SOFTIMAGE®

SOFTIMAGE®|XSI™

Version 1.0

Shaders, Lights & Cameras

Avid

Shaders, Lights & Cameras was written by Judy Bayne and André Demers, edited by Edna Kruger and John Woolfrey, and formatted by Luc Langevin.

Printed in Canada.

Document No. 0130-04613-01 0400

Contents

Roadmap

About This Guide

Shaders, Lights & Cameras contains information and step-by-step procedures on applying materials, textures, and special effects using shaders. It also provides information on creating caustics, global illumination, and motion blur effects.

Chapter 1: Getting the Look You Want—how placing and lighting your objects affects their final look as much as shaders do. Also describes the render region, which lets you render a scene or object interactively as you edit it.

Chapter 2: Shader Basics—the different shaders available: from surface and texture shaders, to volumic and tool shaders.

Chapter 3: Material & Texture Basics—how surface and texture shaders work together to create a texture for your object. Also, how you can use texture projections and blending tools to fine-tune results.

Chapter 4: Advanced Materials & Textures—how you can use the surface and texture shaders to create bump, transparency, reflection, and displacement maps.

Chapter 5: Working with Lights—how to create, manipulate, and edit light properties.

Chapter 6: Global Illumination & Caustics—how to define, render, and fine-tune global illumination and caustic effects.

Chapter 7: Camera Basics—how to create one or several cameras and individually edit their parameters to define how they "see" your scene.

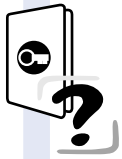
Chapter 8: Blurs, Flares & Other Effects—how to create, apply, and render motion blurs, lens flares, volumic lights, and glows. Also, how to use environment and volume shaders with your scene.

Chapter 9: The Render Tree—the concept and basics of the render tree, which is used to extend or create shaders. Also provides several examples of how the render tree can be used.

Appendix: Shader Descriptions—a brief description of every shader.

Where to Find Information

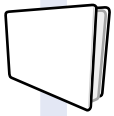
The SOFTIMAGE|XSI package includes a comprehensive set of learning materials. Use this Roadmap to find the information you need to get up and running quickly and effectively.



Start with the ***Setup Guide*** to install and license all components.
Setup Online Help is also available as you go through the process.
We recommend you choose Custom install so that you can perform the tutorials.

Refer to ***Release Notes***, an online listing of known problems and limitations for this version. Also includes workarounds and supplemental information. Access through the web at **www.softimage.com > support**.

Follow the ***Guided Tour*** (available from the Online Library CD).
This is a set of videoclips that provide overviews of features and tools.

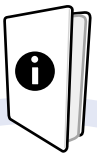


Work through ***Tutorials*** to learn the features in the context of basic productions.
This is a full-color set of lessons showing you step-by-step how to perform typical tasks.
You can install the scenes from the Software CD. (Choose Custom install when installing SOFTIMAGE|XSI). Then choose the **Content** option to install the Tutorials project.

The Softimage Discussion Group

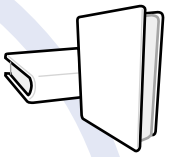
You can join the worldwide network of Softimage users exchanging ideas and techniques by e-mail. To find out more, e-mail **majordomo@softimage.com.** Leave the Subject line empty and type the word "help" in the body of your mail message.

The ***Global Index & Glossary*** is an index to all user guides and *Tutorials*; a glossary of terms; and a list of books, videos, and web sites related to the 3D animation industry.



The **user guides** contain conceptual information and procedures on how to use specific tools. These comprise**:**

- Fundamentals
- Animating
- Modeling & Deformations
- Shaders, Lights & Cameras
- Rendering



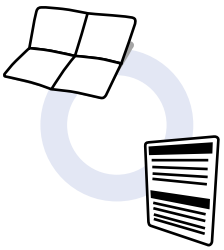
The Online Library CD
The Online Library contains the Guided Tour and all the SOFTIMAGE|XSI and some mental ray documentation in electronic form in both PDF and HTML formats. (See next page for how to use.)

Online Help
On-screen reference information on interface elements, commands, and parameters. There are two ways to access it:

- Click the **?** button in any property editor or tool view.
- Choose **Help > Contents and Index** from the main-menu bar.

Using SOFTIMAGE|3D with SOFTIMAGE|XSI provides tips and techniques about using the two software packages. Available from the Online Library CD and **softimage.com > support** only)

HTML Scripting Reference
An HTML-based reference help on the syntax for all scripting commands and arguments. It appears in your default HTML browser. Click on the icon (above) to open the script editor, then click **Help > Scripting Reference** or press **F1**.



Pin up the ***SOFTIMAGE|XSI Interface Layout*** and the ***Quick Reference Card*** to help you become familiar with the interface and keyboard shortcuts.

Using the Online Library

The Online Library contains the *Guided Tour* and all the SOFTIMAGE|XSI and some mental ray documentation in electronic form in both PDF and HTML formats.

For full-text searching and printing, we recommend PDF format. If you do not have Acrobat Reader installed, you can install it it free of charge from the Online Library CD: Follow the instructions in the readme file on the CD.

To access the Online Library

1. Insert the Online Library CD in your disk drive.
2. Open one of the following documents:
   - **mainmenu.pdf** (PDF format)
   - **mainmenu.htm** (HTML format)

Document Conventions

The following are ways that information is displayed in the SOFTIMAGE|XSI documentation.

Typography Conventions

Type style	Usage
Bold	Menu commands, dialog-box and property-editor options, and file and directory names.
Italics	Definitions and emphasized words.
`Courier`	Text that you must type exactly as it appears. For example, if you are asked to type `mkdir style`, you would type these characters and the spacing between words exactly as they are appear in this book.
>	The arrow (>) indicates menu commands (and subcommands) in the order that you choose them: *Menu name* > *Command name*. For example, when you see **File** > **Open**, it means to open the **File** menu and then choose the **Open** command.

Visual Identifiers

These icons help identify certain types of information:

Notes are used for information that is an aside to the text. Notes are reminders or contain important information.

Tips are useful tidbits of information, workarounds, and shortcuts that you might find helpful in a particular situation.

The 3D icon indicates information about differences in workflow or concepts between SOFTIMAGE|3D and SOFTIMAGE|XSI. You will find these very helpful when working with the two products.

Warnings are used when you can lose or damage information, such as deleting data or not being able to easily undo an action. Warnings always appear *before* you are about to do such a task!

Keyboard and Mouse Conventions

SOFTIMAGE|XSI uses a three-button mouse for most operations. These are referred to as the *left*, *middle*, and *right* mouse buttons. In many cases, you will use the different buttons to perform different operations; always use the left mouse button unless otherwise stated.

The two-button mouse is not supported in SOFTIMAGE|XSI.

This table shows the terms relating to the mouse and keyboard.

When this term is used...	**...it means this**
Click	Quickly press and release the left mouse button. Always use the left mouse button unless otherwise stated.
Middle-click	Quickly press and release the middle mouse button of a three-button mouse.
Right-click	Quickly press and release the right mouse button.
Double-click	Quickly click the left mouse button twice.
Shift+click, Ctrl+click, Alt+click	Hold down the Shift, Ctrl, or Alt key as you click a mouse button.
Drag	Hold down the left mouse button as you move the mouse.
Alt+*key,* Ctrl+*key,* Shift+*key*	Hold down the first key as you press the second key. For example, "Press Alt+Enter" means to hold down the Alt key as you press the Enter key.

Chapter 1 Getting the Look You Want

What Affects a Scene?

Realizing the vision of your model or scene in your mind's eye takes talent, a good knowledge of 3D animation, patience—and, of course, shaders.

Once you have created a model, several models, or a full scene, your next step is either animating or rendering.

To animate objects, refer to the *Animating* guide. To render, keep reading.

Technically speaking, rendering is the process of outputting a final image. But, in fact, before you output a single frame, you will most likely want to "add life" to your scene. This is usually done in two steps:

- First, you may wish to reposition your lights, your camera, and even your objects. You would have to be an exceptionally talented animator not to touch any of your scene's objects before rendering. See *Placing Your Objects in Your Scene* on page 20 for more information on object placement.
- Second, once you've settled on the placement of your objects, you can start adding realism in the form of shaders. Shaders allow you to control how an object looks, how a light falls, and how a camera sees. See *Using Shaders* on page 24 for an overview of how shaders can be used in your scene.

Placing Your Objects in Your Scene

When you are building a scene, there are countless ways to achieve different looks. The most obvious way is to move your objects, lights, and cameras. Each object has a unique relationship with every other object. Understanding how objects interact with each other is the first step in achieving the look you want. Some of these relationships can be quite subtle and esthetic, but they can greatly influence your scene's look.

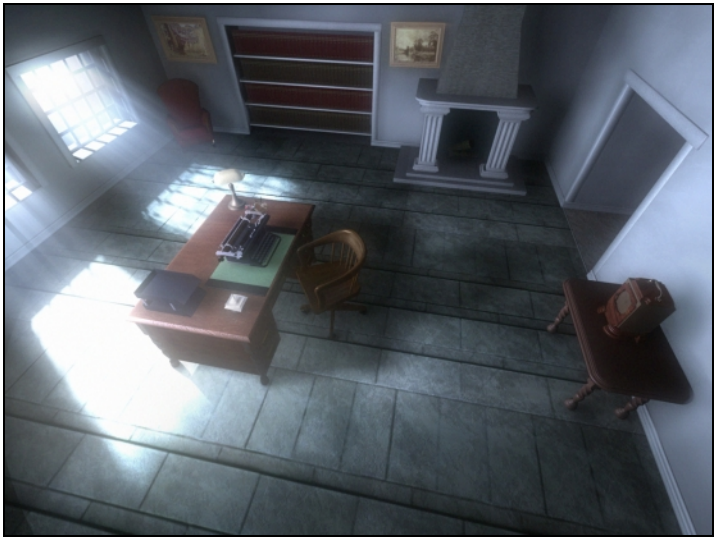
Shaders notwithstanding, elements can achieve different looks, depending on their fellow elements. For example, an object's appearance will depend on how it is lit and where the camera view is located. The following section briefly describes how placing basic objects can affect your scene.

Placing Lights

This is the most important object to consider before rendering your scene. Without a light, it doesn't matter what your object—or your scene—looks like.

Lighting plays a key role in defining surfaces, because the interaction between material properties and the light source creates the visual characteristics of the object. For best results, create your light sources before setting the material.

A light's position relative to an object and camera is vital when trying to achieve a certain look for your scene. Will the object be shaded? Blasted? Or somewhere in between? Will it be transmitting a special effect (such as caustics)? Or perhaps global illumination? Whatever the case, your light's position is crucial to your scene.



The relation between a camera and a light's position is more subtle than the relationship between lights and geometric objects. A camera's position determine's the light's position, and vice versa. When set properly, cameras and lights can create some spectacular effects.

Types of Lights

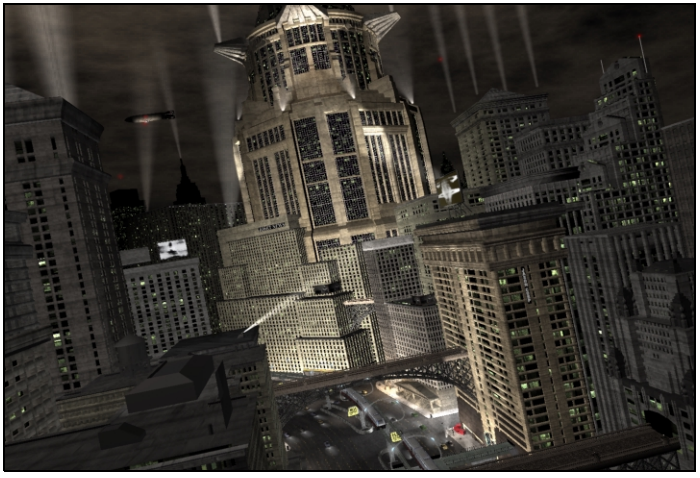
There are several lights, depending on the look you want:

Light	Description
Infinite	Sun-like
Neon	Cylinder-shaped light that acts like a point light
Spot	Standard spotlight
Light Box	Diffused spotlight
Point	A point of light, similar to a light bulb

For more information on lights in general, see *Chapter 5: Working with Lights* on page 131.

Placing Cameras

Whether you have one or one hundred cameras in your scene, a camera's position relative to an object is, most of the time, a matter of esthetics. Using high or low angles, or camera rolls, can add a sense of drama or fun to a scene.



This scene uses a skewed camera angle, several selective point lights, and a few spotlights to create a dramatic effect.

Artistic considerations aside, a camera's position determines how an object should be textured and subsequently rendered. If an object is far from the camera and doesn't require a great deal of detail, there isn't much use creating an ultra-photorealistic texture map that will simply slow your rendering time. On the other hand, if an object will be very close to your camera, you will probably want to spend the time creating a detailed texture that will not look fake when in the foreground.

Types of Cameras

A variety of cameras can be used, depending on the look you want.

Camera	Description
Default Perspective	The default-perspective camera
Wide Angle	A basic camera with a very wide angle of view
Telephoto	A basic camera with a zoom lens
Orthographic	A camera that does not display depth or perspective

For more information on camera types and their characteristics, see *Chapter 7: Camera Basics* on page 169.

Multiple Cameras

You can create as many cameras as your scene requires. Each camera view can be rendered separately or assigned to a different render pass. In addition, each camera can have its own unique settings and format.

Distortion

If your camera is very close to your object, not only might you require a great deal of detail in your texture, but you may also encounter (on purpose or not) distortion. You can use various lens shaders to correct, create, and define camera distortion.

Placing Objects

How you place your objects or how they are rotated and scaled affects rendering just as much as the actual animation of your scene. Where you place your objects will affect where shadows are cast, how cameras will distort, and how you should use certain shaders on objects. That said, the final location of your objects can determine the realism, or "mood," of the scene.

Geometric Approximation

You can easily change an object's "smoothness" by modifying its basic geometry or rendered geometry—surface approximation—to achieve different looks. For example, a sphere with four UV subdivisions is not as smooth or as round as a sphere with 16 UV subdivisions. Geometry also affects how a displacement map will look on a given object. The more subdivisions an object has, the smoother the map will appear. For more information on using geometric approximation, see *Setting an Object's Surface Approximation* on page 83 in the *Rendering* guide.

Motion Blur

How an object is animated or how it moves across the scene is one of the main characteristics that define motion blur. An object that moves quickly from one end of a scene to another produces more motion blur than a slower object. For more information on how to create and render motion blur, see *Blurs, Flares & Other Effects* on page 185.

For more information on how to animate an object, see the *Animating* guide.

Using Shaders

Every object can be affected by a single shader or a complex structure of several shaders. Shaders are responsible for applying materials and textures or telling a surface how to react to light. Every single characteristic on an object's surface is defined by a shader, whether it be its color, how it refracts light, or how its shadow is displayed.

For more specific information on the various types of shaders and how to apply and edit them, see *Chapter 2: Shader Basics* on page 37.

The following section describes just some of the effects you can create on various objects using shaders.

Object Effects

Most shaders that affect geometric objects control their surface characteristics. You usually start with a surface shader and build a texture from there.

Surface/Material Shaders

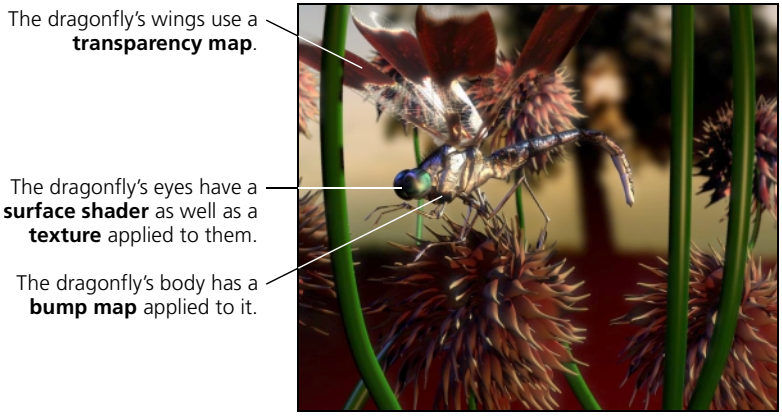
A surface shader is the most basic of shaders. Surface shaders come in different shading models (Blinn, Phong, Lambert, constant, etc.) and define the basic color, reflectivity, and transparency of an object's surface.

Realistic glass surfaces were created using reflection, refraction, and transparency. You can work with all these properties in a surface shader's property editor.

Materials are not the same as they were in SOFTIMAGE|3D. Rather than being a base to which a shader is applied, a material is now a shader that is a base connection point or placeholder for the surface, contour, environment, and so on. Every object has a base material node with no values attached to it. It automatically takes on the values of the default scene material. For more information on materials and textures, see *Chapter 3: Material & Texture Basics* on page 59.

Textures

Most shaders create, define, and refine textures. Most 2D and 3D textures are applied from the toolbar or a property editor.



In SOFTIMAGE|XSI, textures are shaders that are applied in addition to a surface shader. Of course, you can still use images to use as textures, but they are referenced by the texture shaders.

Bump/Displacement Maps

In addition to creating texture maps, you can define bump and displacement maps using the same shaders and tools as you would for textures. For example, you can use shaders to apply a lizard skin onto your model, then use another shader to create a scaly bump map. And that's just a start!

Transparency/Reflection Maps

To increase realism in a scene, you can add transparency and reflection maps to a geometric object's surface. Essentially, these maps are textures or images that are seen on or through an object's reflective or transparent properties, respectively. For example, you can use a reflection map on a car's windshield to reflect a cloud texture and a transparency map of the inside of the car for when the windshield is transparent.

Light Effects

All lights are created with a default shader that let you define a light's color, falloff, cone spread (in the case of a spotlight), and, of course, intensity. But you can easily add a shader to a light that can, for example, customize its falloff, light shards, volume, and so on.

If shadows are used, light shaders normally cast shadow rays to detect obscuring objects between the light source and the illuminated point. For information about lights in general, including shadows, see *Chapter 5: Working with Lights* on page 131.

Shadows

Shadows are an excellent way to create a sense of depth, realism, or mood. You also have a great amount of control over shadows. Of course, the most realistic types of shadows take longer to render, but you can define whether you wish to use raytraced shadows, use a shadow map, or define area lights to create softer shadows. You can then define *how* they will render.

Primary and Secondary Rays

In the real world, light will illuminate objects and reflect off of it. This is called primary and secondary rays, respectively. For example, if you see an object lit by a spotlight, you are seeing its primary rays; if you see that object's reflection, you are seeing its secondary rays. The primary and secondary light rays are not mutually exclusive: you can see one without seeing the other. For example, a vampire character can be seen walking through a room (primary rays), but his reflection in a mirror (secondary rays) will not.

Photon Lighting

Sometimes called radiosity, photon lighting encompasses global illumination and caustic lighting. One of the best ways to create photorealistic lighting is to use photon-lighting effects.

Caustics simulate the seemingly random reflections of highly reflective surfaces such as the reflections or "hot spots" at the bottom of a pool.

Global illumination takes into account the color of objects within a scene and how they affect other objects when light reflects off them. For example, using global illumination, a bright red sofa in a room will create a reddish tinge on a white wall.

For more information on caustics and global illumination, see *Chapter 6: Global Illumination & Caustics* on page 153.

Camera Effects

Every camera in a scene is assigned a basic default shader that controls its format, projection, field of view, and clipping planes. In addition to a camera's basic properties, you can add a shader that distorts (e.g., fisheye), create a flare, or change the way in which a camera "sees" a scene (e.g., cartoon effect).

Lens Flare

Applying a lens flare is quick and easy. You can also add realism to a flare by adding starburst flares and shards that rotate as the camera moves.

Motion Blur

There are several different types of motion blur, each one with a varying degree of realism. Motion blur simulates the effect of using a long shutter speed while photographing moving objects. The objects leave image trails, or smears, depending on their direction and speed.

Depth Fading

Controlling how far your camera can "see" using depth fading is often used to create realistic fading that occurs when viewing far-away objects or sceneries. Lights placed far from the camera can have an influence on how realistic a depth-fading effect will be.

For more information on how to create camera effects, see *Chapter 8: Blurs, Flares & Other Effects* on page 185.

Volumic Lights

Volumic lights simulate light density. As an extreme example, think of a street light shining down on a foggy street. The light cone of the street light is very visible because it is illuminating a volumic (or dense) space.

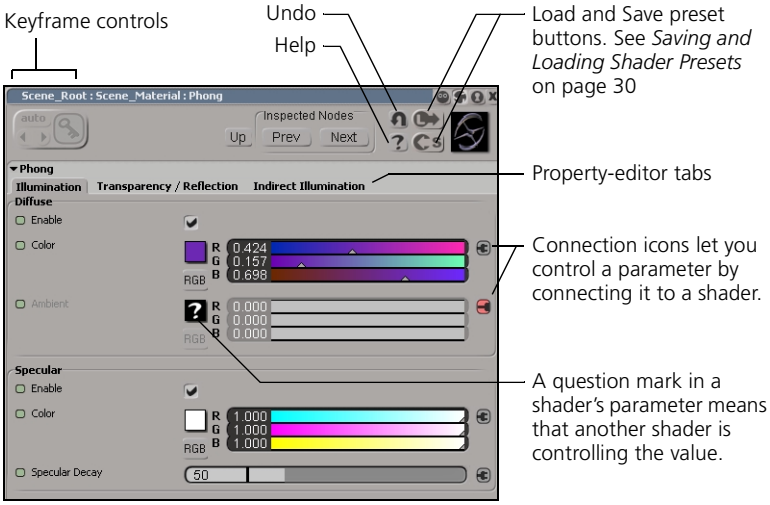
For more information on creating and rendering volumic lights, see *Creating a Volumic Light* on page 197.

Useful Tools

In order to apply materials, textures, and camera and light effects, there are a few tools you should know about. Each one of the following tools helps you connect, disconnect, and edit a shader. In addition, you can load and save presets as well as change the shader's position in a hierarchy.

Shader Property Editors

Every shader is controlled through its respective property editor. Different types of shaders have different property editors associated to them, and you can display and use more than one property editor at one time. For more information on property editors, see the *Fundamentals* guide.



To open a shader property editor

There are three ways to do this:

- Open an object's shader property editor by choosing **Modify** > **Shader** from the render toolbar.

  or

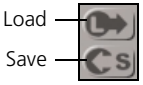
- Select an object in a viewport or render tree view and press Enter. Choosing more than one object and pressing Enter displays the Multiple Selection property editor, which contains the objects' common properties and shaders.

  or

- Click on an object's Iconium the explorer to display its subsets of properties.

If a parameter in a property page is replaced with a question mark (?), this means that the parameter is being driven or controlled by another shader. For more information on connecting a shader to a parameter, see *Using the Connection Icon* on page 44.

Saving and Loading Shader Presets



Presets are files that contain values for all settings in a property editor. You can load a preset or you can save the values in a property editor and name them as a preset. You can then load and use them again later. Presets have a **.preset** file name extension.

Presets are useful if you want to select a particular shader and modify some of its attributes, save the settings, and load them as a shader preset for more than one object. It saves you from having to select each shader and set the same parameters each time for other objects.

Saving Shader Presets

To save a defined shader

3. Open a shader's property editor.
4. In the shader's property editor, click the **Save** icon (left) on the right side of the property editor and use the browser to select a directory and file name for your preset.
5. Click Ok.

Creating Preset Thumbnail Images

You can use the render region to generate a thumbnail image for your shader presets. These thumbnails are displayed in the browser and help you identify your presets.

To create thumbnail presets

1. Draw a render region (press **q**) around an area of your scene that displays the effect of the shader you want to save as a preset.
2. Open the shader's property editor (**Modify** > **Shader**).
3. In the property editor, click the **Save** icon on the right side of the property editor and use the browser to select a directory and file name for your preset.
4. Select **Use Region as Thumbnail**. Your preset is saved with a thumbnail identical to the area shown in the render region. The option is dimmed out if no render region is defined.

Loading Shader Presets

After you have saved a shader preset, you can apply it to any other object in your scene. You can also apply the same preset to several objects at once, if multiple objects are selected.

A preset is loaded as a copy, so if you make changes to it, only that instance of the preset is affected.

To apply a preset to an object

You can do any one of the following:

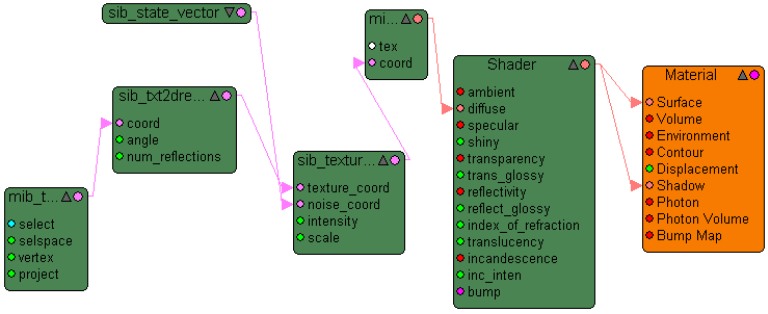
- Open a browser and drag the preset onto an object node shader attachment point in the explorer or render tree.
- Drag and drop a preset from the browser to a viewport. The shader preset is immediately applied to the material node.
- From a shader's property page, click the **Load** icon (previous page) and choose the preset from the browser.

Explorer View

You can use the Explorer view to see a visual representation of object and shader hierarchies. The explorer is capable of focusing on specific properties of a scene. For example, you can use the explorer to filter shaders or materials so only those icons are shown. You can also drag and drop icons to rearrange a hierarchy or right-click on it to access its property editor.

Render Tree View

The render tree view is very useful for creating accurate and complex shader structures. It gives you fine control over every shader and each of its parameters. Shaders are displayed in a hierarchical tree format that lets you connect and disconnect shaders wherever you wish.



For more information on how to use the render tree, see *Chapter 9: The Render Tree* on page 205.

Render Region

The render region allows you to immediately and interactively see how your scene will be rendered. For more information on how to use and define settings for the render region, see *Previewing Interactively with the Render Region* on page 32.

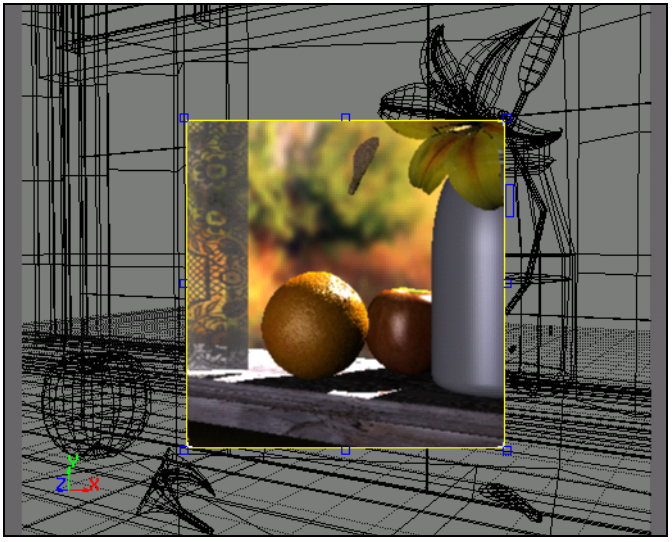
Previewing Interactively with the Render Region

You can view a rendering of any section or object in your scene quickly and easily. Rather than setting up and launching a preview, you can now simply create a render region over any viewport and see how your scene will appear in the final render. You can resize and move a render region, select objects and elements within the region, as well as modify its properties to optimize your preview.

Whatever is displayed inside that region is continuously updated as you make changes. The region lets you select elements inside its bounding box and provides interactive visual feedback when you modify and apply rendering properties to objects in your scene. Only this area is refreshed when changing object, camera, and light properties, when adjusting rendering options, or when applying textures and shaders.

Depending on its settings, the render region can render the scene in the same way as a final output render. This gives you a very accurate preview of what your final rendered scene will look like.

The refreshing of the render region appears as a series of tiles that the mental ray® rendering software re-renders at each change. Each time you change properties that affect the appearance of objects in your scene, the region is automatically refreshed. You do not have to wait for the region to finish refreshing to make more changes to your scene. A change to your scene interrupts the current refresh and restarts it to include the updates you have just made.



Creating a Render Region

You can draw a rectangular region of any size around the objects you want rendered interactively within any of the four viewports. You can draw only one render region at a time in any viewport. A region can render any geometry view or camera point of view displayed in a viewport: User, Top, Front, Right, Spot Lights, and Camera views, including the Render Pass view.

To create a render region

1. Do one of the following:

   - Choose **Render** > **Region** > **Region Tool** from the Render toolbar to activate the render region mode. The mouse pointer shows a square with a sphere in it to indicate that you are in region mode. You can now define a region in any viewport.

     or

   - You can also press the **q** supra key to activate the render region mode. This is the same as choosing **Render** > **Region** > **Region Tool**.

To delete the current render region, press **q** to activate the render region tool and click in any viewport.

2. Drag the mouse pointer diagonally across the viewport to create the render region. The region is drawn as a box with a yellow border and blue resizing handles.
3. As soon as you release the mouse button, the region is drawn and whatever is displayed in that region is rendered.

The render appears as a series of tiles that fill up the region. Each tile is outlined with an angled *L* marker at each corner, signifying which tiles are being updated at each refresh. When you make a change that affects the visual appearance of the scene, only the part of the scene shown in the render region is rendered.

Moving and Resizing a Render Region

You can resize and reposition the region in the viewport by dragging its border. The mouse pointer changes to indicate the type of action you can perform with the region. The placement and size of the render region are limited by the layout and size of the viewports themselves. If you wish to start over, simply draw a new region anywhere you like.

To move a region

1. Position the pointer on the border of the render region's bounding box. The pointer changes to a four-way directional arrow.
2. Click and drag the pointer in any direction to reposition the region. Once positioned, release the mouse button and the region is refreshed.

To resize a region

1. Position the pointer over one of the blue squares on the corners or borders of the render region. The pointer changes to a directional arrow.
2. Click and drag the pointer to resize the region. Release the mouse button and the region is refreshed.

Setting Render Region Options

The render region renders the scene in the same way as a final output render. As such, the rendering options you set for the render region gives you a very accurate preview of what your final rendered scene or current pass will look like.

These rendering options are a subset of the global rendering options set for a render pass, but they affect only the render region and are not used for the final render.

Previewing the changes you make to your scene in a render region is an interactive process. To speed up the "tune-preview-retune" cycle, use the region's rendering options to optimize the display.

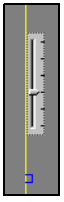
To open the Render Region property editor

Choose the **Render** > **Region** > **Options** from the Render toolbar to open the View Render Options property editor.

For more information on the render settings, see *Chapter 3: Rendering Options* on page 51 of the *Rendering* guide, or refer to the online help in the property editor.

You can quickly set the rendering accuracy (sampling level) with the slider on the right side of the region border:

- Position the pointer over the blue rectangle on the right side of the region border. The **Max Sample** slider appears.
- Drag the slider down to decrease the maximum sample level and increase the render speed in the region.



For more tips on how to optimize a render region, see *Chapter 3: Rendering Options* of the *Rendering* guide.

Refreshing the Render Region

By default, the render region refreshes automatically each time you modify the region or change scene properties. You can optimize region refresh when moving, resizing, or applying changes to your scene elements by choosing to manually refresh the region.

You should deactivate the automatic refresh of the region if you are making a number of changes that require intensive rendering calculations, like caustics and motion blur. You can refresh the render region after you have set these options and made your scene changes.

To deactivate automatic refreshing

1. Draw a render region in a viewport.
2. Choose the **Render** > **Region** > **Auto Refresh** command on the Render toolbar. This turns off the automatic refresh of the render region.
3. Make the necessary changes to your scene or change the settings of the render region options. When you want to view your changes, choose the **Render** > **Refresh** command. Your changes are rendered in the region.
4. At any point choose **Render** > **Region** > **Auto Refresh** again to activate the automatic refreshing of the region.

When the render region is not set to Auto Refresh, the region's outline is red instead of yellow.

Tracking with the Render Region

Rather than always adjusting your render region's size and location while you are working in a scene, you can set the render region to track an object, hierarchy, group of objects, or even a part of an object (cluster).

1. Select an object in any viewport and choose **Render** > **Region**> **Track Selection** from the Render toolbar.
2. If necessary, open a render region in the viewport in which you wish to track the selection.

As the object moves, the render region follows it. If the object changes size, the render region is recalculated to fit around it.

If you are working on a small part of an object and you would like to track a specific area rather than the entire object, select a cluster or tagged points on the object and click **Render** > **Region**> **Track Selection**. The render region focuses on and tracks only the selected portion of the object.

The track selection always remains on the selected object. Selecting another object causes the render region to track the new selection.

Chapter 2 Shader Basics

Shaders are basic rendering tools that let you create a variety of special effects, such as lens flares, fog, textures, surfaces, and more. Shaders help you achieve just about any visual effect you can imagine.

A shader is a miniature computer program that controls the behavior of mental ray rendering software during or immediately after (in the case of output shaders) the rendering process. Some shaders are invoked by mental ray to compute the color values of pixels. Other shaders can displace or create geometry on the fly.

There are many different types of shaders: surface, environment, volume, shadow, photon, volume photon, light, light photon, lens, texture, displacement, contour, and output. Each of these types of shaders are described on page 41.

Some shaders consist of multiple shaders that are packaged together and work together to create a single, complex visual effect. Such a shader contains a structure of individual elements—or nodes—called a render tree (which is a type of DAG—directed acyclic graph). Each node on these trees corresponds to a single shader.

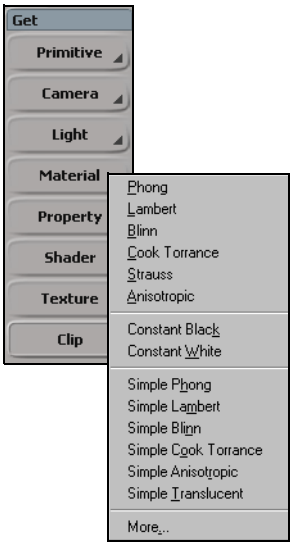
Every object is defined by its material. A material shader acts as a placeholder and accepts nine different types of shaders to define a material an object's look. These nine inputs are:

- Surface
- Volume
- Contour
- Displacement
- Shadow
- Photon
- Environment
- Bump Mapping
- Photon Volume

All aspects of the shader are automatically integrated into the scene; you do not have to worry about which shaders are dependent on others.

For information how to apply and edit a material shader, see *Chapter 3: Material & Texture Basics* on page 59.

Working with Shaders



Shader Icon

If you want to create rendering effects for the objects in your scene, the following provides an overview of the many ways shaders can be used:

- Shaders are used to achieve almost any effect you want. Materials and textures are shaders, as are lens flares, camera effects, and volumic effects. In addition, you are supplied with a hefty library of tool shaders that allow you to create your own shader from scratch or extend an existing shader. For a complete list of each shader and a brief description, see *Shader Descriptions* on page 233.
- Apply a shader to a geometric object, light, or camera using the **Get** > **Material** or **Get** > **Texture** commands from the Render toolbar. You have the choice of applying preset surface shaders, such as Phong, Lambert, Strauss, and so on. You can attach shaders to geometric objects, lights, cameras, render passes, pass partitions and to the scene root. For more information, see *Attaching Shaders to Elements* on page 41.
- Save and load the properties of a shader as a preset. For more information about property editors in general, including how to load and save presets, see *Saving and Loading Shader Presets* on page 30.
- View a shader in its hierarchy by using the explorer. From any viewport, select the explorer view, select an object, and expand its icon. A shader is a subnode of the material node of an object. It is represented by a colored sphere icon in the explorer (left).
- Modify a shader's properties using its property editor. To open its property editor, right-click on the shader in the explorer and choose **Properties** from the pop-up menu or use the **Modify** > **Shader** command on the Render toolbar. For more information about the specific parameters in a shader's property editor, click the **Help** icon (left) in that property editor.
- Connect an almost endless amount of shaders together to create a complex or very specific effect. Connecting a shader's parameters to one or more other shaders can be done with the connection icon or through the render tree. For more information, see *Using the Connection Icon* on page 44 and *Chapter 9: The Render Tree* on page 205.
- Animate a shader's properties in the same way as you animate other properties. For a complete description of how to animate objects, see *Animating with Keys* in the *Animating* guide.
- When you render, you can select and deselect different shaders. This lets you optimize rendering time while still achieving the effect you want, because only the necessary types of shaders are invoked. These settings can be made independently for each render pass, and they can be found on the Active Effects page of the Render Options property editor. For more information, see *Chapter 3: Rendering Options* on page 51 of the *Rendering* guide.

Attaching Shaders to Elements

There are several ways you can attach a shader to an object.

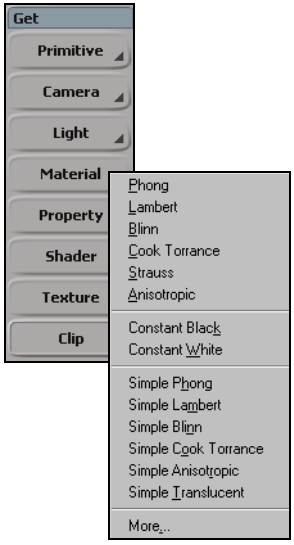
You can:

- Attach shaders to geometric objects, cameras, lights, and render passes.
- Attach a shader to multiple objects, on a group, and to a hierarchy of objects.
- Apply a shader and have its properties propagate to all objects in a branch or model hierarchy.
- Attach a shader to a cluster of polygons to define local materials and textures.
- Assign a tree of shaders to a single property of another shader.

You can also drag and drop a shader from the browser to elements in your scene displayed in the explorer or render tree.

If you attempt to attach a shader to a non-compatible object type (for example, you cannot attach a lens shader to a light), the mouse pointer displays a circle and bar icon to indicates that you cannot perform the action.

Attaching a shader to a camera or light is very similar to connecting a shader to an object. For more information on how to specifically use shaders with cameras and lights, see *Chapter 7: Camera Basics* on page 169 and *Chapter 5: Working with Lights* on page 131.



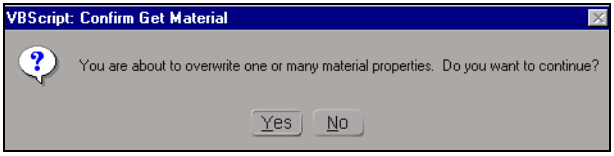
To attach a new material and surface shader to an object

A material node and surface shader are the basic shaders needed to create effects using textures or tool shaders.

1. Select an object. You can select the object from the explorer or from a viewport.
2. Choose **Get** > **Material**. From the list of presets that appears, choose one of the surface shaders to apply it to the object.

   In the case of a multiple selection that includes the same type of objects, the shader is attached to each object in the selection.

   If the selected objects already have a material or other shader applied to them, you are prompted to overwrite the existing shared material.



3. Click **Yes** to overwrite the old materials properties, or click **No** to leave your selection as is.

   For a multiple selection that includes different types of objects, such as a selection of lights and geometric objects, a warning appears in the mouse/status line at the bottom of the main window. You are prompted to **Continue** and attach the shader to only the appropriate objects or to **Cancel** the entire operation.

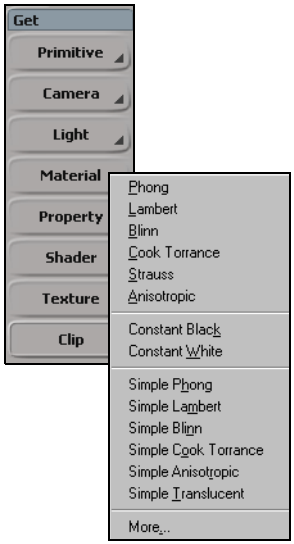
4. After the shader is attached to the selection, its property page is automatically opened. You can now edit its parameters.

Detaching Shaders

There are several ways to detach a shader:

- Attach a surface shader to an object that already has a surface shader attached; the old shader is removed and replaced by the new surface.
- Delete a shader from the explorer: it is removed and no longer affects the object.
- Use the render tree to graphically disconnect and/or delete the shader. For more information on how to use the render tree, see *Chapter 9: The Render Tree* on page 205.

Applying and Editing Shaders



When defining what your object will "look" like, the most basic and fundamental step is to apply a material and surface shader. You can apply a multitude of materials and surfaces simply by clicking a button.

To apply a new material and surface shader

There are two ways:

1. Select a geometric object. Its name appears in the Current Selection text box in the Selection panel.
2. Choose **Get** > **Material** from a toolbar. From there you can choose a wide range of preset surface shaders.

   or

   Choose **Get** > **Material** > **More** from a toolbar, and from the browser that appears, you can select a surface shader or a Preset. Click OK.

   With either method, the new shader is attached to the object and the old material (if any) is removed.

When choosing **Get** > **Material**, you are actually changing the material node as well as the surface shader. This tool not only replaces the surface shader in a tree, it starts from "scratch" by deleting every shader associated to an object and replaces it with a new material node and the chosen surface shader.

For more information on materials and surfaces in general, see *Chapter 3: Material & Texture Basics* on page 59.

Applying Shaders

Once you've attached a material and surface to your object(s), you can start to create virtually any type of surface you want. You can use any of these methods to attach any further shaders:

- Using the Shader button on the Render toolbar
- Using the connection icon in a shader's property editor
- Using the explorer
- Using the render tree

Using the Shader Button (Render toolbar)

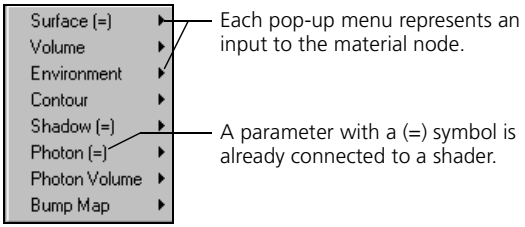
Once you have applied a surface shader to an object, you can map virtually any shader on to any parameter of the material node.

To map a shader

1. Select an object that has a material applied to it.

2. From the Render toolbar, choose **Get** > **Shader**. The menu that appears lists every parameter (input) that can be mapped to the material node.
3. To map the Surface parameter of your material, click on **Surface**. A submenu of frequently used nodes appears. For example, select the **Strauss** shader if you want to connect the Surface parameter of your material to the Strauss surface shader.

Parameters marked with a (=) symbolize an existing connection to another node.



4. Once you select a shader to connect to a parameter, the chosen shader's property editor opens automatically. You can now edit its parameters.

Using the Connection Icon

Connection icon

Once you apply a surface shader, its property editor appears. To the right of each parameter, there is a connection icon—it looks like a plug. Click the icon to open a menu of shaders that you can attach directly to the parameter. The icon lets you control a shader's parameter with another shader instead of a simple color or numeric value.

To connect a shader to a parameter

1. From any shader's property page, click the connection icon for the parameter you wish to connect to another shader.
2. From the pop-up menu select one of the following:

Option	Result
Edit	Opens the property editor for the shader connected to the parameter.
Disconnect	Cuts the connection between the parameter and an attached shader.
Blend with	Opens another pop-up menu from which you can select a texture shader. The selected shader will be blended with the parameters value using a mixing shader.
Various shaders	Lists shaders that are commonly associated with the parameter.
More...	Opens a browser and let you select any shader from the shader library.

Using the Render Tree

Using the render tree to connect shaders to each other gives you the widest possible range of control when connecting shaders.

For more information on how to use and create effects with the render tree, see *Chapter 9: The Render Tree* on page 205.

Regardless of which method you use to apply a surface shader, the shader's property editor will be opened in a floating window as soon as it is applied.

Replacing Shaders

What do you do if you suddenly realize you'd prefer a Blinn shading model instead of the Lambert you already connected? Simply replace it with a drag'n'drop.

To change a shading model

The easiest way to change a shading model on an object is through the render tree:

1. Select the object that you want to edit, and open a render tree view for the object in a viewport.
2. Open a browser and navigate to the Material folder.
3. Drag and drop the new surface shader directly over the one you wish to replace in the render tree.

All of the render tree's connections will remain intact, but the new surface shader replaces the old one.

If you choose **Get** > **Material** when the selected object already has a material applied to it, you are asked if you wish to replace the existing material. If you choose yes, a new material and a surface shader is applied, thereby deleting any previous connections you might have made to the material node.

Editing Shaders

An important part of the process of fine-tuning a scene is editing its shader properties. Every shader is edited through its property editor. Property editors contain the various parameters that define the properties of individual objects, whether they be geometric objects, lights, or cameras. You can display and use multiple property editors simultaneously.

Information about parameters listed in the property editor and its property pages is available from the online help in the property editor.

To edit an object's shader

1. Select an object in the viewport whose shader you want to edit.
2. From the toolbar, choose **Modify** > **Shader**. The property editor opens in a floating window. You can now edit any of its parameters. Click the various tabs to select different parameter groups.

Other ways to open a property page

Shader Icon

- With the object selected, click the **Property** button on the Selection panel to display the selected object's node. Expand its material node and click the surface shader icon beneath the surface node to open its related property editors.

  or

- Select the object and open a render tree view. Double-click on the shader node you wish to edit.

Deleting a Shader

You can delete a shader by selecting it in either the Explorer, Render Tree, or Schematic view and pressing the **Delete** key. The shader will be removed from all views.

The Shader Library

Shaders are divided into several different categories, based on how they are invoked and where they are used. Every shader can be opened in an explorer or a render tree and edited through its property editor.

The shader library can be found in:
<install directory>/DSPresets/Shaders/

For a complete list of all shader names and a brief description, see the *Shader Descriptions* appendix. For a more detailed description of each shader, refer to online help available through its property page.

All shaders fall into the following categories and are similarly divided in the Shaders folder:

Surface

Surface shaders are one of the most important types of shaders. All geometric objects in a scene have an associated surface shader, even if it is only the default shader of a scene. Surface shaders determine the basic color of an object, possibly in conjunction with volume, environment, shadow, texture, displacement, and contour shaders. Surface shaders are also responsible for casting reflected, refracted, and transparency rays. For more information on surface shaders and how they are used, see *Surface Shader Basics* on page 67. For a description of the material node, see *The Material Node* on page 52.

Soft_Material shader

All geometric objects defined in a scene are assigned the soft_material shader when imported into SOFTIMAGE|XSI. The shader's material property editor lets you edit and set the following properties: shading model, ambient, diffuse, specular, specular decay, reflection, transparency, and refractive index.

For more information, see *Using the soft_material Shader* on page 73.

Textures

2D texture shaders apply a two-dimensional texture onto an object, as 3D texture shaders implement a three-dimensional texture into an object. They are used by a surface shader when an object has a defined texture.

Texture shaders also include the necessary shaders to create bump maps.

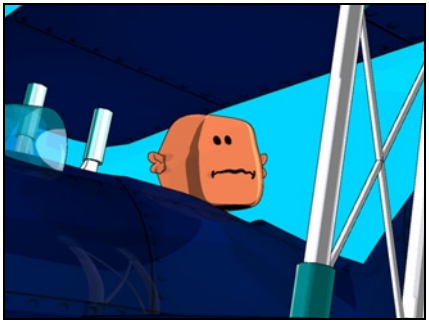
For more information on texturing concepts, see *Texture Basics* on page 83.

Light

Light shaders implement the characteristics of a light source. For example, a spotlight shader uses the illumination direction to attenuate the amount of light emitted. A light shader is used whenever a surface shader uses a built-in function to evaluate a light. If shadows are used, light shaders normally cast shadow rays to detect obscuring objects between the light source and the illuminated point. For information about lights in general, see *Chapter 5: Working with Lights* on page 131.

Lens

Lens shaders are used when a primary ray is cast by the camera. They may modify the ray's origin and direction to implement cameras other than the standard pinhole camera, and they may modify the result of the primary ray to implement effects such as lens flares or a cartoon effect (below). For more information on working with cameras, see *Chapter 7: Camera Basics* on page 169 and *Chapter 8: Blurs, Flares & Other Effects* on page 185.



Volume

Volume shaders are used to modify rays as they pass through an object (local volume shader) or the scene as a whole (global volume shader). They can simulate effects such as clouds, smoke, and fog. For more information on how to apply volume shaders, see *Volume Shaders on page 201*.

BBC "Everyman": Animation by Aldis Animation

Output

Output shaders operate on images after they are rendered but before they are written to a file. They can perform operations such as filtering, blurring, compositing with other files, and writing to different file formats. For more information on how to apply output shaders, see *Chapter 8: Blurs, Flares & Other Effects* on page 185.

Tool Shaders

The tool shaders are the basis for creating a shader from scratch or extending an existing one. Although some can be used on their own, many of them must work in conjunction with another to achieve a highly customized effect. They each have a specialized function:

- **Color Channels** manipulate red, green, blue, and alpha components of a color.
- **Conversion** changes one value to another. Especially useful for changing a scalar-type node to a color one. Scalar, color, vector, Boolean, and integer nodes can be converted to any other type of output using these tools.
- **Illumination** creates some of the more common shading algorithms or shaders. Besides the usual Blinn, Phong, and Lambert shadings, you'll find a few more specialized tools, including photon shading.
- **Image Processing** defines, manipulates, and tweaks color and scalar values.
- **Math** performs math functions such as interpolation, multiplications, additions, and exponential functions.
- **Mixers** uses one or several equations to mix a few or several colors or textures into a single color output.
- **Raytracing** creates raytracing effects, such as refraction, transparency, or reflection. These tool shaders give you unprecedented access to deep raytracing values inside a node, and they let you "get your hands dirty" by tweaking small yet complex values.
- **Share** coordinates the sharing of a single value among several others.
- **Surface** creates custom effects or fine-tunes a surface.
- **Switch** changes an object's color based on location, angle of view, or another specified value.
- **Texture Generators** are the basis for creating textures within a shader. These shaders create basic 2D and 3D textures.
- **Texture Space Controllers** manipulate a texture space by perturbing, rotating, scaling, or using a number of other functions.
- **Texture Space Generators** generate a specific texture space to be mapped onto an object in a variety of ways.

Shadow

Shadow shaders determine how the light coming from a light source is altered when it is obstructed by an object. They are used to define the way an object's shadow is cast, such as its opacity and color. Shadow shaders are used instead of surface shaders when a shadow ray intersects with an object. Shadow rays are cast by light sources to determine the visibility of an illuminated object. Shadow shaders are basically "lightweight" surface shaders that calculate the transmitted color of an object without casting secondary or shadow rays. For general information about creating shadows, see *Creating Shadows* on page 144.

Photon

Photon shaders are used for global illumination and caustics. They process light to determine how it floods the scene. Photon rays are cast from light sources rather than from a camera. For more information on global illumination and caustic effects, see *Chapter 6: Global Illumination & Caustics* on page 153.

Environment

Environment shaders are used instead of surface shaders when a visible ray leaves the scene entirely without intersecting an object or when the maximum ray depth is reached. For example, an environment shader might evaluate a texture mapped on an imaginary infinite sphere enclosing the scene. For more information on environment shaders, see *Environment Shaders* on page 203.

Displacement

Displacement shaders alter an object's surface by displacing its points. The resulting bumps are visibly raised and can cast shadows. For information about setting displacement properties, see *Setting an Object's Surface Approximation* on page 83 of the *Rendering* guide.

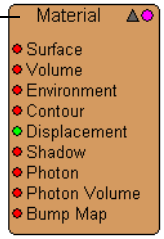
Geometry

Geometry shaders are evaluated before rendering starts. This allows the shader to introduce procedural geometry into the scene. For example, a light shader might install a transparent cone in the scene that can bound a volume effect.

The Material Node

Every object has a material node. Without it, an object wouldn't render—it acts like a placeholder for every shader that can be applied to an object. Shaders can alter an object by invoking one or several of nine shader input types: Surface, Volume, Environment, Contour, Displacement, Shadow, Photon, Photon Volume, and Bump Map. See *Attaching Shaders to Elements* on page 41 for a description of the available shader types.

Every type of shader that can affect how an object looks is eventually connected to the material node of an object. It acts as a placeholder for all shaders that determine an object's surface/material.



For more information on the Material node's function, see *The Material Node* on page 62.

To learn more about how the Material node is used in the render tree, see *Where to Start?* on page 218.

The Material node in SOFTIMAGE|XSI doesn't have quite the same function as it does in SOFTIMAGE|3D. What was called "materials" are now more accurately named "surface shaders." The Material node acts like a container for all of the possible shaders that can be applied to an object, much like the Material Editor in SOFTIMAGE|3D.

Shader Inputs and Outputs

In order to create customized or detailed effects, you have to know how to connect shaders to one another whether you are working in the render tree or simply mapping shaders together via their connection icons.

The following section explains how a shader's output affects another's input. For example, when connecting a wood shader to the surface input of a material, you expect to see your object's surface resemble wood. But what if you wanted to connect that same wood shader to the Shadow input? Or the Volume input? What would happen then?

In fact, SOFTIMAGE|XSI has an almost endless amount of input- and output-shader combinations.

The following pages describe the fundamentals of connecting different node types and what will result.

Inputs and Outputs

Types of Input and Output	Result
Color	The node returns or outputs a color (RGB) value. These input/outputs are most often used in conjunction with the surface of an object or when defining a light or camera.
Scalar	A scalar input/output is any value between 0 and 1.
Vector	This output/input corresponds to vector positions or coordinates. As an output, it is returning a specific vector position. As an input, it is required to map a texture, for example, to a specific location.
Boolean	This type of input/output corresponds to a 0 or 1 value. It could also be equated to On and Off.
Image	This input only accepts an image file.

Parameters: The Ins and Outs

As long as the output of a shader matches the input of a parameter, you can connect the two nodes. What will happen depends on the value of the outputting node, the setting of the inputting node, and, of course, where you're connecting the node. There are hundreds of different parameters amid all of its shaders and tools. For more information on how to connect shaders using the render tree, see *Connecting Nodes* on page 214.

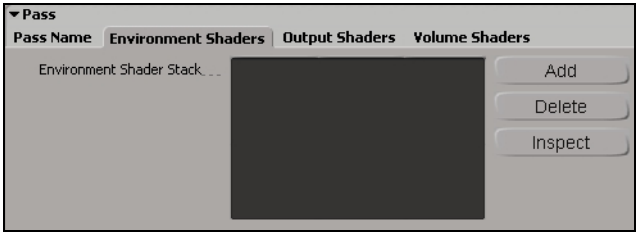
The following chart lists the most common parameters and how they can be driven with some common and some not-so-common shaders.

Parameter Name	Description
Ambience	Defines a geometric object's ambient color. You can use a texture (or gradient) instead of a color.
Bias	Often used to control the contrast between two values. Rather than use a numerical value, use a function such as a scalar texture like cell or flagstone.
Blending	Used to blend two colors or textures. You can also use a gradient or other tool shaders (fractal, wave, etc.) to create non-standard blends.
Color	Defines a color value. Rather than setting a color for this parameter, try controlling it with a texture or a tool shader that generates textures based on vector information.
Cone Spread	Defines a spotlight's cone spread. Try using a noise-generating tool shader (fractal) to create an interesting cone spread.
Diffuse	Defines a geometric object's diffuse color. You can use a texture (or texture tool shader) instead of a color.
Factor	Defines a shader's (bump, intensity, etc.) factor. Try using a fractal or noise shader to control a shader's factor.
Falloff	Defines a light's attenuation. Because the shader needs a scalar input, use a scalar texture shader to control how the light will fall off.
Intensity	Defines a color or light's intensity. Try to control this parameter with a fractal or warp shader to create an uneven intensity.
Light Color	Defines the light's color. Instead of setting a color for the light, you can use the light as a projector by making it use a texture or image clip.
Mix	Defines the mix ratio between two color inputs. As in the Blending parameter, try using a color or texture to achieve an interesting mix.
Reflection	Defines a geometric object's reflection. Use a texture or fractal noise to create a non-standard reflection.
Refraction	Defines a geometric object's refraction. Use a texture or fractal noise to create a non-standard refraction.
Shadow	Controls an object's shadow. Apply a texture shader to make a texture visible in an object's shadow.
Specular	Defines a geometric object's specular color. You can use a texture (or texture tool shader) instead of a color.
Transparency	Defines a geometric object's transparency. Use a texture or fractal noise to create a non-standard transparency.

Local and Global Shaders

When a shader is attached to an object, it is considered *local*. If it is attached to a scene, it is called a *global* shader. Local shaders appear as subnodes of their attached object node, and global shaders are attached to the render-pass node and affect the entire scene.

Output, volume, and environment shaders can be defined both locally and globally. As global shaders, they are attached to the default render pass when imported into SOFTIMAGE|XSI. Global shaders are listed in the shader stack for the particular shader type.



The shader stack lets you apply global shaders to a pass or scene. By using the shader stack, you can define environment, volume, and output shaders.

For example, if you have applied the Soft Fog and Soft Layered Fog shaders to your scene, they appear in the Volume Shaders stack for the default render pass. Any additional render passes you create will include these global shaders. You can add more shaders, delete them, or arrange their order of execution from the Render Pass property page.

Defining Global Shaders

As previously mentioned, a shader applied to an entire scene is called a global shader. Unlike local shaders, they are not applied to a single object but applied globally via a render pass.

A render pass—or simply "pass"—creates a picture layer of a scene that can then be composited back with any other passes during the post-production video-editing process. Rendering a scene in layers can be extremely useful in isolating specific elements of a scene in order to have more flexibility when using special effects.

In the explorer, you can expand a render-pass node to display its Render Pass property page—also called a shader stack. A render pass can be defined by a stack of environment, volume, and output shaders.

For example, a volume shader attached to a render pass is a global volume shader. Global volume shaders are listed in the Render Pass property page for that pass.

You can add or delete shaders in the Render Pass property page.

To edit global shaders

1. Open the explorer and select the **Passes** scope. This displays a list of the defined render passes in your scene. If you haven't created any new passes, you will simply see the Default render pass.
2. Click on the icon of the render pass you wish to edit. The Render Pass property editor opens in a new floating window.
3. Click either the Environment, Volume, or Output tabs to display its specific shader stack. Any volume, environment, or output shaders assigned to the scene appears in their respective shader stack.
4. Select a shader from the list and click the **Inspect** button to display its property editor. You can now edit its properties.
5. Click the **Add** button to open a browser that displays a list of available shaders. Select the shader and click OK to add it to the global shader list.
6. You can delete any shader in its respective shader stack by selecting the shader and clicking the Delete button. This means that the shader is no longer applied to the scene.

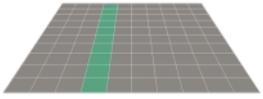
Applying Local Shaders

You can also apply one to individual polygons (local) on a polygon mesh object. You do this by assigning a material locally to its individual polygon clusters (groups of polygons).

A local property or shader will always override a global one applied to the entire object.

All materials (the global material applied to the entire object as well as each local material) have their own 2D and 3D texture lists. Global materials are applied directly to the object, whereas local materials are applied to specific polygon clusters.

Polygon mesh object with global surface shader



Model with selected polygons

Darker surface shader applied to selected polygons

To attach a local material to polygons

1. Select one or more polygons (**y** supra key); the selected polygons are highlighted.
2. Choose **Get** > **Material** from a toolbar and select a shader preset. It will be attached to the selection.
3. Set the parameters for the new material as desired in the property editor.

You can create any pattern you want by repeating the procedure for each new material desired and assigning it to any polygon on the object.

If a local material is detached from the polygons, the material reverts to the default material assigned to the object.

To access a local material from the explorer, you must open the object's node > **Polygon Mesh** > **Clusters** > **Polygon**.

Attaching Shaders to Hierarchies

You can attach shaders to hierarchies as easily as you can attach shaders to a single object. Shaders can be used successfully in a hierarchy to minimize the number of shaders used in a scene. However, there are certain factors that need to be considered when applying a surface shader to a hierarchy.

For example, if you have an object such as a table, you may want the legs and top to be the same color. If you color the parent (table top), the material definition is transmitted (propagated) to its children (table legs). In other words, objects without materials inherit the materials from their parents when in a hierarchy.

An object (in a hierarchy or not) that does not inherit a material from a parent and does not have a locally defined material reverts to the scene's default material. For a complete description of the rules of propagation, see *Hierarchy Propagation* on page 79.

For more information on setting your scene's default material, see *Default Scene Material* on page 64.

To attach a shader to a hierarchy

1. Select the hierarchy you want to apply a shader middle-clicking to branch-select or right-clicking to model-select.
2. Apply a shader by selecting **Get** > **Material**, or by any other method described in *Attaching Shaders to Elements* on page 41.

Chapter 3 Material & Texture Basics

The keystone of photorealism is accurate and life-like texturing. Simply stated, an object's material, surface, and texture properties define how the object looks or reacts to light. Everything from refractive values to transparency and bump mapping to texture maps are applied using texture, surface, and material shaders.

Surface and texture shaders are the two most important shader types to use when defining what your object will look like.

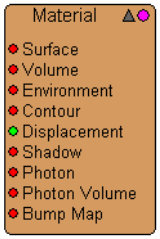
Surface shaders provide a rich variety of parameters for describing surface attributes, including ambient, diffuse, and specular colors, a specular exponent, reflectivity, transparency, and index of refraction. Almost every surface-shader parameter can be connected to one or more textures or texturing tools.

The Material node in SOFTIMAGE|XSI doesn't have quite the same function as it does in SOFTIMAGE|3D. What were called material shaders are now more accurately named *surface shaders*. The Material node acts like a container for all of the possible shaders that can be applied to an object, much like the full-screen Material Editor in SOFTIMAGE|3D.

The surface properties of a geometric object define how the object appears. They are also determined by a texture projection and any texture that is applied to the geometric object.

The Material Node

Every object has a material node. The node acts like a placeholder for every shader that can be applied to the object. Shaders can alter an object by invoking one or several of nine shader input types. The basic types of shaders that can alter a scene or element's look are:



Input Type	Function
Surface	Determines the basic color of an object as well as the casting of reflected, refracted, and transparency rays. These usually use texture shaders to further define a material.
Volume	Modifies rays as they pass through an object or the scene as a whole. Can simulate effects such as clouds, smoke, and fire.
Environment	Used instead of surface shaders when an eye ray leaves the scene entirely. Defines what will be seen in the background or infinity.
Contour	Used to add contour effects to objects' edges, such as a cartoon effect shader.
Displacement	Alters an object's surface by displacing its points; the resulting bumps are visibly raised and can cast shadows. Displacement is only visible in a rendering.
Shadow	Determines how the light coming from a light source is altered when it is obstructed by an object. Used to define the way an object's shadow is cast, such as its opacity and color.
Photon	Used for global illumination and caustic effects. Without this connection, an object could not receive or transmit photons.
Photon Volume	Used to define a volume for a photon effect.
Bump Map	Determines how and where an object will display a bump map on its surface.

For more information on the different types of shaders available, see *Chapter 2: Shader Basics* on page 37. To learn how the material node is used in the render tree, see *Where to Start?* on page 218.

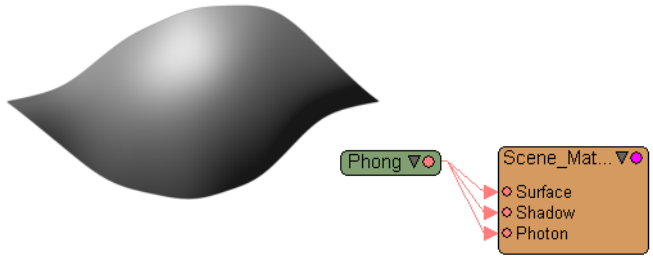
How Surface and Texture Shaders Work Together

Surfaces and textures are usually combined to create an object's look. The surface shader defines the object's surface characteristics such as base color, transparency, refraction, reflectivity, and so on. A texture, on the other hand, applies either a 2D image or a procedural 3D texture onto the surface. The texture doesn't "cover" the surface shader; rather, it simply enhances it.

In the following example, the texture is connected to the surface shader's ambient and diffuse parameters only. The specular value (and decay) set in the surface property page are still maintained after the texture is applied.

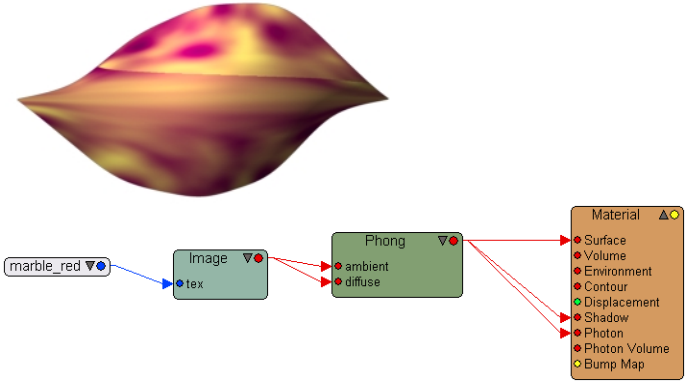
Although a texture shader can be connected directly into the Surface input of the Material node, applying it through a surface shader will give you much more control.

Both surface and texture shaders have a color output that can easily affect an object's surface. In most cases, a texture is connected to the Ambient and Diffuse parameters of a surface shader. The following example illustrates how, by using the render tree, connecting an image texture shader into a surface shader's various inputs can affect the final result.



The material node is simply connected to a Phong surface shader. By default, the surface shader is connected to the surface, shadow, and photon inputs.

A surface shader is automatically connected to the photon parameter so that the material will recognize and render any caustic or global illumination effects when applied to an object.

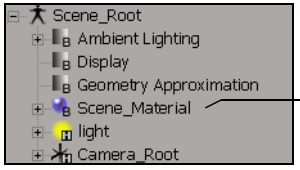


Here, a texture is connected to the Ambient and Diffuse parameters of the surface shader. The surface's Ambient and Diffuse values are overridden and output to the material node's surface input, which makes the object display the texture.

The previous example is, of course, a simple use of a texture with a surface shader. Using the render tree, the explorer, or a property page's connection icon allows you to create an almost endless amount of effects on any one of a material node's connection points.

Default Scene Material

The default material of a scene (Phong) is used for any objects that do not have another surface shader explicitly assigned to them. You can change the default surface shaders by modifying values in their property editors or by dragging and dropping a different surface shader onto the scene root. In the Explorer, the default surface shader appears as a subnode of the scene.



The Default Scene Material is always under the Scene Root level. By default, the scene material is a Phong surface shader.

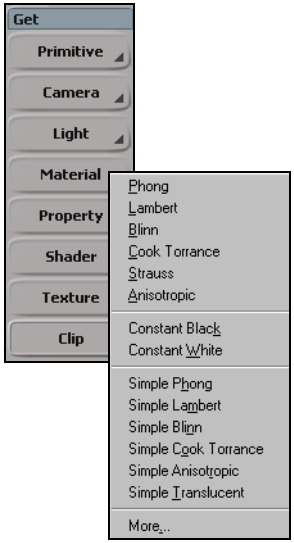
Each object has its own surface. If you delete a surface from an object, the default surface shader is assumed.

For more information on how the default scene material is propagated, see *Hierarchy Propagation* on page 79.

Setting a Scene's Ambient Color

You can also set a scene's ambience. For more information on what a scene's ambience is and how you can edit it, see *Setting a Scene's Ambience* on page 135.

Applying and Editing a Surface Shader



You can apply a multitude of materials or textures simply by clicking a button. One of the most basic and fundamental steps is to apply a surface shader. With a surface shader in place, you can begin applying more shaders to refine your object's look.

To apply a surface shader

1. Select a geometric object, hierarchy, or group of objects.
2. Do one of the following:

   Choose **Get** > **Material** from the toolbar. From there you can choose a basic surface shader (illumination model).

   or

   Choose **Get** > **Material** > **More** to open a browser in which you can select a surface shader or a Preset.

   With either method, the new surface shader and material node are attached to the object and the old material (if any) is removed.

   When selecting a material type from the toolbar, you have the following options (see *Shading Models* on page 68 for a detailed description of each shading model):

Surface Shader	Result
Phong	Applies the default (gray) Phong surface shader.
Lambert	Applies the Lambert shading model, which has no specular value.
Blinn	Applies a Blinn shading model. Similar to Phong.
Cook-Torrance	Produces results that are somewhere between the Blinn and Lambert shading models. It is useful for simulating smooth and reflective objects, such as leather.
Strauss	Applies the Strauss shading model, which is widely used to simulate realistic metal surfaces.
Anisotropic	Applies the anisotropic shading model, which lets you control the direction of the specular.
Constant Black	Applies a constant black illumination surface shader.
Constant White	Applies a constant white illumination surface shader.
More...	Opens a browser in the Material directory and lets you choose from a wide range of illumination models.
Simple surfaces	Apply the same type of surface as described above, except you do not have control over the shader's transparency or reflection.

When using the **Get** > **Material** command, you are actually changing the material node as well as the surface shader. This tool not only replaces the surface shader in a tree, it starts from "scratch" by deleting every shader associated to an object and replaces it with a new material node and the chosen surface shader.

Instead of using the **Get** > **Material** command in the toolbar, you can also apply shaders using the explorer, the connection icon in a property page, the **Shader** button on the Render toolbar, or the render tree. For more information on how to apply shaders using these tools, see *Attaching Shaders to Elements* on page 41.

Regardless of which method you use to apply a surface shader, the shader's property editor opens in a floating window as soon as it is applied to the material node, allowing you to edit its properties.

Displaying a Surface Shader's Property Editor

Property editors contain the various parameters that define the properties of individual elements, such as surfaces. You can display and use multiple property editors at once.

To edit a surface

1. Select an object in the viewport whose surface shader you want to edit.
2. Do one of the following:

From the Render toolbar, choose **Modify** > **Shader**. The Surface property editor opens in a floating window.

or

With the object selected, right-click the **Property** button on the Selection panel on to display the Selection pop-up menu. Click **Surface Shader** to open its property editor.

If a parameter in a property page is replaced with a question mark, this signifies that the parameter is being driven or controlled by another shader.

3. You can now edit any surface parameter from the property page. Click on the various tabs to select different property pages, i.e.: Illumination, Transparency/Reflection, etc.For more information on a surface shader's parameters, see *Surface Shader Basics* on page 67.

When you define the color of an object's surface, you should work with a white light, because colored light sources affect the material's appearance. You can color your light source afterward to achieve the final look of the scene.

For more information on how to edit and delete shaders, see *Applying and Editing Shaders* on page 43.

Surface Shader Basics

A surface shader defines an object's material surface attributes. Although different surface shaders provide different types of properties you can edit, there are a number of basic properties that are defined for most surfaces.

The Material node in SOFTIMAGE|XSI doesn't have quite the same function as it does in SOFTIMAGE|3D. What were called "material" shaders are now more accurately named "surface" shaders. The Material node acts like containers for all of the possible shaders that can be applied to an object, much like the full-screen Material Editor in SOFTIMAGE|3D.

This section describes how to work with the following surface-shader properties:

- Shading models
- Illumination
- Reflectivity
- Transparency
- Refraction
- Colors
- Propagation in a hierarchy

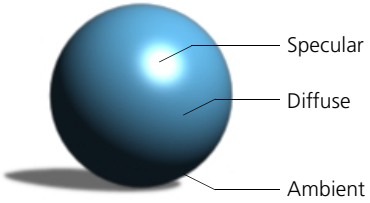
Uses for Surface Shaders

You can also do the following with surface shaders:

- Save a surface shader as a preset and apply it to another object in your scene—see *Saving and Loading Shader Presets* on page 30.
- Animate surface-shader properties in the same way as any other property. For more information about animation in general, see *Animating with Keys* in the *Animating* guide.
- Detach a surface shader and replace it by another type of surface. For information on attaching and detaching shaders, see *Attaching Shaders to Elements* on page 41.

Surface Illumination

You can create a very specific color for an object by defining its specular, diffuse, and ambient colors separately in the surface-shader property page.



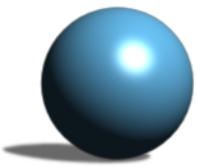
- The *diffuse* color is the color that the light scatters equally in all directions so that the surface appears to have the same brightness from all viewing angles. It usually contributes the most to an object's overall appearance, and it can be considered the main, or basic, color of the surface.
- The *specular* color is the color of shiny highlights on the surface. It is usually set to white or to a brighter shade of the diffuse color. Specular highlights are visible only on Phong-, Blinn-, anisotropic- and Cook-Torrance–shaded surfaces (see *Shading Models* on page 68). The size of the highlight depends on the defined Specular Decay value.
- The *ambient* color appears on areas that are shielded from light but still visible due to an ambient light, which is a non-directional light that pervades the entire scene. As an alternative to ambient light and color, you can use global illumination, which is more realistic but takes longer to calculate (see *Global Illumination & Caustics* on page 153).

For more realistic results, you may want to make the Ambient color identical to the Diffuse color and let the Scene's Ambient color control the ambient value.

Shading Models

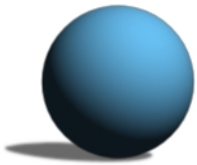
Shading models determine how an object's surface appears under different lighting conditions. Several kinds of mathematical models can be used to calculate the shading. Each shading model processes the relation of surface normals to the light source to create a particular shading effect.

Each of these shading models are available from a toolbar (**Get** > **Material**).



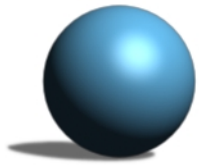
Phong

The Phong shading model uses ambient, diffuse, and specular colors. It reads the surface normals' orientation and interpolates between them to create an appearance of smooth shading. It also processes the relation between normals, the light, and the camera's point of view to create a specular highlight. The result is a smoothly shaded object with diffuse and ambient areas of illumination on its surface and a specular highlight so that the object appears shiny like a billiard ball or plastic. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with a Phong shader.



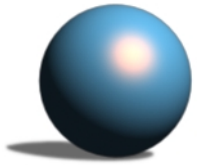
Lambert

The Lambert shading model uses the ambient and diffuse colors to create a matte surface with no specular highlights. It interpolates between normals of adjacent surface triangles so that the shading changes progressively, creating a matte surface. The result is a smoothly shaded object, like an egg or a ping-pong ball. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with a Lambert shader.



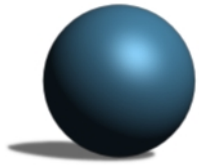
Blinn

The Blinn shading model uses diffuse, ambient, and specular color, as well as a refractive index for calculating the specular highlight. This shading model produces results that are virtually identical to the Phong shading model except that the shape of the specular highlight reflects the actual lighting more accurately when there is a high angle of incidence between the camera and the light. This shading model is useful for rough or sharp edges and simulating a metal surface. The specular highlight also appears brighter than the Phong model. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with a Blinn shader.



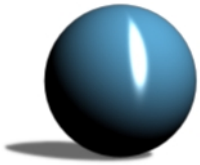
Cook-Torrance

The Cook-Torrance shading model uses diffuse, ambient, and specular color, as well as a refractive index used to calculate the specular highlight. It reads the surface normals' orientation and interpolates between them to create an appearance of smooth shading. It also processes the relation between normals, the light, and the camera's point of view to create a specular highlight. This shading model produces results that are somewhere between the Blinn and Lambert shading models and is useful for simulating smooth and reflective objects, such as leather. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with a Cook-Torrance shader. Because this shading model is more complex to calculate, it takes longer to render than the other shading models.



Strauss

The Strauss shading model simply uses a diffuse color to simulate a metal surface. The surface's specular is defined with smoothness and "metalness" parameters that control the diffuse to specular ratio as well as reflectivity and highlights. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with a Strauss shader.



Anisotropic

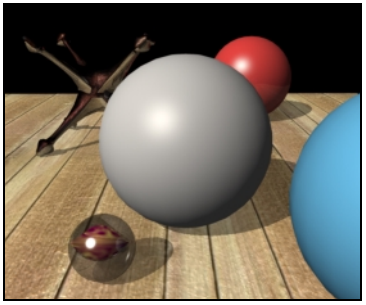
The anisotropic shading model—sometimes called Ward—simulates a glossy surface using an ambient, diffuse, and a glossy color. To create a "brushed" effect—such as brushed aluminum—it is possible to define the specular color's orientation based on the object's surface orientation. The specular is calculated using UV coordinates. Reflectivity, transparency, refraction, and texture can be applied to an object shaded with an anisotropic shader.

Constant

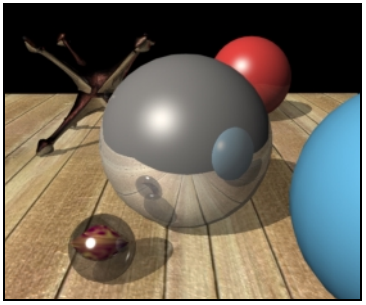
The Constant shading model uses only the diffuse color. It ignores the orientation of surface normals. All the object's surface triangles are considered to have the same orientation and be the same distance from the light. It yields an object whose surface appears to have no shading at all, like a paper cutout. This can be useful when you want to add static blur to an object so that there is no specular or ambient light. It also provides good support for textures because there are no attributes to interfere with the texture's definitions.

Reflectivity

You can define an object's surface to be reflective from the surface shader's property page. Reflectivity controls whether other objects in the scene appear as reflections on the object's surface. Reflectivity values usually range from 0 to 1, with 0 representing no reflectivity and 1 representing complete reflectivity, giving the object a perfectly mirrored surface.



No reflectivity in ball's material



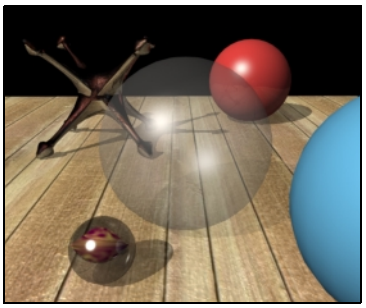
Reflectivity (35%)

The visibility of the other surface parameters, such as those related to diffuse, ambient, and specular areas of illumination, decrease as the reflectivity value increases. You can compensate for this by drastically raising their values; for example, a specular highlight that was 1 on a non-reflective object could be reset to 1000. If an object's material is fully reflective, its other material attributes are not visible at all, so reflectivity should always be set to a value less than 1.

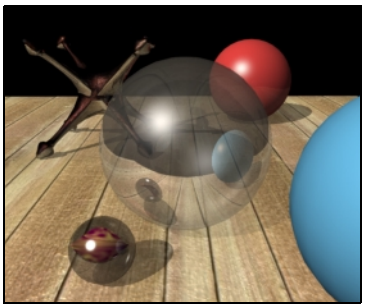
Although a parameter's slider bar can be moved to only a maximum value of 1, you are able to increase the values by typing in a number higher than the slider bar permits.

Transparency

You can define a surface shader's transparency by setting any Transparency options in its property editor. Transparency values usually range from 0 to 1, with a value of 0 representing no transparency and 1 making the object completely transparent. For a transparent surface such as glass, a value of 0.9 is more convincing.



Transparency (75%)

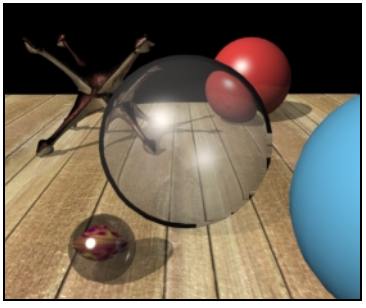


Transparency (70%) with reflection (30%)

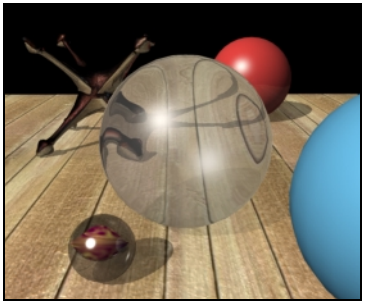
As with reflectivity, transparency affects the visibility of an object's other surface attributes. You can compensate for this by increasing their values, such as changing specular color values that were around 1 on an opaque object up to 10 or higher on a transparent object.

Refraction

When transparency is incorporated into an object's surface definition, you can also define the refraction value. Refraction is the bending of light rays as they pass from one transparent medium to another, such as from air through glass or water.



Refraction value of 0.9



Refraction value of 1.1

When you use refraction and reflections, you may need to increase the Ray Depth value in the Rendering Options property editor (refer to *Global Illumination & Caustics* on page 153). The use of refraction and reflections may increase rendering time.

You can set the refraction from a surface shader's property editor. The default value is 1, which represents the density of air. This value allows light rays to pass straight through a transparent surface without deviation. If you increase this value (greater than 1), it simulates light passing into a denser material such as from air to glass: the higher the index value, the more the light rays deviate. If you decrease the index below 1, light rays bend in the opposite direction, simulating light passing from air into an even less dense material such as a vacuum.

Refractive-index values usually vary between 0 and 2, but you can type in higher values as needed.

You can enter the actual, physical refraction values for real-life matter, and the behavior of light passing through it will be simulated.

The following table lists some common transparent materials and their refractive value.

Material	Refractive Value
Air	1.00
Ice	1.31
Water	1.33
Acetone	1.36
Alcohol	1.39
Glass	1.50–1.89
Polystyrene	1.55
Emerald	1.57
Topaz	1.61
Ruby/Sapphire	1.77
Crystal	2.00
Diamond	2.42

Using the soft_material Shader

Any scene imported from SOFTIMAGE|3D uses the soft_material surface shader to define its materials and how they interact with any associated textures.

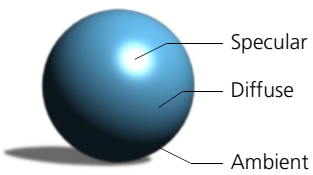
A shader imported into SOFTIMAGE|XSI from SOFTIMAGE|3D retains the name it was given prior to importing. For our purposes, we will continue to call it the soft_material shader.

The soft_material shader combines texturing capabilities within one monolithic shader so you can set parameters for blending 2D textures with a material from within the soft_material shader.

You cannot build a render tree from the soft_material shader; that is, you cannot add additional texture nodes or tool nodes to augment the effect. If you wish to do this, replace soft_material with one of the new surface shaders (**Get** > **Material**) and build from there as described in *Applying and Editing Shaders* on page 43.

The picture file and the current material on an object are blended together. One picture can be used to create several texture maps at the same time, and different picture files used for different textures can be applied to the same object's material both globally and locally. The texture-image editing options of the soft_material shader are found in the property editor under the tab named after the image you wish to edit.

Blending the Material with a Picture File



Using the color sliders in the property editor, you can control the amount of blending between the picture file and the object's current material. Values range from 0 to 1: if the value is set to 0, only the material is visible; if set to 1, only the picture file is visible.

- If you choose the **Ambient** slider, you can control the intensity of the picture file on the object's ambient area of illumination. Values range from above 0 to 1. If the value is set to 0, the picture file is replaced by the object's material (global or current local).
- If you choose the **Diffuse** slider, you can control the intensity of the picture file on the object's diffuse area of illumination. Values range from above 0 to 1. If the value is set to 0, the picture file is replaced by the object's material (global or current local).
- If you choose the **Specular** slider, you can control the intensity of the picture file on the object's specular area of illumination. Values range from above 0 to 1; the value may be set higher to compensate for reflectivity or transparency. If the value is set at 0, the picture file is replaced by the object's material (global or current local).

Blending Materials and Textures

The alpha channel and RGB intensity values of the picture file's pixels can also be used as a mask to further control the various texture-mapping effects. You can mask using either the RGB values or the alpha channel (transparency) of the picture file as a blending factor.

The blending parameters are found on the Map tab of the soft_material shader's property editor.

The **Mask** parameter uses the SOFTIMAGE|3D mask blending method:

1. **Without**
2. **Alpha**
3. **RGB Intensity**
4. **RGB Modulate**

The **Component** parameter defines which component of the texture is used to blend.

1	**Alpha:** When using the Alpha Channel Mask option, the picture file is blended with the object's current material according to the different alpha-channel values of the picture pixels. When the alpha-channel value is high (white), the color of the picture file is visible; when the alpha-channel value is low (black), the object's current material is visible.
2	**RGB Intensity:** When using the RGB Intensity Mask, the RGB color intensity of the picture file is used as a blending factor. The picture file is blended with the object's current material according to the different RGB intensity values of the picture pixels. When the RGB intensity is high (white), the color of the picture file is visible; on low-intensity pixels (black), the object's current material is visible.

Defining Colors

Every surface shader and most of the texture shaders allow you to manipulate color. You can define a color using the color sliders or the color editor. These controls appear in various shader property editors, including the Light property editor and the property editors of some 2D and 3D textures.

When you define the color of an object's material, you should work with a white light, since colored light sources affect the appearance. You can color your light source afterward to achieve the final look of the scene.

Color Models

The additive color system is used to define colors for materials as well as for lights; this is the system used by the renderer to calculate colors. You can toggle between three different color models: RGB (red, green, and blue—the default), HLS (hue, lightness, and saturation), and HSV (hue, saturation, and value).

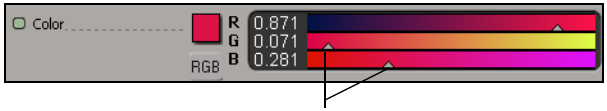
Although all three color models are displaying the same gray, their RGB, HLS and HSV values differ.

- **RGB:** This color model defines color by mixing light rather than by mixing pigments. The primary hues from which all colors are derived from are red, green, and blue. These three hues mixed together at their highest values (1) make white, while an absence of all hues (0) gives black. When you create a color, you must define the values of each red, green, and blue component to get exactly the desired hue.
- **HLS:** In this color model, hue refers to the position of the color in the spectrum. Lightness is the amount of white mixed in a color, creating the difference between a pure red and pink. Saturation is the purity of the color, creating the difference between a pure red and a dusty rose. Low saturation means that there is more gray in the color.
- **HSV:** This color model—also known as HSB—defines the hue and saturation like the HLS model. Value is similar to Lightness as in HLS; however, a value of 1 represents a pure color when Saturation is 1, while a Lightness of 1 yields white no matter what the Saturation. In both systems, 0 is black.

Defining Colors with Sliders

First, select the color model that best suits your needs. Below the color box, you can click on the channel name to toggle between RGB, HLS, and HSV.

To define a color using the color sliders, click and drag the sliders to change the strength of each channel independently, or type a numerical value directly in the space provided.



Drag the sliders to interactively adjust a color

To move all three sliders at once, hold the **Ctrl** key down while moving the sliders with the cursor. For fine-tuning a single color value, hold down the **Shift** key while moving the cursor over a color slider bar.

You can quickly copy a color by dragging and dropping the color chip from one color box to another—from the same property page or another one. This is especially useful when you want to match ambient or diffuse colors among objects.

As you begin working with the color sliders, you will notice that they are interactive; that is, each color slider immediately updates to show you a gradient of the color you have selected. The slider shows you what color will result if you move the slider to a new position.

The specific channel corresponding to a slider depends on the color channel you selected; for example, if you selected RGB, the sliders correspond to the red, green, and blue channels. The values range between 0 and 1.

When you are defining materials, you can type values greater than 1 to compensate for transparency and reflectivity.

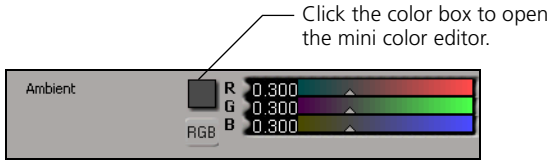
Defining Colors with the Color Editors

To define colors more precisely or choose a color from any viewport, you can use the color editor or the mini color editor. In the color editor, you can create and modify colors for your materials and lights. This can be useful for creating specific palettes to be applied consistently to a group of materials.

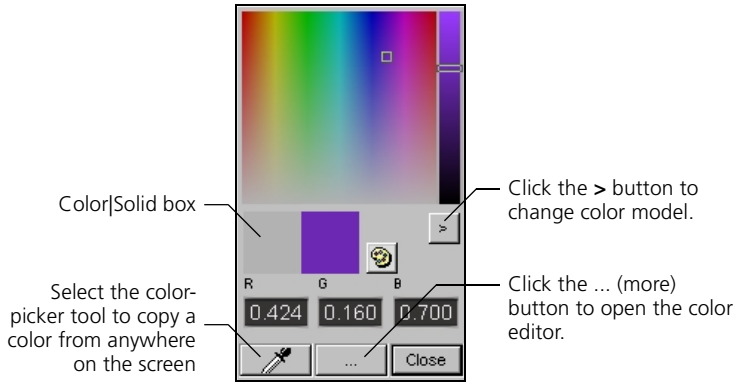
You can also use the color-picker tool to pick a color displayed in any viewport, including Rotoscope views or the Image Clip editor.

To select a color using the mini color editor

1. Click the color box next to the color sliders in any shader's property editor.



2. A palette of colors appears in the mini color editor. The current color appears in the **Color|Solid** box.



- To select a color, click one of the colors inside the color spectrum. The color appears on the right side of the color views box, while the previous color remains on the left side to use as a reference.
- To modify a color, use the slider bar at the right of the mini color editor or enter a numeric value. In addition, you can also change the color model from RGB, to HLS, to HSV. The corresponding Hue, Sat (saturation), and Lum (luminance) values are numerically displayed above. See *Hierarchy Propagation* on page 79 for more information.
- Select the color-picker tool to pick a color from anywhere in the color editor or from one of the viewports. The color-picker tool takes the color seen on the screen rather than the true color of the objects.

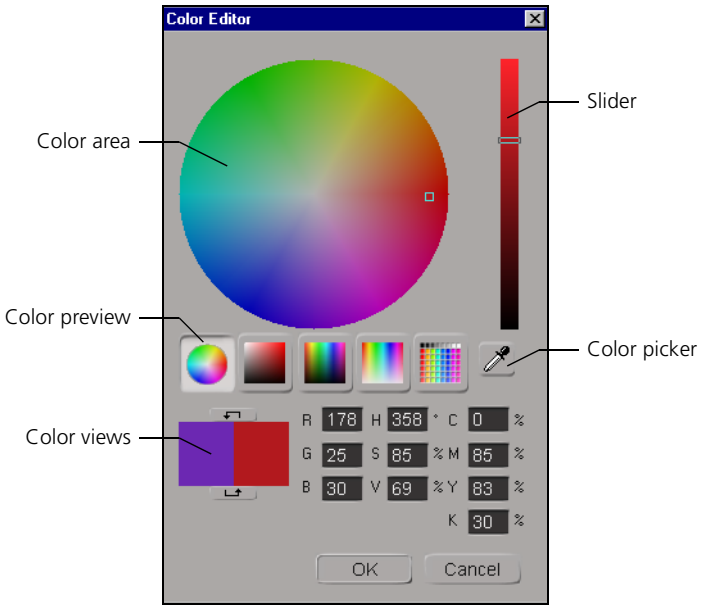
If you wish to match a color of an object outside of SOFTIMAGE|XSI, you can load an image file into a viewport using the Flipbook. Simply load a file into flipbook and select it for viewing. Then you can pick the desired color once the file is visible in a viewport. For more information, see the *Fundamentals* guide.

- Click on the (...) button to open the color editor. The color editor allows you to accurately pinpoint and create custom colors.

3. **Close** the color editor to use the color you created.

To select a color using the color editor

1. From the property editor, click the color box to open the mini color editor.
2. From the mini color editor, click the (...) button.



- To select a color, click within the color area. The color area can be defined by selecting a color-preview mode. Each mode offers a variety of spectrums and hues. Once a color is selected, the color appears on the right side of the color-views box, while the previous color remains on the left side to use as a reference.
- To modify a color, use the slider bar at the right of the color editor or enter a numeric value in any one of the RGB, HSV, or CMYK fields. As a color is modified, the corresponding RGB, HSV, and CMYK values are updated. See *Hierarchy Propagation* on page 79 for more information on the RGB, HSV, and HLS color channels.
- Select the color-picker tool to pick a color from anywhere in the color editor or from one of the viewports. The color-picker tool will take the color seen on the screen rather than the true color of the objects. This tool can be especially useful when trying to match a color in a viewport set to Flipbook or the Image Clip editor.

Hierarchy Propagation

Propagation is the method in which an object "passes down" its material to its children, or how children in a hierarchy inherit their parents' attributes.

Propagation works in a unique way. Every object has three possible levels of propagation:

- **Scene Root propagation**—a property passed on from the scene's root. This is also the default property that an object reverts to if no other material is defined for that object.
- **Branch propagation**—a property passed from parent to child in a hierarchy. For more information on how to create parent-child groups, see *Hierarchies* on page 139 of the *Fundamentals* guide.
- **Local propagation**—a property that is unique to a single object, hierarchy, or group of objects. The material is not being inherited from another object. Local propagation takes priority over every other type of propagation.

Although an object can have all three types of propagation at once, it only displays one type at a time. When SOFTIMAGE|XSI is looking for a propagated material to display, it always selects a local material over a branch material, and a branch material over a scene root material. For more information on the Default Scene Material, see *Default Scene Material* on page 64

You can use either the explorer or the Schematic view to see how propagation affects different objects in different hierarchies.

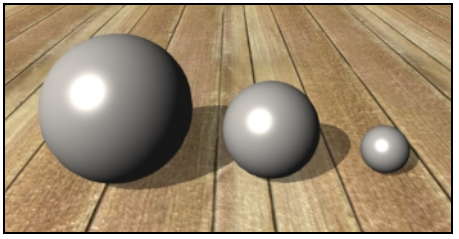
A material is only propagated if the parent is branch-selected before the material is applied. Otherwise, a local material is applied and the object's children do not inherit any properties.

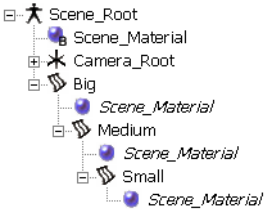
When you are about to modify or delete a shared material, you are warned and asked for confirmation.

The following examples illustrate how a material and/or texture is propagated.

Default Propagation

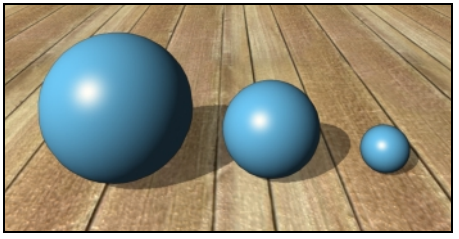
In this sphere hierarchy, each sphere is parented to a smaller one beside it. Because no other surface shader has been applied, each sphere inherits the default scene material.

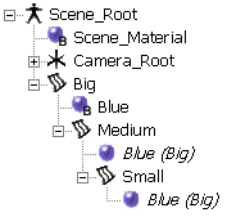


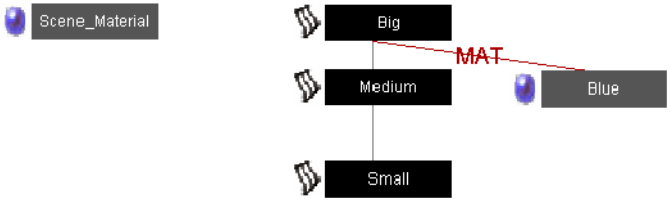


Branch Propagation

The Big sphere was branch-selected and given a Blue surface; the remaining spheres inherit the blue surface because the Big sphere was branch-selected. Notice how the Blue surface is shown in italics in the explorer: this denotes that the Blue surface is being inherited from the Big sphere. The propagating object is always listed in parentheses (Big). In addition, the parent propagating the surface is noted with a B symbol on the material nodes: this signifies that the material is being propagated in Branch mode.

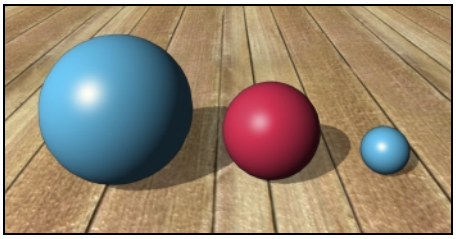


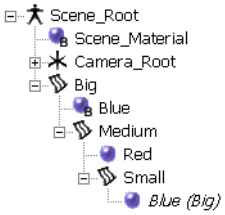


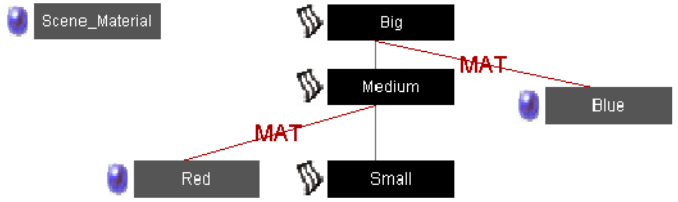


Local Application

The Medium sphere was single-selected and given a Red surface. This applies a local surface shader that is applied to the selected object only (and none of its children). Notice how the Medium sphere's child (Small sphere) still displays the Blue surface propagated from the Big sphere.







Unlike SOFTIMAGE|3D behavior, SOFTIMAGE|XSI allows you to apply a material in local mode. This material will not be propagated unless it is applied in branch mode.

Applying a Local Material

You can apply a local material to any object regardless if it is in a branch or is the parent of other objects. A local material affects only the selected object(s).

To apply a local material

1. Select an object or several objects to which you wish to apply a local material.
2. Choose **Get** > **Material** from any toolbar. A single material becomes attached to each individual object. Each material is independent of the other.

   If you are about to modify or delete a shared material, you are warned and asked for confirmation,

You can right-click the Surface tab of any property editor to make the surface shader local.

Applying a Branch Material

A branch material will be passed on to each child that does not have a local material already applied to it. If a local material is applied, it will be displayed above all others.

To apply a branch material

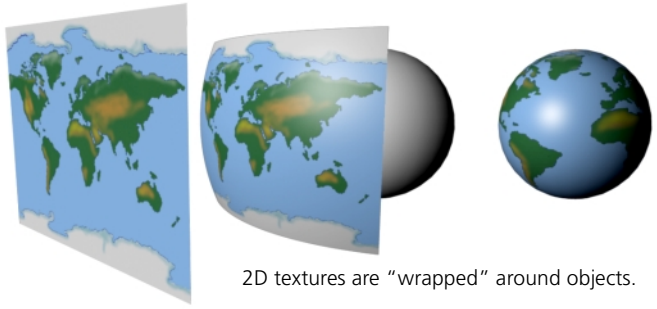
1. Choose a branch (middle-click) to which you wish to apply a branch material.
2. Click **Get** > **Material** from the toolbar to apply a material and surface shader, or use any other method described in *Applying and Editing a Surface Shader* on page 65.

   If you are about to modify or delete a shared material, you are warned and asked for confirmation.

Texture Basics

Texture shaders apply an image or "feel" to an object's surface. These shaders use images and bump maps to simulate a texture, while a surface shader affects an object's color, transparency, reflectivity, and so on. Textures fall into two simple categories: 3D (procedural) and 2D (images).

2D textures are simply 2D images that you can apply to—or "wrap" around—the surface of an object. Applying textures to your objects can give them an increased sense of realism.



2D textures are "wrapped" around objects.

Unlike 2D textures, 3D textures are projected "into" objects rather than onto them. Often defined by fractals, 3D textures are applied inside objects as well as outside. Common uses of 3D textures are wood, clouds. and marble.

3D textures are applied throughout an object

2D and 3D textures can be used in a number of different ways. You can:

- Apply textures to an entire object.
- On a polygon mesh object, you can apply texture shaders locally to individual polygons or polygon clusters.
- Attach textures to specific properties of a surface shader. This lets you use a texture to vary a parameter, such as the diffuse or specular color, over every point on a surface.
- Use 2D textures as bump, displacement, reflectivity, and environment maps.
- Use textures in combination with shadow-casting lights to create complex patterns of colored light and shadow on other objects in a scene.
- Animate all texture properties.

2D Textures and Images

2D textures are images that can be wrapped around an object's surface, much like a piece of paper that's wrapped around an object. To use a 2D texture, you start with any type of picture file (PIC, GIF, TIFF, PSD, etc.). These can be picture files imported from SOFTIMAGE|3D, imported scanned photos, or imported picture files. A picture file is any file containing data that describes all the pixels in an image, RGB, or RGBA data.

Image Clips

Every time you select a picture to use as a texture, an image source file and a clip instance of the image—called an *image clip*—are created. For more information on image sources and image-clip files, see *Using Image Sources & Image Clips* on page 95.

Several image clips can be created from the source file. An instance or clip of the image source file is created every time it is used. You can then edit each one slightly according to your needs. For example, if you are using the same picture file for both your surface texture and your bump map, you can apply a slight blur onto the picture file without affecting the picture being used as a surface texture. For more information on editing image clips, see *Using Image Sources & Image Clips* on page 95.

3D Textures

3D textures are solid "procedural" textures. Procedural means that they are mathematically calculated at points in 3D space—not wrapped around a 2D surface like 2D textures. This means they can be used to represent substances having internal structure, like the rings and knots of wood or the veins in rocks or marble.

Commonly Used Shaders

There are some shaders and tool shaders that are commonly used to create or manipulate surfaces and/or textures. Most of these shaders can be found in the Texture folder and sorted by function (Mixer, Conversion, Image Processing, etc.) The following is a list of these shaders and a brief description:

Shader	Description
Image	Allows you to apply any image to a surface (or any other parameter) and edit it. It also permits you use the image as a bump map.
Gradient	Can be used on its own or in conjunction with another texture to create a color/texture gradient.
3D_turbulence 3D_fractal 3D_Warp	Any one of these shaders can be used to create a pattern or disrupt a texture's coordinates in a logical or random manner.
Kaleidoscope	Creates a series of mirror images of the texture. Although 2 or 3 mirror images may not be too exciting, try using a pattern of 16 or more.

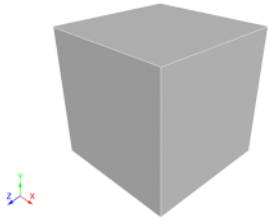
Texture Projection

2D textures are applied to an object's surface by a process known as *projection*. Projection basically decides *how* to apply a texture onto an object or group of objects. When a 2D texture is projected onto an object, a correspondence between pixels in the texture and points on the object's surface is calculated. This process can be accomplished by projecting the texture onto the object's XY, XZ, or YZ axes, or by mapping cylindrically, spherically, or using an object's UV directions to create different effects.

For information on how to define a texture projection, see *Creating a Texture Projection* on page 107.

Planar Projection

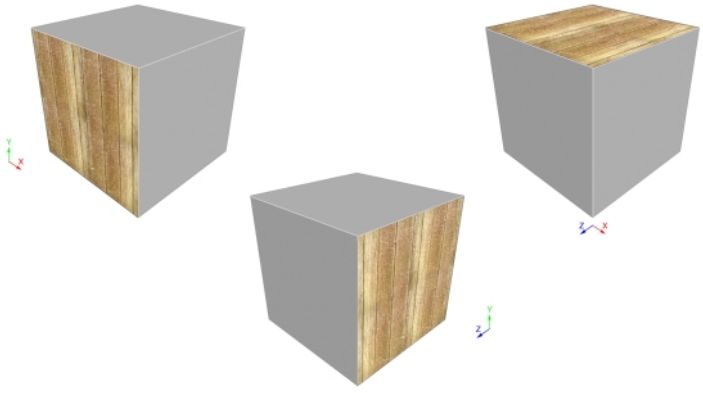
For XY, XZ, and YZ mapping, imagine an image being projected onto a cube. If the picture file is mapped onto XY coordinates, its pixels are projected accurately onto the XY surface plane of the object. The projection plane is (by default) one pixel smaller than the surface plane, therefore no "streaking" or distortion will occur on the object's other planes.



Picture file used

Above: Cube (left) with XYZ orientation.

Below: Resulting projection with (from left to right) XY, XZ, and YZ mapping methods.



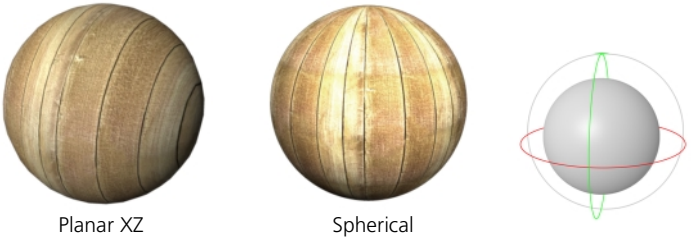
Cylindrical and Spherical Projection

If the picture file is mapped cylindrically, it is projected as if it is a cylinder wrapped.

Planar XY Cylindrical

Polygon mesh cylinder with XY projection map (left) and Cylindrical map (right). The picture file was tiled by setting the number of repeats.

When mapped spherically, the picture file surrounds the object and covers the entire surface with some distortion.



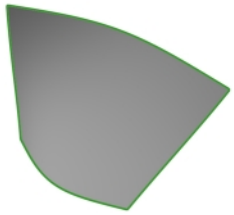
Planar XZ Spherical

Polygon mesh sphere with XZ projection map (left) and Spherical map (right). The texture was blended at 50% to show shading on the object.

UV Projection

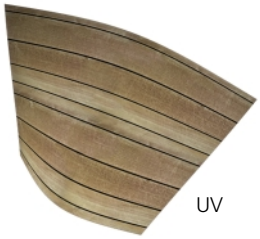
UV mapping is the most versatile mapping method, and can be used on NURBS objects as well as polygon mesh. It behaves like a rubber skin stretched over the object's surface. The points of the object correspond exactly to a particular coordinate point in the texture. This lets you map a texture accurately to the object's geometry. Even when you deform an object, its texture follows the object's geometry.

Reparameterizing a UV surface will reparameterize the UV projection as well.



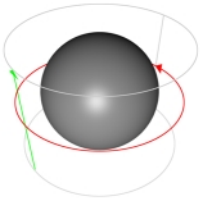
Planar XZ

A NURBS surface (above) with XZ map (above, right) and UV map (below, right). Pattern follows the contours of the object with UV map.



UV

Texture Support Object

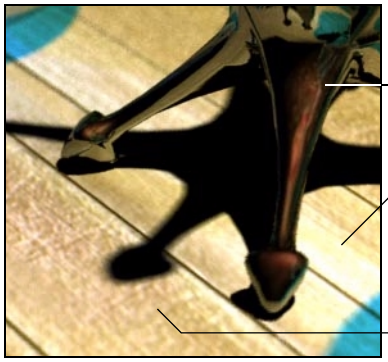


As you apply texture projections, a graphical representation of the projection is added to the object(s) in the viewports. This object is called the texture support object. It is displayed in dark-green wireframe and uses red and light green to show the U and V directions. For more information on how to use and manipulate the texture support object, see *Using the Texture Support Object* on page 111.

If the texture support object is not visible, press **j** to turn its visibility on.

Applying a Texture

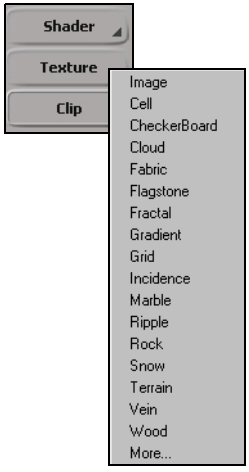
The easiest way to apply a texture to an object is to use the Texture button on the Render toolbar. This tool will apply a surface shader (if none exists) and a selected texture shader to the selected object or objects.



You can choose where you want to apply a texture. By default, a texture is applied to the ambient and diffuse parameters of a surface shader when the Texture button is used.

Three Basic Texturing Methods

The following three texturing methods can all accomplish the same results. Although some methods may be better suited for advanced users, others can quickly accomplish basic texturing with a few mouse clicks.



Using the Texture Button

The fastest and most thorough way to texture an object's surface is to use the Texture button The appropriate mixing shaders are attached to the surface and texture shaders.

To texture using the Texture button

1. Select an object to texture.

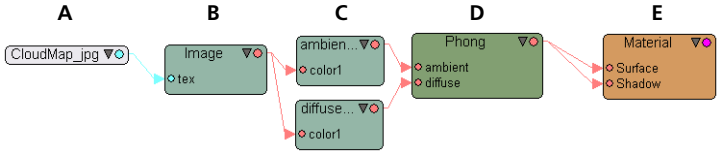
There are several presets available at the click of a mouse:

Texture Shader	Effect
Image	Applies an image over a surface using a user-defined texture space. You can also use the texture image to create a bump map.
Cell	Creates a 3D cell-like surface pattern on an object's surface.
Checkerboard	Creates a definable checkerboard pattern.
Cloud	Creates a 3D cloud pattern texture on an object's surface.
Fabric	Creates a 3D highly definable fabric pattern on an object's surface.

Texture Shader	Effect
Flagstone	Creates a flagstone/stained-glass pattern on the surface of an object.
Fractal	Creates a procedural fractal texture on the surface of an object. This texture is often used to drive another texture's parameter; i.e.: Transparency.
Gradient	Creates a definable color gradient with editable colors and adjustable patterns.
Grid	Creates a grid pattern on the surface of an object.
Incidence	Lets you control the shading of an object's edges in relation to its angle from the camera. Possible applications are an X-ray look, varnish, or glass transparency.
Marble	Creates a 3D marble texture on an object's surface. The marble's age, flaws, and colors are customizable.
Ripple	Creates a water-drop-like ripple on the surface of an object.
Rock	Creates a highly definable rock texture on an object's surface.
Snow	Creates a 3D snow texture on an object. Often used to simulate snow-covered objects.
Terrain	Creates a highly definable terrain pattern on the surface of an object.
Vein	Creates a 3D vein texture on the surface of an object.
Wood	Creates a highly definable 3D wood texture on the surface of an object. The wood's age, rings, and colors are customizable.
More...	Opens the browser and allows you to select any type of texture preset.

2. Click the **Texture** button on the Render toolbar, and a menu of your texturing choices appears beside the toolbar.
3. You can easily apply more textures to the same object by repeating this step.
4. Select a texture to apply to your object. The selected texture is connected to the Ambient and Diffuse parameters of the surface shader.

Adding a texture with the Texture command creates a render tree that looks like this:



Mix 8 Colors shader tools (C) are used to blend a texture and/or image with each parameter. This gives you much more control over each surface attribute. For example, you could choose to blend the texture with the Phong's Specular and Reflection parameters but blend another texture with the Diffuse and Ambient. The above shader nodes perform the following basic tasks:

Shader	Function
A	The actual image clip (PIC, JPG, or other type of image file). For more information on Image Clips, see *Using Image Sources & Image Clips* on page 95.
B	The Image shader that holds the image clip and defines a texture projection for it.
C	The Mix 8 Color tool shader that blends the image file with any parameter and/or any other image. A separate mix node is used for each surface parameter (one for the Ambient and another for the Diffuse) and named for the parameter it is controlling.
D	The default surface shader Phong.
E	The Material node that accepts almost any shader to define an object's look.

Clicking the Texture button once a texture has already been applied to a surface will continuously add textures to the surface shader via the Mix 8 Colors shader.

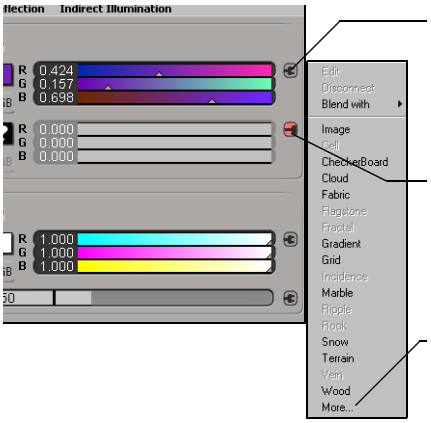
Although the Mix 8 Colors shader only accepts eight color inputs, there is no limit to the number of textures or images you can apply to a shader. Subsequent textures are blended using a second mixing shader.

You can easily apply more textures to the same object by repeating step 2 above.

For more information on how to use the Mix 8 Colors shader, see *Blending Textures* on page 99.

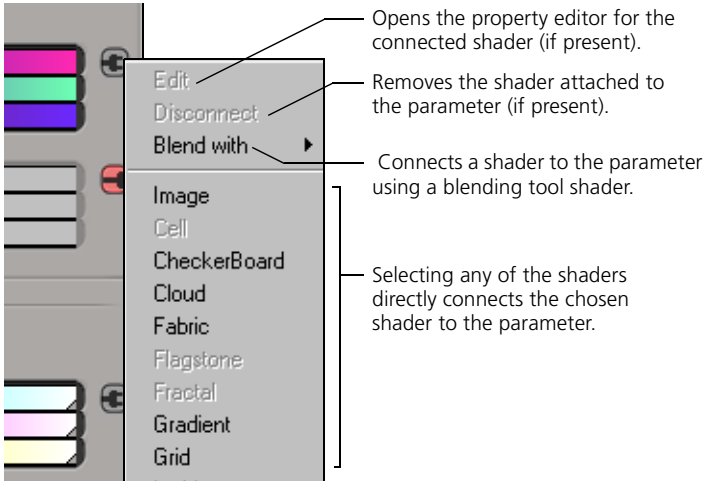
Connecting Shaders Using the Property Editor

You can also texture an object using the connection icons within each shader's property editor. Every parameter has its own connection icon that opens a pop-up menu listing common shaders for that parameter.



To texture using the Connection icon

1. Select an object you wish to texture, or texture further, and open its Surface or Texture shader property editor (**Modify** > **Shader**).
2. Select the connection icon at the end of the parameter you would like to attach a shader to, as shown above. A drop-down menu appears.
3. From the drop-down menu you can either edit, disconnect, blend, or attach a shader as shown.



If a parameter is already mapped to a shader, its connection icon is replaced by a red icon. To connect a new shader, or disconnect the present shader, open the pop-up menu by right-clicking the icon. A left-click will automatically open the property editor for the connected shader.

Texturing with the Render Tree

When you apply a texture to an object by using the render tree, you are creating everything from "scratch." While the Texture button from the Render toolbar automatically connects a series of mixing shaders between a material and texture, working in the render tree requires that you construct the tree manually.

To texture using the render tree

1. Open a Render Tree view in a viewport or floating window.
2. Select an object and click **Update** on the render tree command bar.
3. If necessary, apply a surface shader to the material node (e.g.: Phong) using the **Get** > **Material** command from the toolbar.
4. From the **Nodes** > **Texture** drop-down menu in the render tree command bar or from a browser, select or drag and drop a texturing shader into the render tree work area and connect it to the surface shader as desired.

For more information on how to work with the render tree, see *Chapter 9: The Render Tree* on page 205.

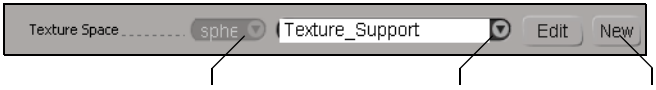
Drag'n'Drop

You can also drag and drop a surface or texture shader from a browser onto your object in a viewport. Although this method is quick and simple, it does not allow for very much control over individual parameters.

Defining a Texture Projection

Before the texture can be "attached" to an object or hierarchy, you must define a texture projection for the texture to be replaced relative to the scene.

Regardless of the number or types of textures you wish to apply, each texture must have a texture space defined to be applied to your selection. Every texture shader allows you to create a texture space on the fly using the Texture Space controls in the Texture Shader property page.



This drop-down menu defines which object or objects the texture space will be applied to.

Click on the arrow to open a drop-down menu of all existing texture spaces.

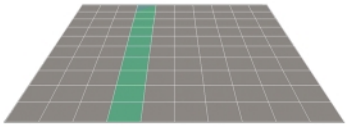
Click the New button to create a new texture space.

For more information on the different types of texture projections, see *Texture Projection* on page 85. Also, see *Blending Textures* on page 99 and *Creating a Texture Projection* on page 107 to learn how to manually create, edit, combine, and manipulate texture spaces.

Applying a Local Texture

On a polygon mesh object, you can also assign textures locally to groups of surface polygons or globally to all the polygons of the polygon mesh. Since textures can be applied in both ways, each object has a global and local (where applicable) material "list," which, in turn, can have its own texture list.

Polygon mesh object with global texture shader



Model with selected polygons

Darker texture applied to selected polygons

Click the Polygon button to select polygons on a polygon mesh.

To apply local textures to polygons

1. Click the selection button on the main command area and then click the **Polygon** button to activate polygon selection.
2. Select one or more polygons on an object. The selected polygons are highlighted.
3. Choose **Get** > **Texture** > **More** from a toolbar. A cluster is automatically created. The browser opens at the location of the shader library defined in your user-preferences file.
4. Select the texture preset you want and click OK to apply the texture to the selected polygons.
5. Set the parameters for the new texture as desired.

To access a local material's shaders from the explorer, you must open the object's node > **Polygon Mesh** > **Clusters** > **Polygon**.

Copying Textures

It is more than likely that you will, at one point or another, copy an object's texture(s) to another object. This can easily be done by saving a texture as a preset and then loading it onto another object or objects.

The best way to save texture- and surface-shader information is to save all the shaders from either the material node or from the surface shader property editor.

To save a texture preset

1. Open the surface-shader property editor of the object whose texture you wish to save.

2. Click the Save icon on the upper-right corner of the property page.
3. Select a path to save to, and name the preset. Click OK.

It is important to save the texture preset from the surface shader's property page, since this will save everything attached to the surface shader itself. Otherwise, shaders that are connected to a surface shader's transparency or reflection inputs may not be loaded correctly.

To load a texture preset

1. Open the surface-shader property editor of the object whose texture you wish to load from another location.
2. Click the Load icon on the upper-right corner of the property page to open the Load Preset browser.
3. Navigate to the location you saved a preset, select the file, and click OK. The texture preset is applied to the object as it was saved from the original.

Using Image Sources & Image Clips

Every time you select an image to use as a texture or for rotoscopy, an image clip and an image source of the selected image is created, regardless of its format (PIC, GIF, JPEG, TIFF, PSD, etc.).

Image Sources

A source image is the original image file copied into a SOFTIMAGE|XSI folder. It is defined as read-only and is saved with your scene in the Sources folder of the Scene Root. It does not have to be reloaded when you re-open your scene.

Image Clips

An image clip is a copy or instance of an image source file. Each time you use an image source, an image clip of it is created. You can have as many clips of the same source as you wish. You can then edit, crop, or even blur the image clip without affecting the original source image.

Clips are useful because they allow you to create different representations of the same texture image (source), such as five different blur levels of the source image. Also, clips are memory efficient because the source is only loaded once, regardless of the number of clips.

Loading/Creating Sources and Clips

You can load as many image files into SOFTIMAGE|XSI as you wish and as you need them. For example, before beginning a scene you could load all of your images into the Source folder so they are readily available to you in clip form.

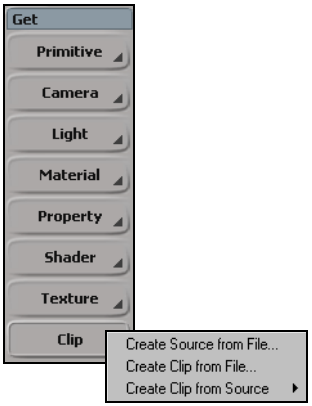
You can access the Sources or Clips folder from the explorer (set the scope on Project). To view your clips, set a viewport to the Image Clip Editor. From the Image Clip editor command bar, select **Clips** > <***filename***>.

To load or create a source from the Render toolbar

1. Once a scene has been created or opened, choose **Get** > **Clip** from the Render toolbar.
2. From the pop-up menu, choose **Create Source from File**. A browser opens, from which you can select an image file. Click OK.

You can choose from a wide variety of image files, not just .PIC files. Click the arrow beside the File Types text box to see a list of the supported graphic file formats.

3. Once you have selected a file, the Image Source property editor appears. By default, the source file takes the same name as the image file. You can rename the source by typing a new name in the **Source Name** field.
4. Close the property editor. The source file has now been created for use in your SOFTIMAGE|XSI scene. It is now available to use in the Image shader property editor.

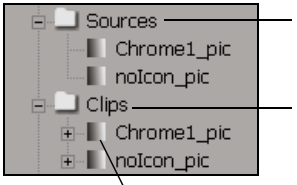


To create sources and clips from the Render toolbar

1. Once a scene has been created or opened, select **Get** > **Clip** from the Render toolbar. A menu offers you one of three choices:

If you select...	
Create Source from File:	A browser opens from which you can select an image file. Once a file is selected, an image source is created for it as well as the image source property editor, where you can rename the source, is opened.
Create Clip from File:	A browser opens from which you can select an image file. Once a file is selected, an image source is created for it as well as an image clip immediately afterward. The image clip property editor appears. You can use it to edit or manipulate the newly created image clip
Create Clip from Source:	Another sub-menu opens prompting you to select an existing source file to be copied as an image clip. Once a source is selected the image clip property editor appears. You can use it to edit or manipulate the newly created image clip.

Once you use a source image as a texture, a clip (or instance) of the image source is created every time it is used.

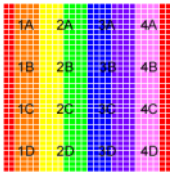


The Sources folder contains the original image file. Source files are read-only.

The Clips folder contains copies or instances of an image in the Sources folder. Clips can be modified or edited without affecting the original source image.

Clicking on a clip's icon will open the image-clip editor. From there you can perform color corrections, crops, blurs and more.

noIcon.pic

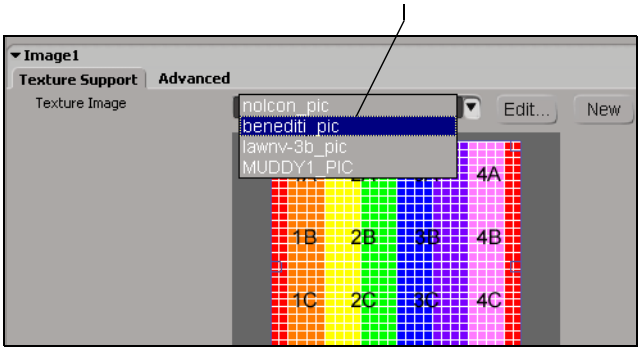


Every image file used is stored in the Sources and Clips folders of the Scene Root. Images can be dragged and dropped from a browser to any viewport or a Sources folder. Sources are only copied into the Clips folder once they are used.

The only picture file included with SOFTIMAGE|XSI is the default noIcon pic—a color bar (shown left). This image replaces the question mark image of SOFTIMAGE|3D.

When an image is loaded into the Sources folder (as described above), it is available in the Texture Image drop-down menu in the property editor of the Image shader as an image selection.

All defined image sources and image clips are available in the Image shader's drop-down menu.



In the drop-down menu, Sources are listed first and clips are listed with an indent. You can produce an endless amount of image clips per source. Each clip of an image source retains the source's name but has a number appended to it.

Selecting an image clip makes the clip current, while selecting a source creates a new image clip and makes it current.

Editing a Clip

Because an image source file is read-only, you cannot modify it. Clips, on the other hand, can be edited as much as you want. The easiest way to edit a clip is to use the Image Clip Editor.

Within the Image Clip property editor, you can perform a variety of commands that can make one image clip very different from another clip that originates from the same source.

To open the Clip Property editor from the Image shader

1. Open the Image shader that uses the clip you wish to edit. The Image shader can be opened from the render tree (double-click the shader), or from a property editor (using the Up, Prev. or Next buttons).
2. Once the Image shader property editor is open, right-click the thumbnail image and select **Clip Properties** from the pop-up menu. This opens the Clip property editor.

For more information on how to use the specific tools in the Clip property editor, refer to the online help in the property editor by clicking the question-mark icon (left).

Selecting the Image Clip editor opens it in the selected viewport.

To open the Image Clip property editor from a viewport

1. From the Views viewport menu, select Image Clip editor. The viewport's view is replaced with a blank work area.
2. From the Image Clip editor's command bar, select an existing clip from the Clips drop-down menu. Any source that has been defined as a clip is available from this menu. Once selected, the image clip appears in the workspace.
3. Open the Clip properties editor and select **File** > **Clip Properties** from the command bar. The Image Clip property editor appears in a floating window.
4. To Crop or Uncrop the image clip, select the desired tool from the **View** menu on the command bar.

For more information on how to use multi-resolution texturing (MipMapping) for an image clip, see *Image Pyramid Mapping (Rendering)* on page 129.

Viewing an Animated Image Clip

Because not all textures are static images, you can view an image clip from within the Image shader's property editor as you would any other type of texture. You can use the playback controls in the property editor to play, pause, stop, and loop an animated image or a sequence of images.

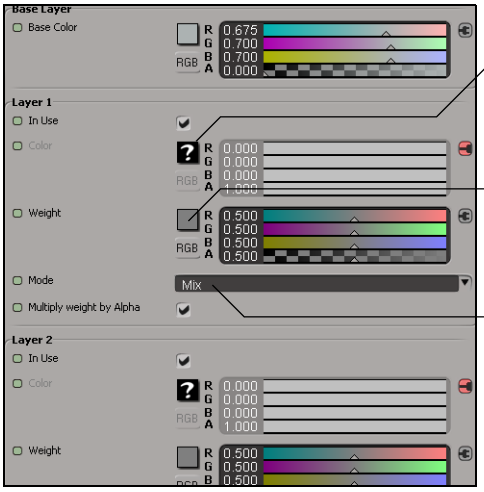
For more information on how to create an animated texture, see *Creating a Sequence Texture* on page 125.

Removing a Texture Shader

Select a texture shader and press **Delete**. When deleting a texture, you are not removing the texture projection from the object. To delete the texture support object, select it and press Delete.

Blending Textures

Whether you are blending two or a dozen textures, a shader mixing tool is used to blend colors, shaders, and images together. Each texture can have its own weight and mixing mode assigned to it.



The question mark signifies that this parameter is already connected or being driven by a color or image input or shader.

The Weight parameter lets you assign a weight variable—or strength—to the input color.

You can select the mixing mode the shader will use.

The Mix 8 Colors
This shader uses a base color, at least one input, a mix mode, and a weight value to mix colors and/or textures with one another.

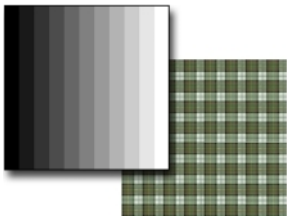
The Mix 8 Colors shader allows a wide range of control over how your colors and/or textures are blended. All of the SOFTIMAGE|3D blending methods are still available. Here's how they translate:

SOFTIMAGE|3D Mode	SOFTIMAGE|XSI Equivalent
Without	Mix (Alpha option not checked)
Alpha	Mix (Alpha option checked)
RGB Intensity	RGB Intensity
RGB Modulate	Hide/Reveal

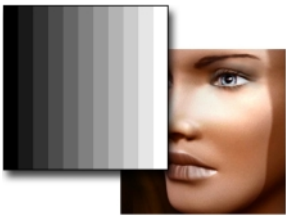
In SOFTIMAGE|XSI, the default is equivalent to SOFTIMAGE|3D's Alpha option.

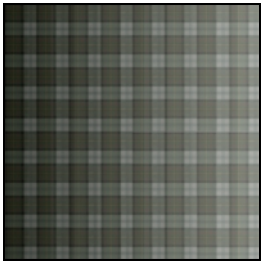
The following list defines each mixing mode and how it reacts according to a gradient, a repetitive texture, and an image.

These two images(left) illustrate how the mix_8_colors shader reacts to a "white-to-black" gradient and a repetitive, colored texture.



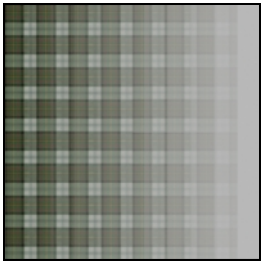
These two images (right) illustrate how the mix_8-colors shader reacts to a "white-to-black" gradient and a texture map.





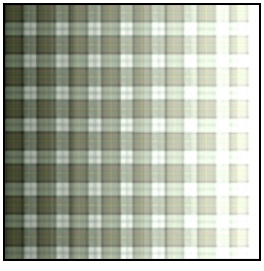
Mix
Calculates a simple average of each layer's pixel values. This is the default mode for the mixing shaders.

Tip: Try scrubbing the Weight slider to obtain a cross-fade.

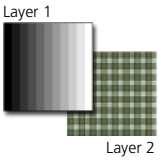


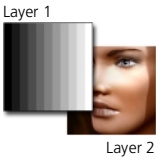
Add Compensate (Blend)
Makes the brighter sections of the second layer gradually (and proportionally) screen out the darker sections of the first layer. This mode compensates the first layer prior to adding, so the resulting color never exceeds 1 (100%).

Tip: You can use this mode to blend two high-intensity images without losing detail.

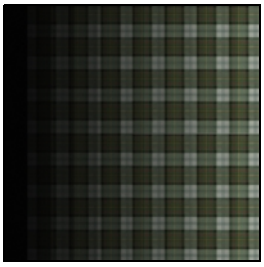


Add
The Add mode simply adds the first and second layers' pixel values. The result is not clipped at 1 (100%).

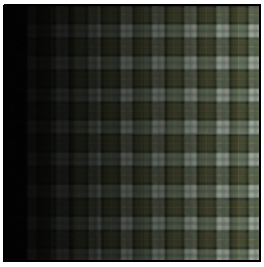




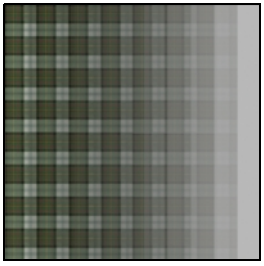
Add (Bound)
Performs a similar calculation to the Add mode, but all values are clipped at 1 (100%).



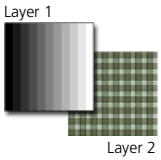
Hide/Reveal (Multiply)
Multiplies all of the layers' pixel values. Using a 1 (white) Weight value, the second layer's pure white pixels reveals the color values of the first layer. If the Weight is set to 0 (black), a black image is displayed. Because this mode often produces dark results, consider the full range of the layers. Good, intense blends can be achieved by using two light-colored sources.

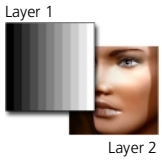


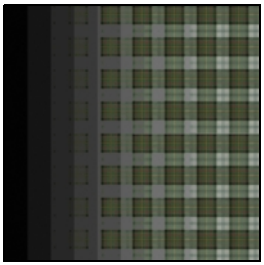
Hide/Reveal (Bound)
Performs a similar calculation to the Hide/Reveal (multiply), except that all values are clipped at 1 (100%).



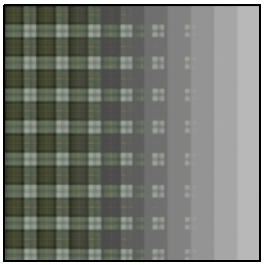
RGB Intensity
Calculates a proportional average of each layer's pixel values. The average is dependent on the brightness of each layer.





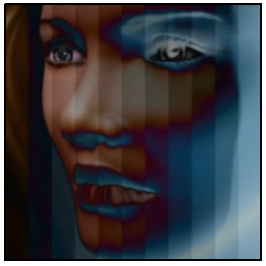


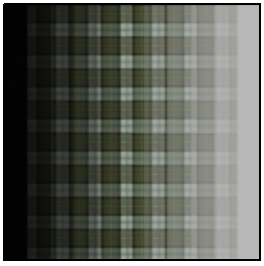
Darker
Sets the darkest color (between the first and second layer) as the result color. Pixels darker than the blend color are not replaced.



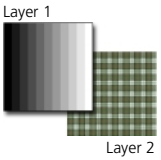
Lighter
Sets the brightest color (between the first and second layer) as the result color. Pixels darker than the blend color are not replaced.

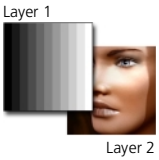
Difference
Takes the brightest color (between the defined color and the base color) and subtracts it from each of the color channel's values.

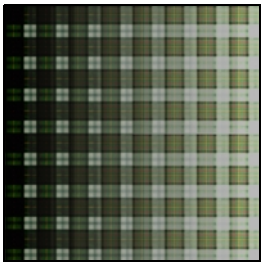




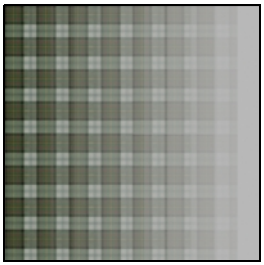
Hard Light
Causes the first layer's bright pixels to brighten the next layer's light pixels. Also, the first layer's dark pixels dims the second layer's darkest pixels. Visually, each layer is pushing the other layer's extreme pixel values; i.e.: the darks become darker and the lighter parts become brighter.







Hue Offset
Makes the darkest pixels of the second layer dim the gamma and boost the contrast of the first layer. Meanwhile, the brighter pixels of the second layer boost the gamma and dim the contrast of the first layer. The brighter the layer, the more it offsets the first layer's hue. Visually, you're controlling both the gamma and the contrast values simultaneously on different parts of the image.



Screen
Multiplies each color channel's values with the inverse of the defined color and the base color. The Screen function results in a lighter color as though the original color has been faded.

Soft Light
Mixes the first and second layers, then reveals the resulting color with the brightest pixels of the second layer.

Chapter 4 Advanced Materials & Textures

Creating a Texture Projection

Although you can quickly create a texture projection on the fly as you apply textures, you can also create a texture projection (or many!) before applying a texture.

How an object's texture is applied is almost entirely dependent on its texture projection. An object's texture projection is visible in any viewport, represented by a specific icon—called the texture support object—depending on the type of projection that you have defined.

For more information on the texture control object, see *Using the Texture Support Object* on page 111.

To see texture support objects on SOFTIMAGE|3D scenes, you may need to unhide them by pressing **Shift+h**.

You can define an object's texture projection before applying a texture or while you are applying the texture.

By default, any texture projection created is hidden. You can unhide the texture-support object by pressing **j** or **Shift+j** to unhide all of a scene's texture support objects.

To define a texture-space projection before applying a texture

1. Select the object, objects, group, branch, or model to which you want to assign a texture projection.
2. From a toolbar, select **Get** > **Property** > **Texture Support** > **Create Texture Control** to open the Create Texture Support property editor.
3. In this property editor, you can assign a texture projection (spherical, planar, etc.) or a projection plane or define a UV projection. The following lists your texture-projection options:

Option	Result
Planar (XY, XZ, YZ)	Defines a standard planar projection and allows you to select the projection plane.
Cylindrical	Defines a standard cylindrical projection and allows you to select the projection plane.
Spherical	Defines a standard spherical projection.
Spatial	Defines a three-dimensional UVW texture projection.
Camera	Defines the camera's "view" as the texture projection.
UV	Defines a standard UV texture projection.

4. You can then choose to parent the projection to the object by checking the **Parent to Object** box. This will constrain the projection to the object or group to which you apply the projection.
5. If you want, you can name the texture projection by typing a name into the **Texture Space Name** field. This is useful to differentiate between several types of projections assigned to a single object or group.

If planar XY, XZ, or YZ projection is used, the outline is displayed as a box; if cylindrical mapping is used, it is displayed as a cylinder; and if spherical mapping is used, it is displayed as a sphere.

If Camera projection is used, you will be prompted to select a camera from a viewport or explorer.

If the defined texture projections aren't visible in a viewport, you can make them appear by selecting **Show** > **Texture** supports from the viewport menu bar or toggling their visibility by pressing the **j** key.

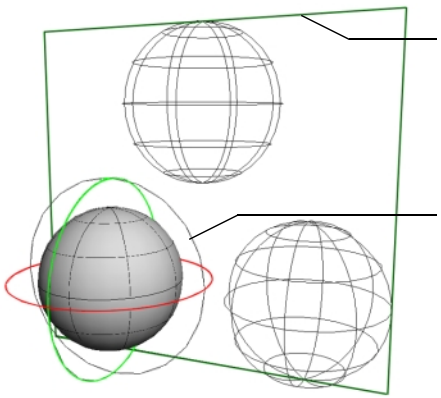
To define a texture-space projection while applying a texture

1. Select the object, objects, group, branch or model to which you want to assign a texture projection and apply a texture using the **Get** > **Texture** button from a toolbar.
2. Select any texture shader from the drop-down menu. The shader's property editor opens.
3. From this property editor, click the **New** button beside the Texture Space text box to define a new texture projection. A drop-down menu allows you to select any of the texture projections listed previously. Select **Advanced** to open the Create Texture Support property page. For more information on how to use the Create Texture Support property page, see page 107.
4. Once a texture projection is defined, it is automatically made active and the texture (if any) is applied using that projection.

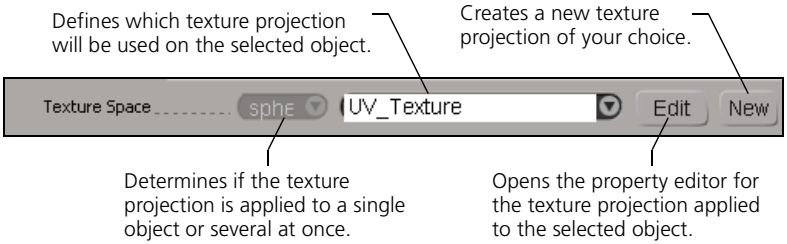
Simultaneous Texture Support

An object can have more than one type of texture projection assigned to it and display as many texture supports as you wish at any one time. For example, a single sphere in a group of a five spheres can have a texture projection assigned specifically to it or to the entire group.

Which or how many projections it displays is up to you. You can even animate which projections it will use at what frame.



You can define the texture projection of each object from the same texture-projection window. Use the drop-down menu to the left of the Texture projection's name in a property page to define which object a projection is assigned to.



Applying a Texture Support to a Group or Branch

How a group or hierarchy of objects has been selected determines what kind of texture support is applied to the group or hierarchy.

To apply multiple texture supports

1. Multiple-select each object to which you want to apply an individual texture projection.
2. Select **Get** > **Property** > **Texture Support** > **Create Texture Control** from a toolbar. The Create Texture Support property editor opens.

3. Select a **Projection Type** and a **Texture Support Name**.
4. Click OK to apply the new texture projections on to each of the selected objects.

To apply a single texture support to a hierarchy or group

1. Branch- or model-select (middle-click or right-click) the group or hierarchy you wish to apply a single texture projection to.
2. Select **Get** > **Property** > **Texture Support** > **Create Texture Control** from a toolbar. The Create Texture Support property editor opens.
3. Select a **Projection Type** and a **Texture Support Name**.
4. Click OK to apply the new, single texture projection onto all of the selected objects.

If the defined texture projections aren't visible in a viewport, you can make them appear by selecting **Show** > **Texture** supports from the viewport menu bar or toggling their visibility by pressing the **j** key.

Freezing a Texture Projection

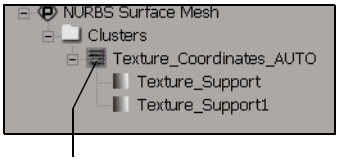
Once you have scaled, rotated, or translated a texture projection to your liking, you can freeze it—permanently or temporarily. Freezing a texture projection is the equivalent of freezing the operator stack, and it's useful if you want to avoid accidentally editing or moving your texture projection.

As the name suggests, a temporary freeze isn't permanent and can be altered at any time using the same property page you opened to originally freeze it.

To permanently freeze a texture projection

1. From any viewport, select the texture-support icon to freeze.
2. Press Delete to remove the icon and permanently freeze the texture's position and scaling.

To temporarily freeze a texture projection

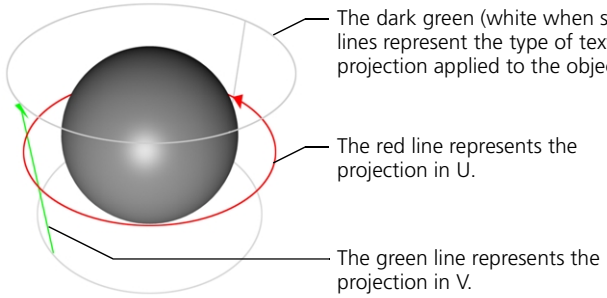


Texture Control Object icon

1. From any viewport, select an object with a texture support you wish to temporarily freeze.
2. From the Selection panel, choose **Explore** > **Texture Supports**. A floating explorer window opens showing the texture support object's node.
3. Click the texture-control object's icon to open its property page.
4. Select the Texture Support tab and click the **Mute** checkbox to temporarily freeze the texture projection.
5. Click the checkbox again to unfreeze the projection.

Using the Texture Support Object

You can interactively manipulate a texture's projection in a viewport using a 3D manipulator called a *texture-support object*. A texture-support object is a graphic representation outlining how the texture is projected, allowing you to visualize where the texture falls on the object.



In this example, the texture-support object represents a Cylindrical texture projection on the sphere.

Texture supports let you control the way the texture is projected on the object. For example, if you choose a spherical object and a spherical mapping type, the texture support wraps around the sphere and defines a spherical shape from which the texture is projected.

You can also perform transformations on texture supports that alter the way a texture's image is projected onto an object. The texture is constrained to the support so that when the support is rotated and scaled, the texture is also affected. For example, if the texture support is scaled up, the texture image is also enlarged.

The texture's center of projection (calculated according to the center of the texture support) is automatically displayed and can be translated anywhere within the texture support, allowing you to further distort the texture's image on the object.

The center of the texture support is calculated according to the center of the object's bounding box.

To display the texture support

1. Select the object whose method of texture projection you want to change.
2. Press the **j** key to display the texture-support object. All texture-support objects are hidden by default. If the support object still isn't visible, try selecting the **Show** > **Texture Controls** option from a viewport's Show menu.

To hide all of a scene's texture control objects, press **Ctrl+j** or **Shift+j**. These commands hide and display (respectively) the control objects in all viewports.

Manipulating Texture Support Object

In addition to viewing a texture projection's orientation and position, you are able to manipulate it (scale, rotate, or translate) as you would any object in a viewport.

By default, an object's texture projection is constrained to the object; otherwise, animated objects would move through space without their projection.

If you have the Texture Support display active in a viewport, you can see the results immediately. You can also view the results in the render region.

Scaling, Rotating, and Translating the Texture Support Object

You can interactively scale (**x** supra key), rotate (**c** supra key), or translate (**v** supra key) a texture-support object—hence the texture itself—directly in a viewport using the Scale, Rotate, and Translate commands in the Transform panel. These transformations can also be animated. You can also align textures to one or more selected polygons of the object to which the texture is applied.

Copying and Connecting Texture Support

You can duplicate a texture-support object as you would any element in a viewport. But what do you do with the new texture projection? You can easily connect any texture projection to an object so its projection can be used anywhere it is connected.

To connect a texture support to an object

1. In any viewport, select the object you wish to connect to a texture support.
2. Select **Get** > **Property** > **Texture Support** > **Connect to Txt Ctrl.**
3. Enter a name for the new texture space in the dialog box.
4. From any viewport, select a texture-support object to attach to the object.

Connecting a Texture Support to a Camera

A useful trick to get a texture projection looking just right on an object's surface is to project the texture from the camera. This is done by applying the texture-support object to the camera and then freezing the texture.

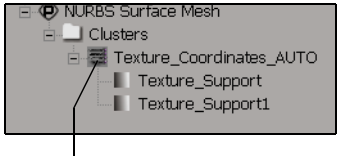
To project a texture from a camera

1. From any viewport, select an object you wish to texture.
2. Select **Get** > **Property** > **Texture Support** > **Create Texture Control** from a toolbar.
3. From the Create Texture Support property page, select a Camera projection type and click OK.
4. You are prompted to select a camera from either a viewport or the explorer. The texture is projected from the camera you select.

Now the texture will be recomputed every time the camera moves.

Once you apply the texture's projection to your liking, or you want to stop the texture from following the camera, you can freeze the projection.

To freeze a camera projection



Texture Control Object icon

1. Select the object with the camera projection applied to it.
2. From the Selection panel, choose **Explore** > **Texture Supports**. A floating explorer window opens, showing the texture-support object's node.
3. Click the texture-control object's icon to open its property page.
4. Select the Camera Texture tab and click the **Mute** checkbox to temporarily freeze the texture-control object.

Constraining a Texture Support to a Bounding Box

The **Get** > **Property** > **Texture Support** > **Constrain to BBox** command allows you to constrain any object to an existing texture projection that has been applied to a group, branch, or tree hierarchy.

This command cannot be applied to a multi-selection.

Swapping the U and V Directions

By selecting the **Swap UV** option in a texture-projection property editor, you can exchange the U and V directions of the texture coordinates. The texture is rotated and flipped.

Removing a Texture Support Object

To remove a texture support object, simply select it and press Delete.

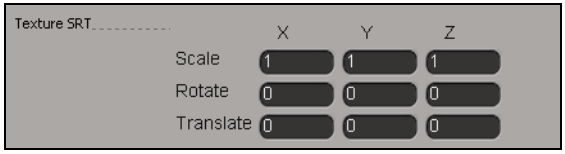
Note that if a texture was applied to the support object, it will be frozen to the object as it was applied at the time the support object was deleted. The texture is still editable, however.

Manipulating Textures

Once a texture is projected onto an object, you will most likely want to make a few adjustments to it. These adjustments can usually be made by scaling, rotating, or translating the texture-control object. But if you want to have a little more "hands-on" control, you can use the texture shader's main page and its **Advanced** page in its property editor. The main property page usually contains the standard SRT controls, and the Advanced property page lets you control tiling, repetitions, and remapping.

Scaling, Rotating, and Translating Textures

You can scale, rotate, and translate a texture from any texture-shader property editor. The Scale, Rotate, and Translate text boxes reflect any changes you may have made in the Scale, Rotate, or Translate commands in the Transform panel.



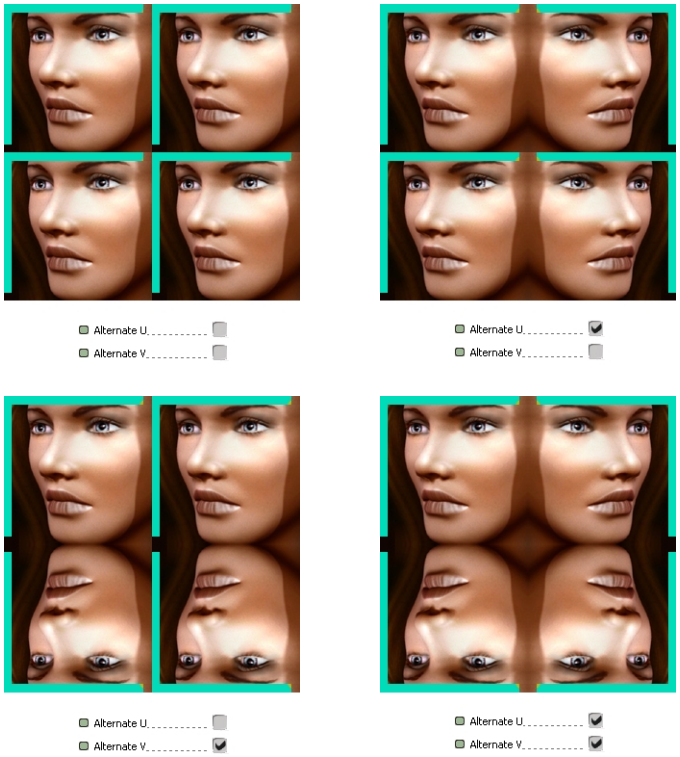
The Scale, Rotate, and Translate options scale and offset using the upper-left corner of the picture file as a reference point instead of the texture's "center."

The Scale, Rotate, and Translate options may be useful when using the Repeat option (under the **Repeats/Cropping** tab) to tile a texture. For example, say you repeat the texture to be tiled 12 times but you want to increase that number. Using Scale, Rotate, and Translate, you can quickly adjust the size and placement of the tiled texture map from within the Texture property editor.

Tiling a Texture

Most textures provide repeat parameters that let you create a wallpaper-like pattern by tiling the texture repeatedly.

When a picture is repeated, its edges should be absolutely symmetrical; if not, you will see seams.



If you have a Texture display type active in a viewport, you can see the results immediately. You can also view the results in the render region.

If you are working with a 3D texture, you can also define its tiling in the Z axis.

Repeating a Texture

Texture repeats are special because a texture's repetition lines can be defined and displayed as well as interactively edited in any viewport.

You can define the number of repetitions in X, Y, and Z. The Repeat parameter contains the repetition factor within X, Y, and Z. For example, a value of six will shrink the texture so it fits onto the object six times.

Steps and Textures

The Steps parameter is mainly used to control the smoothness or jaggedness of a bump map. The slider lets you define the U, V, and Z steps of the bump texture. For more information on how to control a bump map, see *Creating Bump Maps* on page 117.

Implicit and Explicit Texture Projection

The difference between implicit and explicit texture projection is subtle, but it can make a big difference when it comes to rendering.

When using an explicit texture projection, the texture shader doesn't convert the projection, it uses it as is. Conversely, an implicit texture projection rebuilds the texture coordinates for every pixel.

UV projection is only available through explicit texture projection. Implicit texture projection is capable of XY, YZ, XZ, spherical, and cylindrical projection.

Explicit Projection

Explicit texture projection uses the object's UV data and is faster to render because it uses the projection that has already been made available on the object's defined coordinates, whether they be UV, Cylindrical, Spherical, etc. Texture projections generated from the toolbar (**Get** > **Property** > **Texture Support** > **Create Texture Support**) are explicit texture projections.

Explicit texture projection is more friendly to use since you can see the texture-control object, or texture manipulator, in the viewports; hence, it is more flexible.

Implicit Projection

Implicit texture projection is slower to render but produces more reliable and accurate results for spherical and cylindrical projections.

You would use implicit texture projection to obtain a slightly better rendering performance (1%) and a better overall result for spherical and cylindrical projections over a low-polygon-count model. For example, when mapping a texture onto a sphere (using either spherical or cylindrical projection), implicit texturing produces more accurate results at the spheres' poles than does explicit projection.

Although more mathematically accurate, implicit projection only has a Scale, Rotate, Translate (SRT) box in the texture property page, which makes it a little trickier to manipulate than using a manipulator icon.

Implicit texture projection is slower to render because it performs its own computation (based on a predefined projection model; i.e., spherical, planar, etc.) at each pixel, as opposed to using predefined UV data like explicit projection. For example, if you're creating a predefined implicit projection like spherical, planar, or cylindrical, rendering will be slower because the shader has to calculate the projection and the final result. Implicit texture projection is performed by the mental ray **mib_texture_vector** shader.

Creating Bump Maps

Although real surfaces can be perfectly smooth, you are more likely to encounter surfaces with flaws, bumps, and ridges. One of the most basic ways to create these types of "bumps" on a surface is to use a bump map.

On an object, bump maps are used as regularly as textures. They are necessary when you want your object to look realistic. Although bump maps do not change the geometry of an object, they do affect the geometry of a surface shader. This affects the shading, giving the illusion of a pattern being embossed on the surface. Bump maps, like texture maps, require a texture space.

To create a bump map

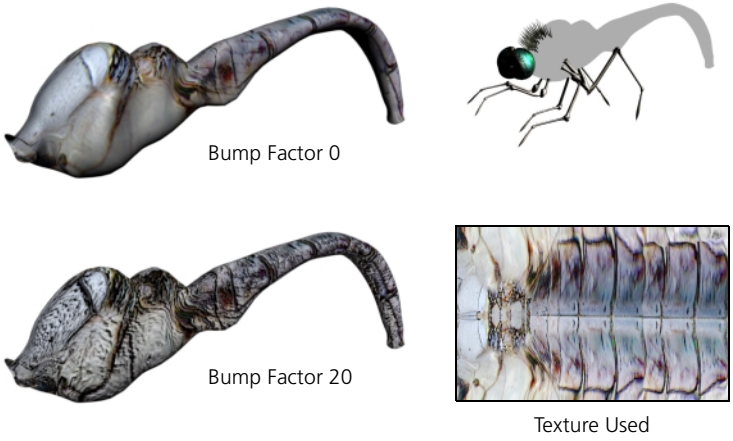
1. Select the object(s) to which you want to apply a bump map.
2. Apply a texture shader to the object, group or hierarchy by clicking **Get** > **Texture**. The selected texture's property editor appears. If you chose the Image file, select an image clip to use as a bump map. You can choose from available image clips or create a new one by selecting **New**.
3. Define a texture space for the texture (by selecting **New**), or use an existing texture space from the Texture Space drop-down menu.

It isn't necessary for an image clip and a bump map to have the same texture space. If you want one to use a different texture space, you must use another shader and create another image clip of it.

If you have a render region open, you will notice that the texture has been applied to your object.

4. To activate the bump map, click **Enable Bump Map** in the texture shader's property editor.
5. Determine the strength of the bumping by adjusting the **Bump Map Factor** parameter. The higher the factor, the "bumpier" the texture will appear. A positive value bumps outward, and a negative value bumps inward.

6. Close the texture shader's property editor.



The dragonfly's body is textured with a bug-like texture, but the bump mapping is set to 0, which makes the effect not apparent. In the image above, the bump mapping is enabled and the Bump Factor is set to 20. A positive value bumps outward and a negative one bumps inward.

Applying a Bump Map without Texture

There may be instances when you wish to apply a bump map without using the bump map's texture or color on the surface of the object. The easiest way to do this is to negate the color values of the texture you are using to bump map. This can be done easily from the Mix 8 Colors shader.

To set a textureless bump map

1. Apply an image as a bump map as described in steps 1 to 5.
2. From the texture shader property page, click the **Up** button to bring you to the next shader up in the shader hierarchy. The texture shader's property page should be replaced with the Mix 8 Colors property page. If not, you can open the Mix 8 Colors shader from the render tree view by double-clicking its shader node.
3. In the Mix 8 Colors property editor, each image connected to the surface shader is represented by a Layer. Set the RGB Weight values to 0; this negates the color information for a color layer and leaves the bump information.
4. Repeat steps 2 and 3 for each Mix 8 Colors shader the bump map is connected to.

Mixing Textures for a Bump Map

To create a more refined or detailed bump map, you may want to combine two or more bump maps onto the same object. This can be done as easily as adding a new texture.

1. Select the object to which you will apply another bump map.
2. Choose **Get** > **Texture** from the render toolbar. The texture shader property editor will appear.
3. Edit the texture as you wish. If you selected the Image texture shader, select an image clip you will use as a bump map.
4. Define a texture projection for the texture (UV is recommended to better follow the surface's curves). The new texture shader is added to your object using the Mix 8 Colors mixing shader. Because the shader can support multiple textures, the original bump map(s) and textures are not disturbed.
5. To activate the new texture's bump map, click **Enable Bump Map** from the texture shader property editor.
6. Determine the strength of the bumping by adjusting the **Bump Map Factor** parameter. You may wish to open a render region to tweak this parameter.
7. Click the **Up** button from the texture shader property page to open the Mix 8 Colors shader property page. From here you can adjust the way the bump maps and textures are blended with each other. For more information on how to work with the Mix 8 Colors node, see *Blending Textures* on page 99.

Bump Mapping with the Alpha Channel

If you wish to use an image or texture's alpha channel to define a bump map, simply repeat the steps needed to apply a regular bump map (page 117). Once the bump map is in use, click the **Use Alpha** option so that the texture shader calculates the alpha channel instead of the RGB values.

Bump Mapping Using the Render Tree

There are several ways to achieve bump maps using the render tree. All of the methods involve using the shaders found in the **Nodes** > **Bump** pull-down menu of the render tree's command bar.

- **Bump Tree**: This shader can be used to bump at the surface shader level or the material node level. It requires an image clip to apply a bump map.
- **Bump map Generator**: Generates a bump map using a texture or color input.

For more information on how to use the render tree and create bump maps with it, see *Where to Start?* on page 218.

Creating a Displacement Map

In addition to bump maps, you can also use displacement maps, which, unlike bump maps, perturb the geometry of an object to create ripples, ridges, or just plain bumps.

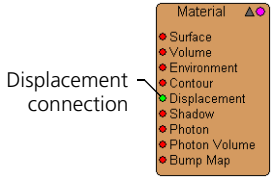
A displacement map is a scalar map that displaces a surface at each point in the direction of the object's normal; the geometry is distorted according to the map during the rendering process. Unlike regular bump mapping that "fakes" the look of real texture, the edges are visibly raised and can cast shadows that follow the displacement effect.

Displacement map | Bump map

Displacement (left) affects the geometry of the object during the rendering process. The object's edges are visibly raised and the displacement casts shadows, while the bump map leaves the surface smooth. On a bump-mapped image (right), however, the edges remain smooth although the bump gives the appearance of displacement.

To create a displacement map

1. Select the object whose surface you want to displace.
2. Open a render tree view in a viewport. The material that defines the object's material is displayed as an orange node in the render tree, which is connected to the surface parameter of the default node.
3. From the Nodes pull-down menu, select a Texture you wish to use as a displacement map. Once selected, it appears in the render tree work area.
4. Again from the **Nodes** pull-down menu, select the **Conversion** > **Color2Scalar** shader.
5. Connect the output of the texture node (click on the red dot and drag the connector) to the input of the Color2Scalar shader (green dot). Then connect the Color2Scalar shader to the **Displacement** parameter of the Material node.

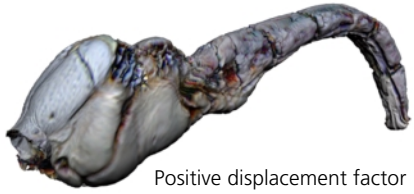


Material Node

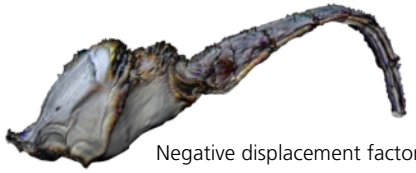
If the texture you selected does not connect to the Displacement input, it is because they are incompatible connections. The Displacement input only accepts scalar inputs. If you want to connect a color-output texture, such as marble, you have to use the Color2Scalar converter between the nodes. This converter can be quickly accessed from the **Nodes** > **Conversion** button on the render tree command bar.

To control the amount of displacement

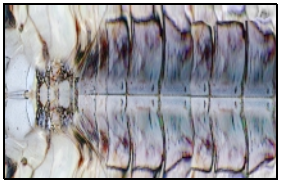
6. Now that the displacement has been applied, you will probably want to adjust its intensity. You can do this by getting the **Nodes** > **Image Processing** > **Intensity** shader from the render tree command bar.
7. Connect the Intensity shader between the texture shader that is creating the displacement and the Color2Scalar shader.
8. Double-click the Intensity shader to open its property page, and adjust the **Factor** parameter to control the displacement on the object.



Positive displacement factor



Negative displacement factor



Texture used

A displacement map on an object affects the rendered geometry (not the wireframe) of an object. Hence, creating a jagged displacement map (like the one above) would cause jagged shadows of the texture. Displacement is controlled via the **Intensity** shader placed between the material node and the texture used to create displacement.

To edit displacement's geometric approximation

To edit the displacement quality in the render region and in the final render, you can set the object's geometric approximation.

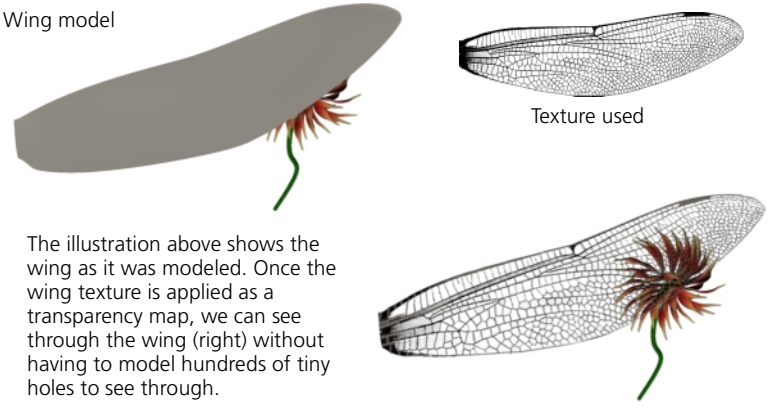
9. With the displaced object still selected, right-click the **Property** button in Selection panel and select **Viewing Properties** from the pop-up menu. This opens the Viewing property editor.
10. From the property editor, click the Geometric Approximation tab and then the Displacement tab beneath it.
11. Set the **Parametric Subdivision** parameter. Start with a low value, such as 4, and gradually increase it to see the displacement on the object rendered in the render region.

Mapping Effects

In addition to bump and displacement maps, you can apply any combination of transparency, reflection, and environment maps to add a sense of realism to your scene. In most cases, these maps are derived from an image file, but they can also be defined by a shader or a combination of an image file and a shader.

Creating a Transparency Map

A picture can be used to define areas of transparency on your object. The alpha channel or RGB intensity of the picture file is used, and a factor is applied to map a pattern of varying degrees of transparency on the object's material.



The illustration above shows the wing as it was modeled. Once the wing texture is applied as a transparency map, we can see through the wing (right) without having to model hundreds of tiny holes to see through.

To create a transparency map

You can create a transparency map by connecting a texture to a surface shader's transparency parameter.

1. Select the object to which you want to apply a transparency map. Make sure your object has a surface shader applied to it. If it doesn't, you can quickly apply one by selecting **Get** > **Material** > **Phong**.
2. Open the surface shader's property page by selecting **Modify** > **Shader** from the Render toolbar.
3. From the surface shader property editor, select the **Transparency/ Reflection** tab.
4. Click the connection icon to the right of the Transparency color sliders. From the pop-up menu, select a texture shader. Once selected, the texture is connected to the Transparency color parameter of the surface shader, and the render region displays a transparency map on the selected object.

Connection icon

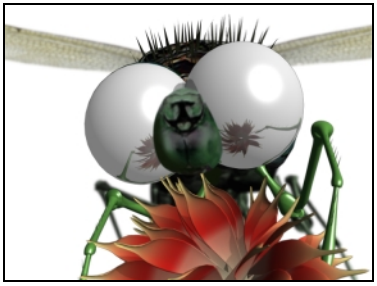
5. The texture shader's property page opens. Define a texture projection for it from the Texture Support property page.

If you wish to control the level of transparency of your surface shader, use a **Nodes** > **Image Processing** > **Intensity** shader between the texture shader and the surface shader in the render tree.

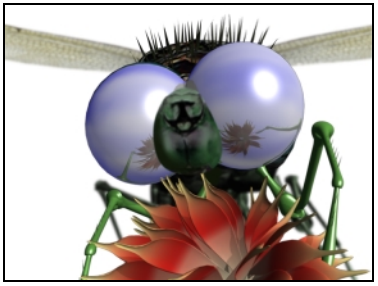
If you don't see your effect, try increasing the Ray Depth values in the render region setup (**Render** > **Region** > **Options**) or the rendering options (**Render** > **Render** > **Options**).

Creating a Reflection Map

Reflection mapping can be used to simulate an image reflected on the object's material but without needing to use raytracing. It can also be used to add an extra reflection to an object's reflective, raytraced surface.



Reflection



Reflection and reflection map

The illustration to the left shows an insect's eyes with its reflection property enabled; notice the flower's reflection in the eyes. The illustration to the right has a reflection map in addition to the reflection property. The reflection map reflects a defined shader or texture in the surface's reflective areas.

When used without raytracing, only the reflection map appears on the object's surface; when used after raytracing, the map is combined with raytraced reflections.

To create a reflective mapping on an object, you have to connect a texture map to the object's environment input.

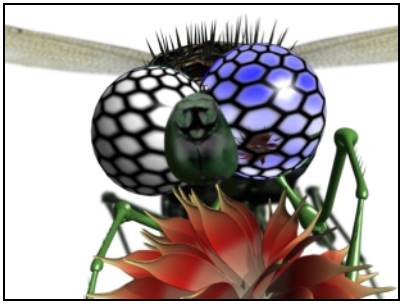
To create a reflection map

1. Select the object to which you want to apply a reflection map. Make sure your object has a surface shader applied to it. If it doesn't, you can quickly apply one by selecting **Get** > **Material** > **Phong**. A Phong shader is applied to your object.
2. Open the surface shader's property page by selecting **Modify** > **Shader** Render toolbar.
3. From the surface-shader property page, select the **Transparency/ Reflection** tab.
4. Click the connection icon to the right of the Reflection color sliders. From the pop-up menu, select a texture shader. Once selected, the texture is connected to the Reflection color parameter of the surface shader, and the render region displays a reflection map on the selected object.

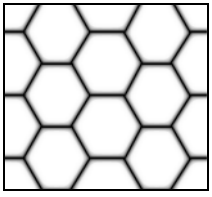
Connection icon

5. The texture shader's property page opens. Define a texture projection for it from the Texture Support property page. See *Creating a Texture Projection* on page 107 for more information on texture projections.

If you wish to control the level of reflectivity of your surface shader, use a **Nodes** > **Image Processing** > **Intensity** shader in a render tree.



The eye above has the honeycomb texture applied to it. The eye to the right has a reflection map applied to it. Notice how the white areas of the texture become reflective and the dark areas opaque.



With texture reflectivity, white areas of texture become non-reflective. Using a negative reflection value, black areas become reflective.

For more information on how to apply environment maps to a scene or object, see *Environment Shaders* on page 203.

Creating a Sequence Texture

Instead of applying a static image file as a texture, you may want to apply an animated sequence as a texture to an object. For example, a raindrop sequence applied to a window.

To apply a sequence texture, you can either

- Create a script with a SCR extension or a simple text file containing a series of picture file-locations with absolute paths. Instead of applying an image file, you would choose this script file.

  or

- In the script editor, type the following:
  `CreateImageClip` "<Path and sequence name>.[<first frame>..<last frame>].pic"
  e.g.: CreateImageClip "\\mymachine\000\sequence.[1..30].pic"

Creating Memory-Mapped Textures

Memory mapping means that the texture is not loaded into memory but is accessed directly from disk when a shader uses it. If a texture is memory-mappable, the mental ray renderer recognizes it and memory-maps it automatically. Only the map image file format (extension **.map**) can be mapped.

If the textures and the scene are so large that they do not fit into physical memory, loading a texture is equivalent to loading the file into memory, decompressing it, and copying it out to swap (the disk partition that acts as a low-speed extension of the physical memory that exists as RAM chips). From then on, accessing a texture means accessing the swap. Memory mapping eliminates the "read-decompress-write" step and accesses the texture from the file system instead of from the swap; therefore, less space is needed.

If the texture and scene are not large and fit into memory, and if the texture is accessed frequently, memory-mapped textures are slower than regular textures because the swap would not have been used.

Memory mapping requires several preparation steps:

- Convert your textures to **.map** format using the **imf _copy -p** standalone. You must update the scene file to reference these textures. For more information on the standalone, see *Converting Image Files to Memory-Map Textures* on page 239 of the *Fundamentals* guide.

The mental ray renderer recognizes **.map** textures even if they have an extension other than **.map**. This means that you can continue to use the "non-**.map**" texture references in your scene file.

- Memory-mapped textures are automatically considered local by mental ray. If the scene is rendered on multiple computers, each computer tries to access the given path instead of transferring the texture across the network, which would defeat the purpose of memory mapping. This path must be valid on every computer participating in the render.
- The texture should not be on an NFS-mounted file system (one that is imported across the network from another host). Although this simplifies the requirement that the texture must exist on all hosts, the necessary network transfers reduce the effectiveness and can easily make memory mapping slower than regular textures by slowing down the render.
- Memory mapping works best if there are extremely large textures containing many tens of megabytes that are sampled infrequently, because then most of the large texture files are never loaded into memory.

Memory-mapped textures are not decoded or converted by mental ray when they are read in: memory mapping expects the textures to be available in the exact format that mental ray used during rendering. Normally, mental ray attempts to automatically convert image-data formats. For example, if a color image file is to be used as a scalar texture, mental ray converts the color pixels to scalars as the texture is read in. This does not work for memory-mapped textures.

OGL Display Settings

As explained in *Chapter 3: Material & Texture Basics* on page 59, the material node acts as a placeholder for shaders that affect an object's look. By opening the material node's property editor, you can define the OGL settings it will use to display textures in the Textured view of a viewport.

The following OGL options are specific to the Textured view in a viewport and not the display options in a render region.

The OGL settings are set per object and not for the entire scene. Every geometric object has the same default OGL settings, but you can alter any of their settings to suit your needs.

To open the material node property page, do one of the following

- Open a render tree view and double-click on the orange material node.

  or

- Select a geometric object or model and click **Modify** > **Shader** to open its surface-shader property page. Press the **Up** button to open its material-node property page.

Texture Display Mode

When viewing a texture in a viewport, you can control whether you want the texture to be affected by the scene's light (Shaded) or appear as a "decal" in a constant (Constant) lighting mode.

Select between the **Shaded** and **Constant** Texture Display Modes in the material property editor to define how a texture is displayed.

Texture Clamping

The clamping setting lets you define how you want a texture to be applied in the Textured mode of a viewport. You can select **Clamp** or **Repeat** from the Wrap U and V drop-down menus.

- **Clamp** rigidly applies a single instance of the texture to the object's surface according to the defined texture projection.
- **Repeat** repeats the texture across the object's surface using the defined texture projection.

Pyramid Mapping

Instead of using the memory-hungry antialiasing option, you can use pyramid mapping to avoid resampling a texture when it is far from the camera.

Pyramid mapping performs a type of "MipMapping" or pre-blur on a texture, according to the texture's distance from the camera. In turn, this prevents having to increase the antialiasing settings and removes flickering that is often associated with detailed textures being far from the camera.

Pyramid mapping can be used on both 2D and 3D textures.

From the Material node's property editor, you can use pyramid mapping and define the maximum texture-map resolution it will use by checking the **Generate MipMaps** option and entering a value for the **Maximum Width or Resolution** parameter. The parameter converts the input to a 2x-factor resolution; e.g.: 2 x 2, 4 x 4, 8 x 8, 16 x 16, 32 x 32, etc.

In addition, you can specify how a texture will be pre-blurred when it is far or near the camera using the **Texture Minification Filter** (when a texture is far from the camera) and **Texture Magnification Filter** (when a texture is near the camera) parameters.

Image Pyramid Mapping (Rendering)

You can use pyramid mapping—or "MipMapping"—to smooth textures when rendering. Pyramid mapping performs a pre-blurring of a texture that is far from a camera.

The classic example is a very long grid texture that seemingly goes off into infinity but starts near the camera. As the grid gets farther away, the texture flickers and breaks up as the renderer samples only a single, far pixel.

This can be corrected with antialiasing, but it may become very time-consuming when rendering. Pyramid mapping blurs the distant texture so the render doesn't return a specific pixel color; instead, it returns an interpolation of several distant pixels. This solution is less time-consuming than antialiasing and yields excellent, realistic results.

To use pyramid mapping, click the **Enable Multi Resolution Texture** check box from the Image Clip property editor (property tab). Use the **Blurring** slider to control how much interpolation—or blurring—the textures will have applied to them. More complex a textures should have a higher blur applied to them.

Chapter 5 Working with Lights

Without lights, it doesn't really matter what your scene looks like—you can't see it. Plain and simple.

Each light in a scene contributes to the scene's illumination and affects the way all objects' surfaces appear in the rendered image. You can dramatically change the nature and mood of your images by modifying lights and their properties. Also, you can add lights directly, or you can work with lights from imported scenes.

Rendering Properties

Every light in a scene is defined by a set of properties. Some of these properties do not vary, no matter what type of light or the 3D program from which it was imported. These properties are referred to as the light's rendering properties, as they determine its appearance when rendered. These include:

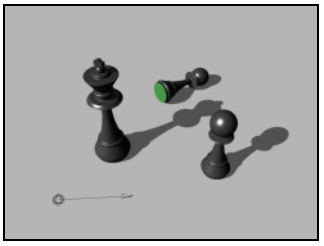
- **Selectivity**—you can specify which objects are affected by which lights.
- **Shadows**—you can create several different kinds of shadows, including using area lights to define soft shadows.
- **Effects**—you can create global-illumination effects such as indirect lighting from diffuse reflections and color bleeding, as well as caustic lighting effects such as shimmering highlights on sand beneath rippling water. For more information on special light effects, see *Chapter 6: Global Illumination & Caustics* on page 153 and *Chapter 8: Blurs, Flares & Other Effects* on page 185.

Light Effects

For more information on lighting effects, including global illumination, caustics, and flares, see *Chapter 6: Global Illumination & Caustics* on page 153.

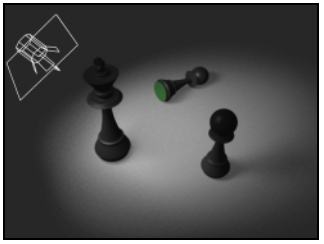
Types of Lights

There's a light for every occasion. Although you cannot change an imported light's type (except to define it as an area light), you can create several different kinds of light:

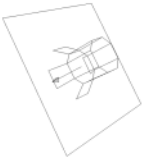


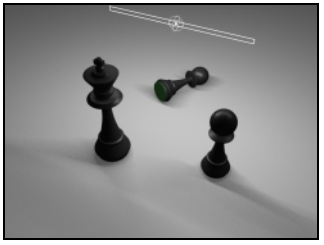
Infinite
Simulates light sources that are infinitely far from objects in the scene. There is no position associated with an infinite light, only a direction. All objects are lit by parallel light rays. This is a scene's default light

Light Icon

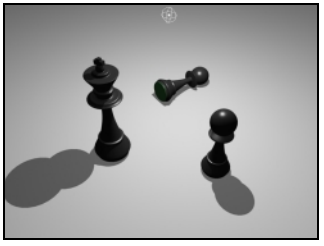


Light Box
Simulates a light diffused with a white fabric, like a studio's light box. The light and shadows created by this light are very soft. Speculars are still visible, but noticeably weaker.

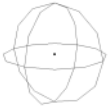


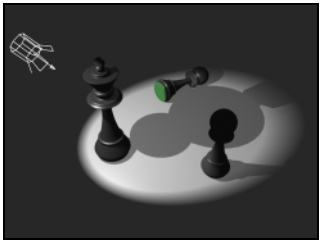


Neon
Simulates a neon light, which is essentially a point light with its settings and shapes altered so it resembles a neon or fluorescent tube.



Point
Casts rays in all directions from the position of the light. This is similar to a light bulb, where the light rays emanate in all directions from the bulb.





Spot
Casts rays in a cone, simulating a real spotlight. This is useful for lighting a specific object or area.

Setting a Scene's Ambience

A scene's ambient color is multiplied with an object's ambient color. If the scene ambience is set to black, nothing can alter the ambient color of an object except, of course, a light.

A scene's ambient lighting is equivalent to a SOFTIMAGE|3D scene's Atmosphere Ambience.

For more information on an object's ambient color, see *Chapter 3: Material & Texture Basics* on page 59.

To edit a scene's ambient color

1. Open the explorer in any viewport and set the explorer's scope to **Scene**.

If the Scene Ambience icon isn't visible, try selecting **Local Properties** from the Show menu in the explorer.

2. Click on the **Ambient Lighting** icon to open the scene's ambient lighting property editor.
3. Edit the scene's ambient value as desired using the color sliders. Keep this value dark or low so as not to affect your scene's lighting in an unpredictable way.

Although a Scene Ambience node is visible at the object level in the explorer, editing its value will affect the ambience of the whole scene.

Setting a Realistic Ambient Color

Perhaps the most realistic way to set a scene's ambient color is to match a color from a rotoscoped image from the scene you use for compositing.

To set a realistic ambient color

1. Open the Ambient Lighting property editor as described above.
2. Open a Rotoscope view in any viewport, and load an image or scene you will be using for compositing.
3. From the Ambient Lighting property editor, click the color chip to open the color editor.
4. From the color editor, select the color picker(left); the cursor changes to an eye-dropper.

Color Picker

5. Use the color picker to select the ambient color from the image or scene you will be using for compositing. For best results, choose an ambient color you find in the shadows of an image.
6. Close the color editor to accept the chosen color as the new ambient lighting color.

Keep in mind that there are some color-range limitations to safely output for TV (or film). A true black or a full red isn't supported by NTSC and some other formats. Also, adjusting the gamma value will not alter a black color generated by a non-lit scene/object; i.e., you can adjust the shadowed area of a scene without adjusting or adding lights.

Creating Lights

You can add several types of lights to your scene from within SOFTIMAGE|XSI. Once you have added a light, you can set its properties or you can load a preset as described in *Saving and Loading Shader Presets* on page 30.

In addition, you can set area-light options for point lights and spotlights. For more information about area lights, see *Creating Soft Shadows with Area Lights* on page 148.

To create a new light with the default options

1. Choose **Get** > **Light** from a toolbar.
2. Select either **Spot**, **Neon, Light Box, Point**, or **Infinite** as the type of light to create. A new light is added to the scene and positioned at the origin. For a description of the different types of lights, see *Types of Lights* on page 134.

The light's property editor appears in a floating window. For more information on how to set a light's properties, see *Setting Light Properties* on page 138. For a complete description of all the light properties, refer to the online help (?) in the appropriate property editor (left).

You can select and manipulate lights as you would any other object.

When creating a new light, the default light remains in the scene until deleted. A new light does not replace the scene's default light.

The Light Shader

A light is also defined by a light shader that describes the light's color, falloff, and other settings.

While rendering properties for a light do not change, the light shader can be detached and replaced by another shader of the same type. For information on attaching and detaching shaders, see *Attaching Shaders to Elements* on page 41.

You can animate all of a shader's properties to vary their values over time. For more information about animation in general, see the *Animating* guide.

Imported Lights

The light shader attached to a light varies depending on the 3D program from which the scene was originally imported. Lights imported into SOFTIMAGE|XSI from a SOFTIMAGE|3D scene are assigned a default light shader called soft_light. This shader appears as a subnode under the light object node.

The soft_light shader implements the infinite (directional) lights, point lights, and spotlights supported by SOFTIMAGE|3D. When lights are imported into SOFTIMAGE|XSI, the soft_light shader recognizes the light and behaves according to the light type: spot, point, or infinite.

The soft_light shader incorporates all the spot, point, and infinite light properties. Any spot, point, or infinite light you create uses the soft_light shader by default.

Setting Light Properties

Every light created or imported with your scene is defined by a light shader. All light sources need a light shader, such as the built-in soft_light shader, or a custom light shader you built and linked in with SOFTIMAGE|SDK.

The properties of the soft_light shader include its color, intensity, falloff, and raytraced shadows. Spotlights also have options that control the light cone's angle and spread.

Regardless of which light shader you are defining (even custom ones), you can edit its properties in its property editor. The properties you can define depend on the particular shader: they include the light color, attenuations, and spotlight directions.

You can edit and perform a number of operations on a light shader:

- You can save these properties as light-shader presets and apply them to another light shader of the same type in your scene. See *Saving and Loading Shader Presets* on page 30.
- Animate a light's properties so that the value changes over time. For example, you can make a light gradually change color or grow dim. For more information about animation in general, see the *Animating* guide.
- You can detach a light shader and replace it with another shader of the same type. For information on attaching and detaching shaders, see *Attaching Shaders to Elements* on page 41.

Setting the Light's Color

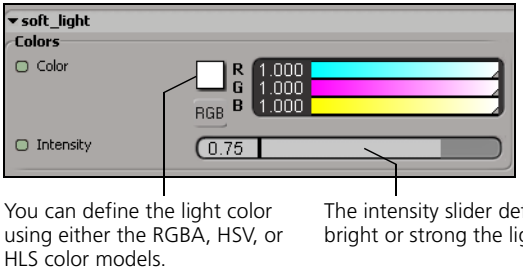
The color of a light controls the color of the rays emitted by the light. The final result depends on both the color of the light and the color of objects. For example, a red object reflects red light but absorbs blue and green light. On the other hand, a yellow object reflects both red and green light but absorbs blue light. If you illuminate a yellow object with a blue light, it appears dark. Objects illuminated by white light appear in their natural colors.

When you define the color of an object's material, you should work with a white light because colored light sources affect the material's appearance. You can color your light source afterward to achieve the final look of the scene.

To set a light's color

1. Select a light in the viewport and choose **Modify** > **Shader** on the Render toolbar to display the selected light's property editor.
2. In the light's property editor, set a color for the light. The current color is shown in the box to the left of the color sliders and under the light's name. You can set the color using the color sliders or by opening the Color

editor. You can also toggle between the RGB, HSV, and HLS color channels. For more information on color models and defining a color, see *Defining Colors* on page 75.



- Drag the color sliders to change the strength of each red, green, and blue channel independently, or type a numerical value directly in the space provided. The slider values range between 0 and 1, but you can enter higher values yourself. Hold the **Ctrl** key to move all sliders at once.

  or

- Click the color box to the left of the sliders to open the Color editor. For more information about using the editor to create, save, and use your own predefined colors, see *Defining Colors* on page 75.

You can also drag and drop a color from one color box to another. For example, if you have two property pages open, click and drag a "color chip" from one property page to another. Drop the "chip" directly onto the color box you wish to edit.

Setting the Light's Intensity

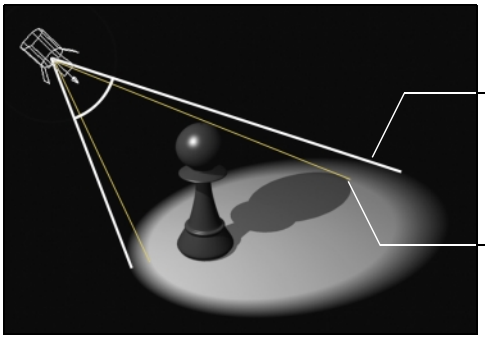
The intensity of a light is controlled indirectly through the strength of its separate color channels. If you want to adjust the strength of a light without changing its color, you can use the HSV or HLS color models and adjust V (value) or L (lightness). Click on the RGB button beneath the color box of a light's property editor.

To set values higher than 1 for a stronger light, type directly in the space to the left of the slider.

Switch between the RGB, HSV, and HLS color models by clicking the RGB button beneath any color chip to the left of a color slider. For more information on color models, see *Color Models* on page 75.

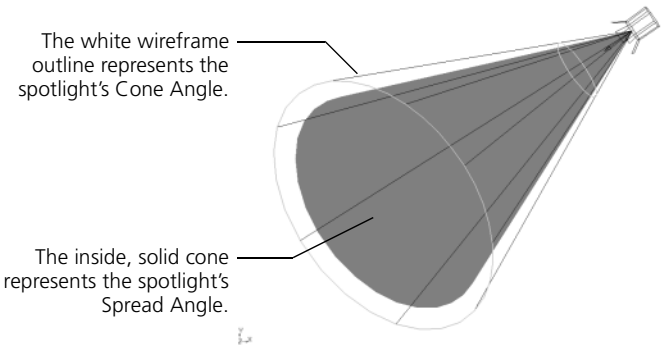
Setting a Spotlight

A spotlight casts its rays in a cone aimed at its interest. Spotlights have special options that control the size and shape of the cone. You can set these options using the spotlight's property editor or its 3D manipulators.



To set the cone and spread angles from a property editor

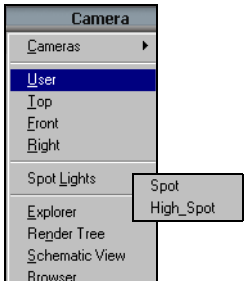
1. Select a spotlight in the viewport.
2. Choose **Modify** > **Shader** from the Render toolbar to display the selected light's property editor.
3. In the light property editor, set the **Cone Spread**. This is the angle in degrees of the inner, solid cone of full-intensity light.
4. Click on the General tab and set the **Cone Angle**. This is the angle in degrees of the exterior cone light. The cone defines the maximum spread of light.



To set the cone and spread angles using 3D manipulators

1. Select a spotlight.
2. Press **b** on the keyboard to display the light's manipulators.
3. Press the **Tab** key until the spotlight' cone and spread-angle manipulators appear.

4. The exterior wireframe cone controls the light's cone angle. Click the edge of the cone and drag outward or inward to interactively increase or decrease the Cone Angle value, respectively.
5. The interior yellow shaded cone controls the light's spread angle. Click the edge of the cone and drag outward or inward to interactively set the Cone Spread value.
6. Shift+click and drag the edge of either cone to increase or decrease both angles simultaneously. The distance between the angles of each cone remains the same.



Viewing from the Spot Light

The Spot Light view in a viewport lets you select from a list of spotlights available in the scene. Selecting a spotlight from the view menu switches the point of view in the active viewport relative to the chosen spotlight. The point of view is set according to the direction of the light cone defined for the chosen spotlight.

Changing the view from the viewport directly affects the spotlight's position or cone angle.

This view is useful to see what objects a spotlight is lighting and from what angle. For more information about other viewpoints, see *Chapter 4: Viewing Your Work* in the *Fundamentals* guide.

Setting a Light's Falloff

Falloff refers to the diminishing of a light's intensity over distance, also called *attenuation*. This mimics the way light behaves naturally. These options are available only with point and spotlights.

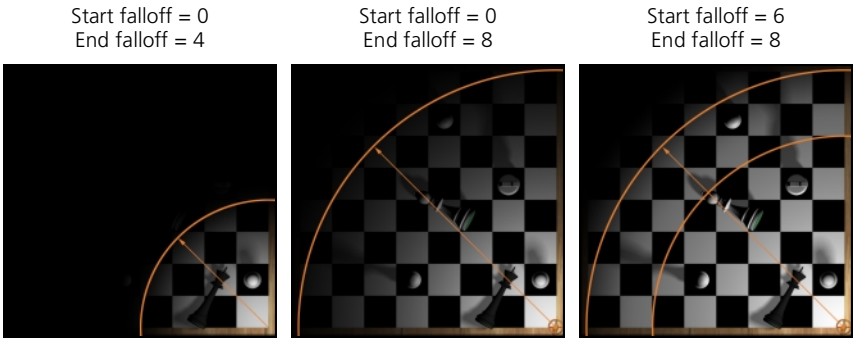
You can set the distance at which the light begins to diminish, as well as the distance at which the falloff is complete (darkness). This means you can set the values so the falloff affects only those features you want. In addition, you can control how quickly or slowly the light diminishes.

A light's falloff parameters are sometimes defined in conjunction with a depth-fading shader or an atmosphere shader, such as in a smoky bar or heavy fog. See *Chapter 2: Shader Basics* on page 37 for more information on applying and defining shaders.

You can set the falloff value using the light's property editor or its 3D manipulators.

To set the falloff from a property editor

1. Select a light in a viewport.
2. Choose **Modify** > **Shader** from the Render toolbar to display the selected light's property editor.
3. In the light property editor, select the **Falloff** option.
4. Set the required **Falloff Start** and **Stop** distance values in the Attenuation controls.

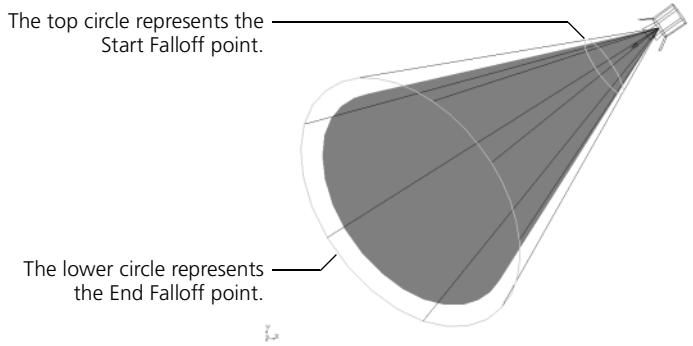


Falloff
Start and End Falloff values. Using a point light, umbra = 0; bottom corner of chess board is 0; top, left corner is 10.

5. From the **General** tab, choose what type of falloff you want: Linear or Use Exponent.
   - **Linear** falloff is sharper and less realistic. When selected, the Exponent parameter is automatically set to 1.
   - **Use Light Exponent** falloff is an inverse square falloff, which is more natural and realistic. This option uses the value in the **Exponent** field to determine the falloff. The default value is 2. A high value, such as 5, causes light to fall off very quickly, whereas a smaller value, such as 0.1, makes the falloff more gradual.
6. Finally, show whether you want the light to be diffused or not by selecting either the **Diffuse Only** or the **Diffuse and Specular** option in the light property page's drop-down menu. Diffusing softens the light and removes its specular highlight; that is, specular values are no longer computed.

To set the falloff using 3D manipulators

1. In the viewport, select the spotlight or point light for which you want to set the falloff.
2. Press **b** on the keyboard to display the light's 3D manipulators.
3. Press the **Tab** key until the light's start and end falloff manipulators appear.



4. The exterior cone-angle control displays a light blue start falloff disc (if falloff is already activated, you see this before displaying the 3D manipulators). Ctrl+click the disc at the base of the white cone angle cone to activate the falloff.
5. Click and drag this disk toward or away from the light to set the Start value.
6. The exterior Cone Spread control also displays a magenta end falloff disk at its base. Click and drag this disk toward or away from the camera to set the End value.
7. Shift+click and drag either disk to increase or decrease both Start and End Falloff values simultaneously. The distance between the planes of each disk remains the same.
8. To deactivate falloff, Ctrl+click either the start or end falloff disk.

Creating Shadows

Shadows can make all the difference in a scene. A lack of them can create a sterile environment, whereas the right amount can make the same scene delightfully moody. If you want your scene to have a more realistic look, you can create shadows that appear to be cast by the objects in your scene.

Shadows are controlled independently for each light source. This means that a scene can have some lights casting shadows and others not.

To create a shadow, you must set up three things:

- A light that generates shadows
- An object that casts shadows
- Rendering options that render shadows

Types of Shadows

There are three basic kinds of shadows you can create:

- **Raytraced** shadows that use the raytracing renderer. The shadows are very realistic but take longer to render. For information about creating raytraced shadows, see *Creating Raytraced Shadows* on page 150.
- **Shadow-mapped** shadows that use the scanline renderer. They are quick to render but not as accurate as raytraced shadows. Shadow-mapping works only with spotlights. See the next section (on page 146) for information about creating shadow-mapped shadows.
- **Soft** shadows that are created by defining area lights. Area lights need to be rendered with the raytracing renderer to obtain soft shadows. For more information, see *Creating Soft Shadows with Area Lights* on page 148.

Rendering Methods

In addition, you can render all of these types of shadows using different rendering methods. They are:

- **Regular** shadows—performs a basic, simple rendering of the shadows. The amount of light from a light source that passes through a shadow-casting object is determined. The shadow shaders are used in random order.
- **Sort** shadows—similar to Regular shadows but uses the shadow shaders differently. The shadow-casting objects are sorted so that the shadow shader of the object closest to the illuminated point is processed first and the object closest to the light is preprocessed last.

- **Segment** shadows—also sorts the shadow shaders in a specific fashion. When Segment is chosen, shadows are computed by tracing the segments (between the illumination point, the occluding objects, and the light source) and applying volume shaders to these segments (shadow segments). This process slows down rendering but is required if volume effects are to cast shadows.
- **None**, as the name indicates, does not allow the light to compute shadows. This option is usually used to speed up rendering.

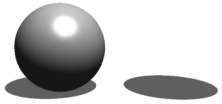
Creating Shadow Objects

Besides creating shadows from an object, you can also create shadows without an object being there! This is called a shadow object and is used exclusively with lights for casting shadows. The object itself is not visible when rendered, only its shadow appears. Material and textural effects defined for the object can be used in conjunction with the **Shadows** option to cast complex patterns of light and shadows in the scene (the object is not visible to the camera).

It is important to remember that when you want to render shadows you must set up one or more lights to generate shadow-mapped or raytraced shadows.

To create a shadow object

1. Select an object in the viewport for which you want to create a shadow object.
2. Click the **Property** button in the Selection panel to display the selected object's node.
3. Click the **Visibility** node to open the object's Visibility property editor.
4. On the Visibility property page, deselect **Primary rays** and **Secondary rays**, and select **Shadows**.



Object with Primary Rays Turned On

Shadow Object with Primary Rays Turned Off and Shadows left On

Creating Shadow-Mapped Shadows

Shadow mapping, also known as *depth-mapped shadows*, works only with spotlights having a cone angle less than 90 degrees. It uses a modified z-buffer (depth) algorithm to create shadows more quickly but less precisely than those created with raytracing.

Normal point source light has sharp shadow edges.

Shadow-mapped shadows blur with distance.

This algorithm calculates color and depth (z-channel) information for each pixel, based on its surface and distance from the camera. Before rendering starts, a shadow map is generated for the light if one does not already exist. This map contains information about the scene from the perspective of the light's origin. This information describes the distance from the light to objects in the scene and the color of the shadow on that object. During the rendering process, the map is used to determine if an object is in a shadow.

When working with shadow-mapped shadows, avoid using the Area Light function at the same time. Artifacts may occur within your scene.

To create shadow-mapped shadows

1. Open a spotlight's property editor using the **Modify** > **Shader** button on the render toolbar.
2. In the light property editor, select **Shadows Enabled** by clicking its check box.
3. To use shadow maps, select the Shadow Map tab and select the **Use Shadow Map** option.
4. Set the **Resolution** to determine the quality of the shadow map (width and height of the map buffer). Note that a high-resolution setting will increase memory usage and rendering time.
5. Set the **Softness** to determine the type of shadow. A value of 0 results in hard-edged shadows. Higher values create longer, smoother shadows but increase rendering time.

6. Set the **Samples** adjust the shadow's resolution. High sample values yield a higher quality render, but it increase rendering time.
7. Repeat steps 1 to 6 for other lights, if desired. Note that the more lights used to generate shadows, the longer render times will be.

Although the default settings use shadows in the render region and during the final render, you may need to do the following to complete the shadow map effect:

8. Select the objects whose shadows you want to cast and display their property editor by clicking on **Property** on the Selection panel. Open the **Visibility** property page and select the **Shadows** option.
9. To view the shadow map in the render region, draw a region (press **q**) **and** choose **Render** > **Region** > **Options** from the render toolbar to open the View Rendering Options property editor. Select the **Shadows** tab.
   - Select **Enable Shadow Maps** to activate shadow maps in the render region.
   - Select the **Rebuild** option to recalculate the shadow map at every frame.

For more information about the render region, see *Previewing Interactively with the Render Region* on page 32.

10. Before rendering, choose **Render** > **Render** > **Options** from the render toolbar to open the Rendering Options property editor for the current render pass and set the following options:
    - From the Optimization page, select the **Scanline** renderer option (deselect **Raytracing**).
    - From the Active Effects tab, select the **Shadow Type rendering method for** your scene: Regular, Sort, or Segment. For more information on Shadow types, see *Creating Shadows* on page 144.
    - From the Shadows tab, select the **Enable Shadow Maps** and, if desired, the **Rebuild Shadow Map** options.

Creating Soft Shadows with Area Lights

Area lights are a special kind of point light and spotlight. The rays emanate from a geometric area instead of a single point. This is useful for creating soft shadows with both an umbra (the full shadow where an object blocks all rays from the light) and a penumbra (the partial shadow where an object blocks some of the rays).

Normal point source light has sharp shadow edges.

Area light creates blurred shadows.

The shadow's relative softness (the relation between the umbra and penumbra) is affected by the shape and size of the light's geometry. You can choose from three shapes and set the size as you wish.

When working with area lights to create shadows, avoid using the Shadow Map function at the same time. Artifacts may occur within your scene.

To determine the amount of illumination on a surface, a sample of points is distributed evenly over the area light geometry. Rays are cast from each sample point; all, some, or none of the rays may be blocked by an object. This creates a smoothly graded penumbra.

To define an area light

1. Select a Point or Spot Light and click the **Modify** > **Shader** button on the render toolbar to display the selected light's property editor.
2. From the light property page, click the Area tab and select the **Area Light**.
3. Click on the Area Light icon checkbox and specify the **Geometry**: **Rectangle**, **Disc**, or **Sphere**. This determines the shape of the surface from which the light rays emanate.
   - Set the **Sampling** sliders to control the size of the grid of sample points on the surface of the area light in the **U** and **V** directions. Values over five take longer to render.

Because each area light is sampled multiple times, the sampling level for the render pass can be lowered even for high-quality renders. For more information on rendering options, see *Chapter 3: Rendering Options* on page 51.

4. Use the **Area Transformation** sliders to set the size of the surface from which the light rays emanate:
   - For rectangles, the X and **Y** sliders control the length and width when scaling.
   - For discs and spheres, the X slider controls the radius and the Y slider is switched off when scaling. If desired, use the X, Y, and Z Orientation sliders to further control the rotation of the surface from which the light rays emanate. X, Y, and Z rotations are always defined according to the geometry's local space; that is, according to its own center of orientation.

The center of the area light's geometry is always the position of the underlying point or spotlight. The Z axis of the local coordinates points from the light's center toward its interest. Orientation has no effect on area lights with sphere geometries.

Creating Raytraced Shadows

Raytracing involves calculating how light rays are reflected, refracted, and obstructed. It gives very realistic results, but it can be a time-consuming process.

To create raytraced shadows

1. Select a light in the viewport and choose **Modify** > **Shader** on the Render toolbar to display the selected light's property editor.
2. From the light property page, click the **Shadow Enable** checkbox.
3. Set the **Umbra** value with its slider. This defines a transparency factor on the umbra (main) area of the shadow (default 0.75). It controls how the shadow blends with the material on which it is cast to create a more realistic shadow.
4. Repeat steps 1 to 3 for other lights, if desired.
5. Before rendering, choose **Render** > **Render** > **Options** from the Render toolbar to open the rendering options property editor for the current render pass, and set the following options.
6. From the Optimization page, set the **Render Mode** to Raytracing (deselect Scanline).
7. Click the Shadows tab and, from the Raytraced section, select a Shadow Type rendering method, either Sort, Segmented, or Regular. For more information on these options, see *Creating Shadows* on page 144.
8. To view raytraced shadows in the render region, select **Render** > **Region**> **Options** and repeat steps 8 and 9.

Note that the more lights used to generate shadows, the longer the scene takes to render.

If you wish to turn the shadows off or make them invisible, the **Shadows** option must be unselected in an object's Visibility property page, as well as in the render options on the Optimization page.

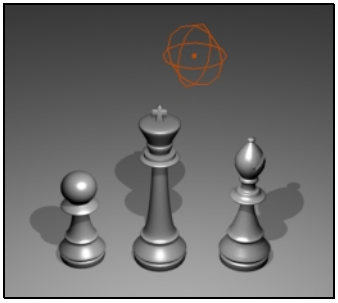
Using Selective Lights

When you create a light, by default all visible objects are affected by it. However, you can select specific objects to be affected by the light; such lights are called *selective*. This can be useful if you are using a number of lights in a scene but want to reduce the rendering time (the more lights there are in a scene, the longer it takes to render) or create a specific lighting effect. Only the objects that are affected by the light source are illuminated by it when rendered. Affected or unaffected objects are defined as **Associated Models**.

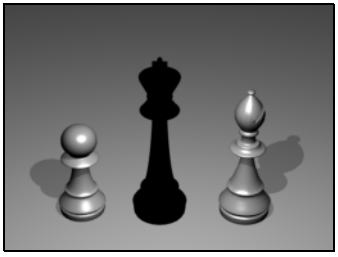
A selective light can be either Inclusive or Exclusive. Each light's selective property can be changed through its property editor.

- **Exclusive** illuminates every object *except* for those in its Associated Models group.
- **Inclusive** illuminates every object defined in its Associated Models group.

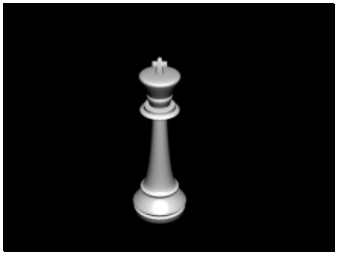
A simple scene illuminated by a point light. None of the geometric objects are included in the light's **Associated Models**' list, hence, they are all lit by the light source.



The King piece (center) has now been added to the light's Associated Models list, thereby making it affected by the light's selective property. In this example, the light has been defined as **Exclusive**, thereby not illuminating the objects on the light's Associated Models list.



This image shows the light when set to **Inclusive**. Now the light source affects only the objects listed in the Associated Models list (just the King piece) and ignores the rest.



To define a selective light

1. Select a light in the viewport. Any type of light can be selective.
2. Open an explorer that displays both objects and lights (a Scene scope is useful).
3. Click the light's icon to open its property editor. Leave it open for now.
4. Expand the Light node of the light you wish to define as selective. A light's last node is a group named (by default) **Associated Models**. The contents of this group define which objects are affected by either an exclusive or inclusive light.
5. Click and drag any of your scene's objects into the Associate Models group of the light you want to be selective. Copies of the objects are placed in the Associated Models list. None of your object's properties are affected.

The Associated Models group can only contain geometric objects. You can drag and drop hierarchies, models, groups, or as many geometric objects as you wish.

6. In the light property editor, define the light as **Inclusive** or **Exclusive**. **Inclusive** causes the light to affect only the objects in the Associated Models group. **Exclusive** causes the light to affect every object in a scene *except* those contained in the Associated Models group.

To remove the selective property

If you change your mind and decide that you no longer want an object to be affected by a selective light, do the following:

1. Expand the Associated Models group in the explorer.
2. Right-click the object you wish to remove from the Associated Models group and select **Remove form Group** from the pop-up menu.

Chapter 6 Global Illumination & Caustics

We see them every day but we take them for granted: around a fountain, on the walls of a sunlight room, even beside your favorite beer. In the 3D world, caustics and global illumination are one of the easiest ways to create realistic lighting. Although memory hungry, these effects—also called photon effects, or *radiosity*—are well worth the time. Very fast and physically accurate global-illumination and caustic lighting can be simulated. To do this, energy is emitted into the scene in the form of simulated photons. The final density of these photons on diffuse surfaces in the scene is computed and stored in the Photon Map™.

Global Illumination

Global illumination is a physical simulation of all lighting in a scene. This includes both direct and indirect lighting caused by diffuse reflections. Photon rays emanate from the light source in all directions and bombard the scene. When these photons hit an object, some stick and others are reflected and refracted. These reflected and refracted rays go on to illuminate other surfaces. For example, the underside of a table is not completely dark, even if it is not illuminated directly by a light or reflections from a shiny surface. With global illumination, no ambient light is shed, which is a non-directional light that pervades a scene.

Global illumination includes the simulation of effects such as color bleeding. For example, diffuse light reflected off a red object creates a pink tinged area on a nearby white object.

When working with global illumination, you can make the render region render faster by selecting the Grid Acceleration method instead of BSP Tree in the Render Setup. For more information on render-acceleration methods, see *Acceleration Methods for Raytracing* on page 62.

Caustic Effects

Caustics are light patterns created when light illuminates a diffuse surface by reflection or refraction: for example, a bright patch under a magnifying glass, reflections on a wall from a mirror ball, or shimmering highlights under rippling water.

Caustics are a subset of full global-illumination simulation where photons are only emitted toward any caustic-generating objects in the scene. These photons are then reflected or transmitted by these objects until they hit diffuse surfaces in the scene. When a photon hits a caustic-receiving object, it is stored in a caustic Photon Map. Photons are stored only on diffuse surfaces, so the caustic-receiving objects must have a diffuse component in the reflection model.

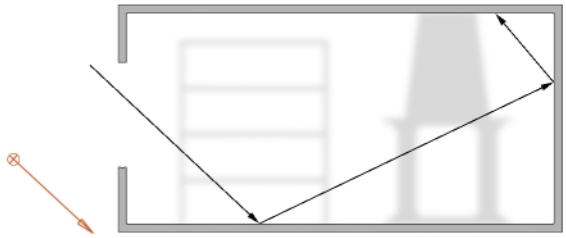
Materials of caustic-emitting objects specify if and how photons hitting this object should continue. In addition, the materials of caustic-receiving objects decide how to "pick up" photons absorbed by the object during processing.

For best results, your object's material has to be mainly specular (little or no diffuse reflection), and the sum of reflection and transparency has to be close to or larger than 1.

Anatomy of a Photon

If a photon hits an object that transmits the effect, photon information is stored and the photon continues to trace indirect-lighting information. Objects that transmit photons are either diffuse (that is, not black), have a high specularity, are transparent, refract light, or any logical combination of these. A photon stops bouncing (is absorbed) when it hits an object that neither receives nor transmits the effect. Using an object's Visibility property editor, you can control which objects receive or transmit the effects. And from the Render Options property editor, you can control the number of bounces that a photon makes. See *Raytracing* on page 60 for more information on controlling ray depth.

This information is stored in a Photon Map, which is a three-dimensional representation of where photons are stored in the scene. You can save a Photon Map so that once you're happy with the way your scene looks, you can avoid having to recalculate it every time you render a scene.



This illustration shows the path of photons that occurs when using global illumination.

What Photons Do

The more photons you throw into a scene, the more refined the effect will be. Unlike most other settings, where ranges often go from 0 to 1 or 1 to 100, when working with photons you'll be specifying high numbers: 1000, 10 000, 100 000, 500 000 or more. The more photons there are in the scene, the longer it takes to render.

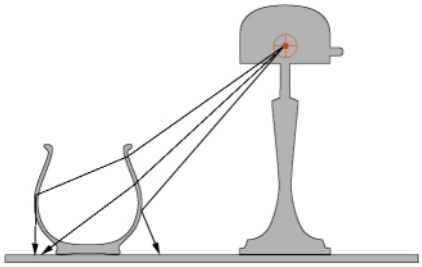
To do all this bouncing around, photons need energy. You need to specify a numerical strength for them so that they can both bounce from object to object and have an effect on distant objects. Photons leave energy behind, which create the photon effect, every time they "stick" to a surface.

Energy Intensity controls the brightness of the effect. Each photon uses an equal amount of the energy specified. For example, 1000 photons will use 1/1000th of the photon energy. Generally, the more distant objects are from the light source, the more energy each photon needs. Again, you'll be working with large numbers: 100, 1000, 10 000, 100 000, 1 000 000 and more. Energy values don't affect rendering time. To set a photon's energy, see *Rendering Global Illumination and Caustics* on page 165.

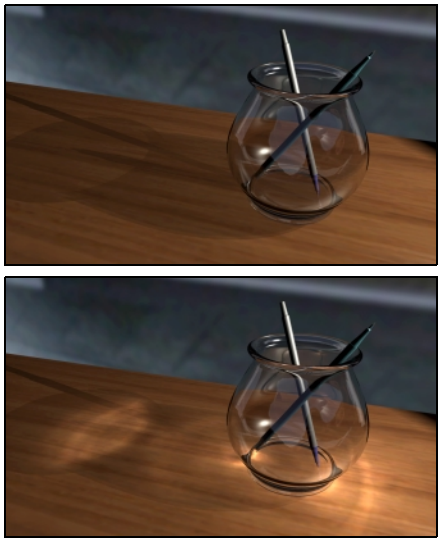
The Photon Map

The Photon Map is essentially a table that stores the positions of the photons in the scenes along with their RGB energy levels for each of the thousands of photons. When SOFTIMAGE|XSI wants to determine how the Photon Map affects a given point in space, the raytracer consults the Photon Map to determine which photons are nearby (within a given radius—see page 166) and determines, based on energy level of every photon within the radius, the photon contribution to the total lighting in the scene. Only one map is created regardless of the number of lights.

For more information on when to rebuild the Photon Map, see page 167.



This illustration shows the path of photons that occurs when using caustic illumination.



Both scenes have the same number of lights and types of shaders. The image on the bottom used caustic effects. Notice the reflections on the table caused by the glass pencil holder. Both the glass's reflect-ivity and refraction play a part in how the caustic effects are rendered.

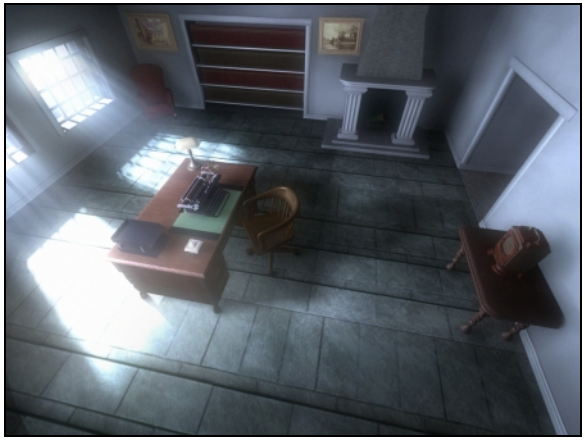
Preparing Object Surfaces

In order to achieve a good caustic- or global-illumination effect, you have to set up your geometric objects' shaders in such a way as to enhance the effect. Here is a quick checklist of things to keep in mind:

- Assign surface shaders to the objects that will be receiving caustic- or global-illumination effects. Otherwise, the default gray render settings are used.
- For global-illumination effects, use the Lambert, Phong, Cook-Torrance, anisotropic or Blinn surface shaders. You cannot use constant.
- For caustic effects, use Phong, Blinn, or anisotropic surface shaders, because the caustic effect is driven by the object's specularity.
- Objects that cast, receive, or create global illumination or caustics cannot be black (in other words, the object's diffuse color cannot be RGB 0,0,0). Purely black objects absorb photons, so it won't create an effect.
- Some surface shaders seem more difficult to balance in a final global-illumination rendering. An object's ability to transmit light is related to its diffuse value. Objects with the usual diffuse values rendered with global illumination may look more saturated when hit by direct light (particularly point lights). Lowering the diffuse intensity greatly helps manage the color-saturation balance between direct and indirect illumination. Use the Luminance of the HLS color slider in the surface shader's property editor to change the intensity value quickly.

Caustics are actually the reflection of an object's specularity. You should adjust the object's specularity, not its reflectivity, to control the appearance of caustics.

- Caustics are influenced by the caustic-casting object refractivity. No refraction (a setting of 1) produces caustics that are very subtle. Settings higher than 1 create a "focusing" effect (try settings of 1.33 or 1.50 for this). Settings lower than 1 greatly diffuse the effect.



Both these scenes have the same setup, same shaders, and an equal number of lights. The scene on top has not been rendered with global illumination. The scene below uses a high Photon Reflection setting that causes the light from the windows to hit the floor and bounce off the walls. Notice how the drawer-side of the desk is illuminated with ambient light, whereas the image above has a very sharp falloff that appears less realistic when compared to a render using global illumination.

Global Illumination and Caustics Workflow

Global illumination and caustic effects are handled in a very similar way. The workflow for each effect is nearly identical, but the results can be stunningly different.

When working with global illumination and/or caustics, you may want to turn off your scene's ambient lighting in order to make photon effects more visible.

The light energy for both the global-illumination and caustic effects are measured in photons. There is a wide range of photon controls that allow you to fine-tune a photon effect in either the render region or for the final render pass.

To achieve a photon effect

1. **Define an Emitter:** Create a light in your scene that will emit photons for either a caustic or global-illumination effect. For more information on designating a light as a photon emitter, see *Defining a Light as a Photon Source* on page 162.
2. **Define the Transmitters:** Select an object in your scene that will transmit the photons (such as a lamp or sun). For more information on defining a transmitter, see *Setting Up Transmitters and Receivers* on page 163.
3. **Define the Receivers:** Select an object in your scene that will receive the photons (such as a desk or water surface). For more information on designating a receiver, see *Setting Up Transmitters and Receivers* on page 163.
4. **Displaying Photons:** Set up your render-region and/or render-pass settings to display global-illumination and/or caustic effects. For more information on setting your render options, see *Displaying Photons in the Render Region* on page 164 and *Rendering Global Illumination and Caustics* on page 165.

Because calculating photon effects is slower, you may want to turn off the Auto-Refresh option for the render region. You can do this by selecting **Render** > **Region** > **Auto-Refresh** from the Render toolbar.

Defining a Light as a Photon Source

The first step in creating a global-illumination or caustic effect is to define a light as a source for photon rays. To generate photons using a particular light source, you must specify the number of photons to be generated by this light source and the energy being distributed into the scene.

Photons can be emitted from either point lights, spotlights, or area lights.

To define a photon light source

1. Select the light you want to use as the global-illumination or caustic source and choose the **Modify** > **Shader** from the Render toolbar to display the selected light's property page.
2. Click on the Photon tab to display all of the light's photon properties.
3. Define the light as a **Global Illumination** source or a **Caustic** source (or both) by clicking in the appropriate check box.
4. Define a custom **Color** for the global illumination or caustic light by using the color sliders or the Color Editor (click the color box beside the light color sliders). For more information on defining colors, see *Defining Colors* on page 75.
5. Set the caustic energy **Intensity** for the light. Intensity is the option that creates the caustic effect. You can tune the brightness of caustics to make them either more or less visible by setting a value from 10 000 to 1 000 000.

It is important to understand the difference between color and intensity: *color* is the power of the direct illumination; *intensity* is the power of the caustic. You set the color of the actual light in the light property editor, but the caustic light color does not affect the color emitted by the light itself.

6. Define the number of **Emitted Photons** for your caustic or global-illumination effect using the slider.

   This value specifies the number of photons in the Photon Map (see page 158), so it's a good indication of the quality of the generated caustics. It is also a direct indication of the memory usage, which will be proportional to the number of photons in the Photon Map. For quick, low-quality caustics, a value of 10 000 is adequate. Medium-quality typically needs 100 000, and highly accurate effects may require up to 1 000 000.

   Lower values, in the case of global illumination, cause fuzzy puddles of light—this means there are too few photons hitting the surfaces in the scenes. In the case of caustic effects, the caustic looks blurry or indistinct.
7. If desired, repeat steps 1 to 6 for other lights that you want as sources for global illumination or caustics.

Setting Up Transmitters and Receivers

Defining which objects will transmit and which will receive is perhaps the most integral step in creating photon effects. The caustic or global-illumination effect you're trying to achieve greatly depends on an object's position to the light, its position to other objects, and its surface.

1. Select an object in the viewport that you want to set as a caustic transmitter or receiver. Remember to select a geometric object and not a light. For example, in the case of a magnifying glass, you would select the glass object rather than the light source.
2. Right-click the **Property** button on the Selection panel and select **Viewing Properties** from the drop-down menu. The Visibility property editor opens.
3. Select the Visibility tab to access the render-visibility options.
4. Select whether you want your object to receive or transmit global-illumination photons or caustic photons or any combination of the four.

There is no reason that an object cannot be assigned all four properties; that is, an object can be a global-illumination receiver and transmitter as well as a caustic transmitter and receiver. Defining all of your objects in this way, however, increases render times.

Before you render, you must activate global illumination and/or caustics for the render pass. To view the caustic and global-illumination effects in the render region, see *Displaying Photons in the Render Region* on page 164 and *Rendering Global Illumination and Caustics* on page 165.

Displaying Photons in the Render Region

You can see global-illumination and caustic effects in the render region. Because they take longer to render, they are turned off by default. To view the global-illumination and/or caustic effects in the render region, do the following steps:

1. Create a render region in a viewport. Make sure you include the area where caustic-light or global-illumination effects will be displayed.
2. Do one of the following:

   Choose **Render** > **Region** > **Options** from the Render toolbar to open the View Rendering Options property editor.

   or

   Right-click the render region's frame to display its pop-up menu and choose **Properties.**
3. Click the Photon tab, then select **Caustics** (this option automatically activates the Caustics option on the Active Effects page) and/or **Global Illumination**. The selected effect is now visible in the render region.

From this page, you can also set the Caustic Accuracy, Global Illumination Accuracy and various Photon Depths for your render. For more information on photon-rendering options, see *Rendering Global Illumination and Caustics* on page 165.

4. To modify the display of caustics in the region, you can generate a map of the caustic effect by selecting the **Rebuild Photon Map** option. For more information on this option, see *Rebuilding the Photon Map* on page 167.

Rendering Global Illumination and Caustics

To render global illumination and caustics, you must activate these effects for the render pass. But first you must define global illumination and caustics for specific objects and lights in your scene.

When rendering caustics or global illumination, use Grid acceleration—instead of BSP—as the rendering method. Your renders will be a little faster. For more information on render-acceleration methods, see *Acceleration Methods for Raytracing* on page 62 in the *Rendering* guide.

To activate photons in a render pass

1. Make sure you have already defined global illumination for specific objects and lights in your scene. For information on setting up lights for global illumination, see *Setting Up Transmitters and Receivers* on page 163.
2. Do one of the following:

   Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass (the current pass is the one displayed in the pass selection box on the Render toolbar).

   or

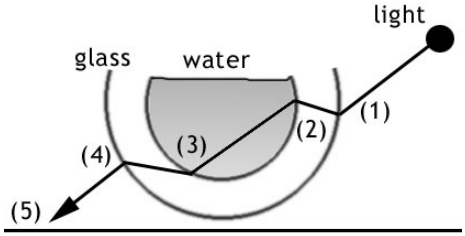
   Select a render pass in the explorer, right-click on the render pass, and choose **Render Options** from the pop-up menu. The Rendering Options property editor appears.
3. Click on the Photon tab to display the global-illumination and caustic property page. Select **Caustics** (this option automatically activates the Caustics option on the Active Effects page) and/or **Global Illumination**. The selected effect is rendered once a render is started.
   The rest of the settings are described in the following sections.

Photon Depth Options

The Photon Depth options define the depth of a photon, which means you are actually defining the maximum number of times a photon ray can be reflected and refracted—or bounced—in the scene.

The default is 2 for both reflective and refractive bounces. The Max. Depth parameter limits the total number of bounces. If you keep the default of 2, if a photon reflects off a diffuse wall (first bounce) and then a floor (second bounce), the photon is absorbed, even if its next bounce was to be through a refractive surface (like a glass).

If you study the scene, you can trace the way the photon bounces. For example, if a spot light illuminates a glass fish bowl, the photon will hit the bowl (1), go through the bowl (2), go through the water (3), go out the other side (4) and hit the table (5).



- **Photon Reflection** ray depth lets you specify the maximum number of a photon ray's reflected branches in a scene. For example, in a totally reflective scene, the photon ray continuously bounces around in the scene creating an infinite number of branches. This option lets you set an upper limit on these calculations.
- **Photon Refraction** ray depth allows you to set the maximum number of times a photon ray can be refracted in a scene.
- **Maximum Photon Depth** adds the total number of a ray's reflections and refractions in the scene. If the total exceeds the number you have specified, the branch is not cast. If the sum of the two numbers is an odd number, there will be more reflection rays because reflection always gets calculated first.

Photon Render Options

The Global Illumination and Caustic Accuracy settings help you get a very refined look to your scene.

- **Caustic Accuracy** is like a sampling parameter. It controls how many photons are calculated during rendering.
- **Radius** defines the size or scope of the sampling value to use on the caustic effect. A large value will wash out the photon effect.

Caustic Accuracy is looking for a specific number of photons stored in a given area. Starting from a point on a surface, accuracy looks at the surrounding area (controlled by the **Radius** parameter) and accumulates the number of photons specified in the **Accuracy** parameter. For a sharp and precise result, you need a high-accuracy and a low-radius value. If the radius is too big, it washes out the surface by including a too large sampling value—there's too much surface to evaluate at a given point. If the radius is too small, it takes longer to render.

If you specify more photons in the Accuracy setting than are stored in the Photon Map, searching will not stop.

The default accuracy is 100 and a default radius is calculated. The default radius is calculated based on the volume/size of the bounding box of the scene.

Generally, if caustics look spotty, increase the Accuracy value. The default is 100 photons. Note that higher numbers (200) will refine the effect, but they may also blur it. For some effects, like the highly focused light you see in the classic gold-ring example, lowering this value (as low as 10) makes the effect look very sharp; however, you may lose some refinement on the surface receiving the effect. Low settings may also help you create the large spots of light that you see at the bottom of a pool; but again, this setting may cause the other caustics in the scene to look too big and bright.

Caustic Filtering

You can also apply a filter with a step value to control the sharpness of the caustic. If you change the default Filter type (Box) to Cone and set the Constant value to 1.1, caustic effects may look sharper but noisier. The larger the Constant value, the blurrier the caustic effect will be.

Global Illumination Render Options

You can also set Accuracy and an accuracy Radius for global illumination. These parameters behave the same way as the caustic Accuracy parameters. If global-illumination effects look grainy, increase the Accuracy value. (The default is 200.) Note that higher numbers (2000) make the effect smoother but increases rendering time as well.

Rebuilding the Photon Map

To modify the display of caustics in the region, you can generate a map of the caustic effect by selecting the Rebuild Photon Map option (wait while the Photon Map is calculated and displayed). You can then zoom, pan, or play back your animation in the render region using the precomputed map of the caustics instead of computing the caustic effect each time the region is refreshed, which takes much longer.

You can also specify a Photon Map in the Map File Name text box. Click the browse (...) button to open a browser and select a map file.

Final Gathering

Think of it as a poor man's global illumination. Final Gathering is another way global illumination can be calculated without using photon energy. It calculates both direct and indirect lighting, using rays cast from the object's pixels rather than from the light. This method is particularly appropriate for scenes in which the light source is far from its targets. Think of a marble in the middle of a football field—you would need so many photons to get noiseless results that it may be impossible.

Please note that while Final Gathering may give more pleasing results under certain circumstances, it is also slower than standard rendering.

With Final Gathering, you specify a Minimum and Maximum Radius in the Render Options property editor. The radius identifies the hemisphere above the pixels that are sampled for both direct and indirect illumination. Photon Map contributes to the information Final Gathering "gathers" about the indirect illumination.

For example, scenes with large, open spaces will use a larger radius, while scenes with very small objects require a smaller radius. The larger the radius, the better the chance the final-gathering effect will hit an object in the scene. Rendering will be slower, since a larger number of objects are hit by a final-gathering ray.

An Accuracy parameter specifies how many rays should be fired from each pixel to calculate the indirect illumination. The default is 30, which yields a poor quality but is fast to render. A higher-quality Accuracy is between 500 and 1000.

With Final Gathering on, you should need fewer photons in the scene.

Chapter 7 Camera Basics

The best way to make an audience see what you want them to see is to get the camera to do the work. The camera in SOFTIMAGE|XSI is analogous to a physical camera in the real world. It defines the view that you can render. In addition, you can add as many cameras as you want in a scene.

Cameras and Viewpoints

There are both cameras and viewpoints. A *viewpoint* allows you to see a scene in a viewport in a specific way. You can choose from four default viewpoints in the viewport: User, Front, Top, and Right. For more information about selecting views, see *Chapter 4: Viewing Your Work* in the *Fundamentals* guide.

In many ways, cameras and viewpoints are similar, except that viewpoints are not actual objects: they are only tools for viewing your scene. You cannot animate a viewpoint, nor can you render from a viewpoint as you can from a camera.

By selecting a camera view in a viewport, you can see what the camera sees. The active camera is the camera that is associated with a particular render pass. For information about setting the active camera for a render pass, see *Controlling the Active Camera in the Render Pass* on page 32 in the *Rendering* guide.

Multiple Cameras

Nothing stops you from adding a second, third, or fourth camera to your scene. In fact, there is no limit to the number of cameras you can use to view your scene or render it.

Lens Shaders

A lens shader is a rendering subroutine that affects rays as they are emitted from the camera. Lens shaders let you achieve effects like fish-eye views, depth of field, and lens flares or cartoon effects.

You can apply a number of lens shaders to a camera. You apply lens shaders the same way you would any other shader or use the lens-shader stack. For more information about applying shaders, see *Attaching Shaders to Elements* on page 41.

If you load a scene with shaders attached to the camera, they appear as subnodes of the camera. You can also apply additional shaders or detach any existing lens shader.

Creating Cameras

You can create additional cameras and work with cameras from imported scenes. Cameras created in SOFTIMAGE|XSI are saved to the scene file and are not saved back to the imported 3D scene. You can view and render your scene through any camera.

To create a new camera with a defined projection

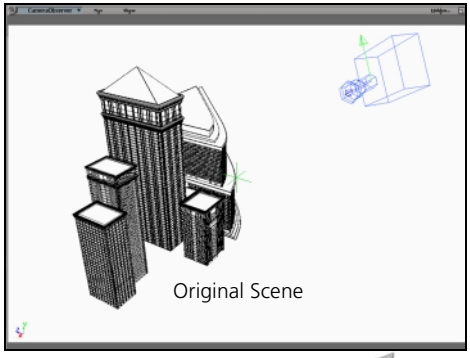
You can also create a new camera with a defined projection directly from the Render toolbar.

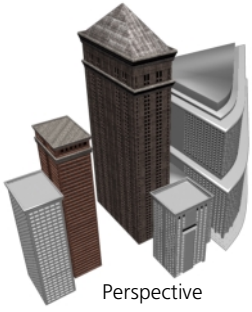
1. Choose **Get** > **Camera**. A pop-up menu gives you the following list of cameras:

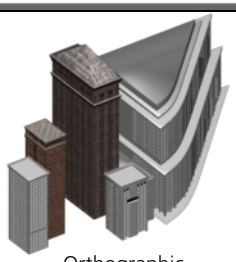
(Default) Perspective	Uses a perspective projection, simulating depth. Useful for simulating a physical camera.
Telephoto	Uses a perspective projection and a small angle of view (5°) to simulate a telephoto lens view.
Wide Angle	Creates a wide-angle view by using a perspective projection and a large angle (100°) of view.
Orthographic	Uses an orthographic projection. All camera rays are parallel, and objects do not change size as they change distance from the camera. This projection is useful for architectural and engineering renderings. The depth of field settings on the DOF shader property editor have no effect with an orthographic projection.

2. A new camera, with its preset projection, is added to the scene and positioned at the origin. The property editor for the new camera is displayed once a camera is selected. Adjust the values if required.
3. Select and manipulate cameras as you would any other object. You can use its manipulators to rotate or translate the camera (press **b**).

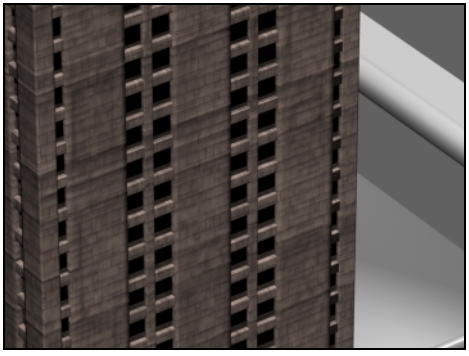
Each of these images uses the same scene with a different camera in the same position (right). Each of the four camera types available from the toolbar create a new camera with a different perspective.

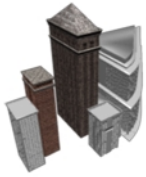



Perspective

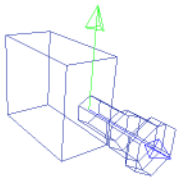

Orthographic

The Default perspective camera view (above left) creates a projection that simulates depth. The Orthographic camera (above right) causes all of a scene's lines to be parallel. Objects remain the same size regardless of the camera's distance. The Telephoto view (lower left) has a much smaller angle of view than the Default camera, but objects are "zoomed." The Wide Angle camera has a much larger field of view than the other cameras and, in some cases, distorts the perspective.


Telephoto


Wide Angle

Using the Camera Icons



In a viewport, cameras are represented by camera icons. Arrows indicate the camera's interest and up direction. The camera icons are only a visual reference, and they are not rendered when you draw a render region, preview a frame, or render.

You can select and move cameras in a viewport as you would any other object. For more information, see *Chapter 5: Working with Scene Elements* in the *Fundamentals* guide.

You can also keyframe a camera's viewpoint as described in *Chapter 2: Animating with Keys* in the *Animating* guide.

The Camera Interest

The camera's interest—what the camera is always looking at—is represented by a null in a scene. You can translate and animate the null as you would any other object.

Camera Direction

The camera icon displays a blue and a green arrow. The blue arrow shows where the camera is "looking"; that is, the direction the lens is facing. The green arrow shows the camera's up direction. You can change its direction by rolling the camera (**l** key).

Distance to Camera

You can see how far an object is from the render-pass camera (or output camera) in any or all viewports. This option can be useful when trying to build a scene or set up your lighting. It is also used to help define clipping-plane distances. For more information on how to set up clipping planes, see *Setting Clipping Planes to Hide or Display Objects* on page 180.

To display the object-to-camera distance

There are two ways:

- From a viewport's command bar, select **Show** > **Distance to Output Camera**. This displays the distance in the selected viewport.

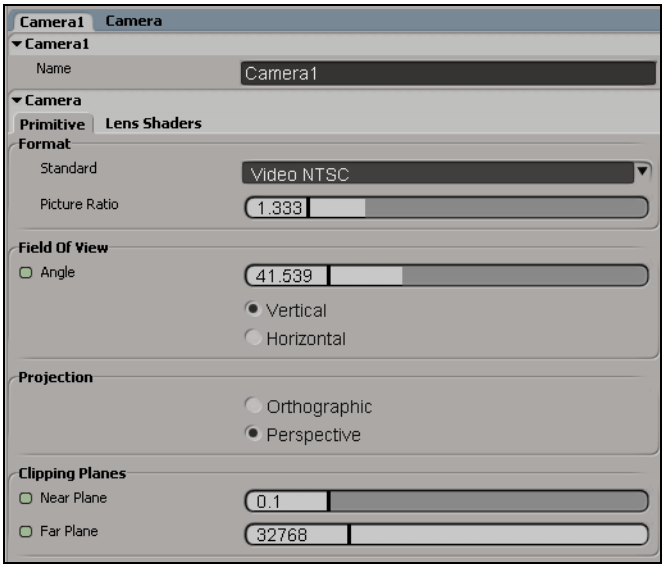
  or

- From the menu bar, select **View** > **Distance to Output Camera**. This displays the distance in all viewports.

Once activated, the Distance to Output Camera tool displays a reading of how far a selected object is from the render pass camera. The value is displayed in SOFTIMAGE units.

Opening the Camera Property Editor

The Camera property page contains every parameter needed to define how a camera will "see" your scene. It can be accessed by clicking its icon in the explorer or double-clicking its node in the render tree.



To open the camera property editor

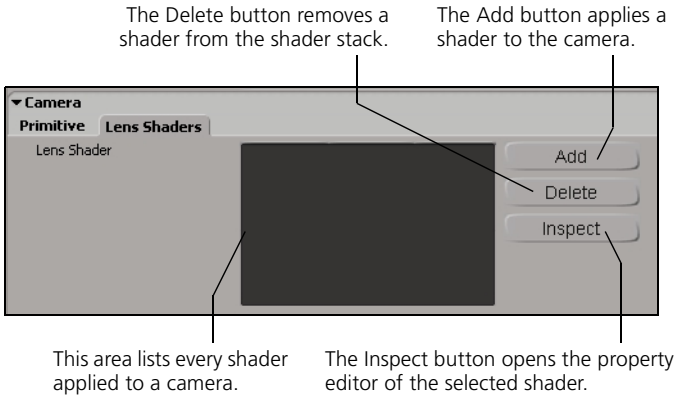
1. Select a camera whose properties you want to edit.
2. Click **Modify** > **Shader** from the Render toolbar; the selected camera's property page will open.

When you open a camera property page, you will notice two tabs above the page. The first tab simply relates to the Camera Name field, the second tab corresponds to the actual camera-parameters property page.

The Lens Shader Stack

Although the shader stack is useful for assigning multiple shaders to a camera, you will still need to use the render tree if you wish to build a complex tree of shaders.

- To display the shader stack editor, click the Lens Shader tab. From here you can define which shaders the camera will use and edit their properties.



For more information on extending shaders with the render tree, see *Where to Start?* on page 218.

Selecting a Camera View

Camera views let you display your scene in a viewport from the point of view of a particular camera. If you have created more than one camera in your scene, you can display a different camera view in each viewport (a maximum of four viewports can be displayed in your workspace). You can also choose to display the viewpoint of a camera associated to the current render pass or a different camera for each pass. For more information, see *Chapter 2: Passes & Partitions* on page 23 in the *Rendering* guide.

Selecting a camera or the Render Pass command from a viewport's Cameras menu switches the viewpoint to that of a "real" camera in your scene. Every camera that has been created or imported with your scene is available from any viewport views menu. All other views such as User, Top, Front, and Right are orthogonal viewpoints and are not associated to an actual camera.

To display a camera viewpoint

- Choose **Cameras** from the viewport's views drop-down menu and select from a submenu of cameras available in the scene.

Choose Render Pass to switch to the camera view defined for your render pass. This camera is defined in the render options property editor

Aside from the user-created cameras, you can select a pre-defined orthographic view.

To display the viewpoint of the camera associated with the current render pass, choose **Cameras** > **Render Pass**.

The Render Pass view is also a camera view. It shows the viewpoint of the camera associated with the current render pass. Only a camera associated with a render pass is used in a final render. You can define which camera a render pass will use by selecting **Render** > **Render** > **Options** from the Render toolbar, then setting camera options in the **Format** tab of the rendering options property editor.

You can also define which camera will be used to render which frames. for example, you can use camera A to render frames 1 to 30 and camera B to render frames 31 to 100. For more information on how to define a camera to render from, see *Controlling the Active Camera in the Render Pass* on page 32 in the *Rendering* guide.

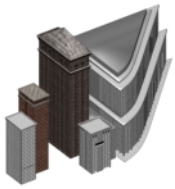
The Pass Selection box on the Render toolbar indicates which render pass is current for the scene. For more information, see *Chapter 2: Passes & Partitions* on page 23 in the *Rendering* guide.

Selecting a Projection Method

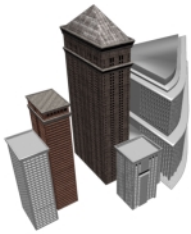
You can easily change any camera from an orthographic to a perspective projection.

To select a projection

1. Select a camera from a viewport.
2. Choose **Modify** > **Shader** from the Render toolbar. The selected camera's property editor opens.
3. From the property editor, you can select one of two projection methods.
   - **Orthographic** uses an orthographic projection, in which all camera rays are parallel and objects do not change size as they change distance from the camera. This projection is useful for architectural and engineering renderings. The depth of field settings on the DOF property page have no effect on orthographic projections.
   - (**Default**) **Perspective** uses a perspective projection, simulating depth. This projection is useful for simulating a physical camera.



Orthographic projection



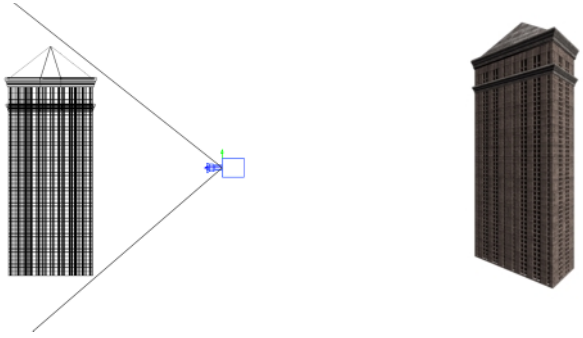
Perspective projection

Setting the Field of View

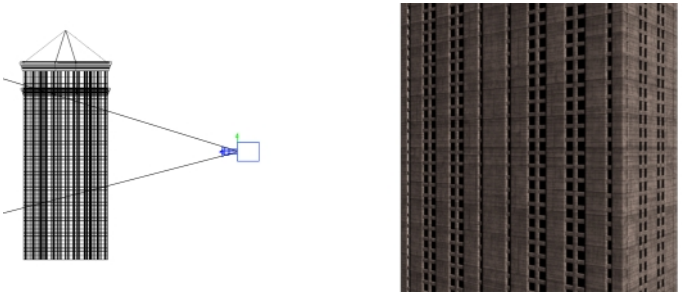
The field of view is the angular measurement of how much the camera can see at any one time. By changing the field of view, you can distort the perspective to give a narrow, peephole effect or a wide, fish-eye effect.

To set the field of view

1. Select a camera from a viewport and choose **Modify** > **Shader** from the Render toolbar. The selected camera's property editor opens.
2. From the Field of View section, select whether you want to specify the **Vertical** or **Horizontal** field of view. The other dimension is automatically calculated using the aspect ratio.
3. Set **Angle** to the desired value in degrees.



The camera's Vertical field of view was made large enough to accommodate the entire building. The Horizontal field of view was automatically calculated according to the aspect ratio.

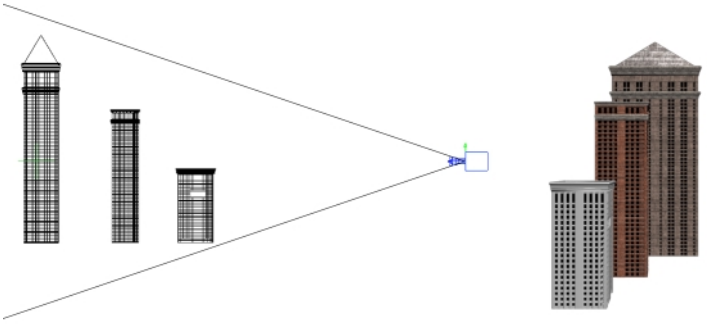


Using the same camera in the same location, the Vertical field of view was made much smaller, thus making only a small part of the building visible.

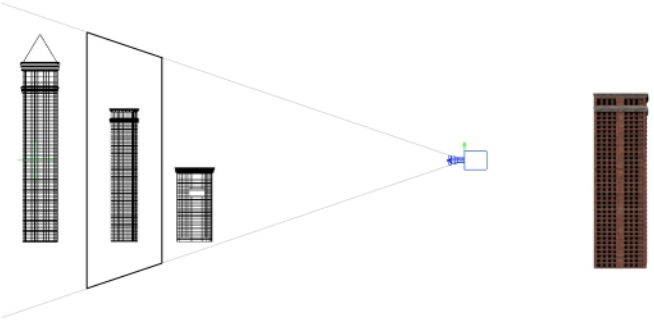
Each camera projection type has its own default field of view. For example, a wide angle projection has a large field of view, whereas a telephoto camera has a smaller field of view.You can edit these cameras' fields of view at any time.

Setting Clipping Planes to Hide or Display Objects

You can use clipping planes to set the minimum and maximum viewable distances from the camera. Objects outside these planes are not visible. By default, the near plane is very close to the camera and the far plane is very far away, so most objects are usually visible. You can set clipping planes to display or hide specific objects.



A camera with no clipping planes set—which means the resulting image (right) is every object in the scene.



A similar camera with near and far clipping planes set. The near plane is between the first two buildings and the far clipping plane is between the last two buildings. Everything before the first plane will not be visible, and everything beyond the far clipping plane will also be invisible, as seen in the resulting image (right).

To set clipping planes

1. Determine acceptable minimum and maximum distances from the camera.

To help determine what distances are acceptable within a scene, you can use the **Show > Distance to Output Camera** option from a viewport's command bar. The distance is displayed in SOFTIMAGE units.

2. Select a camera and display its property editor by clicking **Modify > Shader** from the Render toolbar.
3. Set the **Near Plane** and **Far Plane** values according to the near and far distances you calculated.

You can sometimes reduce rendering time by choosing appropriate clipping planes. By default, the near clipping plane is very close to the camera, and the far clipping plane is far away. However, if an image contains distant, complex objects, you may want to set the far clipping plane so that these objects are not rendered.

You can also enhance the depth precision (or resolution) displayed in the Open GL views if you make the near and far clipping planes closer to your objects. For more information on how to set OGL view options, see *OGL Display Settings* on page 127.

Setting Aspect Ratio and Shutter Speed

Aspect ratio and shutter speed are global rendering options that are set specifically for the pass using the Render Options property editor.

Aspect Ratio

The aspect ratio is the ratio between the width and the height of the image. It is very important to set the aspect ratio when you first start working on a scene. To ensure that what you see in a window accurately reflects what will be within the frame in the final render, you can load the camera's ratios into the render setup. For more information on render passes and their settings, see *Chapter 3: Rendering Options* on page 51.

The aspect ratio is also set in the rendering options of a render pass. For more information, see *Specifying the Format Options (Global Only)* on page 57 in the *Rendering* guide.

Setting the Shutter Speed for Motion Blur

If you are using motion blur in your scene, you must set the shutter speed of the camera from the Rendering Options property editor to see the motion blur effect. For more information about using motion blur, see *Chapter 8: Blurs, Flares & Other Effects* on page 185.

Resetting a Camera's Position

You can reset the camera's position and aspect ratio to its default values.

To reset a camera's position

1. Select the camera whose position you want to reset.
2. Choose **Transform** > **Reset all Transforms** on the Transform panel. This returns the object you just transformed back to its original size, orientation, and location.

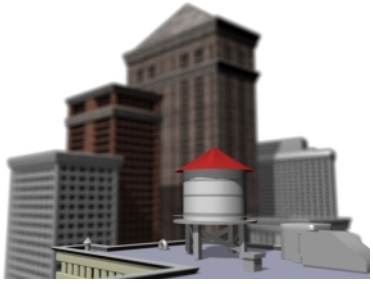
   If you have done a series of transformations to an object, you can reset only the currently active transformation. **Transform** > **Reset Active Transform** resets the currently active transformation you applied to your object back to its original size, orientation, or location.

You cannot undo this command. Saving your scene after using this command saves it with the new transformations.

Creating Depth of Field

In real life, it is rare that all objects around us are in focus all the time, so why should it be any different in your scene? In order to add a sense of realism to a scene, you can add some depth of field. Depth of field is an effect that simulates a plane of maximum sharpness and blurs objects in front of, close to, or beyond this plane.

High Depth of Field
This scene has a far depth of field setting so that only objects farther from the camera are in focus.



Low Depth of Field
This scene uses a near depth of field setting so that only objects close to the camera are in focus.

There are two ways of using depth of field:

- You can use a lens shader such as a Depth of Field shader that takes multiple samples using different paths to reach the same point on the focus plane to interpolate the depth effect.

  or

- You can use a combination of a volume shader and an output shader that collects depth information during rendering and then applies a blurring filter as a post-processing step over the finished image.

Using the Depth of Field Shader

A camera imported from a SOFTIMAGE|3D scene may already have a depth of field shader attached to it. You can also apply any of the depth of field shaders available from the shader library within SOFTIMAGE|XSI.

To set depth of field

1. Select a camera and display its property editor (**Modify** > **Shader**).
2. Click the Camera tab and set the camera's projection method to **Perspective**.
3. Select the Lens Shaders tab; the lens shader stack appears. Click **Add** to open a browser.
4. From the shader library, select the depth of field lens shader from the Lens folder and click OK to attach the shader to the selected camera.
5. To edit the lens shader's parameters, open its property editor by clicking the **Inspect** button while the shader is selected in the shader stack.
6. Set its parameters such as the **Focal Length**, **F-Stop**, and focal **Distance** to the values you want.

   These settings work the same way as on a physical camera. The options define the focus (sharpness) of objects according to their distance from the camera, similar to the way a real camera works. Depth of field lets you set the minimum and maximum distance from the camera for objects to be in focus. Objects closer than the minimum distance and farther than the maximum distance become increasingly out of focus.

For more information on setting parameters for the depth of field lens shader, click the online help button (left) in the shader's property editor.

Chapter 8 Blurs, Flares & Other Effects

Creating Motion Blur

You can easily achieve a photorealistic motion-blur effect per object. Motion blur gives a natural look to fast-moving objects. It is also used as an alternative to field rendering for reducing the strobing effect.

By default, motion blur is off for every object in a scene.

Defining Motion Blur

To view the motion blur of an object, simply switch on the motion blur settings in either the render-region options or the render-pass options or both. As long as these options are on and you have a moving object in your scene, the motion blur will be visible.

To edit the motion blur of a specific object in a scene, you must assign it a Motion Blur property.

To create the motion blur property

1. Select the object for which you want to create the motion blur property.
2. From the Render toolbar, select **Get** > **Property** > **Motion Blur**. This creates a motion blur property for the selected object.The motion blur property editor appears.
3. From this property page, you can toggle the motion blur **On** and **Off**. The motion blur property has been defined for the object and you can now edit its parameters. For more information on setting a motion blur's values and render options, see *Rendering Motion Blur* on page 188 and *Rendering Motion Blur* on page 188.

Motion Blur Property for Groups

If you wish to remove the motion blur from more than one object in a scene, repeat the same steps, but first create a group with all the non-motion blurred objects. Then open the explorer and expand the group. You should see the motion blur subnode beneath the group. Proceed as above.

Rendering Motion Blur

There are two different rendering methods with which to render motion blur with mental ray rendering software: raytracing and scanline.

While motion blur is useful at times, it can significantly increase rendering time.

You can also use the motion blur standalone tool to render motion blur effects. For more information, see *Creating 2D Motion Blur* on page 266 in the *Fundamentals* guide.

- **Raytracing** correctly blurs attributes applied to moving objects, such as highlights, textures, shadows, and reflections as well as refractions, transparency, and intersecting objects.
- **Scanline** rendering applies a quick local illumination motion blur. This is much faster, but it cannot handle reflections and refractions (and shadows must be approximated with shadow maps). However, motion blur of highlights, textures, transparency, and intersecting objects still work with scanline rendering.

Raytracing

Scanline

Although the differences are subtle, they are quite visible. Raytracing motion blur (above) is clearer and more accurate. The scanline motion blur (below) is a quicker motion blur that cannot calculate reflections and refraction on a moving object. Scanline motion blur renders more quickly than raytracing.

Either one of these rendering methods can be defined on the Optimization tab of the Rendering Options property editor. For more information on these rendering methods, see *Chapter 3: Rendering Options* on page 51 in the *Render toolbar Rendering Guide*

For a more realistic effect, combine motion blur with antialiasing. For more information about antialiasing, see *Antialiasing and Motion Blur* on page 191.

Setting Motion Blur Options

1. Choose **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass (the current pass is the one displayed in the Pass text box on the Render toolbar).

   or

   You can also repeat the following steps for the Render Region. Open the Render Region options property editor by selecting **Render** > **Region** > **Options** from the Render toolbar.

2. Click on the Motion Blur tab and select **Motion Blur**. Switching on the motion blur on this page automatically activates the motion blur option on the Active Effects page.

3. On the Optimization tab, select **Raytracing** to use the raytracing algorithm when rendering motion blur. If you want to render a fast scanline motion blur, select the **Scanline** option and deselect **Raytracing**. By default, both Scanline and Raytracing are activated.

4. Set the **Shutter Speed** value. This specifies the length of time the shutter is open. The shutter speed defines how long the camera's shutter remains open.

Longer shutter speeds (>0.6) create a wider and/or longer motion blur effect simulating a faster speed. Shorter shutter speeds (<0.3) create more subtle motion blurs. Of course, all shutter speeds are relative to the speed at which the blurred object is animated. Objects that move from one end of a scene to another in a few frames produce a large amount of motion blur regardless of the shutter speed. A value of 0 turns motion blurring off.

The value is in time, emulating the shutter on a camera being open for a percentage of time between frame x and x + 1.

The speed at which an object is animated has the greatest impact on the motion blur effect. But the blur can be made more subtle or enhanced using different shutter speeds. In the first image (left), a quick shutter speed (<0.1) is used, then a slower shutter speed (middle), and finally (right) a very slow shutter speed (>0.6).

Shutter speeds higher than 1 will produce intense motion blurs, but may cause artefacts to appear in a rendered image.

5. Set the **Motion Blur Sampling Threshold** by adjusting the RGBA sliders. If the difference between neighboring samples is greater than this level, another level of sampling occurs.

The **Motion Blur Sampling Threshold** determines how often the motion-blurred object will be sampled during the rendering process. Although each process is similar, the motion blur and antialiasing techniques are not computed at the same time.

The smaller the value for each color (R, G, B), the greater the amount of sampling that must be done to close the contrast gap between the rendered image and the threshold settings, and the smoother the motion blur.

Both of these images have the same animation and motion blur. The image on the left uses a low Motion Blur Sampling Threshold, whereas the image on the right uses a higher value. Notice how the image with a higher sampling value has more contrast and detail than the image with low sampling. Keep in mind that higher sampling levels take longer to render.

If you're also using antialiasing, set the **Motion Blur Sampling Threshold** values higher than the antialiasing **Sampling Threshold** values (on the Aliasing property page of the same property editor) to speed up motion blur rendering. This results in more grainy images, but the quality of antialiasing is not degraded.

For imported scenes, motion blur works only if the scene contains motion vectors. Make sure that the .dsc or .mi file has enough motion vectors to produce motion blur.

If importing a scene from SOFTIMAGE|3D, make sure you the Linear/Exact motion blur flag is on and that you have saved the scene with Motion Blur on.

6. If you prefer to have a "quick and dirty" motion blur, select the **Estimated Motion Blur** option.

Regular (raytraced) Estimated

7. Select **Deformation Motion Blur** when the standard SRT motion blur isn't accurate enough for your scene. Deformation motion blur computes a motion blur any time a vertex is animated.

Deformation motion blur is off by default because it consumes a lot of memory. The most efficient way to work with deformation motion blur is to use localized motion blur properties, as in the following example:

If you have a large static set and three characters in a scene, your best option is to go to the root of the scene and choose **Get** > **Property** > **Motion Blur**. The property page that appears will have blur turned on and deformation blur turned off. Place your characters in a group and put a motion blur property onto the group (or put properties on each separate character) and then turn on deformation blur in that property page. In your render settings, make sure that you turn on motion blur *and* deformation blur. Only the characters use the extra memory required to blur them, while the rest of the scene remains static. If a character picks up a prop, for example, the movement will be SRT based, so the moving prop will blur.

8. Selecting **Motion Blurred shadow maps** activates the shadow mapping option for motion blur effects. Shadow maps are a scanline-based feature that produce shadows faster than raytracing. Activating them with motion blur slows render times but matches the shadow to the motion blur nicely. If you are running a quick or test render, switch this off.

Antialiasing and Motion Blur

You can use antialiasing on motion blurred objects to heighten the level of realism. Applying antialiasing to an object that has a blur on it determines if more subdivisions of the pixel are needed, based on the antialiasing Threshold Level. The **Motion Blur Sampling Threshold** determines how often the motion-blurred object is sampled during the rendering process. Although each process is similar, the motion blur and antialiasing techniques are not computed at the same time.

For more information on antialiasing, see *Controlling Aliasing* on page 66 in the *Rendering* guide.

Creating Lens Effects

Lens effects—such as lens flares—are distortions of the rendered image achieved by changing the light path through the camera lens. These effects are achieved by attaching one or more lens shaders to the camera. If no lens shaders are present, a standard camera is used.

Any lens shader defined in a Render toolbar scene is attached to the camera when imported into Render toolbar. The lens shader appears as a subnode of the camera object node in the Explorer view.

A Render toolbar lens shader automatically turns off scanline rendering because it uses raytracing.

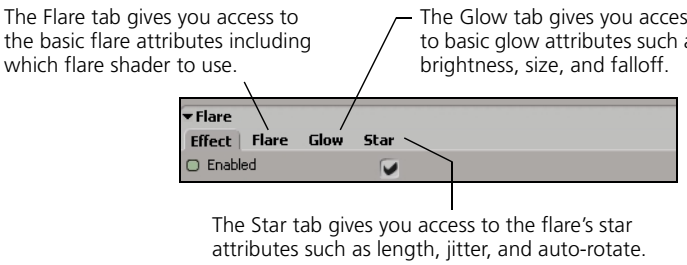
If lens shaders change the origin or direction of a ray, they only work correctly if scanline rendering is turned off.

Creating a Lens Flare

Lens flares occur when a light (usually point) source hits the camera's lens at a small enough angle to show lens "aberrations."

To create a lens flare

1. Select the light you want to create a flare with.
2. From the Render toolbar, click **Get** > **Property** > **Lens Flare**. The lens flare property editor opens.

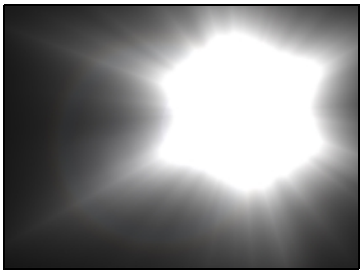


3. Switch the flare effect by clicking the **Enabled** check box. You can still edit a flare's parameters if it is off.
4. Select the Flare tab and choose a flare shader from the **Flare File** drop-down menu. As you can see, there's quite a variety of flares to choose from.

The following reference chart can help you decide what kind of flare you want to apply to your light.

Classic lens flare.

Long Ray lens flare

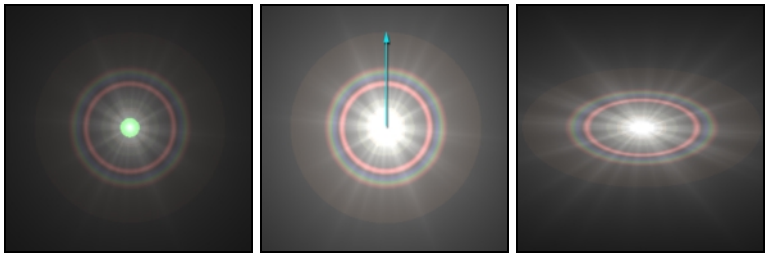


Real lens flare

Sharp Ray lens flare

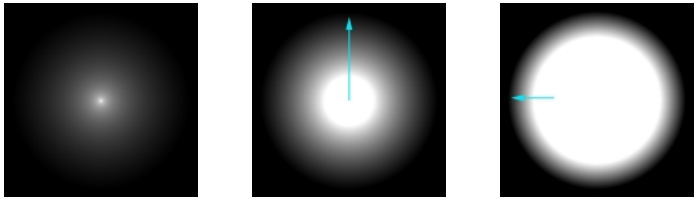
Short Ray lens flare

5. On the Flare property page, edit the flare's size, brightness, and aspect ratio.



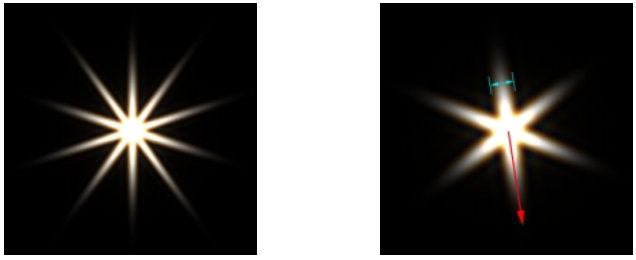
All three scenes above use the same flare type. The left scene is the default flare.The middle one uses increased Size and Brightness values. Notice how the flare's colors become more saturated. The scene on the right "squashes" the effect by editing its Aspect Ratio value.

6. Open the flare's Glow property page to define its glow properties.



The scene on the left is the default glow of a flare (no stars). The middle one shows a glow with a higher brightness value and larger size (glow radius). The scene on the right shows the same glow but with almost no falloff.

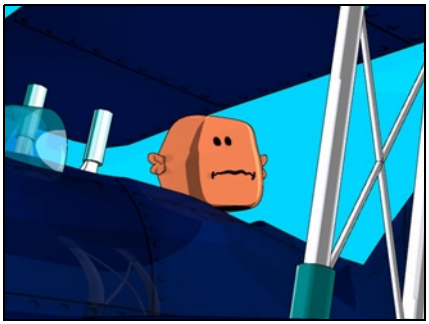
7. Open the flare's Star property page to define its star rays and their properties. You can edit the number of rays, their length (red arrow), and width (blue arrows).



The scene to the left shows a default lens flare with the Star property on.

Creating a Cartoon Effect

In order to give your scene a 2D, cel-animated cartoon look, you must follow two simple steps. First, your objects must use the Toon material shader so they can be rendered as a cartoon. Second, you must apply a cartoon shader to the scene's camera so it "sees" everything in cartoon style.



To create a cartoon effect

1. Select the object or objects to be rendered in cartoon style.
2. Apply a Toon Material shader to each object that will be rendered in a color of your choice. You can quickly apply the shader by selecting **Get** > **Material** > **More,** opening the Soft3D folder, and selecting the Toon Material shader.

To see the effect, you need to apply a Cartoon shader to the camera.

3. Select the camera (or cameras) you will use to render your scene.
4. From the Render toolbar, choose **Modify** > **Shader** to open the camera's property editor.
5. Click the Lens shader tab to open the camera's shader stack.
6. Click the **Add** button on the shader stack and select the Cartoon shader from the Lens folder of the shader library. Click OK.
7. Open the lens shader's property page by selecting it from the shader stack and clicking the **Inspect** button.

For more information on the parameters for the Toon and Cartoon shaders, see the online help in their property editors.

Volume and Environment Effects

Picture this: A camera gently dollies through a forest, panning up to the canopy above, barely seeing the sky above.

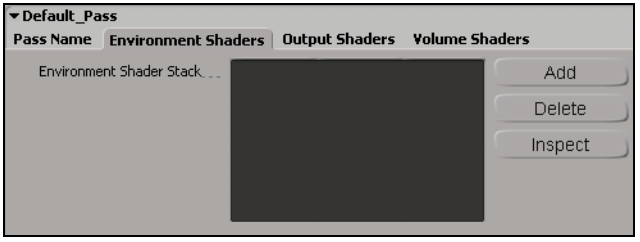
Now let's add some shader effects: A camera gently dollies through a fog-shrouded forest, panning up to the canopy above, taking in the dramatic sunset and faint, but glowing, moon.

Some effects can be added to a scene directly from the Render toolbar, such as volumic lights, glows, and lens flares, instead of assigning individual shaders to several objects. For more information on how to apply a lens flare, see *Creating a Lens Flare* on page 192.

Volume, output, and environment shaders are used to create fog, smoke, glows, halos, realistic backgrounds, and many other effects in a scene that add a sense of realism, drama or fantasy.

BBC "Everyman": Animation by Aldis Animation

While these shaders produce a wide variety of effects on a scene, they are almost all applied in the same way. You can apply any combination of volume, output, and environment shaders to either an object, a group, or a pass. In the case of a pass, a shader stack is used to apply these three types of shaders.



The shader stack lets you select any combination of environment, volume, or output shader. You can also determine in what order each shader is processed. You can open any shader's property page and edit its values using the Inspect button.

If you wish to apply one of these shaders to a pass or object(s), you must first define a pass. You can then apply any type of volume, environment, or output shader to a pass.

- To learn more about creating passes in your scene, see *Creating Render Passes* on page 29 in the *Rendering* guide.
- To learn more about how to apply a volume shader to an object or scene, see *Volume Shaders on page 201*.
- For information on how to create a volumic light, see *Creating a Volumic Light* on page 197.
- To learn more about how to apply an environment shader to a scene, see *Environment Shaders* on page 203.

Using Implicit Objects

Implicit objects are basic shapes defined by a mathematical formula. They cannot be rendered, but they can be used to define bounding boxes for volume effects. For example, if you want to contain some smoke in your scene to a specific area, you would simply assign the smoke shader to an implicit geometric object.

Implicit objects can be created from any toolbar (**Get** > **Primitive** > **Implicit**).

Creating a Volumic Light

You can create a volumic light using volume shaders, but a simpler method is to create one directly from the Render toolbar. Volumic lights created from the Property button are easily edited and can be deactivated or rendered with the click of a mouse.

This scene was created using a single volumic light. It also uses Shards. The Force Volumic Shadows parameter was on, thereby making the leaf block some of the volumic light.

To create a volumic light

1. Select an existing light or create a new one (**Get** > **Light**).
2. From the Render toolbar, select **Get** > **Property** > **Volumic Light**. The Volumic Light property editor opens on the Volumic page.
3. The principle parameter to edit when defining a volumic light is Map Size. Map Size determines the depth map resolution of the effect. If the objects the volumic light is crossing are simple, the map volume can be set very low (100 to 200). But if the objects in the light's path are complex, such as very sharp edges, you need to increase the Map Size value to pick up more of the object's shape detail, creating a better volumic effect.

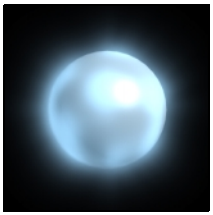
Increasing the Map Size slows rendering.

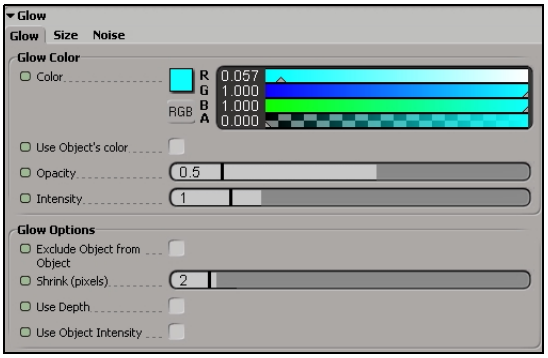
4. Adjust the **Reflectance** parameter to simulate the volume's reflectivity. A low value (0.5) simulates a volumic effect with low reflectivity, such as dust. A higher value (> 2) creates a reflective volume, such as dense smoke.
5. You can set the distance from the light source at which the volumic effect begins by using the **Minimum Distance** slider.
6. Select the **Force Volume Shadows** check box to force the rendering of shadows of the volumic effect. When this is not selected, volumic lights appear to go through objects. When on, objects occlude volumic lights and cause shadows.
7. To make the volumic effect have a transparent quality, select **Transparent**.
8. Click the Shards tab to access the volumic light's shard parameters. They are off by default. From this property page, you can activate shards, set their intensity, complexity, and resolution.

For more information on the parameters for the Volumic light shader, see the online help in its property editors.

Creating a Glow Effect

You can quickly create a post-process 2D glow effect to any object or group of objects directly from the Render toolbar. The glow created from the Property button can be applied to any object in a scene and switched off or rendered with the click of a mouse.

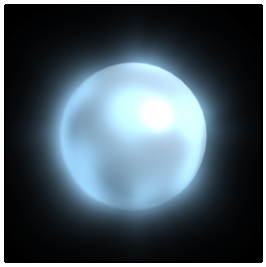




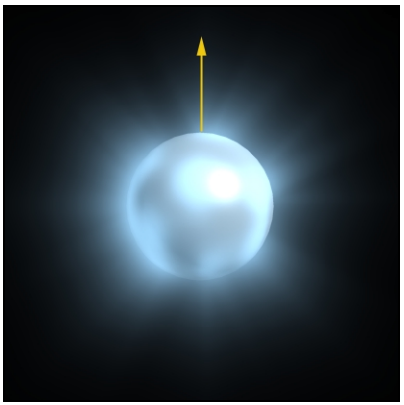
Remember that, since this effect is post-process, it is added to the scene only after rendering is completed (after the last tile is rendered).

To create a glow

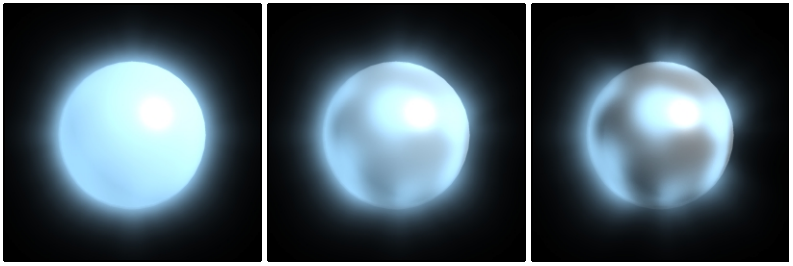
1. Select the object to which you want to apply a glow effect.
2. Create a render region in a viewport (press **q**).
3. From the Render toolbar, select **Get** > **Property** > **Glow**. The Glow property page opens in a floating window.
4. You can determine the color of the glow by adjusting the RGB sliders. You can also set the size and texture noise of the effect from the Size and Noise property pages in the Glow property editor.



Default glow



You can increase the size of the glow (above right) as well as alter its noise-level amplitude.



0.1 Amplitude 0.5 Amplitude 0.9 Amplitude

For more information on a glow's properties, refer to the online help from the glow property editor.

5. Close the property editor. The glow is applied to the selected object once it has finished rendering.

Volume Shaders

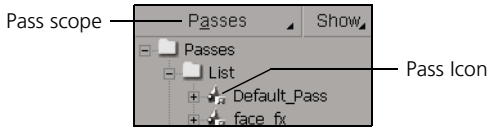
Volume shaders are used to create everything from smoke and fog to lava and slime. There are two basic ways to apply a volume effect: on the scene or on an object.

Creating a Volume Effect in a Scene

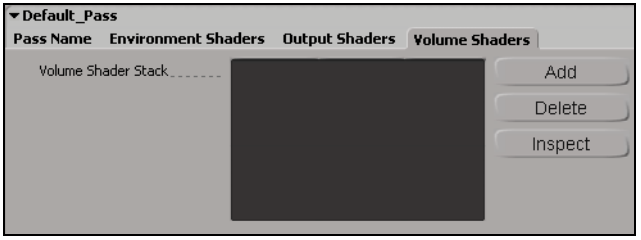
When you apply a volume shader to a render pass, it surrounds the scene rather than being bound to a single object.

To create a volumic scene effect

1. From an explorer, set the scope to Passes. All of your scene's passes are listed. If you haven't created any passes, only the Default Pass is available.



2. Select the pass (or default pass) for which you wish to apply the background, and click the pass icon to open the Pass property editor.



3. Select the Volume Shaders tab and click **Add** to apply a volume shader to the selected pass's shader stack.
4. From the browser, select a volume shader from the shader library. The following is a list of volume shaders that can be applied to a scene:

Shader	Creates a:
Volume_effects	Fog/dust-like effect.
Volume_fog	Fog effect in a scene.
Volume_layered_fog	Fog effect with vertical start and stop parameters.
Volume_cigarette_smoke	Cigarette-like smoke effect.

5. Click OK when you have chosen a shader—it appears in the shader stack list. Select the shader from the stack window and click **Inspect**. This opens the shader's property editor. For more information on how to use a particular volume shader, click the help (?) button on the shader's property page.

Creating Volume Effects with an Object

Depending on whether you want the object to be rendered or just act as a bounding box, you can select either an explicit (rendered) or implicit (unrendered bounding box) geometric object. For more information on the use of implicit objects, see *Using Implicit Objects* on page 197.

To create an object-based volumic effect

1. Select an object (implicit or explicit) to which you will apply a volumic effect.
2. From the Render toolbar, select **Get** > **Shader.** Click Volume to apply a shader to the volume input of the material. A list of available shaders appears. Select one of the following volume shaders:

Shader	Effect
March Fractal	Creates a huge range of volumic effects on either a geometric object or light.
March smoke	Creates a cigarette-like smoke effect.
Fog	Creates a fog effect on an object.
Layered Fog	Creates a fog effect with vertical start and stop parameters
Volume_lightning	Creates lightning within an object.
Constant_density	Defines a density within an object.
More...	Opens a browser from which you can select other types of volume shaders from the shader library.

3. Once you select a shader, it is attached to the object's volume input, and its property page appears in a floating window. You can edit the volume effect's parameters from there. For more information on parameters for a particular volume shader, click the help (?) button on the shader's property page.

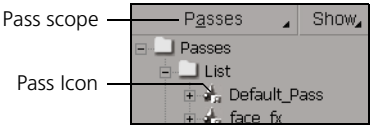
Environment Shaders

Environment shaders are mainly used to create an environment that completely encompasses a scene. Environment shaders are unique because they don't have to be applied to an object but they still simulate a geometric object's characteristics. They can be applied to a geometric object—a sphere surrounding a scene, for example—or a scene's render pass.

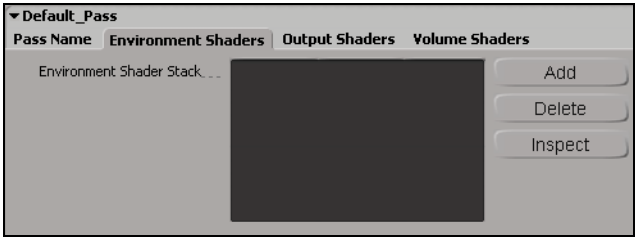
Creating a Background

The most common use for an environment shader is to create a background that is visible from any camera angle at any distance.

To create a background



1. From an explorer, set the scope to Passes. All of your scene's passes will be listed. If you haven't created any passes, only the Default Pass is available.
2. Select the pass (or default pass) for which you wish to apply the background, then click the pass icon to open the Pass property editor.



3. Select the Environment Shaders tab and click **Add** to apply an environment shader to the selected pass's shader stack.
4. From the browser, select an environment shader from the shader library. You can choose from:
   - **Cubic 1**—Maps a single image or texture in a cubic shape around the scene. Less accurate than spherical but faster to render.
   - **Cubic 6**—Maps up to six images in a cubic shape around the scene.
   - **Spherical**—Maps a single image in a spherical shape around the scene. This is more realistic than cubic, but it takes longer to render.
5. Click OK when you have chosen a shader. It appears in the shader-stack list. To give the shader a texture to use as an environment, select the shader from the stack window and click **Inspect**. This opens the shader's property editor.

Creating an Environment on an Object

Rather than create an environment that encompasses an entire scene, you can work on a smaller scale and create a specific environment for a single object. The object is usually reflective and is given its own environment map to speed up rendering.

To give an object an environment map

1. Select the object (or objects) to which you wish to apply an environment shader.
2. From the Render toolbar, select **Get** > **Shader** > **Environment** to open a list of environment shaders you can use. Select one of the following:

Shader	Function
Spherical	Maps a single image in a spherical shape around the scene. More realistic than cubic but takes longer to render.
Cubic	Maps a single image or texture in a cubic shape around the scene. Less accurate than spherical but faster to render.
Cubic 6	Maps up to six images in a cubic shape around the scene.
Atmosphere	Allows you to map realistic planet atmospheres.
Day	Maps a day-like horizon environment, with a sun, clouds, and the ground.
Night	Maps a night-like horizon environment, with stars, a moon, and the ground.

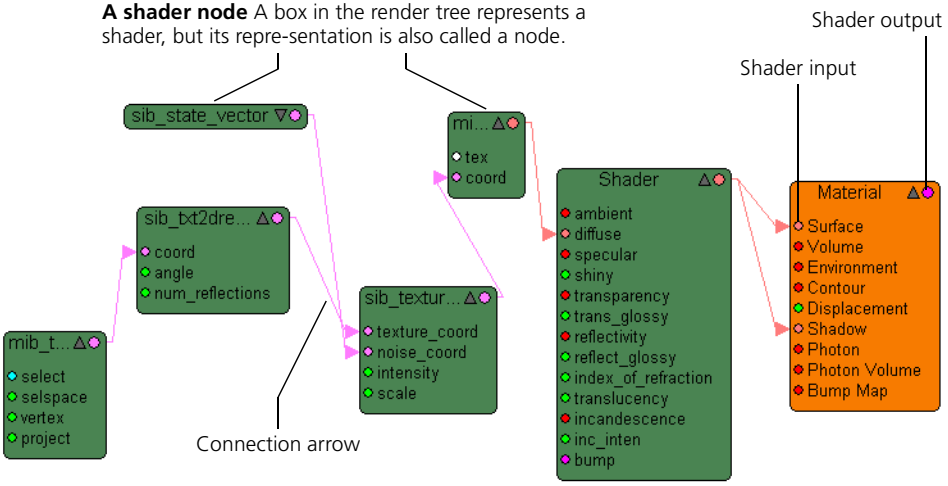
You can quickly edit an environment map's image by right-clicking the thumbnail icon in the shader's property page and selecting **Clip Properties** to open the Clip property editor. Select the FX tab to change the image's color, brightness, or to apply a blur.

For more information on a shader's parameters, refer to its online help by clicking the question mark icon on its property editor.

Chapter 9 The Render Tree

The render tree lets you connect shaders—or nodes—together to build a visual effect. Each node exposes a set of properties that can be dynamically linked by connecting the output of one property to the input of another. This is called *dynamic property assignment* and is as simple as dragging a connection line (arrow) between the input and output attachment points of the shader.

This chapter contains nine examples that provide you with a basic understanding of how to create effects in the render tree. The examples begin on page 220.



In the render tree, a shader is presented as a container; that is, you cannot change its internal content. But you can extend it by connecting nodes to augment the effect.

The render tree does not let you change the internal structure of a shader or build your own from scratch. To do so, you must use a simple text editor.

Besides shaders, render tree nodes can also be textures or functions known as *tool nodes*. Each type of node performs specific operations. Tool nodes are used as pattern generators (to create noise, for example), mathematical functions (such as multiplication), converters (to convert a scalar to a color RGB value), mixers, blends, and so on.

A shader provides a number of properties that you can control. You can set constant values for these properties or you can attach tool nodes or other shaders to these properties to procedurally control their values.

A render tree makes it possible to use a texture or a texture shader to control a component of the shading calculation. For example, you can use a 2D or 3D texture as the color, specularity, or reflectivity of a material. This means that the inputs to the shading model can be manipulated procedurally. Color and transparency textures, reflection mapping, bump mapping, displacement mapping, and solid texturing can all be implemented using the render tree.

Opening the Render Tree

There are two ways of opening the render tree:

- Click the arrow or the name of the current view displayed on any viewport title bar to open the **View** menu and choose **Render Tree**.

  or

- Choose **View** > **Views** > **Render Tree** from the main-menu bar. This displays the render tree in a floating window.

Working with Render Tree Nodes

A node is the generic term for a shader's "box" representation in the render tree.

Double-clicking on a node in the render tree automatically opens its property page in a floating window. From the property editor, you can manipulate the node's various parameters as well as select help to get a contextual help topic for the shader.

You can also multi-select several nodes in the render tree and press Enter to open a single property page for all of them. In this case, each node is represented by a tab at the top of the property editor.

Selecting Nodes

The selection mode in the render tree is mapped to the mouse buttons. The left, middle, and right mouse buttons correspond to node-, branch-, and tree-level sections, respectively. For example, if you select a node that is part of a hierarchy, left-clicking only selects that node, middle-clicking selects all the nodes in that branch, and right-clicking selects the entire tree structure.

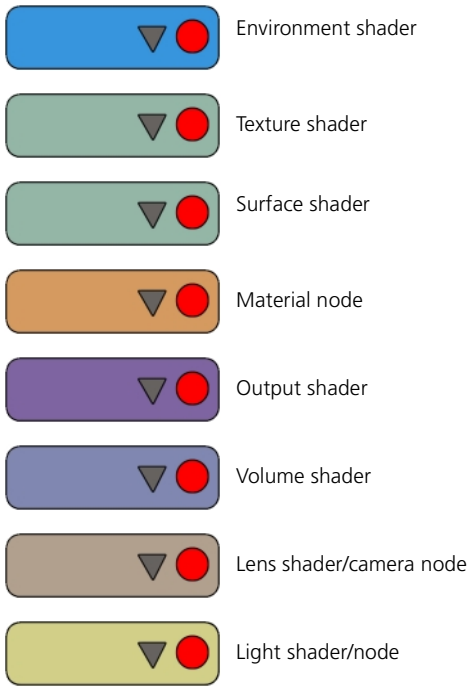
Understanding the Color Codes

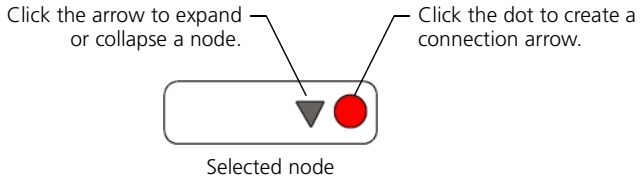
Every node that appears in the render tree is color coded, as are each of its parameters. This coding system helps you visualize which shaders are doing what within their respective render tree structures.

The coding also speeds up the construction of a tree by clearly displaying how and where parameters and nodes can be connected.

Node Colors

As a render tree structure grows in size and complexity, color coding can be extremely useful to locate a specific node at a glance. Here is the node color color code:





Navigating in the Render Tree

In order to navigate quickly and efficiently through the render tree workspace, you should know a few basics. Along the top of the render tree workspace is a command bar with the Tools and Show drop-down menus. You can select a specific option in each of these menus to customize a navigation method. These options can be changed at any time, depending on your needs and the size of your render tree.

Panning and Zooming in the Render Tree

Moving about the render tree work area is as simple as navigating through any viewport using the Pan and Zoom tool (**z** key).

There are two ways to pan and zoom:

You can also use the arrow keys to scroll a render tree in any direction.

- Select the **Pan and Zoom** option from the Tools menu. You can now pan in any direction, zoom in or zoom out using the left-, middle-, and right-mouse buttons, respectively.

  or

- Select the **Rectangular** Zoom option from the Tools menu, then use the cursor to click and drag a rectangle in the render tree work area. When you release the mouse button, the contents of the rectangle are sized to fit the viewport.

You can access all of the view, pan, zoom, and arrange commands by right-clicking on the render tree work area. When the contextual menu appears, click the desired task.

Rearranging and Grid Snapping

When working with a large and complex render tree, it may not take too long for nodes to start overlapping and cluttering the workspace. You can rearrange the nodes by having them snap to an invisible grid that covers the entire render tree work area. From the Tools menu, you can select the following:

- **Rearrange**—places every node in an orderly and logical manner without any overlapping.
- **Snap to Grid**—causes the nodes to align themselves with an invisible grid that covers the whole render tree workspace.
- **Enable Grid**—creates an invisible grid in the render tree workspace. Any future rearranging or moving of the nodes will be along this grid. When this grid is off, you can move the nodes however and wherever you wish.

If you have edited or moved nodes in the render tree, their positions may not necessarily be the same when re-opening the render tree view.

Clearing the Render Tree Workspace

You can quickly clear the render tree work area by clicking the Clear button from the render tree command bar. The Clear command clears only the workspace but does not remove or undo any of the connections that were made in the render tree. Unconnected nodes are deleted.

Updating the Render Tree Workspace

When you click **Update** in the render tree command bar, any unconnected node is removed from the workspace. This applies to any node not connected to the selected element (geometric object, camera or light). For example, if you have created a series of interconnected shaders that are not attached to your object, they are lost if you click **Update**.

Framing the Render Tree

In order to see the entire render tree structure, select **Tools** > **Frame All** from the render tree command bar. This command causes the work area to zoom out until every node is visible in the viewport.

To frame part of a render tree

Instead of framing an entire render tree, you can frame a selected node or branch you are working on.

1. Select a single node or branch by using the left or middle mouse button, respectively.
2. Chose **Tools** > **Frame Selection** from the render tree command bar.

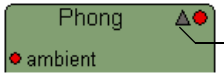
As in any view, you can use the **f** key to Frame Selection and **a** to Frame All.

Panning in the Render Tree

Depending on the size of the render tree viewport and the complexity of your render tree structure, it may become necessary to navigate from one end of your tree to another. Select **Show** > **Pan View** to create a small window in the upper left of the render tree work area. The render tree work area is represented by a gray square inside the window. As you move the square with your cursor, your structure's nodes move accordingly.

Collapsing and Expanding Nodes

As your render tree grows, it may become necessary to collapse some nodes to gain some real estate in your work area. This can be done by selecting the **Collapse All** option from the Show menu. Collapsing a node reduces its "box" to the bare minimum and will only display its name and output. You can also click the triangle icon on a node to do this.



Clicking the triangle icon collapses and expands a node.

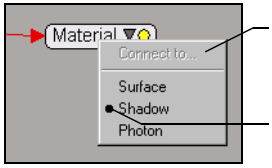
To "open" the node again, select **Expand All** from the Show menu. The full node is displayed with all of its parameters.

Select the **Show Connections Only** option to display the inputs that are connected to another node.

Connecting and Disconnecting Collapsed Nodes

Although a node's parameters are hidden in the collapsed mode (or only partially hidden in the Show Connected Only mode), you can still connect and disconnect inputs to any parameter.

- Pull a connection arrow onto a collapsed node and release it to open a small window next to the node. The window lists which parameters accept a connection from the arrow. Click on one of the parameters to connect it to the output of the node the arrow was "pulled" from.



When connecting a node to a collapsed node, a drop-down menu appears, prompting you to select a parameter to connect to.

A dot beside a parameter means that it is already connected to a shader.

Disconnecting nodes from the collapsed mode is just as easy.

1. Click on a collapsed node from which you want to remove an input. A small window appears listing each node that is connected.
2. Click on the parameter you want to remove an input from and the connection is cut.

Copying and Pasting Nodes

You can also copy and paste nodes within the same render tree or to another view with the following commands:

Command	Result
Ctrl+x	Cuts a selected node, removing it from the tree and keeping it in memory.
Ctrl+c	Copies a selected node, leaving it in the render tree work area but keeping it in memory
Ctrl+v	Pastes a copied or cut node and places it in the render tree work area.

Accessing Shaders

The **Nodes** button on the render tree command bar gives you quick access to almost all the shaders. Once you click the button, you will notice that shaders are categorized by function and type. For example, the texturing shaders are under the Texture heading, convertors are under Conversion, and so on.

For more information on the shader categories and the shaders themselves, see the *Appendix: Shader Descriptions* on page 233 and *Frequently Used Shaders* on page 217.

Getting Image Clips

Often when working in the render tree, you'll want a quick way to load image clips to use as textures, backgrounds, or bump maps.

1. Click the **Clips** button from the render tree command bar to display a list of all your project's image clips.
2. Select a clip to make it appear in the render tree work area.

You can quickly load clips by dragging and dropping image files from any type of browser into the render tree work area.

Connecting Nodes

Dynamic property assignment is a feature of the render tree that lets you extend shaders by using the output of one shader as the input for one or several shader parameters. You can also use the output of one shader as the input of another, which, in turn, provides the input for one or several shader parameters. In fact, there is no end to the number of nodes that can be connected to each other. For example, you can use a 2D or 3D texture to control the reflectivity of a material shader and another texture to control the specularity.

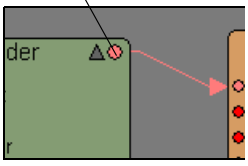
What to Connect and Where

Some render tree connections are very basic. For example, when you connect a color output into the surface input of an object, you have a pretty good idea of what your result is going to be because you're only changing the object's color. But, what if you connect a fractal shader to a shader's Reflection input, or a cloud texture to a material's Displacement input? The combinations are endless! For a full description of how inputs and outputs react with each other, see *Shader Inputs and Outputs* on page 53 for more information.

To connect nodes

1. In a viewport, get a geometric object.
2. Draw a render region (press **q**) so that you can interactively preview the effect you are building in the render tree.
3. Open the render tree in another viewport (select it from the Views menu); it automatically opens horizontally across two viewports so you have a larger working area. You should see your selected element with any connected nodes displayed as a render tree.
4. From the Nodes drop-down menu, select a shader you want to connect as a node. Once selected, it is immediately placed in the work area.
5. Click and hold the colored button (red, magenta, blue, etc.) in the upper right of the new node to display a connection arrow. Drag the arrow and connect it to a compatible attachment point—Surface, Volume, Environment, Contour, Displacement, Shadow, or Photon—of the material node.

Click the Output button to create a connection arrow.



If the output color does not correspond to the input color, a connection won't be possible. Once a connection arrow is pulled from a node, the parameter name becomes colored according to its accepted input as the pointer passes over a parameter inside a node. For example, if a parameter attachment point turns light blue when the pointer passes over it, this input connection only accepts a texture connection.

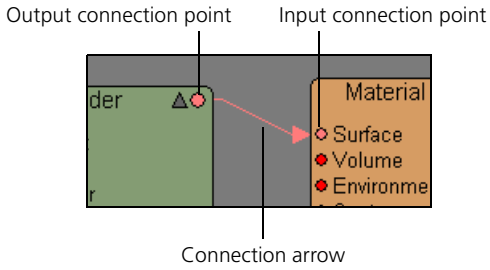
Disconnecting Nodes

To disconnect a connected node, simply click on the tip of the light-red connection arrow. As soon as the arrow turns red, you can release it, thereby severing the connection.

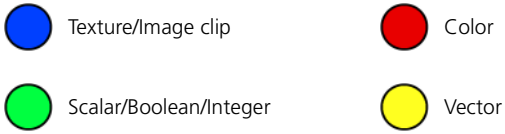
Input and Output Colors

Every parameter of every shader is color coded. A node's output is indicated with a connection point (colored dot) in the top right of the node, and a parameter's input is indicated with a connection point (small dot) to the left of the parameter's name. This color identifies what type of input the parameter will accept.

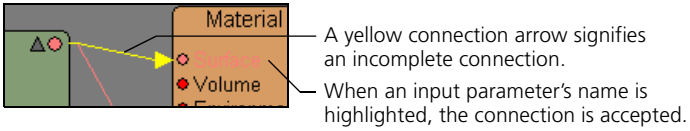
Only inputs and outputs of the same color can be connected; that is, scalar to scalar, color to color, and so on.



Here is the node output and input color code:



Also, once a connection arrow has been created from a node's output, you can drag the cursor over a parameter inside a node to see if a parameter will accept the connection. If the parameter name changes color, it will accept the connection arrow. For example, if a parameter attachment point turns red when the cursor passes over it, this input connection will only accept a color connection.



Replacing a Shader

To replace a shader with another, simply drag and drop a shader directly on top of the existing shader in a render tree structure. The old shader is deleted and replaced with the new shader. If possible, the new shader will keep the same connections (both input and input) of the previous shader.

Editing Nodes

You can move nodes within the render tree workspace and copy, paste, edit, and connect them to each other. The render tree is highly interactive: you just point and click to select nodes and connect related nodes by drawing lines between them as described in *Connecting Nodes* on page 214.

By double-clicking on a shader node in the render tree, you can open its property editor to edit that shader's properties as well as find a contextual help topic by clicking the Help icon (?).

Previewing a Node

Use the **p** key to directly preview a single node's result. This function attaches the selected node directly into the Surface input of the Material node so you can get a quick preview of its values in the render region.

Don't forget to press **p** again to restore the previous render tree connection and switch off the preview mode.

Grouping Materials

You can group several materials to make it easier to work with multiple objects. Although you can only work on one render tree per render tree view, a multi-selection's render tree can be displayed if all the objects share the same material.

Frequently Used Shaders

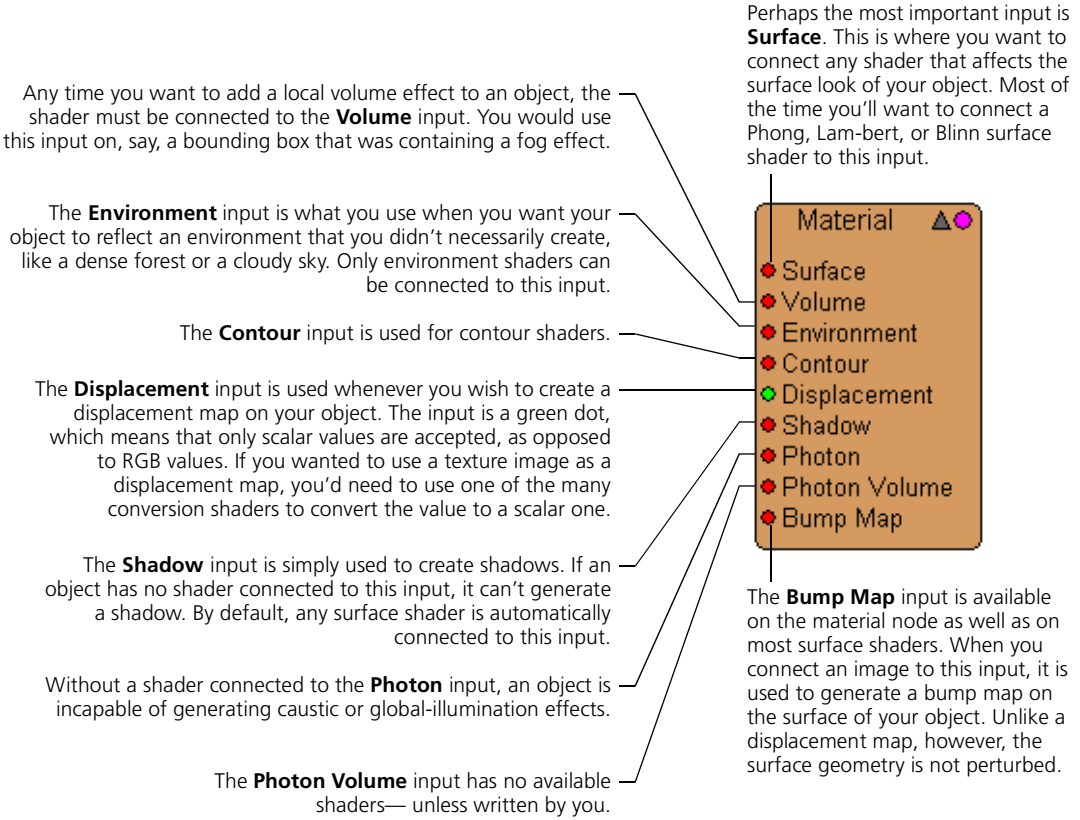
As you begin creating your own render trees in the render tree, you will discover some invaluable tool shaders. The following is a quick description of some very useful tools that you may not find immediately (most of these shaders can be found in the Texture folder of the shader library).

Shader	Function
Image	Applies and deforms an image over a surface in a selected projection.
Phong	Applies a Phong surface material to an object.
Front_back	Defines a color for each side of a surface.
Color_share	Shares a color among several nodes.
Mix_2colors	Mixes one color with another color.
Incidence	Controls the density of a transparent object's edges, depending on angle of the camera.
RGB Keyer	Lets you key a specific RGB value from a color, shader, or texture.
Gradient	Accepts up to eight colors or textures to create a gradient.
Invert	Inverts a color.
Color_correction	Controls a color's contrast/brightness/hue/luminance/ saturation/gamma.
Scalar2color	Converts a scalar (numeric) value to a color value (RGB.)
Color2scalar	Converts a color (RGB) value to a scalar (numeric) value.

Where to Start?

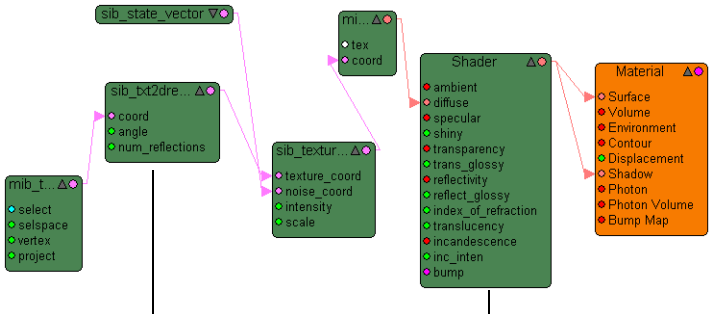
The first time you open the render tree, it may be a little daunting. There are a lot of nodes and thousands, if not millions, of possible connections and more shaders than you know what to do with!

The trick to keeping focused is knowing what shader goes where and what it does. Of course, there are no hard and fast rules, but knowing what the Material node will accept is a great start. Use the following illustration to help you understand.



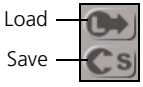
Saving and Copying Render Trees

It is possible to save an object's render tree and apply it as a preset to another object or tree altogether. The most important thing to consider when saving a render tree preset is which node to save. Trees are saved from right to left, following the nodes' hierarchy. For example, a node on the right side of the screen will save every node under it to the left.



To save a render tree preset

1. Double-click the node you wish to save to open its property editor.



2. Click the Save icon (left) on the top right of the page.
3. Navigate to where you wish to save the render tree preset and name it. Click OK.

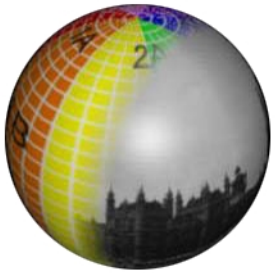
To load a render tree preset

1. Double-click the node to which you wish to load a preset.
2. From its property page, click the Load icon (left).
3. Use the browser to navigate to the location you saved your presets, select one, and click OK.

You can achieve more accurate results if you load a material node preset on a material node preset, or a surface shader on a surface shader, and so on.

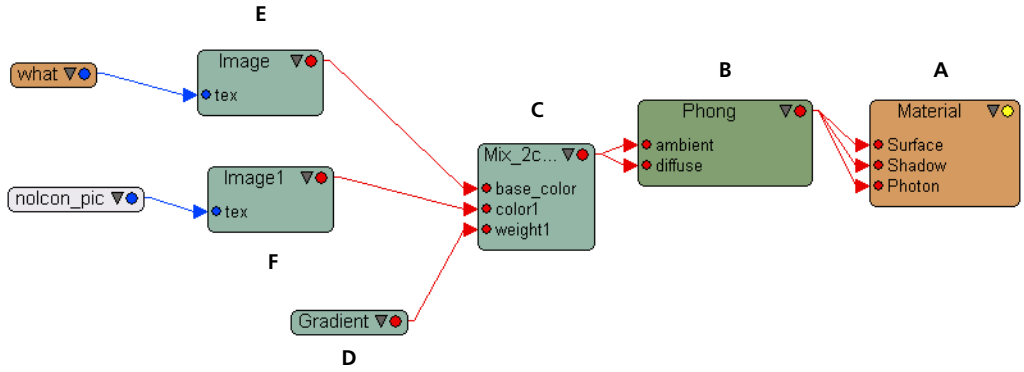
Example 1: Mixing with a Gradient

This example illustrates how two textures or images can be gradually mixed. But rather than connect both images into a mixer node and adjusting from there, you can use a Gradient shader (or any other shader) to drive the mix parameter. The result is two images that merge into one another and whose gradient can be precisely controlled.



This sphere is textured with two images that are blended with a Gradient shader.

Node	Function
A	Material acts like a placeholder for any shader that can affect an object's look.
B	The Phong surface shader defines the surface properties of the object.
C	Mix 2 Colors blends two color inputs: in this case, two images. The actual mixing parameter is driven by a gradient texture shader (node D).
D	Gradient Texture drives the mixing of the mixer node (C), which causes the mixing of the two images to be a gradient instead of a hard-edge mix.
E	This image shader references an image of the London cityscape
F	This image shader references an image of a standard color bar with numbers.


E
what
Image
tex
B
A
C
Phong
ambient
diffuse
Material
Surface
Shadow
Photon
Mix_2c...
base_color
color1
weight1
nolcon_pic
Image1
tex
F
Gradient
D

Example 2: Blending Images with a Mixer

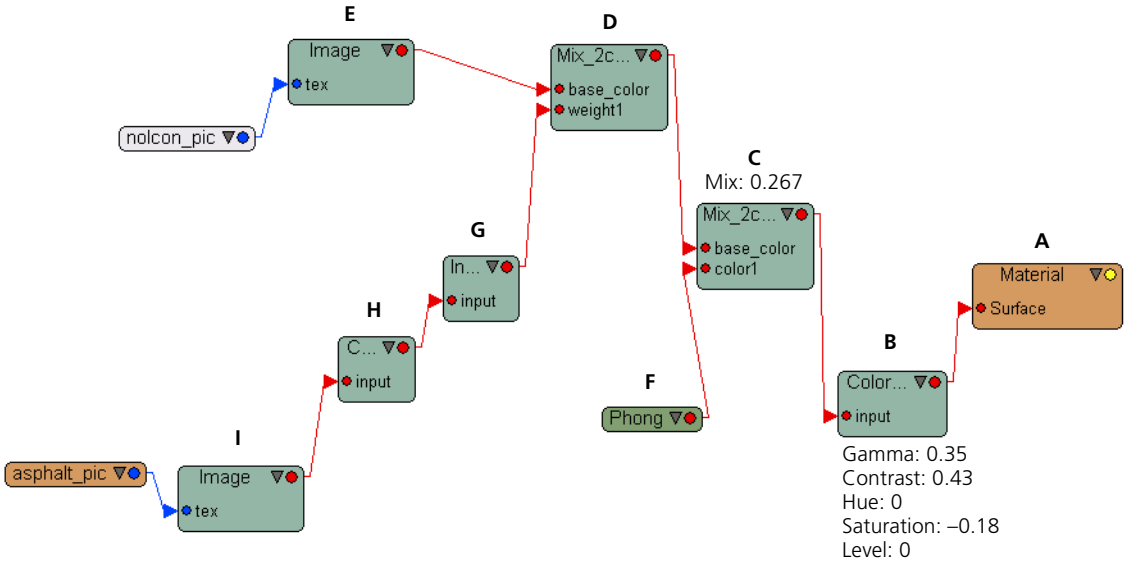
This example blends two images on a single object. Both images are connected to a mixing node and animated so it appears as though the object is gradually changing from one image to another.

The example is an exposed sheet of photographic paper slowly developing in a photo tray. The white sheet begins to darken with the most heavily exposed (dark) areas first and the highlights last.

The photo paper in the tray slowly develops from one image to another with the darkest colors appearing first.

The two images used are the final developed image and an inverted alpha of the final image. Both images are connected to a mixer and animated so it appears as though the image is developing. The nodes are used as follows:

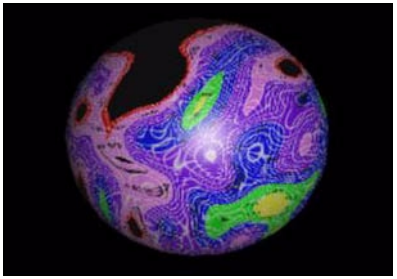
Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look.
B	Color Correction node: Used to correct slight variations in color and tweak the final image's look.
C	Mix 2 Colors: Basic node used to mix the two images. In this example, it is mixing the emerging image with the blank sheet of photographic paper. This tool is animated over the initial frames of the scene, say, 1–10 out of 100 frames.
D	Mix 2 Colors: Blends the final image with the dark areas of the image (which are also animated), so it appears as though an image is developing. This mixer is animated from, say, frames 11–99.
E	The Image shader that references the image of the fully developed photo.
F	Phong surface shader: The surface shader of the white sheet. It is also mixed with the emerging image.
G	The resulting animation from Node H is inverted before it is mixed with the final image.
H	Color Math Exponent tool shader: Also animated, it drives the alpha from black to white.
I	This Image shader references the developing image. It is an alpha of the final image.


E
Image
tex
nolcon_pic
D
Mix_2c...
base_color
weight1
C
Mix: 0.267
Mix_2c...
base_color
color1
A
Material
Surface
G
In...
input
H
C...
input
B
Color...
input
F
Phong
I
asphalt_pic
Image
tex
Gamma: 0.35
Contrast: 0.43
Hue: 0
Saturation: –0.18
Level: 0

Example 3: Warping and Deforming a Texture Space

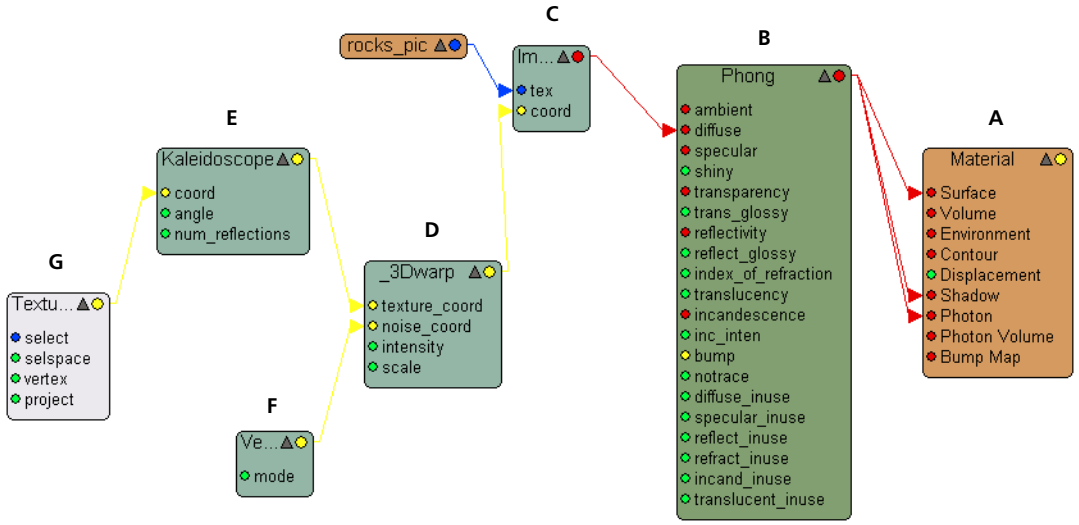
This example illustrates how tool shaders can deform and warp a texture space.

The following render tree is applied to an object that has had a UV texture projection assigned to it.



You can create thousands of different texture patterns with the render tree. Using vector information to drive colors and/or textures can create some pretty interesting looks.

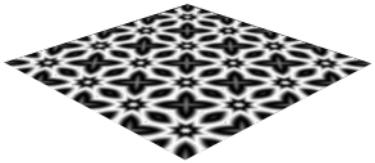
Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look.
B	The Phong surface shader defines the surface of the object.
C	Texture Lookup: This texture-generator tool shader defines the texture applied to the diffuse parameter of the material shader.
D	3D Warp: This texture space controller shader "warps" the texture space. Instead of explicitly defining the parameters within, they are defined by another two shaders (E and F).
E	Kaleidescope: This texture space controller is used to control the texture coordinates of the 3D Warp. Adjusting the Number of Reflections parameter can create a wide range of patterns.
F	Vector State: This tool shader is used to drive the noise parameter of the 3D warp node.
G	Texture Space Generator: This tool shader is defined on the UVNurbs texture space the object has had assigned to it.


C
rocks_pic
Im...
tex
coord
B
Phong
ambient
diffuse
specular
shiny
transparency
trans_glossy
reflectivity
reflect_glossy
index_of_refraction
translucency
incandescence
inc_inten
bump
notrace
diffuse_inuse
specular_inuse
reflect_inuse
refract_inuse
incand_inuse
translucent_inuse
A
Material
Surface
Volume
Environment
Contour
Displacement
Shadow
Photon
Photon Volume
Bump Map
E
Kaleidoscope
coord
angle
num_reflections
D
_3Dwarp
texture_coord
noise_coord
intensity
scale
G
Textu...
select
selspace
vertex
project
F
Ve...
mode

Example 4: Creating Textures with the Render Tree

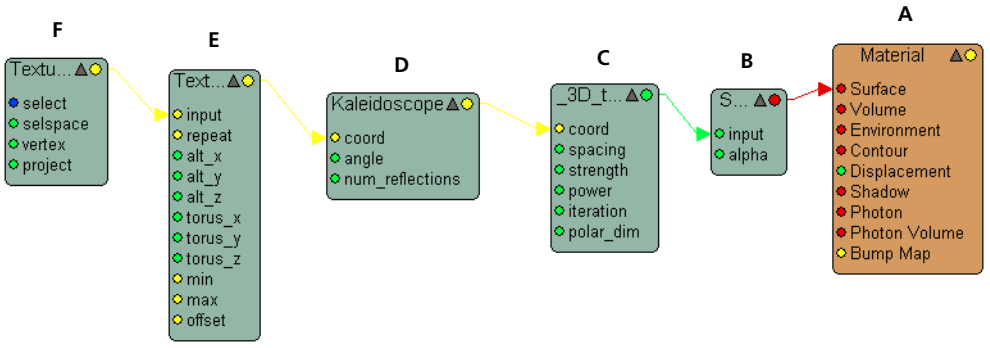
This example illustrates how a few shaders can manipulate simple vector coordinates to create complex texture patterns on a given object.

The following render tree is applied to an object that has had UV texture coordinates assigned to it.



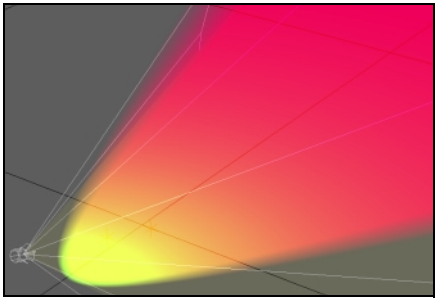
Using texture-space controller shaders, like the Kaleidoscope shader, are a simple way to repeat a texture in creative but editable texture patterns.

Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look.
B	The Scalar2color tool shader converts the shader's output from a scalar to a color value.
C	3D_turbulence: This texture shader creates a turbulence to the vector input it is receiving from node D.
D	Kaleidescope: This texture space controller tool shader rearranges vector coordinates into a kaleidescope pattern. Adjusting the number_reflections parameter creates a wide range of patterns.
E	Texture_edit: This tool shader remaps vector coordinates as defined by the parameters in the shader's property editor.
F	Texture Space Generator: This tool shader redefines the UVNurbs texture space the object has had assigned to it.



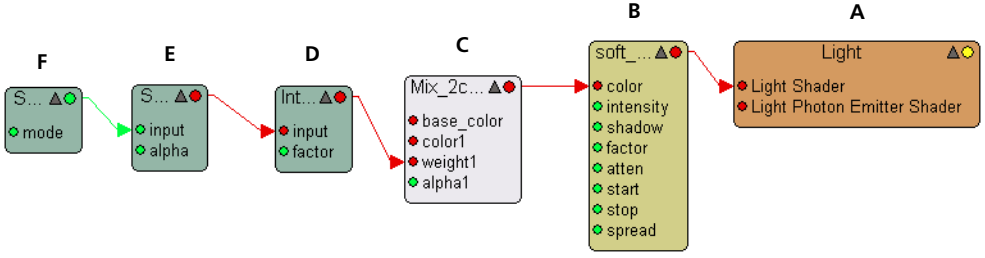
Example 5: Defining Light Properties

Although one tends to think of geometric objects and textures when working with the render tree, you can also create custom effects with lights and cameras. The following example simply illustrates how you can control a light's color based on its distance from the light source.



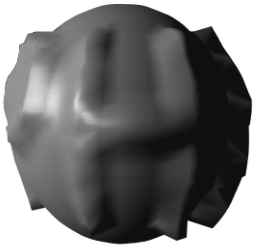
This simple example using a light is just the tip of the iceberg. From the Soft Light shader, you can use textures and vector information to control nearly every aspect of a light.

Node	Function
A	Light Node: Shaders connected to this node define the material, surface, or projection of the light.
B	Soft Light: The basic light shader used to control its properties. All lights created have this shader applied by default.
C	Mix 2 Colors: The two colors the light returns are defined with this tool shader. In this example, red and blue are used. Alternately, you could drive the base_color and color 1 inputs with any shader that outputs a color value.
D	Intensity: This tool shader adjusts the intensity of the Scalar2color node's output, which then drives the color mixing node.
E	Scalar2color: Converts the scalar input into a color output.
F	Scalar_state: This node modulates the color with the distance. It returns the distance to node E.



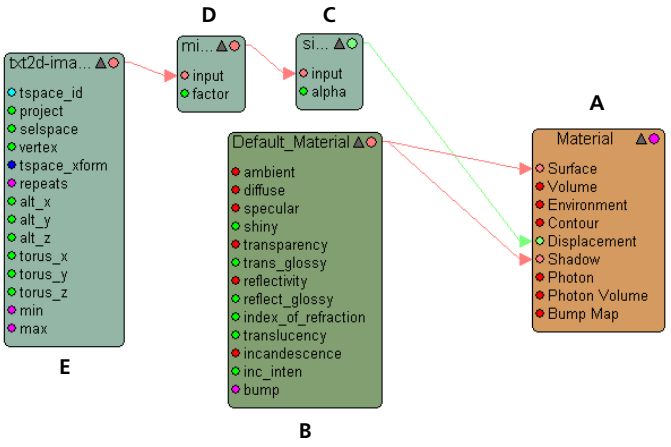
Example 6: Creating a Displacement Map with an Alpha Channel

This example lets you use an image's alpha channel to create a displacement map on an object. You don't have to use the same texture to drive the diffuse or specular values of the object. In addition, the Alpha can be independently controlled with the Factor slider.



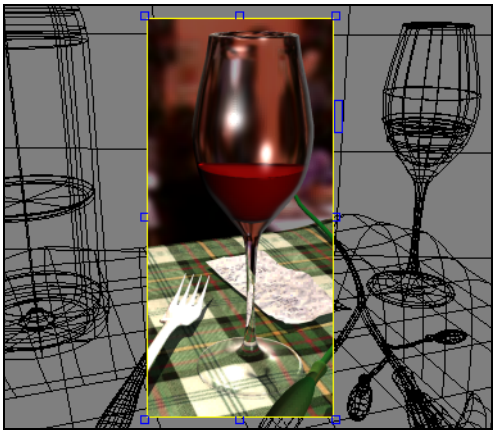
Although this example creates displacement, it is just as easy to create a bump map or use the alpha to define a texture.

Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look.
B	Phong: The Phong surface shader, which defines the surface of the object.
C	Color2scalar: This tool shader converts the color input into a scalar output (the only type of input the Displacement parameter accepts).
D	Color2Alpha: This color channel tool shader extracts the Alpha channel from an image. It can be independently controlled using the Factor slider in its property editor.
E	The Image shader defines which image you wish to use as a texture and how to project it. In this example, the word "Alpha" was applied to a texture but is only visible in the Alpha channel.



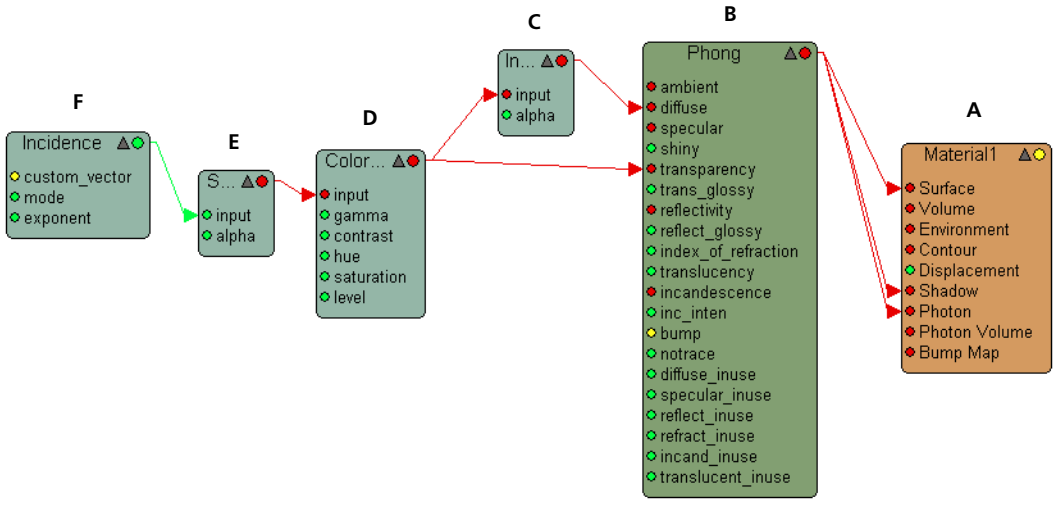
Example 7: Creating Realistic Glass

Creating a realistic glass texture is surprisingly easy when using the render tree. The key to the effect is using the Incidence shader, which lets you control the shading on an object's edges, depending on the camera's viewing angle.



Glass is a little more complex to create than slapping on a Phong surface shader and making it transparent. Notice how this wine glass's edges and stem are darker and less transparent than its center. This effect changes depending on your angle of view.

Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look.
B	Phong: The surface shader node, which defines the surface of the object including transparency, reflection and refraction.
C	Invert: This image-processing shader is simply inverting the output of the color correction node so the white areas defined by the Incidence shader are now opaque.
D	Color Correction: This image processing shader is used to tweak the results of the Incidence shader. Since the Incidence shader outputs a black and white image, this shader controls the contrast and intensity of each color.
E	Scalar2Color: Converts the scalar (Incidence shader) input into a color output so it can be manipulated by the Color Correction shader.
F	Incidence: This illuminaiton shader controls an object's shading in relation to the camera angle or any other axis.


F
Incidence
custom_vector
mode
exponent
E
S...
input
alpha
D
Color...
input
gamma
contrast
hue
saturation
level
C
In...
input
alpha
B
Phong
ambient
diffuse
specular
shiny
transparency
trans_glossy
reflectivity
reflect_glossy
index_of_refraction
translucency
incandescence
inc_inten
bump
notrace
diffuse_inuse
specular_inuse
reflect_inuse
refract_inuse
incand_inuse
translucent_inuse
A
Material1
Surface
Volume
Environment
Contour
Displacement
Shadow
Photon
Photon Volume
Bump Map

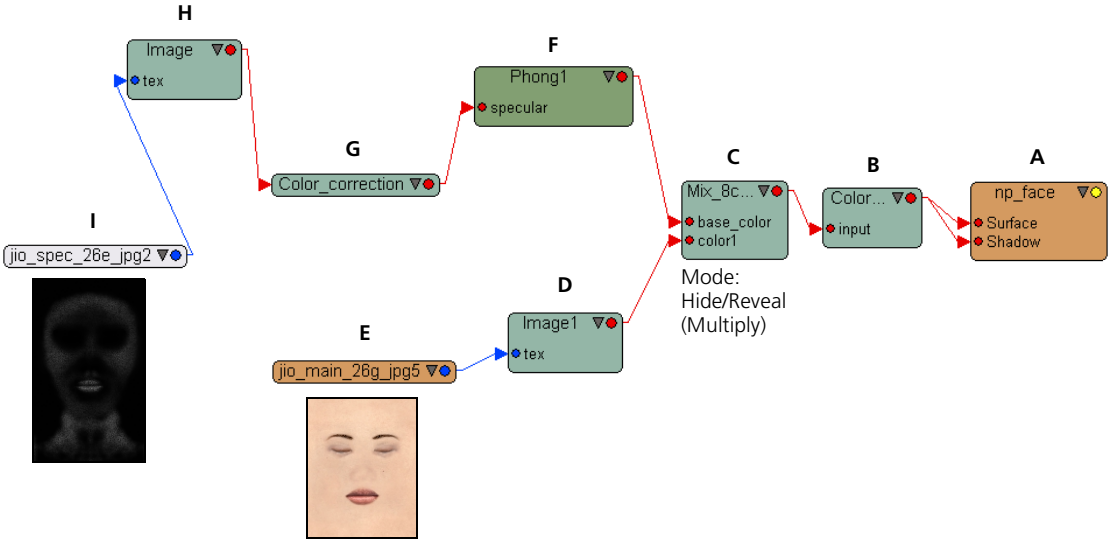
Example 8: Creating Realistic Skin

Perhaps one of the more complex tasks in photorealistic rendering is creating a skin texture. Using the render tree's mixing capabilities, this example shows you how to use a minimal amount of shaders and textures to create incredibly realistic skin on a NURBS patch.

To create realistic skin textures, it helps to have a high-resolution texture to apply. However, a nice texture alone won't be enough to mimic real skin when it comes to shadows and highlights. This example uses only two image clips, one mixer, and a few tool shaders to create life-like skin.

Node	Function
A	Material node: Acts like a placeholder for any shader that can affect an object's look. By default, any shader that affects the object's surface is connected to the Surface and Shadow input; otherwise, none of these properties are visible.
B	Color Correction: This image-processing shader lets you make some final corrections to the overall effect, such as gamma, hue, or contrast correction.
C	Mix 8 Colors: This mixer shader uses one image (H) as a base color and multiplies another image (D) with it. The mixing Mode is set to Hide/Reveal (Multiply) with a Weight set to 1 (white). Playing with the Weight parameter determines how much of each texture is shown or hidden by the other.
D	Image: This texture shader processes the Image Clip E that is blended with another texture in shader C.
E	Image Clip: This is an image clip of the high-resolution texture.
F	Phong: This surface shader is used to give the specular texture a skin-like color and provide a highly editable surface shader to use as a base color when mixing with the skin texture.

Node	Function
G	Color Correction: This image-processing shader is used to increase the gamma of the specular map texture.
H	Image: This texture shader processes the Image Clip I that is blended with another image clip (E).
I	Image Clip: This is the image clip of the specular map. The subtle white areas on this image define the highlights (lips, forehead, etc.) of the final image once blended with the skin texture.



Appendix **Shader Descriptions**

SOFTIMAGE|XSI Shaders

Shaders and tools are essential in creating, sculpting, or just tweaking a scene or image. The descriptions are listed in this appendix in order to help you understand which shader does what and where it should go. Although this may be obvious with some shaders, others, such as the mathematical-function tools, may be a little more obscure. This is why the shaders and tools have been categorized by directory as they are found in SOFTIMAGE|XSI. The following table lists the preset name (as it appears in the browser), the shader's output (scalar, color, etc.), in what situation the shader can be used (environment, volume, lens, etc.), and a brief description.

Scenes are imported with their assigned shaders. Once in, you can detach existing shaders and attach other ones. You can choose from the available shaders in the shader library.

The shaders are described as they are found in the browser:

Environment

Basic environment mapping tools for defining how a texture is mapped as an environment around a scene. Environment shaders are applied to a pass rather than a geometric object.

Preset name	Output	Description
Cubic_mapping_1	Color	Maps a single texture on all of a cube's faces.
Cubic_mapping_6	Color	Maps a different texture on each face of a cube (up to six textures).
Spherical_mapping	Color	Maps a texture using a sphere around the object.

Lens

Applied to a lens to create specialized camera effects.

Preset name	Output	Description
Bump_lens	Color	Applies a bump map on a lens to distort an image.
Camera_projection	Color	Maps an image as though it were projected from a camera.
Cartoon	Color	Creates cartoon-like edges and paint colors.
Depth_of_field	Color	Creates and simulates a camera's depth-of-field lens parameters.
Fast_lens_effects	Color	Basic lens shader used to create quick effects such as flares, stars, etc.
Fisheye	Color	Creates a fisheye lens effect.
Flare	Color	Creates a highly editable lens flares.
Foreground_mask	Color	Creates a foreground (alpha) mask.

Preset name	Output	Description
Panoramic	Color	Renders a cylindrical (surround) view of a scene. Often used to create surround video.
Self_antialiasing	Color	Allows antialiasing without using the background. Often used to create game sprites.
Starburst_flare	Color	Create a flare with starburst specular highlights.
Stereoscopic	Color	Creates a stereoscopic image.
True_lens_emulator	Color	Simulates several popular camera lenses and their attributes.

Light

Light-based shaders used to create a type of light or manipulate an existing one.

Preset name	Output	Description
Soft_light	Color	Creates a basic light with adjustable parameters. This shader is attached to all newly created lights by default.
Fast_light_effects	Color	Creates a basic light with a wide range of adjustable effects.
Slide_projector	Color	Projects a defined .pic file from a spotlight.

Surface (Material)

Base surfaces applied to an object prior to adding a texture shader.

Preset name	Output	Description
Anisotropic	Color	Creates a Ward-type surface illumination with two adjustable speculars.
Blinn	Color	Applies a Blinn surface to an object.
Cooktorrance	Color	Applies a Cook-Torrance surface to an object.
Flat_light	Color	Applies a flat-light surface to an object (viewing angle not considered).
Lambert	Color	Applies a Lambert surface to an object.
Phong	Color	Applies a Phong surface to an object.
Shadow	Color	Extracts a shadow projected onto an object for pass purposes.
Strauss	Color	Applies a Strauss surface to an object.

Surface (Material) Illumination Shaders

This directory contains the basic tools used to create some of the more common shading algorithms. Besides the usual Blinn, Phong, and Lambert shadings, you'll find a few more specialized tools.

Preset name	Output	Description
Anisotropic_shading	Color	Creates a Ward surface illumination with two adjustable light directions.
Anisotropic_ shading_uv	Color	Creates a Ward surface illumination in the U and V directions.
Blinn_shading	Color	Creates a Blinn illumination texture on an object.
Cooktorrance_ shading	Color	Creates a Cook-Torrance surface illumination on an object.
Flat_light_shading	Color	Creates a moviescreen-like illumination texture on an object.
Lambert_shading	Color	Creates a Lambert illumination texture on an object.
Phong_shading	Multiple Colors	Creates a Phong illumination on an object.
Translucent_shading	Color	Applies a editable translucency onto an object.

Softimage Material Presets

The Materials folder includes all of the old color and surface presets that were included in SOFTIMAGE|3D. The folder is separated into primary and secondary colors as well as glass, metals, and other surfaces. All of the Softimage material presets are based on the Phong shading model.

SOFTIMAGE|3D Legacy Shaders

SOFTIMAGE|XSI provides support for all SOFTIMAGE|3D shaders. Supported here are a wide range of the most popular SOFTIMAGE|3D shaders.

The Legacy shaders cannot be used within the render tree. They are only used as generators and/or root nodes.

Preset name	Output	Description
Dusty	Color	Simulates the dirt and dust that accumulates on surfaces.
Easy_Atmosphere	Color	Allows you to quickly create planet atmospheres.
Env_atmosphere	Color	Simulates realistic planet atmospheres.

Preset name	Output	Description
Env_ball	Color	Reproduces environments by "photo-capturing" whole environments.
Env_cubic	Color	Creates a cubic environment. You assign six images to the up, down, right, left, front, and back of the cube.
Env_horizon_day	Color	Creates a day-like horizon environment, with a sun, clouds, and the ground.
Env_horizon_map	Color	Creates a day-like horizon environment, with clouds, and the ground.
Env_horizon_night	Color	Creates a night-like horizon environment, with stars, a moon, and the ground.
Env_sphere	Color	Takes a single image file and maps it to the environment using a spherical mapping.
Fire	Color	Simulates a surface-projection fire effect. Not volumic.
Peekaboo	Color	Creates surface transparency by using the light hitting it, allowing you to "see" into objects.
Toon_material	Color	These three toon shaders work together to create a cartoon-like effect in a scene.
Toon_soft	Color	
Toon_wire	Color	
Velvet	Color	Creates a satin or velvet-like effect on a surface.

Output

These shaders are applied to a final rendered image to create an additional effect, such as a glow or fur. These effects are rendered on the image, hence their 2D description.

Preset name	Output	Description
2d_auto_depth_of_field	Color	Automatically calculates and applies the depth of field of an image using a start and stop focus point.
2d_background_color	Color	Defines a background color for a scene based on the alpha channel.
2d_background_pic	Color	Allows a background picture (.pic file) to be composited based on an alpha.
2d_contour	Color	Creates lens-like diffraction and diffusion effects on an object's edges.
2d_depth_cue	Color	Creates a fast fog effect.
2d_depth_of_field	Color	Lets you apply a depth of field on an image.

Preset name	Output	Description
2d_fur	Color	Creates fur on a selected object.
2d_glow	Color	Creates a glow on the inside and outside of an object's edges.
2d_halo	Color	Creates a halo effect around an object.
2d_motion_blur	Color	Applies a post-process motion blur onto a rendered image.

Texture

These shaders create either a 2D or 3D texture on or through an object. They can also be used to define a bump map.

Preset name	Output	Description
Bumptree	Color	Lets you connect an image to a surface shader or the material node to use as a bump map.
Cell	Scalar	Creates a cell-like surface pattern on an object's surface.
Checkerboard	Color	Creates a checkerboard pattern with adjustable colors and formats.
Cloud	Color	Creates a cloud pattern texture on an object's surface.
Constant	Color	Creates a constant color or texture that is not affected by any illumination.
Constant_black	Color	Creates a constant black color that is not affected by any illumination.
Fabric	Color	Creates a fabric pattern on an object's surface.
Flagstone	Scalar	Creates a flagstone/stained-glass pattern on the surface of an object.
Fractal	Scalar	Creates a fractal texture on the surface of an object.
Gradient	Color	Creates a definable color gradient with an adjustable pattern.
Grid	Color	Creates a grid pattern on the surface of an object.
Image	Color	Applies and deforms an image over a surface in a selected projection.
Image_ bumpmap	Vector	Deforms an image over a surface to produce bump mapping.
Marble	Color	Creates a highly definable marble texture on an object's surface.
Ripple	Scalar	Creates a water-drop ripple on the surface of an object.

Preset name	Output	Description
Rock	Scalar	Creates a rock texture on an object's surface.
Snow	Color	Creates a snow texture on an object. Often used to simulate snow-covered objects.
Terrain	Color	Creates a highly definable terrain pattern on the surface of an object. Often used as a displacement map.
Vein	Scalar	Creates a vein texture on the surface of an object.
Wood	Color	Creates a highly definable wood texture on the surface of an object.
Z_depth	Color	Returns a pixel depth range based on a start/end value.

Texture Tool Shaders

These shaders are the basis for creating almost any effect. Although some can be used on their own, many of them must work in conjunction with another to achieve a highly manipulable effect. Each one has a specialized function. They are:

Color Channels

Let you manipulate the red, green, blue, and alpha components of a color.

Preset name	Output	Description
Color2alpha	Color	Converts a color to an RGBA grayscale; the result is copied to alpha.
HLSA_combine	Color	Grabs separate H, L, S, A components and places them back into a color or texture.
HSVA_combine	Color	Grabs separate H, S, V, A components and places them back into a color or texture.
Pick_channel	Color	Lets you isolate a specific color model and channel from a texture or shader.
Rgba_combine	Color	Grabs separate R,G,B,A components and places them back into a color or texture.

Conversion

Convert one value to another. Especially useful for changing a scalar-type node to a color one. Scalar, color, vector, Boolean and integer nodes can be converted to any other type using these tools.

Preset name	Output	Description
Color2scalar	Scalar	Converts a color (RGB) value to a scalar (numeric) value.
Color2vector	Vector	Converts a color (RGB) value to vector information.
Hsv2rgb	Color	Converts HSV color to RGB color.
Integer2scalar	Scalar	Converts an integer to a scalar value.
Rgb2hsv	Color	Converts RGB color to HSV color.
Scalar2color	Color	Converts a scalar (numeric) value to a color value (RGB).
Scalar2integer	Integer	Converts a scalar value to an integer.
Scalar2vector	Vector	Converts a scalar value to vector information.
Vector2color	Color	Converts vector information to an RGB value.
Vector2scalar	Scalar	Converts a vector value to a scalar one.
Vector_coordinate_convertor	Vector	Converts a vector parameter to or from different space coordinates (e.g.: camera to world).

Illumination

Let you to further define a texture's photon illumination. Used only when your scene has photon light defined (either caustics or global illumination).

Preset name	Output	Description
Photon_irridiance	Color	Returns photon illumination (caustic) at an intersection point.

Image Processing

For defining, manipulating, and tweaking color and scalar values.

Preset name	Output	Description
Color_correction	Color	Controls a color's constrast/brightness/hue/luminance/saturation/gamma.
Intensity	Color	Defines a color's intensity.
Invert	Color	Inverts a color.

Preset name	Output	Description
RGBA_keyer	Color	Lets you key a specific color from a shader or texture and obtain a matte that can be color corrected.
Scalar_invert	Scalar	Inverts a scalar value.

Math

Math functions such as interpolation, conversions, and attenuation.

Preset name	Output	Description
Average	Color	Converts an input color to an RGBA grayscale.
Boolean_math_logic	Boolean	Performs a basic mathematical calculation on two Boolean inputs.
Boolean_negate	Boolean	Simply switches off a Boolean input.
Change_range	Scalar	Remaps a value to a new range.
Color_math_Basic	Color	Performs a basic mathematical calculation on two color inputs.
Color_math_ Exponent	Color	Performs an exponential function using two color inputs.
Color_math_Logic	Color	Performs a logical argument using two input colors to output a single color.
Color_math_Unary	Color	Performs a basic operation on a single color value.
Exponent_falloff	Scalar	Attenuates a value exponentially.
Inrange	Color	Ensures that an input color, texture, or shader remains in range between two other RGB values.
Linear_falloff	Scalar	Attenuates a value in a linear manner.
Scalar_inrange	Scalar	Ensures that an input scalar value remains in range between two other scalar values.
Scalar_math_Basic	Scalar	Performs a basic mathematical calculation on two scalar inputs.
Scalar_math_ Exponent	Scalar	Performs an exponential function using two scalar inputs.
Scalar_math_Logic	Scalar	Performs a logical argument using two input scalar values to output a single scalar value.
Scalar_math_Unary	Scalar	Performs a basic operation on a single scalar value.

Preset name	Output	Description
Vector_math_scalar	Scalar	Permits scalar math operations with vectors.
Vector_math_vector	Vector	Permits vector math operations with vectors.

Mixers

Use one or several equations to mix a few or several colors (or a scalar value) into a single color output.

Preset name	Output	Description
Color_interpolate	Color	Performs an interpolation of a color map based on a scalar value.
Mix_2colors	Color	Mixes one color or texture with another color or texture.
Mix_8colors	Color	Blends, mixes, or adds up to eight color/image inputs to produce a single color output.

Raytracing

Basic tools for creating raytracing effects such as refraction, transparency, or reflection. They giveyou unprecedented access to deep raytracing value inside a node, and they allow you to “get your hands dirty” by tweaking with small yet complex values.

Preset name	Output	Description
Dielectric	Color	Returns the "reflection intensity" or "refraction intensity" of a surface.
Opaque	Color	Makes a transparent surface using the surface color (opacity).
Photon	Boolean	Allows the manipulation of photon lighting properties (specular, refraction, etc.).
Photon_rendertree	Color	Used in the render tree, it creates photon illumination on an object.
Reflection	Color	Defines a reflection on an object or shader.
Reflection_diffuse	Color	Applies a diffused reflection to an object or shader.
Refraction	Color	Applies a transparency or a refraction on an object or shader.
Refraction_diffuse	Color	Applies a diffused refraction to an object or shader.
Scalar_state	Scalar	Accesses low-level raytracing scalar values at pixel level.
Thickness	Scalar	Creates thickness at an intersection point.

Preset name	Output	Description
Transparency	Color	Makes a transparent surface by subtracting the surface color.
Vector_state	Vector	Accesses low-level raytracing vector values at pixel level.

Share

Coordinate the sharing of a single value among several others.

Preset name	Output	Description
Boolean_share	Boolean	Shares a Boolean among several nodes.
Color_share	Color	Shares a color among several nodes.
Integer_share	Integer	Shares an integer among several nodes.
Scalar_share	Scalar	Shares a scalar among several nodes.
Vector_share	Vector	Shares a vector among several nodes.

Surface

Let you create custom effects or fine-tune a surface.

Preset name	Output	Description
Bump_map	Vector	Selects, creates, and maps a bump map to a surface.
Bump_map_texture _orientation	Structure	Selects, creates, and maps a bump basis to a surface in a given projection.
Incidence	Scalar	Shades the edges of objects according to the viewing angle. Often used to simulate glass transparency or varnish.

Switch

Let you change an object's color based on location, angle of view, or another specified value.

Preset name	Output	Description
Boolean_switch	Color	Permits switching between two color types based on Boolean value.
Front_back	Color	Defines a color for each side of a surface.
Ray_type	Color	Defines which ray (eye, reflect, etc.) hitting an object will be seen. You can switch between several color inputs based on a ray type.

Preset name	Output	Description
Scalar_switch	Color	Permits switching between two color types based on a scalar value.
Vector_switch	Color	Permits switching between two color types based on a vector value.

Texture Generators

The basic tools for creating textures within a shader.

Preset name	Output	Description
2D_checkerboard	Color	Creates a basic checkerboard texture.
2D_dot	Color	Creates an adjustable polkadot pattern.
2D_fabric	Color	Basic tool to create a fabric pattern.
2D_gradient	Color	Creates a definable color gradient with an adjustable pattern.
2D_grid	Color	Basic tool to create a grid pattern.
2D_Image_implicit	Color	Applies an image as a texture using an implicit projection.
2D_ripple	Scalar	Basic tool to create concentric waves/ripples.
2D_terrain	Color	Basic tool to create a terrain texture.
3D_cloud	Color	Basic tool to create a cloud texture.
3D_fractal	Scalar	Basic tool to create a fractal texture.
3D_lattice	Color	Basic tool to create a lattice (grid) pattern with definable cell and crease colors.
3D_marble	Color	Basic tool to create a marble texture.
3D_rock	Scalar	Basic tool to create a rock texture.
3D_snow	Color	Basic tool to create a snow texture.
3D_turbulence	Scalar	Generates a definable turbulent pattern in an area.
3D_vein	Scalar	Basic tool to create a vein texture.
3D_wave	Color	Creates waves in the U, V, and W directions.
3D_wood	Color	Basic tool to create a wood texture.
Image_filtered_ lookup	Color	Looks up an image based on texture coordinates; the result is filtered (midmap).
Image_lookup	Color	Looks up an image based on texture coordinates; the result is non-filtered.

Texture Space Controller

Let you manipulate the texture of an object to suit the scene.

Preset name	Output	Description
3d_Warp	Vector	Distorts/perturbs a texture space.
Kaleidoscope	Vector	Creates a versatile Kaleidoscope look by altering texture space.
Texture_edit	Vector	Scales, rotates, repeats, and crops a texture space.
Texture_rotate	Vector	Rotates a texture projection.

Texture Space Generators

Lets you manipulate a texture space by perturbing, rotating, scaling, or using a number of other functions.

Preset name	Output	Description
Texture_space_ generator	Vector	Creates a Texture Space Projection to map an image/texture.

Volume

These shaders create a volumic effect within a scene, such as fog, clouds, or smoke.

Preset name	Output	Description
Constant_density	Color	Defines a density within an object.
Fast_volume_effects	Color	Volume shader used to create volumetric spotlights.
Volume_cigarette_ smoke	Color	Creates a cigarette-like smoke effect.
Volume_effects	Color	Creates a fog/dust-like effect.
Volume_fog	Color	Creates a fog effect in a scene.
Volume_layered_fog	Color	Creates fog with a vertical start and end point.
Volume_lightning	Color	Creates lightning within an object.
Volume_smoke	Color	Creates smoke within an object.

Index

Numerics

A

B

C

D

R

S

T

U

V

X

Y

Z