

3Delight for XSI 1.1

RenderMan© production rendering for SOFTIMAGE|XSI

Short Contents

.....	1
1 Introduction	2
2 Installation	3
3 Features Summary	5
4 The User Interface	8
5 Rendering Guidelines	19
Credits	27
List of Figures	28
Concept Index	29

Table of Contents

.....	1
1 Introduction	2
1.1 Further Readings.....	2
2 Installation	3
2.1 Getting a License.....	3
2.2 Local Install.....	3
2.3 Network Install.....	3
2.4 Starting the License Server.....	4
2.5 The Settings File.....	4
2.6 Licensing More Than One Product.....	4
3 Features Summary	5
3.1 Integration with <i>XSI</i>	5
3.2 Core Features of 3Delight.....	6
3.3 Pipeline.....	6
4 The User Interface	8
4.1 Renderer Properties.....	8
4.1.1 Quality.....	8
4.1.2 Motion Blur.....	9
4.1.3 Depth Of Field.....	10
4.1.4 Optimization.....	10
4.1.5 Archives and RIB Export.....	11
4.1.6 Search Paths.....	12
4.1.7 Displays.....	13
4.1.8 Shadow Maps.....	13
4.1.9 Render Tree Conversion.....	13
4.1.10 Network Cache.....	14
4.1.11 Diagnostics.....	14
4.2 3Delight Attributes.....	15
4.2.1 Quality.....	15
4.2.2 Ray Tracing.....	15
4.2.3 Subsurface Scattering.....	15
4.2.4 Geometry.....	16
4.2.5 RIB Archive.....	17
4.2.6 RIB Box.....	17
4.3 RenderMan Shaders Properties.....	17

5	Rendering Guidelines	19
5.1	A Few Words About Tessellation	19
5.2	RIB Export and Import	19
5.3	Ray Tracing in 3Delight for XSI	20
5.4	Efficient Rendering of Hair	20
5.4.1	Render Settings	20
5.4.2	Quality Settings	20
5.5	Rendering XSI Curves	21
5.6	Using RenderMan Shaders	21
5.6.1	Connecting RenderMan Shaders	22
5.6.2	Accessing XSI Vertex Properties	22
5.6.3	Accessing XSI Render Channels	23
5.6.4	Writing Shaders for XSI Curves	24
5.6.5	Accessing Reference Geometry	25
5.6.6	Writing “Cluster Aware” Shaders	25
5.7	Supported XSI Shaders	26
	Credits	27
	List of Figures	28
	Concept Index	29

1 Introduction

3DELIGHT FOR XSI is a fully integrated rendering plug-in for SOFTIMAGE® | XSI®¹. It is based upon a heavy-duty, production ready and fully featured RenderMan®-compliant renderer: 3DELIGHT.

The primary aim of the product is to provide XSI users with realistic rendering solutions to some known issues with the internal renderer. This includes, among other things:

- Automatic conversion of XSI shaders into RenderMan® shaders.
- Fast 3-D motion blur and depth of field.
- Efficient rendering of “hair” and “fur”.
- Fast and accurate displacements.
- Fast and accurate rendering of smooth surfaces such as subdivision meshes, NURBS and curves.
- Beautiful anti-aliasing, as suitable for high-end film rendering.
- Efficient rendering of complex scenes.
- Complete geometry support: polygons, subdivision surfaces, NURBS, curves, particles.
- Advanced RIB export and import features.

The second aim is to introduce some rendering concepts and methodologies found in RenderMan® pipelines into the XSI world. More about 3DELIGHT FOR XSI features in [Chapter 3 \[Features Summary\], page 5](#).

1.1 Further Readings

This document contains everything you need to work with 3DELIGHT FOR XSI. For the interested, here is a list of additional readings:

3Delight User’s Manual.

A recommended reading for users interested about the internals of 3DELIGHT.

The RenderMan Companion, by Steve Upstill.

Although quite aged, this book is still a very good introduction² to the RenderMan® standard and “spirit”.

Advanced Renderman, by Anthony A. Apodaca and Larry Gritz.

A production oriented RenderMan textbook.

The RenderMan Specifications 3.3

An essential reading for serious RenderMan® practitioners and hobbyists.

¹ And to our knowledge, the first after XSI’s integrated renderer.

² And in author’s opinion, the best.

2 Installation

3DELIGHT FOR XSI comes as an XSI add-on and is thus installable as any other add-on. The only additional step is to start the license manager¹. If you have already a 3DELIGHT license server running, please refer to [Section 2.6 \[Licensing More Than One Product\]](#), page 4.

2.1 Getting a License

Please contact us at sales@3delight.com to get the 3DELIGHT FOR XSI package. The package is a standard XSI *add-on* that can be installed by a drag-and-drop of the uncompressed package onto the XSI window.

In order to obtain a license, you must send us the *hostid* of your machine, which you can obtain by going through the following steps:

1. Install the add-on
2. Open the *Plug-in Manager* window (*File -> Plug-in Manager*)
3. Right click on the 3DELIGHT FOR XSI add-on and select *Command Prompt*. This will open a command prompt with the current directory being the root of the add-on installation.
4. In the command prompt, issue the following command: `cd Application\bin\nt-x86`
5. And then issue: `licutils hostid`

This will give you a unique hexadecimal identifier of the current machine, you can then send us this number get your license. *Note that you have to run this on your license server machine.*

2.2 Local Install

If you are installing 3DELIGHT FOR XSI on your own machine and you plan to run the license server on the same machine, then the only thing to do is to install the license file ('`license.dat`') in add-on's '`/License`' directory and start the license server as explained in [Section 2.4 \[Starting the License Server\]](#), page 4.

2.3 Network Install

If you plan to install 3DELIGHT FOR XSI on a network (and you probably have a workgroup) then:

- Put the license file ('`license.dat`') in add-on's '`/License`' directory.
- Fill-in in the '`settings.txt`' files, as explained in [Section 2.5 \[The Settings File\]](#), page 4.
- Start the license server if not started yet, as explained in [Section 2.4 \[Starting the License Server\]](#), page 4.

¹ There is no license management in the |Demo version.

2.4 Starting the License Server

The license server binary (`licserver`) is installed in the `/Application/bin` directory of the add-on. There are two batch files to start the license server:

`licserver-foreground.bat`

Starts the license server in *foreground* mode. This is useful to see all the log messages and we recommend this when starting the license server for the first time.

`licserver-background.bat`

Starts the license server in *background* mode. This starts the license server and returns control to the user. This can be executed at startup time on the license serving machine.

The license server can also be started “by hand” if necessary (`licserver -h` for more infos) but we recommend using the handy batch files.

2.5 The Settings File

The settings file contains information about the location of the license manager² and is located in the root of the add-on directory. The standard settings file goes as follow:

```
# put the name of the license server here (if not your local machine)
/3dfx/licserver 127.0.0.1
```

```
# 3DFX can wait for available licenses if all are taken. Put "true"
# here if this is the wanted behaviour.
```

```
/3dfx/waitforlicense false
```

The following two lines are recognized:

```
/3dfx/licserver
```

Points to the machine that is running the license server. The default value means that 3DELIGHT FOR XSI will look for the license server on the local machine (putting *localhost* in this line is the same as *127.0.0.1*). Machines can be specified either by their name or by their IP address.

```
/3dfx/waitforlicense
```

Toggle this to `‘true’` if you want 3DELIGHT FOR XSI to wait for a license if none are available. The default action is not to wait and report an error.

2.6 Licensing More Than One Product

If you are running both 3DELIGHT and 3DELIGHT FOR XSI licenses at the same time, it is **important** to run only one license server. The license file should then contain two license keys, one for each product and both the `‘settinx.txt’` file (3DELIGHT FOR XSI) and the `‘renderman.ini’` file (3DELIGHT) should point to the same license server.

² Although it could be used for additional purposes in the future.

3 Features Summary

3.1 Integration with XSI

Particular care has been taken to ensure a seamless integration with XSI:

Full Render Region Support

Simply select ‘3Delight’ as the current renderer in XSI’s *render options* and launch renderings as usual. 3DELIGHT FOR XSI can render into XSI’s render region and frame buffer. All framebuffer channels are supported (such as depth, normals and alpha).

Complete *Render Tree* support

3DELIGHT FOR XSI automatically converts XSI’s render trees into RenderMan© shaders and compiles them on the fly¹. All textures are converted (if needed) into 3DELIGHT’s mip-mapped and tiled TIFF format for efficient rendering and filtering. Material clusters, vertex colors, implicit and explicit projections are all supported. The following material slots are recognized:

- Surface
- Displacement
- Bump
- Environment

Additionally, the conversion system has been designed in such a way so that advanced users can convert their in-house shaders to 3DELIGHT format easily.

NOTE: This means that XSI, in conjunction with 3DELIGHT FOR XSI, can be used as a RenderMan© shaders authoring tool.

XSI Shader Balls rendering

3DELIGHT FOR XSI will render XSI shader balls if the user enables this features in *File -> Preferences*.

XSI Properties Sharing

3DELIGHT FOR XSI accesses all the common rendering properties so that its behaviour is as close as possible to XSI’s internal renderer.

Rich Geometry Support

Here is a complete list of supported geometric primitives:

Polygons All types of polygons are supported. Additionally, polygonal geometry with a *subdivision step* greater than 1 will be rendered as a *smooth* subdivision surface. Edges and vertices tagged as “hard” are rendered properly.

NURBS All NURB surfaces are supported. Trim curves on surfaces are also fully supported and rendered to sub-pixel accuracy.

¹ Compilation is multi-threaded to take full advantage of multi-core machines.

- Particles** All particle types are supported with proper shading. Additionally, 3DELIGHT FOR XSI can take advantage of 3DELIGHT's *lightweight* particles to render extremely large clouds of small particles.
- Hair** Hair is rendered using 3DELIGHT's efficient **RiNuCurves** primitive. Linear, quadratic and cubic curves are all supported. Special RIB export options for hair enable memory efficient off-line rendering.
- Curves** 3DELIGHT FOR XSI can render XSI curves of any shape and degree.

NOTE: 3DELIGHT has no tessellation controls other than the **ShadingRate**. This means that, by default, all surfaces (hair, subdivisions, nurbs) are rendered as smoothly as visually required for the given resolution, *and this is the normal operational mode of 3DELIGHT*.

3.2 Core Features of 3Delight

Motion Blur and Depth of Field

Multi-Segment motion blur and realistic camera shutter simulation contribute to high quality rendered images. Compared to other rendering software, motion blur in 3DELIGHT FOR XSI is *fast*. Depth of field is fully supported and simulates a realistic camera *bokeh*.

High Quality Anti-Aliasing

Edge anti-aliasing, motion blur and depth of field quality are all controlled using very simple and *predictable* options. Contrary to other rendering software, increasing pixel samples for higher quality anti-aliasing does not affect performance significantly².

Geometric Displacements

Displacements are efficiently rendered to sub-pixel accuracy. XSI's displacement shaders as well as **RenderMan©** shaders can be used for displacement.

Subsurface Scattering

Automatic subsurface scattering can be enabled on a per-object basis and delivers impressive results, fast. All standard XSI materials can be used without modification to achieve subsurface scattering effects.

3.3 Pipeline

RenderMan© Shaders Support *[Studio]*

It is possible to connect **RenderMan©** shaders to geometry (or groups) and use them instead of XSI's materials. Surface, displacement and volume shaders are supported.

Scene Caching *[Studio]*

This features enables you to "cache" a scene for efficient re-rendering. Once a scene is cached, the plug-in doesn't have to parse it for subsequent renders. This

² By the principle of dissociation between shading and visibility in 3DELIGHT, as explained in 3DELIGHT FOR XSI *User's Manual*.

minimizes the time necessary to start the render of complex scenes, especially those with hair³.

RIB Export [*Studio*]

XSI scenes can be exported into efficiently constructed RIB files. RIBs can be monolithic or fragmented into smaller archives for memory efficient rendering. RIB can be outputted in binary and/or compressed format.

RIB Archive Import [*Studio*]

Individual objects or groups of objects can be replaced by RIB archives. *Delayed* RIB archives⁴ are also supported.

RIB “Boxing” [*Studio*]

The purpose of *RIB boxing* is to enable the insertion of **RenderMan**© commands during scene rendering. This is a common facility in **RenderMan**© oriented work flow.

³ This features is intended primarily for the lighting stage, where rapid feedback is of prime importance.

⁴ Mainly used in depth-complex scenes, for faster and memory efficient rendering.

4 The User Interface

3DELIGHT FOR XSI has only three interface pieces and those are well integrated inside XSI: the renderer properties, the object properties and the RenderMan© shaders properties¹.

4.1 Renderer Properties

Renderer properties are general properties that apply to the entire render engine. This UI piece is accessible through the *Render -> Render -> Renderer Options* menu².

4.1.1 Quality

Shading Rate

The *shading rate* is a fundamental quality control parameter and formally speaking, it defines the *rate* (or frequency) at which a surface is shaded. From *The RenderMan Companion*, page 214:

" ... [The Shading Rate] controls the frequency of shading a surface. *size* is specified as an area in pixels: if *size* is 1, a surface is shaded about once per pixel. Large values of *size* cause cruder but faster shading. "

In most micro-polygon driven implementations, including 3DELIGHT, the shading rate actually controls how finely surfaces are tessellated. Since the shading rate is specified in "area of pixels", micro-polygon renderers simply tessellate surfaces so that each micro-polygon is of the specified area. And since every micro-polygon is an *atomic unit of shading*³, we automatically obtain the desired shading rate. There is an important implication to this:

Higher shading rates (such as 20) not only make objects look blurrier (less frequent shading), but also make surface silhouettes look less smooth. This is because the shading rate and the tessellation are *interconnected* in a micro-polygon based renderer.

Note however that shading rate has *absolutely no effect* on edge anti-aliasing, which is controlled through the filtering parameters described next.

In practice, the following shading rate values are common:

0.1 to 1.0	For high quality renders. Even lower shading rates can be used for extremely detailed shaders or textures. Some shaders are not written with anti-aliasing in mind ⁴ and can exhibit artifacts (especially in animation), those also require low shading rates.
1.0 to 2.0	For medium quality renders.
> 2.0	Draft renders and shadow map renders.

Table 4.1: Common shading rate ranges.

¹ RenderMan© shaders properties are only available in 3DELIGHT FOR XSI | *Studio*.

² Note that 3DELIGHT has to be selected as the active renderer in the *Render -> Render -> Pass Options* in order to be accessible in the renderer options.

³ Contrary to ray-tracers, where the smallest unit of shading is a sub-sample.

⁴ Shaders generated by 3DELIGHT FOR XSI are properly anti-aliased.

Oversampling

The *oversampling* (or *supersampling*) defines the number of sub-samples (or sub-pixels) there is in each pixel and in each direction. This means that an *Oversampling* of 2 tells the renderer to use 4 (2x2) samples for each pixel. Oversampling has a direct impact on the following:

Edge anti-aliasing

The higher the oversampling, the lower the probability of aliasing on geometric edges and *sharp* shading variations.

*Motion blur quality**Depth of field quality*

The higher the oversampling, the smoother the motion blur and depth of field effect.

Filter Type

Specify what filter to use when combining the pixel samples together. The most useful filters are: ‘*sinc*’, ‘*mitchell*’ and ‘*catmull*’. ‘*box*’ filter is commonly used for shadow map rendering or when oversampling is disabled (*Oversampling* set to 1) ‘*gaussian*’ filter is good when softer results are desired but tends to be too blurry when *Filter Width* is higher than 2 or 3.

NOTE: Filter type does not affect rendering time. Filter settings that affect render time are the oversampling and filter width.

Filter Width

The width of the super sampling filter tells the renderer how many pixels to consider in the final result (also known as the filter *support*). It is well known that film-grade quality anti-aliasing can only be achieved using sharper filters (such as ‘*sinc*’, ‘*mitchell*’ and ‘*catmull*’) and wide filter supports. Values of 2 to 4 are common for draft renders and values from 6 to 8 are common in high-end final renders.

Note that because ‘*sinc*’ and ‘*catmull*’ filters have negative lobes, it makes no sense to use a filter width smaller than 4 for these; to better understand the problem please refer to *The RenderMan Companion*, page 176.

Table 4.2 gives general guidelines on the common filter parameter ranges for various render qualities. Additionally, please refer to Section 5.4 [Efficient Rendering of Hair], page 20 for the particular case of hair rendering.

Quality	Filter	Width	Oversampling	Shading Rate
Low	box	1	2	10
Low-Medium	gaussian	2	2	4
Medium	mitchell	3	3	2
Medium-High	mitchell	4-6	4	1
High	sinc/mitchell	8	8	0.5

Table 4.2: Filter settings guidelines.

4.1.2 Motion Blur

Motion Steps

Camera Specifies how many motion blur steps to use for the camera. It makes sense to increase this value for cameras that are moving rapidly and following a *non-linear* path.

Transformation Specifies how many motion blur steps to use for transformation motion blur. It makes sense to increase this value for rapidly *rotating* objects such as a car's wheel.

Deformation Specifies how many motion blur steps to use for deforming objects. Unless an object has an extreme deformation movement, this value should be set to 1. Having more deformation steps is more expensive (in terms of memory usage) than having more transformation steps since this implies geometry duplication inside the renderer.

Shutter Settings

Shutter

Delay

Shutter Efficiency

4.1.3 Depth Of Field

Enables Turns depth of field on or off.

fStop The *f-stop* of the camera being used. In photography, *f-stops* usually define discrete steps (because of click-stopped apertures), each step represents a halving of the light intensity from the previous stop. In 3DELIGHT an f-stop can be any scalar, so a more formal naming for this parameter would be *f-number*.

Focal Length Gives the focal length of the lens being used.

Focal Distance Gives the distance at which the surface appear in focus.

Extreme Motion Blur with DOF When rendering effects with extreme motion blur and depth of field, the supersampling as specified in *Oversampling* (see [Section 4.1.1 \[Quality\], page 8](#)) might not be enough to produce smooth images. This parameter lets the user control an alternate sampling algorithm that is well suited for this problematic case. The value specified here dictates the number of *iterations* of that specific algorithm and a value of 1 is generally enough to produce smooth results.

4.1.4 Optimization

Multi-Threading Specifies the total number of threads to use.

Tile Size A tile, or a *bucket*, is the smallest unit of rendered data that the renderer gives back to the user. From *3Delight's Users Manual*:

"This option specifies the dimension of a bucket (tile), in pixels. Using smaller buckets may reduce memory use at the expense of slight performance loss. In our experience, this default value (16) is the best compromise for 3DELIGHT."

IMPORTANT: Because of a performance problem of the render region in XSI versions prior to 6.5, using higher values for tile size will greatly accelerate the render *in the render region*. This problem should be fixed in newer versions of XSI.

Grid Size *Grids* represent small sets of micro-polygons that are shaded together in an SIMD manner. From *3Delight's Users Manual*:

During the rendering process, geometry is split into small grids of micropolygons. Shading is performed on one grid at a time. This option specifies the maximum number of such micropolygons per grid. The default value ensures that one grid covers approximately one bucket (tile). Since by default `RiShadingRate` (the *Shading Rate* property) is one and buckets are 16 by 16 pixels wide, the default grid size is $16 \cdot 16 / 1 = 256$.

Texture Memory

Specifies how many megabytes of memory is dedicated to *texture caching*. From *3Delight's User Manual*:

"The memory needed to hold the texture for a scene may exceed the amount of core physical memory available. To render such scenes efficiently, 3DELIGHT uses a memory caching system to keep the texture memory footprint below a predefined threshold. This option specifies the amount of memory dedicated to texture mapping algorithms. Increasing this parameter may improve texture, shadow and environment map access performance ...".

Disable Ray Tracing

Disables ray-tracing. *By default, ray tracing is disabled.*

Run Shader For Shadows

Enables the execution of shaders for ray traced shadows (including ambient occlusion). There is a significant gain in not running shaders when using ray traced shadows so this feature is disabled by default. This means that objects will cast shadows as if they were fully opaque. Note that if there is no shaders connected to material's 'Shadow' slot, 3DELIGHT FOR XSI will not run shaders either.

4.1.5 Archives and RIB Export

Archive-Rendering and Exporting

Enable When turned on, this feature makes 3DELIGHT FOR XSI use RIB archives for rendering. The implications are the following:

- For render region rendering, 3DELIGHT FOR XSI will create archives for individual objects in the path specified in *Archives Cache Directory*.
- For RIB export, 3DELIGHT FOR XSI will create an ‘archives’ directory at the same directory level as the main RIB and puts archives in there. The location of the main RIB file is specified through the standard XSI *Pass -> Archive -> Filename* property.

More about RIB export in [Section 5.2 \[RIB Export and Import\]](#), page 19.

Scene Caching

Freeze Archives

When enabled, this option tells 3DELIGHT FOR XSI to *reuse* already created archives, effectively implementing a form of *scene caching*. Subsequent renders of the same scene will use the saved RIB archives instead of asking XSI for the actual geometry. *This can lead to important time gains for complex scenes or scenes containing hair*. We highly recommend to switch this option on for the *lighting* phase of your pipeline.

NOTE: Users can still modify shaders or lights and have correct renders since those are not cached with the archives. Only when geometry is modified that this toggle should be switched off.

More about this option and scene caching in [Section 5.4 \[Efficient Rendering of Hair\]](#), page 20.

Archive Cache Directory

Specifies where to put archives *when rendering in the render region*.

RIB Output Options

Binary When enabled, all RIB files and archives are written in their binary form. Binary RIB’s are about 2 times smaller than corresponding ASCII RIBs.

Compressed

When enabled, all RIB files and archives are written in compressed form. Compressed RIBs are about 4 times smaller than their uncompressed counterpart. Compressed binary files are only slightly better than compressed ASCII files (compression ratios do not multiply, obviously).

NOTE: Compressed RIBs are ordinary *gzipped* files and can be uncompressed using **gunzip**. This also means that 3DELIGHT natively reads gzipped RIB files and archives. It is common to name such compressed RIBs with a ‘.gz’ extension, such as in: ‘scene.rib.gz’.

4.1.6 Search Paths

Shaders Specifies the paths where to look for RenderMan© shaders. Note that this path has absolutely nothing to do with the automatic translation of XSI shaders: it only applies to shaders that are loaded through the renderman shaders properties page (see [Section 4.3 \[The RenderMan Shaders Properties\]](#), page 17).

Textures Specifies the paths where to look for textures. This path is considered when specifying textures in RenderMan© shaders loaded through the renderman

shaders properties. They do not apply to XSI textures: these are automatically handled by the plug-in.

Archives Specifies where to look for archives that are specified in the RIB archive property page (see [Section 4.2.5 \[RIB Archive Object Property\]](#), page 17).

4.1.7 Displays

4.1.8 Shadow Maps

Render Shadow Maps

Specifies if shadow map rendering is needed. A good use of this toggle is to set it to ‘off’ after rendering the shadows once (since when lights do not move, there is no need to re-generate shadows).

Use Shadow Maps

Disable the use of shadow maps in the render, but still generate them (unless *Render Shadow Maps* is off).

4.1.9 Render Tree Conversion

Generate RenderMan© shaders

Toggles automatic shader conversion on and off. Of course, it is usually necessary to have this toggle turned ‘on’ in order to render XSI scenes correctly⁵. This toggle can be switched ‘off’ if shaders are not modified and they have been already translated once; *this can accelerate rendering since the shader translation phase is skipped*.

NOTE: Normally, 3DELIGHT FOR XSI will detect which shaders have been modified and translate only those but a glitch in XSI versions prior to 6.5 makes recompilation necessary if the render region is changed (which normally has no incidence on shaders). The existence of this toggle is in part related to this problem.

Shaders Destination Directory

Specifies the path of the generated shaders. The naming of the shader itself is ‘[Material Name]_surface.sl’ for surface shaders and ‘[Material Name]_displacement.sl’ for displacement shaders.

NOTE: There is no “bump” or “environment” shaders in the RenderMan© specifications. Those are still supported by 3DELIGHT FOR XSI and dealt with inside the surface shader. The way this is done is clearly visible in the ‘.sl’ files, for the interested.

Textures Destination Directory

3DELIGHT FOR XSI will automatically convert XSI textures into mipmapped TIFF files suitable for 3DELIGHT. This property specifies where those textures will be stored. Supported texture formats include: `pic`, `tga`, `tiff`⁶, `psd`, `iff`, `gif`, `openEXR`, `radiance`, `sgi` and `jpeg`.

⁵ Unless you are using only RenderMan© shaders as explained in [Section 5.6 \[Using RenderMan Shaders\]](#), page 21.

⁶ The high dynamic range LogLuv encoding is supported, as well as floating point TIFFs.

4.1.10 Network Cache

NOTE: Network caching is only available in 3DELIGHT FOR XSI| *Studio*. This feature, unique to 3DELIGHT, can substantially improve rendering times, network traffic and server load in networked environments.

From *3Delight Users Manual*:

3DELIGHT offers a special extension for more efficient rendering in networked environments⁷: a local file cache for textures and archives to minimize network traffic and file server load. This proves particularly powerful when using a large quantity of rendering servers.

The working principle of 3DELIGHT's network cache is to copy files to a local file system and reuse them when needed. If either a texture or a RIB archive is cacheable and the file cache is full, one or more files are removed from the cache to make space for the new one; a LRU (Least Recently Used) strategy is used to choose which texture(s) or archive(s) to remove from the cache.

Enable Network Caching

Enables the network cache.

Local Network Cache Directory

The directory, on each rendering machine, where the textures and RIB files will be cached.

Cache Size

The total cache size in megabytes.

Cache Network Writes

Enables *network write caching*: instead of writing data over the network in small bursts, 3DELIGHT will write images locally and then transfer them efficiently at the end of each render. This feature can have a very significant impact on *large* render farms.

4.1.11 Diagnostics

Level Of Detail

Specifies how verbose the statistics will be. The default ('None') disables statistics.

Format

Chooses how to output statistics: XSI console, text file or HTML file.

Message Suppression - Enable

Enables message suppression. All 3DELIGHT messages have an identified, for example R2020. If you want to suppress certain messages just type them in this text field, separated by a comma.

Display Plug-in Performance Statistics

Displays additional timings about the plug-in. This was mainly used during the development cycle and is not of real interest to users⁸.

⁷ Not available on Windows platforms.

⁸ It was used to time the performance of XSI's SDK when getting the Hair and when sending data to the render region.

4.2 3Delight Attributes

3DELIGHT attributes are ordinary object properties that are useful in 3DELIGHT but unavailable through the XSI interface. It is important to note that 3DELIGHT FOR XSI uses the standard XSI properties whenever possible, to not disrupt users habits.

This particular UI piece is accessible through the *Render -> Property -> 3Delight -> Attributes* menu.

4.2.1 Quality

Shading Rate Multiplier

Controls the shading rate on a per-object basis. The specified value *multiplies* the global shading rate set in [Section 4.1.1 \[Quality\]](#), page 8.

4.2.2 Ray Tracing

Trace Displacements

If enabled, the affected objects will be tessellated and displaced for ray tracing⁹. If disabled (the default) the 3DELIGHT render engine will not tessellate surfaces for ray tracing, instead, it will use analytical intersection methods to keep memory usage low and will use bump-mapping to simulate the effect of the displacement (which should be enough in most cases).

NOTE: Ray tracing displacements is a costly feature, both in terms of speed and memory, it should be used with great care.

Trace Motion Blur

Specifies if motion blur effects are to appear in reflections and refractions. This doesn't effect the motion blur for "primary rays".

4.2.3 Subsurface Scattering

Subsurface Light Scattering

Enables or disables subsurface light scattering simulation.

Subsurface Strength

Relative strength of the sub-surface lighting.

Scattering Parameters - Type

Specifies which method to use when specifying surface properties for subsurface scattering simulation. This is either 'Absorption & Scatterin' or 'Mean Free Path & Reflectance'.

Scattering Parameters - Preset

Selects a particular subsurface preset.

Refraction Index

Specifies the index of refraction of the underlying object

⁹ Still, tessellation is performed "just-in-time" and a geometry caching system keeps memory usage in check.

Scale Specifies the scale of the underlying model. A scale of 1.0 means that the model is specified in millimeters. The larger the scale the smaller is the model (.1 means centimeters and 10 means a tenth of a millimeters). The scale is necessary because the absorption, scattering and mean free path are provided in millimeters¹⁰ and the renderer needs to know how to interpret the world metrics. Making the models smaller (by using a scale bigger than 1.0) will make them more “translucent”.

Quality - Shading Rate

Specifies how finely the geometry will be tessellated when running the subsurface scattering simulation. This is similar to *Shading Rate* property described in [Section 4.1.1 \[Quality\], page 8](#).

4.2.4 Geometry

Output Vertex Properties

Output all vertex properties. This should always be ‘on’ since it is needed for XSI shader support. One could toggle this off if only **RenderMan©** shaders are used and no vertex properties are needed. Please refer to [Section 5.6 \[Using RenderMan Shaders\], page 21](#) for an example on how to access vertex properties in a **RenderMan©** shaders.

Output Reference Geometry

Include reference geometry with the main geometry. This is potentially useful when developing **RenderMan©** shaders as explained in [Section 5.6 \[Using RenderMan Shaders\], page 21](#).

Output Cluster Sort Orders

Output the cluster sort orders along with the geometry. This is potentially useful when developing **RenderMan©** shaders as explained in [Section 5.6 \[Using RenderMan Shaders\], page 21](#).

Surfaces

Enable Trimming

Enables NURBS trimming. This is ‘on’ by default.

Curves

Render Curves

The default behaviour in 3DELIGHT FOR XSI is to *not* render curves. Set this option to ‘on’ to render XSI curves smoothly. All curve types are supported.

Width at Root

Width at Tip

Specifies curve’s width at the root and tip, respectively.

¹⁰ Actually, in 1/mm.

Particles

Use Lightweight Particles

If ‘on’, 3DELIGHT FOR XSI will use 3DELIGHT’s *lightweight* particles instead of the geometry specified in XSI (sphere, billboard, etc ...). This is useful to efficiently render large clouds of very small particles.

4.2.5 RIB Archive

Enable Archive Loading

Enables the loading of the archive specified in *RIB Archive Name*.

RIB Archive Name

The name of the RIB archive to read.

Archive Loading Type

Specifies if the archive has to be loaded in *immediate* or *delayed* mode:

‘Immediate’

The archive is loaded as soon as the renderer reaches the associated object. This is often useful when the archive contains only **RenderMan**© attributes and no geometry.

‘Delayed’

The archive is loaded when the renderer encounters the bounding box of the associated object. This should be the default mode for archives containing geometry since this can greatly improve rendering performance.

Please refer to [Section 5.2 \[RIB Export and Import\]](#), page 19 for more about archive loading.

4.2.6 RIB Box

RIB boxing is a common technique to pass custom **RenderMan**© commands to the renderer. Any valid RIB command which is appropriate in the current context, can be used.

4.3 RenderMan Shaders Properties

NOTE: This feature is only available in 3DELIGHT FOR XSI|*Studio*.

This property is a shader assignment UI to connect **RenderMan**© shaders to objects or groups of objects. The UI is accessible through the *Render -> Property -> 3Delight -> Shaders* menu.

The interface will show differently depending if it is connected to a light or not:

- If it is connected to a light, the only possibility is to connect a light source shader.
- If it is not connected to a light, the interface shows three tabs that correspond to three **RenderMan**© shader types: surface, displacement and volume¹¹.

¹¹ Those are actually *interior* shaders.

After specifying the shader, its parameters will appear in the interface and can then be modified or animated. The *Reload* button is handy during shader development: it re-reads all shader parameters and rebuilds the interface appropriately. More at [Section 5.6 \[Using RenderMan Shaders\]](#), page 21.

NOTE: When specifying a shader with a relative path, 3DELIGHT FOR XSI will look in the *3Delight Option* property that applies to the entire scene. It has the following XSI path: 'Passes._3Delight_Options'.

5 Rendering Guidelines

5.1 A Few Words About Tessellation

There is many differences between the 3DELIGHT render engine and XSI's internal renderer but one that is very striking for XSI artists is the *absence of almost all surface tessellation controls*. The following *geometric approximation* properties are not needed in 3DELIGHT FOR XSI:

Geometric Approximation - Surface

All the refinement settings are ignored. 3DELIGHT will tessellate surfaces as required by the *shading rate*, which is specified in [Section 4.1.1 \[Quality\]](#), page 8 and can be modified on a per-object basis as in [Section 4.2.1 \[Quality Object Property\]](#), page 15.

Geometric Approximation - Polygon Mesh

The *Render Level* property is interpreted as follows: a value of '0' will render the polygon mesh as usual and any value greater than '0' will render the polygon mesh as a *smooth subdivision surface*. Users will notice that rendering perfectly smooth subdivisions is almost a no-cost operation in 3DELIGHT FOR XSI.

Geometric Approximation - Displacement

Displacements are rendered to sub pixel accuracy as dictated by the shading rate. The *Max Disp.* property is considered by 3DELIGHT and is of prime importance for the quality and speed of displacements: values that are too large will slow down the renders and values that are too small may cause *cracks* in the rendered image. 3DELIGHT will issue warnings if the displacement bounds are too large or too small.

Geometric Approximation - Surface Trim

3DELIGHT uses an analytical algorithm for trim curves (not relying on surface tessellation). This means that all the parameters in this section are not considered and trim curves are always smooth.

Geometric Approximation - Hair

If *Curve Type* is set to 'Quadratic' or 'Cubic', 3DELIGHT FOR XSI will render them as smooth curves. The *Render Level* property is thus not useful.

5.2 RIB Export and Import

IMPORTANT: RIB export and import functionalities are only exposed in 3DELIGHT FOR XSI|*Studio*.

3DELIGHT FOR XSI can export RIB files using the standard *Render -> Export* menu in XSI. Depending on the settings in the *Archives and RIB Export* property page (see [Section 4.1.5 \[Archives and RIB Export\]](#), page 11, 3DELIGHT FOR XSI will export a *monolithic* RIB or a *multi-archive* RIB. A monolithic RIB contains the entire scene and a multi-archive RIB will contain a series of *delayed reads* that will load smaller archives.

5.3 Ray Tracing in 3Delight for XSI

By default, ray tracing is disabled in 3DELIGHT FOR XSI. It is so because memory management is more efficient without ray tracing and large scenes will render efficiently. Ray tracing is needed for the following two features:

- Ray tracing shadows
- Reflections

To enable ray tracing, uncheck the *Ray Tracing -> Disable* toggle in the render properties.

Anti-Aliasing

When tracing rays for shadows, reflections and refractions, a REYES based renderer such as 3DELIGHT behaves differently than a ray tracer: rays are shot per micro-polygon and *not per screen sample*. This means that to increase the quality of ray traced effects one has to decrease the shading rate. This is usually very costly since decreasing shading rate has an effect on overall performance and not only ray tracing.

A better solution is to increase the *Samples* parameter in your illumination XSI shaders (such as ‘Phong’). Even if the *Gloss* parameter is set to 0, increasing the number of *Samples* will increase the quality of your renders¹.

5.4 Efficient Rendering of Hair

3DELIGHT FOR XSI exposes a few features that are well-suited for rendering of large amount of hair². This includes an efficient archive-based RIB exporter, a scene caching mode, and the efficiency of the core renderer.

5.4.1 Render Settings

A very important thing to consider when rendering hair in 3DELIGHT FOR XSI is that there is a costly phase even before the rendering starts: it is the phase where XSI actually computes the hair positions and give them back to 3DELIGHT FOR XSI. In the lighting stage, this phase can be greatly optimized by using archive-base RIB export and scene caching:

1. In the *Archives and RIB Export* tab of the render properties (see [Section 4.1.5 \[Archives and RIB Export\]](#), page 11), enable the *Archive-Rendering and Exporting* property.
2. Enable the *Freeze Archives* property.

When these two features are set, 3DELIGHT FOR XSI will only ask XSI for the hair data *once*, write the data into RIB archives, and re-use the archives on all subsequent renders. Note that this work flow is very good for any complex scene that is not being modified (which usually happens in the lighting stage).

5.4.2 Quality Settings

In the particular case of hair rendering, quality settings should be influenced by the following observations:

¹ Since 3DELIGHT uses *ray differentials*, it can properly guess what is the angle span of any traced ray, even if *Gloss* is set to 0.

² Or similarly complex scenes

- Generally, hair have a smooth shading variation (low frequency in the shading space) and sharp geometric variation (high frequency in the screen space). This means that using shading rates that are too low (see [Section 4.1.1 \[Quality\], page 8](#)) is usually wasteful and using oversampling that is too low will produce aliasing, especially in animation.

A good setting could be 2.0 for the shading rate, 8x8 oversampling, 6x6 pixel size and using a ‘sinc’, ‘mitchell’ or ‘catmull-rom’ filter.

- Hairs are usually thinner at their tip, so a common practice is to specify this in the XSI hair properties. This is quite often a mistake: making hairs very thin at their tip is a challenge to almost all sampling renderers (including ray-tracers) since there is a greater probability that not enough samples will catch hair geometry, and this is the only needed ingredient to produce bad aliasing. A wiser thing to do is to leave the tip as wide as the root but decrease its transparency proportionally to the desired thickness. So if the tip is two times thinner than the root, use a transparency of 50%. This “technique” will give far smoother renderers.

5.5 Rendering XSI Curves

3DELIGHT FOR XSI can render XSI’s curves and there is no limitation on curve’s shape or degree. The default behaviour is to *not* render curves so that existing scenes render exactly as in XSI’s internal renderer. To enable curves’ rendering:

- Select the desired curve and create a 3DELIGHT object property (see [Section 4.2 \[The 3Delight Attributes\], page 15](#)).
- In the properties, enable the *Render Curves* toggle that is part of the *Geometry* tab.
- Select an appropriate width for the root and the tip of the curve.

Please note the following:

- Curves are rendered using a 3DELIGHT primitive that is meant to render *thin* curves³. Rendering thick curves is better done through proper NURB *surfaces*. Quality guidelines explained in [Section 5.4 \[Efficient Rendering of Hair\], page 20](#) also apply in this case.
- Since XSI will not let you assign materials to curve geometry, the only option is to assign RenderMan© shaders (see [Section 4.3 \[The RenderMan Shaders Properties\], page 17](#)). To get you started, an example shader is provided in [Section 5.6.4 \[Writing Shaders for XSI Curves\], page 24](#).

5.6 Using RenderMan Shaders

NOTE: This section contains some more advanced RenderMan© concepts. One should be familiar with the basics of shader writing before reading this section. Additionally, RenderMan© shaders assignment is only available in 3DELIGHT FOR XSI|*Studio*.

As explained in [Section 3.1 \[Integration with XSI\], page 5](#), 3DELIGHT FOR XSI automatically translates XSI shaders to compiled RenderMan© shaders. This means that no shader

³ Rendered using the RiNuCurves primitive.

programming is necessary to use 3DELIGHT FOR XSI. However, it is logical to provide XSI users with the added flexibility of using RenderMan[®] shaders, if needed. The following sections explain how to connect RenderMan[®] shaders and how to write shaders that are tightly integrated into XSI.

5.6.1 Connecting RenderMan Shaders

IMPORTANT: RenderMan[®] shaders are not connected using XSI material manager but by using a special object property. The reasons for this are explained below.

Connecting a RenderMan[®] shader is a three step process:

- a. Select the object to which the shader will be applied. Note that it is impossible to connect RenderMan[®] shaders to *clusters* since the RenderMan[®] standard has no support for such elements.
- b. Get a RenderMan[®] *Shaders Property* by going to *Render -> Property -> 3Delight -> Shaders*.
- c. Select the surface, displacement, volume or light shader tab and specify a shader path.

Please note the following important details:

- A shader property can be inserted *anywhere in the scene hierarchy*. For example, it is possible to put a RenderMan[®] shader property in the **Scene_Root** and that property will be applied to all objects in the scene. *This is the main reason of using properties to connect RenderMan[®] shaders.*
- It is possible to connect a RenderMan[®] displacement shader, leaving the surface shader empty, and this will effectively use an XSI material for the surface and a RenderMan[®] shader for the displacement.

5.6.2 Accessing XSI Vertex Properties

All XSI vertex properties (such as vertex colors and UV coordinates) are available to shader writers. Accessing a vertex property is as simple as naming it in the parameters list of the shader.

NOTE: If there is 3DELIGHT attributes attached to geometry, make sure that *Export Vertex Properties* is 'on'. This parameter is on by default, as explained in [Section 4.2 \[The 3Delight Attributes\]](#), page 15.

[Listing 5.1](#) shows how to access the *Vertex_Color* vertex property.

```
surface test(
    varying float Vertex_Color[4] = 0; )
{
    /* Put the color in the surface color */
    Ci = color( Vertex_Color[0], Vertex_Color[1], Vertex_Color[2] );
}
```

Listing 5.1: An example RenderMan[®] shader accessing XSI vertex color properties.

Two noteworthy details about [Listing 5.1](#):

- a. We use an array of 4 scalars to pass colors since those can also contain the alpha. Also note that if you export Tangents they will end up as color properties since those are encoded as a *4-tuple*.
- b. The `varying` keyword is needed in front of the parameter since vertex colors vary across the surface.

Listing 5.2 shows how to access a UVW XSI property named *Texture_Projection*.

```
surface test(  
    string texturename = "";  
    varying float Texture_Projection[3] = 0; )  
{  
    /* Access a texture using the given uvs */  
    Ci = texture( texturename,  
        Texture_Projection[0], Texture_Projection[1] );  
}
```

Listing 5.2: An example RenderMan© shader accessing XSI uv properties.

Note that we are using a `float[3]` instead of a `point` in **Listing 5.2** to declare the parameter. This is needed because a `point` would have been transformed by the transformation matrix applied to the underlying object⁴; passing an array of floats bypasses this “problem”.

5.6.3 Accessing XSI Render Channels

All XSI render channels are easily accessed by naming them as *output varying* shader parameters. The names are the same as in the XSI interface: ‘**Diffuse**’ for the diffuse channel, ‘**Specular**’ for the specular channel, ‘**Ambient**’ for the ambient channel and so on. **Listing 5.3** shows an example shader that sets some of XSI render channels.

⁴ As per the RenderMan© specifications.

```

surface renderchannels(
    output varying color Specular = 0;
    output varying color Diffuse = 0;
    output varying float Depth = 0; )
{
    /* This will set the Diffuse/Specular render channels in XSI */
    Specular = noise( P );
    Diffuse = diffuse( N );

    /* Store the depth, in XSI infinity is set to 0. */
    Depth = depth(P);
    if( Depth > 1e36 )
        Depth = 0;

    /* Put some different color pattern in the color */
    Ci = cellnoise( P * 2 );
    Ci *= Oi;
}

```

Listing 5.3: An example RenderMan© shader to access XSI render channels.

Accessing *store-in* render channels is similarly easy: the store-in has to be declared as parameter of the shader, with the correct type.

5.6.4 Writing Shaders for XSI Curves

As explained in [Chapter 3 \[Features Summary\]](#), [page 5](#) and [Section 4.2.4 \[Geometry Object Property\]](#), [page 16](#), 3DELIGHT FOR XSI can render XSI curves of any shape and of any degree. Since XSI does not allow connecting a material to such geometric primitives, the only option for shading is to assign a RenderMan© shader. A few remarks:

- 3DELIGHT FOR XSI exports curves' length with each curve using the *xsi_curve_length* variable so shader writers can have access to it by declaring it as a parameter of the shader.
- RenderMan© curves vary in 'u' across their width and in 'v' across their length.
- Curves *knot vector* is normalize between 0 and 1 so 'v' varies between 0 and 1.

[Listing 5.4](#) gives a simple example of such a renderman shader. Note the presence of the *xsi_curve_length* variable that is available to shader writers and automatically passed as a *primvar* to all curve primitives.

```

/*
  A simple shader that demonstrates how to cut a curve at a specific
  length. This example shows how to:
  -- access curve's length
  -- set the opacity using an anti-aliased step function
*/
surface curve_cutoff(
  float xsi_curve_length = 0; /* This is set by 3Delight for XSI */
  float cutoff = 0; )
{
  /* compute distance from root */
  float current_length = v * xsi_curve_length;

  float f = filterstep( current_length, cutoff );

  /* Opacity becomes 0 after the cutoff */
  Oi = f;

  /* Set the color to white and modulate by opacity as usual in
  RenderMan shaders */
  Ci = 1 * Oi;
}

```

Listing 5.4: A simple shader for XSI curves.

5.6.5 Accessing Reference Geometry

Reference geometry can be output as the usual *Pref* and *Nref* RenderMan© *primvars* by enabling reference geometry output in [Section 4.2.4 \[Geometry Object Property\]](#), page 16. [Listing 5.5](#) illustrates how to access the reference geometry from inside a shader.

```

surface prefaccess(
  varying point Pref = 0;
  varying normal Nref = normal(0, 0, 1); )
{
  Ci = diffuse( Nref ) * noise( Pref );
}

```

Listing 5.5: Accessing XSI reference geometry from inside RenderMan© shaders.

Note that exporting reference geometry will double the size of the object so this feature should be enabled only when strictly necessary.

5.6.6 Writing “Cluster Aware” Shaders

It is possible to access clusters’ *sort order* property from inside a RenderMan© shader as shown in [Listing 5.6](#). This is an easy way for a TD to create clusters and write particular code for each one of them.

NOTE: To export the cluster sort order, it is necessary to attach 3DELIGHT attributes to geometry and switch the *Export Cluster Sort Order* property to ‘on’. Refer to [Section 4.2 \[The 3Delight Attributes\]](#), page 15.

```
surface test(  
    uniform float xsi_cluster_sort_order = 0; )  
{  
    /* Just put the sort order as a gray scale value */  
    Ci = xsi_cluster_sort_order / 255;  
}
```

Listing 5.6: An example **RenderMan**© shader accessing cluster's sort order.

Note that a **uniform** variable is enough here since the sort order of each cluster doesn't vary across one single polygon.

5.7 Supported XSI Shaders

Credits

We would like to thank the following people for helping us with this product:

Softimage As a company, for making great products and for always being helpful during the development cycle.

Halfdan Hivgarsson, *Softimage*

For the continuous technical help and the great XSI rendering SDK that made this work possible.

Jason Brynford-Jones, *Softimage*

For all the help, encouragements and support as well as the efforts to show 3DELIGHT FOR XSI to the XSI community.

Luc-Eric Rousseau, *Softimage*

For continuously trying to make the link between *d n' a* and *Softimage*. It finally worked out.

All the beta-testers

For beta-testing, obviously :).

And last but not least, the author would like to thank Moritz Moeller (and its employer at the time, *Rising Sun Pictures*) for the tremendous and pioneering work on *Affogato*, for making the source available, and for the continuous harassment to make 3DELIGHT FOR XSI.

List of Figures

Listing 5.1: An example RenderMan © shader accessing XSI vertex color properties...	22
Listing 5.2: An example RenderMan © shader accessing XSI uv properties.	23
Listing 5.3: An example RenderMan © shader to access XSI render channels.	24
Listing 5.4: A simple shader for XSI curves.	25
Listing 5.5: Accessing XSI reference geometry from inside RenderMan © shaders.	25
Listing 5.6: An example RenderMan © shader accessing cluster's sort order.	26

Concept Index

A

accessing 3delight attributes	15
accessing clusters in renderman shaders	25
accessing renderer properties	8
affogato	27
anti-aliasing	2, 6
anti-aliasing, ray tracing	20
archive-based rendering	12
archive-based rendering, render region	12
archives, freeze	12
automatic shader conversion	13
automatic subsurface scattering	6
automatic texture conversion	13

B

background, running license server in	4
box filter	9
bump shaders	5

C

caching, scene	6
camera, motion blur	10
cluster sort order, accessing in shaders	25
cluster sort orders	16
clusters	5
clusters in renderman shaders	25
configuration, settings files	4
connecting renderman shaders	22
curves, curve length	24
curves, example RenderMan shader	24
curves, rendering	16, 21
curves, width	21
curves, XSI shaders	21

D

deformation blur steps	10
delayed archives	7
depth of field	2, 6
diagnostics, format	14
diagnostics, level of detail	14
diagnostics, message suppression	14
diagnostics, plug-in timings	14
directory, shader destination	13
directory, texture destination	13
disabling trimming	16
displacement shaders	5, 17
displacement, geometric approximation	19
displacements	2, 6

E

edge anti-aliasing	9
efficient rendering of hair	20
enabling ray tracing	11, 20
environment shaders	5
explicit projections	5
exporting cluster sort order	25
exporting RIB	7

F

f-number	10
f-stop	10
filter type	9
filter width	9
freeze archives	12
frequency of shading	8

G

gaussian filter	9
geometric approximation, displacement	19
geometric approximation, hair	19
geometric approximation, polygon mesh	19
geometric approximation, surface	19
geometric approximation, surface trim	19
geometry, nurbs	5
geometry, polygons	5
gloss, ray tracing	20
grid size, optimization	11

H

hair support	6
hair transparency	21
hair, geometric approximation	19
hair, rendering	20
hostid	3

I

iff textures support	13
implicit projections	5
installation	3
installation, local	3
installation, network	3
introduction	2

J

jpeg textures support	13
-----------------------	----

L

license 3
 lightweight particles 17
 local installation 3
 logluv encoding 13

M

material clusters 5
 micro-polygons, shading rate 8
 mitchell filter 9
 motion blur 6
 motion blur, camera steps 10
 motion blur, deformation steps 10
 motion blur, delay 10
 motion blur, ray trace 15
 motion blur, shutter 10
 motion blur, shutter efficiency 10
 motion blur, transformation steps 10
 motion-blur 2
 multi-segment motion blur 6

N

network installation 3
 non opaque shadows 11
 nurbs 5

O

openexr textures support 13
 optimization, grid size 11
 optimization, texture memory 11
 optimization, tile size 11
 oversampling 9
 overview 2

P

particles, lightweight 17
 pic textures support 13
 pixel samples 9
 polygon mesh, geometric approximation 19
 polygons 5
 preview, shaderballs rendering 5
 product description 2
 projections 5
 properties, binary rib output 12
 properties, camera motion blur 10
 properties, compressed rib output 12
 properties, deformation motion blur 10
 properties, delay 10
 properties, disable ray tracing 11
 properties, enable depth of field 10
 properties, enable rib archives 11
 properties, enable trimming 16
 properties, f-stop 10
 properties, filter width 9

properties, output cluster sort orders 16
 properties, output reference geometry 16
 properties, output vertex properties 16
 properties, oversampling 9
 properties, render curves 16
 properties, shading 5
 properties, shaders destination directory 13
 properties, shading rate 8
 properties, shading rate multiplier 15
 properties, shutter 10
 properties, shutter efficiency 10
 properties, texture memory 11
 properties, textures destination directory 13
 properties, tile size 11
 properties, trace displacements 15
 properties, trace motion blur 15
 properties, transformation motion blur 10
 properties, use lightweight particles 17
 properties, width at root, curves 16
 properties, width at top, curves 16
 property, archive cache directory 12
 property, freeze archives 12
 properties, generate renderman shaders 13
 psd textures support 13

Q

qualities of the product 2
 quality controls 6
 quality settings for hair rendering 20
 quality, depth of field 6
 quality, filter type 9
 quality, filter width 9
 quality, motion blur 6
 quality, oversampling 9
 quality, shading rate 8

R

radiance textures support 13
 ray differentials 20
 ray traced shadows 11
 ray tracing motion blur 15
 ray tracing, anti-aliasing 20
 ray tracing, disable 11
 ray tracing, enabling 11, 20
 reference geometry 16
 render channels, accessing 23
 render region support 5
 renderer properties 8
 rendering curves 21
 rendering shaderballs 5
 rendering xsi curves 16
 rendering, integration 5
 renderman shaders 6
 renderman shaders, available variables 22
 renderman shaders, displacement 17
 renderman shaders, render channels 23

renderman shaders, search paths	12
renderman shaders, store-in render channels	24
renderman shaders, surface	17
renderman shaders, volume	17
rib archives	7
RIB export	7

S

scene caching	6, 12
search paths, shaders	12
search paths, textures	12
settings file	4
sgi textures support	13
shader assignment panel	17
shader destination directions	13
shaderballs, rendering	5
shaders search paths	12
shaders, for xsi curves	24
shaders, renderman	6
shading frequency	8
shading rate	8
shading rate multiplier property	15
shading rate, common parameters	8
sinc filter	9
smooth surfaces	2
sort order, accessing in shaders	25
sort orders, clusters	16
starting the license server	4
store-in render channels, accessing	24
sub-pixel displacements	6
subsurface scattering	6
supersampling	9
supported texture formats	13
surface shaders	5, 17
surface trim, geometric approximation	19
surface, geometric approximation	19

surfaces	5
----------------	---

T

taarga textures support	13
texture conversion	5, 13
texture memory, optimization	11
textures search paths	12
thickness of hair	21
tiff textures support	13
tile size	11
trace displacements	15
transformation blur steps	10
transparency in ray traced shadows	11
trimming, object property	16

U

using renderman shaders	22
-------------------------------	----

V

vertex colors	5
vertex properties	16
volume shaders	17

W

workgroup installation	3
------------------------------	---

X

xsi curves	6
xsi hair	6
xsi standard properties	5
xsi texture conversion	13
xsi_curve_length shader variable	24