SOFTIMAGE®

SOFTIMAGE®|XSI™

Version 1.0

Rendering

Avid

Rendering 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. 0130-04616-01 0400

Contents

Roadmap

About This Guide

Rendering contains information and step-by-step procedures on how to set up a scene for rendering, how to optimize, and render. It also provides help on how to batch render using a script.

Chapter 1: Rendering Basics—the steps you may wish to follow during the rendering process in order to maximize workflow and optimize rendering. Also explains how you can preview a single frame of your final render.

Chapter 2: Passes & Partitions—how to divide a scene into passes and further customize their properties with partitions. Also explains how to quickly create matte, shadow, depth, and highlight passes.

Chapter 3: Rendering Options—describes all the options you can define when rendering a scene. Everything from format to aliasing is explained as you would set it. Also explains how you can optimize render time.

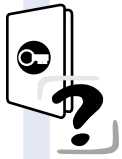
Chapter 4: Rendering—how to actually begin a render, whether you wish to use a single computer, a distributed-rendering network, or batch render using scripts.

Appendix: ***Setting Up Distributed Rendering***—describes the steps required to establish a distributed-rendering network.

This book is meant to be used in conjunction with the *Shaders, Lights & Cameras* guide.

Where to Find Information

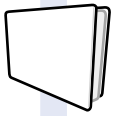
The SOFTIMAGE|XSI package includes a comprehensive set of learning materials. Use this Roadmap to find the information you need to get up and running quickly and effectively.



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.

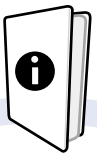


Work through ***Tutorials*** to learn the features in the context of basic productions.
This is a full-color set of lessons showing you step-by-step how to perform typical tasks.
You can install the scenes from the Software CD. (Choose Custom install when installing SOFTIMAGE|XSI). Then choose the **Content**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

The Softimage Discussion Group

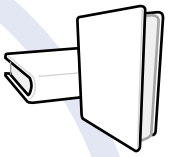
You can join the worldwide network of Softimage users exchanging ideas and techniques by e-mail. To find out more, e-mail **majordomo@softimage.com.** Leave the Subject line empty and type the word "help" in the body of your mail message.

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



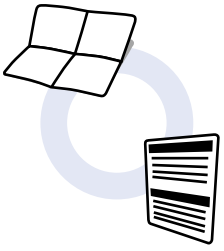
The Online Library CD
The Online Library contains the Guided Tour and all the SOFTIMAGE|XSI and some mental ray documentation in electronic form in both PDF and HTML formats. (See next page for how to use.)

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.

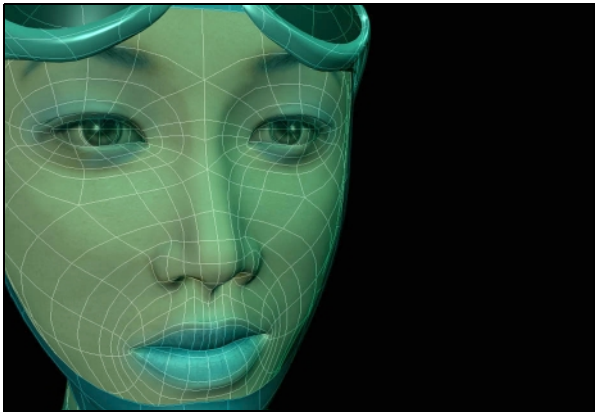
The two-button mouse is not supported in SOFTIMAGE|XSI.

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 Rendering Basics

After adjusting all of your lights and objects and defined your camera settings, you're ready to render your scene out. Rendering is one of the final steps you take once a scene has been created. Whether you're rendering a single frame or hours of animation, rendering a scene is like developing a photograph. The process is often done more than once, and you will most likely have to tweak and adjust your options to achieve the look you set out to create.



Once you've created and edited your scene to your liking, you can tweak the scene, break it up into passes, or perform a variety of tasks that not only enhance a scene but optimize rendering quality and speed.

SOFTIMAGE|XSI uses mental ray® version 2.1 rendering software.

Workflow Overview

This workflow outlines a simple but typical sequence of tasks you might follow when rendering. You do not need to follow these tasks strictly in order.

For more information on how to get the look you want in a scene, refer to *Chapter 1: Getting the Look You Want* on page 17 of the *Shaders, Lights & Cameras* guide.

Workflow outline

1. **Set up render passes** and define their options. Render passes let you render different aspects of your scene separately, such as a matte pass, a shadow pass, a highlight pass, or a complete beauty pass. You can define as many render passes as you want. Within each pass, you can create partitions of lights and objects, then apply shaders and control their settings together. For more information about using render passes, see *Chapter 2: Passes & Partitions* on page 23 of the *Shaders, Lights & Cameras* guide.
2. **Set rendering properties.** All objects—geometric objects, lights, and cameras—are defined by their rendering properties. For example, you can determine whether a geometric object is visible, whether its reflection is visible, and whether it casts shadows. Rendering properties can be set per pass as well. For more information, see ***Rendering Options* on page 49**.

3. **View the results** of any modifications. The viewports can display your scene in different modes, including wireframe, shade, texture, and rotoscope. In addition, you can view a portion of a viewport rendered with mental ray by defining a render region. You can preview a full frame using the Frame Preview button. For more information about viewing and previewing your scene, see *Setting Output Rendering Options (Global Only)* on page 53.

4. **Render the passes**. After previewing a few rendered frames, you can render on your local computer or perform a distributed render across a network of computers. Through a series of command-line entries, you can define computer groups, specify availability, verify the setup, schedule rendering jobs, and initiate rendering. For more information about rendering, see *Chapter 2: Passes & Partitions* on page 23.

5. **Composite the passes** using a post-production tool such as SOFTIMAGE®|DS or the composite standalone included with SOFTIMAGE|XSI (see the *Using Standalones* appendix in the *Fundamentals* guide).

After you have worked with SOFTIMAGE|XSI for a while, you might want to customize certain features**.** You can change the layout of your workspace as well as save various preferences. For more information, see *Customizing SOFTIMAGE|XSI* on page 211 in the *Fundamentals* guide.

Tools for Rendering

To help you obtain a final render, there are a few tools you should know about. These tools simplify your render tasks and help optimize your work time as well as your render times.

Layers

When you load a scene, all of the scene elements are loaded into memory. A large and complex scene can significantly slow down the refresh rate and clutter the display with many objects. Creating scene layers is useful when you wish to concentrate on only a few objects in a crowded scene or to save time required in refreshing the display when applying rendering options and effects.

Layers help you organize, view, and edit the elements in your scene but do not affect the final render and are not associated with render passes. You can put different objects into each layer and you can hide a particular layer if you don't want to render or see that part of your scene. You can also make a layer unselectable. Layer definitions are saved with the scene file.

For more information on how to use layers, see *Chapter 5: Working with Scene Elements* on page 144 of the *Fundamentals* guide.

Render Passes

A render pass creates a picture layer of a scene that can then be composited back with any other pass during the post-production 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. Passes also allow you to quickly re-render a single layer without re-rendering the entire scene.

Each scene can contain as many render passes as you need. You can use preset passes such as matte, shadow, and highlight passes, or you can create your own. One of the main reasons for creating passes is that you can change and re-render any one of these passes without needing to re-create each one from "scratch."

Each pass has its own rendering options. This lets you further speed up rendering by selecting only those options you need for each pass. For example, you could have shadow calculations only in the shadow pass. See *Chapter 2: Passes & Partitions* on page 23 for information on how to set your passes or *Rendering Options* on page 49 for information on defining your rendering options.

Render Options

Render options can be defined separately for both a render pass and the render region. A render pass and the render region can have completely different render options or, conversely, can share identical values so a rendering in the render region looks exactly like your final render.

These options include settings such as the rendered file name, format, and frames, antialiasing settings, effects such as motion blur and global illumination, and the rendering algorithm, acceleration method, and global-approximation methods for all geometric objects. Global-render settings are defined separately for each render pass; for example, you can define separate settings for highlight, matte, and shadow passes.

For some effects to be rendered, you must set global-rendering options. For example: for shadows, you first set rendering properties for the appropriate light, then set rendering properties for the objects you want to cast shadows, and finally set global options to render shadows on a particular pass.

For more information on how to set render options for your render pass or the render region, see *Chapter 3: Rendering Options* on page 49.

Previewing a Single Frame

The **Render** > **Preview** command in the Render toolbar displays a menu that lets you preview the current frame at fully rendered quality in a floating window. You can preview selected elements only, the entire scene and all its layers, or just its current or visible layers. The frame is rendered with the global-rendering options defined for the current render pass.

When the image is bigger than the frame-preview window size, as is the case with large 4K x 4K resolution images, scroll bars appear for you to view any part of the rendered image.

To preview a single frame

1. Choose **Render** > **Preview** from the Render toolbar.
2. Choose the type of render preview from the displayed list. The options perform the following type of render:
   - **All Layers:** Renders every layer regardless of whether it is visible or not. Layers that are set not to render will not be rendered.
   - **Visible Layers:** Only renders layers visible at the time of rendering.
   - **Current Layer:** Only renders the current layer. The current layer's name is displayed in the Layer panel.
   - **Selection:** Only renders the selected object or objects.

Setting Frame-Preview Options

The Frame Preview property page lets you specify the render characteristics of the current frame as selected by the **Render** > **Preview** command.

By default, Frame Preview uses the same render settings as defined for the current render pass

To set frame-preview options

1. Choose **Render** > **Preview** > **Options** from the Render toolbar to open the Frame Preview property editor.
2. If you wish the frame-preview area to have the same characteristics as those specified for the render region under the **Render** > **Region** > **Options** command, select **Use Region Settings**. When off, the preview uses the final output render options defined in **Render** > **Render** > **Options**.
3. In the **Format** box, specify the size (in pixels) of the preview area using the **X** and **Y Resolution** slider bars.

4. If you chose not to use the render region's settings, you can specify your own resolution by clicking on the arrow in the **Resolution Factor** text box. The default is 1:1, but you can select up to a 1:8 resolution factor.

5. If you wish to smooth the oblique edges of objects in the rendered frame by antialiasing, click the **Custom Sampling** check box and modify the **Min** and **Max** controls.

   The **Min** and **Max** controls allow you to specify the minimum and maximum sample rate. This is the number of samples taken to compare surrounding color values that are averaged to define the color of a pixel. If the Min value is 0, each pixel is sampled at least once.

   A Min value of –1 and Max value of +1 means that the image is subsampled every four pixels (a 2-by-2-pixel square). SOFTIMAGE|XSI takes two samples, one on pixel 1 and one on pixel 3, compares them, and, if the difference is greater than the contrast specified, it divides each pixel into four subpixels. If this still does not meet the supersampling threshold, aliasing appears.

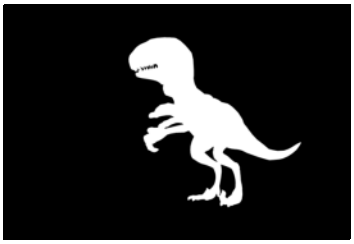
Chapter 2 Passes & Partitions

A render pass creates a picture layer of a scene that can be composited with any other passes during the post-production video-editing process. Passes also allow you to quickly re-render a single layer without re-rendering the entire scene. Later you can composite the rendered passes using the composite standalone or a compositing tool like SOFTIMAGE|DS.

Each scene can contain as many render passes as you need. When you first create a scene in SOFTIMAGE|XSI, it has a single pass named Default Pass. This is a "beauty pass" that is set to render every element of the scene—you can render a single beauty pass or you can render separate passes. Also, you can use preset passes such as matte, shadow, and highlight, or you can create your own. One of the main reasons for creating passes is that you can change and re-render any one of these passes without needing to re-create them all.

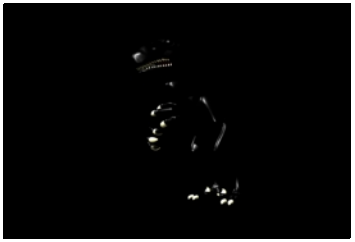
The above photograph (background pass) is the background scene the dinosaur will be composited with.

This pass is simply a rendered image of the dinosaur. Compositing him to the background would make the scene rather flat and unrealistic.



The matte pass is used when the dinosaur will be passing in front of or behind another rendered image (such as a telephone pole or mailbox). The matte "cuts out" a section of another rendered image so another image can be composited over or beneath it.

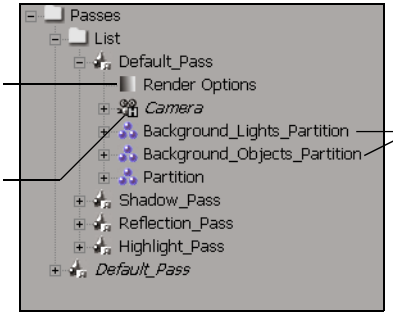
The shadow pass acts much like a matte pass but is created solely for an object's shadow that falls on other rendered images. You can manipulate this pass's transparency or sharpness to add realism when compositing with your final scene.



The specular pass is used to edit an object's highlights.

Here the passes are all composited together to create the final image.

Each pass has its own rendering options. This lets you further speed up rendering by selecting only those options you need for each pass. For example, you could have shadow calculations only in the shadow pass (see *Chapter 3: Rendering Options* on page 49 for information on how to set your rendering options). You can also define render-pass partitions (see *Defining Partitions* on page 38) and use them to organize the objects and lights in your passes.



What Is a Partition?

A partition is a division of a pass. Partitions are a simple method used to organize scene elements within a pass. There are only two types of partitions, object and light. Light partitions can only contain lights, and object partitions can only contain geometric objects. You can create as many partitions as you need and apply shaders to them. For more information on how to create partitions, see *Defining Partitions* on page 38.

What about Layers?

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.

Although similar in concept, there is a big difference between passes and layers. Layers are used to separate elements of a scene so they can be visually organized or not rendered to optimize workflow. Passes, on the other hand, separate scene elements for rendering so they can be individually tweaked or edited.

For more information on layers and how they are used, see *Chapter 4: Viewing Your Work* in the *Fundamentals* guide.

Working with Passes

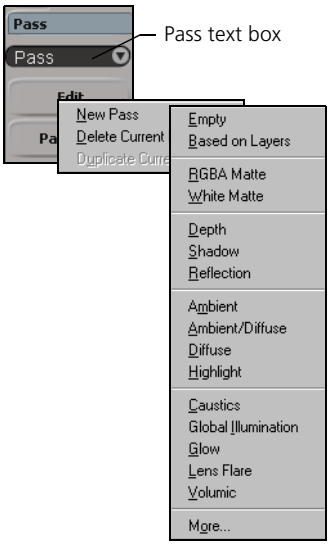
If you want to render or edit only certain aspects or areas of your scene, the following steps provide an overview of how to use render passes:

To use render passes

1. Create and name a new render pass. For details, see *Creating Custom Render Passes* on page 42.
2. When working with several passes, you can use the explorer scope to display either the Current Pass or All Passes of a scene. For more information on how to use the explorer, see the *Fundamentals* guide.
3. Define partitions and use them to organize and edit the objects and lights in your render passes. How you divide elements into partitions depends on what effect you want to achieve with the pass. This is described in more detail in *Defining Partitions* on page 38.
4. Specify the active camera for the pass, and apply a lens shader if desired. For more information, see *Applying Shaders to Passes and Partitions* on page 44.
5. Apply shaders (including special effects such as glows, environments, and volumic effects) to the pass and its partitions—see *Applying Shaders to Passes and Partitions* on page 44.
6. If necessary, define an override for a partition. An override lets you control an object's parameters using shaders without replacing any of an object's material properties. For more information on overrides, see *Applying Shaders Using Overrides* on page 46 and *Creating Property Overrides* on page 72 of the *Fundamentals* guide.
7. Set rendering options for the objects in each pass partition—see *Controlling the Active Camera in the Render Pass* on page 32.
8. Set global rendering options for each pass—see *Chapter 3: Rendering Options* on page 49.
9. After you have set up render passes, you can render them as described in *Chapter 4: Rendering* on page 85. You can then composite the passes using the composite standalone (see *Compositing Images* on page 109) or a compositing program like SOFTIMAGE|DS.
10. You can save and re-use your passes as described on page 42.

A new pass is created for every imported SOFTIMAGE|3D scene.

Creating Render Passes



When you create a scene, it has a single pass, named *Default_Pass*. This is a beauty pass that is set to render every element of the scene "as is." It is also selected as the current pass for the scene and its name appears in the pass text box.

You will most likely want to create more passes as your scene grows in size and complexity. You can create a variety of pass types from the Render toolbar.

To create a render pass

1. From the Render toolbar, click **Pass** > **Edit** > **New Pass** and select a type of pass to create. Your choices are as follows:

Pass	Description
Empty	Empty pass with empty object and light partitions
Based on layers	Pass with a partition for every defined layer
RGBA matte	Matte pass of the selected objects. This type of pass is regularly used to composite overlapping objects. Background objects and lights are not rendered but matte objects are rendered with their original RGB settings.
White matte	Matte pass with a constant white RGBA channel of selected objects. This type of pass is regularly used to composite overlapping objects. The matte pass renders matte objects in white. Background objects and lights are not rendered (black).
Depth	Pass of the entire scene's depth information. Also called a Z-pass, this type of pass contains depth information for objects based on the distance from the camera.
Shadow	Shadow pass of the entire scene. Rendering a shadow on its own and being able to composite it back in the final render allows you to edit a shadow's blur, intensity, and color without any additional rendering.
Reflection	Pass containing every selected object's reflections.
Ambient	Pass containing all of a scene's ambient color.
Ambient/Diffuse	Pass containing all of a scene's ambient and diffuse information.
Diffuse	Pass containing all of a scene's diffuse color.
Highlight (or Specular)	Pass containing the highlight of the entire scene. A highlight pass (sometimes called specular) contains all of a scene's specular highlights or hotspots.
Caustics	Pass of all of the scene's caustic effects.

Pass	Description
Global illumination	Pass of all of the scene's global illumination effects.
Glow	Pass of every selected objects' glow property.
Lens flare	Flare pass for the selected camera or light.
Volumic	Volume pass for each selected object in your scene.
More...	Opens a browser allowing you to select a pass preset.

There are many more types of passes you can create from scratch by selecting the empty option and custom-defining the pass. In addition, you can create your own variations of existing passes.

To load a pass preset

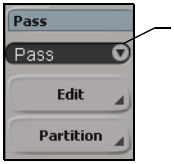
If you do not want to use any of the pass presets, you can easily load your own.

1. Choose **Pass** > **Edit**> **New Pass** > **More...** from the Render toolbar.
2. In the browser, navigate to a folder in which you have a saved preset pass.
3. Click OK to load the preset.

You can also drag a preset pass from the browser or custom toolbar and drop it directly on the Render Passes root in the explorer. The preset pass appears as a subnode under the Rendering Passes node.

Setting the Current Pass

The current pass is the pass to which all pass and partition properties are applied. The current pass is also the pass displayed in the viewport when the Render Pass view is chosen from the viewport View menu.



To set the current pass

1. Click the arrow button beside the Pass text box on the toolbar to display the drop-down pass list. The list contains all available passes defined for the scene.
2. Choose the render pass you want to set as current; all work is then applied to the current pass. To make a change to another pass, set it as the current pass.

Use the Pass commands on the toolbar to set the camera for the pass, add a partition, or delete the current pass. See *Creating Custom Render Passes* on page 42.

Every time you create a new pass, it immediately becomes the current pass.

Controlling the Active Camera in the Render Pass

You can specify the camera you want to use for each render pass. The active camera provides the viewpoint from which the pass is rendered. You can add new cameras to your scene and set them as active if needed.

In the explorer, a Pass Camera node appears as a subnode of each render pass. This doesn't signify that a new camera is created with each pass. The Camera node represents the camera for that pass only.

To set the active camera for a render pass

1. Choose the current pass from the Render toolbar. To do this, click the arrow button beside the Pass text box to display the drop-down pass list. The list contains all available passes defined for the scene. Choose the render pass you want to set as current.
2. Click on **Render** > **Render** > **Options** to open the Render Options property editor of the current pass.
3. Click the **Output** tab and define from the Listed cameras which camera you want to be active for your render pass. The drop-down menu lists all of a scene's cameras.

To view the current pass from the viewpoint of the active camera, click the arrow button on the left side of any viewport's title bar to choose **Views** > **Render Pass**.

When working in the explorer, you can also drag and drop the camera icon into a pass to set the active camera for that pass.

The Beauty Pass

When you start SOFTIMAGE|XSI, the default pass in place is a complete pass that includes everything in the scene. It contains all the elements for the imported scene to be fully rendered in a single "beauty pass." You can render the beauty pass as is—with its default settings—or you can customize the pass by renaming it, setting new options, or adding partitions.

A new beauty pass is created for every imported SOFTIMAGE|3D scene.

You can also edit the pass to suit your needs by loading a pass preset or creating a new pass from scratch. See *Creating Custom Render Passes* on page 42 for information on how to do this.

When you first create a scene, the default beauty pass is the current pass. Any pass commands that you apply, such as setting the pass camera or adding a partition, affects the current pass (see *Setting the Current Pass* on page 31).

Imported Scenes

Global-rendering options for the default pass are read in from the imported scene. All settings defined in **.mi**, **.dsc,** or other scene files are added to the beauty pass. This includes any environment, output, and volume shaders that have been set for the scene.

Default-Pass Settings

The default pass includes the following partitions, nodes, and settings:

- **Background Objects Partition**—automatically includes all objects in the scene.
- **Background Lights Partition**—automatically includes all lights in the scene.
- **Environment, Volume, and Output shaders**—all global shaders imported with the scene are applied to the default pass. Each new pass you create has its own list of output, environment, and volume shaders; therefore, you can add a specific shader to a specific pass.
- **Pass Camera**—automatically set to the camera defined for the imported scene.

The Shadow Pass

Rendering a shadow on its own and being able to composite it back in the final render allows you to edit a shadow's blur, intensity, and color without any additional rendering. This can be a great asset when production time is limited.

The preset creates an alpha matte of the shadows cast by every object in the scene and by all of the scene's lights.

To set up the shadow-pass preset

1. Load the shadow-pass preset from the Render toolbar by selecting **Pass** > **Edit** > **New Pass** > **Shadow**. A shadow pass for the entire scene is created. The Shadow Pass property editor appears.
2. Rename the pass and, if you wish, set the active camera and rendering options. For general information about choosing rendering options, see *Chapter 3: Rendering Options* on page 49. For specific information about each of the options in this property editor, see Online Help by clicking (?) in the property editor.

3. If not already set, change the viewport display type to view the render pass by selecting **Cameras** > **Render Pass** in the viewport Views menu.
4. You can further customize the shadow preset by renaming the pass and adding more partitions. See *Creating Custom Render Passes* on page 42.
5. When you are satisfied with the settings, preview the fully rendered frame by choosing **Render** > **Preview** > **All Layers** from the toolbar.
6. To render the pass, see *Chapter 4: Rendering* on page 85.

The Matte Pass

You can use two types of matte passes: RGBA and white. Matte passes are usually used to create a "cut-out" of one object over another. For example, if you wish to render a car driving along a street, you must create a matte pass of the lamp posts; otherwise, the car render will overlap the lamp posts. The lamp posts' matte is used to obscure the render of the car so it appears to be *behind* the posts.

Basic Matte Settings

- **RGB Matte**: Creates a matte pass of selected objects. Regularly used to composite overlapping objects. Background objects and lights are not rendered, but matte objects are rendered with their original RGB settings.
- **White Matte**: Creates a matte pass with a constant white RGBA channel of selected objects. Regularly used to composite overlapping objects. The matte pass renders matte objects in white; background objects and lights are not rendered.

The matte pass includes the following settings:

- **Background Objects Partition**—automatically includes all objects that are not explicitly assigned to the Matte Objects partition. Visibility is turned off for primary and secondary rays for these objects so that they are not rendered.
- **Background Lights Partition**—automatically includes all lights for the scene.
- **Matte Objects partition**—these objects are rendered in white, including their shadows and any motion blur defined for them. The Constant material shader is usually applied to objects in a matte pass partition.

To set up the matte-pass preset

1. Select the object or objects for which you wish to create a matte (RGBA or White).
2. Load the matte-pass preset from the Render toolbar by selecting **Pass** > **Edit** > **New Pass** > **RGBA Matte** or **White Matte**.

   All other objects are contained in the Background Objects Partition and are rendered black.
3. The Render Pass property page opens. Rename the new pass and, if you wish, set an active camera and render options. For general information about choosing rendering options, see *Chapter 3: Rendering Options* on page 49. For specific information about each of the options in this property editor, see Online Help by clicking (?) in the property editor.

4. When you are satisfied with the settings, preview a fully rendered frame by choosing **Render** > **Preview** > **All Layers** from the toolbar.
5. To render the pass, see *Chapter 4: Rendering* on page 85.

The Depth Pass

The depth pass creates a black and white gradient based on depth information. White represents a distance close to the camera, and black is very far. A depth map's color and unit scaling can be customized.

The depth pass isolates depth information. When it is composited with the rest of the scene, it applies a fog or depth fading to objects that are further from the camera. In this example, the smaller dino-saur is farther back in the scene, hence darker in the depth pass. If you were to composite this scene with-out a depth pass, the dino-saur in the background would not be faded or dimmed and look like a baby dinosaur at its mother's feet.

To create a depth pass

1. Load the depth-pass preset from the Render toolbar by selecting **Pass** > **Edit** > **New Pass** > **Depth**. A depth pass for the entire scene is created, and the Pass property editor appears.
2. Rename the depth pass and, if you wish, set the active camera and rendering options. For general information about choosing rendering options, see *Chapter 3: Rendering Options* on page 49.
3. To edit the depth-pass parameters, click the **Volume Shader** tab from the depth-pass property editor.
4. Select the Constant Density shader from the shader stack and press **Inspect** to open the Constant Density property editor.

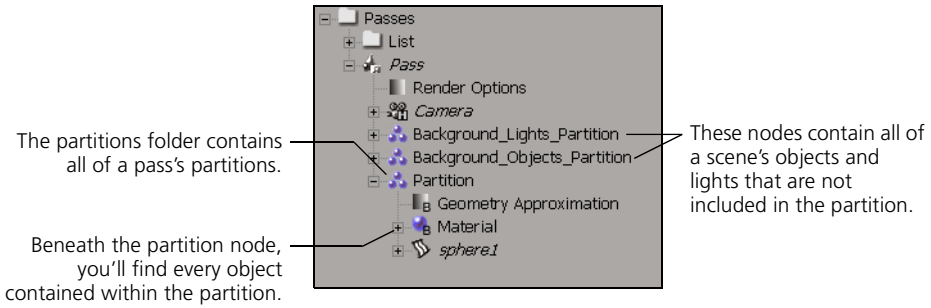
   From the property editor, you can define the Unit Scaling of the depth pass. The scaling determines the depth use (in SOFTIMAGE units) to display the defined Density. Use 1:100 to create a detailed depth pass of a scene that extends far from the camera and 1:1 for a scene that is relatively flat.

For specific information about each of the options in this property editor, see Online Help by clicking (?) in the property editor.

5. If not already set, change the viewport display type to view the Render Pass by selecting **Cameras** > **Render Pass**.
6. When you are satisfied with the settings, preview the fully rendered frame by choosing **Render** > **Preview** > **All Layers** from the toolbar.
7. To render the pass, see *Chapter 4: Rendering* on page 85.

Defining Partitions

Partitions let you organize elements (either geometric objects or lights) within each render pass. You can add properties to a partition as well as apply shaders to an entire partition at once. The properties or shaders can then be edited through the explorer or the render tree. For more information, see *Applying Shaders to Passes and Partitions* on page 44.



When working with several layers or partitions, you can set the explorer scope to either **All Passes** in a scene or only the **Current Pass**.

Background Partitions

Every render pass contains two special partitions: Background Objects Partition and Background Lights Partition. These partitions automatically include all the objects and lights not explicitly assigned to other partitions. Scene elements can only belong to one partition per pass, and you can add only geometric objects to object partitions; only lights to light partitions.

This lets you conveniently treat all the background elements together. For example, when you are defining a matte pass, you can turn off the visibility of all objects in the Background Objects partition so that they are excluded from the rendering process. You can define the contents of the background partitions by removing objects from the background partition.

Creating Partitions

There are several ways to create a partition: you can create an empty one and add elements to it, or you can instantly create one that already includes the elements you have selected. Either way, you can add or remove elements as described in the next section.

To create an empty partition

1. Select the render pass into which you want to add the partition. To do this, choose the pass name from the Render toolbar and click the arrow button beside the Pass text box to display the drop-down pass list; the list contains all available passes defined for the scene. Choose the render pass you want to set as current.

2. Make sure that no objects are selected in your scene.
3. Choose **Pass** > **Partition** > **New** from the toolbar to add a partition to the current pass. An empty object partition has now been created.
4. You can now name and add the appropriate objects to your new partition. See *To add elements to a partition* on page 39.

If a geometric object is selected when a new partition is created, the object is placed inside the (object) partition. If a light is selected when you make a new partition, the (light) partition contains the selected light. For more information, see the following section.

To create a partition that includes elements

There are two ways to create a partition that already includes a selected object.

1. Make the render pass (to which you want to add the partition) the current pass.
2. Select the objects or lights you want to be included in your partition.
3. Choose **Pass** > **Partition** > **New** from the Render toolbar to add a partition to the current pass. A new partition is created with a partition that contains the elements you selected. If you select both objects and lights, two separate partitions are created.

Either a light or object partition is created, depending on what type of object you selected. If you selected both lights and objects, two separate partitions are created: one for the lights and one for the objects. The new partitions appear as subnodes of the selected render pass in the explorer.

Dragging and Dropping Partitions

You can also drag and drop partitions from one pass to another instead of recreating them for each render pass. This method is equivalent to copying and pasting the objects and attributes of one partition into another. Dragging and dropping a partition keeps all of the changes made to that partition until it is edited. You can also delete or remove partitions from passes by dragging and dropping them.

To add elements to a partition

Rather than using the drag and drop technique, you can add elements to your partitions by choosing **Add to Partition** in the Render toolbar.

1. In the explorer, select the partition to which you want to add elements.
2. In the viewport, select the element or elements you want to add to the partition. Select only elements of the same type: you can add only geometric objects to object partitions, and only lights to light partitions.

3. In addition to selecting elements, **Shift+click** to select the partition to which you wish to add the elements.

To add elements to a partition, the objects must be multi-selected with the partition node. Press the space bar to activate selection mode, and select the elements as well as the partition; hold the Shift key down for multiple selection.

To remove an element from the current selection, hold down the Ctrl key and select the unwanted element. Press the Shift key again and continue your multiple selection.

4. Choose **Pass** > **Partition** > **Add to Partition** from the toolbar. The selected elements are added to the selected partition in the current render pass

You can also add elements to partitions by dragging and dropping the elements from one pass directly onto a partition node in the explorer. The elements are added as subnodes of the partition. Notice that the elements still appear under the Scene node as usual, but now they are also associated to a particular render partition.

To remove elements from a partition

While building your render-pass partitions, you may have to remove elements from an existing partition. To do this:

1. In the explorer, expand the pass partition (click the + sign) you want to modify to display the elements it contains.
2. Select the elements you want to remove from the partition.
3. Choose **Pass** > **Partition** > **Remove from Partition** from the Render toolbar. The selected elements are removed from the partition in the current render pass.

To delete a render-pass partition

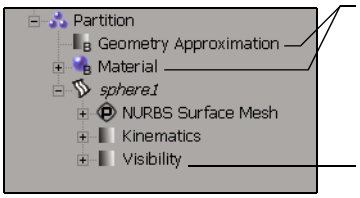
1. In the explorer, click on a render-pass partition node to select it.
2. Press the **Delete** key to delete the selected partition in the current render pass.

Setting Partition Properties

When you create a new pass partition, it has no default properties. You have to assign the partition properties, depending on the element in the partition (light or object). You can then set properties for all elements in a partition by opening the property editor for that partition and specifying values. The partition properties override the individual objects' properties for the current render pass.

There are two ways to set the properties in a partition: through the object or by creating a partition property that applies to every object or light in a given partition.

You can access a partition's property editors by opening an explorer view set to Passes. Expose the property editors by expanding the partition nodes you wish to edit.



Any values you set or options you change at the partition level are propagated to the objects as though they were children nodes of the partition property. If you make no changes to the partition properties, SOFTIMAGE|XSI uses the properties for the individual objects when rendering the pass.

You can also set properties for the individual elements in your partitions. For more information, see *Applying Shaders Using Overrides* on page 46.

Example: Setting Partition Properties

Let's say you have two geometric objects in partition A—one has a reflective surface and the other object is a simple, flat gray. When you open the property editor for partition A, the Material option is displayed in an empty state; that is, they are neither selected nor deselected.

1. With Partition A selected, choose **Get** > **Material** > **Phong** from a toolbar. A Material and surface shader are assigned to the partition, and the Phong property editor appears in a floating window. The newly created partition property is marked with a B. This signifies that the property was created in a branch and its values are propagated down to the objects in the partition.
2. If you want to change the surface's color to yellow, for example, adjust the RGB Diffuse slider of the Phong property editor.
3. Close the property editor. Now both objects are yellow but their reflection settings remain as is.

For more information about setting object properties, see *Chapter 1: Getting the Look You Want* on page 17 of the *Shaders, Lights & Cameras* guide.

Creating Custom Render Passes

If you want to isolate and render particular aspects of your scene that are not defined by any of the preset passes—matte, shadows, highlights, or others—then you can create your own render passes. Your scene can contain as many render passes as you need.

After you have created a new render pass, you can define its objects and light partitions and apply shaders to the pass and its partitions. Keep in mind that you can also load a preset pass and customize it to fit your particular needs.

You can create a custom render pass in several ways. The following covers the many tools and views you can use to do this.

There are two ways to create a custom pass

- Choose **Pass** > **Edit** > **New Pass** > **Empty** from the Render toolbar. This not only creates an empty new pass named "Pass," it also creates empty background partitions. Each new pass is independent of the original Default_Pass. For more information on this pass, see *The Beauty Pass* on page 33.

  or

- Select **Pass** > **Edit** > **New Pass** > **Based on Layers** from the Render toolbar. This creates a new pass and a partition for each layer you have defined in the Layers Control property editor.

  The new pass is automatically the current pass now. Any further pass or partition commands are applied to this pass until you select or create another pass.

If you want to duplicate a pass, simply select **Edit** > **Duplicate Single** from the Edit panel.

4. The pass has been created; now you have to fill it with objects. Select the objects you wish to include in your pass and click **Pass** > **Partition** > **New**. This places a copy of each selected object in a new partition.

5. Once your passes and partitions are organized, you can begin applying properties to the passes and override properties to the partition elements. For a complete description of these steps, see *Setting Partition Properties* on page 41, *Applying Shaders Using Overrides* on page 46 and *Creating Property Overrides* on page 72 of the *Fundamentals* guide.

6. When you are ready to render your pass, there are two ways to proceed:

   Choose **Render** > **Render** > **Options** from the toolbar to open the Rendering Options property editor for the current pass.

   or

   In the explorer, open the render pass node and click on the **Render Options** icon:

- Make sure you specify a new file name for the rendered frames in the File text box in the Output section of the General property page; otherwise, the rendered frames of different passes overwrite each other.
- Some of the tasks you can accomplish in this property editor, such as sampling, antialiasing, and applying motion blur, are described in *Chapter 3: Rendering Options* on page 49. You can display this property editor again to modify options at a later time.
- For specific information about each option in this property editor, see Online Help by clicking (?) in the property editor.

Render options are controlled per pass; that is, the changes you make to the Render Options property page are for the current pass only. You can define the options for each render pass before it is rendered.

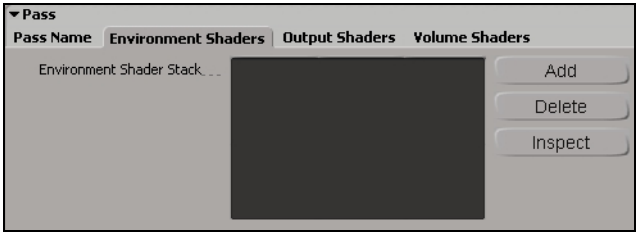
To delete a render pass

1. Make the render pass you want to delete the current pass. To do this, click the arrow button beside the Pass text box to display the drop-down list containing all the passes defined for the scene. Choose the render pass you want to delete.
2. Choose **Pass** > **Edit** > **Delete Current Pass** from the Render toolbar to delete the current pass.

Applying Shaders to Passes and Partitions

When you apply shaders to partitions, they override the shaders applied individually to elements in the scene for the selected pass only. This means you can change the properties of elements on a particular pass without losing the properties of elements as defined for the whole scene. Once you have applied a shader to a partition, you can open its property editor and change values as necessary.

Applying global shaders—that is, environment, volume, and output shaders—to a pass is not done the same way. Although both steps use the explorer, applying a shader to a pass uses the Render Pass property editor, also called the *shader stack*. Since global shaders are attached to the pass, you can have different sets of global shaders per pass.

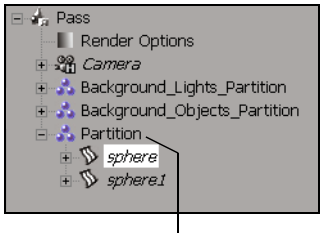


To apply a shader to a pass

A render pass can be defined by a stack of environment, volume, and output shaders. You can add more shaders, delete them, or arrange their order of execution from the Render Pass property page.

Click the Pass icon to open the pass property editor.

1. Open the explorer and choose the **Passes** view mode.
2. Select a pass to which you will add a shader, and click on the render-pass icon of the pass you wish to edit. The Render Pass property editor opens in a 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 the shader stack.
4. 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.
5. Select any shader from the list and click the **Inspect** button to display its property editor. You can now edit its properties.
6. Click the **Delete** button to remove the shader from the list. This means that it is no longer applied to the scene.



Select a partition to apply a shader to it.

To apply a shader to a partition

Applying a shader to a partition is very similar to applying a shader to an object using the explorer.

1. In the explorer, expand a pass (click the + sign) to display its partitions.
2. Select the partition node to which you want to apply a shader.
3. Choose **Get** > **Material** > **More...** from a toolbar to open a browser.
4. In the browser, select the desired shader and click OK to apply a shader to a partition.

You can also drag and drop a shader from either another partition or a browser.

5. Once the shader is applied, its property editor is automatically opened for you to open its properties. For more information about applying shaders, see *Chapter 2: Shader Basics* on page 37 of the *Shaders, Lights & Cameras* guide.

Because there is a wide range of setups and possible shader combinations, some invalid results may appear if a conflicting shader setup has been created. For example, two conflicting environment shaders would not produce a result.

Applying Shaders Using Overrides

Objects with a texture
(and diffuse, ambient, and specular values)

An *override* allows you to override a parameter, shader, or any type of node to any shader parameter of any object or objects in a partition. For example, if a scene contains several hundred objects and you wish to add a texture to each object's specular value, you would simply create a pass partition that contains all the objects you want to affect, and you'd define an override property linked to all of their materials' specular parameters.

From there you can simply attach a texture shader node to a single override node in the render tree or explorer. By using the override, the texture is only added on to the specular value and the remaining values (ambient, diffuse, etc.) remain intact. In addition, the original material—or other type of shader—applied to the object is not removed. In turn, this lets you control the specific shader connection of multiple materials with different render trees.

Override
Same objects with an override applied to remove the Ambient and Diffuse values

No override
The same objects but without an override. Instead, a Constant Black surface shader was applied and the Diffuse and Ambient values. Notice how the texture has been overridden and the bump value lost.

Shader overrides are extremely useful when it comes to adding extra properties onto an existing material. Using the example above (texturing a specular value), if you were to drop or add the texture directly into the partition it would override any textures or materials assigned to your objects.

Overrides also make it easy to add a shader to several parameters at once. Again, using the above example, it would be relatively easy to add a texture to a single specular parameter. But what if you had 315 objects, each one with a different material?

The override tool locates every specular parameter for every object in a partition and applies it on top—not instead—of its existing properties.

You can also use the override tool to control the property sets of several objects. For more information, see *Creating Property Overrides* on page 72 of the *Fundamentals* guide.

Override properties only work on SOFTIMAGE|XSI shaders. The soft_material shader does not permit any override connections.

To add an override

Once you have created a non-empty partition pass, you can select specific properties from one or several of its objects and add another property to them.

1. In the explorer view, select the partition to which you want to add an override.
2. Select **Get** > **Property** > **Override** from a toolbar; the Override property editor appears, and you can name the override. Don't close the property editor yet.

To define an override

3. Click on the **Add Parameters** button. A mini explorer window appears listing each object in the partition. Expand the object or light's surface node until you expose its sub-nodes.
4. Click on any shader parameter's name (diffuse, specular, etc.) to include it with your override property; each parameter you select is highlighted blue. To select several parameters, use Shift+click.
5. Click outside the explorer window once you have selected the properties to close the window. The override property editor lists the parameters you have chosen.

To edit an override

1. From an explorer, select the override by clicking on its name.
2. Open the render tree to see the node representation of the override property. If nothing is visible in the render tree work area, click on **Update**.
3. Add and edit shader nodes as you normally would in the render tree. For more information on working with the render tree, see *Chapter 9: The Render Tree* on page 208.

For more information on how to use overrides with property sets, see *Creating Property Overrides* on page 72 of the *Fundamentals* guide.

Chapter 3 Rendering Options

Global-rendering options apply to the entire render pass and, to a lesser extent, to the render region. They include settings such as the rendered file name, format and frames, antialiasing settings, effects such as motion blur and global illumination, and the rendering algorithm, acceleration method, and geometric approximation methods for all geometric objects.

Global render settings are defined separately for each render pass; for example, you can define separate settings for highlight, matte, and shadow passes.

Render-region render settings are very similar to the global options with the exception of the Format and Output property pages. There is no specific output other than the image rendered in the render region itself. The render region renders the current pass.

For some effects to be rendered, you must set specific rendering options. For example, for shadows, you first set rendering properties for the appropriate light, then set rendering properties for the objects you want to cast shadows, and finally set global options to render shadows on a particular pass. Similarly, you must also set the proper render option to render motion blur.

Setting Global Rendering Options

Set rendering options in the Rendering Options property editor of the current render pass or from the Render Setup of the render region. Most of these options are described in this chapter.

To set global-rendering options

1. 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 text box on the toolbar).

   or

   Choose **Render** > **Region** > **Options** from the Render toolbar to open the render-region Rendering Options property editor for the current pass (the current pass is the one displayed in the Pass text box on the toolbar).

2. Select the following options as you need them:

   - **Output**—for information about the output options**,** see *Setting Output Rendering Options (Global Only)* on page 53.
   - **Format**—for more information on image format and image resolution, see *Specifying the Format Options (Global Only)* on page 55.
   - **Aliasing**—for information about controlling aliasing and image resolution, see *Controlling Aliasing* on page 64.
   - **Image Process**—for more information on image processes that can be applied to your render, see *Applying an Image Process* **on page 67**.
   - **Activating Effects**—for information about enabling shaders and light options, see *Activating Effects* on page 68.

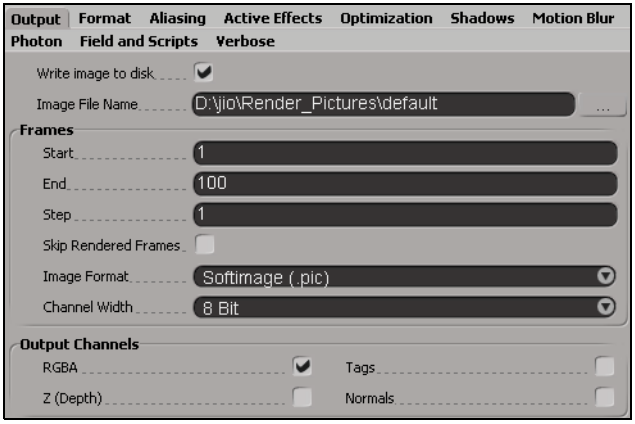
- **Shadows**—for information about enabling and disabling shadows, see *Setting Shadow Options* on page 68.
- **Motion Blur**—for information about motion blur, see *Setting Motion Blur Options* on page 69.
- **Photon**—for information about caustics and global illumination, see *Setting Photon Options* on page 69.
- **Optimization**—for information about the scanline and **r**aytracing renderers and the raytracing settings, see *Selecting a Rendering Method* on page 57. For more information on rendering acceleration, see *Acceleration Methods for Raytracing* on page 60.
- **Field and Scripts**—for information on field rendering and scripts, see *Field Rendering* on page 70 and *Using Pre- and Post-scripts* on page 73.

For a complete description of every option in the Render Options property editor, click (?) in the property editor to view Online Help.

In addition, *General Memory Requirements* on page 75 gives some suggestions for increasing rendering speed, and *Setting Output Rendering Options (Global Only)* on page 53 can help you tweak a final render frame before starting a full render.

Setting Output Rendering Options (Global Only)

Output rendering options you can set include the output file's name, which frames to render, their format, and which channels to output (e.g., RGB, depth, etc.). They are set on the Output page of the Rendering Options property editor. All of these rendering options are saved separately for each render pass.



The Output options specify the file name, frame range, and channels of the rendered output.

To specify an output file

1. 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 text box on the toolbar).

     or

   - In the explorer, right-click on a render pass and choose **Render Options** from the context menu. The Rendering Options property editor opens.

2. Open the **Output** property page and type an absolute or relative path and file name for the output in the **Image File Name** text box. You can also click the browser button (...) to specify a file name and directory from a browser. The rendered images have the file-name format **`filename.framenumber.extension`**, where the extension is determined by the image format.

3. Set **Start** to the first frame you want to render and **End** to the last frame you want to render.

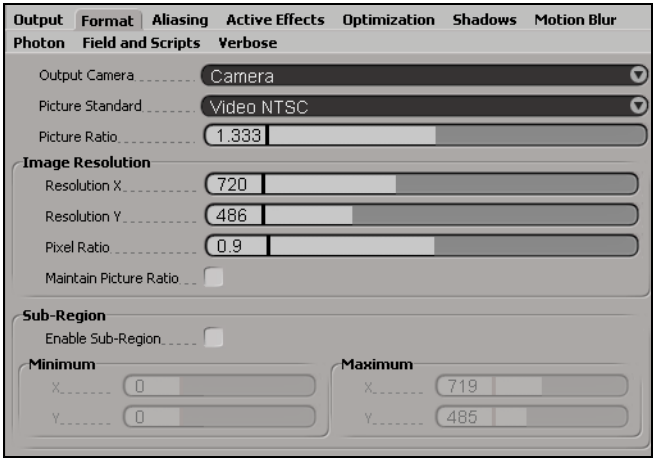
4. Set **Step** to the desired increment between rendered frames. This is useful for rendering a quick test. For example, if you set the **Step** to 2, every other frame is rendered. The default of 1 renders every frame between the Start and End frames.

You can use the **Skip Rendered Frames** checkbox to skip over images that are already rendered (or being rendered) in the directory.

5. Select a format for the image files from the **Image Format** list. The default is the **PIC** format.
6. Chose a **Channel Width** for your render. Your options are 8-bit and 16-bit. The default value is 8-bit.
7. Select the channels you want to render:
   - **RGB** for red, green, and blue channels.
   - **Alpha** for the alpha channel. If you select a format that does not support an alpha channel (for example, Windows Bitmap, TIFF, etc.), a separate **.a** file is written.
   - **Z depth** for the Z channel. The Z-channel information from a 3D scene provides depth information so you can position an object in front of and behind the background image when compositing. This allows for more advanced compositing operations. The rendered file name has the **.zpic** extension.
   - **Tags** for tag channels. With tag-channel information from a SOFTIMAGE|3D scene, post-processing utilities can isolate pixels associated with a given object and process only those pixels. Without the tag information, the post-processing utility would have to either locate the edges of a specific object or apply an effect to the entire image. The rendered file name has the format ***tagname.framenumber.tag***.
   - **Normals** for the surface normal of the triangle nearest the camera instead of the RGB value for each pixel. The rendered file name has the **.n** extension.

Specifying the Format Options (Global Only)

The Format options specify the resolution and aspect ratio of the rendered files. You can select a standard format or specify your own custom format and define an output camera for the render pass. With custom formats, you can compensate for non-square pixels on the final viewing equipment.



To define a render-pass camera

You must specify which camera a render pass will use, otherwise it uses the scene's default camera (or the first camera from an imported scene). The camera's picture ratio is displayed as a reference.

1. 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 text box on the toolbar).

     or

   - In the explorer, right-click on a render pass and choose **Render Options** from the context menu. The Rendering Options property editor opens.

2. Open the **Format** property page and click on the arrow at the right end of the **Output Camera** text box. A list of all of the scene's cameras is displayed. Select the camera you want the render pass to use when rendering your scene.

Using the Output property page, you can specify a camera to be used for a certain frame range and another camera for a different range of frames. For example, Camera A can be used to render frames 1–30 and Camera B could be used for frames 31–100.

3. Select the picture standard you wish to output your render in. The default is the NTSC Video standard.

4. The **Picture Ratio** for the camera you have selected appears beneath the output camera's name. This value cannot be edited unless you select **Custom** from the **Picture Standard** drop-down menu. Editing this value changes the X and Y resolution of your output image(s).

To render with a custom resolution and aspect ratio

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass (the current pass Pass text box on the toolbar).

2. Open the **Format** property page. In the **Resolution X** text box, specify the width of the image in pixels or use the slider to set a value. In the **Resolution Y** text box, specify the height of the image in pixels or use the slider to specify a value.

If you select **Maintain Picture Ratio** and change **Size X, Size Y** scales automatically, and vice versa, to maintain a constant and proportional picture ratio.

To render a subregion

If you are performing rendering tests and are only interested in a portion of the image, you can render a subregion.

1. Open the Rendering Options property editor for the current pass (the current pass is the one displayed in the Pass text box on the toolbar).

2. Open the **Format** property editor and click on **Enable Sub-region**. This activates the position sliders.

3. Set the **X** and **Y Minimum** and **Maximum** borders of the subregion you want to render in terms of the units you specified. The origin is the upper-left corner of the whole image.

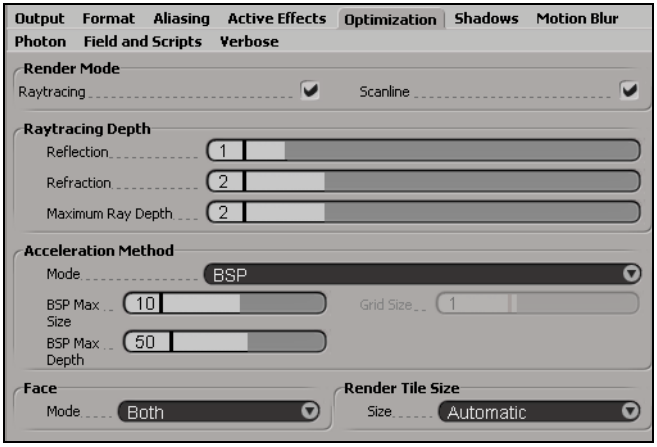
To render the whole image again, deselect **Enable Sub-region.**

Selecting a Rendering Method

The mental ray rendering software combines two rendering methods:

- **Scanline**
- **Raytracing**

Normally, mental ray uses a combination of scanline and raytracing algorithms to calculate samples of the scene. Without scanline rendering, the render is usually slower. Without raytracing, reflection rays cannot be cast and refraction rays are computed like transparent rays, which do not allow control over the ray direction based on the index of refraction of the surface shader. Lens shaders cannot alter ray origin and direction. However, environment maps can still work without raytracing.



Scanline

Scanline rendering is a rendering method used to determine primary visible surfaces. The image is rendered scanline by scanline rather than object by object. Scanline rendering is faster than raytracing but does not produce as realistic results.

This scene was rendered using the scanline rendering method. Notice how the reflections and transparency have little to no depth.

To render using the scanline method

1. 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 text box on the toolbar).
2. On the Optimization property page of the Rendering Options property editor, select **Scanline**.

Raytracing

Raytracing calculates the light rays that are reflected, refracted, and obstructed by surfaces and volumes. Raytracing produces realistic and precise results, but it can be time-consuming.

In raytracing, each refraction or reflection of a light ray creates a new branch of that ray when it bounces off a solid object and is cast in another direction. The various branches of rays in a scene constitute a ray tree. Each new branch can be thought of as a layer: if you add together the total number of a ray's layers, it represents the depth of that ray.

If you have a scene containing reflections, refraction, transparency, or shadows, it will take a fairly long time to render; however, mental ray lets you render only what you want. For example, if you want to render only the reflections, you can deselect all of the other features and show only these options. This takes less time to render, yet you benefit from the realistic results of raytracing.

This scene was rendered using the raytracing render method. Notice how the glass reflections, transparencies, and refractions are much more realistic than with Scanline rendering.

To use raytracing

1. 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 text box on the toolbar).
2. Open the Optimization property page and select **Raytracing**.
3. In the Ray Depth section, specify the **Ray Depth** options. These options define the depth of a ray, which means you are actually defining the maximum number of times a ray can be reflected or refracted in the scene. This helps limit the rendering time.
   - **Reflection** ray depth lets you specify the maximum number of a ray's reflected branches in a scene. For example, in a totally reflective scene, the ray continuously bounces around in the scene creating an infinite number of branches. This option lets you set an upper limit on these calculations.
   - **Refraction** ray depth allows you to set the maximum number of times a ray can be refracted in a scene.
   - **Maximum Ray 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.
4. Specify an acceleration method: **BSP** or **Grid** acceleration. These methods and their options are described in more detail in the following section.

Acceleration Methods for Raytracing

You can specify what type of raytracing acceleration method to use. This option is accessed from the Rendering Options property page.

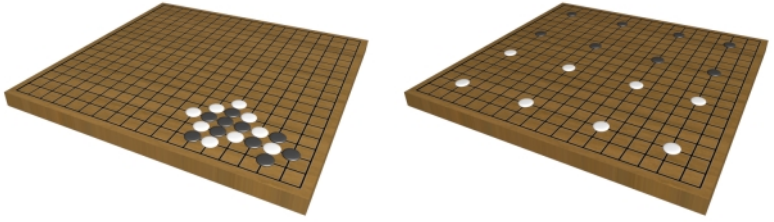
Various acceleration methods are used to accelerate the search of ray triangle hits in a scene. Acceleration algorithms do this by "bundling" a scene's triangles into cells and optimizing cell-stepping speed and/or minimizing the number of ray-triangle intersection computations by making assumptions about the scene that is being rendered.

The available acceleration methods for raytracing are BSP (binary space partitioning) and Grid acceleration. A general rule for choosing between these methods is the overall complexity of your scene. If the scene information shows that there are less than 150 000 triangles, then you should choose the BSP tree acceleration method.

You can determine how many triangles are in your scene by selecting **Edit** > **Info Scene** from the Edit panel.

BSP or Grid?

Because there are two rendering-acceleration methods, you may ask yourself which is most beneficial to scene. Although both render a scene just as easily, the two methods have subtle differences that can affect the time of your render.

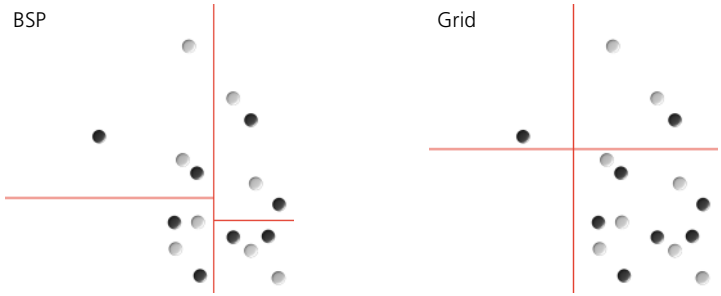


Here we use a very simple board game to illustrate ideal scenes for both the BSP and Grid acceleration methods. The board on the left has many objects close together in a small section of a larger scene; this object distribution would be ideal for BSP. On the right, the objects are evenly distributed throughout the scene; in this case, the Grid acceleration method would be ideal.

BSP

The BSP algorithm is aimed at accelerating "teapot-in-stadium" scenes; that is, a scene with a small and highly complex object in a very large environment. Grid acceleration does not handle these types of scenes very well, since it tries to cluster the scene in an isometric fashion. Each cell in a BSP tree has a different size, depending on the depth of the tree. The BSP tree algorithm often leads to faster rendering times, particularly in complex scenes with many reflections and refractions, as well as in scenes with motion blur.

Although BSP is a great all-around acceleration method, the algorithm does have some drawbacks. For instance, BSP does not work well in distributed rendering because the entire BSP tree must be present on all the computers that are rendering. Another drawback of the BSP algorithm is that it is nearly impossible to determine the tree's boundaries (triangles per leaf and depth) without building the tree first. Therefore, it is difficult to estimate and control the amount of memory required to store the acceleration data.



These illustrations show how the same scene is rendered using BSP acceleration (left) and Grid acceleration (right). The BSP method divides the scene in equal parts, whereas the Grid acceleration divides the scene into equal-sized boxes.

Grid

The Grid rendering algorithm provides faster preprocessing, especially on multiprocessor systems or a distributed-rendering network. Grid is also more memory hungry than the BSP tree algorithm. The rendering speed with Grid acceleration is comparable to BSP tree, but it depends more on the scene.

The Grid acceleration method partitions a scene into equal-sized boxes. Because the boxes are all the same size, cell stepping is very fast because it is mostly done with integer arithmetic. Unlike the BSP method, Grid is aimed at well-distributed objects of similar size. The "teapot-in-stadium" example above would prove chaotic for the Grid algorithm.

You can determine how many triangles are in your scene by selecting **Edit** > **Info Scene** from the Edit panel.

BSP Tree Acceleration Method

The BSP tree (binary space partitioning) method divides the scene into cubes to reduce the number of computations. It builds a hierarchical spatial data structure by recursively subdividing a bounding volume surrounding the entire scene. The resulting binary tree consists of branch nodes that correspond to a subdivision of a bounding volume into two smaller volumes, and leaf nodes that contain the geometric primitives (triangles).

You can set the BSP Max Depth to accelerate the renderer by limiting the level of space subdivision. The default for maximum depth is 50. In addition, you can adjust the rendering process more precisely to limit the number of triangles calculated in each cube by setting the Max size. The default is 10.

There is a trade-off between speed and memory consumption. Memory usage of the BSP tree algorithm depends on the parameters. An estimate of the amount of memory needed is not available before rendering begins. Large leaf sizes and small tree depths reduce memory usage but increase rendering time because larger leaves need to be searched. Increasing the tree depth also slightly increases the preprocessing time required for building the acceleration data structure.

You can fine-tune the rendering speed by experimenting with maximum leaf sizes of 1 to 10 and maximum tree depths of 20 to 27. When rendering with the verbose option (see *Using Pre- and Post-scripts* on page 73), the size of the largest leaf node (before rendering) and the number of candidate triangles per ray (after rendering) is reported. If these numbers are much larger than 10, you should try to build a deeper BSP tree by setting a larger BSP Maximum Depth such as 30 or higher. This can have a dramatic effect on rendering speed. When there are more triangles than can be stored in the BSP tree of the specified depth and leaf size, the maximum leaf size is ignored, so large leaves can result even if it is set to a small number.

Grid Acceleration Method

The grid algorithm subdivides on a regular grid of isolines in space.

Set the **Grid Size** to adjust the grid voxel size. Voxels are volume elements arranged on a fixed regular grid that define objects in space. The default grid size is 1. Larger values increase the number of voxels and shrink each voxel accordingly.

Rendering Faces

By default, both faces of an object are rendered; that is, those whose normals are facing the camera as well as those that are not. This could be useful for ribbons, flags, or pages in a book, where you need to see both sides of the object at the same time. If you like, you can render just the front face of an object. In this instance, the back faces are "culled" (ignored).

You have the option of rendering the front, back, or both faces.

To render faces

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass.
2. Select the Optimization property page and choose one of the **Face** options: Front, Back, or Both.

   For example, if you select Front, only the polygons facing the camera are rendered. You should select this option if the back faces of the object are not influential in the scene.

   To ensure the highest-quality image, you should select Both; you should also select Both if you are rendering a double-sided object.

Defining the Task Size

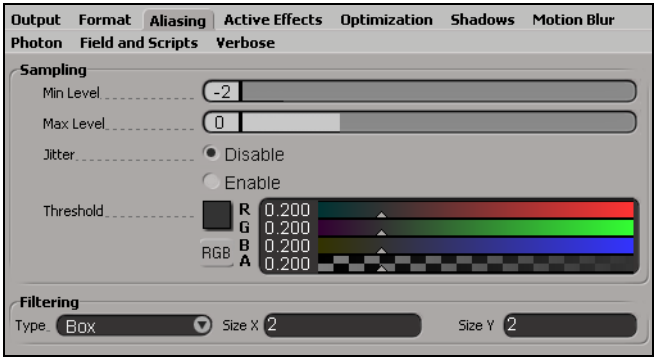
Task size allows you to define the size of the tile used when rendering an image. Usually, you would use a smaller task size when you are performing a distributed rendering that has a host and/or slaves operating at different speeds. Assigning a smaller task size avoids delaying the faster hosts/slaves by not making them wait for slow slaves that may be taking more time to render a large tile.

To define task size

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass.
2. Open the **Optimization** property page and select any one of the **Task Size** options: 4 × 4, 8 × 8, 16 × 16, 32 × 32, 64 × 64, or 128 × 128.

Controlling Aliasing

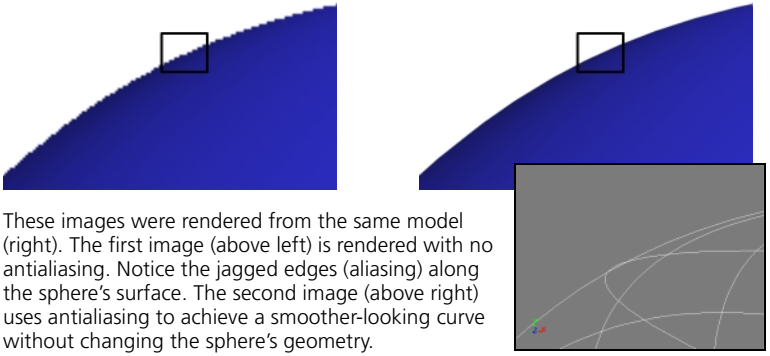
Aliasing, which depends more on contrast and image detail than image size, can occur at any resolution. Fortunately, aliasing can be visually curbed by antialiasing. Antialiasing is a method of smoothing out and sharpening rough or jagged edges of images to produce a more polished look. It uses a mathematical process that subsamples the pixel area, then averages the values of neighboring samples. As a result of these calculations, antialiasing can increase rendering time.



You can also 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.

No antialiasing

Antialiasing



These images were rendered from the same model (right). The first image (above left) is rendered with no antialiasing. Notice the jagged edges (aliasing) along the sphere's surface. The second image (above right) uses antialiasing to achieve a smoother-looking curve without changing the sphere's geometry.

Avoid adding antialiasing when you render a texture map that fills up the screen, since its edges aren't visible anyway. If the texture map itself is already antialiased, you don't need to add more antialiasing when you render.

To use antialiasing

1. 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 text box on the toolbar).

     or

   - Select a render pass in the explorer and right-click on the render pass and choose **Render Options** from the pop-up menu. The Rendering Options property editor is displayed.

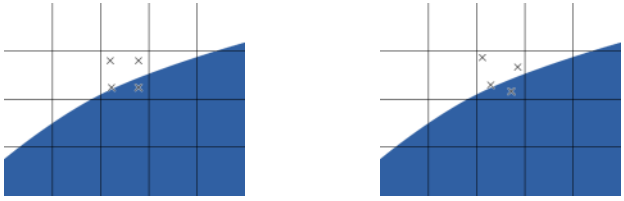
2. Open the Aliasing property page and set the **Min Level** and **Max Level** values for the Sampling options. These parameters describe the number of samples taken to compare surrounding color values that are averaged to define the color of a pixel. The default settings for **Min Level** and **Max Level** are –2 and 0, respectively.

   If the **Min Sample** is 0, the pixel is sampled at least once. The default minimum of –1 means that the picture is sampled once every four pixels (a 2-by-2 pixel square).

   With the default sampling, pixel 1 is compared with pixel 3. If the difference is greater than the specified contrast, the level of samples is increased. The new sampling compares pixel 1 with pixel 2, and a third pass takes place and each pixel is divided into four subpixels. If this still does not meet the supersampling threshold, aliasing appears in the result because the default maximum of 1 limits sampling to three levels. If this is the case and more antialiasing is required, change the **Min Level** and **Max Level** values to modify the limit.

   The maximum difference between the maximum and minimum sampling levels is 3. This limitation is meant to limit the amount of memory required and thus optimize performance.

3. Set the **Jitter** level using the slider. Jitter creates a random sampling that makes the antialiasing appear more natural and less interpolated.



Two methods of sampling. In the image on the left, we see an example of default sampling. On the right, some Jitter is added to the sampling. Notice how the sampling area is more random and the surface becomes smooth but less rounded.

4. Set the **Sampling Threshold** sliders. If the difference between neighboring samples is greater than this level, another level of sampling occurs.

   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 antialiasing.

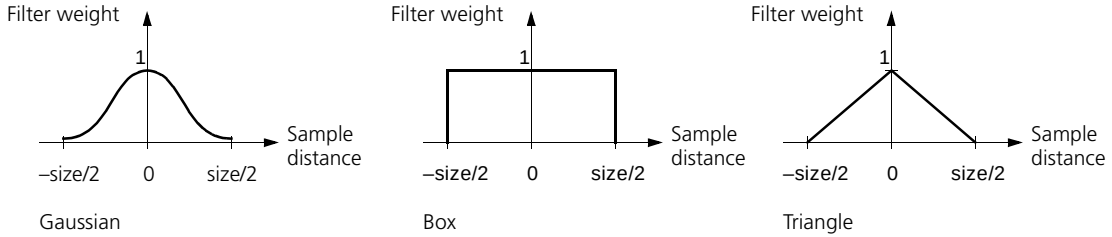
Filtering Processes

If you like, you can select a sampling filter and filter type to apply to your scene render. The sampling procedure can be complemented in post-processing to ensure even more antialiasing. Each of the three filter types (**Gaussian, Box**, and **Triangle**) process subsamples—surrounding and including the pixel that is being rendered—by using the height and width of the filter. Based on the value of the pixel, mental ray takes the average of every pixel and its surrounding pixels and removes aliasing artifacts.

The **X** and **Y** options for each of these filter types define the size of the filter to be applied.

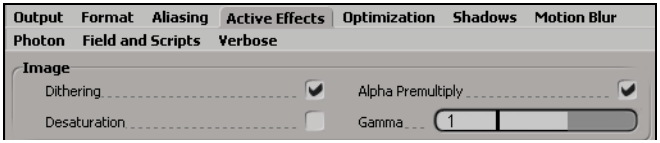
When you select a filter, it is applied as an algorithm defining a curve, peaking at the center of the pixel sampled.

- The **Gaussian** filter uses a sloped curve, weighting the sampling gently at the top of the peak and toward the edge of the sampled area.
- The **Box** filter sums up all the samples in the filter area with an equal weight.
- The **Triangle** filter uses a linear curve that affects the pixels so that the least filtering happens at the edges of the sampled area.



Applying an Image Process

It is possible to apply an image process or post-process to a scene during its render.

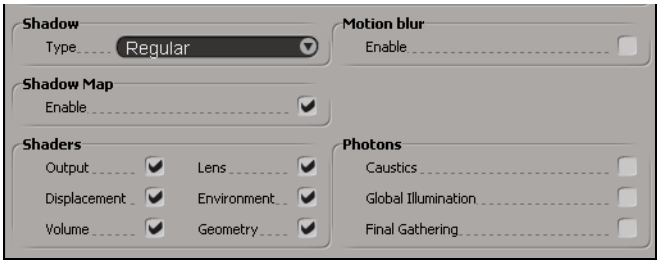


To apply an image process to a render

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass.
2. Open the Active Effects property page and set the following:
   - **Dithering** simulates colors in the image to create a smooth gradation between areas of light and dark.
   - **Desaturation** activates the desaturation color-clipping method that looks at each RGB color component to identify any pixels that may have a value greater than 1. If such values exist, the values of all other pixels are reduced proportionally.
   - **Alpha Premultiply** automatically multiplies an object's RGB values with their associated alpha value. This precomputation is helpful if you wish to composite the object image later on.
   - **Gamma** applies a gamma correction value to the render-region color. For most purposes, the default value of 1 is suitable.

Activating Effects

When rendering, you can activate only those effects and shaders that you require for the current render pass. You can choose to use different types of shaders per render pass as well as define which lighting effects are rendered.



Setting Shadow Options

When rendering a scene with shadows, you can determine what type of shadows are rendered. You can also determine if and how shadow maps are used.

To render shadows

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass.
2. Open the Active Effects property page, click on the **Shadow Type** text box to select what type of shadows are rendered.
   - **Regular** shadows performs a basic, simple rendering of the shadows. SOFTIMAGE|XSI determines how much light from a light source passes through a shadow-casting object. The shadow shaders are used in a random order.
   - **Sort** shadows is similar to regular, but it 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 used 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; for example, for a fur shader.
   - **None**—no shadows are rendered. This is usually used to speed up rendering time.
3. Click on **Enable Shadow Map** if you wish to use shadow map.

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. The 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 whether an object is in a shadow.

For fast previewing of scenes, shadow maps can be used by the scanline renderer to produce fast approximate renderings with shadows without using any raytracing. For more information about shadows, see *Creating Shadows* on page 144 of the *Shaders, Lights & Cameras* guide.

Setting Motion Blur Options

From the **Active Effects** page, you can define whether or not the output image renders motion blur by checking the **Enable Motion Blur** check box.

A full description of motion blur and its rendering options can be found in *Blurs, Flares & Other Effects* on page 185 of the *Shaders, Lights & Cameras* guide.

Setting Shader Options

You can deactivate shaders and other effects to allow for faster preview rendering. All shaders and effects that are deactivated are ignored.

To set shader options

1. Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options property editor for the current pass.
2. On the Active Effects property page, select or deselect the appropriate shaders and effects.

For more information on shaders and how to use them, see *Shader Basics* in Chapter 2 of the *Shaders, Lights & Cameras* guide.

Setting Photon Options

From the Active Effects page, you can define whether or not the output image renders caustics, global illumination, and final gathering effects by checking the **Enable Caustics**, **Global Illumination** or **Final Gathering** check boxes.

A full description of photon lights, both caustic and global illumination, and their rendering options can be found in *Chapter 6: Global Illumination & Caustics* of the *Shaders, Lights & Cameras* guide.

Field Rendering

Field rendering consists of rendering two alternating fields of horizontal scanlines (odd and even). When creating images for video, you can render to fields instead of frames. This reduces the strobing effect that results from fast-moving objects or from a large object moving very close to the camera.

Field rendering is a technique that allows smooth animations on interlaced video displays. To reduce flickering, video displays first display only every other scanline of the picture and then the remaining scanlines in a second sweep. Each sweep is called a *field*, and two fields make up one *frame*. Since sweeps occur at one half the frame rate, animated objects may have moved between sweeps. Not taking this into account results in rough animations.

In general, use field rendering if you are rendering to video and you want very smooth motion in animation without strobing, or when you need to match material that was shot with a video camera. Field rendering creates a more "electronic" look that resembles video camera shots, and it will make motion blur look like material shot on film.

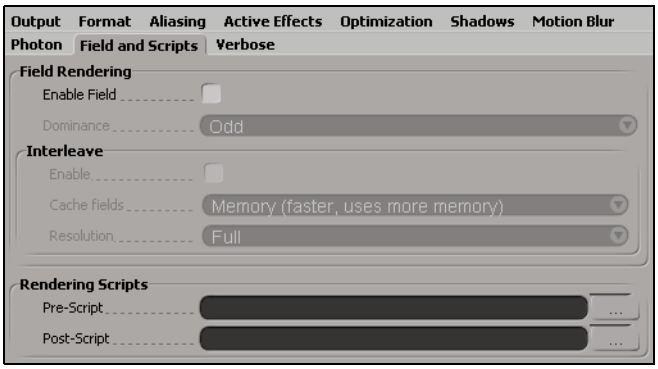
Instead of rendering double frames or post processing and interleaving the frames, use a field rendering with no interleave at high resolution. Then, post-process the images and interleave the frames.

When using field rendering you should take into consideration the following:

- Field-rendered objects have half the vertical resolution per frame than that of frame-rendered objects when using the medium-resolution option. This may create some antialiasing artifacts, particularly in objects that move slowly in the vertical.
- Interleaved field-rendered material is more difficult to use in post processing because some filter types can destroy the interleaved image. This is particularly true of blur filters applied to the image in post production.

By default, mental ray renders full frames, resulting in a non-interlaced output picture. If you set the dominant field to even, every consecutive pair of rendered frames is combined such that the first frame contributes the even scanlines (counted from the top) and the second frame contributes the odd scanlines. This is reversed if you set the dominant field to odd.

The even field of one frame is rendered, then the odd field of the next frame is rendered. This effectively doubles the frame rate. Because of the dependence on frame numbers, scenes should start with an odd frame and end with an even frame, meaning that there must be an even number of frames.



To set up a pass for field rendering

1. 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 text box on the toolbar).
2. On the Field and Scripts property page, select the **Enable Field** and the **Enable Interleave** options from the Field Rendering section.
3. In the **Dominant Field** group, select one of the following:
   - **Even** renders to fields using even dominance. With mental ray, this means that even frame numbers contain the even fields. This is the dominance used by the PAL video standard.
   - **Odd** renders to fields using odd dominance. With mental ray, this means that odd frame numbers contain the odd fields. This is the dominance used by the NTSC video standard.
4. Open the **Output** property page and select a **Format** that is appropriate for field rendering: Video NTSC, Video PAL, or Video HDTV.

Fields are output as .pic files and are named `name.frame.field.pic`

Interleaving the Fields

You can immediately merge the field-rendered pictures together by interleaving their fields. A built-in interleave function is used to merge the fields. Before interleaving, however, the first two "intermediate" images must be rendered (one for each field). These "intermediate" images can then be placed into memory or onto disk. Once the interleave is finished, the memory/disk is cleaned. The resulting image is always saved onto disk at the determined **Output** path.

To interleave fields

1. Select the **Enable Interleave** option.
2. Open the **Cache Fields** menu and select one of the following:
   - **Memory—**saves the rendered fields into memory. Once the final image is rendered, the memory is cleared. This option is faster but uses more memory than Disk.
   - **Disk**—interleaves to rendered fields and saves them to the **Output** path you specified on the Output property page. This option is slower than Memory but uses less memory.

If you do not use the **Interleave** option, you have to combine your field-rendered files outside of the interface or by using a post-frame script.

- You can use the **interleave** standalone to merge your rendered pictures together (see *Appendix: Standalones* on page 223 of the *Fundamentals* guide).
- You can use an interleave post-script to combine the rendered frames.

Selecting a Resolution

When field rendering, you must determine the quality of your image by selecting either:

- **Full resolution**

  or

- **Half Resolution—**half the number of scanlines needed (half the scanlines for field 1 and half for field 2). Although this option is faster, it can create some antialiasing artifacts.

Using Pre- and Post-scripts

Pre- and post-scripts allow you to run extra processes before or after each frame is rendered. These scripts can be used for interleaving fields, dumping to Accom or Abekas, moving, renaming files, and any other process you need to perform. You can use the scripts provided with SOFTIMAGE|XSI or you can write your own. On Windows NT you write batch files, and on IRIX you write C-Shell scripts

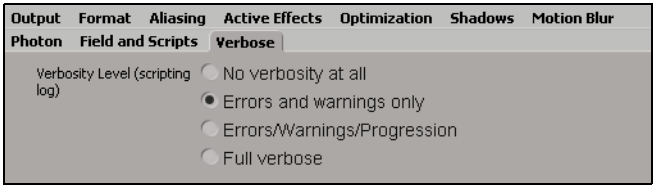
After you have created a pre- or post-script file, you can access it from within SOFTIMAGE|XSI.

To access a pre- or post-scrip file

1. 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 text box on the toolbar).
2. Open the Field and Scripts property pages and select the pre- or post-script you want to run by clicking the browser icon (...).

Receiving Messages

You can control the level of message verbosity directly from the Render Options property page.



To choose a verbosity level

(You can choose from four.)

1. 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 text box on the toolbar).
2. Open the Logging Messages property page.
3. Select a level of verbosity.
   - **No Verbosity at all**: Provides no feedback during the render.
   - **Errors and Warnings Only**: Only warns of errors that may halt the render.
   - **Errors/Warnings and Progression**: Warns of errors that may halt the render and the progression of the rendering itself.
   - **Full Verbose**: Provides full messaging, progress and warnings.

General Memory Requirements

A general rule for calculating memory requirements is to allow one MB of memory for every 1000 surface triangles. For every 512 × 512 texture, you would need an additional 1 MB of memory.

You can determine how many triangles are in your scene by selecting **Edit** > **Info Scene** from the Edit panel.

These rules assume that you're using raytracing features (reflection, shadows, refraction, etc.). If you are not using raytracing, you can render much larger scenes.

For information on how to optimize the rendering of your scene, see *Optimizing Rendering Performance* on page 76.

Optimizing Rendering Performance

The rendering process usually involves a trade-off between rendering speed and image quality. This section provides some useful tips to consider when rendering. Some suggestions can improve speed dramatically without sacrificing image quality, while others may involve a small but acceptable reduction of quality.

In addition to the following suggestions, you can obtain important information for performance tuning of mental ray rendering software from the diagnostic messages.

Modeling

Before you add a scene to a project, you can improve rendering speed by considering your options:

- Fewer objects in a scene take less time.
- The more geometric detail an object has, the longer it takes to render. Try to substitute some of an object's geometric detail with a texture map. From a distance, the difference may not be distinguishable.
- You can simplify the calculations required for shadows, reflections, and refractions by adding invisible objects to the scene. Suppose you have an object that is generally spherical but has a great deal of surface detail. Set its object properties so that it is visible to primary rays and secondary rays but does not cast a shadow. In the same position, add a second, simpler object that casts a shadow but is otherwise invisible. The end result is an image that appears correct but for which the shadow computations are much less expensive. A similar technique can be employed so that secondary (reflected and refracted) rays see a much simpler object.

Clipping Planes

Another way to reduce image complexity is to define clipping planes to set a maximum and minimum distance for the camera to render. 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 objects that are complex but very far away, you may want to set the far clipping plane so that such objects are not rendered. These options are set in the Camera property editor.

Rendering Faces of Objects

You have the option of rendering only the front or back faces of objects, which is faster than rendering both faces. This feature is set with the Face options on the Quality/Speed page in the Rendering Options property editor (see *Rendering Faces* on page 63). Because it takes less time to render one side instead of two, you should make sure that the normals of all visible surfaces point toward the camera.

Render Passes and Compositing

You can save rendering time by setting up render passes properly. For example, if you have a separate highlight pass and want to change the highlights, you only need to re-render that pass. You can then composite the new highlight pass with the other old passes. For information on setting render passes, see *Passes & Partitions* on page 23.

Compositing is a practical way of optimizing rendering time. By rendering constant elements of a scene (such as the background, shadows, highlights, etc.) separately from the other elements and then compositing them together later, you save lots of rendering time. The number of passes you can composite is unlimited.

Image Resolution and Subregions

If you are testing settings, you can decrease the resolution to obtain an image more quickly. Another possibility for test images is to render only a subregion. These options are set on the General page of the Rendering Options property editor (see *Setting Output Rendering Options (Global Only)* on page 53).

Lights

Rendering time is also directly affected by the number and type of lights. Use the minimum number of lights necessary in each pass, and use selective lights whenever possible to reduce the amount of calculations per light. Avoid using area lights unless you require soft shadows.

You can toggle off all of a scene's area lights before rendering by typing `ToggleAreaLights` in the Scripting text box.

Shadows

Shadows are time consuming to render, so remember to select None as the Shadow Type on the Active Effects page of the Rendering Options property editor for passes that do not require shadows.

In addition, shadows on distant objects do not need to be as accurate as those on objects that are close to the camera. For example, you could use a selective area light to generate soft shadows for close objects, and use a selective point light in the same location to generate simpler shadows for distant objects. Alternatively, you could use two separate render passes: one with raytraced shadows for close objects and one with shadow-mapped shadows for distant objects. For more information about shadows, see *Creating Shadows* in Chapter 5 of the *Shaders, Lights & Cameras* guide.

For fast previewing of scenes, shadow maps can be used by the scanline renderer to produce fast approximate renderings with shadows without using any raytracing.

Global Illumination

Global-illumination simulation can be very time consuming to render. Unless you require effects like color bleeding from diffuse inter-reflection and/or caustic effects, you should consider using ambient light or final gathering to imitate a global-illumination effect. For more information, see *Chapter 6: Global Illumination & Caustics* of the *Shaders, Lights & Cameras* guide.

Caustic Effects

Caustics are a subset of the complete set of effects achieved by global-illumination simulation. If you only want to see caustics, activate only the **Caustics** option for the render pass where they are used (**do not** activate Global Illumination as well).

If you need to use caustics, you can specify a smaller amount of photons emitted by each caustic light source, or you can use a pre-computed photon map.

Shaders

The total rendering time is affected by the number and complexity of shaders. The amount of calculations required by individual shaders can vary greatly. In general, volume shaders are very time consuming whereas surface shaders are quick.

Experiment with different shaders to see whether you can get a similar effect that renders more quickly. When rendering, you can activate only those shaders that you require for the current render pass on the Active Effects page of the Rendering Options property editor. For more information about shaders in general, see *Chapter 2: Shader Basics* of the *Shaders, Lights & Cameras* guide.

Ray-Depth

Specific surface-shader properties such as reflectivity, transparency, and refraction can increase rendering time. To render these properties effectively, you usually have to increase the ray-depth setting for the raytracing rendering algorithm, which slows down the rendering. To save time for reflections, you can apply a reflection map without raytracing to simulate reflectivity. You can also apply reflectivity or transparency to only one or two objects in a scene rather than to all of them (check for any unnecessary surface shader properties on each object).

Memory-Mapped Textures

Rendering of textures can be made quicker with the use of memory-mapped textures (UNIX only). Memory-mapped textures are not loaded into memory but are accessed directly from the disk whenever the shader needs it. Since it isn't possible to toggle this option, SOFTIMAGE|XSI recognizes a memory-mappable texture and automatically maps it. Only the map image-file format (with the **.map** extension) can be mapped. Before a texture can be memory-mapped, a few steps have to be taken:

- The texture must be converted to **.map** format. The scene file must reference this texture.
- Memory-mapped textures are considered local by mental ray, as though the local keyword was used in the scene file. So, if the scene is rendered on multiple hosts, each one accesses the given path rather than transfer the scene across the network.
- The texture should not be on an NFS-mounted file system. Although it may seem faster to have the texture on all the hosts, the network transfers make memory-mapping slower than regular texturing.

- Memory-mapping is ideal for very large textures containing dozens of megabytes that are sampled infrequently, since most of the texture file isn't loaded into memory.

If the texture and the scene are too large to fit into the physical memory, then loading a texture is equivalent to loading the file into memory, decompressing it, and copying it out to swap. From that point on, accessing a texture means accessing the swap. On the other hand, a frequently accessed texture and scene are smaller and fit into memory. Memory-mapped textures are slower than regular textures because the swap isn't used.

Textures

Rendering can be slowed by the amount of textures in a scene. Each 2D texture file must be loaded into memory, and this can result in memory swapping (transferring data between RAM and disk). Large image files used as textures require more memory than small files, and lots of different textures in the same frame require memory for each one. You can sometimes reduce the size of texture files for objects that are far from the camera without losing image quality. For more information about 2D textures, see *Chapter 3: Material & Texture Basics* of the *Shaders, Lights & Cameras* guide.

3D textures are procedural and must be calculated from points in 3D space. This requires more computation time than that for 2D textures. For more information about 2D and 3D textures, see *Material & Texture Basics* on page 59 of the *Shaders, Lights & Cameras* guide.

Antialiasing

Antialiasing generally increases the rendering time because it performs a larger number of calculations. However, you can reduce the number of samples per pixel and still obtain a satisfactory result. You can also increase the sampling contrast: this requires a careful examination of the resulting images to determine an acceptable level, but it can make a significant difference in rendering time, especially if the number of samples is high. See *Controlling Aliasing* on page 64 for more information on these options.

Motion Blur

You can set any of the following options in the Rendering Options property editor of a render pass to optimize the render of motion blur:

- Reduce the shutter speed to speed up motion blur calculations.
- Select **Estimated Motion Blur** from the motion blur property page to render a "quick and dirty" motion blur effect.
- Select the **Scanline** option to render a faster scanline motion blur than is done with raytracing. Make sure to deselect the **Raytracing** option.
- Use 2D motion blur instead of raytraced or scanline motion blur algorithms. 2D motion blur is handled as an output shader and lens shader and applies the motion blur effect after the image is rendered. When you use 2D motion blur, the output shader is automatically attached to the current pass. You can also use the 2D motion blur standalone tool. For more information, see *Standalones* on page 223 of the *Fundamentals* guide.

Rendering Methods

There are two separate rendering methods: raytracing and scanline. Raytracing is more realistic than scanline, but it takes longer. To save time, you may want to use only the scanline renderer. To do this, select the **Scanline** option on the Optimization page of the Rendering Options property editor and switch off Raytracing. For more information about rendering methods, see *Selecting a Rendering Method* on page 57.

Raytracing Settings

If you use the raytracing rendering method, there are ways to optimize its use:

Acceleration Methods

There are two rendering-acceleration methods that work with raytracing: binary space partitioning (BSP) and grid acceleration. They are set on the Optimization page of the Rendering Options property editor. Each has its own unique advantages (see *Acceleration Methods for Raytracing* on page 60 for more information) depending on your scene.

Ray Depth

With raytracing, the rendering time is closely related to the total number of rays cast. Physically, a ray may be reflected or refracted an infinite number of times, but you usually want to place a low maximum limit on this number. You can set limits on reflections and refractions separately, as well as on their sum. These options are set on the Optimize page of the Rendering Options property editor (see *Raytracing* on page 58 for more information).

Pyramid Mapping ("MipMapping")

Pyramid mapping performs a type of 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. For more information on how to set pyramid mapping options, see *Pyramid Mapping* in Chapter 4 of the *Shaders, Lights & Cameras* guide.

Adjusting an Object's Surface Approximation

Depending on your method of surface approximation, you can reduce the number of triangles in an object but still obtain a quality render. Reducing triangles decreases the amount of time necessary to transmit the information required to render a frame across a network, and it also decreases the amount of memory and swap space required to hold that information.

You can determine how many triangles are in your scene by selecting **Edit** > **Info Scene** from the Edit panel.

For more information on setting an object's surface approximation, see *Setting an Object's Surface Approximation* on page 81.

Setting an Object's Surface Approximation

Surface approximation specifies how polygons, NURBS surfaces, and curves should be tessellated (divided into triangles at rendering time). The various methods of approximation let you reduce the number of triangles in the geometry of an object and still render a very smooth surface.

What You Can Approximate

On the Geometric Approximation Parameters property page, you can change or adjust the surface approximation of an object's various surface types. For each type of approximation, you are given the choice of Parametric approximation or Length, Distance, Angle (L/D/A) approximation. Your choices are listed as follows:

- **Hardware Display**—Does not affect the geometry of the object. Used for viewport display purposes only.
- **Surface**—Affects the entire geometry of an object. Surface approximation is often used in conjunction with displacement approximation.
- **Polygon Mesh**—Defines how a polygon object's surface is approximated. You can specify the discontinuity angle or use the (default) automatic-discontinuity option.
- **Surface Trim**—Defines how the trimmed edge of an object are approximated. For example, the rim of a hole cut into a sphere.
- **Displacement**—Approximates areas that have had a displacement assigned to them through a texture. Displacement approximation is directly influenced by the surface-approximation values. Note that if you have a render region defined, increasing this value dramatically increases rendering time.

Where to Use Which Approximation

Although there is no hard and fast rule on which approximation to use for a given object, here are a few rules of thumb you can follow to optimize a rendered scene's quality:

- Parametric approximation should be used if you have a largely flat-based surface as well as some displacement.
- Length approximation is useful—especially when combined with view dependency—because it lets you control the approximate number of triangles per pixel.
- Distance approximation is often used on objects that have a gentle curve, since they don't usually have two neighboring triangles with a large normal distance.
- Angle approximation is best used on objects that have localized areas with high curvature, such as a cube with rounded edges.

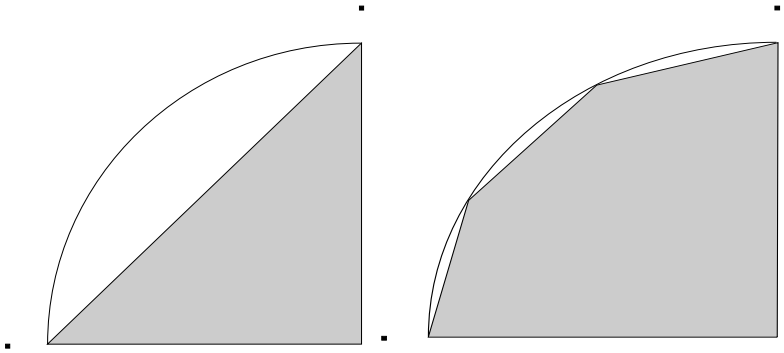
To set the surface approximation for an object

1. Select an object in the viewport whose surface you want to edit.
2. Click the **Property** button on the Selection panel to display the selected object's node.
3. Click the icon beside the Geom Approximation Parameters node to open its property editor.
4. On the Surface page, select a type of surface approximation: **Parametric** or **LDA** (see the next sections for more information).

   Depending on the surface approximation you choose, SOFTIMAGE|XSI tries to create the perfect curve between two points.

Parametric Approximation

The Parametric method of surface approximation defines the U and V steps of the selected object as seen by the geometry. This type of approximation uses non-adaptive sampling between parametric intervals.



The surface on the left uses a Step Value of 1, while the surface on the right uses a Step Value of 3. Notice how using more steps determines how smooth an object's surface looks.

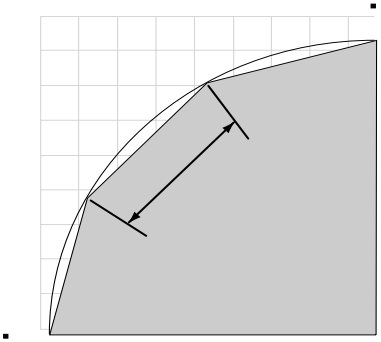
Changing these values changes the number of steps in the object's surface. These parameters remain the same throughout the animation, and if the camera gets very close to the object the steps might be seen.

The Displacement property page uses a Step parameter instead of defining both the U and V steps of a surface's geometry. The Subdivision Step parameter controls the U step only.

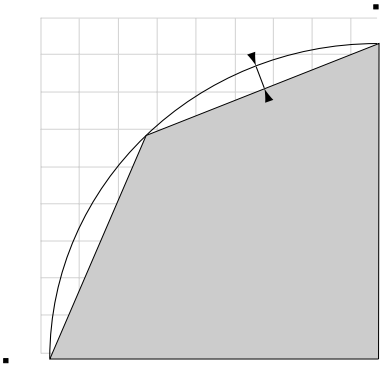
LDA (Length/Distance/Angle) Approximation

The LDA approximation method is a curvature-dependent approximation based on subdivisions by length, distance, and angle. This type of approximation uses adaptive sampling between parametric intervals.

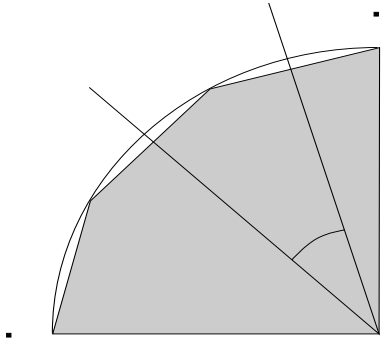
- **Length** subdivides the surface or curve so that no edge length of the tessellation exceeds the **Length** parameter. Length is given as a distance in camera space, or as a fraction of a pixel diagonal in raster space. Small values such as 1 are recommended. Length is affected by the **Min** and **Max Subdivision** parameters.



- **Distance** specifies the maximum distance between the tessellation and the actual curve or surface. The value is a distance in camera space, or a fraction of a pixel diagonal in raster space. As a starting point, a small distance such as 0.1 is recommended. Distance is affected by the **Min** and **Max Subdivision** parameters.



- **Angle** specifies the maximum angle in degrees between normals of adjacent tiles of a displaced polygon or the tessellation of a surface or its displacement or between tangents of adjacent segments of the curve approximation. Large angles such as 45 degrees are recommended. Angle is affected by the **Min** and **Max Subdivision** parameters.



Use the **View Dependent** check box to go from camera space to raster space.

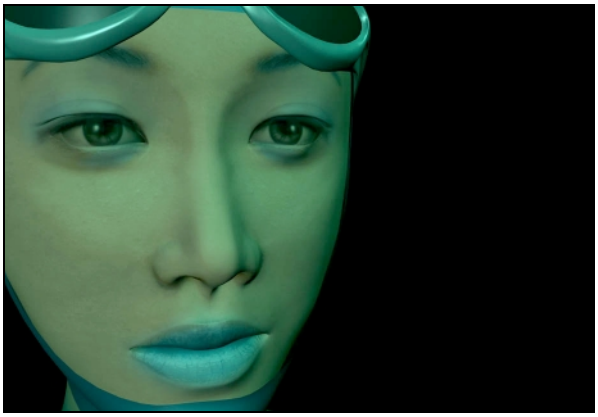
Setting the subdivision limits

To prevent massive amounts of subdividing, a limit can be assigned to each selected object, which helps to optimize the quality of surfaces with areas of different curvatures.

The **Recursive steps minimum** and **Recursive steps maximum** parameters specify the minimum and maximum number of recursion levels of the adaptive subdivision. The **Minimum** parameter specifies the minimal triangulation. Edges are further subdivided until they satisfy the given criterion or the **Maximum** subdivision level is reached. The defaults are 1 and 3. You can usually achieve good results with a maximum of three subdivisions (default).

Chapter 4 Rendering

Rendering is the process of using the information in a 3D scene to compute a 2D image. When you preview a scene or display it in a viewport, it is implicitly rendered to the computer screen; however, when you explicitly render your scene, the frames are rendered as image files on disk.



Whenever you render, SOFTIMAGE|XSI uses the options that you have set for the selected render pass. For more information about render passes, see *Chapter 2: Passes & Partitions* on page 23. Rendering options set for render passes (global) are described in *Chapter 3: Rendering Options* on page 49.

Single vs. Multiple Computers

When it's time to render, you will have to decide whether you will use a single computer or render your scene over several computers on a network. The most obvious advantage of using several computers is speed. But assigning more than one computer to render a scene sometimes involves a certain amount of preparation on each rendering computer.

You can render locally using your own computer, as described on page 88. For more information on how to render across a network see *Setting Output Rendering Options (Global Only)* on page 53.

Rendering on a Single Computer

Rendering locally means using your own computer to calculate images the other computers on the network do not participate in the rendering process.

When you render locally, you have a choice of rendering the current frame or the entire scene of frames in a render pass. You can also render the current frame to memory and preview it before saving, or you can render it directly to disk.

To preview the current frame in a separate window

1. First, select the pass you want to render from the **Pass** list in the Render toolbar.

   If you have not defined any render passes, the default is a full beauty pass that includes everything in your scene. For more information about passes in general, see *Chapter 2: Passes & Partitions* on page 23.

2. Next, do one of the following:

   - Choose the **Render** > **Preview** > **All Layers** from the Render toolbar. You can change the current frame using the timeline.

     or

   - To save the image, choose **File** > **Save As** from the menu bar of the preview window, then use the browser to specify a directory and file name.

To render the current frame directly to disk

Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options window. Render a single frame by entering the same start and end frame in the **Start Frame** and **End Frame** text boxes. You can also define a path and file name as well as an image format for your rendered frame. Click the **Render** button to render your frame and save it to disk. For more information, see *Specifying the Format Options (Global Only)* on page 55.

To render the entire scene of frames in a render pass

Choose **Render** > **Render** > **Options** from the Render toolbar to open the Rendering Options window, then choose which frames you wish to render by defining the first and last frames to be rendered in the **Start Frame** and **End Frame** text boxes. You can also define a path and file name as well as an image format for your scene. Click on the **Render** button to render your frame and save it to disk using the defined Output and Format options. For more information, see *Setting Output Rendering Options (Global Only)* on page 53.

Distributed Rendering

With distributed rendering, you can share the job of rendering among a group of computers on a network. This reduces the time needed to render frames, and it takes advantage of any unused processing power on the network.

When you start a render task, your computer instructs the others in the group (defined in the **.ray2hosts** file) to start mental ray rendering processes, and it then sends a separate portion of a frame to each remote process. It collects and assembles the rendered "tiles" when they are complete, and then sends more portions of frames until the task is done.

The host that reads and translates the scene is called the Master host. This computer is responsible for connecting to all other hosts, called Slave hosts.

For more information on verifying a distributed rendering network's setup, verifying computer availability, and defining a rayhost, see *Setting Up Distributed Rendering* on page 111

Launching Distributed Renders

There are three ways to launch a distributed render:

- ray2.exe command line
- Interactively from the interface
- Command-line scripting

For testing purposes, the ray2.exe option is convenient; it allows you to have a verbose output in a command prompt, therefore allowing you to "see" what is going on and recognize any point of failure. The verbose output is not recommended for actual production renders as it slows down the process.

For a full list of command-line options, type `help` in a command prompt or see *Chapter 3: Rendering Options* on page 49.

Environment Variables

The following three environment variables can help control the transmission of data to remote hosts (set these in the System Environment variable):

- **MI_RAY_LOAD_NUM**—specifies the maximum number of servers concurrently loaded. Default is 10.
- **MI_RAY_LOAD_TIME**—specifies the time-out period for loading a server, in seconds. Default is 900.

Skip Frame Render Mode

Using the Skip Frame mode will make each computer "reserve" a frame and warn the other computers that it will render it so they can skip the frame and proceed to the next one.

Rendering with Scripts

In addition to using the Render button, you can batch render using scripts. You can use the RenderPass command to render the current pass with its current rendering options, or you can use the `RenderAllPasses` command. Use **GetValue**() and **SetValue** to work with rendering options.

A good example of a script that includes rendering commands is Rendering.vbs in the DSScripts subdirectory. You can open this file in the script editor to see how it works. To run this script, you must drag it to a toolbar and click **Parse** in the Add Scripting Command dialog box, because the script contains no global code.

The Rendering.vbs is a core system file. Modifying this file may corrupt your installation. Do not edit and save the file.

You can also create a script to render in batch mode from the command line, without invoking the interface. Remember that you must provide an argument to specify which scene file to render, because scripts run from the command line cannot require any interactivity such as selecting a scene.

For information about scripting and the Script Editor in general, see *Chapter 7: Commands & Scripts* on page 169 of the *Fundamentals* guide.

Preparing a Script for Rendering

So that your scene can be opened for rendering, it must be specified within the script file. You can add the following lines using the Script Editor.

The proper syntax to include as the first line(s) of your script is one of the following:

```
OpenScene "PATH/SCENE.SCN"
```

or

```
ImportFromSI3D "PATH/SCENE.DSC" or "PATH/MODEL.HRC"
```

The last line to include in your script—to make it a render—is:

```
renderpass
```

Batch Rendering with a Script

The batch renderer is a renderer you start from a command-prompt window. You can render MI format files as well as SOFTIMAGE|XSI scene files using scripts.

Launching a Render with a Script

The basic usage to batch render with a script is as follows:

```
xsi -script <scriptfilename> [ -lang <langname> ] [ -main <entrypoint> ]
[-args -<arg1name> <arg1value> -<arg2name> <arg2value> ... ]
```

The following is an example of how you would start the batch renderer:

```
xsi -script batchrender.vbs -function main -args -scene myScene.scn -pass
Default_Pass -startf 1 -endf 10 -step 1 -output c:\temp\test
```

Batch Render Options

This section describes the options available for the SOFTIMAGE|XSI standalone batch renderer. Use these options to render with a script and define a list of arguments.

Option	Function
-script <scriptname>	Script to be executed
-lang <language>	Desired scripting language (optional)
-main <function>	Function entry point (optional if no arguments are specified). If this is not specified, global code is executed.
-args <argument list>	List of function arguments. All flags given after '-args' will be passed to the script function.

If the `-lang` argument is missing, the scripting language is deduced from the script file's extension.

Example of a Render Batch VBS Script

The following example is a full VBS script used to render batch a SOFTIMAGE|XSI scene.

```
'START batchrender.vbs
function main (scene, pass, startf, endf, step, output)
if Typename(pass) <> "String" and TypeName(scene) <> "String" then
   Msgbox "Sorry No Pass Name Define or scene to open"
   exit Function
   else
      OpenScene scene
      if IsNumeric(startf) then
      SetValue "Passes."& pass &".RenderOptions.StartFrame", startf
      end if
      if IsNumeric(endf) then
         SetValue "Passes."& pass &".RenderOptions.EndFrame", endf
               end if
      if IsNumeric(step) then
         SetValue "Passes."& pass &".RenderOptions.Step", step
      end if
      if TypeName(output) <> "Error" then
      SetValue "Passes."& pass &".RenderOptions.ImageFileName", output
      end if
   end if
   Msgbox "Starting Rendering...."
   RenderPass "Passes."& pass
   Msgbox "Rendering Done!"
end function
'END batchrender.vbs
```

Syntax Conventions

These are the typographical conventions used to describe the usage (syntax) of the batch renderer.

- A dash (-) followed by one or more letters (as in -z) indicates an option name. For example, -z usually means zoom, -s usually means sequence, -v usually means verbose mode, etc. The available options are listed and explained in the description of each standalone utility.
- Angle brackets(< >) indicate a parameter for which you must substitute a name or value. When entering a parameter, do not type the angle brackets. For example, <filename> means that you must type the name of the file you want to use. Unless otherwise specified, do not include the extension (such as .pic, .lin, .hrc, .mi, etc.) when specifying file names.

- Commands and parameters not enclosed in angle brackets must be typed exactly as they appear.
- Strings are quoted within double quotes; this includes all names. The double quotes protect names from interpretation by the shell or command prompt window.
- Brackets ([ ]) indicate an option along with its input. Do not type the brackets. For example, [-z <fact>] means that, if you use the -z option (zoom), you must also specify its <fact> (zooming factor).
- A vertical bar (|) separates exclusive options. For example, the usage for -face includes front|back|both. This means that you must specify one of these three options for defining which face of the object to be rendered. When specifying an option, do not type the vertical bar.
- Braces ({ }) enclose a group of possible values related to an option. When entering parameters, do not type the braces.
- The ellipsis (...) indicates that the option or group of options can be repeated.
- Options and inputs appearing on the same command line must be separated by spaces.
- Commands should be entered as a single line in a shell or command prompt window, or in a C-shell file.

Ray2.exe Rendering

Although many of the Ray2.exe commands are available in the interface, you may wish to use the ray2.exe standalone command line tool to manually edit exported .mi files. Ray2.exe lets you read and render an .mi file and define extra shaders, create objects, swap textures, or perform other special effects.

When started from a shell command line, mental ray accepts a large number of options. When an option is given on the command line, it overrides the corresponding command or statement in the scene file, which in turn overrides the defaults. The defaults for certain options given in the option list below apply only if the corresponding command or statement is not present in the scene file.

Ray2.exe Options Overview

The following table provides an overview of the command-line options available through the ray2.exe command.

Command	Function
-acceleration S	r[ay_classification], b[sp], or g[rid]
-aperture N	width of aperture
-aspect N	aspect ratio
-bsp_depth N	maximum depth of BSP tree
-bsp_memory N	maximum memory allowed for BSP
-bsp_size N	maximum number of primitives in a BSP leaf
-caustic B	toggle caustic computation
-caustic_accuracy N [N]	caustic estimation parameters
-clip N N	hither and yon clipping planes
-code F ... --	compile and dynamically link C files
-colorclip S	color clipping mode: a[lpha], rg[b] (default), or ra[w]
-contrast R G B	threshold for adaptive oversampling
-core B	allow core dumps, for debugging
-cut_windows N	render frame as N*N sub-frames
-c_compiler F	C compiler for dynamic linking
-c_flags S	compiler flags for dynamic linking
-c_linker F	linker command for dynamic linking
-desaturate B	toggle desaturation color clipping
-diagnostic option arg	enable diagnostics, options: grid, photon, samples

Command	Function
-displace B	enable/disable all displace shaders (default on)
-dither B	toggle intensity dithering
-echo F options --	echo scene DAG to file, options: ascii, approx, source, textures, norendercommand, render
-face S	faces to render: f[ront], ba[ck], or bo[th]
-file_type S	file type (pic, rla, tiff etc)
-file_name F	output picture file name
-filter S [X [Y]] b[ox], t[riangle], g[auss], m[itchell] or [l]anczos	filter; and optional filter size in pixels units
-finalgather B	toggle final gathering
-finalgather_accuracy N [N [N]]	final gathering number of rays, max and min reuse radius
-focal N	focal length N, or orthographic if N=i[nfinite]
-gamma N	gamma exponent, default 1.0
-geometry B	enable/disable all geometry shaders (default on)
-globillum B	toggle global illumination computation
-globillum_accuracy N [N]	global illumination estimation parameters
-grid_size N	relative grid size, default 1.0
-?	print help information
-h	print help information
-help	print help information
-hosts S ... --	servers to use, use host[:port] or 'host[:port] options'
-imgpipe N	send preview image to file descriptor N
-I P	same as -include_path
-include_path P	replace standard include path for $include <file>
-jitter N	jitter image samples
-L S	library search path
-ld_libs S	libraries for dynamic linking
-ld_path S	library search path
-lens B	enable/disable all lens shaders (default on)

Command	Function
-link F ... --	dynamically link .OBJ or .DLL files
-merge B	enable/disable surface edge merging where specified (default on)
-message M S,S,... --	enable messages S in module M (all modules if M is "all")
-o F	same as -file_name
-offset X Y	offsets rendered area by X,Y pixels
-output B	enable/disable all output shaders (default on)
-photonmap_file F	use photonmap file F
-photonvol_accuracy N [N]	volume global illumination estimation parameters
-photon_depth N [N [N]]	max photon depth for reflect, refract, total
-premultiply B	premultiply RGB by alpha, default on
-raycl_memory N	ray classification: megabytes for ray space subdivision
-raycl_subdivision N N	ray classification: adjust # of eye and shadow subdivisions
-render N1 N2 [N3]	only render frames N1...N2, increment N3 (first frame is 1; default 1 <last> 1)
-resolution X Y	image resolution X by Y pixels
-samplelock B	sample lock or flicker
-samples Nmin Nmax	min,max sampling level, 0 means 1 sample/pixel
-scanline S	scanline rendering mode: on (software), off, op[engl]
-server	ignored for compatibility
-shadow S	shadow casting: on, off, so[rt], or se[gment]
-shadowmap options --	modify shadow map options: on, off, opengl, motion, nomotion, rebuild, reuse, only

Ray2.exe Options

The following section describes the options listed above in more detail.

Options can be abbreviated as long as the given substring is unambiguous. mental ray checks for ambiguities and prints an error message listing the choices. For example, -resolution can be abbreviated as -res. For frequently used options such as -verbose and -filename, short forms are available. The available options are:

-acceleration bsp

Selects the binary space partitioning (BSP) rendering algorithm. This algorithm is often, but not always, the fastest. It is controlled by the -bsp_size and -bsp_depth options.

-acceleration grid

Selects the grid rendering algorithm. It provides faster preprocessing, especially on multiprocessor systems. Memory usage is more conservative than with the BSP algorithm. Speed is comparable to BSP but more scene-dependent. It is controlled by the -grid option.

-acceleration raycl

Selects the ray-classification rendering algorithm. This algorithm is recommended for very large scenes. It operates with a constant acceleration memory size, controlled by the -subdivision_memory and -subdivision options.

-aperture aperture

The aperture is the width of the viewing plane. The height of the viewing plane is aperture divided by aspect. Together with the focal and aspect viewdefs, aperture defines the lens of the camera.

-aspect aspect

This is the aspect ratio of the camera. The default is 1.33. In camera space, aperture is the width of the viewing plane and aperture divided by aspect is the height. The viewing plane is divided into pixels as specified by the resolution viewdef, so the aspect will result in nonsquare pixels if it is not equal to the X resolution divided by the Y resolution.

-bsp_depth depthint

The maximum number of levels in the BSP tree. This option is used only if binary space partitioning is enabled. Larger tree depths reduce rendering time but increase memory consumption; they also slightly increase preprocessing time. The default is 40.

-bsp_memory memoryint

The maximum memory in megabytes used in the BSP preprocessing. A value of zero indicates that there is no limit on the memory consumption; this is the default. This flag is useful only on multiprocessor computers, since the memory consumption increases with the number of rendering threads. When the specified amount of allocated memory is reached, mental ray will prevent threads from being scheduled for preprocessing, thus reducing the memory requirements.

-bsp_size sizeint

The maximum number of primitives in a leaf of the BSP tree. This option is used only if binary space partitioning is on; it has no effect on the ray-classification algorithm. Larger leaf sizes reduce memory consumption but increase rendering time. The default is 10.

-caustic on|off

Toggles the generation of caustics defined in the scene. The default is off. To actually create caustics, lights must have an energy and materials must have photon shaders.

-caustic_accuracy nphotonsint [**radius**]

Controls how caustics are estimated from the photon map during rendering. nphotons is the maximum number of photons to examine, and radius is the maximum radius to search. If nphotons = 0, all photons are examined up to the radius limit. Radius = 0.0 means that a scene-size dependent radius will be used. The defaults are 100 and 0.0.

-clip hither yon

The hither (near) and yon (far) planes are planes parallel to the viewing plane that delimit the rendered scene. Points outside the space between the hither and yon planes will not be rendered (this does not apply to the infinite-radius environment maps because they are not geometric objects). The clip statement specifies the distance of the hither and yon planes from the camera.

-code "filename" ... -

The named filename is interpreted as a C source file, ending with the extension ‘‘.c”, is compiled and linked into mental ray. The shaders it defines are available in mental ray as shading functions. Multiple file names can be given. The list must be terminated with a double minus sign.

-colorclip rgb|alpha|raw2.1

Controls how colors are clipped into a valid range [0, 1] before being written to a non-floating point frame buffer or file. The rgb mode is the default. In this mode, RGB is first clipped to [0, 1] and alpha subsequently to [max(R, G, B), 1]. In alpha mode, alpha is first clipped to [0, 1] and RGB subsequently to [0, A]. In raw mode, RGB and A are both clipped to [0, 1] independently of each other. In all modes, the RGB components are clipped as specified by the desaturate option. The RGB and alpha modes ensure that the resulting color is a valid premultiplied color. RGB should be used if the alpha channel is considered less important than preserving the RGB color and intensity. Alpha mode is intended for alpha compositing, where the alpha channel is more important than the absolute color value to preserve correct transparencies. Raw mode should only be used if no layering based on alpha is going to take place. This mode also forces the premultiply mode to On. It should be used with care, because shaders might receive ‘‘illegal” colors (colors that cannot be composited in standard ways).

-contrast r g b [**a**]

The contrast controls supersampling. If neighboring samples differ by more than the color r, g, b, a, supersampling is done as specified by the sampling options (see above). Default for a is the average of r, g, and b. The recursive supersampling algorithm controlled by -samples modifies the contrast based

on the recursion level: at sample level 0, the contrast is used directly; at sample level 1, the contrast is doubled (effectively requiring a higher contrast to force another subdivision), and so on. Negative levels divide the contrast, i.e. use a fraction 12, 14, and so on. In general, the contrast is multiplied by 2level at the supersampling level level, which is bounded by -samples. See the Scene Description chapter for more information on how to optimize performance and quality with contrast values.

-cut_window numint [expand]

Cut a frame into num x num sub-frames, each of which is rendered individually. When all are finished, they are combined to the final image. This can greatly reduce memory consumption if the scene is built such that each sub-frame is much less complex than the entire frame, but it may take much longer to run all the extra rendering cycles. If the scene contains few polygons and many free-form surfaces and/or displacement maps, memory usage often goes down by factors of five or ten, while rendering time doubles. Mostly, polygonal scenes usually do not benefit from cut windows, and in fact can behave worse. Tessellation caching in animations is disabled in cut window mode.

Before deciding whether geometry outside the cut window frustum should be tessellated, the frustum is enlarged by 10% all around to catch geometry outside the frustum that is displaced into the frustum. This value can be changed with the optional expand argument, which is a factor of the cut window size. 0.1 expands the cut window by 10% all around, which increases its area by a factor of 1.44.

-c_compiler "filename"

If this option is given, the standard C compiler "cc" is replaced with filename.

-c_flags "options"

Replaces the standard options given to the C compiler . The defaults depend on the computer used; for example, the default-option string for single-processor SGI IRIX computers is "-O2". The -c option is always inserted, as is -o if the compiler supports it.

-c_linker "filename"

If this option is given, the standard linker "ld" is replaced with filename.

-desaturate on|off

If a color is output to a frame buffer that does not have 32-bit (floating-point) precision, and its RGB components are outside the range [0, max], mental ray will clip the color to this legal range. If desaturation is turned off, the individual components are simply clipped into range. Otherwise, SOFTIMAGE|XSI tries to maintain the brightness of the color by moving it towards the grayscale axis of the color cube until the RGB components are in the legal range. Max is determined by the colorclip2.1mode. Desaturation is turned off by default.

-diagnostic grid off|object|world|camera size2.1

This diagnostic mode is intended for scene debugging. Unless off, it draws a colored grid on all objects in the scene that indicate object, world, or camera space coordinates. Steps on the X, Y, and Z axes are shown with red, green, and blue grid lines, respectively. The distance between grid lines is size units. This is useful to estimate the size and distances between objects in mental ray, or to visualize object space coordinates. The off argument disables the grid. The default is off.

-diagnostic photon off|density|irradiance max2.1

When rendering caustics or global illumination, this option disables all material shaders in the scene and produces a false-color rendering of photon density, or the average of the red, green, and blue irradiance components. Photon density is the number of photons per unit surface area.

-diagnostic samples on|off2.1

Switches to sampling visualization mode and creates grayscale images representing sampling densities instead of color images. A black pixel has had no samples, whereas a white pixel has had the maximum amount as specified by the -samples option. In addition, a red grid is drawn indicating task boundaries. The default is off.

-displace on|off

Ignores all displacement shaders when off. The default is on.

-dither on|off

mental ray supports both 8 and 16 bits per color component. In some cases, 8 bits per pixel, as supported by all popular picture file formats, can cause visible banding when the floating-point color values calculated by the material shader are quantized to the 8-bit values used in the picture file. Dithering mitigates the problem by introducing noise into the pixel such that the round-off errors are randomly distributed. Note that this can cause run-length encoded picture files to be larger than without dithering. Dithering is turned off by default.

-echo "filename" [**ascii**] [**source**] [**approx**] [**norendercommand**] [**textures**] -

Echo the current scene to the file filename. The options specify the format of the echoed file. Allowed options are:

- **ascii** uses ascii format for the vectors (default is binary)
- **source** prefers source geometry over triangles if available (default)
- **approx** prefers triangles over source geometry if available
- **norendercommand** disables the echo of the render command
- **textures** includes texture pixel data verbatim.

Note that triangle echos have displacement mapping already applied to the triangles, but the displacement shaders are not removed from the materials, so the echoed file will get displaced twice when rendered. The echo option must be terminated with a double minus.

-face front|back|both

The front side of a geometric object in the scene is defined to be the side its normal vector points away from. By specifying that only front-facing triangles are to be rendered, speed can be improved because fewer triangles need to be tested for a ray. This works well unless there are objects whose back side is seen by refracted or reflected rays - with face front, the back side would not be visible. The default is both faces.

-file_name "filename"

Overrides the file name given by the first file output statement in the camera in the scene file. The full file or path name must be given, including extension if desired.

-file_type "format"

Overrides the file format given by the first file output statement in the camera in the scene file. File formats include rgb for SGI RGBA files, jpg for JPEG files, "pic" for SOFTIMAGE image files, "rla" for Wavefront RLA files, and "ps" for PostScript files if contour mode is enabled. For a complete list, see the Output Shaders subsection in the Functionality chapter.

-filter box|triangle|gauss|mitchell|lanczos [**width** [**height**]]

The - filter option specifies how multiple samples are to be combined into a single pixel value. The filter defaults to a box filter of width and height 1.0, optimized for speed. This option allows replacing the box filter, or its size (in pixels). If the size of the filter is not specified, default values are used. These are 1.0 for box, 2.0 for triangle, 3.0 for Gauss and 4.0 for Mitchell2.1and Lanczos2.1. If the height is omitted, it defaults to the width. Other values can be specified: larger filter sizes result in softer images. If the size is too small, artifacts may appear. Filters must be larger than 0.0 but sizes smaller than 1.0 are generally wasteful since they will discard some samples. Filters larger than 1.0 can reduce rendering speed. For further details on filtering, see the description of the filter statement in the Scene Description chapter.

-finalgather on|off2.1

Toggles final gathering. It is off by default. Final gathering is a rendering technique used for computing indirect illumination with a one-generation raytracing step.

-finalgather_accuracy nraysint [maxdist [mindist]]2.1

nrays is the number of rays cast in a final-gathering step during rendering. The default is 1000. maxdist is the maximum distance within which a final gather result can be reused. The default is scene dependent. mindist is the minimum distance within which final gather results must be reused. The default is scene dependent.

-focal distance | infinity

The focal distance is set to distance. The focal distance is the distance from the camera to the viewing plane. The viewing plane is the plane in front of the camera that the rendered scene is projected onto; its edges correspond to the edges of the rendered image. However, objects between the camera and the viewing plane will still be rendered; a common approach is to place the viewing plane in the middle of the interesting objects in the scene and then set the aperture such that it is a bit larger than the horizontal extent of the objects in camera space. If infinity is used in place of the distance, an orthographic view is rendered. An orthographic view turns off perspective; all camera rays are parallel. View-dependent surface tessellation is not possible in orthogonal mode.

-gamma gamma_factor

Gamma correction can be applied to rendered color pixels to compensate for output devices with a nonlinear color response. All R, G, B, and alpha component values are raised to gamma_factor. The default gamma factor is 1.0, which turns gamma correction off.

-geometry on|off

Ignore all geometry shaders if set to off. The default is on.

-globillum on|off

Enable or disable the computation of global illumination. The default is off. To actually compute global illumination, lights must have an energy, and materials must have photon shaders.

-globillum_accuracy nphotonsint [radius]

Controls how global illumination is estimated from the photon map during rendering. nphotons is the maximum number of photons to examine and radius the maximum radius to search. if nphotons = 0, all photons are examined up to the radius limit. radius = 0.0 means that a scene-size dependent radius will be used. The defaults are 500 and 0.0.

-grid_size factor

Adjusts the scene-dependent default grid voxel size. The default is 1.0. Larger values increase the number of voxels and shrink each voxel accordingly.

-help

Prints a summary of all options with their allowed parameters, and terminates.

-hosts "hostname[:portnumber] [remote parameters]" ... -

The host list overrides the host list taken from the .rayhosts file, if present. One server is started on each host specified. Host names must be given as expected by the local name-resolving method (such as /etc/hosts) or as a numeric internet address (nnn.nnn.nnn.nnn). Computers used as servers must be correctly configured; see the installation notes. The host list must be terminated by a double minus.

-I path

Overrides the path used to resolve $include statements in the .mi scene file. The default path is /usr/include. Note that only one -I option can be specified. It may contain a colon-separated (Unix only) or semicolon-separated (Unix and NT) list of directory paths that are tried in sequence if a $include statement using angle brackets is used in the .mi scene file. Paths introduced with an exclamation point are special; they are applied to quoted $include paths too, and substitute the entire directory path. This can be used to force mental ray to use a specified path regardless of the path specified in the $include statement, for example, because that path points to an obsolete (1.9) version of a declaration file. For example, -I /a:/b:!/new tries to find a path <x/y/z> first as /a/x/y/z, then /b/x/y/z, then /new/z.

-imgpipe fdint

Normally, mental ray prints connection information into the output image file that let programs like imf_disp connect and display pictures while being rendered. If -imgpipe is used, the relevant information is printed to the given file descriptor fd instead. This can be used for command lines such as `‘‘ray -imgpipe 1 scene.mi | imf_disp -’’`. The imf_disp program is a viewer provided by mental images that supports image piping.

-jitter jitter

The jittering factor introduces systematic variations into sample locations. Without jittering, samples are taken at the corners of pixels or subpixels. Jittering displaces the samples by an amount calculated by lighting analysis, limited to jitter pixels. This is used to reduce artifacts. Jittering is turned off by default or by specifying a jitter of 0.0. Jittering works best in raytracing mode. The jitter value is always set to 1.0 when jittering is turned on.

-ld_libs "libraries"

The libraries string replaces the standard library options given to the linker. The defaults depend on the computer used, typically "-lm -lc". Linker options are highly computer dependent and operating-system dependent and cannot be changed.

-ld_path "path1;path2;..."

Supply a list of paths that mental ray searches for dynamic shared objects (DSO) containing shader code. The paths given here precede those that can be given by an environment variable (MI_LIBRARY_PATH) and the built-in search path (/usr/local/mi/lib;.). -ld_path is synonymous with -L.

-lens on|off

Ignore all lens shaders when off. The default is on.

-link "filename" ... -

Like the code command, the link command attaches external shaders to mental ray, which can then be used as shading functions. While the code command accepts ‘‘.c” files as filename, the link command expects either object files ending in ‘‘.o” or dynamic shared-object (DSO) files ending in ‘‘.so”. Object files are linked, while DSOs are just attached without any preprocessing. DSOs are the fastest way of attaching an external shader, and they require no compilers or development options, which are sometimes sold separately by system vendors. On Windows NT, .so extensions are automatically converted to .dll for cross-platform compatibility. 5.1 However, not all systems support DSOs; see ‘‘Dynamic Linking of Shaders”. The file name list must be terminated with a double minus sign.

-merge on|off

Ignore all merge epsilons and all connections in the scene.

-message moduleclass_list ... -

Toggles individual message classes, per module. The module names are printed at the beginning of every message printed by mental ray; all can be used to modify the message classes of all modules. The class_list is a comma-separated list of classes to print. Supported message classes are phase, progress, vprogress, time, scene, memory, render, vrender, resources, network, files, and debug. The special words default, all, and none are also supported. A class can be inverted by prepending an exclamation point. For example, to print less verbose RC progress messages and make all modules report every file accessed, specify

```
-message rc default,!vprogress all default,files -
```

-o "filename"

This is an abbreviation for -file_name.

-offset x y2.1

Specifies an offset for the rendered image. The default is 0.0 for both values, which means that the image will be centered on the camera’s Z axis. Positive values translate the image up and to the right. The offset is measured in pixel units.

-output on|off

Ignores all output shaders when off. Default is on. File-output statements are not affected.

-photonmap_file "filename"

Uses filename for the photon map, in all frames. If the photon map file does not exist, it is created and saved. If it exists, it is loaded and used. For multiple frames it is only created for the first frame and then loaded for the remaining frames.

-photonvol_accuracy nphotonsint [**radius**]

Controls how global illumination2.1or caustics in participating media are estimated from the photon map during rendering. nphotons is the maximum number of photons to examine, and radius is the maximum radius to search. If nphotons = 0, all photons are examined up to the radius limit. Radius = 0.0 means that a scene-size dependent radius will be used. The defaults are 30 and 0.0.

-photon_depth reflectint [**refractint** [**sumint**]]

photon_depth is similar to trace_depth except that it applies to photons. Reflect thus limits the number of recursive reflection photons. When = 0, no photons are reflected; When = 1, one level is allowed but a photon cannot be reflected again, and so on. Similarly, refract controls the maximum depth of refracted photons. Additionally, it is possible to limit the sum of reflected and refracted photon levels with sum. Note that custom shaders may override these values. The default value is 5 5 5.

-premultiply on|off

Premultiplication means that colors are stored with alpha multiplied to R, G, and B. For example, white at 10% opacity is not stored as (1, 1, 1, 0.1) but as (0.1, 0.1, 0.1, 0.1). This is the standard method in computer graphics to represent colors; mental ray always uses it internally and in all shaders. One implication is that R, G, and B can never exceed A. mental ray normally enforces this when storing color values into frame buffers. The -premultiply off option instructs mental ray to always store colors unpremultiplied into frame buffers and files. It does this by undoing the internally applied premultiplication. (mental ray internally always works with premultiplied colors to present a uniform interface to shaders; since this is done with floating-point values there is no precision penalty.) This option is ignored if the -colorclip raw2.1mode is in effect.

-raycl_memory memoryint

This option sets the amount of memory to be used by the ray space subdivision algorithm for acceleration data structures to memory megabytes on each CPU. It has no effect if the BSP algorithm is used. mental ray allows presetting the amount of rendering acceleration memory independently of scene complexity without sacrificing speed. The default is set to 6 megabytes, which is sufficient for most scenes. Even for extremely large scenes, little can

be gained from memory sizes greater than 12 megabytes. Note that this option does not affect the amount of memory used for the scene description, which depends on the complexity of the geometry in the current frame.

-raycl_subdivision subdivint subdiv_2dint

mental ray uses a raytracing algorithm that subdivides the space of all rays. The optimal subdivision is determined automatically by a built-in scene analysis. The -subdivision option can be used to modify the result of this analysis; arguments of 0 leave the calculated subdivision unchanged, positive numbers increase and negative numbers reduce the subdivision. Subdiv controls general subdivision; subdiv_2d controls primary (eye) and shadow rays. This option has no effect when the BSP algorithm is used.

-render beginint [**endint** [**incint**]]

Renders only frames begin through end5.2. If end is omitted, begin and all following frames are rendered. If inc is given, only every inc-th frame is rendered. Frame 1 is considered the first render statement in the scene file; camera frame specifiers are not considered. For example, 4 8 2 will skip the first three frames, then render frame 4, 6, and 8, and omit the rest. It is not an error if the scene file has fewer frames than requested.

-resolution xint yint

Specifies the width and height of the output image in pixels.

-samplelock on|off

Toggles wether to let the sampling of area light sources, motion blur, and depth-of-field be static or depend on the frame number. The default is on, meaning static sampling.

-samples minint maxint

This option determines the minimum and maximum sample rate. Each pixel is sampled at least 22 . min times in each direction. If min is 0, each pixel is sampled at least once. Positive values increase the minimum sample rate; negative numbers reduce the sample rate to less than one initial sample per pixel (infrasampling). min has default -2, which means that at least one sample per 4 x 4 pixels is taken. If min is chosen too small, small features may be lost if all samples happen to miss it (if it is found just once in any pixel of a task, mental ray will analyze the feature and render it correctly). If a filter camera statement is used to set a filter other than box 1 1, min must be set to -1 or greater (mental ray 2.1) or 1 or greater (mental ray 2.0).

The max value sets the maximum sample rate. If neighboring samples find a difference in contrast exceeding the contrast limit, the area that contains the contrast is subdivided until the maximum recursion depth specified by max is reached. At most 22 . max samples per pixel are taken. If a filter camera statement or option is used to set a filter other than box 1 1, max must be set to 0 or greater (mental ray 2.1) or 1 or greater (mental ray 2.0).

-scanline on|off|opengl

This statement controls the scanline rendering algorithm. By default, mental ray uses the scanline algorithm to compute rays traveling in a straight line from the camera, such as primary rays. In most cases, this gives better performance than pure raytracing. Turning scanline off forces mental ray to rely entirely on raytracing. This will generally slow down rendering but in some cases, for example when the task size is very small, the overhead of initializing the scanline algorithm may outweigh its benefit and turning it off can result in a speed improvement.

-scanline opengl2.1 mode will cause mental ray to use OpenGL hardware, if available, to further accelerate rendering. OpenGL is not used to compute actual colors but only projection and intersection information, so the full mental ray shader feature set remains available. However, there may be a precision loss. Since OpenGL rendering is typically extremely fast compared to network transfer times, only the client host uses its OpenGL hardware. The results are broadcast to server hosts if present. Server hosts never use their OpenGL hardware. Also see -task_size below.

OpenGL rendering makes certain demands on the OpenGL hardware present. If a 24 bit frame buffer with a Z buffer is not present, OpenGL rendering may fail. OpenGL may also fail or become slow if the image to be rendered is larger than the workstation display. The max parameter to the -samples option should not be higher than 0, or image artifacts may appear. During rendering with OpenGL, a window may briefly appear containing seemingly random colors. On Unix workstations, mental ray will need access to the X server and OpenGL hardware for OpenGL rendering to work. This is usually achieved through the use of the DISPLAY environment variable. See your workstation manuals for details. Finally, OpenGL rendering does not work with motion blurring. If, for any reason, OpenGL rendering is not possible on the client host, mental ray falls back on regular scanline rendering.

-shadow off

All shadowing is disabled.

-shadow on

This flag enables simple shadowing. Shadow shaders determine how much light from a light source passes through a shadow-casting object between the light source and an illuminated point on some other object. In shadow-on mode (the default), shadows are computed. Shadow shaders are called in random order.

-shadow sort

This flag also turns on shadowing but alters the way shadow shaders are called. The shadow-casting objects are sorted such that the shadow shader of the object closest to the illuminated point is called first and the one closest to the light is called last (unless the sequence terminates early because light is completely blocked).

-shadow segments

As with shadow sort, the shadow shaders are called in a sorted fashion. 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 slows down rendering but is required if volume effects should cast shadows (as for certain complex shaders such as fur shaders).

-shadowmap [**on**] [**off**] [**opengl**] [**only**] [**rebuild**] [**reuse**] [**motion**] [**nomotion**] -

Can be used to control shadow maps: The allowed options are:

- **on** enables use of shadow maps.
- **off** disables use of shadow maps. This is the default.
- **only** causes only shadow maps to be rendered, without rendering a color image. By default, the color image is rendered also.
- **rebuild** causes all shadow maps to be recomputed, even if they are found in memory or on disk. By default, reuse is in effect, meaning that shadowmaps are computed only if they are not found in memory or on disk.
- **reuse** causes shadowmaps to be reused. First, internal memory is searched for shadowmaps from a previous frame. If they are not found, a search is made on disk in the current directory, according to the filenames specified in the .mi file, if given. If neither is found, the shadowmaps are recomputed.
- **motion** activates motion blurred shadow maps. This is the default.
- **nomotion** disables motion blurred shadow maps. This improves rendering speed.
- **opengl2.1** causes mental ray to try to use OpenGL acceleration when rendering shadow maps. The same limitations apply as mentioned with the -scanline opengl option. Additionally, because of the difference of the rendering algorithm, shadow maps rendered with this option contain slightly different information from those generated with the regular algorithm, and the resulting shadows may look different. In particular, soft areas of shadows tend to be smaller and some areas may incorrectly be determined to be not in shadow due to the lower precision of OpenGL acceleration. When OpenGL rendering of shadow maps is enabled, only the client host will participate, since the computation cost of the map is so small that the network transfer time would be prohibitive.

-shutter shutter

Specifies the shutter open time. A shutter value of 0.0 turns motion blurring off; values greater than 0.0 turn motion blurring on. The shutter value scales the motion vectors attached to object vertices; if shutter is 1.0, each vertex moves the distance given by its motion vector and is blurred in the image over this distance. Values less than 1.0 reduce this path.

-task_size task_sizeint

Specifies the size of the image tasks during rendering. Smaller task sizes are convenient for previewing but also increase the overall rendering time. This option can also be used in order to optimize load balancing for parallel rendering. Note that very small task sizes can cause the scanline algorithm to perform poorly and in such cases it may be desirable to turn it off. See -scanline above. If the task_size is not specified, an appropriate default value is used.

-threads nthreadsint

Normally, mental ray starts one thread for each CPU in the system. In a single-processor host, the default is always 1. This option changes the number of threads. There is normally no advantage in increasing the number of threads, but it may be lowered to reserve CPUs for other users; to avoid monopolizing a multi-processor computer. Note that Convex computers do not make the number of CPUs in the system available to user programs, so -threads should always be used.

-time_contrast r g b [**a**]

Controls temporal supersampling for motion blurred scenes. It works similar to the spatial-contrast parameter explained above: If neighboring samples in time differ by more than the color r, g, b, a, supersampling is done. Default for a is the average of r, g, and b; the default for r, g, and b is 0.2. Using values for -time_contrast that are higher than -contrast can speed up motion blur rendering at the price of grainier images without degrading the quality of spatial antialiasing.

-trace on|off

Normally, mental ray uses a combination of a scanline algorithm and raytracing to calculate samples of the scene. If -trace off is specified, raytracing is disabled, and mental ray will rely exclusively on the scanline algorithm. Since the scanline algorithm can only compute straight rays from the pinhole camera, reflection rays cannot be cast and refraction rays are computed like transparent rays, which do not allow control over the ray direction based on the index of refraction of the material. Lens shaders cannot alter ray origin and direction. However, reflections onto environment maps do work. Motion blurring and shadows are also affected if raytracing is turned off. Raytracing is on by default.

-trace_depth reflectint [**refractint** [**sumint**]]

reflect limits the number of recursive reflection rays. When = 0, no reflection rays will be cast; when = 1, one level is allowed but a reflection ray cannot be reflected again, and so on. Similarly, refract controls the maximum depth of refraction and transparency rays (which implement transparency with and without index of refraction). Additionally, it is possible to limit the sum of reflection and refraction rays with sum. For example, if 3 3 4 is given, an eye ray may be reflected 3 times, or refracted 3 times, or reflected twice and then refracted twice, or any other combination that sums up to at most 4. Shaders may override this setting. The default is 1 1 1.

-v on|off|levelint

An abbreviation for -verbose.

-verbose on|off|levelint

Controls verbose messages. There are seven levels: fatal errors (0), errors (1), warnings (2), progress reports (3), informational messages (4), debugging messages (5), and verbose debugging messages (6). All message categories numerically less than level are printed. Verbose off is equivalent to level 2 (fatal errors and errors); verbose on is equivalent to level 5 (everything except debugging messages).

-view_samples on|off2.1

Toggles the sample view mode. When on, the rgb image will be replaced with an image that shows where samples were collected during rendering. This is useful for tuning the -samples and -contrast parameters and identifying troublesome areas in the image. A pixel's intensity reflects the number of samples collected within the pixel and on its lower and left edges. It is normalized to the maximum number of samples per pixel. In addition, task boundaries are shown as red in the image.

-volume on|off

Ignore all volume shaders if set to off. The default is on.

-window x_lowint y_lowint x_highint y_highint

Only the subrectangle of the image specified by the four bounds will be rendered. All pixels that fall outside the rectangle will be left black.

-xcolor [**"control"**]

Print colored messages. Error messages, for example, are printed in red, which makes them stand out much better in verbose reports. The control string allows customization; it consists of seven characters describing the severity level (fatal, error, warning, info, progress, debug, and vdebug). Each character is one of k (black), r (red), g (green), y (yellow), b (blue), m (magenta), c (cyan), w (white), or a dot (no color).

Appendix Setting Up Distributed Rendering

This appendix is meant to guide you through the distributed-rendering setup for both IRIX and Windows NT platforms. It also highlights common problems and error messages.

Distributed rendering is a method of dividing up the task of rendering an image or a scene among many different computers across a network using the mental ray 2.1 renderer. You can also divide a rendering task among several processors on multi-processor computers. By spreading the workload, your overall rendering time can be decreased considerably. The number of computers you can use is only limited by the number of mental ray 2.1 licenses you have. In the case of multiprocessor computers, each processor requires a separate license.

How Distributed Rendering Works

Distributed rendering is started automatically once a render is initiated on a computer. The initiating computer is referred to as the *master*, and the other computers on the network are *slaves*. The master and slaves communicate via the rayserver service (called **ray2xsiserver**), which runs the **ray2xsi.bat** batch file (Windows NT) or **ray2.sh** script file (IRIX) on each computer. These files set the environment variables required for distributed rendering.

The master reads a local file called **.ray2hosts**, which lists the slaves to be used for the render. The image to be rendered is broken up into segments (*tiles*), which are placed in a queue. Each computer, master or slave, requests tiles from the queue to render. Once a tile is finished, it is sent back to the master, and another tile is requested from the queue. The master assembles all the tiles to create a complete rendered image.

During distributed rendering, the master will also send any extra information the slave might need to accomplish a render, such as textures. When using large or memory-mapped images, you can use the **linktab.ini** file to coordinate file sharing.

FLEX*lm* Installation

The FLEX*lm* server handles the allocation of mental ray renderer licenses when distributed rendering is initiated. For details on installing the FLEX*lm* software, see the *Setup Guide*.

It is recommended that you use a dedicated computer to act as the FLEX*lm* license server; this computer should not take part in the network rendering or any other memory-intensive tasks. If a computer's connection to the FLEX*lm* license server is lost during rendering, the render image will be corrupted.

Setting Up mental ray Rendering Software

The setup for distributed rendering is mostly automatic; however, there are some procedures you should follow before and during setup to ensure that everything runs smoothly. For a more detailed description of using mental ray for distributed rendering, see *Programming mental ray* (on the Online Library CD).

In order for the master to properly communicate with the slaves, each computer should have the same version of the mental ray rendering software installed. The master should have more memory than the slave computers, and as much virtual memory as possible.

The .ray2hosts File

The **.ray2hosts** file contains a list of the computers that can participate in distributed rendering, not including the master. Each computer must be listed by name on a separate line; any line preceded by a hash (#) symbol is considered a comment and ignored by SOFTIMAGE|XSI. Here's a sample **.ray2hosts** file:

```
# The first three computers are always part of the
# render network.  The last computer listed is only
# used for overnight renders; remove the # to make
# it available.
larry
moe
curly
# shemp
```

The **.ray2hosts** file must be present on the master computer. Once a render is initiated, SOFTIMAGE|XSI searches for **.ray2hosts** in %HOMEDRIVE%%HOMEPATH% (Windows NT) or ~/ (IRIX), then in SI_HOME.

Depending on your network settings, it can take up to several minutes for a computer to tell mental ray that a host does not answer, so be sure to modify your .ray2hosts file if you know that certain computers are down.

The linktab.ini file

The **linktab.ini** file is used when the render network uses a mix of Windows NT and IRIX operating systems. Each line in a **linktab.ini** file contains a Windows NT path and an IRIX path that indicates where SOFTIMAGE|XSI or resources such as textures are located on both operating systems. This allows the rendering master to find required files on slaves, regardless of operating system.

Most **linktab.ini** files contain only one line, indicating where SOFTIMAGE|XSI is located on both platforms. Here's a sample **linktab.ini** file:

```
c:\Softimage     /usr/Softimage
```

- The Windows NT path must come before the IRIX path, and they must be separated by a tab—not spaces.
- All path names, even for Windows NT paths, are case-sensitive.

You can use an exclamation mark to distinguish a mounted volume from the rest of the path. For example, if \\foobar\users\fred (Windows NT) is equivalent to /home/fred (IRIX) and f:\foobar\users is a mounted volume, the linktab.ini line would look like this:

```
\\foobar\users!fred      /home/fred
```

The **linktab.ini** file must be present on the master computer. SOFTIMAGE|XSI searches for **linktab.ini** in SI_LINKTAB_LOCATION, then in SI_HOME. If the scene was imported from SOFTIMAGE|3D, SOFTIMAGE|XSI searches in SI_LOCATION.

If you are using textures or memory-mapped images, you must have entries that point to the Windows NT and IRIX directories containing them.

Environment Variables

Three environment variables are required for distributed rendering.

- LM_LICENSE_FILE must be set to a directory containing license files (for example, **%SI_HOME%\Flexlm\licenses**) or a specific port on a host (for example, **744@larry**). LM_LICENSE_FILE is set in the environment script (**setenv.bat** on Windows NT and **.xsi_1.0** on IRIX) during setup. For more information on the LM_LICENSE_FILE environment variable, see the *Setup Guide*.
- MI_ROOT must be set to the **rsrc** directory in the SOFTIMAGE|XSI application directory. In a default install, MI_ROOT would be set as **C:\Softimage\XSI_1.0\rsrc** (Windows NT) or **/usr/Softimage/XSI_1.0/rsrc** (IRIX). MI_ROOT is set in **setenv.bat** (Windows NT) and **ray2.sh** (IRIX) during setup.
- SI_HOME must be set to the directory in which SOFTIMAGE|XSI is installed. SI_HOME is set in the environment script (**setenv.bat** on Windows NT and **.xsi_1.0** on IRIX) during setup.
- SI_LINKTAB_LOCATION must be set to the location of the **linktab.ini** file.

If you will be importing scenes from SOFTIMAGE|3D, set SI_LOCATION in the **ray2.sh** script (IRIX) or as a system variable by choosing **Start** > **Settings** > **Control Panel** > **System** > **Environment** (Windows NT).

Sample setenv.bat File

```
@echo off
set SI_HOME=C:\Softimage\XSI_1.0
set HOME=%SystemDrive%\users\%USERNAME%
set Path=%SI_HOME%\bin;%Path%
set MI_ROOT=%SI_HOME%\rsrc
set MI_RAY2_SERVICE=mi-ray2xsi
set SI_DBDIR=%HOME%
set SI_IMAGE_PATH=%SI_HOME%\bin\sil
set LM_LICENSE_FILE=744@ludovm4;%LM_LICENSE_FILE%
```

Sample ray2.sh File

```
#!/bin/csh -f
setenv SI_HOME "/var/tmp/Sumatra_RC/Real/XSI_1.0"
setenv MI_ROOT "${SI_HOME}/rsrc"
setenv MI_RAY2_SERVICE "mi-ray2xsi"
setenv LM_LICENSE_FILE "744@ludovm4"
exec /var/tmp/Sumatra_RC/Real/XSI_1.0/bin/ray2 $argv
```

Sample ray2xsi.bat File

```
@echo off
call "C:\Softimage\XSI_1.0\bin\Setenv.bat"
C:\Softimage\XSI_1.0\bin\ray2.exe
```

Troubleshooting

This section contains solutions to common causes for problems with distributed rendering.

Slaves Not Rendering

If a slave is not participating in the distributed rendering, check to see if:

- It uses a different version of mental ray from the other computers in the network. All computers in a distributed rendering network must use the same version of mental ray.
- Its shaders are located in the same path as the master.

To find the specific cause of the problem, open a SOFTIMAGE|XSI shell and enter `ray2xsi -verbose on`. This will produce a detailed account of ray2xsi's actions.

Render Not Starting

If the render will not start, make sure that you have enough licenses available. Check the FLEX*lm* log for license request information.

Corrupted Images

If your rendered images have corrupted tiles, a slave may be losing its connection to the license server. Ensure that the license server is not running any memory-intensive tasks.

Index

M

N

O

P

R