



Rendering

A Comprehensive User's Guide

The *SOFTIMAGE 3D User's Guide* was written by Susan Duciaume, Camille Lemberg, Suzanne Gerrior, Grahame Fuller, and Edna Kruger; edited by Edna Kruger; illustrations by Richard Z'Graggen; and layout by Paul Verrall.

Special thanks to Craig Hall, David Eu, and Pierre Tousignant for their assistance in assuring the technical integrity of this guide.

© Copyright 1996 Softimage Inc., a wholly owned subsidiary of Microsoft Corporation. All rights reserved.

Microsoft®, MS®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States and/or other countries.

SOFTIMAGE® is a registered trademark of Softimage Inc., in the United States, Canada, and/or other countries.

3D Studio Max™ is a trademark of Autodesk, Inc.

Adobe Illustrator® is a registered trademark of Adobe Systems, Inc.

Abekas® is a registered trademark of Abekas Video Systems, Inc.

Alias® is a registered trademark of Alias Research Inc.

Apple® and Macintosh® are registered trademarks of Apple Computer, Inc.

Aurora® is a registered trademark of Chyron Corporation.

AutoCAD® is a registered trademark of AutoDesk, Inc.

Flock of Birds® is a registered trademark of Ascension Technology Corp.

Indigo® is a registered trademark and Indy™ is a trademark of Silicon Graphics, Inc.

Magellan™ is a trademark of Logitech, Inc.

Macromedia® is a registered trademark and Freehand™ is a trademark of Macromedia, Inc.

PostScript® is a registered trademark of Adobe Systems, Inc.

SEGA Saturn™ is a trademark of Kabushiki Kaisha Sega Enterprises.

Targa® is a registered trademark of Truevision Inc.

TIFF™ is a trademark of Adobe Systems, Inc.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

V-LAN® is a registered trademark of Videomedia, Inc.

Wacom® is a registered trademark of Wacom Co., Ltd.

WSD® is a registered trademark of Accom, Inc.

All other product names mentioned in this guide may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

This document is protected under copyright law. The contents of this document may not be copied or duplicated in any form, in whole or in part, without the express written permission of Softimage Inc. This document is supplied as a guide for SOFTIMAGE 3D products. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Softimage Inc. does not accept responsibility of any kind for customers' losses due to the use of this document.

Document No. 90442-0596

Printed in Canada.

Table of Contents

CHAPTER ONE

Introduction	1
Introduction to Rendering	3
The Camera	4

CHAPTER TWO

Rendering Basics	11
Previewing before Rendering	13
Optimizing Rendering Time	16
Rendering a Sequence	19
Rendering in the Background	25
Rendering a Subregion	29
Showing and Hiding Object Edges	32

CHAPTER THREE

Advanced Rendering	33
Introduction	35
Raytracing	36
Antialiasing	39
Blurring a Moving Object	42
Field Rendering	47
Rendering Tag and Z Channels	51
Rendering Faces of an Object	54

CHAPTER FOUR

Rendering with mental ray	55
Introduction	57
Setting Up	59
Overview for Using mental ray	63
Using Shaders	65
Raytracing Using Mental Ray	68
Antialiasing Using mental ray	71
Blurring a Moving Object Using mental ray	74

Displacement Mapping77

Rendering Faces of an Object78

Rendering Shadows.....79

Surface Approximation80

Using Output Shaders84

Saving to a File88

I N D E X

C H A P T E R O N E

Introduction

Introduction to Rendering

An object in SOFTIMAGE 3D is defined by points in space that are connected together to create a surface. An important part of the process of creating a 3D object or scene in SOFTIMAGE 3D is the material definition of the object's surface, the lighting of the scene, and rendering. The Matter module contains most of the tools designed to accomplish all these tasks.

For the purpose of rendering, SOFTIMAGE 3D divides these surfaces into triangles, which is known as *tessellation*. Rendering is an integral part of finalizing an object's material definition. When you render an object, SOFTIMAGE 3D processes the object's surface triangles and normals relative to the light source and creates a visible surface that is shaded according to the parameters set for material and texture attributes.

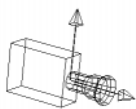
Each triangle is a planar surface with its front face oriented in one direction. The orientation of this surface is shown by a vector (direction line) called a *normal*, located on the points (vertices) of each triangle. The normal always needs to be oriented in the direction of the camera to make that surface visible (you can, however, make all surfaces visible to the camera – see *Rendering Faces of an Object* on page 52).

Lighting

If you render an object without defining a light source, SOFTIMAGE 3D automatically creates a light. The default light is white and infinite (it's far away like the sun), and does not cast shadows. When you create your own light source, the default light disappears.

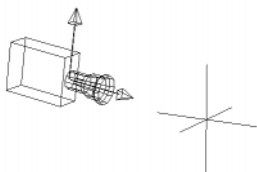
When the surface has been defined, the rendering process calculates all these attributes to create a final image. SOFTIMAGE 3D calculates the relation between the orientation of surface normals and the light source to determine the surface attributes of each triangle. For more information about light, see the *Light* chapter in the *Defining Materials and Textures User's Guide*.

The Camera



The camera is the device that lets you view your scene – what the camera sees is displayed in the Perspective window. Ultimately, the camera's view is what is rendered for your final image. By default, this is whatever is shown in window **B**.

When rendering with the Depthcue, Hardware, Hidden-Line, Wireframe, Ghost, Rotoscope (Wire), or Rotoscope (Shade) rendering types, you can use any of the displayed windows. Rendering with the other rendering types must be done in window **B** with the Perspective view.



You can see the camera itself in the parallel projection windows by choosing the **Camera → Show** command. The camera is represented by a camera icon. It also has an interest to which the camera always points. The interest's icon is three intersecting lines, like a null object.

You can use more than one camera while working on your scene by displaying more than one Perspective window. This allows you to view your scene in different ways and help you determine exactly how to render your scene. These additional cameras are non-rendering.

There are many parameters related to the camera, including its roll, field of view, and depth of field, as well as the picture format, etc. Many of these parameters can be animated.

For more information on moving the camera around in a scene, see *Using the Camera* on page 70 in the *Working with SOFTIMAGE 3D User's Guide*.

Setting Up and Resetting

The **Camera → Settings** command lets you set up all parameters for the camera that renders your scene. These include things such as position of the camera and its interest, camera angle, depth of field, lens shaders, etc. To reset the camera and all its parameters to their default values, choose the **Camera → Reset** command.

When you choose the **Camera → Settings** command, the Camera Settings dialogue box opens in which you can set the parameters. The following sections describe some of the parameters that have a large influence on how a scene is viewed and rendered.

Changing the Angle

The camera lens allows you to define your angle of view in degrees. The **Automatic** setting calculates the camera angle for you depending on its current attributes. For example, if you zoom in to your scene in the Perspective window, the camera angle automatically increases.

The **Custom Angle** option lets you specify the angle of the camera in degrees. A wider angle lets you see more of the scene.

The camera lens angle is based on a 35mm Cine format. Default values for the camera angle are based on 1 SOFTIMAGE 3D unit = 1 foot. SOFTIMAGE 3D units are arbitrary units that you can define, so you can effectively use the camera lens angle values as you like.

Depth of Field

The **Depth of Field Simulation** options provide you with additional control over camera settings, as well as support for lens shaders used when rendering with mental ray. The options define the focus of objects according to their distance from the camera, similar to the way a real camera works. Depth of field refers to the minimum and maximum distance from the camera that objects are in focus. Objects closer than the minimum distance and farther than the maximum distance become increasingly out of focus.

If you select the **Automatic** option in the Depth of Field area of the dialogue box, it allows you to set standard camera parameters to modify depth of field. These three parameters work together to achieve the desired result: **Focal Length**, **F/stop**, and **Distance**.

- **Focal length** allows you to define the length of the camera lens. A larger value increases the lens length and decreases depth of field. For example, 50mm is the standard lens size of a 35mm camera.
- **F/stop** allows you to set the size of the aperture opening. A smaller value results in a larger opening, but decreases the depth of field.
- **Distance** allows you to define the distance from the camera that objects are at their sharpest focus. Objects located in front of and beyond this point become out of focus.

If you select the **Custom** option, you can custom determine your own settings. These are the parameters that you can set:

- **Near Focus** is the nearest distance from the camera where objects are in focus. This is determined in system units.

- **Far Focus** is the farthest distance from the camera where objects are in focus. This is determined in system units.
- **Max COC** is the Maximum Circle of Confusion (COC). It controls the out-of-focus effect in terms of pixels. The higher the value, the more out of focus the object is. The default is 20.

For example, if you preview with a resolution of 100 x 100 and the Max COC is set to 5, you must compensate during the final render. If the final render is 1000 x 1000, the Max COC must be increased to 50. The Max COC is proportional to the image resolution.

- **Max occurs at** is the distance from the camera where Max COC occurs. This is determined in system units.

Choosing an Aspect Ratio

When you render an image, you can select its aspect ratio of the image by choosing the **Camera → Picture Format** command.

It is very important to set the aspect ratio when you first start working on a production. This ensures that what you see in the Perspective window accurately shows you what is within the frame in the final rendering.

When you choose this command, the Picture Format dialogue box appears. You can select standard aspect ratios for film, slides, and video. To customize a non-standard picture format, specify the aspect ratio, or define the size of the image (width and height).

Using Lens Shaders

Lens shaders are only available when you are using the mental ray renderer. You can apply one or more lens shaders to the camera. The effects that lens shaders can produce can be compared to the lens filters of a real camera and produce such effects as lens flares or fish-eye lens.

The following is an example of a fish-eye lens.

1. Choose the **Camera → Settings** command. The Camera Settings dialogue box appears.
2. In the Lens Shader area, click the **Select** button. This opens the database browser to the **Camera Shaders** chapter.
3. Select a camera shader from the list and click **Load**. The camera shader appears in the list of lens shaders.

4. Choose **Preview** → **Setup** in the Matter module and select mental ray as the Preview Renderer.
5. Choose **Preview** → **All** to preview your scene through the fish eye lens.

Using the Lens Flare Shader

The following is an example of how to use a lens shader called a Lens Flare.

1. Create a small city scene with two or three cubes over a plane and one point light.
2. Animate the camera over 10 frames so that the light disappears behind one building.
3. Choose **Camera** → **Settings**. In the Lens Shaders area of the dialogue box, click the **Select** button and select the **Flares_stars** lens shader from the browser that appears.
4. Edit this shader by clicking the **Edit** button, clicking **Select**, and selecting the light source. Select the **Circles** option and deselect **Streaks**.
5. Select the light source and choose **Light** → **Edit**. Give it a raytraced shadow and change the Umbra value to 0.04. You need to lower the umbra value to make the light fade when it is behind an object. An umbra of 0 makes the light fade out completely behind an object.
6. Choose the **Render** menu cell in the Matter module to render the animation using mental ray as the rendering type.

For more information, see *Rendering with mental ray* on page 53.

Controlling the Camera's Movement

When a camera reaches a vertical axis with its interest, it flips. To prevent this, you can constrain the y-axis (up vector) of the camera to another object. There are two reasons to control the up vector of the camera:

- To prevent the camera from flipping when it reaches a vertical axis with the interest. For example, you can produce a proper rollercoaster-like action with the camera using an up vector constraint.
- To control the banking of the camera using a separate control path.

To use the up vector constraint in its simplest form, follow these steps. This uses a camera as the example object:

1. Select the camera.
2. Choose **Constraint → Up Vector** in the Actor or Motion module and select the object to which you want to constrain the camera's up vector.

Using an independent object is not always the most effective way to control the up vector. If the up vector constraint coincides with the interest, a flip occurs on the current vertical axis. This problem can be solved by creating the following nulls and constraints. This procedure makes the up vector follow the “point of view” of an animated object exactly, allowing you to animate the camera the same way as you would animate any object:

1. Select an object.
2. Create three nulls:
 - Place one null in front of the object.
 - Place one null in the centre of the object as its “point of view.”
 - Place one null on the y-axis directly above the “point of view” null.
3. Make these nulls children of the object so that they follow the object's movements.
4. Select the camera and constrain it positionally (**Constraint → Position**) to the null in the centre of the object (the “point of view” null).
5. Select the camera interest and constrain it positionally (**Constraint → Position**) to the null in front of the object.
6. Select the camera, choose **Constraint → Up Vector**, and pick the null directly above the object's centre on the y-axis.
7. Animate the object as you like.

Now when the object moves, the camera's up vector is oriented correctly because it is constrained to the null above the object.

Representing an Asymmetrical Camera Projection

The Camera Projection Plane allows for the representation of an off-axis/asymmetrical camera projection.

The perspective projection is in one of two modes: symmetrical (normal SOFTIMAGE 3D camera) or asymmetrical (if the projection plane parameters are activated).

These parameters are accessible only through the Spreadsheet window. To have them listed in the spreadsheet, do the following:

1. In the Spreadsheet window, click **QUERY**. The Set Query dialogue box is displayed.
2. At the bottom of the dialogue box, in the **Edit Clause** section, set the following clause:

chptyp == CAM

and click Accept. Be sure that you are editing the main query, do not add another clause or a subquery

3. Click **Columns** to open the columns dialogue box, and expand the camera parameters in **Available Columns** by double-clicking on **CAMERAS**.
4. Scroll to the bottom of the list to see the camera projection plane attributes. Highlight the following:
 - cmpdst (camera projection plane distance)
 - cmpsxx (camera projection plane size in x)
 - cmpszy (camera projection plane size in y)
 - cmpofx (camera projection plane offset in x)
 - cmpofy (camera projection plane offset in y)
 - cmpact (camera projection plane active)
5. Click Add to place them in the **Displayed Columns** list.
6. Click Exit to return to the Set Query dialogue box. If you want to save your query, see *Saving a Query* on page 93 of the *Working with SOFTIMAGE 3D User's Guide* for more information.
7. Click Ok to run your query. The Spreadsheet displays a column for editing the projection plane values, and enabling and disabling the **cmpact** (camera projection plane active) parameter. Middle-click on a cell to toggle this parameter on and off.

C H A P T E R T W O

Rendering Basics

Previewing before Rendering

Before you render a scene, you will probably want to preview it first. Previewing lets you render a scene but not save it to file. It uses a modified version of general rendering settings so that your previewed scene is a fairly good approximation of the final look.

If you have not assigned material attributes to the object, it is rendered with its default parameters. If you have not defined a light source, SOFTIMAGE 3D creates a default light.

Setting Up for Previewing

Before you preview a scene, you should choose the **Preview → Setup** command in the Matter module and enter an image resolution and select the Preview Renderer (either SOFTIMAGE or mental ray) at the minimum. The attributes you set up here also affect the previews you do from the Material, 2D Texture, and 3D Texture dialogue boxes.

Previewing in Different Ways

You can preview different objects or areas in a scene, or the whole scene.

- The **Preview → All** command in the Matter module allows you to preview the whole scene.
- The **Preview → Selection** command module allows you to preview only the selected object (or objects) in the scene. This is obviously much quicker than previewing a whole scene if you don't need to.

Previewing a Subregion

To preview a subregion, you can define an area in a scene according to the parameter settings defined using the **Preview → Setup** command. Objects in the defined area are rendered with material attributes, textures, and lights. Hidden objects are not rendered.

To preview a subregion:

1. Choose **Preview → Setup**. In the Subregion area, select the **Draw Subregion on Previewing** option.
2. Click the **Draw Subregion** button. A red box appears in window B that defines the region you want rendered.

3. Click and drag the left mouse button to resize the box. Middle-click to accept the size.
4. The **Preview Setup** dialogue box reappears. Click Exit.
5. Choose the **Preview → Subregion** command so it is active.
6. Choose **Preview → All** or **Selection** and then middle-click. The preview appears on the screen showing only the subregion you selected.

You can also define the subregion by assigning values to Subregion Pixel and Percent (left, right, bottom, top).

For more information on subregion rendering, see *Rendering a Subregion* on page 29.

Previewing a Hidden Line Image

In addition to previewing a fully shaded image, you can preview a hidden-line picture in any window *before* you render it, which is unlike the **Line → Show** command in the Tools module which requires a rendered .lin file.

The command you choose (**Faceted** or **Smoothed**) corresponds to the type of hidden-line renderer you can select in the Render Setup dialogue box.

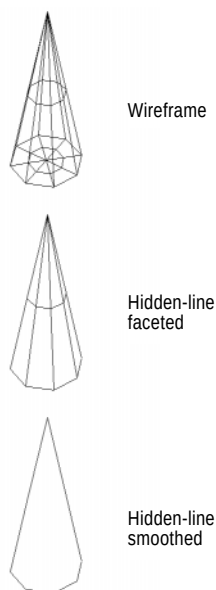
Faceted means that lines are drawn for all the object's surface polygon edges. This makes the object look like a typical wireframe object, except that the hidden lines are removed.

Smoothed means that the hidden lines are also removed, but only the lines at the object's edges are drawn, which creates an outline of the object with no lines inside it.

To adjust the way the hidden-line is processed for polygon mesh objects, select the **Automatic Discontinuity** option in the dialogue box that appears when you choose the **Info → Selection** command. The **Edge_Flag** menu commands in the Matter module also let you adjust the hidden-line process (see *Showing and Hiding Object Edges* on page 32).

To preview a hidden-line image, follow these steps:

1. Select any window by clicking the letter identifier box (A, B, C, or D) in the upper-left corner of the desired window. The window is highlighted in red. If you don't select a window, window B is used.



2. Choose the **Hidden_Line** command in the Tools module (**Faceted** or **Smoothed** relates to the type of hidden-line renderer you can choose in the Render Setup dialogue box).

The image is rendered and displayed in the window you selected.

3. Middle-click to close the window.

Optimizing Rendering Time

Before rendering, it is important to think of ways to optimize rendering time, since this is one of the most time-consuming parts of creating your animation. By keeping efficiency in mind when modelling objects, defining their material attributes, and creating light sources, projects which might otherwise be very time-consuming to render will become much more feasible.

Modelling

When modelling, try to minimize the total number of triangles in a scene. You can do this starting from the earliest stages of the working process. For example, if you are drawing a curve to be used for extrusion or revolution, you should use the fewest points possible to create the desired contour: the resulting object will have fewer triangles.

If you plan to include lots of smooth rounded objects in your scene, you can model them as polygonal mesh objects rather than patch or NURBS, then simulate their surface smoothness by setting the Automatic-Discontinuity option in the dialogue box opened by the **Info → Selection** command: if the objects were extruded from triangles instead of circles, they would have fewer triangles.

If you are working with patch or NURBS surface objects, you can minimize the number of triangles for objects that are located further away from the camera by lowering their geometric resolution, again by using the **Info → Selection** command and its dialogue box for patch models. Also, instead of modelling very complex objects that have many triangles, you can use texture mapping techniques for transparency and roughness to simulate the desired effects.

Material Attributes

You can further optimize rendering time by keeping material attributes such as reflectivity, transparency, and refraction, as well as raytracing depth, and the number of lights and shadows in a scene to a minimum. When used in combination, these attributes can increase rendering time significantly. To save rendering time you can, for example, 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. If you need to increase raytracing depth, perform tests to determine the actual

required depth. Also, remember that each light you add to a scene also affects the rendering: in general, most lighting effects can be created using two or three lights, with only one casting a shadow.

mental ray

The mental ray renderer is a high-quality, photo-realistic renderer available in SOFTIMAGE 3D. This renderer allows you to perform many special effects as part of the rendering process instead of having to create these effects in the scene itself. This helps to keep your scenes as small as possible. This feature of mental ray also lets you save the settings you used during one rendering process and use them in another one. For more information, see *Rendering with mental ray* on page 55 of the *Reference Guide*.

Compositing

Compositing is another very practical way to optimize rendering time. By rendering constant elements of a scene, such as the background, separately from other elements then compositing them together afterwards, you can save a lot of rendering time. The time saved is even greater if raytracing is required. The number of layers that can be composited is unlimited. For more information, see *Compositing Images* on page 73 of the *Using Tools User's Guide*.

Render/Memory Guidelines

These are general rendering guidelines and should not be taken as system limitations because there are ways of maximizing memory usage or adding to available memory.

SOFTIMAGE 3D uses a Binary Space Partitioning (BSP) scheme (for raytracing) which needs to “see” an entire scene to compute the rays’ trajectories. This allows for fast rendering, but very large scenes can put large demands on memory.

Memory can be increased by adding RAM (Random Access Memory) or by partitioning the disk allotting more disk space for swap space (virtual memory). Because of the relatively slow access time of reading information from disk, a scene that requires disk swapping renders more slowly than a scene that fits into RAM. Swapping to disk varies in degrees of relative slowness, depending on where and how often the system needs to retrieve information from the disk.

General Memory Requirements for Rendering

A general rule for calculating memory requirements is to allow 1 Mbyte of memory for every 1,000 surface triangles. For every 512 x 512 texture, you would need an additional 1 Mbyte of memory. These rules assume that you're using raytracing features (reflection, shadows, refraction). If you are not using raytracing, you can render much larger scenes.

Fitting a Scene into Memory

If a scene won't fit into memory, there are a number of ways to correct the situation. The simplest way is to increase the available memory of the computer (RAM and/or virtual).

If increasing the amount of available memory is not possible, there are ways of reducing memory demands. One easy way to reduce memory demands is to use scanline rendering, meaning to not use features which require raytracing.

Another common way to reduce memory demand is to break up large scenes into separate, smaller scenes that can be rendered separately. The rendered images can then be composited back together. Memory demands can also be reduced by adjusting the BSP (Binary Space Partitioning) tree which may, however, affect rendering speed adversely.

Textures are a good place to look to for reducing memory requirements. Be sure to use images that are as small as possible, yet still give an acceptable result. No matter how much disk space images take up, they all use the same amount of RAM depending on the pixel resolution of the image. A 400 x 400 pixel image would use $400 \times 400 \times 4 = 640,000$ bytes or 640 Kbytes of RAM.

A good trick for saving memory with texture and reflection maps is when the colours are the same across the image, you could composite the image so that it is only one pixel wide (this saves a lot of memory). SOFTIMAGE 3D automatically "pulls" the texture across the length of the texture-mapped surface. Texture maps can be quite small with very acceptable results by using the Pixel Interpolation option. This allows the renderer to average the in-between pixels to give the effect of a higher definition map.



The **Edit** button in the 2D Texture File dialogue box displays the Cropping Utilities dialogue box in which you can crop and save a new picture file directly to the disk again.

Rendering a Sequence

To render an object or scene and save the image to file, you can use the **Render** menu cell in the Matter module or the standalone renderer that is run from the command line in a shell.

With the **Render** menu cell, its accompanying Render Setup dialogue box allows you to specify the type of renderer to use, the file output name, the frame numbers to be rendered, image resolution, antialiasing filter, pixel ratio, motion blur, and raytracing depth, as well as other options.

Choosing a Renderer

To choose a render type, choose the **Render** menu cell and select a type from the **Rendering Type** menu. Here's a brief description of the various renderer types available to you:

Softimage Renderer

The SOFTIMAGE renderer is the default renderer. The extension of the output file is **.pic**. The SOFTIMAGE renderer uses most of the parameters in the Render Setup dialogue box so that you can perform complete rendering tasks. You can also run this renderer from the command line (see *Running the Renderer from the Command Line* on page 24).

mental ray

The mental ray renderer is a high-quality, photorealistic renderer. It provides an extensive set of built-in functions and can be dynamically linked with user-defined shaders during the rendering process. You don't have to use shaders with mental ray, but there are many different types of shaders which you can use to create procedural textures (including bump and displacement maps), materials, camera lenses, atmospheres, light sources, etc.

After selecting mental ray from the Rendering Type list, you can set the parameters specific to it. These are accessed by selecting the **Antialiasing** and **Motion Blur** options, as well as the **Options** button.

The extensions of the output files available for mental ray are: **.qntntsc** (Abekas NTSC), **.qntpal** (Abekas PAL), **.ct16** (RGBA 16 bits), and **.rgb**.

For more information on rendering in mental ray, see *Rendering with mental ray* on page 55.

Hidden Line (Faceted or Smooth).

The hidden line renderer removes the lines that are usually hidden from view in a wireframe object. This results in a more realistic looking object than if you used the wireframe renderer (which shows all object lines). The image is saved in **.lin** file format.

- If you select the **Faceted** option, the renderer draws lines for all the object's surface polygon edges. This makes the object look like a typical wireframe object, except that the hidden lines are removed.
- The **Smoothed** option also removes hidden lines, but draws lines only at the object's edges, which creates an outline of the object with no interior lines.

Wire Frame

The wireframe renderer renders wireframe images, which are made up of the edges of objects and drawn as lines (this is the default view shown in the windows when you open SOFTIMAGE 3D). This renderer displays tracing features, such as edges or contour lines without attempting to remove invisible or hidden parts, or fill surfaces. The resulting image is in **.lin** file format.

Depthcue

The Depthcue renderer provides colour wireframe rendering. The portion of the model closest to the camera appears “brighter” in colour, giving the sense of depth in the z-axis. It is usually used to derive more visual feedback from the models displayed on a screen because the hardware shading is often too slow for complex models and scenes.

Since the alpha channel is not calculated in the depthcue renderer, compositing must be done using an external compositor, such as SOFTIMAGE® Eddie.

Hardware Renderer

The hardware renderer renders the same view you see when you choose the Shade view mode in a window. The resulting image is saved in **.pic** file format. The image produced is of lower quality; objects show colour and lighting effects, but not shadows, reflection, or transparency.

Note

Remember to disable your screen saver while rendering with this renderer. This process takes a snapshot of the screen when the image is rendered.

Ghost

The Ghost renderer renders the same view you see when you choose the Ghost view mode in a window. It allows you to display a series of frozen snapshots of your animated objects at the current frame, next frames, and previous frames. These frames remain in the view regardless of the position of the time line pointer and are used as reference point.

Rotoscope (Wire and Shade)

The Rotoscope renderer renders the same view you see when you choose the Rotoscope Wire or Shade view mode in a window. Rotoscoping is a technique in which individual frames of a video or film are imported into SOFTIMAGE 3D and “traced.”

The 2D background can be a static picture or a sequence of images. The scene can be zoomed and panned while maintaining perfect registration with the background.

When zooming in window B, you must use the rectangle zoom option (**Shift-z Supra** key) to define the zoom area; this takes a snapshot of the rectangular area and magnifies it to fill the screen. The camera position does not change. You can then use the zoom option (**z Supra** key) to pan, zoom in, and zoom out on the defined area. When you pan, it has the effect of moving the snapshot in front of a still camera. When you select one of the **Rotoscope** options, the Rotoscopy dialogue box is displayed, as well as the window type in which you are displaying the scene.

Rendering a Scene

When rendering with the Depthcue, Hidden Line, Wireframe, Hardware, Ghost, or Rotoscope (Wire or Shade) rendering types, you can use any of the displayed windows. Rendering with the other rendering types must be done using window B with the Perspective view.

To do a basic rendering of a scene, follow these steps:

1. Select the window you want to render by clicking the letter identifier (A, B, C, or D) icon in the upper-left corner of the

window's title bar. The selected window is outlined in red. If no window is selected, rendering is done in window B.

2. Choose the **Render** menu cell in the Matter module. The **Render Setup** dialogue box is displayed. The parameters described in this chapter refer to this dialogue box.
3. Select a **Rendering type** (one of the previously described renderer types).
4. Specify the sequence by setting the **Start** and **End frame**, and the incrementation step.
5. Leave the default resolution as it is, but you can change it as described in the next section.
6. Specify the name of the scene that you want to render by clicking the **Select** button in the Output Image area and selecting a file from the browser that appears. You can also type a name by which you want to save the rendered image.
7. If you have the choice of file formats, select one from the File Format menu.

For a very basic rendering, this is all you really need to set. If you want to set raytracing, antialiasing, motion blur, or other more sophisticated options, see *Advanced Rendering* on page 33.

8. Click the **Render Sequence** button to start the rendering process.

By default, the rendered picture is saved to the RENDER_PICTURES chapter of your working directory.

9. To view the rendered image or sequence, see the *Using Tools User's Guide* for information on different viewing commands and utilities available.

Setting the Resolution

The resolution sets the frame resolution for your image in pixels. A higher resolution produces a more detailed image.

The higher the resolution, the longer it takes to render. There is no limit for image resolution, but if a resolution is greater than the monitor screen ($x = 1280$), it cannot be displayed on screen while rendering.

In the Render Setup dialogue box, you can assign the x-resolution the value of 720 (**Specify X**) for the NTSC television standard. Once

you specify the x-value, the y-value is automatically calculated according to the x-resolution, so when x is 720, the y-resolution is 486.

You can also set the y-resolution independently of the x-resolution by selecting the **Specify Y** option. This command is more appropriate if, for example, you are printing the image on paper. If you are doing an advertisement that has unusual dimensions, you can set the x and y-resolution values separately.

Resuming an Incomplete Rendering Sequence

If, for some reason such as a power outage or other interruption, you have to interrupt the rendering process, you can continue rendering at a later time using the **Incomplete Rendering Sequence** option. You can resume rendering an incomplete rendering sequence, rendering unfinished frames, and frames not yet started.

This is how to resume an incomplete rendering sequence:

1. Click on the **Options** button in the Render Setup dialogue box. This opens the **Options** dialogue box.
2. Select on the **Complete Unfinished Frames** option.

Adding a Time/Date Stamp

The **Time/Date Stamp** button in the Render Setup dialogue box stamps the hour, month, day, and year of the position of the Sun light type in the scene as it is rendered. This feature is useful for sending images to a client, or keeping track of projects. This option is available only with the SOFTIMAGE renderer.

When you click the **Time/Date Stamp** button, the Time/Date dialogue box appears. Select **Active** and pick the corner and the colour of text you want the stamp to be. When you render, this information shows up where you specified.

Running the Renderer from the Command Line

When you type **soft -R** instead of **soft** when starting SOFTIMAGE 3D, the standalone SOFTIMAGE renderer loads. This allows you to render an image without loading all of SOFTIMAGE 3D.

The following example shows a rendering of the scene **RENDER-test.1-0.dsc** in the **local** database. The **-j** option allows more than one computer to render the scene without having two computers render the same frame. The **-s** option specifies which frames in the sequence to render and the sequence step, bypassing the original setup.

```
soft -R RENDER-Test.1-0 -d local -j -s 10 352 1
```

The following example renders the latest version of the **Render-Plan3** scene, which is in the **final** database. The rendered frames are recorded in **/EX8a/RENDER/Plan3** and will be named **back3**. The rendering can be run on more than one computer because of the **-j** option. The ampersand (&) sends the process to the background so that you can use the current shell to work on other tasks.

```
soft -R RENDER-Plan3 -d final -L -j  
-o /EX8a/RENDER/Plan3 -n back3 &
```

Rendering in the Background

If you're using the standalone renderer (**soft -R**), one of its main advantages is that you can render in the background while you continue working.

There are two standard ways to send an IRIX process to the background. The first way is to pause the process by typing **Ctrl-Z**, then move the paused task to the background by entering the **bg** command. For example:

```
soft -R final_logo_job -d tutorials -L -s 1 450 1
```

^Z

bg

 The **Ctrl-Z** command can be useful to simply pause a rendering in progress so that you can do something else, such as perform IRIX file management, etc., without the huge overhead of a heavy rendering job using up all of the CPU cycles. To return the job to the foreground, enter the **fg** command.

The second way to send a process to the background is to add an ampersand (**&**) to the end of the command. For example:

```
soft -R final_logo_job -d tutorials -L -s 1 450 1&
```

These two techniques put the process into the background, and allow the current shell to be used for other things. The process continues to run until the window they are running in is terminated.

If you desire the process to continue after the window is terminated, or even after logging out, use the **nohup** command. To do this, create an alias such as:

```
alias doit nohup soft -R
```

To run your alias, use the following syntax:

```
doit filename >& logfile &
```

For example:

```
doit final_logo_job -d tutorials -L -s 450 1 >&  
log &
```

filename represents a scene and **logfile** represents the text file to which messages are redirected. It is important to redirect messages

to a file because messages are normally sent to the local window. However, if the local window does not exist because it has been terminated or if you have logged out, this will cause the process to stop.

Writing Rendering Scripts

Rendering scripts allow you to use C-shell scripts to do extra processes before, or after each frame. These are used with the **Pre-frame** and **Post-frame Script** options in the Render Setup dialogue box.

The advantages of writing rendering scripts are numerous because they automate tasks. Here are some of the advantages:

- They can be used to save disk space by dumping files.
- They can be used for processing and conversion.
- Post-frame can convert the file format as it runs.
- Once a file is rendered, you can use a script to start a composite using a standalone; for example, you can use them to start Painterly Effects as a standalone and then modify the images as they are rendered.

When writing rendering scripts, **Pre-frame** executes the specified C-shell before rendering an image and **Post-frame** executes the specified C-shell script after the rendering of an image. Note that arguments are passed to the program: **\$1** is the frame file name (**.pic** sequence name); **\$2** is the current frame number (**.pic** frame); **\$3** is the frame counter of the sequence; and **\$4** is the field. A C-shell script must start with **#!/bin/csh -f** to define it as a C-shell; otherwise, it is assumed to be a Bourne shell.



A script file needs to have execution permission. Type **chmod 777** and the file name to give the file all the necessary permissions.

The following is an example script of parameter passing.

If you have the following file:

```
echo $1 $2 $3 $4 $5
```

and you specify this file as a **Pre-frame** or **Post-frame** script in the Render Setup dialogue box, and then render in the database **/obladi** to the picture name **blada** from frames 1 to 3 without field rendering, you'll get this output in the shell from which you started SOFTIMAGE 3D:

```
/usr/people/you/obladi/RENDER_PICTURES/blada 101
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 2 1 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 3 2 1
```

If you render the same thing from frames 11 to 13, you'll get:

```
/usr/people/you/obladi/RENDER_PICTURES/blada 11 0 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 12 1 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 13 2 1
```

If you render the same file with field rendering on, with the dominant field even or odd, you'll get:

```
/usr/people/you/obladi/RENDER_PICTURES/blada 11 0 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 11 0 2
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 12 1 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 12 0 2
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 13 2 1
```

```
/usr/people/you/obladi/RENDER_PICTURES/blada 13 0 2
```

If you specify any parameters after the name of the file in the **Pre-frame** or **Post-frame** script text boxes, the script finds these at the beginning, not the end, of the variable list provided to the script. For example, if you specify the following path in the script text box:

```
/path/example_script hello
```

the example script gives:

```
hello/usr/people/you/bladi/RENDER_PICTURES/blada 101
```

```
hello/usr/people/you/bladi/RENDER_PICTURES/blada 2 1 1
```

```
hello/usr/people/you/bladi/RENDER_PICTURES/blada 3 2 1
```

If you specify the following path:

```
/path/example_script hello goodbye
```

the result is:

```
hello goodbye/usr/people/you/obladi/RENDER_PICTURES/blada  
1 0
```

```
hello goodbye/usr/people/you/obladi/RENDER_PICTURES/blada  
2 1
```

```
hello goodbye/usr/people/you/obladi/RENDER_PICTURES/blada  
3 2
```

The doesn't show the field number at the end of each line because the example script only prints out the first five arguments.

Rendering a Subregion

If you don't want to render your whole scene, you can render a portion of it, called a subregion. There are two ways to do this: by either choosing certain options in the Render Setup dialogue box (**Render** menu cell) or by using the standalone renderer (**soft -R**).

Using the Render Menu Cell

If you want to render a subregion using the **Render** menu cell in the Matter module:

1. Choose the **Render** menu cell in the Matter module.
2. In the Render Setup dialogue box, select the **Subregion** option to click the **Set** button. The subregion only needs to be defined once, since the Preview and Render processes share the same subregion definition.
3. In the Define Subregion dialogue box, select the exact area to be rendered by entering either the pixel coordinates or by typing the coordinates as percentages.

For example, if you want to test some material parameters, but only want to preview the area affected by change and don't want to redefine the subregion for each test, you can set the pixel or percentage coordinates for this area.

Note

The pixel values are dependent on the resolution. For example, if you set the subregion area with pixel coordinates and the resolution changes (becomes larger or smaller), the area that is rendered still uses those pixel coordinates. However, the percentage values are independent on the resolution. If you set the subregion area with percentage coordinates, the pixel values change as the resolution changes (if you set the percentage values to 10 percent, you always see 10 percent of the image being rendered).

4. You can also draw the subregion to be rendered with the cropping rectangle by selecting the **Draw Always** option. When you select this option, the cropping rectangle is displayed in window B, the default Perspective window. If another view is selected in window B (such as Right, Front, etc.) when you chose the renderer, SOFTIMAGE 3D switches it to the Perspective window.

5. Resize the red rectangle to indicate the subregion you want to render by clicking on a corner or side and drag the mouse. You can move the rectangle by clicking inside of it and dragging the mouse.
6. Then click the **Render Sequence** button to render.

Using the Standalone Renderer

To render subregions using the standalone SOFTIMAGE renderer, use the following format:

```
soft -R <scenename> -n output\  
-b <distance from left edge>  
    <distance from right edge>  
    <distance from bottom edge>  
    <distance from top edge>
```

The subregion values should be in the range [0, 1] and be a percentage of image resolution. They must overlap by 1 percent because of the antialiasing of edges. To do this, use Adaptive supersampling; (see *Antialiasing* on page 39). Also, note that the values saved in the **SETUP_SOFT.sts** file work differently.

The following example renders four quadrants (whatever the resolution may be):

```
soft -R test.2-0 -n a -b 0.00 0.51 0.49 1.00  
soft -R test.2-0 -n b -b 0.49 1.00 0.49 1.00  
soft -R test.2-0 -n c -b 0.00 0.51 0.00 0.51  
soft -R test.2-0 -n d -b 0.49 1.00 0.00 0.51
```

The following example renders nine quadrants:

```
soft -R test -L -n a -b 0.00 0.34 0.65 1.00  
soft -R test -L -n b -b 0.32 0.67 0.65 1.00  
soft -R test -L -n c -b 0.65 1.00 0.65 1.00  
soft -R test -L -n d -b 0.00 0.34 0.32 0.67  
soft -R test -L -n e -b 0.32 0.67 0.32 0.67  
soft -R test -L -n f -b 0.65 1.00 0.32 0.67  
soft -R test -L -n g -b 0.00 0.34 0.00 0.34  
soft -R test -L -n h -b 0.32 0.67 0.00 0.34  
soft -R test -L -n i -b 0.65 1.00 0.00 0.34
```

Later, you can simply composite the rendered images by defining the full resulting resolution. The **composite** standalone uses the information in the “comments” area of the picture file. A sample composite script exists for nine quadrants resulting in a 1270 x 714 image. Execute this in the directory where the picture files reside.

```
composite -S 1270 714 final -d -v -s 1 1 1 a b c d e
f g h i
```

Showing and Hiding Object Edges

Hiding Edges

The **Edge_Flag → Hidden/Rect hidden** commands in the Matter module allow you to define one or more edges of a polygon mesh object as hidden when using the Hidden Line renderer. The hidden edges are hidden regardless of the position of the object.

1. Select a polygon mesh object.
2. Choose the **Edge_Flag → Hidden/Rect hidden** command.
3. Pick the edges individually or perform a rectangular selection using the left mouse button.

The edges are highlighted in cyan. The middle mouse button deselects the edge, and the right mouse button toggles.

4. Press Esc to end the mode.

Showing Edges

The **Edge_Flag → Visible/Rect visible** commands in the Matter module allow you to define one or more edges of a polygon mesh object as visible when using the Hidden Line renderer. The visible edges are visible regardless of the position of the object.

1. Select a polygon mesh object.
2. Choose the **Edge_Flag → Visible/Rect visible** command.
3. Pick the edges individually or perform a rectangular selection using the left mouse button.

The edges are highlighted in yellow. The middle mouse button deselects the edge, and the right mouse button toggles.

4. Press Esc to end the mode.



The **Show → Edge Flags** command also shows or hides visible or hidden edges. For curves and surface objects, these are defined automatically upon creation.

C H A P T E R T H R E E

Advanced Rendering

Introduction

In addition to the basic rendering process, there are some rendering features available that are not necessary for doing a basic render, however, they can produce some useful effects. With such rendering effects as raytracing, antialiasing, field rendering, or motion blur you can produce effects that make your animation look much more realistic, and give it a polished edge.

- Raytracing lets you render a scene with more realistic and precise details, giving you a high quality render. You need to use raytracing to calculate reflections, refractions, and shadows.
- Antialiasing is a method of smoothing out and sharpening rough or fuzzy edges of graphics to produce a more polished look. This is done by a mathematical process that subsamples pixels.
- Field rendering is used to reduce the strobing effect on fast moving objects for rendering to video.
- Motion blur defines the relative blur of a moving object, usually for special effects and for fast moving objects with lateral motion.

You can also render the Tag and Z channels of an image to use tag channel and depth information, which is useful when compositing and image-processing.

Details for each of these options is described on the following pages. Each of these can be accessed from the Render Setup dialogue box.

Raytracing

Raytracing calculates the light rays that are reflected, refracted, and obstructed by surfaces. When rendering with raytracing, you will have more realistic and precise results. However, it takes much longer to render with a higher raytracing value.

If you have a scene with reflection, refraction, transparency, or shadows in it, it will take a fairly long time to render. The beauty of raytracing is that you can render what you want: if you want to see only the effects of the animation, or just the effects of reflection, you can deselect all of the other features and show only these options. This will take less time to render, yet you benefit from the realistic results that rendering with raytracing offers.

If there is no reflection or refraction in your scene, raytracing occurs at about the same speed as hardware rendering.

The following illustrates when you would want to use raytracing using the SOFTIMAGE renderer.

1. Select an object.
2. Choose the **Material** menu cell in the Matter module. In the Material Editor, select a glass material and apply it to the sphere.
3. Increase the Reflection and Transparency values.
4. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
5. Select SOFTIMAGE as the rendering type.
6. Click the **Options** button. In the Options dialogue box, set the **RayTraced Depth** to 3 or 4. This means that the light rays will bounce around the scene this many times.
7. Specify the sequence by setting the **Start** and **End frame**, and the incrementation step.
8. Click **Render Sequence** and look at the results of rendering with raytracing.

Note

If you want to put restrictions on the rendering process, such as only rendering with the reflection turned on, choose **Preview → Setup**. Make sure **Global On** is selected and choose which other parameters you want to have selected, such as 2D Textures, reflectivity, Shadows, etc.

Raytracing in SOFTIMAGE 3D

SOFTIMAGE 3D raytracing uses a modification of the Glassner/Kaplan spatial subdivision algorithm. In preprocessing the database, first the bounding box (containing every object in the scene) is found. This box is then divided in half on the x-axis. Both of the boxes are then divided in half on the y-axis. The resulting four boxes are each divided in half on the z-axis. These eight boxes are divided in half on the x-axis... ad infinitum. This subdivision is halted when either:

- A box contains less than the preset number of triangles (triangles per leaf)

or

- The number of divisions is greater than the preset maximum (maximum tree depth).

In the rendering, the ray is passed from box to box until it hits something. At each box, the ray must be tested against all contained triangles. This triangle intersection testing is relatively slow compared to the calculation of the next appropriate box, so it is to your advantage to have a small number of triangles in each box. However, as the number of triangles grows smaller, the amount of memory required to hold all the boxes increases, as does the time taken at the preprocessing stage.

Triangles per Leaf

Triangles per leaf allows you to optimize the raytracing process by limiting the number of triangles per leaf that are rendered. Object surfaces are divided into triangles to simplify rendering.

The smallest practical number for the **Triangles per Leaf** option is 10. You may need to increase this if the program runs out of memory (that is, you get an error message like `xsplrit: malloc failed`).

Max Tree Depth

The **Max Tree Depth** option defines the limit on the number of subdivisions. It is necessary to prevent infinite recursion in the preprocessing stage. Think, for instance, of the top of a sphere where 40 triangles touch. If the **Triangles per Leaf** is set to 10, the program keeps on subdividing space around this point, trying to find a box that contains only 10 triangles and failing. The **Max Tree Depth**

makes the software stop subdividing even though there are more than 10 triangles in the box.

Finding the proper **Max Tree Depth** is more of an art than a science. Empirical studies suggest that this is just beyond the point where the number of leaves begins to decrease. To find this point, set the **Max Tree Depth** to some large number (such as 50), select the **BSP Tree** statistics in the Render Setup dialogue box, and run the program (it is not necessary to render a complete picture; the program can be stopped when the first scan line appears). Exit the program and examine the “stats” file. There should be a relatively obvious point at which the number of leaves begins to decrease. Set the **Max Tree Depth** to this number plus 1.

This is not an exact method, but by experimenting with the number of **Triangles per Leaf** and **Max Tree Depth**, it is quite possible to find a combination that results in a faster rendering time.

Reflection Mapping with Raytracing

Objects that are both reflective (have a **Reflective** value of more than 0 in the Material dialogue box) and have an image assigned as a **Reflection Map** (in the 2D Texture dialogue box) use reflection mapping. This takes the 2D image and wraps it around an infinitely large sphere.

A reflection map without raytracing ignores the environment in which an object exists. The 3D environment is not reflected on the reflective object, only the reflection map picture will be. A reflection map *with* raytracing reflects the 3D environment with the reflection map behind it. If your reflective object is in a closed environment (in a room) and you use reflection map with raytracing, only the room is reflected; the room would hide the reflection map behind it. If you need to reflect both the enclosed environment and the reflection map, you should render twice and composite both layers.

For more information on reflection mapping, see *Creating a Reflection Map* on page 79 of the *Defining Materials and Textures User's Guide*.

Antialiasing

Aliasing usually occurs when there is limited pixel resolution. 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.

The number of pixels in a scene depends on the screen resolution. The greater the resolution, the greater the number of pixels. The smaller the resolution, the smaller the number of pixels.

The more pixels there are, less aliasing occurs and the edge is smoother. When there are not enough pixels, you need to use antialiasing to make the lines look smoother.

 **Tip** Avoid adding antialiasing when you render a texture map that fills up the screen because 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 with the SOFTIMAGE renderer, follow these steps:

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select the SOFTIMAGE Rendering type. To use antialiasing with the mental ray renderer, see page 71.
3. Specify the sequence by setting the **Start** and **End frame**, and the incrementation step.
4. Select **Antialiasing**. The Antialiasing dialogue box appears where you can select either the Bartlett or the Adaptive supersampling option as antialiasing method.

To help you make a choice on the type of antialiasing you should use, here's an explanation of the differences between Adaptive supersampling and the Bartlett filter.

Bartlett

Bartlett is a static oversampling algorithm based on a variable width box filter. As you increase the filter level, more of the adjacent subpixel samples are taken into account while filtering a pixel. The expense of this algorithm is caused by all pixels being supersampled at the same rate (sample level) regardless of content affecting that

pixel; that is, a pixel with no geometry covering it gets sampled at the same rate as a pixel with lots of geometry. One side effect of this algorithm is due to using the box filter. As the sample rate (and thus the width of the box) increases, more subsamples are taken into account for a pixel; at high sample rates, this will sometimes lead to “soft” looking images, meaning there are no well-defined edges in the resulting image. Based on empirical data and practical limits of time, a sample level of 4 is about the maximum advisable value.

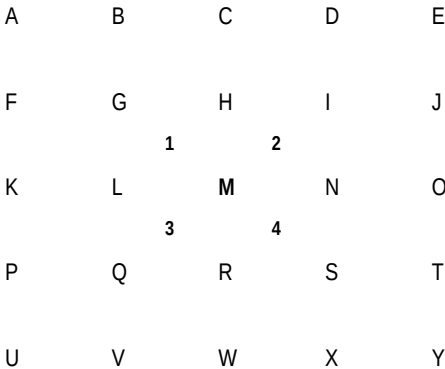
Adaptive

For Adaptive supersampling, the algorithm is intended to decrease the oversampling rate for those pixels that do not require it. This is based on the contrasted thresholds specified. First, subpixel samples are shot at the corners of each pixel. The variation in contrast is evaluated and, if required, the pixel is subdivided into four subpixels by sampling the full-sized pixel five more times, once in the centre and once in the middle of each edge of the pixel. From here, each of the four subpixels has the algorithm applied recursively, treating each of the four as if they were actually a full pixel. This process continues until the recursion limit is reached.

Visible Differences

You can see the difference between images filtered with the Bartlett or Adaptive algorithm. These differences are primarily from the number of subpixel samples shared between adjacent full size (result) pixels.

In the Adaptive algorithm, only the subpixel samples along the edges of the full pixel are shared between adjacent pixels (that is, the four corner samples are shared among the six adjacent pixels). Here’s a picture:



This matrix represents image pixels: 1, 2, 3, and 4 are the first level subsamples made for filtering pixel M. Samples 1 and 3 are also used in the filtering of pixels G, H, L, Q, and R. Thus only these subsamples are shared (along with any samples made between 1 and 3, are used for filtering pixel L, etc.).

In contrast, the box filter used in Bartlett can be viewed as a window that moves along the scan line being filtered. All subsamples that fall in the window are used to obtain the resulting pixel value for a given pixel on the scanline. Assume we are filtering area M again with a box filter that covers only two adjacent pixels: the letters now represent subpixel samples, not image pixels. The window will cover all of the subpixel samples A-Y, with M weighted most since it is central. Now move the window to the next sample to filter N: now samples A, F, K, P, and U are unused, but all the rest are used, as well as the samples that are outside the diagram past E, J, O, T, and Y. So you can see that subsample M still contributes to filtered pixels at both N and O. The subpixel samples have an effect on filtered pixels that are half the width away of the box filter.

Choosing an Algorithm

Results are usually a matter of preference.

The filtering effect of the Adaptive algorithm is slightly more localized than the Bartlett. The reduction of computing time is often found favorable and the filtering adequate with Adaptive. It only recursively subdivides at the sub-pixel level, so the image will remain crisper even at high filter levels.

If you don't mind slightly "softer" looking images, and can deal with the overhead of a static supersample, you will like Bartlett. Bartlett gives you a progressively "softer" image the higher the filter level because it is similar to blurring the image (more pixels are averaged into one).

Blurring a Moving Object

Motion blur allows you to define the relative blur of a moving object, usually for special effects and for fast moving objects with lateral motion.

Note Motion blur is used with animation rendered for film output to reduce the strobing effect.

You can use motion blur from the Render Setup dialogue box or from the command line using the **mb** standalone program.

The following illustrates how to apply motion blur to an object using the Render Setup options:

1. Select an object or objects.
2. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
3. Select the SOFTIMAGE rendering type.
4. Select **Motion Blur**. The Motion Blur dialogue box appears.
5. Enter the **Shutter Speed** which allows you to set the length of time the shutter is open. Values are measured in frames.
6. Enter the **Min. Movement** which sets the sampling rate while the shutter is open and the minimum amount of movement to be blurred. The value is measured in pixels. A low value increases the sampling rate and the memory required. The default value is 5.
7. Click Accept to return to the Render Setup dialogue box.

You can also use the motion blur with the mental ray renderer. This may be especially useful if you want to apply a motion blur to a single object in your scene so it is much quicker. For more information on using mental ray, see *Blurring a Moving Object Using mental ray* on page 74.

Using the mb Standalone

You can also create motion blur for objects in a scene by using the **mb** standalone.

The following example applies a backward motion blur of 80% on the **test** files, included in the **final-roger.3.0** scene which is in the **ROGER** database. The processed images (numbered from 1 to 98) will be named **testmb**.

```
mb /usr/softimage3.5/rsrc final-roger3.1-0 -b -c  
.8 -d ROGER -n test -N testmb -s 1 98 1
```

This example applies a backward motion blur of 100% on the **saloon** images, numbered from 325 to 1082. Those images come from the last version of the scene **test-saloon**, which is in the **GEORGETTE** database. The processed images will be saved in **/EX8a/RENDER/saloon**, and will be named **saloonmb**.

```
mb /usr/softimage3.5/rsrc test-saloon -b -L -d  
GEORGETTE -n saloon -N saloonmb  
-p /EX8a/RENDER/saloon -s 325 1082 1
```

For more information on the command line syntax for this standalone, see the **mb** description in the *Standalones* HTML file on the *On-line Documentation* CD.

How Motion Blur Is Calculated

SOFTIMAGE 3D makes a copy of each object being blurred, and each copy is updated at a different position in time in the animation. As the number of requests for time steps increases, more object copies are stored into memory which places high memory demands on the system.

Note

Since most people use RAM to capacity, motion blur usually uses memory swap space for its calculations.

Tips for Using Motion Blur

Half Blur

You can choose to activate or deactivate the blur effect in front of or behind an object. To produce a more traditional animation, you can control the blur motion by activating its effect only behind the moving object. Half Blur is available in only the **mb** standalone.

Alpha Channel

By default, resulting images will also have a blurred alpha channel. If the sequence of animation is not to be composited, it is possible to turn off the blurring of the alpha channel to reduce the time required to finish blurring the sequence.

Depth Computation

To reduce computation time, you can disable use of the depth information in the blurring process. This is only recommended when the objects in motion do not cross paths.

- Motion blur is extremely handy because the colour element can be rendered first. Make sure to render colour elements intended for motion blurring using only small amounts of antialiasing, or no antialiasing at all. This is because motion blur masks the aliased edges in the resulting image anyway.
- Motion blur used on 3D procedurally texture-mapped objects produces poor results.
- Motion blur has no effect on the apparent motion caused by camera movements. This is due to the number of moving objects that would over-extend the limits of the buffer memory.



If you use field rendering, you can produce a smoother effect because you are doubling the number of frames by 2.

Memory Usage

Calculating motion blur can be very memory intensive. When the message “malloc failed, reduce memory demands” is displayed, it indicates that animated objects have exhausted the memory of the system and that all available RAM and swap space have been exceeded. This situation can be solved by using the following techniques:

- Limit rendering to one object at a time, and once completed, composite the rendered objects together. Applying motion blur to a single object usually achieves the desired effect.
- Reduce the physical size of the objects being blurred because the number of replicated objects is a combination of object size, rendered resolution, shutter speed, and the minimum movement values.

- Render the objects requiring motion blur at half the original resolution. Enlarge the objects with the **zoom** standalone to return objects to their original size, and then composite them back over the rest of the scene.
- Reduce or eliminate shadows, reflection, refractions, and transparency in scenes because they use large amounts of memory and reduce rendering speed.

Note

Usually, when the error message “malloc failed, reduce memory demands” appears, the only way to clear memory is to exit the software and retrieve the scene; otherwise, results that reflect the modified parameters may not take effect.

Even if physical memory limits are not reached, certain scenes may still lack memory to complete rendering tasks. This is because of the method the system uses to calculate its memory allocation before a task is executed. The system calculates an estimated value of required memory, and does not proceed if that estimate exceeds the system’s current memory capacity.

Limitations with Motion Blur

Extreme Rotation or High Curvature Movement

Objects that show extreme rotation or movement over a small number of frames will show some perceptible artifacts. For example, one rapidly rotating cylinder end placed at a perpendicular angle to the camera will produce unrealistic motion blur. The true motion would show the moving end of the cylinder sweeping a smooth arc between the three position (X-1, X, X+1). This algorithm, however, will draw a straight lines between the three positions, giving a more angular appearance to the resulting blur.

First and Last Animation Frame

Since the motion blur algorithm bases its evaluation on the previous and the next frames, a conflict arises since no previous frame exists before the first frame and no next frame exists after the last frame in an animation sequence. A solution is to add an extra frames to the fcurve at the beginning and one at the end. For example, for an animation that initially runs from frame 1 to 100, you would extend the fcurve to include frame 0 and 101. This will provide the algorithm the necessary position data to evaluate the blur motion to include frame 1 and 100 of the animation.

Rapid Object Growth and Scale

Motion blur's ability to interpret the one-to-one pixel correlation is lost if an object grows rapidly relative to its frame size or if an object is scaled up over a small number of frames. This loss is because that the geometry covers fewer pixels at frame X than at frame $X+1$. Without the pixel correlation information, the algorithm produces a visible artifact that appears as "tears" along the edge of the blurred object.

Inaccessible Data

Since motion blur uses image-based information, inaccuracies can occur from loss of data as objects enter and exit the frame. This is because the object position data that originates outside the frame is inaccessible to the motion blur's algorithm.

Transparent Objects

Transparent objects may cause unexpected artifacts to appear in blurred images. Scenes that contain moving transparent objects passing in front of static objects create visible artifacts. The pixels associated with the moving object are blurred even though the colour contribution of those pixels is primarily from the background object. Conversely, if a moving object passes behind a static transparent object, the portion of the object visible through the transparency will be blurred incorrectly.

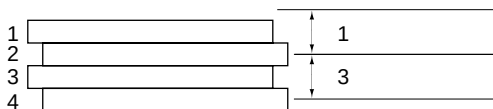
Antialiasing/Inaccurate Colour Choices

An artifact will result from antialiasing that is used to create the original in blurred images. When the colour of a pixel is derived mainly from the background colour with a small colour contribution from a moving piece of geometry, the blur algorithm incorrectly includes the background colour as part of the moving geometry pixels.

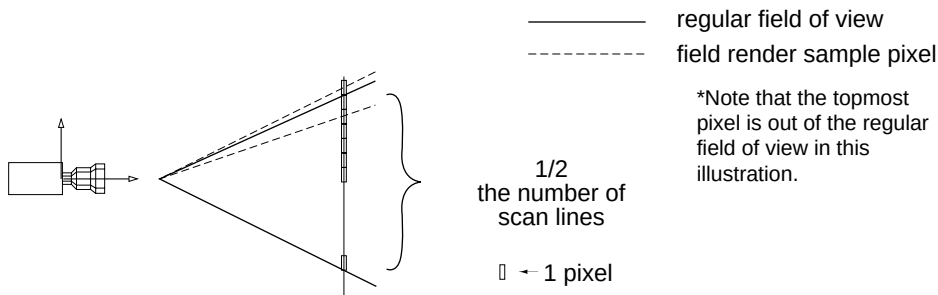
Field Rendering

Field Rendering is used to reduce the strobing effect that results from fast moving objects when rendering to video. If you are rendering to film, refer to motion blur (see page 42).

SOFTIMAGE 3D uses a wide-pixel rendering technique. Internally, the camera's field of view is perceived as being half as high, but each scan line is amplified to twice its thickness. The sampling of the pixel, therefore, is estimated over a larger area than normal. The camera first considers the even and then the odd lines. It is raised or lowered to ensure that the two different images have the correct visual orientation. The direction of compensation depends on whether your dominant field is set to even or odd. The following figure illustrates the use of the wide-pixel rendering technique.



The following example shows how the scene is viewed by the camera at render time. The direction of compensation corresponds to your dominant field setting (even or odd). Each line is scanned in at 50% wider than the normal sample line to ensure correct picture proportions.



There are two versions of the post-frame script file: inter01 and inter02. File version inter01 corresponds to a dominant field setting of 2,1 and inter02 is for a 1,2 setting. Select and use the post-frame file version that corresponds to your dominant field setting. These files ensure that merging occurs after each full frame (two fields) has been processed. The post-frame script files invoke the standalone

program called **interleave**. The script files are located in `/usr/softimage/3D/tools/scripts/inter01`.

Note The `inter03` and `inter04` scripts perform the same function as `inter01` and `inter02` respectively, except that they remove the separate image fields after the composite frames have been created.

To use field rendering with the SOFTIMAGE or mental ray renderer, follow these steps:

1. Choose the **Render** menu cell in the Matter module.
2. In the Render Setup dialogue box, select either SOFTIMAGE renderer or mental ray as the rendering type.
3. Select the **Field Rendering** option. The Field dialogue box is displayed.
4. Select the **Active** option, then select the **Even** or **Odd** option to correspond to your Dominant field environment.
5. Click the Accept button.
6. Enter a **Post-frame** script in its text box, or click the **Post-frame** button to open a browser in which you can locate the `/usr/softimage/3D/tools/scripts/inter01` file which is used to combine the rendered frames. Select the post-frame script file `inter01` or `inter02` that corresponds to your video editing environment. You may want to copy this file to your working directory.

Note If the fields fail to combine after they have been rendered, verify to ensure that the appropriate ASCII file has been entered in the **Post-frame** text box and not mistakenly placed in the **Pre-frame** text box.

Pre and Post-Frame Scripts

These scripts allow you to use C-shell scripts to do extra processes before or after each frame.

Arguments:

\$1 = frame file name (.pic sequence name)

\$2 = current frame number (.pic frame)

\$3 = frame counter of the sequence

\$4 = field

Creating a Setup

These two steps will help you recognize the four possible resulting field combinations produced by the post-frame script files:

1. Select the appropriate post-frame script file to suit your dominant field setting.
2. Render a single frame from a merged sequence and record a few seconds on tape. A cube moving over 10 frames with a stationary patch sphere was used for this test. The cube is used to verify correct field order, and the sphere is used to verify environment type.

Compare your results with the following:

- A normal image where the correct field setting and appropriate post-frame script were selected.
- If the cube appears to jump back and forth, the incorrect field setting and inappropriate post-frame script were selected. To view this clearly, use the VTR in slow frame-by-frame motion. Correct this by changing your dominant field setting and using the alternate post-frame script file to reverse the order of the frames when combined.
- If the cube appears broken at the top and bottom pixels, but the motion is smooth, an incorrect dominant field setting and correct post-frame script were selected. Correct this by changing your dominant field setting.
- If the sphere is stable and the motion is not smooth, a correct dominant field setting and incorrect post-frame script were selected. Correct this by using the alternate post-frame script file.

Note

The Avanzar video output board has the ability to read and combine the fields as it records to tape. This may eliminate the need to use post-script files, unless you need to evaluate the quality of the combined frame or have access to the flipbook display on the workstation.

Tip

It is recommended that NTSC users set the Dominant field to ODD in the Render Setup dialogue box, and save the Setup File with the **Preferences → Setup File → Save** command. By including the -s option at the end of the soft alias string, this feature is automatically set to the correct

position for each new session. The Picture format and other preview and rendering parameters will also be saved. Look at the ASCII file to verify the various parameters available.

Rendering Tag and Z Channels

Tag and Z Channels provide additional channel information to increase flexibility and enhance compositing and image-processing results. You can use Z Channels with the **composite** standalone program and Tag Channels with the Painterly Effects program for post-processing.

Tag Channels

The information contained in the Tag Channels consist of 1-bit stored layers that are identified by using the **Select → Set Named Selection** command specifying a name, and selecting the User Tag option. The rendered file has the form of `<name>.<frame number>` with a `.tag` extension instead of the usual `.pic` extension.

User Tag

Items to be included in a user tag selection. Tagged objects are rendered into a separate file which contains a mask showing all pixels in the image affected by the tagged model.

Note

You can define multiple objects within each Tag layer, but since there is no depth information, Painterly Effects considers the whole layer as one section. Only one pixel is designated as the “tagged” pixel. Assign objects that require a specific effect to a separate layer. In a case where two different scene images (layers) have the same tagged pixel, the topmost layer will be used.

Using the Tag Channel Option

With the Tag Channel information, post-processing utilities (such as Painterly Effects) can isolate pixels associated with a given object and process only those pixels. For example, a post-processing effect such as glow can be used to affect a specific part of an image or the entire image. Without the tag information, the post-processing utility would have to either locate the edges of a specific object or apply the effect to the entire image.

The following example using three cylinders describes the typical procedure to follow when using Tag Channels:

1. Create a “target” using three different-sized and different-coloured cylinders.

2. Superimpose all three, placing the smallest one in front.
3. Select the second cylinder, then choose the **Select → Set Named Selection** command.
4. In the **Set Named Selection** dialogue box, select the User Tag option, then select the Create button.
5. Repeat the same operations for the third and largest cylinder. Notice that you now have two user tag items in the Schematic window.
6. Choose the Render menu cell and render your picture with the **Render Tag Channel** option selected.
7. Exit SOFTIMAGE 3D and open Painterly Effects.
8. Load your image and select an effect (such as Rough Pastels), then click on the Tag Channels button.
9. Select Use Single Tag Channel and enter the number of the Tag Channel to affect using the Rough Pastels effect (1 or 2).
10. Select Ok and preview.

You will notice that the effect affected only the object with the Tag Channel 1. If you add a ripple effect on Tag Channel 2, it will only affect the cylinder with Tag Channel 2.

Z Channel

The Z channel information allows for more advanced compositing operations. Z channel provides depth information so that you can position an object in front of and in back of the background image. One useful application would be to use the Z channel information to allow an object in a scene to interact with a background image. Without the Z channel information, the compositor can only decide which layer should be placed on top, confining selected objects to the front of the background image.

Z channel information is also useful when compositing a SOFTIMAGE 3D scene with particles created in the SOFTIMAGE® Particle program.

Using the Z Channel Option

The following example of a computer-generated airplane flying around a scanned image of a building describes the typical procedure for using Z Channels:

1. Simulate the Z Channel for the background image using the Rotoscopy view mode to create a simple model of the building.
2. Choose the **Render** menu cell and render this model with the **Render Z Channel** option selected.
3. Rename the **.Zpic** file created using the same base name as the building image (such as **stuff.Zpic** or **building.Zpic**.) to inform the compositor that the building image has an associated Z Channel.
4. At this point, you need only ensure that the path of the airplane passes behind the simulated position of the building, and then composite.

The resulting animation will show the airplane passing behind the static image of the building.

Limitations

The quality of the Z compositing resolution diminishes when you physically intersect any object. This is because the polygonal triangle of one object is overlapped by the object placed before it. The program averages and filters the colour pixel information, but cannot do the same for object location in space. Consequently, the resulting image will appear non-filtered at object intersection points. Presently, it is impossible to antialias depth information due to the program's single sample per pixel calculation formula.

Rendering Faces of an Object

By default, only the front faces of an object are rendered; that is, the ones whose normals are facing the camera. The back faces are “culled” (ignored). If you like, you can render all faces of an object. This could be useful if you are rendering something like a flag waving in the wind, in which case you would want both sides of the flag to be rendered.

The faces that are rendered are defined by the **Back Culling** option in the Render Setup dialogue box.

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select the SOFTIMAGE renderer as the Rendering type. To use back culling with mental ray, see *Rendering Faces of an Object* on page 78.
3. Click on the **Options** button.
4. Select **Back Culling** to render only the front face (the default), or deselect it to render all faces.

C H A P T E R F O U R

Rendering with mental ray

Introduction

The mental ray renderer is a high-quality, photo-realistic renderer available in SOFTIMAGE 3D. This renderer allows you to perform many special effects as part of the rendering process instead of having to create these effects in the scene itself, which keeps your scenes as small as possible. This feature of mental ray also lets you save the settings you used during one rendering process and use them in another one.

mental ray gives you more versatility, better image quality, and the ability to run on a network or on a multi-processor machine. Another great aspect of mental ray lies in its open architecture which gives you the option of writing and applying custom shaders.

Taking Advantage of a Network

The mental ray renderer allows several imaging possibilities which were not previously attainable by the SOFTIMAGE renderer. One of these is the capability to increase the speed of the rendering process. Through network rendering, you can use other machines licensed with SOFTIMAGE 3D to capture unused processor power. To do this, simply create a file called `.rayhosts` in the home directory. This file needs to contain the names of the other machines, each written on a separate line. There should be a noticeably decreased time in the rendering of each frame.

As each frame is rendered, tiles appear within it. These tiles are the rendered portion of the scene sent back from the host machines. The trade-off of parallel processing is that the scene information must be sent to each of the host machines. This makes network rendering a useful choice when the time necessary to render the frame is actually greater than the time required to load and dispatch the scene data to the rendering servers.

Custom Shaders

The mental ray renderer provides an extensive set of built-in functions and can be dynamically linked with user-defined shaders during the rendering process. You don't have to use shaders with mental ray, but there are many different types of shaders you can use to create procedural textures (including displacement maps), materials, camera lenses, atmospheres, light sources, etc.

SOFTIMAGE 3D supplies sample shaders of each type, but you will probably want to use mental ray to its full advantage and create your own shaders (see the *mental ray Programmer's Reference Guide*). You can use any combination of these shaders to achieve specific results, such as using a lens flare lens shader for the camera with a “star” shader for the light.

Setting Up

The installation program sets all the paths and environment variables. The following steps are necessary for SOFTIMAGE 3D to fully function with mental ray options, however, they are automatically specified when during a standard installation procedure:

1. Set the environment variable of **mental ray** in your **.softimage** file to point to the renderer. This can be entered using a text editor. Open the **.softimage** file and type **SI_MI_TRACER** (as it is shown), and then the correct path in which the mental ray executable resides. Save the file.
2. Edit the **DatabaseSys.rsrc** file which supports the reference to the multiple shader libraries. Using a text editor, specify the name of the database which contains the shaders. On the next line, type the full path in which the shader descriptors reside. Finally add a third line by typing **TYPE SHADERLIB** (as it is shown) to indicate that the database contains shader descriptors rather than regular SOFTIMAGE 3D scene data.

The **DatabaseDir.rsrc** file supports references to multiple shader libraries. A shader library is specified by the following syntax in this file:

DATABASE [*name*]

PATH [*path*]

TYPE SHADERLIB

For example:

DATABASE playtime

PATH /usr/people/elmo/shaders

TYPE SHADERLIB

When you open a Database browser in SOFTIMAGE 3D, you can see that the shader libraries are identified with “Shd” at the end of the file names.

Setting Up for Single and Multiple Objects

The mental ray renderer traces the entire scene like other rendering software. However, instead of treating every object with the same ray information, object “flags” can be set to ignore verbose rendering information of specified objects. This means that you can use mental ray’s options for only the objects that you choose, which can make the rendering process go much quicker.

Render Options for Single Objects

To use mental ray on a single object:

1. Select a single object in the scene (this must be a 3D object such as a polygon mesh or patch or NURBS surface object).
2. Choose the **Info → Selection** command. The Info dialogue box for the object type you selected appears.
3. Click the **Render Setup** button in this dialogue box.
4. In the mental ray Render Setup dialogue box that appears, set the parameters for the object, such as motion blur, visibility, and surface approximation (see *Options for Selected Objects* on page 61).

Render Options for Multiple Objects

To use mental ray for more than one object on a scene:

1. Select multiple objects with the **Multi** menu cell active.
2. Choose the **Info → mental ray** command. This displays the mental ray Propagation dialogue box.

This box is identical to the mental ray Render Setup dialogue box as previously described except for two modes for editing the parameters: the **Edit fields values** option lets you edit the state of the selected parameters, but the **Select/unselect propagated fields** mode is the default. This mode allows you to select only those parameters that you want to apply to the selected models.

With the **Select/unselect propagated fields** mode selected, all the options will be dimmed. However, you can select any options you want to use by clicking on them (you can deselect them in the same way).

The options in this dialogue box are “smart” because they edit only the parameters which relate to a particular type of object. If there is a

patch object and a polygon mesh object in the selected group, the Propagation parameters edited will affect only the objects to which they are applicable.

3. Select the **Select/Unselect propagated field** option for the Edit mode. Pick the fields you want to edit and they become unghosted.
4. Change the parameters that you want (see the next section), and click the **Apply on Selected** button. This applies the parameters set to the selected objects, but leaves the dialogue box open.
5. Click Exit to close the dialogue box.

Options for Selected Objects

If the final rendering of the scene will use mental ray, these parameters should be taken into account throughout the creative process.

Visibility of Objects

The three Object Visibility options determine if an object is visible when rendered with mental ray. You can also choose to have a visible shadow and reflection of that object. These options can be selected individually or combined to achieve the desired effect. The default is that all three options are selected.

- **Primary rays** refers to all rays emitted from the camera. It allows you to specify the visibility of an object in the scene. When this option is selected, the object is visible.
- **Secondary rays** allows you to turn an object's reflection in a mirrored surface on or off. When selected, a reflection is visible.
- **Shadows** allows you to turn an object's shadow on or off. When selected, a shadow is cast.

For example, if you create two characters (an angel and a devil) and make them both walk past a mirror, you can set the Object Visibility parameters so that only the angel is visible to the camera, but the shadow and the reflection in the mirror belong to the devil. To do this, select *only* the **Primary rays** option for the angel, and select *only* the **Secondary rays** and **Shadows** options for the devil.

For information on how to create shadow objects, see *Creating Shadows* on page 17 of the *Defining Materials and Texture's User's Guide*.

For information on using the **Secondary rays** option for creating a reflection map, see *Creating a Reflection Map Using mental ray* on page 80 of the *Defining Materials and Texture's User's Guide*.

Motion Blur

The motion blur options allow you to specify varying degrees of motion blur for a selected object. For information, see *Blurring a Moving Object Using mental ray* on page 74.

Surface Approximation

For information on Surface Approximation, see *Surface Approximation* on page 80.

Overview for Using mental ray

The mental ray renderer is used like any other renderer, but has a many more options that can be set, which are not all found in one place. This outline shows you where to find these new options and describes how to use them to achieve the rendering results you want.

Tip

The first step in using mental ray is to inspect the scene to be rendered. From the observations you make doing a triangle count and checking material assignment, you can evaluate how to best use the various options of mental ray.

1. Create the scene or object you want to render.
2. If you select a single object to render, choose the **Info → Selection** command. If you select multiple objects in the scene to render, choose the **Info → mental ray** command (see page 60 for both these).
3. With the **Light → Define** command, you could set the Area lights geometry options to bring more realistic effects to light sources. The area lights model the light source as a piece of geometry, which allows soft shadows to be created (see page 19 in the *Defining Materials and Textures User's Guide*).
4. You can also choose shaders for the different effects that you want, such as camera lens shaders, light shaders, material shaders, volume shaders, shadow shaders, and textures (2D and 3D) shaders. You can use any combination of shaders to achieve specific results, such as using a fish-eye lens shader for the camera with a cloud volume shader for the atmosphere.
 - Choose the **Camera → Settings** command to access lens shaders.
 - Choose the **Light → Define** command to access light shaders.
 - Choose the **Material** menu cell to access material, volume, and shadow shaders.
 - Choose the **Texture → 2D Global/2D Local** command to access 2D texture shaders.
 - Choose the **Texture → 3D Global/3D Local** command to access 3D texture shaders.
 - Choose the **Atmosphere → Depth-Fading** command to access volume shaders.
 - Choose the **Render** menu cell and select mental ray **Options** to access output shaders.

5. Choose the **Render** menu cell, select mental ray as the rendering type in the Render Setup dialogue box, and set all the **mental ray** rendering options as described in this chapter.

Previewing with mental ray

If you want to preview your image in mental ray, you must change the Preview Setup. This also sets up for previewing from within the Material, 2D Texture, and 3D Texture dialogue boxes, which is necessary if you want to preview the effects of shaders from within these dialogue boxes.

1. Choose the **Preview** ➔ **Setup** command in the Matter module.
2. Select **mental ray** as the Preview Renderer.
3. Click Ok.

Using Shaders

Shaders are the key to adding special rendering effects to SOFTIMAGE 3D. A shader is a simple program written outside of SOFTIMAGE 3D and then accessed through the mental ray interface in SOFTIMAGE 3D. A shader is an opening in the architecture of mental ray which lets you program rendering variables. Instead of choosing a pre-programmed shader, you could write your own shader to achieve quality improvements, performance optimization, or simply create a rendering effect not available in the shaders which accompany SOFTIMAGE 3D.

Once compiled, the parameters of the programmed shader can be edited as easily as editing the standard shaders. For more information, see the *mental ray Programmer's Reference Guide*.

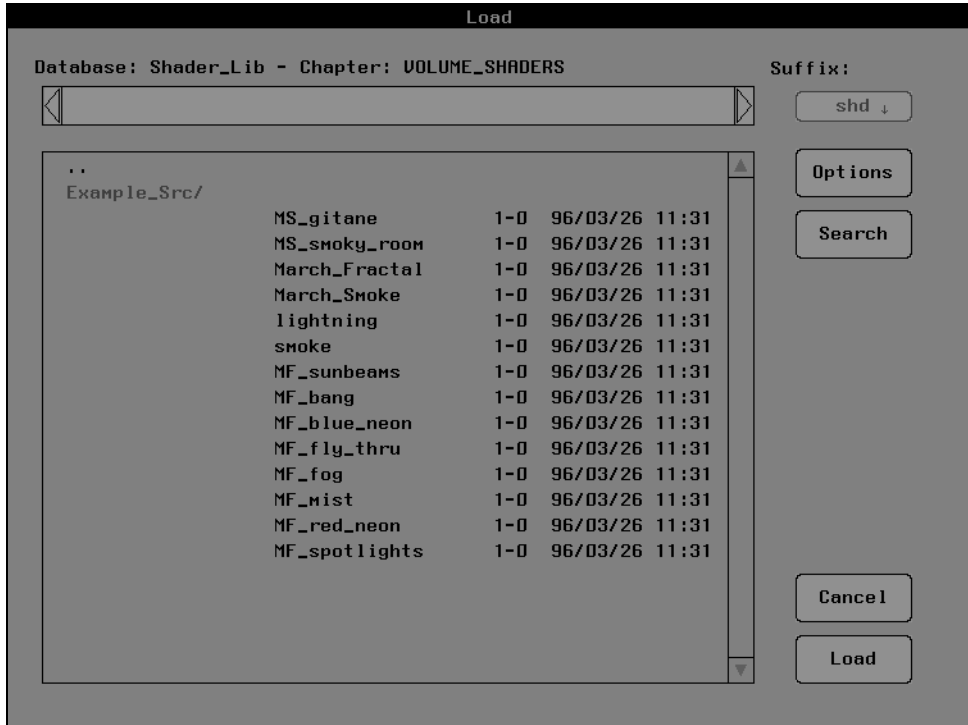
SOFTIMAGE 3D ships with a variety of shaders. There are several areas in its interface for applying a shader, depending on the type of shader it is. There are shaders for material, volume, shadows, 2D and 3D textures, camera lenses, and lights, and atmosphere. The output shaders perform post-processing on the rendered image.

- Choose the **Camera** → **Settings** command to access lens shaders.
- Choose the **Light** → **Define** command to access light shaders.
- Choose the **Material** menu cell to access material, volume, and shadow shaders.
- Choose the **Texture** → **2D Global/2D Local** command to access 2D texture shaders.
- Choose the **Texture** → **3D Global/3D Local** command to access 3D texture shaders.
- Choose the **Atmosphere** → **Depth-Fading** command to access volume shaders.
- Choose the **Render** menu cell and select mental ray **Options** to access output shaders.

This is a general procedure for selecting a shader once you have chosen the appropriate command. The exception for this is output shaders:

1. To select the shader to apply, **Shader** option in the mental ray area of the dialogue box you have open.

- The browser opens to the chapter for that type of shader appears. In this example, it is the VOLUME_SHADERS chapter that is automatically loaded.



- Select one of the shaders and click the **Load** button. Its name appears in the text box below the shader type's name in the main dialogue box.
- To edit the shader's parameters, highlight the shader's name in the text box and click on the **Edit** button. The dialogue box showing all parameters for that shader appears.
- If you want to save the shader by another name and then edit the parameters, you can modify its name in the text box. This creates a new shader with the new name, but with the current parameters. You can then edit the new shader's parameters as described above.

The new shader can then be saved and recalled by name in other scenes.



Before previewing, go into the Preview Setup dialogue box and select **mental ray** as the rendering type.

Linking a Custom Shader to a Script File

You can enter commands for the custom shader linking into the mental ray file so that you don't need to modify script files manually to include link, code, and \$include commands for non-standard shaders. The syntax of the commands must be exact.

To link a custom shader, follow these steps:

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select mental ray as the Rendering type.
3. Click the **Options** button. The mental ray Options dialogue box appears.
4. Click the **Edit** button next to the **Default Link/Code** button to display the Custom Script Line dialogue box.
5. If you want to edit the command line, click the **Edit** button inside this dialogue box. The **Custom Script Lines** dialogue box is displayed again. Make any changes, and click Ok to exit the dialogue box.
6. The **Add** button lets you add a new command to the script. Click the **Add** button and the **Append New Line** dialogue box appears. You can add a maximum of 50 command lines.
7. Enter the new command line and click Ok to Accept and return to the **Custom Script Lines** dialogue box.
8. If you want to delete a command line from the script, click the **Delete** button.

Raytracing Using Mental Ray

Raytracing calculates the light rays that are reflected, refracted, and obstructed by surfaces. When rendering with raytracing, you will have more realistic and precise results. However, it takes much longer to render with a higher raytracing value.

If you have a scene with reflection, refraction, transparency, or shadows in it, it will take a fairly long time to render. The beauty of raytracing is that you can render what you want: if you want to see only the effects of the animation, or just the effects of reflection, you can deselect all of the other features and show only these options. This will take less time to render, yet you benefit from the realistic results that rendering with raytracing offers.

If there is no reflection or refraction in your scene, then rendering occurs at about the same speed as hardware rendering.

Each refraction or reflection of a ray creates a new branch of that ray when it bounces off a solid object and is immediately cast in another direction. 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.

To use raytracing with mental ray, follow these steps:

1. Select an object. If it has some reflection and transparency, the effects of raytracing are more obvious.
2. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
3. Select mental ray as the Rendering type.
4. Click the **Options** button. In the Options dialogue box, specify the **Ray Depth** parameters.

The **Ray Depth** option allows you to 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. The various branches of rays in a scene constitute a *ray tree*.

- **Reflected ray depth** allows you to specify the maximum number of a ray's reflective 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.

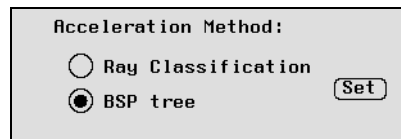
- **Refracted** ray depth allows you to set the maximum number of times a ray can be refracted in a scene.
- **Sum** adds the total number of a ray's reflections and refractions in the scene. If the total exceeds the number you have specified, the ray is not cast.

If the **Sum** exceeds the value specified, the ray 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.

5. Click Accept when you have set the values and then render the scene.

Acceleration Method

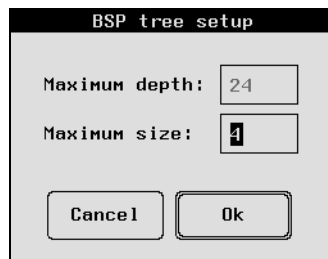
When choosing an acceleration method for using raytracing with mental ray, it is wise to return to the findings that you made when you initially evaluated your scene. The two choices are **BSP tree** and **Ray classification**.



A general rule for choosing between these methods is the overall complexity of the scene. If the scene information shows that there are less than 150,000 triangles, then the acceleration method should be **BSP tree** (Binary Space Partitioning).

BSP Tree

With **BSP tree**, the scene is divided into cubes to reduce the number of computations. Click the **Set** button beside this option to open the BSP tree setup dialogue box in which you can set the maximum depth and size.



A **Maximum depth** can be set to accelerate the renderer by limiting the level of space subdivision that occurs. The default for maximum depth is 24.

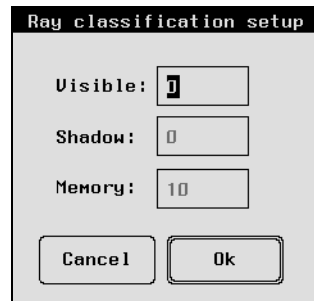
In conjunction with the maximum size of the BSP tree, the rendering process can be adjusted more finely to limit the number of triangles calculated in each cube by setting the **Maximum size**. The default for the size is 4 (triangles).

Ray Classification

The other method of acceleration, **Ray classification**, deals with larger scenes more efficiently than BSP tree. It provides better management of memory consumption and can help avoid massive memory swapping. Ray classification exploits the coherence of the rays.

Ray classification checks for intersections between every solid object to define refraction or reflection.

Click the **Setup** button beside this option to open the Ray classification setup dialogue box. In it, there are three variables of optimization to increase the speed of the ray classification.



The **Visible** option specifies the number of divisions a ray will make in order to accurately describe the object's surface.

The **Shadow** option allows you to specify the shadow ray space division.

The **Memory** option is a safety device that allows you to specify the maximum amount of memory to be used for the data structures. The default value is 10 megabytes.

Antialiasing Using mental ray

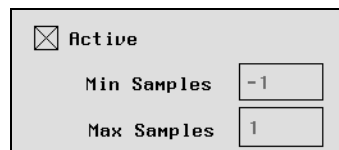
Aliasing usually occurs when there is limited pixel resolution. Antialiasing is a method of smoothing out and sharpening rough or jagged edges of images to produce a more polished look. It uses a mathematical process that subsamples the pixel area.

The number of pixels in a scene depends on the screen resolution. The greater the resolution, the greater the number of pixels. The smaller the resolution, the smaller the number of pixels.

The more pixels there are, less aliasing occurs and the edge is smoother. When there are not enough pixels, you need to use antialiasing to make the lines look smoother.

To use antialiasing with mental ray, follow these steps:

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select mental ray as the Rendering type.
3. Select the **Antialiasing** option to open the mental ray Antialiasing dialogue box. When the **Active** option is selected, the parameters affect the rendering of the scene.



4. Set the **Min Samples** and **Max Samples** values. This describes the number of samples taken to compare surrounding colour values which are averaged to define the colour of a pixel. The default settings for **Min Samples** and **Max Samples** are -1 and 1, respectively.

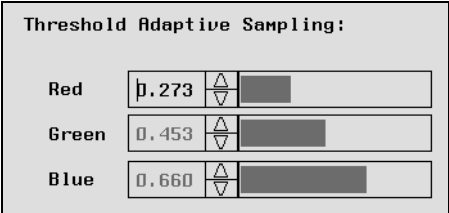
If the minimum value is zero, the pixel is sampled at least once. The default minimum of -1 means that the picture will be subsampled every four pixels (a 2 pixel-by-2 pixel square).

If the default sampling is used, what happens? mental ray 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 tries advancing the level of samples. Now the new sampling examines pixel 1, pixel 2, and pixel 3, looking again at the contrast level.

If the contrast is still too high between pixels, mental ray does a third pass dividing each pixel into four subpixels within itself.

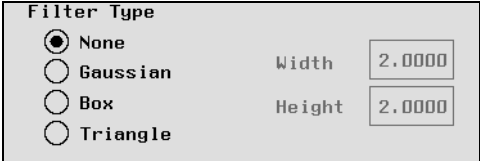
If this still does not meet the supersampling threshold, aliasing appears because the limit of three levels of sampling has been executed as the maximum. If this is the case and more antialiasing is required, you must change the minimum or maximum values to modify the limit.

5. Set the **Threshold Adaptive Sampling** sliders. These describe the contrast level variable for deciding if another level of sampling will occur.



The smaller the value for each colour (R, G, B), the greater 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.

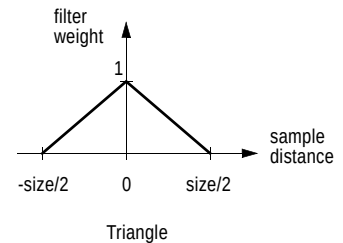
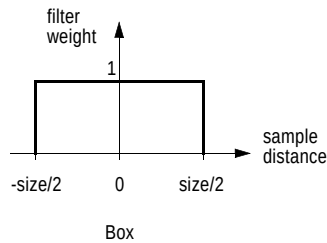
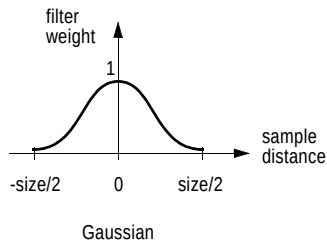
6. Select the **Filter Type**. The sampling procedure can be complemented in post-processing to ensure even more antialiasing. Each of the three filter types (Gaussian, Box, and Triangle) process subsamples, surrounding and including the pixel which is being rendered, by using the height and width of 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 **Width** and **Height** options for each of these types define the size of the filter to be applied.

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

- The Gaussian filter uses a sloped curve weighting the sampling gently at the top of the peak and towards 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 which affects the pixels by the least filtering happening at the edges of the sampled area.



Blurring a Moving Object Using mental ray

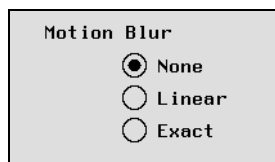
Motion blur can be used for special effects and on fast moving objects with lateral motion moving away from to toward the camera. Motion blur with mental ray uses raytracing to correctly blur attributes applied to moving objects such as highlights, shadows, and reflections as well as refractions and intersecting objects.

Because you can apply a motion blur to a selected object in your scene, it can make rendering go much quicker than when using the SOFTIMAGE renderer.

Motion blur must be set up for the selected object in both the mental ray Render Setup dialogue box (accessed from the **Info** → **Selection** command) and the Render Setup dialogue box (accessed from the **Render** menu cell) for the effect to work.

To use mental ray on specific objects in a scene, follow these steps:

1. Select an object or objects in a scene.
2. Choose **Info** → **Selection** for a single object or **Info** → **mental ray** for multiple objects (click the **Render Setup** button in the Info dialogue box).
3. Select the **Motion Blur** option in the appropriate dialogue box: **Linear** or **Exact**.

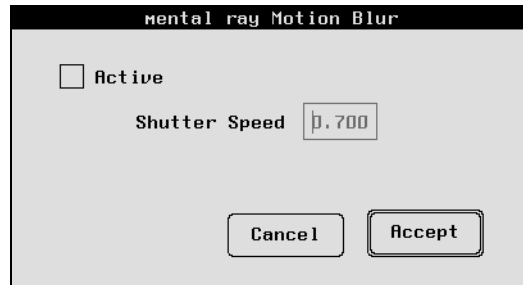


If you want the selected object to have a motion blur based only on a translation keyframe, select the **Linear** option. If the object is animated only in translation, using **Linear** can actually be an optimization since only one motion vertex is used for the whole object. **Linear** is mandatory for objects whose topology is not consistent throughout the animation. This affects objects such as meta-clay and animated Booleans.

If the desired effect requires motion blur by animation of rotation, scale, cluster, shape, lattice, or materials, select the **Exact** option.

4. Click Ok to exit the dialogue boxes.

5. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
6. Select mental ray as the Rendering type.
7. Select the **Motion blur** option. The mental ray Motion Blur dialogue box is displayed.



8. Select the **Active** option to activate motion blur, then set the shutter speed value. **Shutter Speed** allows you to set the length of time the shutter is open. A value of zero automatically turns all motion blur calculations off. The default value is 0.7, with the range of acceptable values between 0 and 1. The value is in time, emulating the shutter on a camera being open for a percentage of time between frame x and $x + 1$. A large number means a slower shutter speed.

Example

A few modifications have to be made to any object that you want to have motion blur. This is an example of blurring two primitive objects:

1. Get a default Dodecahedron and an Icosahedron.
2. Select the Dodecahedron and choose **Boolean** → **Static**. Click Ok in the dialogue box to confirm the other parameters. Select the Icosahedron to create a union between these two objects.
3. The **Boolean** → **Static** command creates a completely new object, so, you must hide the original objects by choosing **Display** → **Hide** → **Unselected**.
4. Choose **Polygon** → **Automatic Colourize** in the Matter module.
5. Choose the **Material** menu cell and change the shading model to Lambert.

6. Select the new “jewel” object and choose **Info → Selection** to open the Polygon Info dialogue box.
7. Change the **Automatic Discontinuity** value to 44, then click the **Render Setup** button and change the **Motion Blur** option to **Exact**. Click Ok to accept the set parameters in the **mental ray** Render Setup dialogue box and click Ok again to exit the Polygon Info dialogue box.
8. Choose the **Render** menu cell. In the Render Setup dialogue box, select mental ray as the Rendering type, and then select **Motion Blur**. In the mental ray Motion Blur dialogue box, select **Active** and set the **Shutter Speed** value to 1.
9. Click Accept to accept these changes and exit the Render Setup dialogue box.
10. In the playback box, change the Start frame to 0 and the End frame to 30.
11. Spin the object: save a rotation keyframe at frame 0 using **SaveKey → Object → Rotation → All**. Then go to frame 30, rotate the object 180 degrees and save another keyframe.
12. Choose the **Render** menu cell again. Set a resolution of at least 200 pixels.
13. Set the mental ray **Antialiasing Max Samples** to at least 3. Lesser values will have a coarse result and probably won't look very realistic.
14. Render the sequence using mental ray.



To get rid of the noisy look, increase the antialiasing level and/or add a Blur function from the Antialiasing menu in the Render Setup dialogue box.

Displacement Mapping

Displacement mapping allows you to move an object's vertices so that during the rendering process, the object's geometry is altered to create a bumpy surface. The **Roughness** slider in the 2D Texture File dialogue box is used to adjust the degree to which the object's geometry is modified by the **Displacement (mr)** option.

Displacement mapping only alters the object's geometry in the rendered image and not the scene, so you can create highly complex objects without having to actually model them.

To use displacement mapping, you can try the following example. There is also more general information on it and bump mapping in *Bump Mapping* on page 75 in the *Defining Materials and Textures User's Guide*.

This example creates an asteroid.

1. Choose the **Render** menu cell in the Matter module. The Render Setup dialogue box is displayed.
1. Select mental ray as the Rendering type and click **Accept**.
2. Get a B-spline patch sphere and make it 4 steps by 4 steps.
3. Choose the **Info → Selection** command and in the Info dialogue box, click the **Render Setup** button.
4. Change the **Surface Approximation** to Spatial, keep the default value of 5, and select **Pixel length** to define the units. Click Ok in both dialogue boxes.
5. Choose the **Material** menu cell in the Matter module and select the Lambert shading in the Material Editor dialogue box.
6. Choose the **Texture → 2D Global** command in the Matter module. In the 2D Texture File dialogue box, click the **Select** button beside the Picture Filename text box. In the Picture library, under the SURFACES directory, select the **moon** picture, and click **Load**.
7. Select UV as the Mapping Method.
8. Select the **Displacement (mr)** option above the **Roughness** slider. Adjust the roughness with this slider to 0.5, and click Ok to accept the texture parameters.
9. Choose the **Render** menu cell again and render frame 1.

Rendering Faces of an Object

By default, only the front faces of an object are rendered; that is, the ones whose normals are facing the camera. The back faces are “culled” (ignored). If you like, you can render all faces of an object. This could be useful if you are rendering something like a ribbon or pages in a book, in which case you would want both sides of the object to be rendered.

In mental ray, you have the option of rendering the Front, Back, or Both faces. The default is Both.

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select mental ray as the Rendering Type.
3. Click the **Options** button. The mental ray Options dialogue box appears.
4. Selecting a **Face** allows you to specify which polygons are to be rendered. For example, if you select Front only the polygons facing the camera are rendered.

If the back faces of the object are not influential in the scene, then you could select only the **Front** face option. However, to ensure the highest quality image, perhaps for the final render, you should select the **Both** option. You would also select Both if you are rendering a grid (such as a flag) where you need to see both sides of the object.

Rendering Shadows

For render tests, you can turn off all reflectivity, refraction, or textures. You can make sure that no shadows are cast for objects. It is advantageous to be able to turn off shadow components because they are time-consuming to render. When you're ready to see the reflectivity, refraction, textures, or shadows again, you can select them to make your scene more realistic looking.

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select mental ray as the Rendering type.
3. Click the **Options** button. The mental ray Options dialogue box appears.
4. Select **Trace** to activate the raytracer and/or **Shadow** to create shadows. The raytracer is automatically selected if a lens shader is active.

A set of global switches for raytracing parameters is also located here.



Note

For information about creating shadow objects with mental ray, see *Creating Shadow Objects* on page 21 in the *Defining Materials and Textures User's Guide*.

Surface Approximation

It is possible to reduce the number of triangles in the geometry of an object and still render a very smooth surface. Dealing with large models while using parallel processing can slow down the time to distribute the information as well as occupy unnecessary swap space.

To set the surface definition for an object:

1. Select the object.
2. Choose the **Info → Selection** command and click the **Render Setup** button.
3. In the mental ray Render Setup dialogue box that appears, you can decrease the step size, and the number of triangles decreases drastically.
4. Select a type of surface approximation: **Static** or **Adaptive** (see the next section for more information).

Depending on the surface approximation you choose, mental ray tries to create the perfect curve between two points.

5. Set 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.

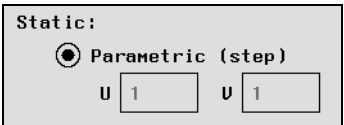
There can be a minimum (**Min**) amount required guaranteeing a smooth curve, and a maximum (**Max**) amount guaranteeing the utmost limit of computation.

6. Click Ok in this dialogue box and the next to save your changes.

Choosing an Approximation Method

Static

The **Static Parametric** method of surface approximation defines the U and V steps of the selected object as seen by the geometry.

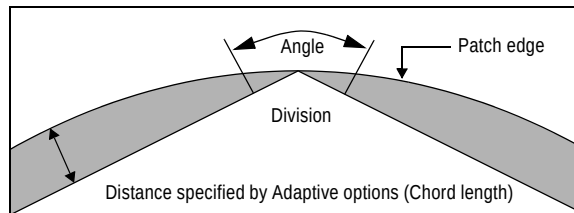


Changing these values also change the number of steps in the object's Info selection dialogue box. These parameters remain the same throughout the animation and if the camera gets very close to the object, the steps might be seen.

This is the same as the method of surface approximation used by the SOFTIMAGE renderer.

Adaptive

The **Adaptive** method of surface approximation interprets the size of the object in proximity to the camera and adapts the number of steps to account for the size in the render.



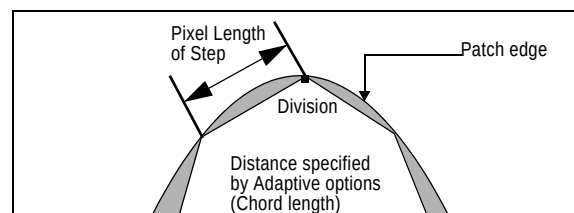
This is the best method for surface approximation for both NURBS and patch surfaces.

The size of the step can be measured in either the number of pixels in the length of the step or the SOFTIMAGE 3D units, defined by the grid setup.

Adaptive surface approximation can be accomplished by one of two methods: **Spatial** or **Curvature**.

Spatial

The **Spatial** surface method using the pixel length is more intuitive than using the system units for definition of the step's length. Within the object curve, the steps are as long as the number of pixels specified in the text box.

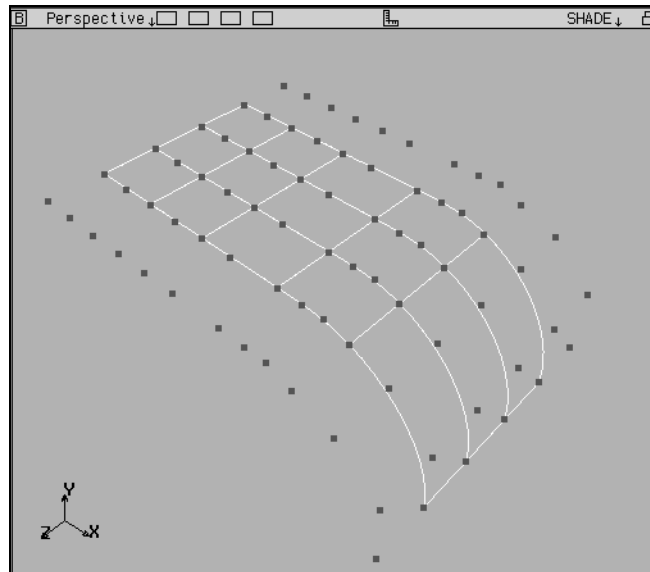


If the step is longer than the specified number of units, the steps are subdivided until they are equal to that number.

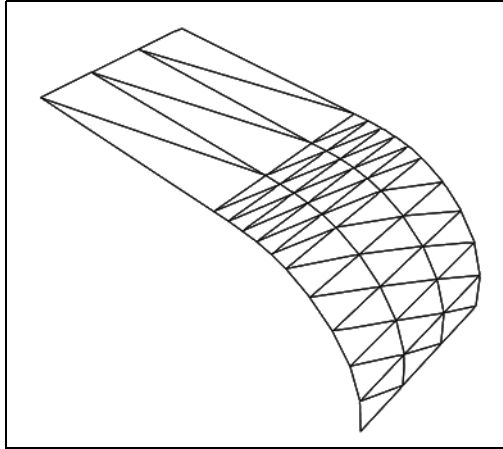
Curvature

The **Curvature** method chord length uses the distance between the most distant arc of a curve and the step as a parameter by which the steps will be subdivided. This method is more effective when dealing with larger flat areas which converge into curved surfaces.

It approximates the flat area with less tessellation than **Spatial** and approximates the curved area with more subdivisions of steps where necessary.



If the combination of the **Chord Length** and the **Angle** of the steps do not meet the parameters in the text boxes, the steps are subdivided.



Using Output Shaders



Output shaders operate on rendered images before they are written to a file, but after the normal rendering process is complete. They can perform operations such as filtering, compositing with other files, and writing to different file formats. The benefits of using output shaders is that the rendering time is usually very quick and can produce excellent results when used correctly. However, they have some limitations because they are simply added to the existing rendered image.

Before you choose an output shader, you can set up these options: **Contour Rendering**, the **Surface Normals**, and **Z Pic** information.

These options are available when you:

1. Choose the **Render** menu cell.
2. Select mental ray as the Rendering type.
3. Click the **Options** button.

The parameters are in the Output Shaders area of the mental ray Options dialogue box.

Setting Up to Use the Output Shaders

Using Contour Rendering

This method of rendering defines surfaces by contrast rather than by geometry and produces antialiased contour lines between surfaces as well as in areas where surface normals show discontinuity. Contour line rendering is particularly useful for cartoon animation because the contour lines that are produced look similar to line drawings of traditional animation, and the output can be painted cell by cell.

When you click the **OFF/ON** button beside the **Contour rendering** option, the Contour Rendering Parameters dialogue box appears:

- Selecting **Activate** makes the Contour Rendering option active.
- **Line Width** defines the width of the contour lines in pixels. This is 0.5 by default.
- **Contour Depth** is used to give a tolerance to distinguish between objects that are almost in the same position in space. This is 0 by default, which means it will distinguish between all objects.

Display is disabled during contour rendering, with the result written to a .pic file. This is the only output file generated.

Note

You cannot use contour rendering with the BSP tree acceleration method. If you do so, SOFTIMAGE 3D switches the Ray Classification method instead to render the contour images.

Surface Normals

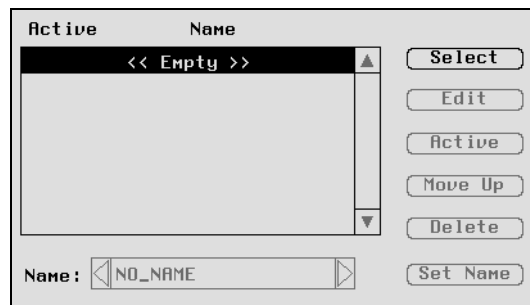
The **Surface Normals** option activates surface normal output and creates an output file with the .n extension. It will have the same resolution as the actual picture, except that instead of storing the RGB value for each pixel, it stores the surface normal used for rendering the triangle nearest the camera. Normal encoding is used mainly for post-processing applications.

Z pic

The **Z pic** option activates depth picture output and creates an output file with the extension .Zpic. This method is similar to Normal encoding, except that it is the distance between the camera and the nearest triangle that is stored instead of the RGB value.

Selecting an Output Shader

If you need to search for an output shader, click the **Select** button beside the Output shader scroll box and choose a shader from the appropriate chapter or database in the browser that appears.



- **Edit** displays the dialogue box containing the parameters that define the shader you have selected. You can then edit the parameters of that shader. For example, if you are using the same shader two or more times in the list, you can modify the parameters slightly for each time it is used.

- **Active** makes the shader active or inactive, depending on the current state of the shader. If a shader is active, it is highlighted.
- **Move Up** lets you rearrange the order of the shaders when you have more than one active shader in the list. Since you can have more than one lens shader active at the same time, it makes a difference how you order them in the list. The shaders are processed starting from the top of the list.
- **Delete** deletes the currently selected shader from the shader list.
- **Set Name** lets you change the name of a output shader. To change names, select a shader from the list, modify the name in the **Name** text box, and click **Set Name**. This creates a new shader with the new name, but with the current parameters. You can edit these parameters in the dialogue box that appears when you click **Edit** (see the previous description of **Edit**). The new shader can then be saved and recalled by name in other scenes.
- The **Name** text box lets you save a shader by another name (see **Set Name** above).

Example

1. Select an object or load a scene.
2. Choose the **Render** menu cell. In the Render Setup dialogue box, select mental ray as the Rendering Type and set the **Start** and **End** frames 1.
3. Click the **Options** button. In the Output Shaders area of the mental ray Options dialogue box, click **Select**.
4. In the browser that appears, go to the `/Shader_Lib/GLOWS` directory. Select **DGlow** as the Output shader and click **Load**.
5. With the **DGlow** shader selected, click **Edit**.
6. In the **OZ-Diffusive Glow Postfilter** dialogue box, click the **Select** button beside the **Object list** text box. The Object list allows you to apply the **DGlow** shader to any or all objects in your scene. Select the object you want and click **Ok** to exit the Object list.
7. Click **Ok** to exit the **OZ-Diffusive Glow Postfilter** dialogue box and then click **Accept** in the mental ray Options dialogue box.
8. Click **Accept** to exit the Render Setup dialogue box.
9. Choose **Preview** → **All** to view the results.

10. If you are satisfied, choose the **Render** menu cell. In the Render Setup dialogue box, name your scene appropriately, and then click **Render Sequence** to begin the rendering process.

Note You will notice that mental ray only applies the output shader to the specified object after it has completely rendered the scene.

11. To view the final result, choose the **Picture** menu cell in the Tools module. Use the browser to go to the RENDER_PICTURES chapter. Select your rendered scene and then click **Display**.

Saving to a File

You can write the commands received by the raytracer to a file instead of actually rendering the scene.

1. Choose the **Render** menu cell. The Render Setup dialogue box is displayed.
2. Select mental ray as the Rendering type.
3. Click the **Options** button. The mental ray Options dialogue box appears.
4. Select the **Output to File** option.
5. Enter the complete path name or a relative path name in the text box; the file will be written to the current working directory. This output can then be edited and manually sent to the mental ray renderer using the `irix` command.

If the **Output to File per frame** option is not selected, all frame are created in the same `.mi` file. By default, all information for the sequence of output frames are written into a single file. If the **Output to File per frame** option is selected, each frame in the sequence is written to a separate file.

Index

A

- Acceleration method in mental ray 69
- Adaptive antialiasing 40
 - mental ray 72
- Alpha channel and motion blur 44
- Angle of camera 5
- Antialiasing 35, 39
 - filters
 - adaptive 40
 - Bartlett 39
 - comparisons 40
 - mental ray 71
 - motion blur 46
- Aspect ratio of camera 6

B

- Back culling 54
- Background, rendering in 25
- Bartlett antialiasing filter 39
- BSP tree in mental ray 69

C

- Camera 4
 - angle 5
 - aspect ratio 6
 - depth of field 5
 - lens shaders 6
 - movement 7
 - projection 8
 - setting and resetting 4
- Colours with motion blur 46
- Command line
 - motion blur 43
 - rendering 24, 30
- Compositing 17
- Contour rendering 84

D

- Depth computation
 - mental ray 85

- motion blur 44

- rendering 52

- Depth of field 5

- Depth of trees 37

- Depthcue rendering 20

- Displacement mapping 77

- Dithering 35

E

- Edges, hiding and showing 32

F

- Faces, rendering 54, 78

- Field rendering 35, 47

- scripts 48

- Files

- .lin 14

- .mi 88

- rayhosts 57

- saving raytracer commands 88

- scripts 26, 48, 67

- .Zpic 85

- Filters, antialiasing

- adaptive 40

- Bartlett 39

- comparisons 40

- mental ray 72

G

- Ghost rendering 21

H

- Half blur 43

- Hidden-line rendering 20

- hiding and showing edges 32

- previewing 14

- Hiding edges for rendering 32

L

- Lens shaders 6

- Lights 3

- optimizing rendering time 17

- .lin files 14

M

Materials

- optimizing rendering time 16
- previewing 13

mb standalone 43

Memory

- guidelines 17
- motion blur 44
- requirements 18

mental ray renderer 19, 57, 63

- acceleration method 69
- adaptive supersampling 72
- antialiasing 71
- BSP tree 69

contour rendering 84

depth computation 85

displacement mapping 77

faces 78

motion blur 74

multiple objects 60

networks 57

optimizing time 17

output shaders 84, 85

previewing 64

ray classification 70

rayhosts file 57

raytracing 68

saving raytracer commands to file 88

script files 67

setting up 59, 61

shaders 57, 65, 67

shadows 79

single objects 60

surface approximation 80

surface normals 85

Z channels 85

.mi files 88

Modelling, optimizing rendering time 16

Motion blur 35, 42, 43

alpha channel 44

antialiasing 46

calculating 43

colour 46

depth computation 44

first/last frames 45

growth and scale 46

half blur 43

high curvature movement 45

inaccessible data 46

limitations 45

mb standalone 43

memory 44

mental ray 74

rotation 45

transparent objects 46

N

Network rendering 57

Normals 3

mental ray 85

O

Optimizing rendering time 16

Output shaders 84, 85

setting up 84

surface normals 85

P

Pre and post-frame scripts 48

Previewing 13

.lin files 14

mental ray 64

subregions 13

Projection, camera 8

R

Ray classification 70

rayhosts file 57

Raytracing 36

acceleration method 69

mental ray 68

reflection mapping 38

saving commands to file 88

Reflection mapping 38

optimizing rendering time 16

Rendering 21
 in background 25
 command line 24, 30
 contours 84
 depth information 52
 depthcue 20
 faces 54, 78
 field rendering 35, 47
 ghost 21
 hidden-line, see *Hidden-line rendering*
 mental ray, see *mental ray renderer*
 motion blur 42, 74
 optimizing time 16
 output shaders 84
 rotoscope 21
 scenes 21
 scripts 26
 sequences 19
 resuming 23
 SGI hardware 20
 shadows 79
 SOFTIMAGE 19
 standalone 24, 30
 subregions 29
 surface normals 85
 tag channels 51
 wireframe 20
 Z channels 52
Resolution 22
Rotoscope rendering 21

S

Saving raytracer commands to file 88
Scenes
 previewing 13
 rendering 21
Scripts
 field rendering 48
 linking to shaders 67
 pre and post frame 48
 rendering 26
Sequences, rendering 19
 resuming 23

SGI hardware renderer 20
Shaders 57, 65
 lens 6
 linking to script files 67
 output 84
Shading 3
Shadows, rendering 79
Showing edges for rendering 32
SOFTIMAGE renderer 19
Standalones
 motion blur 43
 renderer 24, 30
Subregions
 previewing 13
 rendering 29
Surface approximation 80
 adaptive 81
 curvature 82
 spatial 81
 static 80
Surface normals 3
 mental ray 85

T

Tag channels 51
Tessellation 3
Textures, previewing 13
Time, optimizing 16
Time/date stamp 23
Tree depth 37
Triangles 3, 37

U

User tag 51

W

Wireframe rendering 20

Z

Z channels 52, 85
.Zpic files 85

