KATANA

USER GUIDE

VERSION 1.0v3

Katana software engineering: Andy Lomas, Jonathan Attfield, Chris Hallam, João Montenegro, Örn Gunnarsson, Andrew Bulhak, Steve LaVietes, Jeremy Selan, Brian Hall
Product testing: Phillip Mullan, Richard Ellis
Katana customer support engineer and supplementary testing: Iulia Giurca
Initial writing and layout design: Eija Närvänen
Main user guide author: Gary Jones
Proof reading: Eija Närvänen, Joel Byrne, Tytti Pohjola, Charles Quinn

The Foundry
6th Floor, Communications Building,
48 Leicester Square,
London
WC2H 7LT

Rev: January 6, 2012

Contents

PREFACE

Katana is a 3D application specifically designed for the needs of look development and lighting in an asset-based pipeline. Originally developed at Sony Pictures Imageworks, Katana has been their core tool for look development and lighting for all their productions since "Spider-Man 3", "Beowulf", and "Surf's Up!".

Katana provides a very general framework for efficient look development and lighting, with the goals of scalability, flexibility, and supporting an asset-based pipeline.

Key Concepts

Katana's operation revolves around two tabs, the **Node Graph** and the **Scene Graph**.

Within the **Node Graph** tab, Katana utilizes a node-based workflow, where you connect a series of nodes to read, process, and manipulate 3D scene or image data. These connections form a non-destructive **recipe** for processing data. A node's parameters can be viewed and edited in the **Parameters** tab.

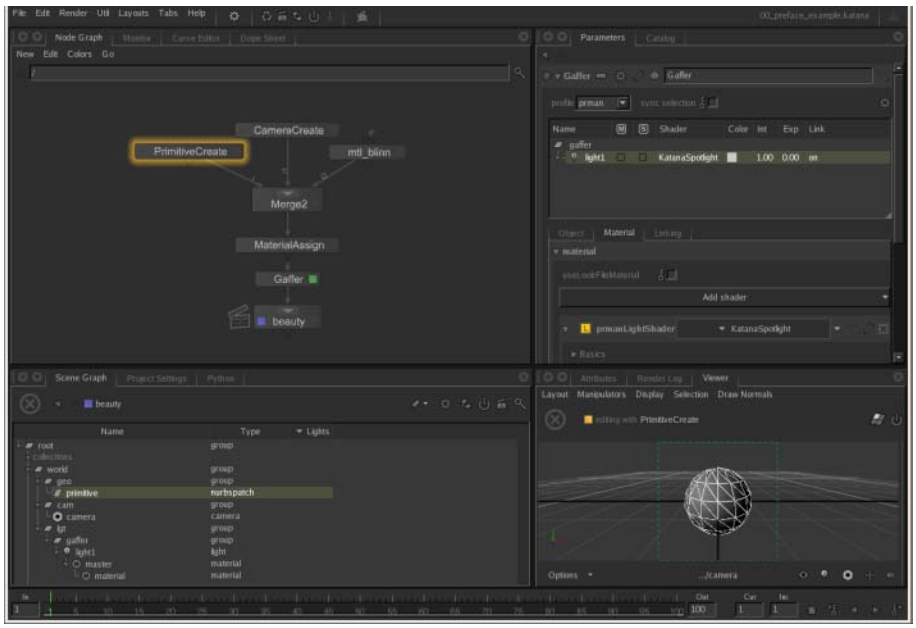
To view the scene generated up to any node within a recipe, you use the **Scene Graph** tab. The Scene Graph's hierarchical structure is made up of locations that can be referenced by their path, such as /root. Each location has a number of attributes which represent the data at that location. You can view, but not edited, the attributes at a location within the **Attributes** tab.

These key concepts are explained in greater depth in the chapter The Way of The Katana.

An example recipe

In this example of a very basic recipe:

- the **Node Graph** tab contains the recipe for creating the scene,
- the **Scene Graph** tab shows the scene generated at the **beauty** node (a renamed **Render** node),
- the **Parameters** tab shows the current parameters of the **Gaffer** node, and
- the **Viewer** tab shows a 3D view from the point of view of the camera.



User Interface and Naming Conventions

For clarity, some naming conventions have been adopted throughout this User Guide.

User interface (UI) elements are in **bold**, such as the **Node Graph** tab, **MaterialAssign** node, and **cameraName** parameter.

Panes are user interface areas that contain one or more tabs. For instance, when you first open Katana there are four panes displaying four tabs. In the top left pane are the **Node Graph**, **Monitor**, **Curve Editor**, and **Dope Sheet** tabs.

As mentioned briefly above, the **Scene Graph** tab displays the scene data generated up to a certain node. Sometimes the data displayed is mentioned without the UI being relevant, when this is the case, Scene Graph is used.

1 INSTALLATION AND LICENSING

Installing and licensing new applications can be a boring task that you just want to be done with as soon as possible. To help you with that, this chapter guides you to the point where you have a default Katana workspace in front of you and are ready to start...

System Requirements

Before you do anything else, ensure that your system meets the following minimum requirements to run Katana effectively:

- Linux 64-bit (CentOS/RHEL 5.4 supported).
- A graphics card which supports OpenGL shader model 4.
- A supported renderer, such as:
  - Arnold 3.3.x.x (minimum 3.3.7.0)
  - PRMan 15.x
  - PRMan 16.x

Install Katana

The current version of Katana can be obtained from our website: http://www.thefoundry.co.uk/products/katana/product-downloads/

Once you have downloaded the tarball, follow the installation instructions below:

1. Move the tarball into a temporary folder.
2. Extract and decompress the tarball inside the temporary folder.
   tar xvf Katana<version>-linux-x86-release-64.tgz
3. Start the install script:
   ./install.sh
4. After reading the End User License Agreement, if you agree with it, type:
   yes
5. Enter the installation directory for Katana.
6. That's it! Proceed to Launch Katana below.

Tip *You can also use the **--path** option which assumes you have read, and agree with, the EULA. For instance, to use the **--path** option to install Katana to the /opt/foundry/katana directory, execute the install script with:*

./install.sh --path /opt/foundry/katana

Connecting Katana to a Renderer

Katana is not tied to any one renderer, in fact, it is designed to be renderer agnostic. By using the Renderer API, plug-in developers connect renderers and renderer-specific settings to Katana. Included with Katana are plug-ins for Solid Angle's Arnold and Pixar's Photorealistic RenderMan (PRMan).

Before trying to connect Katana to a renderer, make sure the renderer is installed correctly. Consult the manual that accompanies the renderer for details.

Note *For more information on writing a renderer plug-in for Katana utilizing the Renderer API, see the developers documentation that accompanies the installation. The developers documentation can be accessed through the* ***Help > Documentation*** *menu option inside Katana.*

Katana uses the KATANA_RESOURCES environment variable to find the renderer plug-ins it needs. The renderer plug-ins included with Katana reside in the PLUGINS/Resources/Renderers directory inside the installation location. Currently, the renderer plug-ins included are:

- Arnold v3.3
- PRMan v16
- PRMan v15

To have Katana utilize the Arnold v3.3 and PRMan v16 plug-ins, set the environment variable to:
KATANA_RESOURCES=$KATANA_HOME/PLUGINS/Resources/Renderers/Arnold3.3:$KATANA_HOME/PLUGINS/Resources/Renderers/PRman16

Note *This assumes the KATANA_HOME environment variable has been previously set to the current installation location.*

Arnold Specific Notes

The current version of the Arnold plug-in (Arnold3.3) is compiled against Arnold v3.3.11.

Compiling the Arnold plug-in to match your Arnold version

Included within PLUGINS/Resources/Renderers/Arnold3.3/src are two source directories so studios can compile the Arnold renderer plug-in to match the version of Arnold they are currently using. Makefiles are included within both directories (RendererInfo and RendererPlugin) to make this compilation easier.

PRMan Specific Notes

Katana has renderer plug-ins for both PRMan v15 and PRMan v16. Although it is possible to have Katana work with multiple renderers at the same time, utilizing the best features of them all, it is currently not possible to have multiple versions of the same renderer. Therefore, it is not possible to have both PRMan v15 and PRMan v16 plug-ins referenced in the KATANA_RESOURCES environment variable.

Including the Katana PRMan shaders

Katana includes a number of basic PRMan shaders. Although not production optimized, some studios may find them useful as example code. They are located at PLUGINS/Resources/Renderers/PRmanXX/Shaders (where XX is the PRMan major release number). Katana's PRMan plug-in finds shaders through the RMAN_SHADERPATH environment variable. To include the example shaders, append their location to the environment variable. For instance:
RMAN_SHADERPATH=$RMAN_SHADERPATH:$KATANA_HOME/PLUGINS/Resources/Renderers/PRman16/Shaders

Licensing Katana

About Licenses

To use Katana, you need a valid **floating license** and server running the Foundry Licensing Tools (FLT).

Floating Licenses—also known as **counted** licenses—enable one of our products to work on any networked client machine. The floating license should be put on the server and is locked to a unique number on that server. Floating licenses on a server require additional software to be installed. This software manages those licenses on the server, giving licenses out to client stations that want them. The software you need to manage these licenses is called the Foundry License Tools (FLT) which can be freely downloaded from our web site. Floating licenses often declare a port number on the server line and a port number on the vendor line.

Setting up the Floating License Server

All the tools necessary for setting up a license server are included with the Foundry Licensing Tools (FLT). The latest version can be downloaded at http://www.thefoundry.co.uk/support/licensing/manage-floating-licences/rlm/

Note *Although Katana is only available for Linux, you can install the license server software on Mac, Windows, or Linux. See the license server system requirements for its own supported operating systems.*

Setting up the License on the Client Machine

Once the license server is up and running, the client needs to point to the license server. To install your license, please download the Foundry License Installer (FLI) for the operating system you are using. This can be found at http://www.thefoundry.co.uk/support/licensing/tools/rlm/

It is also possible to do this manually by changing the environment variable foundry_LICENSE to point to the server. The correct syntax for the environment variable is <PORT_ID>@<SERVER_NAME>. For instance: foundry_LICENSE=4101@our_license_server.

Launch Katana

1. Open a terminal.
2. Navigate to the directory that has the Katana installation.
3. Enter **./katana**.

   If a license is present, the interface displays. Otherwise, you need to license Katana. See Licensing Katana.

Further Reading

For more information on licensing Katana, displaying the System ID number for the license server, setting up a floating license server, adding new license keys and managing license usage across a network, you should read the Foundry Licensing Tools User Guide available on our web site at http://www.thefoundry.co.uk/support/licensing/

2 THE WAY OF THE KATANA

Katana Introduction

For users of 3D applications and compositors, Katana has familiar elements, such as a timeline, a Node Graph, a hierarchical scene view, and a 3D viewer. Whether you are a 3D veteran or diving into your first 3D application, this chapter explains how some of these elements are used within Katana and tries to get you into the mind-set of a Katana user.

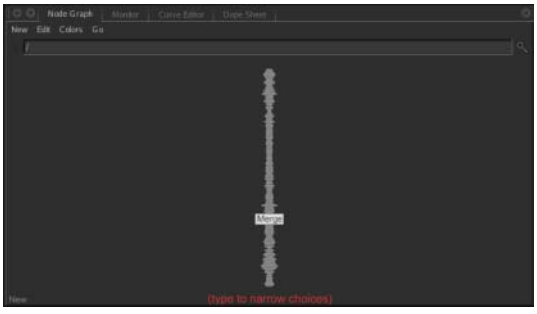
Unlike most of the chapters, this one guides you through a number of steps and then follows up with an explanation of how these steps influence Katana behind the scenes. If you don't understand everything at first glance, don't panic, it's all explained in greater detail in later chapters. In fact, there are cross-references to more in-depth explanations dotted throughout this chapter.

The steps in this chapter are:

It's time to launch the application and get ready to embrace *The Way of the Katana!*

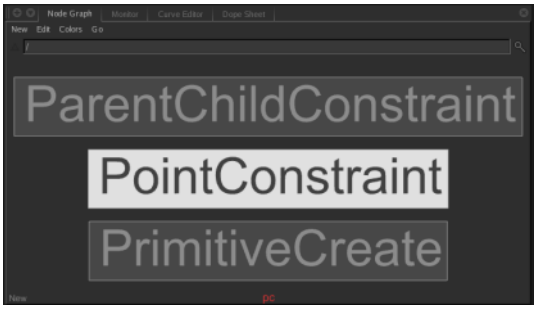
Step 1: Learning about the Node Graph, recipes, and node creation

1. Hover the mouse over the Node Graph and press **Tab**.
   All available nodes are displayed.



2. Type **PC**.

   This narrows the node list down to those with **PC** at the start of their name and those with **PC** as their name's starting capital letters.

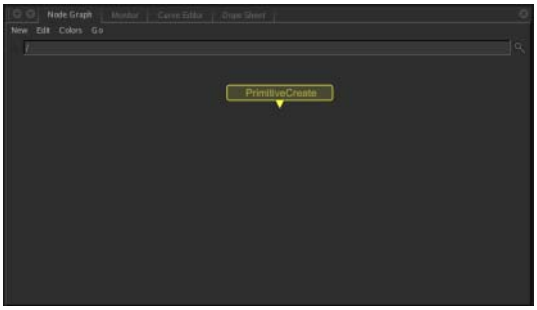


3. Click **PrimitiveCreate**.

   The PrimitiveCreate node is selected. Once selected, it floats with the cursor, ready to be placed.

4. Click somewhere towards the top of the **Node Graph**.

   The node is added to the **Node Graph**, beginning a new recipe.



The **Node Graph** is where it all starts. It is here that you create a recipe by connecting various nodes, which add, override, or modify scene data.

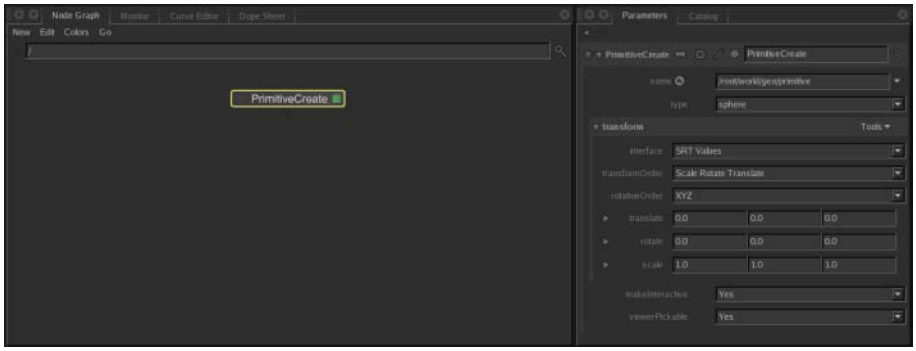
Most **recipes** start by reading in the 3D elements—such as camera data, geometry caches, or particle caches—that comprise the scene. In this example we use a PrimitiveCreate node that defaults to a sphere.

One of the most important things to understand about the **Node Graph**, and its resulting recipe, is that it is non-destructive. The recipe is a description of how to bring in the ingredients (the various assets), modify them to suit the shot, add materials and assign them to objects, add lights, and finally send everything off to a renderer. The recipe approach is extremely flexible, allowing the assets to be continually updated in an iterative workflow.

For more on nodes and the Node Graph, see Working with Nodes.

Step 2: Editing a node and using the Parameters tab

Hover the mouse over the PrimitiveCreate node and press **E**.
A green square appears at the right-hand side of the node and the node's parameters are displayed in the **Parameters** tab.



This is one way to make a node editable. You can also:

- select one or more nodes and press **Alt+E**, or
- click the faint square at the right-hand side of the node.

Most nodes within Katana have **parameters** that modify their behavior. You can change these parameters in the **Parameters** tab. A node that is being edited within the **Parameters** tab displays a green square on its right-hand side.

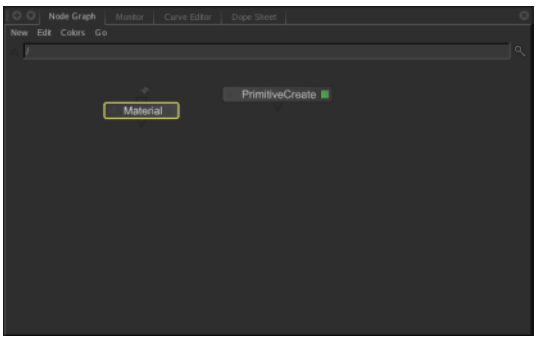
To find out more about parameters and the **Parameters** tab, see Editing a Node's Parameters.

On with the recipe!

Step 3: Creating and assigning materials

1. Press **Tab** in the **Node Graph**.
2. Type **MAT** to filter the node list.
3. Select **Material**.

4. Click to the left of the PrimitiveCreate node to add the Material node to the **Node Graph**.

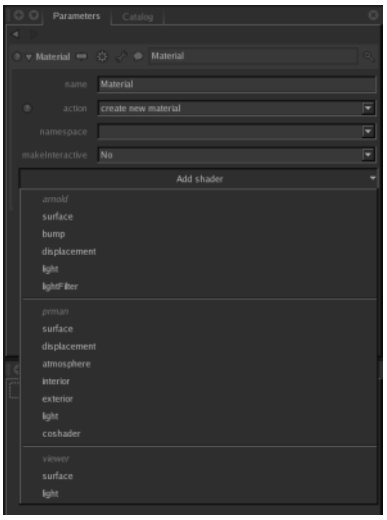


5. Hover the mouse over the Material node and press **E**.

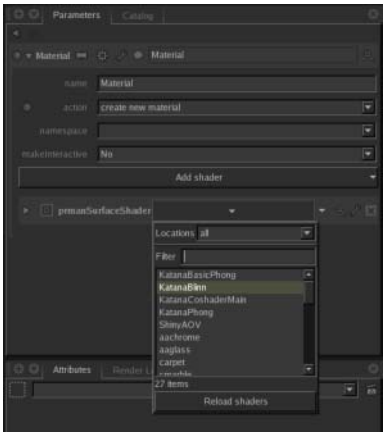
   The Material node becomes editable within the **Parameters** tab.

6. In the **Parameters** tab, click **Add shader** and select a shader type from the list, for instance **prman surface**.

   The shader list varies depending on the renderers installed.

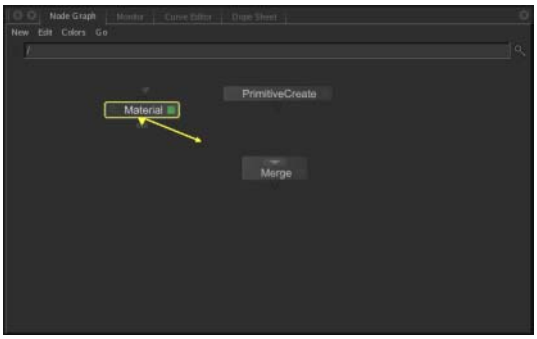


7. Click the large dropdown to the immediate right of the shader type and select a shader, for instance **KatanaBlinn**.



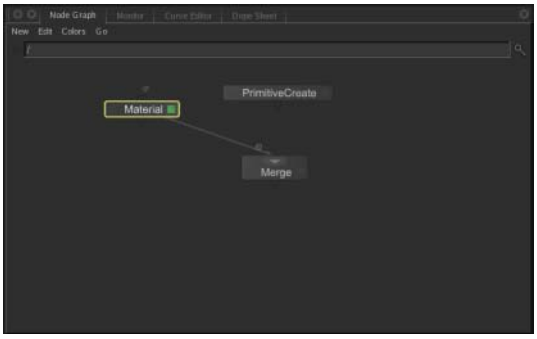
8. Create a Merge node and place it below the PrimitiveCreate node.
9. Hover the mouse over the Material node and press ` (**Backtick**).

   A connection from the Material node is started.



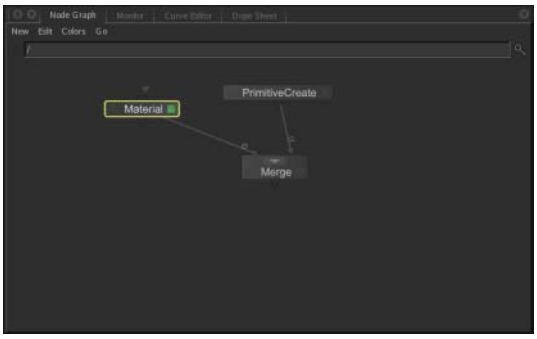
10. Hover the mouse over the Merge node and press ` (**Backtick**).

    Pressing **Backtick** a second time connects the Material node to the input of the Merge node. It is also possible to manually connect two nodes by dragging from the output triangles to the input squares, see Connecting Nodes.

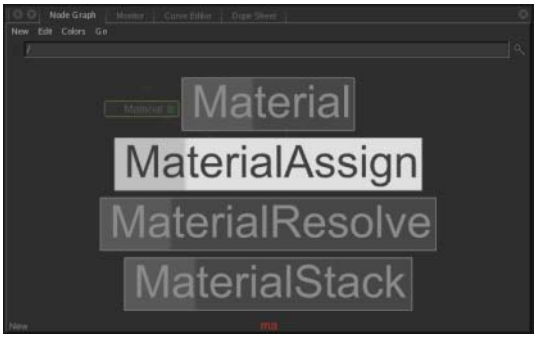


11. Connect the PrimitiveCreate node to the Merge node, as described above.

    The **Merge** node brings different branches of the recipe together, combining their scene data. Right now, the Merge node brings together the sphere created by the PrimitiveCreate node and the material created by the Material node.

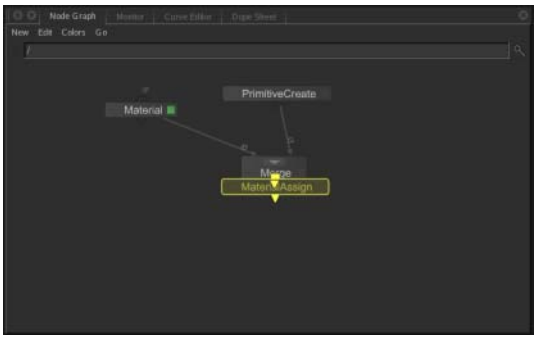


12. Create a MaterialAssign node using the **Tab** key menu.

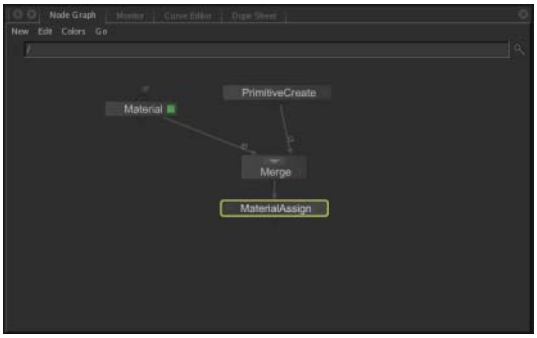


13. With the MaterialAssign node floating with the cursor, hover the node over the output from the Merge node until it turns yellow, then click to connect.

    This connects the MaterialAssign node to the output of the Merge node. The MaterialAssign node continues to float with the cursor.



14. Click below the Merge node to place the MaterialAssign node.

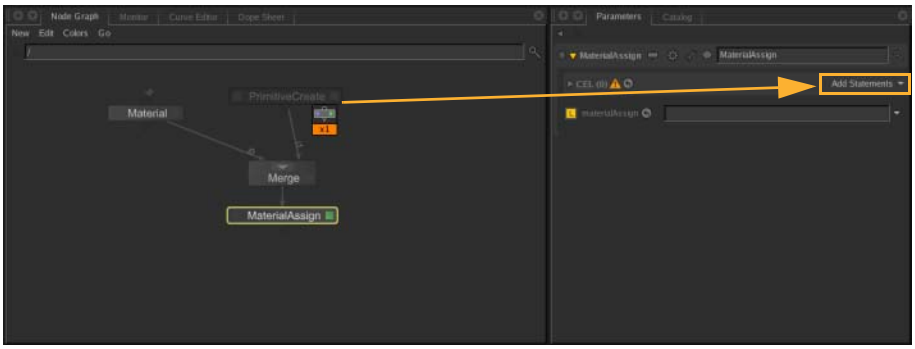


15. Hover the mouse over the MaterialAssign node and press **E**.

    The MaterialAssign node becomes editable within the **Parameters** tab.

16. **Shift+middle**-click and drag from the PrimitiveCreate node to the **Add Statements** menu in the **Parameters** tab.

    The Scene Graph location of the object created by the PrimitiveCreate node is added to the **CEL** parameter.



    After you've created a material it needs to be assigned. In the MaterialAssign node, Katana uses the **Collection Expression Language** (**CEL**) to create a list of Scene Graph locations to be used in the material's assignment.

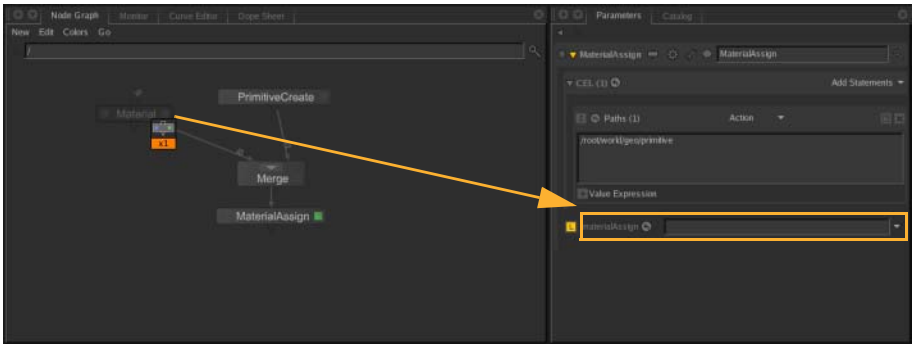
    For a brief explanation of the **Scene Graph**, see Step 5: Using the Scene Graph. For a more comprehensive explanation, and an explanation of locations, see Using the Scene Graph.

    For further details on CEL, see Assigning locations to a CEL parameter.

17. **Shift+middle**-click and drag from the Material node to the **materialAssign** parameter in the **Parameters** tab.

    An expression is created that links the material created by the Material node to the **materialAssign** parameter. If the location created by the

Material node changes, the expression automatically updates the **materialAssign** parameter with the new location.

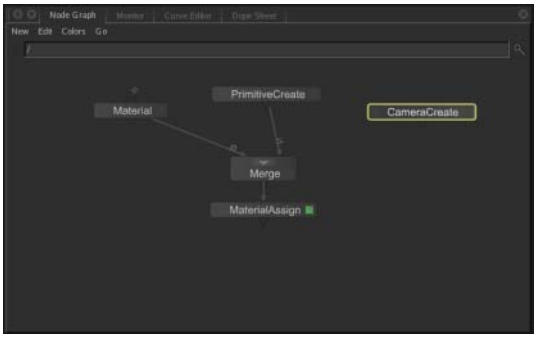


Materials define how geometry and lights are rendered. Each material can have one or more shaders for each renderer, as well as a shader that defines how that object is displayed in the 3D Viewer. Materials can be assigned to geometry or lights and then saved as a **Katana Look File** (.klf). This part of a production pipeline is commonly referred to as **look development**.

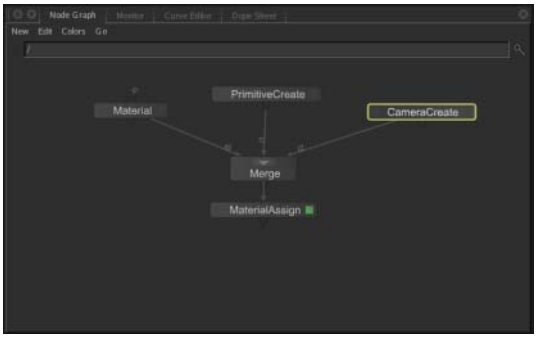
Katana Look Files are extremely powerful. For a more in-depth explanation, see Look Development with Look Files.

Step 4: Lights, camera, action

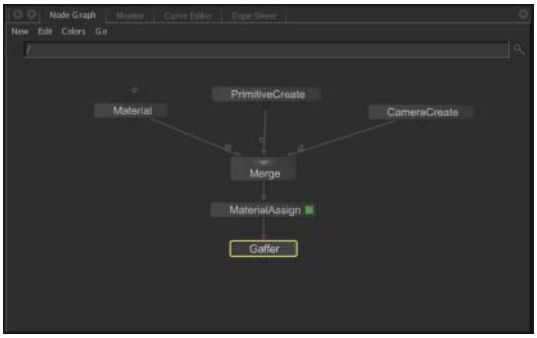
1. Create a CameraCreate node in the same way as the previous nodes and place it to the right of the PrimitiveCreate node.



2. Connect the CameraCreate node to the Merge node.



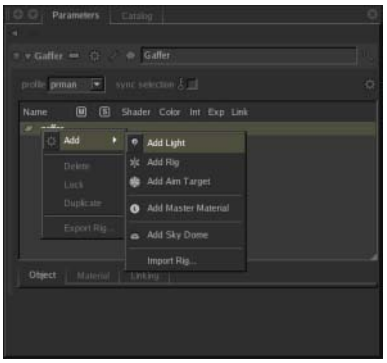
3. Create a Gaffer node and connect it to the output of the MaterialAssign node.



4. Hover the mouse over the Gaffer node and press **E**.

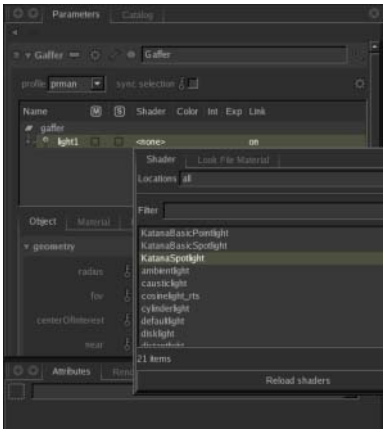
   The Gaffer node becomes editable within the **Parameters** tab.

5. In the light list for the Gaffer node now displayed in the **Parameters** tab, right-click on **gaffer** and select **Add > Add Light**.



   This creates a light and places it below the **gaffer** in the light list.

6. In the light list, under the **Shader** column, click **<none>** and select a light shader from the list, for instance **KatanaSpotlight**.



Note *The shader list varies depending on the renderers installed.*

In the example, the camera is created within the recipe. It could just as easily be brought in from an external file, such as Alembic (ABC). Supplementary cameras may be placed in Katana for various rendering techniques, such as camera based projections or stereo.

Lights are usually created within the **Gaffer** node. The name comes from the role of the person on-set responsible for the setting up of the lights — the Gaffer. The Gaffer node provides a one-stop-shop for a number of convenient lighting functions, for instance: light creation, shader assignment, and light soloing and muting.

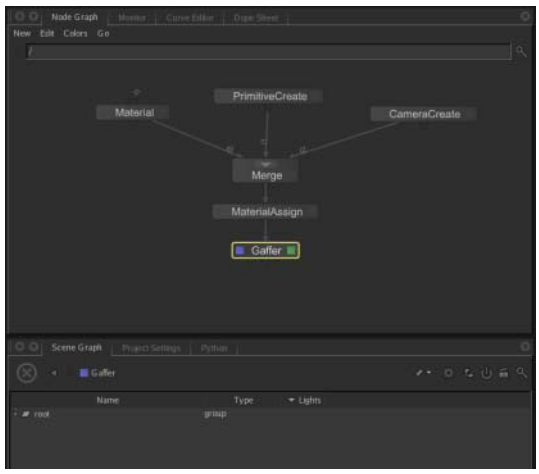
For more information on the Gaffer node, see Getting to Grips with the Gaffer Node, or for lighting in general, see Lighting Your Scene.

Step 5: Using the Scene Graph

1. Hover the mouse over the Gaffer node and press **V**.

   The **Scene Graph** tab now displays the 3D scene generated up to the Gaffer node.

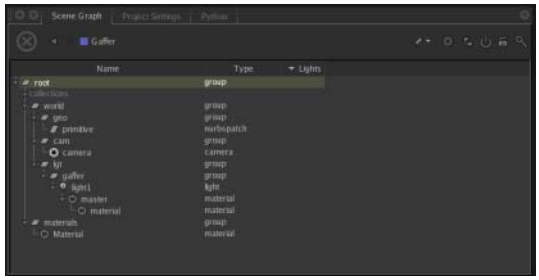
   This is one way to view the Scene Graph generated at a node. You can also click the faint square at the left-hand side of the node.



   When the **Scene Graph** tab is displaying the scene generated at a node, that node (referred to as the view node) has a purple square displayed on the left-hand side.

2. In the **Scene Graph** tab, right-click on the **/root** location and select **Expand All**.

   The /root location expands to display everything within the Scene Graph.



There are a few key concepts regarding the Scene Graph:

The first key concept is that: **the Scene Graph is just a viewer.** The Scene Graph displays the 3D scene generated for the current frame by stepping through the recipe up to the node with the blue square—this node is the current view node.

The second concept is that: **there is no such thing as *the scene* in Katana.** You can merge and branch the recipe, pruning and adding to one of the branches. Therefore, pressing **V** at different nodes can generate vastly different scenes. To see this for yourself, hover the mouse over the CameraCreate node and press **V**. The Scene Graph changes because at this node in the recipe, only the camera has been created. You can view the 3D scene generated at any of the other nodes in the same way. When you are happy, hover the mouse back over the Gaffer node and press **V** again.

Lastly, **the Scene Graph only loads data as it is needed.** This deferred loading makes it possible for Katana to contain recipes dealing with scenes of incredible and potentially even infinite size. As you control which elements are loaded—by expanding locations within the Scene Graph—you can light extremely complicated scenes by only loading the required data. You don't need to load all the scene data to light the scene.

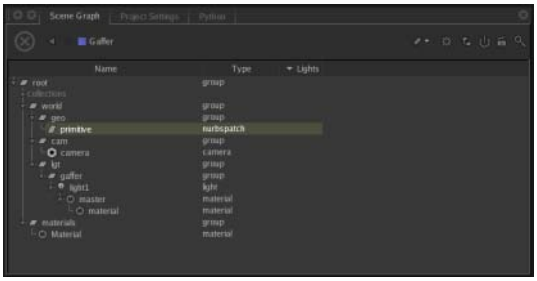
Each location, such as /root or /root/world, within the Scene Graph has attributes that you can view within the read-only **Attributes** tab.

To find out more about the Scene Graph, locations, and attributes, see Using the Scene Graph.

There are three main viewers for the underlying Scene Graph data in Katana, the **Scene Graph** tab, the **Attributes** tab, and the **Viewer** tab.

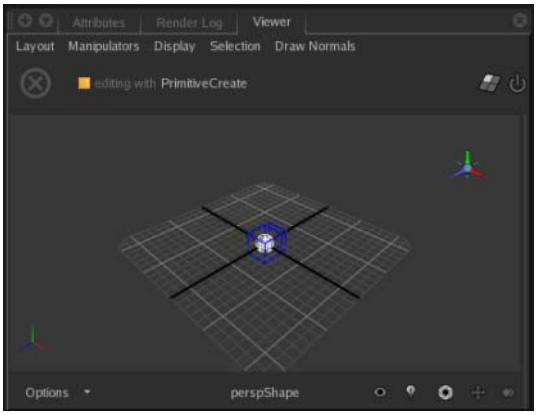
Step 6: Using the Viewer

1. With the Gaffer node as the view node (designated by the purple square) and the **Scene Graph** fully expanded, select **primitive** in the **Scene Graph** tab.



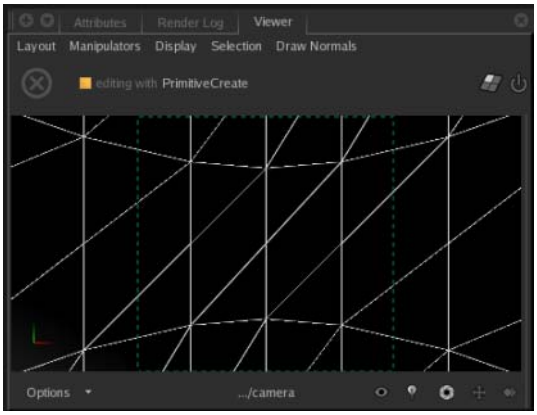
2. Towards the bottom of the **Viewer** tab (located in the bottom right pane by default), click **perspShape**.

   A list of objects you can look through displays. This is a combination of the light list (to list just lights, click ) and the camera list (to list just cameras, click ).



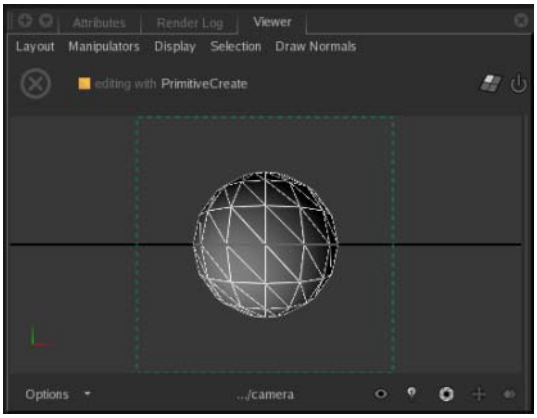
3. Select **../camera**.

   The **Viewer** tab now shows the view from the point of view of the camera.



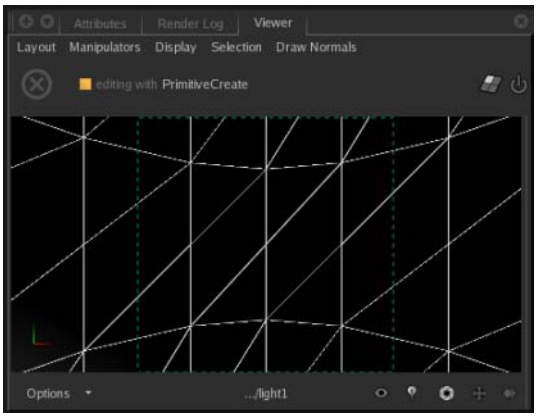
4. With **primitive** still selected in the **Scene Graph**, press **F** in the **Viewer** tab.

   The camera moves to frame the currently selected object and makes it the camera's point of interest.

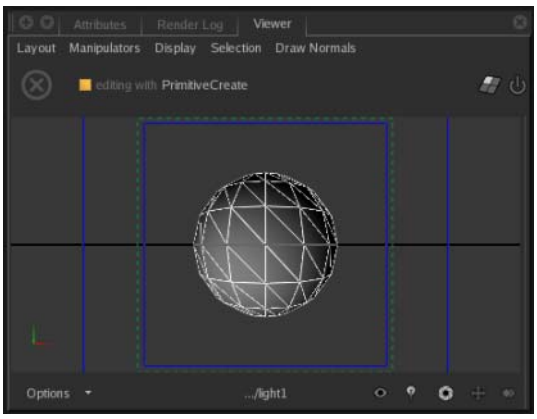


5. Click **../camera** in the **Viewer** tab and select **../light**.

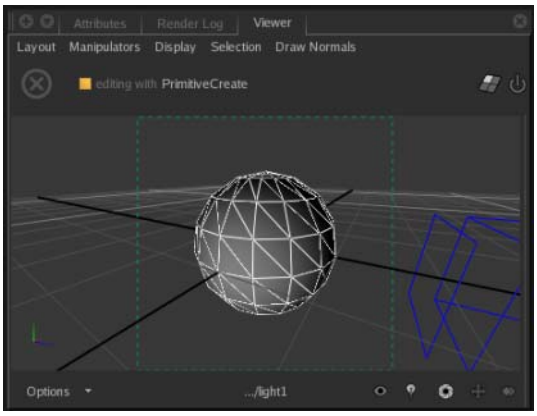
   The view changes to that of the light.



6. Press **F** in the **Viewer** tab to frame the sphere again, this time for the light.



7. **Alt+left**-click and drag to rotate the light around the sphere and position the light.

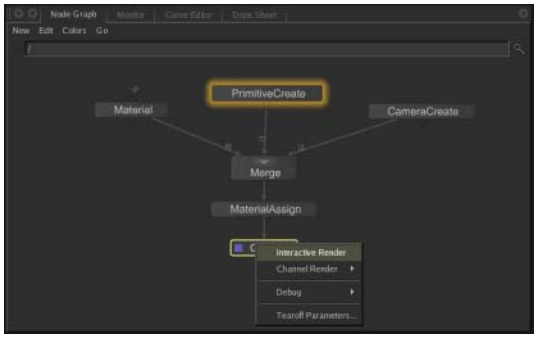


The Viewer is a 3D representation of the scene within the Scene Graph **but only locations exposed within the Scene Graph are displayed**. Therefore, if you first view the Scene Graph for a node and only /root is exposed **the Viewer is empty.**

To learn more about how to move around and manipulate objects within the **Viewer**, see Using the Viewer.

Step 7: Starting a render

Right-click on the Gaffer node in the **Node Graph** and select **Interactive Render**.
The scene generated at the Gaffer node is sent to the renderer. This is the actual production renderer that is used, not an internal Katana renderer.



You can view the progress of the render in the **Render Log** tab and the results are displayed in the **Monitor** tab (click the **Monitor** tab next to the Node Graph tab when using the default workspace to access the **Monitor**). To have the render fit the size of the **Monitor** tab, press **F**.

For more on rendering, saving your renders, and changing render settings, see Rendering a Scene. For more on viewing the results of your renders, see Viewing Your Renders.

That's it! You now know the basics on wielding Katana. Keep going!

3 Customizing Your Workspace

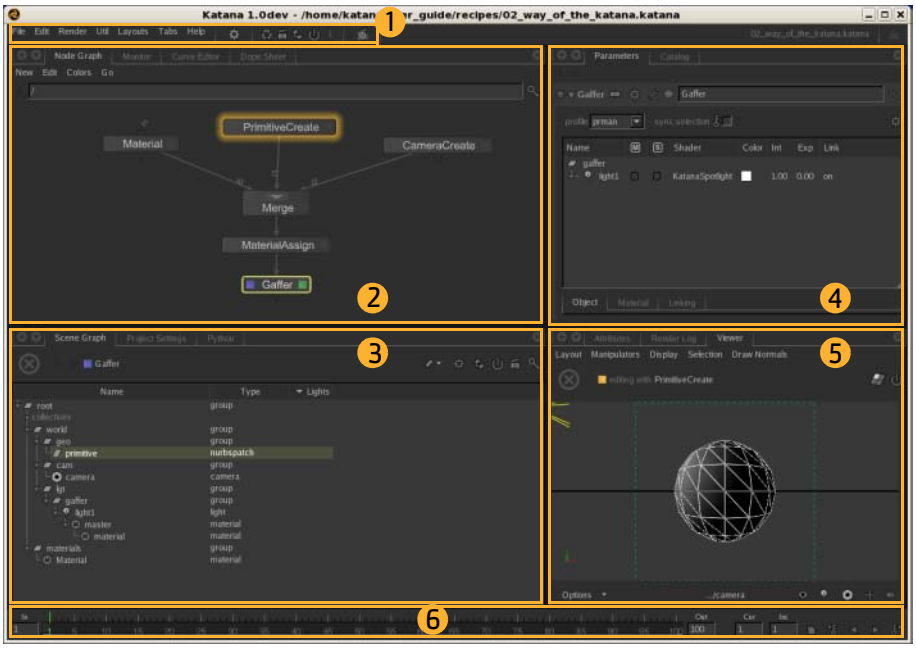
This chapter is designed to help you understand the Katana workspace, and how to customize it to meet your particular needs.

Workspace Overview

If you have used 3D applications in the past, you may notice that Katana's workspace has many familiar features, such as a timeline, a hierarchical Scene Graph view, an OpenGL viewer, and a 2D monitor.

The Default Workspace

Here is an illustration of a simple Katana workspace.



1. The menu bar, complete with menus, such as File, Help, etc. and menu icons, such as the Interactive Render Filter icon. For further details, see Menu Bar Components.
2. The top left pane, containing the **Node Graph**, **Monitor**, **Curve Editor**, and **Dope Sheet** tabs.
3. The bottom left pane, containing the **Scene Graph**, **Project Settings**, and **Python** tabs.
4. The top right pane, containing the **Parameters** and **Catalog** tabs.

5. The bottom right pane, containing the **Attributes**, **Render Log**, and **Viewer** tabs.

   For more on the contents of the various tabs, see the The Default Tabs below.

6. The **Timeline**. The **Timeline** is explained in greater depth in Using the Timeline.

The Default Tabs

The following are the tabs displayed by default. More tabs are available in the **Tabs** menu.

Tab	Function
Node Graph	This is where you build your node tree (a tree graph that represents the recipe for manipulating a 3D scene).
Monitor	This is where you view the results of your renders and composites.
Curve Editor	Lets you edit animation keys as curves.
Dope Sheet	Lets you edit animation keys as a spreadsheet of keys and ranges.
Scene Graph	This is where you view the scene data, generated at the current view node in the Node Graph, in a hierarchical representation. The objects—such as geometry, particle data, volumetric data, materials, cameras, and lights—that make up the Scene Graph are called locations, and are referenced by their path, such as /root/world/cam/camera.
Project Settings	This is where you can view and edit parameters for the whole project.
Python	This is where you can enter Python commands as well as view their outputs. It acts as a Python interactive shell within Katana.
Parameters	This is where you adjust the parameters associated with nodes currently selected for editing.
Catalog	Lets you view and organize previous renders.
Attributes	Lets you view the attribute values held at each location in the Scene Graph.
Render Log	Lets you view text output from the renderer.
Viewer	This is where you can view and manipulate your scene using a 3D representation. Only objects whose locations that are visible in the Scene Graph are displayed.

Menu Bar Components

The Katana menu bar includes the following functions:

Menu	Functions
File	Commands for disk operations, including creating, loading, and saving Katana projects.
Edit	Undo, redo, and preferences.
Render	Rendering the output.
Util	A group of miscellaneous menu items including farm management and cache handling.
Layouts	Adjusting, saving, activating, and deleting layouts.
Tabs	Adding floating panes to the interface.
Help	Accessing documentation, APIs, and information on the current version.
	Collection of Python shelf scripts.
	Flush caches: forces assets, such as look files, to be dropped from memory and reloaded when needed.
	Toggles implicit resolvers. This gives a better impression of the data sent to the renderer at the cost of extra computation. For more on implicit resolvers, see Turning on Implicit Resolvers.
	When enabled, rendering only includes items selected in the **Scene Graph** tab.
	Stops the Scene Graph from being regenerated.
	The auto key icon: when enabled, changing parameters automatically adds a new key.
	Specify what interactive render filters to use for any new interactive renders. For more on interactive render filters, see Setting up Interactive Render Filters.

Customizing Your Workspace

You can create layouts designed for whatever function you happen to be performing. For instance: lighting, look development, or material editing. You can then save your preferred layouts for future use.

During the customization process, you can:

- Resize panes to create space where it's most needed.

- Maximize the pane under the mouse cursor.
- Move and split panes to create new work areas, for example, to have two **Viewers** side-by-side.
- Remove panes and all tabs nested inside them.
- Add and remove tabs as required.
- Move tabs to easily access the elements you often need.
- Float and nest tabs to create more space or group similar functions together in the same pane.
- Add a Timebar to the main Katana window or any tab.
- Make the main Katana window fullscreen, hiding the window borders.

Once you are happy with the layout, you can save it for future use.

Note *The **Layouts** menu also includes four preset layouts to get you going.*

Adjusting Layouts

To make accessing the elements you often need as quick and easy as possible, it's a good idea to adjust the default layout(s).

Resizing panes

To resize individual panes, hover the mouse over the divider line until the cursor changes to the resize icon. Click and drag the cursor to resize the pane.

Tip *When moving the divider line, by default, if it crosses multiple panes, the entire line is moved. To only move the divider line for the local pane, **Ctrl**+drag.*

Maximizing panes

To maximize a pane so that it expands to the size of the window:

- Click  in the top left corner of the pane, or
- hover over the pane and press **Spacebar**, or
- double-click the tab of the pane to maximise.

To return to the regular interface, click  or press **Spacebar**.

Note *Pressing the **Spacebar** in the **Monitor** tab does not maximize the pane, instead it swaps the **Front** and **Back** images.*

Moving and splitting panes

To move an existing pane to a new location in the interface:

1. Hover over the  icon in the top left corner of the pane until the cursor changes to the move icon.
2. Click and drag the pane to a new location.

   The orange highlight around the destination pane helps you determine where the pane is moved and whether the destination pane is split horizontally or vertically.

Removing panes

To remove a pane and all tabs nested inside it, right-click on any of the tab names and select **Close all**.

Adding panes

To add a floating pane to the interface, use the **Tabs** menu in the menu bar and select the tab you want to add.

To add a tab to a specific pane, click  in the top left corner of the pane and select the tab you want to add.

Moving tabs

To move an existing tab to a new location in the interface, click and drag the tab to a new location.
The orange highlight around the destination pane helps you determine where the tab is nested and if the destination pane is split horizontally or vertically.

Removing individual tabs

- Make sure you are viewing the tab you want to remove and click on  in the top right corner of the pane, or
- right-click on the name of the tab and select **Close tab**.

Floating and nesting tabs

To turn a tab into a floating window, right-click on the name of the tab and select **Detach tab**.

To nest a floating tab, click on the name of the tab and drag it to where you want it to dock. Use the orange highlight around the destination pane to

help you determine where the tab is nested and whether the destination pane splits horizontally or vertically.

Showing and hiding timelines

To show or hide a Timebar at the bottom of the main Katana window, select **Layouts** > **Show Main Timeline**.

To show or hide a Timebar at the bottom of any tab, right-click on the tab name and select **Show Timeline**.

Making the main window fullscreen

To make the main Katana window fullscreen, select **Layouts** > **View Fullscreen**.

To return to normal, select **Layouts** > **View Fullscreen**.

Saving Layouts

You can save as many of your favorite layouts as needed, retrieving them as necessary.

To save a layout:

1. Once you are happy with your layout, select **Layouts** > **Save current layout**.

   The **Save Current Layout** dialog opens.
2. In the dialog, enter a name for the new layout.
3. If your layout includes any floating tabs and you want those to be saved with the layout, check **Save # Floating Panes** (where # corresponds to the current number of floating panes).
4. Click **Save** to preserve your layout.

Loading Layouts

To load a previously saved layout, select it from the **Layouts** menu in the menu bar.

Deleting Layouts

1. Select **Layouts** > **Edit saved layouts**.
2. In the dialog that opens, select the layout you want to delete from the list of available layouts.
3. Click **Delete Layout** and **Save**.

4 Creating a Katana Project

Katana Projects and Recipes

There are no fixed rules as to what constitutes a Katana project. A Katana project is simply a collection of recipes that are worked on together and stored in a single .katana file. A project could be a shot, a scene, or look development for one or more assets.

Each recipe within a project can be totally self contained or it can be linked to others through dependencies. As an example, look development could have one recipe which creates a **Katana look file** (**.klf**) for a piece of geometry and another recipe which renders out a turntable of that same geometry complete with its newly created Katana look file assigned.

How you group your recipes into Katana projects is up to you and your studio.

Creating a new Katana project

To create a new Katana project:

1. Select **File > New** (or press **Ctrl+N**).
2. If needed, click **New Project** in the **Unsaved Changes** dialog window to confirm.

Note *Ctrl+N does not work within the **Node Graph**.*

Saving a Katana Project

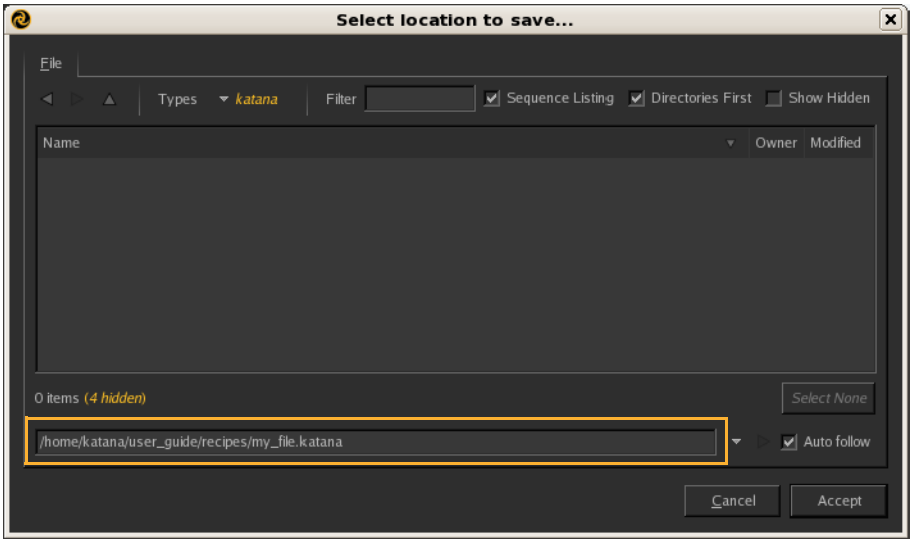
To save your current Katana project:
Select **File > Save** (or press **Ctrl+S**).
If the file has not been saved before, the **file browser** appears. See steps 2 to 4 below to select a location to save.

Saving to a new file

To save your current Katana project to a new file:

1. Select **File > Save As...** (or press **Ctrl+Shift+S**).
   The **file browser** appears.
2. Navigate to the directory to save the file.

3. Add the filename to the text field below the main window.



4. Click **Accept**.

Note *If using a custom asset management system, a different dialog may display.*

Loading a Katana Project

To load a Katana project:

1. Select **File > Open...** (or press **Ctrl+O**).
2. If needed, click **Load New Scene** in the **Unsaved Changes** dialog window to confirm.
3. Select a Katana project from the **file browser** (see Using the File Browser below).
4. Click **Accept**.

Loading a recently saved Katana project

To load a recent Katana project:

1. Select **File > Open Recent > ...** and select from one of the previously saved projects in the list.
2. If needed, click **Load New Scene** in the **Unsaved Changes** dialog window to confirm.

Tip *You can clear the list of recently opened projects by selecting **File > Open Recent > Clear Menu.***

Reverting back to the last save

You can revert back to the last time you saved, to do so:

1. Select **File > Revert**.
2. Click **Revert Scene** in the **Unsaved Changes** dialog window to confirm.

   The Katana project reverts back to the last save.

Importing a Katana Project

To import a Katana project into the current project:

1. Select **File > Import...** (or press **Ctrl+I**).
2. Select a Katana project from the **file browser** (see Using the File Browser below).
3. Click **Accept**.

   The imported project's nodes float with the cursor inside the **Node Graph**.
4. Click somewhere within the **Node Graph** to place the imported project at that location.

Importing a Katana project as a LiveGroup

A LiveGroup is an imported Katana project (inside a Group node) which is re-imported every time the current project is loaded. The nodes themselves cannot be updated and are locked. To import as a LiveGroup:

1. Select **File > Import LiveGroup...**
2. Select a Katana project from the **file browser** (see Using the File Browser below).
3. Click **Accept**.

   A **LiveGroup** node, named after the imported file, floats with the cursor inside the **Node Graph**.
4. Click somewhere within the **Node Graph** to place the **LiveGroup** at that location.

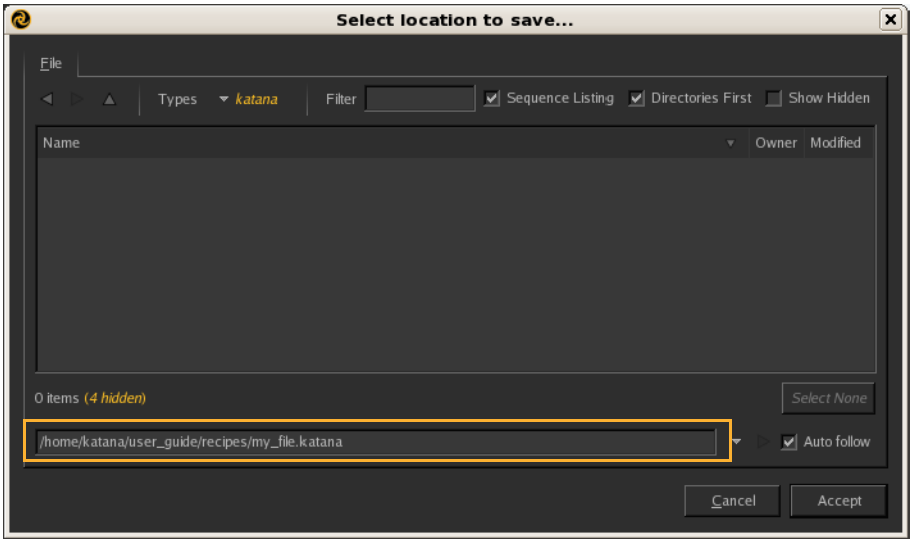
Exporting a Katana Project

Exporting from Katana gives you the ability to do the equivalent of **File > Save As...** but for a limited number of nodes. To export part of the current project:

1. Select the nodes you wish to export.
2. Select **File > Export Selection...** (or press **Ctrl+E**).

   The File browser appears.
3. Navigate to the directory to export the file.

4. Add the filename to the text field below the main window.



5. Click **Accept**.

Changing a Project's Settings

Projects' settings are shared between each of the recipes created within that project. These can all be changed from within the **Project Settings** tab.

Setting	Description
inTime	The starting frame number for the timebar.
outTime	The ending frame number for the timebar.
currentTime	The current frame number.
timeIncrement	Changes the frame increment for the move forward and backwards icons in the timebar.
resolution	The default resolution for 2D source files, such as ImageColor. Not used for the rendering of 3D scenes, they use the RenderSettings node instead.
plugins	
asset	The asset manager to use (defaults to **File**).
fileSequence	Plug-in to determine how to interpret a file sequence.
compDefaults > fileIn	
missingFrames	How an ImageRead node behaves when a frame is missing.
inMode	What an ImageRead node displays for frames before its first frame.

Setting	Description
outMode	What an ImageRead node displays for frames after its last frame.
compDefaults	
useOverscan	Whether to use overscan when rendering. Overscan is extra pixel information around the main render window.
viewerSettings	
normalsDisplayScale	Changes the size of normals when displayed in the **Viewer** tab.
proxyCacheSize	Number of proxy geometry objects to keep in memory.
monitorSettings	
overlayColor	Color to use when displaying alpha overlays.
ROI	
left	Left coordinate of the region of interest rectangle.
bottom	Bottom coordinate of the region of interest rectangle.
width	Width of the region of interest rectangle in pixels.
height	Height of the region of interest rectangle in pixels.

Dealing With Assets

Katana has been designed from the ground up to work within an asset based production environment. In fact, the philosophy behind Katana—the non-destructive recipe based approach—works to its fullest when used with assets that change and update in an iterative workflow. The decoupling of asset creation and their use in shots, allows a team to work in parallel.

Whether in a small, medium, or large studio, an asset management system helps maintain the large number of assets and revisions that artists create and use.

With its extensible **Asset Management API**, Katana can be made to slot into any production workflow that incorporates an asset management system. Examples of how to incorporate an asset manager using the Asset Management API are included with the Katana install. A full explanation of this process goes beyond the scope of this guide. For all examples within this guide, we assume you are using the **File** asset manager that ships as the default with Katana. For further details on the asset manager employed by your facility, consult your pipeline manager.

Selecting an Asset Manager

By default, Katana uses the file system to store assets. But Katana has the ability to plug into a studio's asset management system through its Asset Management API. Connecting Katana using this system is beyond the scope of the User Guide and you should consult your pipeline manager and the technical documentation that accompanies the installation for further information (the technical documentation is found under **Help > Documentation**).

Once connected, you can change the asset manager from within the **Project Settings** tab.

Changing the current asset manager

You can select which asset manager to use by doing the following:

1. In the **Project Settings** tab, click the **plugins > asset** dropdown.
2. Select the asset manager from the filterable list.

Using the File Browser

The file browser is the basis for the **File** asset manager.

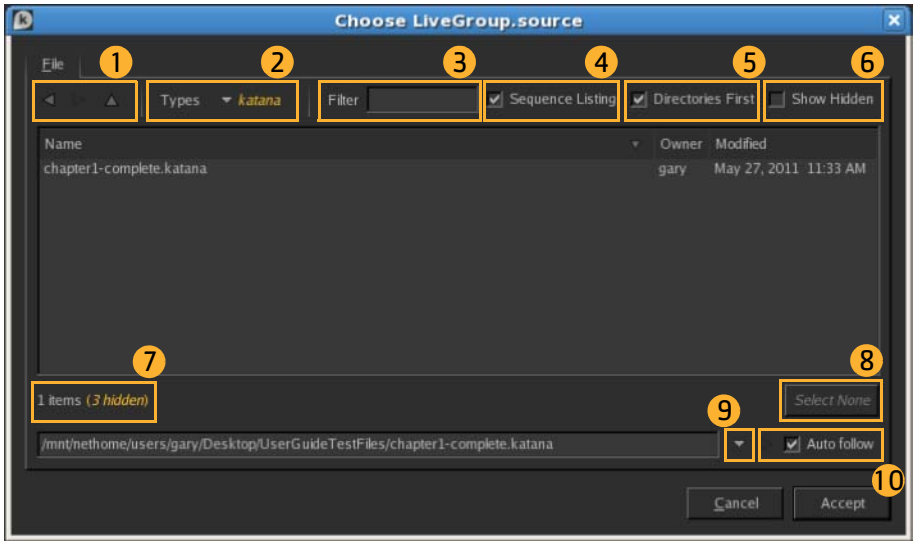


Figure 5. The file dialog.

1. To step through the file browser's history, click the left and right arrows. To move up a directory, click the up arrow.
2. To change the filter controlling which files are displayed within the file dialog, click the dropdown next to **Types**.
3. Type in the **Filter** field to narrow the list of items within the main area. Only items that contain the string you entered are displayed.

4. Enable **Sequence Listing** to display image sequences as a single item. If disabled, image sequences are displayed as individual files.
5. Click the **Directories First** checkbox to toggle whether directories are displayed at the top or mixed in with the other files.
6. Click the **Show Hidden** checkbox to toggle whether hidden files are displayed.
7. The count for displayed (and filtered) items is displayed on the left below the main window.
8. To deselect all items, click **Select None**.
9. To quickly navigate to recent and parent directories, click the down arrow next to the filename.
10. To have Katana automatically update the main window as you navigate, enable **Auto follow**. If **Auto follow** is off, use  to navigate into a directory.

5 Working with Nodes

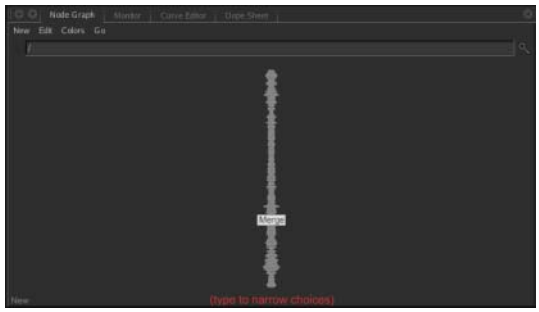
Nodes are the basic building blocks of a Katana recipe. You create and connect nodes to form a tree of the operations you want to perform.

Adding Nodes

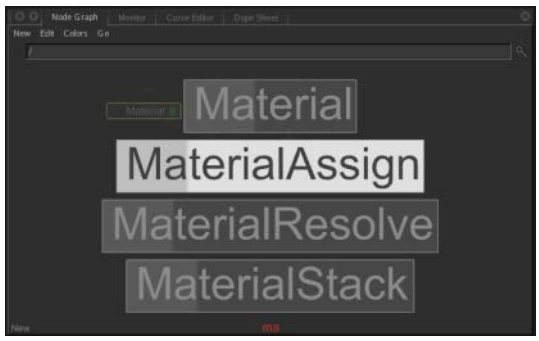
You add nodes to the **Node Graph** using the **Tab** menu, the **New** menu, or the right-click menu.

Adding a node using the Tab key menu

1. With the mouse over the **Node Graph** tab, press the **Tab** key.
   Katana displays a list of all available nodes.



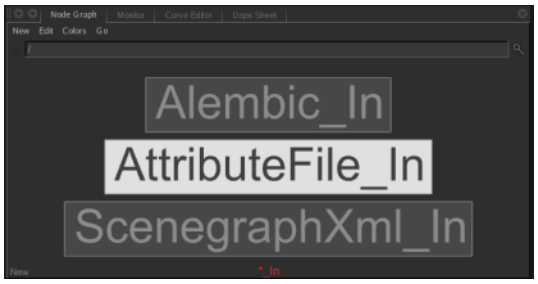
2. Narrow the list of nodes by either:
   - typing the starting letters of the node name, or
   - typing the capital letters that make up the node name (for instance, typing **MA** for the MaterialAssign node).



3. To select the node you want to add from the list, either:
   - click on it, or
   - scroll to it with the **Up** and **Down** arrow keys and press **Return**.
4. Click on an empty space in the **Node Graph** to place the node.

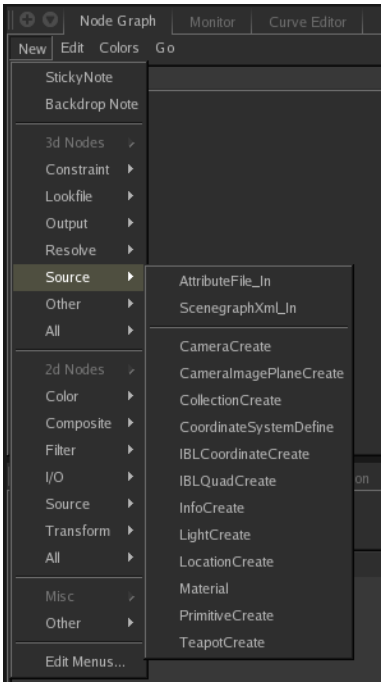
Tips *To add another copy of the last node created using this method, simply press **Tab** and then **Return**.*

*Katana accepts wildcards while typing the name of the node to create. For instance, ***_In**.*



Adding a node using the New menu

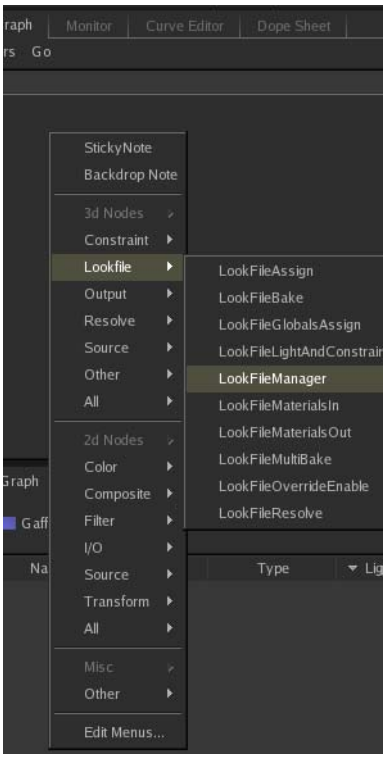
1. In the **Node Graph** tab, select **New** and the node you want to add.



2. Click on an empty space in the **Node Graph** to place the node.

Adding a node using the right-click menu

1. Right-click on the **Node Graph** (or press **N**) and select the node you want to add from the menus.



2. Click on an empty space in the **Node Graph** to place the node.

Tips *While the node is floating with the mouse cursor, you can cancel the nodes creation by pressing **Esc**.*

*To have Katana automatically connect the new node to the currently selected node, check the option **Edit > Auto Connect New Nodes Based On Selection** within the **Node Graph**.*

Instead of placing the node and then connecting it, you can connect the node straight into the node tree by either:

- *clicking on a connection, or*
- *clicking on another node's input or output, followed by clicking an empty space in the **Node Graph**.*

Selecting Nodes

Katana offers a number of options for selecting nodes. Selected nodes are highlighted in yellow.

Selecting a single node

Click once on the node.

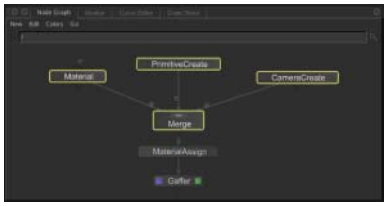
Selecting multiple nodes

Press **Shift** while clicking on each node you want to select.
OR
Drag on the **Node Graph** to draw a marquee. Katana selects all nodes inscribed by the marquee.

Selecting all nodes upstream

You can select all the nodes upstream of the currently selected node(s). To do this:

1. Click on a node.
2. Press **Ctrl**+**Up Arrow**.

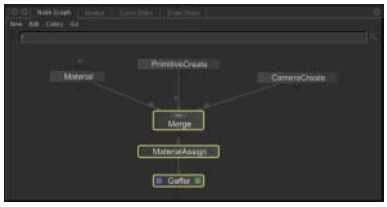



Katana selects all nodes that feed data to the selected node.

Selecting all nodes downstream

You can select all the nodes downstream of the currently selected node(s). To do this:

1. Click on a node.
2. Press **Ctrl**+**Down Arrow**.

Katana selects all nodes downstream from the selected node.

Adding to a selection

Shift+click to select more nodes without clearing the current selection.

Deselecting a node

To deselect a node:
Shift+click.

Connecting Nodes

As you build up a scene, you'll need to create connections between nodes or change the connections that already exist. Any nodes that are not connected to the overall node tree do not have any effect.

Nodes have input and output ports that are used to connect one node to another. Input ports are rectangles, usually located at the top of a node. Output ports are triangles, usually located at the bottom.

Connecting a Node into the Recipe

There are a number of different ways to connect a node into the recipe, you can:

1. Click the output port of the first node you want to connect.
2. Drag the resulting arrow to the input port of the second node.
3. When the input highlights in yellow, release the mouse button.

OR

1. Hover the cursor over the first node you want to connect.
2. Press the **Backtick** key (`) once.
3. Hover the cursor over the second node and press the **Backtick** key again.

OR

1. Drag one node over the input or output of a second node, and release the mouse button to establish a connection.
2. Click on an empty space in the **Node Graph** to then place the node there.

Adding a node between two other nodes

1. Drag the node into the space between two already connected nodes.

   When the cursor is over the connection, the connection becomes active (turns yellow).
2. Release the node you are dragging.

   It automatically wires itself into the network between the two nodes.

Removing a Node from the Recipe

There are two different ways to disconnect a node without deleting it:

- remove its input/outputs manually, or
- extract it—which removes all connections and attempts to repair the recipe by connecting the nodes that are around the extracted node.

Disconnecting a node

To disconnect a node, drag the head or tail of the connecting arrow to an empty area of the workspace.

Extracting a node

You can remove all the connections to a node, extracting it from any recipes without deleting it. To extract a node:

1. Select the node you wish to extract.
2. In the **Node Graph**, select **Edit > Extract Selected Nodes** (or press **X**)

   This removes all connections from the selected node, extracting it from the recipe.

Tidying the Recipe with a Dot node

Dot nodes are used to help tidy a recipe and make the flow of the connections clearer.

They also have a unique ability in that disabling a Dot node ignores the contribution of all the nodes upstream.

To insert a Dot node:

1. Decide where to place the Dot node by:
   - selecting the node before the connector you want to bend, or
   - hovering the mouse over the connection you wish to bend.
2. Press **.** (**Full stop**) to create a Dot node.
3. Drag the Dot node as necessary to reposition the connections.

Tips *You can also create the Dot node in the same way as any other node (through the **Tab** menu, **New** menu, or right-click menu) and connect it manually.*

*You can create a Dot node while connecting two nodes. With the connector connected at one end, press **.** (**Full Stop**) to place the Dot at the current mouse location, then continue as normal.*

Replacing Nodes

Replacing one node with another

You can use the **R** key to replace a node using the same **Tab** key menu. To replace a node using the **R** key:

1. In the **Node Graph**, select the node you want to replace.
2. Press **R** and start typing the name of the node you want to create.

   Katana displays a list of matches.

3. To select the node you want to add from the list, either:
   - click on it, or
   - scroll to it with the **Up** and **Down** arrow keys and press **Return**.

   The new node replaces the selected node in the **Node Graph**.

Copying and Pasting Nodes

To copy, paste, and perform other editing functions in the node tree, you can use the standard editing keys (for example, **Ctrl**+**C** to copy, and **Ctrl**+**V** to paste). Copied nodes inherit the values of their original, but these values, unlike those in cloned nodes (see below), are not actively linked—that is, you can assign different values to the original and the copy.

Copying nodes to the clipboard

1. Select the node or nodes you want to copy.
2. In the **Node Graph**, select **Edit** > **Copy** (or press **Ctrl**+**C**).

Pasting nodes from the clipboard

In the **Node Graph**, select **Edit** > **Paste** (or press **Ctrl**+**V**).
Katana adds the nodes to the scene.

Cutting nodes from the Node Graph

1. Select the node or nodes you want to cut.
2. In the **Node Graph**, select **Edit** > **Cut**.

   Katana removes the node(s) from the scene and writes the node(s) to the clipboard.

Cloning Nodes

You can clone nodes and place them elsewhere in a recipe. Cloned nodes inherit the values of their parent, but unlike copied nodes, they also maintain an active link with their parents' values. If you alter the values of the parent node, the clone automatically inherits these changes.

To clone nodes:

1. Select the node or nodes you want to clone.
2. In the **Node Graph**, select **Edit** > **Clone**.

   Katana clones the node or nodes and creates an expression between each parameter of the parent node and that of the clone. Any change on the parent is therefore reflected in the child.

To declone nodes:

1. Select the node or nodes you want to declone.
2. In the **Parameters** tab, select  > **Reset Parameters**.

   Katana removes the clone status of the selected nodes and resets all its parameters to the nodes' defaults.

Disabling Nodes

Disabling and re-enabling nodes

You can toggle a node between enabled and disabled. To toggle whether a node is enabled:
Hover over the node and press **D**.
OR

1. Select the node or nodes.
2. In the **Node Graph**, select **Edit** > **Toggle Ignore State of Selected Nodes** (or press **Alt**+**D**).

Deleting Nodes

Deleting selected nodes

1. Select the node or nodes you want to delete.
2. Press **Delete**.

   Katana removes the node(s) from the scene.

Deleting all nodes not contributing to the current Scene Graph

In the **Node Graph**, select **Edit** > **Delete All Non-Contributing Nodes**.
Disabled nodes that would contribute if enabled are not deleted.

Navigating Inside the Node Graph

As recipes grow in complexity, you need to be able to move between clusters of nodes quickly. Katana offers various methods for doing so.

Panning

Panning with the mouse

Middle-click and drag the mouse pointer over the workspace. The recipe moves with your pointer.

Zooming

You can zoom in on or out of the recipe in a number of ways.

Zooming in

Move your mouse pointer over the area you want to zoom in on, and press **+** (**Plus** key) repeatedly until the workspace displays the recipe at the desired scale.
OR
Alt+Middle-click and drag right.
OR
Move the mouse pointer over the area you want to zoom in on, and scroll up with the mouse wheel.

Zooming out

Move your mouse pointer over the area you want to zoom out from, and press **-** (**Minus** key) repeatedly until the workspace displays the recipe at the desired scale.
OR
Alt+Middle-click and drag left.
OR
Move the mouse pointer over the area you want to zoom out from, and scroll down with the mouse wheel.

Note *On Linux, **Alt+Middle**-click and drag may zoom the entire Katana window instead of the **Node Graph**. This is the default functionality on Gnome. To get around it, you can use the **Windows** key instead of **Alt** when zooming.*

Alternatively, you can change your window preferences on Gnome to fix the problem:

*1. Select **System** > **Preferences** > **Windows** to open the Window Preferences dialog.*

*2. Under **Movement Key**, select **Super ("or Windows logo")**.*

*You should now be able to zoom in and out of the **Node Graph** using the **Alt** key.*

Fitting Selected Nodes in the Node Graph

To fit selected nodes in the **Node Graph**, press **F**. If no nodes are selected then the entire node tree fills the **Node Graph**.

Fitting the Node Tree in the Node Graph

To fit the entire node tree in the **Node Graph**, press **A**.

Improving Readability and Navigation with Backdrop Notes

You can use Backdrop Notes to help document your recipes, making them easier to read and navigate. They can be placed at the side of important nodes to explain their use for future users, around a collection of nodes that perform a particular function, or just as a title for your entire recipe. How you use them is up to you!

Creating a Backdrop Note

A Backdrop Note is created in the same way as any other node, through the **Tab** key menu, the right-click menu, or with the **New** menu within the **Node Graph**. As well as these methods you can also create a Backdrop Note around a number of nodes using the method below.

To fit a Backdrop Note around the currently selected nodes:

1. Select the nodes the Backdrop Note is to encompass.

   A minimum of two nodes must be selected.
2. Select **Edit > Fit Backdrop to Selected Nodes**.

   If you select a Backdrop Note with the selected nodes, Katana uses that Backdrop Note, otherwise a new Backdrop Note is created.

Editing a Backdrop Note

To change the parameters of a Backdrop Note:

1. Double-click within the horizontal lines at the top of the node.

   This brings up the **Edit Backdrop Note** dialog.

2. In the dialog you can:
   - Enter or edit the text in the main text box.
   - Change the size of the text with **fontScale**.
   - Change the background color.
   - Toggle whether this Backdrop Note should be part of the jump-to menu with **Show In Bookmarks** (See Navigating with Backdrop Notes below).
   - Toggle whether this Backdrop Note should be drawn behind other notes with **Send to Back**.
3. Click **Ok** to save changes.

Tip *The first line of a Backdrop Note is used as its title for the **Jump to Bookmark** menu mentioned below.*

Resizing a Backdrop Note

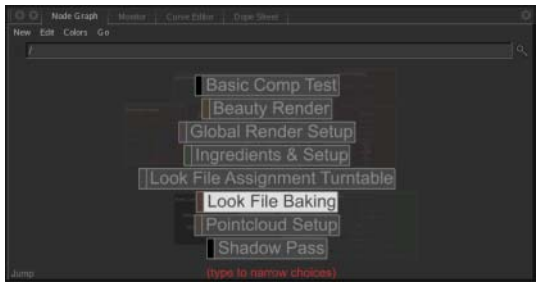
You can resize a Backdrop Note by dragging from the bottom right corner.

Navigating with Backdrop Notes

One extremely useful function of Backdrop Notes is their ability to act as jump to points throughout a project.

1. In the **Node Graph**, select **Go > Jump to Bookmark** (or press **J**) to bring up the Backdrop Notes jump to menu.

   Katana displays all the Backdrop Notes that have the bookmark flag enabled with their background color displayed to the left.



2. Start typing the name of the note you wish to navigate to.

   This narrows down the displayed list.
3. To select the Backdrop Note to navigate to, either:
   - click on it, or
   - scroll to it with the **Up** and **Down** arrow keys and press **Return**.

Selecting nodes within a Backdrop Note

You can select all nodes within the bounds of a Backdrop Note (as well as the note itself) by **Ctrl**+clicking within the two horizontal bars at the top of the note.

Locking and unlocking Backdrop Notes

To lock Backdrop Notes so they can't be edited or selected, select **Edit > Lock All Backdrop Notes**. All Backdrop Notes are locked, but if you create a new Backdrop Note it is not locked.

To unlock all Backdrop Notes, select **Edit > Unlock All Backdrop Notes**.

Editing a Node's Parameters

Each node has parameters that alter how the node behaves within the recipe. These parameters can be changed within the **Parameters** tab.

A parameter's value comes from one of three things:

- A constant.
  For example: 9, test, or /root/world/cam/camera
- An expression.
  For example: 16-3, scenegraphLocationFromNode(getNode('CameraCreate')), or getNode('CameraCreate').fov. See Appendix B: Expressions.
- A curve—only available for numeric inputs. See Animating Within Katana.

Accessing a Node's Parameters

To edit a node's parameters, they need to be in the **Parameters** tab. To do this, you can:

1. Select the node(s) whose parameters you want to edit.
2. In the **Node Graph**, select **Edit > Edit Selected Nodes** (or press **Alt+E**).

OR

1. Hover the mouse pointer over the node you wish to edit.
2. Press the **E** key.

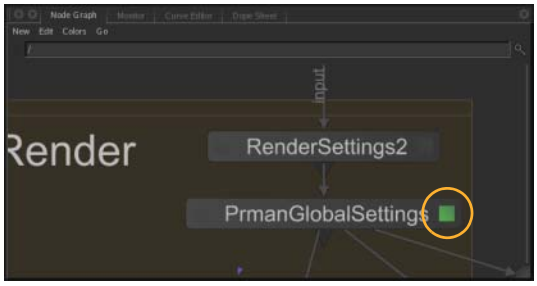
OR

Click within the faint square to the right of a node.

OR

Double-click on a node. (This also sets the current **Scene Graph** view to that node. See Using the Scene Graph.)

A node that has its parameters in the **Parameters** tab has a green square on the right hand side.



Opening and Closing a Node's Parameters

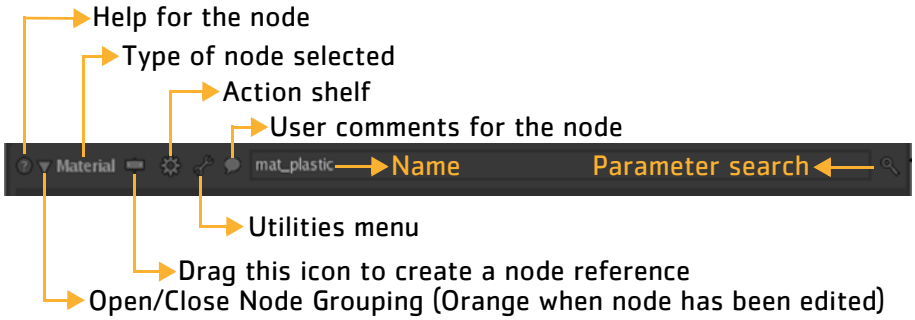
Once a node's parameters are visible within the **Parameter** tab they are grouped with the node type and name at the top. This can be opened and closed with the / icons next to the node type.

Note *If the **Parameter** tab is not visible you can either:*

- *add it to a pane by clicking the*  *icon on the relevant pane and selecting **Parameters**, or*
- *create a new floating pane, by clicking **Tabs > Parameters**.*

Default Parameters Tab Icons

Nodes displayed in the **Parameters** tab have a number of default icons.



Changing a Node's Name

Different nodes have their name edited in different ways. The two most common places to get the node name are:

- The name in the input field at the top, next to the node type.

- One of the parameters—for instance the **passName** in the **Render** node, or the **name** in the Material node.

Changing a Node's Parameters

Each parameter type has a control associated with it. How to make changes to some of the more common parameter types is listed below.

Changing a numeric value

You can change a numeric value by:

- Entering a new value in the input field.
- Pressing the **Up Arrow** key to increment its value, or **Down Arrow** to decrement.
- **Click**+dragging on the parameter name, also known as scrubbing. Dragging to the left decreases the value, and dragging to the right increases.

Tips *To make the changes coarser, hold down the **Shift** key while scrubbing, to make them finer, hold down the **Ctrl** key.*

*Pressing **Shift** with the up and down arrows makes the change coarser, or pressing **Ctrl** makes it finer. Also, to change the increment and decrement amount, right-click and select the sensitivity from the **Sensitivity** menu.*

Changing a color value

Use the color picker or the pixel probe to change the color.

Changing the value of a dropdown menu

To change the value in a dropdown menu:

1. Click on the dropdown menu.
2. Then, either:
   - Click on the new value from the list.
   - Use the **Up** and **Down Arrow** keys to highlight the new value and press the **Return** key.

Changing a text string

A string can be used to represent a texture name, Scene Graph location, node name, or whatever a plug-in may need. Depending on what it is representing it can be displayed in a number of ways. These can be:

- a plain text input field, or
- a Scene Graph location, or
- a filename.

Manipulating a Scene Graph location parameter

Scene Graph location parameters are used to either point to where a new location is inserted into the **Scene Graph** or to reference an existing location.

When the node creates a new location within the **Scene Graph**, the  icon presents you with common path prefixes to aid in placing the new location.

When the node modifies an existing location, the  icon allows you to get the path from either:

- the current **Scene Graph** selection, or
- the current **Node Graph** node selection.

  If you choose the second option, Katana creates an expression that points to the **Scene Graph** location created by the selected node.

To find the location that the parameter references and select it within the **Scene Graph**, click  .

Assigning locations to a CEL parameter

CEL parameters can be made up of one or more statements. Each statement can be one of three things:

- a path,
- a collection (a CEL statement stored on a Scene Graph location), or
- a custom CEL statement.

For more on valid CEL statements, see Appendix C: Collection Expression Language & Collections.

Parameter and Attribute Icons

Some parameters, and all attributes, have an icon to help you determine how the current value is being assigned.

Icon	What it means
	This parameter or attribute has not been set and is getting its value from a predefined default.
	This parameter is being forced to use the predefined default value.
	This parameter has a local change and is being set at this node.
	This parameter or attribute has been set and is not getting its value from the default. A parameter with this icon would have already been set further up the node tree.
	This attribute is inherited from a parent location further up the Scene Graph hierarchy.
	This parameter or attribute has an active reference to a parameter in another file. Changes to the other file update this parameter when reloaded.
	This attribute is currently being updated. The displayed value is an estimate and may change when the update is complete.

Customizing How a Node is Displayed

You can change how a node is displayed to improve clarity and readability and to provide additional information about a node's behavior.

Changing a Node's Background Color

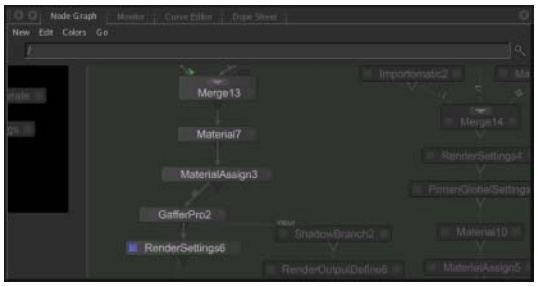
To change a node's background color:

1. Select the node or nodes to change.
2. In the **Node Graph**, select **Colors > [Color]**.

Note *If in the **Node Graph, Edit > Dim Nodes Unconnected to View Node** is selected, or a node is ignored, its background color does not change.*

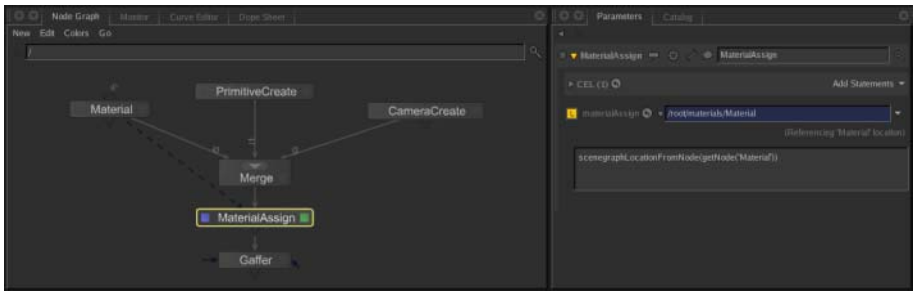
Dimming Nodes not Connected to the View Node

To improve visibility you can dim all nodes not relevant to the currently viewed **Scene Graph**. In the **Node Graph**, select **Edit > Dim Nodes Unconnected to View Node** (or press **Alt+D**).



Displaying Nodes Linked by an Expression

Some nodes are linked to other nodes through expressions. To display this relationship with a dark dashed line in the **Node Graph**, select **Edit > Show Expression Links** (or press **Q**) from within the **Node Graph** tab.



The **materialAssign** parameter of the MaterialAssign node uses an expression to get the **Scene Graph** location created by the Material node.

Drawing the Node Graph with Reduced Contrast

To reduce the contrast around nodes and their connections, in the **Node Graph**, select **Edit > Draw Graph with Low Contrast**. You can do this in conjunction with dimming unconnected nodes.

Aiding Project Readability with Node Icons

By default, some nodes have icons displayed to their left making it clearer what their function is. This behavior is toggled within the **Preferences** dialog.

To toggle node icons:

1. Select **Edit > Preferences...** to bring up the **Preferences** dialog.

2. Click **nodegraph** in the list on the left.
3. Change **showNodeIcons** to **Yes** to display the icons, or **No** to hide.
4. Click **Ok** to make the changes permanent.

Image Thumbnails

Thumbnails provide a guide to the image generated at a particular node within the recipe. Most 2D nodes can display thumbnails, as can the Render node. Although some nodes display thumbnails by default, others need it activated.

To toggle thumbnail display for thumbnail capable nodes:

Right-click and select **Display Thumbnail**.

To update a thumbnail:

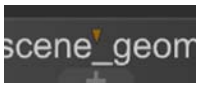
Right-click and select **Regenerate Thumbnail**.

Note *Thumbnails don't update automatically!*

Indicators on Nodes

There are several indicators that can appear on the nodes in the **Node Graph**. The following table describes what each indicator means.

Indicator	What it means
	This node is selected.
	This node's parameters are being edited in the **Parameters** tab.
	This node is being viewed. The Scene Graph generated up to this node is currently displayed in the **Scene Graph** tab.
	This node is disabled.
	Edits to the currently selected location using an interactive manipulator within the **Viewer** tab are fed back to this node.
	An error occurred in the processing of the **Scene Graph** at this node.

Indicator	What it means
	An error occurred in the processing of the **Scene Graph** at a node within this node. *Tip: To see which node, **Ctrl+middle**-click on the node to view inside.*
	Edits to the currently selected location using an interactive manipulator within the **Viewer** tab are fed back to the node inside this node. *Tip: To see which node, **Ctrl+middle**-click on the node to view inside.*
	A node inside this node has its parameters being edited in the **Parameters** tab. *Tip: To see which node, **Ctrl+middle**-click on the node to view inside.*
	A node inside this node is being viewed. The Scene Graph generated up to that node is currently displayed in the **Scene Graph** tab. *Tip: To see which node, **Ctrl+middle**-click on the node to view inside.*

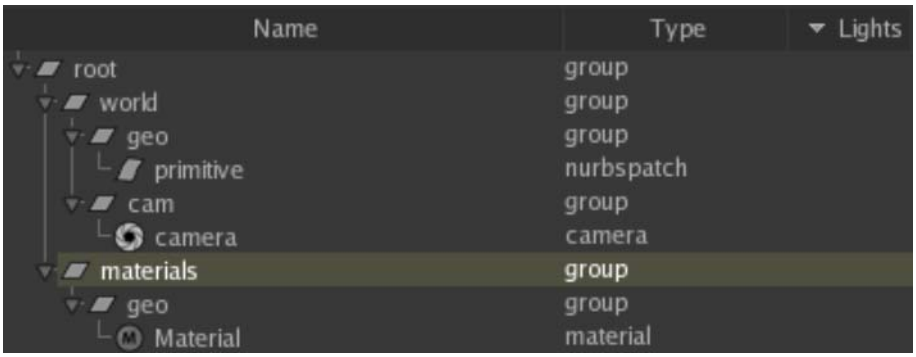
6 Using the Scene Graph

Overview

The **Scene Graph** is a hierarchical structure that represents the scene generated by stepping through the recipe up to the node in the **Node Graph** with the purple square. The node with the purple square is sometimes referred to as the **view node**, this is because the Scene Graph is just a view of the 3D scene generated up to that node. The information within the Scene Graph contains—but is not limited to—geometry, materials, lights, cameras, and render settings. Each node within the **Node Graph** describes a step within the recipe which adds, deletes, or modifies Scene Graph locations or Scene Graph data.

Scene Graph data is stored as attributes on locations.

Scene Graph Terminology



The selected location has a **path** of /root/materials.
The location /root is the **parent** of /root/materials.
The location /root/materials/geo is a **child** of /root/materials.
The location /root/world is a **sibling** of /root/materials.
The location /root/materials/geo/Material is a **leaf** of /root/materials. A leaf is a location with no children.
The locations /root/world and /root/materials are two **branches** from /root.

Locations within Katana have a special attribute called **type**. This attribute tells Katana what type of information to expect at that location. In the example above, there are five group locations and one geometry material location.

Viewing the Scene Graph

You can view the **Scene Graph** generated at any node within the **Node Graph**. This shows the 3D scene generated by the recipe up to that point. To view the **Scene Graph** at a particular node:

1. Select the node in the **Node Graph**.
2. In the **Node Graph**, select **Edit > View Selected Node**.

OR

1. Hover the mouse over the node.
2. Press the **V** key.

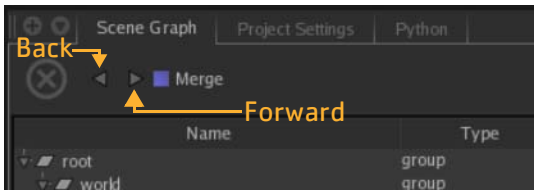
OR

Click within the faint square to the left of the node.

Note *A purple square highlights the current node in the **Node Graph** from which the Scene Graph is generated. This node is known as the **view node**. If the node moves off the screen, or is hidden within another node, its location is indicated by a small purple triangle.*

Navigating the Scene Graph History

Katana keeps a history of the **view node** which can be traversed. To go back and forward through the history, use the icons in the upper left area of the Scene Graph tab.



Manipulating the Scene Graph

Katana's **Scene Graph** is designed to work with scenes of almost unlimited complexity by only displaying the elements that are needed. By default the **Scene Graph** starts with its locations collapsed, so only **/root** is visible.

Selecting and Deselecting Locations in the Scene Graph

Selecting multiple Scene Graph locations

1. Select the first location.
2. **Shift+click** a second location.

   Katana selects both locations and all in between locations that are visible within the **Scene Graph**.

OR

1. Select the first location.

2. **Ctrl+click** the locations to add.

Selecting the parent of the selected location(s)

1. **Right-click** on the selected location(s).
2. Select **Select > Select Parents**.

Selecting the children of the selected location(s)

1. **Right-click** on the selected location(s).
2. Select **Select > Select Children**.

Selecting the leaves of the selected location(s)

1. **Right-click** on the selected location(s).
2. Select **Select > Select Visible Leaves**.

Inverting the selection with its siblings

1. **Right-click** on the selected location(s).
2. Select **Select > Invert Selection**.

Selecting the material location assigned to the currently selected location

1. Select a location with a **materialAssign** attribute.
2. **Right-click** on the selected location.
3. Select **Select > Select Assigned Material Location**.

   Katana selects the location of the material that is assigned at this location. That material location is stored in the **materialAssign** attribute.

Deselecting a location

Use **Ctrl+click**.

Selecting Locations with the Search Facility

Katana **Scene Graphs** can get extremely complicated. To make it easy to find the location you need, Katana has a search facility.

To use the search facility:

1. Click  to bring up the search dialog.
2. To narrow the search results you can:

- Select the type of locations to search for in the dropdown at the top of the dialog:

  Selected

  Pinned

  Cameras

  Lights

  All

- Type text in the **Filter** field to narrow the search to only include locations with matching text.

3. To select a location, select its path within the dialog.

   OR

   To select all the locations displayed in the dialog, click **Select All Matching**.

Note *Only locations that are exposed within the Scene Graph are searched.*

Expanding the Scene Graph

Expanding the Scene Graph completely below a location

1. **Right-click** on the location to expand.
2. Select **Expand All**.

Note *Use with caution on big scenes!*

Expanding to a limited level of the Scene Graph

Assemblies, components, and lod-group (level of detail group) locations are special locations designed to help organize complicated Scene Graphs. They are explained in greater depth at Making Use of Different Location Types and Proxies.

1. **Right-click** on the location to expand.
2. Select the level of the **Scene Graph** to expose:
   - **Expand To > assembly, component or lod-group**

     Expands the Scene Graph from the selected location until it reaches either an assembly, component, or lod-group location. If none are found down a Scene Graph branch, it expands to the leaf location.

     Note: This is the same as **double-clicking** a Scene Graph location.

- **Expand To > component**

  Expands the Scene Graph until it reaches a component location. If none are found down a Scene Graph branch, it expands to the leaf location.

- **Expand To > assembly**

  Expands the Scene Graph until it reaches an assembly location. If none are found down a Scene Graph branch, it expands to the leaf location.

- **Expand To > lod-group**

  Expands the Scene Graph until it reaches an lod-group location. If none are found down a Scene Graph branch, it expands to the leaf location.

Expanding a location only one level

- Click  next to the location name.

OR

1. **Right-click** on the location to expand.
2. Select **Expansion > Open**.

Collapsing the Scene Graph

Collapsing a location and all its children

1. **Right-click** on the location to collapse.
2. Select **Close All**.

Collapsing a Scene Graph location

- Click  next to the location name.

OR

1. **Right-click** on the location to collapse.
2. Select **Expansion > Close**.

Collapsing the Scene Graph completely

- **Right-click** on **/root** and select **Close All**.

OR

- Click  **> Reset Scenegraph**.

Bookmarking a Scene Graph State

Katana allows you to save what parts of the current Scene Graph are open using **bookmarks**. This feature is extremely useful as Katana only loads the locations that are exposed and this allows you to return to a carefully organized Scene Graph state.

Saving a Katana project also saves its bookmarks.

Creating a Scene Graph bookmark

1. Click  **> Create Bookmark**.

   The **Create Scene Graph Bookmark** dialog appears.
2. Type a bookmark name in the **Bookmark name** field or select **Last File Save** from the dropdown.
3. To create the bookmark within a folder, type the folder name in the **Create in folder** field.
4. Select the information to include within the bookmark:
   - **Save Open State** stores which locations are open or closed.
   - **Save Selection State** stores which locations are selected.
   - **Save Pin State** stores which locations are pinned. For more on pins, see Changing What is Shown in the Viewer.
5. Click **Save** to complete the bookmark.

Deleting unused bookmarks

1. Click  **> Organize Bookmarks ...**

   The **Organize Scene Graph Bookmarks** dialog appears.
2. **Right-click** on the bookmark to delete.
3. Select **Delete**.
4. Click  in the top right of the dialog.

Exporting and Importing Bookmarks

You can export your bookmarks for use in other Projects.

Exporting the Project's bookmarks

1. Click  **> Export Bookmarks ...**

   The **Select a File** dialog appears.
2. Choose a location and filename.

3. Click **Accept**.
   A file containing the bookmarks is saved.

Importing previously exported bookmarks

1. Click  **> Import Bookmarks ...**
   The **Import Scene Graph Bookmarks** dialog appears.
2. Navigate to the bookmarks file to import.
3. Click **Accept**.
   The bookmarks are loaded into the Project.

Viewing a Location's Attributes

To view the attributes stored at a location within the **Scene Graph**, select the location within the **Scene Graph** and the attributes appear in the **Attributes** tab. The **Attributes** tab is read-only.

Changing What is Shown in the Viewer

The **Viewer** tab is a 3D representation of the scene currently open in the Scene Graph. Part of Katana's ability to deal with extremely complex scenes comes from it only loading Scene Graph data when it is needed.

The **Viewer** tab only shows locations that are exposed in the Scene Graph and pinned locations.

Note *If your **Viewer** is empty, your Scene Graph is probably closed and no locations with geometry are visible.*

Pinning a location or locations

1. Select the location(s) to pin.
2. **Right-click** and select **Pin > Set Local Pin**.

Pinning all the visible leaves

1. Select the top level location(s) to pin the leaves.
2. **Right-click** and select **Pin > Pin Visible Leaves**.
   Katana descends the **Scene Graph** from the selected location(s) and pins all the leaf locations.

Clearing the pin at a location or locations

1. Select the location(s) to remove the pin.
2. **Right-click** and select **Pin > Remove Local Pin**.

Clearing the pins below a location or locations

1. Select the top level location(s) to remove the lower level pins.
2. **Right-click** and select **Pin > Clear Pins Below**.
   Katana descends the **Scene Graph** from the selected location(s) and removes any pins.

Disabling Scene Graph Updates

To keep the current **Scene Graph** view, and not change its contents as the **View Node** changes, click  .

Rendering only Selected Locations

For speed it is sometimes preferable to only render a subset of the objects within a scene. To limit the objects being sent to the renderer, select the objects in the Scene Graph and click  .

Note *This is only for interactive renders. Performing a hotrender uses the entire Scene Graph.*

Turning on Implicit Resolvers

Katana defers some procedures, such as a material copy, until they are needed by the renderer. This deferring has a number of positive results:

- It speeds up the initial Scene Graph generation.
- You can keep everything at a higher level making it easier to edit and override. For instance, you can change what material is at a given location rather than having to edit or override all the individual shader values.

Some examples of procedures that are deferred are:

- The copying of all the material details to a location.
- The copying of all the texture details to a location.

These deferred procedures are also known as implicit resolvers. To turn on implicit resolvers click  .

Making Use of Different Location Types and Proxies

Only loading what is needed, when it is needed, is a key part of the philosophy of Katana. If you want to position the specular highlight on your main character, for instance, you don't need to load the entire scene. One way to avoid unnecessary loading is to define your scene with special hierarchies and proxies.

The hierarchical scene structure can be created using special location types. Special types that can be used are **assemblies, components, level-of-detail groups,** and **level-of-detail** locations.

Proxies enable you to get a good idea of a scene without opening up too much of the hierarchy. Placing proxies on assemblies and components enables you to open a hierarchy to convenient levels of scene complexity.

Using Assemblies and Components

Assemblies and components help you define convenient points of expansion for the **Scene Graph**. They are usually defined as part of the asset creation process, but you can also define them within Katana. An asset's hierarchy usually consists of an assembly and then below the assembly are other assemblies or components, and below the components is the full geometry data.

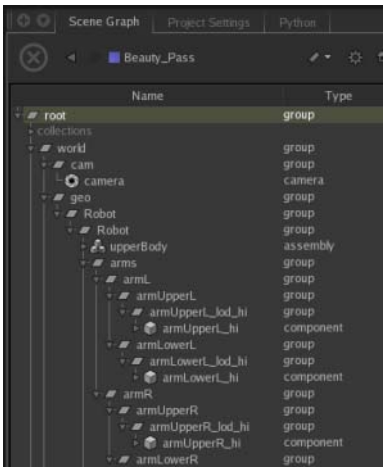


Figure 6.1: A Scene Graph example containing assembly and component locations.

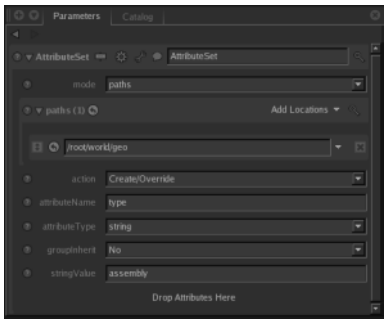


Figure 6.2: A simple example of using an AttributeSet node to change the location type (which is simply the **type** attribute for a location) to **assembly**.

7 Adding 3D Assets

Overview

The most common way to start a recipe is by defining the steps that bring in your 3D assets. Possible assets include static geometry, animated geometry in the form of a geometry cache, a particle cache, or an animated camera from a camera tracking package.

Katana's most common nodes for bringing in scene assets are:

- **PrimitiveCreate**
  The PrimitiveCreate node contains a list of basic geometry shapes used in most 3D packages. These range from simple shapes such as planes and cylinders to teapots and gnomes.
- **CameraCreate**
  A simple node designed to create a camera. You can also import cameras using the Alembic_In node.
- **Alembic_In**
  The Alembic open standard has been adopted by Katana as its preferred means of asset interchange. Alembic is covered in more depth in Adopting Alembic.
- **ScengraphXml_In**
  Katana can also bring in assets defined using XML. With Scene Graph XML you can define level of detail groups, assemblies, and components. For more on these and ScenegraphXml_In, see Describing an Asset Using XML.
- **Importomatic**
  The Importomatic is a one-stop-shop for bringing in assets. It has a plug-in structure enabling assets to be imported from different formats. It ships with plug-ins for Alembic_In and ScenegraphXml_In. To learn more on its use, see Using the Importomatic.
- **ScenegraphGeneratorSetup** and **ScenegraphGeneratorResolve**
  These nodes are used to take advantage of Katana's Scene Graph Generator API. For a brief overview, see Generating Scene Graph Data with a Plug-in. For a more comprehensive explanation, please refer to the development documentation.

Note *Your studio may be using its own geometry format, complete with a custom node to bring that format into Katana.*

Adopting Alembic

Alembic is an open source scene information interchange framework. Alembic distills complex, animated scenes into non-procedural, application-independent, baked geometric results. It stores only the baked information and not how that information was obtained. For instance, a fully rigged and animated character would have its vertices efficiently stored for each frame of the animation but the control rig itself would not be stored. You can export to Alembic from most popular 3D applications.

For more information on Alembic, see http://code.google.com/p/alembic/.

Adding an Alembic Asset

To add an Alembic asset:

1. Create an Alembic_In node and add it to your recipe (assets are usually added first to any recipe).
2. Select the Alembic_In node and press **Alt+E**.

   The Alembic_In node becomes editable within the **Parameters** tab.
3. In the **name** parameter, enter the Scene Graph location to place the Alembic data.
4. Enter the asset filename in the **abcAsset** parameter.

Describing an Asset Using XML

XML is a simple way to describe a hierarchical structure. Katana leverages this format to provide a rich descriptive asset language. Through XML, assets can be structured so they are loaded and manipulated in stages. Simpler versions of the asset (proxies) load quicker and use less memory, allowing you to only load the full asset when absolutely necessary.

Some asset elements that you can describe within a ScenegraphXml file are:

- Assembly locations
- Component locations
- Level-of-detail group locations
- Level-of-detail locations
- Other XML locations
- Geometry caches

Adding an Asset Described Using XML

To add an asset described using XML:

1. Create a ScenegraphXml_In node and add it to your recipe (assets are usually added first to any recipe).
2. Select the ScenegraphXml_In node and press **Alt+E**.

The ScenegraphXml_In node becomes editable within the **Parameters** tab.

3. In the **name** parameter, enter the Scene Graph location to place the data.
4. Enter the asset filename in the **asset** parameter.

Note *Providing a full description of how to describe a scene using XML is beyond the scope of the User Guide. For more information, consult the developer's documentation accessed through the **Help > Documentation** menu.*

Using the Importomatic

The Importomatic node is used to bring in multiple assets and—if needed—assign them a look file or attribute file. Packaging this into one node keeps the recipe simpler to understand and debug.

With the Importomatic node you can:

- Read in multiple Alembic and ScenegraphXML assets in a single node.
- Assign look files to any of the assets (for more on look files, see Look Development with Look Files).
- Assign attribute files to any of the assets.
- Branch from the Importomatic node, allowing multiple outputs.

Tip *It is also possible for a studio to expand on the list of Importomatic asset types. For more information, consult the developer's documentation accessed through the **Help > Documentation** menu.*

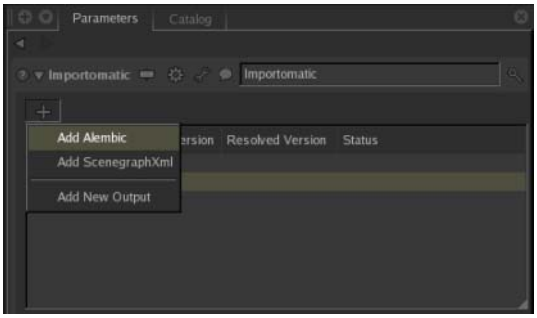
Adding Assets Using the Importomatic

To add assets using the Importomatic:

1. Create the Importomatic node and place it within the project.
2. Select it and press **Alt+E**.
   The Importomatic node becomes editable within the **Parameters** tab.

3. Click  within the **Parameters** tab.

   The asset and output menu is displayed.



4. Select **Add Alembic** or **Add ScenegraphXml** and select the asset from the file browser or asset management browser.

   The new asset is added to the Importomatic's hierarchy.

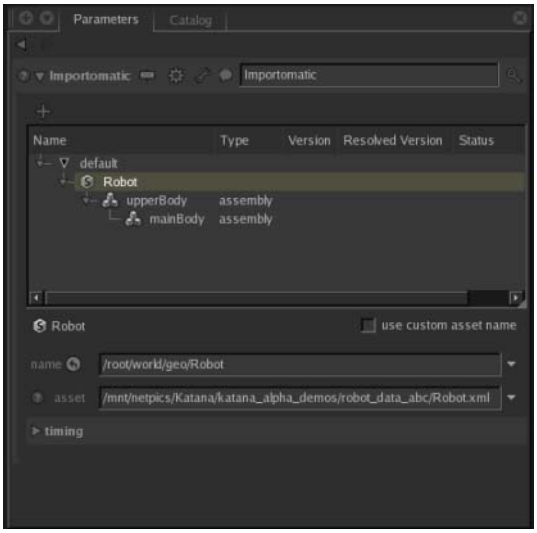
Editing an Importomatic Asset's Parameters

To edit an asset's parameters:

1. Select the asset within the Importomatic's hierarchy within the **Parameters** tab.

   The asset's parameters are displayed below the hierarchy.

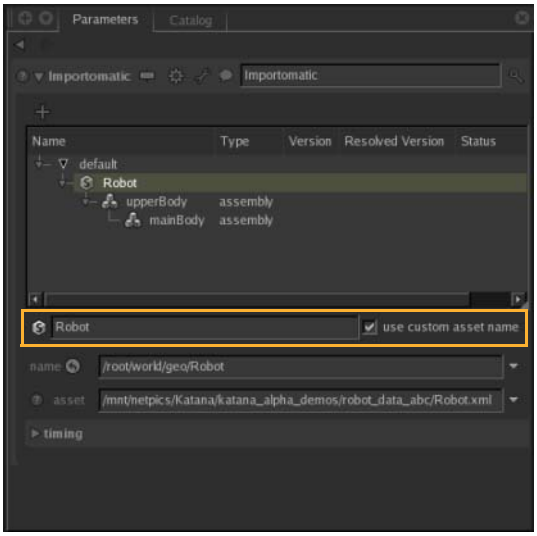
2. Make any changes to the asset that are needed. The parameters that are available are dependent on the asset type.



Editing an Asset's Name

To edit the name:

1. Select the asset within the Importomatic's hierarchy.
   The asset's parameters are displayed below the hierarchy.
2. Toggle **use custom asset name** on.
   The asset name becomes editable.



3. Change the asset name in the field directly below the hierarchy.
   Changing the asset's name within the Importomatic does not influence its location within the **Scene Graph**.

Disabling an Asset

To disable an asset:

1. In the Importomatic parameters, right-click on the asset name within the hierarchy.
2. Select **Ignore Asset** (or press **I**).
   The asset is no longer created.

Enabling an Asset

To enable an asset:

1. In the Importomatic parameters, right-click on the asset name within the hierarchy.
2. Select **Unignore Asset** (or press **I**).

Deleting an Asset from the Importomatic

To delete an asset:

1. In the Importomatic parameters, right-click on the asset name within the hierarchy.
2. Select **Remove Item** (or press **Delete**).

Assigning a Look File to an Asset

To assign a look file:

1. In the Importomatic parameters, right-click on the asset name within the hierarchy.
2. Select **Assign Look File**.

   The file browser or your studio's asset picker appears.
3. Select the look file from the file browser or asset picker.

Assigning an Attributes File to an Asset

To assign an attributes file to an asset:

1. In the Importomatic parameters, right-click on the asset name within the hierarchy.
2. Select **Assign Attribute File**.

   The file browser or your studio's asset picker appears.
3. Select the attribute file from the asset picker or file browser.

Note *Use attribute files to add attributes to existing locations. For a full explanation on using attribute files, see the accompanying PDF, which is accessed through the **Help > Documentation**.*

Adding Additional Outputs to the Importomatic

To add an additional output:

1. In the Importomatic parameters, click  .

   The asset and output menu is displayed.
2. Select **Add New Output**.

   A new output is added to the node and hierarchy.

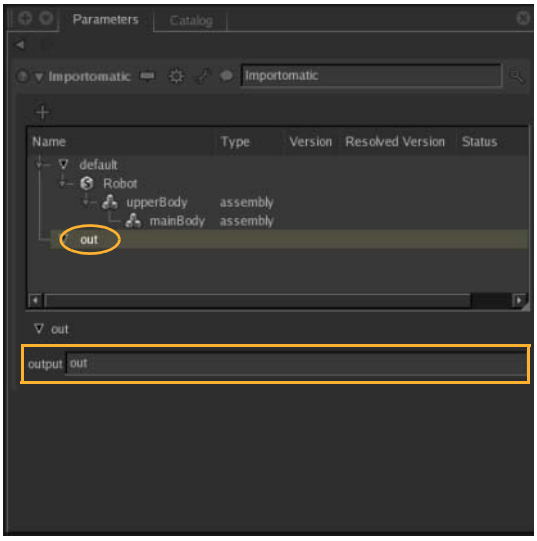
Changing an Output's Name

Apart from the default output, the outputs from the Importomatic can be changed.

To change the name of an output:

1. In the Importomatic parameters, select the output in the hierarchy.

2. Type the new output name in the **output** parameter.



Generating Scene Graph Data with a Plug-in

Using the Scene Graph generator nodes, ScenegraphGeneratorSetup and ScenegraphGeneratorResolve, you can write a plug-in to generate locations and attributes. The coding of Scene Graph generators is explained within the developer documents that accompany the installation. To access the documentation, select **Help > Documentation**.

The first node of the pairing is ScenegraphGeneratorSetup. You can use this node to place the arguments needed for the plug-in into the Scene Graph. The procedure itself is not executed until either, the recipe reaches the second node, the ScenegraphGeneratorResolve, or the renderer requests it at render time.

Adding a Scene Graph Generator Location to Your Scene Graph

To add a Scene Graph generator location to your Scene Graph:

1. Create the ScenegraphGeneratorSetup node and place it within the project.
2. Select it and press **Alt+E**.

   The ScenegraphGeneratorSetup node becomes editable within the **Parameters** tab.
3. Select the Scene Graph generator from the **generatorType** dropdown.

   The arguments for the Scene Graph generator appear.

The ScenegraphGeneratorSetup node creates a location of type **scene graph**

generator. The attributes at that location include the **scenegraphGenerator.generatorType** (which is the procedure that is loaded) and **scenegraphGenerator.args** (which contains all the arguments needed by the procedure).

Forcing the Scene Graph Generator to Execute

To force the Scene Graph generator to execute:
Create the ScenegraphGeneratorResolve node and place it downstream in the recipe, at the point you want the generator to execute.

Note *Anything below the **scene graph generator** location at the time the generator is resolved gets deleted.*

8 Adding and Assigning Materials

Overview

A material is a Scene Graph location that holds one or more shaders. **Shaders** define how an object (a piece of geometry, a light, an area) interact within a renderer to create an image.

The most common types of material are:

- **light materials** (complete with a light shader), which get assigned to light locations to illuminate a scene, and
- **geometry materials** (with surface shaders and possibly displacement or bump shaders), which get assigned to 3D geometry and particles.

The process of creating a basic material is broken down into two stages (although you only need one node):

1. Create the Scene Graph material location to hold the shaders.
2. Add the shaders to that location.

You can assign one material to multiple lights or pieces of geometry. To define this relationship between a material and its objects, use a MaterialAssign node.

An object with a material assigned keeps a reference to its material on the **materialAssign** attribute. The material is actually copied to the object's location either at render time, or at a MaterialResolve node.

At render time, a number of resolvers are applied automatically. These resolvers perform just-in-time resolving of certain operations that are usually best done at the last minute. These resolvers are called implicit resolvers. This method allows data to remain at a higher level for longer. For more details, see Turning on Implicit Resolvers.

Note *Katana is a renderer agnostic application, and the shader types available depend upon the renderer plug-ins and how they locate their shader libraries.*

Creating a Material

The first stage in creating a material is the creation of that material's location. This is the Scene Graph location that acts as a container for one or more shaders.

To create a material location:

1. Create a Material node and add it to your recipe.

   Materials are usually created in their own branch of the recipe and a Merge node is used to connect them to the rest of the recipe. If you need multiple materials, use a MaterialStack node. See Creating Multiple Materials with the MaterialStack Node.
2. Select the Material node and press **Alt+E**.

   The Material node becomes editable within the **Parameters** tab.
3. Enter the material's name in the **name** parameter.

   Although strictly not needed as Katana handles name clashes gracefully, it is good practise to name the material, as the name is used for both the node name and the material's Scene Graph location.
4. In the **namespace** parameter, enter the location below /root/materials to place the material.

   By default the material is placed below /root/materials in the Scene Graph. If **namespace** is not blank, the material is placed below /root/materials/<namespace>. Some of the most common namespaces are included as a dropdown to the right of the parameter. You can also specify nested namespaces, for instance, if the **namespace** parameter is geo/metals, the material is placed in the Scene Graph below /root/materials/geo/metals.

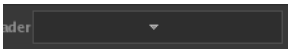
Adding a Shader to a Material Location

A material location needs to have one or more shaders attached.

To add shaders to the material location:

1. Follow steps 1 to 4 in Creating a Material above to create a material location.
2. Click **Add shader** and select a shader type.

   The list of shader types varies depending on the renderers installed.
3. Add a shader to the new shader type's parameter. You can:

   - Click  to the immediate right of the shader type and select the shader from the list.

     OR

   - Browse for a shader with  **> Browse...** and navigate to the shader using the **Shader Browser** dialog, select it and click **Accept**.

4. If you want to set any of the shader's parameters to non-default values, expand the parameters for the shader by clicking  and enter the changes where needed.
5. Repeat steps 2 to 4 for any additional shaders for this material.

   A possible combination might be a surface shader and a displacement shader. Material locations can have shaders from more than one renderer, only shaders for the appropriate renderer are selected at render time. This makes it possible for a single material to control how an object looks in a number of different renderers.

Creating a Material from a Look File

Materials previously baked out into Katana look files can also be assigned to material locations. Look files and the look development process is explained in greater detail in Look Development with Look Files.

Note *This is different from reading in all the materials from a Katana look file, such as a material palette look file created during look development. Material palettes and their creation is covered in* Using Look Files to Create a Material Palette*.*

To use a material from a look file at this material location:

1. Follow steps 1 to 4 in Creating a Material above to create a material location.
2. Select **create from Look File** in the **action** parameter dropdown.
3. Enter the path to the look file in the **lookfile** parameter, or click  > **Browse...**, navigate to the look file and click **Accept**.
4. Select a material from the **materialPath** dropdown list.

   This is the list of materials contained within the look file. The list is automatically populated when a valid look file is assigned to the **lookfile** parameter.
5. If you don't want to import the material as a reference, select **No** for the **asReference** parameter dropdown.

   When Katana imports the material by reference, a reference to the original location of the material is kept. This enables any changes to the original material to be propogated downstream, even if this material is itself baked as part of a look file.
6. If you need to change any parameters, expand the parameters for the shader(s) by clicking  and entering the changes where needed.

Creating a Material That's a Child of Another Material

A child material inherits all the shaders from the parent, but changes you make to the child do not influence the parent.

To create a child material:

1. Follow steps 1 to 4 in Creating a Material above to create a material location.
2. Select **create child material** in the **action** parameter dropdown.
3. Enter the Scene Graph location of the parent material in the **location** parameter within the **inheritsFrom** parameter grouping. See Manipulating a Scene Graph location parameter for details on Scene Graph location parameter fields.

   The child material now has the same attribute values as the parent.

   You can make any changes needed to the parameters in this node without changing the parent. This includes adding additional shaders.

Editing a Material

Once a material is created, it is not locked down. Later in the recipe, you can edit the material using another Material node.

To edit a material location:

1. Create a Material node and connect it to the recipe downstream of the target material.
2. Select the Material node and press **Alt+E**.

   The Material node becomes editable within the **Parameters** tab.
3. Select **edit material** in the **action** parameter dropdown.
4. Enter the Scene Graph location of the material to edit in the **location** parameter within the **edit** parameter grouping. See Manipulating a Scene Graph location parameter for details on Scene Graph location parameter fields.

   The shaders and their current parameter values are displayed below.
5. Edit the shaders for that material location wherever needed. This includes adding additional shaders.

Overriding a Material

As a material location can be assigned to multiple pieces of geometry, sometimes a geometry-specific change is needed. One way to perform this change is to use a material override. You point the Material node at the location(s) to override. Then any changes made are stored on the **materialOverride** attribute of the location.

It is also possible to override material locations directly. In this case, the override acts in the same way as an edit.

You can also override multiple materials at once, but only edit one.

To override the material at a geometry location:

1. Create a Material node and connect it to the recipe downstream of the target material.
2. Select the Material node and press **Alt+E**.

   The Material node becomes editable within the **Parameters** tab.
3. Set the **action** dropdown to **override materials**.
4. Assign the Scene Graph locations of the geometry locations to the **CEL** parameter (located in the **overrides** parameter grouping). See Assigning locations to a CEL parameter for more on using **CEL** parameter fields.
5. In the **Scene Graph** tab, select the material location of the material assigned at the geometry location (or select  **> Select In Scene graph** on the **materialAssign** attribute of the geometry location).
6. **Middle**-click and drag from the attribute to override to the **Drop Attributes Here** hotspot at the top of the **attrs** parameter grouping.

   The attribute displays within the **attrs** parameter grouping and can now be overriden inside the **Parameters** tab.

   All changes you make are added as attributes to the location(s) specified by the **CEL** parameter (under the **materialOverride** attribute).

Creating Multiple Materials with the MaterialStack Node

Having a chain of Material nodes would soon clutter up a recipe. To create multiple materials within one node, use the MaterialStack node.

Adding a Material

To add a material inside the MaterialStack node:

1. Select **Add > Add Material**.

   A new material is added to the **Add** list.
2. Enter a new name in the **name** parameter.
3. Follow steps 2 to 5 in Adding a Shader to a Material Location.

Adding a Material From a Look File

To add a material from a look file inside the MaterialStack node:

1. Select **Add > Add Look File Material**.
   A new material is added to the **Add** list.
2. Enter a new name in the **name** parameter.
3. Follow steps 3 to 6 in Creating a Material from a Look File.

Adding a Material as a Child

To add a material as a child of an existing material:

1. Select a material in the **Add** list.
2. Select **Add > Add Child Material**.
   A new material is added below the selected material.
3. Enter a new name in the **name** parameter.
4. Make any changes needed to the parameters, you can also add additional shaders.

Note *The parent has to be within the MaterialStack node, otherwise the menu options are not available.*

Duplicating a Material

To duplicate a material within the MaterialStack node:
Select the material node in the **Add** list, right-click and select **Duplicate Material**.

Disabling a Material

To disable a material within the MaterialStack node:
Select the material node in the **Add** list, right-click and select **Ignore Material** (or press **I**).

Deleting a Material

To delete a material from the MaterialStack node:
Select the material node in the **Add** list, right-click and select **Delete Material** (or press **Delete**).

Adding a Material Node from the Node Graph

To add Material nodes from the **Node Graph** into the MaterialStack node:
Shift+middle-click and drag the nodes into the **Add** list.

Moving Materials Within the Add List

To move materials within the **Add** list:
Middle-click and drag.

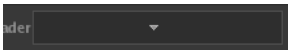
Incorporating PRMan Co-Shaders

RenderMan co-shaders enable shader components to be connected together and plugged into the parameter(s) of one or more shaders. This modular form of shader writing is very powerful.

In order for a shader to make use of a co-shader, the co-shader has to be defined higher (or in the same location) in the Scene Graph hierarchy. For instance, for a material assigned to /root/world/geo/Robot/leg to use a co-shader, that co-shader must be assigned to one of:

- /root/world/geo/Robot/leg
- /root/world/geo/Robot
- /root/world/geo
- /root/world
- /root

Setting up a co-shader material location:

1. Follow steps 1 to 4 in Creating a Material above to create a material location.
2. Click **Add shader** and select **coshader** (under the **prman** heading).
   A new **prmanCoshaders** parameter grouping appears with a co-shader sub-parameter grouping called **shader**.
3. To the right of the **shader** sub-parameter grouping, select  **> Rename...** .
   The **Rename Parameter** dialog appears.
4. Enter a new name for the co-shader and click **Accept** (or press **Return**).
5. Add a co-shader to the **shader** parameter. You can:
   - Click  in the large dropdown and select a co-shader from the list.
     OR
   - Browse for a co-shader with  **> Browse...** and navigate to the co-shader using the **Shader Browser** dialog, select it and click **Accept**.
6. Edit the default parameters where needed.
7. Repeat steps 2 to 6 if additional co-shaders are needed.
   You can create multiple co-shaders in the same node.

Note *You reference the co-shaders inside the main shader by the name selected in step 4 above.*

Assigning Materials

As mentioned in the introduction, a material location needs to be associated with a geometry or light location. This is achieved with the MaterialAssign node.

To assign a material to a Scene Graph location:

1. Create a MaterialAssign node and connect it to the recipe after both the geometry and material locations have been created.
2. Select the MaterialAssign node and press **Alt+E**.

   The MaterialAssign node becomes editable within the **Parameters** tab.
3. Add the Scene Graph locations where the material is to be assigned to the **CEL** parameter. See Assigning locations to a CEL parameter for more on using **CEL** parameter fields.
4. Enter the Scene Graph location of the material to assign in the **materialAssign** parameter. See Manipulating a Scene Graph location parameter for details on Scene Graph location parameter fields.

Tip *The best way to enter a material into the **materialAssign** parameter is to **Shift+middle**-click and drag from the Material node in the **Node Graph** tab to the **materialAssign** parameter. This creates an expression linking the material created by the Material node to the **materialAssign** parameter.*

Assigning Textures

Shaders may be responsible for how a geometry location is rendered, but a lot of the time, the richness of the render comes from a number of asset-specific textures. These textures need to be passed to the shader on a per-asset basis.

Katana uses a render-time script to copy any texture attributes, of the form **textures.<attribute>**, to a shader attribute with the same name **<attribute>**. For instance, if the geometry location has the attribute **textures.ColMap** with value /tmp/colmap_lnf.tx and the **materialAssign** of the same location points to a material with shader attribute **ColMap**, then the shader's **ColMap** attribute becomes /tmp/colmap_lnf.tx at render time. This happens after the material resolve stage. See below for more on material resolving.

Forcing Katana to Resolve a Material

By default, Katana connects a geometry or light location with its respective material using the location's **materialAssign** attribute. This attribute points to where the material is located within the Scene Graph. At render time, an implicit resolver copies the material, pointed to by the **materialAssign** attribute, to the geometry or light's location. For more on implicit resolvers and their benefits, see Turning on Implicit Resolvers.

You can force material resolving at an earlier point within a recipe using the MaterialResolve node.

To force materials to be resolved earlier within the recipe:
Create a MaterialResolve node and connect it to the recipe at the point materials should be resolved.

9 Lighting Your Scene

Overview

Lights are light Scene Graph locations with a light material assigned. The light material contains a shader which defines how that light illuminates the scene.

One strength of Katana is its ability to only load parts of the Scene Graph at render time if they are needed. Lights can potentially be anywhere within the Scene Graph hierarchy. Katana needs to keep a list of all lights so it doesn't need to traverse what could potentially be a very complicated Scene Graph, just to find all the lights. The light list is stored in the **lightList** attribute under the /root/world location.

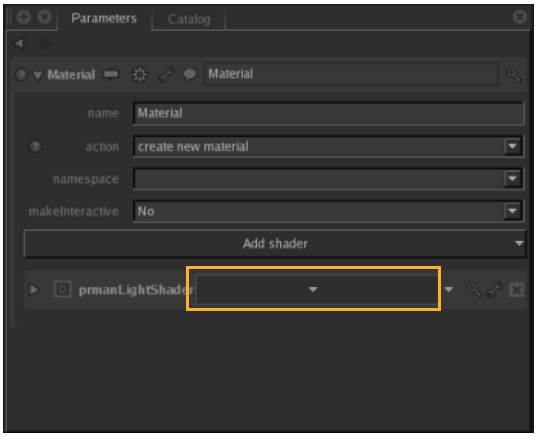
Creating a Light in Katana

Creating a light inside Katana can be done in two ways: using simple nodes (such as LightCreate and Material) or by using the Gaffer node which packages up light creation with a number of other useful functions.

To create a light from its core components:

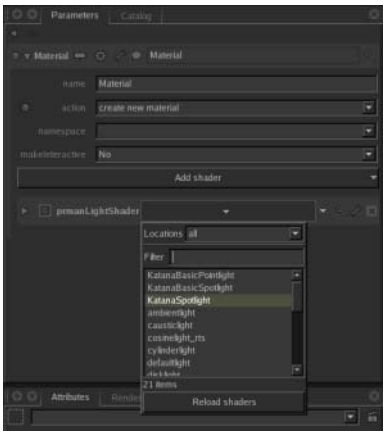
1. Create a LightCreate node and place it within the project.
2. Create a Material node and connect it to the output of the LightCreate node.
3. Select the Material node and press **Alt+E**.

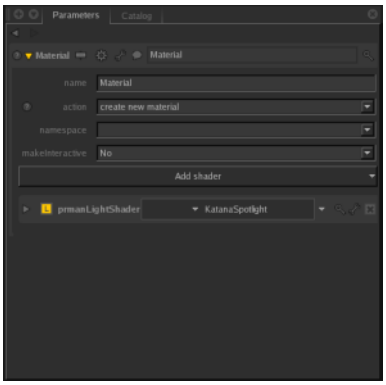
   The Material node becomes editable within the **Parameters** tab.
4. Select **Add shader > prman > light** within the **Parameters** tab.
5. Click next to **prmanLightShader** to display the shader options.



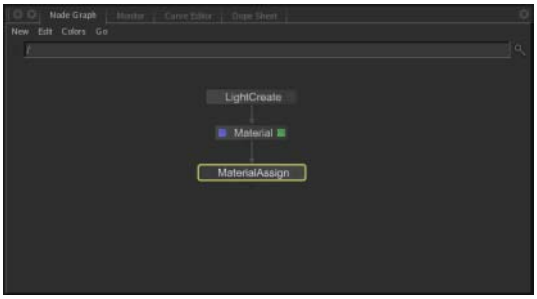
6. Select **KatanaSpotlight** from the dropdown.

   KatanaSpotlight is a simple PRMan light shader that ships with Katana. Depending on your studio's setup, you may need to choose another light shader.





7. Create a MaterialAssign node and connect it to the output of the Material node.

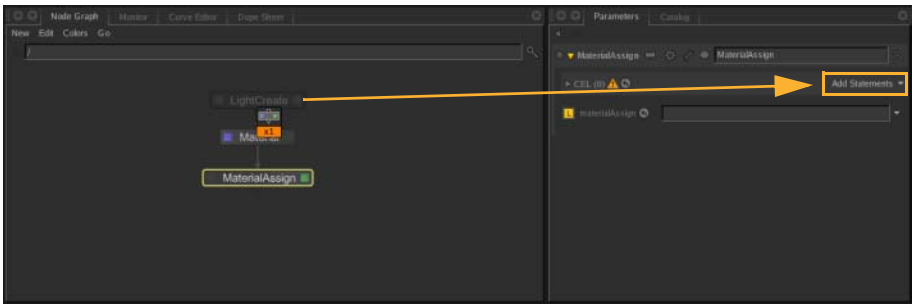


8. Select the MaterialAssign node and press Alt+**E**.

   The MaterialAssign node becomes editable within the **Parameters** tab.

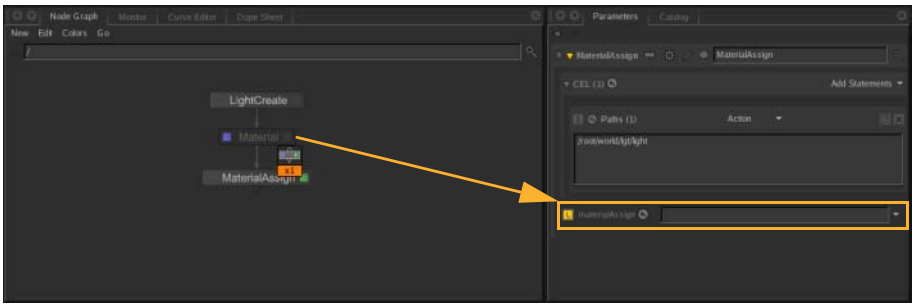
9. **Shift+middle**-click and drag from the LightCreate node in the **Node Graph** tab to the **Add Statements** dropdown in the **Parameters** tab.

   The location created by the LightCreate node becomes a path for the MaterialAssign node.



10. **Shift+middle**-click and drag from the Material node in the **Node Graph** tab to the **materialAssign** field in the **Parameters** tab.

    An expression is created is created for the **materialAssign** parameter that evaluates to the location created by the Material node.



Getting to Grips with the Gaffer Node

The method above, although valid, would be slow for a large number of lights. Katana's Gaffer node wraps this into a single node and adds the ability to:

- Create more than one light.
- Use light profiles for different types of light.
- Add light rigs to group lights together.
- Mute and solo lights and groups of lights.
- Link lights to specific geometry.
- Add aim constraints to lights.

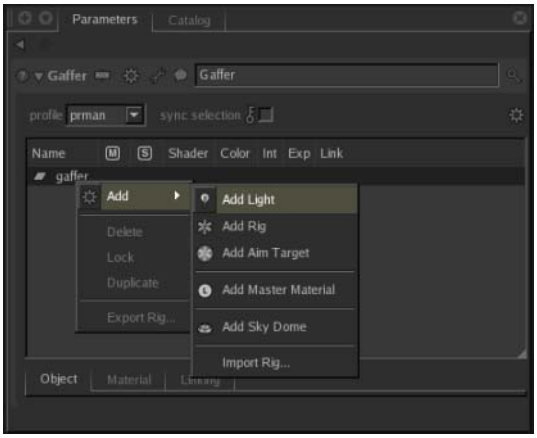
Note *Some of the options listed below may not be available due to the extensive customizability of Katana. Some of the Gaffer node's menu options are created using profiles which can result in different light creation menu options.*

Creating a Light Using the Gaffer Node

To create a light with the Gaffer node:

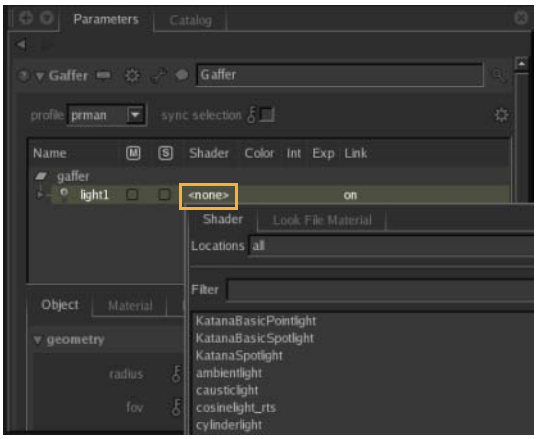
1. Create a Gaffer node and place it within the project.
2. Select the Gaffer node and press **Alt+E.**

   The Gaffer node becomes editable within the **Parameters** tab.
3. Right-click the **gaffer** location in the Gaffer node's parameter hierarchy and select **Add > Add Light**.



4. Click **<none>** below the **Shader** heading in the parameter hierarchy and select **KatanaSpotlight** from the list.

   KatanaSpotlight is a simple PRMan light shader that ships with Katana. Depending on your studio's setup, you may need to choose another light shader.



Making Use of Light Rigs

Light rigs create a Scene Graph group complete with transform attributes and the ability to easily add constraints.

Lights created below the light rig inherit its transformations which enables you to move the lights around as one.

Light rigs can also be exported and imported.

Creating a light rig

Right-click the **gaffer** location in the Gaffer node's parameter hierarchy and select **Add > Add Rig**.

A rig is created below the **gaffer** in the parameter location. It is possible to nest rigs by right-clicking a rig location instead of the **gaffer** location.

Importing a light rig

1. Right-click a location within the Gaffer node's parameter hierarchy, such as **gaffer**, and select **Add > Import Rig...** .

   The **Import Rig** dialog displays.
2. Select the light rig file in the dialog and click **Import**.

   The light rig is imported below the selected location.

Exporting a light rig

1. Right-click on the light rig to export and select **Export Rig...** .

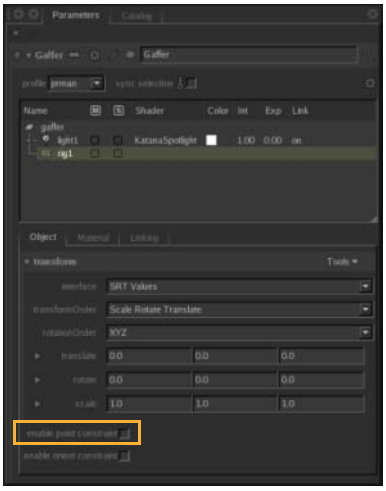
   The **Save Rig** dialog displays.
2. Navigate within the dialog to where you wish to save the light rig and enter a rig name.
3. Click **Save**.

   Light rigs are saved with a .gprig file extension.

Adding a point constraint to a light rig

1. Select the light rig and click the **Parameters > Object** sub-tab.

2. Check **enable point constraint** and open the **point constraint options** parameter grouping.



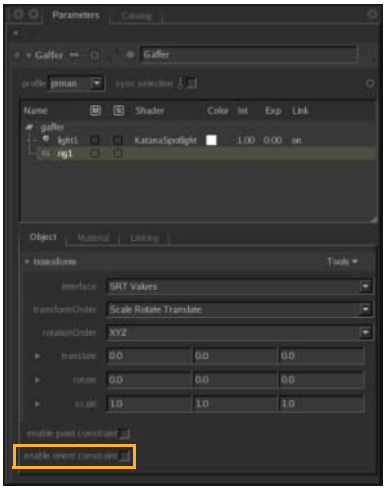
3. Enter the Scene Graph location for the target in the **targetPath** parameter. For more on using Scene Graph location parameters, see Manipulating a Scene Graph location parameter.
4. Select from the **targetOrigin** dropdown which part of the target to use as the point constraint:
   - **Object**—the object's transform position is used.
   - **Bounding Box**—the center of the object's bounding box is used.
   - **Face Center Average**—the center of all the faces for the object are averaged to create the point constraint position.
   - **Face Bounding Box**—the center of the face's bounding box is used.

Adding an orient constraint to a light rig

To add an orient constraint:

1. Select the light rig and click the **Parameters > Object** sub-tab.

2. Check **enable orient constraint** and open the **orient constraint options** parameter grouping.



3. Enter the scene graph location for the target in the **targetPath** parameter. For more on using Scene Graph location parameters, see Manipulating a Scene Graph location parameter.
4. Select the axes to constrain (by default it's all three). To remove the constraint for:
   - the x-axis, select **No** from the **xAxis** dropdown.
   - the y-axis, select **No** from the **yAxis** dropdown.
   - the z-axis, select **No** from the **zAxis** dropdown.

Defining a Master Light Material

At times it is best to have a master material and set local overrides per light. This can be done within the Gaffer node by creating a master material and assigning it to a light. Any changes made within the light's **Material** sub-tab act as an override for the master.

Creating a master material

Inside the gaffer node's hierarchy, right-click and select **Add > Add Master Material**.
A master material location is created inside the Gaffer node. Its shaders and attributes are only visible within the **Scene Graph** when it is assigned to a light.

Assigning a master material to a light

Click **<none>** below the **Shader** heading in the parameter hierarchy in line with the light and select the master material from the list (master materials are displayed in green).

Adding a Sky Dome Light

Sky domes are generally used for image based lighting where an image is placed around the scene and points from the skydome provide illumination. Their exact use depends on the shader assigned.

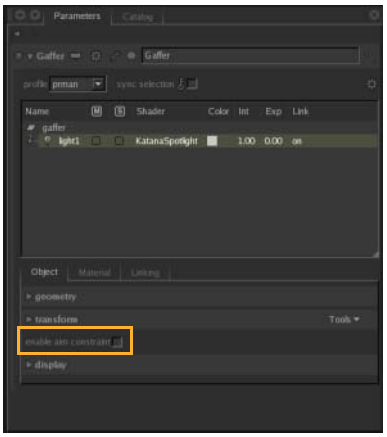
To add a sky dome:
Inside the Gaffer node's hierarchy, right-click and select **Add > Add Sky Dome.**
A sky dome is added to the Gaffer node's hierarchy. The sky dome needs a shader assigned, the shaders available depends on your studio.

Adding an Aim Constraint to a Light

Lights created inside the Gaffer node come with the ability to use an aim constraint. Using an aim constraint makes the light point at an object (the target) within the scene.

Enabling an aim constraint for a Gaffer light

1. Select the light within the **gaffer** hierarchy inside the **Parameter** tab.
2. Select the **enable aim constraint** checkbox within the **Object** sub-tab.
   The **aim constraint options** parameter grouping displays.



3. In the **aim constraint options** parameter grouping, enter the aim target in **targetPath** (for ways to enter a Scene Graph location, see Manipulating a Scene Graph location parameter).

Changing the aim constraint's center point

Select from the **targetOrigin** dropdown:

- **Object**—the point defined by the transform of the object.
- **Bounding Box**—the center of the bounding box.
- **Face Center Average**—the average from all the face centers.
- **Face Bounding Box**—the bounding box of all the faces.

Note *Using **Face Center Average** or **Face Bounding Box** could be slow for heavy geometry.*

Creating an aim target

Inside the Gaffer node's hierarchy, right-click and select **Add > Add Aim Target**.
A locator is created which can be used as the target for an aim constraint.

Linking Lights to Specific Objects

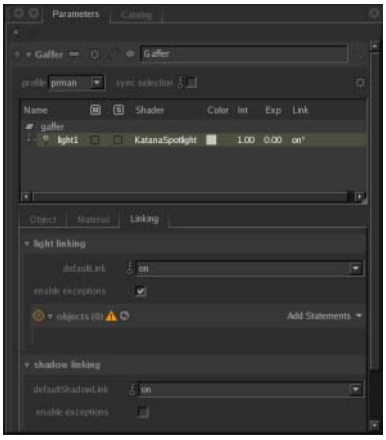
Light linking enables you to create a set of objects which can either be lit (while the others aren't) or unlit (while others are lit).

Setting the default behavior for a light

1. Select the light within the **gaffer** hierarchy inside the **Parameter** tab.
2. Inside the **Linking** sub-tab, select **light linking > defaultLink** :
   - **on**—everything is lit by default, exceptions are unlit.
   - **off**—everything is unlit by default, exceptions are lit.

Creating a light's exception list

1. Toggle the **enable exceptions** checkbox to **on**.
2. Assign the Scene Graph locations of the objects to be excluded to the **light linking > objects** parameter (see Assigning locations to a CEL parameter).



Linking Shadows to Specific Objects

Linking shadows is handled in the same manner as linking lights above. Each location within the scene graph below /root/world has a **lightList** attribute. This is where light linking and shadow linking information is stored.

Note *Currently only the Arnold renderer supports shadow linking within Katana.*

Deleting from the Gaffer Hierarchy

To delete an item from the gaffer hierarchy:
Right-click on the item in the hierarchy and select **Delete** (or press **Delete**).

Locking a Light or Rig's Transform

Once a light is in the correct position, you can lock it to prevent accidental movement. Locking a light does not prevent it from being edited or deleted.

To toggle whether a light is locked:
Right-click on the light and select **Lock**.

Note *This also works for light rigs and aim targets.*

Duplicating an Item within the Gaffer Hierarchy

To duplicate an item within the gaffer hierarchy:
Right-click on the item in the hierarchy and select **Duplicate**.

Creating Shadows

Different renderers support different types of shadows. The most common types of shadows are:

- Raytraced
- Shadow Map
- Deep Shadow Map

As Katana supports any renderer that implements its Renderer API, the shadow types available depends on the renderer.

The renderers that ship with Katana support the following shadow types:

- Arnold: raytracing (default).
- PRMan: raytracing, shadow maps, deep shadow maps.

Raytracing Shadows in Arnold

As long as the light shader supports them, your Arnold renders use raytraced shadows by default.

You can turn off shadows by disabling them within the **Linking** sub-tab of the Gaffer node.

Raytracing Shadows in PRMan

Raytracing shadows within PRMan involves two steps:

- set raytracing in the light,
- define which objects within the scene cast shadows.

Turning on raytracing for PRMan lights

To turn on raytracing:
Type **raytrace** in the material shader's shadow file parameter. For instance, the **KatanaSpotlight** shader uses a **lightShader > Shadows > Shadow_File** parameter.

Specifying which objects cast shadows in PRMan

To specify which objects cast shadows:

1. Create a PrmanObjectSettings node and connect it into the recipe.

2. Select the PrmanObjectSettings node and press **Alt+E.**

   The PrmanObjectSettings node becomes editable within the **Parameters** tab.
3. Assign the Scene Graph locations of the shadow casting objects to the PrmanObjectSettings node's **CEL** parameter (see Assigning locations to a CEL parameter).
4. Select **Yes** in the **attributes > visibility > transmission** dropdown.

Tip *While raytracing shadows using the **KatanaSpotlight**, you can:*

- *change the light radius using **lightShader > Shadows > shadowblur**, or*
- *change the number of shadow rays using **lightShader > Shadows > shadownsamps**.*

Creating a Shadow Map

A shadow map is a depth render from a light. This render is later used by the light shader to work out whether an object is occluded by another object by testing its depth from a light against the depth recorded in the shadow map.

To create a shadow map:

1. Create a ShadowBranch node and connect it into the recipe.
2. Create a RenderOutputDefine node and connect it below the ShadowBranch node.
3. Select the RenderOutputDefine node and press **Alt+E.**

   The RenderOutputDefine node becomes editable within the **Parameters** tab.
4. Select **file** from the **locationType** dropdown.
5. Enter a filename for the shadow map in the **renderLocation** parameter.

   When using the OpenColorIO standard, shadow map files should have an **_ncf** suffix. For more on OpenColorIO, see Managing Color Within Katana.
6. Create a RenderSettings node and connect it below the RenderOutputDefine node.
7. Select the RenderSettings node and press **Alt+E**.

   The RenderSettings node becomes editable within the **Parameters** tab.
8. Enter the scene graph location of the light whose shadow map you are creating in the **cameraName** parameter.
9. Select the shadow map resolution from the **resolution** dropdown or type in the resolution to the right of the dropdown.
10. Create a Render node and connect it below the RenderSettings node.

11. Right-click on the Render node and select **HotRender** to generate the file.

Creating a Deep Shadow Map

A deep shadow map is rendered from a light and stores the transparency levels as you step through the image until fully opaque. With the use of multiple samples, fine geometry and curves can cast realistic shadows (particularly useful for hair and fur). Motion blur can also be included within deep shadow maps.

A shadow map generates more information, and hence larger files, than a normal shadow map.

To create a deep shadow map:

1. Create a ShadowBranch node and connect it into the recipe.
2. Select **primary deepshad** from the **defineOutputs** dropdown.
3. Create a RenderOutputDefine node and connect it below the ShadowBranch node.
4. Select the RenderOutputDefine node and press **Alt+E.**

   The RenderOutputDefine node becomes editable within the **Parameters** tab.
5. Select **file** from the **locationType** dropdown.
6. Enter a filename for the deep shadow map in the **renderLocation** parameter.

   When using the OpenColorIO standard, shadow map files should have an **_ncf** suffix. For more on OpenColorIO, see Managing Color Within Katana.
7. Create a RenderSettings node and connect it below the RenderOutputDefine node.
8. Select the RenderSettings node and press **Alt+E**.

   The RenderSettings node becomes editable within the **Parameters** tab.
9. Enter the Scene Graph location of the light, whose deep shadow map you are creating, in the **cameraName** parameter.
10. Select the deep shadow map resolution from the **resolution** dropdown or type in the resolution to the right of the dropdown.
11. Create a Render node and connect it below the RenderSettings node.
12. Right-click on the Render node and select **HotRender** to generate the file.

Using a Shadow Map In a Light Shader

A shadow map (whether deep or normal), once generated, is just a file. In order to utilize the file, a light shader must know where it is.

Connecting a Gaffer light to a shadow map file:

To connect a shadow map:

1. Select the Gaffer node and press **Alt+E**.
   The Gaffer node becomes editable within the **Parameters** tab.
2. Select the light in the Gaffer's hierarchical view.
   The lights parameters display below the hierarchical view.
3. Click the **Material** sub-tab in the light's parameters.
4. Click  next to the **material** parameter grouping.
   The connected shaders list displays.
5. Click  next to the **lightShader** parameter.
6. Click  next to the **Shadows** parameter grouping.
7. **Shift+middle**-click and drag from the Render node that is generating the shadow file to the **Shadow_File** parameter.
   An expression link is created between the output from the Render node and the **Shadow_File** parameter.

Note *__Shadow_File__ is the parameter used for __KatanaSpotlight__, your shaders may use a different parameter name.*

Tip *While using shadow maps with the __KatanaSpotlight__, you can:*

- *blur the shadow map using __lightShader > Shadows > shadowblur__,*
- *change the number of shadow map samples using __lightShader > Shadows > shadownsamps__, or*
- *move the shadow map (to avoid artifacts) using __lightShader > Shadows > shadowbias.__*

Positioning Lights

To position a light it first needs to be visible within the **Scene Graph** tab (see Changing What is Shown in the Viewer) then positioned within the **Viewer** tab.

Moving a Light Within the Viewer

To move a light, you can:

- Translate and rotate the light with the manipulators, (see Transforming an Object in the Viewer).

OR

- Look through the light and change its view position, (see Changing What You Look Through).

10 Rendering a Scene

Overview

Katana provides two rendering choices: **Interactive Render** and **Hotrender**.

The RenderOutputDefine node sets the type of render (such as color, point cloud, shadow, etc.), the channels (including arbitrary output variables—AOVs), the colorspace, the pass name, and the final render destination.

The RenderSettings node sets the render camera, the choice of renderer, and the resolution.

Renderer specific settings are set using the renderer's global settings node. For instance, when rendering with Arnold you use the ArnoldGlobalSettings node. You can also give locations within the Scene Graph renderer object settings using that renderer's object settings node, using the ArnoldObjectSettings node.

All of the renderer settings manipulate attributes at the **/root** location within the Scene Graph. The **specific** node changes are stored under the **renderSettings** attribute. Inside the **renderSettings** attribute are the output passes from the RenderOutputDefine node. They are stored under **renderSettings.outputs**. Renderer specific globals are stored under the <xxx>GlobalStatements attribute, for instance **arnoldGlobalStatements**.

Performing an Interactive Render

You can perform an interactive render at any node within the recipe. The Scene Graph is generated up to that node. The generated scene data is then sent to the actual production renderer and the results are visible in the **Monitor** tab (see Viewing Your Renders).

To perform an interactive render:

1. **Right**-click on a node.
2. Select **Interactive Render**.

Setting up Interactive Render Filters

Interactive render filters enable you to set up common interactive render recipe changes without having to add them at each point in the recipe to test. These filters are designed to only be included when performing an interactive render and are ignored for hotrenders.

You can set up an interactive render filter to reduce the render image size,

thus making debugging and light tests much quicker. Other examples might be anti-aliasing settings, shading rate changes (if using RenderMan), or the number of light bounces. A filter can consist of more than one change to the recipe and it is the equivalent of appending the filter nodes to the end of the node you selected to render.

These filters are bundled together inside the InteractiveRenderFilters node and toggled using  at the top of the application.

Creating interactive render filters

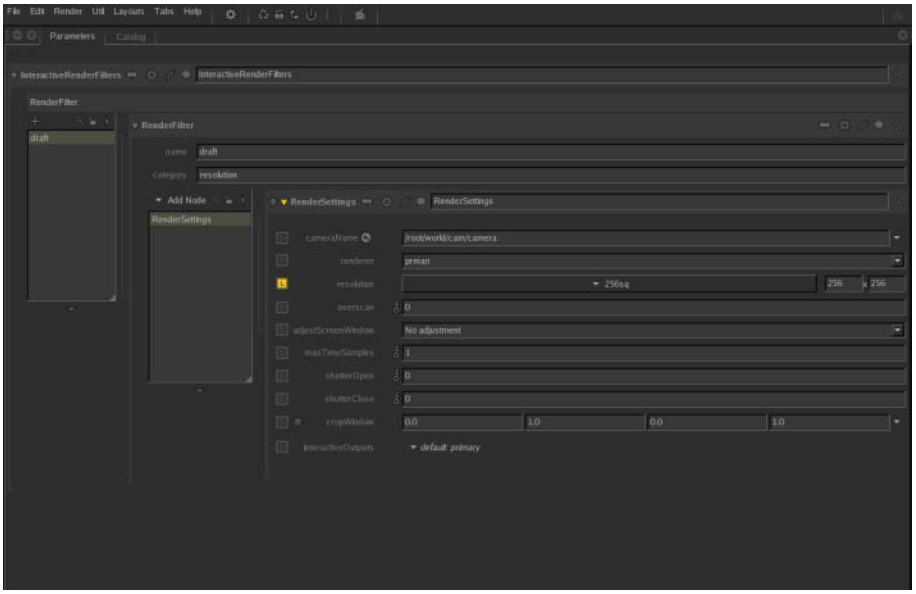
1. Create an InteractiveRenderFilters node and place it anywhere within the **Node Graph**.
2. Select the InteractiveRenderFilters node and press **Alt+E**.

   The InteractiveRenderFilters node becomes editable within the **Parameters** tab.
3. Click  below **RenderFilters** in the **Parameters** tab.

   A new **RenderFilter** is created.
4. To help you remember what this filter does, type a name in the **name** parameter.
5. To create a group of filters, type a group name in the **category** parameter.
6. Click **Add Node** and select a node from the list.

   The list can be filtered by using the **Categories** dropdown or the **Filter** field.

7. Make any changes to the node.



8. Repeat the previous two steps for any additional nodes for this filter.
9. Repeat steps 3 to 8 for any additional filters.

Tip *It is also possible to **middle**-click and drag nodes from the **Node Graph** tab into the **Add Node** list of the InteractiveRenderFilters node.*

Note *The InteractiveRenderFilters node doesn't need to be connected into a recipe to work.*

Activating and deactivating a render filter

By default, render filter nodes aren't active. To toggle whether a render filter is active:

1. Click  at the top of the application.
2. **Middle**-click and drag filters from one side to the other to toggle whether they are active. (You can remove all the active filters by clicking the **clear** button.)

Setting Up a Render Pass

The RenderOutputDefine node is used to define render outputs inside Katana. With it, you can set:

- The type of render output (such as color, point cloud (ptc), etc.).
- The output's file type, colorspace, and location.
- The outputs name.

Defining and Overriding a Color Output

The RenderOutputDefine node can be used to create a new render output or override the settings for an existing one.

To define or override a color output:

1. Create an RenderOutputDefine node and add it to the recipe.

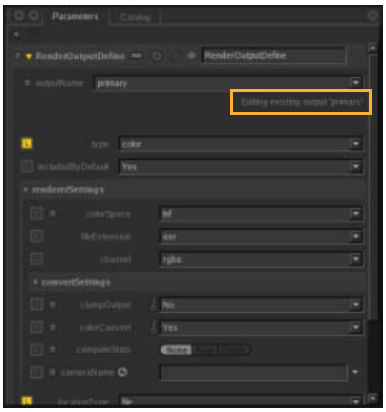
   RenderOutputDefine nodes are usually placed just before Render nodes (see below for more on Render nodes).

2. Select the RenderOutputDefine node and press **Alt+E**.

   The RenderOutputDefine node becomes editable within the **Parameters** tab.

3. Type the pass name to define or override in the **outputName** parameter.

   The **primary** pass is the default pass. Setting the pass name to something other than **primary** results in more than one pass. Katana provides feedback below the **outputName** parameter that displays whether or not you are creating a new pass or editing a previous one.



4. Select the output file's colorspace using the **colorSpace** dropdown.

   The output colorspace is ignored if the **colorConvert** dropdown is set to **No**. For more on colorspaces within Katana see Managing Color Within Katana below.

5. Select the file type to use from the **fileExtension** dropdown.

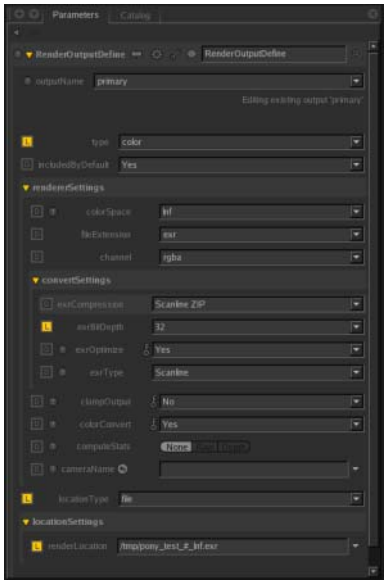
   The file type should have sufficient bit depth for the colorspace selected in 4. For instance, the lnf colorspace requires 32 bits and, as such, some file formats won't support. Use the **convertSettings** parameter grouping to access the file type specific settings, including bit depth.

6. Select the type of location for the output file using the **locationType** dropdown.

   The **locationType** can be:

   • **local**—the output is saved to a temporary directory off /tmp.

- **file**—the **locationSettings** parameter grouping gains a renderLocation parameter so you can select a file location.
- **studio's asset manager**—your studio may have an asset manager which appears here, details are implementation specific.



Defining Outputs Other than Color

The exact options available in the RenderOutputDefine node's type parameter depends on the current renderer. Each renderer plug-in is queried for the list of output types it supports.

The types available for PRMan are:

- **color**—used for most renders.
- **deep**—used for deep shadow map creation, see Creating a Deep Shadow Map.
- **shadow**—used for normal shadow map creation, see Creating a Shadow Map.
- **raw**—used when no color management is needed and allows you to directly set the **Display** line as it is output into the PRMan RIB stream.
- **ptc**—used to define a point cloud output (although the actual point cloud is created by a shader).
- **script**—used to inject a command line script into the render process that depends on a previous render (usually for txmake, ptfilter, or brickmake commands).
- **prescript**—used to inject a command line script into the render process that runs before the render is started.

- **merge**—used to merge a number of render outputs (usually AOVs) into a single OpenEXR file.
- **none**—clears the pass, removing it from the output list.

The types available for Arnold are: **color**, **raw**, **script**, **prescript**, **merge**, and **none**. They behave in the same manner as their PRMan counterparts.

Defining an AOV Output

Arbitrary output variables (AOVs) allow data from a shader to be output during render calculations (usually data that is being calculated as part of the beauty pass and hence at no extra processing cost) to provide additional options during compositing. The ability to define AOVs is fully supported in Katana and is easy to set up.

To define an AOV output:

1. Follow Defining and Overriding a Color Output to set up a normal output.
2. In the **channel** parameter of the RenderOutputDefine node, enter the name of the AOV (this is the actual variable name that is being output from the renderer), such as **_occlusion** or **P**.
3. Create a renderer specific OutputChannelDefine node, for instance PrmanOutputChannelDefine, and add it to the recipe above the RenderOutputDefine node.
4. Select the <Renderer>OutputChannelDefine node and press **Alt+E**.

   The <Renderer>OutputChannelDefine node becomes editable within the **Parameters** tab.
5. Enter the same name as the **channel** parameter in step 2 into the **name** parameter, in the previous examples **_occlusion** or **P**.
6. At this point, the parameters needed by the renderer specific OutputChannelDefine node vary depending on the renderer, see below.

For the PrmanOutputChannelDefine node:
Select the data type of the AOV from the **type** dropdown.

For the ArnoldOutputChannelDefine node:
Make sure the parameters match the data type of the AOV. Consult the Reference Guide that accompanies this User Guide for details on the various parameters.

Previewing Interactive Renders for Outputs Other Than Primary

By default, the output displayed in the **Monitor** tab after an **Interactive Render** is the output from the **primary** pass. When additional outputs are available, such as from AOVs, you can view those in the **Monitor** tab alongside the primary pass.

To view additional interactive render outputs:

1. If there isn't a RenderSettings node below the RenderOutputDefine node then create one and add it.
2. Select the RenderSettings node and press **Alt+E**.

   The RenderSettings node becomes editable within the **Parameters** tab.
3. Select the outputs to view from the **interactiveOutputs** parameter's list.

   All outputs selected are available in the **Monitor** tab the next time you perform an interactive render downstream of this RenderSettings node. For more on viewing these renders, see Selecting Which Output Pass to View.

Executing Your Renders with the Render Node

The Render node acts as a render point within a recipe (possibly one of many).

To write a render pass to disk:

1. Create a Render node and add it to the recipe.

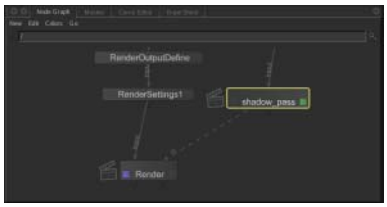
   Add the Render node at the point in the recipe where you are happy with the interactive render.
2. Right-click on the Render node and select **HotRender**.

   The Scene Graph is generated up to that node and then the generated Scene Graph data is sent to the renderer. The outputs of which are saved to the locations specified by the RenderOutputDefine node which generated the output.

Note *Unlike an interactive render (which shows the render as it is generated in the **Monitor** tab), the results of a hotrender are only visible after the render has been completed.*

Setting up Render Dependencies

Some renders may require another render to be completed first, for instance the generation of a shadow map. You can tell Katana that one Render node depends on another by connecting the output from the Render node that needs to be run first to the large connector at the top of the other Render node.



The dependency is shown with a dashed line.

Managing Color Within Katana

As well as communicating with one or more renderers, Katana also reads in image data from a number of different formats. Managing the color of the data within Katana is accomplished through the OpenColorIO standard originally developed by Sony Pictures Imageworks.

A typical workflow within Katana involves:

1. Reading in the images from various formats, such as DPX, TIFF, or OpenEXR.
2. Converting those images into the scene-linear colorspace.

   This is handled automatically by Katana as long as the filenames use the OpenColorIO naming scheme. Files should use a suffix which denotes the files colorspace, for instance: beautypass_lnf.exr (for a 32-bit linear file). For further details, see the OpenColorIO standard at http://opencolorio.org/.
3. Rendering within scene-linear colorspace.
4. Compositing and manipulating the images in scene-linear colorspace.

   Compositing with image data that has not been converted yields inconsistent results.
5. Viewing the scene-linear image data through a device specific look-up-table (LUT) in the **Monitor** tab. The LUTs can include additional manipulations to show the image data converted to film or log (or any other potential output if you have the correct LUT) so you can see the image as it would appear in that target's colorspace.
6. Writing the file out, specifying the colorspace to use in the relevant node. Use the **rendererSettings.colorSpace** parameter in the RenderOutputDefine node for 3D renders and the **image.colorspace**

parameter in the ImageWrite node for 2D composites. Make sure **colorConvert** is enabled in both cases.

This is a best practise guide on how to work within Katana. That said, it is perfectly possible for you to work outside the OpenColorIO standard or even manipulate your images within log or some other colorspace. Doing so forces you to manage all image manipulations manually.

11 Viewing Your Renders

Monitor and Catalog Overview

The **Monitor** tab is a viewer for current and previous renders. The **Catalog** tab acts as an archive for renders and imported images. Within the **Catalog** tab you can manage images by placing them into different **slots** for later comparison or reference.

When a slot is active (its slot number is displayed beneath the **Front** image in the **Monitor** tab), new renders are placed at its top. If the current slot's top image is not locked, it is replaced by any new renders. If the top image is locked, a new render is placed above it in the slot.

Using the Monitor tab

Toggling whether the Monitor tab is maximized

Press **Ctrl+Space** or **double**-click on the tab name.

Switching the Front and Back images

Press **Space**.

Changing which Catalog slot to use

Press the number that corresponds to the slot, for instance **3**,
OR
Change the current **Front** image using the **Catalog** tab, see To change the Front and Back images within the Monitor tab:.

Viewing the Catalog from inside the Monitor tab

Press the **Tab** key (pressing the **Tab** key again returns to the **Monitor** view).

Changing the Image Size and Position

There are numerous ways to get the image to the right size and location within the **Monitor** tab.

Moving the image around the Monitor tab

Middle-click and drag.

Fitting an image to the Monitor tab

At the top of the **Monitor** tab, select **[Current display ratio]** (for instance **1.23 : 1**) **> Frame Display Window** (or press **F**).

Viewing the image at a 1:1 ratio

Select **[Current display scale] > Reset Viewport** (or press **Home**).
The image changes size so the displayed image is one image pixel to one screen pixel, the bottom left of the image moves to the bottom left of the **Monitor** tab.

Changing the size of the image within the Monitor tab

To change the displayed image size:
Scroll the **mouse-wheel** up to zoom in (or press **+**) or scroll the **mouse-wheel** down to zoom out (or press **-**).
The image size changes by a factor of two, for example: 1:8, 1:4, 1:2, 1:1, 2:1, 4:1, 8:1. The change is reflected in the display scale at the top of the tab.
OR
Alt+middle-click and drag (drag right to zoom in, drag left to zoom out).

Tip *Katana zooms in and out around the location of the cursor.*

Changing How To Trigger a Render

By default you have to manually start a render either through right-clicking on a node or by using one of the menu options under **Render**.

Katana can also be made to render when you release the mouse after a change (**Pen Up Render** mode) or as you drag a parameter or the Timeline (**Drag Render** mode).

These render modes only work with 2D renders.

To change when Katana starts a render:

- Select **Render > Manual Render** to only start a render manually.
- Select **Render > Pen-Up Render** to start a render when you release the mouse after changing a parameter or the current time.
- Select **Render > Drag Render** to start a render while you are changing a parameter or the current time.

Tip *These options are also available at the top of the **Monitor** tab.*

Changing The Displayed Channels

To change the displayed channel:

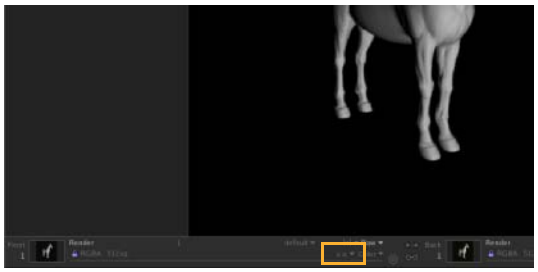
1. Click the channel display dropdown (labelled **Color** by default) towards the bottom of the **Monitor** tab.
2. Select the channel to display:

- **Color** (or press **C**)
- **Luma** (or press **L**)
- **Red** (or press **R**)
- **Green** (or press **G**)
- **Blue** (or press **B**)
- **Alpha** (or press **A**)

Tip *If you are viewing a channel other than the color channel, press the key that corresponds to that channel to toggle back to color. For instance, click **R** once to view the red channel, click **R** again to go back to the color channel.*

Changing How the Alpha Channel is Displayed

The alpha channel menu is located next to the color display menu at the bottom of the **Monitor** tab.



Toggling Premultiply in the Monitor tab

Select **[Alpha display] > Premultiply**.

Displaying the alpha channel as an overlay

It is possible to display the alpha channel as an overlay (either as a mask or a matte).

Using the alpha channel menu the overlay is set to one of three states:

- **Mask**—The area of the image with no alpha channel becomes the overlay color.
- **Matte**—The area of the image with an alpha is overlayed with the overlay color.
- **None**—No alpha overlay is displayed.

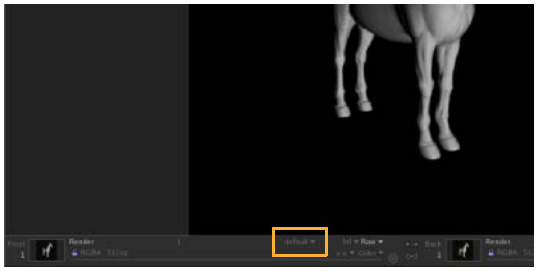
Changing the color used for alpha overlays

1. Select **[Alpha display] > Overlay: Color**.
   The color picker dialog appears.
2. Select a color with the color picker.
3. Press **Ok**.

Selecting Which Output Pass to View

When more than just the primary pass is outputted during an interactive render, you can view all the outputs within the **Monitor** tab.

To view outputs other than the default (primary) pass:
Select the output from the outputs dropdown towards the bottom of the **Monitor** tab. By default it is **default** or **primary** (depending on the render settings).



For details on setting up multiple outputs, see Defining an AOV Output. For more on sending those outputs to the **Monitor** tab, see Previewing Interactive Renders for Outputs Other Than Primary.

Using the Catalog tab

The **Catalog** tab acts as an archive for your renders. It has a number of slots where you can place each render. Each slot acts as a stack, you replace the top render in the stack by starting a new render. If the top item is locked, any new renders become the new head of the stack.

For more on changing which slot to use, see Changing which Catalog slot to use.

Viewing the RenderLog for a Catalog Entry

The **RenderLog** output for renders during this session of Katana are saved as part of its catalog entry. Catalog entries saved with a project do not include their **RenderLog**.

To view a **Catalog** entry's **RenderLog** output:

Click its thumbnail.
The entry becomes the **Front** render in the **Monitor** tab and its **RenderLog** entry is displayed within the **RenderLog** tab.

Removing Renders from the Catalog

To remove all unlocked images from the **Catalog**:
Select **Edit > Flush Unlocked Images** (from within the **Catalog** tab).

To delete the selected images from the **Catalog**:

1. Select the image(s) to delete.
2. Select **Edit > Delete Selected Images** (or press **Delete**).
3. If the images are locked, confirm deletion by clicking **Accept**.

To clear the entire **Catalog**:

1. Select **Edit > Clear Catalog**.
2. Click **Delete** to confirm.

Changing the Catalog Renders Displayed in the Monitor tab

To change the **Front** and **Back** images within the **Monitor** tab:

- left click a thumbnail to make it the **Front** image, or,
- right click a thumbnail to make it the **Back** image.

Manipulating Catalog Entries

Moving catalog entries from one slot to another

To move entries:
Middle-click and drag.

Toggling the lock status of a render

To toggle the lock status:

Click  /  next to the image's thumbnail.
Locked images are not overridden by a subsequent render to the same slot.

Toggling whether an image is saved within this catalog

To toggle the save status:

Click  /  next to the image's thumbnail.
The first time the icon is pressed a file is saved to the directory specified by the KATANA_PERSISTENT_IMAGES_PATH environment variable (if not set, it defaults to /tmp/katana_persist). The naming of the file consists of the show name, shot name, file size, and colorspace. To add a prefix to the

filename, use the KATANA_PERSISTENT_IMAGES_PREFIX environment variable.

Note *The show name comes from the $SHOW environment variable and the shot name comes from the $SHOT environment variable.*

Changing the region of interest (ROI) to match the ROI of the render

1. **Right**-click to the right of the renders thumbnail.
2. Select **Adopt Render ROI**.

Changing the current frame to match the frame of the render

1. **Right**-click to the right of the renders thumbnail.
2. Select **Adopt Frame Time**.

Selecting the node the Catalog render was generated from

1. **Right**-click to the right of the renders thumbnail.
2. Select **Find in Node Graph**.

Regenerating the thumbnail within the Catalog tab

1. **Right**-click to the right of the renders thumbnail.
2. Select **Regenerate Thumbnail**.

Creating a copy of a catalog item

1. **Right**-click to the right of the renders thumbnail.
2. Select **Duplicate Catalog Item**.

Making a comment for a Catalog render

- Type under the comment heading in the same row as the relevant thumbnail.

OR

- If the image is the current **Front** or **Back** image, click  at the top of the **Monitor** tab and type the comment in the **Front** or **Back** field.

Importing an image or file sequence to the Catalog

1. Select **File > Import Image / Sequence** (from the **Catalog** tab).
   The **File Browser** appears.

2. Select whether you want an individual frame or a sequence by toggling **Sequence Listing**.
3. Select the image or sequence to import.
4. Click **Accept**.

Toggling the lock for new 2D renders

Click the checkbox marked **Lock 2D**.
When ticked, any new renders automatically have the lock icon. Being locked prevents the image being overridden by a subsequent render to the same slot.

Changing the Catalog View

By default the **Catalog** tab displays renders under their respective slot. It is also possible to view the renders in order of when they were rendered.

To change the **Catalog** tab to a **Slot** centric view:
Click **Slot View** in the upper right corner of the tab.

To change the **Catalog** tab to a **Time** centric view:
Click **Time View** in the upper right corner of the tab.

Using the Histogram

Katana comes equipped with a **Histogram** tab for checking RGBA levels within an image.

Note *The Histogram tab works in conjunction with the Pixel probe in the Monitor tab. To view anything within the Histogram tab you must have a point or area selected with the probe.*

The image's channels are plotted with the value along the horizontal axis and the count for that value along the vertical axis. The top histogram matches the **Front** image and the bottom histogram matches the **Back** image.

Viewing the count at a particular value

Click anywhere within either histogram.
The RGBA channel's count for that value displays towards the top of each histogram. The format of the display is <value> : <red count> <green count> <blue count> <alpha count> .

Changing the colorspace used for plotting values within the Histogram

Click the **colorspace** dropdown and select one of the provided options — these options come from the current OpenColorIO profile. For more on OpenColorIO, see Managing Color Within Katana.

Changing the scale of the plotted values in the y axis

Enter a new value within the **vScale** field.

Toggling a channels display within the Histogram tab

Click the letter that represents the channel at the top right of the tab.

Tip *Along the bottom of each histogram a colored dot shows the lowest and highest value for each channel displayed.*

Note *The plotted value does not necessarily correspond to an actual value. For instance, the range for the **Inf** colorspace within the **Histogram** tab is 0 to 1023 whereas the actual values are 32-bit floating point. Katana maps the colorspace values to a range for display purposes.*

Viewing the Pixel Values of the Front and Back Images

Turning on the pixel probe

Click  or press . (**full stop**).
The pixel probe toolbar appears.

Changing the colorspace for the displayed pixel values

Select from the top dropdown in the pixel probe toolbar.

Changing what type of pixel value you want displayed

Click the lower dropdown in the pixel probe toolbar and select:

- **ave**—the mean average of the area selected.
- **min**—the lowest value for each channel from the area selected.
- **max**—the highest value for each channel from the area selected.
- **stdDev**—the standard deviation of the area selected.

Changing where the pixel probe samples

- **Ctrl**+click to change the sample centre point.

- Click  to sample an area and  to change back to sampling a point.
- Click and drag the centre point.
- Click and drag the vertical bar to move the sample's centre left and right.
- Click and drag the horizontal bar to move the sample's centre up and down.
- When sampling an area, click and drag the bounding border lines to change the area's bounding rectangle.

Turning off the pixel probe

Click  or press **.** (**full stop**).
The pixel probe toolbar disappears.

Comparing Front and Back Images

If you need to compare the **Front** and **Back** images, for instance: to see how changes are affecting an image or that colors are consistent across shots, you can use the swipe feature within Katana. There are two types of swiping within Katana, line swipe (where a line acts as a curtain from one image to the next) and a rectangle swipe (where a bounding rectangle displays the **Back** image inside the rectangle and **Front** image outside).

Using the swipe line feature

Select **[Swipe menu] > Swipe Line**.
The swipe line handle appears in the Monitor tab and the names for the **Front** and **Back** images become **A** and **B** respectively. You can:

- Click and drag the centre of the handle to move its origin.
- Click and drag the lines either side of the handles centre to change the swipe angle.

Using the swipe rectangle feature

Select **[Swipe menu] > Swipe Rect**
The swipe rectangle appears in the **Monitor** tab and the names for the **Front** and **Back** images become **Outside** and **Inside** respectively. You can:

- Click and drag the centre of the handle to move its origin.
- Click and drag the bounding box lines to change the swipe rectangle.

Turning on a Red/Cyan 3D mix between the Front and Back images

Select **[Swipe menu] > Red/Cyan 3D**.

Turning off swiping (and any swipe menu's Red/Cyan 3D mix)

Select **[Swipe menu] > Swipe Off.**

Toggling 2D Manipulator Display

Some 2D nodes, such as **Transform2D**, provide a manipulator within the **Monitor** tab. It is possible to toggle the display of these manipulators.

To toggle the display of 2D nodes' manipulators:

Click  or  .

Underlaying and Overlaying an Image

The **Monitor** tab within Katana has the ability to overlay or underlay an image with the current render. The underlay and overlay can be composited with either the **Over** or **Add** function.

Displaying the Underlay and Overlay controls

Click  .
The **Underlay/Overlay** toolbar is added to the **Monitor** tab.

Adding an image to the Underlay or Overlay fields

Middle-click and drag from either the **Front/Back** images in the **Monitor** tab or one of the renders from the **Catalog** tab. The checkbox toggles on and the Underlay/Overlay function becomes active.

Turning off the Underlay or Overlay composition

Uncheck the checkbox to the left of the field name.

Removing an image from the Underlay or Overlay fields

Click  to the right of the image.

Changing the compositing function used

1. Click the dropdown on the left of the toolbar.
2. Select the compositing function, the options are **Add** or **Over**.

Removing the Underlay/Overlay toolbar

Click  .

Rendering a Region of Interest (ROI)

To reduce render time while making changes, you can render a smaller section of the image—this section is called a region of interest (ROI). The region of interest is only used for interactive renders and is ignored when doing a hotrender.

Switching on region of interest rendering

Select **[ROI menu] > ROI on** or **ROI on (visible)**.

Switching off region of interest rendering

Select **[ROI menu] > ROI off** or **ROI off (visible)**.

Note *If the region of interest's bounding rectangle is visible, you can change the bounds by dragging the edges.*

12 Using the Viewer

Overview

The **Viewer** tab provides one or more 3D windows into the scene described by the **Scene Graph**. Only locations that are exposed within the **Scene Graph** are represented in the **Viewer**—the exception being pinned locations. For more on pinning a location see Pinning a location or locations.

You can move most objects within the Viewer with **manipulators**. The manipulators available vary depending on the type of object selected. It is also possible for additional manipulators to be implemented by your studio using the Viewer Manipulator API. Consult the developer documentation and example code for further details.

Changing the Layout

The **Viewer** tab can be split into multiple panes allowing multiple views of the same scene.

Splitting the Viewer tab into multiple panes

To split the **Viewer** tab, select **Layout > ...:**

- **Single Pane**
  A single pane takes up the whole **Viewer**, this is the default.
- **Two Panes Side by Side**
  This displays two panes split vertically, sitting side by side.
- **Two Panes Stacked**
  This displays two panes split horizontally, one above the other.
- **Three Panes Split Top**
  This displays three panes, one large on the bottom, and two more split vertically above.
- **Three Panes Split Left**
  This displays three panes, one large on the right, and two more split horizontally on the left.
- **Three Panes Split Bottom**
  This displays three panes, one large on the top, and two more split vertically below.
- **Three Panes Split Right**
  This displays three panes, one large on the left, and two more split horizontally on the right.

- **Four Panes**
  This displays four panes.

You can change each pane to have a different view of the Scene Graph data. The current view is either an object within the scene—such as a camera or light—or a **Viewer** camera. A **Viewer** camera is not a part of the **Scene Graph** and cannot be used outside the **Viewer**. Four **Viewer** cameras are created by default (**persp**, **top**, **front**, and **side**). Others can be created if needed.

Changing How the Scene is Displayed

You can change the 3D scene within the **Viewer** to suit your needs, the computer's specifications, or a scene's specific demands.

Changing the Overall Viewer Behavior

To change the overall shading model, select **Display > ...**:

- **Points**
  This displays the current 3D scene with each vertex (or control point for a NURBS patch) as a point.
- **Wireframe** (or press **4**)
  This displays the current 3D scene with each edge (or surface curve for a NURBS patch) as a line.
- **Simple Shaded** (or press **6**)
  This displays the current 3D scene with a very simple shader which ignores scene lights and shadows.
- **Shaded (raw)**
  This displays the current 3D scene with each object using its viewer shader (or the default if none is assigned). Changing an object or lights viewer shader is done in the same way as assigning any other shader. See Creating a Material.
- **Shaded (filmlook)** (or press **5**)
  This is identical to the **Shaded (raw)** shading model but applies an adjustment designed to approximate the **filmlook** OpenColorIO LUT. For more information on OpenColorIO within Katana see Managing Color Within Katana.

Note *__Shaded (raw)__ and __Shaded (filmlook)__ use an OpenGL shader and not the shader used for the final render. This can cause the __Viewer__ to display a drastically different look to your final render depending on how closely the OpenGL shader matches the production shader.*

Changing which lights to use

To change the lighting used for the Shaded (raw & filmlook) shading models:

- Select **Display > Lighting > Off**.
  Removes all lights from the **Viewer**.
- Select **Display > Lighting > Selected Lights** (or press **8**).
  All selected lights contribute to the lighting in the **Viewer**.
- Select **Display > Lighting > All Lights** (or press **7**).
  All lights within the scene contribute to the lighting in the **Viewer**.

Changing shadow behavior

To change whether shadows are used for the Shaded (raw & filmlook) shading models:

- Select **Display > Shadows > Off**.
  No shadows from lights are used in the Viewer.
- Select **Display > Shadows > Selected Lights**.
  All selected lights create shadows for the lighting in the **Viewer**.
- Select **Display > Shadows > All Lights**.

Changing the anti-aliasing settings

To change the anti-aliasing for lines and points:

- Select **Display > Smoothing > Off**.
  Anti-aliasing is not applied to either points or lines.
- Select **Display > Smoothing > Points**.
  Toggles point anti-aliasing in the **Viewer**.
- Select **Display > Smoothing > Lines**.
  Toggles line anti-aliasing in the **Viewer**.

Changing how proxies are displayed

To change how proxies are displayed:

- Select **Display > Proxies > Bounding Box** (or press **Ctrl+B**).
  Only proxy bounding boxes are displayed.
- Select **Display > Proxies > Geometry** (or press **Ctrl+G**).
  Only proxy geometry is displayed.
- Select **Display > Proxies > Both** (or press **Ctrl+Shift+G**).
  Both proxy geometry and proxy bounding boxes are displayed.

Note *If no proxies have been associated with the geometry, bounding boxes are not automatically calculated.*

Changing the Viewer Behavior for Locations that are Selected

By default the **Viewer** tab highlights (with a white wireframe) the location(s) that are currently selected.

To change the way Katana displays selected locations:
Select **Display > ... while selected > ...**

Note *Any display changes made only affect locations while they are selected.*

Changing the Viewer Behavior While Dragging

For some scenes with complicated geometry or lighting it may make sense to lower the display quality while dragging geometry or lights around the scene.

To change the way Katana displays the scene while dragging:
Select **Display > ... while dragging > ...**

Note *Any settings within this menu override the default display behavior while something within the viewer is being dragged.*

Changing the Background Color

The background color for the pane can be changed to make the scene easier to read, to reduce eye fatigue, or to better match the background color when rendered.

To change the background color, select **Display > Background Color > ...** :

- **Black** (or press **T**)
- **Gray** (or press **Y**)
- **White** (or press **Shift+T**)

Overriding the Display Within a Specific Pane

You can change the shading settings in a specific pane to reduce or improve the quality. This is useful when positioning a light in one pane while viewing the effect in another.

To change a **Viewer** pane's display, use the **Options** menu in the bottom left of the pane. Each menu option corresponds to a similar one under the **Display** menu and acts as an override. To remove any override use **No Change**.

Selecting within the Viewer

You can use standard selection behavior within the **Viewer**.

Action	Behavior
Click	Selects the first object below the mouse.
Drag	Selects all objects within or touched by the marquee.
Shift+Click	Selects an object if it is not selected, deselects it if it is.
Shift+Drag	Selects any object within the marquee that is not selected, deselects it if it is.
Ctrl+Click	Deselects the first object below the mouse.
Ctrl+Drag	Deselects everything within the marquee.

Stepping Through the Selection History

Katana tracks what is selected in the Scene Graph. You can step back and forward through this selection history.

To step backward through the selection history:
Select **Selection > History Backward** (or press **Backspace**).

To step forward through the selection history:
Select **Selection > History Forward** (or press **Shift+Backspace**).

Changing the View Position

You can change which object you are viewing through and that object's position and orientation. This makes light and camera positioning easy.

Viewport Movement

To change the view's current position and orientation:

Shortcut	Action
Alt+left-click and drag	Tumbles the view around its center of interest.
Alt+middle-click and drag	Tracks the view.
Alt+right-click and drag	Dollies the view forward (drag right) and back (drag left).

Note *Looking through a location with no xform attribute does not allow you to move the object within the viewport. To enable transformation of a Scene Graph location, add a **Transform3D** node and assign the location to the node's **path** parameter.*

Changing What You Look Through

The view from a viewport comes from either a light or a camera. You can change the view to a different light or camera to make placement easier or to help with composition.

To change the view with the camera and light list:

1. Click the text at the bottom of the viewport (such as **perspShape**).
   This brings up a list of available lights and cameras.
2. Filter the list to find the camera or light you want. To filter the list you can:
   - Uncheck the **Cameras** checkbox to remove cameras from the list.
   - Uncheck the **Lights** checkbox to remove lights from the list.
   - Type text into the **Filter** field to only display items that contain the text.
3. Select the required light or camera from the list.

OR

1. Click the text at the bottom of the viewport (such as **perspShape**).
   This brings up a list of lights and cameras.
2. Click **New persp view** to look through a new perspective camera.

Note *The camera and lights displayed in the filter list are populated in four ways:*

- *Cameras from the **globals.cameraList** at the **/root/world** location.*
- *Lights from the **lightList** attribute at the **/root/world** location.*
- *The default four cameras (persp, top, front and side) along with any new cameras created with the **New persp view** button in the filter list.*
- *The current render camera (such as set with the **RenderSettings** node).*

To change the view with the  icon:

1. Click  to bring up the camera list.
2. Type text into the **Filter** field to only display cameras that contain the text.
3. Select the camera to look through from the list.

OR

1. Click  to bring up the camera list.
2. Click **New persp view** to look through a new perspective camera.

To change the view with the  icon:

1. Click  to bring up the light list.
2. Type text into the **Filter** field to only display lights that contain the text.
3. Select the light to look through from the list.

To change the view to the currently selected location:

Click  .

Tip *Text entered into the **Filter** field of the view selection dialogs may contain some basic regular expression patterns, such as ranges [a-z].*

Looking Around the Viewport by Offsetting and Overscanning

Looking around the viewport without actually moving the camera is especially useful when a camera has been brought in from another package—representing a camera track for instance—and you don't want to change its position or orientation.

To look around inside the viewport:

1. Click  to bring up the pan/zoom toolbar.
2. To make changes to the current view:
   - Type in the **hOff** field to pan left (negative value) or right (positive value).
   - Type in the **vOff** field to pan up (positive value) or down (negative value).
   - Type in the **overscan** field to zoom in (value between zero and one) or out (value above one).
3. Click **Reset** to restore defaults.

Tip *All three text fields can be scrubbed by dragging on their names.*

While you have the toolbar up the **Pan-zoom active** warning text is displayed in the top left corner of the viewport.

When **hOff**, **vOff**, or **overscan** values change from their defaults, Katana displays a warning icon  on the left of the toolbar.

Changing What is Displayed Within the Viewport

Customizing the **Viewer** or individual viewports to only display the information you need can help speed up your workflow.

Hiding and Unhiding Objects Within the Scene

Objects within the **Viewer** can be hidden from view.

To hide an object(s) within the Viewer:

1. Select the object(s) within the Viewer (or select the locations within the Scene Graph).
2. Select **Selection > Hide** (or press **H**).

Elements are hidden is displayed in all viewports when one or more objects are hidden.

To unhide all hidden objects within the Viewer:

Select **Selection > Unhide All** (or press **U**).

Changing the Subdivision Level of a Subdivision Surface

Subdivision surfaces (Subds) are a form of polymesh that allows greater detail to be defined in certain areas of a mesh while keeping the rest of the mesh at a rough lower level.

To change the displayed level of a subdivision surface:

1. Select the object(s) you want to change.
2. Select **Selection > Subd Level ...** (or press **0**, **1**, **2**, or **3**).

Note *Use higher levels of subdivision with caution as they can be expensive to calculate.*

Toggling Grid Display

Katana displays a **Grid** to help you get a sense of scale, the origin's location, and the orientation of the XZ plane.

To toggle displaying the Grid:

Select **Display > Grid** (or press **G**).

Toggling Manipulator Display

Manipulators provide a visual way for you to change the parameters on an object within the scene—such as a light or piece of geometry. You can hide the **Manipulators** from view.

To toggle displaying the Manipulators:

Select **Display > Manipulators** (or press **Backtick (`)**).

Toggling Annotation Display

Some manipulators have **Annotations** to display a parameter's current value. You can turn these **Annotations** off.

To toggle displaying Annotations for manipulators:

Select **Display > Annotations** (or press **Shift+~**).

Toggling the Heads Up Display (HUD)

Within Katana each **Viewer** pane has its own axis orientation guide in the bottom left corner. The default perspective camera (and any other perspective cameras made with the **New persp view** button) has a manipulator in the top right corner to change the cameras position to a view axis, or three quarter view, centered on the current selection. You can hide these features.

To toggle the display of the Heads Up Display (HUD):

Select **Display > HUD**.

Displaying Normal Information Within the Viewer

Katana gives you the ability to display object normals.

To toggle normal display within the **Viewer** select **Display > Normals** (or press **N**).

To change the normals display length:

- select **Draw Normals > Scale ...** , or
- enter the required normal size in **viewerSettings.normalsDisplayScale** in the **Project Settings** tab.

Freezing the Viewer from Updates

You can freeze the current scene displayed within the **Viewer**. With the scene frozen you can change the **View Node** and change what is exposed within the **Scene Graph** without those changes influencing the **Viewer**.

To stop **Scene Graph** changes from influencing the **Viewer** click  in the top right of the **Viewer** tab, **Scene Graph** tab, or top of the Katana window.

Transforming an Object in the Viewer

You can move, rotate and scale objects within the Viewer in order to get them into the correct position and orientation.

To translate an object in its local coordinate system:

1. Select the object to translate.
2. Select **Manipulators > Translate** (or press **W**).

To translate an object in the world coordinate system:

1. Select the object to translate.
2. Select **Manipulators > Translate (world)** (or press **S**).

To rotate an object in its local coordinate system:

1. Select the object to rotate.
2. Select **Manipulators > Rotate** (or press **E**).

To rotate an object in the world coordinate system:

1. Select the object to rotate.
2. Select **Manipulators > Rotate (world)** (or press **D**).

To scale an object:

1. Select the object to scale.
2. Select **Manipulators > Scale** (or press **R**).

To have no transform manipulator press Q.

Tip *With the Quick Editor (to display the Quick Editor press the **A** key or select the menu option **Layout > Show Quick Editor**) you can change the translate manipulator's coordinate system, its plane axis, and whether it snaps.*

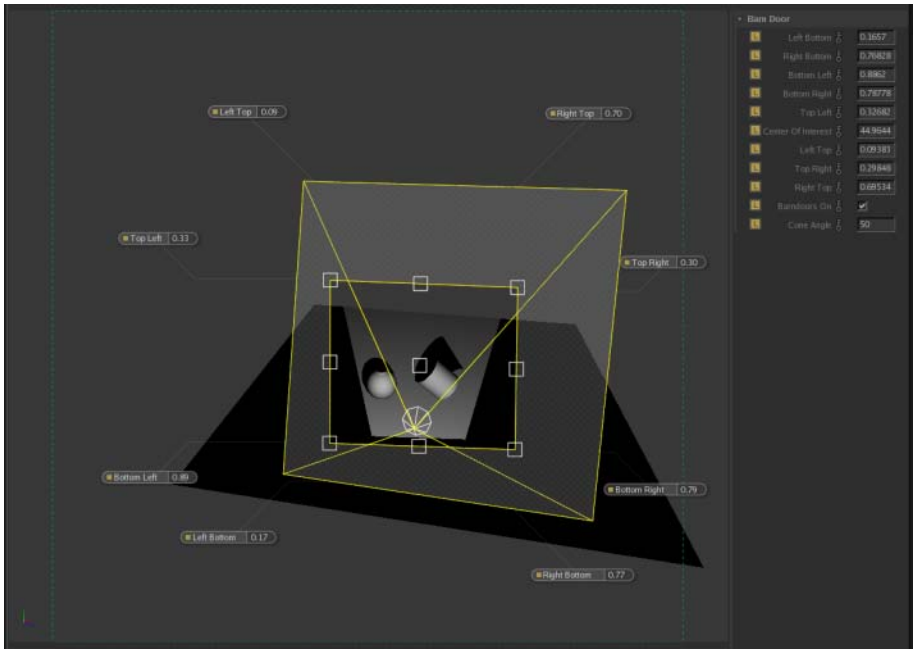
Manipulating a Light Source

Katana provides a visual way to manipulate the parameters of a light. Some parameters which can easily changed with a manipulator are barn doors, the cone angle, decay regions, and its slide map. The light itself must support the function in order to use the manipulator, for instance you cannot use

the barn door manipulator on a light which does not support barn doors — that menu option would not displayed.

To manipulate the barn doors for a light:

1. Select the light to manipulate.
2. Select **Manipulators > Barn Door**.
3. Move one or more of the nine square manipulators to the desired position.



Note *Each parameter is defined by a value between 0 and 1.*

To change a light's center of interest:

1. Select the light to manipulate.
2. Select **Manipulators > Center of Interest**.
3. Move the circular manipulator to where you wish the light to point.

To change a light's cone angle:

1. Select the light to manipulate.
2. Select **Manipulators > Cone Angle**.
3. Move the two manipulators to change the inner and outer cone angles.

To change a light's decay regions:

1. Select the light to manipulate.
2. Select **Manipulators > Decay Regions**.

To change the radius of a light:

1. Select the light to manipulate.
2. Select **Manipulators > Radius**.
3. Move the manipulator away from the center of the light to increase the radius, and towards the center to decrease.

To rotate the light around its center of interest:

1. Select the light to manipulate.
2. Select **Manipulators > Rotate Around COI**.
3. Use the rotate manipulator to move the light around the center of interest.

To move the light while keeping it pointed at its center of interest:

1. Select the light to manipulate.
2. Select **Manipulators > Translate Around COI**.
3. Move the light with the translate manipulator.

   The light remains pointed towards its center of interest.

To position the light so its specular highlight is at a specific point:

1. Select the light to manipulate.
2. Select **Manipulators > Place Specular**.
3. **Click** the place you want the specular highlight to appear.

   Katana moves the light so its position and orientation reflect in the polygon clicked.

Tip *To move forward through the light manipulator list press the **Tab** key, to move backward through the list press **Shift+Tab**. To have no light manipulator press **Shift+Q**.*

13 Look Development with Look Files

Overview

Within the look development department, Katana provides the tools for:

- a material designer—someone who creates a material palette by combining shaders and shader attributes to produce materials that are used downstream in the production.
- a look dev artist—someone who, through the assignment of materials and/or textures, creates an asset's look.

Using Look Files to Create a Material Palette

Look files can be used to create a material palette. This material palette can be brought into other recipes, allowing material presets to be setup and shared across assets, shots, and scenes.

Creating a Material Palette

The **LookFileMaterialsOut** node writes all materials at or below the location /root/materials to a Katana look file. This look file is designed to be a material palette that can then be read in by those in look development to help design an asset's look.

To create a material palette:

1. Create the materials for the material palette. For information on the creation of materials, see Adding and Assigning Materials.
2. Create a **LookFileMaterialsOut** node and connect it to the bottom of the recipe.
3. Select the **LookFileMaterialsOut** node and press **Alt+E**.

   The **LookFileMaterialsOut** node becomes editable within the **Parameters** tab.
4. Enter the location for the Katana look file (.klf) in the **saveTo** parameter.
5. Click **Write Look File**.

   The **Save Materials to Look File** dialog appears.
6. Confirm the location of the Katana look file within the dialog and click **Accept**.

   The look file is saved.

Reading in a Material Palette

The material palette is then easily added to any asset's look development recipe.

To read in a material palette:

1. Create a **LookFileMaterialsIn** node and connect it to the recipe. It is usually added in a seperate branch and joined with a Merge node.
2. Select the **LookFileMaterialsIn** node and press **Alt+E**.

   The **LookFileMaterialsIn** node becomes editable within the **Parameters** tab.
3. Enter the location for the material palette's Katana look file (.klf) in the **lookfile** parameter.
4. Select the pass from the Katana look file to use for this palette with the **passName** parameter.
5. Select whether to bring in the materials palette by reference or not using the **asReference** dropdown.

   When reading the material palette by reference, any materials assigned keep a reference to the Katana look file from which they got their material. Thus, if the material in the materials palette Katana look file is updated, so is the material assigned to the asset. This happens even if the asset's look development is saved in a new Katana look file. If by reference is not used, the asset's look development Katana look file is baked and not updated.
6. Using the **locationForMaterials** dropdown, select where in the scene graph to import the materials from:
   - **Load at original location**—the materials maintain the same location.
   - **Load at specified location**—provides a parameter, **userLocation**, that acts as a namespace for the material palette. For instance, a material at /root/materials/geo/chrome with **userLocation** default_pass is placed at /root/materials/lookfile/default_pass/geo/chrome.

Note *If a location already exists, it is overwritten.*

Using Look Files in an Asset's Look Development

Katana look files (.klf) can be used for an asset's look development. They are created by comparing the scene graph generated at two points within the Node Graph and then recording the difference. When that same asset is used within another recipe, the look file can be applied, restoring the state created during look development. Multiple looks (within the same file) can be created for different passes, the first pass is always called **default**.

Creating a Look File Using LookFileBake

The **LookFileBake** node is used to compare the scene graph generated at two points within the node graph, an original and a second point downstream of the original. At each location below the **LookFileBake** node's **rootLocations** parameter the difference between the original scene graph and the downstream scene graph is recorded.

To create a look file:

1. Create a **LookFileBake** node and place it anywhere within the **Node Graph**.
2. Connect from a point in the recipe where the bake asset has no materials assigned to the **orig** input of the **LookFileBake** node.

   Tip: Connecting it straight after the geometry has been imported usually produces the best results.
3. Connect an output from downstream in the recipe, where the asset has the look you want to bake, to the **default** input of the **LookFileBake** node.
4. Select the **LookFileBake** node and press **Alt+E**.

   The **LookFileBake** node becomes editable within the **Parameters** tab.
5. In **rootLocations**, enter the scene graph location to traverse.

   Tip: It is a good idea to make sure **rootLocations** matches the location the asset was initially imported.

   Multiple locations can be traversed by using **Add Locations** to the right of the **rootLocations** parameter. For more information on adding path locations using location parameters, see Manipulating a Scene Graph location parameter.
6. Enter the asset name for the look file in the **saveTo** parameter.
7. Click the **Write Look File** button.

   The **Write Look File** dialog appears.
8. Select where the asset is going to be saved (it defaults to the **saveTo** parameter) and click **Accept**.

   Katana starts to bake out the look file. This may take some time as all locations in **rootLocations** must be fully expanded for each pass and all their attributes compared. Any differences detected between the scene graph generated at the **orig** input and the scene graph generated at the pass inputs are written to the look file.

To add additional passes to a look file:

1. In the **LookFileBake** node, select **Add > Add Pass Input** to the right of the **passes** parameter grouping.

   A new pass **name** parameter appears.

2. Type the name of the new pass in the **name** parameter.
3. Connect the new input of the **LookFileBake** node (labelled with the pass name) to the output of the node to record the look of.

To have the look file include any changes to /root:

Select **Yes** for the **includeGlobalAttributes** dropdown inside the **options** parameter grouping of the **LookFileBake** node.

To include level of detail changes within the look file:

Select **Yes** for **includeLodInfo** dropdown inside the **options** parameter grouping of the **LookFileBake** node.

Including materials within the look file

Look files automatically include materials that are assigned to geometry below locations it traverses (as are renderer procedurals). On occasion it might be useful to include extra materials created during look development to be read in later using the **LookFileMaterialsIn** or **Material** nodes.

To force materials to be included within the look file:

1. In the **options** parameter grouping of the **LookFileBake** node, select **Yes** for the **alwaysIncludeSelectedMaterialTrees** dropdown.

   A locations widget appears.
2. In **selectedMaterialTreeRootLocations**, enter the material root scene graph location of the materials to include.

   Multiple locations can be included by using **Add Locations** to the right of the **selectedMaterialTreeRootLocations** parameter. For more information on adding path locations using the location widget, see Manipulating a Scene Graph location parameter.

Note *Two things that are not recorded when a look file is written: changes over time (only differences for the current frame are recorded) and deleted locations (locations cannot be removed by look files — for geometry, a similar effect can be achieved by setting its visibility to off).*

Assigning a Look File to an Asset

The easiest way to assign a Katana look file to your asset is by using the **Importomatic**, see Using the Importomatic. It is also possible to assign a Katana look file using **LookFileAssign**.

To assign a Look File using LookFileAssign:

1. Create a **LookFileAssign** node and connect it to the recipe.

2. Select the **LookFileAssign** node and press **Alt+E**.
   The **LookFileAssign** node becomes editable within the **Parameters** tab.
3. Assign the scene graph locations of the 3D assets to the **LookFileAssign CEL** parameter (see Assigning locations to a CEL parameter).
4. In the **asset** parameter, enter the Katana look file to assign.

Resolving Look Files

A look file is assigned to a location in much the same way a material is assigned. An attribute on the location, **lookfile.asset** in this case, stores where to retrieve the look file without actually copying the details to that location. In order to apply the changes specified in the look file for a particular pass, use a **LookFileResolve** node. The alternate, and preferred method, is to use the **LookFileManager** node, see Managing Passes in the LookFileManager.

To resolve the look file for a particular pass:

1. Create a **LookFileResolve** node and connect it to the recipe at the point you want to resolve for a specific pass.
2. Select the **LookFileResolve** node and press **Alt+E**.
   The **LookFileResolve** node becomes editable within the **Parameters** tab.
3. In the **passName** parameter, enter the look file pass to use.

Note *To force a reload for a look file that is being resolved, click **Flush Look File Cache** in the **LookFileResolve**'s parameters.*

Overriding Look File Material Attributes

When a Katana look file is assigned to a location, the details of where to find the look file are stored, not the contents of the look file itself. To retrieve the actual contents, a **LookFileResolve** or **LookFileManager** node is needed. These nodes enable you to select a pass, stored within the Katana look file, and retrieve the scene graph locations for that pass.

While this behaviour has a number of advantages, scene specific overrides need access to the information within the look file. To make scene specific changes you bring in a look file's materials and then change those material locations. This is achieved with either the **LookFileOverrideEnable** or the **LookFileManager** nodes. For details on overriding with the **LookFileManager** node, see Overriding Look Files.

To override a look file material using the LookFileOverrideEnable node:

1. Create a **LookFileOverrideEnable** node and connect it to the recipe.

   The **LookFileOverrideEnable** node should be connected at some point downstream of a **LookFileAssign** node but before the look file is resolved.
2. Select the **LookFileOverrideEnable** node and press **Alt+E**.

   The **LookFileOverrideEnable** node becomes editable within the **Parameters** tab.
3. Enter the name of the look file to override in the **lookfile** parameter.
4. Enter the look file's pass name to use in the **passName** parameter.

   The materials within the look file are brought into the recipe and can be overridden.
5. Edit the material as needed. See Editing a Material for further details.

Activating Look File Lights and Constraints

Katana maintains a list of lights, cameras, and constraints at /root/world within the scene graph. When a look file brings in a light or constraint, the lists at /root/world need to be updated. The **LookFileLightAndConstraintActivator** node activates look file lights and constraints by updating the respective lists.

To activate lights and constraints from within a look file:

1. Create a **LookFileLightAndConstraintActivator** node and connect it to the recipe at some point downstream of a **LookFileResolve** or **LookFileManager** node.
2. Select the **LookFileLightAndConstraintActivator** node and press **Alt+E**.

   The **LookFileLightAndConstraintActivator** node becomes editable within the **Parameters** tab.
3. Find the lights or constraints to activate by either:
   - selecting **Action > Search Entire Incoming Scene...**,

     OR
   - selecting a location within the scene graph and then selecting **Action > Search Incoming Scene From Scene graph Selection...** .

   Any look files with lights or constraints, found during the search, populate the node's hierarchical display (located below the **Action** menu in the **Parameter** tab).

4. Enable the lights and constraints for activation by right-clicking the .klf file in the hierarchical display and selecting **Enable** (or expanding the hierarchy and doing it individually).

Using Look Files as Default Settings

It is often desirable to have consistent default render settings across an entire show. Most render settings reside in the scene graph at /root. These settings can be stored in a Katana look file and brought in to each recipe of a show.

Creating a look file for a show's default settings is the same as creating any other look file but you need to have the look file record changes at /root, which is not recorded by default.

To save changes to /root as part of a look file:

With the **LookFileBake** node's parameters in the **Parameter**'s tab, open up the **options** parameter grouping and select **Yes** for the **includeGlobalAttributes** dropdown.
The look file now records changes to /root.

Setting a globals look file for a recipe

Look files for assets are assigned to the location of the asset. As a look file for a show's settings is designed to repeat the changes made to /root, a **LookFileGlobalsAssign** node associates a look file with the /root location (this can also be achieved with the **LookFileManager** node, see Assigning a Global Look File in the LookFileManager).

To have a look file associated with /root:

1. Create a **LookFileGlobalsAssign** node and connect it to the recipe at the point you want to setup the show's default settings.
2. Select the **LookFileGlobalsAssign** node and press **Alt+E**.
   The **LookFileGlobalsAssign** node becomes editable within the **Parameters** tab.
3. Enter the look file to use in the **asset** parameter.
4. If you want the look file to be resolved immediately, select **Yes** from the **resolveImmediately** dropdown.

Tip *You can force a reload of the look file at anytime by either: clicking the **Flush Look File Cache** button in the **Parameter** tab (when the **LookFileGlobalsAssign** node is editable), or by clicking*  *at the top of the Katana window.*

Making Look Files Easier with the LookFileManager

The **LookFileManager** node has a lot of the functionality mentioned above, but it does it all in one node!

The **LookFileManager** node can:

- Assign a look file to /root, thus providing a show's default settings, in the same way as the **LookFileGlobalsAssign** node.
- Bring in a look file's material locations enabling them to be overridden, in the same way as the **LookFileOverrideEnable** node.
- Define which passes to resolve, in the same was as the **LookFileResolve** node. The **LookFileManager** node can resolve multiple passes, providing an output for each.

Connecting the LookFileManager node into the recipe:

Create a **LookFileManager** node and connect it to the recipe at the point you want to resolve any look files into their respective passes.

Bringing a Look File into the Scene Graph

You can bring in a look file into the scene graph for later overriding or assigning to /root (to set a shot's global settings). This is done by adding the look file to the **Look Files** list of the **LookFileManager** node.

You can add a look file to the **Products** list in a number of ways:

To add the look file currently assigned to a scene graph location:

With one or more Scene Graph locations selected, right-click inside the **Look Files** list (or click ) and select **Add Look File Asset From Scene graph Selection**.

To add a look file from all the look files in the current scene graph:

1. Right-click inside the **Look Files** list (or click ) and select **Find All Look File Assets In Incoming Scene...** .

   The **Find Look File Assets On Incoming Scene** dialog appears. The dialog is populated with all the look files in the current scene graph.
2. Right-click on a look file you want to add and select **Add Look File Asset**.

   You can repeat this step for as many look files as you want to add.
3. Click **Close** when you have finished.

To add a look file from a list of look files at or below a scene graph location:

1. With one or more scene graph locations selected, right-click inside the **Look Files** list (or click ) and select **Find All Look File Assets Beneath Selection In Incoming Scene...** .

   The **Find Look File Assets On Incoming Scene** dialog appears. The dialog is populated with all the look files assigned at or below the scene graph location selected.
2. Right-click on a look file you want to add and select **Add Look File Asset**.

   You can repeat this step for as many look files as you want to add.
3. Click **Close** when you have finished.

To add a look file that is not assigned anywhere within the scene graph:

1. Right-click in the **Look Files** list (or click ) and select **Advanced > Add Look File Asset From Browser...** .
2. Select the look file within the browser and click **Accept**.

   The look file is added to the **LookFileManager Products** list and assigned as the look file to /root. To unassign it, uncheck the **Add As Look File Root Asset** checkbox.

Assigning a Global Look File in the LookFileManager

You can replicate the behaviour of the **LookFileGlobalsAssign** node inside the **LookFileManager**.

To assign a look file to /root that is not currently in the scene graph:

1. With the **LookFileManager** node's parameters in the **Parameters** tab, right-click in the **Look Files** list (or click ) and select **Advanced > Add Look File Asset From Browser...** .

   The **Load Look File** dialog appears.
2. Select the look file within the browser and click **Accept**.

   The look file is added to the **LookFileManager Products** list and assigned as the look file to /root.

To assign a look file that is currently within the scene to /root:

1. Bring the look file into the **LookFileManager** node's **Look Files** list.

2. Right-click on the look file (or select it and click ) and select **Use Look File For Scene Globals**.

Unassigning a Global Look File in the LookFileManager

It is possible to unassign a look file previously assigned to /root within the **LookFileManager** node without deleting it.

To unassign a look file from /root:

Within the **Look Files** list, right-click on the look file (or select it and click ) and select **Disable Use of Look File For Scene Root Attribute**.

Removing a Look File from the Products List

You can remove a look file from the **Look Files** list of the **LookFileManager** node. Removing a look file that has previously been assigned to the /root scene graph location unassigns it. Also, any look file that is removed from the **Look Files** list is no longer available for material overrides within the scene graph.

To remove a look file from the LookFileManager's Look Files list:

Within the **Look Files** list, right-click on the look file (or select it and click ) and select **Remove Look File From Manager**.

Managing Passes in the LookFileManager

Each look file has one or more passes. The **LookFileManager** can resolve as many of these passes as needed, creating an output for each (the **default** pass is always resolved). One technique is to have the look file that is assigned to /root contain all the necessary passes for that shot. This method means only one look file needs to be brought into the **LookFileManager** node to define all the passes that need resolving.

The **Passes** list to the right of the **Look Files** list inside the **LookFileManager** shows a list of passes that are both being resolved and are available within a look file to be resolved. Each pass name has one of three states:

-  —this pass is not only being resolved, the **LookFileManager** is the view node and the **Scene Graph** shows the results of resolving for this pass.
-  —this pass is being resolved, it has an output from the **LookFileManager**.

- no icon—this pass is within the currently selected look file but is not being resolved.

To have the LookFileManager resolve additional passes:

1. Within the **Products** list for the **LookFileManager** node, click on the look file with additional passes. The **Passes** list to the right of the **Products** list shows additional unresolved passes that are contained within the look file. These additional passes are displayed with no accompanying icon.
2. In the **Passes** list, right-click on the pass to resolve and select **Add Selected Pass Name Output.**

   The pass is now resolved and an output is added to the **LookFileManager** node.

To change which pass to use when the LookFileManager is the current View node:

- right-click on the pass in the **Passes** list and select **View Scene graph For Pass**, or
- select the pass in the **Passes** list and select  > **View Scene graph For Pass**, or
- click  next to the pass name.

Overriding Look Files

When a look file is added to the **Products** list, its materials are added to the scene graph under the location /root/materials/lookfile. You can then override/edit these materials.

To override/edit a material within a look file:

1. Add the look file to the **Products** list.
2. In the **Parameters** tab, select **Add Override > Material**.

   You can narrow the list of nodes in the **Add Override** menu using the **Filter** field.

   To have the new **Material** node override affect all passes, toggle the **New Overrides Active For All Passes** to on.
3. Follow the steps for overriding and editing a material at Editing a Material.

Note *It is also possible to **Shift+middle**-click and drag a node into the overrides list from within the Node Graph tab.*

To toggle ignoring an override once it has been created:

In the overrides list, right-click on the override (or select it and click ) and select **Toggle Ignore State**.

To duplicate an existing override:

In the overrides list, right-click on the override (or select it and click ) and select **Duplicate Override**.

To view the parameters for an override in a separate panel:

In the overrides list, right-click on the override (or select it and click ) and select **Tearoff Parameters of Override...** .

To delete an override:

In the overrides list, right-click on the override (or select it and click ) and select **Delete Override** (or with it selected, press **Delete**).

Note *You can change which passes the overrides are valid for using the **active for passes** menu to the right of **Add Override**.*

Tip *Although the most common use of the **Add Override** menu is for adding material overrides, any kind of override may be created so long as the node has both an input and an output.*

14 MANIPULATING ATTRIBUTES

Overview

At its core, Katana is a way to create and manipulate attributes. These attributes, stored at locations within the Scene Graph, represent the information a renderer needs to render a scene.

Although almost all nodes in essence manipulate attributes, Katana provides a number of nodes that give you free reign to directly influence the attributes at one or more location. Two of the most common are AttributeSet and AttributeScript.

- The AttributeSet node is used to create, override, or delete attributes at one or more locations.
- The AttributeScript node is used to run a Python script at one or more locations. The script can access the attributes of the location (and others) and use Python to make changes.

Making Changes with the AttributeSet Node

To add an AttributeSet node to a recipe:

1. Create an AttributeSet node and connect it to the recipe at the point you want to make the change.
2. Select the AttributeSet node and press **Alt+E**.

   The AttributeSet node becomes editable within the **Parameters** tab.
3. Select the assignment mode from the **mode** dropdown:
   - **paths**—the locations influenced by this node are selectable by their path.
   - **CEL**—the locations influenced by this node are selectable using CEL.
4. Assign the locations to influence with this node to either the **paths** or **celSelection** parameter (depending on your selection in 3).
5. Select what type of action this node is performing:
   - **Create/Override**—adds a new attribute or overrides an existing one.
   - **Delete**—if it exists, removes an attribute from the location.
   - **Force Default**—forces the attribute back to its default.
6. Enter the name of the attribute to influence in **attributeName**.

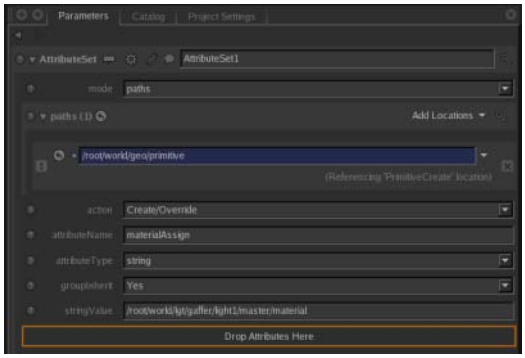
   You can enter a grouped attribute by seperating the parts of the attribute with a full stop, for instance **geometry.point.P**.

If the **action** parameter is **Create/Override**:

7. Select the type of the attribute using the **attributeType** dropdown.

8. With the **groupInherit** parameter, select whether you want the attribute changes to be inherited by any Scene Graph children. For instance, a new attribute on /root/world/geo created with this option set to **Yes** is inherited by all children of /root/world/geo.
9. Enter the new attribute value in the **<type>Value** parameter, for instance **stringValue** for a string.

Tip *It is possible to **middle**-click and drag from an attribute in the **Attributes** tab to the **Drop Attributes Here** hotspot to set the attribute details in the AttributeSet node.*



Using Python within an AttributeScript Node

The AttributeScript node allows you to use Python to influence the attributes on locations in the Scene Graph. Although a full explanation of Python is beyond the scope of this User Guide (it is a fully fledged scripting language in its own right), example scripts are included here to help get you started.

To add an AttributeScript node to the recipe:

1. Create an AttributeScript node and connect it to the recipe at the point you want to insert the Python script.
2. Select the AttributeScript node and press **Alt+E**.

   The AttributeScript node becomes editable within the **Parameters** tab.
3. Assign the Scene Graph locations this Python script is to run on to the **CEL** parameter (see Assigning locations to a CEL parameter).
4. Select when to run the script using the **applyWhen** dropdown:
   - **immediate**—run the script immediately.
   - **during attribute modifier plug-ins resolve**—run the script when other attribute modifier plug-ins are being resolved.

- **during katana look file resolve**—run the script when any katana look files are resolved (or would be resolved if there are none in the recipe).
- **during material resolve**—run the script when material resolving occurs.

5. If you want to run an initial script before the main script, select **Yes** in the **initializationScript** parameter.

   If **Yes** is selected, a **setup** parameter appears. Enter your initialization script in the **setup** parameter.
6. Finally, enter the Python script in the **script** parameter.

Note *Initialization scripts only get run once. Once the initialization script is run, the main script is run for every location assigned to the AttributeScript's **CEL** parameter.*

Example Python Scripts

These example scripts assume a basic knowledge of Python and cannot be used to learn the language in isolation.

Sometimes you may get an error if you copy and paste statements from another source, like an e-mail, into the Python tab or a parameter. This may be caused by the mark-up or encoding of the source you copied the statement from. To fix the problem, re-enter the statement or correct the indentation manually.

Texture path manipulation

If the filename for a texture is included when publishing an asset, the path of where to retrieve that texture needs to be prefixed. If the attribute that contains the filename is geometry.arbitrary.ColMap and the path where textures are stored is on the user.textureRoot variable, we can write a script to append the two and store the output on the textures.ColMap attribute (which is automatically assigned to a parameter of the same name on any shaders, see Assigning Textures).

1. Create a PrimitiveCreate node and add it anywhere in the recipe.
2. Create an AttributeSet node and connect it below the PrimitiveCreate node.
3. Select the AttributeSet node and press **Alt+E**.

   The AttributeSet node becomes editable within the **Parameters** tab.
4. **Shift**+**middle**-click and drag from the PrimitiveCreate node to the **paths** parameter.

   The location created by the PrimitiveCreate node is assigned to the **paths** parameter using an expression.
5. In the **attributeName** parameter, enter **geometry.arbitrary.ColMap**.

6. Select **string** from the **attributeType** dropdown.

   The **stringValue** parameter appears.

7. Enter a filename in the **stringValue** parameter, for instance **test_file.tx**.

Note *The PrimitiveCreate and AttributeSet nodes are only included to provide sample data. In a normal recipe, the data comes from a published asset which is brought into the Scene Graph using the Alembic_In node (or whatever your studio uses to read in its data). This data would already have the geometry.arbitrary.ColMap attribute assigned as part of the asset generation and publication process.*

8. Create an AttributeScript node and connect is below the AttributeSet node.
9. Select the AttributeScript node and press **Alt+E**.

   The AttributeScript node becomes editable within the **Parameters** tab.

10. **Shift+middle**-click and drag from the PrimitiveCreate node to the **CEL** parameter.

    The location created by the PrimitiveCreate node is assigned to the **CEL** parameter.

11. Create a new user parameter called **texturePath**. For details on creating user parameters, see [INSERT SOME REF].
12. In the **script** parameter, enter:

```
colMap = GetAttr("geometry.arbitrary.ColMap")

if colMap is not None:
   textureName = user.texturePath[0] + colMap[0]
   SetAttr("textures.ColMap", [textureName])
```

The GetAttr(<name>) function returns the value assigned to the attribute <name>. In this example, the GetAttr() function is returning the value we assigned using the AttributeSet node previously in the script. The value returned is always a list (as all attributes in Katana are lists of one particular type, a string in this case).

It is also possible to retrieve attributes at locations in the Scene Graph (for instance /root) and not just the one the script is being run on. An example might be: GetAttr("renderSettings.cameraName", atLocation="/root").

Note *It is not possible to return the default value for an attribute using the GetAttr() function. Therefore, if the value has not been previously set within the script, the GetAttr() function returns None.*

Also, GetAttr() doesn't return inherited values by default, if attribute assigned to /root/world/geo that is inherited by /root/world/geo/robot is not returned by GetAttr() unless you include inherit=True.

The script starts by assigning the attribute at geometry.arbitrary.ColMap to the variable colMap using the AttributeScript specific function GetAttr(). If colMap has a value, create a new variable (called textureName) from the concatenation of the user.texturePath parameter and the colMap variable (both of which are lists that need to be referenced by their first element). Finally, set the attribute textures.ColMap with the variable textureName (once again, set as the first element of a list).

The textures.ColMap attribute automatically populates any shaders with a parameter name ColMap. For more on textures, see Assigning Textures.

A new Katana primitive

Everything in Katana is just a collection of attributes attached to locations, even the geometry. In this example script we show how you can create a new Katana primitive using three nodes and a small piece of Python script.

1. Create a LocationCreate node and add it to a new recipe.
2. Select the LocationCreate node and press **Alt+E**.

   The LocationCreate node becomes editable within the **Parameters** tab.
3. In the **locations** parameter, enter /root/world/geo/pyramid.
4. Create an AttributeScript node and connect it below the LocationCreate node.
5. Select the AttributeScript node and press **Alt+E**.

   The AttributeScript node becomes editable within the **Parameters** tab.
6. Add the path /root/world/geo/pyramid to the **CEL** parameter of the AttributeScript node.
7. In the **script** parameter, enter the following:

```
bounds = [-0.5, 0.0, -0.5, 0.5, 1.0, 0.5]
points = [-0.5, 0.0, -0.5,
          -0.5, 0.0, 0.5,
          0.5, 0.0, 0.5,
          0.5, 0.0, -0.5,
          0.0, 1.0, 0.0]
vertexList = [0, 1, 2, 3,
              1, 0, 4,
              2, 1, 4,
              3, 2, 4,
              0, 3, 4]
startIndex = [0, 4, 7, 10, 13, 16]
```

```
SetAttr("type", ["polymesh"])
SetAttr("bound", ScenegraphAttr.Attr("DoubleAttr", bounds))
SetAttr("geometry.point.P",
        ScenegraphAttr.Attr("FloatAttr", points, 3))
SetAttr("geometry.poly.vertexList", vertexList)
SetAttr("geometry.poly.startIndex", startIndex)
```

At this point, you can view the newly created primitive in the **Viewer** tab (although it can't be moved). To enable the pyramid to be moved, it needs a transformation matrix assigned and a node where changes can be stored. This is handled by the Transform3D node.

8. Create a Transform3D node and connect it to the recipe below the AttributeScript node.
9. Select the Transform3D node and press **Alt+E**.

   The Transform3D node becomes editable within the **Parameters** tab.
10. Enter /root/world/geo/pyramid in the **path** parameter to associate this Transform3D node with the pyramid's Scene Graph location.
11. Select **Yes** from the **makeInteractive** parameter dropdown.

The script makes use of a Python function SetAttr(). This function is only available inside the AttributeScript node (and not the **Python** tab). It sets the value for an attribute at the current location. The data for Scene Graph attributes are stored using the ScenegraphAttr.Attr object type.

A more complicated example

The next example is designed to:

- Demonstrate how to pass information from the initialization script to the main script.
- Show how to build a group of attributes.
- Give a brief example of how to format arbitrary data so that it can be interpreted by renderers.

1. Create an AttributeScript node and connect it into a recipe after geometry creation.
2. Select the AttributeScript node and press **Alt+E**.

   The AttributeScript node becomes editable within the **Parameters** tab.
3. Add one or more locations to the **CEL** parameter of the AttributeScript node.
4. Select **Yes** from the **initializationScript** dropdown.

   The **setup** parameter appears.
5. In the **setup** parameter, enter:

```
import GeoAPI

gb = GeoAPI.Util.GroupBuilder()

scope = ScenegraphAttr.Attr("StringAttr", ["primitive"])
value = ScenegraphAttr.Attr("FloatAttr", [1.0, 0.0, 0.0])
color = ScenegraphAttr.Attr("StringAttr", ["color3"])

gb.set("scope", scope)
gb.set("value", value)
gb.set("inputType", color)

user.geom = gb.build()
```

6. In the script parameter, enter:

```
SetAttr("geometry.arbitrary.myVariable", user.geom)
```

Any variables that need to be passed from the initialization script to the main script should be prefixed with user (for instance, user.geom in the example above).

The GroupBuilder object is used to assemble a group attribute with other attributes below. As it is possible to use SetAttr() with a full explicit hierarchy (without having to first create the sub groups) the GroupBuilder is not usually necessary.

Tip *The initialization script is best used to generate shared data when that shared data is expensive to generate. This example, although valid, would probably be done using the SetAttr() function in the main script.*

Arbitrary Attributes Within Katana

Katana provides the ability to export arbitrary attributes to renderers. Currently, only attributes defined in **geometry.arbitrary** is written out. This information takes on a renderer specific form at render time, such as **user data** for Arnold and **primvars** for PRMan.

For each arbitrary attribute to export there are a number of attributes that can be set:

- **scope**—this defines the scope of the attribute. For instance: per object, per face, per vertex etc. Katana uses its own internal naming for the scope (**primitive**, **face**, **point**, and **vertex**) and then converts them to renderer specific interpretations. For instance, when using PRMan:
  - **primitive** is interpreted as **constant**,

  - **face** is interpreted as **uniform**,
  - **point** is interpreted as **varying**, and
  - **vertex** is interpreted as **face varying**.
- **value**—the actual value for the arbitrary attribute.
- **elementSize**—defines a two-dimensional array by setting the size of each element. For instance, for a list of 10 RGB values (with 30 floats), an **elementSize** of 3 is used.
- **inputType**—specifies the attribute's data type, such as a color, vector, normal etc.
- **outputType**—converts the input data type into a different type for output.
- **indexedValue**—in conjunction with **index,** defines an indexed array.
- **index**—the indices of the array of values stored in **indexedArray**.

Some example arbitrary attributes

This is the simplest form an arbitrary attribute can have, a scope and a value.

```
SetAttr("geometry.arbitrary.myFloat.scope", ["primitive"])
SetAttr("geometry.arbitrary.myFloat.value", [1.0])
```

Here, a single float is converted to a color3 data type.

```
SetAttr("geometry.arbitrary.myFloatToColor3.scope",
        ["primitive"])
SetAttr("geometry.arbitrary.myFloatToColor3.value", [0.1])
SetAttr("geometry.arbitrary.myFloatToColor3.outputType",
        ["color3"])
```

In this example, two colors are defined and then converted to uniform color[2] when rendering with PRMan.

```
SetAttr("geometry.arbitrary.myColorArray.scope",
        ["primitive"])
SetAttr("geometry.arbitrary.myColorArray.value",
        [0.1, 0.2, 0.3, 0.4, 0.5, 0.6])
SetAttr("geometry.arbitrary.myColorArray.inputType",
        ["color3"])
SetAttr("geometry.arbitrary.myColorArray.elementSize", [2])
```

A simple example with each face of a cube assigned an alternating texture using an indexed array.

```
SetAttr("geometry.arbitrary.myIndexedArray.scope",
        ["face"])
SetAttr("geometry.arbitrary.myIndexedArray.indexedValue",
        ["textureA.tx", "textureB.tx"])
SetAttr("geometry.arbitrary.myIndexedArray.index",
```

```
                    [0, 1, 0, 1, 0, 1])
```

Beyond the AttributeSet and AttributeScript Nodes

Using the AttributeScript node does have its limitations, most notably speed. Katana provides a C++ API that allows you to manipulate attributes using a plug-in. The API is called the Attribute Modifier Plug-in (AMP). The API documentation can be found through the **Help > API Reference > Plugin API** menu.

An example AMP implementation is included with Katana that enables you to define attributes in XML and assign them to Scene Graph locations. The example node is AttributeFile_In and the technical documentation that accompanies the node can be found in the **Help > Documentation** menu.

15 Animating Within Katana

Animation Introduction

Computer based animation owes its core concepts to the techniques employed by pencil-drawn animators since the dawn of the animation business. In order to reduce time, the lead animators of large studios would draw key poses—known as **keyframes** or **keys**—defining the extreme positions within a scene. A different animator would then fill in the poses between the keyframes using a technique called tweening, thereby creating the illusion of movement. For some scenes, **breakdowns** were created to show how the transition from one keyframe flowed to the next.

Nearly one hundred years later, this technique—known as **keyframing**—is still alive and kicking within Katana.

Katana does the animation heavy lifting by interpolating the values between keyframes. You can tell Katana how you want these **in-between** frames to be generated by specifying a **segment function**.

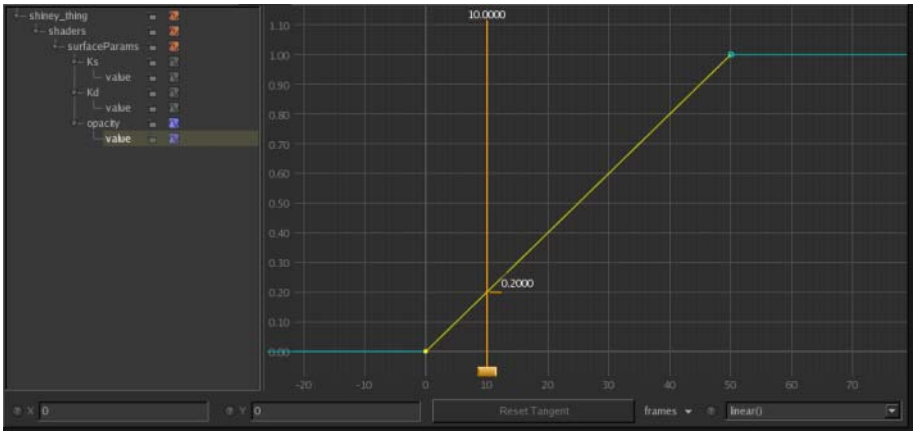


Figure 15.1: Two keyframes on frames zero and fifty with a linear segment function applied to the first.

The most versatile segment function is the **bezier** curve; it uses a mathematical formula to calculate a curve between two anchor points. Bezier curves use four points to interpolate a curve: two anchor points (these are the keyframes) and two control points.

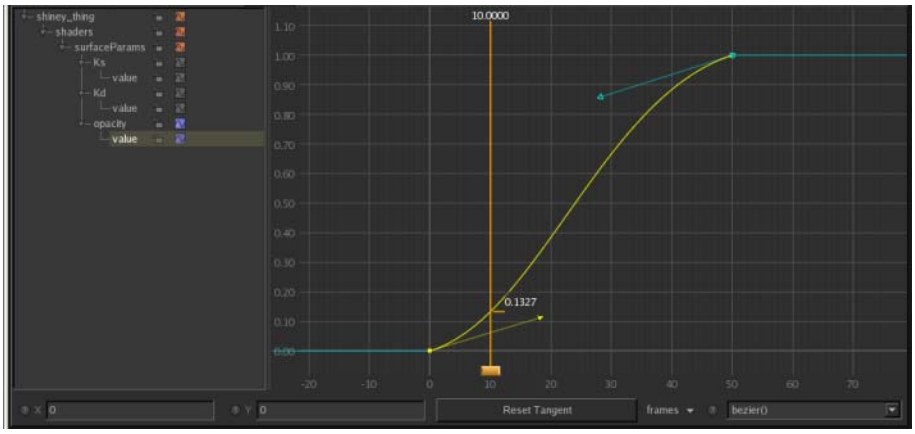


Figure 15.2: The same two keyframes with the bezier segment function applied. The arrowheads represent the location of the two control points.

A tangent and its control points control the slope of the curve around the tangent's keyframe.

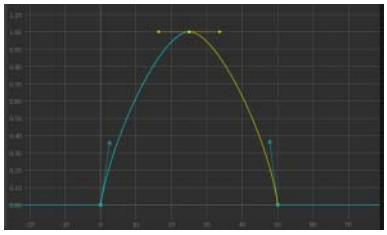


Figure 15.3: The selected control point handles, shown in yellow, form a tangent around the keyframe.

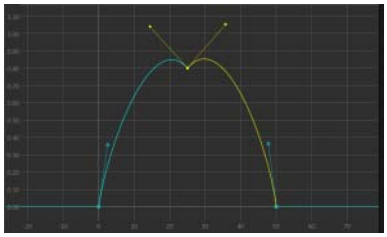


Figure 15.4: Here, a straight line between control points would not pass through the keyframe; hence the tangent is broken.

Breakdowns within the **Curve Editor** maintain the relative time between the keyframe before and the keyframe after.

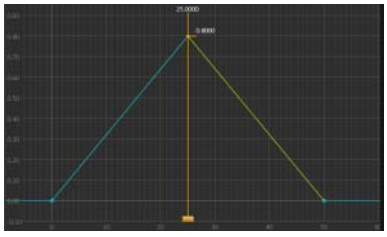


Figure 15.5: Keyframes have been placed on frames 0, 25, and 50. The middle keyframe, on frame 25, has been converted into a breakdown.

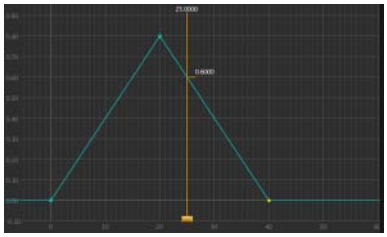


Figure 15.6: Moving the third keyframe from frame 50 to frame 40, automatically moves the breakdown to frame 20.

Keyframes, breakdowns, segment functions, and tangents all combine to create a **curve** that represents how a value changes over time. A curve is plotted on a graph within the Curve Editor tab with time (in frames) along the x axis and the parameter's value plotted on the y axis. When a parameter uses a curve, its background color within the **Parameter** tab changes to green. Light green signifies that the parameter has a keyframe at the current frame; a dark green parameter signifies that the value is interpolated.

Figure 15.7: A bright green parameter signifies a keyframe on the current frame.

Figure 15.8: A dark green parameter signifies the value for the current frame is interpolated.

Setting Keys

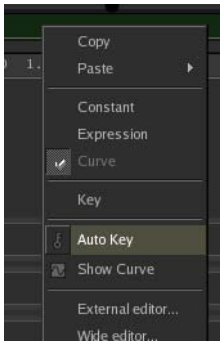
You can set keys either manually or Katana can automatically set a key every time you change the parameter value. To have Katana automatically create keys when you enter a new value, you need to turn on **Auto Key** mode for that parameter.

Toggling Auto Key

While a parameter has the **Auto Key** icon highlighted  , entering a value in the parameter field creates a new keyframe at the current frame.

You can toggle Auto Key mode by:

- Right-clicking on the parameter to toggle and selecting **Auto Key**.



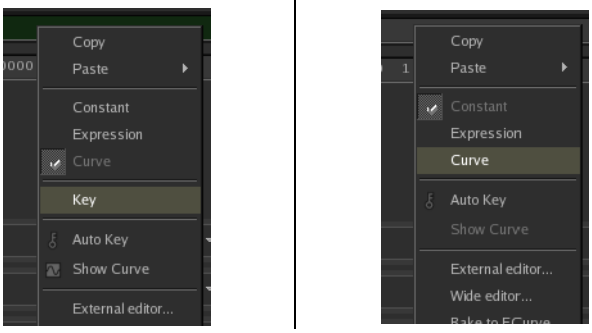
OR

- Clicking the **Auto Key** icon,  or  , next to the parameter.

Setting Keys Manually

To set a key manually:

1. Move the **Timeline** to the correct frame.
2. Set the parameter to the desired value.
3. Right-click the parameter and select **Key**. If a key has not been set on the parameter before, select **Curve**. Selecting Curve not only sets a key, it also converts that parameter from a Constant or Expression to a Curve.



Note *You can also set keys within the Curve Editor tab using Insert mode—see* Setting Keys in the Curve Editor *—as well as converting an interpolated value into a key — see* Baking a Segment of the Curve*.*

Baking a Curve

Whether from an expression or a keyframed curve, you can convert part or all of it to keyframes.

To generate keyframes from a curve or expression:

1. Right-click on the parameter.
2. Select **Bake to FCurve...** .

   The **Bake to Curve** dialog appears.

3. Change the dialog values to suit the curve you are creating. You can change the:
   - **startFrame** — the frame to start generating keys.
   - **endFrame** — the last frame to generate a key.
   - **interval** — how often to generate a key (in frames).
4. Click **Bake**.

   The parameter changes from an expression to a curve and keys are generated from **startFrame** to **endFrame**.

   All the newly generated keys are assigned the linear segment function.

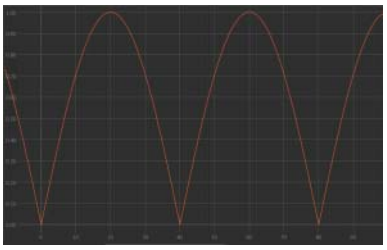


Figure 15.9: The expression abs(sin(frame*pi/40)) displayed in the Curve Editor.

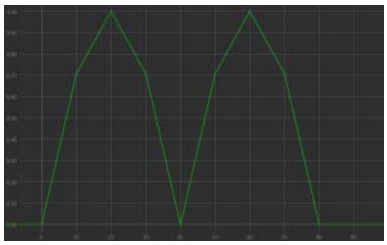


Figure 15.10: The curve generated by baking with a startFrame of 0, endFrame of 80, and an interval of 10.

Note *Although most commonly used with expressions, **Bake to FCurve...** can be used to automatically generate keyframes for any type of parameter, whether it's an expression, a constant, or already a curve.*

Exporting and Importing a Curve

Curves can be exported and imported.

To export a curve:

1. Right-click on the parameter to export.
2. Select **Export FCurve...** .

To import a curve:

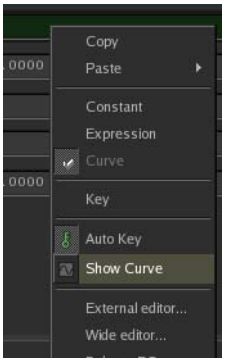
1. Right-click on the parameter to change.
2. Select **Import FCurve...** .

Displaying Keyframes

You can use the **Curve Editor** tab, **Dope Sheet** tab, and **Timeline** to view and manipulate keyframes. They only show a parameter's keyframes if the parameter has the **Show Curve** icon highlighted.

To toggle the Show Curve icon:

1. Right-click on the parameter.
2. Select the **Show Curve** menu item.

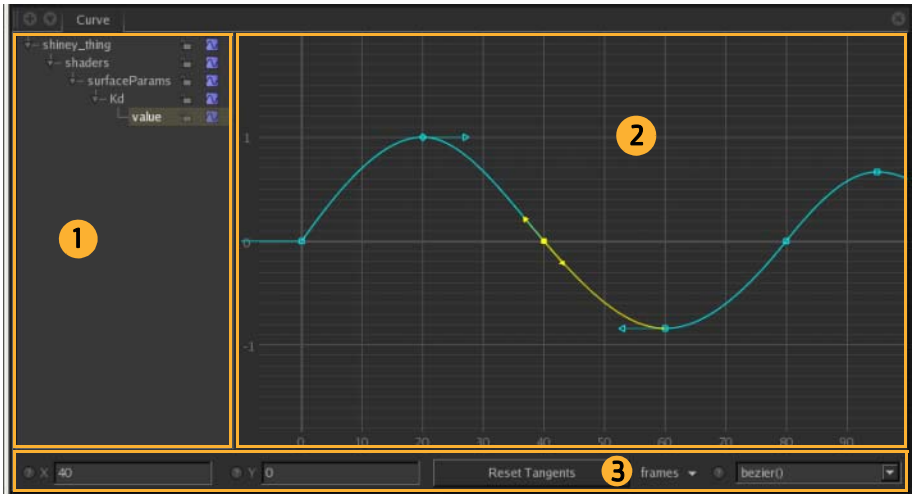


OR

Click  or  to the left of the parameter input field.

Curve Editor Overview

The Curve Editor is the heart of animating within Katana. Here you can move keyframes; change their segment function, tangents and weights; set breakdowns; and make any curve manipulations necessary to get the curve you need.



The **Curve Editor** is split into three areas:

1. The left-hand side is a hierarchical view of all parameters with **Show Curve** enabled.
2. The right-hand side shows these parameter values plotted over time. The parameter value range is on the left and the time frame across the bottom. This area is referred to as the **Curve Editor** graph.
3. The bottom of the **Curve Editor** has a toolbar containing ways to manipulate the keyframes.

Tip *Although the **Curve Editor** is primarily for manipulating curves, it can also be used to view the results of an **Expression**. To view an **Expression** in the **Curve Editor,** enable **Show Curve** for the **Expression** parameter.*

Using the Hierarchical View

On the left of the **Curve Editor** is a hierarchical view of the curves and expressions that have **Show Curve** enabled. You can use this view to expand and collapse the parameters, lock the curves against editing, and toggle the curves that are shown in the **Curve Editor** graph.

To expand or collapse a curve:

Double-click on the part of the parameter name to expand or collapse.
OR

Click  to expand or  to collapse.

Note *Collapsing a parameter in the hierarchical view only changes whether its children are displayed in the hierarchical view. Its only use is to keep the hierarchy more manageable.*

To select a curve in the hierarchical view:

Click on a parameter name to select its curve — it must be the leaf name as that corresponds to the actual parameter.

Tip *You can select more than one parameter by **Ctrl+clicking** further parameters and **Shift+clicking** to select all the parameters from your last selection to where you click.*

Locking a Curve

You can lock a parameter to stop its curve from being editable within the **Curve Editor**.

To lock a parameter and stop it from being editable within the Curve Editor:

Click  within the hierarchical view in the **Curve Editor**.

To unlock a parameter:

Click  .

Note *Parameters that are expressions are always locked and cannot be modified within the Curve Editor.*

Hiding and Showing a Curve

Even though a parameter has **Show Curve** selected, you may not want to display it within the **Curve Editor** graph.

To hide a parameter curve within the Curve Editor:

Click  within the hierarchical view in the **Curve Editor**.

To show a parameter curve that has previously been hidden:

Click  within the hierarchical view in the **Curve Editor**.

Switching the Display of a Parameter's Children

When only some of the children of a parameter are shown,  is displayed.

To switch the display state of the children of a parameter name:

Click  within the hierarchical view in the **Curve Editor**.

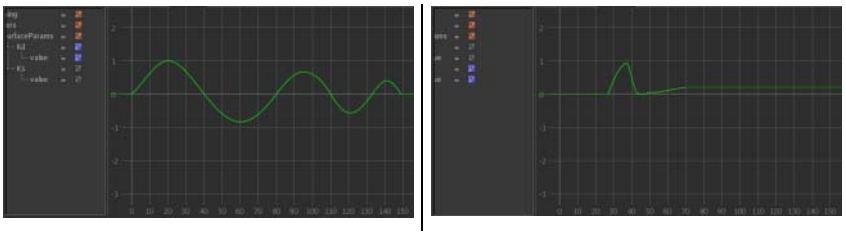


Figure 15.11: By clicking  the two child curves have changed their display states — one becoming hidden and the other visible.

Setting Keys in the Curve Editor

To set keys quickly and easily within the **Curve Editor**, you can use insert mode. The insert mode enables you to click on the graph at any point and insert a new key at that position.

To insert keys with insert mode:

1. Select the curve for the new keys.
2. Press the **Insert** key.

   This puts you into insert mode.
3. Click a point on the graph to insert a new key at that position.
4. Repeat step 3 to insert as many keys as required.

   Select a different curve within the hierarchical view to insert keys on that curve.
5. To finish adding keys and disable insert mode, press **Insert** again.

Selecting Keyframes in the Curve Editor

To select keyframes you can:

click on them,
OR
marquee drag over them.

To select all keyframes for a curve:

Double click the curve.

To add to a current selection:

Hold **Shift** while selecting the keyframe(s).

To remove from a current selection:

Hold **Ctrl** while selecting the keyframe(s) to remove.

Moving Keyframes in the Curve Editor

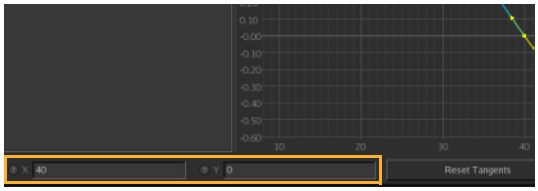
You have two ways to move keyframes within the Curve Editor: using the mouse or using the **X** and **Y** input fields.

To move keyframes using the mouse:

1. Select the keyframe(s) you want to move.
2. Click-and-drag one of the selected keyframes.

To move a single keyframe using the input fields:

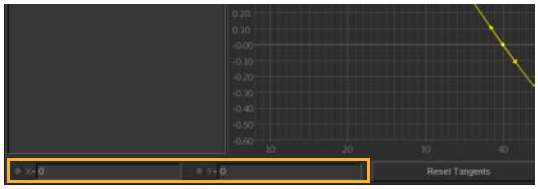
1. Select the keyframe you want to move.
2. Make any changes in the input fields below the graph:
   - Enter a new frame number in the **X** input field.
   - Enter a new value in the **Y** input field.



The values entered into **X** and **Y** are absolute and not relative. For instance, entering 10 in the **X** input field, moves the keyframe to frame 10.

To move multiple keyframes using the input fields:

1. Select the keyframes you want to move.
2. Make any changes in the input fields below the graph. All changes are relative, for instance 3 would add 3 to the current value or frame number and -3 would subtract 3 from the current value or frame number:
   - Enter a relative frame number in the **X+** input field.
   - Enter a relative value in the **Y+** input field.



Changing the Display Range in the Curve Editor Graph

Katana provides a number of ways to change the frame range and parameter value range in the Curve Editor graph.

To pan the Curve Editor graph:

Middle-click and drag within the graph area.

To pan in an axis:

Shift+middle-click and drag within the graph area.

To zoom the Curve Editor graph in and out:

Use the scroll wheel — Scroll up to zoom in and down to zoom out.
OR
Press **+** (**Plus** key) to zoom in or press **-** (**Minus** key) to zoom out.

To frame all the keyframes in the Curve Editor graph:

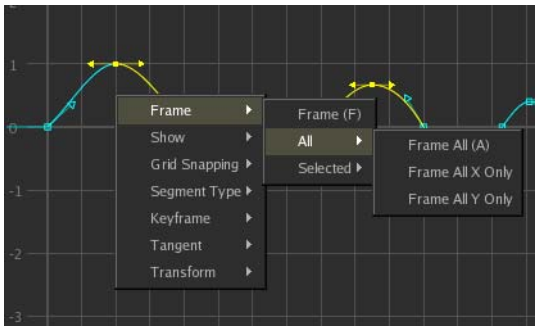
Right-click and select **Frame > All > Frame All** (or press **A**).

To frame all the keyframes in the Curve Editor graph in the x axis:

Right-click and select **Frame > All > Frame All X Only.**

To frame all the keyframes in the Curve Editor graph in the y axis:

Right-click and select **Frame > All > Frame All Y Only.**



To frame the selected keyframes:

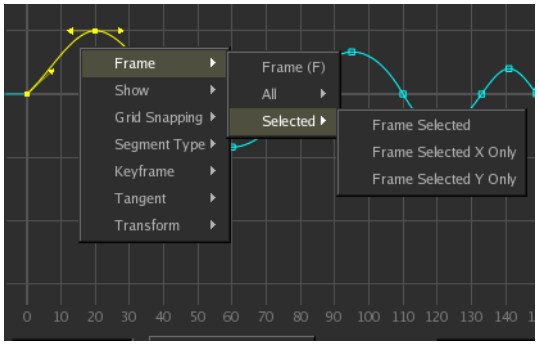
Right-click and select **Frame > Frame** (or press **F**).
OR
Right-click and select **Frame > Selected > Frame Selected**.

To frame the selected keyframes in the x axis:

Right-click and select **Frame > Selected > Frame Selected X Only.**

To frame the selected keyframes in the y axis:

Right-click and select **Frame > Selected > Frame Selected Y Only.**



Changing Display Elements within the Curve Editor Graph

You can display other information in conjunction with the parameter curves. Additional elements that can be displayed include: a domain slider to show the value on a curve for a given time; a curves velocity and acceleration; and a label to identify which curve corresponds to which parameter.

To toggle the display of the Domain Slider:

Right-click and select **Show > Domain Slider** (or press **D**).

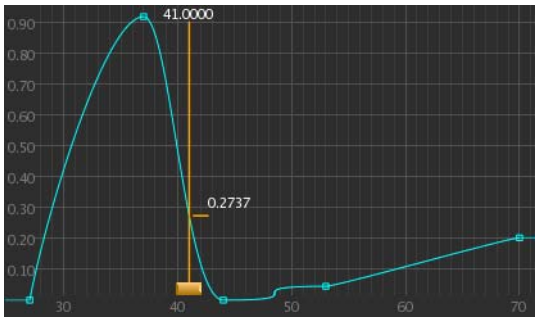


Figure 15.12: The **Domain Slider**, the orange vertical bar, can be moved left and right across the frame range to display the value for the highlighted curve at a particular frame.

Displaying a Velocity Curve

You can use a velocity curve for a parameter to help you spot non-tangential keyframes; these are characterized by breaks in the velocity curve. Non-tangential keyframes can be jarring when making realistic movement through animation. The velocity curve is calculated by analyzing

the changes in the y axis of the curve at small increments along the x axis.

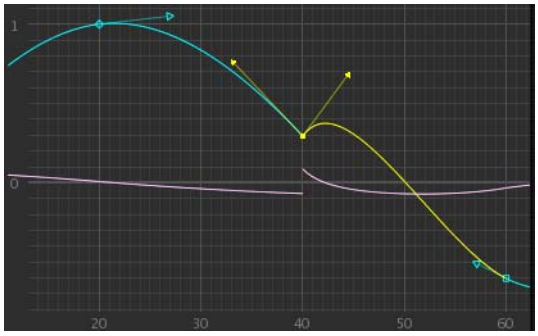
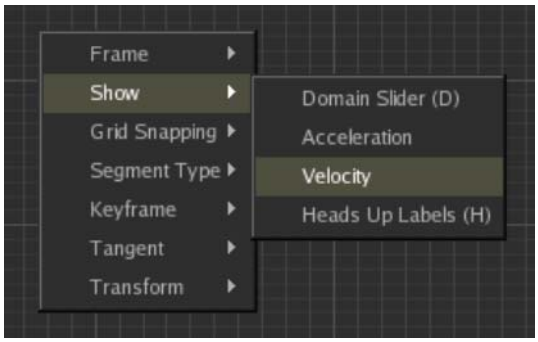


Figure 15.13: The lavender velocity curve is broken (not continuous) at frame 40, as is the highlighted tangent.

To toggle the display of a curve's velocity:

1. Select the curve(s) within the hierarchical view.
2. Right-click an empty part of the graph and select **Show > Velocity**. The velocity curve is shown in lavender.



Displaying an Acceleration Curve

You can use the acceleration of a curve to provide a useful insight into the forces that act on that curve. For instance, an object whose only force is gravity should have a horizontal acceleration curve (assuming it doesn't hit

anything).

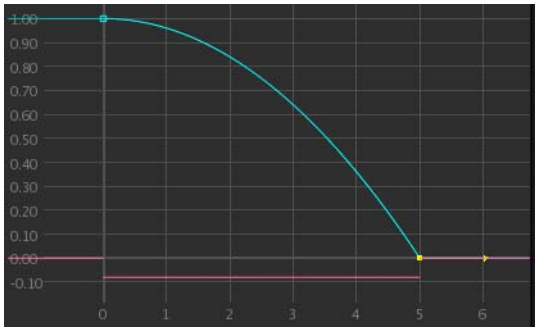
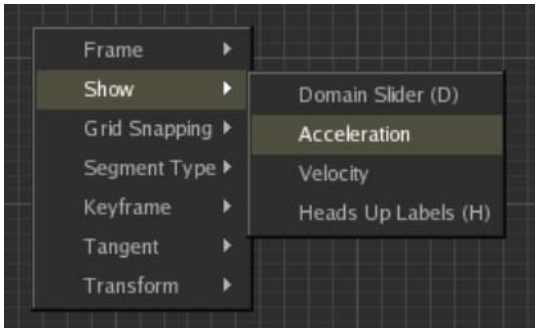


Figure 15.14: Between frames 0 and 5 the acceleration curve shows a consistent force is acting on the parameter (the acceleration curve is straight).

To toggle the display of a curve's acceleration:

1. Select the curve(s) within the hierarchical view.
2. Right-click anywhere on the graph and select **Show > Acceleration**.
   The acceleration curve is shown in pink.



To toggle the display of curve labels:

Right-click and select **Show > Heads Up Labels** (or press **H**).

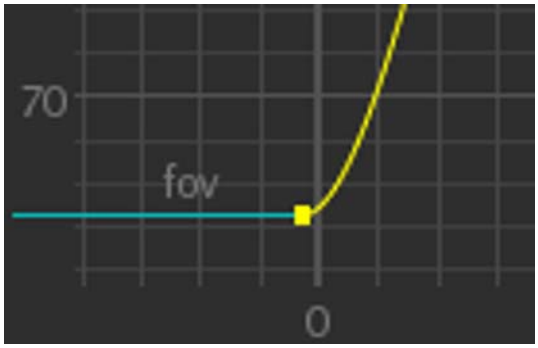


Figure 15.15: The curve label, based on the parameter name, sits just above the curve on the left-hand side.

Snapping Keyframes

When moving keyframes within the **Curve Editor** tab, you can snap their values in place. Snapping to the x axis affects the frame number and snapping to the y axis affects the parameter's value.

To snap a keyframe's frame number while moving it within the Curve tab, you can:

- Right-click and select **Grid Snapping > X Snap to Integers** — Katana snaps keyframe changes to whole frame numbers.
- Right-click and select **Grid Snapping > X Snap to Grid** — Katana snaps keyframe changes to the vertical grid lines.

Note *Selecting either of these menu options does not change the current y axis snap settings.*

To snap a keyframe's value while moving it within the Curve tab, you can:

Right-click and select **Grid Snapping > Y Snap to Grid** — Katana snaps value changes to the horizontal grid lines.

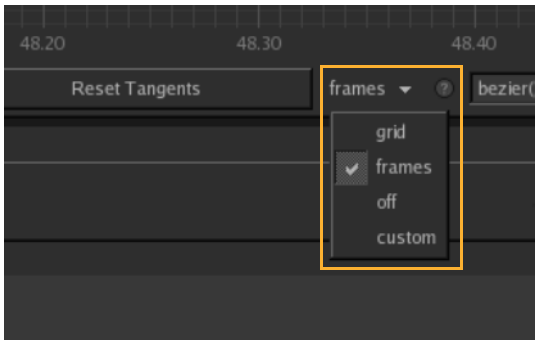
To stop snapping a keyframe, you can:

- Right-click and select **Grid Snapping > X Snapping Off** — Katana no longer snaps keyframe changes in the x axis.
- Right-click and select **Grid Snapping > Y Snapping Off** — Katana no longer snaps keyframe changes in the y axis.

- Select **off** from the dropdown menu to the right of the **Reset Tangents** button at the bottom of the **Curve Editor** — Katana no longer snaps keyframe changes in any direction.

Katana also comes with some predefined snapping options in a dropdown menu to the right of the **Reset Tangents** button at the bottom of the **Curve Editor**. These are:

- **off** — Katana no longer snaps keyframe changes in any direction.
- **frames** — Katana snaps the x axis to whole frame numbers but does not snap the keys in the y axis.
- **grid** — Katana snaps the keyframes to grid intersection points.
- **custom** — The last snap setting you selected that does not match **frames**, **grid**, or **off**. (This option only becomes available once you have made a snap setting change that does not match frames, grid, or off.)



To cycle through the preset snapping options (off, frames, and grid):

Right-click and select **Grid Snapping > Cycle Snapping** (or press **S**).

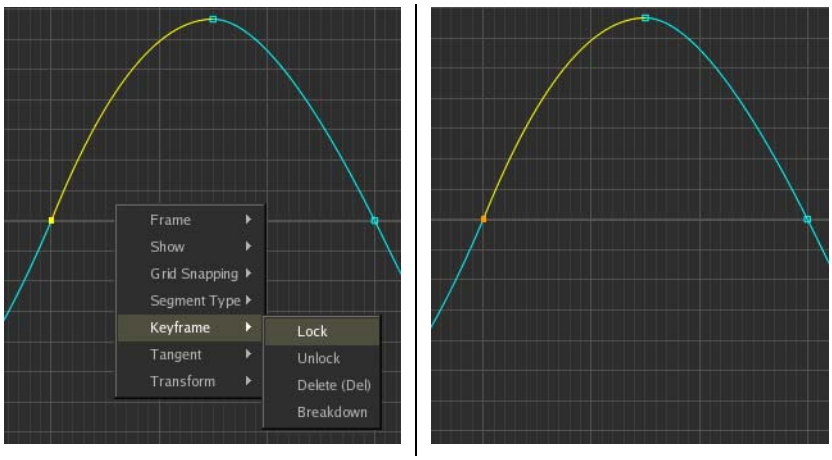
Locking, Unlocking, and Deleting Keyframes

To lock keyframes to prevent accidental editing:

1. Select the keyframes to lock.
2. Right-click and select **Keyframe > Lock**.

   Katana locks the keyframes and turns them orange.

Note *Locking a keyframe only applies to inside the **Curve Editor** tab.*



To unlock keyframes:

1. Select the keyframes to unlock.
2. Right-click and select **Keyframe > Unlock**.
   Katana unlocks the keyframes and turns them yellow.

To delete keyframes:

1. Select the keyframes to delete.
2. Right-click and select **Keyframe > Delete** (or press **Delete**).

Turning a Keyframe into a Breakdown

Katana supports a special kind of keyframe known as a breakdown. Breakdowns help you describe the motion between two keyframes by providing an intermediate value. Breakdowns maintain the same relative time with the keyframes either side, this helps maintain timing. For instance, with keyframes on frames 0 and 60 and a breakdown on frame 20, moving the keyframe on frame 60 to frame 30 would automatically move the breakdown to frame 10, thereby maintaining the 1:2 ratio of frames before and after. If a breakdown falls at the beginning or end of a curve, then moving the keyframe next to it moves the breakdown.

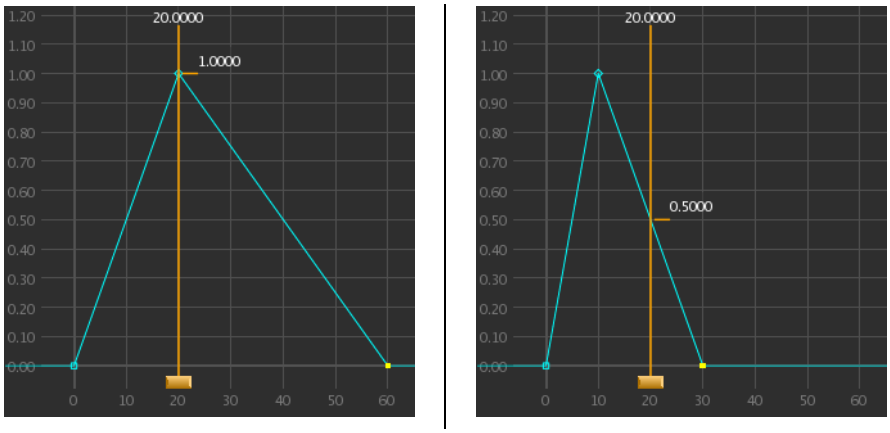
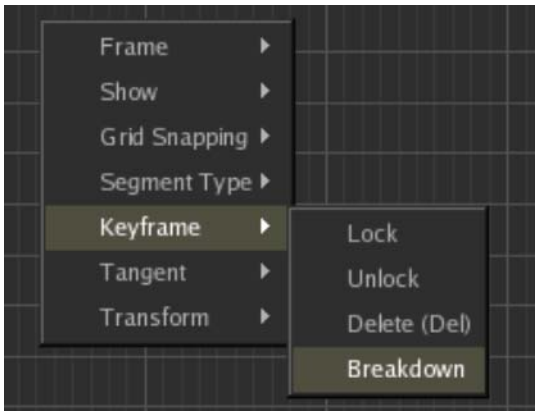


Figure 15.16: When the keyframe on frame 60 is moved to frame 30, the breakdown on frame 20 automatically moves to frame 10.

To convert a keyframe into a breakdown:

1. Select the keyframe(s) to convert.
2. Right-click and select **Keyframe > Breakdown**.



Tip *To change a breakdown back to a keyframe, repeat the steps above.*

Note *Breakdowns are only different to keyframes while within the Curve Editor. Elsewhere, such as within the Dope Sheet, breakdowns are treated as normal keyframes.*

Changing a Segment Function

Katana interpolates the values between one keyframe and the next based on the segment function assigned to the first of the two keyframes. Three

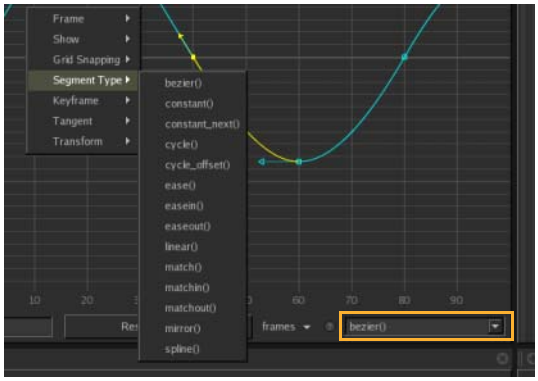
special segment functions can also be assigned to the segment before the first keyframe or after the last: cycle(), cycle_offset(), and mirror().

To change the segment function for either a keyframe or for the segment at the beginning or end of a curve:

1. Select the keyframe(s) or segment to change (to select a segment click on it).
2. Then, either:
   - Right-click and select **Segment Type > ...** .

     OR

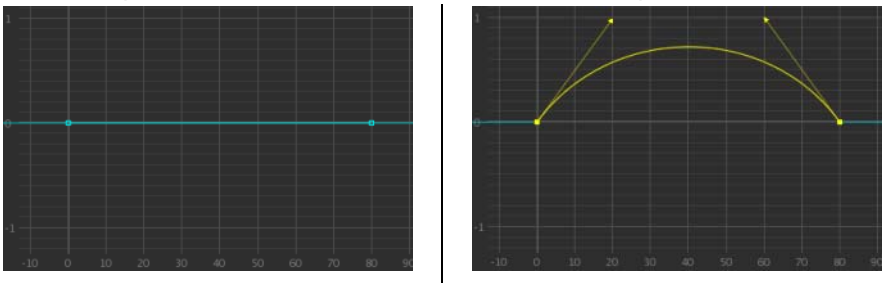
   - Select the segment function from the dropdown menu in the bottom right corner of the **Curve Editor**.



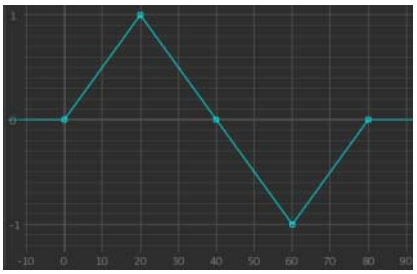
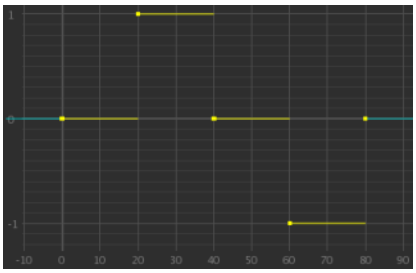
You can choose one of the following segment functions:

- **bezier()**

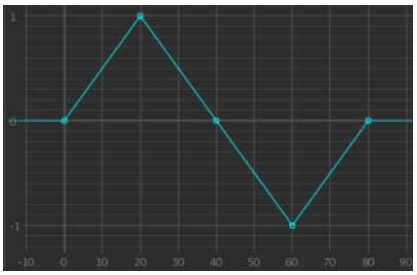
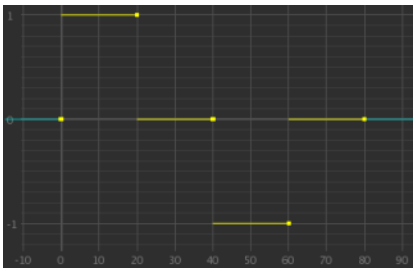
  The bezier segment function is the most versatile. It uses four points—the keyframes at the start and end, and two control points—to define the segment. The control point position is shown with an arrowhead. The weight of a control point, which determines how strong its influence is over the generated curve, is determined by the length of the handle.



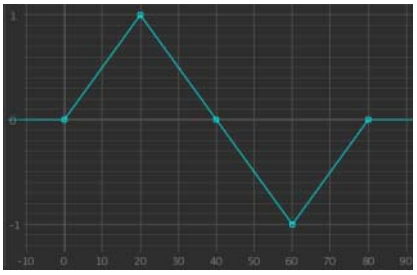
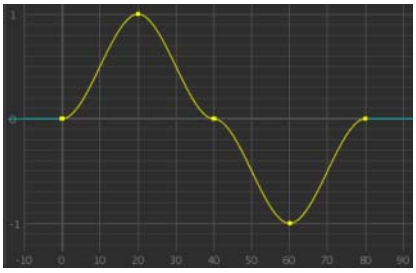
- **constant()**
  The constant segment function uses the keyframe's value for the entire segment.

- **constant_next()**
  The constant_next segment function uses the next keyframe's value for the entire segment.

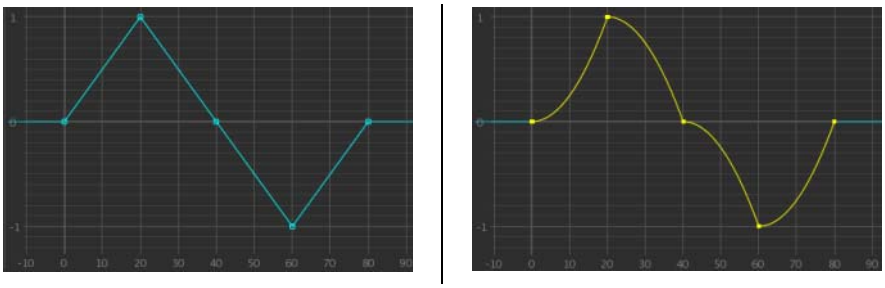



- **ease()**
  The ease segment function flattens out the segment at its beginning and end. This is similar to having flat tangents on the two control points when using bezier curves.

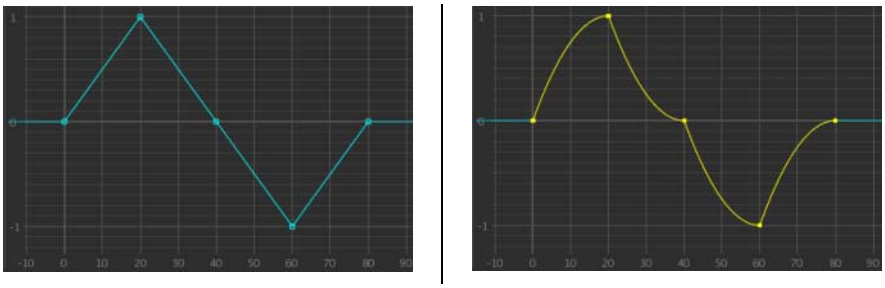



- **easein()**
  The easein segment function starts the segment flat and then maintains the same acceleration until it reaches the next keyframe. This results in

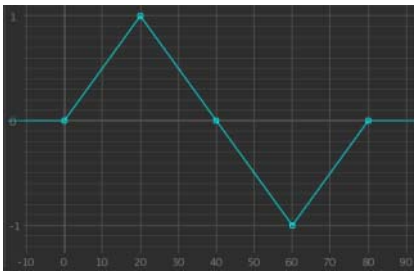
the velocity curve for the segment being a straight line that starts at zero.



- **easeout()**
  The easeout segment function finishes the segment flat while maintaining a constant acceleration throughout the segment. This results in both the velocity curve for the segment being a straight line that ends at zero.

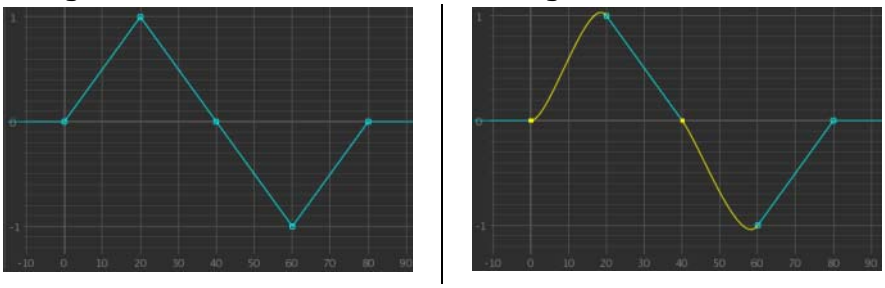


- **linear()**
  The default segment function. The values from one keyframe move in a straight line to the next keyframe.



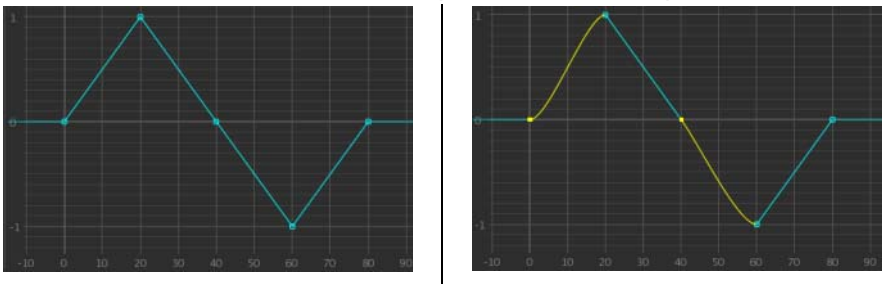
- **match()**

The match segment function gives the segment the same velocity (rate of change) at both the start and end of the segment.



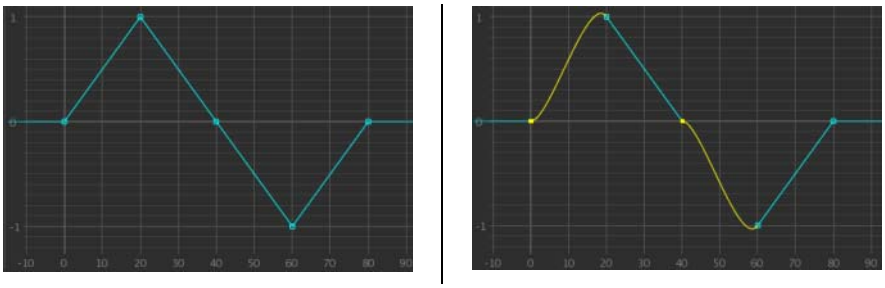
- **matchin()**

  A segment with the matchin segment function begins with a velocity that matches that at the end of the previous segment, the segment ends with zero velocity. This has the effect of making the tangent at the start match the slope of the previous segment and the tangent at the end flat.



- **matchout()**

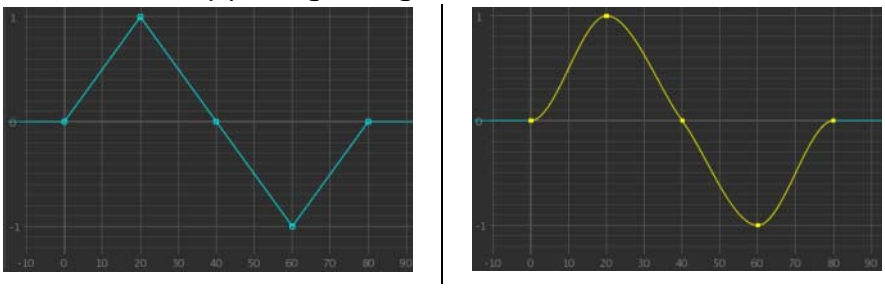
  A segment with the matchout segment function begins with zero velocity and ends with a velocity that matches that at the beginning of the next segment. This has the effect of making the tangent at the start flat and the tangent at the end match the slope of the next segment.



- **spline()**

  The spline segment function uses the Catmull-Rom spline function that uses four keyframes to calculate the value at a given frame. As the frame

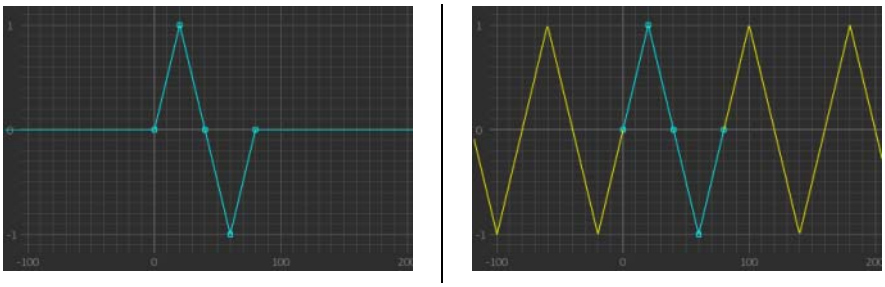
approaches a keyframe, the curve tends towards the value at the keyframe, eventually passing through it.



You can choose one of the following extrapolation functions:

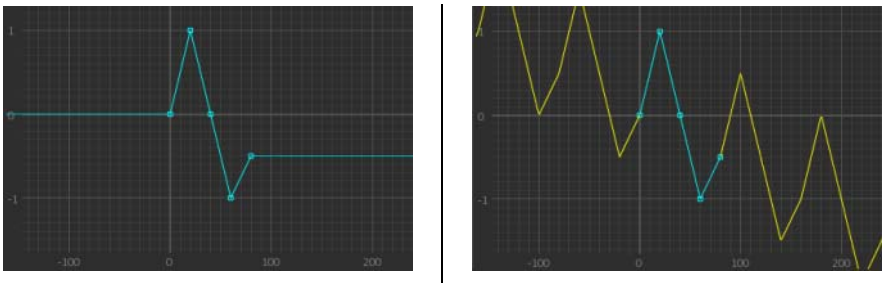
- **cycle()**

  The cycle extrapolation function repeats the curve an infinite number of times either before (if applied to the segment before the first keyframe) or after (if applied to the segment after the last keyframe).



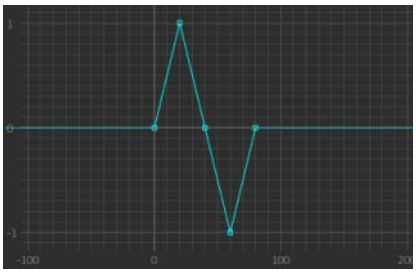
- **cycle_offset()**

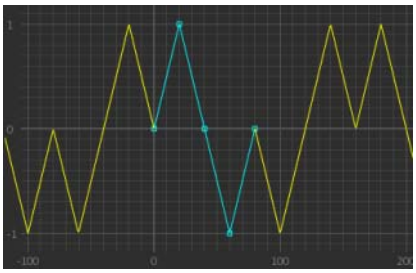
  The cycle_offset segment function only works on the segments at the start or end of a curve. It should not be used on a keyframe. It repeats the curve an infinite number of times; each time the curve repeats the new beginning keyframe starts from the end keyframe from the previous cycle, thus offsetting the curve.



- **mirror()**

The mirror segment function only works on the segments at the start and end of a curve. It continuously flips the curve vertically.





Tip *It is also possible for you to type your own segment or extrapolation function in the dropdown menu. The function must use python syntax;* ***x()*** *can be used to represent the current frame. For instance, sin(x()*pi/20).*

Changing the Control Points of a Bezier Segment Function

Of all the segment functions, the bezier is the most versatile. With the addition of two control points, you have much finer control over how the curve flows between keyframes.

When you change the segment function at a keyframe, you change how the curve is interpolated from that keyframe to the next. When you change the tangent at a keyframe, you affect the control points that sit either side of that keyframe.

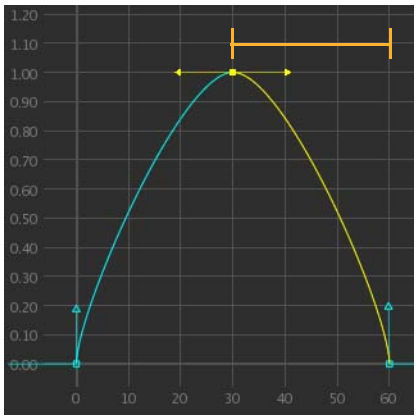


Figure 15.17: The range of any changes to the segment function.

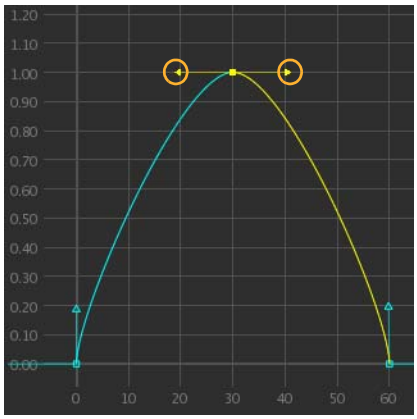


Figure 15.18: The control points influenced by tangent changes.

To change the tangent type at a keyframe:

1. Select the keyframe(s) to change the control points.
2. Right-click and select **Tangent > Type > ...** .

You can choose one of the following tangent types:

- **Fixed**

  The Fixed tangent type doesn't change the current control points but they no longer update as keyframes around them are moved. This becomes the tangent type once any tangent has been manually moved.

- **Flat**

  The Flat tangent type makes the control points sit horizontally either side of the keyframe.

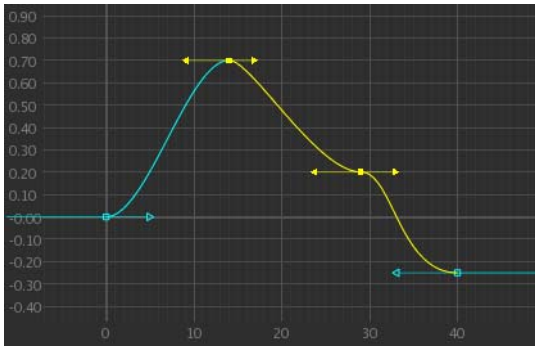


Figure 15.19: All the keyframes are using the Flat tangent type.

- **Linear**

  The Linear tangent type places the control point directly in line with the keyframe that acts as the other anchor point for the segment. If both control points for a bezier segment are linear, the segment is a straight line from one keyframe to the next.

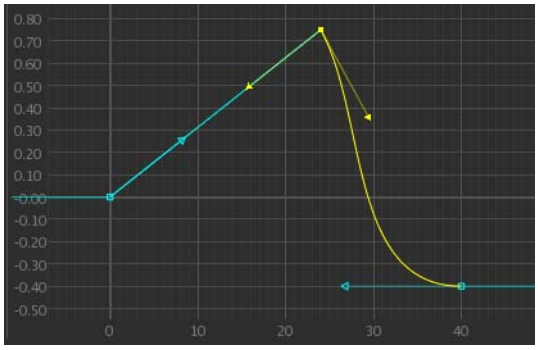


Figure 15.20: The first and middle keyframes use the linear tangent, the right keyframe does not.

- **Smooth**

  The Smooth tangent type places the control points either side of a keyframe forming a line that runs parallel to a line formed by the keyframes either side.

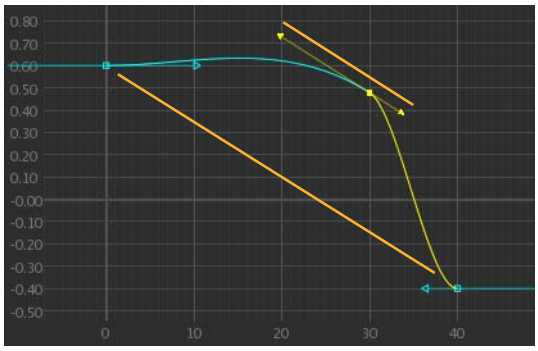


Figure 15.21: The line formed by the control points remains parallel to the line created by the keyframes.

- **Smooth Normal**

  The Smooth Normal tangent type places the two control points vertically in line with the keyframe. Whichever keyframe is higher between the keyframes to the left and right, controls the direction of the curve. Should the keyframes to the left and right be equal, both control points are placed vertically below.

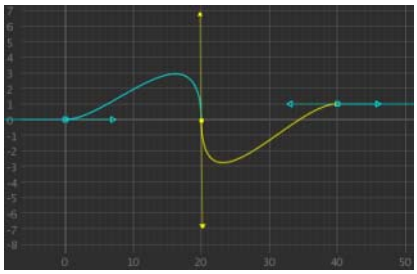


Figure 15.22: With the right keyframe above the left, the curve goes down through the middle keyframe.

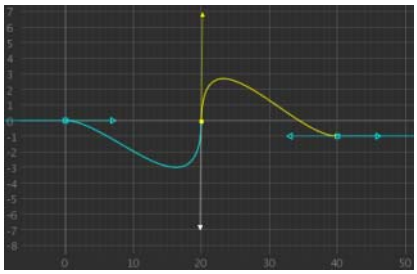


Figure 15.23: With the right keyframe below the left, the curve goes up through the middle keyframe.

- **Plateau**

  The Plateau tangent type uses the Flat and Smooth tangent types depending on its keyframes location relative to the keyframes on either side. If the keyframes on either side are both above or both below the tangent's keyframe, then the Flat tangent type is used. If the tangent's keyframe falls between the values for the keyframes on either side, then the Smooth tangent type is used. When using the Smooth tangent type, if one of the control points for the tangent would fall outside the range

between the keyframes on either side, then that control point converts to the Flat tangent type instead.

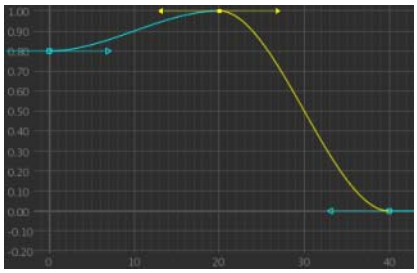


Figure 15.24: Here the Plateau tangent type uses the same algorithm as the Flat tangent type.

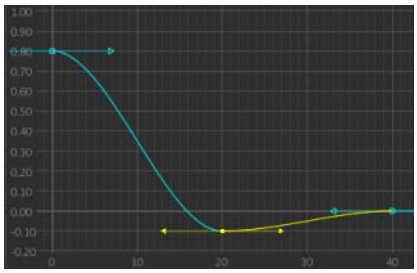


Figure 15.25: Once again the Flat tangent type is used.

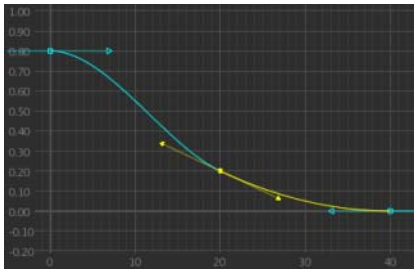


Figure 15.26: Here the Plateau tangent type uses the same algorithm as the Smooth tangent type.

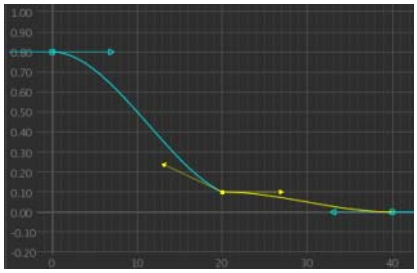


Figure 15.27: As the lower control point would drop below the keyframe to the right, that control point becomes Flat.

To change between weighted and non-weighted tangents:

1. Select the keyframes whose tangents you want to change.
2. Right-click and select **Tangent > Weighted**.

   Katana toggles the tangent between weighted and non-weighted.

What are weighted and non-weighted tangents?

With a non-weighted tangent using the manipulator only changes the angle of the control point. Weighted tangents enable you to change the amount of influence a control point has over the segment function by changing the distance from the keyframe to the end of the tangent. The bigger the distance, the more influence the control point has.

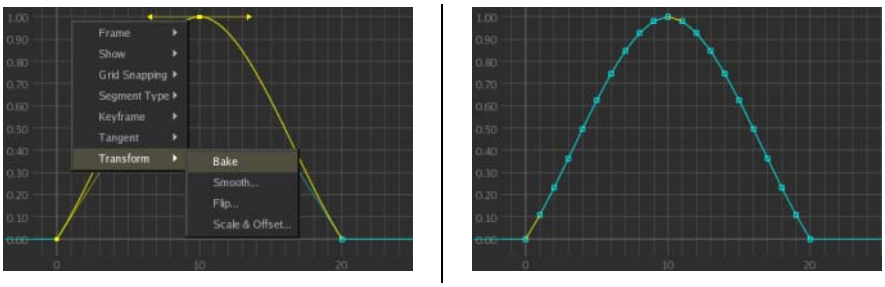
Baking a Segment of the Curve

Baking a segment of the curve converts the interpolated values at each frame of the segment into keyframes.

To bake a segment of the curve:

1. Select the keyframe at the start of the segment.

2. Right-click and select **Transform > Bake**.



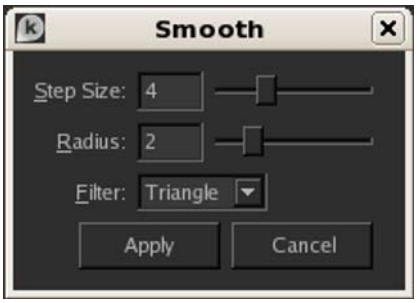
Tip *Multiple segments can be baked at once by selecting multiple keyframes.*

Smoothing a Segment of the Curve

Smoothing a segment of the curve makes the curve flatter — reducing its peeks and troughs.

To smooth a segment of the curve:

1. Select the keyframe at the start of the segment you want to smooth.
2. Right-click and select **Transform > Smooth...** .
   The **Smooth** dialog appears.



3. Change the values within the dialog where appropriate:
   - **Step Size** — how often to create a keyframe.
   - **Radius** — how much to smoothen the curve (higher values for smoother, lower values for closer to original).
   - **Filter** — which algorithm to use, **Triangle** or **Box**.
4. Click **Apply** to smooth the curve.

Note *For the best results, smooth multiple segments at once by selecting a number of keyframes together.*

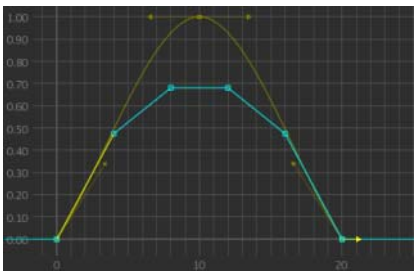


Figure 15.28: Smoothing with the default settings: **Step Size** 4, **Radius** 2, and Triangle **Filter** (the original curve is ghosted out).

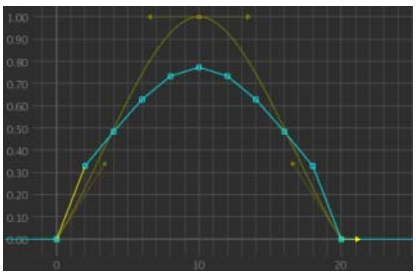


Figure 15.29: Smoothing with **Step Size** 2 and **Radius** 4.

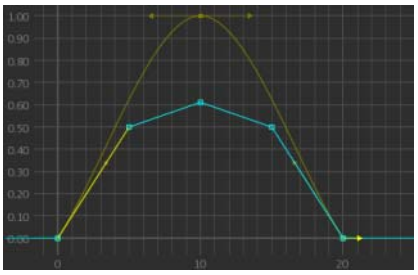


Figure 15.30: **Step Size** 5 and **Radius** 2.

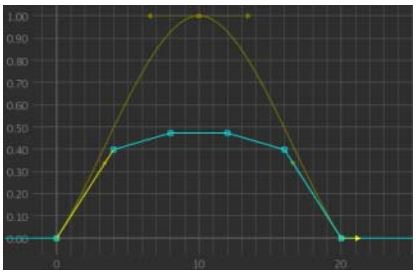


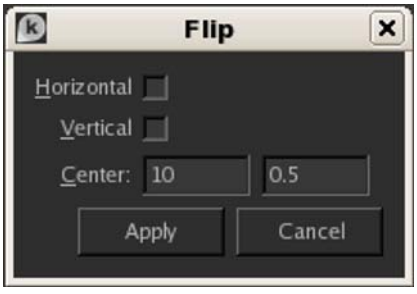
Figure 15.31: **Step Size** 4 and **Radius** 4.

Flipping the Curve Horizontally or Vertically

You can flip a curve either horizontally or vertically.

To flip a curve horizontally or vertically:

1. Select a curve or a curve's keyframe.
2. Right-click and select **Transform > Flip...** .
   The **Flip** dialog appears.



3. Select whether you want to flip the curve horizontally or vertically or both:
   - **Horizontal** (press **Alt+H**) — flips the curve horizontally.
   - **Vertical** (press **Alt+V**) — flips the curve vertically.
   - **Center** (press **Alt+C**) — the point at which to flip the curve (if a keyframe is selected it defaults to that keyframe position).
4. Click **Apply** to flip the curve.

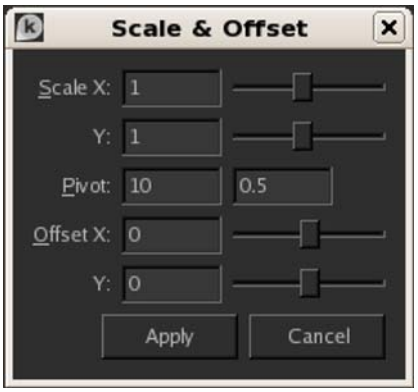
Scaling and Offsetting a Curve

Katana gives you the ability to scale or offset a curve.

To scale or offset a curve:

1. Select the curve or a curve's keyframe.
2. Right-click and select **Transform > Scale & Offset...** .

   The **Scale & Offset** dialog appears.

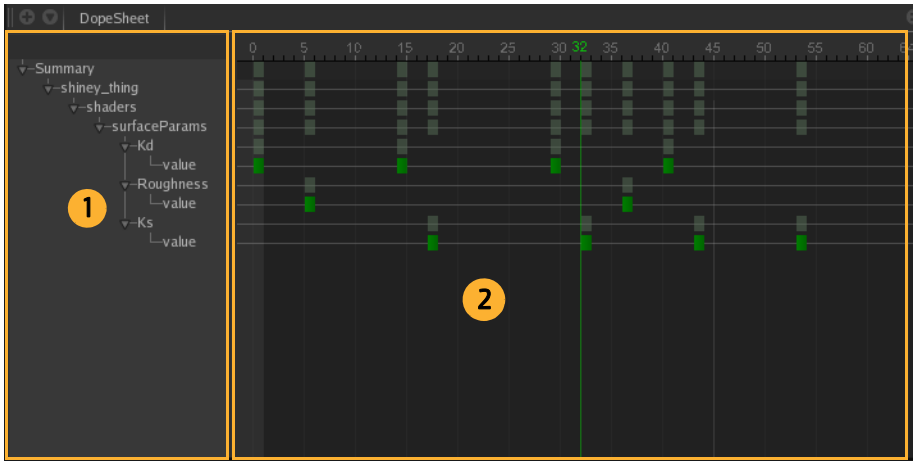


3. Change the values within the dialog to get the desired effect:
   - **Scale** (press **Alt+S**) — scales in either the x direction (changing timing) or y direction (changing the parameter value). Negative values reflect the curve about the values entered in the **Pivot** fields.
   - **Pivot** (press **Alt+P**) — the point about which to scale (if a keyframe is selected it defaults to that keyframe position).
   - **Offset** (press **Alt+O**) — moves the curve in the direction of the offset.
4. Click **Apply** to effect the curve.

Dope Sheet Overview

In the Dope Sheet, you can manipulate keyframes by either retiming (sliding them left or right) or copy and pasting. The Dope Sheet's simple interface makes it easy for you to see keyframe timings across multiple parameters,

whereas the Curve Editor can become cluttered when dealing with more than one curve.



1. The Dope Sheet has a hierarchical view down the left-hand side.
2. The main area has time (in frames) across the top and blocks (to signify keyframes) at the intersection of their parameter on the left-hand side and their frame number above.

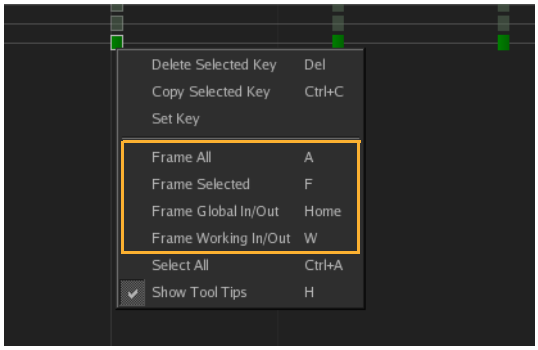
Note *Within the Dope Sheet breakdowns are treated as normal keyframes — this means they do not move automatically when the keyframes either side are moved.*

Changing the Displayed Frame Range

There are multiple ways for you to change the frame range displayed within the Dope Sheet. You can:

- Scroll the mouse-wheel; to zoom in scroll up and to zoom out scroll down.
- **Alt+middle**-click and drag.
- Press **+** (plus key) to zoom in or **-** (minus key) to zoom out.
- Right-click and select **Frame All** (or press **A**) to have the frame range zoom to fit all the keyframes.
- Right-click and select **Frame Selected** (or press **F**) to have the frame range zoom to fit only the selected keyframes.
- Right-click and select **Frame Global In/Out** (or press **Home**) to have the frame range go from the project settings' **inTime** to the project settings' **outTime**.

- Right-click and select **Frame Working In/Out** (or press **W**) to have the frame range go from **In** to **Out** from the **Timeline**.



Panning the Displayed Frame Range

To pan the displayed frame range within the Dope Sheet, middle-mouse drag.

Selecting Keyframes

The Dope Sheet has standard controls for selecting single or multiple keyframes.

To select a keyframe:

- click on it,
  OR
- drag a marquee around it.

To select multiple keyframes:

- drag a marquee around all the keyframes you want to select,
  OR
- right-click and select **Select All** (or press **Ctrl+A**) to select all visible keyframes.

To add to a selection:

Click or drag a marquee over the keyframe(s) while holding the **Shift** key.

To remove from a selection:

Click or drag a marquee over the keyframe(s) while holding the **Ctrl** key.

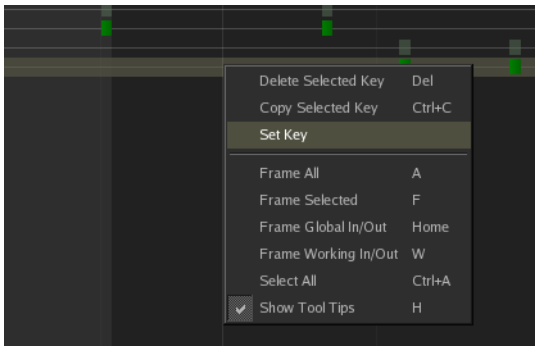
Moving Keyframes

To move keyframe(s):

1. Select the keyframe(s) to move.
2. Click on one of the selected keyframe(s) and drag left or right.

Creating a Keyframe from an Interpolated Value

At times you may want to convert an interpolated value into a keyframe; you can achieve this by right-clicking at the intersection of the frame and parameter (this is where the keyframe block appears) and selecting **Set Key**. A new keyframe is created with the same value as previously interpolated at that frame.

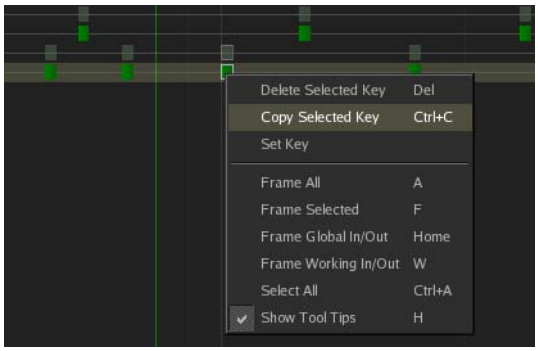


Copy and Pasting Keyframes

The **Dope Sheet** provides the simplest method for copying and pasting keyframes.

To copy keyframe(s):

1. Select the keyframe(s) to copy.
2. Right-click and select **Copy Selected Key(s)** (or press **Ctrl+C**).



To paste keyframe(s):

Right-click and select **Paste Key**.

If you right-click on an empty part of the **Dope Sheet**, the keyframe(s) are inserted in the same parameter from which it was copied at the point shown by a ghosted vertical line.

If you right-click horizontally in line with a parameter, the keyframe(s) are added there. The precise positions are highlighted when you first right-click.

OR

1. Using the **Timeline**, move the current frame to where you want to insert the new keyframe(s).
2. Press **Ctrl+V**.

Deleting Keyframes

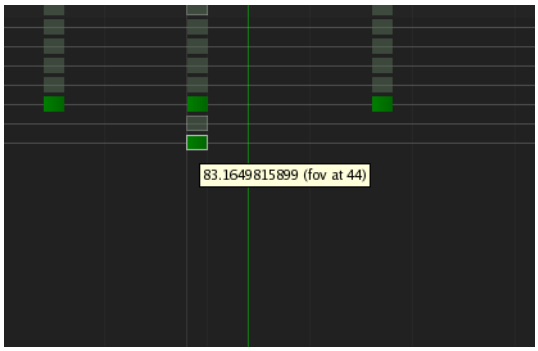
To delete keyframe(s):

1. Select the keyframe(s) to delete.
2. Right-click and select **Delete Selected Key(s)** (or press **Del**).

Tip *When creating, copying, or deleting keyframes within the Dope Sheet, it is a good idea to keep checking the new curve within the Curve Editor to make sure the curve segments are interpolated using the right segment function.*

Toggling Tooltip Display

To toggle tooltip display, right-click and select **Show Tool Tips** (or press **H**). The value, parameter name, and frame number for the keyframe display.



16 Using the Timeline

Using the Timeline

Katana's timeline allows you to move from one frame to another and view keyframes over the frame range.

Changing the Current Frame

To change frames in Katana, you can:

- Press the **Right Arrow** key to increment the current frame by **Inc**, or **Left Arrow** key to decrement.
- Click  to increment the current frame by **Inc**, or  to decrement.
- Click on the timeline at the relevant frame.
- Type the frame number in the field marked **Cur**.
- Press **Ctrl+Right Arrow** to jump to the next keyframe, or **Ctrl+Left Arrow** to jump back to the previous.
- Click  to jump to the next keyframe, or  to jump back to the previous.

Panning the Frame Range

To pan the current frame range, you can:

- Drag the timeline with the middle mouse button, or
- drag the scrollbar directly under the time range.

Zooming the Frame Range

To zoom into/out of an area of the frame range, you can:

- **Ctrl**+drag to select an area of the frame range, then upon release of the mouse button the timeline zooms to that range.
- Scroll up with the mouse wheel over a frame to zoom in at that point, or scroll down to zoom out.
- Press the **+** key to zoom in, or the **-** key to zoom out.
- Click  to set the range from **inTime** to **outTime** in the **Globals** panel.
- Press the **Home** key to set the range from **inTime** to **outTime** in the **Globals** panel.
- Press the **F** key to set the range to fit all keyframes on the timeline.

Changing the Frame Range In and Out Points

To change the frame range in and out points, you can:

- Press the [key to set the in point to the current frame, or press the] key to set the out point.
- Type the in frame number into the **In** field on the timeline, or type the out frame number in the **Out** field.

APPENDIX A: HOTKEYS

Hotkeys

Keystroke shortcuts, or hotkeys, provide quick access to the features of Katana. The following tables show these keystrokes.

Conventions

The following conventions apply to instructions for mouse-clicks and key presses.

- When you see the word "drag" after a mouse button, this tells you to press and hold the mouse button while dragging the mouse pointer.
- Keystroke combinations with the **Ctrl**, **Alt**, and **Shift** keys tell you to press and hold the key and then type the specified letter.

  For example, "Press **Ctrl+S**" means hold down the **Ctrl** key, press **S**, and then release both keys.

Important

*Keystrokes in the tables appear in upper case, but you do not type them as upper case. If the **Shift+** modifier does not appear before the letter, press the letter key alone.*

General Hotkeys

Keystroke(s)	Action
Right Arrow	Increment the current frame by **Inc**. (**Inc** can be found on the **Timeline**.)
Left Arrow	Decrement the current frame by **Inc**. (**Inc** can be found on the **Timeline**.)
\	Rerender the last render.
Alt+middle-click and drag	Pans any scrollable area. (When used in the **Node Graph** it zooms in and out.)
Ctrl+Right Arrow	Move the current frame to the next keyframe.
Ctrl+Left Arrow	Move the current frame to the previous keyframe.
Ctrl+E	Export the currently selected portion of the script as highlighted in the **Node Graph**. This saves the selected nodes as a script.
Ctrl+I	Import a script into the current script.
Ctrl+N	Create a new script. (Doesn't work inside the **Node Graph**.)
Ctrl+O	Open a previously created script.
Ctrl+Q	Quit the application.
Ctrl+R	Redo the last undone action.
Ctrl+S	Save the current script.
Ctrl+Shift+S	Save the current script to a new file (**Save As**).
Ctrl+Z	Undo the last action.
Esc	Cancel the current render.
P	Start an interactive render from the current view node.
Spacebar	Maximizes the pane currently below the mouse pointer. If the pane is already maximized, **Spacebar** restores it to its previous size.

Node Graph

Keystroke(s)	Action
/	Pans the view to the node at the other end of the connection. (Only works when the mouse is hovering over one side of a connection.)
[	Moves the **Backdrop Note** the mouse is over to the back.
]	Moves the **Backdrop Note** the mouse is over to the front.
A	Frames the complete node tree within the Node Graph.
Alt+Any Arrow	Nudges the selected node or nodes a small distance in the direction of the arrow.

Keystroke(s)	Action
Alt+D	Toggles the menu **Edit > Dim Nodes Unconnected To Viewed Node** within the Node Graph. When selected, all nodes not currently contributing to the current Scene Graph are dimmed.
Alt+E	Opens the currently selected node's parameters tab within the **Parameters** panel.
Alt+G	Creates a **GroupStack** node with the currently selected node moved inside.
Alt+I	Toggles the ignore state of the currently selected node(s).
Alt+S	Toggles snapping nodes to grid while dragging within the Node Graph. When selected, moving nodes happens in steps that correspond to a grid.
Backtick (`)	Creates a connection between nodes. Press it first with the mouse over the starting node, and a second time over the node to connect to.
Ctrl+C	Copies the currently selected node or nodes to the buffer.
Ctrl+Down Arrow	Selects all nodes downstream of the currently selected node(s).
Ctrl+Up Arrow	Selects all nodes upstream of the currently selected node(s).
Ctrl+V	Pastes the buffer to the Node Graph.
Ctrl+X	Deletes the currently selected node or nodes from the Node Graph and copies them to the buffer.
Full stop (.)	Adds a **Dot** node to the Node Graph. A **Dot** is only created if the mouse is over a connection, you are connecting two nodes with one end connected, or a node is selected.
D	Toggles the disable state of the node currently under the mouse pointer.
Delete	Deletes the selected node from the Node Graph.
E	Opens the parameter tab of the node currently under the mouse pointer in the **Parameters** panel.
Esc	Cancels whatever operation you are in the middle of, such as connecting nodes.
F	Frames the currently selected node(s) within the Node Graph.
G	Creates a **Group** node with the currently selected nodes moved inside.
J	Displays the Jump-to menu which comprises all **Backdrop Notes** that have the bookmark flag set. Selecting one of the menu options frames its corresponding **Backdrop Note** within the Node Graph.
N	Displays the right-click node creation menu at the current mouse location. (Same behavior as RMB only it works when the mouse is over a node.)
Q	Toggles showing expression links within the Node Graph. When selected, nodes that derive their parameters via an expression that references another node have this relationship shown via a black dashed line.
R	Replaces the currently selected node with a node selected from the node creation menu, which is displayed when the key is pressed. Typing additional characters filters the displayed list.
Shift+~	Swaps inputs one and two on the node below the mouse pointer.
T	Displays the node type of the node below the mouse pointer.
Tab	Displays the node creation menu. Typing additional characters filters the displayed list.
U	Removes all nodes from within the currently selected **Group** node and deletes the **Group** node.

Keystroke(s)	Action
V	Makes the currently selected node the view node for the **Scene Graph**. After pressing this key the Scene Graph displays a snapshot of the scene at that point within the script.
X	Removes all connections to or from the selected node, extracting it from the node tree. Expression references to or from the node remain unchanged.

APPENDIX B: EXPRESSIONS

Expressions

Katana uses Python to evaluate its expressions. Code that would evaluate as a variable assignment within Python can be used as an expression within Katana. The following appendix lists functions and variables that are specific to the expression editor and also lists which modules are imported. This is not meant to be an introduction to Python but a quick reference for those wishing to leverage some of the expression specific functions and variables.

Variables Within Expressions

Variable	Description
frame	The current frame. For example: frame - 1
globals	The project settings as shown within the **Project Settings** tab exposed as object parameter references. For example: globals.inTime
katanaVersion	The current Katana version - complete with build number. For instance **1.1.3**
katanaBranch	The current Katana version - major and minor release numbers only. For instance **1.1**

Katana Expression Functions

Function	Description
getRenderLocation(<nodeObject>, <renderPass>)	Returns the file asset path created by the given node object <nodeObject> for the render pass <renderPass>. Example: `getRenderLocation(getNode('shadow_key'), 'primary')`
getNode(<nodeName>)	Returns the node object with the given name <nodeName>. Example: `getNode('BakeCameraCreate').fov`
getParam(<nodeName.param>)	Returns the parameter object representing the node graph parameter referenced by its node name <nodeName> and parameter path <param>. Example: `getParam("mat_blinn.shaders.surfaceParams.opacity.value")`
scenegraphLocationFromNode(<nodeObject>)	Returns the scene graph location created by the given node object <nodeObject>. Example: `scenegraphLocationFromNode(getNode('mat_katana_blinn'))`
fcurve(<fileName>, <curveName>, <frameNum>)	Returns the value stored within the FCurve file <fileName> from the curve <curveName> for the given frame <frameNum>. Example: `fcurve("/tmp/fcurve.xml", "lgt_spot.shaders.lightParams.intensity.value", frame)` The FCurve file should be an XML file such as those generated by the menu option **Export FCurve...** which is obtained by right-clicking on a float parameter within the **Parameters** tab.
getenv(<envVarName>, <defaultValue>)	Returns the value of the environment variable <envVarName> or if not found the default <defaultValue>. Example: `getenv("HOME", "/tmp")`

Function	Description
getres(<resolutionName>)	Returns a tuple of the resolution named by <resolutionName> with the first index being the width and the second being the height. Example: `int(getres(globals.resolution)[0]) if int(getres(globals.resolution)[0]) > 1024 else 1024` (use the width of the resolution if it is greater than 1024 otherwise use 1024)
getExrAttr(<fileName>, <attrHeader>, <frame>, <default = None>)	Returns the string value of the attribute <attrHeader> within the file <fileName> for the given frame <frame> or the default if no attribute is found. Example: `getExrAttr('/tmp/white_ncf.exr', 'spi:package', frame)`

Note

Both **getNode()** and **getParam()** update automatically based on node name changes. When <nodeName> is changed within the **Node graph** the change is reflected within the function call. For example:

If the node named MainCameraCreate within the **Node graph** were changed to BakeCamera any expressions which have the line:

```
getNode('MainCameraCreate')
```

would become:

```
getNode('BakeCamera'
```

Python Modules Within Expressions

Module	Scope	Brief Description	Module Contents

re	global	Regular expression support.	**Functions:** compile, escape, findall, finditer, match, purge, search, split, sub, subn, template
			Example:
			`re.sub(r'x', ' by ', '2048x2048')`
			For further help:
			`help(re)`

math	expression	Standard mathematical functions and variables	**Functions:** acos, acosh, asin, asinh, atan, atan2, atanh, ceil, copysign, cos, cosh, degrees, exp, fabs, factorial, floor, fmod, frexp, fsum, hypot, isinf, isnan, ldexp, log, log10, log1p, modf, pow, radians, sin, sinh, sqrt, tan, tanh, trunc
			Variables: e, pi
			Example:
			`cos(pi/3)`
			For further help:
			`help(math)`

array	global	Efficient class based array handling	**Methods:** append, buffer_info, byteswap, count, extend, fromfile, fromlist, fromstring, fromunicode, index, insert, pop, read, remove, reverse, tofile, tolist, tostring, tounicode, write
			Example:
			`array.array('u', "efficient").buffer_info()[1]`
			For further help:
			`import array`
			`help(array)`

os.path	as path	Common functions for manipulating pathnames	**Functions:** abspath, basename, commonprefix, dirname, exists, expanduser, expandvars, getatime, getctime, getmtime, getsize, isabs, isdir, isfile, islink, ismount, join, lexists, normcase, normpath, realpath, relpath, samefile, sameopenfile samestat, split, splitdrive, splitext, walk
			Example:
			`path.exists(getenv("HOME", "")+"/shaders")`
			For further help:
			`help(os.path)`

Module	Scope	Brief Description	Module Contents

ExpressionMath	expression	Python interface to SPI ExpressionMath library	**Functions:** cfit, clamp, fit, hsvtorgb, ifelse, isfinite, isinf, isnan, lerp, matmultvec, noise, randval, retime, rgbtohsv, smoothstep, snoise, softcfit, stablehash
			Example:
			`randval(0, 1, frame)`
			For further help:
			`help(ExpressionMath)`

To access the help for the modules type the help examples within the **Python** tab.

APPENDIX C: COLLECTION EXPRESSION LANGUAGE & COLLECTIONS

Collection Expression Language (CEL)

CEL is a grammar for specifying a subset of the locations within the Scene Graph. The locations can be selected based on their path (including wildcards), attributes, or other collections. Basic set notation is included within the grammar.

Basic CEL Syntax

CEL	Description
/root/world/cam_main	Explicitly selects the location.
/root/world/lgt*	Select all immediate children of /root/world whose name starts with lgt. Use * as a wildcard.
/root/materials//*	Selects all locations recursively beneath /root/materials. Use // to represent recursively.
//shape	Selects all locations named shape anywhere within the **Scene Graph**. Use // at the start of the line to represent anywhere within the **Scene Graph.**

Value Expressions

CEL	Description
/root/world/*{ attr("type") == "camera" } **OR** /root/world/*{@type == "camera"}	Selects all immediate children of /root/world whose **type** attribute has a value of **camera**. Use attr("<attribute>") or @<attribute> to get the value of a local attribute.
/root/world//*{ hasattr("textures.ColMap") }	Selects all locations recursively beneath /root/world which have a local attribute named **textures.ColMap**. Use hasattr("<attribute>") to check if an attribute exists on a location.
/root/world//*{ globalattr("material.surfaceParams.Ks") > 0.0 }	Selects all locations recursively beneath /root/world which have or inherit an attribute named **material.surfaceParams.Ks** with a value above **0.0**. Use globalattr("<attribute>") to get the value of an attribute which is either locally assigned or inherited.
//*{@materialAssign ~= "^ambient"}	Selects all locations recursively that have the **materialAssign** attribute beginning with ambient. Use ~= for regular expression syntax.

CEL Sets

CEL	Description
/root/world/lgt_key + /root/world/lgt_fill **OR** /root/world/lgt_key /root/world/lgt_fill	The **lgt_key** and **lgt_fill** locations are combined (this is the default if no operator is specified). To get the **union** of two sets, use the + operator (or no operator).
/root/world/lgt* - /root/world/lgt_rim	Selects all of the immediate children of /root/world that start with **lgt** except **lgt_rim**. To get the **difference** of two sets, use the - operator.
/root/world/*a* ^ /root/world/*b*	Selects all of the immediate children of /root/world that contain both an **a** and a **b**. To get the **intersection** of two sets, use the ^ operator.

Collections

Collections are used to store a CEL statement. Collections can also be used to make the CEL expressions local to a branch of the Scene Graph (or kept global under /root). They are stored as attributes at the location defined by the **location** parameter in the **CollectionCreate** node. As they are simply attributes within the Scene Graph, Collections can be included within Katana Look Files.

Collection Syntax

CEL	Description
/$my_collection/* **OR** FLATTEN(/$my_collection)	Select the contents of the Collection in /root called my_collection. Use $<collection_name> or FLATTEN(<collection_name>) to use the contents of the CEL statement.
/primitive (within a local collection)	For a Collection with location /root/world/geo, /primitive resolves to /root/world/geo/primitive. Use / to represent the root directory of the collection. The exception being a global collection where the full location path is needed.

APPENDIX D: THIRD PARTY LICENSES

Third Party Licenses

This appendix lists third party libraries used in Katana, along with their licenses.

Library	Description	License
Alembic	Geometry format library	TM & © 2010-2011 Lucasfilm Entertainment Company Ltd. or Lucasfilm Ltd. All rights reserved. Industrial Light & Magic, ILM and the Bulb and Gear design logo are all registered trademarks or service marks of Lucasfilm Ltd. © 2010-2011 Sony Pictures Imageworks Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of Industrial Light & Magic nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Library	Description	License
Boost	Source code function / template library	Boost Software License - Version 1.0 - August 17th, 2003 Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following: The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
cgkit	Python Computer Graphics Kit	Copyright (C) 2004 Matthias Baas (baas@ira.uka.de) This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
Expat	C XML Parser library	Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
FreeType	Font support	Portions of this software are copyright © 2008 The FreeType Project (www.freetype.org). All rights reserved.

Library	Description	License
FTGl	OpenGL font rendering library	Herewith is a license. Basically I want you to use this software and if you think this license is preventing you from doing so let me know. Copyright (C) 2001-3 Henry Maddocks Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
GLEW	OpenGL support	The OpenGL Extension Wrangler Library Copyright (C) 2002-2008, Milan Ikits <milan.ikits@ieee.org> Copyright (C) 2002-2008, Marcelo E. Magallon <mmagallo@debian.org> Copyright (C) 2002, Lev Povalahev All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: • Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • The name of the author may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Library	Description	License
GraphViz		Eclipse Public License - v 1.0 THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT. 1. DEFINITIONS "Contribution" means: a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and b) in the case of each subsequent Contributor: i) changes to the Program, and ii) additions to the Program; where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program. "Contributor" means any person or entity that distributes the Program. "Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program. "Program" means the Contributions distributed in accordance with this Agreement. "Recipient" means anyone who receives the Program under this Agreement, including all Contributors. 2. GRANT OF RIGHTS a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form. b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder. c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

Library	Description	License
GraphViz	(continued)	d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement. 3. REQUIREMENTS A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that: a) it complies with the terms and conditions of this Agreement; and b) its license agreement: i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose; ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits; iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange. When the Program is made available in source code form: a) it must be made available under this Agreement; and b) a copy of this Agreement must be included with each copy of the Program. Contributors may not remove or alter any copyright notices contained within the Program. Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution. 4. COMMERCIAL DISTRIBUTION Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

Library	Description	License
GraphViz	(continued)	For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages. 5. NO WARRANTY EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations. 6. DISCLAIMER OF LIABILITY EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 7. GENERAL If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable. If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed. All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Library	Description	License
GraphViz	(continued)	Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Library	Description	License
HDF5	Hierarchical Data Format Library	Copyright Notice and License Terms for HDF5 (Hierarchical Data Format 5) Software Library and Utilities ---------- HDF5 (Hierarchical Data Format 5) Software Library and Utilities Copyright 2006-2010 by The HDF Group. NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities Copyright 1998-2006 by the Board of Trustees of the University of Illinois. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution. 3. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change. 4. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by The HDF Group and by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and credit the contributors. 5. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively. DISCLAIMER: THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no event shall The HDF Group or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

Library	Description	License
HDF5	(continued)	Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois, Fortner Software, Unidata Program Center (netCDF), The Independent JPEG Group (JPEG), Jean-loup Gailly and Mark Adler (gzip), and Digital Equipment Corporation (DEC). --- Portions of HDF5 were developed with support from the Lawrence Berkeley National Laboratory (LBNL) and the United States Department of Energy under Prime Contract No. DE-AC02-05CH11231. --- Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials: This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL. DISCLAIMER: This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Library	Description	License
log4cplus	C++ Logging library	Two clause BSD license: Copyright (C) 1999-2009 Contributors to log4cplus project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Library	Description	License
NVIDIA Cg Library	High-level shading language	Copyright (c) 2002, NVIDIA Corporation. NVIDIA Corporation("NVIDIA") supplies this software to you in consideration of your agreement to the following terms, and your use, installation, modification or redistribution of this NVIDIA software constitutes acceptance of these terms. If you do not agree with these terms, please do not use, install, modify or redistribute this NVIDIA software. In consideration of your agreement to abide by the following terms, and subject to these terms, NVIDIA grants you a personal, non-exclusive license, under NVIDIA's copyrights in this original NVIDIA software (the "NVIDIA Software"), to use, reproduce, modify and redistribute the NVIDIA Software, with or without modifications, in source and/or binary forms; provided that if you redistribute the NVIDIA Software, you must retain the copyright notice of NVIDIA, this notice and the following text and disclaimers in all such redistributions of the NVIDIA Software. Neither the name, trademarks, service marks nor logos of NVIDIA Corporation may be used to endorse or promote products derived from the NVIDIA Software without specific prior written permission from NVIDIA. Except as expressly stated in this notice, no other rights or licenses express or implied, are granted by NVIDIA herein, including but not limited to any patent rights that may be infringed by your derivative works or by other works in which the NVIDIA Software may be incorporated. No hardware is licensed hereunder. THE NVIDIA SOFTWARE IS BEING PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR ITS USE AND OPERATION EITHER ALONE OR IN COMBINATION WITH OTHER PRODUCTS. IN NO EVENT SHALL NVIDIA BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, EXEMPLARY, CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, LOST PROFITS; PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) OR ARISING IN ANY WAY OUT OF THE USE, REPRODUCTION, MODIFICATION AND/OR DISTRIBUTION OF THE NVIDIA SOFTWARE, HOWEVER CAUSED AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Library	Description	License
OpenColorIO	Color management library	Copyright (c) 2003-2010 Sony Pictures Imageworks Inc., et al. All Rights Reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: • Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • Neither the name of Sony Pictures Imageworks nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
OpenEXR	Image file format library	Copyright (c) 2002, Industrial Light & Magic, a division of Lucas Digital Ltd. LLC All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of Industrial Light & Magic nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Library	Description	License
OpenImageIO	Library for reading and writing images	Copyright 2008 Larry Gritz and the other authors and contributors. All Rights Reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of the software's owners nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (This is the Modified BSD License)
OpenSceneGraph	3D graphics toolkit	Copyright (C) 2002 Robert Osfield. Everyone is permitted to copy and distribute verbatim copies of this licence document, but changing it is not allowed. OPENSCENEGRAPH PUBLIC LICENCE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION This library is free software; you can redistribute it and/or modify it under the terms of the OpenSceneGraph Public License (OSGPL) version 0.0 or later. Notes: the OSGPL is based on the LGPL, with the 4 exceptions laid out in the wxWindows section below. The LGPL is contained in the final section of this license.

Library	Description	License
OpenSSL	Toolkit that implements SSL	Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)" 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. 5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. 6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)" THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Library	Description	License
OpenSSL	(continued)	Original SSLeay License Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL. This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com). Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-). 4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)" THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The licence and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Library	Description	License
PyGraphviz		Copyright (C) 2004-2010 by Aric Hagberg <hagberg@lanl.gov> Dan Schult <dschult@colgate.edu> Manos Renieris, http://www.cs.brown.edu/~er/ Distributed with BSD license. All rights reserved, see LICENSE for details. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: • Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Thread Building Blocks	Multi-threading / paralellization library	Copyright 2005-2009 Intel Corporation. All Rights Reserved. Threading Building Blocks is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation. Threading Building Blocks is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with Threading Building Blocks; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA As a special exception, you may use this file as part of a free software library without restriction. Specifically, if other files instantiate templates or use macros or in-line functions from this file, or you compile this file and link it with other files to produce an executable, this file does not by itself cause the resulting executable to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the executable file might be covered by the GNU General Public License.

Library	Description	License
Zlib	Compression library	General purpose compression library version 1.2.2, October 3rd, 2004 Copyright (C) 1995-2004 Jean-loup Gailly and Mark Adler This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions: 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 3. This notice may not be removed or altered from any source distribution. Jean-loup Gailly jloup@gzip.org Mark Adler madler@alumni.caltech.edu

APPENDIX E: END USER LICENCING AGREEMENT

End User Licensing Agreement (EULA)

IMPORTANT: BY INSTALLING THIS SOFTWARE YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT DO NOT INSTALL, COPY OR USE THE SOFTWARE.

This END USER LICENSE AGREEMENT (this "Agreement") is made by and between The Foundry Visionmongers Ltd., a company registered in England and Wales, ("The Foundry"), and you, as either an individual or a single entity ("Licensee").

In consideration of the mutual covenants contained herein and for other good and valuable consideration (the receipt and sufficiency of which is acknowledged by each party hereto) the parties agree as follows:

SECTION 1. GRANT OF LICENSE.

Subject to the limitations of Section 2, The Foundry hereby grants to Licensee a limited, non-transferable and non-exclusive license to install and use a machine readable, object code version of this software program (the "Software") and accompanying user guide and other documentation (collectively, the "Documentation") solely for Licensee's own internal business purposes (collectively, the "License"); provided, however, Licensee's right to install and use the Software and the Documentation is limited to those rights expressly set out in this Agreement.

SECTION 2. RESTRICTIONS ON USE.

Licensee is authorized to use the Software in machine readable, object code form only, and Licensee shall not: (a) assign, sublicense, sell, distribute, transfer, pledge, lease, rent, share or export the Software, the Documentation or Licensee's rights hereunder; (b) alter or circumvent the copy protection mechanisms in the Software or reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software; (c) modify, adapt, translate or create derivative works based on the Software or Documentation; (d) use, or allow the use of, the Software or Documentation on any project other than a project produced by Licensee (an "Authorized Project"); (e) allow or permit anyone (other than Licensee and Licensee's authorized employees to the extent they are working on an Authorized Project) to use or have access to the Software or Documentation; (f) copy or install the Software or Documentation other than as expressly provided for herein; or (g) take any action, or fail to take action, that could adversely affect the trademarks, service marks, patents, trade secrets, copyrights or other intellectual property rights of The Foundry or any third party with intellectual property rights in the Software (each, a "Third Party Licensor"). Furthermore, for purposes of this Section 2, the term "Software" shall include any derivatives of the Software.

Licensee shall install and use only a single copy of the Software on one computer, unless the Software is installed in a "floating license" environment, in which case Licensee may install the Software on more than

one computer; provided, however, Licensee shall not at any one time use more copies of the Software than the total number of valid Software licenses purchased by Licensee.

Please note that in order to guard against unlicensed use of the Software a licence key is required to access and enable the Software. The issuing of replacement or substituted licence keys if the Software is moved from one computer to another is subject to and strictly in accordance with The Foundry's Licence Transfer Policy, which is available on The Foundry's website and which requires a fee to be paid in certain circumstances. The Foundry may from time to time and at its sole discretion vary the terms and conditions of the Licence Transfer Policy.

Furthermore, if the Software can be licensed on an "interactive" or "non-interactive" basis, licensee shall be authorized to use a non-interactive version of the Software for rendering purposes only (i.e., on a CPU, without a user, in a non-interactive capacity) and shall not use such Software on workstations or otherwise in a user-interactive capacity. Licensee shall be authorized to use an interactive version of the Software for both interactive and non-interactive rendering purposes, if available.

If Licensee has purchased the Software on the discount terms offered by The Foundry's Educational Policy published on its website ("the Educational Policy"), Licensee warrants and represents to The Foundry as a condition of this Agreement that: (a) (if Licensee is an individual) he or she is a part-time or full-time student at the time of purchase and will not use the Software for commercial, professional or for-profit purposes; (b) (if the Licensee is not an individual) it is an organisation that will use it only for the purpose of training and instruction, and for no other purpose (c) Licensee will at all times comply with the Educational Policy (as such policy may be amended from time to time).

Finally, if the Software is a "Personal Learning Edition," ("PLE") Licensee may use it only for the purpose of personal or internal training and instruction, and for no other purpose. PLE versions of the Software may not be used for commercial, professional or for-profit purposes including, for the avoidance of doubt, the purpose of providing training or instruction to third parties.

SECTION 3. SOURCE CODE.

Notwithstanding that Section 1 defines "Software" as an object code version and that Section 2 provides that Licensee may use the Software in object code form only, The Foundry may also agree to license to Licensee (including by way of upgrades, updates or enhancements) source code or elements of the source code of the Software the intellectual property rights in which belong either to The Foundry or to a Third Party Licensor ("Source Code"). If The Foundry does so Licensee shall be licensed to use the Source Code as Software on the terms of this Agreement and: (a) notwithstanding Section 2 (c) Licensee may use the Source Code at its own risk in any reasonable way for the limited purpose of enhancing its use of the Software solely for its own internal business purposes and in all respects in accordance with this Agreement; (b) Licensee shall in respect of the Source Code comply strictly with all other restrictions applying to its use of the Software under this Agreement as well as any other restriction or instruction that is communicated to it by The Foundry at any time during this Agreement (whether imposed or requested by The Foundry or by any Third Party Licensor); (c) notwithstanding any other term of this Agreement The Foundry gives no warranty whatsoever in respect of the Source Code, which is licensed on an "as is" basis, or in respect of any modification of the Source Code made by Licensee ("Modification"); (d)

notwithstanding any other term of this Agreement The Foundry shall have no obligation to provide support, maintenance, upgrades or updates of or in respect of the Source Code or of any Modification; and (e) Licensee shall indemnify The Foundry against all liabilities and expenses (including reasonable legal costs) incurred by The Foundry in relation to any claim asserting that any Modification infringes the intellectual property rights of any third party.

SECTION 4. BACK-UP COPY.

Notwithstanding Section 2, Licensee may store one copy of the Software and Documentation off-line and off-site in a secured location owned or leased by Licensee in order to provide a back-up in the event of destruction by fire, flood, acts of war, acts of nature, vandalism or other incident. In no event may Licensee use the back-up copy of the Software or Documentation to circumvent the usage or other limitations set forth in this Agreement.

SECTION 5. OWNERSHIP.

Licensee acknowledges that the Software (including, for the avoidance of doubt, any Source Code that is licensed to Licensee) and Documentation and all intellectual property rights and other proprietary rights relating thereto are and shall remain the sole property of The Foundry and the Third Party Licensors. Licensee shall not remove, or allow the removal of, any copyright or other proprietary rights notice included in and on the Software or Documentation or take any other action that could adversely affect the property rights of The Foundry or any Third Party Licensor. To the extent that Licensee is authorized to make copies of the Software or Documentation under this Agreement, Licensee shall reproduce in and on all such copies any copyright and/or other proprietary rights notices provided in and on the materials supplied by The Foundry hereunder. Nothing in this Agreement shall be deemed to give Licensee any rights in the trademarks, service marks, patents, trade secrets, confidential information, copyrights or other intellectual property rights of The Foundry or any Third Party Licensor, and Licensee shall be strictly prohibited from using the name, trademarks or service marks of The Foundry or any Third Party Licensor in Licensee's promotion or publicity without The Foundry's express written approval.

SECTION 6. LICENSE FEE.

Licensee understands that the benefits granted to Licensee hereunder are contingent upon Licensee's payment in full of the license fee payable in connection herewith (the "License Fee").

SECTION 7. UPGRADES/ENHANCEMENTS.

The Licensee's access to support, upgrades and updates is subject to the terms and conditions of the "Annual Upgrade and Support Programme" available on The Foundry's website. The Foundry may from time to time and at its sole discretion vary the terms and conditions of the Annual Upgrade and Support Programme.

SECTION 8. TAXES AND DUTIES.

Licensee agrees to pay, and indemnify The Foundry from claims for, any local, state or national tax (exclusive of taxes based on net income), duty, tariff or other impost related to or arising from the transaction contemplated by this Agreement.

SECTION 9. LIMITED WARRANTY.

The Foundry warrants that, for a period of ninety (90) days after delivery of the Software: (a) the machine readable electronic files constituting the Software and Documentation shall be free from errors that may arise from the electronic file transfer from The Foundry and/or its authorized reseller to Licensee; and (b) to the best of The Foundry's knowledge, Licensee's use of the Software in accordance with the Documentation will not, in and of itself, infringe any third party's copyright, patent or other intellectual property rights. Except as warranted, the Software and Documentation is being provided "as is." THE FOREGOING LIMITED WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES OR CONDITIONS, EXPRESS OR IMPLIED, AND The Foundry DISCLAIMS ANY AND ALL IMPLIED WARRANTIES OR CONDITIONS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, REGARDLESS OF WHETHER The Foundry KNOWS OR HAS REASON TO KNOW OF LICENSEE'S PARTICULAR NEEDS. The Foundry does not warrant that the Software or Documentation will meet Licensee's requirements or that Licensee's use of the Software will be uninterrupted or error free. No employee or agent of The Foundry is authorized to modify this limited warranty, nor to make additional warranties. No action for any breach of the above limited warranty may be commenced more than one (1) year after Licensee's initial receipt of the Software. To the extent any implied warranties may not be disclaimed under applicable law, then ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO NINETY (90) DAYS AFTER DELIVERY OF THE SOFTWARE TO LICENSEE.

SECTION 10. LIMITED REMEDY.

The exclusive remedy available to the Licensee in the event of a breach of the foregoing limited warranty, TO THE EXCLUSION OF ALL OTHER REMEDIES, is for Licensee to destroy all copies of the Software, send The Foundry a written certification of such destruction and, upon The Foundry's receipt of such certification, The Foundry will make a replacement copy of the Software available to Licensee.

SECTION 11. INDEMNIFICATION.

Licensee agrees to indemnify, hold harmless and defend The Foundry, the Third Party Licensors and The Foundry's and each Third Party Licensor's respective affiliates, officers, directors, shareholders, employees, authorized resellers, agents and other representatives (collectively, the "Released Parties") from all claims, defense costs (including, but not limited to, attorneys' fees), judgments, settlements and other expenses arising from or connected with the operation of Licensee's business or Licensee's possession or use of the Software or Documentation.

SECTION 12. LIMITED LIABILITY.

In no event shall the Released Parties' cumulative liability to Licensee or any other party for any loss or damages resulting from any claims, demands or actions arising out of or relating to this Agreement (or the Software or Documentation contemplated herein) exceed the License Fee paid to The Foundry or its authorized reseller for use of the Software. Furthermore, IN NO EVENT SHALL THE RELEASED PARTIES BE LIABLE TO LICENSEE UNDER ANY THEORY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, EXEMPLARY OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS OR LOSS OF PROFITS) OR THE COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, REGARDLESS OF WHETHER THE RELEASED PARTIES KNOW OR HAVE REASON TO KNOW OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL

PURPOSE. No action arising out of or related to this Agreement, regardless of form, may be brought by Licensee more than one (1) year after Licensee's initial receipt of the Software; provided, however, to the extent such one (1) year limit may not be valid under applicable law, then such period shall be limited to the shortest period allowed by law.

SECTION 13. TERM; TERMINATION.

This Agreement is effective upon Licensee's acceptance of the terms hereof and Licensee's payment of the License Fee, and the Agreement will remain in effect until termination. If Licensee breaches this Agreement, The Foundry may terminate the License granted hereunder by notice to Licensee. In the event the License is terminated, Licensee will either return to The Foundry all copies of the Software and Documentation in Licensee's possession or, if The Foundry directs in writing, destroy all such copies. In the later case, if requested by The Foundry, Licensee shall provide The Foundry with a certificate signed by an officer of Licensee confirming that the foregoing destruction has been completed.

SECTION 14. CONFIDENTIALITY.

Licensee agrees that the Software (including, for the avoidance of doubt, any Source Code that is licensed to Licensee) and Documentation are proprietary and confidential information of The Foundry or, as the case may be, the Third Party Licensors, and that all such information and any communications relating thereto (collectively, "Confidential Information") are confidential and a fundamental and important trade secret of The Foundry or the Third Party Licensors. Licensee shall disclose Confidential Information only to Licensee's employees who are working on an Authorized Project and have a "need-to-know" of such Confidential Information, and shall advise any recipients of Confidential Information that it is to be used only as authorized in this Agreement. Licensee shall not disclose Confidential Information or otherwise make any Confidential Information available to any other of the Licensee's employees or to any third parties without the express written consent of The Foundry. Licensee agrees to segregate, to the extent it can be reasonably done, the Confidential Information from the confidential information and materials of others in order to prevent commingling. Licensee shall take reasonable security measures, which such measures shall be at least as great as the measures Licensee uses to keep Licensee's own confidential information secure (but in any case using no less than a reasonable degree of care), to hold the Software, Documentation and any other Confidential Information in strict confidence and safe custody. The Foundry may request, in which case Licensee agrees to comply with, certain reasonable security measures as part of the use of the Software and Documentation. Licensee acknowledges that monetary damages may not be a sufficient remedy for unauthorized disclosure of Confidential Information, and that The Foundry shall be entitled, without waiving any other rights or remedies, to such injunctive or equitable relief as may be deemed proper by a court of competent jurisdiction.

SECTION 15. INSPECTION.

Licensee shall advise The Foundry on demand of all locations where the Software or Documentation is used or stored. Licensee shall permit The Foundry or its authorized agents to inspect all such locations during normal business hours and on reasonable advance notice.

SECTION 16. NONSOLICITATION.

Licensee agrees not to solicit for employment or retention any of The Foundry's current or future employees who were or are involved in the development and/or creation of the Software.

SECTION 17. U.S. GOVERNMENT LICENSE RIGHTS.

The Software, Documentation and/or data delivered hereunder are subject to the terms of this Agreement and in no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication or disclosure by the U.S. Government is subject to the applicable restrictions of: (i) FAR §52.227-14 ALTS I, II and III (June 1987); (ii) FAR §52.227-19 (June 1987); (iii) FAR §12.211 and 12.212; and/or (iv) DFARS §227.7202-1(a) and DFARS §227.7202-3.

The Software is the subject of the following notices:

- Copyright (c) 1/6/12 The Foundry Visionmongers, Ltd.. All Rights Reserved.
- Unpublished-rights reserved under the Copyright Laws of the United Kingdom.

SECTION 18. SURVIVAL.

Sections 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 and 20 shall survive any termination or expiration of this Agreement.

SECTION 19. IMPORT/EXPORT CONTROLS.

To the extent that any Software made available hereunder is subject to restrictions upon export and/or reexport from the United States, Licensee agrees to comply with, and not act or fail to act in any way that would violate, the applicable international, national, state, regional and local laws and regulations, including, without limitation, the United States Foreign Corrupt Practices Act, the Export Administration Act and the Export Administration Regulations, as amended or otherwise modified from time to time, and neither The Foundry nor Licensee shall be required under this Agreement to act or fail to act in any way which it believes in good faith will violate any such laws or regulations.

SECTION 20. MISCELLANEOUS.

This Agreement is the exclusive agreement between the parties concerning the subject matter hereof and supersedes any and all prior oral or written agreements, negotiations, or other dealings between the parties concerning such subject. This Agreement may be modified only by a written instrument signed by both parties. If any action is brought by either party to this Agreement against the other party regarding the subject matter hereof, the prevailing party shall be entitled to recover, in addition to any other relief granted, reasonable attorneys' fees and expenses of litigation. Should any term of this Agreement be declared void or unenforceable by any court of competent jurisdiction, such declaration shall have no effect on the remaining terms of this Agreement. The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement shall be governed by, and construed in accordance with English Law.

The Foundry and Licensee intend that each Third Party Licensor may enforce against Licensee under the

Contracts (Rights of Third Parties) Act 1999 ("the Act") any obligation owed by Licensee to The Foundry under this Agreement that is capable of application to any proprietary or other right of that Third Party Licensor in or in relation to the Software. The Foundry and Licensee reserve the right under section 2(3)(a) of the Act to rescind, terminate or vary this Agreement without the consent of any Third Party Licensor.