

Meridian Rendering Software

# Meridian for Maya User Guide

---

January 10, 2012

Version 2.0.1

Copyright © 2011 Sunfish Studio, LLC

All rights reserved.

Meridian™, MeridianSL™ and RXF™ are trademarks of Sunfish Studio, LLC.

Protected by one or more of the following patents, or by other patents in the United States or other countries, or by patents pending:

*US 7949700*  
*US 7554540*  
*US 7250948*

*AU 2003294327*  
*NZ 563047*  
*NZ 540742*

*JP 4643271*

Autodesk® and Maya® are registered trademarks of Autodesk, Inc. in the United States and other countries.

## Contents

---

Introduction .....	1
About this Document.....	1
Supported Features .....	1
Planned Features .....	3
Installation .....	3
Loading the Plugin.....	4
Getting Started.....	5
Rendering RXF Files.....	5
Simultaneous Export and Rendering .....	7
Render Settings.....	8
Common.....	8
File Output .....	8
Renderable Cameras.....	8
Extension Attributes .....	9
Cameras .....	9
Pixel Resolution.....	9
Transform Motion Blur .....	10
Non-Extended Lights.....	10
Shadow Map Type .....	10
Pixel Resolution.....	11
Hair.....	11
Render As.....	11
Transforms .....	12
Local Attributes.....	12
Shading Rate .....	13
Timebase.....	13
Transparency Scale .....	13

---

The Maya Scene and DAG Nodes.....	15
Geometric Shapes.....	15
Polygons and Smooth Mesh Preview.....	15
NURBS.....	15
Paint FX and Hair.....	16
Particle Systems.....	16
Instancers and Instanced Geometry.....	16
Subdivision Surfaces.....	17
Cameras.....	17
Non-Extended Lights.....	17
Hypershade.....	18
Automatic Translation.....	18
Surface Materials.....	18
Volumetric Materials.....	19
Displacement Materials.....	19
Lights.....	19
2D Textures.....	20
3D Textures.....	20
Environment Textures.....	20
General Utilities.....	21
Scalar Utilities.....	21
Switch Utilities.....	22
Color Utilities.....	22
Particle Utilities.....	22
Image Planes.....	22
Glow.....	22
Texture (TXR) and Displacement (TXD) Files.....	22
Custom Shading.....	23
User-Defined Shaders.....	23
Sunfish Repository Node.....	24
Sunfish Shader Node.....	24
Annotations.....	26

Command-Line Renderer.....	27
Getting Started.....	27
Pipe Mode.....	27
Working Directory.....	28

## Introduction

*Thank you for choosing Meridian.*

*Nathan T. Hayes*

Meridian for Maya is a plugin developed by Sunfish Studio, LLC for Autodesk Maya. The purpose of the plugin is to export the contents of a scene created in Maya to a format that Meridian can render.

[Render Exchange Format \(RXF\)](#) is a format that describes the contents of a scene to Meridian. Meridian for Maya exports scene information as RXF, and Meridian can then render the RXF data to produce beautiful images.

## About this Document

---

This document explains how to install, configure and operate Meridian for Maya. To learn how to install, license and operate Meridian, please see the [Meridian Rendering Software User Guide](#).

Meridian for Maya is a plugin that provides a friendly user-interface for exporting RXF data from Maya. After the RXF data is exported from Maya, it can then be rendered by Meridian.

Meridian is a stand-alone console application. This document assumes the reader has some basic knowledge of how to work at the command prompt in a console window. To learn more about command prompts and console windows, consult your operating system documentation.

## Supported Features

---

Meridian for Maya is tightly integrated with Maya to provide a practically seamless user interface. Just a few of the features in Maya that are automatically supported are described below.

### Polygons, NURBS and Subdivision Surfaces

Meridian for Maya faithfully exports all of these geometry types, along with any shading group assignments made at the component level (even NURBS).

## Smooth Mesh Preview

Polygons with this option render as a smooth surface and no explicit conversion is necessary.

## Paint FX and Hair

Paint FX and Hair are exported as lightweight curved rendering primitives. Important attributes such as tube flatness, orientation, color and opacity are all exported faithfully, and users may choose to export Hair as curved ribbons or tubes.

## Particles

Particles and nParticles are exported as lightweight point rendering primitives. The position and size of each particle is exported faithfully, and per-particle attribute variables may be used in shaders to create spectacular visual effects.

## Instancers and Instanced Geometry

Meridian for Maya provides full support for instancers and instanced geometry, even with motion blur.

## Motion Blur of Rigid Bodies

If motion blur is enabled, Meridian for Maya automatically calculates the motion of objects moving relative to the camera. Motion is decomposed into a set of animation curves parameterizing the position, orientation, scale and shear of moving objects. The result is spectacular nonlinear motion blur that does not rely on a large number of linearly-interpolated segments to correctly render the motion of a spinning propeller or fan blade, for example.

## Displacement Maps

Floating-point images can be applied to surface geometry in order to render surface relief as an offset along the surface normal. Meridian automatically renders beautiful displacements without cracking or creasing, even in the case of very extreme displacement.

## Variance Shadow Maps

Meridian for Maya generates shadow maps for directional, spot and omni light sources when necessary. Shadow maps are automatically pre-filtered using a proprietary extension of the Variance Shadow Map (VSM) method and contain transparency data to provide accurate shadows of transparent and motion-blurred objects. The proprietary VSM method used by Meridian does not suffer from commonly known VSM artifacts such as “light leaks” and over-darkening.

## Hypershade

Meridian for Maya automatically compiles Hypershade networks into Meridian Shading Language source files. An increasingly large set of nodes are supported, including uv-choosers, file textures, procedural noise, bump, reflection, etc.

## Custom Shading Integration

Users can write their own shaders in Meridian Shading Language and then connect the inputs and outputs of the shader to other nodes in a Hypershade network just like any standard Maya node. This provides a “mix and match” philosophy and workflow where custom shading assets can be easily integrated into a shading network that also uses standard Maya nodes, allowing very sophisticated shading graphs to be created with ease.

## Batch Mode

Meridian for Maya can be run from the Maya command-line rendering interface to export RXF data to a file, and a pipe mode allows RXF data to be streamed directly to the Meridian rendering software without consuming any disk space at all.

## Planned Features

Some features did not make it into the initial release and are planned to be added very soon:

- Volumetric shaders
- Deformation (soft-body) motion blur
- Depth of field
- Blobby particles

Sunfish is always working hard to improve Meridian for Maya. If you see a missing feature, let us know what your needs and requirements are!

## Installation

---

Table 1 is a summary of installers included in the Meridian for Maya distribution media.

Installer	Description
MeridianForMaya-2012.msi	Meridian for 32-bit Maya 2012
MeridianForMaya-2012-x64.msi	Meridian for 64-bit Maya 2012

**Table 1.** Installers included in the distribution media.

Before installing Meridian for Maya, be sure to choose the appropriate installer for your version of Maya. For example, choose the 64-bit installer if you are using a 64-bit version of Maya.

If you have multiple versions of Maya installed on the same computer, it is also safe to install all of the corresponding versions of Meridian for Maya.

### **How to install Meridian for Maya**

---

- Locate the appropriate Installer in the Meridian for Maya distribution media
- Double-click on the Installer
- Follow the on-screen instructions to complete the installation procedure

## Loading the Plugin

Once Meridian for Maya is installed, it is necessary to load the plugin.

### **How to load the Meridian for Maya plugin**

---

- Open Maya
- Select [Window > Settings/Preferences > Plugin-in Manager](#) from the Maya main menu
- The [Plug-in Manager](#) window appears
- Look for the [RXF.mll](#) plugin
- Check the [Loaded](#) and [Auto load](#) options for the plugin

After completing all of these steps, Meridian for Maya is ready to be used.

## Getting Started

Meridian for Maya always exports the contents of a scene from the point of view of the [Renderable Camera](#) selected in the [Common](#) tab of the [Render Settings](#) dialog. Before using Meridian for Maya to export a scene, be sure to select the camera you wish to render.

### How to select a renderable camera

---

- Select [Window > Rendering Editors > Render Settings](#) from the Maya main menu
- The [Render Settings](#) dialog appears
- Click on the [Common](#) tab
- Select a camera from the [Renderable Camera](#) drop-down list in the [Renderable Cameras](#) layout

After selecting a renderable camera, it is time to export the scene.

### How to export a scene as a RXF file

---

- Select [File > Export All...](#) from the Maya main menu
- The [Export All](#) dialog appears
- Select [Sunfish Studio \(RXF\)](#) from the [Files of type](#) drop-down list
- Enter a file name in the [File name](#) edit box
- Click [Export All](#)

After completing these steps, Meridian for Maya exports the scene or animation as RXF to the specified file.

## Rendering RXF Files

---

At the time Meridian for Maya exports a RXF file, it may also create an auxiliary folder that contains textures, source files and shadow maps. This folder will always be created in the same directory location and have the same name as the RXF file.

For example, if you export a file called [scene.rxf](#) to a particular directory on your hard drive, you may also notice that Meridian for Maya creates a folder (directory) called [scene](#) in the same location:

```
D:\Data\Draw>dir
Volume in drive D has no label.
Volume Serial Number is 10E5-F04F

Directory of D:\Data\Draw

09/15/2011  01:33 PM  <DIR>          .
09/15/2011  01:33 PM  <DIR>          ..
09/15/2011  01:33 PM  <DIR>          scene
09/15/2011  01:33 PM                110,468 scene.rxf
               1 File(s)                110,468 bytes
               3 Dir(s)  135,058,485,248 bytes free

D:\Data\Draw>
```

The `scene` folder in this example contains additional information that Meridian may need to render a picture.

For this reason, always run Meridian from the same directory location as the RXF file you wish to render:

```
D:\Data\Draw>fish scene.rxf
=====
[      N      ] Meridian Rendering Software
[      |      ] Release 2.1.9.13, 64-bit
[ W--+---E ] Copyright 2010 Sunfish Studio, LLC
[      |      ] U.S. and foreign patents granted. Patents pending.
[      S      ] Using 2 of 2 active processing core(s)
=====
[09/15 13:35] scene.rxf
[09/15 13:35] Compiling shaders...
[09/15 13:35] 0:00:00.03 elapsed time for compiling
[09/15 13:35] Starting program...
[09/15 13:35] Frame: 1
[09/15 13:35] Raster: 0 0 640 480
[09/15 13:35] Grid: 64
[09/15 13:35] Granules: 16
[09/15 13:35] Rendering tiles...
[09/15 13:35] 5/80 complete
```

In the above example, Meridian was run from the same `D:\Data\Draw` directory that the `scene.rxf` file and `scene` folder are both located in. If you do not run Meridian from the same directory, Meridian may not be able to find the additional information located in the `scene` folder:

```
D:\Data>fish draw\scene.rxf
=====
[      N      ] Meridian Rendering Software
[      |      ] Release 2.1.9.13, 64-bit
[ W--+---E ] Copyright 2010 Sunfish Studio, LLC
[      |      ] U.S. and foreign patents granted. Patents pending.
[      S      ] Using 2 of 2 active processing core(s)
=====
[09/15 20:18] draw\scene.rxf

      LINE: 9

      MESSAGE: Unable to open RXF input

Error reported from program location: reader.cpp(36)

*****
****  RXF ERROR  ****
*****

Unable to execute a user request.

Please correct the problem and try again.

D:\Data>
```

In the above example, Meridian was run from the `D:\Data` directory even though this is not the directory where the `scene.rxf` file and `scene` folder are located. This caused an error because Meridian was unable to find the additional information located in the `scene` folder.



**Note.** For similar reasons, if you decide to move the `scene.rxf` file to a new location you should also move the `scene` folder along with it.

If you would like to understand in more detail the relationship between the `scene.rxf` file and `scene` folder, display the contents of the `scene.rxf` file:

```
D:\Data>more data\scene.rxf
# Sunfish Studio Render, version 2.0.0.0
# Autodesk Maya Version 2012 x64
# File created from: D:/Data/Projects/Maya/2012/Beta/scenes/bump3d.mb
# Output File Name: D:/Data/Draw/scene.rxf
# Created: 09/15/11 at 13:33:07
# Rendering Frame(s) : 1

Program #PUSH Program.
Source "scene/shaders/include/Bump3d.txt"
Source "scene/shaders/include/DirectionalLightExpress.txt"
Source "scene/shaders/include/Lambert.txt"
Source "scene/shaders/include/Maya.txt"
Source "scene/shaders/lights/directionalLightShape1.txt"
Source "scene/shaders/lights/directionalLightShape2.txt"
Source "scene/shaders/lambert1.txt"
-- More (1%) --
```

You can see that line 9 contains a reference to a Meridian Shading Language source file `scene/shader/include/Bump3d.txt`. Meridian will not be able to resolve this relative reference unless the `scene` folder is located in the current directory where Meridian was run from.

## Simultaneous Export and Rendering

When exporting a scene, you do not have to wait for Meridian for Maya to complete the export before you begin rendering.

You may begin rendering a RFX file at any time, even if the export is not finished. If the rendering process catches up to the export process, Meridian automatically pauses until more RFX data is available and then resumes rendering. So there is never any danger to render a RFX file concurrently with the export process.

## Render Settings

Meridian for Maya uses options specified in the [Render Settings](#) dialog to specify certain rendering preferences for a RXF file.

### **How to open the Render Settings dialog**

---

- Select [Window > Rendering Editors > Render Settings](#) from the Maya main menu
- The [Render Settings](#) dialog appears

## Common

---

Meridian for Maya supports most of the layout options on the [Common](#) tab of the [Render Settings](#) dialog. Options in the [Color Management](#) and [Render Options](#) layouts are not supported.

## File Output

Meridian for Maya currently supports 8-bit or 16-bit TIFF file output. This can be selected by choosing [Tiff](#) or [Tiff16](#) from the [Image format](#) drop-down list. If any other format is selected, a warning message will be issued and 8-bit TIFF output will be generated by default.

Custom naming and extensions are not supported.

## Renderable Cameras

Meridian for Maya always exports a scene from the camera specified in the [Renderable Camera](#) drop-down list. If the [Alpha channel](#) check box is selected, an alpha channel is included in the output file. If the [Depth channel](#) check box is selected, a separate 32-bit floating-point TIFF image containing depth information is output.

## Extension Attributes

Meridian for Maya uses the *extension attributes* introduced in Maya 2012 so that custom RXF preferences can be attached to specific Maya nodes and stored in the scene graph. If a node has custom extension attributes, they appear automatically in the [Attribute Editor](#) under a layout called [Sunfish Studio](#) when the Meridian for Maya plugin is loaded.

This section documents the extension attributes of all relevant node types.

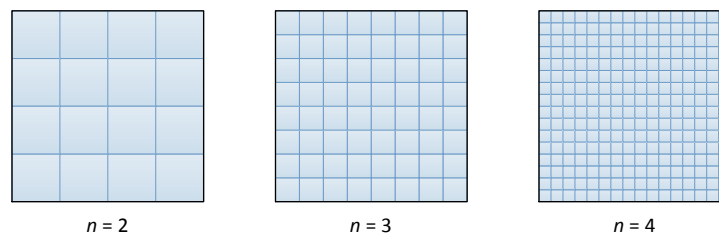
## Cameras

---

Meridian for Maya creates extension attributes for [camera](#) nodes. These extension attributes specify custom RXF preferences that are used to control aspects of the rendered image.

## Pixel Resolution

Use this option to specify the subpixel resolution of each pixel in the image. If  $n$  is the [Pixel Resolution](#), then the total number of subpixel fragments in each pixel is  $2^{2n}$  as illustrated in Figure 1. [Pixel Resolution](#) must be an integer value in the interval  $[1,6]$ .



**Figure 1.** [Pixel Resolution](#) specifies the subpixel resolution of each pixel in the image. If  $n$  is the [Pixel Resolution](#), the total number of subpixel fragments is  $2^{2n}$ .

Choose an increased pixel resolution for improved image quality or a decreased pixel resolution for speed. Large pixel resolutions are recommended only if the image contains motion blur or transparency; otherwise smaller values can be used. The default value of 4 is typically suitable for most high-quality image rendering at a reasonable speed.

## Transform Motion Blur

Use this option to select the type of motion blur that will be exported into the RXF data, as depicted in Table 2.

Transform Motion Blur	Description
None	Disables motion blur
Geometry Only	Compute motion blur only for the transform matrix of geometric shapes
Camera Only	Compute motion blur only for the transform matrix of the camera
Camera and Geometry	Compute motion blur for the transform matrix of geometric shapes and the camera

**Table 2.** Options to specify the type of motion blur that will be exported into the RXF data.

Motion blur can be selectively disabled for each geometric shape in the scene by unchecking the [Motion Blur](#) check box in the [Render Stats](#) layout of the geometric shape.

The motion blur calculation depends on the [Shutter Angle](#) option in the [Special Effects](#) layout of the camera. To simulate the effect of longer exposure times and more motion blur, increase the [Shutter Angle](#) value.

## Non-Extended Lights

Meridian for Maya creates extension attributes for [directionalLight](#), [pointLight](#) and [spotLight](#) nodes. These extension attributes specify custom RXF preferences that are used by the Meridian for Maya plugin only when the [Use Depth Map Shadows](#) check box in the [Shadows](#) rollup is selected.

## Shadow Map Type

Shadow maps are automatically pre-filtered using a proprietary extension of the Variance Shadow Map (VSM) method and contain transparency data to provide accurate shadows of transparent and motion-blurred objects. The proprietary VSM method used by Meridian does not suffer from commonly known VSM artifacts such as “light leaks” and over-darkening.

Use this option to select the type of VSM that will be automatically generated and exported into the RXF data, as depicted in Table 3.

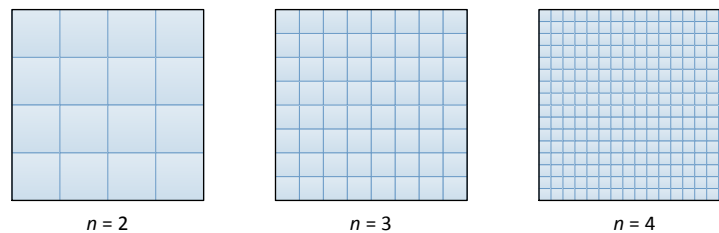
Shadow Map Type	Description
Alpha	High-quality pre-filtered grayscale shadows of transparent and motion-blurred objects
Color	High-quality pre-filtered color shadows of transparent and motion-blurred objects

**Table 3.** Options to specify the type of VSM that will be automatically generated and exported into the RXF data.

Choose the [Alpha](#) option for scenes that do not require colored shadows for transparent objects. This option requires less disk space and is typically faster than the [Color](#) option.

## Pixel Resolution

Use this option to specify the subpixel resolution of each pixel in the VSM prior to filtering. If  $n$  is the [Pixel Resolution](#), then the total number of subpixel fragments in each pixel is  $2^{2n}$  as illustrated in Figure 2. [Pixel Resolution](#) must be an integer value in the interval  $[1,6]$ .



**Figure 2.** [Pixel Resolution](#) specifies the subpixel resolution of each pixel in the VSM prior to filtering. If  $n$  is the [Pixel Resolution](#), the total number of subpixel fragments is  $2^{2n}$ .

Choose an increased pixel resolution for improved image quality or a decreased pixel resolution for speed. Large pixel resolutions are recommended only if the VSM contains motion blur or transparency; otherwise smaller values can be used. The default value of 2 is typically suitable for most high-quality shadows at a reasonable speed.

The VSM is automatically generated by Meridian at render time, pre-filtered and then stored as a mip-map in a file so it can be used in the beauty pass. Increasing the pixel resolution provides better pre-filtered results and generates higher-quality shadows, but it does not actually increase the file size of the VSM. It may take longer to export and render a VSM with a large pixel resolution, though.

---

## Hair

Meridian for Maya creates extension attributes for [pfxHair](#) nodes. These extension attributes specify custom RXF preferences that are used to control aspects of the rendered image.

## Render As

Use this option to specify if hair geometry is rendered as [Ribbons](#) or [Tubes](#).

Rendering hair as ribbons results in significantly faster render times than rendering as tubes. In most cases, hair is far enough away from the camera so that rendering as tubes would provide no noticeable difference, anyways.

However, you can use this option to create interesting effects. If you are a flea on the back of a dog, for example, then rendering the dog hair as tubes may be appropriate!

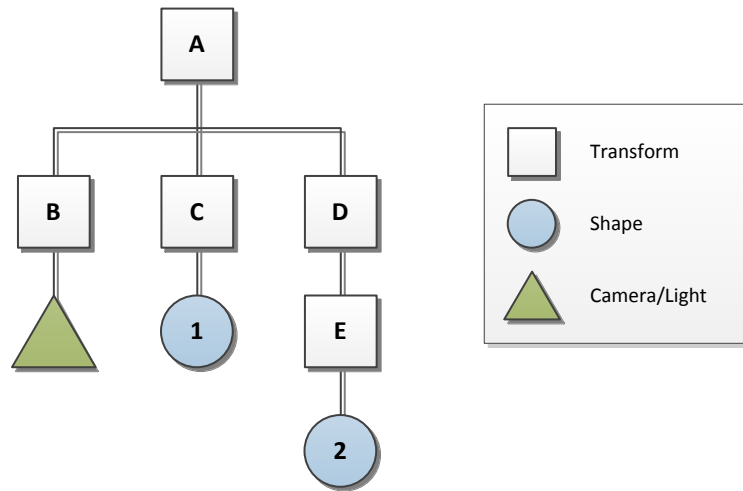
## Transforms

Meridian for Maya creates extension attributes for **transform** nodes. These extension attributes specify custom RXF preferences that are used to control aspects of the rendered image.

Transform extension attributes are hierarchical. If the transform node is a parent of another transform node, each child transform node can selectively choose to inherit or override the extension attributes of the parent.

## Local Attributes

For each shape in the scene, Meridian for Maya examines the transform node parent of the shape to see if the **Local Attributes** option is checked or unchecked. If the option is checked, the extension attributes of the transform node are applied to the shape. If the option is unchecked, Meridian for Maya traverses up one level of the scene graph to the parent of the transform node and repeats the examination procedure.



**Figure 3.** An example scene graph that has two shape nodes, five transform nodes and a camera node.

If none of the transform nodes in the DAG path of the shape have the **Local Attributes** option checked, Meridian for Maya searches the DAG path of the camera by starting at the camera node and then traversing upwards one parent at a time as it did for the DAG path of the shape node.

If none of the transform nodes in the DAG path of the camera have the [Local Attributes](#) option checked, then default values are applied to the shape.



**Note.** For a shadow pass, the camera node is actually the non-extended light node casting the shadow.

Because of this hierarchical traversal process, transform extension attributes may be adjusted at global or local levels in the scene graph. For example, transform extension attributes of a camera or light represent global values applied to all shape nodes in the scene or shadow pass that do not explicitly check the [Local Attributes](#) option of some transform node in the DAG path of the shape node.

Figure 3 is an example. There are two shape nodes in the figure. Meridian for Maya first searches transform node C and then A for the extension attributes applied to shape 1. It searches transform nodes E, D and then A for extension attributes applied to shape 2. In either case, the first transform node found in the search that has the [Local Attributes](#) option checked will be used. If no transform nodes in the search have the [Local Attributes](#) option checked, Meridian for Maya searches transform nodes B and then A, in that order, for extension attributes that may be applied to the respective shape.

## Shading Rate

[Shading Rate](#) is a multiplier value expressed in pixel area. A value of 1.0 has no effect. Larger or smaller values improve rendering speed or image quality respectively. The default value is 1.0.

## Timebase

[Timebase](#) is a multiplier of [Shading Rate](#). A value of 1.0 has no effect. Larger or smaller values respectively improve rendering speed or image quality of geometric primitives with motion blur. The default value is 1.0.

## Transparency Scale

[Transparency Scale](#) controls the quality of transparency effects.

Unlike many rendering engines, Meridian uses a unique method for rendering transparency that operates in a single pass and uses a constant memory footprint. This allows highly complex scenes with transparent objects to render efficiently and never run out of memory. A consequence of this method is that only a fixed dynamic range is available for the transparency effect.

**Transparency Scale** adjusts the dynamic range of the transparency effect based on the depth-buffer values of geometric primitives in the scene. It must be an integer in the interval [0,63]. The default value is 32.

## The Maya Scene and DAG Nodes

For the most part, Meridian for Maya examines the Maya scene graph and determines all necessary and relevant settings from the standard Maya nodes and attributes.

Aside from the extension attributes documented in the previous chapter, there really are no other special knobs, buttons or settings that need to be taken into consideration.

This section of the document simply explains how Meridian for Maya uses some of the standard Maya nodes and attributes to export RXF data.

## Geometric Shapes

---

Meridian for Maya provides automatic support for many of the most popular geometric shapes and primitives in Maya.

## Polygons and Smooth Mesh Preview

Meridian for Maya faithfully exports polygon geometry, along with any *uv*-sets or shading group assignments made at the component level.



**Note.** Meridian is highly optimized to render four-sided polygons (quads). Modeling polygonal geometry as quads can help Meridian render the scene much more efficiently.

Polygons with the [Smooth Mesh Preview](#) option selected automatically render as a smooth surface and no explicit conversion is necessary.

## NURBS

NURBS are exported as true freeform surfaces without any tessellation into polygons. Shading group assignments made at the component level are also supported.

Meridian can render NURBS as perfectly smooth surfaces. No tessellation into polygons is required, even at render-time.

Because of this unique ability, NURBS models can be rendered by Meridian at amazing levels of speed and quality. No more cracking, faceting or creasing of NURBS models, and no more crashing or slow render times on complex scenes that may contain even millions of NURBS.

Trimming of NURBS surfaces is not currently supported.

## Paint FX and Hair

Paint FX and Hair are exported as lightweight curved rendering primitives. Important attributes such as tube flatness, orientation, color and opacity are all exported faithfully.

If the **Brush Type** of a Paint FX brush node is **ThinLine**, then the entire Paint FX is exported as curved ribbon geometry. Otherwise Meridian for Maya exports the main, leaf and flower lines as curved ribbons or tubes, according to the brush attributes.

Extension attributes on **pxHair** nodes allow users to choose if hair is exported as curved ribbons or tubes.



**Note.** Meridian for Maya honors the **Cast Shadows** check-box option located in the **Attribute Editor** under the **Shadow Effects** layout for brush nodes and the **Shading > Hair Color Scale** layout for **hairSystem** nodes. By default, Maya sets this option to unchecked for some brush types. If you are expecting to see your Paint FX cast shadows and they are not, make sure this option is checked.

## Particle Systems

Particles and nParticles are exported as lightweight point rendering primitives. The position and size of each particle is exported, as well as per-particle attribute data.

Meridian renders each particle as a sphere and can efficiently render scenes consisting of millions of particles.

## Instancers and Instanced Geometry

Meridian for Maya faithfully exports instanced geometry produced by instancers, even with motion blur.



**Note.** Instanced geometry with motion blur requires a particle cache in order to render properly. Always be sure to cache your particle data when rendering instanced geometry with motion blur.

## Subdivision Surfaces

Legacy subdivision surface support is provided. Meridian for Maya exports subdivision surfaces as a polygon mesh, as specified in the [Tesselation](#) rollout, along with any *uv*-sets or shading group assignments made at the component level.

## Cameras

---

Meridian for Maya exports the contents of the [Resolution Gate](#) of the [Renderable Camera](#) specified in the [Renderable Cameras](#) layout of the [Common](#) tab of the [Render Settings](#).

Extension attributes on the camera node allow motion blur to be enabled or disabled. If enabled, the simulated exposure of the motion-blur effect is specified by the [Shutter Angle](#) attribute of the camera in the [Special Effects](#) rollout.

## Non-Extended Lights

---

Meridian for Maya exports most of the basic attributes for ambient, directional, point and spot lights. Barn doors and decay regions are not supported for spot lights. Area and volume lights are not supported at all.

If the [Use Depth Map Shadows](#) check box in the [Shadows](#) rollup is selected for a directional, point or spot light, Meridian for Maya automatically exports a shadow map generation pass specified by the [Resolution](#), [Filter Size](#) and [Bias](#) attributes. For directional lights, [Use Auto Focus](#), [Width Focus](#) and [Use Light Position](#) are also used to calculate the shadow map. However, the analogous attributes for point and spot lights are ignored.



**Note.** Meridian for Maya uses the [Bias](#) attribute of a shadow map in a slightly different way than Maya. A value that looks good for Maya may produce shadow acne when rendered with Meridian. A test render might be required in some cases.

Extension attributes on the light node specify additional options if [Use Depth Map Shadows](#) is selected.

## Hypershade

Meridian for Maya provides extensive support for Maya shading networks created in Hypershade.

### Automatic Translation

Meridian for Maya automatically translates Maya shading networks into Meridian Shading Language for a large set of supported Hypershade nodes, as documented in the following sections.

Some Hypershade nodes are only supported in certain shading language contexts. For example, the set of nodes supported in a Surface shader may not be the same as the set of nodes supported in a Light shader. The tables in the following sections document how each node is supported in the various contexts.



**Note.** Using nodes that are not supported in the proper context may give unpredictable results. For example, the scene may export properly but then generate compiler errors at render time.

### Surface Materials

Table 4 is a tabulation of surface material nodes supported in each of the specified shading language contexts.

Surface Material	Surface	Light
Anisotropic	•	
Blinn	•	
Hair Tube Shader	•	
Lambert	•	
Layered Shader		
Ocean Shader	•	
Phong	•	
Phong E	•	
Ramp Shader	•	
Shading Map	•	
Surface Shader	•	
Use Background	•	

**Table 4.** Supported surface materials.

[Hair Tube Shader](#) does not support specular shift, and [Use Background](#) only catches shadows.

Ramp attributes are not supported, so only the value at the beginning of a ramp is used for any ramp attributes of the [Ocean Shader](#).

## Volumetric Materials

Volumetric materials are not currently supported in this release, but are planned to be added soon.

## Displacement Materials

Meridian for Maya supports displacement maps that come from an image file. Always be sure to connect a [file](#) node directly to the [Displacement](#) attribute of the [displacementShader](#) node, otherwise Meridian for Maya may not export the displacement shader.

Meridian for Maya uses the [Displacement](#) and [Scale](#) attributes of the [displacementShader](#) node. A [file](#) node should be connected directly to the [Displacement](#) attribute of the [displacementShader](#) node.

The [Image Name](#), [Use Image Sequence](#), [Alpha Gain](#), [Alpha Offset](#) and [Invert](#) attributes of the [file](#) node connected directly to the [Displacement](#) attribute of the [displacementShader](#) are used by Meridian for Maya to export the displacement map.

If a [place2dTexture](#) node is connected to the [file](#) node, the [Coverage](#), [Translate Frame](#), [Rotate Frame](#), [Wrap U](#), [Wrap V](#), [Repeat UV](#), [Offset](#), and [Rotate UV](#) attributes of the [place2dTexture](#) node are also used by Meridian for Maya to export the displacement map.

## Lights

Table 5 is a tabulation of light nodes supported in each of the specified shading language contexts.

Light	Surface	Light
Ambient Light		•
Directional Light		•
Point Light		•
Spot Light		•
Area Light		
Volume Light		

**Table 5.** Supported lighting nodes.

[Spot Light](#) does not support barn doors or decay regions.

Automatic uv-coordinates are generated, as in Maya, when light attributes are texture-mapped. For example, a texture may be assigned to the [Color](#) attribute of a [Spot Light](#) to create a slide-projector effect.

## 2D Textures

Table 6 is a tabulation of 2D texture nodes supported in each of the specified shading language contexts.

2D Texture	Surface	Light
Bulge	•	•
Checker	•	•
Cloth	•	•
File	•	•
Fluid Texture 2D		
Fractal	•	•
Grid	•	•
Mountain	•	•
Movie		
Noise	•	•
Ocean	•	•
PSD File		
Ramp		
Water	•	•

**Table 6.** Supported 2D texture nodes.

## 3D Textures

Table 7 is a tabulation of 3D texture nodes supported in each of the specified shading language contexts.

3D Texture	Surface	Light
Brownian	•	•
Cloud	•	•
Crater	•	•
Fluid Texture 3D		
Granite	•	•
Leather		
Marble	•	•
Rock	•	•
Snow	•	•
Solid Fractal	•	•
Stucco	•	•
Volume Noise	•	•
Wood	•	•

**Table 7.** Supported 3D texture nodes.

[Wood](#) does not support any of the grain attributes.

## Environment Textures

Table 8 is a tabulation of environment texture nodes supported in each of the specified shading language contexts.

Environment Texture	Surface	Light
Env Ball	•	
Env Chrome	•	
Env Cube	•	
Env Sky	•	
Env Sphere	•	

**Table 8.** Supported environment texture nodes.

## General Utilities

Table 9 is a tabulation of general utility nodes supported in each of the specified shading language contexts.

General Utility	Surface	Light
Array Mapper		
Bump 2d	•	
Bump 3d	•	
Condition	•	
Distance Between	•	•
Height Field		
Light Info		
Multiply Divide	•	•
2d Placement	•	•
3d Placement	•	•
+ - Average		
Projection	•	
Reverse	•	•
Sampler Info	•	
Set Range	•	•
Stencil	•	•
Uv Chooser	•	
Vector Product	•	•

**Table 9.** Supported general utility nodes.

[Sampler Info](#) does not provide accurate sampler information for [Point World](#) and [Facing Ratio](#). Connections made to these fields of a [Sampler Info](#) may yield unpredictable results.

## Scalar Utilities

Table 10 is a tabulation of scalar utility nodes supported in each of the specified shading language contexts.

Scalar Utility	Surface	Light
Add Double Linear		
Add Matrix		
Angle Between	•	•
Blend Two Attributes		
Choice		
Chooser		
Curve Info		
Frame Cache		
Mult Double Linear		
Surface Info		
Unit Conversion		

**Table 10.** Supported scalar utility nodes.

## Switch Utilities

None of the switch utilities are currently supported.

## Color Utilities

Table 11 is a tabulation of color utility nodes supported in each of the specified shading language contexts.

Color Utilities	Surface	Light
Blend Colors	•	•
Clamp	•	•
Color Profile		
Gamma Correct	•	•
Hsv To Rgb	•	•
Luminance	•	•
Remap Color		
Remap Hsv		
Remap Value		
Rgb To Hsv	•	•
Surf. Luminance		

**Table 11.** Supported color utility nodes.

## Particle Utilities

Table 12 is a tabulation of particle utility nodes supported in each of the specified shading language contexts.

Particle Utilities	Surface	Light
Particle Sampler	•	

**Table 12.** Supported particle utility nodes.

## Image Planes

Image planes are supported if the [Type](#) option in the [Image Plane Attributes](#) layout is set to [Texture](#). Meridian for Maya uses all of the options in the [Placement](#) layout to position and render the image plane.

## Glow

[Optical FX](#) and glow are not currently supported.

## Texture (TXR) and Displacement (TXD) Files

Meridian requires texture and displacement files to be converted into a special format prior to rendering. Files in this special format typically have the extensions TXR and TXD, respectively.

Meridian for Maya includes an image plugin that allows Hypershade nodes such as [File](#) to open TXR and TXD files. If a Hypershade node uses a file that has already been converted into one of the special TXR or TXD formats, Meridian for Maya uses the existing TXR or TXD file. Otherwise Meridian for Maya attempts to automatically convert image files such as TIFF, BMP, JPG or PNG into a TXR or TXD file, as the case may be, and then use the converted file.

For this reason, converting all of your texture and displacement files into TXR or TXD format prior to using Meridian for Maya may result in significantly faster export times. For more information about how to create TXR and TXD files, please see the [Meridian Rendering Software User Guide](#).

## Custom Shading

---

Users can write custom shaders in Meridian Shading Language and add them to a Hypershade network just like any other Hyperhade node. User-defined inputs and outputs can be connected to other custom shaders or to standard Hypershade nodes, providing a “mix and match” philosophy that allows very sophisticated shader graphs to be created in a few mouse clicks.

Meridian for Maya allows users to create repository and shader nodes. A *repository* node contains a collection of Meridian Shading Language source files and classes. A *shader* node can be connected to any class in any repository and then inserted as a shader instance into a Hypershade network.

## User-Defined Shaders

*Shaders* are special programs written in Meridian Shading Language that extend the functionality of Hypershade.

To implement a shader, a user must write a class in Meridian Shading Language that derives from the standard [Shader](#) interface:

```
interface Shader {
    procedure evaluate(
        const Light lights[], const int num ) const;
}
```

The [Shader](#) interface only has one method called `evaluate`. The method has no return value because it is declared as a procedure. It accepts an array of [Light](#) interfaces, and the number of elements in the array is specified by `num`.

User-defined shaders derived from the [Shader](#) interface may have inputs and outputs. All of the class members declared with the `output` keyword are outputs of the shader, and the remaining class members are inputs. For example, a simple user-defined shader may look like this:

```
// Simple user-defined shader.
// Derives from the standard Shader interface...
```

```

struct myShader : Shader {
    float3 color;    // input color
    output float3 outColor; // output color
    procedure evaluate(
        const Lights lights[], const int num ) const {
        outColor = color; // Set the output to the input value
    }
}

```

In this example, `myShader` has one input called `color` and one output called `outColor`. The entire implementation of the shader is in the `evaluate` method. This example simply copies the contents of the input to the output, but the implementation can be arbitrarily complex.



**Note.** To learn more about the [Light](#) interface, see the [Meridian Rendering Software Developer Guide](#).

## Sunfish Repository Node

The [Sunfish Repository](#) node maintains a collection of Meridian Shading Language source files and classes.

### How to create a Sunfish Repository node

- Select [Window > Rendering Editors > Hypershade](#) from the main Maya menu
- The [Hypershade](#) window opens
- Select [Maya > Utilities > Sunfish Repository](#) from the create bar on the left side of the [Hypershade](#) window, or
- Select [Create > General Utilities > Sunfish Repository](#) from the main [Hypershade](#) menu

Source files may be added or removed from the repository by clicking on the [Add File](#), [Add Folder](#) or [Remove](#) buttons in the [Source Files](#) layout in the [Attribute Editor](#). Adding or removing files causes the repository to become *dirty*.

A dirty repository can be made *clean* by clicking on the [Compile](#) button. Meridian for Maya then compiles all of the source files and checks for syntax errors. If compilation is successful, a list of compiled classes is displayed in the [Classes](#) layout.

Any number of repositories can be created, but the contents of each repository must be unique. For example, it is not currently possible to include the same source file or class definition in two repositories.

## Sunfish Shader Node

The [Sunfish Shader](#) node represents an instance of a class defined in a [Sunfish Repository](#) node. The [Sunfish Shader](#) exposes member variables of the class in

the [Attribute Editor](#) and allows connections to the inputs and outputs of the class to be made.

### How to create a Sunfish Shader node

---

- Select [Window > Rendering Editors > Hypershade](#) from the main Maya menu
- The [Hypershade](#) window opens
- Select [Maya > Utilities > Sunfish Shader](#) from the create bar on the left side of the [Hypershade](#) window, or
- Select [Create > General Utilities > Sunfish Shader](#) from the main [Hypershade](#) menu

After a [Sunfish Shader](#) node is created, it needs to be connected to a [Sunfish Repository](#) node.

### How to connect a Sunfish Shader node to a Sunfish Repository

---

- Select the [Sunfish Shader](#) node
- Open the [Attribute Editor](#)
- Click on the checkered icon to the right of the [Repository](#) navigation field
- The [Connect to a Repository](#) window opens
- Select one of the items in the list and click [Connect](#)



**Note.** If a [Sunfish Shader](#) is already connected to a [Sunfish Repository](#), clicking on the navigation icon to the right of the [Repository](#) navigation field displays the connected [Sunfish Repository](#) in the [Attribute Editor](#).

The connection between a [Sunfish Shader](#) node and a repository can be broken.

### How to break a connection between a Sunfish Shader and a repository

---

- Select the [Sunfish Shader](#) node
- Open the [Attribute Editor](#)
- Right-click on the [Repository](#) navigation label and select [Break Connection](#) from the pop-up menu
- You may then connect the [Sunfish Shader](#) to a different repository

After the [Sunfish Shader](#) node is connected to a repository, any class in the repository can be selected from the [Class](#) drop-down list of the [Sunfish Shader](#) in the [Attribute Editor](#). This automatically displays the user-interface controls for all of the input member variables of the class.

Connections to the inputs and outputs of the class can be made as usual. This allows the [Sunfish Shader](#) node to interoperate in a shading network with all of the other supported [Hypershade](#) nodes.

## Annotations

Class members of a user-defined shader can be decorated in the Meridian Shading Language with annotations. An *annotation* is a unit of metadata that provides extra information about the class member.

Annotations appear in angle brackets after the definition of the class member. For example:

```
// Simple user-defined shader.
// Derives from the standard Shader interface...
struct myShader : Shader {
    float3 color < // input color
        string UIName = "Input Color";
        string UIWidget = "Color";
        float3 UIDefault = float3( 1.0, 0.0, 0.0 ); >;
    float scale < // input scale
        string UIName = "Input Scale";
        float UIMin = 0.0;
        float UIMax = 1.0;
        float UIDefault = 0.5; >;
    output float3 outColor; // output color
    procedure evaluate(
        const Lights lights[], const int num ) const {
        outColor = color; // Set the output to the input value
    }
}
```

The annotations do not have any effect on the definition or evaluation of the shader. The information is provided purely as a convenience for shader authoring tools.

Meridian for Maya recognizes all of the annotations in Table 13 and attempts to create controls in the Attribute Editor with the specified information. If the string "Color" is specified as the value of a `UIWidget` annotation, Meridian for Maya attempts to display the class member as a color.

Type	Name	Description
string	UIName	The name displayed in the user interface
string	UIWidget	The type of widget
any	UIMin	The minimum value of the class member
any	UIMax	The maximum value of the class member
any	UIDefault	The initial (default) value of the class member

**Table 13.** Supported annotations.

## Command-Line Renderer

Meridian for Maya can be run from the Maya command-line rendering interface to export RXF data to a file. In addition, a pipe mode allows RXF data to be streamed directly to the Meridian rendering software.

### Getting Started

---

Using Meridian for Maya from the Maya command-line rendering interface requires the same setup outlined in the Maya documentation.

In particular, be sure the PATH environment variable includes the location of the Maya command-line renderer as specified in the documentation of the particular version of Maya you are using.

Network rendering requires that files, scenes and textures are accessible to each rendering computer on the network. Meridian for Maya must also be installed on each of the network computers.

Use the `-r rxf` option to export an RXF file from the Maya command-line rendering interface:

```
D:\Data>Render -r rxf \\klendathu\Data\scene.mb
```

In this example, the scene located in a Maya file `\\klendathu\Data\scene.mb` is exported to an RXF file in the current working directory, which in this example is `D:\Data`.

### Pipe Mode

---

RXF files can sometimes grow to gargantuan sizes.

Rendering a RXF file while it is being exported can be a great time saver. But some scenes or animation sequences may be so big that a RXF file may fill up an entire hard drive even before an export is complete!

For scenes and animations of such heroic size and complexity, it is highly recommended to run Meridian for Maya in pipe mode. In pipe mode, RXF data is streamed directly to Meridian without any need to store it on disk. Meridian can then literally “render until the cows come home.”

To use pipe mode, simply use the `-pipe` option and specify `fish.exe` as the program to pipe the RXF data to.

For example, to pipe the RXF content of a Maya scene located in a file `\\Klendathu\Data\scene.mb`, enter the following at the command prompt:

```
D:\Data>Render -r rxf -pipe fish.exe \\Klendathu\Data\scene.mb
```

This will pipe the RXF content directly to Meridian, which will render the scene and then put the output files in the current working directory, which in this example is `D:\Data`.

For pipe mode to function properly, the location of the `fish.exe` program must be in the `PATH` environment variable.



**Note.** The `PATH` environment variable already contains the location of the `fish.exe` program if you run the `Render` command from within a Meridian console window. For more information on Meridian console windows, see the [Meridian User Guide](#) document.

## Working Directory

When operating Meridian for Maya from the Maya command-line rendering interface, all output is created and stored in a working directory. This is true even when using pipe mode.

By default, Meridian for Maya uses the current directory as the working directory. The `-wd` option can be used to specify a different working directory. For example:

```
D:\Data>Render -r rxf -wd E:\Output \\Klendathu\Data\scene.mb
```

forces Meridian for Maya to use `E:\Output` as the working directory. The `-wd` option can also be used with pipe mode.