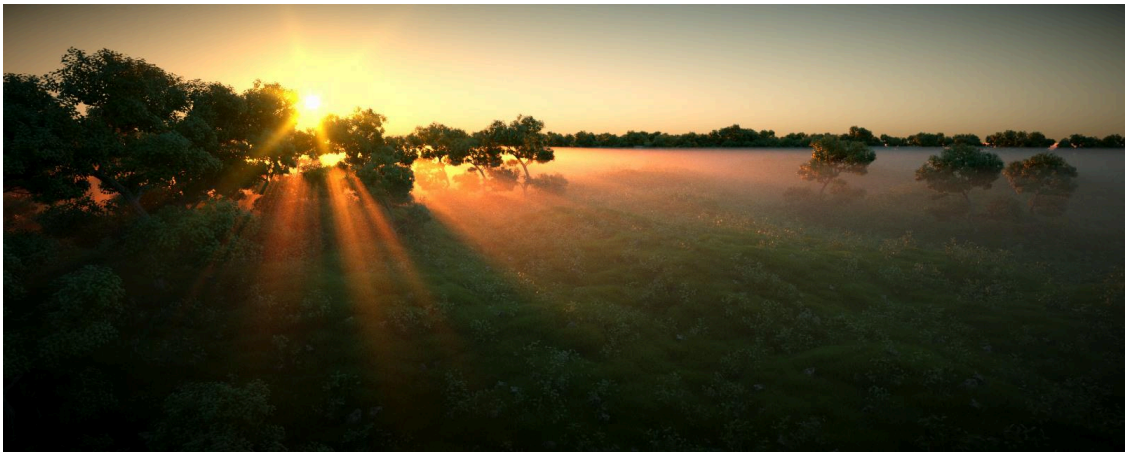


OctaneRender® Unity User Manual

Version 1.1.1.1330

Cover image by OTOY, Inc.

https://unity.otoy.com

Contents

Installation

The OctaneRender® for Unity® installation process consists of three steps:

1. Installing the appropriate version of Unity.
2. Confirming hardware requirements and drivers.
3. Installing the OctaneRender for Unity integration.

Installing the Appropriate Version of Unity®

OctaneRender® for Unity® requires downloading and installing Unity version 2017.3 or later. To get the latest version of Unity, go to https://store.unity.com.

Confirm Hardware Requirements and Drivers

OctaneRender® is a CUDA® 9.1 application. It requires a current NVIDIA® graphics driver (version 391.35 or higher).

Hardware Requirements

OctaneRender® requires an NVIDIA® graphics card with support for compute capability 3.0. More CUDA® cores available on the **GPU**[1] means faster renderings.

System requirements include:

- GPU: NVIDIA Kepler, Maxwell, Pascal™, GTX Titan, or Volta™ GPU and 4 GB+ VRAM
- RAM: 8 GB minimum, 16 GB or higher recommended

[1]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Drivers

OctaneRender® for Unity® requires the latest CUDA® 9.1 drivers. For the latest drivers, please refer to the NVIDIA® Driver Downloads website.

Software Requirements

- Operating system: Windows® 7® or higher (64-bit); Macintosh® - macOS® 10.12 Sierra or higher
- Internet access for license verification

Installing the OctaneRender® for Unity® Integration

You can install the OctaneRender® for Unity® integration by downloading one of the scenes OTOY® provides in the Unity Asset Store. The Installer Scene is built to be lightweight and friendly, so we recommend it for getting started (Figure 1).

- For OctaneRender Prime, all you need to do is download the Installer Scene and open it in the **Unity Editor**.
- To upgrade to OctaneRender Studio or OctaneRender Creator, please visit OTOY's Unity page. The Installation steps are the same. However, once you open OctaneRender in the Unity Editor, you will be directed via email to complete your user account and select plugins, or access to OctaneRender Cloud®.

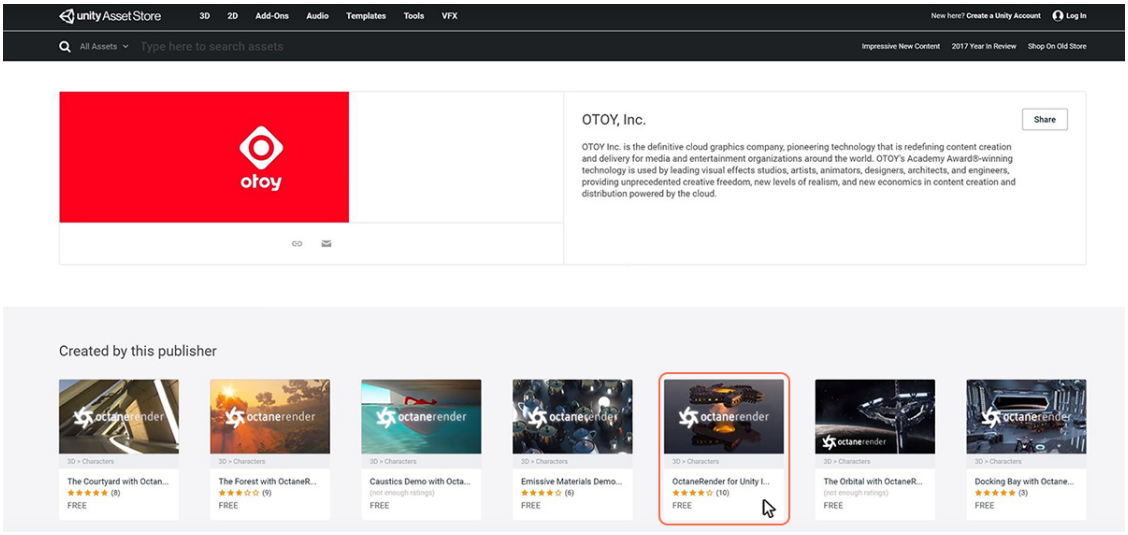


Figure 1: Selecting the Installer Scene from the Unity Asset Store

You can also import the Installer Scene directly from Unity's Asset Store and into a project (Figure 2).

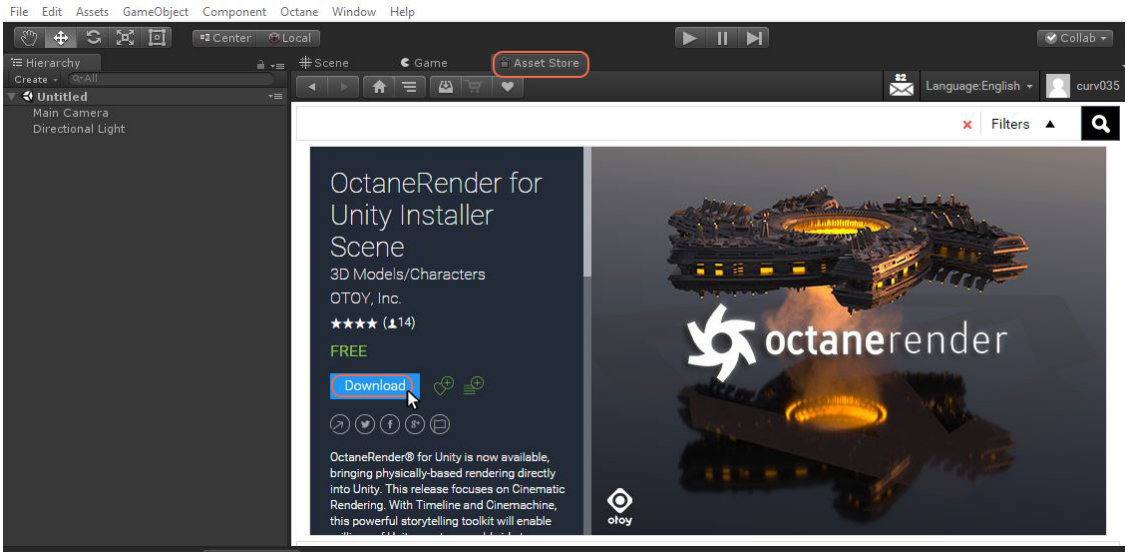


Figure 2: Adding the Octane Installer scene directly in Unity

After the download is complete, the **Import Unity Package** window opens. Click the **Import** button to import all the contents from the Installer Scene into the current Unity project (Figure 3).

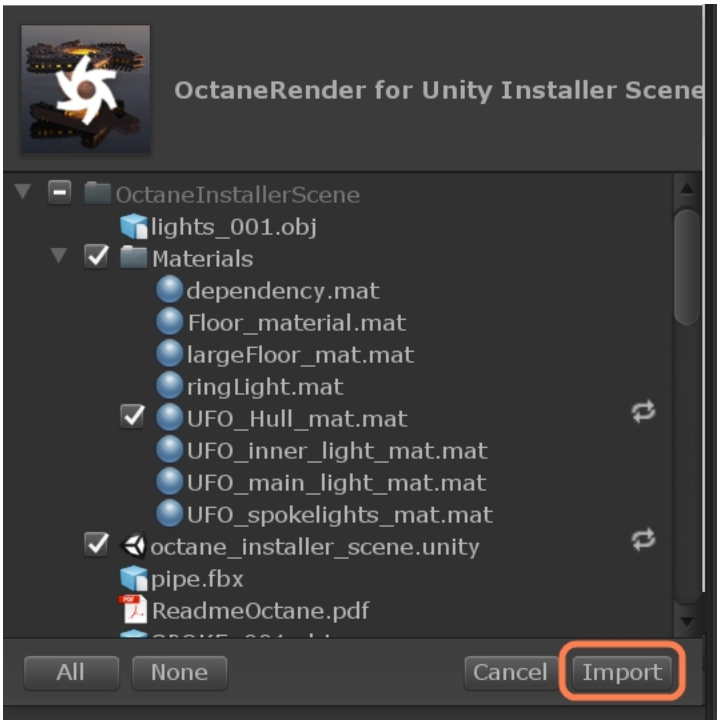
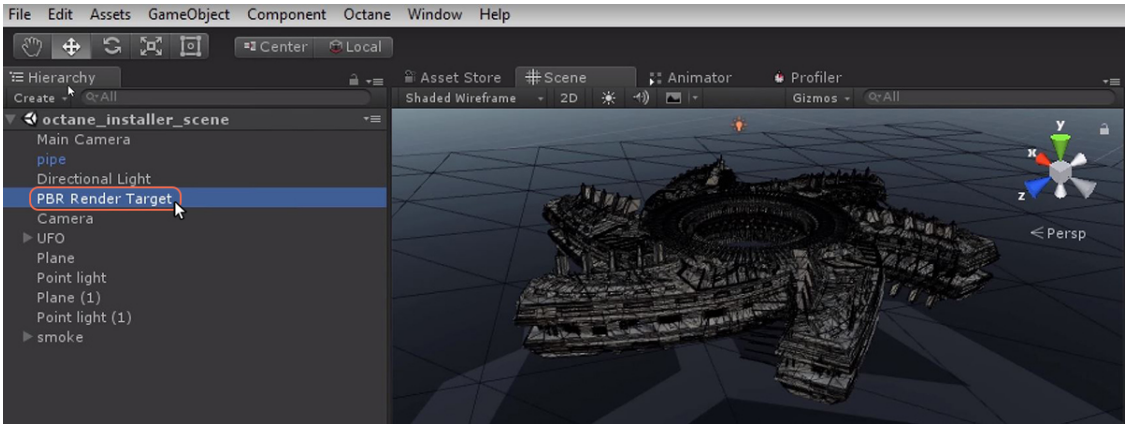


Figure 3: Importing the contents of the Octane Installer Scene into Unity

Once the Installer Scene is imported, select the **PBR**[1] **Render Target** in the **Hierarchy** window (Figure 4).



[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Figure 4: Selecting the PBR Render Target in the Hierarchy window

From the **Inspector** window, click the **Load Octane** button (Figure 5).

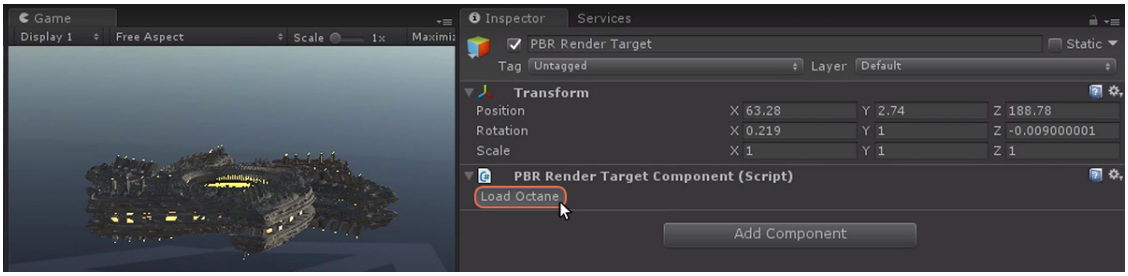


Figure 5: Activating the Octane plug-in from the PBR Render Target's Inspector window

After clicking the Load Octane button and activating the Octane render engine, the Inspector window provides many rollouts for controlling the render process. To start the rendering process, click the **Render** button (Figure 6).

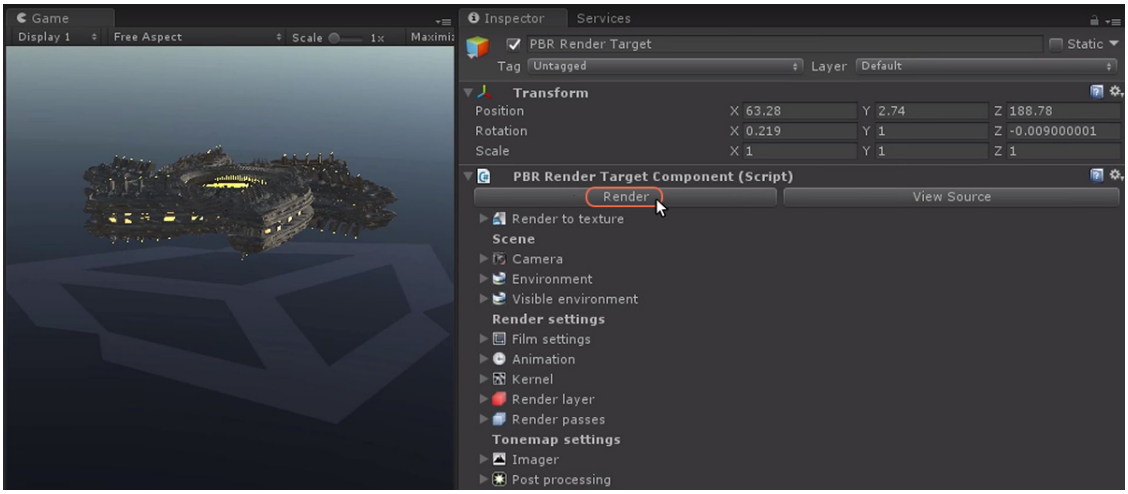


Figure 6: Clicking the Render button to start the rendering process

Clicking the Render button opens the **PBR Viewport** window, where the rendering process continuously updates until OctaneRender reaches the specified sampling level (Figure 7).

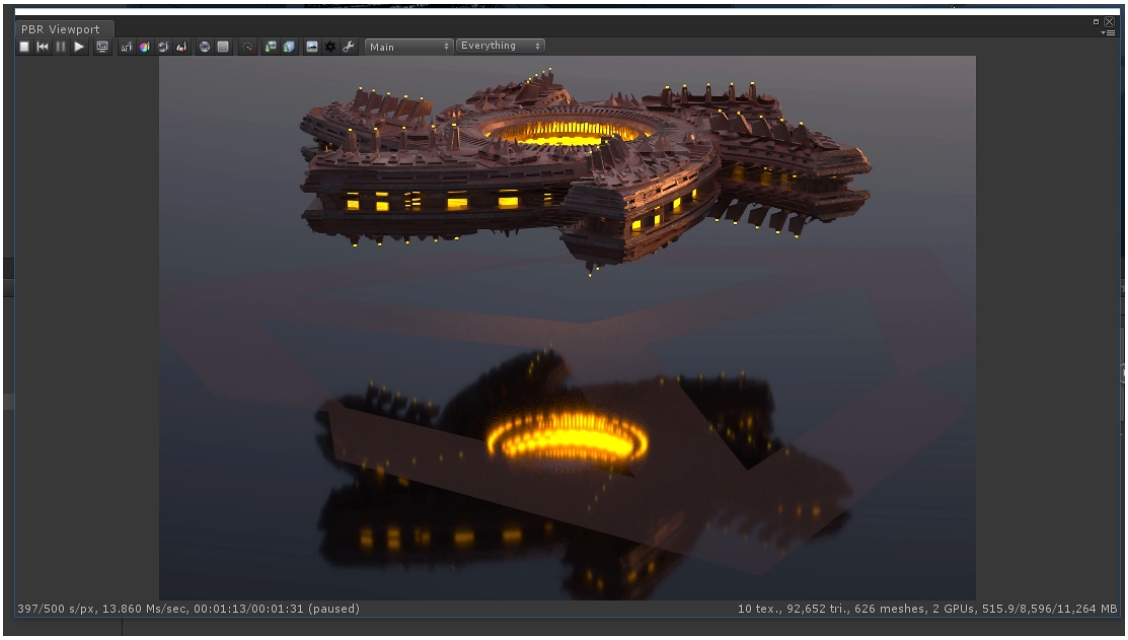


Figure 7: The Octane Installer Scene rendering in the PBR Viewport

The Installer Scene contains the OctaneRender for Unity plug-in, which you can import into any Unity scene as a custom package without the additional assets contained in the Installer Scene. To accomplish this:

1. Create a new Unity scene, then from the **Assets** menu, choose **Import Package**, then click **Custom Package** (Figure 8).

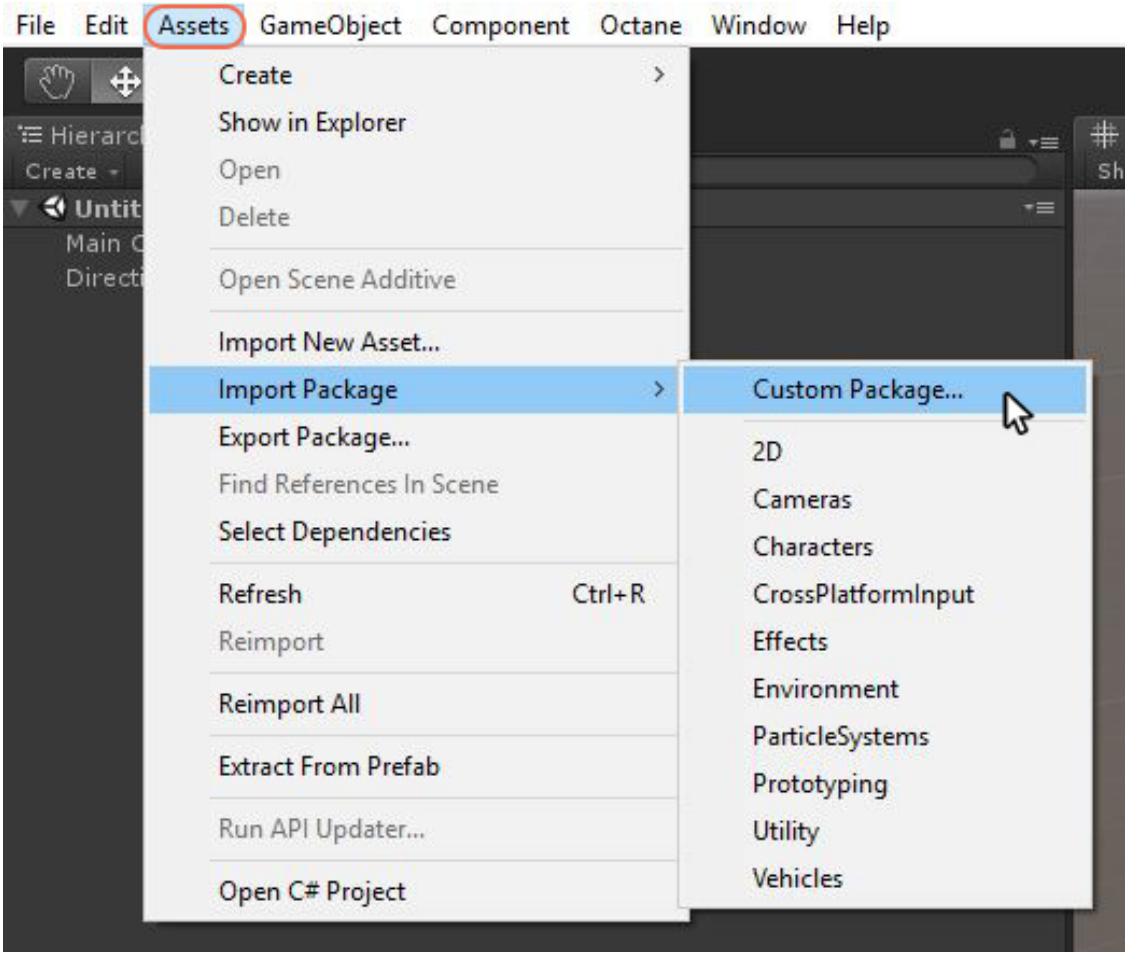


Figure 8: Importing the Octane Custom Package

2. Navigate to where you saved the Octane Installer scene and select the **Hull Material[1].Unitypackage** file (Figure 9).

[1]The representation of the surface or volume properties of an object.

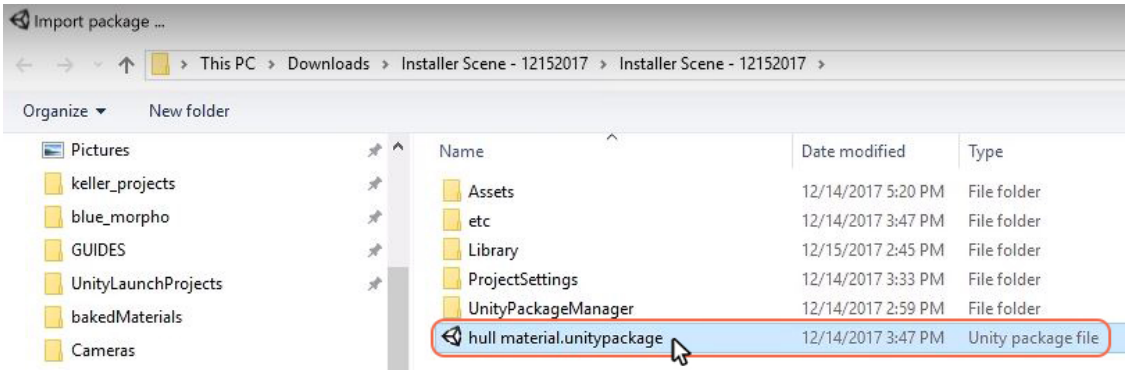


Figure 9: Importing the Hull Material custom package

3. In the **Import Unity Package** window, deselect everything except **Plugins** (Figure 10), then click **Import**.

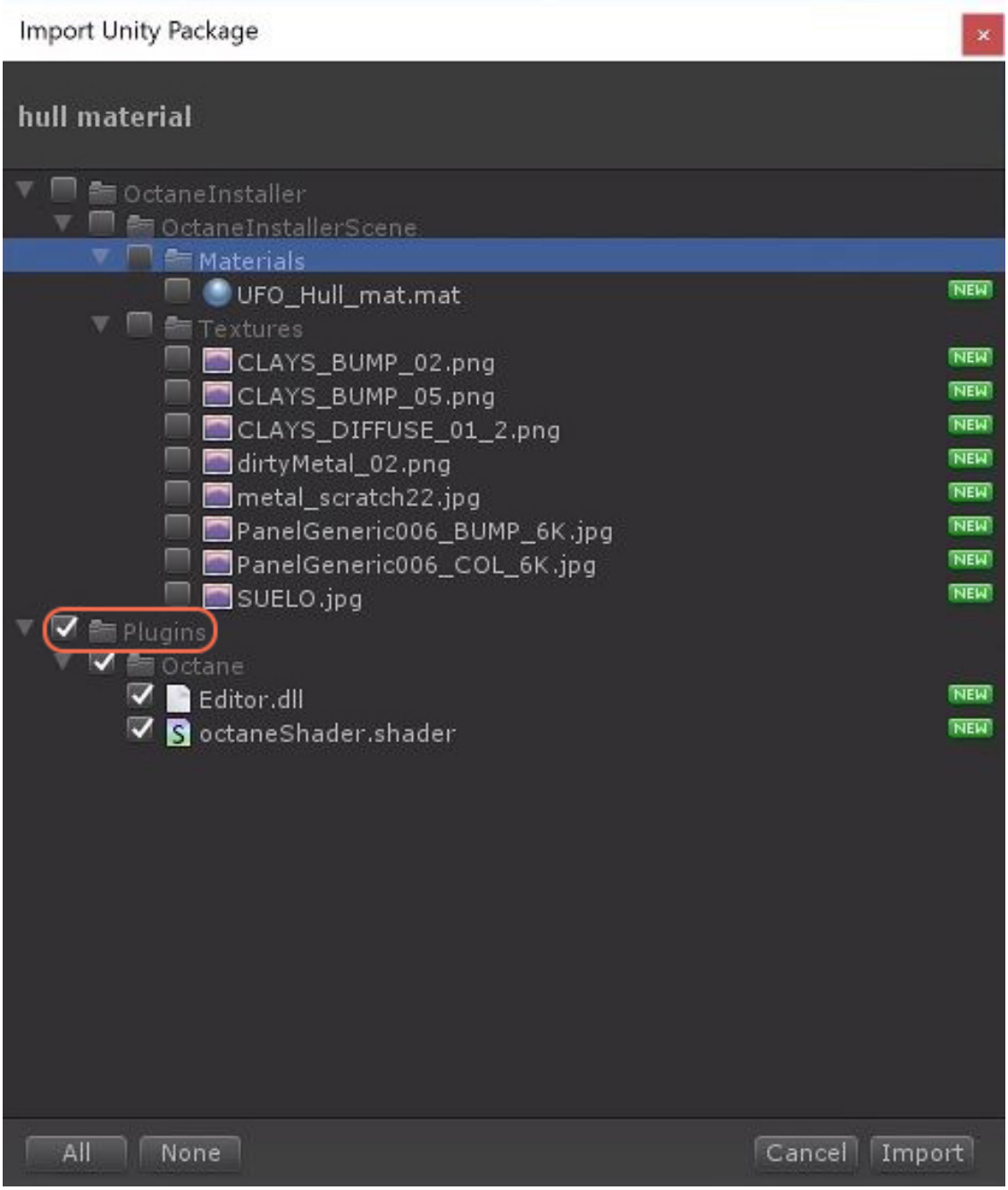


Figure 10: Deselecting all the assets except for Plugins in the Import Unity Package window

4. After importing the OctaneRender custom package plug-ins, the **Octane** Menu displays in the **Unity** menu bar (Figure 11).

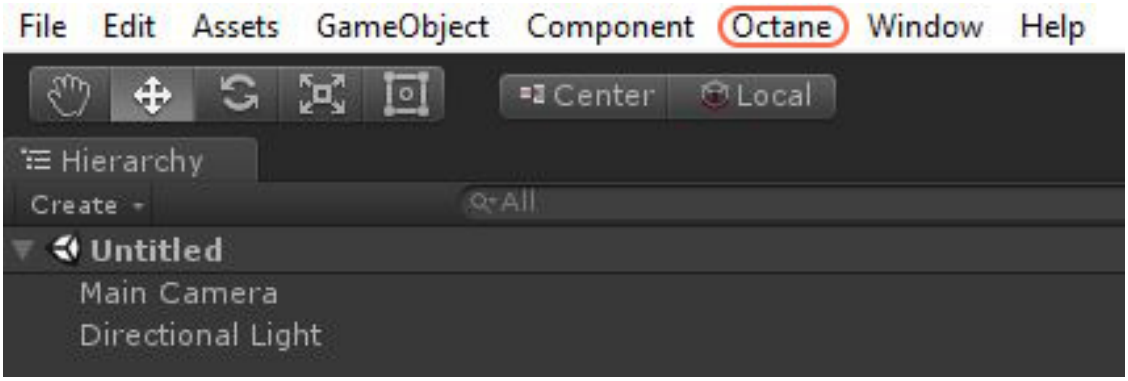


Figure 11: The Octane menu in Unity

5. From the Octane menu, click **Settings** to open the **PBR Settings** window, and click the **Load Octane** button (Figure 12).

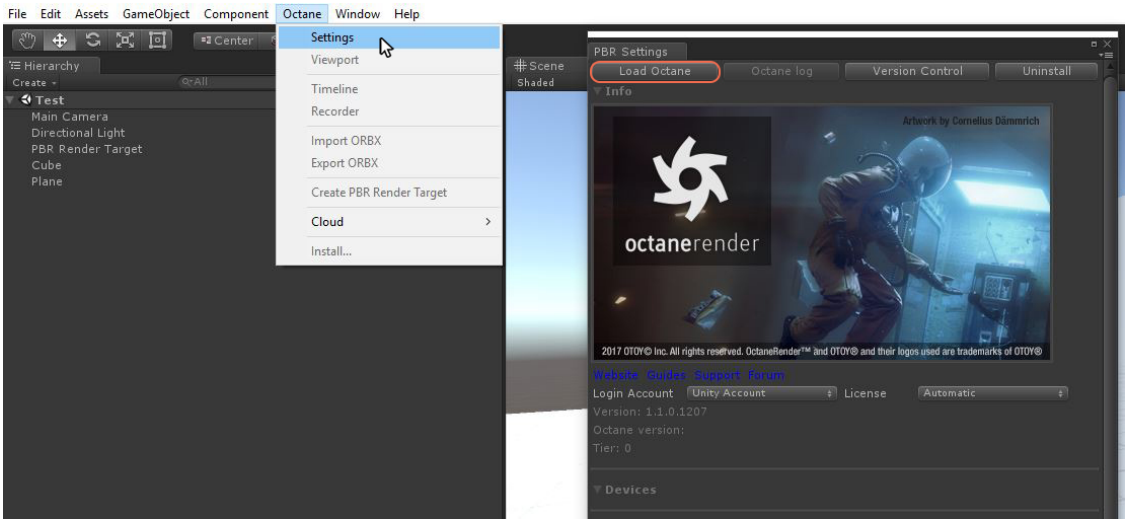


Figure 12: Opening the PBR Settings window from the Octane menu

Overview of OctaneRender® for Unity®

The integration of OctaneRender® in Unity® promises stunning gains in photorealism, **VR**[1] performance, and productivity. Path-tracing techniques respect the physical properties of light, which gives the renderings much better quality than applications using CPU ray-tracing techniques. OctaneRender's **Direct Lighting** mode provides fast, progressive rendering, making it a perfect companion to Unity's real-time rendering capabilities.

As the OctaneRender for Unity integration progresses, future releases will include more features and Octane tools. You can learn more about the roadmap and upcoming releases at https://unity.otoy.com.

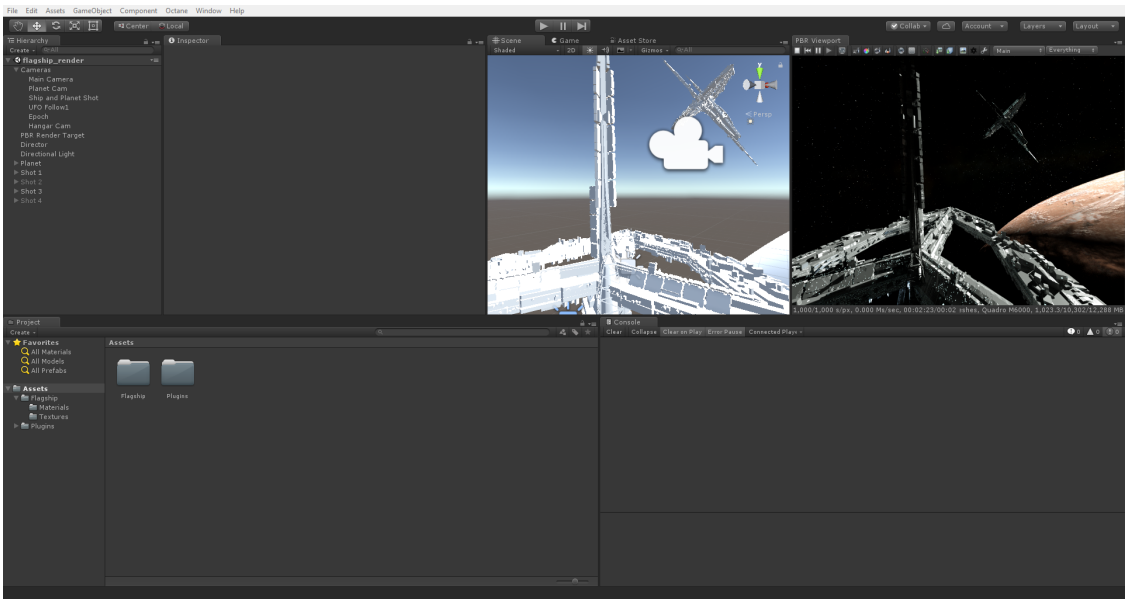


Figure 1: The Unity interface with OctaneRender for Unity loaded

[1]Immersively engaging and experiencing depth perception in a three dimensional scene through stereo vision goggles and head-mounted displays.

FAQ and Performance Tips

- Game mode is not needed for using the timeline to create and render animations.
- Shaders other than the **Standard** and **Standard (Specular[1] Setup)** shaders are not supported by OctaneRender® for Unity®.
- OctaneRender maps the default Unity skybox material and sun directional light to the OctaneRender environment node, but ignores other directional lights or skybox materials in this release.
- Only tri-mesh Unity assets are auto-converted (particles and vegetation are mapped to OctaneRender volumetric and scatter nodes in subsequent releases).
- To optimize speed in the OctaneRender **PBR[2] Vviewport**:
    - Turn off updates to transforms, meshes, and animations in the **PBR Viewport** when only the camera is moving. Due to the nature of the Unity scene graph API that OctaneRender uses in the **Unity Editor**, OctaneRender polls every instance in the scene to check for changes, which can significantly affect speed when there are many objects. Planned improvements to the Unity scene API in future releases will address these performance issues.
    - Detach any inactive objects in the PBR component while it is exposed to scene objects in use by OctaneRender.

1Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

2A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

OctaneRender® for Unity® Interface Overview

Once you load OctaneRender® for Unity®, the **Octane** Menu and other OctaneRender features become available in the **Unity Editor**. You can customize the Unity Editor workspace to whatever layout is most useful to you (Figure 1).

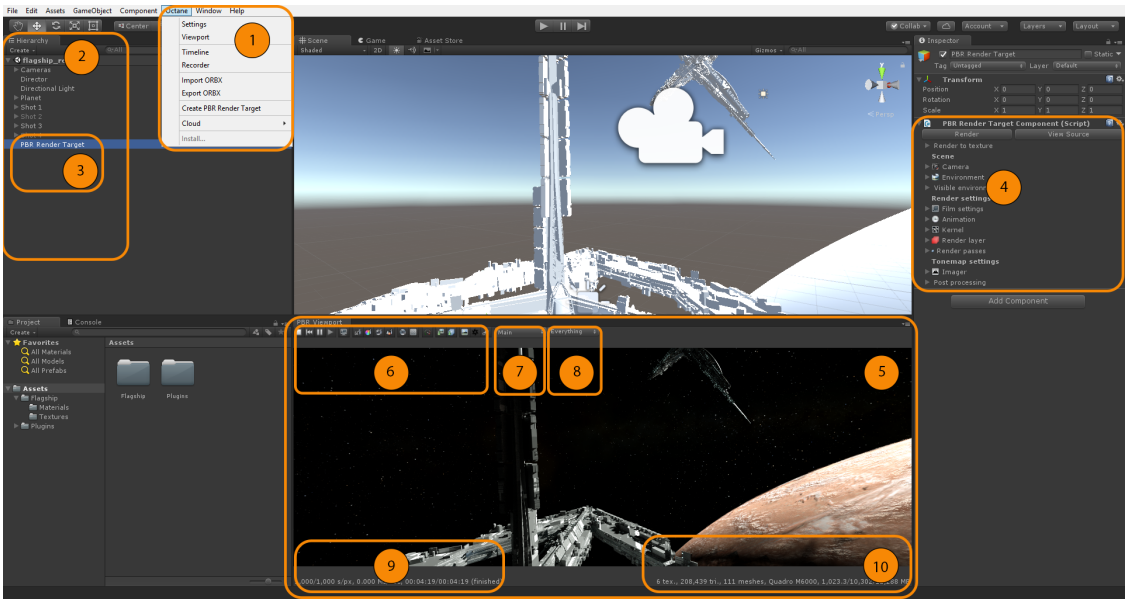


Figure 1: The Unity interface with OctaneRender for Unity loaded

Important Unity interface windows when working with OctaneRender for Unity:

1. **Octane Menu** — This menu gets populated once you install OctaneRender. Otherwise, you need to download the OctaneRender for Unity web installer from the Unity Asset Store, run it with administrative privileges, and then install OctaneRender from within Unity.
2. **Hierarchy Window** — Shows the current project in an outline format. You can add a **PBR**[1] **Render Target** to a Unity scene directly in this window by right-clicking on **PBR Render Target**, and then select **Create PBR Render Target**.

---

[1]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

3. **PBR Render Target Game Object** — This is the essential **Game Object** item that provides the controls for OctaneRender elements in the scene.
4. **Inspector Window** — You can access the PBR Render Target's parameters here.
5. **PBR Viewport** — The render updates in real-time in this window, which also displays the render's progress.
6. **Render Viewport Toolbar** — Controls for working in the PBR Viewport.
7. **Pass Selection** — If you enable OctaneRender **Render Layers**[1] and **Render Passes**[2], you can view each pass through this dropdown menu. Otherwise, only the beauty/main pass is shown by default.
8. **Render Component Filter** — This dropdown menu determines what the OctaneRender engine updates from Unity. It should not be confused with a method for simply hiding or showing scene assets in the PBR Viewport - it is a performance optimization feature. For instance, if a scene has no animations, you can disable the **Animations** checkbox.
9. **Render Progress Indicator** — Displays render samples, render speed, elapsed time, and estimated render time.
10. **GPU**[3] **Status Indicator** — Displays information related to the GPU used for rendering.

---

[1]Render layers allow users to separate their scene geometry into parts, where one part is meant to be visible and the rest of the other parts "capture" the side effects of the visible geometry. The layers allow different objects to be rendered into separate images where, in turn, some normal render passes may be applied. The Render layers are meant for compositing and not to hide parts of the scene.
[2]Render passes allow a rendered frame to be further broken down beyond the capabilities of Render Layers. Render Passes vary among render engines but typically they allow an image to be separated into its fundamental visual components such as diffuse, ambient, specular, etc..
[3]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

PBR Settings Window

The **PBR**[1] **Settings** window is the primary information control center for OctaneRender®. It monitors **GPU**[2] performance, enables out-of-core rendering, and monitors RAM and VRAM usage. Account information and the Octane log are found here as well. You can access this window from the **Octane** menu (Figure 1).

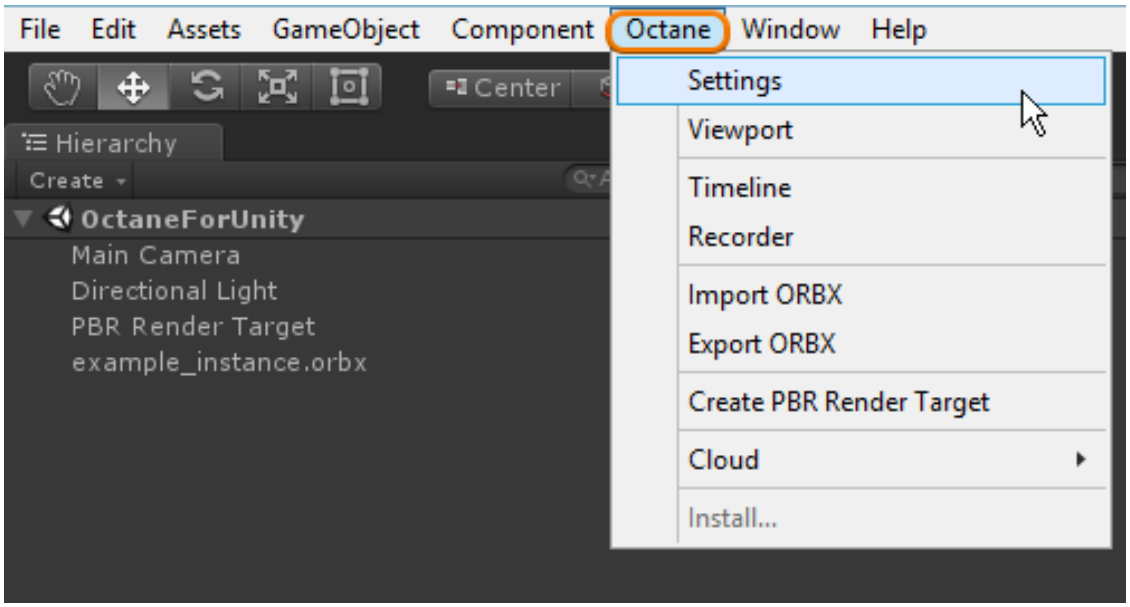


Figure 1: Accessing the PBR Settings window from the Octane menu

Most importantly, this window is where you activate the OctaneRender engine for a Unity® scene. To activate OctaneRender, click on the **Load Octane** button (Figure 2). This activates the information and checkboxes (Figure 3).

[1]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.
[2]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

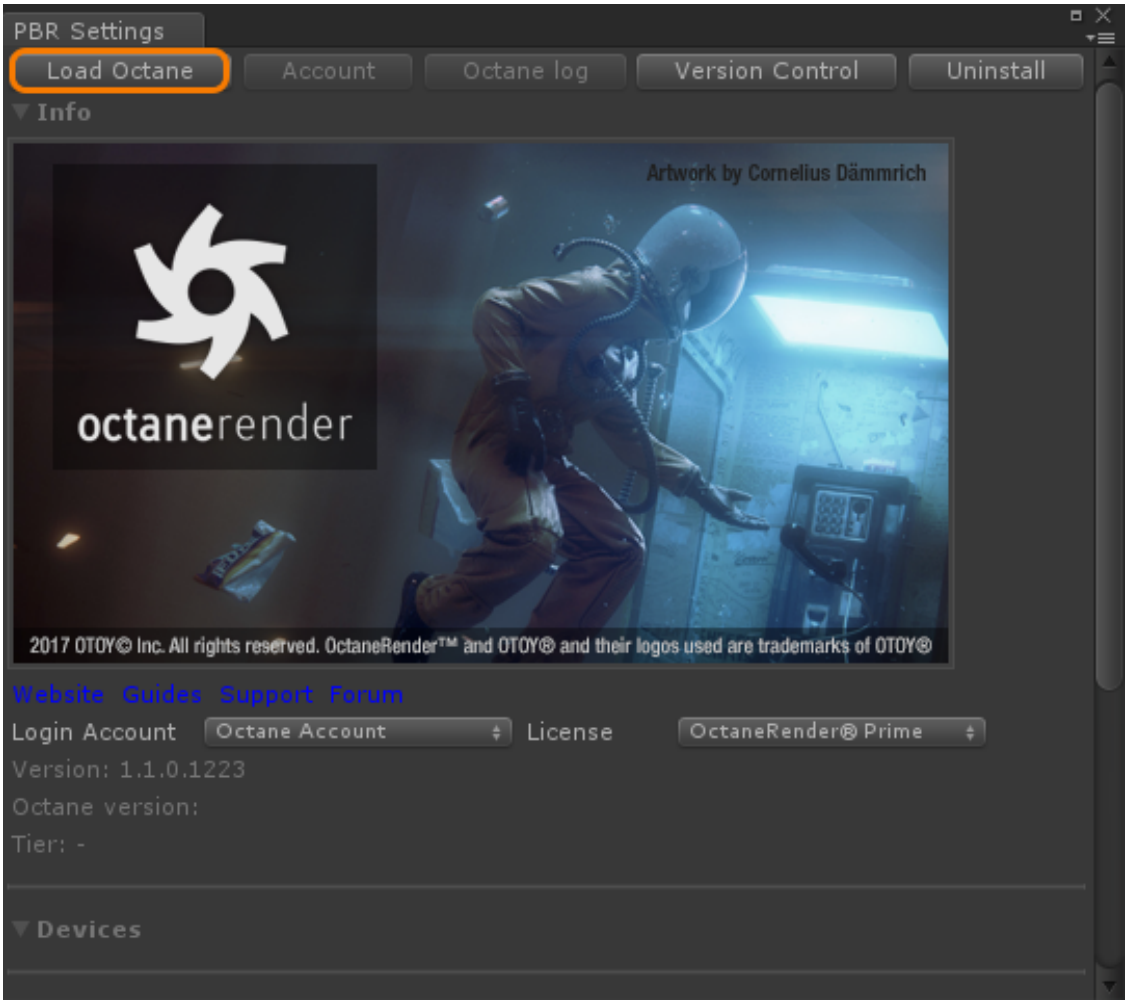


Figure 2: Loading Octane from the PBR Settings window

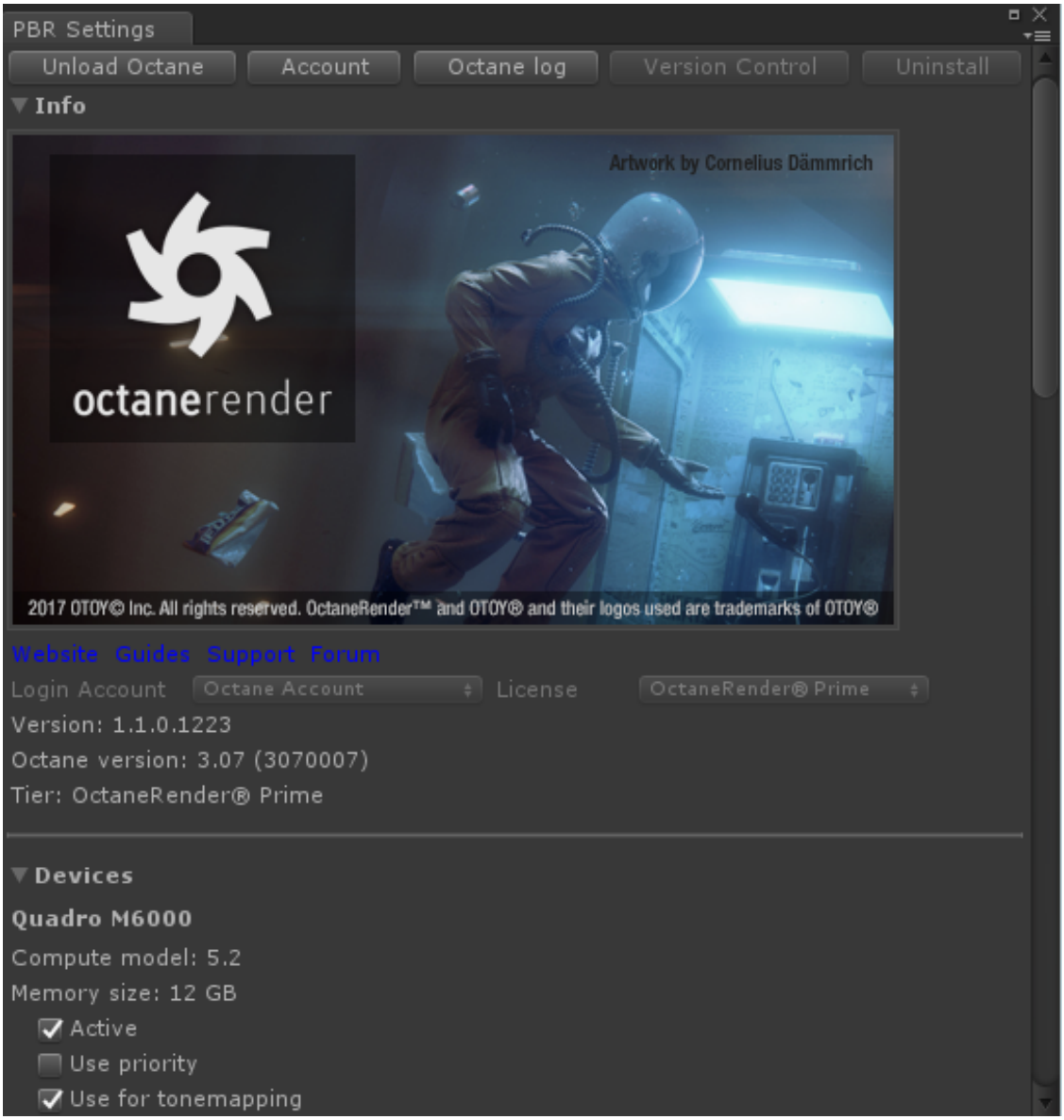


Figure 3: Pressing the Load Octane button activates the parameters

The **Devices** rollout lists the current GPU used for rendering along with a series of checkboxes:

- **Active** - This activates the OctaneRender engine to render a Unity scene. If this checkbox is disabled, you can navigate the Unity scene with the **PBR Viewport** updating the rendering.
- **Use Priority** - Determines the GPU that OctaneRender uses first in multi-GPU machines.

- **Use For Tonemapping** - Enables real-time tonemapping to build the preview image in the PBR Viewport window for specific GPUs.
- **Use For Denoising** - PENDING DESCRIPTION FROM MAXC

The **Out Of Core** rollout addresses how rendering occurs once the GPU runs out of RAM:

- **Enabled** - Activates the Out-Of-Core rendering feature.
- **RAM Limit (GB)** - Specifies the amount of system RAM used for Out-Of-Core rendering.
- **GPU Headroom (MB)** - Minimum amount of GPU memory to keep as a buffer during Out-Of-Core rendering.

The **Baking** section provides functions for batch baking object materials and lighting that are in development and not yet fully supported:

- **Bake** - Starts the batch baking process for all **Baking Group** IDs.
- **Apply Baking Results** - Applies the baked textures to the appropriate assets in the Unity scene.
- **Revert Baking Results** - Removes the baked textures from the scene assets.

Please visit https://unity.otoy.com/ for the latest updates on the integration of OctaneRender into Unity.

PBR Viewport Window

The **PBR**[1] **Viewport** window is where OctaneRender® renders a Unity® scene (Figure 1). The PBR Viewport requires a **PBR Render Target** asset in the scene. This is covered in more detail in the **"PBR Render Target" on page 34** section.

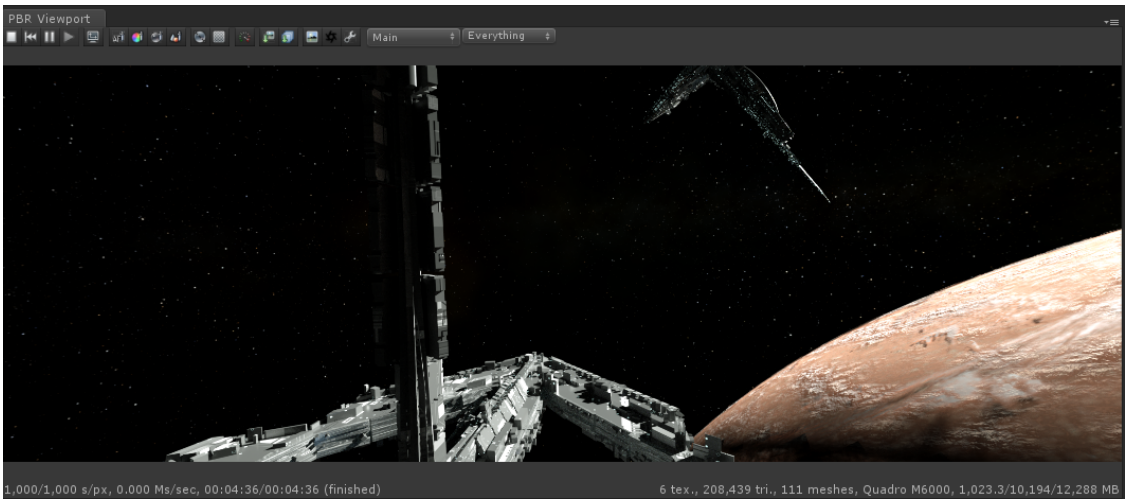


Figure 1: The PBR Viewport window accessed from the Octane menu

You can access the PBR Viewport either from the **Octane** menu (Figure 2), or by clicking on the PBR Render Target, then clicking on the **Render** button in the **Inspector** window (Figure 3).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

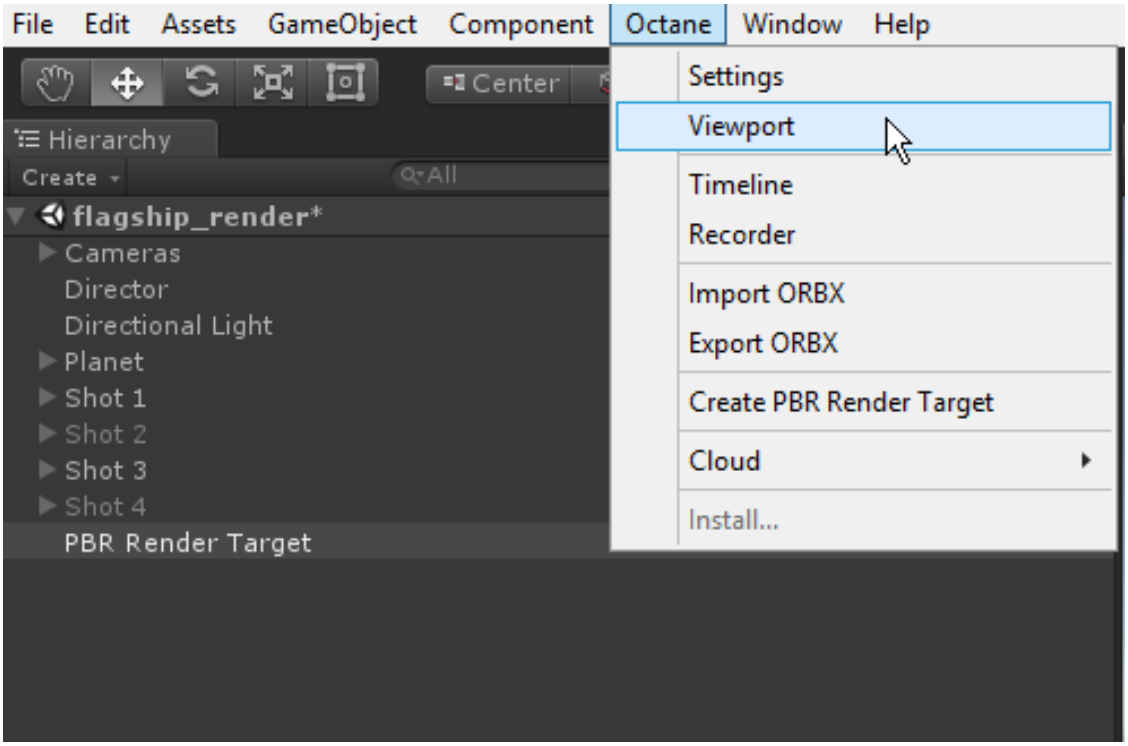


Figure 2: Accessing the PBR Viewport via the Octane menu

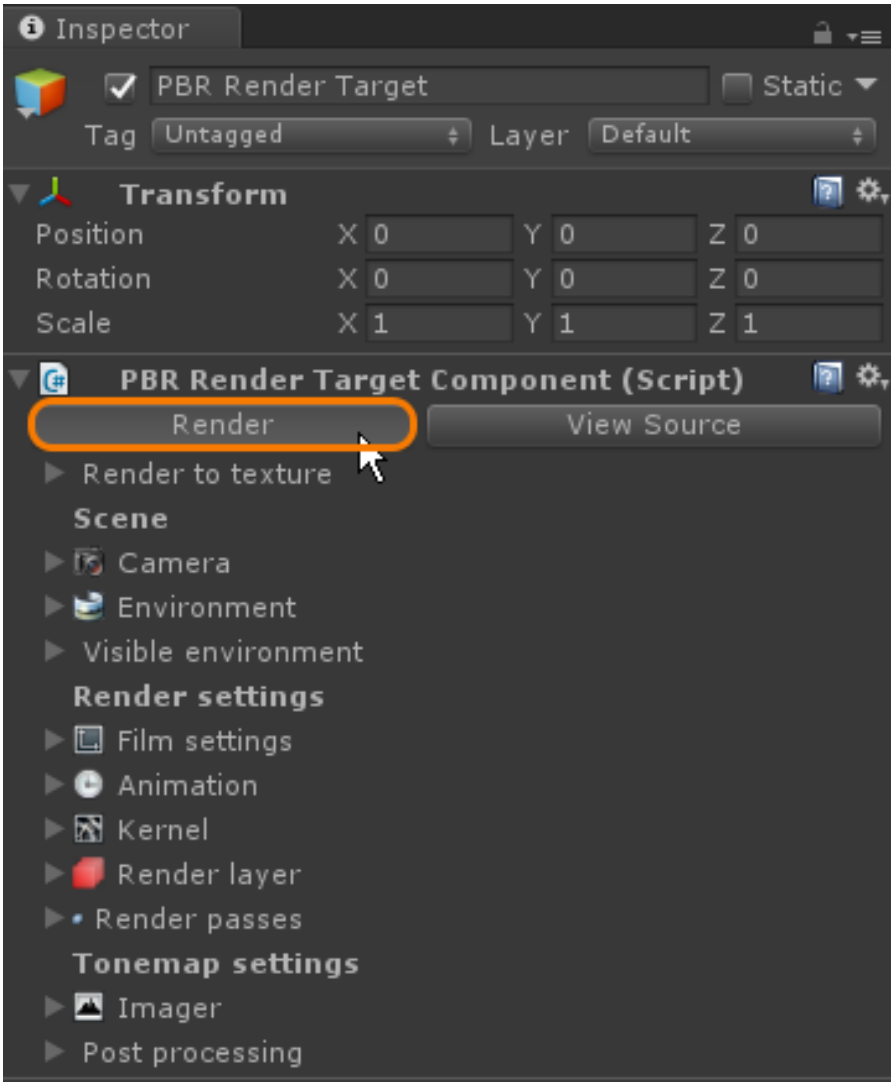


Figure 3: Accessing the PBR Viewport via the Inspector window

The PBR Viewport constantly updates the render, and it displays the render's progress. If you enable OctaneRender **Render Layers**[1] and **Render Passes**[2], you can view each pass in this window. Otherwise, this window shows only the beauty/main pass by default (Figure 4).

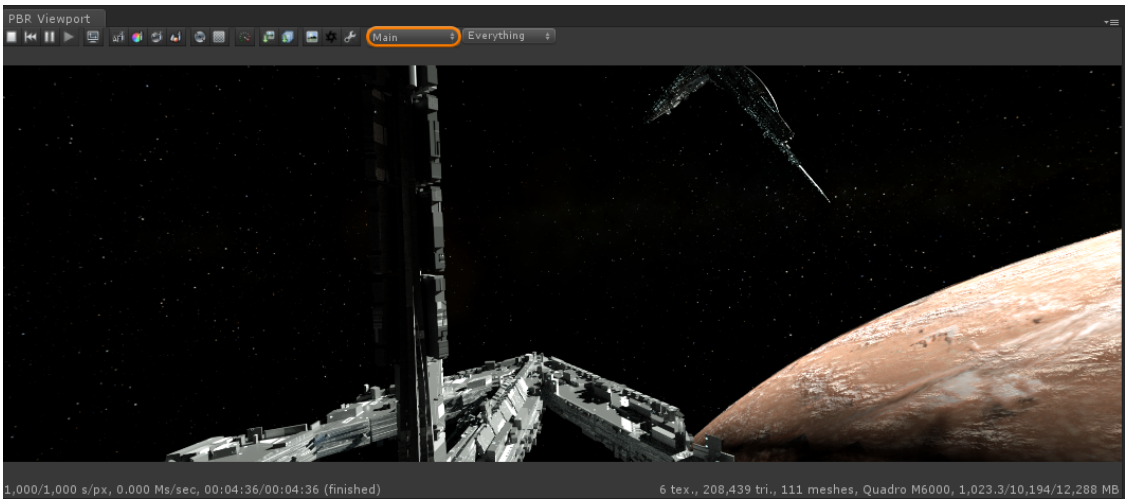


Figure 4: The Main/Beauty pass is rendered by default in the PBR Viewport

The PBR Viewport contains a row of buttons along the top of the window that consists of useful viewport-specific tools (Figure 5).

Figure 5: PBR Viewport toolset

The bottom of the PBR Viewport also displays useful information related to the current render progress (Figure 6) and the **GPU**[3] status (Figure 7).

---

[1]Render layers allow users to separate their scene geometry into parts, where one part is meant to be visible and the rest of the other parts "capture" the side effects of the visible geometry. The layers allow different objects to be rendered into separate images where, in turn, some normal render passes may be applied. The Render layers are meant for compositing and not to hide parts of the scene.
[2]Render passes allow a rendered frame to be further broken down beyond the capabilities of Render Layers. Render Passes vary among render engines but typically they allow an image to be separated into its fundamental visual components such as diffuse, ambient, specular, etc..
[3]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

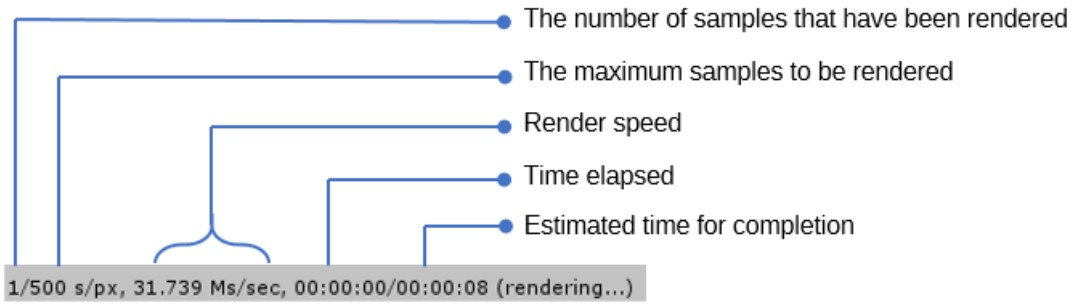


Figure 6: Render progress status displayed in the PBR Viewport window

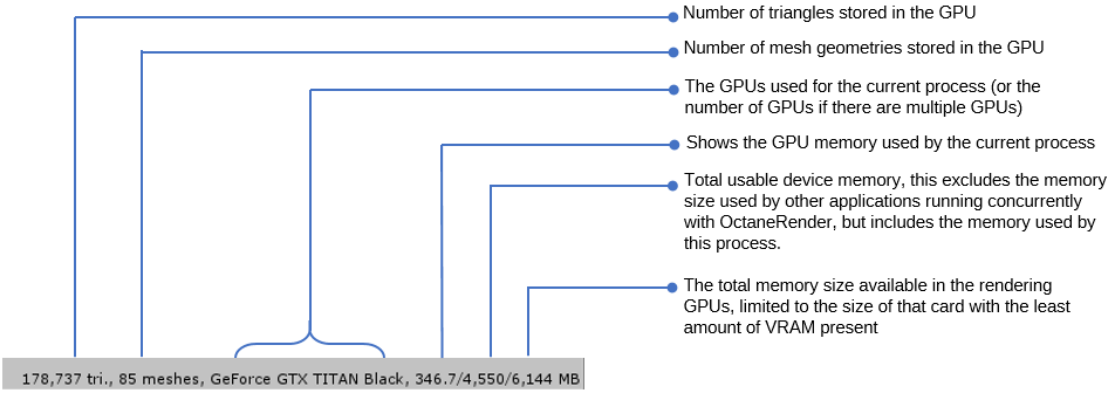


Figure 7: GPU status displayed in the PBR Viewport window

Timeline

The **Timeline** window renders animations designed with the Unity® Timeline tool. You can access it from the **Octane** menu (Figure 1). This window is a batch rendering tool, and is covered in detail in the Animation section.

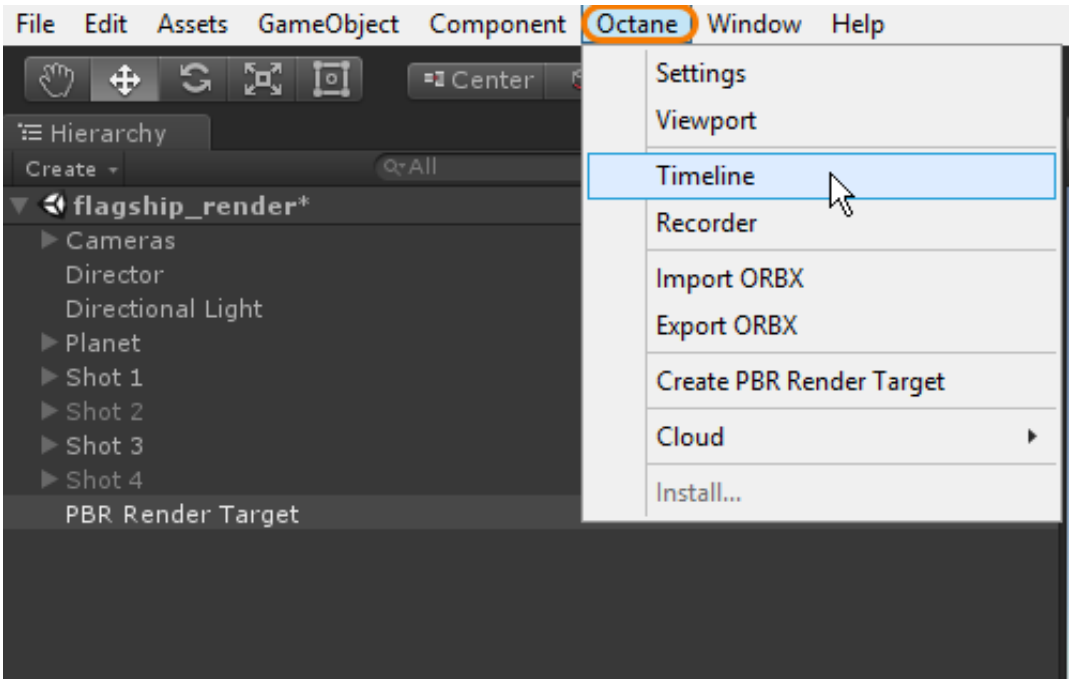


Figure 1: Accessing the Octane Timeline window

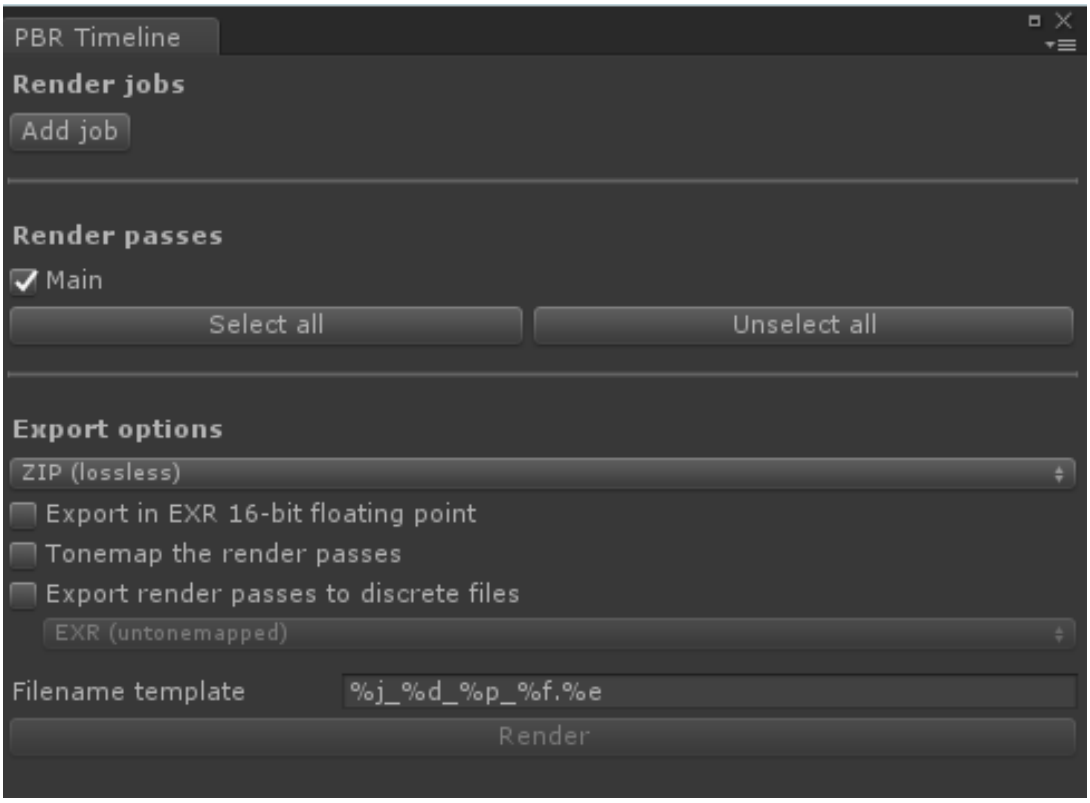


Figure 2: The PBR[1] Timeline window

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Recorder

Recorder combines **Batch Rendering**[1] and **Play** mode so that Play mode can render animations and save the rendered images to disc (Figures 1 and 2). This works for animation built with the Unity® **Timeline** as well as animations triggered during gameplay. The **Recorder** section (under Animation) covers this function in more detail.

[1] The process of assigning sequential portions of frames to be rendered across multiple systems.

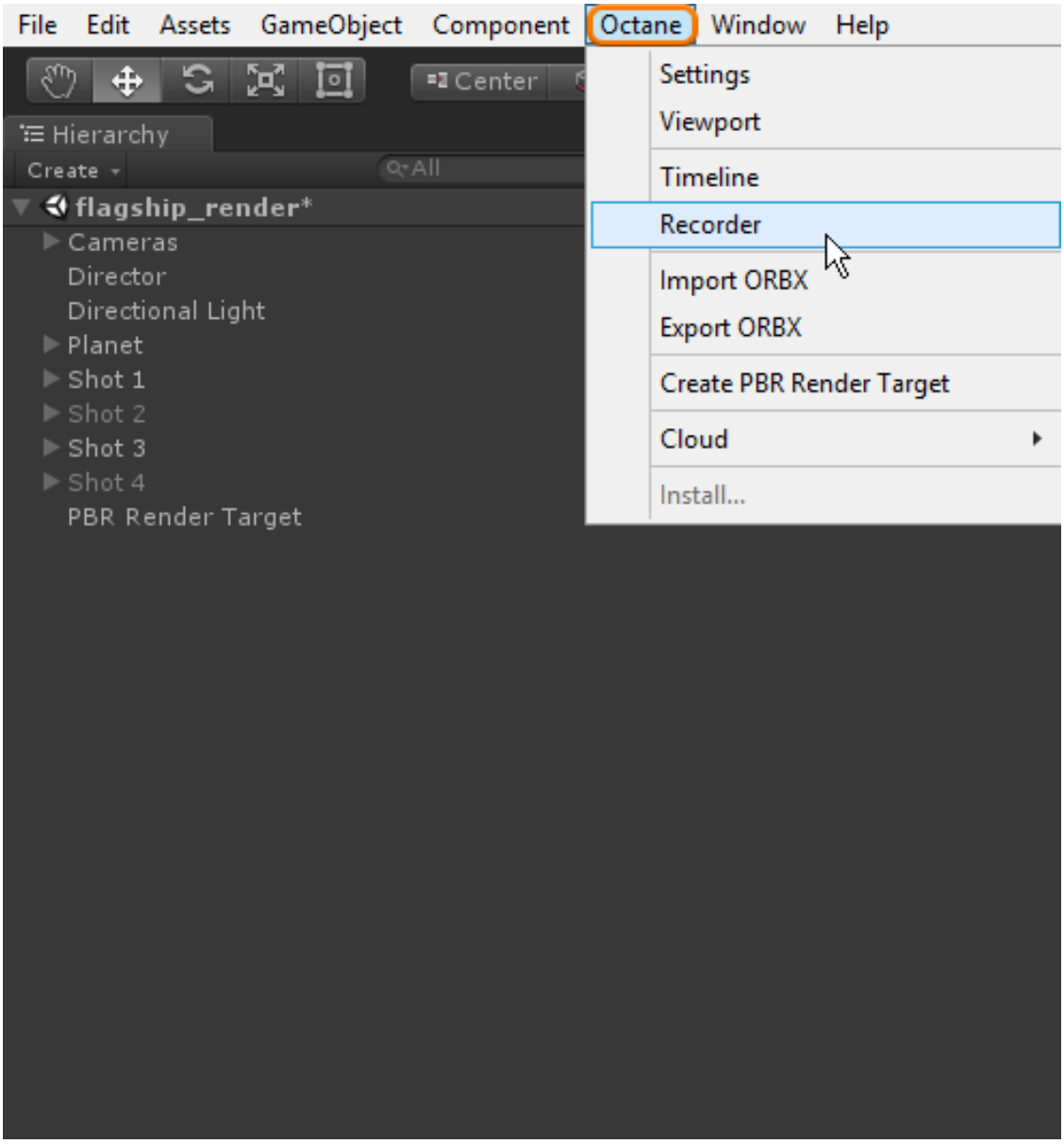


Figure 1: Accessing the Recorder from the Octane menu

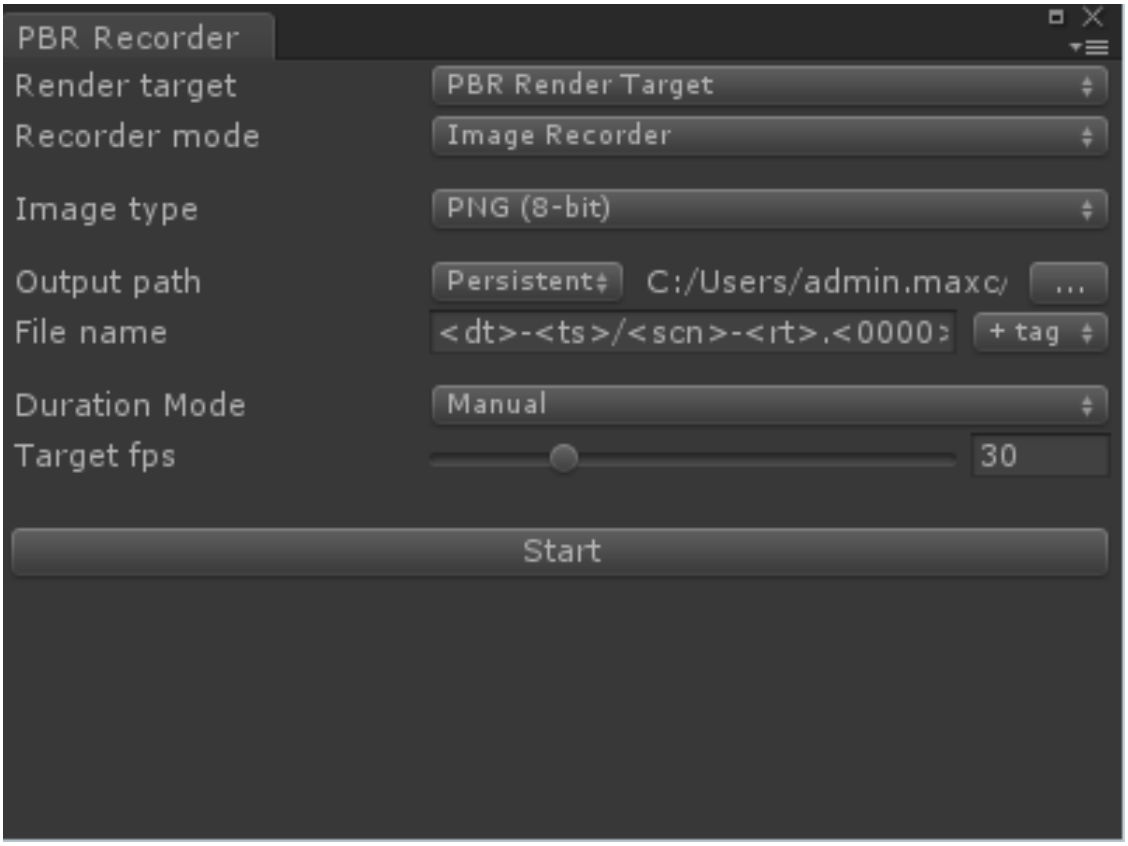


Figure 2: The Recorder window

PBR Render Target

The **PBR**[1] **Render Target** is an essential asset that needs to be added to a Unity® scene in order for OctaneRender® to render the scene. You can access the target from the **Octane** menu (Figure 1), or by right-clicking in the **Hierarchy** window (Figure 2).

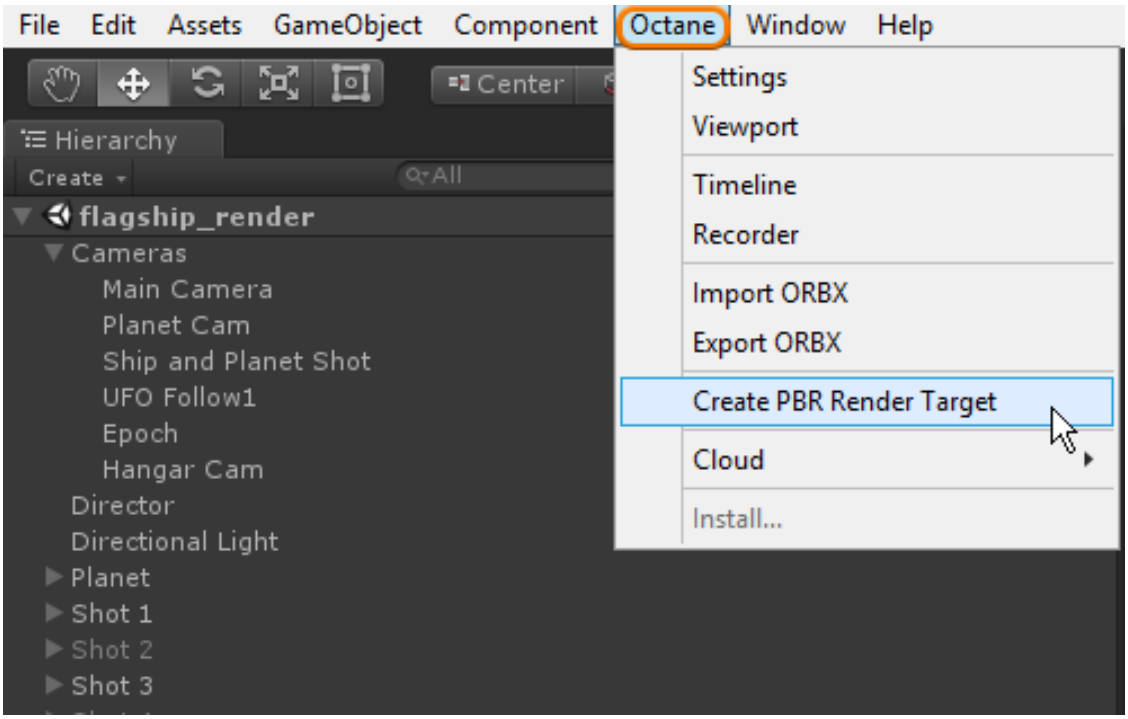


Figure 1: Accessing the PBR Render Target from the Octane menu

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

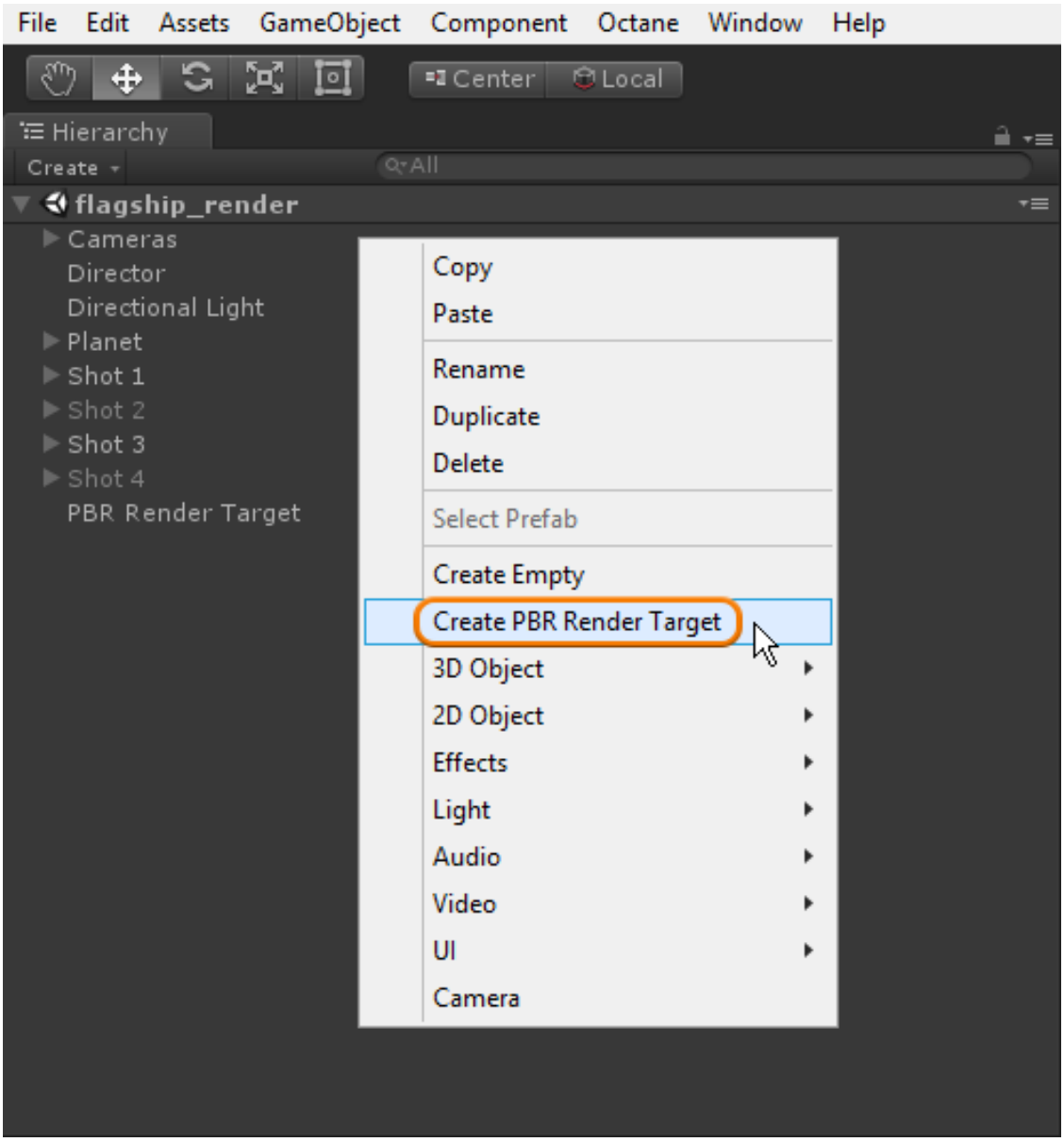


Figure 2: Adding a PBR Render Target by right-clicking in the Hierarchy window

Under the **PBR Render Target Component** rollout, the **Render** button opens the **PBR Viewport** window (if it's not already open) and starts the rendering process (Figure 3). By default, the PBR Viewport syncs to the **Unity Editor** camera view. Navigating in the Unity Viewport alters the view in the PBR Viewport.

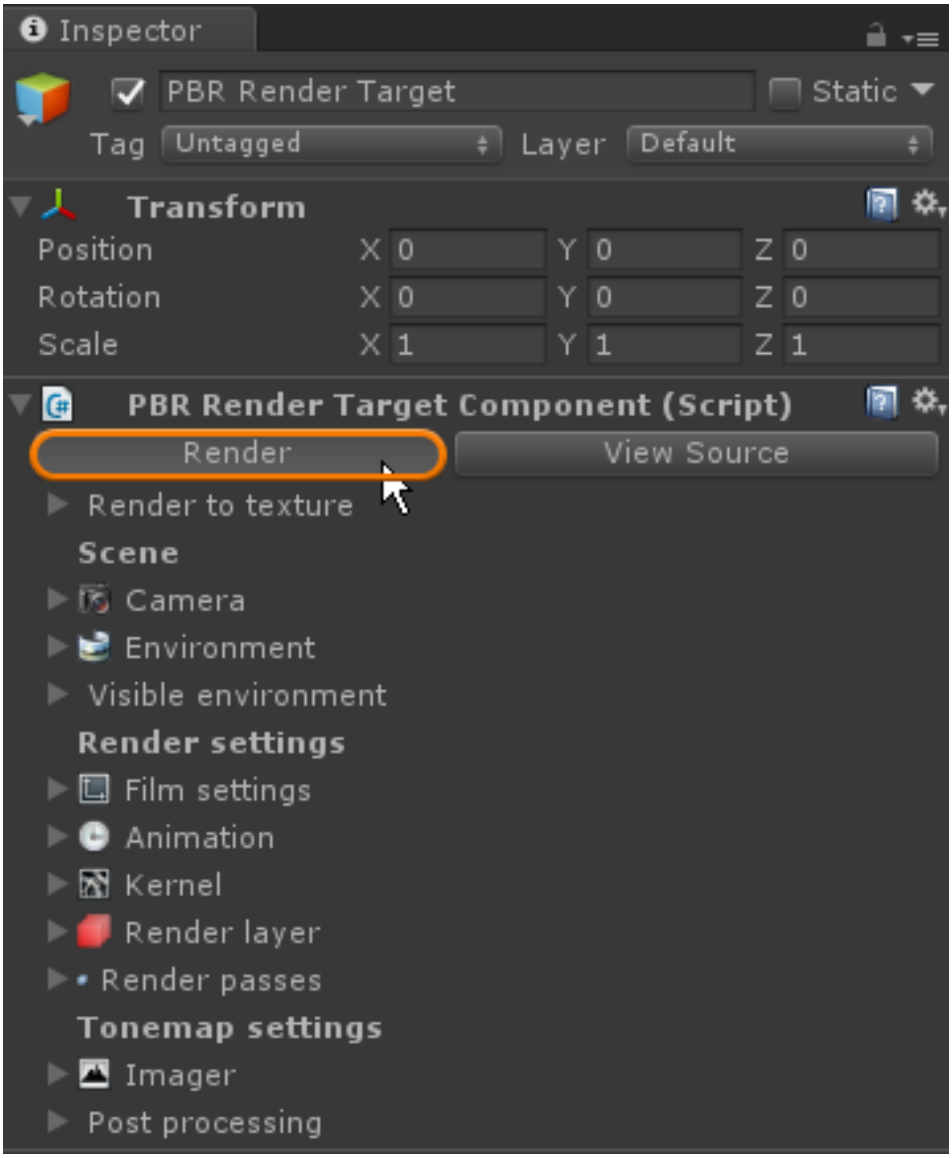


Figure 3: Starting the rendering process via the Render button in the PBR Render Target

The **View Source** button opens the OctaneVR®-specific interface, where you can make additional scene and material edits (see the **OctaneVR Window** section for more details).

The PBR Render Target also contains several categories of parameters, all of which are covered in greater detail later in this manual:

- **Scene** - Contains camera and lighting/environmental controls.
- **Render Settings** - Controls related to render resolution, render kernel types, and render layers/passes.
- **Tonemap Settings** - Contains controls for rendered image processing and post-processing effects.

The OctaneVR® Window

The **OctaneVR**® window is where you edit OctaneRender®-specific materials (Figure 1). You can also edit most OctaneRender-specific assets that you add to a Unity® scene in the OctaneVR window.

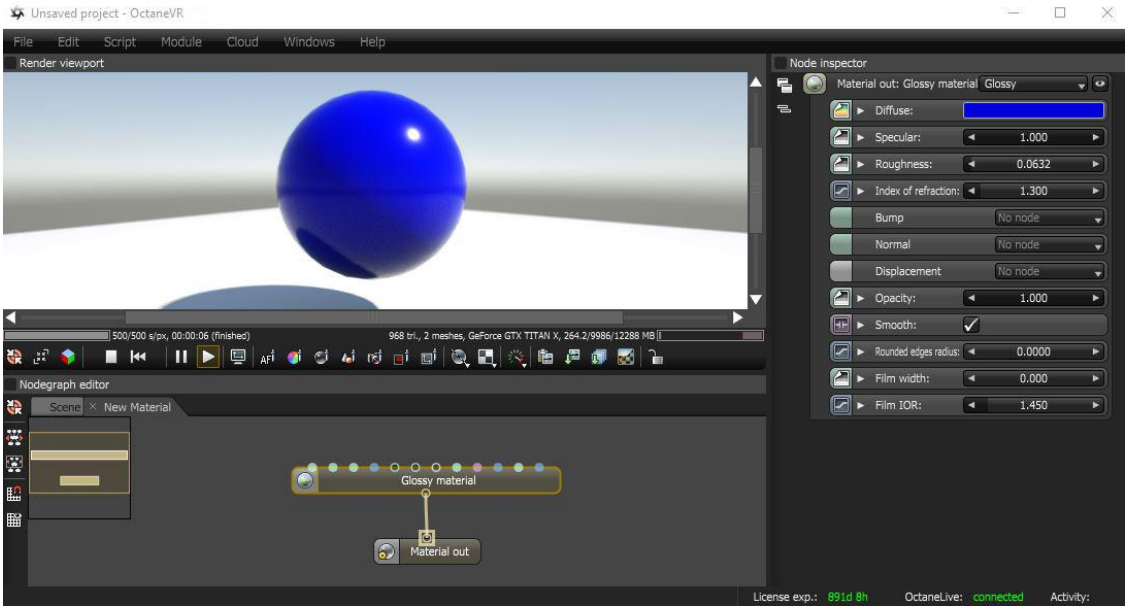


Figure 1: The OctaneVR Interface

You can access the OctaneVR window from the **PBR**[1] **Render Target** by clicking the **View Source** button (Figure 2). You can also access the OctaneVR window by clicking on the View Source button from the **Octane/UserShader** in the Inspector window (Figure 3).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

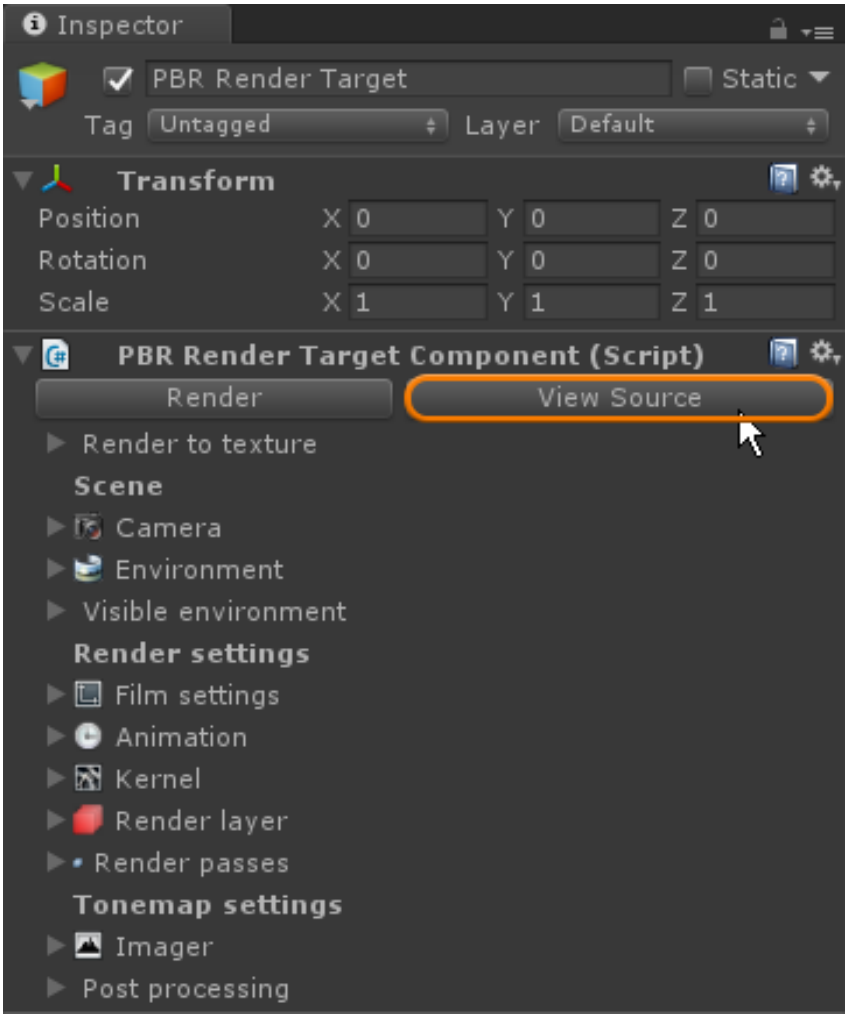


Figure 2: Accessing the OctaneVR window from the PBR Render Target's Inspector window

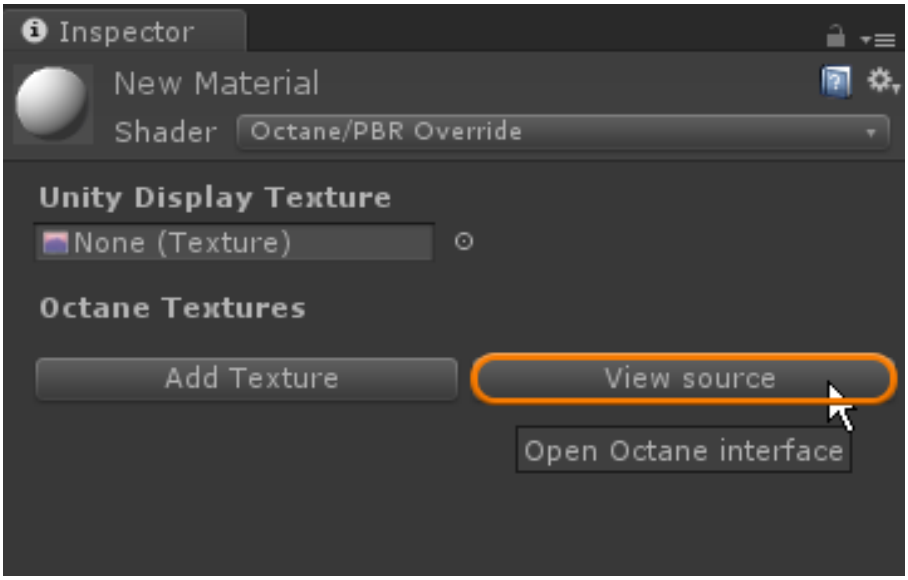


Figure 3: Accessing the OctaneVR window from an Octane Material[1]'s Inspector window

For more information regarding how to work in the OctaneVR window, please refer to the OctaneRender Standalone documentation available on OTOY's website.

Note: Much of the functionality available in the OctaneVR window will be directly integrated into Unity's interface with future releases.

[1] The representation of the surface or volume properties of an object.

What is the ORBX® Format

The **ORBX**[1]® file format is the best way to transfer scene files from 3D authoring software programs that use the OctaneRender® plug-in, such as OctaneRender for Maya® and OctaneRender for Cinema 4D®, as well as OctaneRender Standalone. This format is more efficient than **FBX**[2] when working with OctaneRender-specific data, as it provides a flexible, application-independent format.

ORBX is a container format that includes all animation data, models, textures, etc. that you need to transfer your OctaneRender scene from one application to another.

You can import and export ORBX files from the **Octane** Menu (Figure 1).

[1]The ORBX file format is the best way to transfer scene files from 3D Authoring software programs that use the Octane Plug-in such as Octane for Maya, Octane for Cinema 4D, or OctaneRender Standalone. This format is more efficient than FBX when working with Octane specific data as it provides a flexible, application inde-pendent format. ORBX is a container format that includes all animation data, models, textures etc. that is needed to transfer an Octane scene from one application to another.

[2].fbx (Filmbox) is a proprietary file format developed by Kaydara and owned by Autodesk since 2006. It is used to provide interoperability between digital content creation applications. As of Octane 3.07, a scene node will also be available as an FBX file, allowing for quick and easy transport of assets from industry standard DCC applications

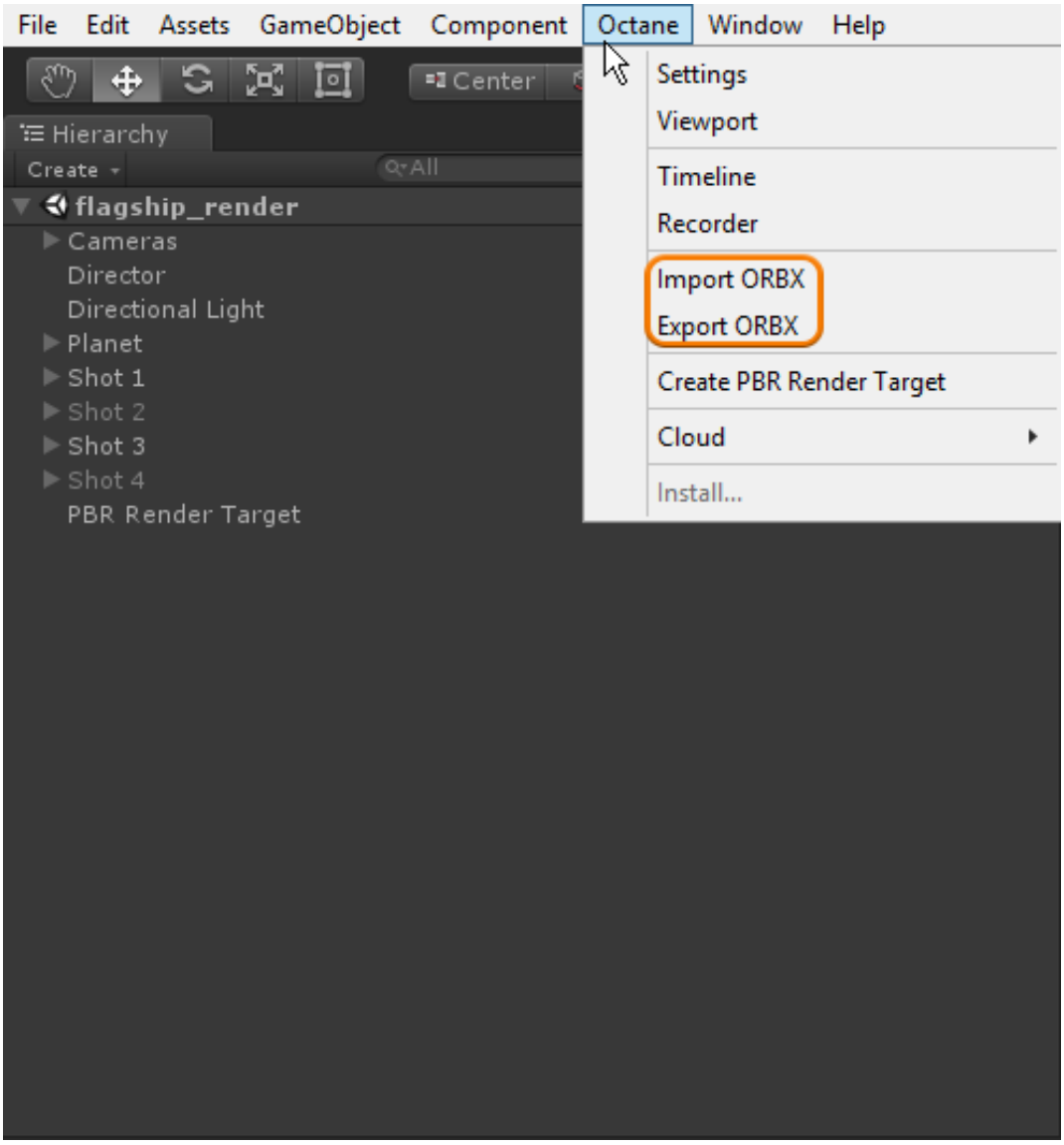


Figure 1: Importing and Exporting ORBX files from the Octane menu

Importing and Exporting ORBX Files

ORBX files are placed as proxies in a Unity® scene. A large cube represents this proxy in the Unity scene. You can only edit ORBX files from within the **OctaneVR**® window. To access an ORBX file from within the **Unity**

Editor, select the ORBX file in the **Hierarchy**, then open the **Inspector** window and click on the **View Source** button (Figure 2). This opens the OctaneVR window. You can edit the scene, shaders, and materials with the OctaneVR **Nodegraph Editor** (Figure 3).

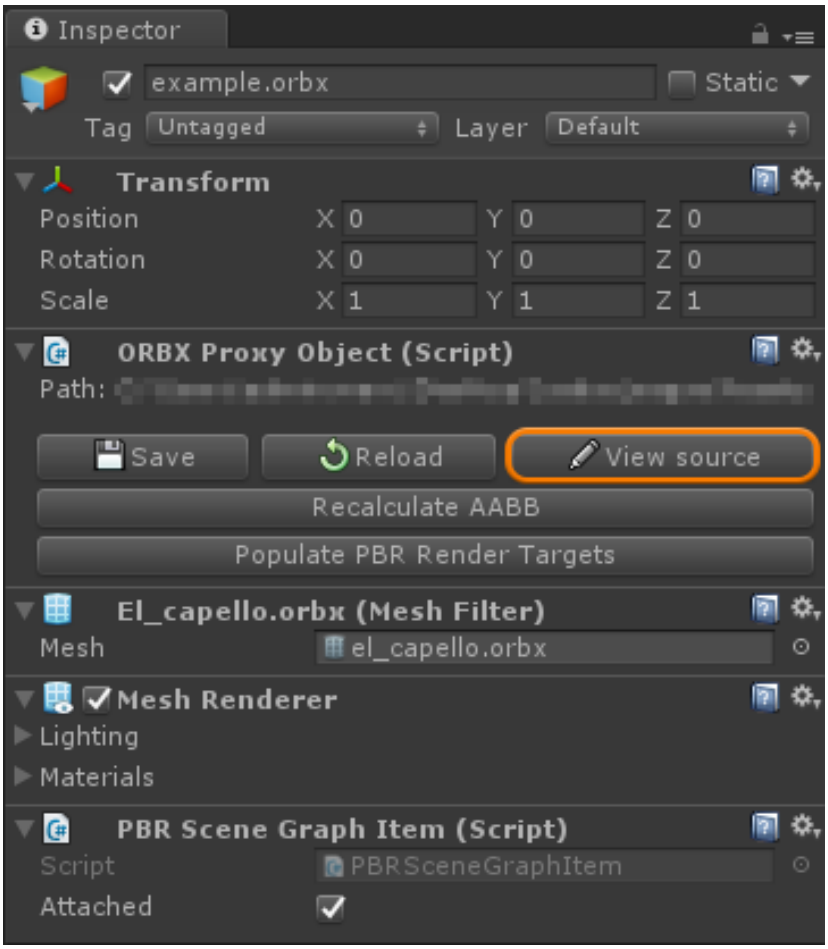


Figure 2: Click on the View Source button in the inspector to edit the ORBX file

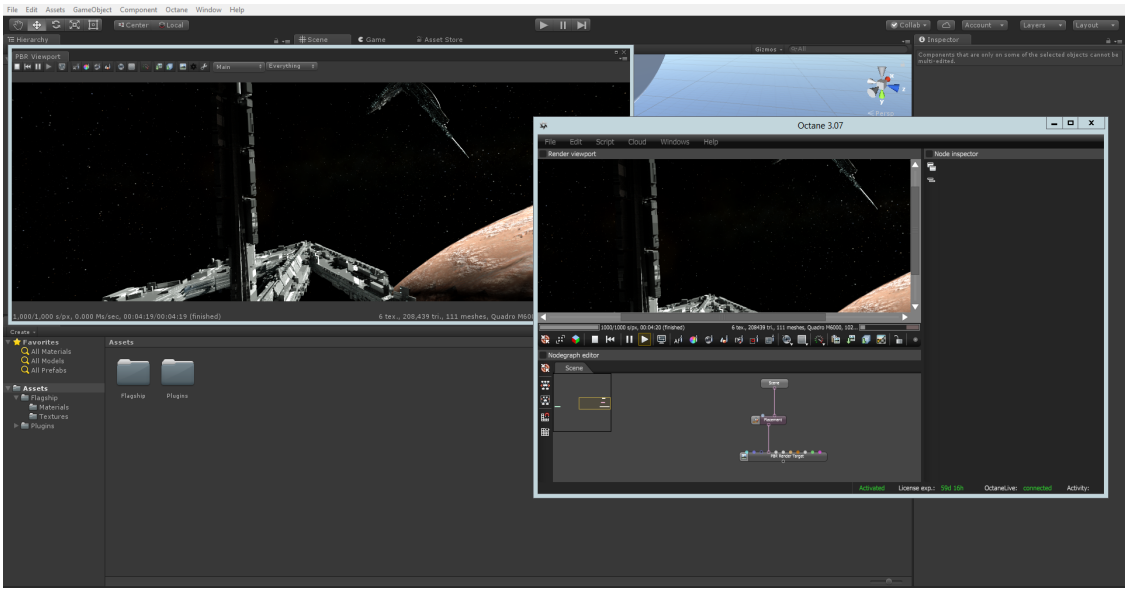


Figure 3: Editing an imported ORBX scene with the OctaneRender Nodegraph Editor

Object Level Script: PBR Instance Properties

OctaneRender® creates a **PBR**[1] **Instance Properties** script on every object in a Unity® scene when the Octane render engine is invoked. You can access its parameters from the **Inspector** window after you select an object in the Unity scene (Figure 1).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

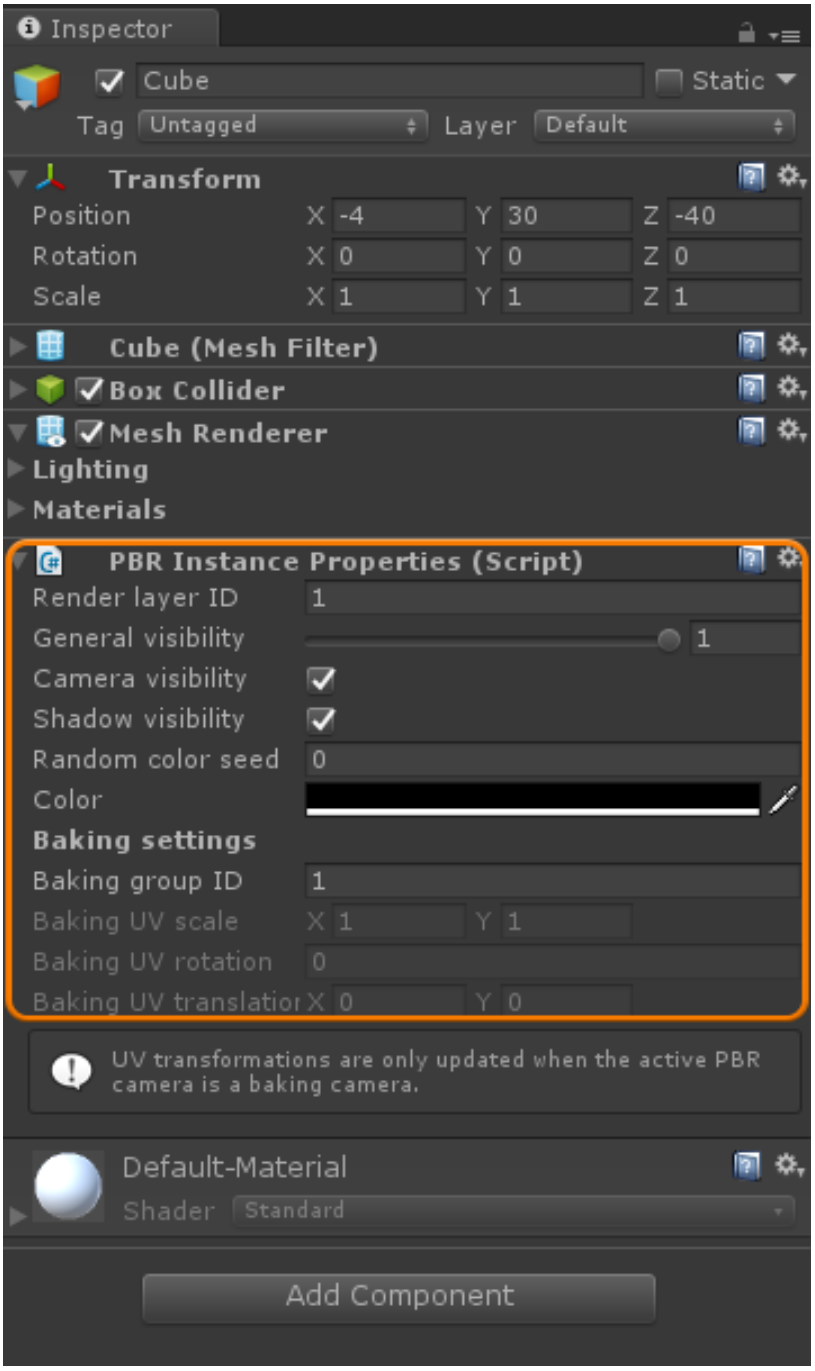


Figure 1: The PBR Instance Properties in the Inspector window

PBR Instance Properties Parameters

- **Render Layer ID** - If using render layers, this parameter specifies the render layer associated with the object.
- **General Visibility** - Controls the overall visibility of the scene object. This is a slider with values ranging from **0** - **1**.
- **Camera Visibility** - Makes the object invisible to the camera, but still casts shadows onto other objects in the scene.
- **Shadow Visibility** - Turns cast shadows on or off for the selected object.
- **Random Color Seed** - Specifies the start point to initialize the color, after which random colors are generated. This is **0** by default, when random colors are currently not in use.
- **Color** - Selects the color for rendering in the object layer render pass.
- **Baking Group ID** - Specifies the Baking Group associated with the object. By default, all objects belong to Baking Group **1**.

Materials

You can use the **OctaneVR**® window and OctaneRender® **PBR**[1] material for deep material design. It is an efficient and fast node-based editing tool, and the interface unlocks the power of OctaneRender's physically-based, unbiased material design environment.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

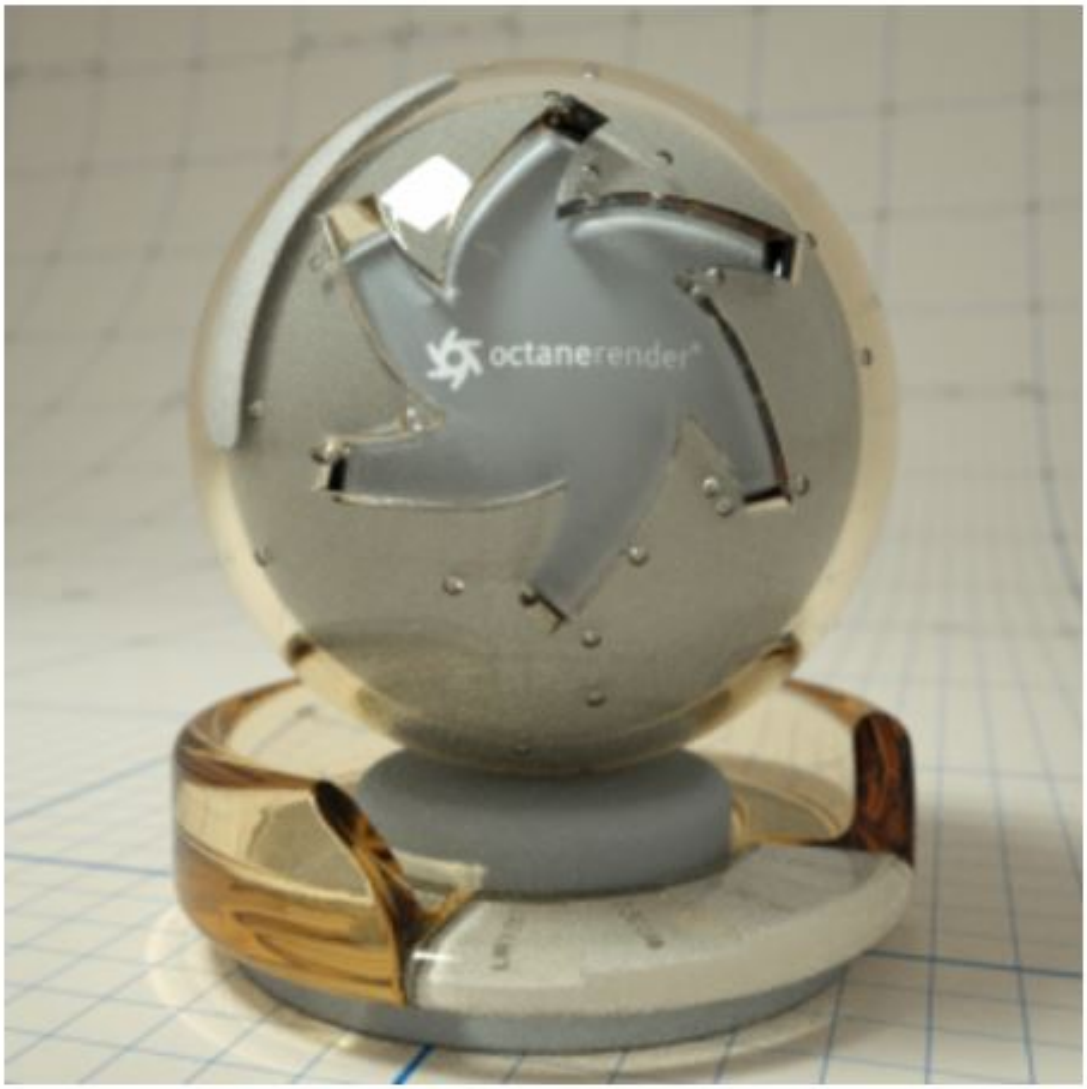


Figure 1: An OctaneRender Specular[1] Material[2]

[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[2]The representation of the surface or volume properties of an object.

Unity® Standard Material

The **Standard** and **Standard (Specular[1] Setup)** materials (Figure 1) are the only two native Unity® materials that work with OctaneRender®. The Standard material differs from the Standard (Specular Setup) material by how it handles shiny highlights.

[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

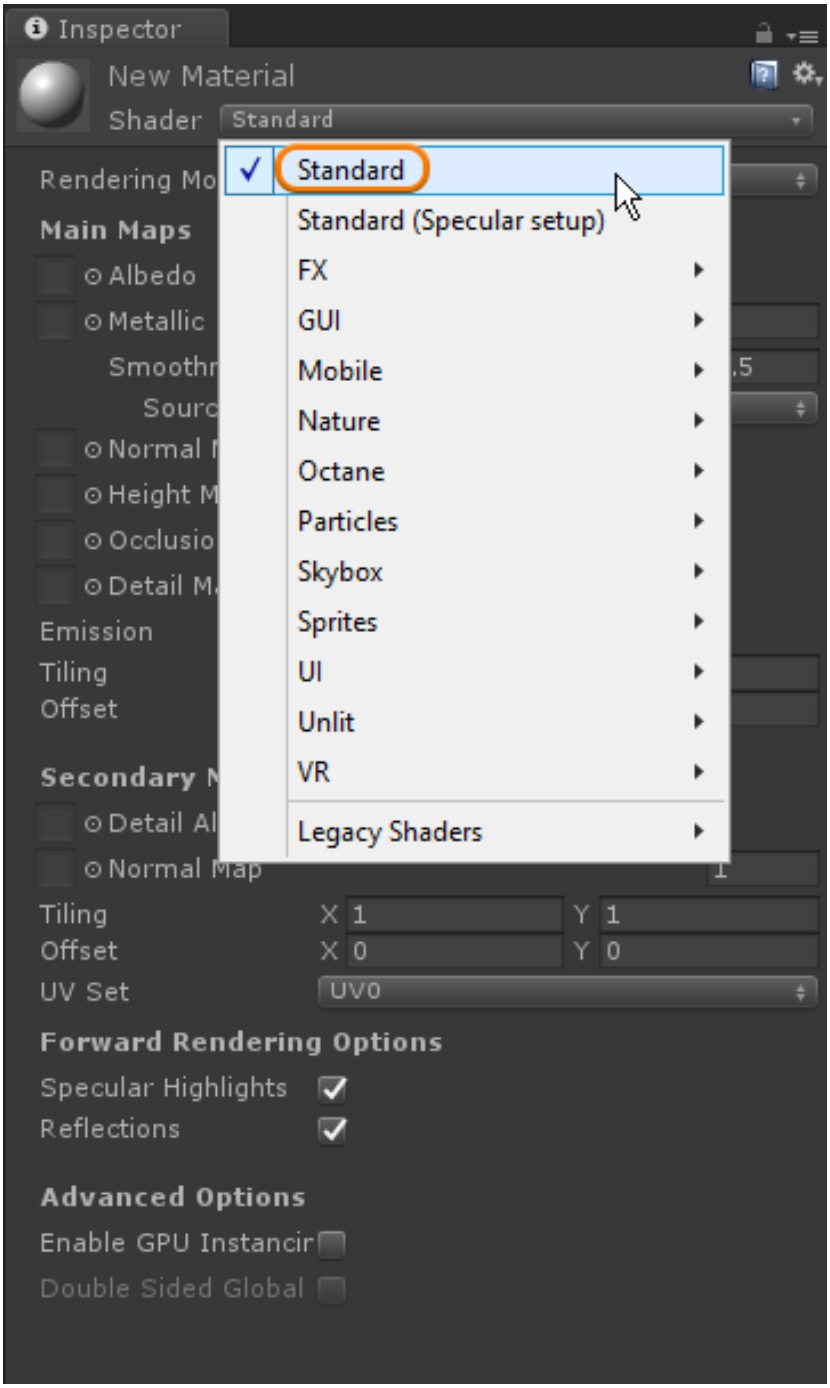


Figure 1: Accessing the Standard material to use with OctaneRender

The Standard material's parameters are compatible with OctaneRender, and function as expected in Unity (Figure 2). You can find further details in the official Unity manual.

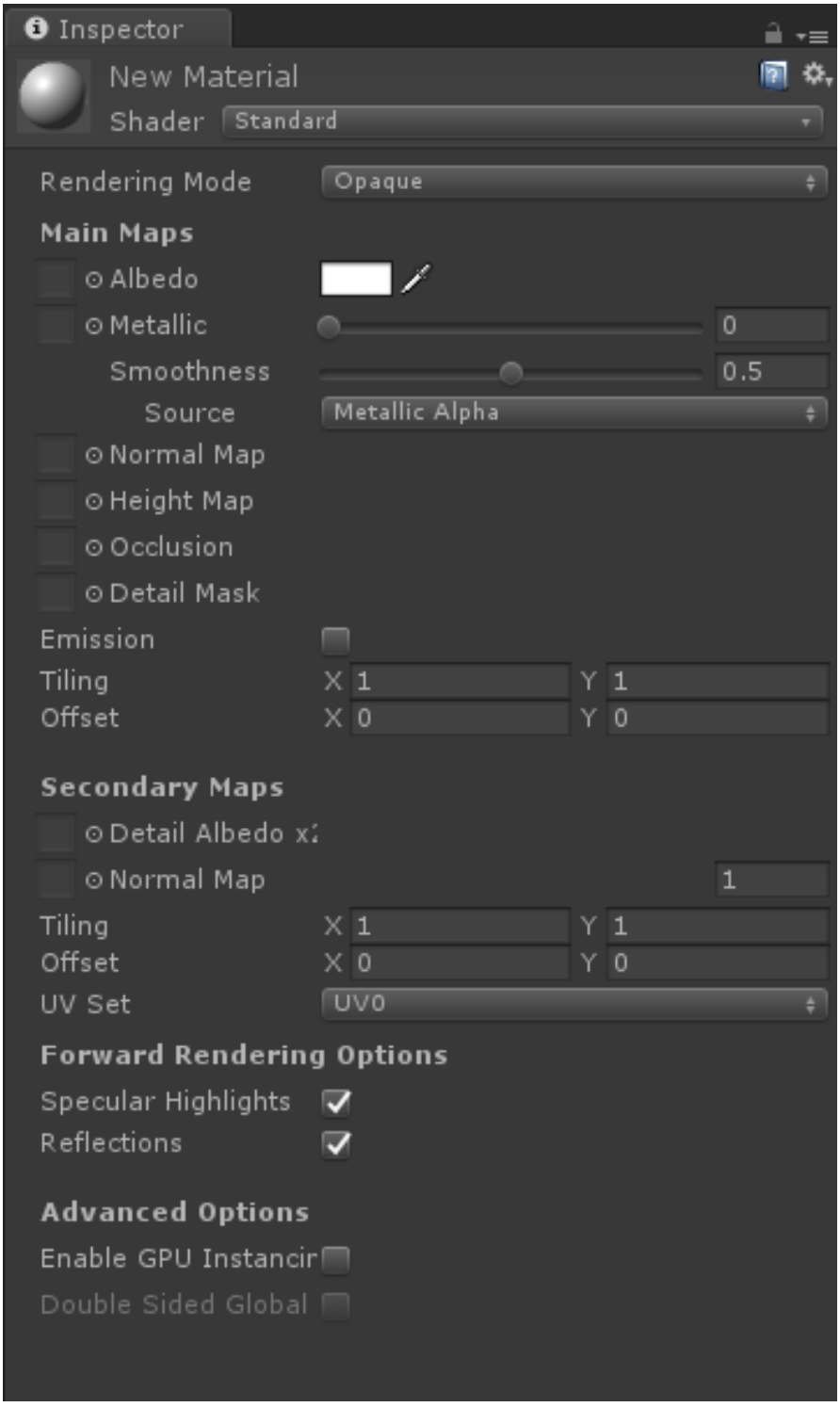

Inspector
New Material
Shader
Standard
Rendering Mode
Opaque
Main Maps
Albedo
Metallic
0
Smoothness
0.5
Source
Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling
X 1
Y 1
Offset
X 0
Y 0
Secondary Maps
Detail Albedo x
Normal Map
1
Tiling
X 1
Y 1
Offset
X 0
Y 0
UV Set
UV0
Forward Rendering Options
Specular Highlights
Reflections
Advanced Options
Enable GPU Instancir
Double Sided Global

Figure 2: The Unity Standard material

Unity® Standard (Specular Setup)

The Unity® **Standard (Specular[1] Setup)** (Figure 1) and **Standard** materials are the only two native Unity materials that work with OctaneRender®. The Standard (Specular Setup) material differs from the Standard material in how it handles shiny highlights.

[1] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

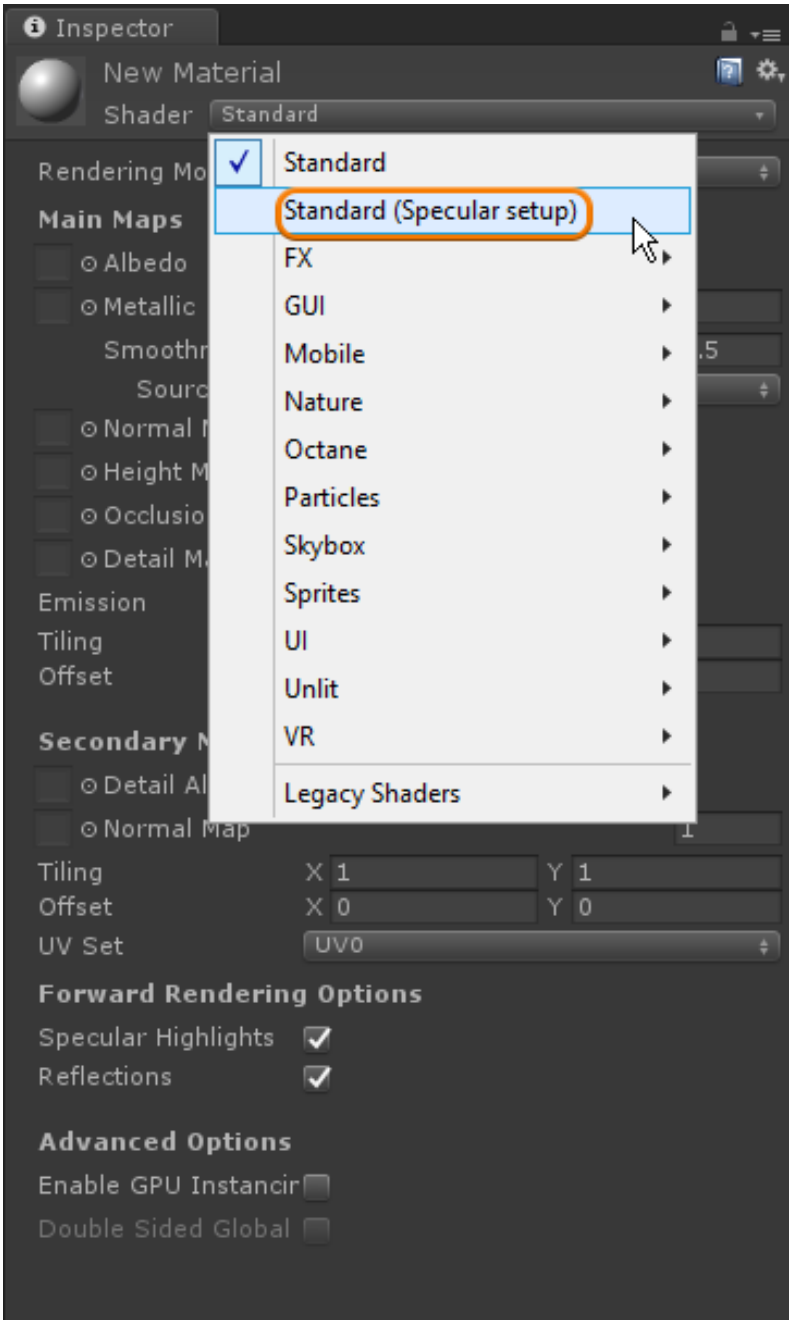


Figure 1: Accessing the Standard (Specular Setup) material for use with OctaneRender

The Standard (Specular Setup) material's parameters are compatible with OctaneRender, and function with OctaneRender as expected in Unity (figure 2). You can find more details in the official Unity manual.

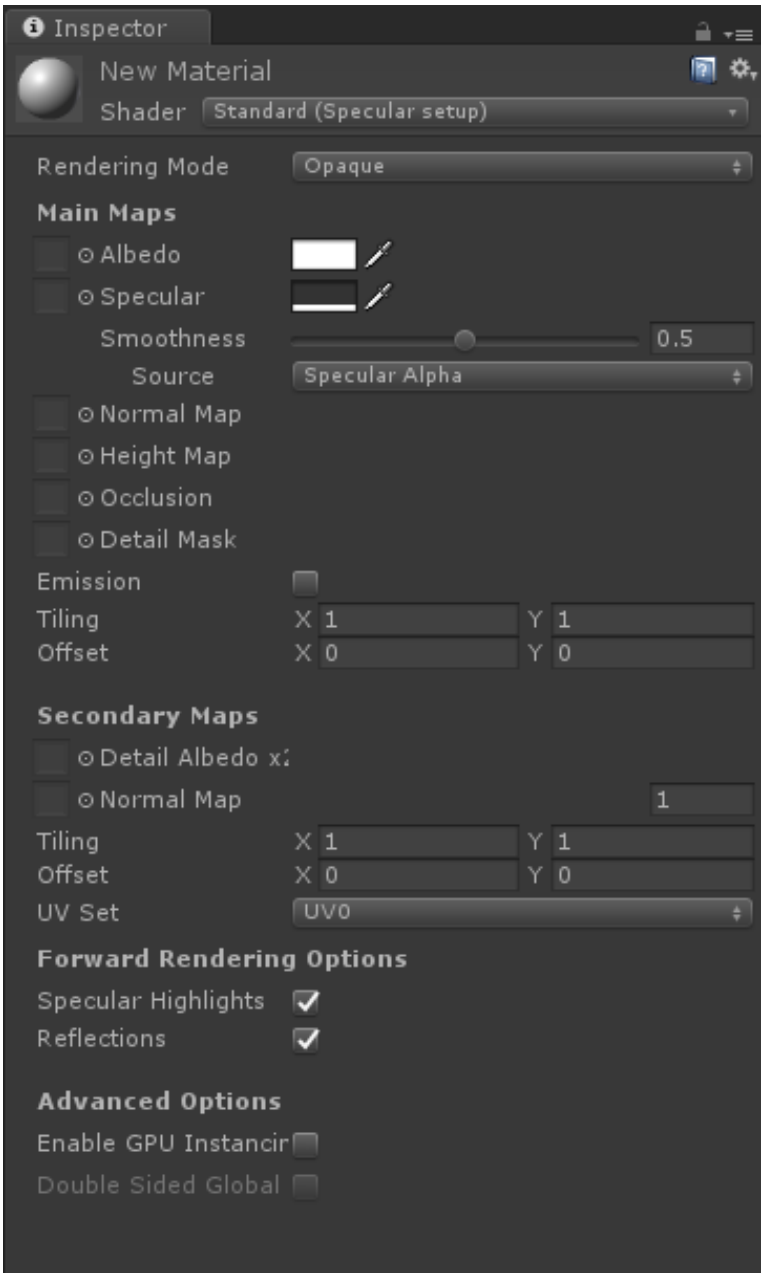


Figure 2: The Unity Standard (Specular Setup) material

OctaneRender®-Specific Materials

The OctaneRender®-specific material types provide greater flexibility than the **Standard** Unity® materials. To add an OctaneRender material to an asset in Unity, you must first right-click anywhere in the **Assets** window, then go to the **Create** category and select **Material**[1] (Figure 1).

[1]The representation of the surface or volume properties of an object.

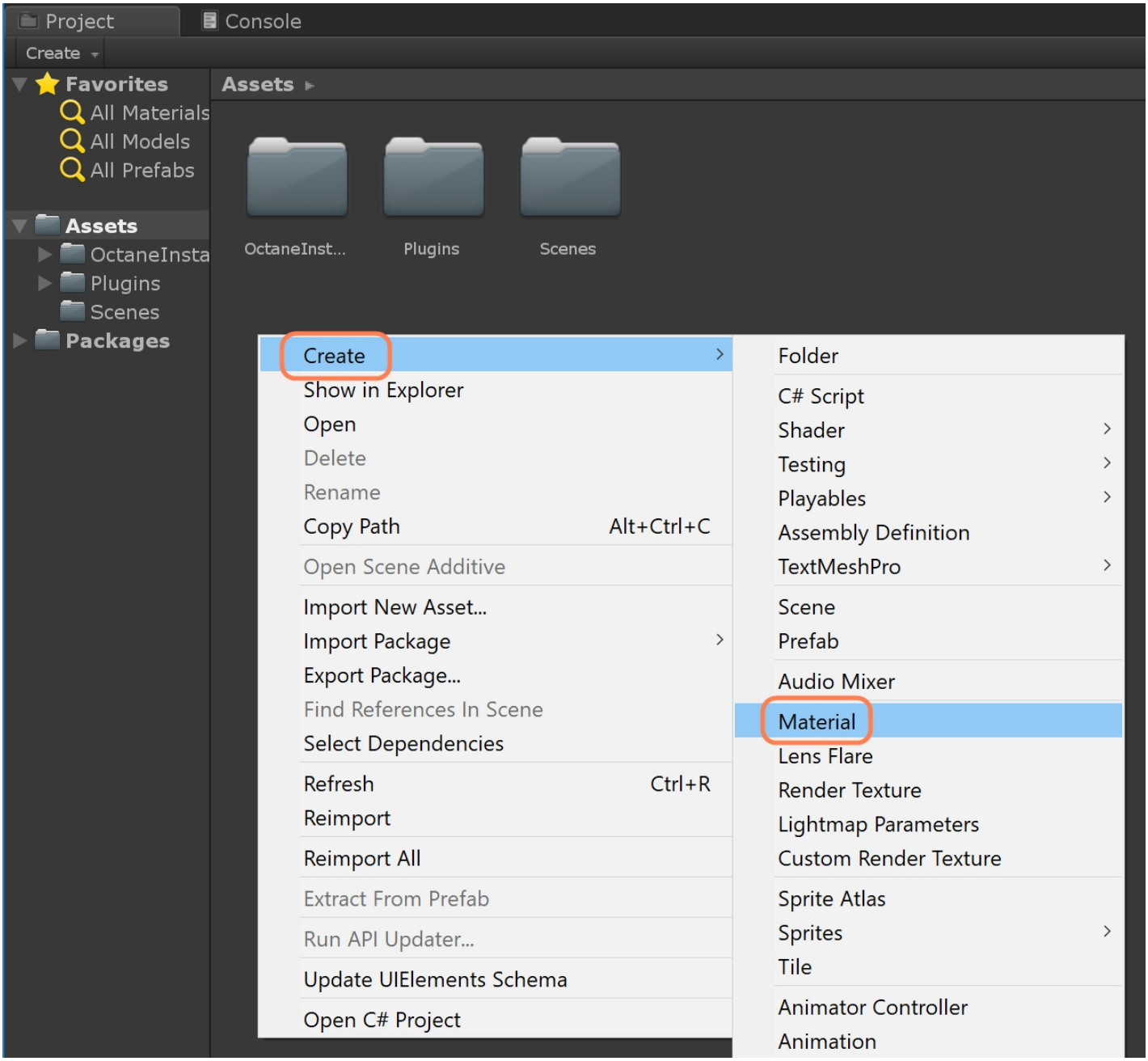


Figure 1: Creating a new Material by right-clicking in the Project Assets window

Next, click on the **New Material** from the Assets window.

From the new material's **Inspector** window (Figure 2), click on the **Shader** dropdown, then choose **Octane**, and then **PBR**[1] **Override**.

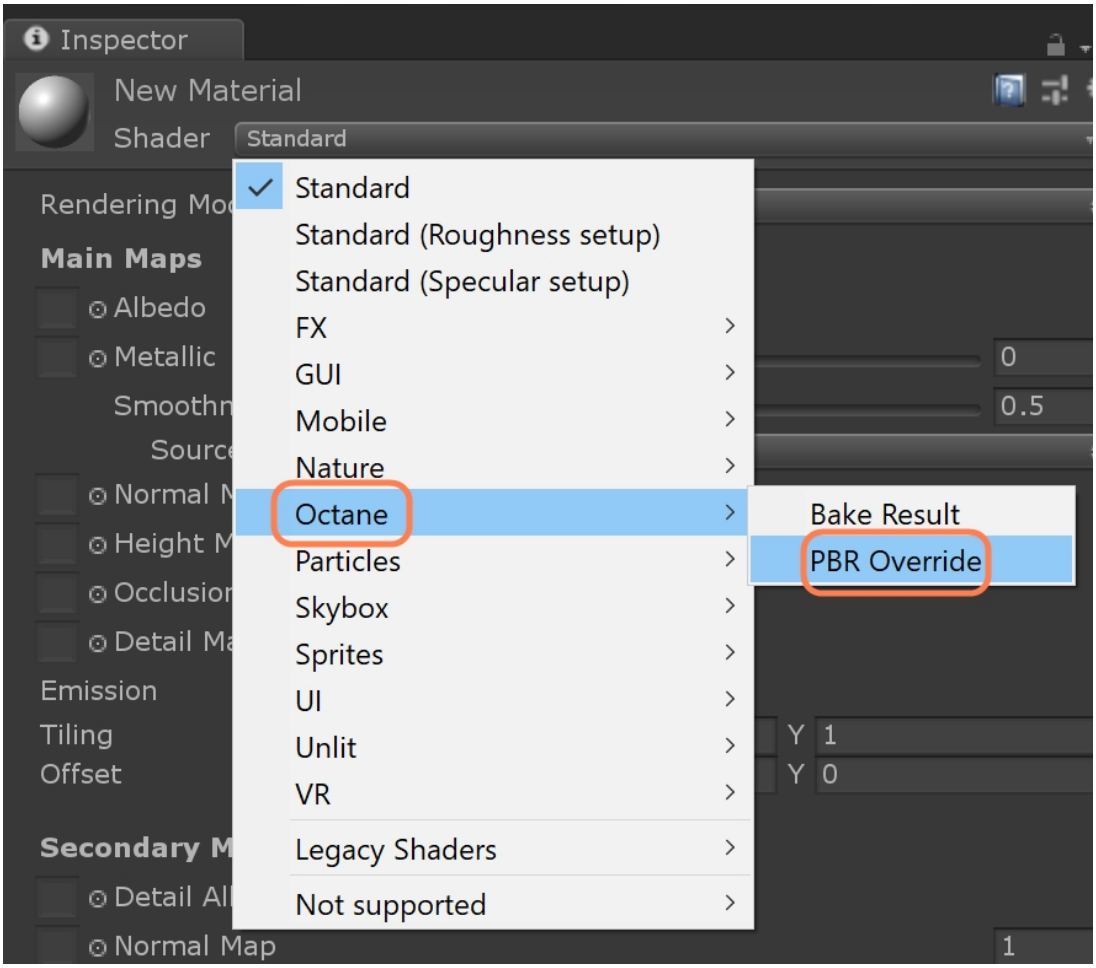


Figure 2: Changing the Shader type to Octane PBR Override

After you make your selection, the parameters in the material's Inspector window changes to reflect the Octane PBR Override parameters.

---

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

You can build and edit the OctaneRender-specific material with the **OctaneVR Nodegraph Editor**. Clicking the **View Source** button in the material's Inspector window opens the **OctaneVR®** window (Figure 3).

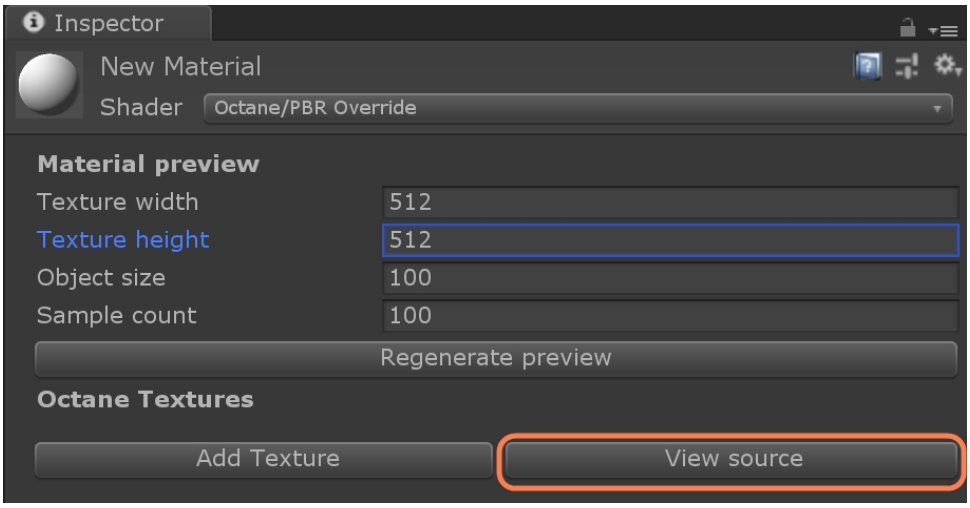


Figure 3: Opening the OctaneVR window by clicking on the View Source button

When you open the OctaneVR window, there is a single node available in OctaneVR's Nodegraph Editor. The **Material Out** node connects the various OctaneRender-specific **Materials**[1] to Unity's Materials (Figure 4).

[1] A set of attributes or parameters that describe surface characteristics.

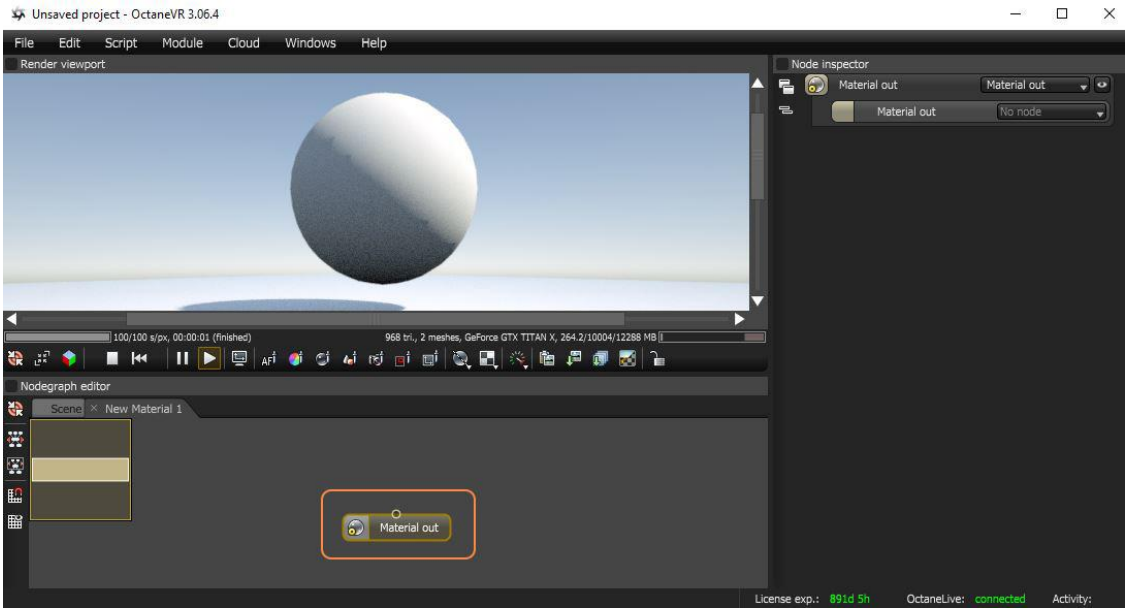


Figure 4: The Material Out node available in the OctaneVR Nodegraph Editor

There are eight material types available in OctaneVR (and Unity):

- **Diffuse**[1] - Creates rough, non-reflecting materials, as well as light emitting meshes.
- **Glossy**[2] - Use for shiny materials such as plastics or metals.
- **Metallic** - These are highly reflective materials with colored reflections (highlights). Similar to glossy material, but more suited to model metals.
- **Mix** - Mixes any two material types.
- **Portal**[3] - Designate openings in scenes to allow the render kernel to better sample light from those areas.
- **Specular**[4] - Used for transparent materials such as glass and water.

---

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
[3]A technique that assists the render kernel with exterior light sources that illuminate interiors. In interior renderings with windows, it is difficult for the path tracer to find light from the outside environment and optimally render the scene. Portals are planes that are added to the scene with the Portal material applied to them.
[4]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

- **Toon** - This material has shadows and highlights appear as blocks of color, creating a flatter-looking image that emulates the style of a cartoon in two-dimensional illustrations.
- **Universal** - Brings substance maps and PBR outputs into OctaneRender.

You can access these materials by right-clicking in the OctaneVR Nodegraph Editor and navigating to the **Materials** category (Figure 5).

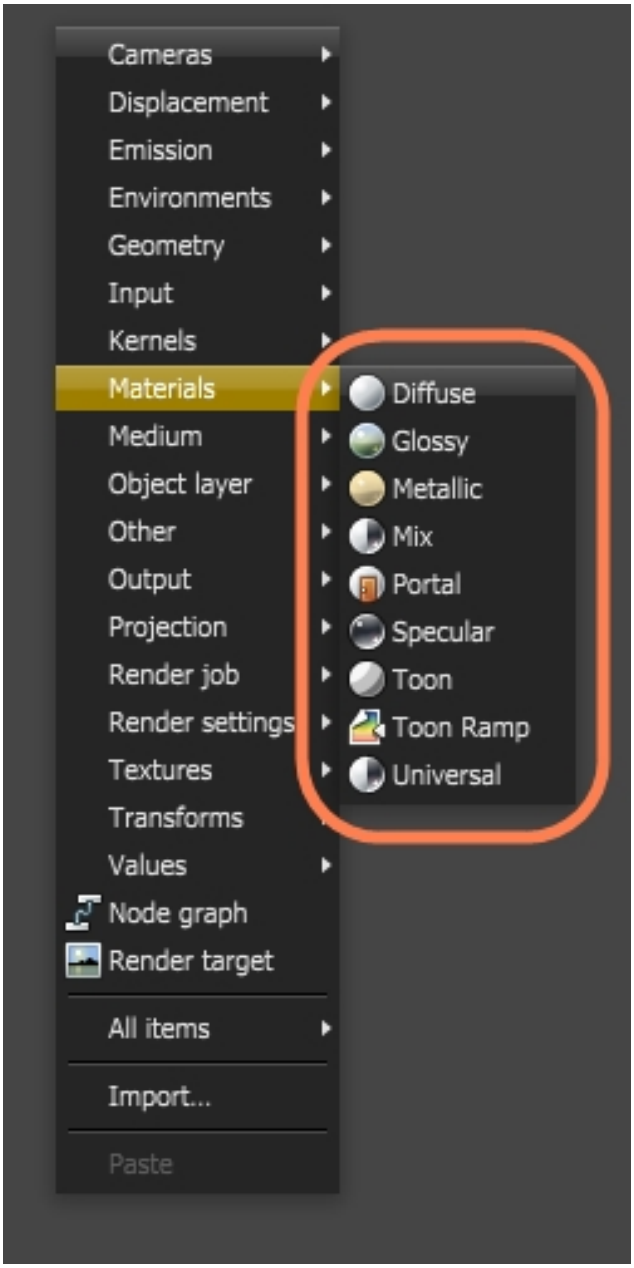


Figure 5: Right-clicking in the OctaneVR Nodegraph Editor to access the OctaneRender material types

Once you select a material type, you need to connect its output pin to the **Input** pin on **Material Out**. Unity then updates the material (Figure 6). For more information on how to work with materials in the OctaneVR window, please refer to the OctaneRender Standalone documentation available on the OTOY® website.

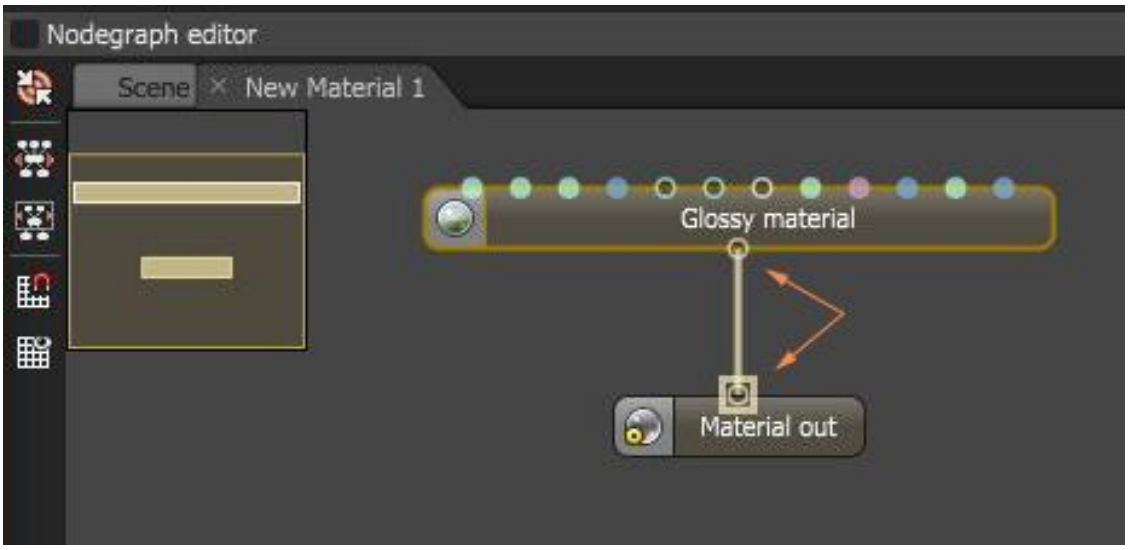


Figure 6: Connecting input and output pins on material nodes

Note: Much of the material functionality available in the OctaneVR window will be directly integrated into Unity's interface with future releases.

Diffuse Material

The **Diffuse**[1] material is used for dull, non-reflecting materials or light emitting surfaces. **Diffuse material**[2] simulates a rough surface that reflects light back into the environment in all directions. **Specular**[3] highlights and reflections do not appear on diffuse surfaces.

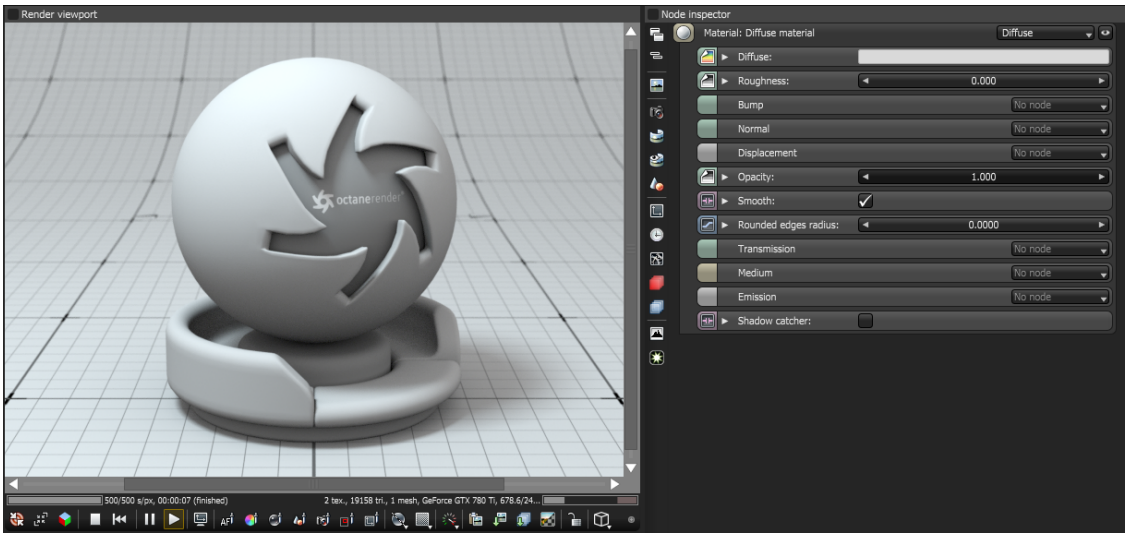


Figure 1: The OctaneRender® Diffuse material

Diffuse Material Parameters

- **Diffuse**- Provides color to the material. This is also known as base color or albedo. You can set Diffuse color by using a value, or connecting a texture (**Procedural** or **Image**).
- **Roughness** - Determines the spread of highlights on the surface. A high Roughness value or light color can simulate very rough surfaces such as sand paper or clay. You can set Roughness using a value, or

---

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]Used for dull, non-reflecting materials or mesh emitters.
[3]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

by connecting a texture (procedural or image-based). A roughness value of **1** (white color) creates a diffuse sheen along the edges of the surface, simulating the look of crushed velvet.

- **Bump**- Creates fine details on the material's surface using a Procedural or Image texture. Typically a **Greyscale** image texture connects to this parameter - light areas of the texture give the appearance of protruding bumps, and dark areas create the appearance of indentation. You can adjust the strength of the bump map by setting the **Power** or **Gamma**[1] values on the Image texture node. These attributes are covered in more detail under the **Texture** category.
- **Normal**- Creates the look of fine detail on the surface. A Normal map is a special type of image texture that uses red, green, and blue color values to perturb the normals of the surface at render time, thus giving the appearance of added detail. They can be more accurate than Bump maps, but require specific software, such as ZBrush®, Mudbox®, Substance Designer, xNormal™, or others to generate. To load a full color Normal map, set the Normal channel to the **RGB Image** data type. Note that Normal maps take precedence over Bump maps, so you cannot use a Normal map and a Bump map at the same time.
- **Displacement**[2]- Adjusts the height of the vertices of a surface at render time using a texture map. Displacement maps differ from Bump or Normal maps in that the geometry is altered by the texture, as opposed to just creating the appearance of detail. Displacement mapping is more computationally expensive than using a Bump or Normal map, but the results are more realistic, especially along the silhouette of the surface. Displacement only works with the **Texture Image** node, and the displaced mesh must have **UV Texture** coordinates. Other OctaneRender Texture nodes, such as **Turbulence** or **Marble**, will not work with Displacement. Displacement mapping is covered in more detail under the **Texture Overview** category.
- **Opacity**- Determines which parts of the surface are visible in the render. Dark values indicate transparent areas, and light values indicate opaque areas. Values in between light and dark create the look of semi-transparent areas. You can lower the opacity value to fade the overall visibility of an object, or you can use a Texture map to vary the opacity across the surface. For example, if you want to make a simple polygon plane look like a leaf, you would connect a black-and-white image of the leaf's silhouette to the Opacity channel of the **Diffuse** shader. When using an Image texture map, set the **Data Type** to **Alpha Image** if the image has an alpha channel, or **Grayscale Image** for black-and-white images, to load an image for setting the transparency. To invert the transparency regions, use the image's **Invert** checkbox.
- **Smooth**- Smooths out the transition between surface normals. If this option is disabled, the edges between the polygons of the surface appear sharp, giving the surface a faceted look. If this option is enabled, then the edges between the polygons blend together.

---

[1] The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

[2] The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

- **Rounded Edges Radius**- This parameter automatically bevels the edges of the surface at render time, without the need to alter or subdivide the geometry. Using this option enhances the realism of objects by eliminating overly sharp edges. The value refers to the radius of the rounded edge - higher values for this setting produce rounder edges.
- **Transmission**[1]- Uses a color or texture that is mixed with the material's Diffuse color, and is most noticeable in areas affected by indirect lighting.
- **Medium**- OctaneRender has three types of mediums to create translucent surfaces:
  - **Absorption**[2] **Medium**- Produces the appearance of a material that absorbs light while passing through a surface. The resulting color depends on the distance that light travels through the material. The Texture Overview category provides more detail about Absorption.
  - **Scattering**[3] **Medium**- Similar to the Absorption medium, but with an additional option for simulating subsurface scattering. Subsurface scattering is the phenomena that gives human skin and similar organic surfaces their characteristic glow under certain lighting conditions. It's a major component for creating the look of realistic skin. The Texture Overview category provides more detail about scattering.
  - **Volume Medium**[4] - Adds color and other qualities to a **VDB**[5] file. VDBs are a generic volume format for creating effects such as smoke, fog, vapor, and similar gaseous objects. VDBs can consist of a single frame, or an animated sequence. 3D software packages like Houdini generate and export VDBs. You can also download VDB files at http://www.openvdb.org/download/.
- **Emission**- Also known as a **Mesh** emitter, this creates a surface that emits light. To activate an emission, connect the Emission input of the Diffuse material to either a **Blackbody** or a Texture emission map. The "Textures" on page 99 Overview and the Mesh Emitters (under Lighting Overview) categories cover maps in more detail.
- **Shadow Catcher**[6]- Converts the material into a Shadow Catcher, which is only visible in areas that are in shadows. All other areas are transparent in the render.

---

[1] A surface characteristic that determines if light may pass through a surface volume.

[2] Defines how fast light is absorbed while passing through a medium.

[3] Defines how fast light gets scattered when traveling through the medium.

[4] A shading system designed to render volumes such as smoke and fog.

[5] Dreamworks' open-source C++ library housing the data structures and tools implementation for storing and manipulating volume data, like smoke and other amorphous materials. The purpose of OpenVDB is mostly to have an efficient way to store volumetric data in memory and on disk. It has evolved into a more general toolkit that also lets you accomplish other things, such as fracturing volumes, converting meshes to volumes and vice versa. However, it does not include a computational fluid dynamics solver, and therefore it cannot procedurally generate smoke or fire. OpenVDB is fully integrated as a library in OctaneRender. For more information about OpenVDB, please see http://www.openvdb.org/.

[6] The Shadow Catcher can be used to create shadows cast by objects onto the surrounding background imagery. The shadows cast are not limited to simply a ground plane but can be cast onto other surfaces of varying shapes.

Glossy Material

The **Glossy**[1] material is used for shiny materials such as plastics or metals.

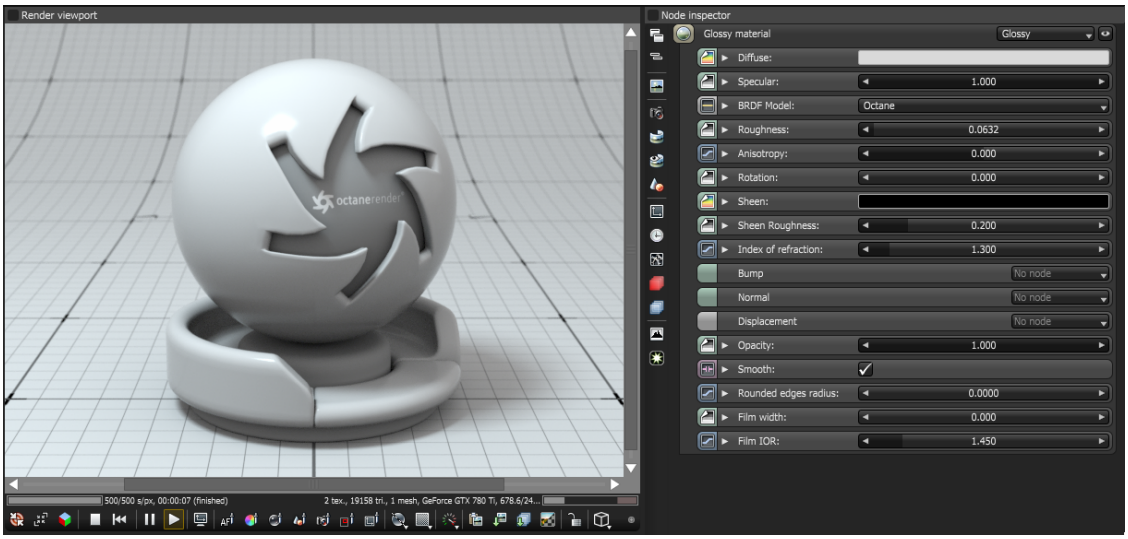


Figure 1: The OctaneRender® Glossy material[2]

Glossy Material Parameters

- **Diffuse**[3]- Gives the material its color, which is referred to as base color or albedo. You can set Diffuse color by using a value, or by connecting a texture (procedural or image-based).
- **Specularity**- Determines the intensity for specular reflections on the surface. This parameter accepts color, values, or textures. In most cases, specular highlights are white or colorless. However, to simulate metallic surfaces, you should tint the specular color using a color similar to the Diffuse parameter,

---

[1]The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
[2]Used for shiny materials such as plastics or metals.
[3]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

like the bright yellow-orange highlights seen on a polished copper kettle. A Glossy node with a black Diffuse color, a **Roughness** of **0**, and an **Index** of **0** produces a perfect mirror (Figure 2).

- **BRDF Model**- The BRDF (Bidirectional Reflectance Distribution Function) determines the amount of light that a material reflects when light falls on it. For Glossy materials, you can choose from four BRDF models. Specific geometric properties (the micro-facet distribution) of the surface affects each BRDF, which describes the surface's microscopic shape (i.e. micro-facet normals) and scales the brightness of the BRDF's reflections. Refer to the topic on BRDF Models for more information.

Figure 24: The four BRDF Models applicable to Glossy materials

- **Roughness** - Determines how much **Specular**[1] reflection spreads across the surface - also known as reflection blur. This parameter accepts a value, color, or texture map (**Procedural** or **Image**). A value of **0** simulates a perfect, smooth reflective surface like a mirror. Increasing the value simulates micro-facets in the surface, which causes reflective highlights to spread. To create a worn plastic look, increase the Roughness value.
- **Anisotropy** - Controls the material's reflectance uniformity. Reflectance changes based on surface orientation or rotation is anisotropic. If the reflectance is uniform in all directions and doesn't change based on the surface's orientation or rotation, then it is isotropic. This parameter's default value is **0**, which sets the metallic material as isotropic. Non-zero values mean the material exhibits anisotropic reflectance, where **-1** is horizontal and **1** is vertical.

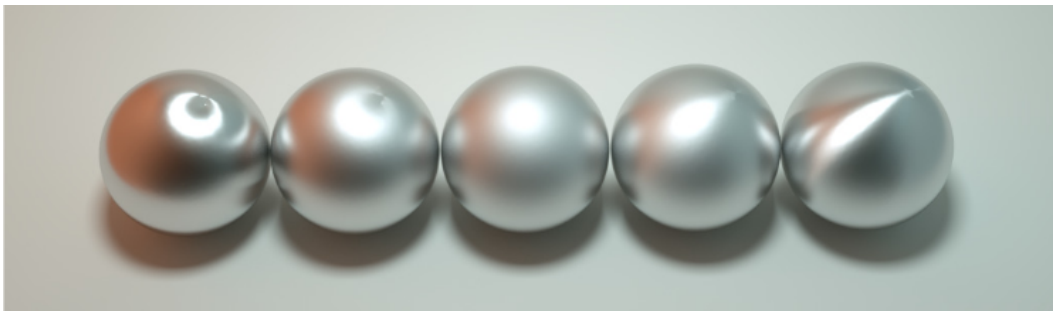
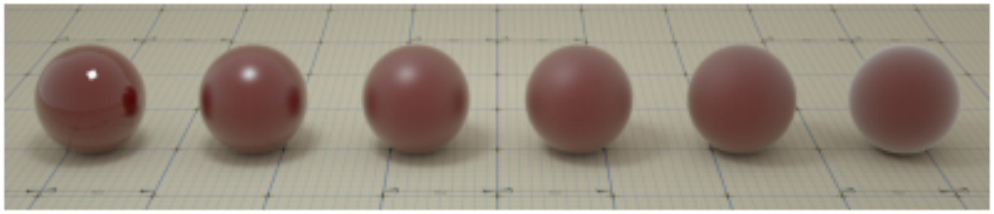


Figure 35: Anisotropic roughness exemplified in materials like brushed metal

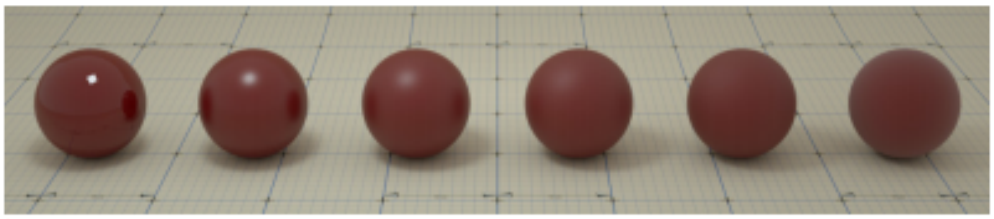
- **Rotation**- The rotation of the anisotropic Specular reflection channel.
- **Sheen**- The material's sheen color.
- **Sheen Roughness**- Roughness channel for the sheen present on **Metallic** and Glossy materials.
- **Index of Refraction**- Determines the reflection strength on the surface based on Fresnel's law. With a value greater than **1**, reflection is strongest on the surface parts that turn away from the viewer's angle (grazing angles), while the reflection appears weaker on the surface parts perpendicular to the viewing angle. This results in a more realistic-looking surface. With a value lower than **1**, the Fresnel effect is disabled, and the reflection color appears as a uniform color across the highlight. The Specular channel's color determines the reflective highlight's color.

  In the following examples, the six balls have a roughness of **0**, **0.2**, **0.4**, **0.6**, **0.8**, **1.0** (left to right) and only the Specular value and Index of Refraction parameters are modified for each rendered image (see Figure 3).

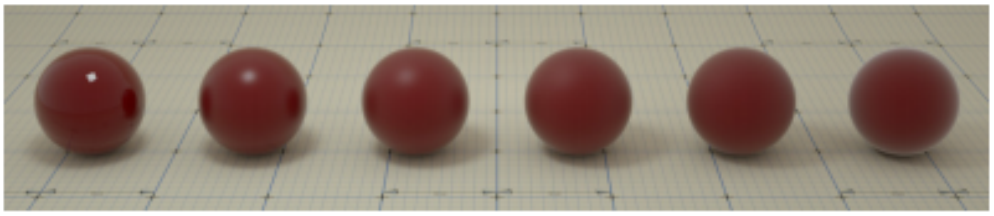
[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.



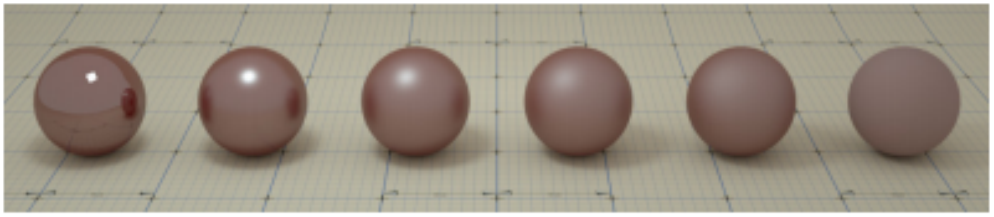
specular = 1.0
index = 2.0 (strong Fresnel reflection)



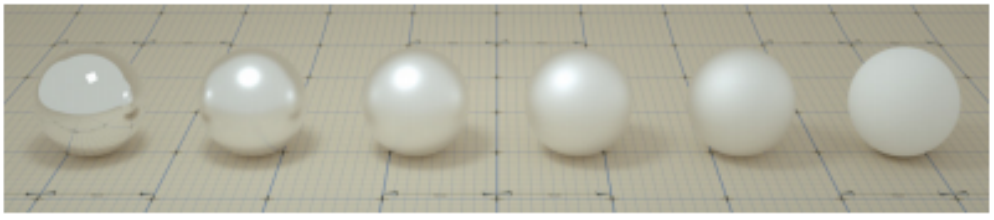
specular = 0.3
index = 3.0 (very strong Fresnel reflection)



specular = 1.0
index = 1.5 (normal Fresnel reflection)



specular = 0.3
index = 0.0 (no Fresnel reflection)



specular = 1.0
index = 0.0 (no Fresnel reflection)

Figure 43: Spheres rendered using different settings for specular and index

- **Bump**- Creates fine details on material surfaces using a **Procedural** or **Image** texture. When you connect a **Grayscale** image texture to this parameter, light areas appear as protruding bumps, and dark areas appear as indentations. You can adjust the Bump map strength by setting the **Power** or **Gamma**[1] values on the Grayscale image texture node. The Texture Overview category covers these attributes in more detail.
- **Normal**- Creates the look of fine detail on the surface. A Normal map is a special type of image texture that uses red, green, and blue color values to perturb the normals of the surface at render time, thus giving the appearance of added detail. They can be more accurate than Bump maps, but require specific software, such as ZBrush®, Mudbox®, Substance Designer, xNormal™, or others to generate. to load a full color Normal map, set the Normal channel to the **RGB Image** data type. Note that Normal maps take precedence over Bump maps, so you cannot use a Normal map and a Bump map at the same time.
- **Displacement**[2]- Adjusts surface vertices' height at render time using a Texture map. Displacement maps differ from Bump or Normal maps by having the texture alter the geometry, as opposed to creating the appearance of detail on the surface. Displacement mapping is more computationally expensive than Bump or Normal mapping, but results are more realistic, especially along the surface silhouette. The Texture Overview category covers Displacement mapping in more detail.
- **Opacity**- Determines what surface parts are visible in the render. Dark values indicate transparent areas, and light values indicate opaque areas. Values between light and dark create the look of semi-transparent areas. You can lower the Opacity value to fade the object's overall visibility, or you can use a Texture map to vary the surface's opacity. For example, if you want to make a simple polygon plane look like a leaf, you connect a black-and-white image of the leaf's silhouette to the Diffuse shader's **Opacity** channel. When you use an Image texture map, set the **Data** type to Alpha image if the image has an alpha channel, or Grayscale image for black-and-white images, to load an image for setting transparency. Use the image's **Invert** checkbox to invert the transparency regions.
- **Smooth**- Smooths out the transition between surface normals. When disabled, the edges between surface polygons appear sharp, giving the surface a faceted look. when enabled, the edges between polygons blend together.
- **Rounded Edges Radius**- Bevels the surface edges automatically at render time, without the need to alter or subdivide the geometry. This option enhances object realism by eliminating sharp edges. Higher values produce rounder edges.
- **Film Width**- Simulates the look of thin film material on a surface, like creating a rainbow color effect

---

[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

[2]The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

that appears on an oil slick's surface. Larger values increase the effect's strength.

- **Film IOR**- Controls the film Index of Refraction. This option adjusts the visible colors in the film.

Specular Material

The **Specular**[1] material creates transparent materials like glass and water.

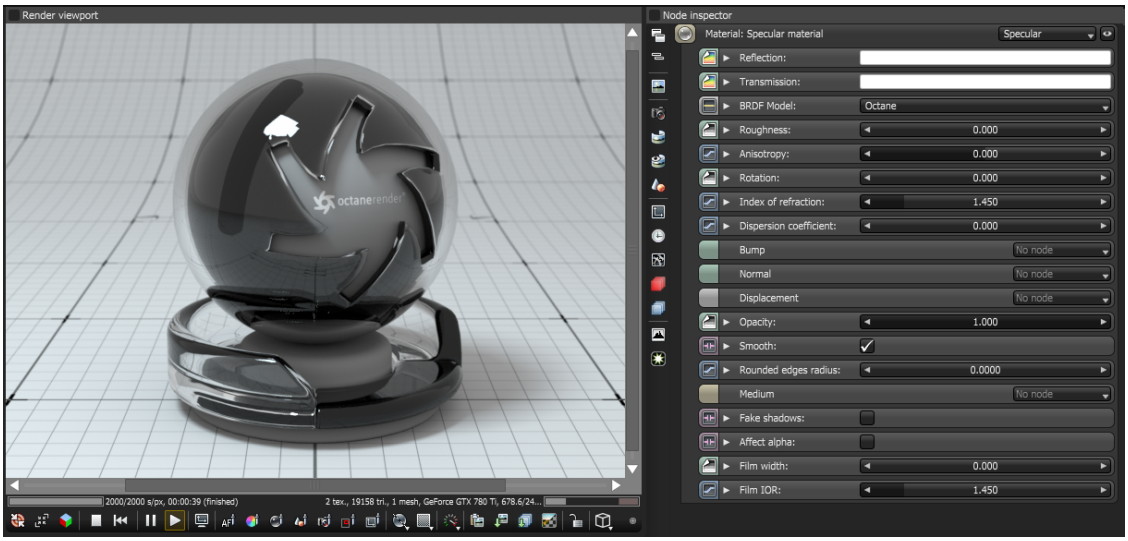


Figure 1: The OctaneRender® Specular material[2]

Specular Material Parameters

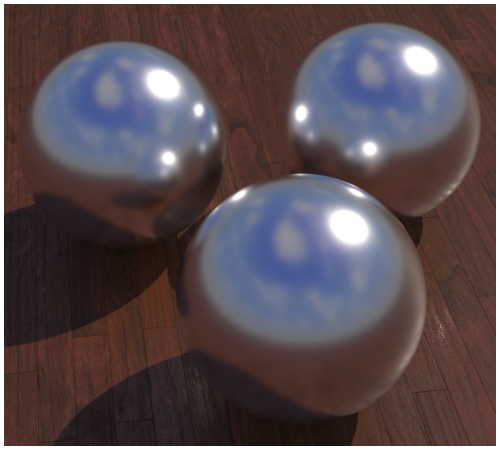
- **Reflection** - Determines surface reflection strength. Lower values increase the ability to transmit light through the object volume. Reflection and **Index of Refraction** work close together to tune Specular material reflectivity.
- **Transmission**[3] - Controls how light passes through a transparent surface. Transmission and Index of Refraction work close together to control surface transparency, and Transmission accepts color or texture input. A value of **1** lets light pass through the surface, making it transparent. To create a mirror surface, set this parameter to a black color, and set Index of refraction to **0** (Figure 2). To create colored

---

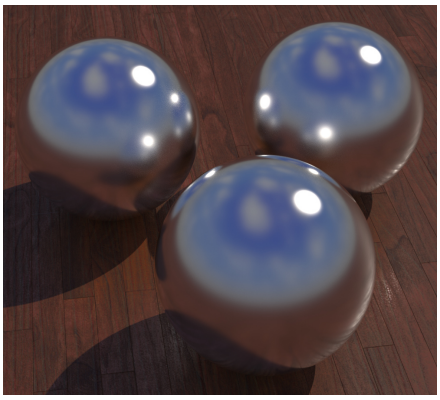
[1] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[2] Used for transparent materials such as glass and water.
[3] A surface characteristic that determines if light may pass through a surface volume.

glass, change the color to something other than white or black. Transmission is different than **Opacity** - Transmission controls transparency, while Opacity controls surface visibility. You can use Transmission to create reflective glass surfaces, and then use Opacity to create a hole in the surface.

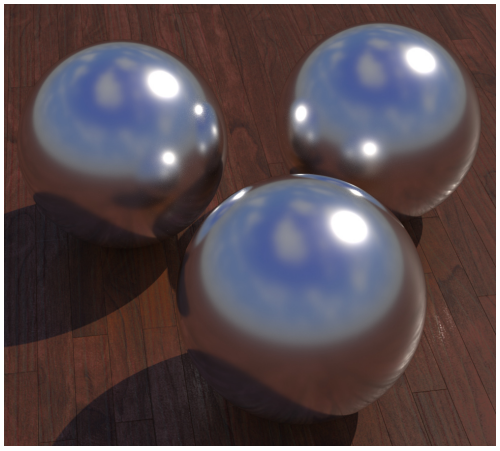
- **BRDF Model**- The BRDF (Bidirectional Reflectance Distribution Function) determines the amount of light that a material reflects when light falls on it. For Specular materials, you can choose from three BRDF models. Specific geometric properties (the micro-facet distribution) of the surface affects each BRDF, which describes the surface's microscopic shape (i.e. micro-facet normals) and scales the brightness of the BRDF's reflections. Refer to the topic on BRDF Models for more information.



Octane 3.07



Beckmann



GGX

Figure 4: The three BRDF Models applicable to Specular materials

- **Roughness** - Simulates the micro-facets effect in the surface, which blurs surface reflections and surface transparency. To create a translucent plastic look, you make a surface with a white or light-colored Transmission color and a Roughness value greater than **0**. This parameter accepts a color value or texture (**Procedural** or **Image**) - you'll want to use an alpha image or value. Hue information won't affect Roughness.
- **Anisotropy** - Controls the material's reflectance uniformity. Reflectance changes based on surface orientation or rotation is anisotropic. If the reflectance is uniform in all directions and doesn't change based on the surface's orientation or rotation, then it is isotropic. This parameter's default value is **0**, which sets the metallic material as isotropic. Non-zero values mean the material exhibits anisotropic reflectance, where **-1** is horizontal and **1** is vertical.

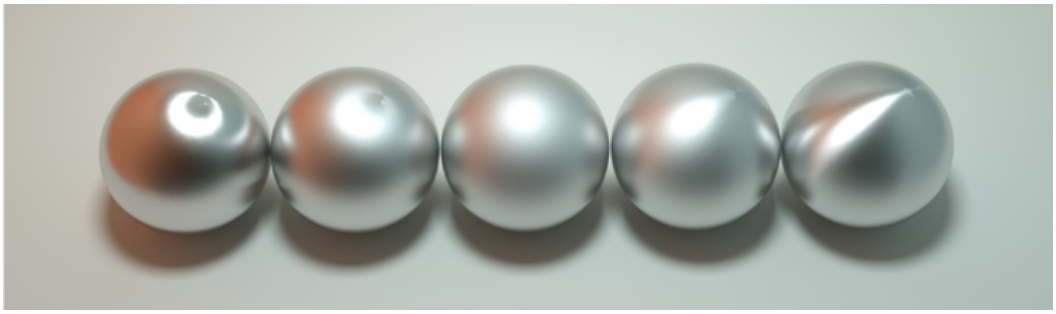


Figure 5: Anisotropic roughness exemplified in materials like brushed metal

- **Rotation** - The rotation of the anisotropic Specular reflection channel.
- **Index of Refraction** - Describes the change in the speed of light as it passes through a medium. As light photons move through surfaces like water, they slow down and change direction. This change appears as the object distorting on the other side of the water's surface. A vacuum's Index of Refraction (IOR) is **1**, and water's IOR is **1.33**, meaning light travels 1.33 times faster through a vacuum than water. Most transparent surfaces' IOR is accessible on the internet. Knowing a surface's correct IOR is important for replicating a surface's look in OctaneRender.
- **Dispersion Coefficient** - Increasing this value increases the coloration amount and dispersion in the object's transmission and in caustics.
- **Bump** - Creates fine details on material surfaces using a **Procedural** or **Image** texture. When you connect a **Grayscale** image texture to this parameter, light areas appear as protruding bumps, and dark areas appear as indentations. You can adjust the Bump map strength by setting the **Power** or

Gamma[1] values on the Grayscale image texture node. The Texture Overview category covers these attributes in more detail.

- **Normal** - Creates the look of fine detail on the surface. A Normal map is a special type of image texture that uses red, green, and blue color values to perturb the normals of the surface at render time, thus giving the appearance of added detail. They can be more accurate than Bump maps, but require specific software, such as ZBrush®, Mudbox®, Substance Designer, xNormal™, or others to generate. to load a full color Normal map, set the Normal channel to the **RGB Image** data type. Note that Normal maps take precedence over Bump maps, so you cannot use a Normal map and a Bump map at the same time.
- **Displacement**[2] - Adjusts surface vertices' height at render time using a Texture map. Displacement maps differ from Bump or Normal maps by having the texture alter the geometry, as opposed to creating the appearance of detail on the surface. Displacement mapping is more computationally expensive than Bump or Normal mapping, but results are more realistic, especially along the surface silhouette. The Texture Overview category covers Displacement mapping in more detail.
- **Opacity** - Determines what surface parts are visible in the render. Dark values indicate transparent areas, and light values indicate opaque areas. Values between light and dark create the look of semi-transparent areas. You can lower the Opacity value to fade the object's overall visibility, or you can use a Texture map to vary the surface's opacity. For example, if you want to make a simple polygon plane look like a leaf, you connect a black-and-white image of the leaf's silhouette to the **Diffuse**[3] shader's Opacity channel. When you use an Image texture map, set the **Data** type to Alpha image if the image has an alpha channel, or Grayscale image for black-and-white images, to load an image for setting transparency. Use the image's **Invert** checkbox to invert the transparency regions.
- **Smooth** - Smooths out the transition between surface normals. When disabled, the edges between surface polygons appear sharp, giving the surface a faceted look. when enabled, the edges between polygons blend together.
- **Rounded Edges Radius** - Bevels the surface edges automatically at render time, without the need to alter or subdivide the geometry. This option enhances object realism by eliminating sharp edges. Higher values produce rounder edges.
- **Medium** - OctaneRender has two mediums you can use to create translucent surfaces: **Absorption**[4] and **Scattering**[5]. To use these mediums, connect the Specular material's Medium input to one of these Medium nodes:

---

[1] The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

[2] The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

[3] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

[4] Defines how fast light is absorbed while passing through a medium.

[5] Defines how fast light gets scattered when traveling through the medium.

- **Absorption Medium** - Produces the appearance of a material that absorbs light while passing through a surface. The resulting color depends on the distance that light travels through the material. The Texture Overview category provides more detail about Absorption.
- **Scattering Medium** - Similar to the Absorption medium, but with an additional option for simulating subsurface scattering. Subsurface scattering is the phenomena that gives human skin and similar organic surfaces their characteristic glow under certain lighting conditions. It's a major component for creating the look of realistic skin. The Texture Overview category provides more detail about scattering.
- **Volume Medium**[1] - Adds color and other qualities to a **VDB**[2] file. VDBs are a generic volume format for creating effects such as smoke, fog, vapor, and similar gaseous objects. VDBs can consist of a single frame, or an animated sequence. 3D software packages like Houdini generate and export VDBs. You can also download VDB files at http://www.openvdb.org/download/.

- **Fake Shadows** - Activates the **Architectural** glass option for all meshes sharing that material. When enabled, Specular materials exhibit Architectural glass characteristics with its transparent feature, allowing light to illuminate enclosed spaces or frame an exterior view.
- **Affect Alpha** - This option lets refractions affect the alpha channel, as long as you enable the alpha channel in the **Kernel** settings.
- **Film Width** - Simulates the look of thin film material on a surface, like creating a rainbow color effect that appears on an oil slick's surface. Larger values increase the effect's strength.
- **Film IOR**- Controls the film Index of Refraction. This option adjusts the visible colors in the film.

---

[1] A shading system designed to render volumes such as smoke and fog.

[2] Dreamworks' open-source C++ library housing the data structures and tools implementation for storing and manipulating volume data, like smoke and other amorphous materials. The purpose of OpenVDB is mostly to have an efficient way to store volumetric data in memory and on disk. It has evolved into a more general toolkit that also lets you accomplish other things, such as fracturing volumes, converting meshes to volumes and vice versa. However, it does not include a computational fluid dynamics solver, and therefore it cannot procedurally generate smoke or fire. OpenVDB is fully integrated as a library in OctaneRender. For more information about OpenVDB, please see http://www.openvdb.org/.

Mix Material

The **Mix** material mixes any two material types, including other **Mix materials**[1]. It accepts any two material nodes as inputs, and you control the mix amount by a value, color, or texture. Figure 1 shows a white glossy material mixed with a red specular material using the default value of 0.5 (or 50%) as input for the **Amount** parameter.

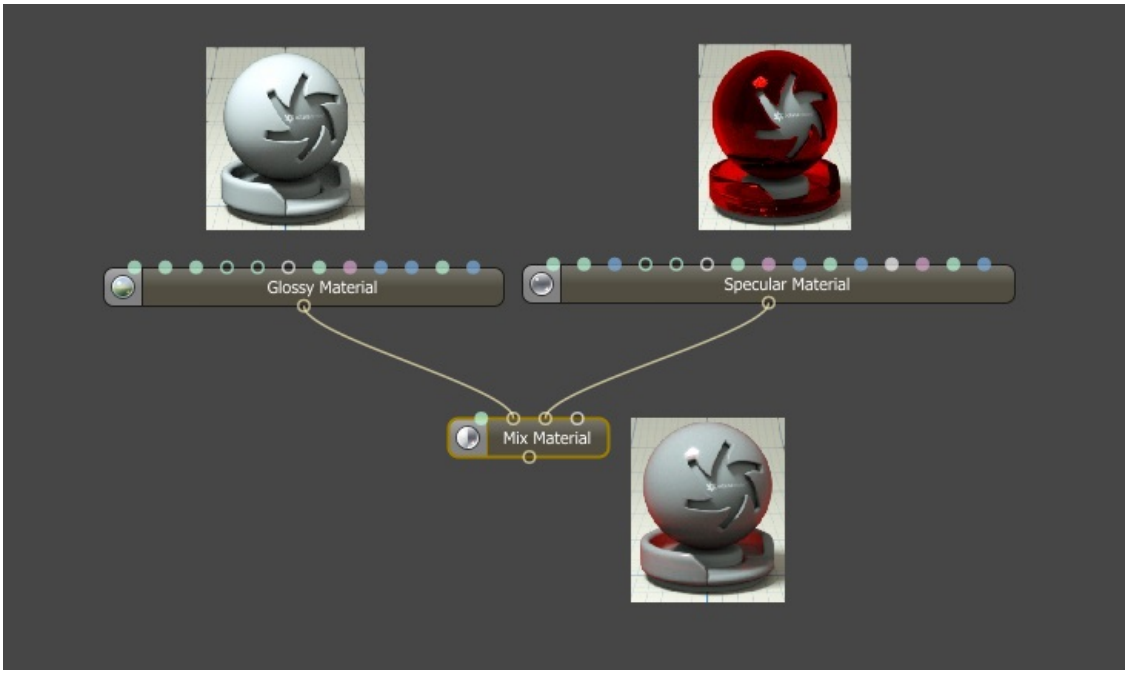


Figure 1: A Mix material is used to mix a glossy and specular materials together

Figure 2 shows the result of the same Mix material setup with a **Checks** texture used an input for the **Amount** parameter.

[1]Used to mix any two material types.

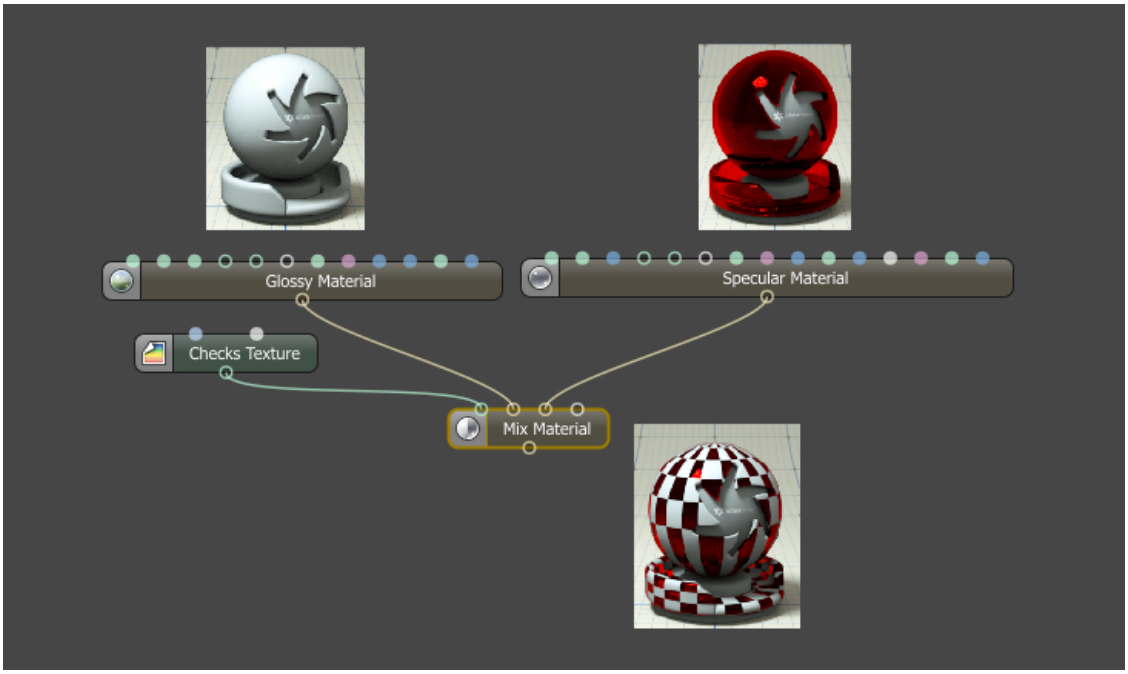


Figure 2: A Mix material mixing Glossy[1] and Specular[2] materials together and using a Checks texture to control the mix amount

---

1The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
2Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

Universal Material

The **Universal** material is used to get substance maps and **PBR**[1] outputs into OctaneRender. Substance Painter and other engines map perfectly to this material.

This material blends between dielectric and metallic with a metallic parameter **0** to **1**. When compared to other materials, the Universal material is equivalent to the **Metallic** material when the Universal material's Metallic attribute is set to **1.0**, while it is equivalent to the **Glossy**[2] material when the Universal material's metallic attribute is set to **0.0**.

The Universal material is designed such that it follows after the workflow in the PBR model, since the Metallic material falls short of the metallic maps and roughness maps that are commonly derived from Substance Painter and other tools. It handles dielectric material (diffuse and glossy BRDF) and also metallic material (glossy BRDF) with assumed **IOR** or custom IOR for both dielectric and metallic surfaces.

Material[3] IOR in the base layer of Universal materials is also not limited to scalar values, and this can be controlled procedurally with texture type nodes and OSL shaders connected to a new **IOR** texture input pin.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.
[2] The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
[3] The representation of the surface or volume properties of an object.

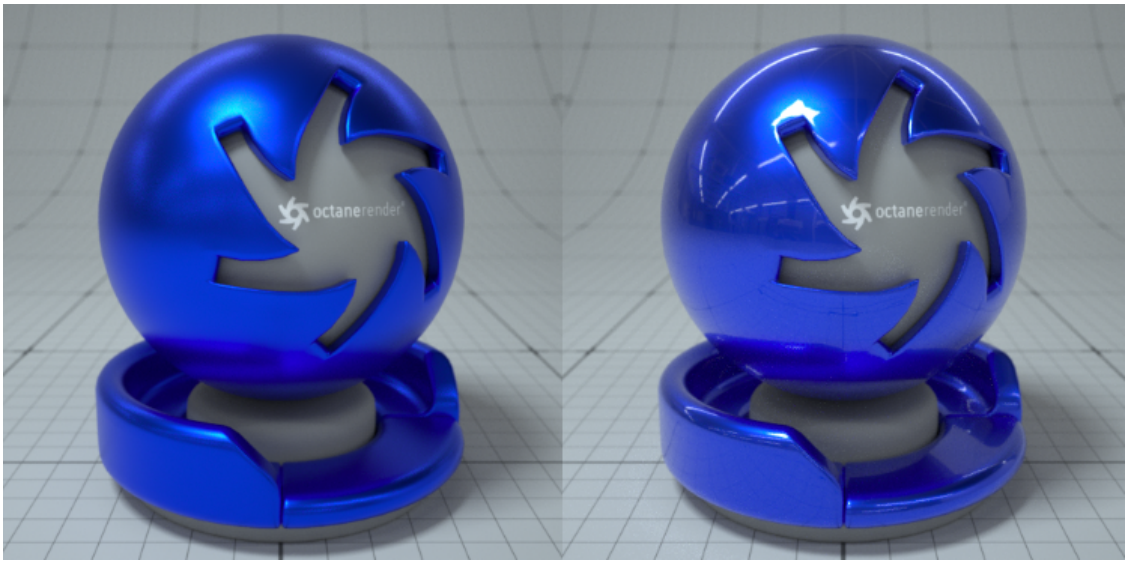


Figure 1: Example of coatings made possible by the the Universal material

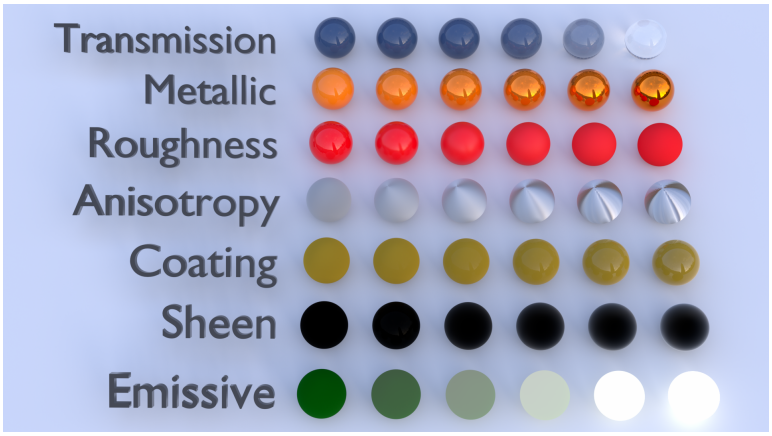


Figure 2: Basic and complex materials can be created procedurally using a Universal Material.

You can also import the BaseColor maps, Height maps, Normal maps, Occlusion maps and other texture maps for a scene that have been derived from major 3D painting software into OctaneRender and then re-link these texture maps to the corresponding pins of the Universal Material node (Figure 3). The Universal Material node will blend the glossy material and the metallic material depending on the settings of the metallic input. You can then begin to adjust the settings of each texture in greater detail. For example, you can place real world IOR

values of metallic objects as part of the IOR metallic input channels (Red, Green and Blue) of the Universal Material (Figure 4).

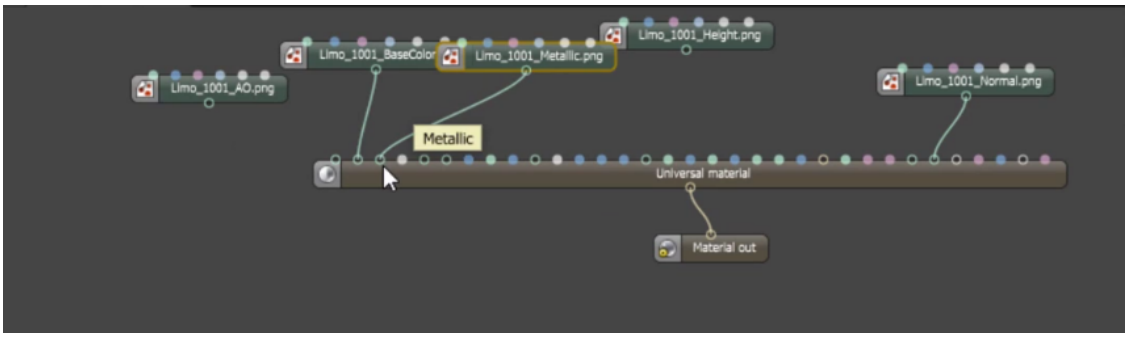


Figure 3: Relinking texture maps to the corresponding pins of the Universal Material.

Figure 4: Placing real world IOR values of metallic objects as part of the IOR metallic input channels (Red, Green and Blue) of the Universal Material.


mat0: Universal material
Universal
Base Layer
Transmission
No node
Albedo:
Metallic:
0.000
BSDF model:
GGX
Specular Layer
Roughness
Roughness:
0.0632
Anisotropy:
0.000
Rotation:
0.000
IOR
Dielectric IOR:
1.500
Dielectric 1/IOR map
No node
Metallic reflection mode:
Default
Metallic IOR:
0.000
0.000
Metallic IOR (green):
0.000
0.000
Metallic IOR (blue):
0.000
0.000
Coating Layer
Coating:
Coating roughness:
0.0632
Coating IOR:
1.500
Thin Film Layer
Sheen Layer
Transmission Properties
Geometric Properties
Bump
No node
Normal
No node
Displacement
No node
Smooth:
Rounded edges radius:
0.0000
Emission
No node
Shadow catcher:

Figure 5: The Universal Material Node.

The Base Layer Attributes

Transmission[1] - This is the transmission channel controlling the light passing the surface of the material (via refraction).

Albedo - This provides the base color of the material.

Metallic - This adjusts how metallic the material appears. It blends between dieclectric and metallic material.

BSDF Model - The BRDF is a function that determines the amount of light reflected from a material when light falls on it. Each BRDF is effected by a specific geometric property - the microfacet distribution - of the surface which describes the microscopic shape (i.e. microfacet normals) of that surface and serves as a function to scale the brightness of the reflections in the BRDF. Refer to the topic on BRDF Models for more information.

The Specular[2] Layer Attributes

Specular - This determines the color of the glossy reflection for dielectric materials. If the index of reflection is set to a value greater than 0, then the brightness of this color is adjusted so that it matches the Fresnel equations.

Roughness - This pertains to the roughness of the specular reflection and the transmission channel.

Anisotropy - This is the anisotropy of the specular and transmissive material:

- -1 = Horizontal
- 1 = Vertical
- 0 = Isotropy

Rotation - This adjusts the rotation of the anisotropic specular reflection and transmission channel.

Attributes affecting the Index Of Refraction

Dielectric IOR - This is the Index Of Refraction which controls the Fresnel effect of the

---

[1] A surface characteristic that determines if light may pass through a surface volume.

[2] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

specular reflection or transmission. By default, if 1/IOR map is empty, then the dielectric specular layer uses this IOR value.

Dielectric 1/IOR map - This is the Index Of Refraction map. Each texel represents 1/IOR. This is empty by default, therefore the material would use **Dielectric IOR** field as the default Index Of Refraction. If this is not empty, this will override the Index Of Refraction set by the **Dielectric IOR**.

Metallic reflection - This channel changes how the reflectivity is calculated for a metallic material:

- Default: This will use only the albedo color.
- IOR+color: This will use the albedo color and further adjusts the brightness using the IOR.
- RGB IOR: This is the most commonly used calculation and it uses only the 3 IOR values (for 650, 550 and 450nm) ignoring the albedo color.

Metallic IOR - Complex-valued Index Of Refraction (n-k*i) which controls the Fresnel effect of the specular reflection for metallic materials. For RGB mode, this serves as the Index Of Refraction for the red light (650nm).

Metallic IOR (Green) - For RGB mode, this is the Index Of Refraction for the green light (550nm).

Metallic IOR (Blue) - For RGB mode, this is the Index Of Refraction for the blue light (450nm).

Coating Layer Attributes

Coating - This sets the coating color of the material.

Coating Roughness - This adjusts the roughness of the coating layer.

Coating IOR - This sets the Index Of Refraction of the coating material.

Thin Film Layer Attributes

Film Width - This sets the thickness of the film coating.

Film IOR - This sets the Index Of Refraction of the film coating.

Sheen Layer Attributes

Sheen - This sets the sheen color of the material.

Sheen Roughness - This adjusts the roughness of the sheen channel.

Transmission Properties

Dispersion Coefficient - This is the coefficient **B** parameter of the Cauchy Dispersion Model where normal dispersion is derived through the relationship between the Index Of Refraction and the wavelength of light passing through transparent materials.

Medium - This is an option to add a medium inside the transparent material.

Opacity - This is the opacity channel that controls the degree of transparency for the material via a greyscale texture.

Fake Shadows - If enabled, light will be traced directly through the material during the shadow calculation, ignoring refraction.

Affect Alpha - If enabled, this allows the Universal Material's refractions affect the alpha channel.

Geometric Properties

Bump - This is used to simulate a relief using a greyscale texture interpreted as a height map.

Normal - This channel is used to distort normals using an RGB image.

Displacement[1] - This channel accepts displacement maps to allow the universal material to have more detailed geometry without adding overhead to hardware memory.

Smooth - If disabled, normal interpolation will be disabled and triangle meshes will appear "facetted". This is enabled by default.

Rounded Edges Radius - This adjusts the radius of rounded edges that are rendered as shading effect.

The Universal Material can also be used to create self-illuminated materials and shadow catchers.

Emission - This allows the universal material to emit light by plugging in a Black body emission or a Texture emission.

---

[1] The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

Shadow Catcher[1] - This switches the material to a shadow catcher. If enabled, the material will be transparent unless these is some direct shadow cast onto the material, which will make it less transparent depending on the shadow strength.

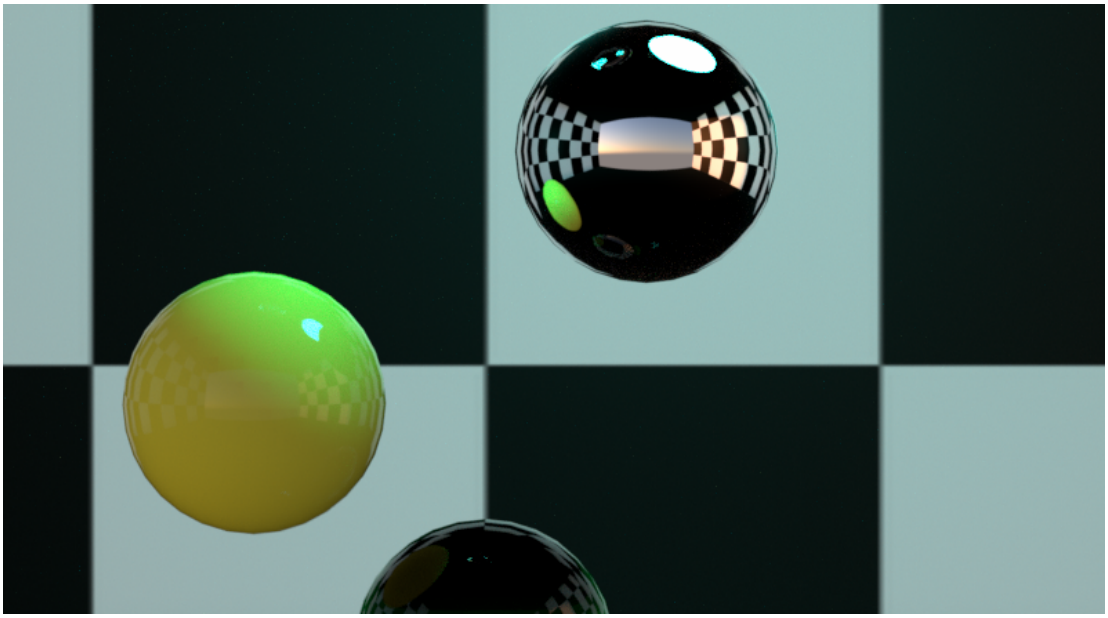


Figure 6: The yellow glossy material is the result of a Universal Material node.

[1]The Shadow Catcher can be used to create shadows cast by objects onto the surrounding background imagery. The shadows cast are not limited to simply a ground plane but can be cast onto other surfaces of varying shapes.

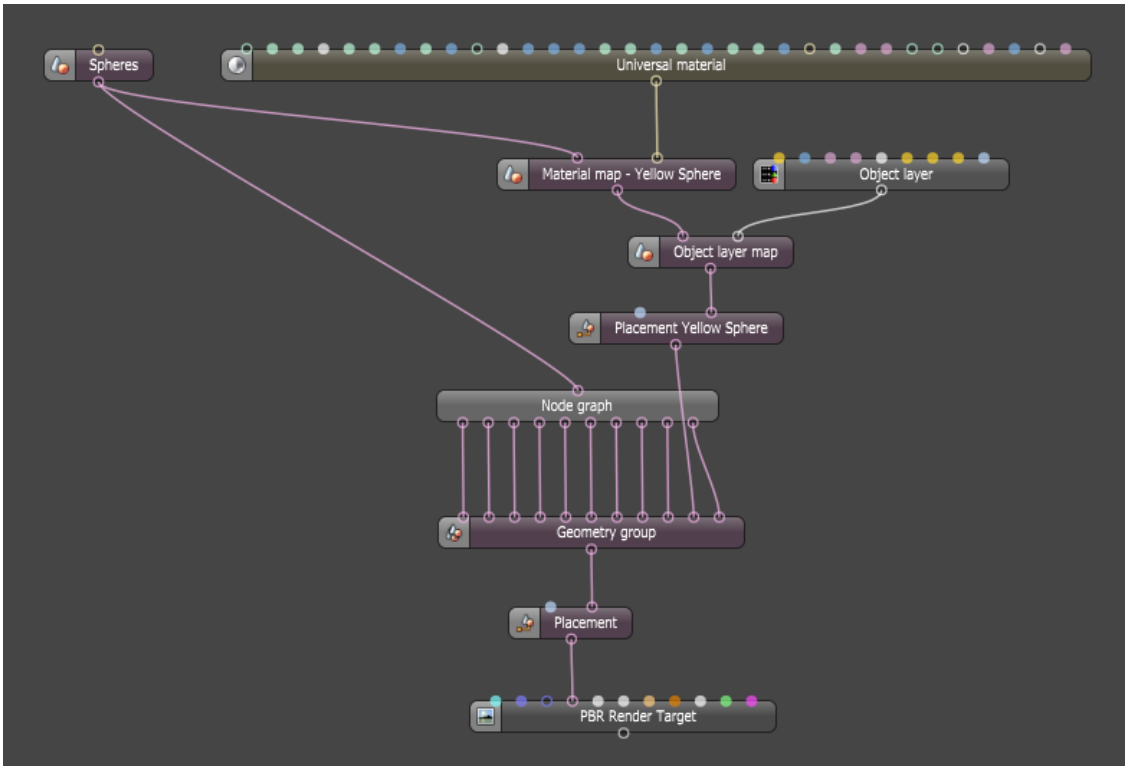


Figure 7: Typical node graph with a Universal Material mapped to a sphere geometry via a Material Map node.

Mediums – Subsurface Scattering and Volumes

OctaneRender® supports participating media inside objects. These settings are stored in **Medium** nodes, which are attached to the corresponding input pin of **Diffuse**[1] or **Specular**[2] material nodes.

There are three types of Medium nodes:

- **Scattering**[3] - Has parameters for **Absorption**[4], **Scattering**, and **Emission**.
- **Absorption** - A simple version with only Absorption parameters.
- **Volume** - See the Effects Overview section for more details.

To render with Medium nodes, the **Path Tracing** or **PMC** render kernels are the best choices. You can render mediums using the **Direct Light** kernel, but only if the Medium node is connected to **Diffuse** material, and if you set the kernel's **Diffuse mode** to **GI**.

To add a Medium node to a scene, right-click anywhere in the **Nodegraph Editor** window and select **Medium** from the pop-up menu (Figure 1). Choose the type of node you want to use. You should connect Absorption and Scattering mediums to the Medium input of the Diffuse or **Specular** material. Volume mediums can connect to **VDB**[5] file inputs. **Schlick Phase Function** and **Volume Gradient** are special nodes that modify the other Medium nodes.

---

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

[2] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

[3] Defines how fast light gets scattered when traveling through the medium.

[4] Defines how fast light is absorbed while passing through a medium.

[5] Dreamworks' open-source C++ library housing the data structures and tools implementation for storing and manipulating volume data, like smoke and other amorphous materials. The purpose of OpenVDB is mostly to have an efficient way to store volumetric data in memory and on disk. It has evolved into a more general toolkit that also lets you accomplish other things, such as fracturing volumes, converting meshes to volumes and vice versa. However, it does not include a computational fluid dynamics solver, and therefore it cannot procedurally generate smoke or fire. OpenVDB is fully integrated as a library in OctaneRender. For more information about OpenVDB, please see http://www.openvdb.org/.

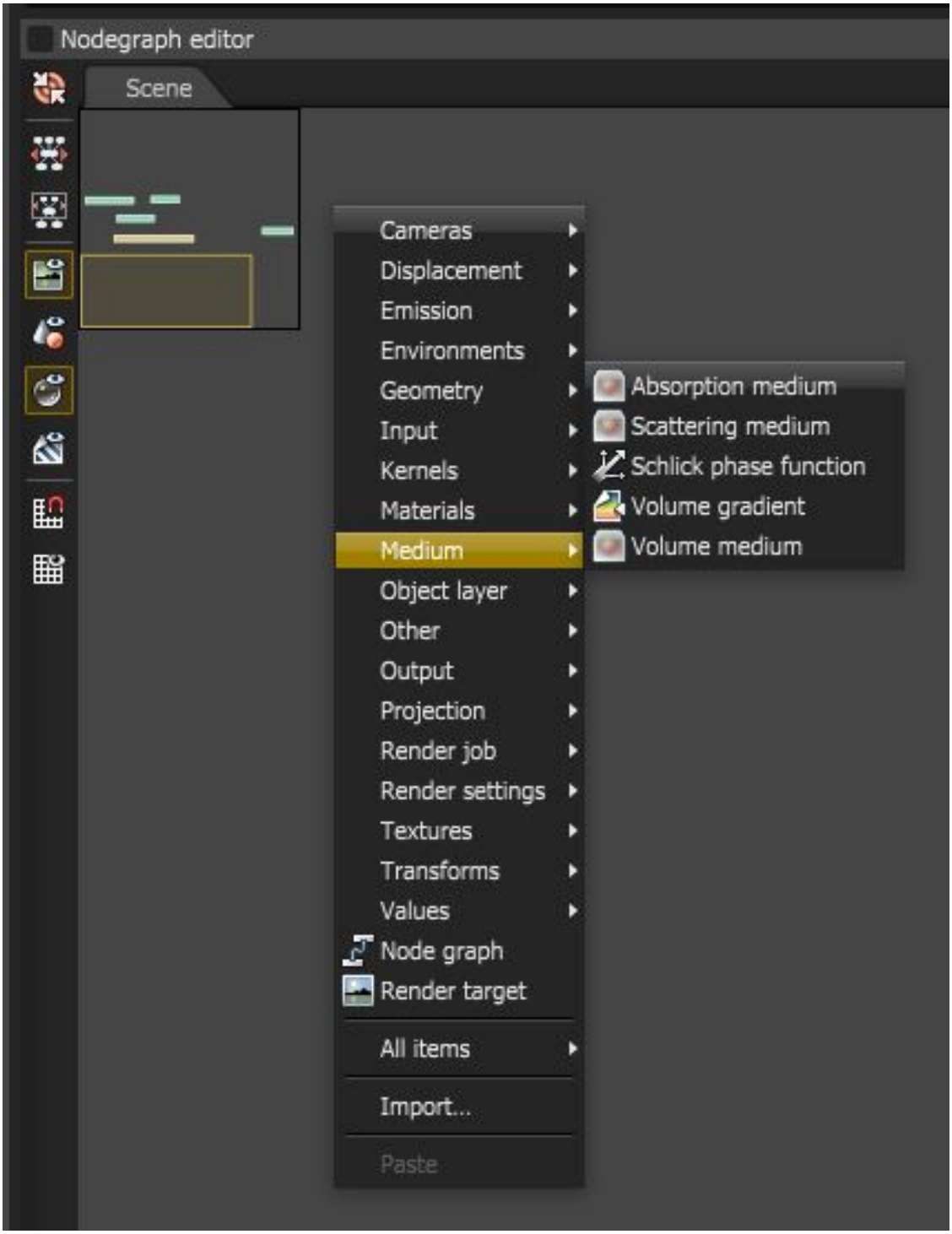

Nodegraph editor
Scene
Cameras
Displacement
Emission
Environments
Geometry
Input
Kernels
Materials
Medium
Object layer
Other
Output
Projection
Render job
Render settings
Textures
Transforms
Values
Node graph
Render target
All items
Import...
Paste
Absorption medium
Scattering medium
Schlick phase function
Volume gradient
Volume medium

Figure 1: Select the Medium node from the pop-up menu when you right-click in the Node-graph Editor

There are some things to keep in mind about using **Mediums**[1] with Meshes and Specular materials:

Meshes

Medium nodes should only be added to materials you've applied to meshes that define a closed volume. A single-sided plane will not work. For example, a plane representing a leaf will not work properly if a material with a medium is applied to it. The one exception is a plane representing the ground: OctaneRender treats the ground plane as an infinitely deep surface.

Specular **Materials**[2]

Specular materials are the best choice when using a Medium node. When using a **Specular material**[3], set the value of the **Reflection** parameter to a low value because only the part of the spectrum that is not reflected can enter the object for scattering. If you set Reflection to **1**, all light reflects regardless of the **Transmission**[4] value. If Reflection is set to **0**, all light transmits through the surface, but the result is an unnatural appearance. Reflection values of **0.1** - **0.2** are good starting points.

---

[1] The behavior of light inside a surface volume described by scatter, absorption, and transmission characteristics.

[2] A set of attributes or parameters that describe surface characteristics.

[3] Used for transparent materials such as glass and water.

[4] A surface characteristic that determines if light may pass through a surface volume.

Absorption Medium

Absorption Parameters

- **Absorption**[1] - Controlled by Absorption color, which defines how fast a medium absorbs light passing through it. A **0.0** or black value means no absorption. Higher values result in faster light absorption. The specified color in the Absorption parameter produces its complimentary color in the rendering (Figure 1). The Absorption texture is multiplied by the **Density** parameter. This allows setting a wide range of values.

[1] Defines how fast light is absorbed while passing through a medium.

Figure 1: Specifying a blue hue in the Absorption parameter absorbs blue more quickly, thus giving the object an orange appearance

- **Volume Step Length** - Depending on the surface, you may need to adjust this parameter. The default value is **4**, but if the volume is smaller than this, you need to decrease the Step Length. Decreasing this value decreases render speed, and increasing the value causes the ray marching algorithm to take longer steps. If the Step Length exceeds the volume's dimensions, then the ray marching algorithm takes a single step through the whole volume. To get the most accurate results, keep the Step Length as small as possible.
- **Invert Absorption** - Inverts the Absorption color so that the Absorption channel becomes a **Transparency** channel. This helps visualize the effect of the specified color, since a neutral background shining through the medium appears in that approximate color.

Scattering Medium

The **Scattering**[1] medium node helps create the look of subsurface scattering. Subsurface scattering is the phenomena where light rays enter a surface, are scattered within the material of surface, and then exit the surface (Figure 1).

Figure 1: Subsurface scattering is the key to creating the look of realistic human skin and other organic surfaces

[1] Defines how fast light gets scattered when traveling through the medium.

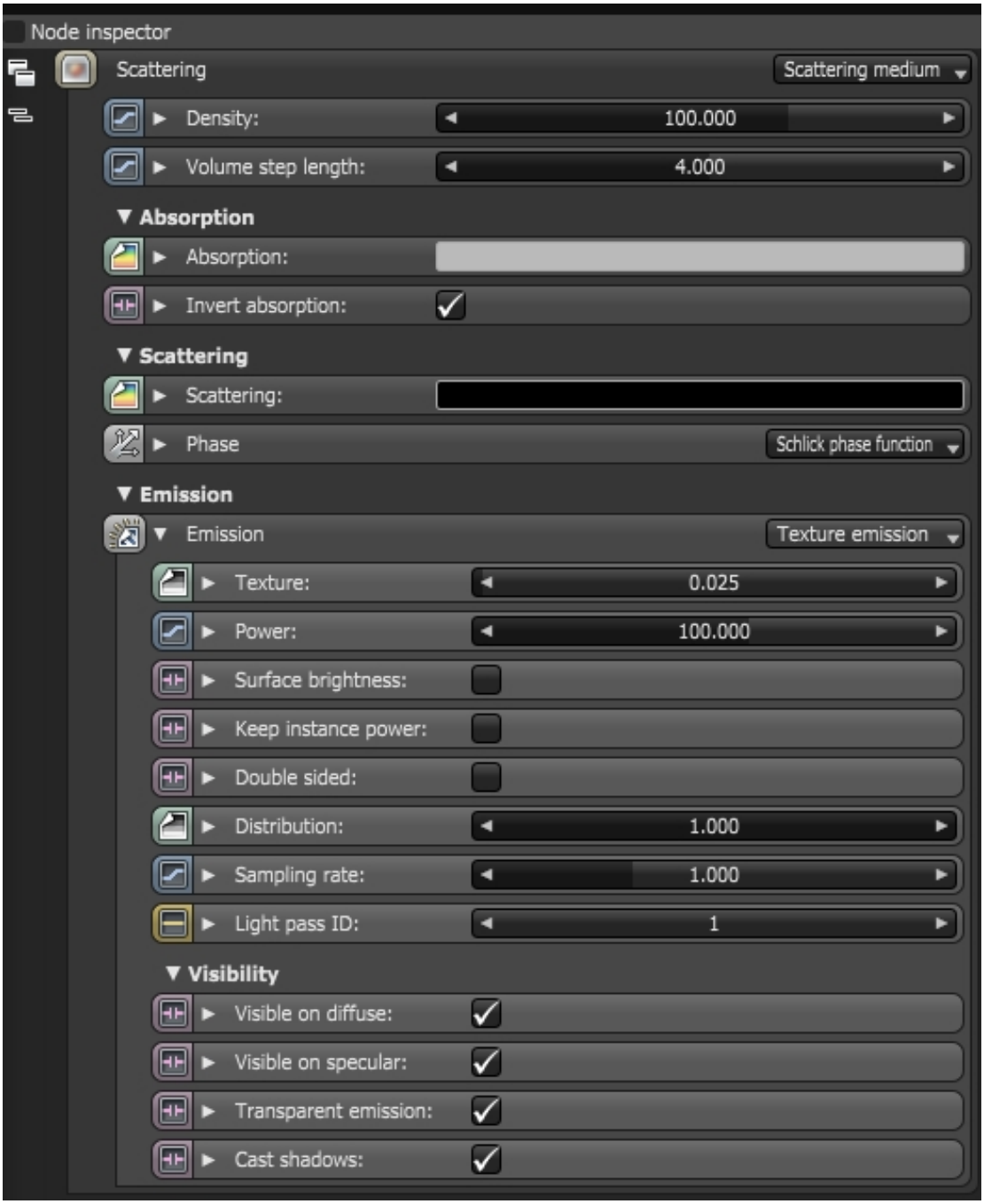


Figure 2: The Scattering node parameters

Scattering Parameters

- **Density** - This parameter multiplies against **Scattering**.
- **Volume Step Length** - Depending on the surface, you may need to adjust this parameter. The default value is **4**, but if the volume is smaller than this, you need to decrease the Step Length. Decreasing this value decreases render speed, and increasing the value causes the ray marching algorithm to take longer steps. If the Step Length exceeds the volume's dimensions, then the ray marching algorithm takes a single step through the whole volume. To get the most accurate results, keep the Step Length as small as possible.
- **Absorption**[1] - Controlled by Absorption color, which defines how fast a medium absorbs light passing through it. A **0.0** or black value means no absorption. Higher values result in faster light absorption. The specified color in the Absorption parameter produces its complimentary color in the rendering (Figure 1). The Absorption texture is multiplied by the **Density** parameter. This allows setting a wide range of values.
- **Scattering** - Determines how fast light scatters as it moves through a surface. High values mean that light scatters sooner as it enters a surface, and low values mean that light passes deeper into the surface before scattering. A **0** value disables Scattering.
- **Phase** - Controls light direction as it scatters through the surface. A 0 value results in light scattering equally in all directions; a positive value results in forward scattering, where photons continue the same approximate direction as when they enter the surface; and a negative value results in backwards scattering, where light moves through the surface in the same direction, but opposite to the angle that they entered the surface (this is known as backscattering).
- **Emission** - Attaches an **Emission** node to the **Emission** input pin. When you connect an Emission node to a Medium node, it defines emission inside the volume instead of on the object's surface. In this case, **Power** controls how fast a ray's radiance increases while traveling through the volume; it doesn't represent total power. It's not multiplied with the **Scale** parameter. This effect works best with large, not-too-bright objects - small, bright objects create lots of noise.

[1] Defines how fast light is absorbed while passing through a medium.

Textures

You can add **Texture** maps to the **Standard** and **Standard (Specular[1] Setup)** materials. They function in OctaneRender® in much the same way as they do in Unity® (Figure 1).

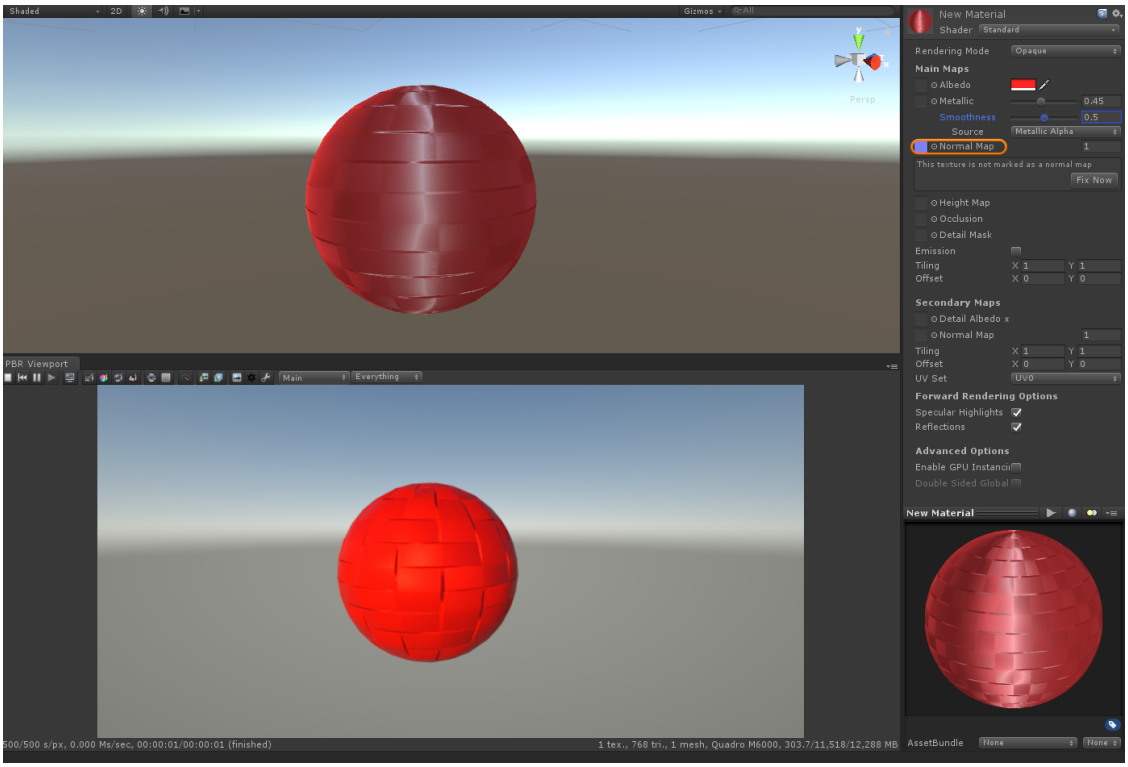


Figure 1: A Texture map added to the Normal map channel of a Standard material

You can further modify **Textures**[2] in the **OctaneVR**® window (Figure 2). This approach provides access to many more Texture nodes, which allows for deeper control over the material design process when using OctaneRender.

---

[1] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[2] Textures are used to add details to a surface. Textures can be procedural or imported raster files.

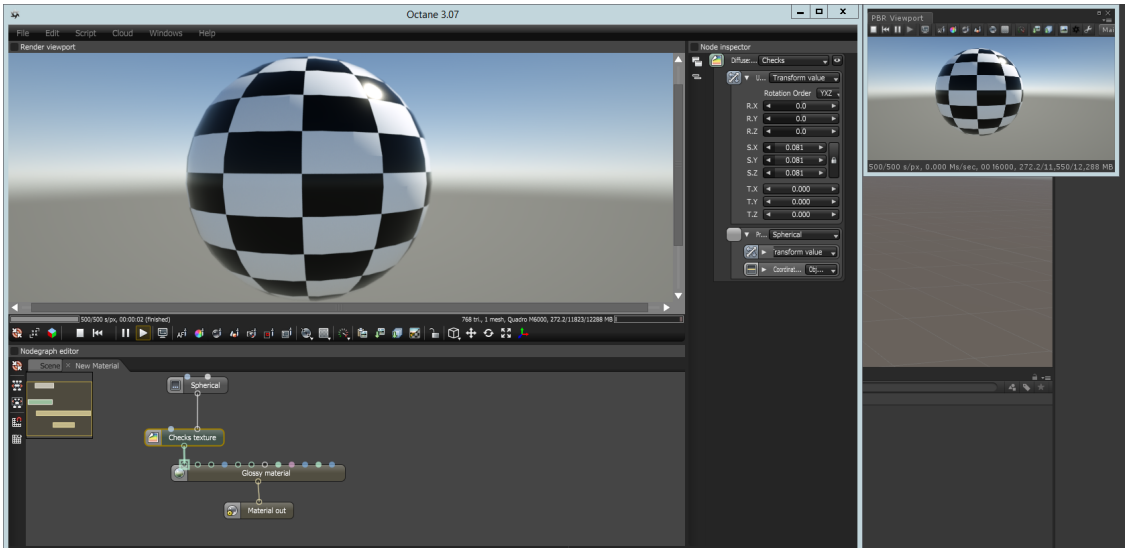


***Figure 2: A Checks texture added to a Glossy material*[1]**

[1] Used for shiny materials such as plastics or metals.

Working with Unity® Textures

Unity® **Textures**[1] used with the **Unity Standard** or **Standard (Specular**[2] **Setup)** materials are fully compatible with OctaneRender® (Figure 1).

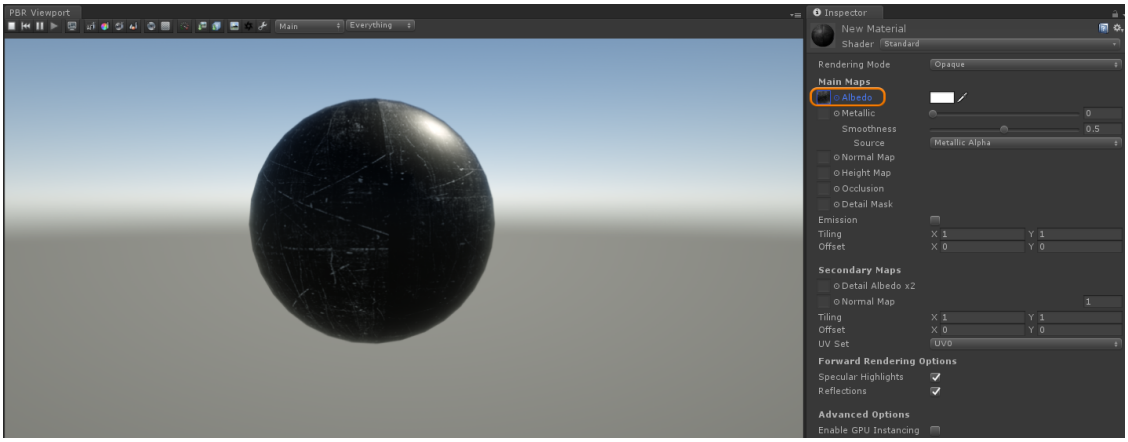


Figure 1: A Texture map connected to the Albedo parameter of a Unity Standard material

[1]Textures are used to add details to a surface. Textures can be procedural or imported raster files.

[2]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

Working With OctaneRender® Textures

OctaneRender® for Unity® provides many additional texture types that allow for the creation of deep material design. In order to access the OctaneRender-specific Texture nodes, you must use the **OctaneVR®** window. You can only use OctaneRender Texture nodes with OctaneRender-specific materials (**Diffuse**[1], **Glossy**[2], **Specular**[3], **Mix**, **Metallic**, **Portal**[4], **Toon**, **Toon Ramp**, **Universal**). All OctaneRender textures can be found by right-clicking in the **Nodegraph Editor** and selecting the **Textures**[5] category (Figure 1).

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
[3]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[4]A technique that assists the render kernel with exterior light sources that illuminate interiors. In interior renderings with windows, it is difficult for the path tracer to find light from the outside environment and optimally render the scene. Portals are planes that are added to the scene with the Portal material applied to them.
[5]Textures are used to add details to a surface. Textures can be procedural or imported raster files.

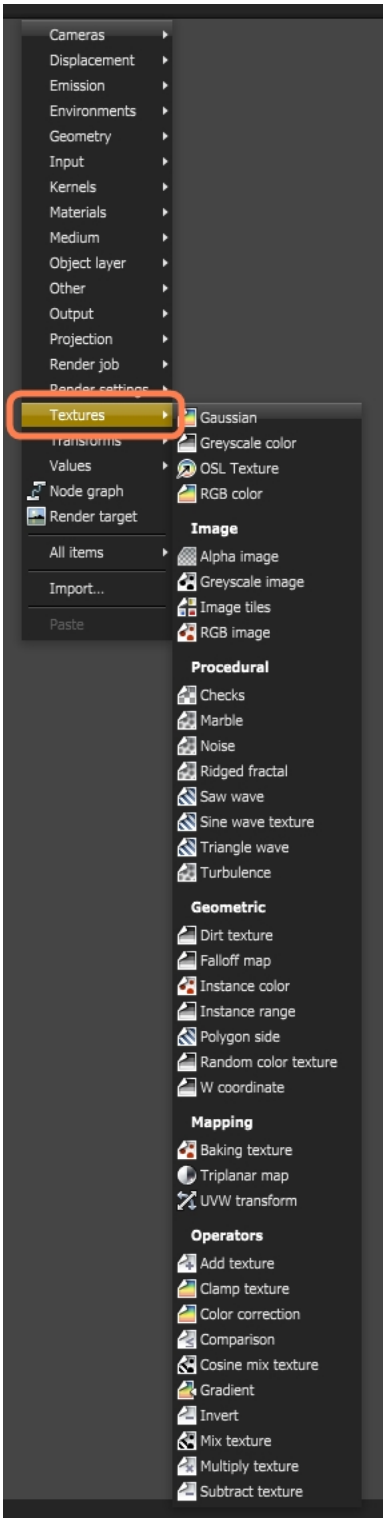


Figure 1: Accessing OctaneRender Texture nodes in the Nodegraph Editor

Gaussian Spectrum

This output uses a Gaussian distribution spectrum. **Wavelength** sets the center of the spectrum, and **Width** sets the curve width. The narrower the Width, the more pure and saturated the color (Figure 1).

Figure 1: The parameters of the Gaussian Spectrum texture in the Node Inspector

Gaussian Spectrum Parameters

- **Wavelength** - This represents the mean wavelength approximation between 380 nm – 720 nm (with a range from **0** - **1**). Lower values appear bluish, and higher values (around 700 nm) appear reddish.

- **Width** - When set to **0**, almost no color is visible. When set to **1**, the color spreads thin over a large space, and the texture appears faint.
- **Power** - The texture brightness.

Greyscale Color

The **Greyscale Color** node generates a float value, which node networks use as an input. When connected to a **Color** input, the result is a greyscale color, with **0** being equivalent to black, and **1** being equivalent to white (Figure 1).

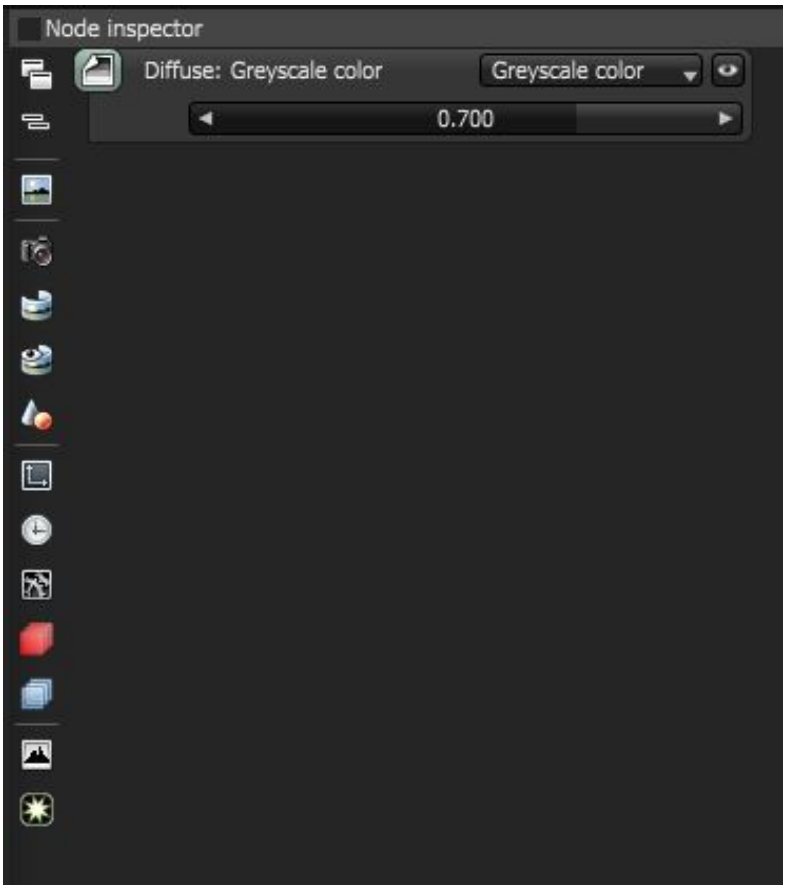


Figure 1: Parameters for the Greyscale Color texture

OSL Texture

The **OSL Texture** node is a scriptable node where users may write scripts using the OSL (**Open Shader Language**[1]) standard programming language to define arbitrary texture types to create highly customized OctaneRender materials and shaders. OSL is a standard created by Sony Imageworks. To learn about the generic OSL standard, information is provided from the OSL Readme and PDF documentation.

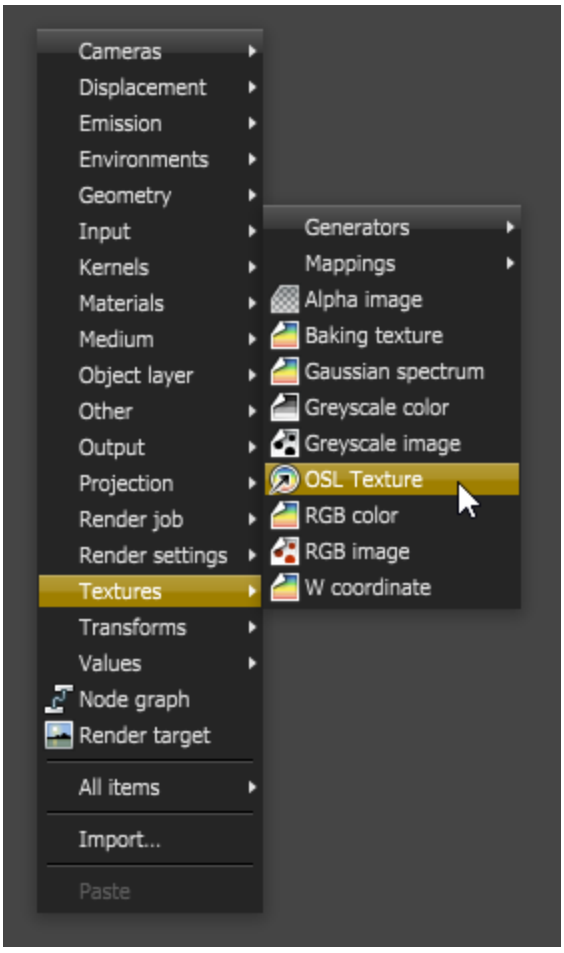


Figure 1: Select the OSL Texture node from the pop-up context menu of the Node Graph Editor pane

[1] A shading language developed by Sony Pictures Imageworks. There are multiple render engines that utilize OSL as it is particularly suited for physically-based renderers.

An OSL script is written into the OSL Texture node via the script editor window. Click on the pencil icon to go to the script editor window. If the script exists as an external .osl file, insert the .osl file into the node through the load icon. Any existing file used within an OSL Texture node may be edited. To refresh file with the edits, reload the file via the reload icon.

The OSL Texture is only for textures and requires one output `color`. One OSL Texture node is one OSL compilation unit which contains only one shader, thus, it has only one output attribute pin that connects to the texture input pin of other Octane Texture nodes (e.g. Turbulence texture node) or to the texture input pin of Octane materials (e.g. **Diffuse**[1] **Material**[2] node).

Note: Multiple outputs for OSL Texture is not yet supported at this stage.

As of version 3.08.x, OctaneRender supports most of the texturing functions (eg. textures or noises) in the OSL Specification.

When an OSL Texture node is invoked, it is provided with an initial script with a declaration component that bares one output variable which represents a color. The initial script's function body then initializes the color to black based on the standard RGB color mode. Customize the script to create a customized OSL Texture shader. A custom script may have many variables, some of which may require user input via input nodes. So depending on the custom script, the OSL Texture may have an input of **0**, or many input nodes.

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]The representation of the surface or volume properties of an object.

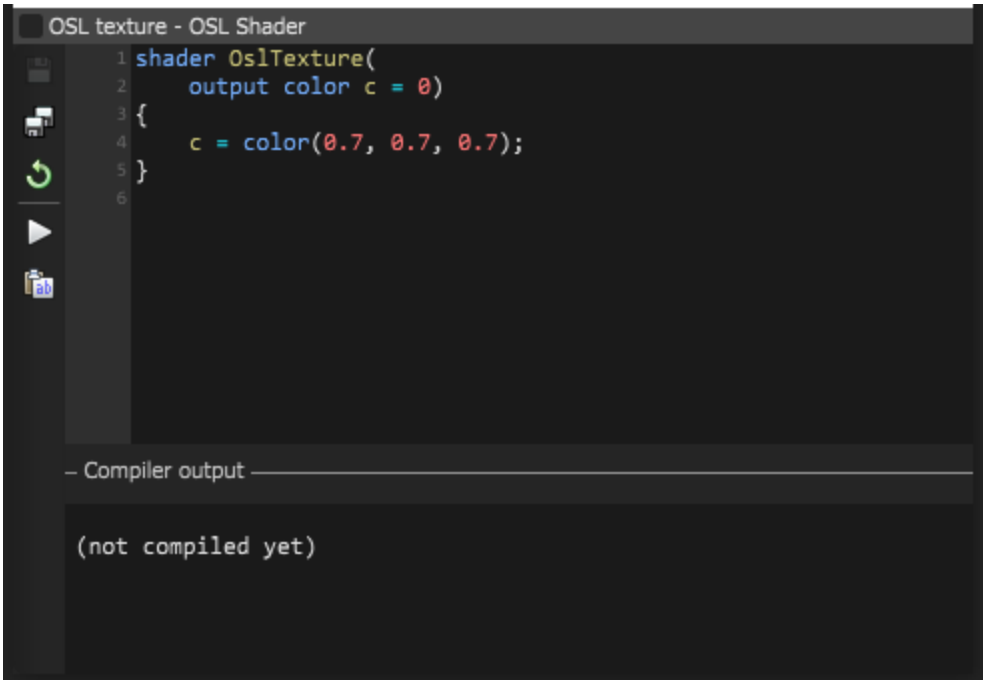


Figure 2: An OSL Texture node's initial script containing an OSL program code to calculate the output color

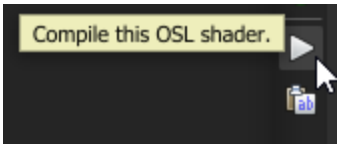


Figure 3: Click on the Compile button to compile the script. Compilation messages will display below the script window

The example in Figure 4 below shows an OSL shader that adds two RGB textures.

```
Shader Add (
    color a = 0,
    color b = 0,
    output color c = 0)
```

```
{
        c = a + b;
}
```

The script's declaration component has three variables, two are input variables of **Input Type** `color` and the third one is the required output represented by a variable of **Output Type** `color`. The OSL **Input/Output Type** "`color`" corresponds to an **Octane Texture Attribute** node. The script's function body adds the two input parameters and places the result into the output variable.

For a list of OSL variable declaration Input/Output types in the OSL Specification that OctaneRender supports, refer to the Appendix topic on OSL Implementation in Octane. To learn more about scripting within OctaneRender using the Open Shader Language, refer to The Octane OSL Guide.

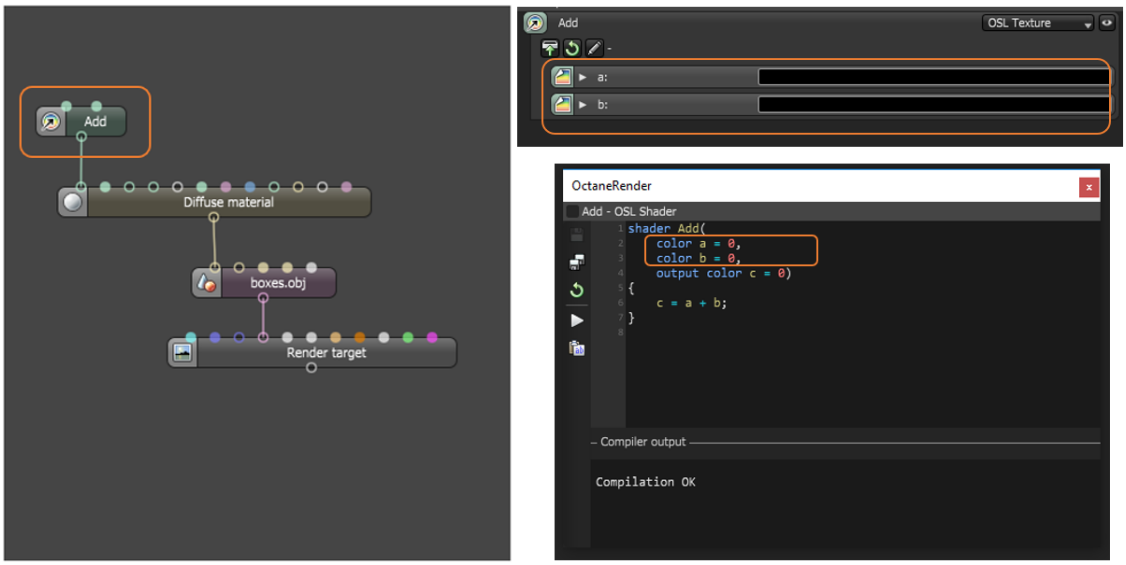


Figure 4: An example of an OSL Texture node with an OSL script to logically add two input texture nodes and send the result to one output

Once the OSL Texture is available, it is plugged into the texture input pin of an OctaneRender material node. And similar to other procedural texture types, its own initial attributes can be replaced with other applicable texture types in the course of the texturing process. For example, the OctaneRender texture attribute in the example provided in Figure 3 is initially an RGB Color texture, however this may be changed to any applicable

OctaneRender texture via its representation on the inspector node to further customize the OSL shader (Figure 5).

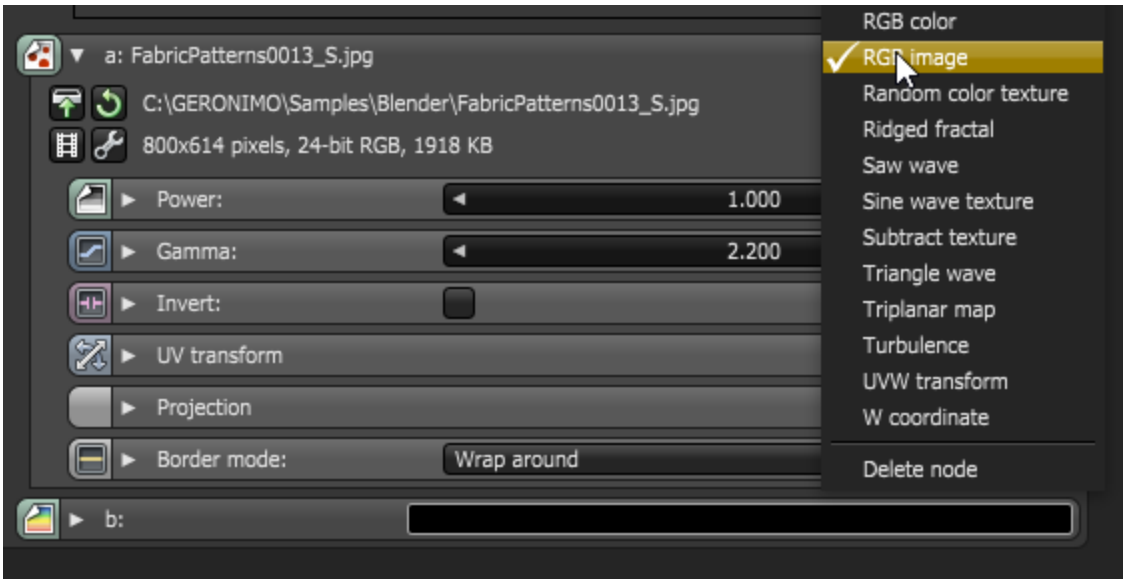


Figure 5: Changing the input node to customize the OSL Shader

To learn more about coding in OSL, refer to the OSL Implementation In Octane topic in the Appendix.

Note: When using OSL, you should ensure shaders never lock up or run for exceedingly long times. This may cause the system to freeze, or to reset the display driver. Also, some operations, like out-of-bounds array access, may cause kernel crashes.

Example of an OSL Script for an OSL Texture Node

This is an example of a shader with a few different inputs with meta data. The script below creates the **Mandelbrot Set** shader.

```
// The obvious piece of programmer art in any OSL intro is a Mandelbrot shader


shader Mandelbrot(
     output color c = 0,
     int maxcount = 100
          [[int min=1, int max=20000, int sliderexponent=4,
          string label = "Max iterations",
          string help = "Maximal amount of iterations (z = z² +
          c)"]],
     int outputScale = 100
          [[int min=1, int max=20000, int sliderexponent=4,
          string label = "Iterations scale factor",
          string help = "Amount of iterations mapped to white"]],
     float gamma = 1.0
          [[float min=0.1, float max=10, int sliderexponent=4,
          string label = "Gamma[1]",
          string help = "Gamma value applied to the gradient"]],
     point projection = 0
```

[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

```
        [[string label = "Projection"]],
    matrix xform = matrix(.33333, 0, 0, -.5, 0, .33333, 0, -.33333, 0,
    0, .33333, 0, 0, 0, 0, 1)
        [[string label = "UV transform", int dim = 2]],
    int smooth = 0
        [[string widget="checkBox",
        string label = "Smooth gradient",
        string help = "Smooth out the gradient outside the Man-
        delbrot set"]]
)
{
    point p = transform(1 / xform, projection);
    float cx = (p[0]-.5);
    float cy = (p[1]-.5);
    float x = cx;
    float y = cy;
    float prevRR = 0;
    float rr = x*x + y*y;
    int count = 0;
    while (rr < 4 && count < maxcount)
    {
        count += 1;
        // z = z² + c
        float x2 = x*x - y*y + cx;
        y = 2*x*y + cy;
        x = x2;
        prevRR = rr;
```

```
            rr = x*x + y*y;
        }
        if (count < maxcount)
        {
            float h = (float)count;
            if (smooth)
            {
            h = h + (4 - prevRR) / (rr - prevRR);
            }
            c = pow(h / outputScale, gamma);
            c = min(c, .999);
        }
        else
        {
        c = 1.0;
        }
}
```

The Mandelbrot Set Shader

First, what is the Mandelbrot set? The Mandelbrot set is a famous example of a fractal in mathematics, it is one of the first images generated in computer graphics that displayed a fractal geometric shape. It showed how visual complexity can be created from simple rules and that a "degree of order" is actually present in things that are typically considered as messy or chaotic (i.e., like clouds or shorelines).

The computer graphic image was created by applying the common fractal equation $Z_{n+1}=Z_n^2+c$ to each pixel in an iterative process. In that equation, c and z are complex numbers and n is zero or a positive integer (natural number). Starting with $z_0=0$, c is in the Mandelbrot set if the absolute value of zn never becomes larger than a certain number (that number depends on c), no matter how large n gets. The resulting image became known as the Mandelbrot Set (Figure 1).

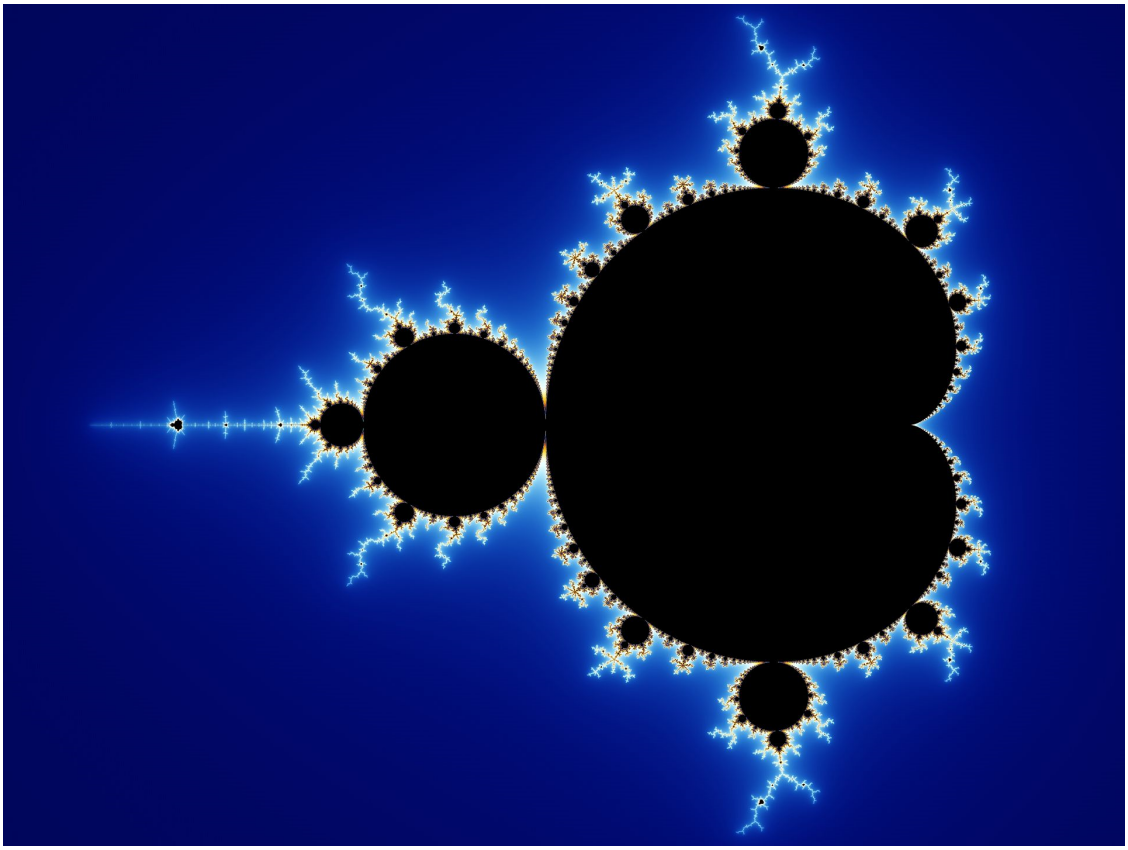


Figure 1: The Mandelbrot set

Therefore, the purpose of the Mandelbrot Set Shader is to recreate the Mandelbrot set fractal procedurally via an OSL script and allow the resulting image to be displayed on a surface when the shader is used on a material for that surface (Figure 2).

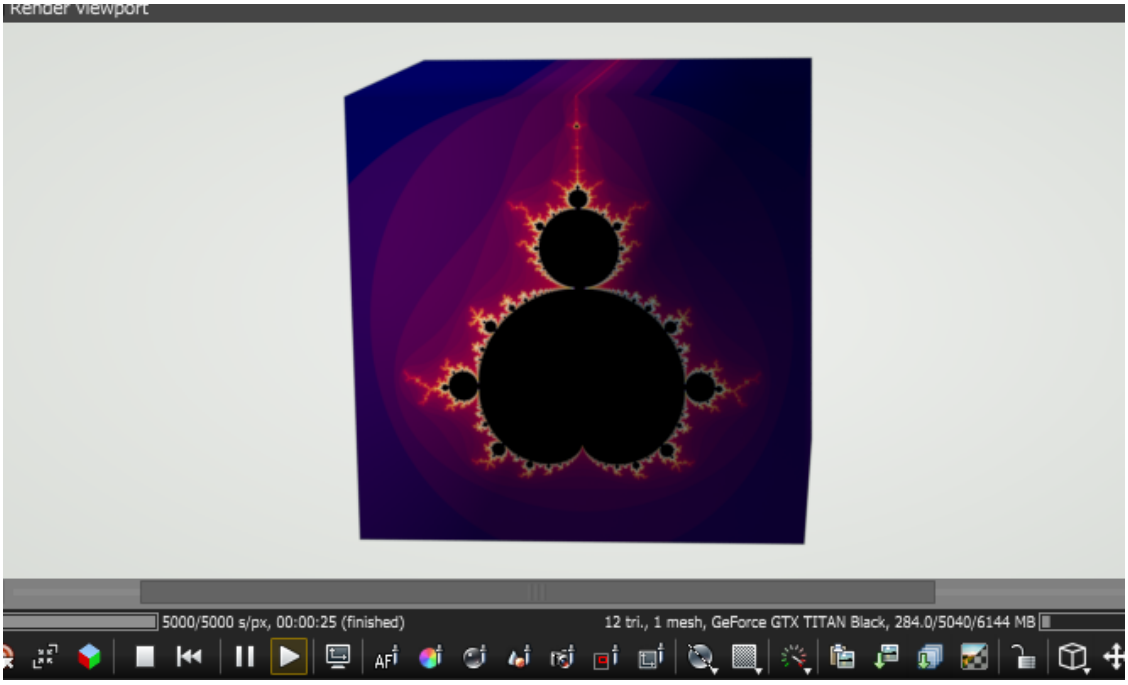


Figure 2: The Mandelbrot Set displayed on the surface of a cube

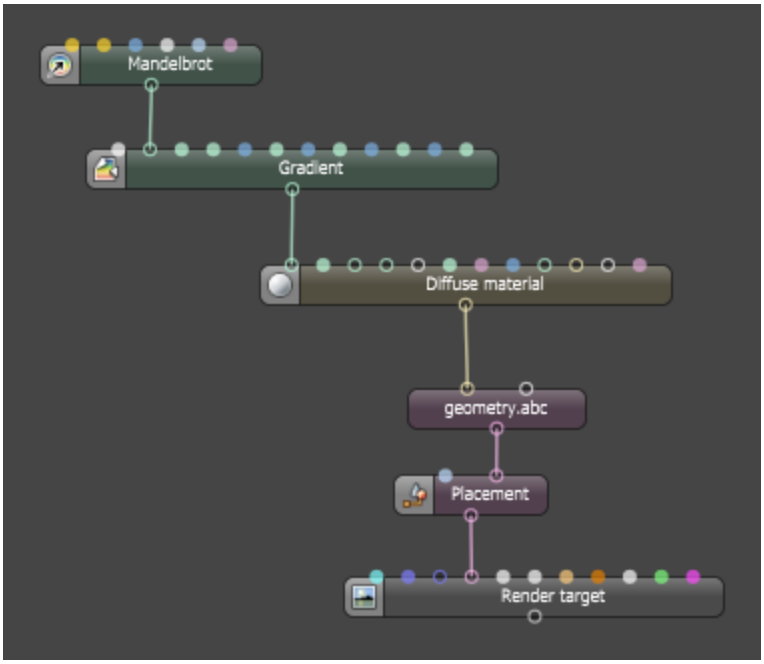


Figure 3: The Mandelbrot Set Shader is plugged into the texture input pin of a Gradient node

Dissecting the Code

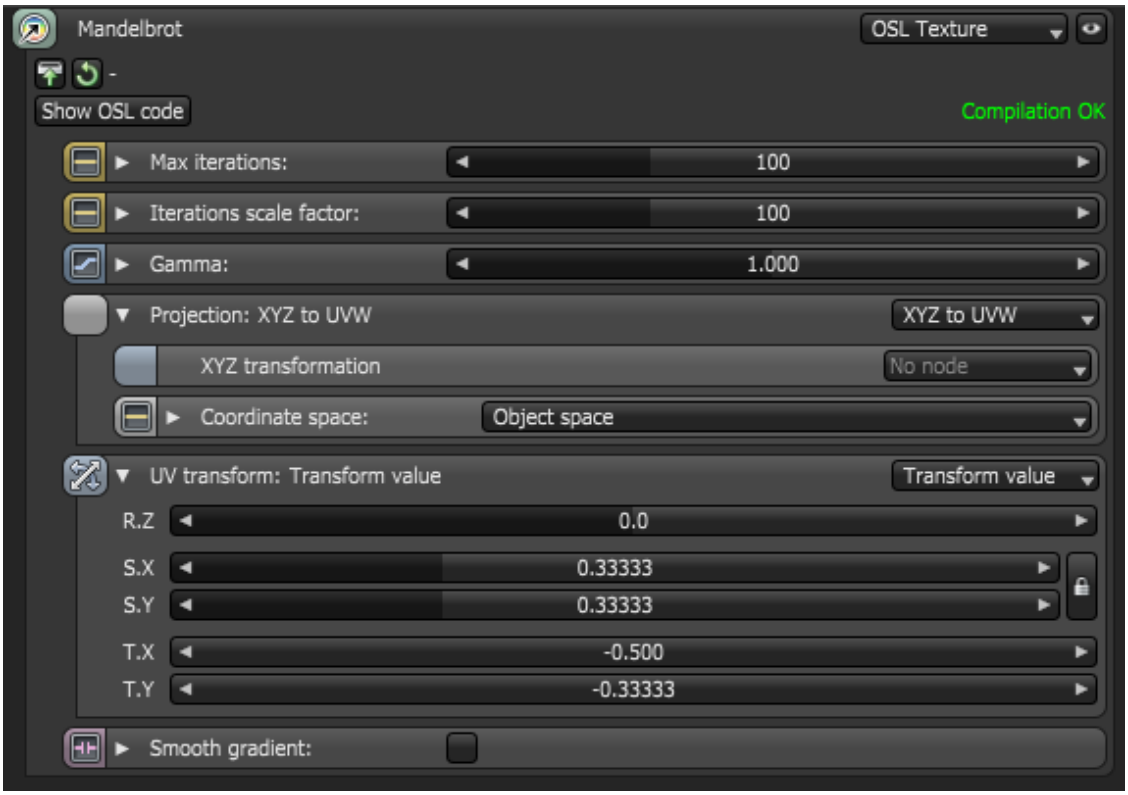
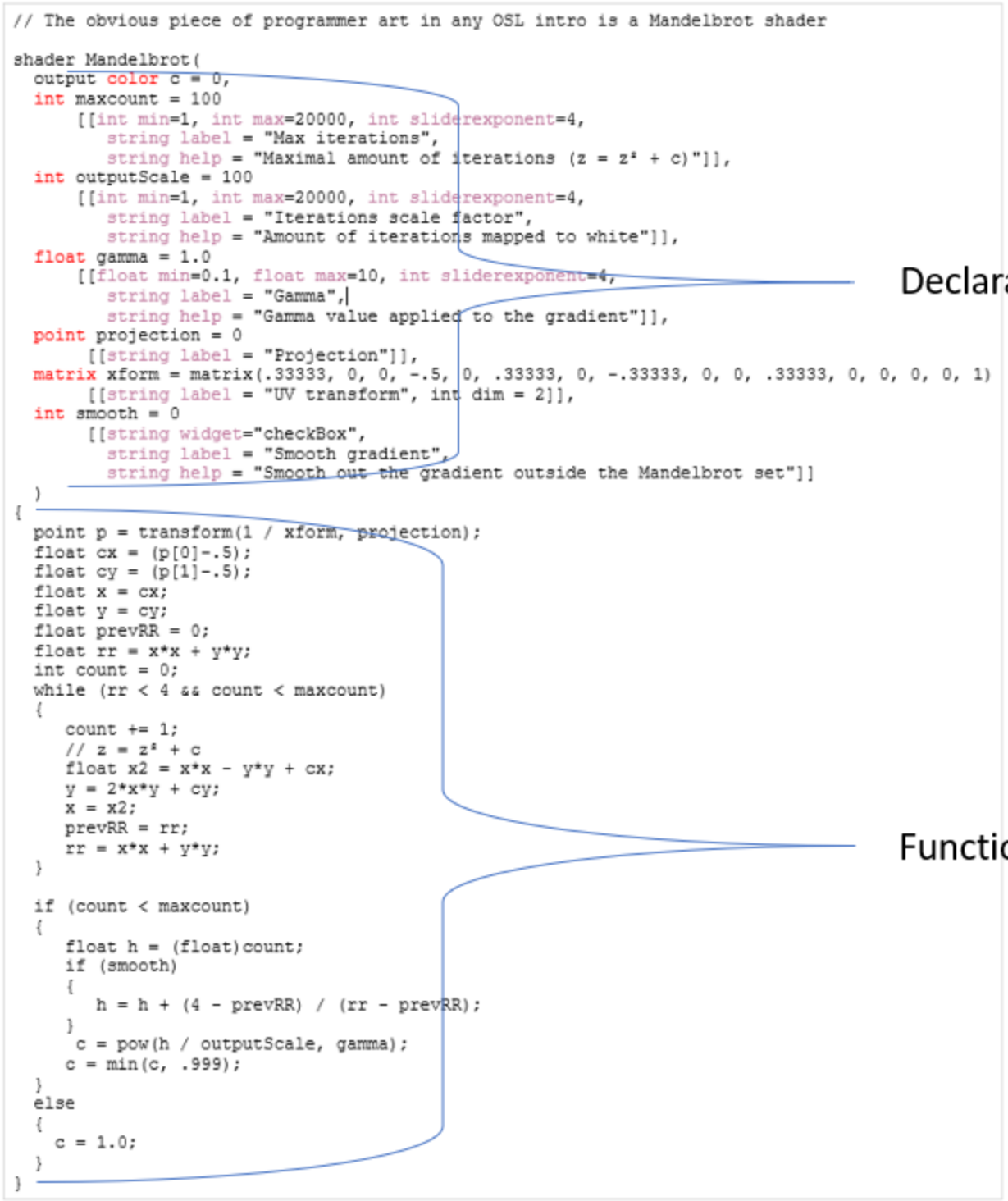


Figure 4: The Mandelbrot Shader implemented via an Octane OSL Texture node




```
// The obvious piece of programmer art in any OSL intro is a Mandelbrot shader

shader Mandelbrot(
  output color c = 0,
  int maxcount = 100
      [[int min=1, int max=20000, int sliderexponent=4,
         string label = "Max iterations",
         string help = "Maximal amount of iterations (z = z² + c)"]],
  int outputScale = 100
      [[int min=1, int max=20000, int sliderexponent=4,
         string label = "Iterations scale factor",
         string help = "Amount of iterations mapped to white"]],
  float gamma = 1.0
      [[float min=0.1, float max=10, int sliderexponent=4,
         string label = "Gamma",
         string help = "Gamma value applied to the gradient"]],
  point projection = 0
       [[string label = "Projection"]],
  matrix xform = matrix(.33333, 0, 0, -.5, 0, .33333, 0, -.33333, 0, 0, .33333, 0, 0, 0, 0, 1)
       [[string label = "UV transform", int dim = 2]],
  int smooth = 0
       [[string widget="checkBox",
         string label = "Smooth gradient",
         string help = "Smooth out the gradient outside the Mandelbrot set"]]
  )
{
  point p = transform(1 / xform, projection);
  float cx = (p[0]-.5);
  float cy = (p[1]-.5);
  float x = cx;
  float y = cy;
  float prevRR = 0;
  float rr = x*x + y*y;
  int count = 0;
  while (rr < 4 && count < maxcount)
  {
     count += 1;
     // z = z² + c
     float x2 = x*x - y*y + cx;
     y = 2*x*y + cy;
     x = x2;
     prevRR = rr;
     rr = x*x + y*y;
  }

  if (count < maxcount)
  {
     float h = (float)count;
     if (smooth)
     {
        h = h + (4 - prevRR) / (rr - prevRR);
     }
      c = pow(h / outputScale, gamma);
     c = min(c, .999);
  }
  else
  {
    c = 1.0;
  }
}
```

119 — Example of an OSL Script for an OSL Texture Node

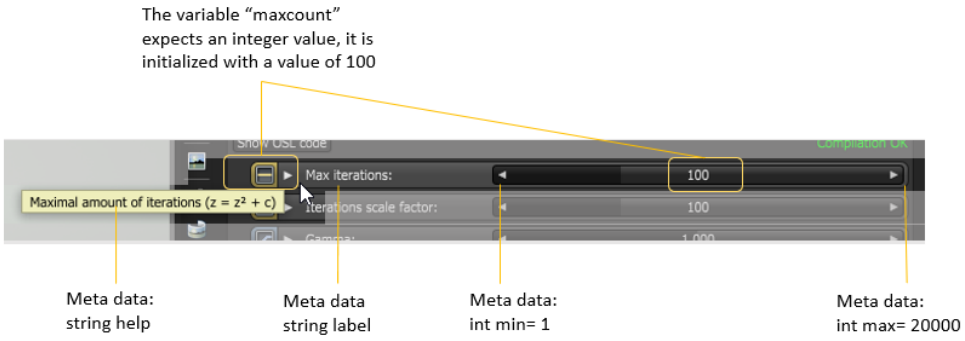
The Declaration Component

Line 1: This is a note for viewers of the code, it is ignored by the compiler.

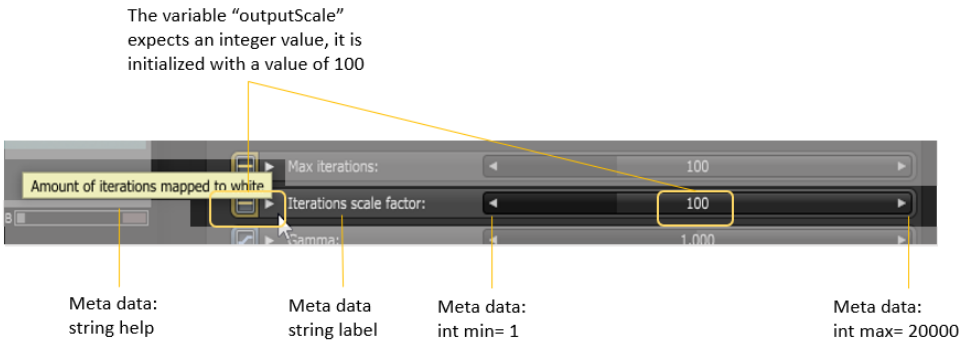
Line 3: This names the OSL Texture node

Line 4: An OSL Texture requires one output attribute, so this is provided by the declaration of the variable c which is of OSL **I/O type** "`color`" corresponding to an ***Octane Texture Attribute*** node.

Lines 5 - 8: This declares input of OSL **I/O type** "`int`" corresponding to an **Octane Int attribute** node (1D-value)



Lines 9 - 12: This declares input of OSL **I/O type** "`int`" corresponding to an **Octane Int attribute** node (1D-value)



The Function Body Component

Line 27: Gets a point. This implements the usual UV transform + projection inputs of Octane texture nodes.

Lines 28-34: A few more local variables to hold values needed in the function body. The variable P holds the pixel value which is taken from the transform matrix provided through Line 19.

Lines 35-44: These are the iterations that results in a fractal shape. Each iteration is checked

Lines 46-59 : This outputs the value 1.0 if the iteration doesn't diverge. This outputs < 1.0 if the iteration diverged at some point.

RGB Color

The **RGB Color** texture outputs the color specified in the color parameters (Figure 1).

Figure 1: The RGB Color texture parameters in the Node Inspector

Image-Based Textures

Image-based textures can be used by themselves or in combination with the **Mapping** textures to create surfaces. There are four image-based textures in OctaneRender:

- "Alpha Image" on the next page
- "Greyscale Image" on page 130
- "Image Tiles Texture" on page 126
- "RGB Image" on page 132 and "Animated Image Textures" on page 134

Alpha Image

The **Alpha Image** texture utilizes the image's native alpha channel to provide transparency. This texture only accepts PNG, TIF, and **EXR**[1] images (Figure 1).

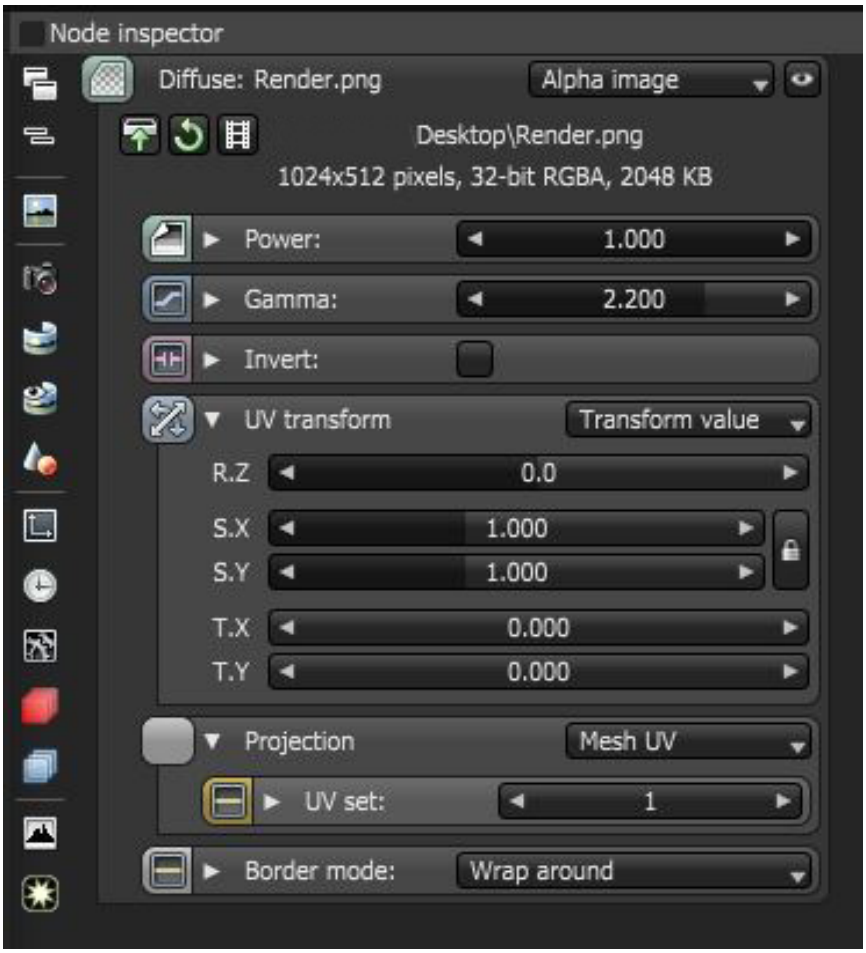


Figure 1: The Alpha Image parameters in the Node Inspector

[1]Also known as OpenEXR. This image file format was developed by Industrial Light & Magic and provides a High Dynamic Range image capable of storing deep image data on a frame-by-frame basis.

Alpha Image Parameters

- **Power** - Controls image brightness. Lowering the value makes the image look darker.
- **Gamma**[1] - Controls input image luminance, and tunes or color-corrects images if needed.
- **Invert** - Inverts the texture values.
- **UV Transform** - Positions, rotates, and scales the surface texture.
- **Projection** - Accepts OctaneRender® **Projection** nodes. If nothing is connected to this input, the Image texture uses the surface's **UV** texture coordinates by default. This also changes the UV set if the original surface contains more than one UV set. For more details, see the Octane **Projections**[2] section of this manual.
- **Border Mode** - Sets the behavior of the space around the image if it doesn't cover the entire geometry. **Wrap Around** is the default behavior, which repeats the image in the areas outside the image's coverage. If you set this parameter to **White Color** or **Black Color**, the area outside the image turns to white or black, respectively.

---

[1] The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.
[2] Methods for orienting 2D texture maps onto 3D surfaces.

Image Tiles Texture

This is used to set up a tile grid similar to UDIM image tiles.

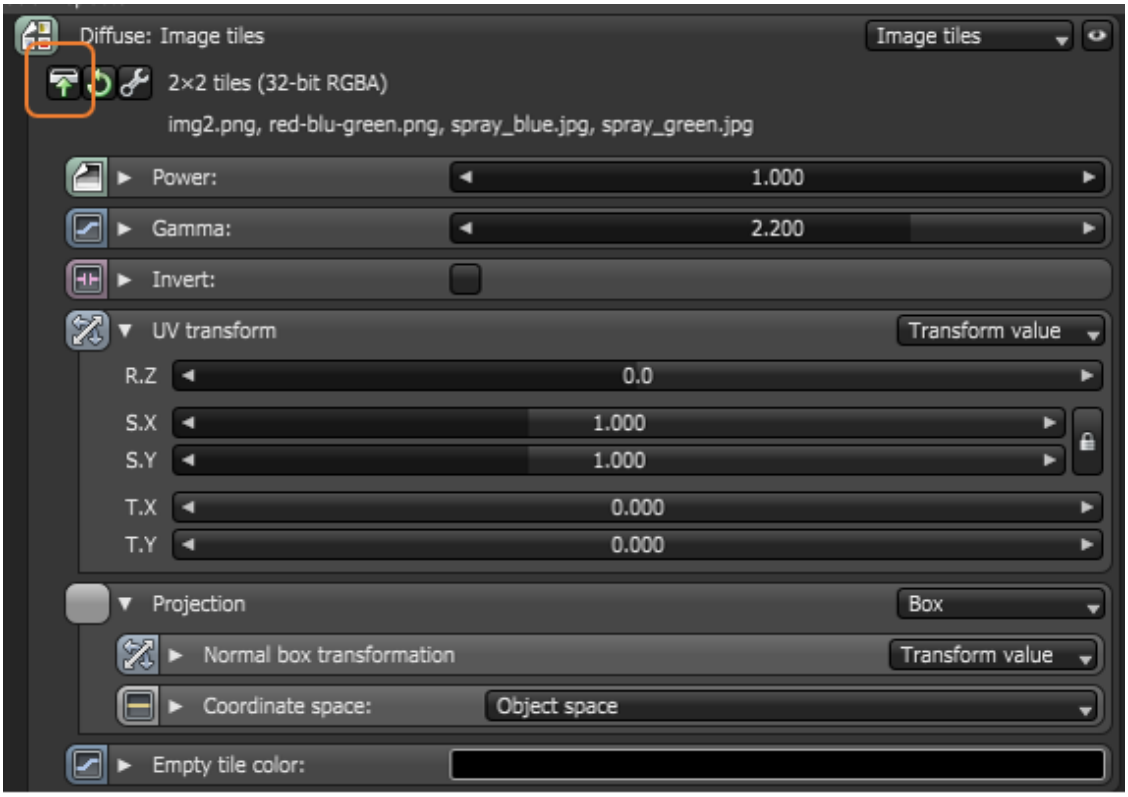
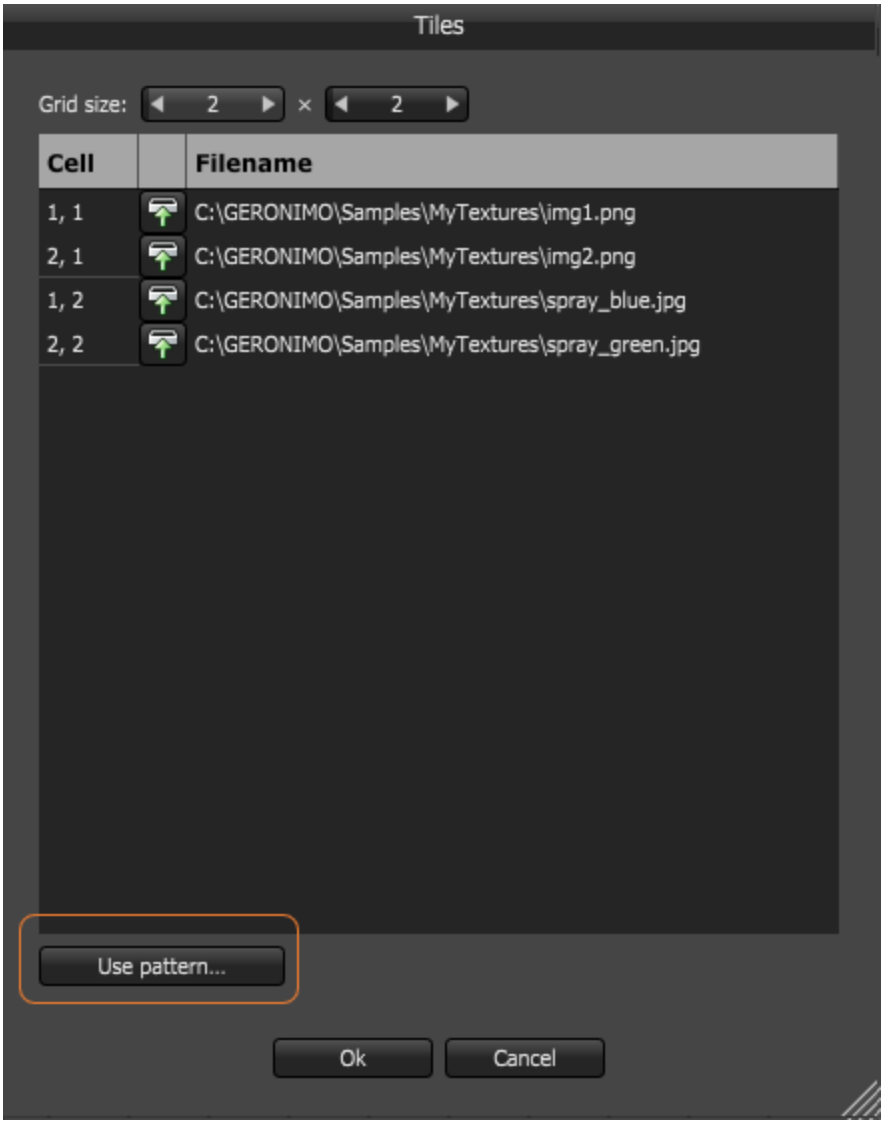


Figure 1: To begin creating the grid, click on the Load Images icon to select and load images into the grid


Tiles
Grid size: 2 × 2
Cell
Filename
1, 1
C:\GERONIMO\Samples\MyTextures\img1.png
2, 1
C:\GERONIMO\Samples\MyTextures\img2.png
1, 2
C:\GERONIMO\Samples\MyTextures\spray_blue.jpg
2, 2
C:\GERONIMO\Samples\MyTextures\spray_green.jpg
Use pattern...
Ok
Cancel

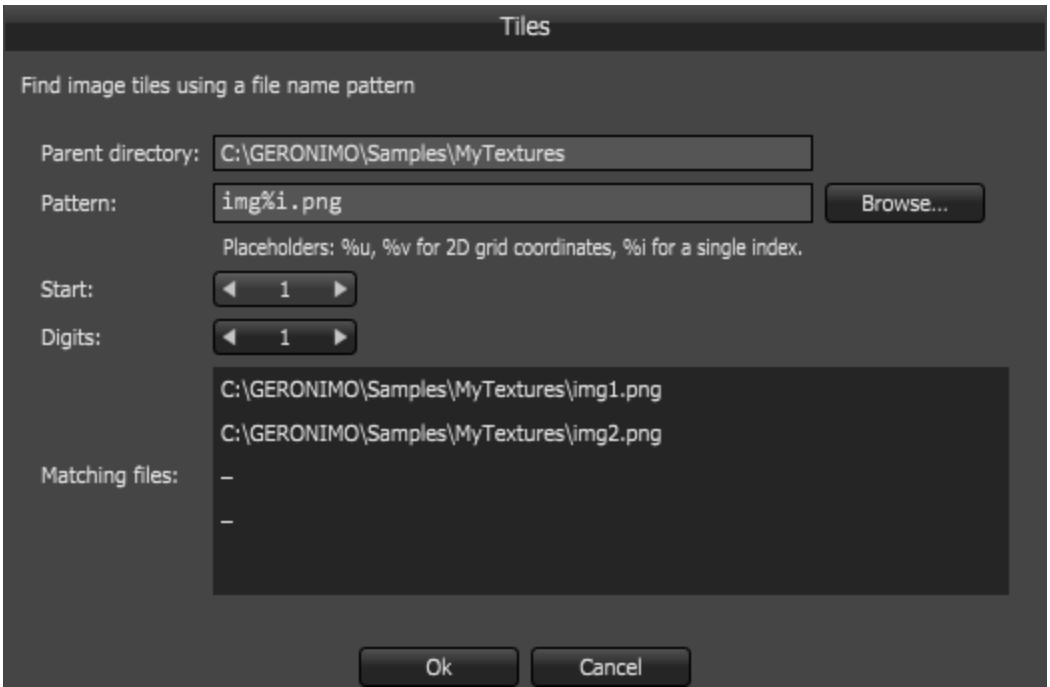

Tiles
Find image tiles using a file name pattern
Parent directory:
C:\GERONIMO\Samples\MyTextures
Pattern:
img%i.png
Browse...
Placeholders: %u, %v for 2D grid coordinates, %i for a single index.
Start:
1
Digits:
1
Matching files:
C:\GERONIMO\Samples\MyTextures\img1.png
C:\GERONIMO\Samples\MyTextures\img2.png
–
–
Ok
Cancel

Figure 2: Use pattern is an option to use a filename pattern to quickly select files

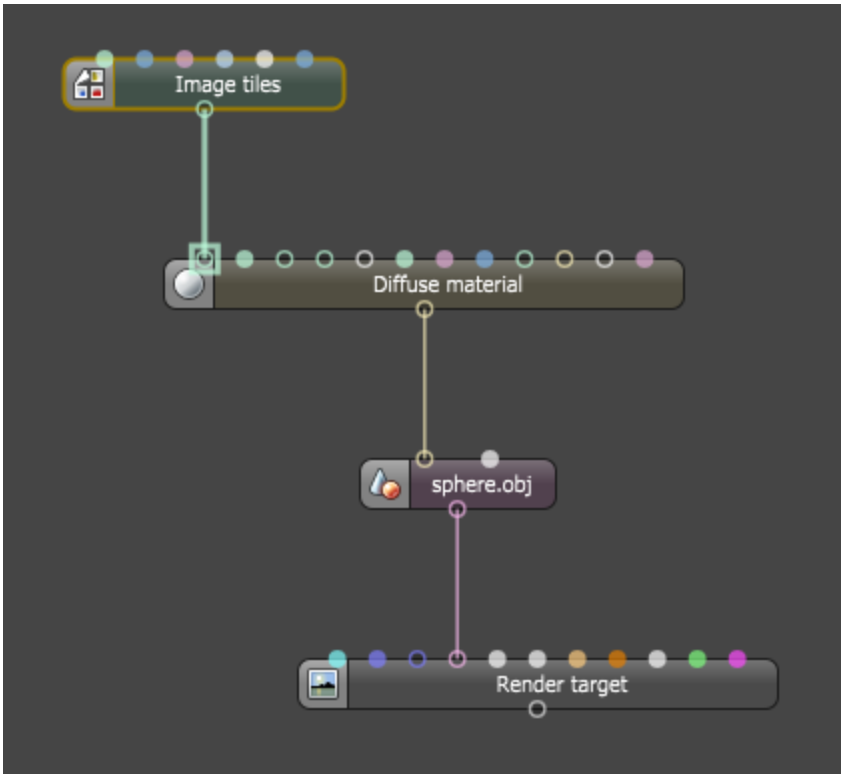


Figure 3: A basic scene graph using Image Tiles texture

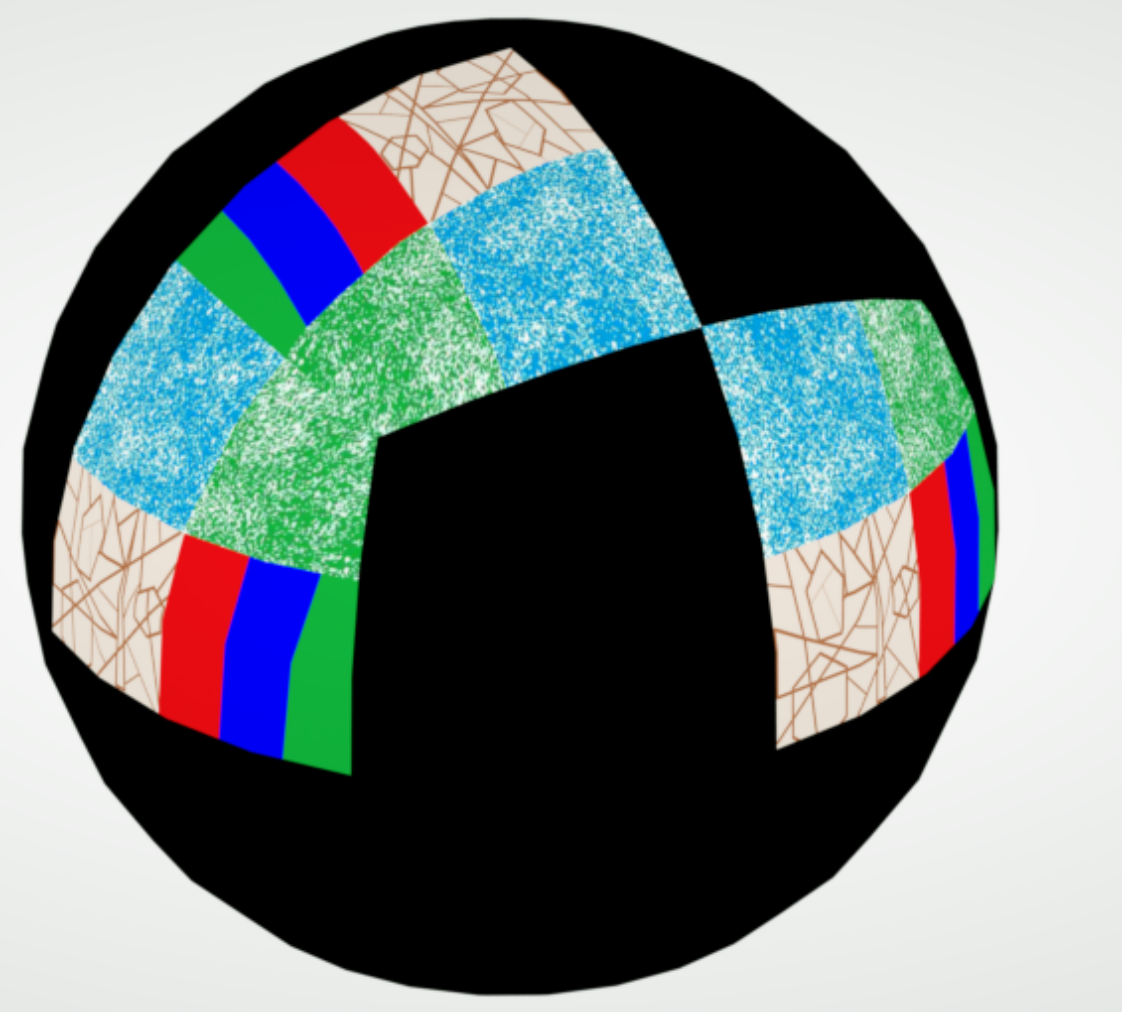


Figure 4: Image Tiles texture with four tiled images applied on a sphere without a mesh UV

Greyscale Image

The **Greyscale Image** converts an RGB image to grayscale. This can conserve RAM when using a color image as an input for **Bump** or **Opacity** channels of an OctaneRender® material. The **Invert** checkbox inverts the image (useful for **Bump** and **Opacity** maps). Figure 1 shows the parameters for the Greyscale texture.

Figure 1: Greyscale image parameters in the Node Inspector

The **Channel Format** (Figure 2) indicates the preferred channel format for loading the image. This is ignored for 8-bit images. This also selects the texture bit depth of High Dynamic Resolution (HDR) images in **Environment** textures.

Figure 2: The Import Settings shortcut

Greyscale Image Parameters

- **Power**- Controls image brightness. Lower values cause the image to appear darker. When used as a Bump map, this setting alters the bump height on the surface.
- **Gamma**[1] - Controls the input image luminance, and it also tunes or color-corrects the image.
- **Invert** - Inverts the texture values.
- **Transform** - Positions, rotates, and scales the surface texture.
- **Projection** - Accepts OctaneRender **Projection** nodes. If nothing is connected to this input, the Image texture uses the surface's **UV** texture coordinates by default. This also changes the UV set if the original surface contains more than one UV set. For more details, see the OctaneRender Projection Node section of this manual.
- **Border Mode** - Sets the behavior of the space around the image if it doesn't cover the entire geometry. **Wrap Around** is the default behavior, which repeats the image in the areas outside the image's coverage. If you set this parameter to **White Color** or **Black Color**, the area outside the image turns to white or black, respectively.

[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

RGB Image

RGB Image textures connect an external image file to any material parameters that accept a **Texture** map (Figure 1).

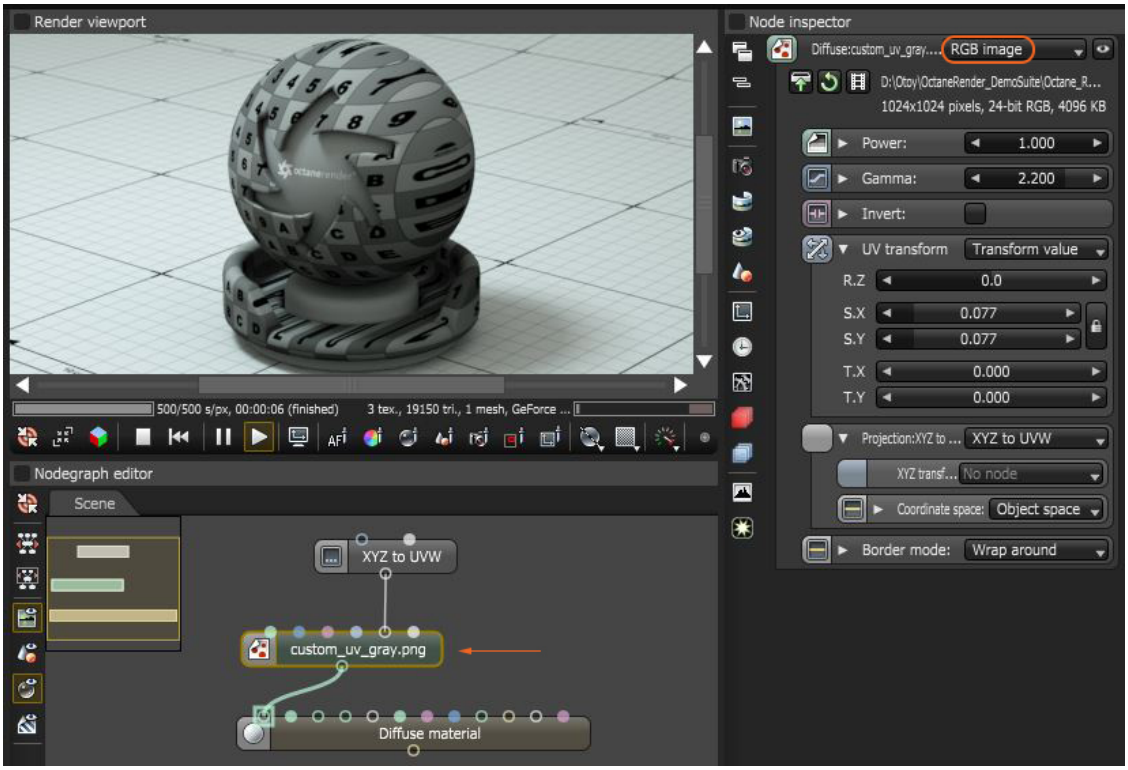


Figure 1: The RGB Image node is used to import a PNG image into the Diffuse[1] pin of a Diffuse material[2]

To create an icon of the image, click on the eyeball icon in the upper-right corner of the **Node inspector**. The drawer icon replaces the image, the circular arrow reloads the image, and the film strip icon loads an animated sequence.

---

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]Used for dull, non-reflecting materials or mesh emitters.

The RGB Image texture converts all images to three-channel images, including greyscale images. To use memory resources efficiently, only use the RGB Image texture for color inputs. For Greyscale channels such as **Bump**, use the **Greyscale Image** texture.

Use the **Import settings** button (Figure 2) to indicate the preferred channel format for loading the image. This is ignored for 8-bit images. You can also use this for explicitly selecting the texture bit depth of High Dynamic Resolution images in **Environment** textures.

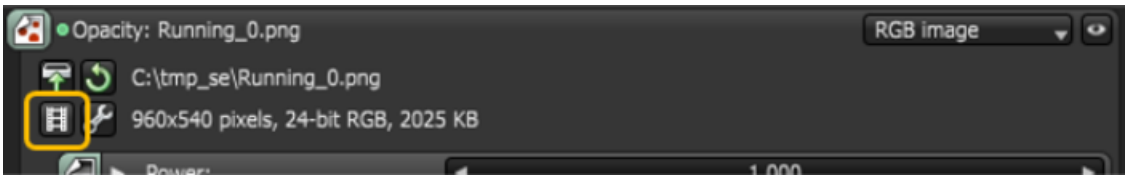
Figure 2: The import settings shortcut

- **Power** - Controls image brightness. Lowering the value makes the image look darker.
- **Gamma**[1] - Controls input image luminance, and tunes or color-corrects images if needed.
- **Transform** - Positions, rotates, and scales the surface texture.
- **Invert** - Inverts the image's color values.
- **Projection** - Accepts OctaneRender® **Projection** nodes. If nothing is connected to this input, the Image texture uses the surface's **UV** texture coordinates by default. This also changes the UV set if the original surface contains more than one UV set. For more details, see the Octane Projection Node section of this manual.
- **Border Mode** - Sets the behavior of the space around the image if it doesn't cover the entire geometry. **Wrap Around** is the default behavior, which repeats the image in the areas outside the image's coverage. If you set this parameter to **White Color** or **Black Color**, the area outside the image turns to white or black, respectively.

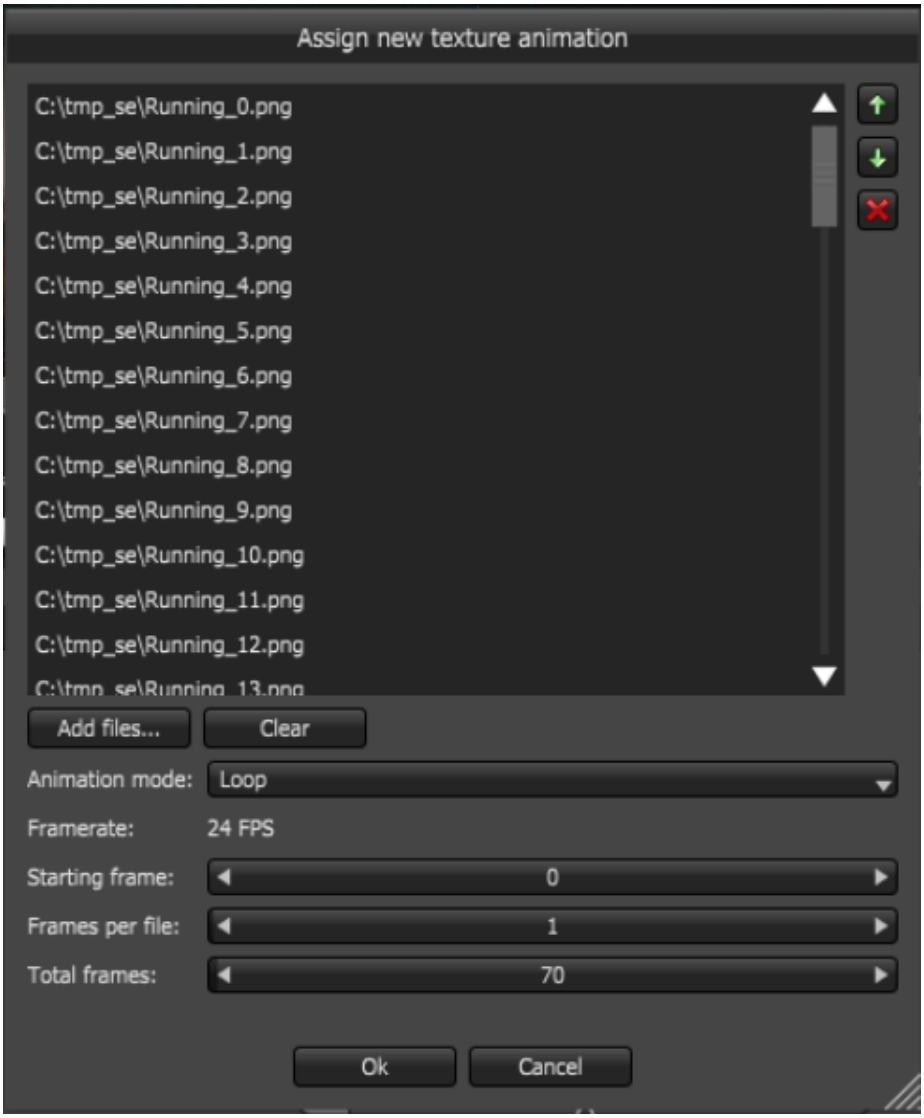
[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

Animated Image Textures

OctaneRender® implements animated image textures by animating the file name attribute in the **Image** texture nodes. To set it up, click the **Animation** button in the **Node Inspector** pane.



This opens the **Assign New Texture Animation** window. From there, you can add the sequence of image files by clicking on the **Add Files** button.



To specify how the animation runs through the file sequence, you can specify an **Animation Mode**. The following modes are available:

- **Once** - Runs through the sequence once.
- **Loop** - Runs through the sequence and repeats it indefinitely.
- **Ping-Pong** - Runs through the sequence from beginning to end, then from the end back to the beginning, and repeats the sequence indefinitely.

To control the duration and speed of the animation sequence, you can adjust the following parameters:

- **Frames Per File** - Sets the number of frames to display each image of the sequence. The **Framerate** is defined in the time slider of the **Render viewport** - i.e., it comes from the project. It's displayed in the dialog just for convenience.
- **Total Frames** - Adjusts the total number of frames to play the animation sequence.

When you save a project as a package, all of the specified images in the sequence are stored in the package, too. After opening this package, you can still remove an image from the sequence and change its order, but you cannot add new files from the file system. This is a limitation of the animated file name attribution (i.e., this avoids having files coming from multiple packages or from the file system in the same sequence).

Procedural Textures

Procedural Textures[1] are a category of textures used to generate patterns that can be used by themselves or in combination with the **Mapping** and **Color** textures to create common surface effects. This is a very memory efficient way to generate patterns as the calculations require less memory than loading bitmap images. Procedurals can be used to create rock and marble surfaces, chequered or wooden textures, bump maps and other advanced materials with minimal impact to **GPU**[2] memory. It is advantageous to explore creating materials using these textures before resorting to image-based textures:

- **Checks**
- **Marble**
- **Noise**
- **Ridged Fractal**
- **Saw Wave**
- **Sine Wave**
- **Triangle Wave**
- **Turbulence**

[1] Textures are used to add details to a surface. Textures can be procedural or imported raster files.

[2] The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Checks

The **Checks** Procedural texture is useful for making stripes, checker board and grid patterns. The parameters consist of **UVW transform** and **Projection** inputs (Figure 1).

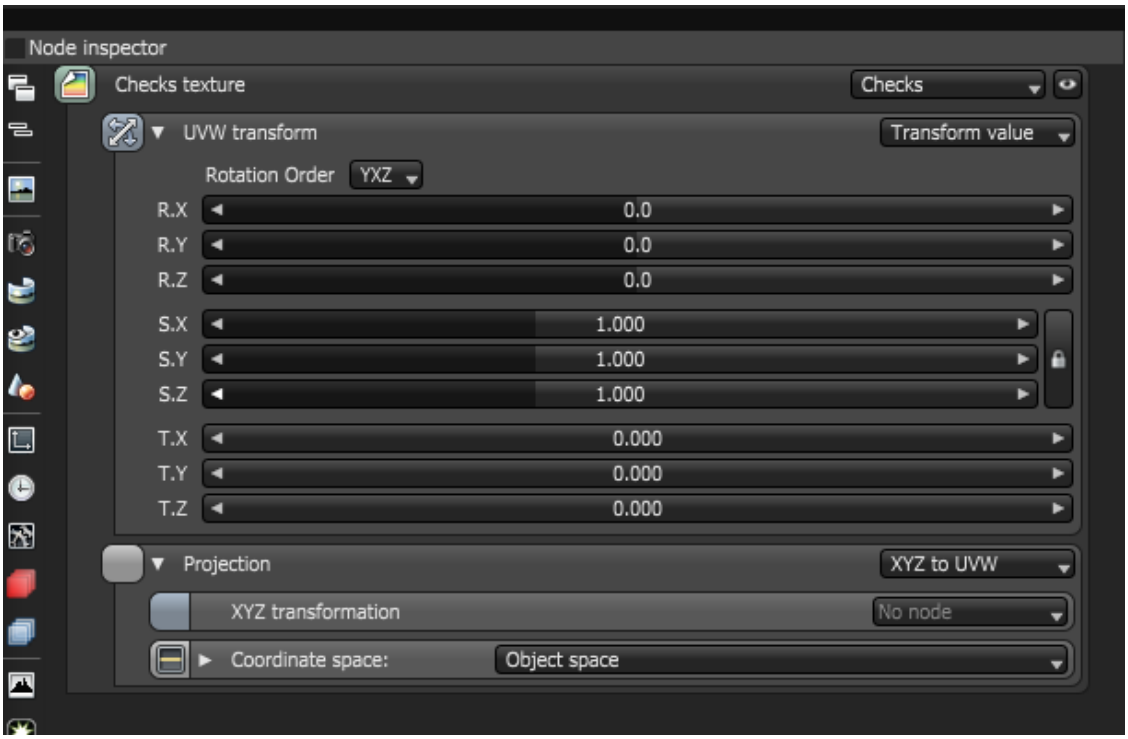


Figure 1: The parameters of the Checks texture

You can use the UVW Transform and Projection parameters to edit the pattern's position and scale (Figure 2).

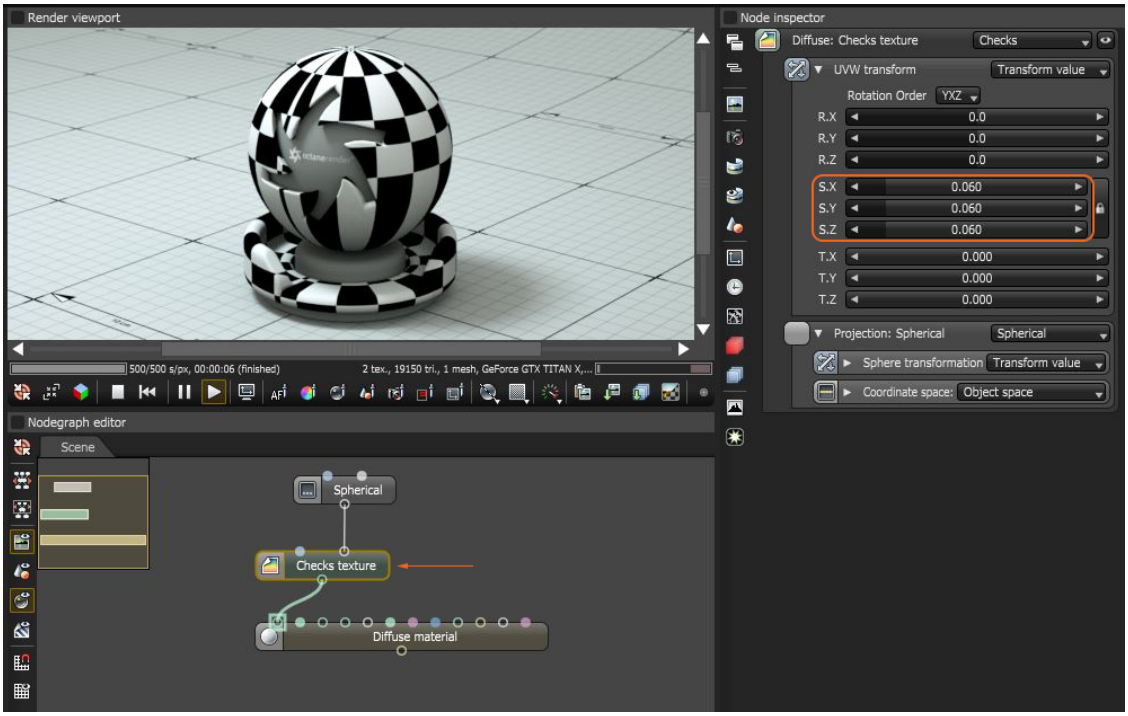


Figure 2: Checks example with X,Y,Z scaled in the UVW Transform parameter

Marble

The **Marble** node is a **Procedural** texture that can create marble-like noise. It is similar to a **Turbulence** texture, but more fine-tuned to create marble-like patterns. Figure 1 shows a Marble texture connected to the **Diffuse**[1] channel of an OctaneRender® material.

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

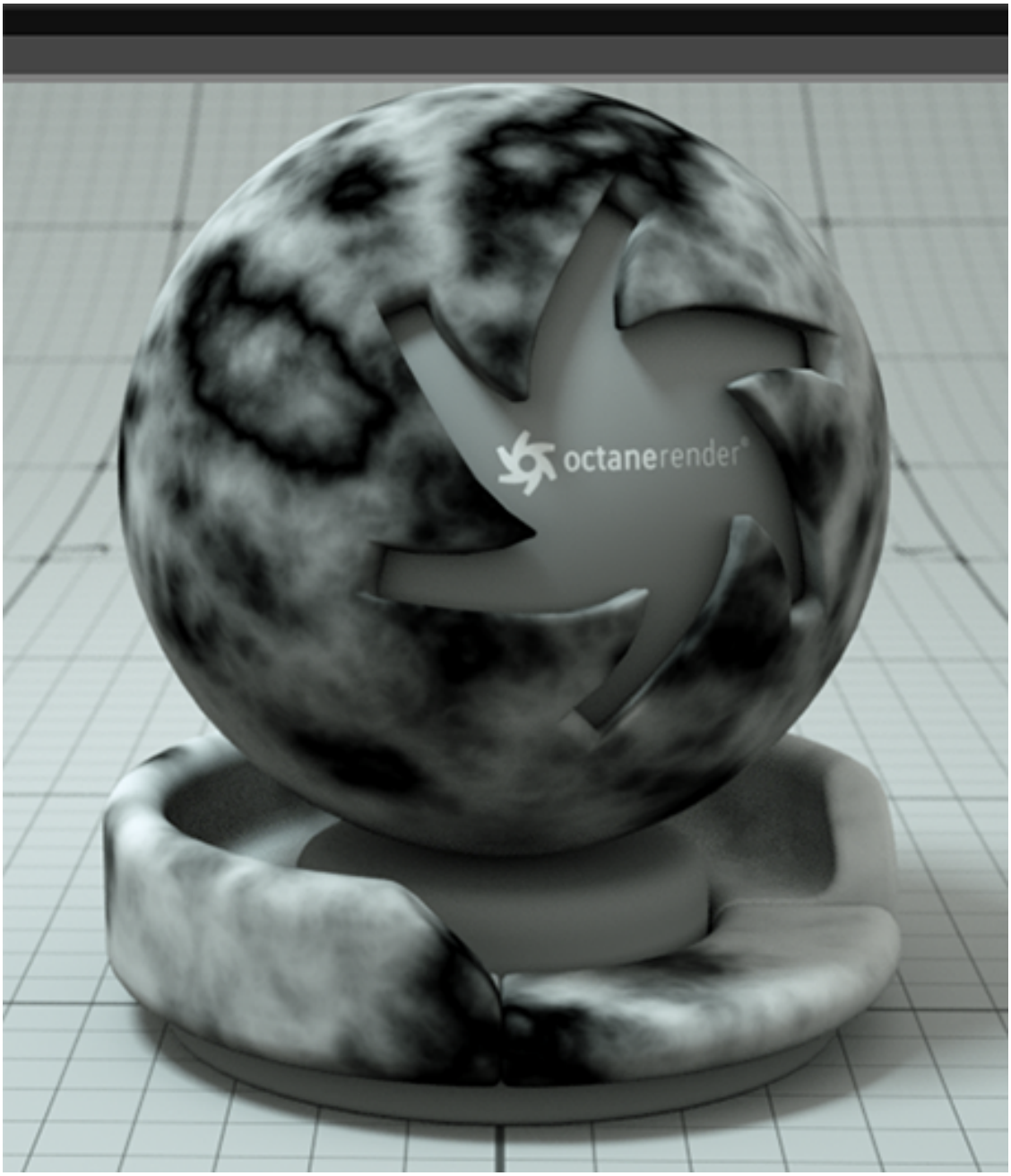


Figure 1: The Marble texture creates a Procedural noise pattern

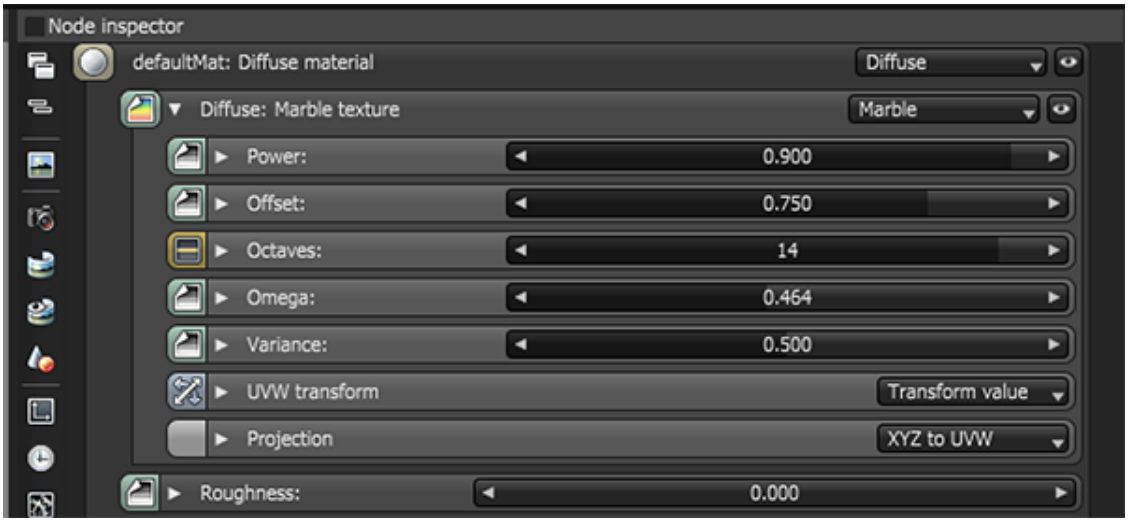


Figure 2: The Marble texture parameters

Marble Texture Parameters

- **Power** - A multiplier that controls the texture's overall brightness.
- **Offset** - Sets the texture position in 3D space.
- **Omega** - Controls detail in the underlying fractal pattern.
- **Variance** - Randomizes the Marble pattern.
- **UVW Transform** - Positions, scales, and rotates the surface texture.
- **Projection** - Sets how the texture projects onto the surface.

Noise

The **Noise** texture generates four different types of **Procedural** noise, and the settings give you the ability to produce a wide variety of noise effects. The four types of noise are:

- **Perlin**
- **Turbulence**
- **Circular**
- **Chips**

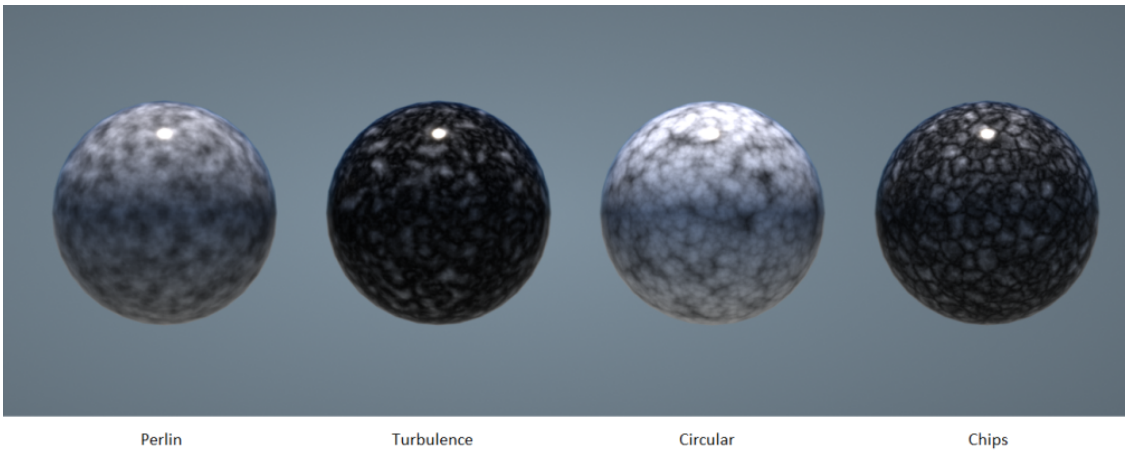


Figure 1: A comparison of the four different noise types

Figure 2 shows the **Noise** texture connected to the **Diffuse**[1] channel of a **Diffuse** material. Figure 3 shows the settings that generate this noise pattern.

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

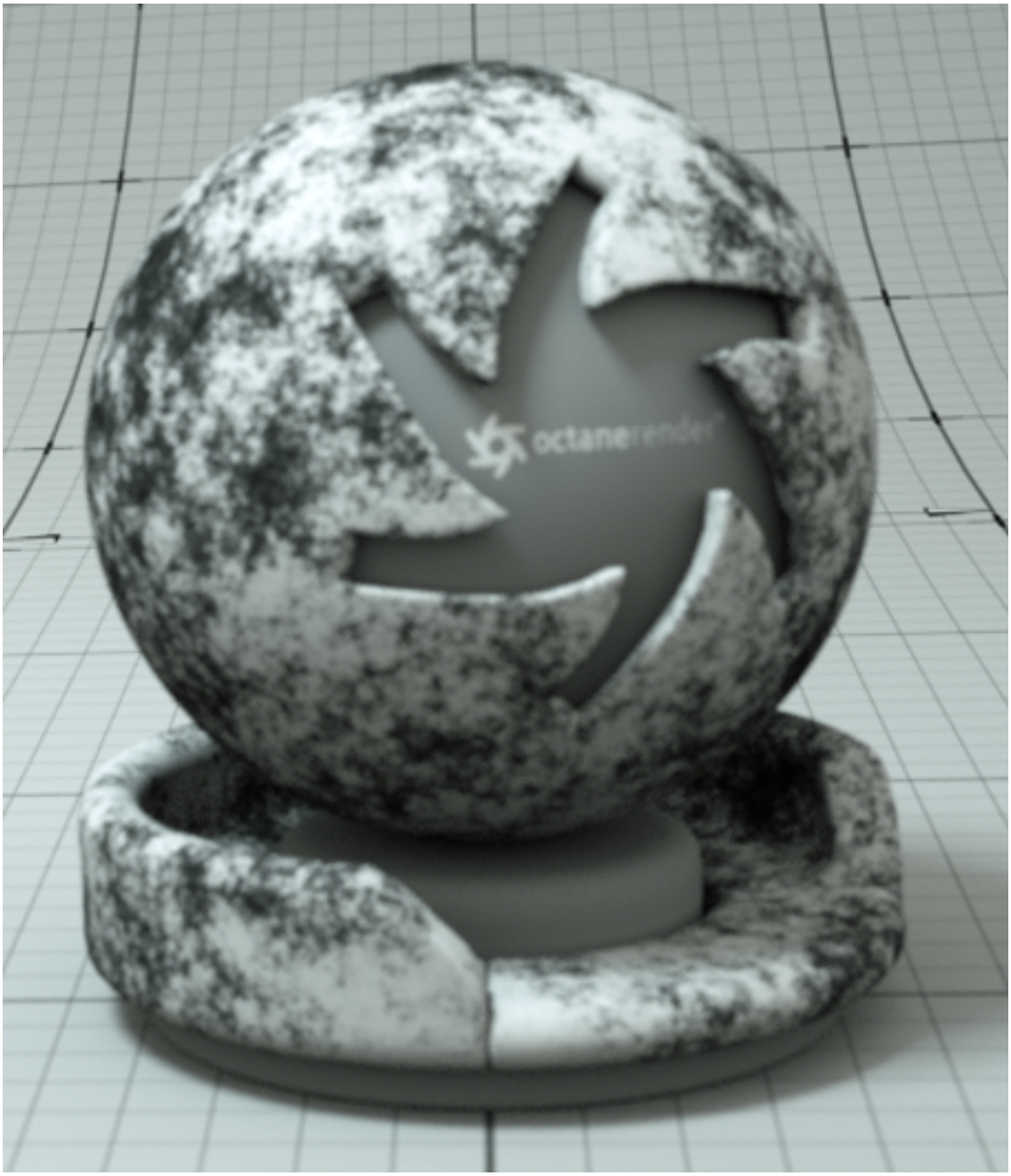


Figure 2

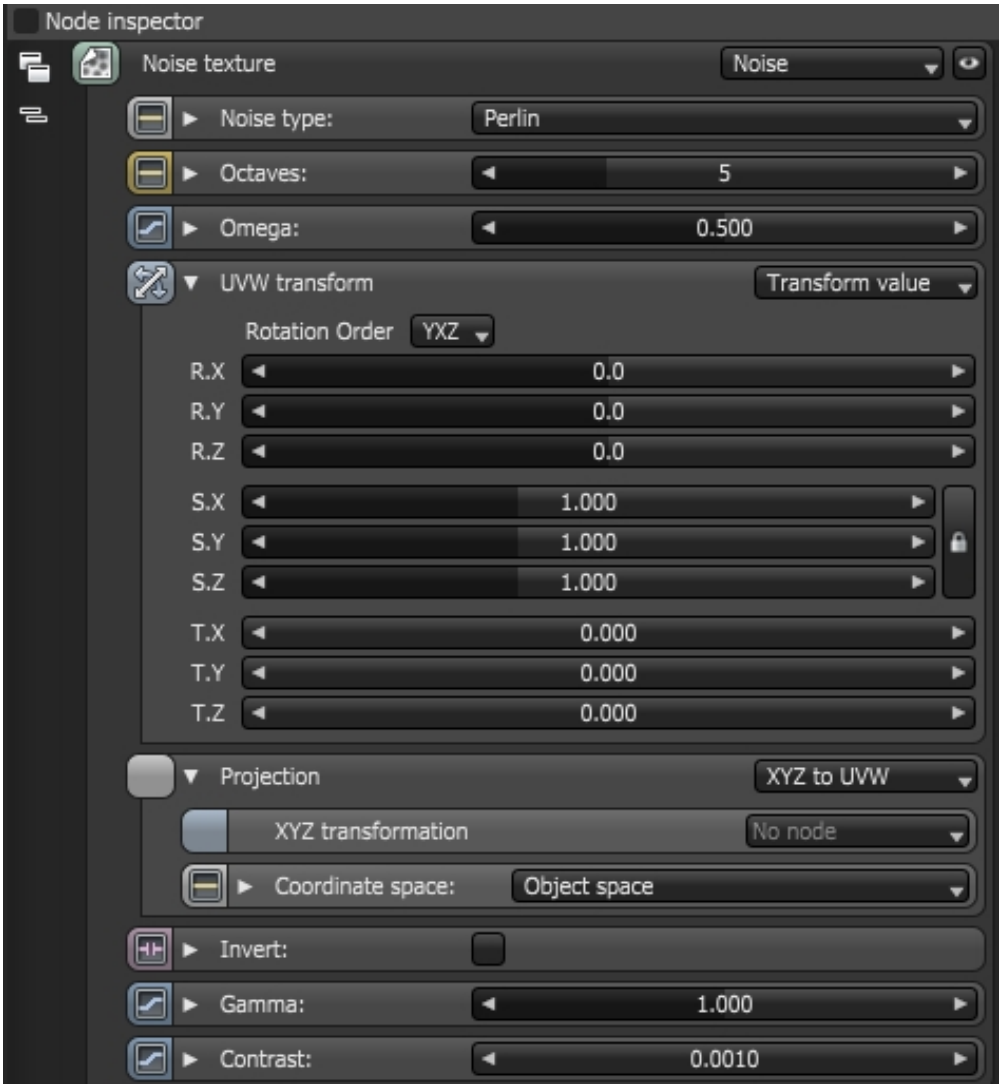


Figure 3: The Noise parameters in the Node Inspector

Noise Texture Parameters

- **Noise Type** - Select from four different noise generators.
- **Octaves** - Sets the noise detail's scale.
- **Omega** - Controls the fractal pattern detail.
- **UVW Transform** - Positions, scales, and rotates the surface texture.

- **Projection** - Sets how the texture projects onto the surface.
- **Invert** - Inverts the Noise texture values.
- **Gamma**[1] - Adjust the Noise texture's luminance values.
- **Contrast** - Adjusts the Noise detail sharpness.

[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

Ridged Fractal

The **Ridged Fractal** node produces a fractal pattern in grayscale format. In Figure 1, the **Ridged Fractal** node has been connected to a **Diffuse material**[1]'s **Diffuse**[2] pin. Note that the UVW Transform scale (S.X, S.Y, S.Z) values have been significantly lowered for the pattern to emerge on the surface.

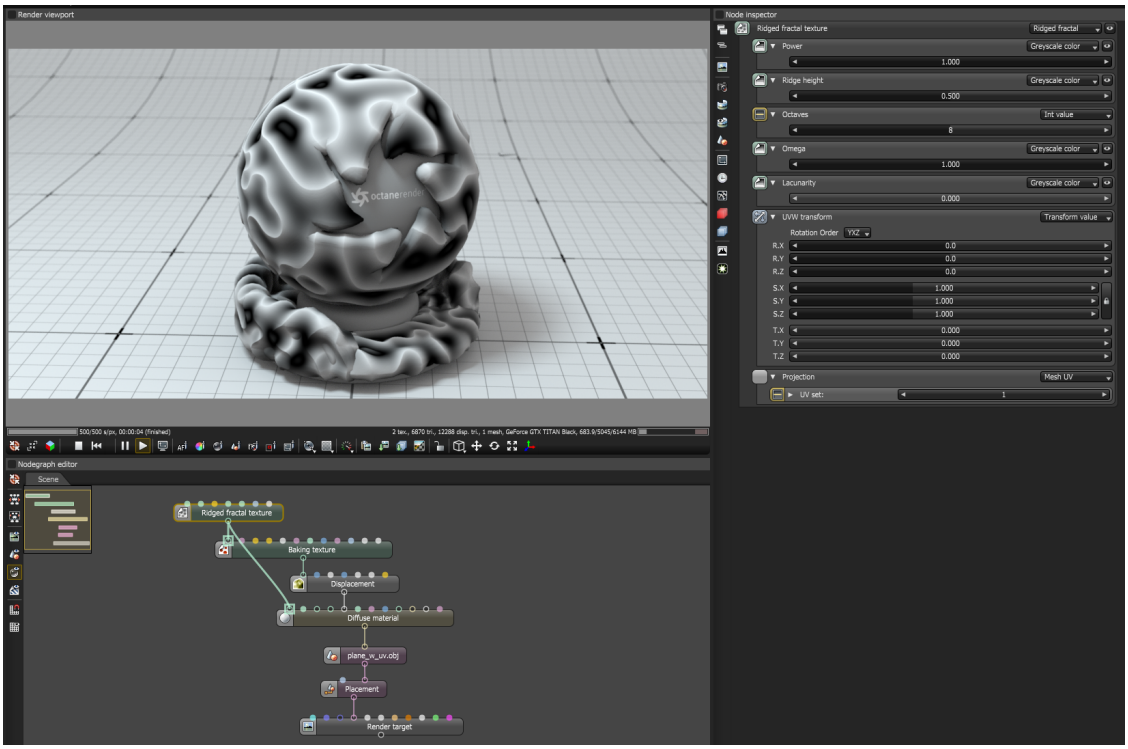


Figure 1: A Ridged Fractal texture is connected to a Diffuse material, and also used as input for a Baking texture for Procedural displacement

Ridged Fractal Parameters

[1]Used for dull, non-reflecting materials or mesh emitters.

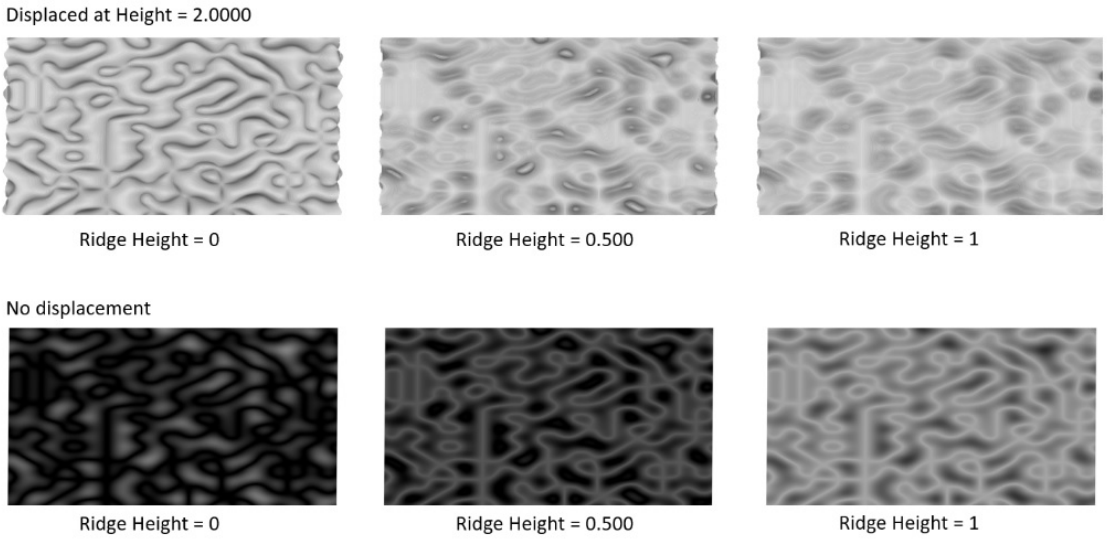
[2]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

Power

Controls the overall brightness of the texture.

Ridge Height

This specifies the height of the elevated parts of the fractal pattern.

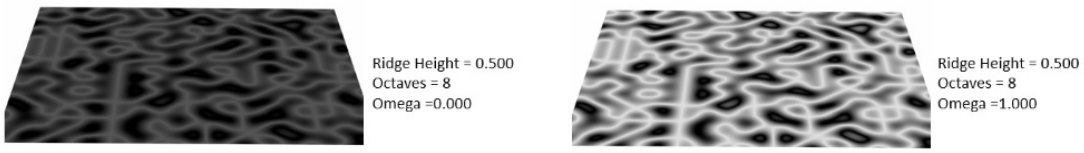


Octaves

Controls the amount of detail in the texture.

Omega

This specifies the difference per interval.



Lacunarity

Controls the size of the gaps in the fractal pattern.

UVW Transform

Controls position, scale, and rotation of the texture on the surface.

Projection

Determines how the texture is projected onto the surface.

Saw, Sine, Triangle Wave

The **Sine Wave**, **Saw Wave**, and **Triangle Wave** textures can create various banding or striped patterns. You can adjust the position, scale, and rotation of the patterns in the **UVW Transform** parameter. Figure 1 shows a comparison of the Sine, Saw, and Triangle Waves, respectively. The **Offset** parameter can shift the position of the surface pattern.

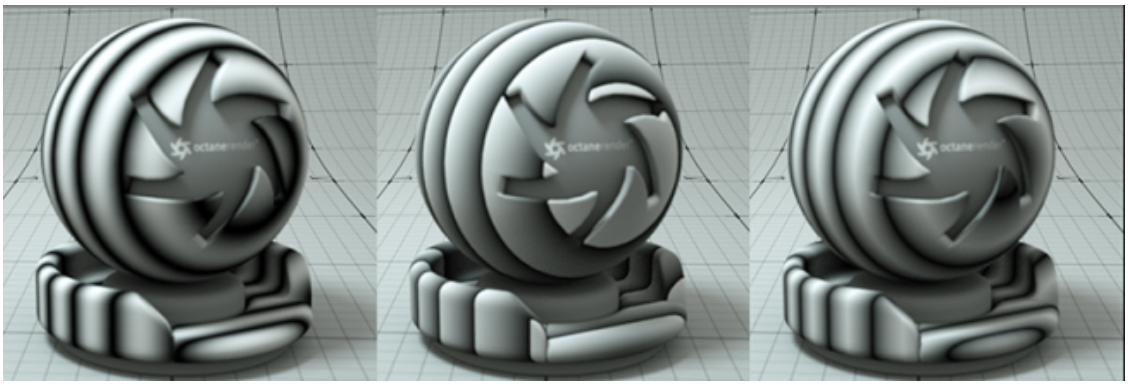


Figure 1: A comparison of the Sine, Saw, and Triangle Wave textures

You can create interesting effects by using one of the wave textures as an input for another **Procedural** texture. Figure 2 shows the result of using a Sine Wave as the input for the **Offset** of a **Turbulence** texture.

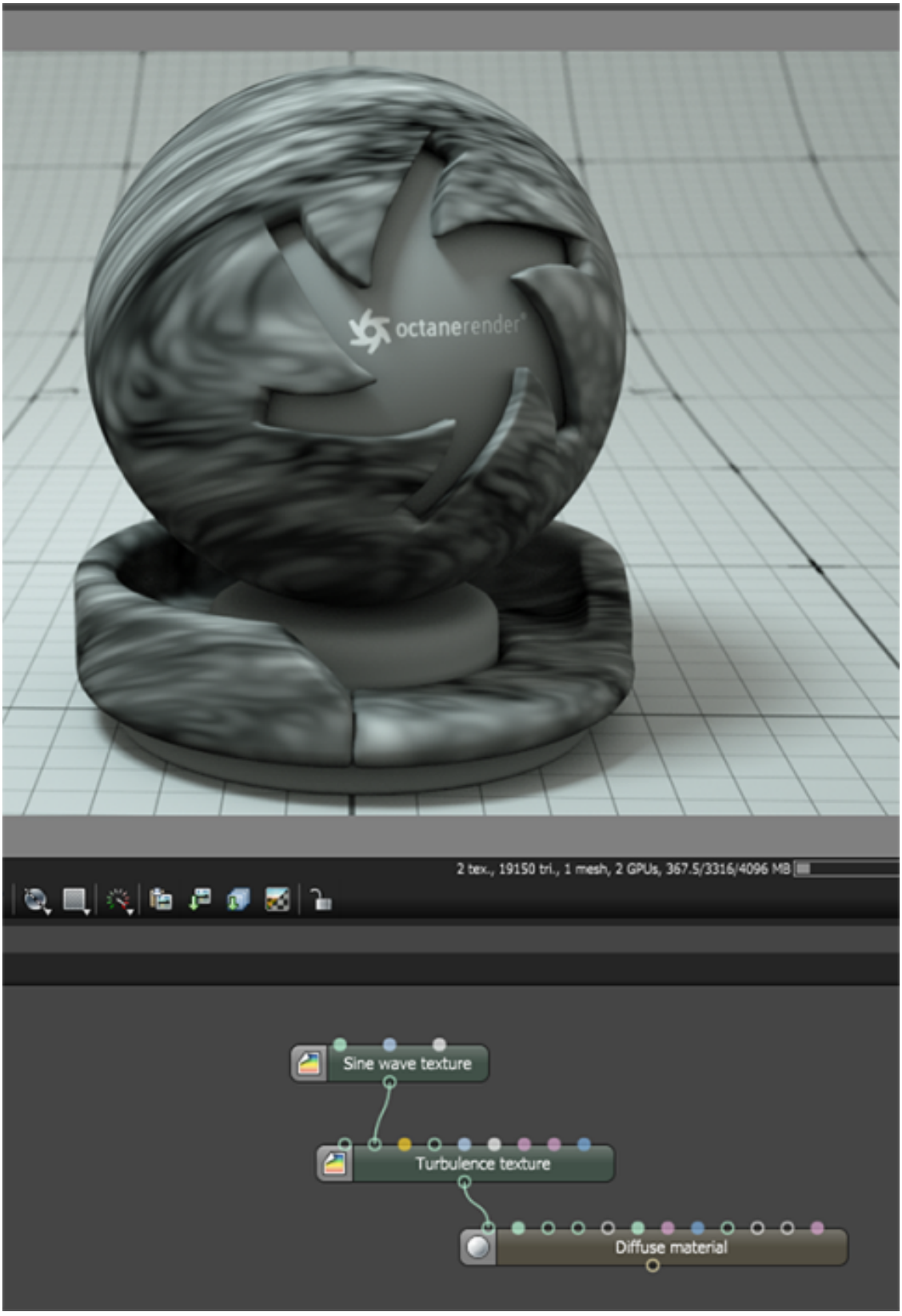
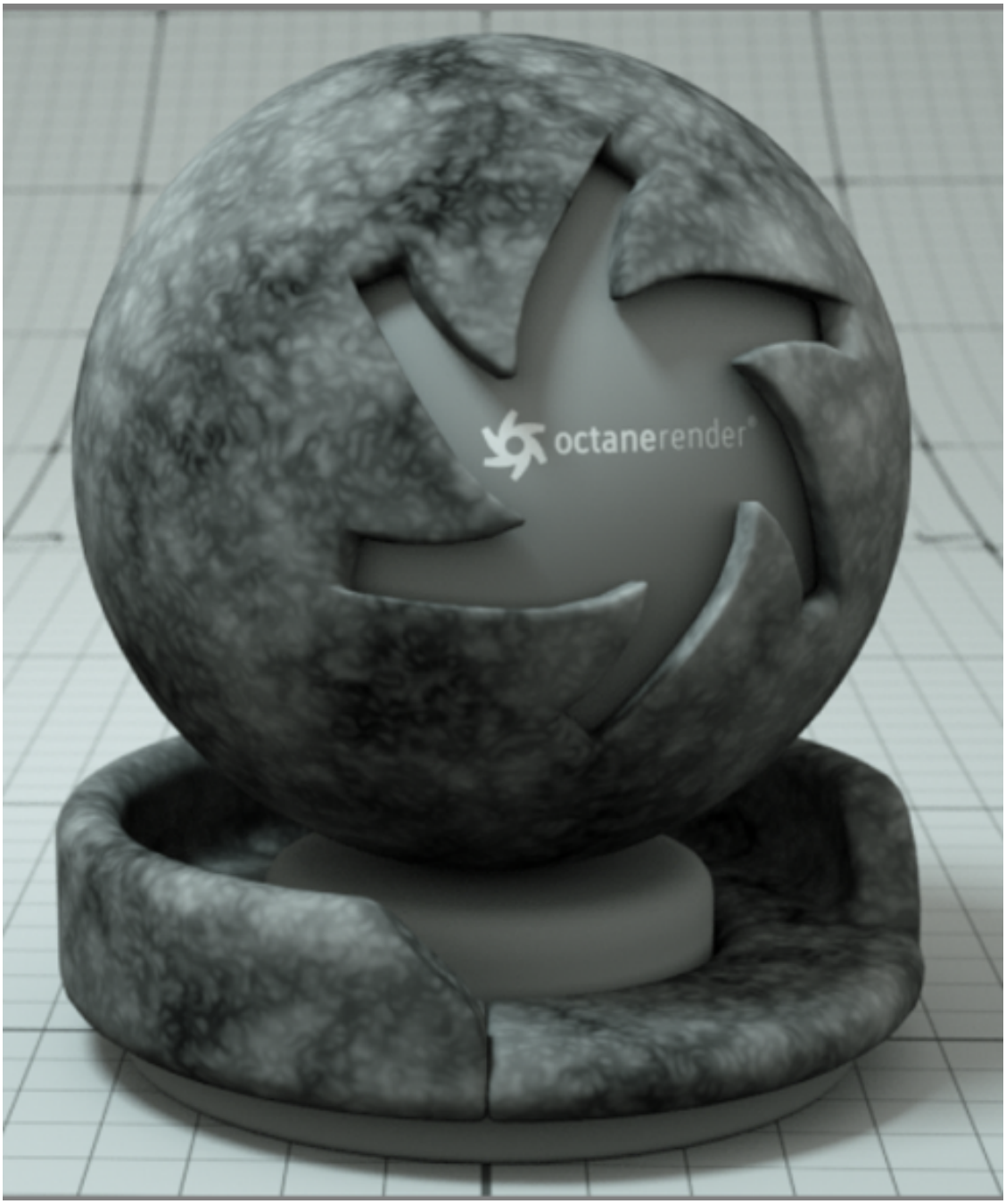

octanerender
2 tex., 19150 tri., 1 mesh, 2 GPUs, 367.5/3316/4096 MB
Sine wave texture
Turbulence texture
Diffuse material

Figure 2 : A Sine Wave controls the Offset of a Turbulence texture

Turbulence

The **Turbulence** texture generates a **Procedural** noise texture that has a different quality than the **Noise** texture. Figure 1 shows the Turbulence texture connected to a **Diffuse**[1] material's **Diffuse** channel.

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.


octanerender

Turbulence Texture Parameters

- **Power** - Controls overall texture brightness.
- **Offset** - Shifts the Turbulence pattern in 3D space.
- **Octaves** - Sets the noise detail's scale.
- **Omega** - Controls fractal pattern detail.
- **UVW Transform** - Positions, scales, and rotates the surface texture.
- **Projection** - Determines how the texture projects onto the surface.
- **Use Turbulence** - Toggles the turbulent noise calculation, which multiplies against procedural noise.
- **Invert** - Inverts the Noise texture values.
- **Gamma**[1] - Adjusts the Noise texture's luminance values.

[1] The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

Geometric Textures

Geometric Textures[1] are a category of textures used to generate geometric patterns that can be used by themselves or in combination with the **Mapping** and **Color** textures to create surface effects. The patterns they produce are the result of geometric algorithms. This is a very memory efficient way to generate geometric patterns as the calculations require less memory than loading bitmap images. These are also used to create bump maps and other advanced materials with minimal impact to **GPU**[2] memory. It is therefore advantageous to explore creating materials using these textures before resorting to image based textures.

- Dirt Texture
- Falloff Map
- Instance Color
- Instance Range
- Polygon Side
- Random Color Texture
- W Coordinate

[1] Textures are used to add details to a surface. Textures can be procedural or imported raster files.

[2] The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Dirt Texture

The **Dirt** texture can create different shading effects based on ambient occlusion calculations. This texture is useful for simulating dirt, dust, or wear and tear, as well as blending textures based on the recesses of a surface.

Users often connect the Dirt texture to the **Diffuse**[1], **Bump**, or **Transmission**[2] inputs of an OctaneRender® material.

Figure 1 shows the result of connecting two **RGB Spectrum** textures to the inputs of a **Mix** texture, which is then connected to the **Diffuse** pin of a **Material**[3] node. The Dirt texture alters the Mix amount.

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2] A surface characteristic that determines if light may pass through a surface volume.
[3] The representation of the surface or volume properties of an object.

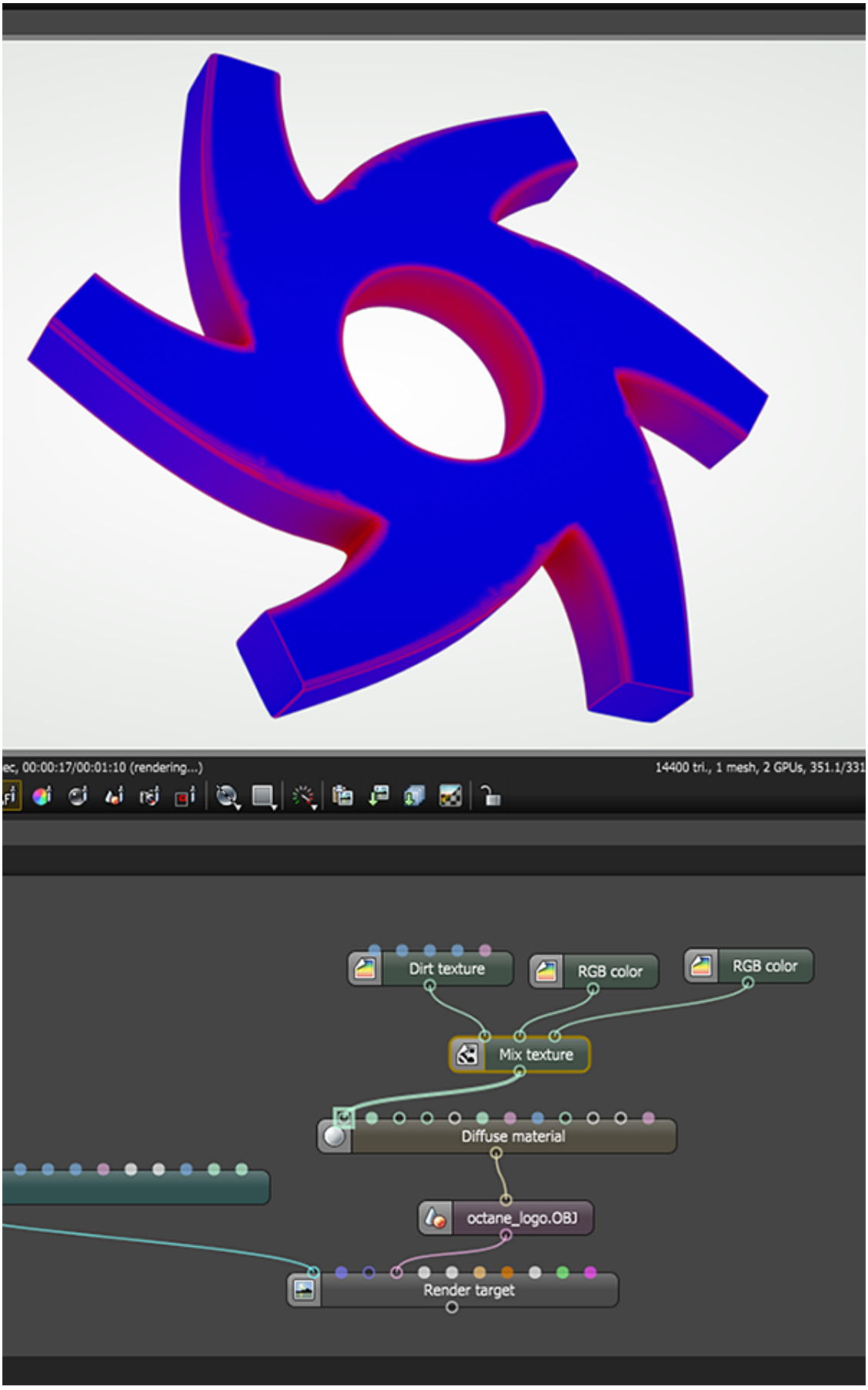

ec, 00:00:17/00:01:10 (rendering...)
14400 tri., 1 mesh, 2 GPUs, 351.1/331
Dirt texture
RGB color
RGB color
Mix texture
Diffuse material
octane_logo.OBJ
Render target

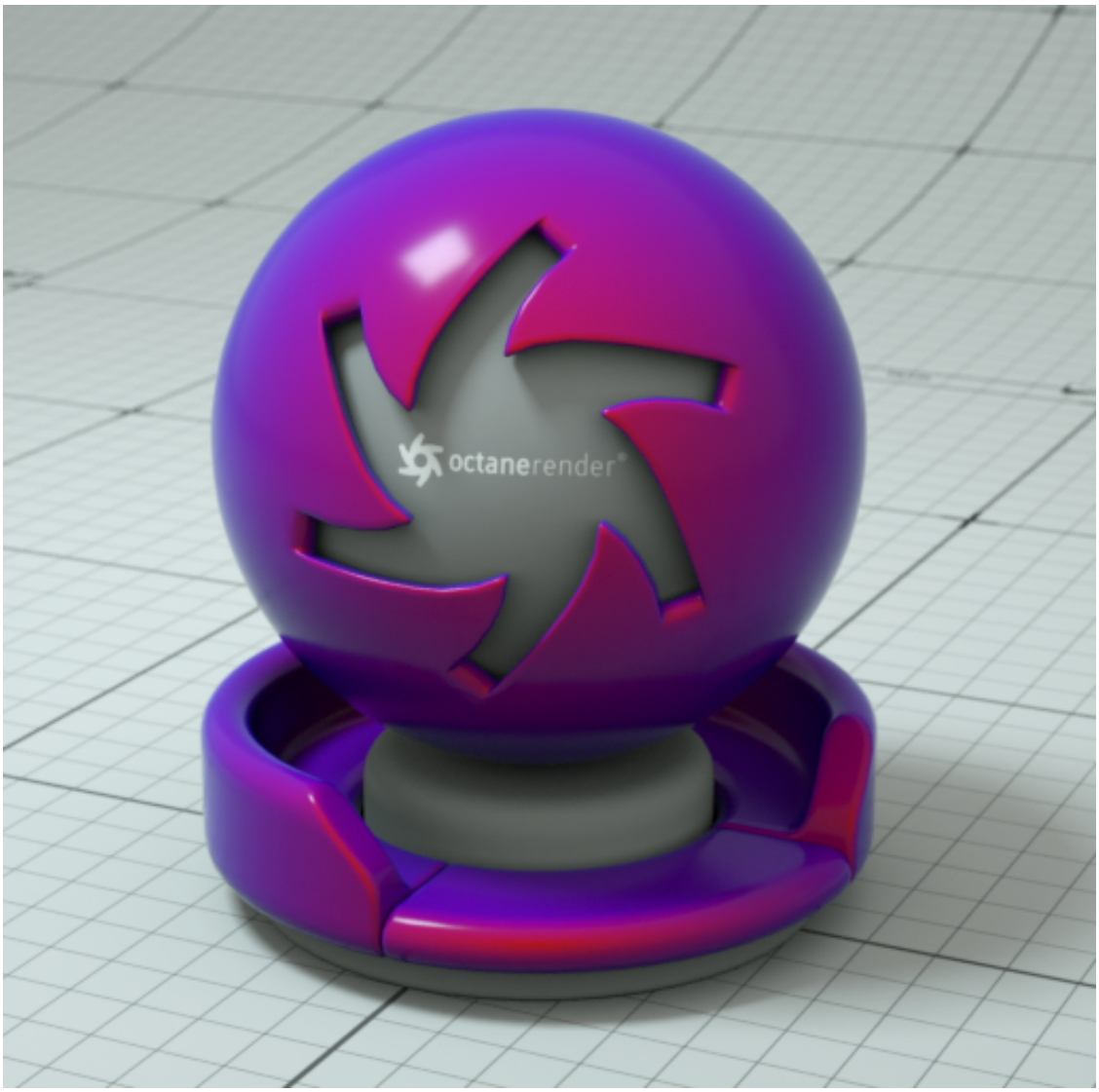
Figure 1: The Dirt texture node used to determine the Mix amount for two RGB Color nodes

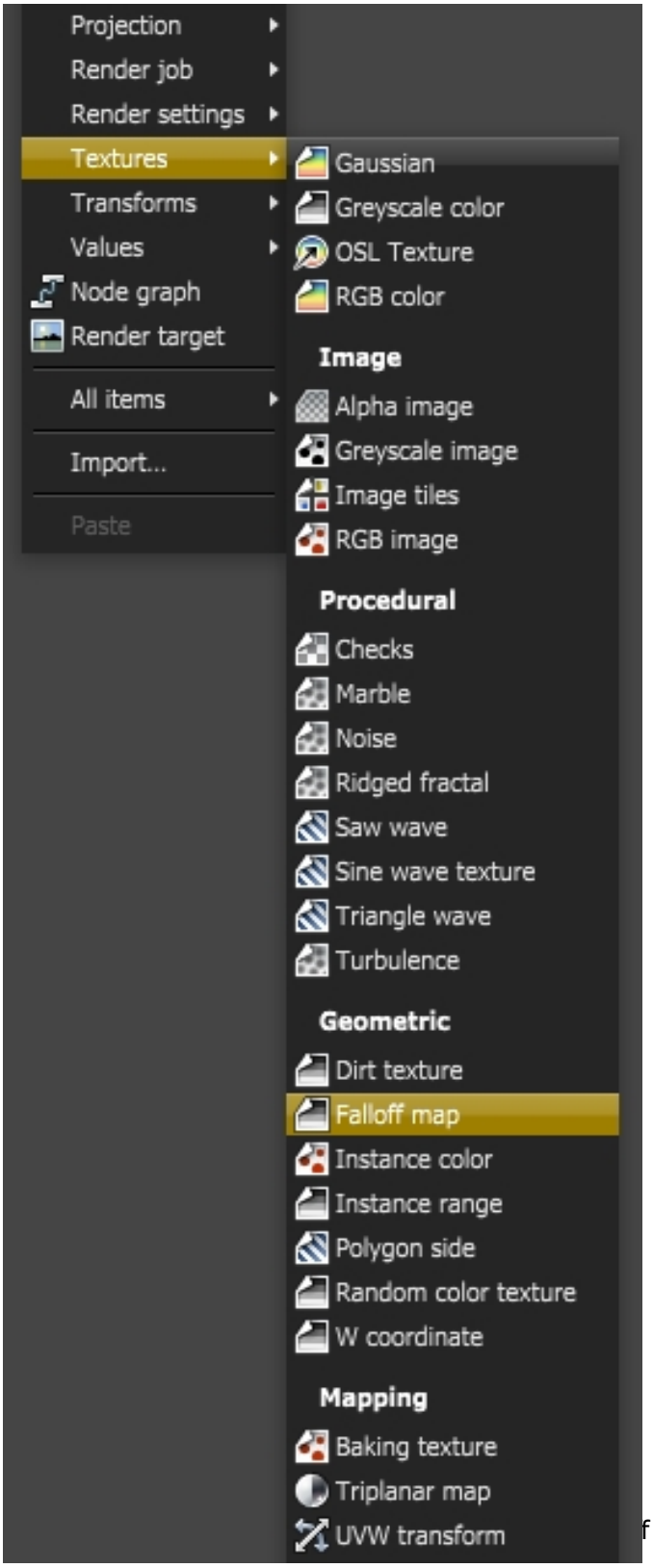
Dirt Texture Parameters

- **Strength** - Controls the Dirt intensity across the geometry surface.
- **Details** - Controls the Details intensity.
- **Radius** - Controls the dirt spread across the model's surface from the recessed parts towards the exposed parts.
- **Tolerance** - Reduces black edges on rough tessellated meshes.
- **Invert Normal** - Reverses the Dirt texture effect based on the normal surface direction.

Falloff Map

The **Falloff Map** texture controls material blending, depending on the viewing angle of the material's geometry.

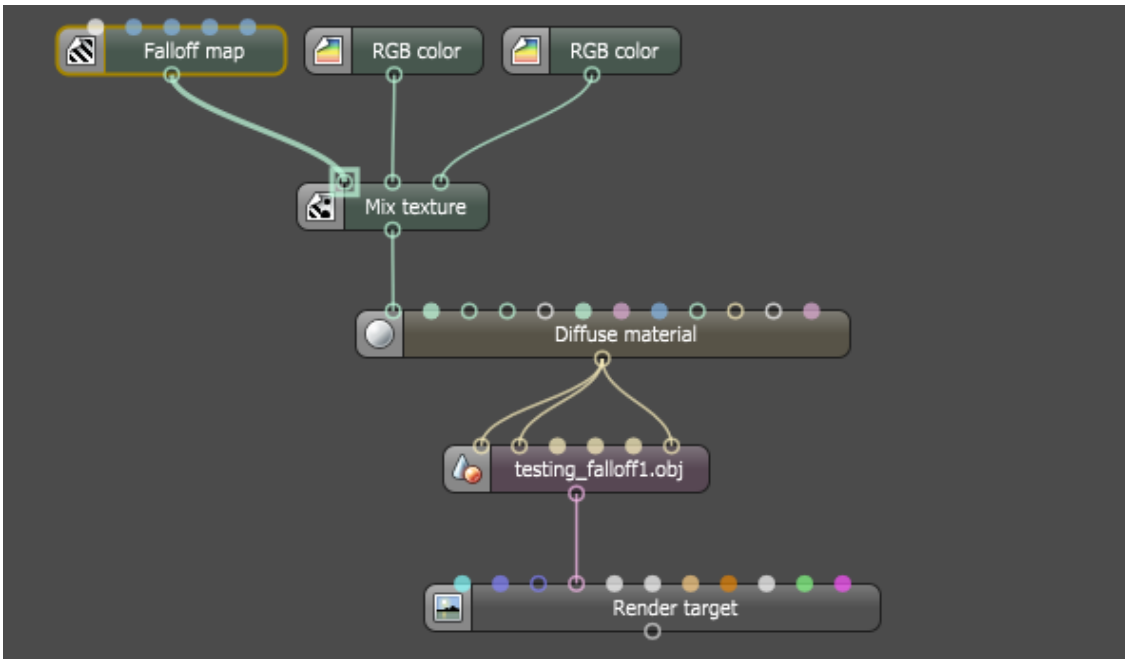



Projection
Render job
Render settings
Textures
Transforms
Values
Node graph
Render target
All items
Import...
Paste
Gaussian
Greyscale color
OSL Texture
RGB color
Image
Alpha image
Greyscale image
Image tiles
RGB image
Procedural
Checks
Marble
Noise
Ridged fractal
Saw wave
Sine wave texture
Triangle wave
Turbulence
Geometric
Dirt texture
Falloff map
Instance color
Instance range
Polygon side
Random color texture
W coordinate
Mapping
Baking texture
Triplanar map
UVW transform

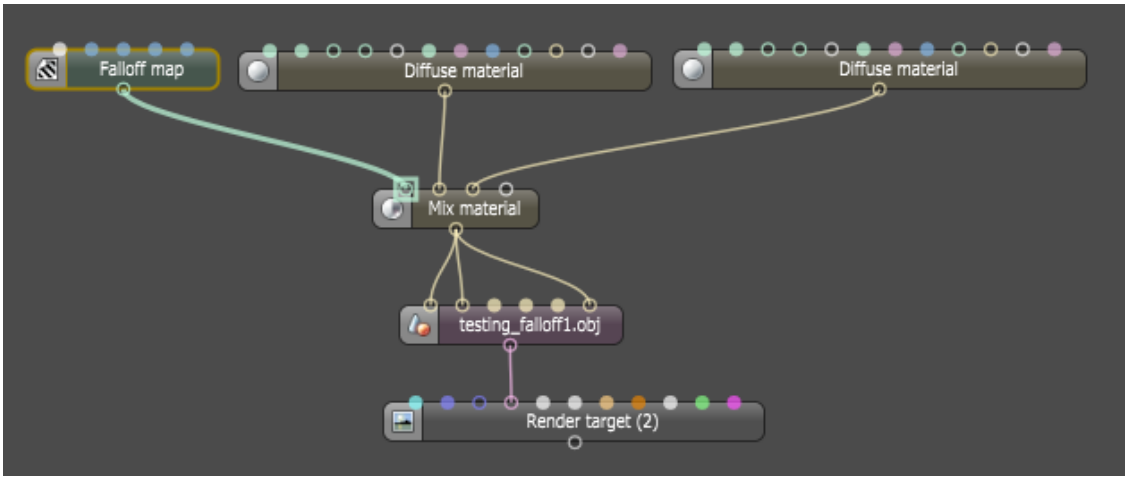
f Map

The angle between the **Eye Ray** and the **Shading Normal** is mapped from [0°, 90°] to [0, 1]. For values larger than 1, the **Falloff** node does a gamma correction using the **Falloff Skew Factor** as an exponent. OctaneRender® uses the Skew Factor to interpolate between the spectral shades resulting from the **Minimum Value** and **Maximum Value** parameters, which are based on the first and second inputs of a **Mix** node.

You can use the Falloff Map to control the blending amount of a Mix node. The Mix node can either be **Mix Texture** or **Mix Material**[1].



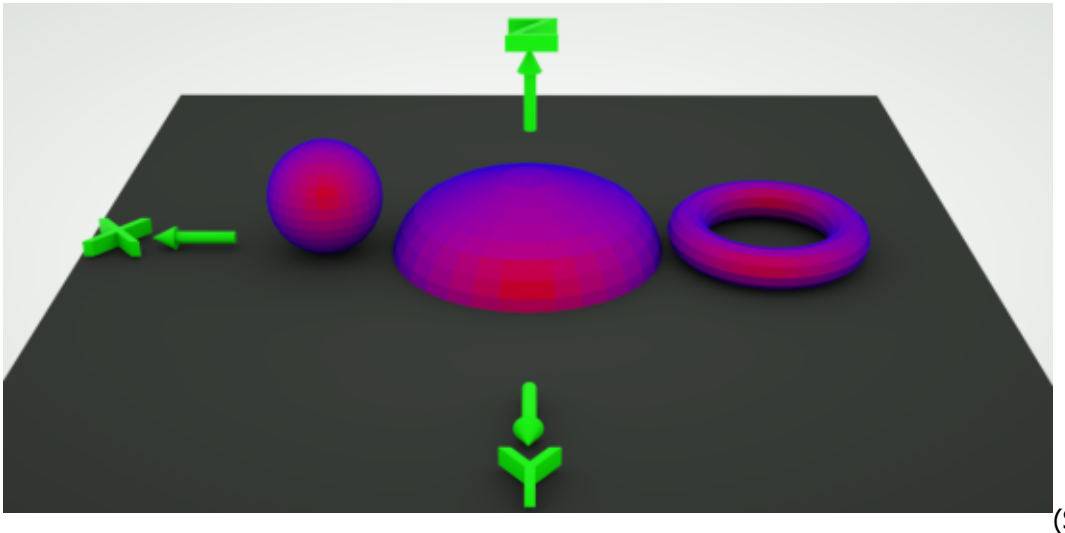
[1]The representation of the surface or volume properties of an object.



When using the Falloff Map, the Falloff node controls the **Amount** parameter of the Mix node.

Three modes of the Falloff Map

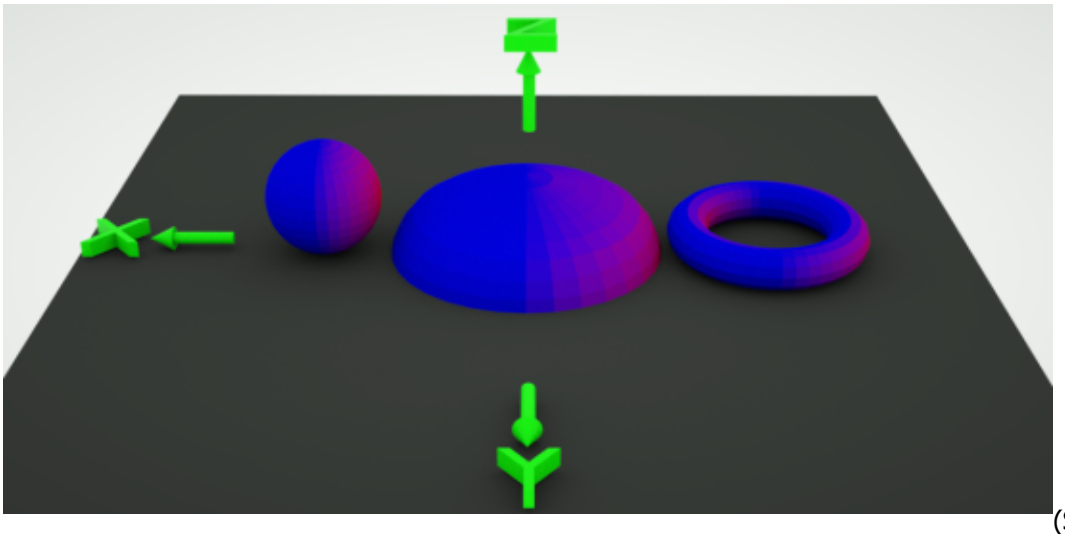
- **Normal vs. Eye Ray**: This is the default mode where OctaneRender calculates the falloff from the angle between the Surface Normal and the Eye Ray. This mode is often used for reflections. The Falloff color range affects faces directly in front of the view, and gradually falls at angled faces towards the sides as it falls away from the straight-on viewing angle. The **Falloff Direction** parameter does not apply.



(Skew

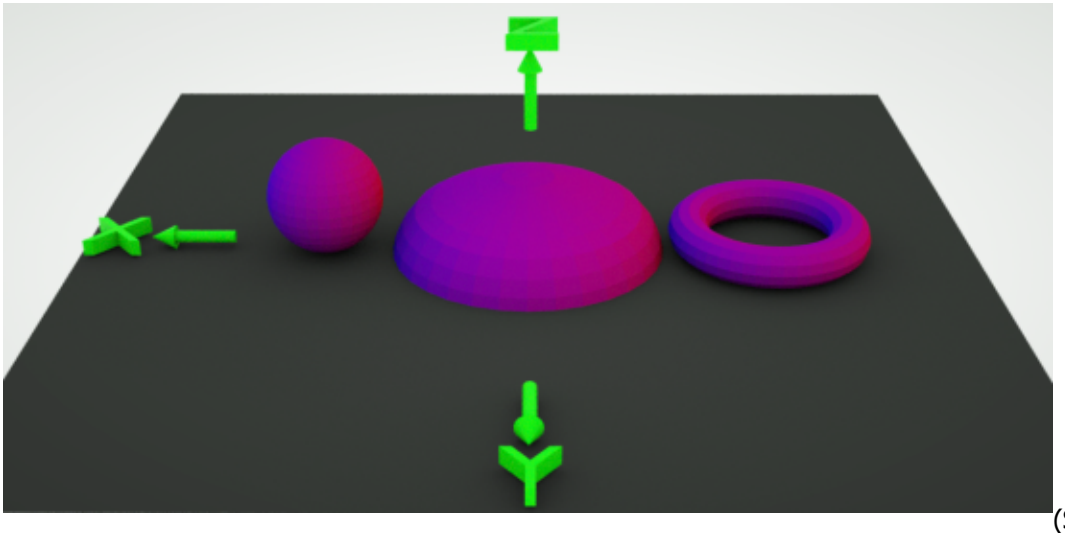
factor = 1; Direction does not apply)

- **Normal vs. Vector 90deg**: OctaneRender calculates the falloff from the angle between the Surface Normal and the specified direction vector, maxing out at 90 degrees. This is similar to the default mode, except that it maintains the effect of the color range according to the Falloff Direction parameter.



(Skew factor = 1; Direction x=1)

- **Normal vs. Vector 180deg**: OctaneRender calculates the falloff from the angle between the Surface Normal and the specified direction vector, maxing out at 180 degrees. This provides a wider color range from the Minimum to the Maximum Values, and maintains the effect of the color range according to the Falloff Direction parameter.



(Skew factor = 1; Direction x=1)

Falloff Map Parameters

- **Minimum Value** - The visible material on the surface facing the camera. A value of **0** displays the material connected to Material pin 2, and a value of **1** displays the material connected to Material pin 1.
- **Maximum Value** - Determines what material displays towards the grazing angles. A value of **0** displays the material connected to Material pin 2, and a value of **1** displays the material connected to Material pin 1.
- **Falloff Skew Factor** - Balances the **Normal** and **Grazing** angles' influence. Low values result in stronger Grazing angle influence - any textures that the Maximum Value controls cover more surface. High values result in stronger Normal angle influence - any textures that the Minimum Value controls cover more surface.

In the figure below, a red **Diffuse material**[1] connects to the **Mix material**[2]'s first Material pin, and a white **Diffuse**[3] material connects to the second Material pin. The Falloff map then connects to the Mix material's **Amount** pin.

[1]Used for dull, non-reflecting materials or mesh emitters.

[2]Used to mix any two material types.

[3]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

A Falloff texture controlling the Mix amount of a Mix material blending two Diffuse materials

A value of **0.1** leads to almost complete coverage by the grazing value regardless of viewing angle, whereas a value of **15** leads to almost complete coverage by the Normal value. This parameter's default setting is **6**.

While the index value on **Glossy**[1] and **Specular**[2] nodes corresponds to a real world **Index of Refraction** (IOR) value on dielectric materials like plastic and glass (OctaneRender doesn't yet support metals and Bezier curves), the Falloff node works differently because of this Falloff Skew Factor.

If set to **1**, then the value is proportional to the angle between the Normal and the Camera Ray (i.e. if view angle is 45°, then the value is **0.5**).

If the value is larger than **1**, then it applies a power curve to the angle.

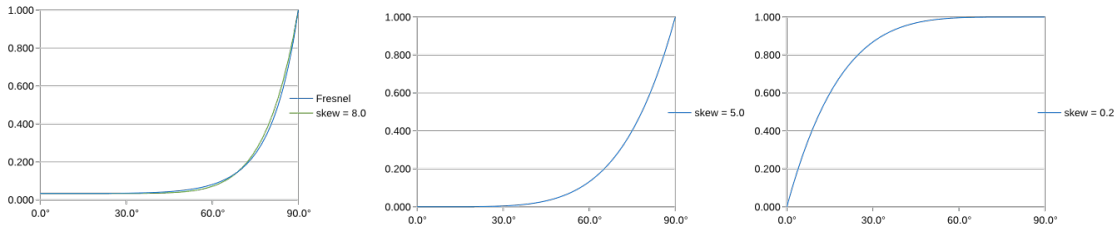
If the value is smaller than **1**, then it inverts the skew factor, and mirrors the power curve.

- $falloff \leq 1 : y = x^{falloff}$
- $falloff \geq 1 : y = 1 - (1 - x)^{(1 / falloff)}$

Falloff Direction

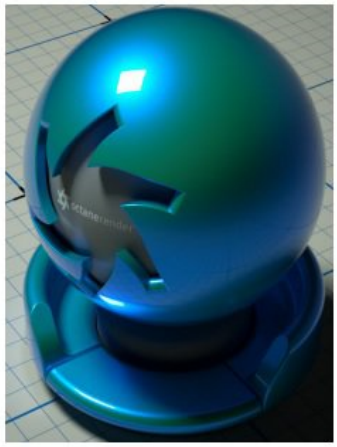
This is used by the Normal vs. Vector 90deg and Normal vs. Vector 180deg modes. For most materials, the Fresnel effect (the default mode) is often correct, while Falloff Direction applies for exceptional cases, which can adjust relative to the camera. Changing the object rotation will not change the Falloff Direction orientation.

You can approximate the behavior of glass with a Skew Factor of **8.0** and a Normal value of **0.034**.



[1]The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.
[2]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

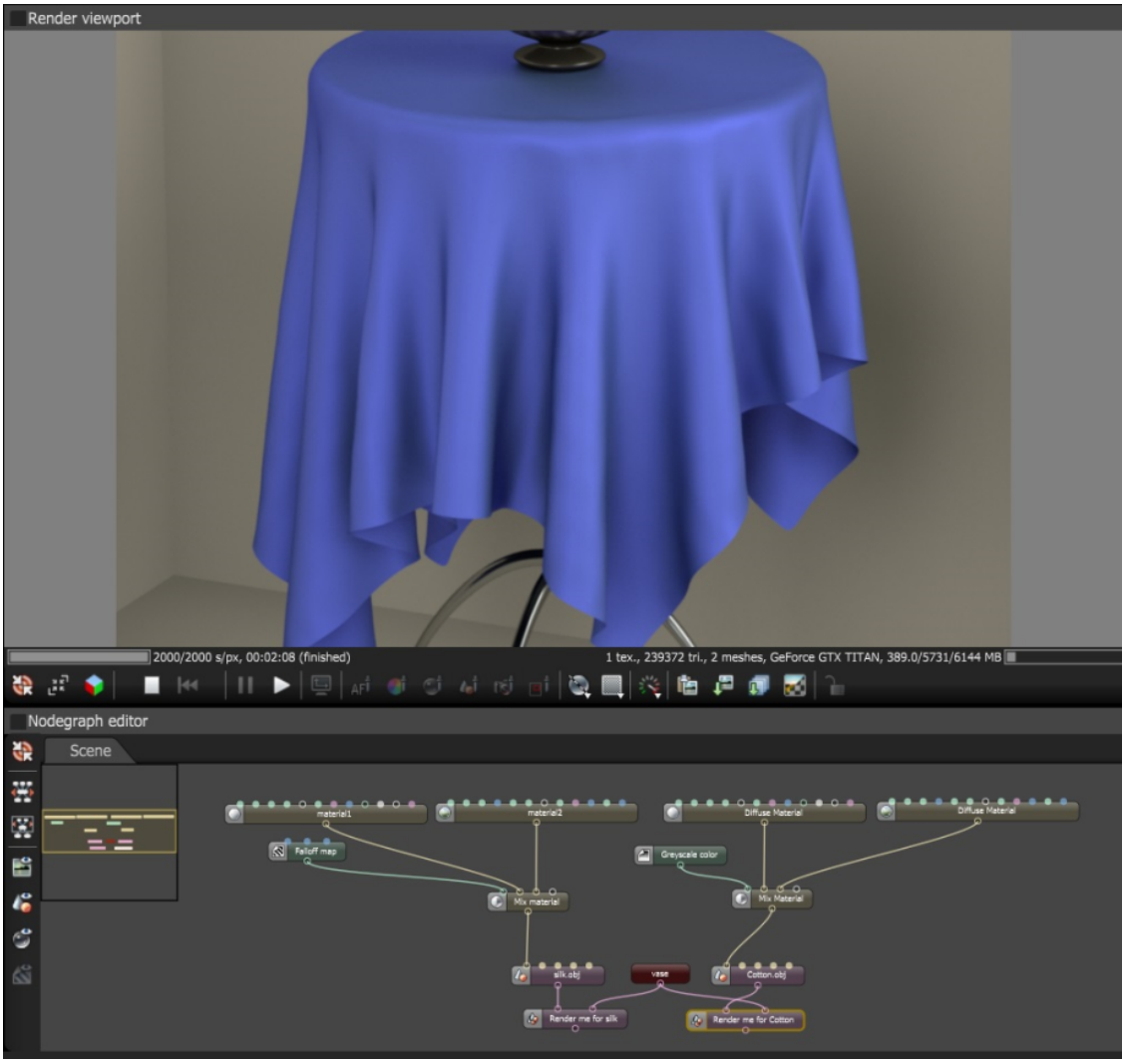
You can also use the Falloff Map for other things, like the input for glass opacity. Falloff is useful for car shaders (at 90 degrees, it should have the desired color and in less somewhat darker, but at the same time a bit more reflective), water shaders (tends to be more reflective to low angles of incidence), and fabrics like velvet (tends to become almost white at low angles). It is also useful for some metals to simulate some coating effects.



<-Image without Falloff

<-Image with Falloff

Sample Node Graph Using The Falloff Texture Map

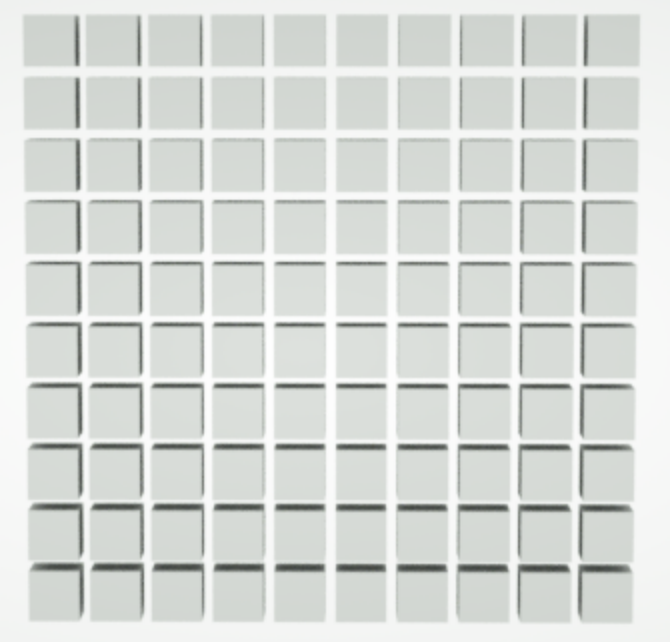

Render viewport
2000/2000 s/px, 00:02:08 (finished)
1 tex., 239372 tri., 2 meshes, GeForce GTX TITAN, 389.0/5731/6144 MB
Nodegraph editor
Scene
material1
material2
Diffuse Material
Diffuse Material
Falloff map
Greyscale color
Mix material
Mix Material
silk.obj
vase
Cotton.obj
Render me for silk
Render me for Cotton

Instance Color

The **Instance Color** texture holds an image, and prepares each pixel of the image for mapping to geometric instance IDs.

Just as a **Lua**[1] script or any of the OctaneRender® plug-ins are able to generate instances of an object, these same processes can also assign an ID to each of the instances generated, which results in a grid of instance IDs. You can then assign colors to each instance ID via **Texture** (in this case with an image in the Instance Color texture), and match the IDs with pixels of the image, starting at the bottom-left and moving up to the top-right.

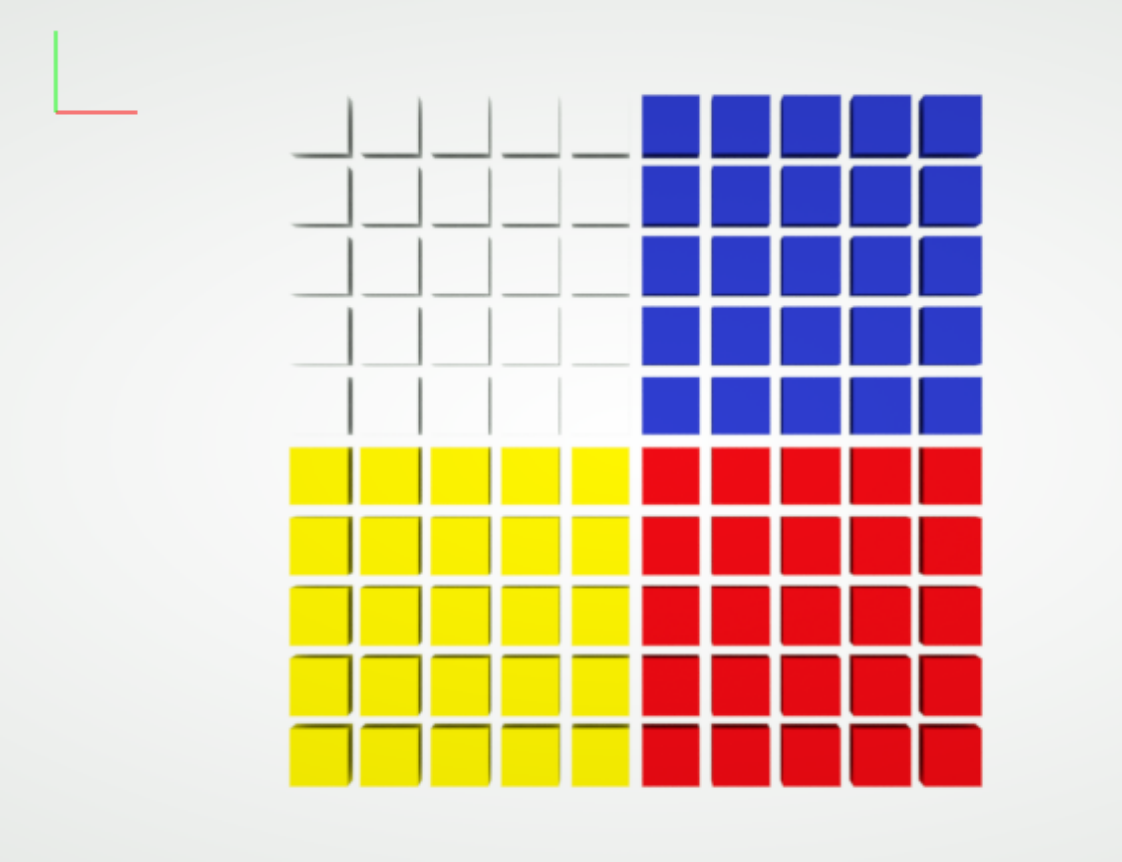
For the example below, there are 10 x 10 instances, and since a Lua script assigns IDs to each instance, OctaneRender generates 100 IDs.

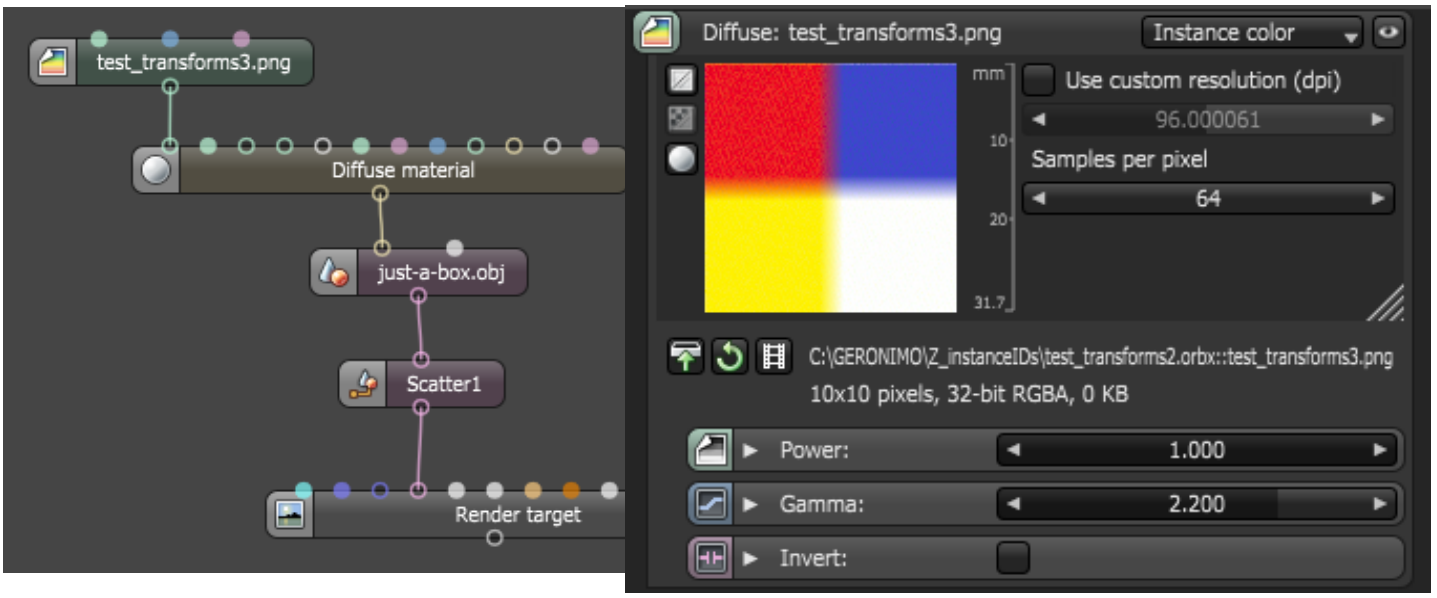


[1]A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.

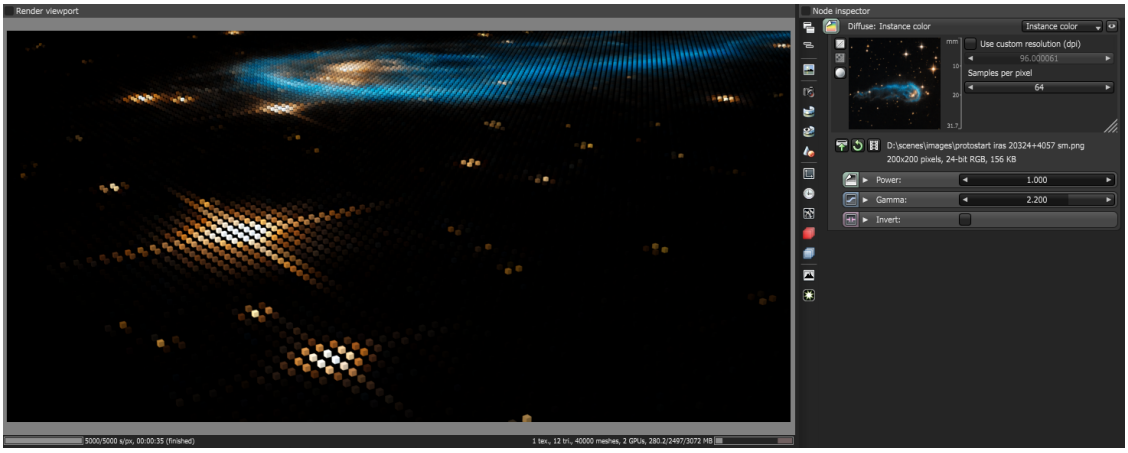
Figure 1: A cube and 99 instances of the same cube are shown here, forming a 10 x 10 grid of cubes

You can plug an image with 10 x 10 pixels into the Instance Color texture to match these dimensions. OctaneRender maps each pixel and assigns them to the instance IDs.





You can also use an existing image's dimensions as the basis for creating the instances. You can create the instances and assign an ID to each instance by a Lua script, or by an OctaneRender plug-in or any other standard object scatter plug-ins supported by OctaneRender. Below is one example of the possibilities using explicitly-defined IDs in conjunction with the Instance Color texture.



Since OctaneRender stores the colors as a texture, this option is more flexible compared to storing the colors directly with the geometry since it allows you to specify more than one color per instance.

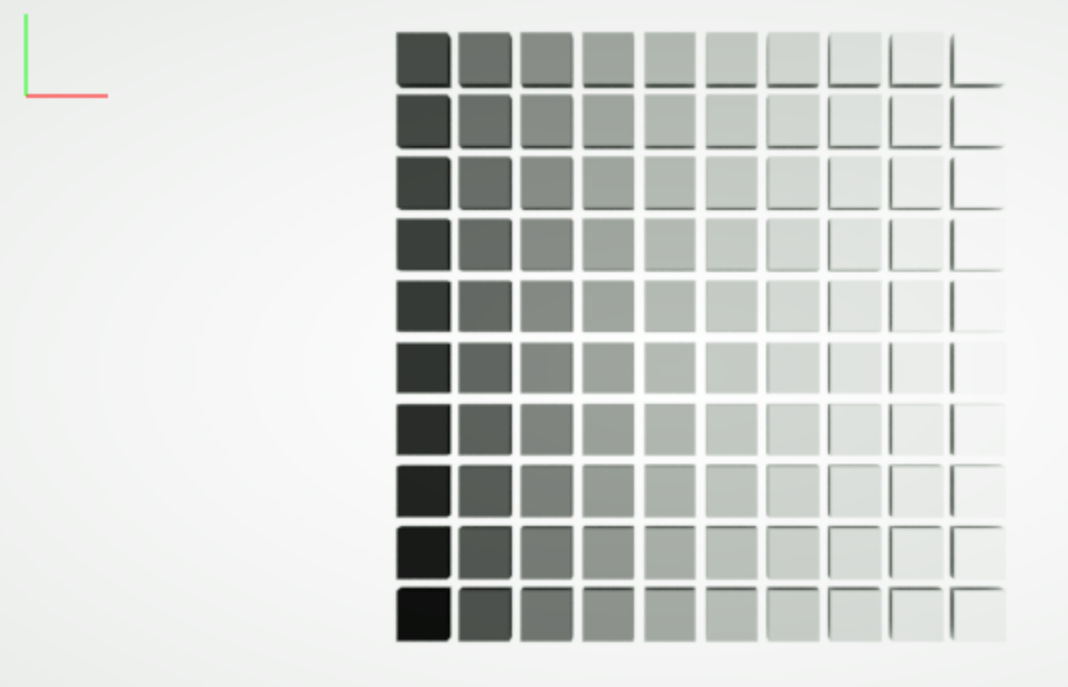
Instance Range

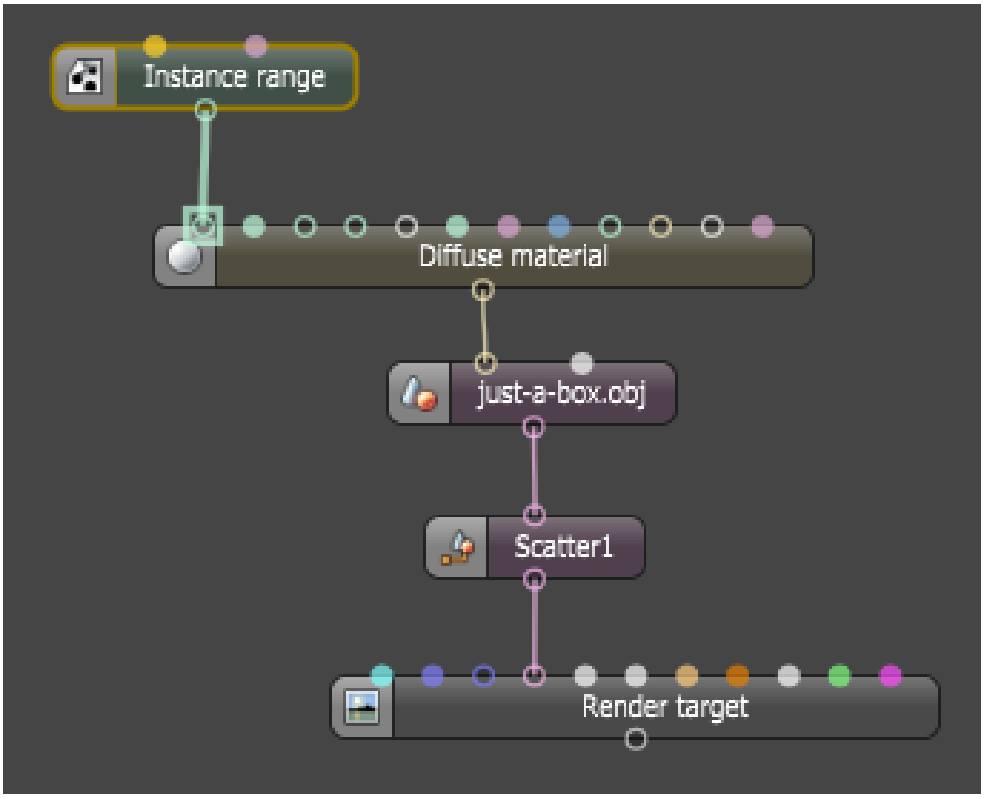
The **Instance Range** texture holds a greyscale color with the **Maximum ID** range of **0** to whatever figure you enter in this parameter, and OctaneRender® prepares this range to map it to geometric instance IDs.

Figure 1: Parameters of the Instance Range texture node

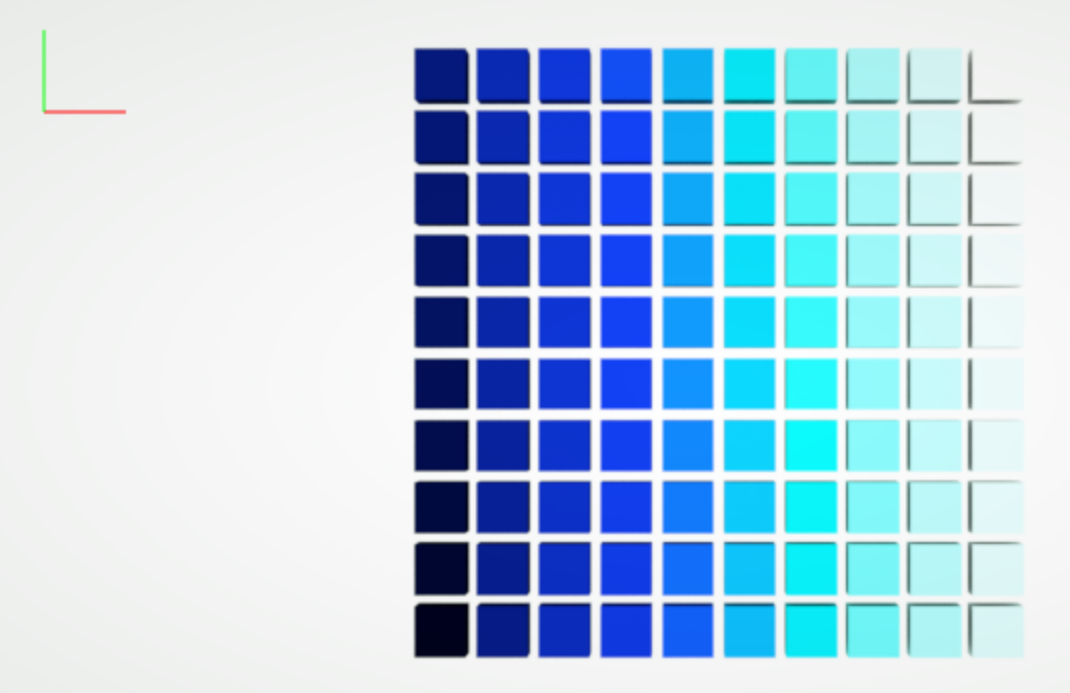
Just as a **Lua**[1] script or any of the OctaneRender plug-ins are able to generate instances of an object, these same processes can also assign an ID to each generated instance, which results in a grid of instance IDs. You can then assign colors to each instance ID via **Texture** (in this case with an image in the **Instance Color** texture), and match the IDs with pixels of the image, starting at the bottom-left and moving up to the top-right. For the example below, there are 10 x 10 instances that a Lua script assigns IDs to each instance, generating 100 IDs. To map the range, the Maximum ID attribute must match the number of generated IDs - 100 in this case.

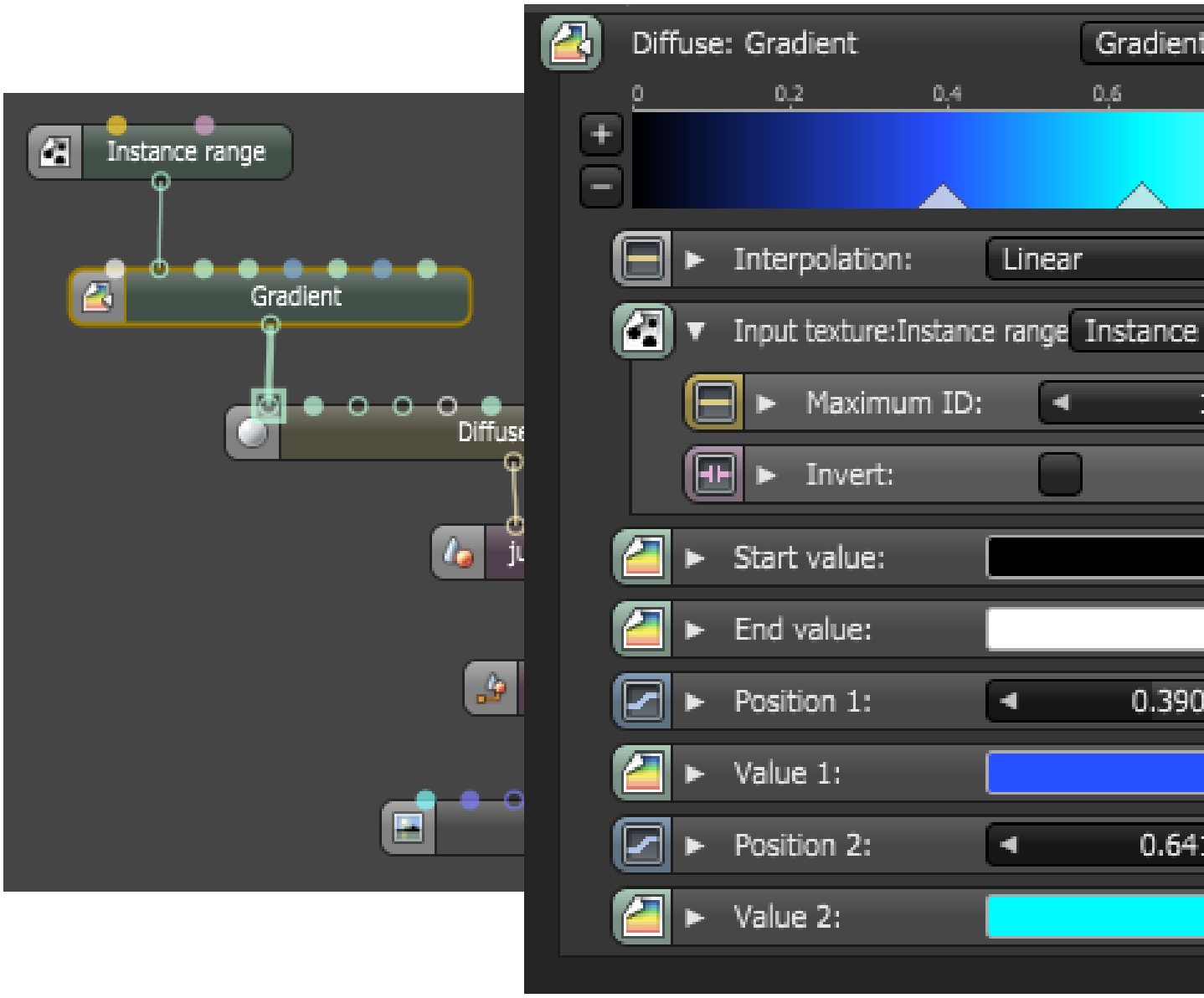
[1]A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.





You can use other mapping textures, such as the **Gradient** texture, in conjunction with the Instance Range to create some interesting variations.





Since OctaneRender stores the colors as a texture, this option is more flexible compared to storing the colors directly with the geometry, since it allows you to specify more than one color per instance.

Polygon Side

The **Polygon Side** node assigns black or white values based on the normal direction of a polygon object. This is useful for assigning different textures or materials to different sides of a polygon object. Figure 1 shows how a Polygon Side node controls the amount of a **Mix**texture (green and red **RGB Color** textures are connected to the Mix texture).

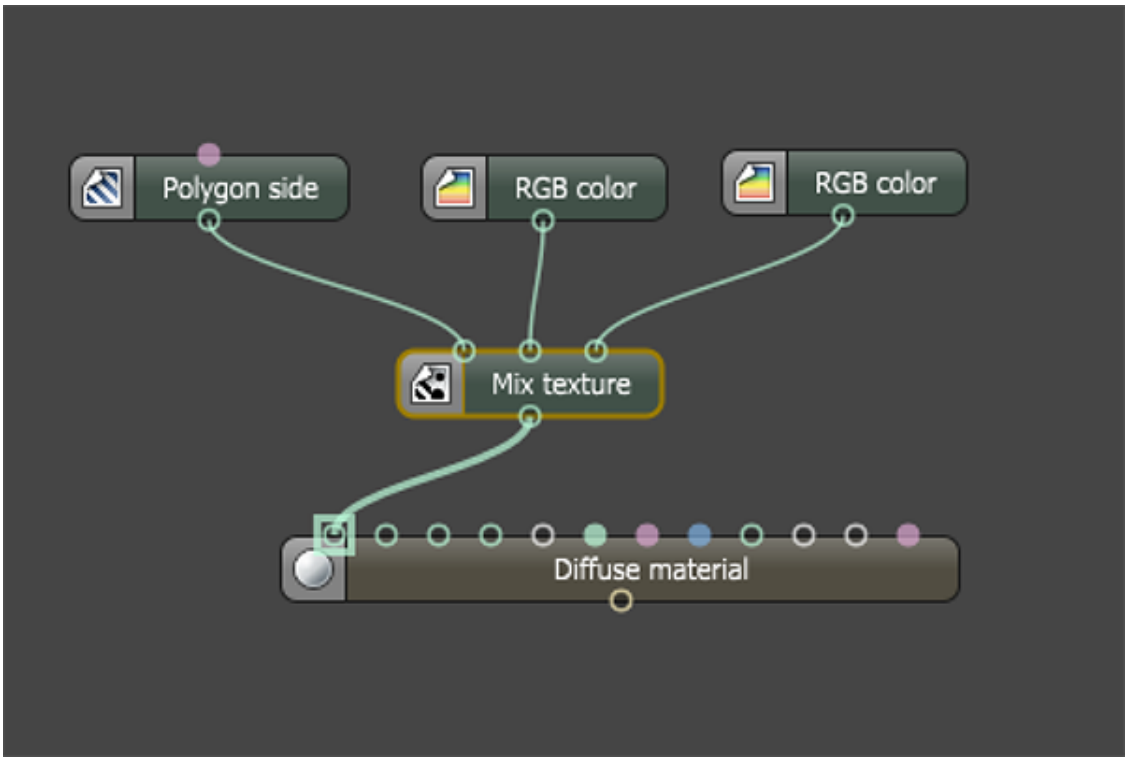


Figure 1: A Polygon Side texture controls the Mix texture blending amount, which blends green and red RGB Spectrum textures

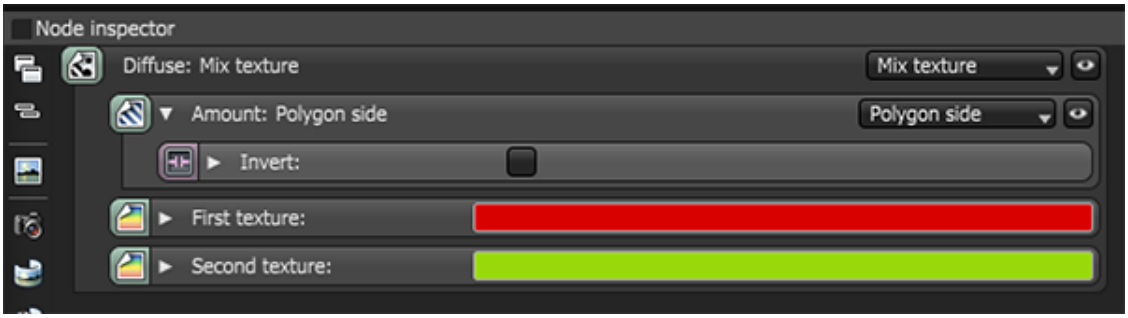


Figure 2: The Diffuse material[1] parameters in the Node Inspector

The result connects to the OctaneRender® material's **Diffuse**[2] channel, which applies to a twisted polygon. Figure 3 shows a rendering of the twisted geometry. This technique is useful for creating a leaf or butterfly wing with different colors or materials applied to each side.

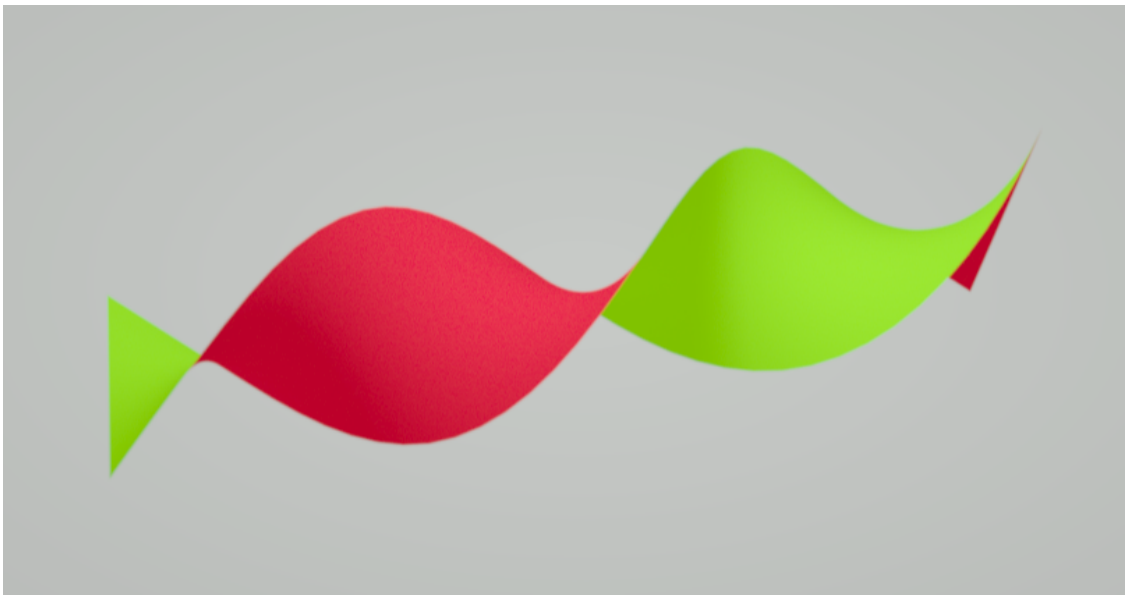


Figure 3: A twisted surface with a red texture mapped to one side of the polygon and a green texture mapped to the other

---

[1] Used for dull, non-reflecting materials or mesh emitters.

[2] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

Random Color Texture

The **Random Color** texture generates a random float value that creates color variations on instances of geometry connected to a single OctaneRender® material. Figure 1 shows a number of instances of the OctaneRender logo geometry. A single **Diffuse**[1] material applies to all of the instances. Each instance has a random shade because the Random Color texture node connects to the material's **Diffuse** channel.

Figure 2 shows a graph of the network. OctaneRender creates these instances by connecting the **Geometry** node to multiple **Placement** nodes. The Random Color texture is useful when importing baked particle simulations that contain thousands of instances. OctaneRender can apply a single material to the instances, and the Random Color texture can connect to different channels of the material to create variations in the instance shading. The Random Color texture has a single parameter - **Random Seed**. Changing this value shifts the random output of the texture.

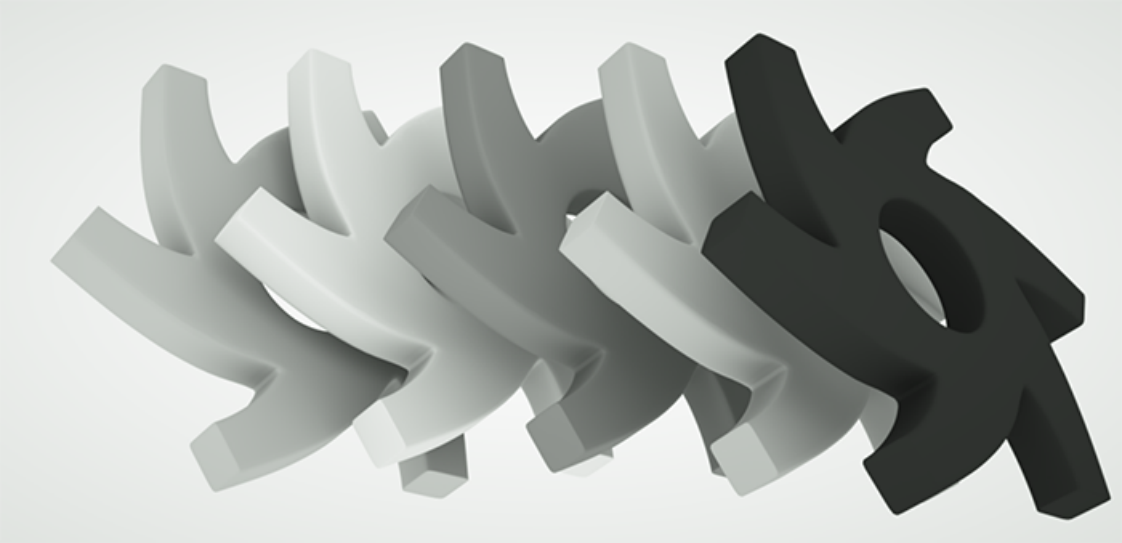


Figure 1: Several instances of the same geometry have random shading after connecting a Random Color texture to the instance's Material[2]

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2] The representation of the surface or volume properties of an object.

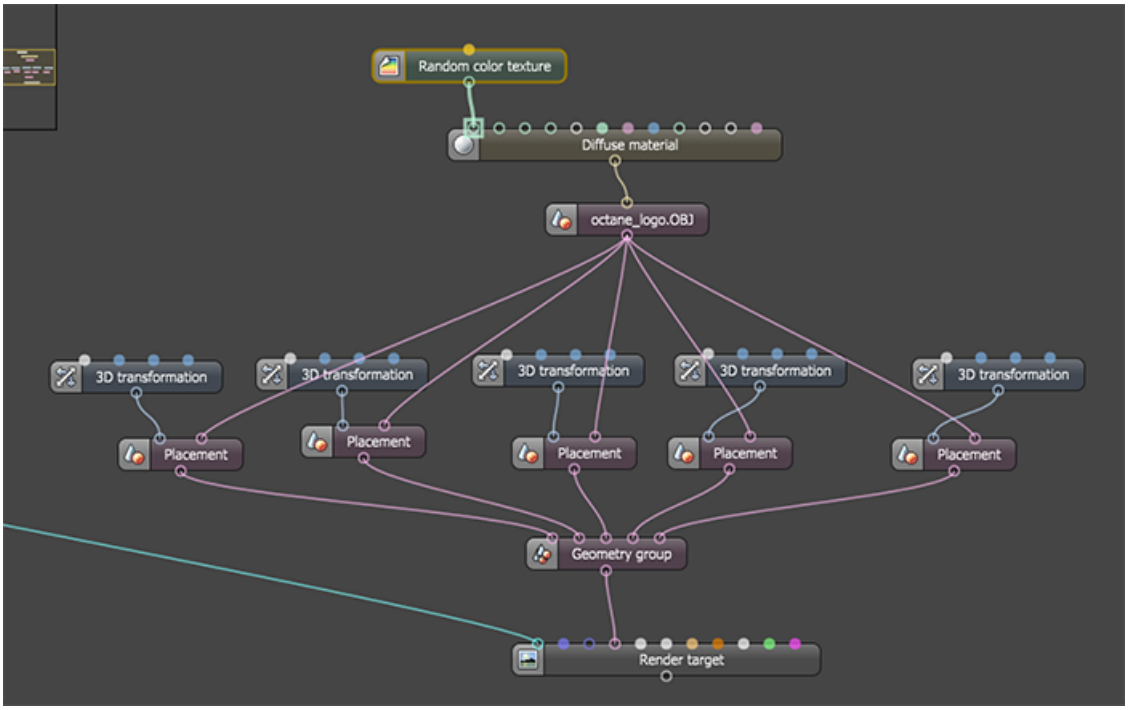
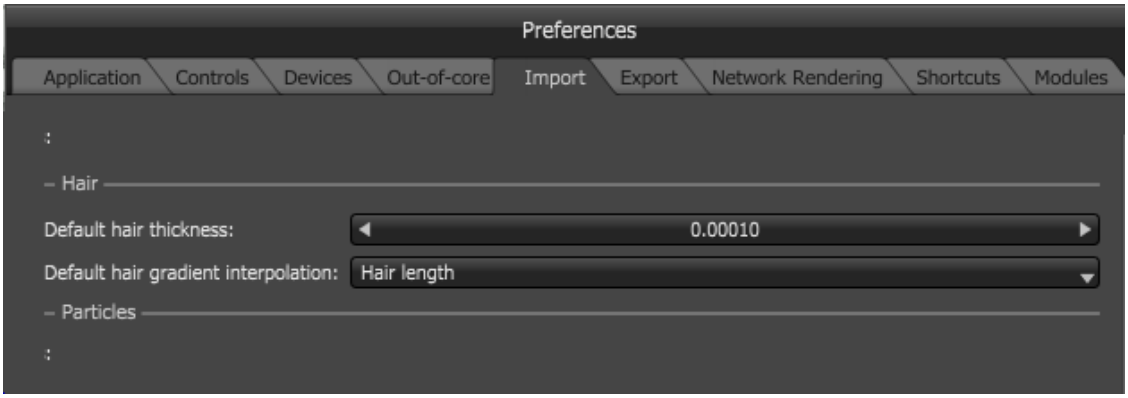


Figure 2: A graph of the geometry instances network

W Coordinate

The **W Coordinate** texture can access the OctaneRender® W Coordinate System, which provides the means to place **Gradients** on hair geometry. The hair geometry stores an inherent **Hair Gradient Interpolation** along with hair data exported from common 3D modeling applications. W is an attribute of the **Mesh** node, which defines a coordinate for every hair vertex per strand. This attribute is loaded from an **Alembic**[1] file input. However, if the attribute is not in the Alembic file, OctaneRender creates the coordinates automatically per strand. OctaneRender uses the resulting vertex coordinates to distribute a gradient per strand, and the gradient interpolation is based on settings in the **Import** tab of the **Preferences** pane.

It takes into account basis of the Interpolation set in the Import tab.

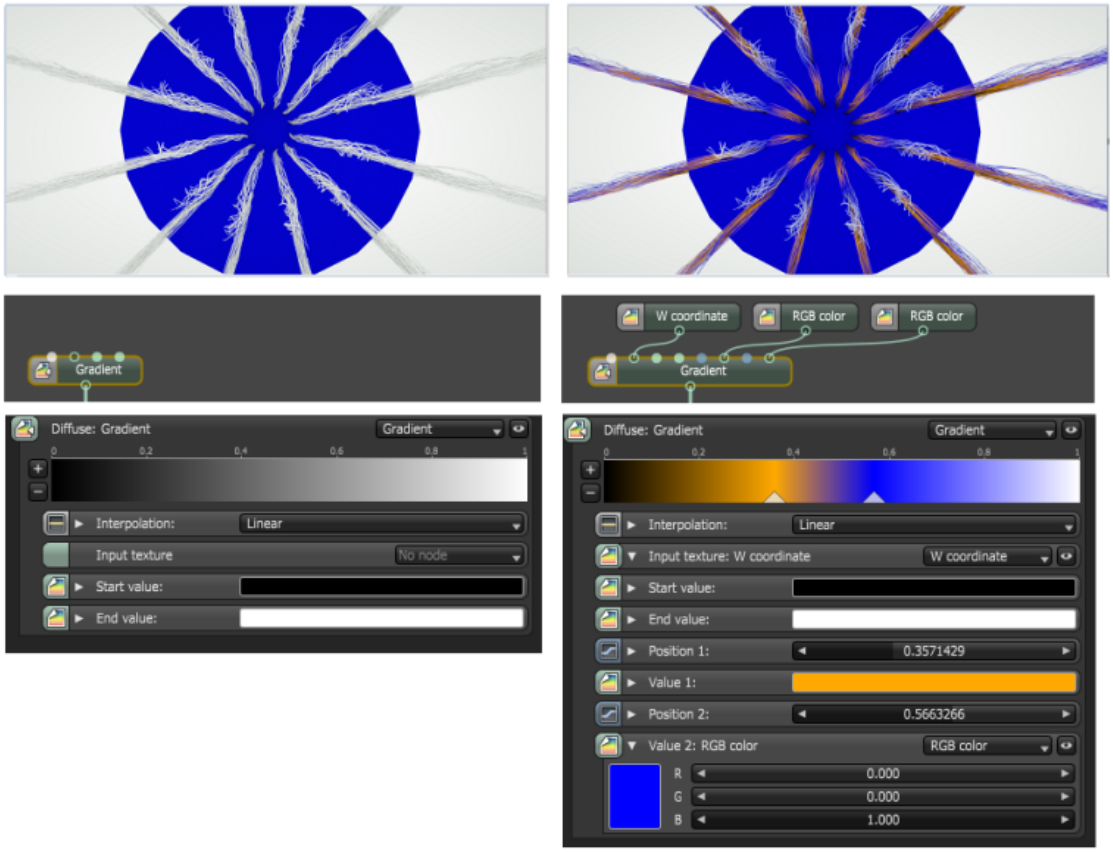


To use the W attribute to apply gradient colors to the hair data, you must plug a W Coordinate texture as the **Input** of an OctaneRender **Gradient** . This tells OctaneRender to render the inputs as a Gradient mapping, and OctaneRender uses the specified Gradient interpolation to distribute the gradient. This is based on either the hair length or the segment count per strand, depending on what is set in the Import tab for hair geometry.

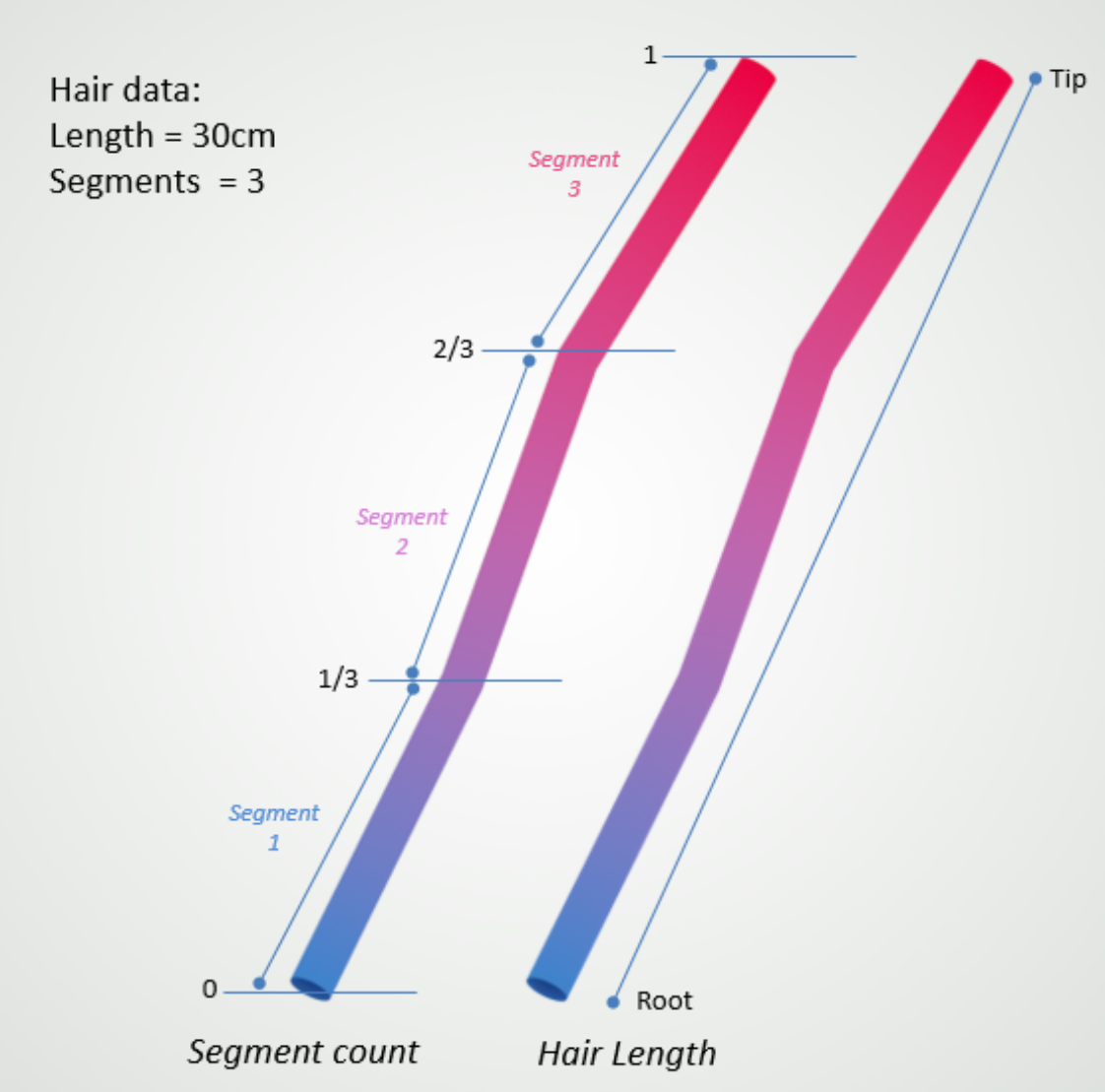
[1]An open format used to bake animated scenes for easy transfer between digital content creation tools.

Gradient texture node without W Coordinate:

Gradient texture node with W Coordinate:



For example, if the hair strand has 3 segments, and each of the hair segments have a different length, the **Hair Length** option distributes the W evenly from root to tip. **Segment Count** distributes the W independent of the segment lengths, so the first segment goes from 0 to 1/3, the second segment goes from 1/3 to 2/3, and the last segment goes from 2/3 to 1.


Hair data:
Length = 30cm
Segments = 3
1
Tip
Segment 3
2/3
Segment 2
1/3
Segment 1
0
Root
Segment count
Hair Length

Hair Segment Interpolation

Hair Length Interpolation

Operators

Operators are a category of textures used to generate patterns as a result of a mathematical function. These can then be used by themselves or in combination with the **Mapping** and **Color** textures to create surface effects that are very memory efficient compared to direct loading of bitmap images. These are also used to create bump maps and other advanced materials with minimal impact to **GPU**[1] memory.

- Add Texture
- Clamp Texture
- Color Correction
- Comparison
- Cosine Mix Texture
- Gradient
- Invert
- Mix Texture
- Multiply Texture
- Subtract Texture

[1]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Add Texture

The **Add Texture** node adds two textures together. The calculation is similar to the Add layer mode used in Photoshop® that adds the color values of two layers (Figure 1).

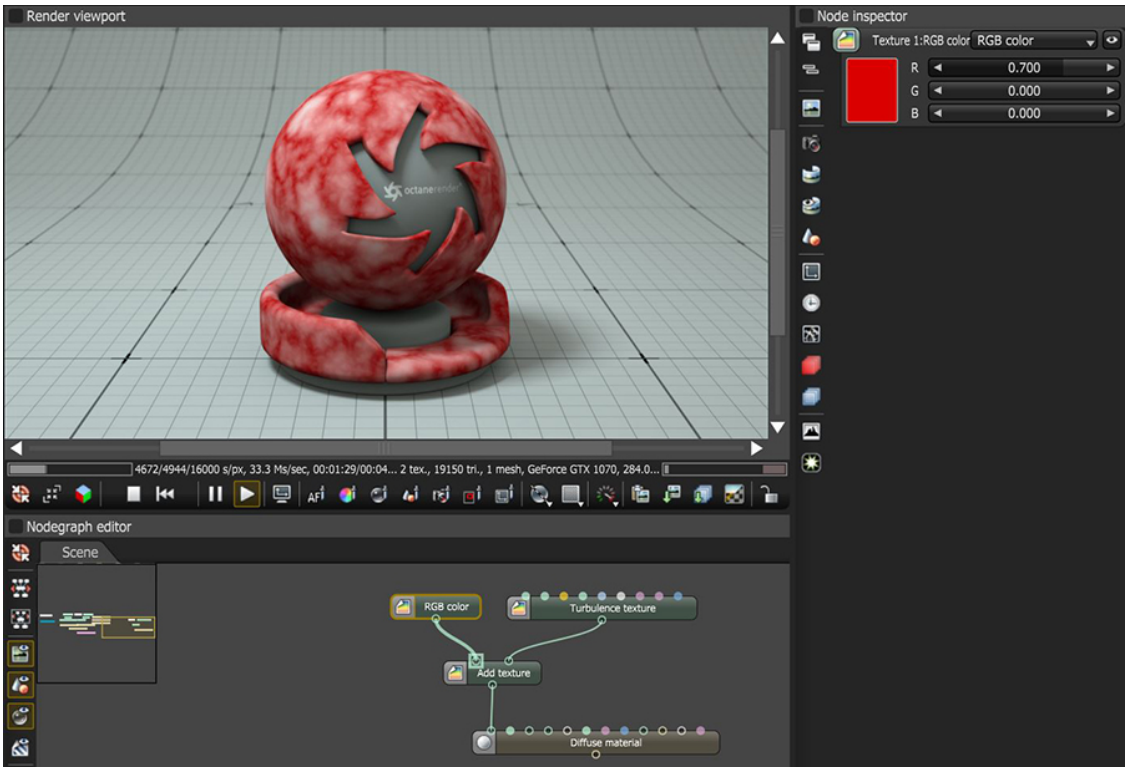
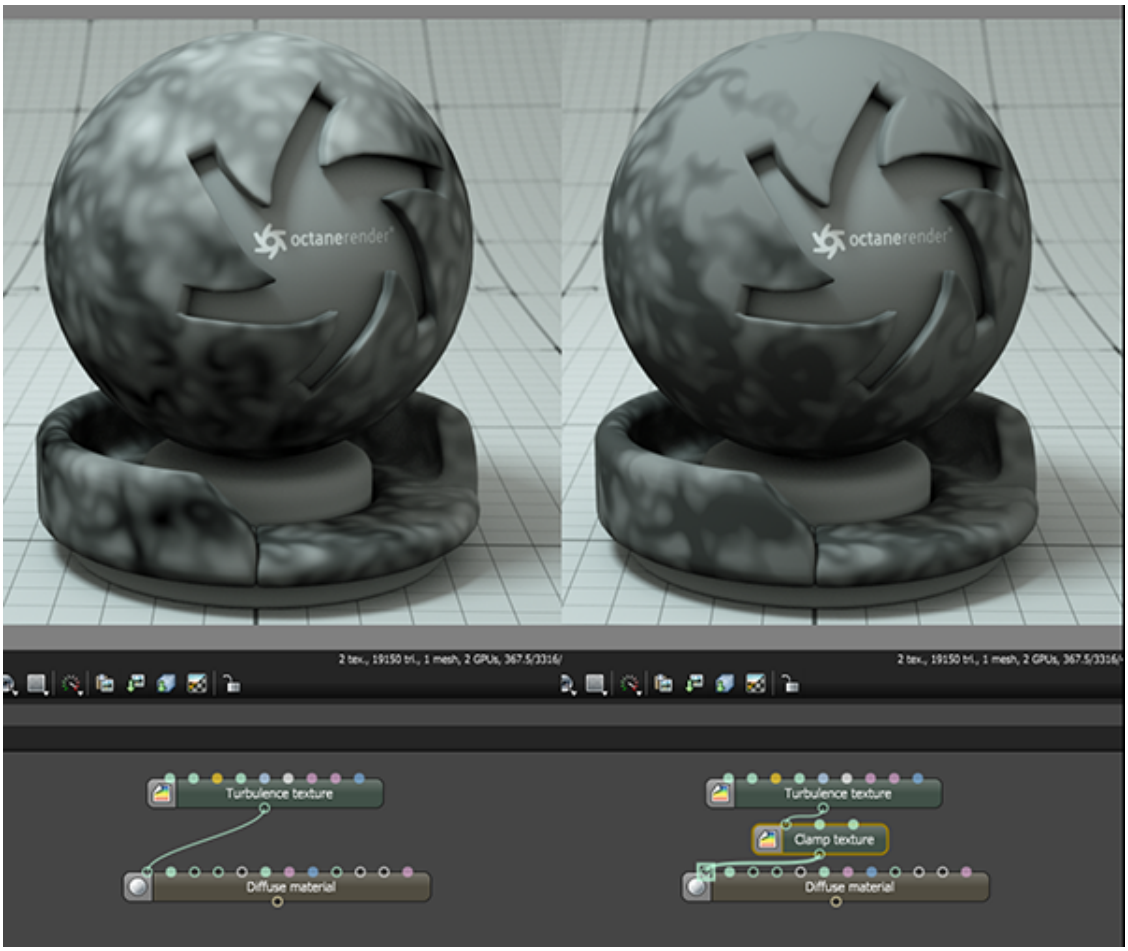


Figure 1: Add Texture adds a red color to a Turbulence texture. The result is that the dark parts of the turbulence pattern are tinted red

Clamp Texture

The **Clamp** texture provides a minimum and maximum value to clamp the values of an incoming texture map. In the example shown in Figure 1, the material on the left has the Turbulence texture connected to the **Diffuse**[1] channel of a **Diffuse** material. The material on the right shows the same Turbulence texture passing through a Clamp texture, and then connected to the Diffuse channel of another **Diffuse material**[2]. The Clamp values are set to a minimum of **0.1** and maximum of **0.3**. The result has a lot less contrast than the material on the left.



[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]Used for dull, non-reflecting materials or mesh emitters.

Figure 1: The Clamp texture alters the values of a turbulence texture

Color Correction

The **Color Correction** node adjusts image map attributes such as **Brightness**, **Hue**, **Saturation**, **Gamma**[1], and **Contrast** (Figure 1).

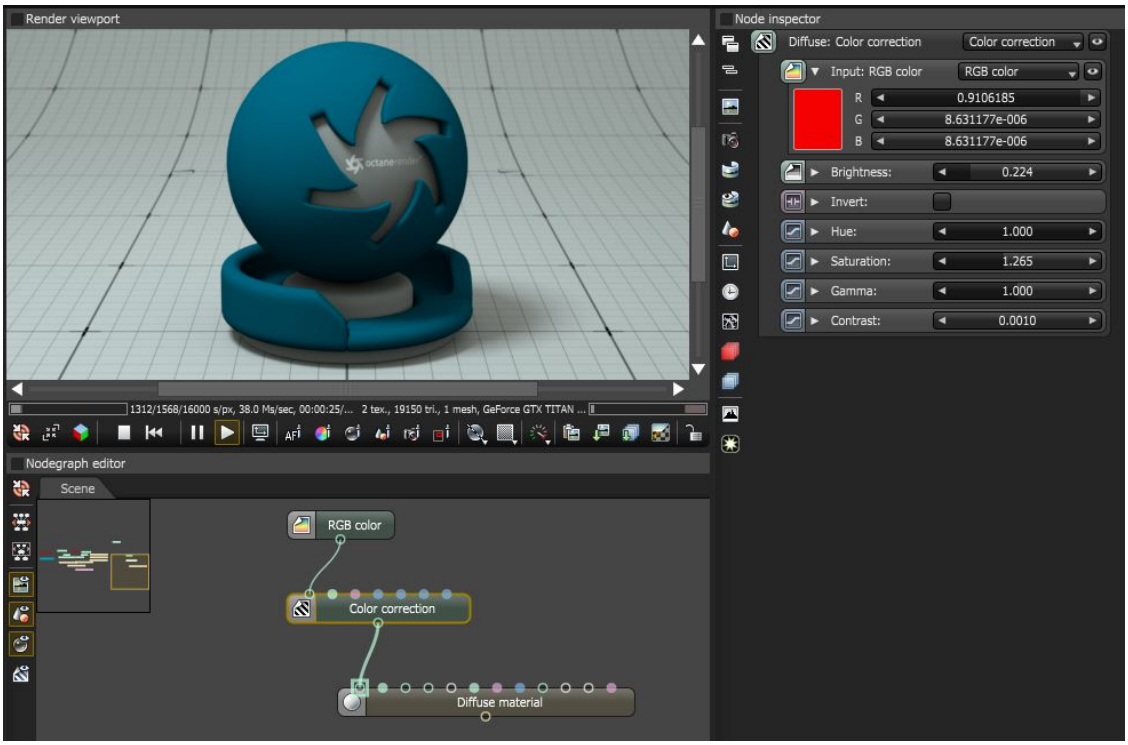


Figure 1: The Color Correction node alters the RGB Color node's Hue and Brightness values before connecting to the Diffuse[2] pin on a Diffuse material[3]

---

[1]The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.
[2]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[3]Used for dull, non-reflecting materials or mesh emitters.

Comparison

The **Comparison** node offers you the ability to use a logical comparison operator as a way to combine textures. The node takes four inputs: the first two inputs are the textures for comparison. The second two inputs are the result of the comparison. (Figure 1).

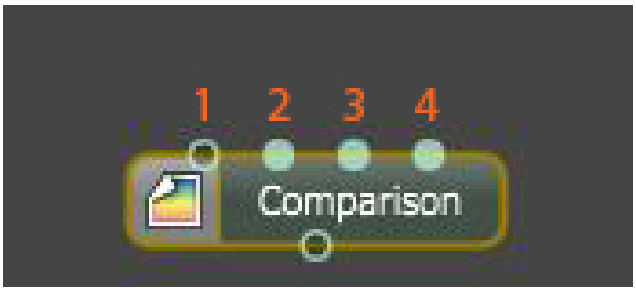
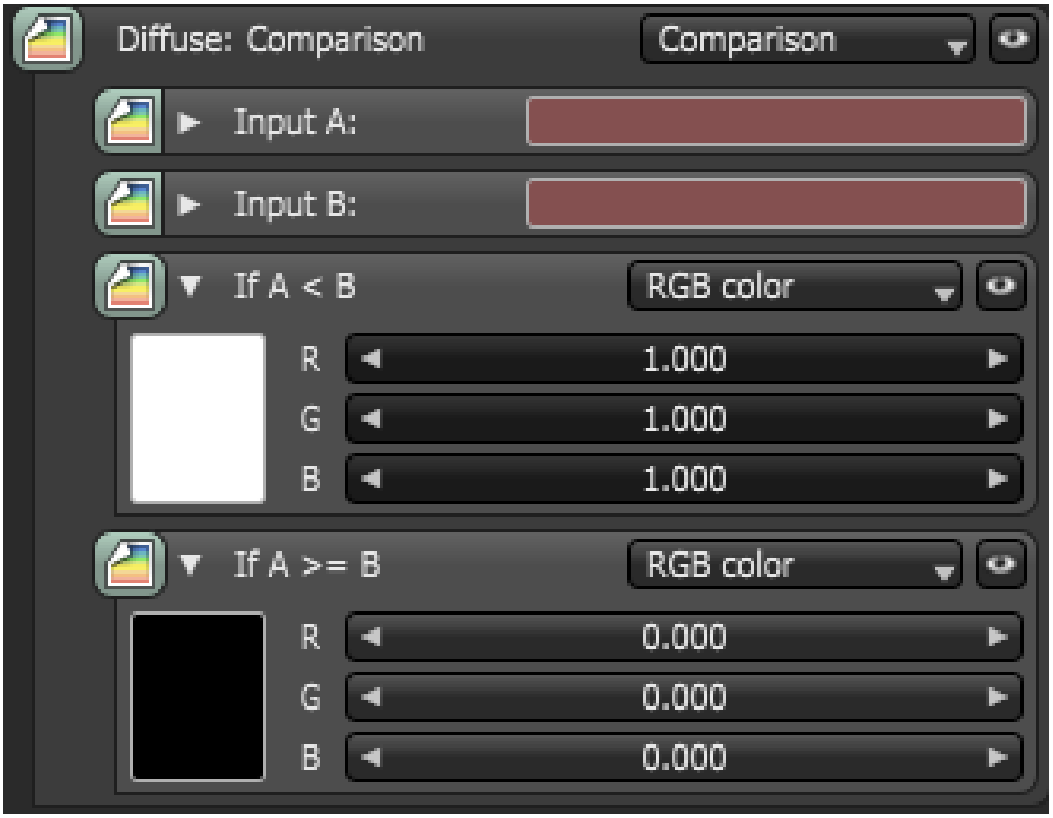


Figure 1: The various input pins for the Comparison node: (1) Input 1, (2) Input 2, (3) If A<B, (4) If A>=B

This is a conditional **Texture** node for altering a given texture (input A) based on another texture (input B). In its simplest form, the Comparison node is used to select between two alternative textures at render time.

In the example below, the result will be black since inputs A and B are the same, and it evaluates to "true" for the second condition.


Diffuse: Comparison
Comparison
Input A:
Input B:
If A < B
RGB color
R
1.000
G
1.000
B
1.000
If A >= B
RGB color
R
0.000
G
0.000
B
0.000

Cosine Mix Texture

The **Cosine Mix** texture mixes two textures together according to a cosine wave. In Figure 1, a **Checks** texture combines with a **Marble** texture using a Cosine Mix texture, and then it connects to the **Diffuse**[1] channel of a **Diffuse** material. It is very similar to the **Mix** texture, but the difference between the Cosine mix texture and the Mix texture is more apparent when the **Mix Amount** parameter is shifted towards **0** or **1**.

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

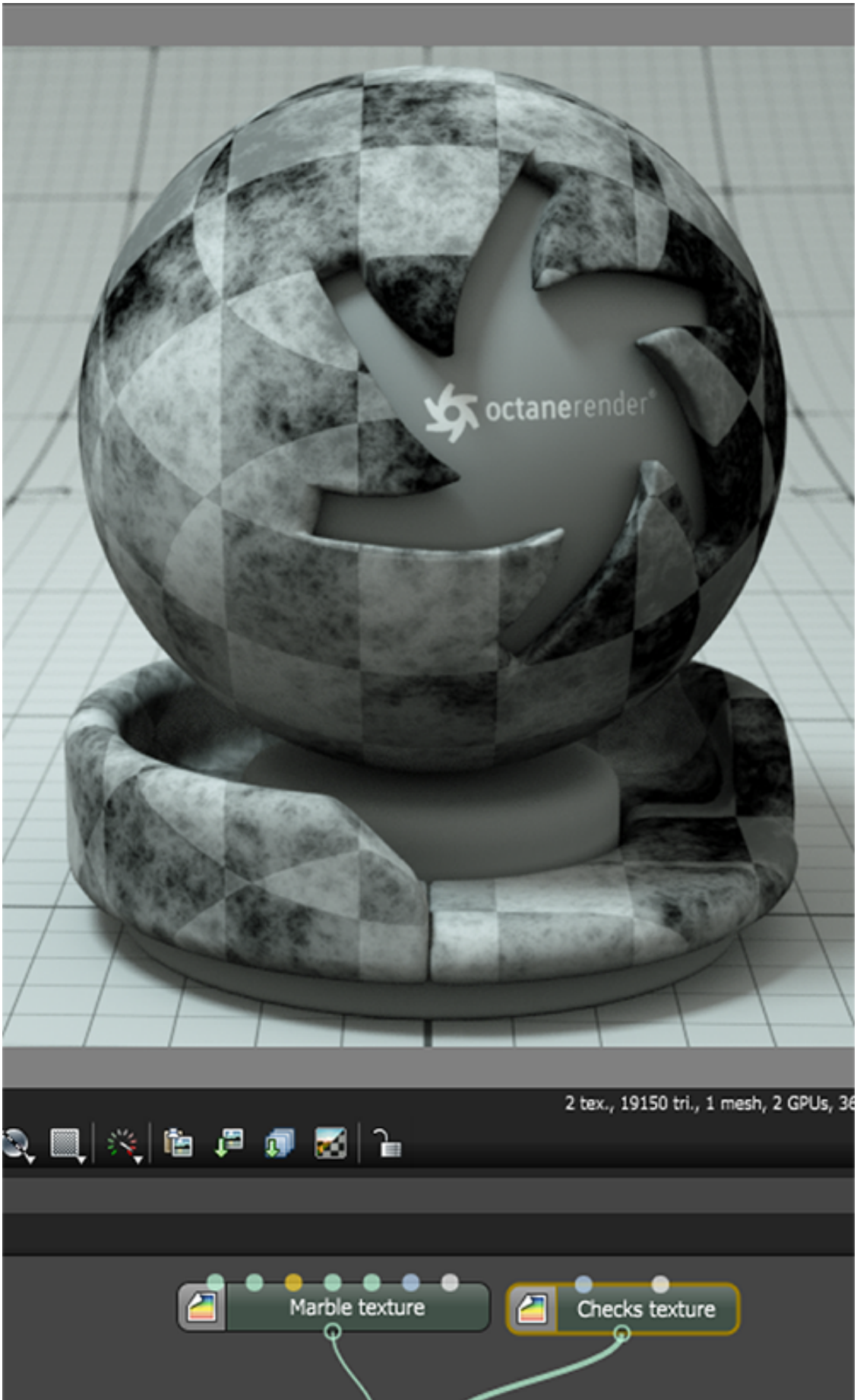

octanerender
2 tex., 19150 tri., 1 mesh, 2 GPUs, 3
Marble texture
Checks texture

Figure 1: A Marble texture is blended with a Checks texture using the Cosine Mix texture

The parameters of the Cosine Mix texture consist of the inputs for the two textures, and the Mix Amount parameter. The Mix Amount parameter accepts a float value or any texture that outputs a float, such as a **Greyscale Image** texture.

Gradient

The **Gradient** texture produces a gradient blend between colors. It accepts an input to determine how the gradient maps to the surface. In Figure 1, a gradient that goes from green to red connects to the **Diffuse**[1] channel of an OctaneRender® material. OctaneRender maps the gradient using a **Falloff** map, resulting in the reddish color of the gradient being more visible on the edges of the surface that face away from the camera, and the green color appearing on the parts of the surface that face the camera. Figure 2 shows the Gradient texture in the **Node Inspector**.

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

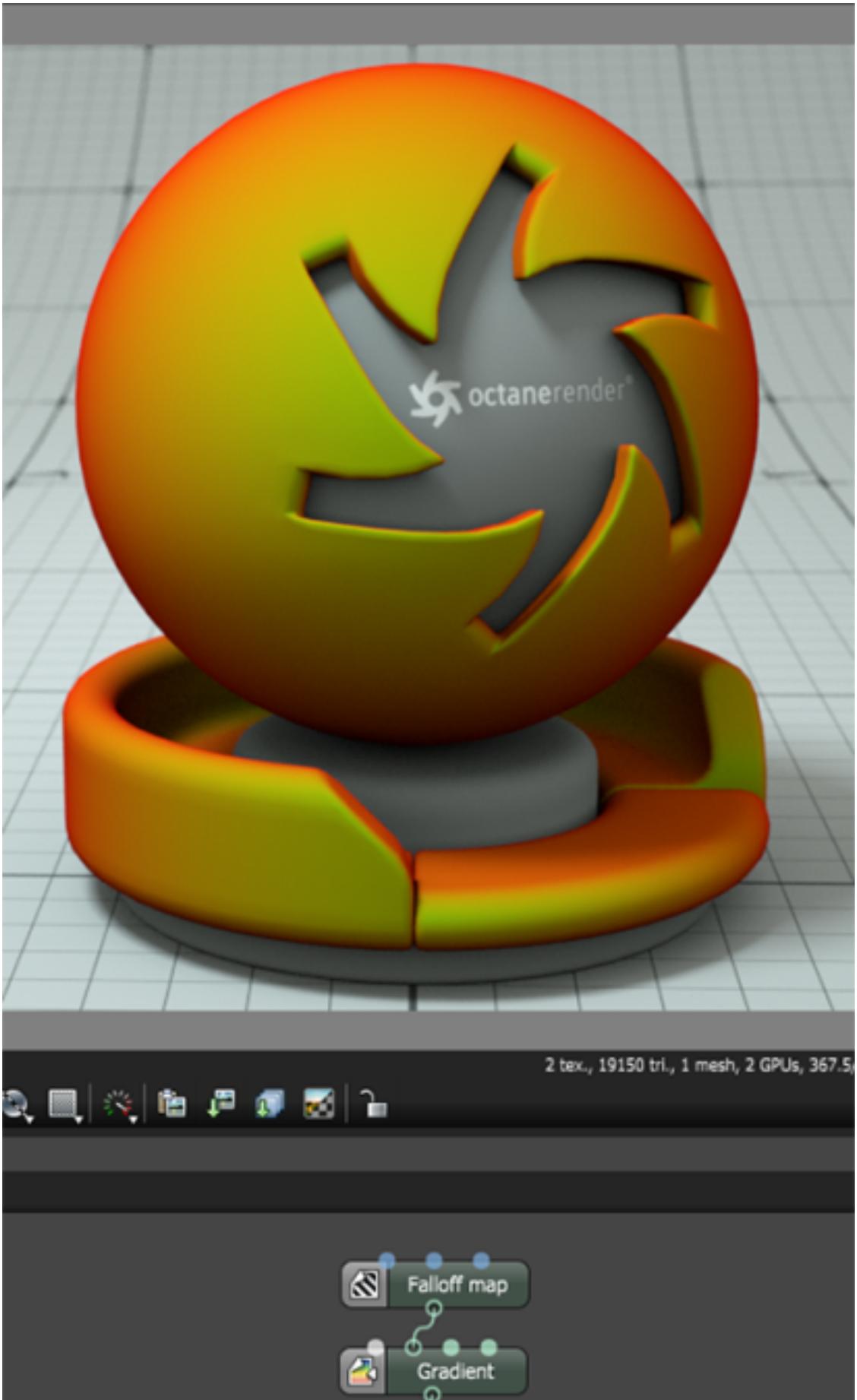

octanerender
2 tex., 19150 tri., 1 mesh, 2 GPUs, 367.5
Falloff map
Gradient

Figure 1: A Falloff map is mapping a colored gradient to a surface

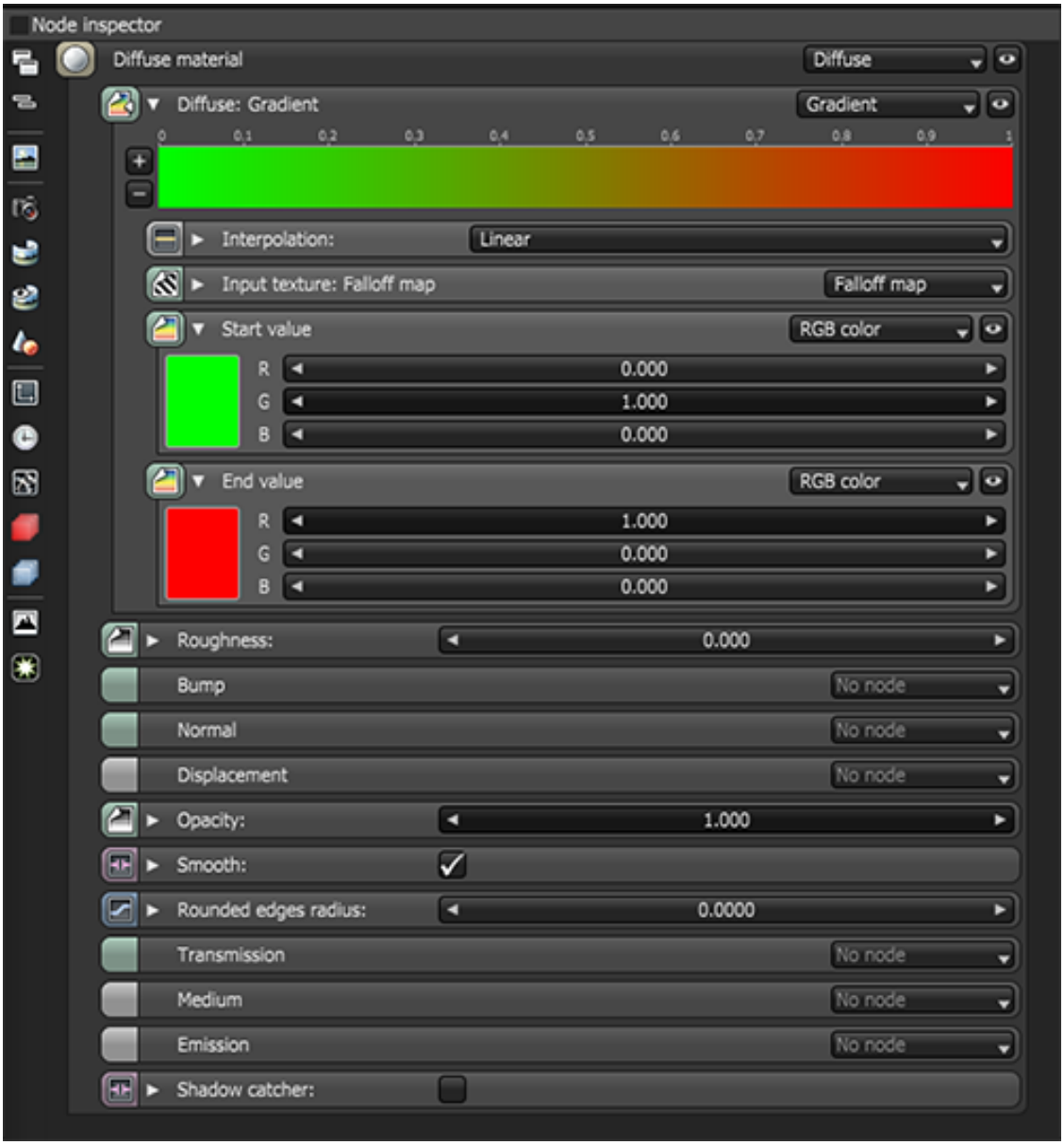


Figure 2: The parameters of the Gradient texture in the Node Inspector

Gradient Parameters

- **Gradient** - Determines the gradient's colors. Use the **+** and **-** buttons to add or remove gradient markers. Each new marker creates an arrow and a new color input option. You can place the color on different parts of the gradient by dragging the marker around.
- **Interpolation** - Select **Constant**, **Linear**, or **Cubic** to determine the color-blending rate from one marker to the next.
- **Input Texture** - Determines how the color maps to the surface.
- **Start Value, End Value** - Use the color swatches or RGB values to set the gradient's starting and ending colors.

Invert

The **Invert** texture can be used to reverse the colors or values in a **Texture** map or **Procedural** map. In Figure 1, the Invert Texture node flips the black and white areas of a **Checks** texture.

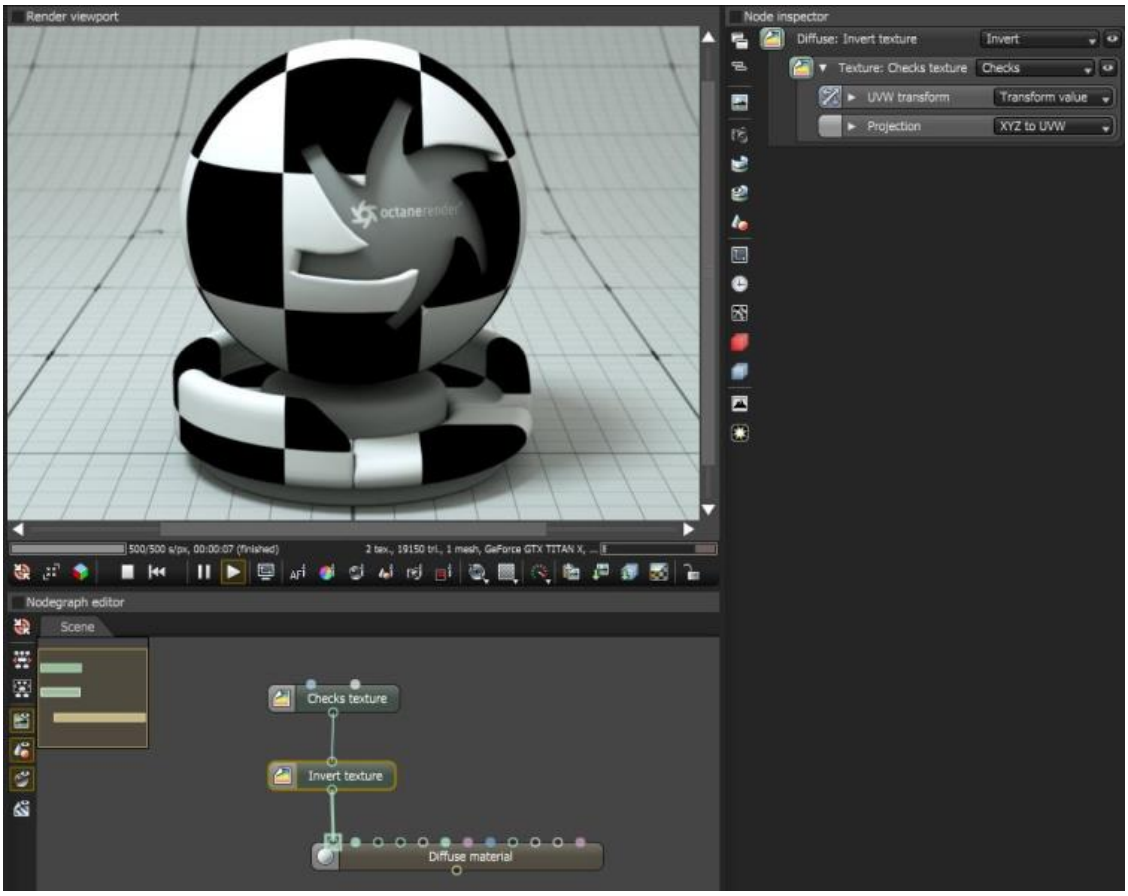


Figure 1: An Invert texture reverses the color values on a Checks texture

Mix Texture

The **Mix** texture mixes two textures together. In Figure 1, a **Marble** texture combines with a **RGB Color** texture using the Mix texture that's connected to the diffuse component of an OctaneRender® material. By default,a float value controls the **Amount**. A value of **0** means the **First Texture** is visible, and a value of **1** means the **Second Texture** is visible. Values in between blend the two textures together in a linear fashion. The Mix texture is similar to the **Cosine Mix** texture, except for the behavior of the mix slider. In Figure 1, the **Mix Amount** is controlled with a **Sine Wave** texture.

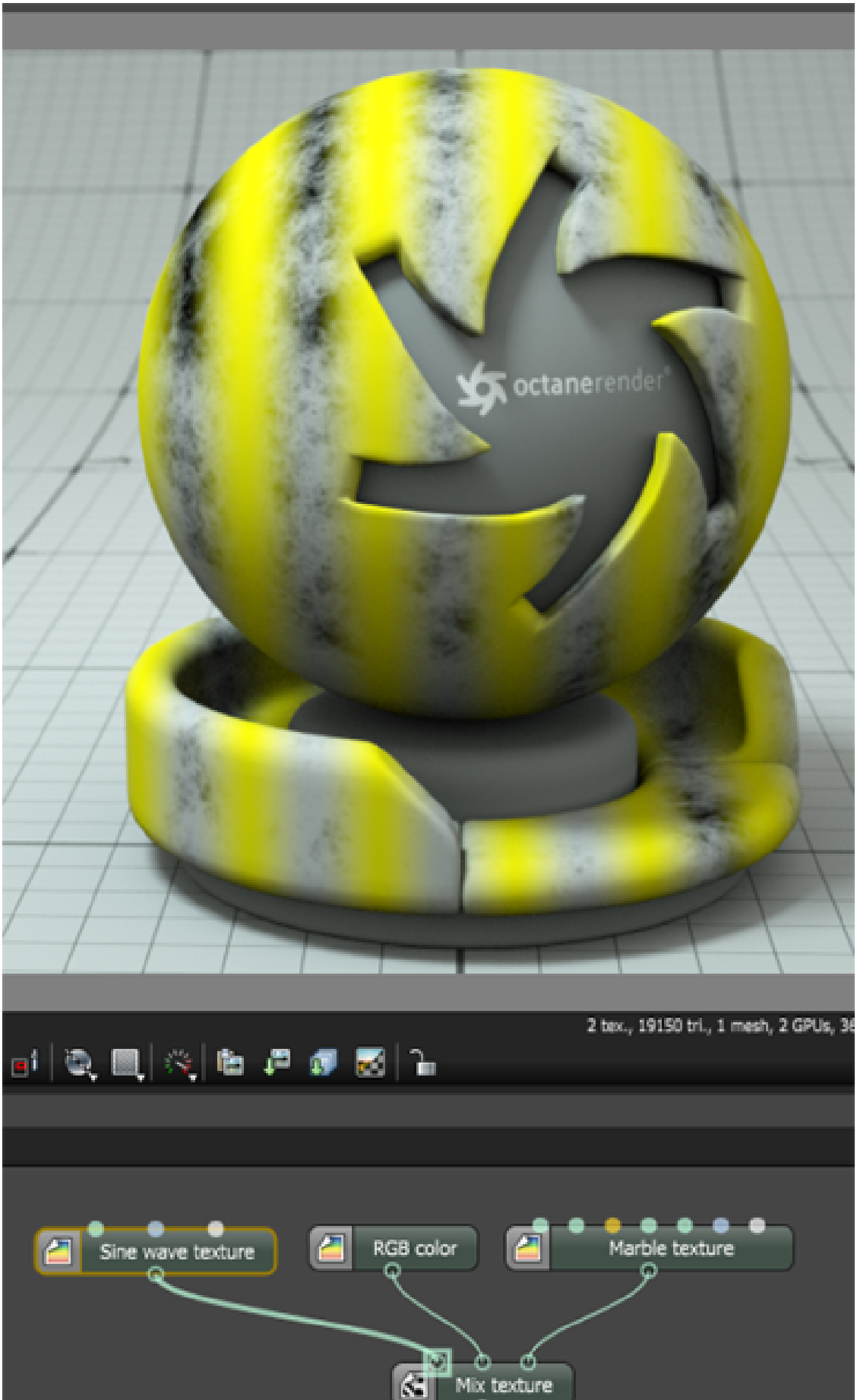

octanerender
2 tex., 19150 tri., 1 mesh, 2 GPUs,
Sine wave texture
RGB color
Marble texture
Mix texture

Figure 1: A yellow color is mixed with a Marble texture, and the mix amount is controlled by a Sine Wave texture

Multiply Texture

The **Multiply** texture multiplies the values of textures or colors together in an overlay fashion. Figure 1 shows a yellow **RGB Color** texture multiplied with a **Marble** texture.

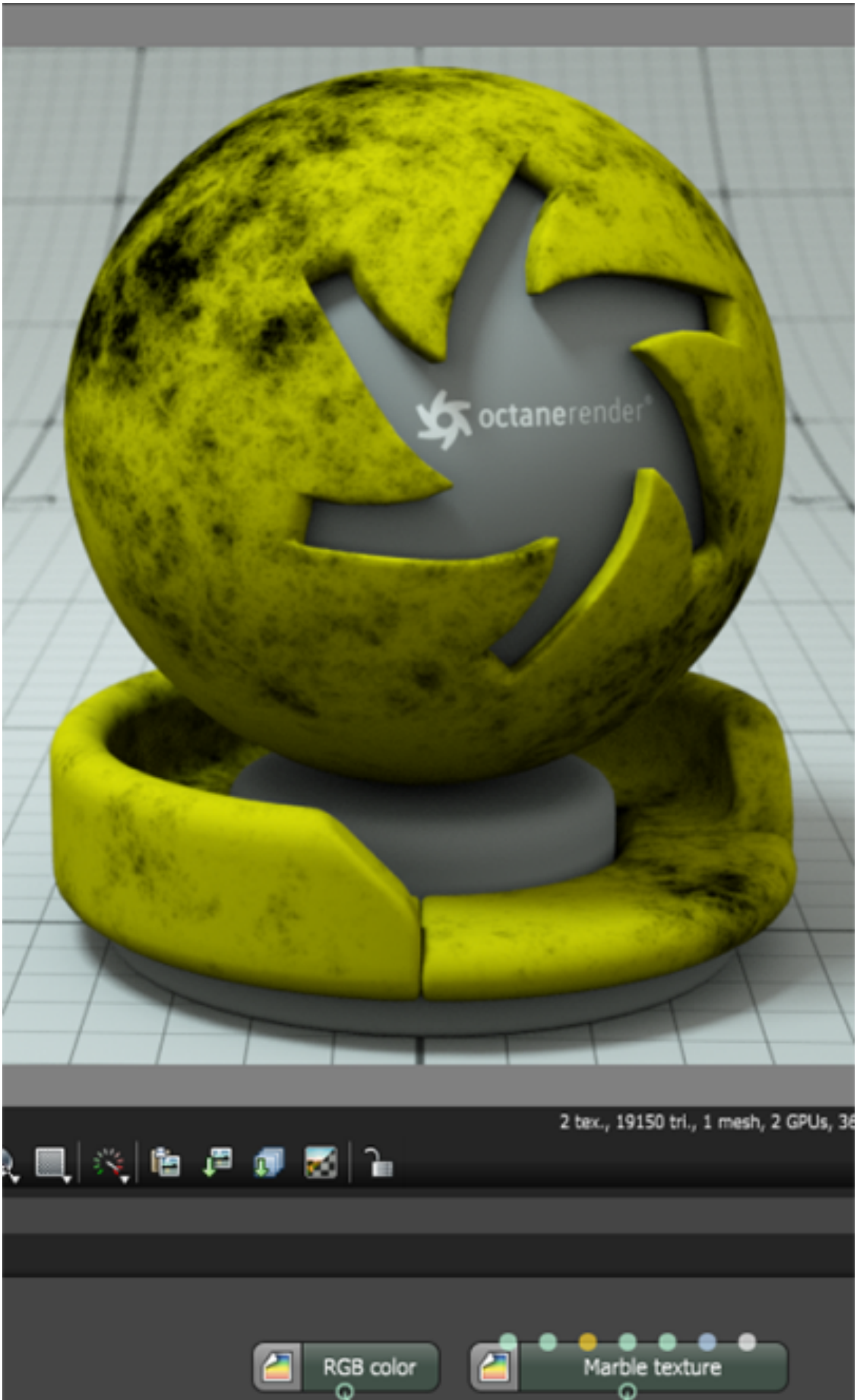

octanerender
2 tex., 19150 tri., 1 mesh, 2 GPUs, 3
RGB color
Marble texture

Figure 1 : A yellow RGB Color texture is multiplied against a Marble texture, and the result is connected to the Diffuse[1] channel of a Diffuse material[2]

1Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
2Used for dull, non-reflecting materials or mesh emitters.

Subtract Texture

The **Subtract** texture subtracts the value of one texture from another, similar to the Subtract layer mode in Photoshop®. Figure 1 shows a graph of the Subtract texture used to subtract a **Marble** texture from a yellow **RGB Color** texture.

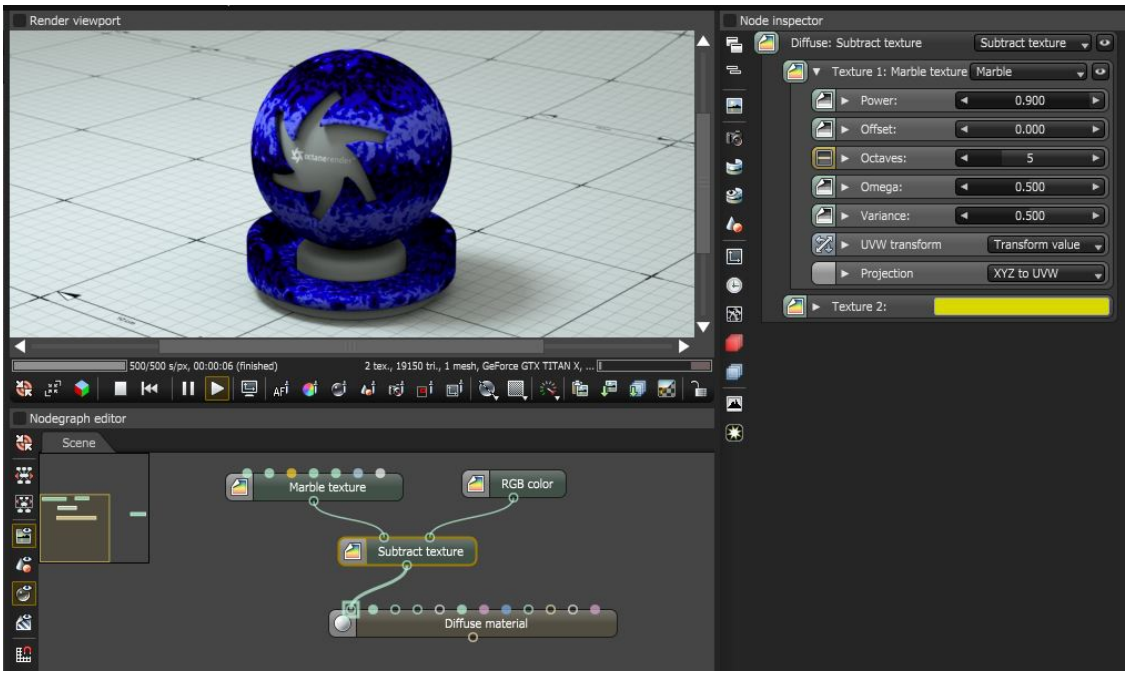


Figure 1: The subtraction of a yellow RGB Color texture from a Marble texture results in a blue color

Mappings

The **Mapping** node category contains nodes that can be used to alter or adjust both **Procedural** and imported texture maps. Mapping is found as a subsection of the **Textures**[1] section of the pop-up menu (Figure 1).

[1]Textures are used to add details to a surface. Textures can be procedural or imported raster files.

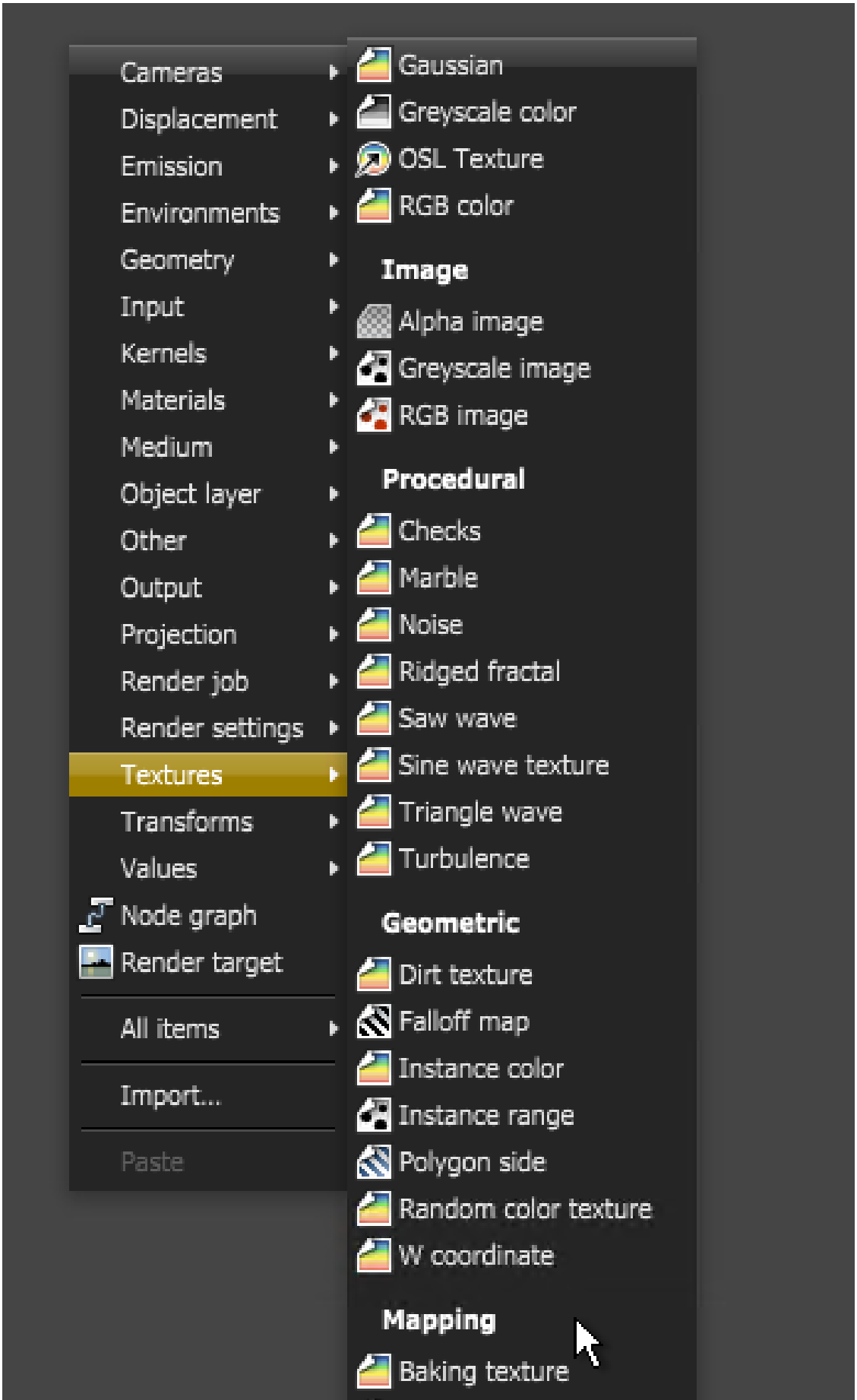

Cameras
Displacement
Emission
Environments
Geometry
Input
Kernels
Materials
Medium
Object layer
Other
Output
Projection
Render job
Render settings
Textures
Transforms
Values
Node graph
Render target
All items
Import...
Paste
Gaussian
Greyscale color
OSL Texture
RGB color
Image
Alpha image
Greyscale image
RGB image
Procedural
Checks
Marble
Noise
Ridged fractal
Saw wave
Sine wave texture
Triangle wave
Turbulence
Geometric
Dirt texture
Falloff map
Instance color
Instance range
Polygon side
Random color texture
W coordinate
Mapping
Baking texture

Figure 1: Mappings are a subcategory of Textures

Baking Texture

This texture allows you to bake an arbitrary texture into an image and make use of **Procedural** textures in displacement mapping. The baking uses the texture preview system, and then the texture appears like an image texture to the rest of the system. The baking is done whenever an input changes, and baking is calculated on-the-fly. The internal image is not stored in the project, so it needs recalculating whenever you load the project. You can access the Baking texture node by right-clicking in the **Nodegraph Editor** and selecting **Baking Texture** from the **Textures**[1] menu item (Figure 1).

[1] Textures are used to add details to a surface. Textures can be procedural or imported raster files.

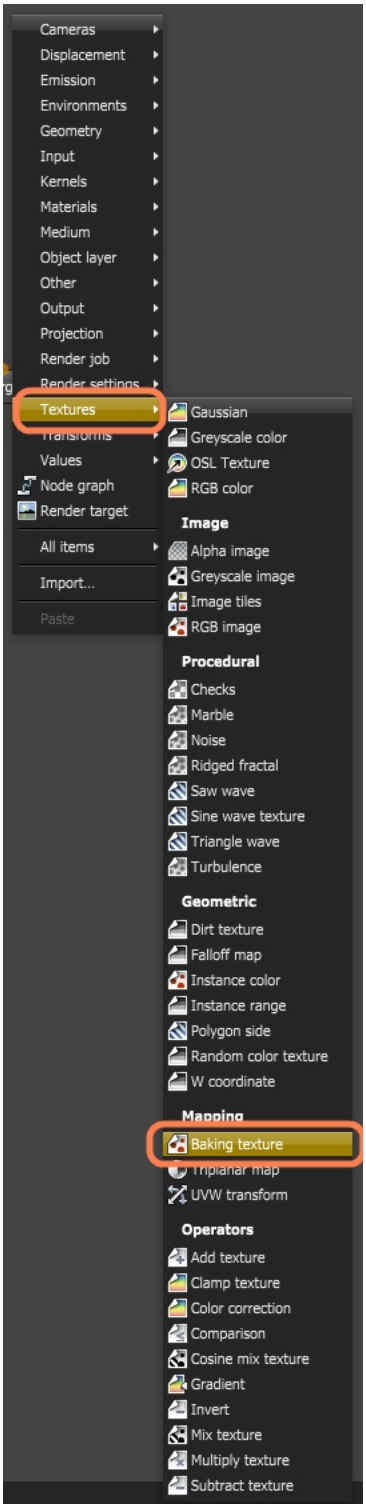
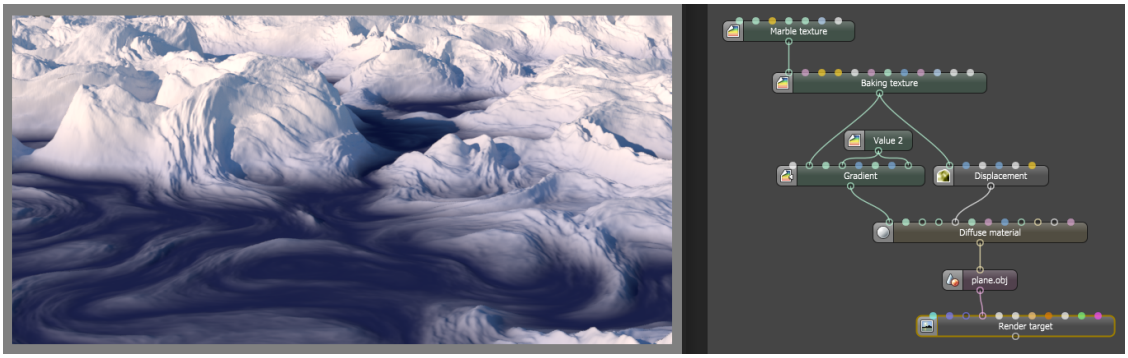

Cameras
Displacement
Emission
Environments
Geometry
Input
Kernels
Materials
Medium
Object layer
Other
Output
Projection
Render job
Render settings
Textures
Transforms
Values
Node graph
Render target
All items
Import...
Paste
Gaussian
Greyscale color
OSL Texture
RGB color
Image
Alpha image
Greyscale image
Image tiles
RGB image
Procedural
Checks
Marble
Noise
Ridged fractal
Saw wave
Sine wave texture
Triangle wave
Turbulence
Geometric
Dirt texture
Falloff map
Instance color
Instance range
Polygon side
Random color texture
W coordinate
Mapping
Baking texture
Triplanar map
UVW transform
Operators
Add texture
Clamp texture
Color correction
Comparison
Cosine mix texture
Gradient
Invert
Mix texture
Multiply texture
Subtract texture

Figure 1: Selecting the Baking Texture node from the pop-up context menu of the Node-graph Editor

With this Texture node, you can utilize the full power of Procedural textures and combine them with **Displacement**[1], which makes it easy to create alien landscapes like this:



In OctaneRender®, Displacement cannot utilize a Procedural texture map. This Texture node provides a way to bake Procedural textures into an image to use as a Displacement map.

In the following example, Displacement uses the **Turbulence** procedural texture by connecting it to the Baking texture node before connecting it to the Displacement node (Figure 2).

[1]The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

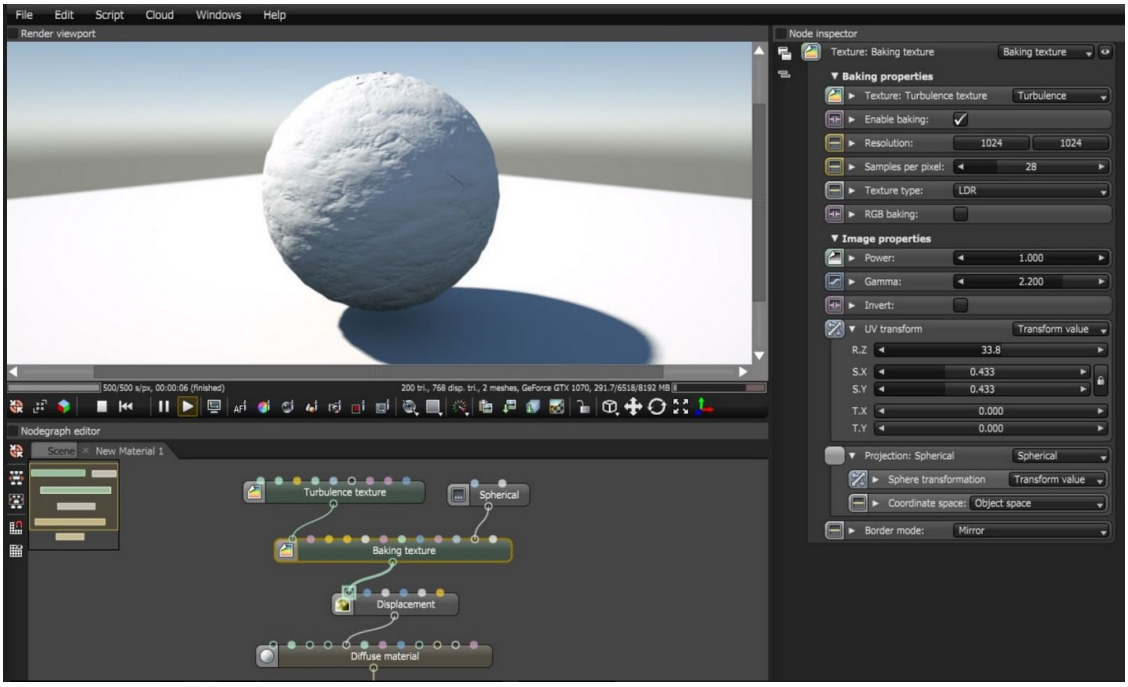
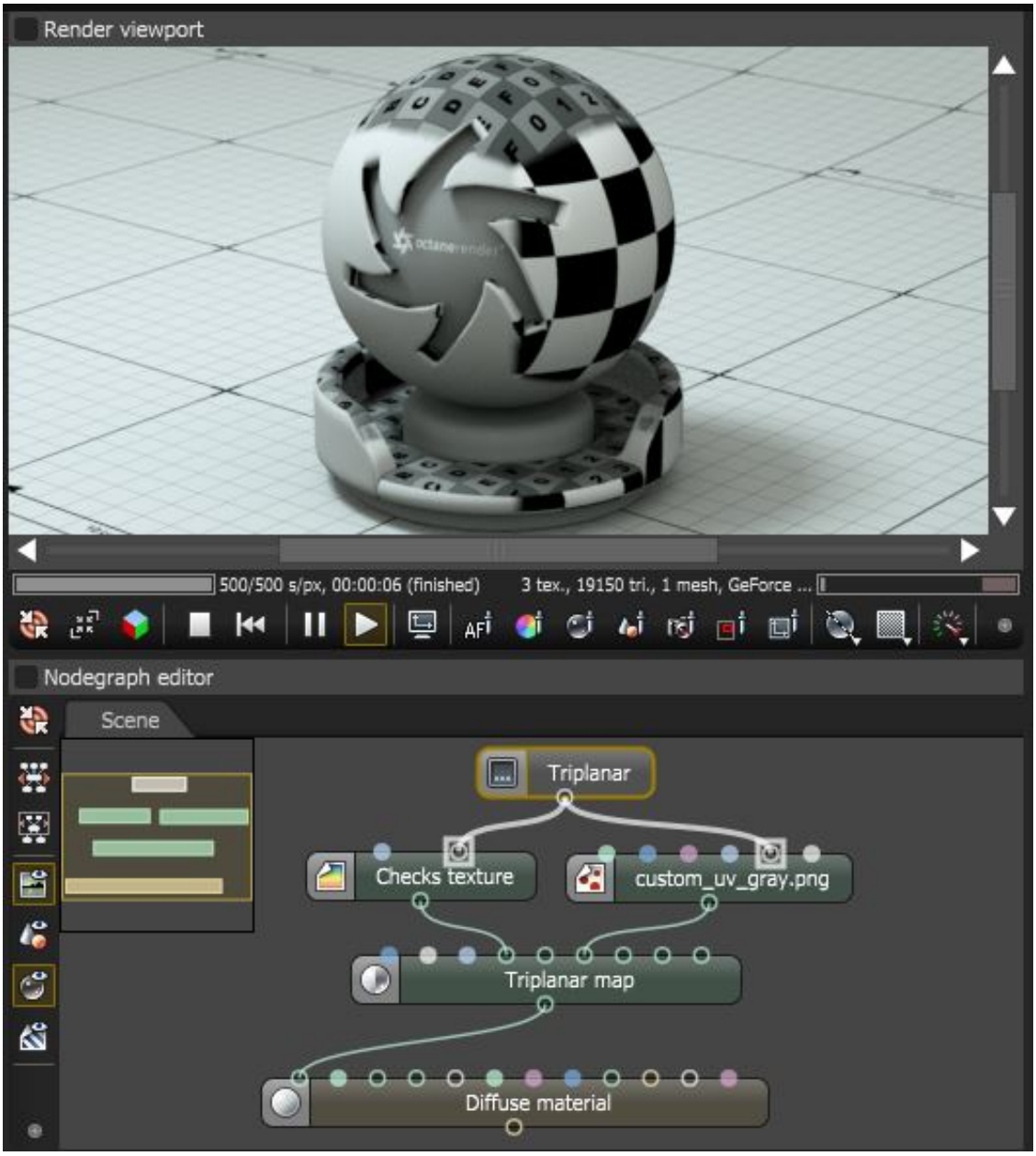


Figure 2: The Turbulence node used for Displacement by filtering it through the Baking texture node

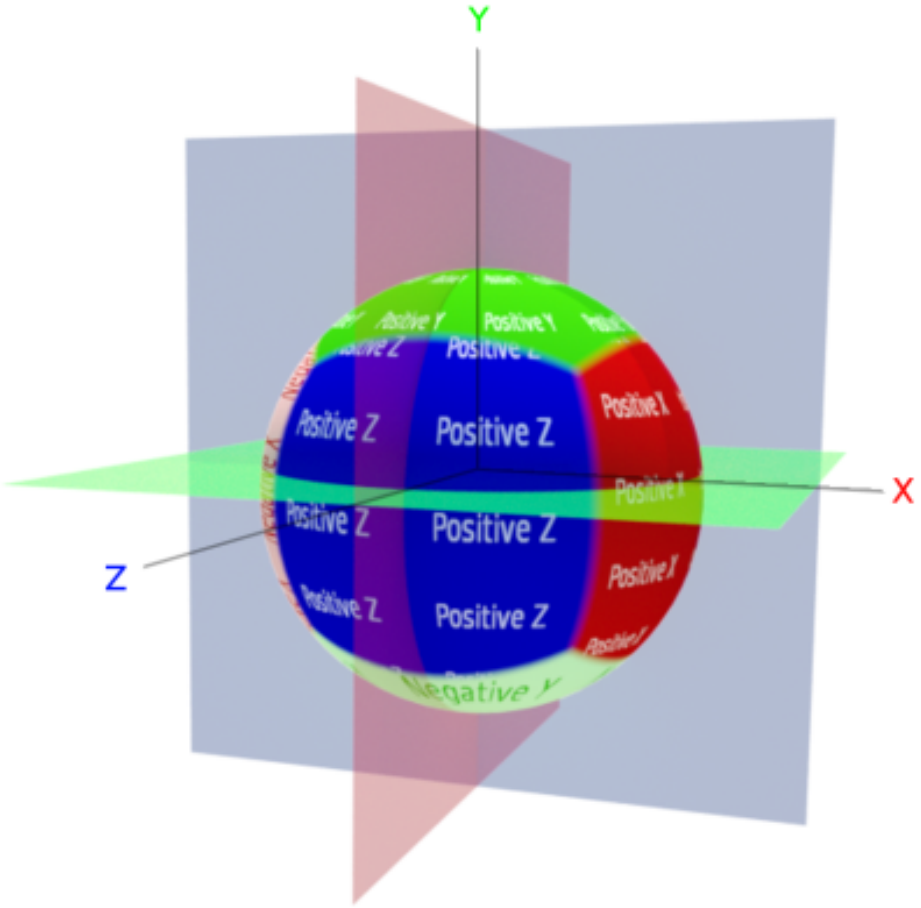
Triplanar Map

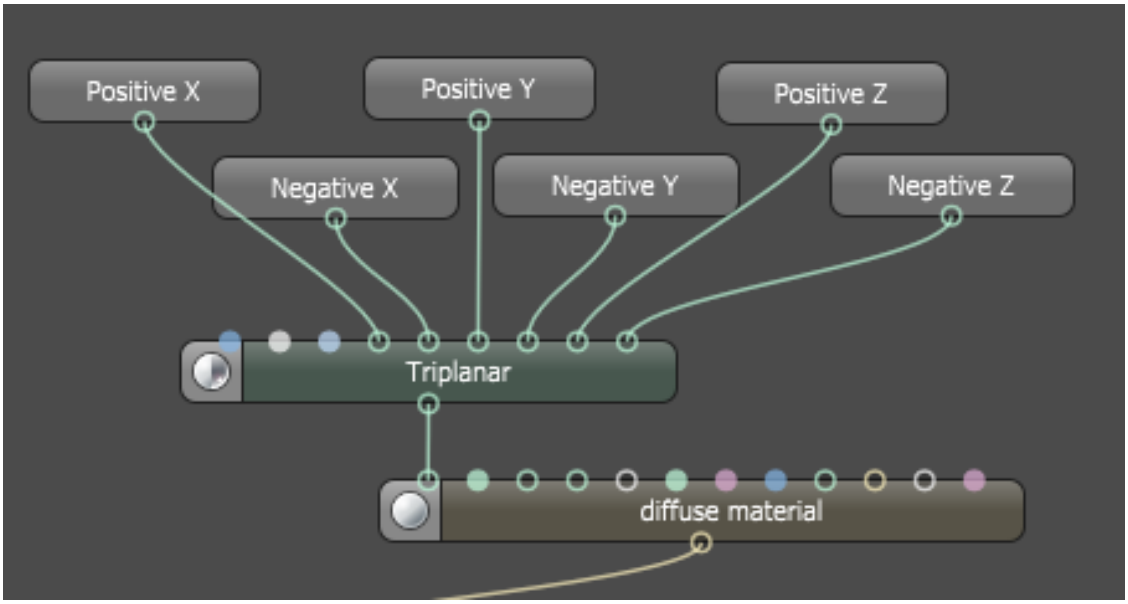
The **Triplanar Map** texture works in conjunction with a **Triplanar** projection. The Triplanar projection takes the coordinates in world or object space and it picks the projection axis, depending on the active axis of the Triplanar Map. This gives a quick way to map a texture on any object, and presents the possibility for texture transforms local to each projection axis. The Triplanar Map has six input pins representing the positive and negative X, Y, and Z planes. The same or different Texture nodes can map to each of these input pins.



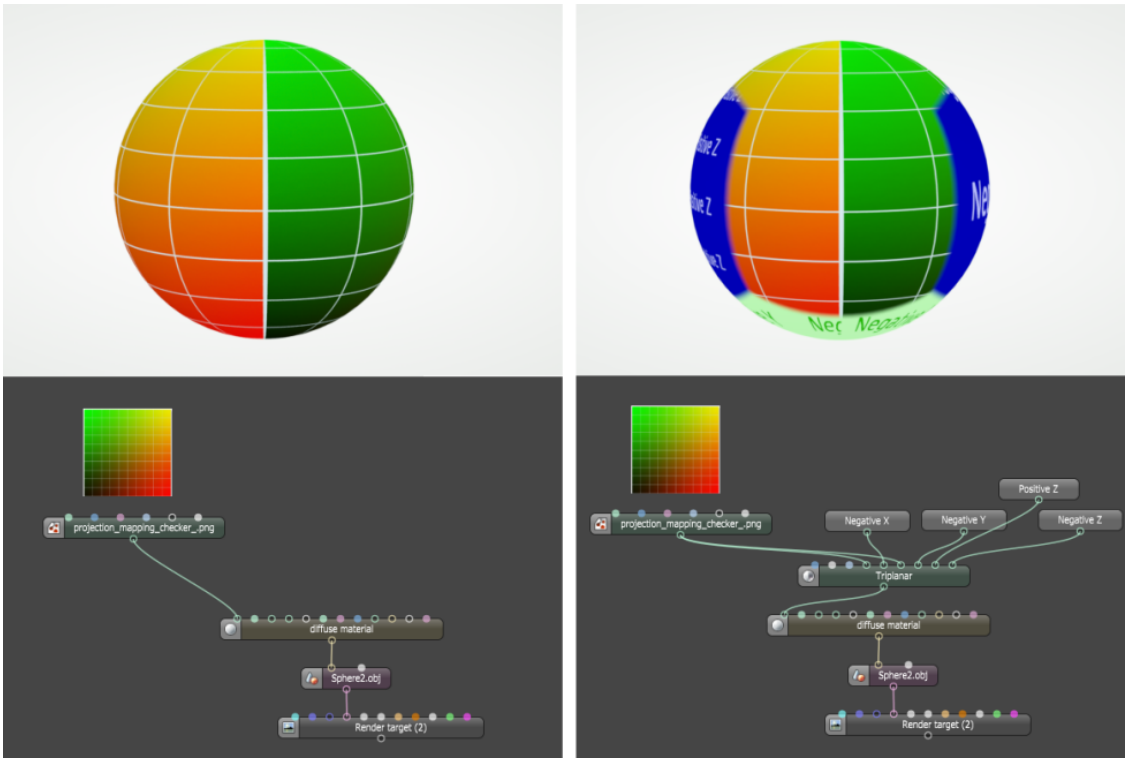
The Triplanar Map and the Triplanar projection are mapping a Check texture and an imported texture to different projection planes of an object

This texture maps the samples of multiple textures along three planes — x, y, and z — in world space or object space coordinates, and blends them to create one seamless texture. In most cases, and depending on the complexity of the model, it generally allow users to map textures without having a UV-mapped mesh.

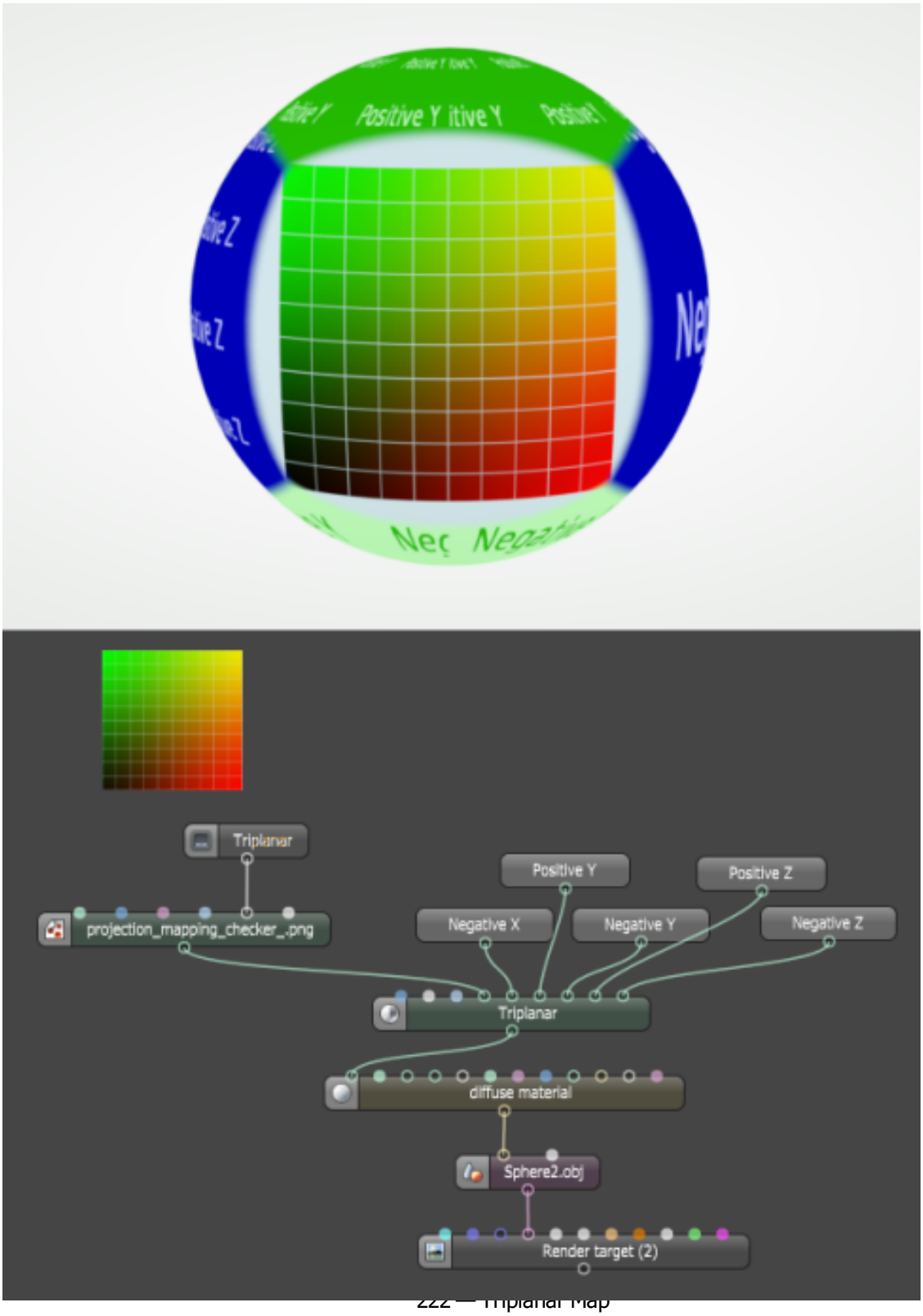




The Triplanar Map divides a material map into six areas corresponding to the x, -x, y, -y, z, and -z axes. Initially, a texture would cover the entire surface of the object, but the triplanar mapping confines visibility of the texture map onto the corresponding axes that are active for that texture. The illustration below compares an image without the triplanar mapping versus one that is plugged into the **Positive X** and **Positive Y** axis pins of a Triplanar Map node.

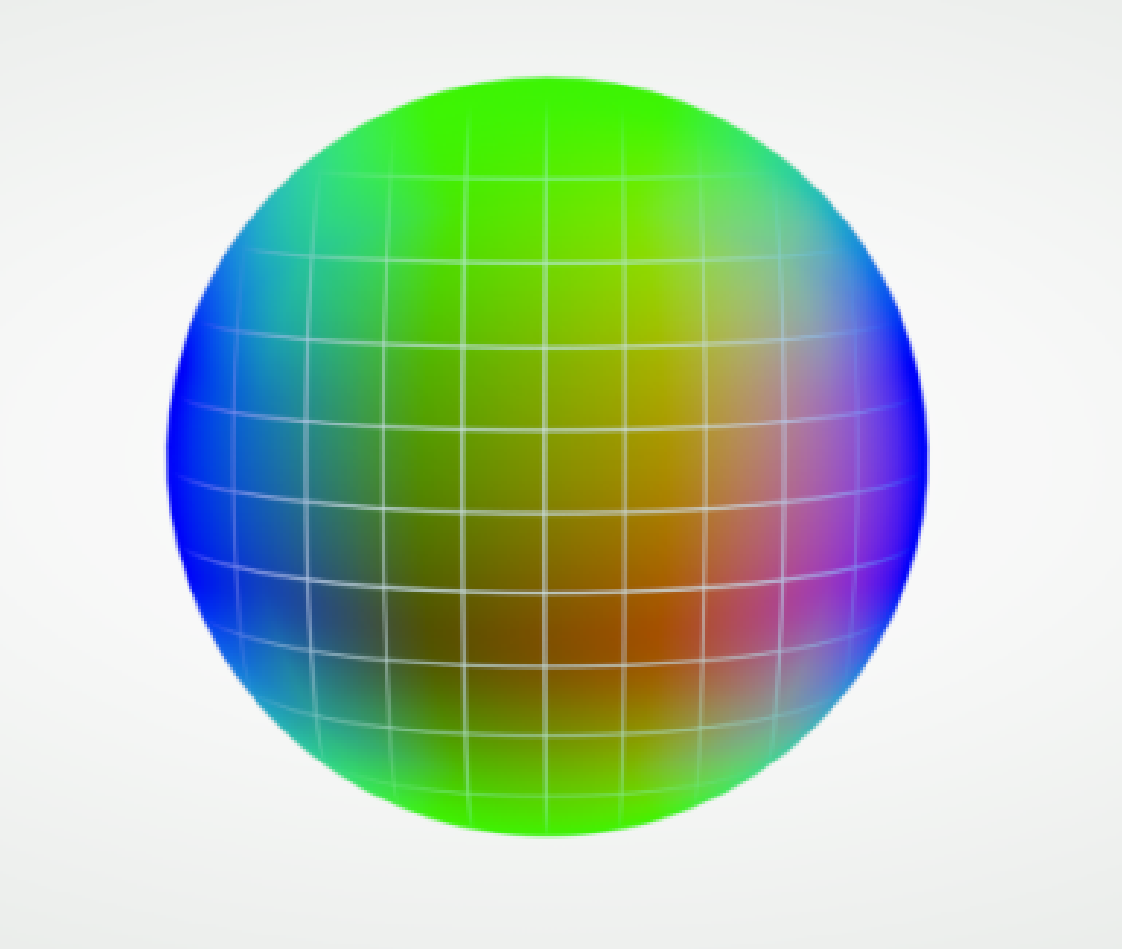


The Triplanar projection can localize the projection of the texture to a corresponding plane and allow texture UV transforms relative to that projection axis.



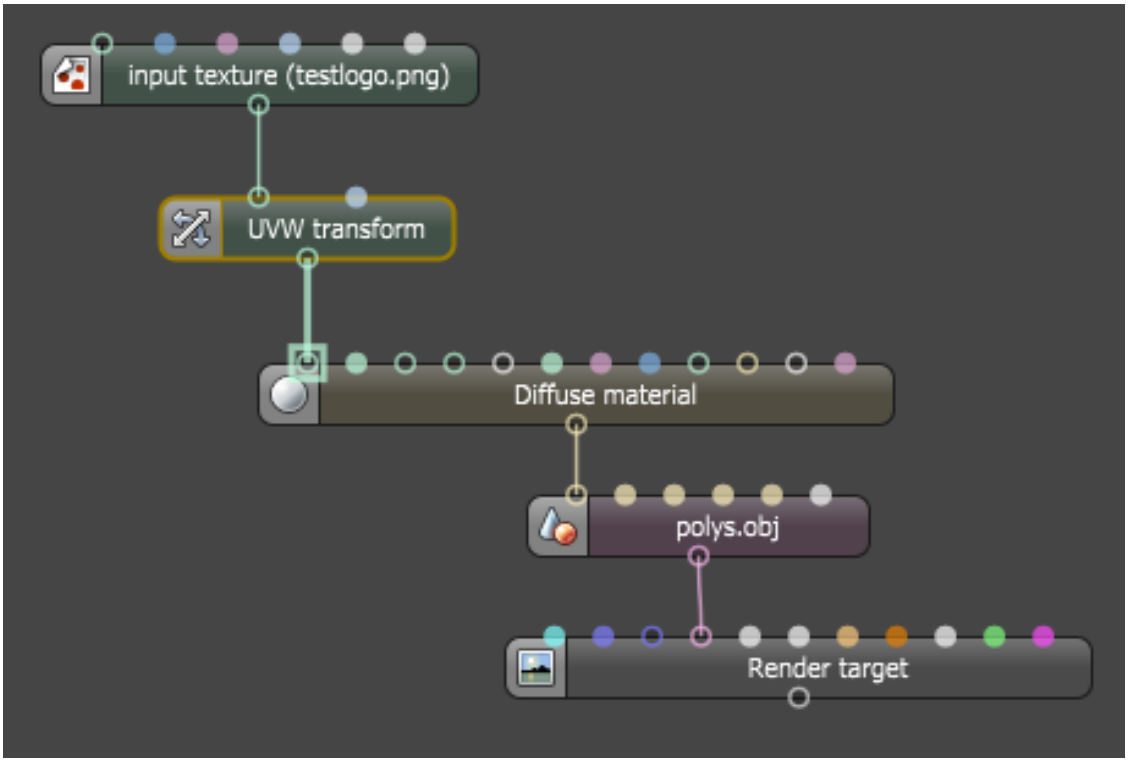
222 — Triplanar Map

You can adjust the Triplanar Map's **Blend Angle** and **Blend Cube Transform** parameters to soften the seams.

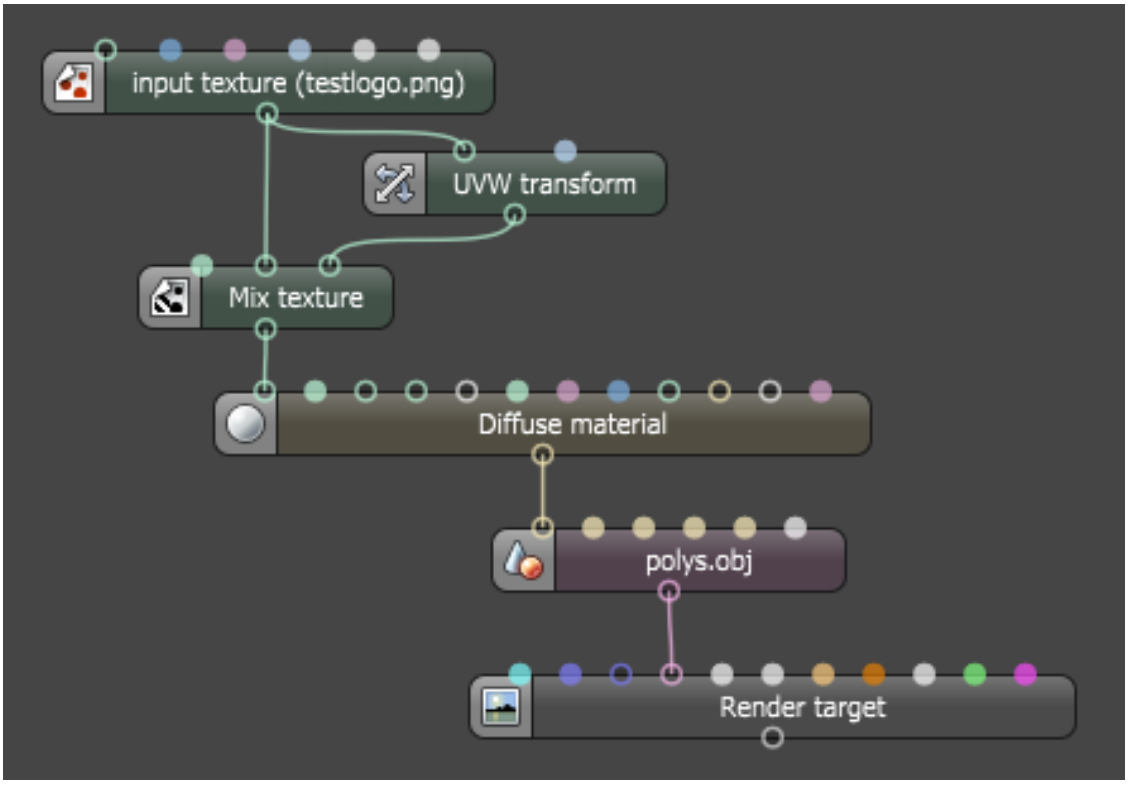


UVW Transform Texture

The **UVW Transform** texture takes an **Input Texture** and applies a map to transform the Input Texture's UV layout on top of the its own UV coordinate transformation.

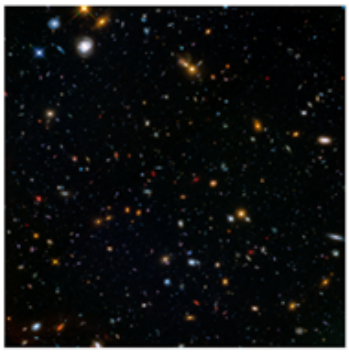


It is also applied to the input texture's **UVW Transform** parameter, and this concept becomes more useful when used in conjunction with other mapping textures to combine different scales/orientations/translations of that same texture to create a larger detail range.

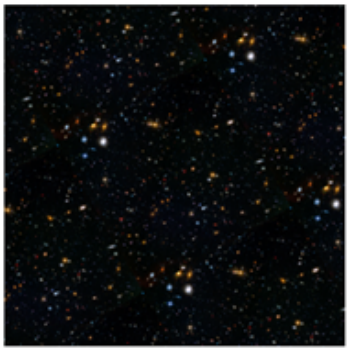


The UVW Transform texture can be used in conjunction with other mapping textures like the **Triplanar Map** texture, **Mix** texture, **Cosine Mix** texture, logical texture maps like **Comparison**, or arithmetic texture maps like **Add**, **Subtract**, and **Multiply**.

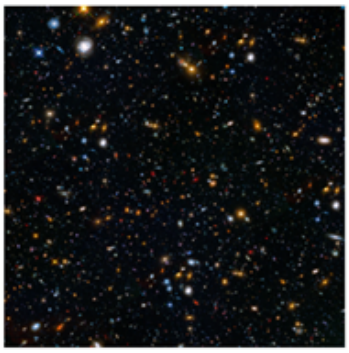
Here is an example where there is a need to combine different scales/orientations/translations of the same texture to create a larger detail without creating obvious patterns:



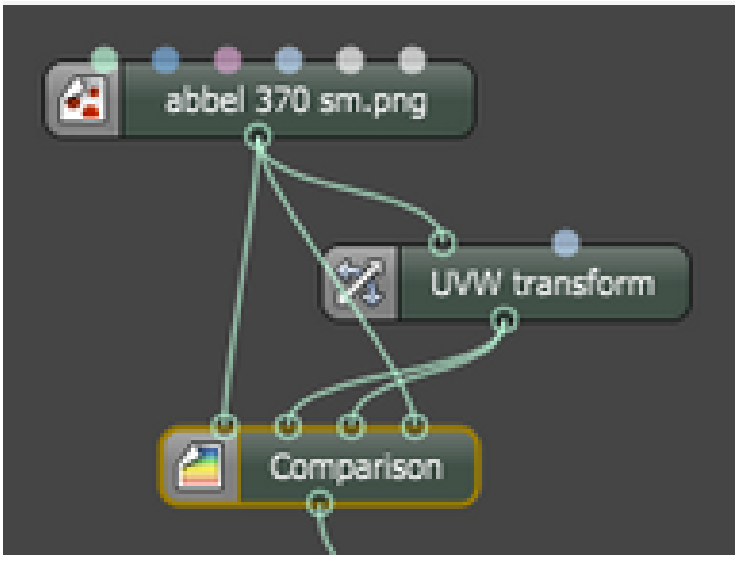
This is the original image texture.



The same image texture scaled down by a factor of 2 using the UVW transform texture map, however, this results in a repetitive pattern that is easy to spot



Overlaying these two with a Comparison texture makes the repetitive pattern less obvious



Displacement

Displacement[1] mapping utilizes a 2D texture map in order to generate 3D surface relief. As opposed to **Bump** and **Normal** mapping, Displacement mapping not only provides the illusion of depth, but it also displaces point positions over the surface based on light and dark values of the Displacement texture. The OctaneRender® Displacement node controls how the texture displaces the surface. Displacement mapping requires a **UV** projection for the object with the displacement. Models created in other 3D applications (ZBrush®, Mudbox®, Maya®, etc.) need UV texture coordinates, and the Displacement map texture should match the model's UV layout. Procedural textures will not work for Displacement in OctaneRender - only **Image** textures will work.

You can find the Displacement node by right-clicking in the **Nodegraph Editor** and navigating to the **Displacement** category (Figure 1).

[1]The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

Figure 1: Creating a Displacement texture

The **Displacement** node connects to the **Displacement** input pin of a **Material**[1] node. The Displacement texture (typically a **Grayscale Image** node) then connects to the **Texture** pin of the Displacement node (Figure 2).

[1]The representation of the surface or volume properties of an object.

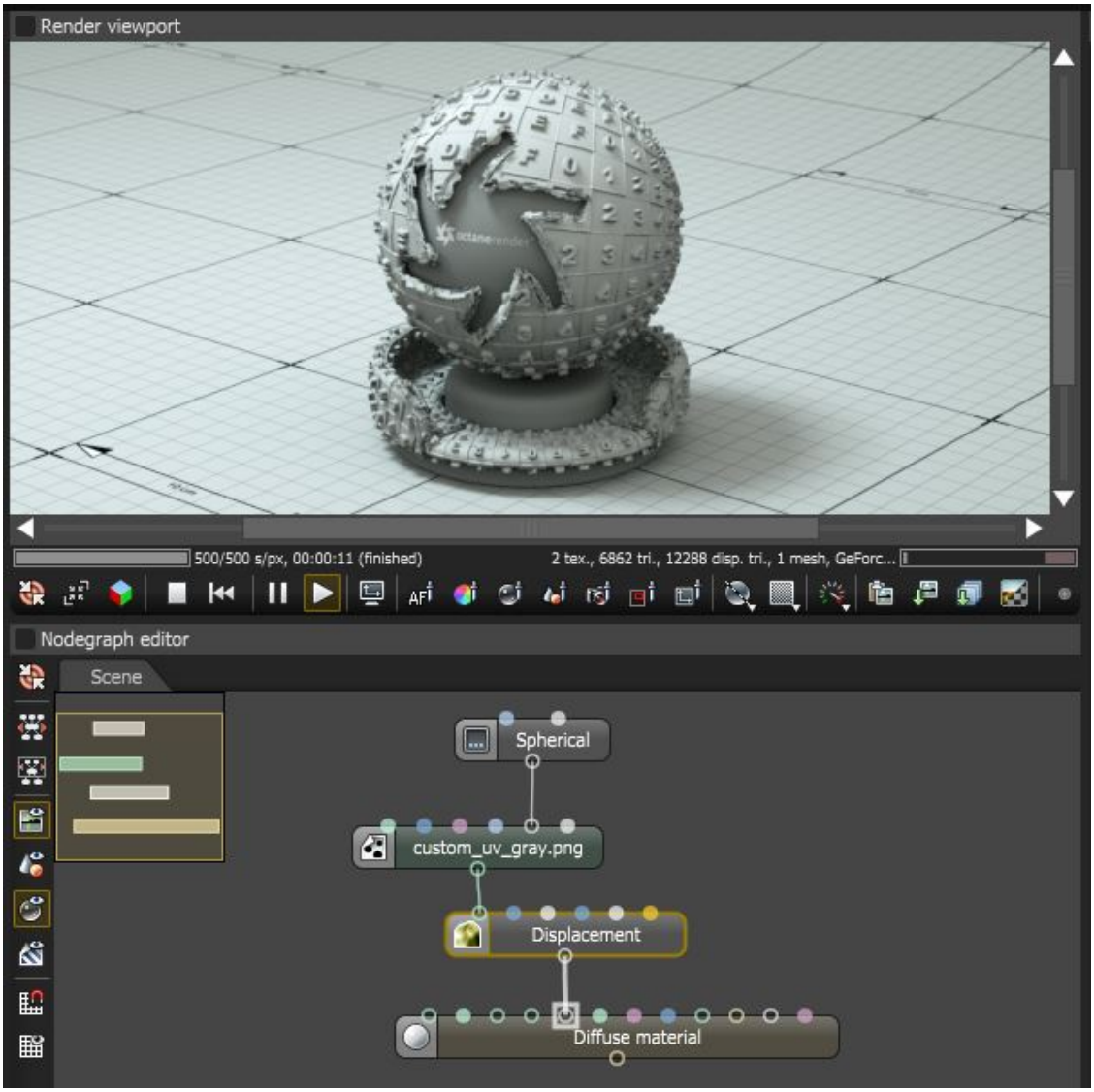


Figure 2: A Material graph with Displacement applied to a Mesh

Displacement Parameters

- **Texture** - This slot provides the Displacement map path. Displacement maps are image textures generated in programs like ZBrush, 3D Coat, Substance Designer, or Photoshop®.

- **Mid Level** - Defines the Displacement shift in texture value range. Set this value to **0.5** for image textures that use 50% to represent no displacement, or for images (such as 32-bit EXRs) that use black to represent no displacement, set this value to **0**. If you use a digital sculpting program like ZBrush to generate Displacement, you can get the best results by setting Mid Level in the sculpting program to **0.5** when it generates Displacement, and then set the Displacement node's Mid Level value to **0.5**.
- **Level Of Detail** - Adjusts map detail quality. Higher values reduce artifacts seen in shadows cast on the Displacement surface and brings out finer details, but it increases render time.
- **Height** - Controls Displacement strength.
- **Displacement Direction** - Lets you choose different Displacement vectors.
- **Filter Type** - Selects the Displacement map filter.
- **Filter Radius** - Adjusts the number of nearest pixels to use for filtering. Higher values result in smoother Displacement maps. This parameter is valid if you enable a **Box** or **Gaussian** filter.

> **Note:** Calculating Displacement geometry places additional **GPU**[1] load. High or low Displacement values (Height) causes issues and GPU errors. Displacement mapping emphasizes details in the scene's textural aspects rather than major features that the geometry (e.g., rugged terrains, large crevices) should provide. You can't use Displacement and a Normal map together on the same material - this results in rendering artifacts on the material. This also applies to Bump maps, but the digital artifacts are less pronounced.

[1]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Projections

Projection nodes can add or edit existing UV mapping coordinates of a texture map. These nodes do not alter the actual 3D model's UV coordinates - they simply apply a new UV projection to its connected texture in the **Nodegraph Editor** (Figure 1).

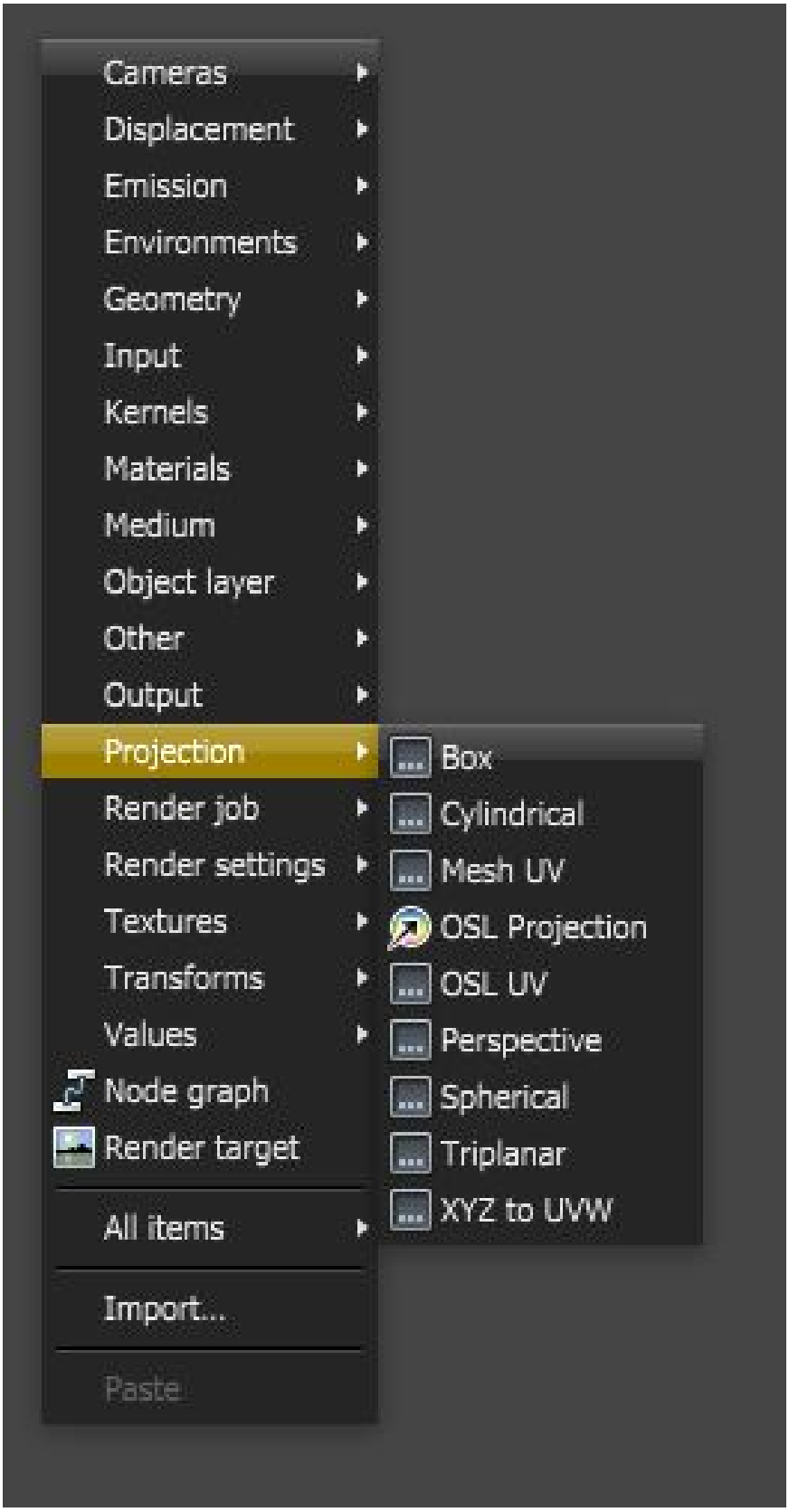

Cameras
Displacement
Emission
Environments
Geometry
Input
Kernels
Materials
Medium
Object layer
Other
Output
Projection
Render job
Render settings
Textures
Transforms
Values
Node graph
Render target
All items
Import...
Paste
Box
Cylindrical
Mesh UV
OSL Projection
OSL UV
Perspective
Spherical
Triplanar
XYZ to UVW

Figure 1: Accessing the Projection nodes in the Nodegraph Editor

Box

The **Box** projection is an extension of **XYZ To UVW** mapping - the projection of the planes of the cube are based on the normal direction of the surface. Box projection provides a quick way to map a texture on any object without too much distortion. However, the seams between the projection planes of the box may be visible in the render, depending on the shape of the surface (Figure 1).

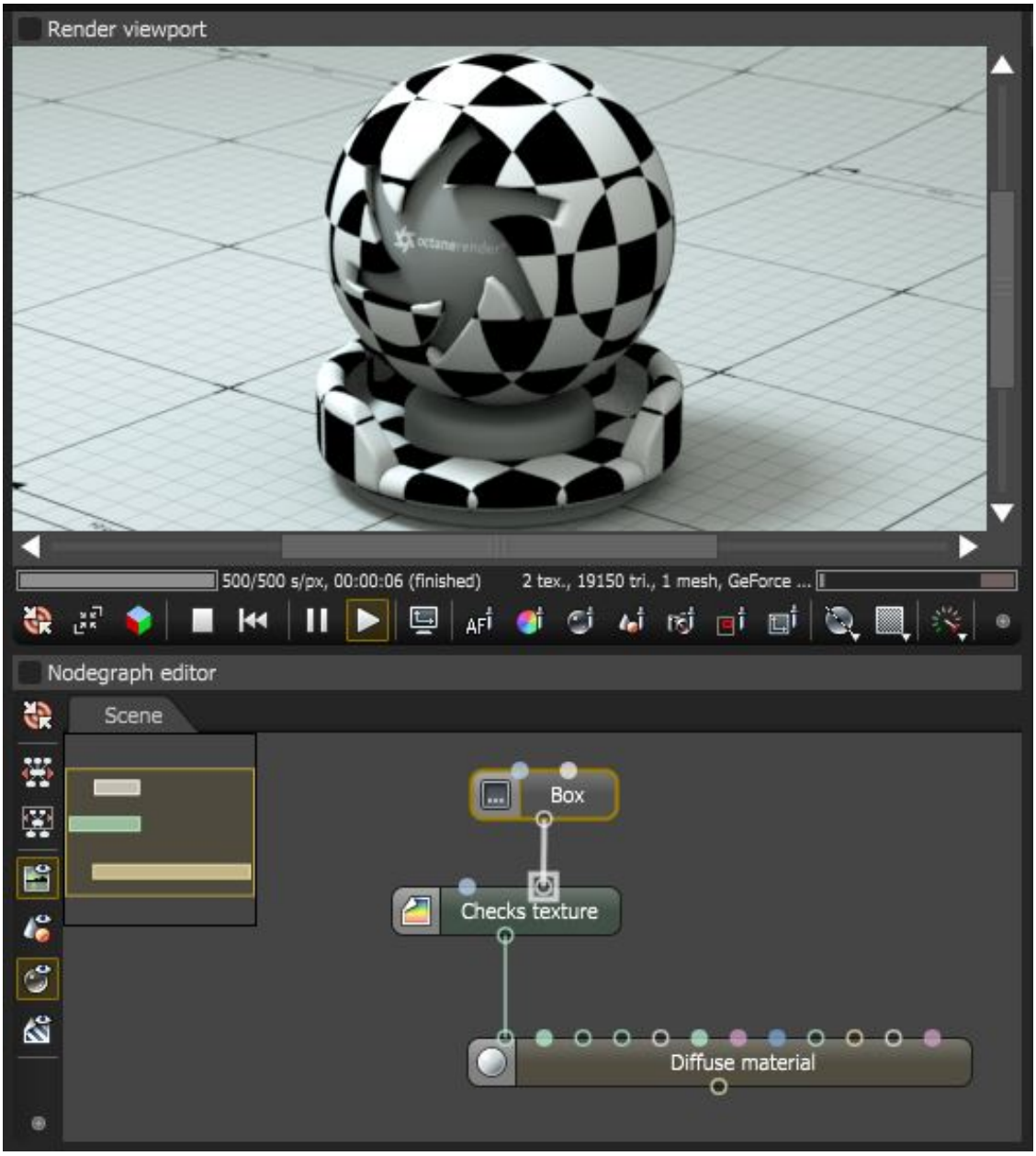


Figure 1: The result of using Box projection applied to a Checks texture node

Cylindrical

Cylindrical projection wraps texture maps on a surface with a cylindrical shape. Cylindrical projection provides a quick way to map a texture on roughly cylindrical-shaped surfaces without too much distortion. However, the seams of the texture may be visible in the render, depending on the shape of the surface (Figure 1).

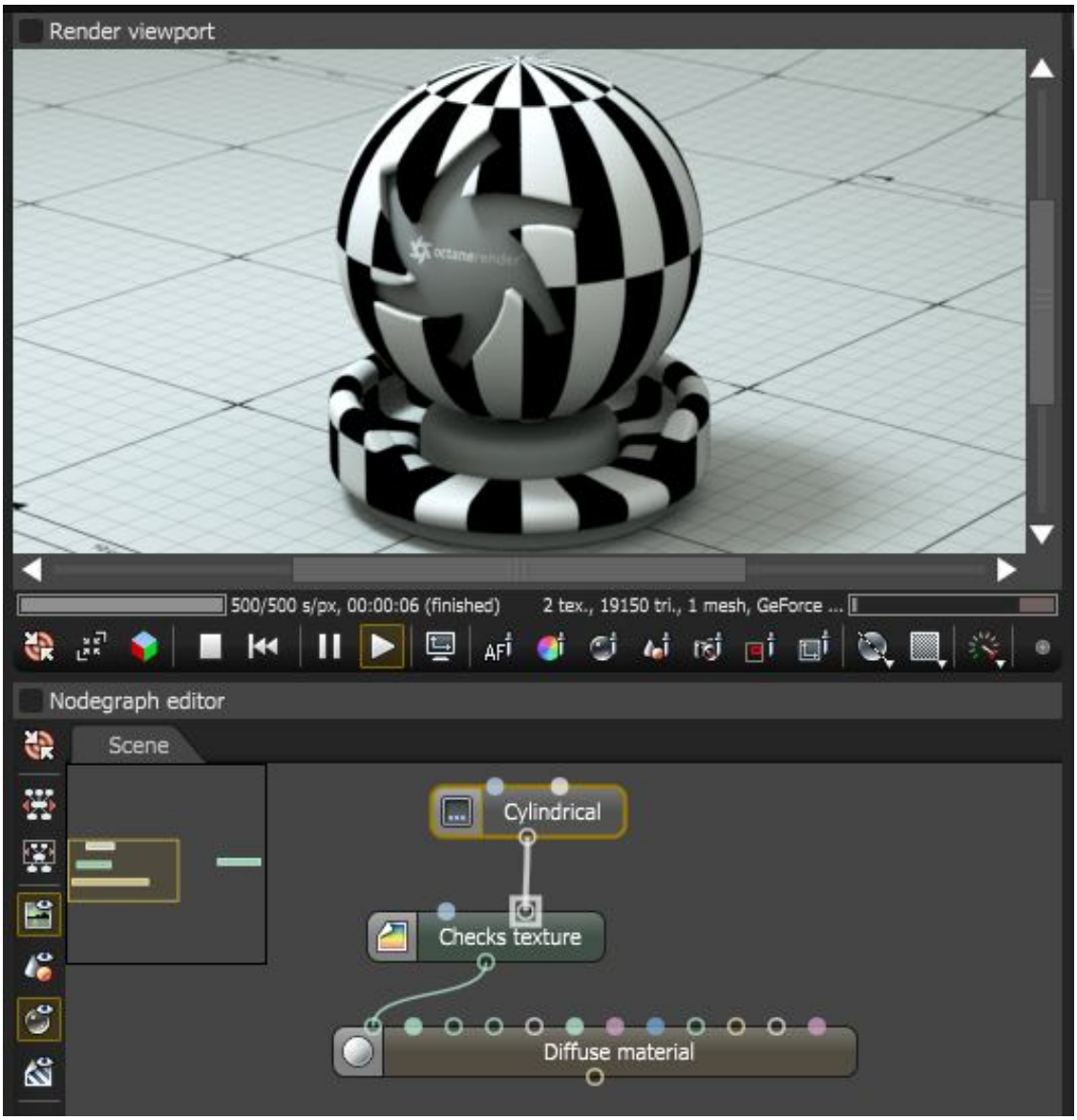


Figure 1: The result of using Cylindrical projection with the Checks texture node

This projection performs cylindrical mapping where the U coordinate is the longitude, and the Y coordinate is the world space Y coordinate. For **Images**, the mapping on the Y axis maps the image to the [-1, 1] interval. For **Procedural** textures, the W coordinate is the distance from the Y axis. For points on the ground plane (Y=0), cylindrical and spherical mappings now map to the same points on the images, or what would be the equator on spherical mapping.

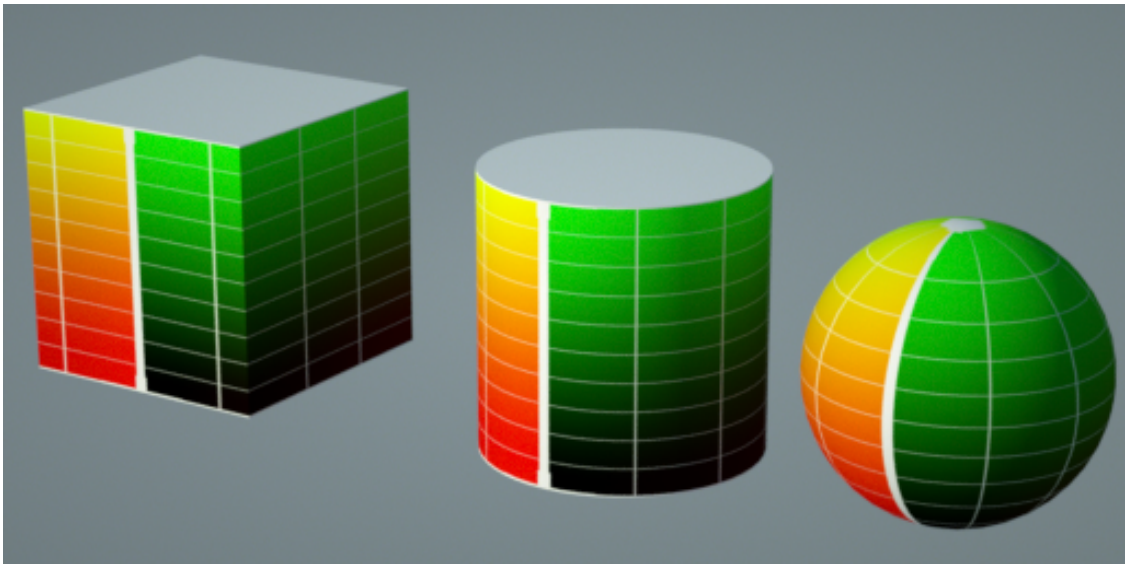


Figure 2: Cylindrical projections on a box, cylinder, and sphere

Mesh UV

The **Mesh UV** projection uses the mesh's UV coordinates to map the texture to the surface. This is the default behavior for all textures, so in many cases, it's unnecessary to use a **Projection** node when mapping a texture based on UVs.

This projection applies a spherical mapping for **Environment** textures and **IES**[1] light distributions. For more control over the projection (mainly rotation), use the **Spherical** projection in these cases.

[1]An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.

Perspective

Perspective mapping takes the world space coordinates and divides the X and Y coordinates by the Z coordinate (Figure 1). This is a useful way to model a projector (i.e., by using a texture with this projection as the distribution, with black border mode). You can also use it for camera mapping.

The same change as the other projections applies here: the image is mapped to (-1, -1) – (1, 1), so by default, offsets are not needed to use this mapping for projectors or camera mapping.

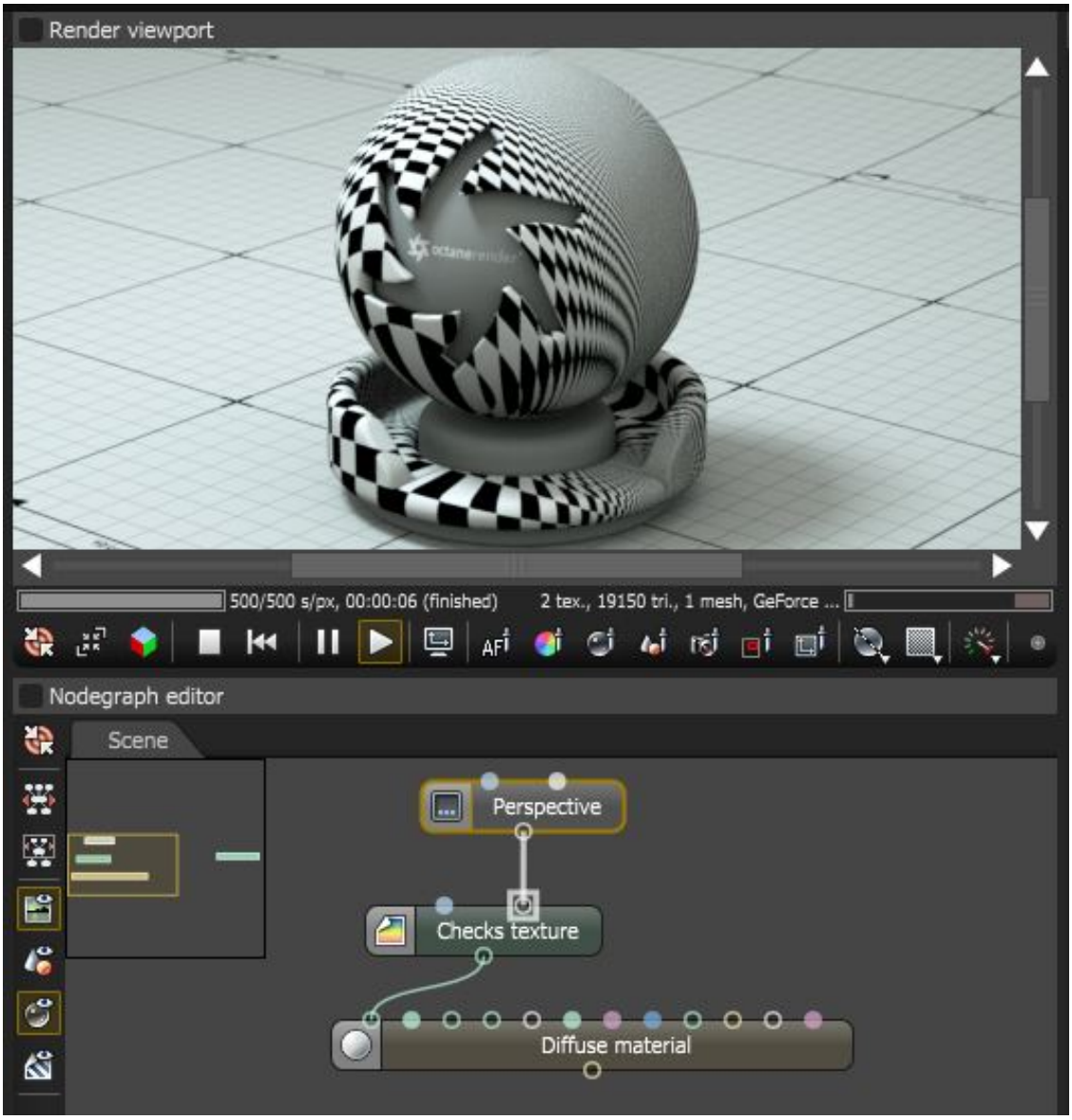


Figure 1: The Perspective projection node orienting a Checks texture

This projection takes the world space coordinates and divides the X and Y coordinates by the Z coordinate. This is useful if you want to model a projector (i.e., use a texture with this projection as the distribution, with black border mode). It is also useful for camera mapping. The image is mapped to (-1, -1) – (1, 1), so by default, an offset is not necessary to use this mapping for projectors or camera mapping.

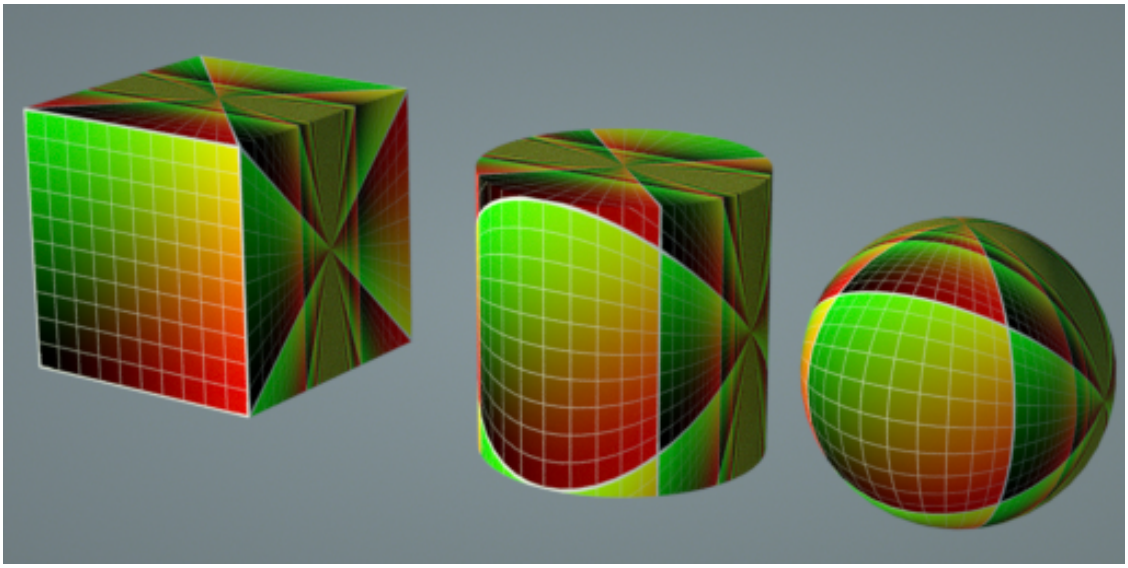


Figure 2: Perspective projection applied to a box, cylinder, and sphere

Spherical

The **Spherical** projection is used for **Environment** textures and **IES**[1] light distributions. It performs latitude-longitude mapping for the U and V coordinates. When used with **Procedural** textures, the W coordinate is the distance from the origin (Figure 1).

[1] An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.

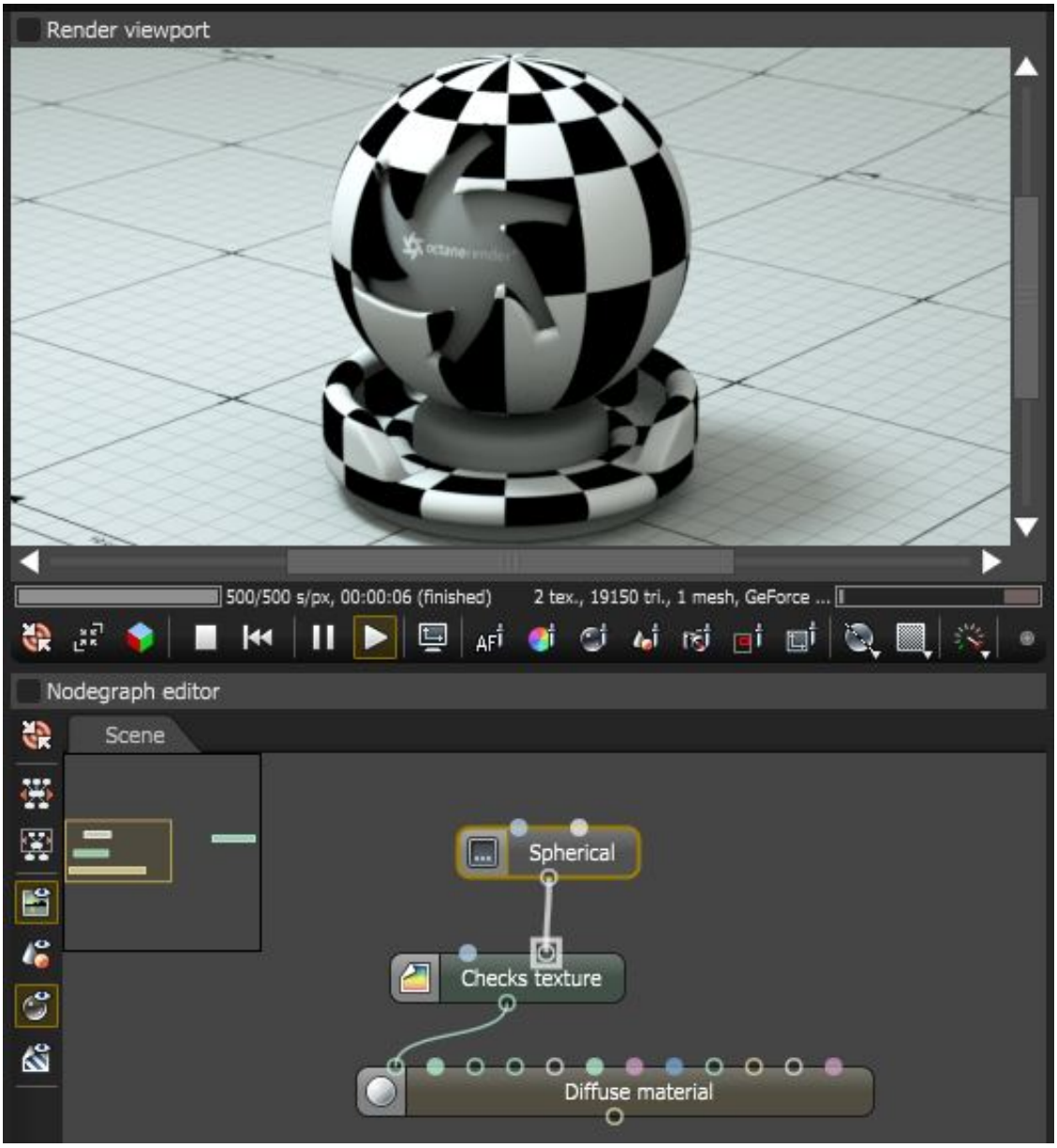


Figure 1: The Spherical projection used with the Checks procedural texture

When using an **Image** texture to light a scene with the OctaneRender® **Environment** node, spherical mapping combined with a **Transforms** node allows for the rotation and translation of the environment sphere's texture.

This projection is mostly used for Environment textures and IES light distributions. It performs latitude-longitude mapping for the U and V coordinates, and for **Procedural** textures, the W coordinate is the distance from the origin. To rotate a texture image (e.g., an HDR image) around a vertical axis, switch the projection of the texture environment image to **Spherical** and rotate it via the Y axis through the **Sphere Transformation** sliders.

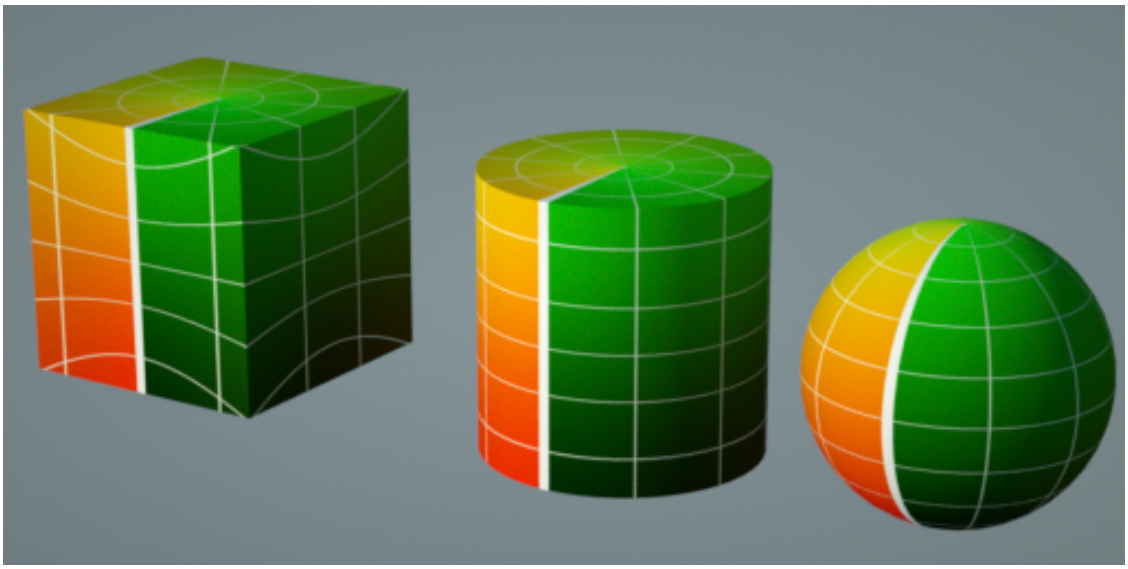


Figure 2: Spherical projection applied to a box, cylinder, and sphere

Triplanar

The **Triplanar** projection works in conjunction with a **Triplanar** map. It takes the coordinates in world or object space and will pick the projection axis depending on the active axis of the Triplanar map node. This gives a quick way to map a texture on any object, and presents the possibility for texture transforms local to each projection axis. The Triplanar map node has six input pins representing the positive and negative X, Y, and Z planes. You can map the same or different texture nodes to each of these input pins. (Figure 1).

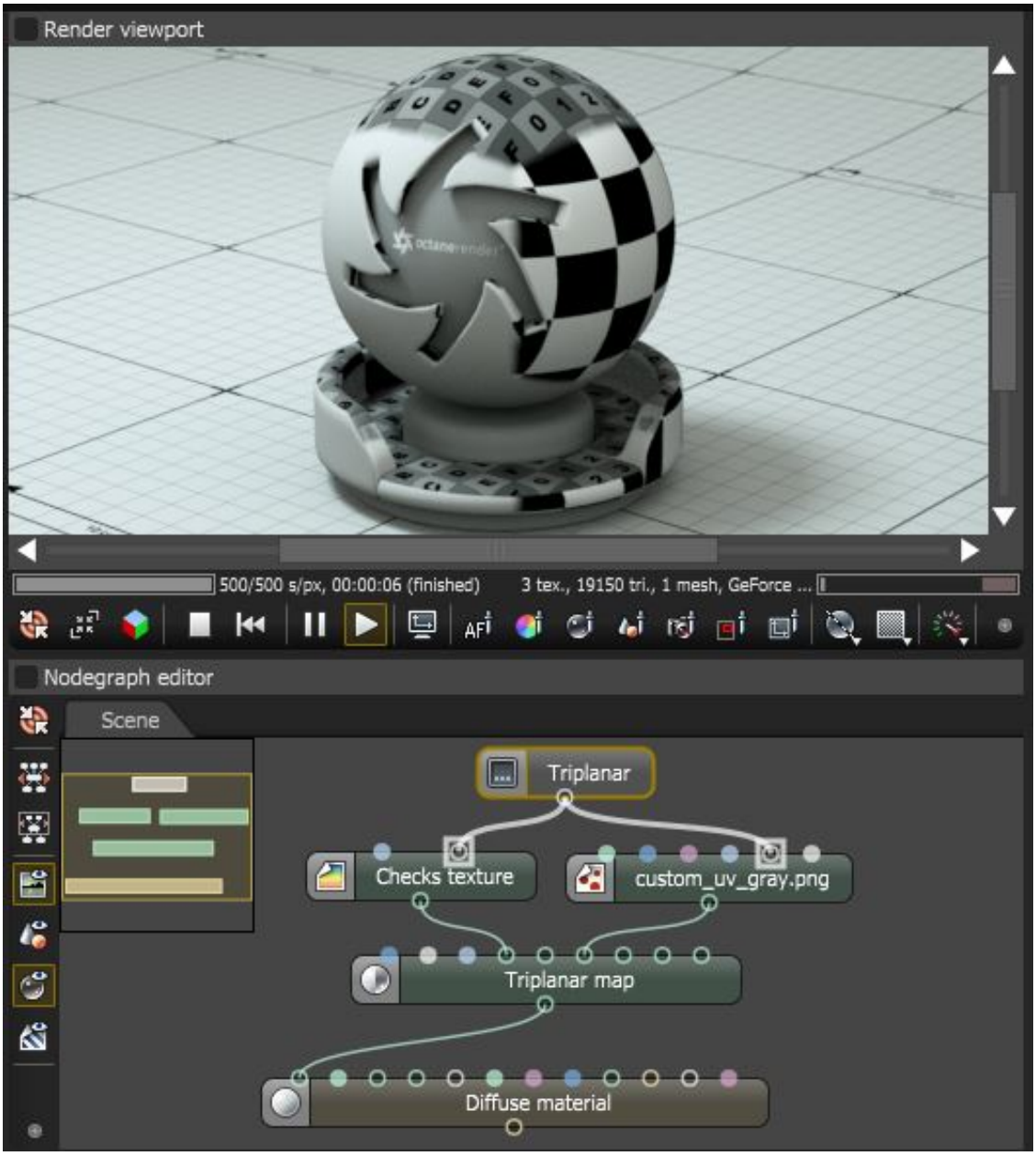


Figure 1: The Triplanar map and the Triplanar projection map a Check texture and an imported texture to an object's different projection planes

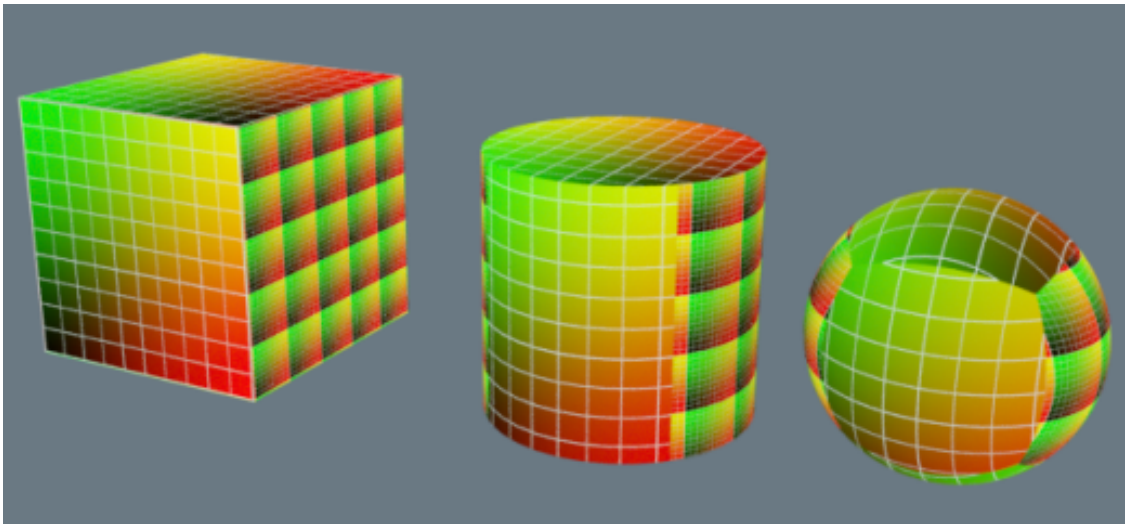


Figure 2: Triplanar projection applied to a box, cylinder, and sphere

XYZ to UVW

XYZ To UVW is also known as planar projection or flat mapping. This mapping type takes the coordinates in world or object space and uses them as UVW coordinates. For images, only the X and Y coordinates are relevant, which are mapped to U and V. In other words, the images use flat mapping projected along the Z axis. XYZ To UVW results differ depending on whether it is applied to a **Procedural** texture or an imported texture. In Figure 1, a procedural texture using XYZ To UVW is oriented in a similar fashion to **Box** projection. In Figure 2, an imported texture using XYZ To UVW is oriented in a planar fashion.

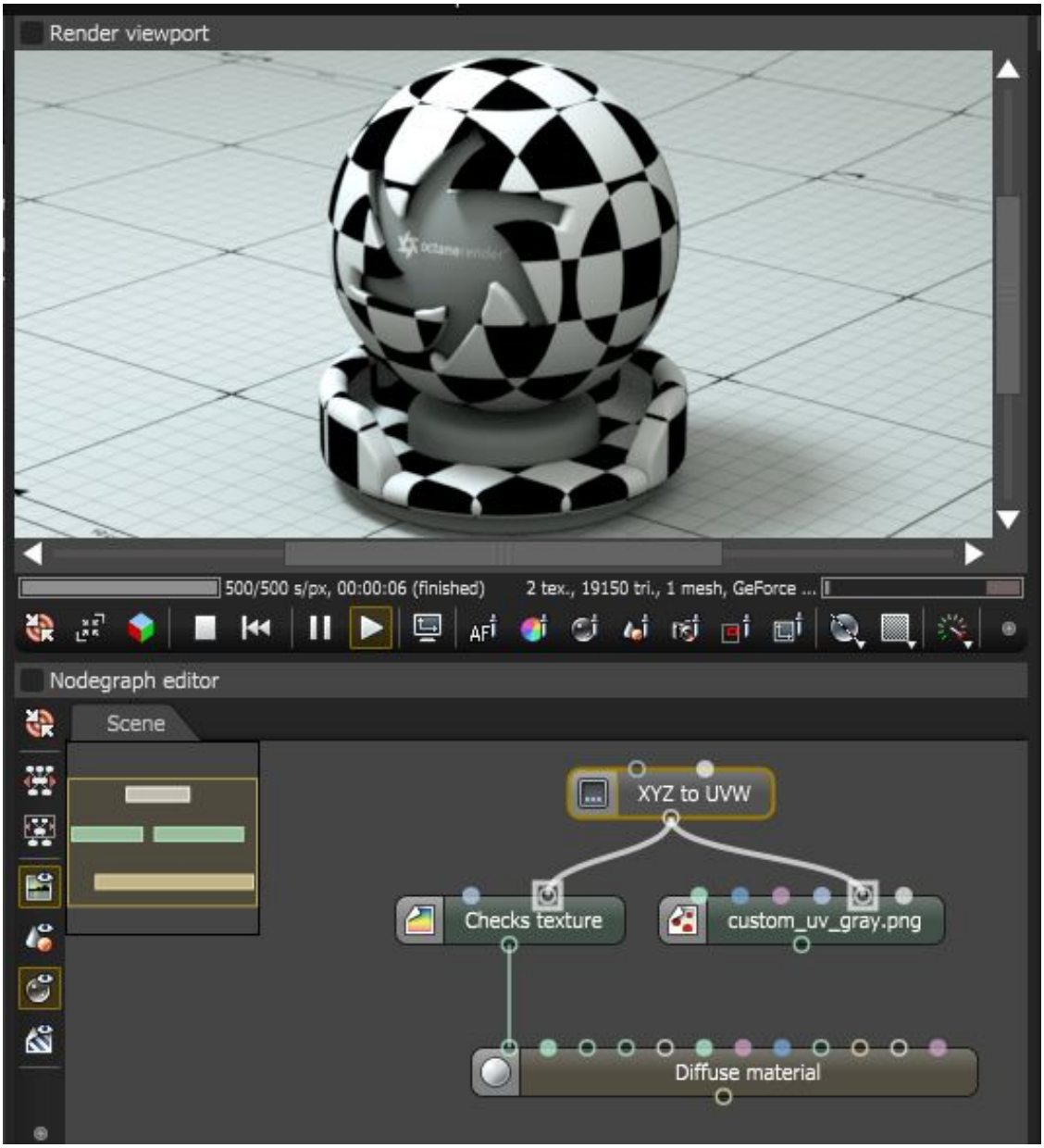


Figure 1: XYZ To UVW projection applied to a Procedural texture map

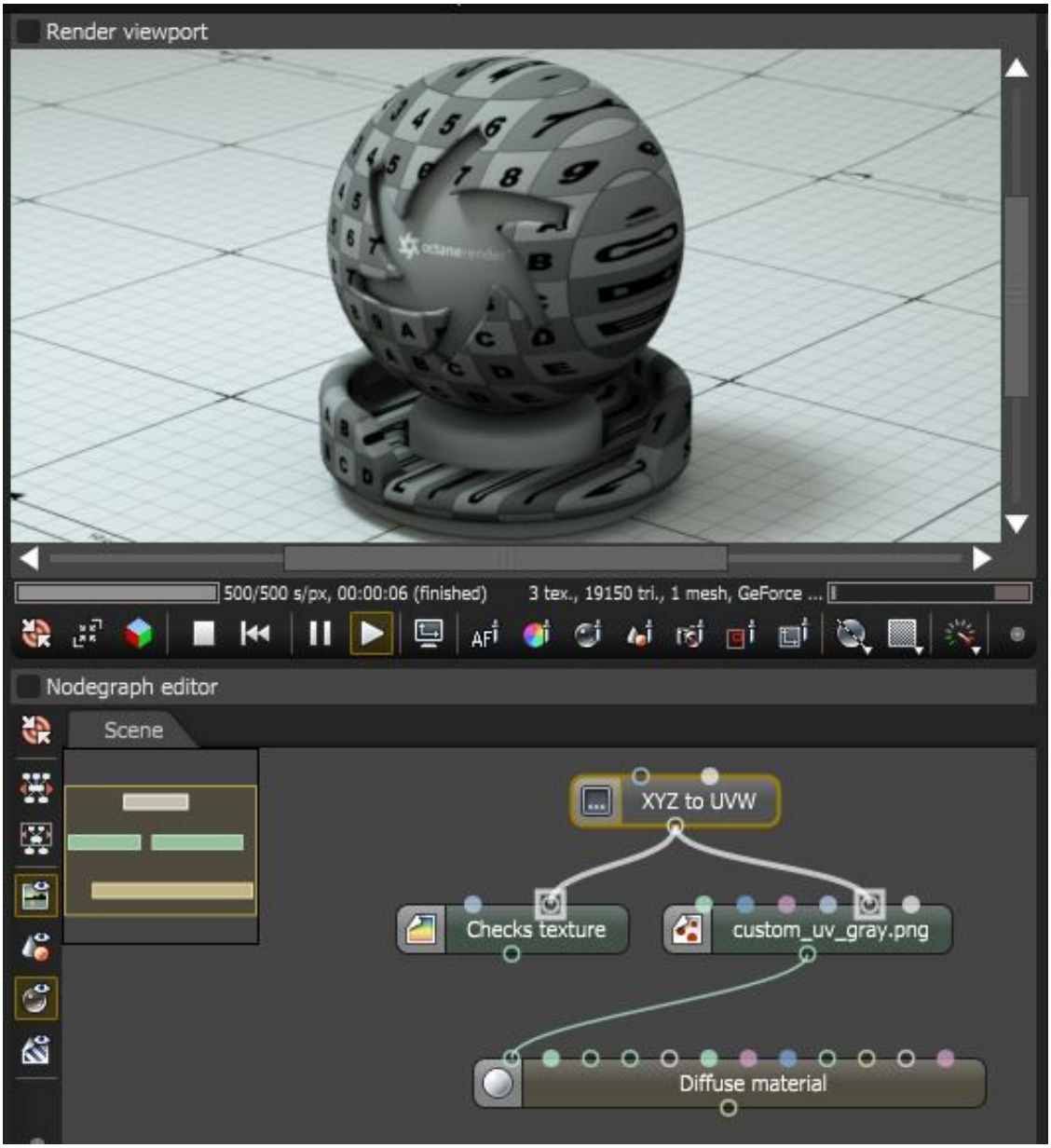


Figure 2: XYZ to UVW projection applied to an imported texture map

XYZ To UVW maps image textures to the (-1, -1) - (1, 1) range. Rotating the mapping around the Z axis rotates the image around the center, as the UVW rotation would do. Octane uses the object coordinate space in a way that the texture projection is in a coordinate space local to each instance. If UV mapping is required, you can apply a transformation in UV space (translation/scale/rotation) via the **UV Transform** pin.

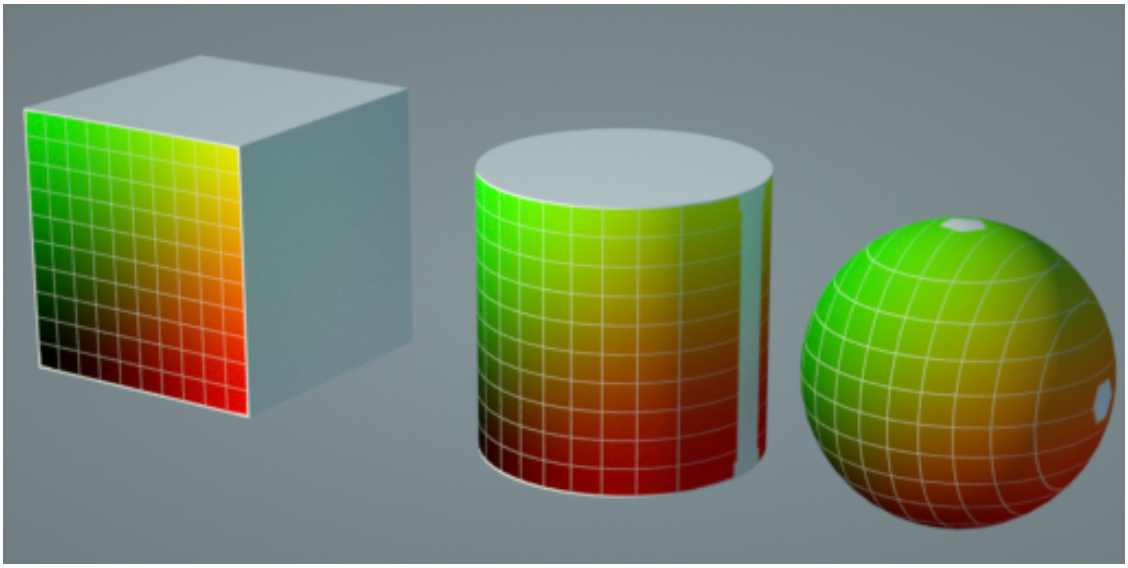
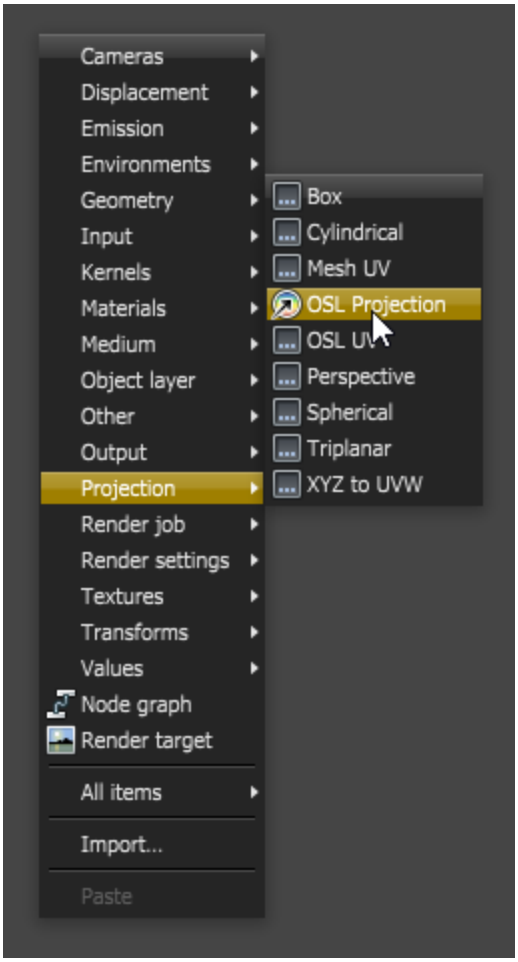


Figure 3: XYZ To UVW projection applied to a box, cylinder, and sphere

OSL Projection Node

The OSL Projection node is a scriptable node where users write OSL (**Open Shader Language**[1]) scripts using the to define arbitrary projection types. It works similar to an OSL Texture node but connects to a projection input. OSL is a standard created by Sony Imageworks. To learn about the generic OSL standard, information is provided from the OSL Readme and PDF documentation.



[1] A shading language developed by Sony Pictures Imageworks. There are multiple render engines that utilize OSL as it is particularly suited for physically-based renderers.

Figure 1: Right-click on the Node Graph Editor pane and select Projections[1] > OSL Projection node from the pop-up context menu

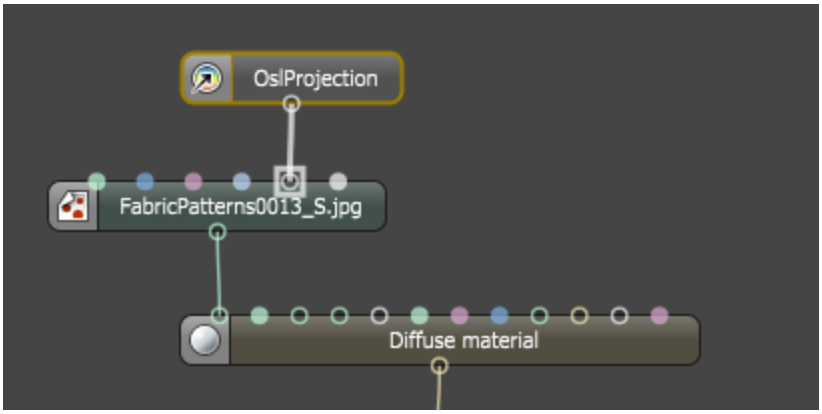


Figure 2: An OSL Projection node is connected to the projection input pin of an Image texture node

The customized OSL script is written into the OSL Projection node to create custom camera types. To edit the script, click on the pencil icon to go to the script editor window. If the script exists as an external .osl file, insert the .osl file into the node through the load icon. Any existing file already used within an Osl Camera node may be edited. To refresh the file and use the edits, reload the file via the reload icon.

[1] Methods for orienting 2D texture maps onto 3D surfaces.

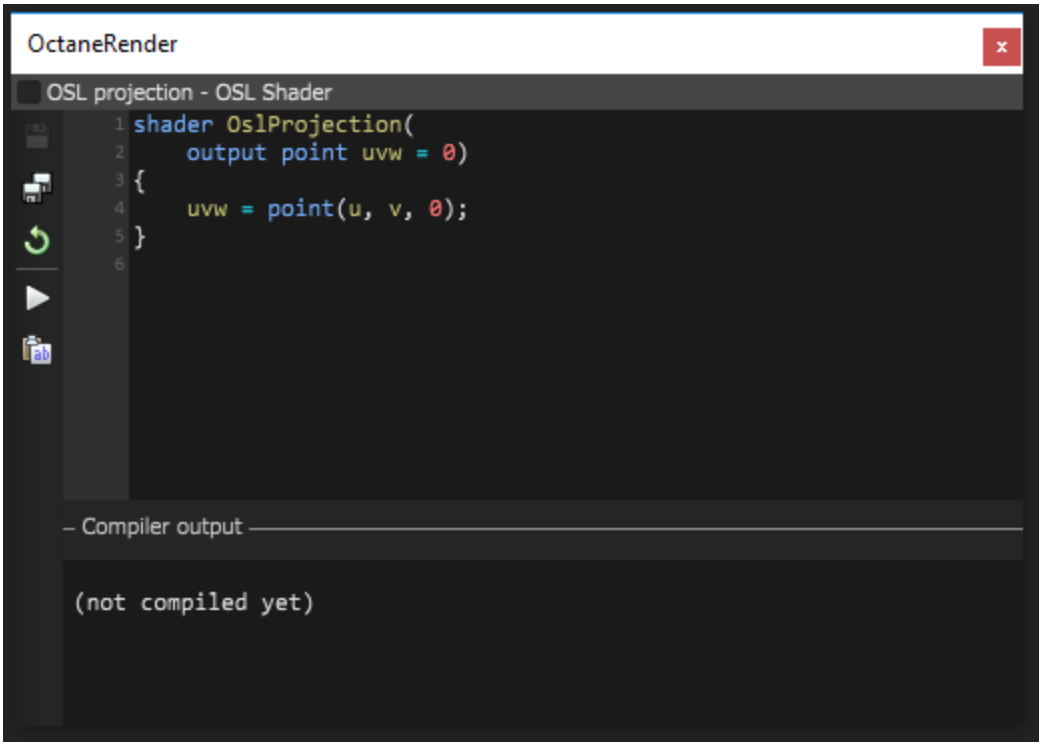


Figure 3: The script editor window showing the initial script of the OSL Projection node

When an OSL Projection node is invoked in Octane's node system, the node is provided with an initial OSL script (Figure 3):

```
Shader OslProjection (
      output point uvw = 0)
{
      uvw = point(u, v, 0);
}
```

The initial script's declaration component includes one required output variable with **output type** `point`. The OSL **I/O type** "`point`" corresponds to an ***Octane projection attribute*** node (Box, Mesh UV, Spherical, Cylindrical...).

A projection shader must have one output of a point-like type. All global variables have the same meaning as within texture shaders. The output value specifies a texture coordinate.

For a list of OSL variable declaration **Input/Output types** in the OSL Specification that Octane supports, refer to the Appendix topic on OSL Implementation in Octane. To learn more about scripting within Octane using the Open Shader Language refer to The Octane OSL Guide.

Transforms

Transforms provide control over the texture map placement on object surfaces. Most **Projection** and **Generator** nodes offer similar controls, but the Transform nodes provide an additional level of control for texture placement. The values in a Transform node are multiplied by the identical values in a Projection or Generator node. Note that you can also use the Transform nodes to control geometry object placement in a scene.

You can access the Transform nodes by right-clicking in the **Nodegraph Editor** and pointing to the **Transforms** menu item (Figure 1).

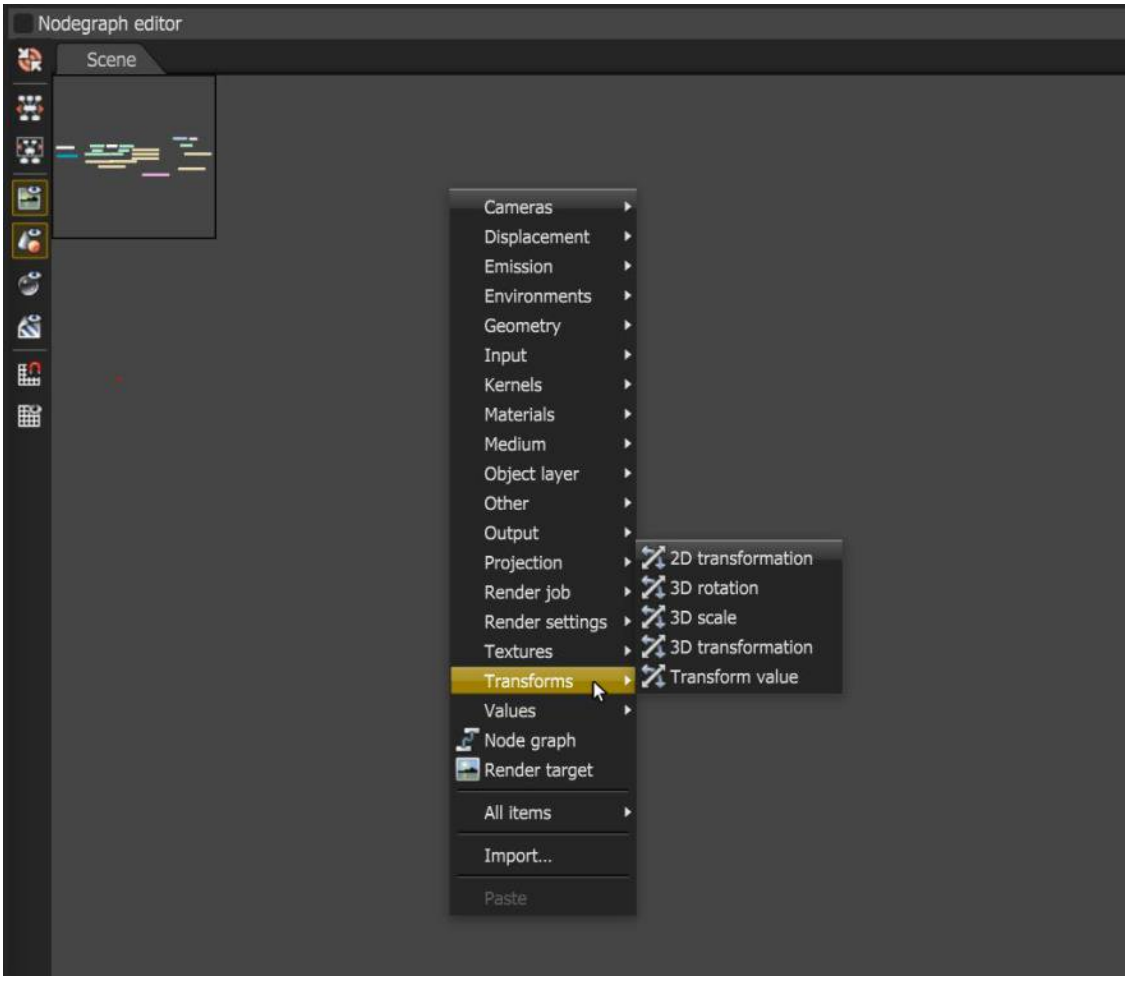


Figure 1: Accessing the Transforms list in the Nodegraph Editor

Transforms are usually paired with a Projection node in order to place a texture map on an object's surface (Figure 2).

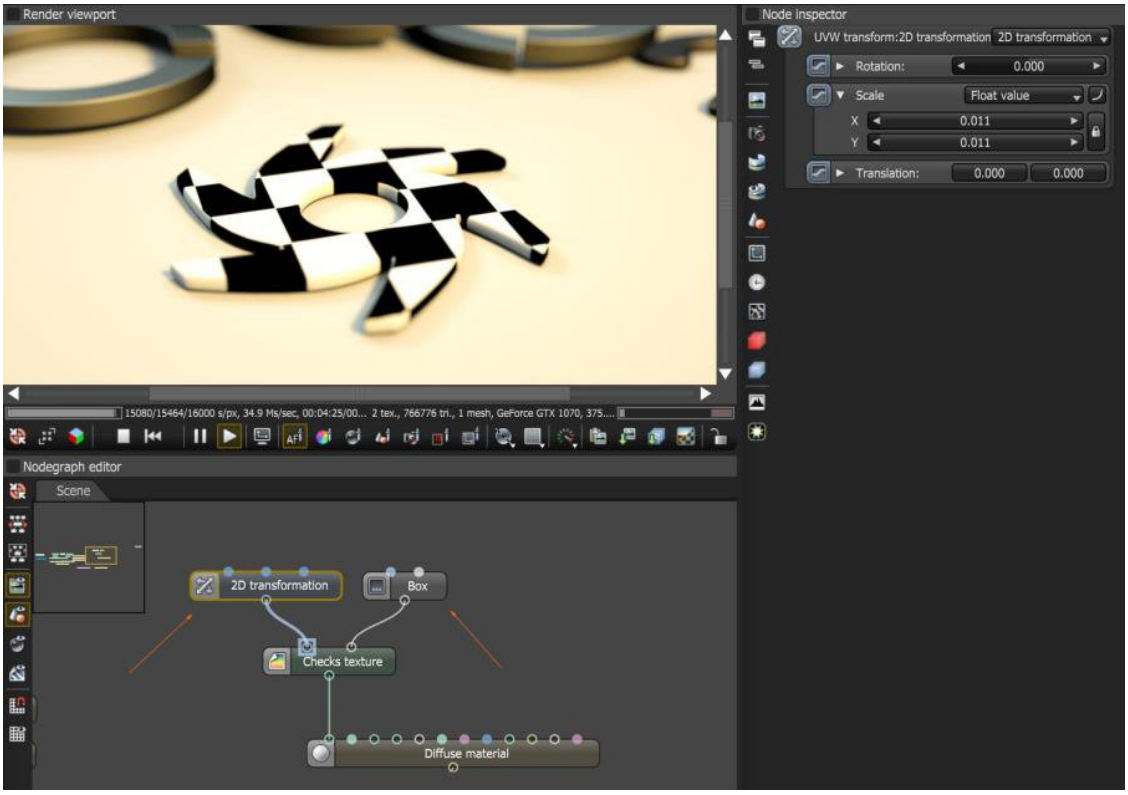


Figure 2: A 2D Transform node paired with a Box projection to orient a Checks texture

2D Transformation

The **2D Transformation** node provides scale and translation parameters for x and y, but not z. The **Rotation** parameter rotates around the z axis, or perpendicular to the object's surface (Figure 1).

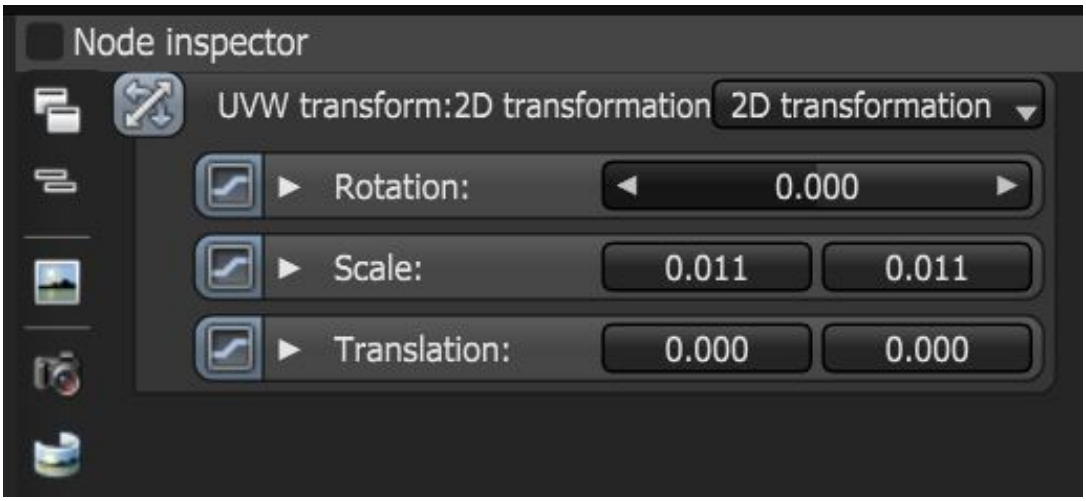


Figure 1: The 2D Transformation node's parameters

3D Rotation

The **3D Rotation** transform controls rotational values on the x, y, and z axes. It also allows you to change the rotation order (Figure 1).

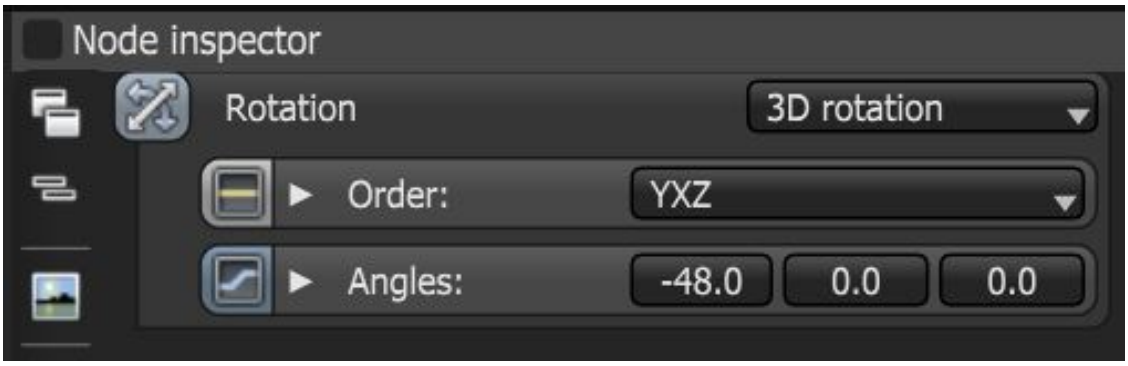


Figure 1: The 3D Rotation node's parameters

3D Scale

The **3D Scale** node provides parameters for controlling the x, y, and z values for a texture map's scale on the surface of an object (Figure 1).

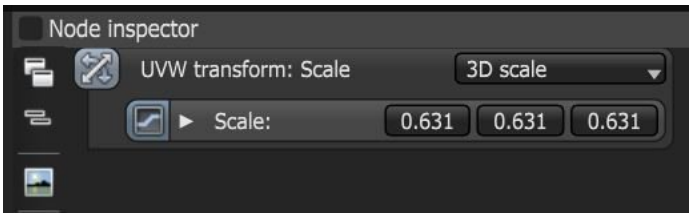


Figure 1: The 3D Scale node's parameters

3D Transformation

The **3D Transformation** node has parameters for **Rotation**, **Scale**, and **Translation** on all three axes. There is also a parameter for selecting different axes combinations for the **Rotation Order** (Figure 1).

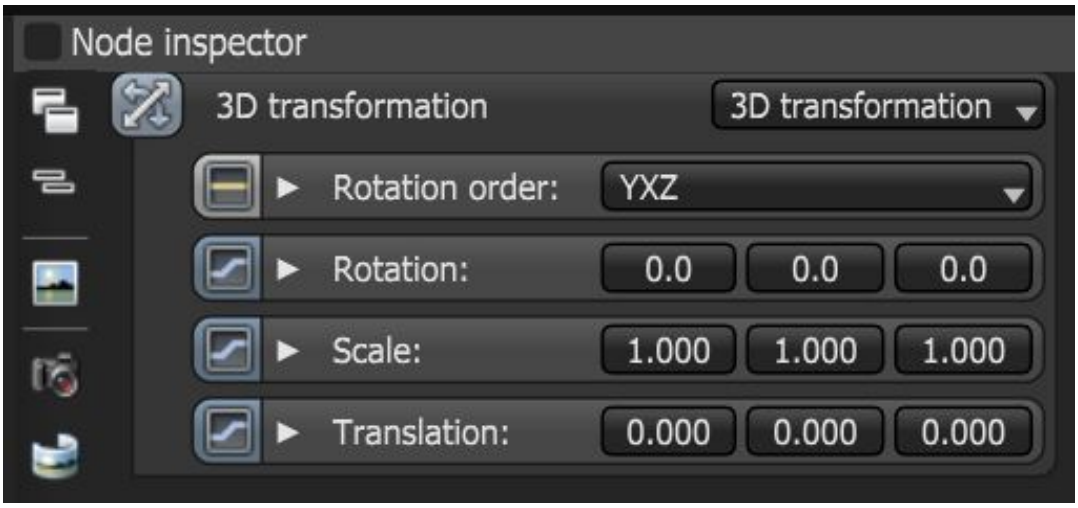


Figure 1: The 3D Transformation node's parameters

Transform Value

The **Transform Value** node is similar to the **3D Transformation** node, except it does not provide any input pins (Figure 1).

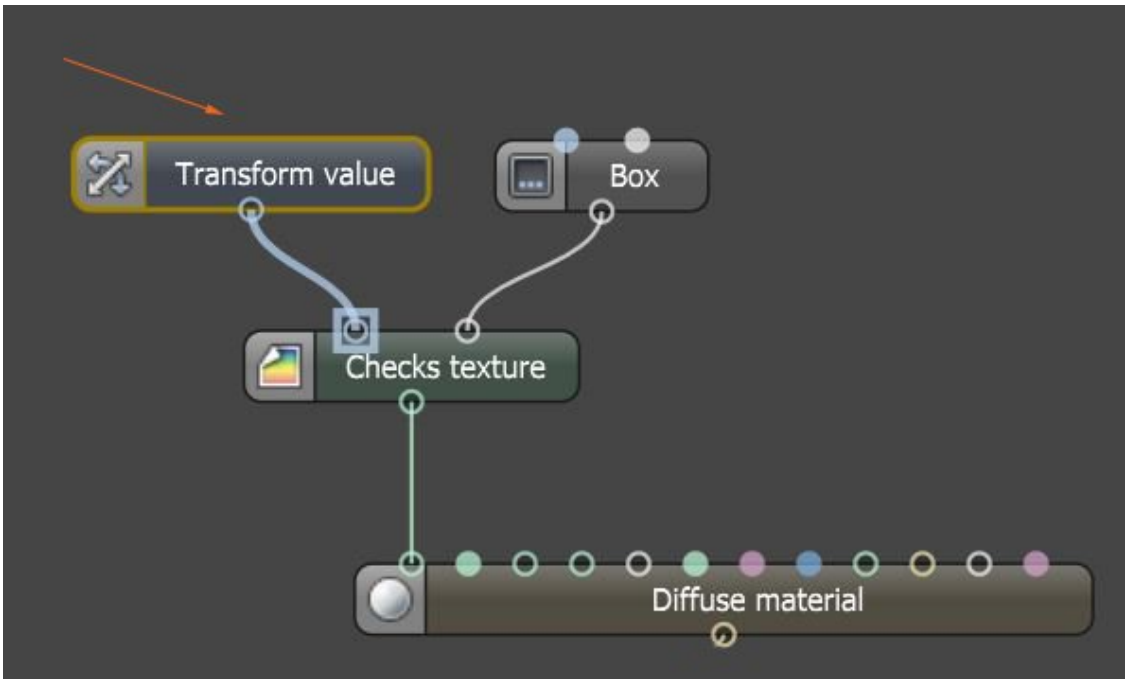


Figure 1: The Transform Value node does not have input pins

Lighting

OctaneRender® for Unity® provides flexible lighting solutions directly in the Unity interface, and more complex solutions when utilizing the power of the OctaneVR® interface.

Figure 1: A Unity Area light (left) and an OctaneRender Mesh Emitter (right)

Unity Lights

OctaneRender® for Unity® is capable of rendering the Unity **Point**, **Spot**, and **Area** light sources. OctaneRender also supports the Unity **Directional** light, but only as the **Sun** parameter in a **Daylight** environment (Figure 1).

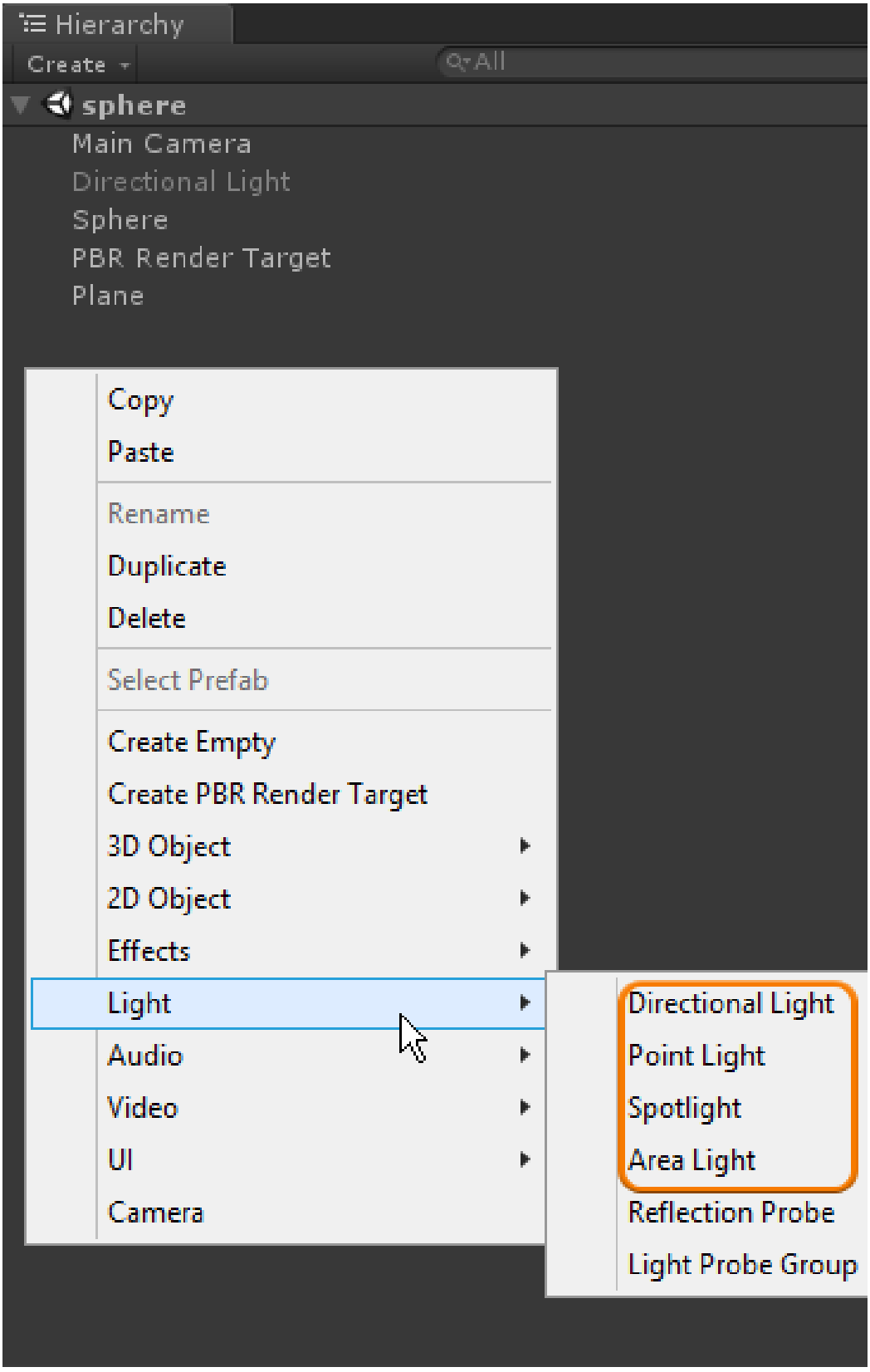

Hierarchy
Create
All
sphere
Main Camera
Directional Light
Sphere
PBR Render Target
Plane
Copy
Paste
Rename
Duplicate
Delete
Select Prefab
Create Empty
Create PBR Render Target
3D Object
2D Object
Effects
Light
Audio
Video
UI
Camera
Directional Light
Point Light
Spotlight
Area Light
Reflection Probe
Light Probe Group

Figure 1: Adding Unity lights to a scene rendered with OctaneRender

Although you can use the Unity light parameters to alter the light source, using the parameters in the **PBR**[1] **Unity Light Component (Script)** generates more predictable results. You can find details on using this component in the **"PBR Unity Light Component" on page 274 (Script)** section.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Point Light

The Unity® **Point** light emits light in all directions from a finite location (Figure 1).

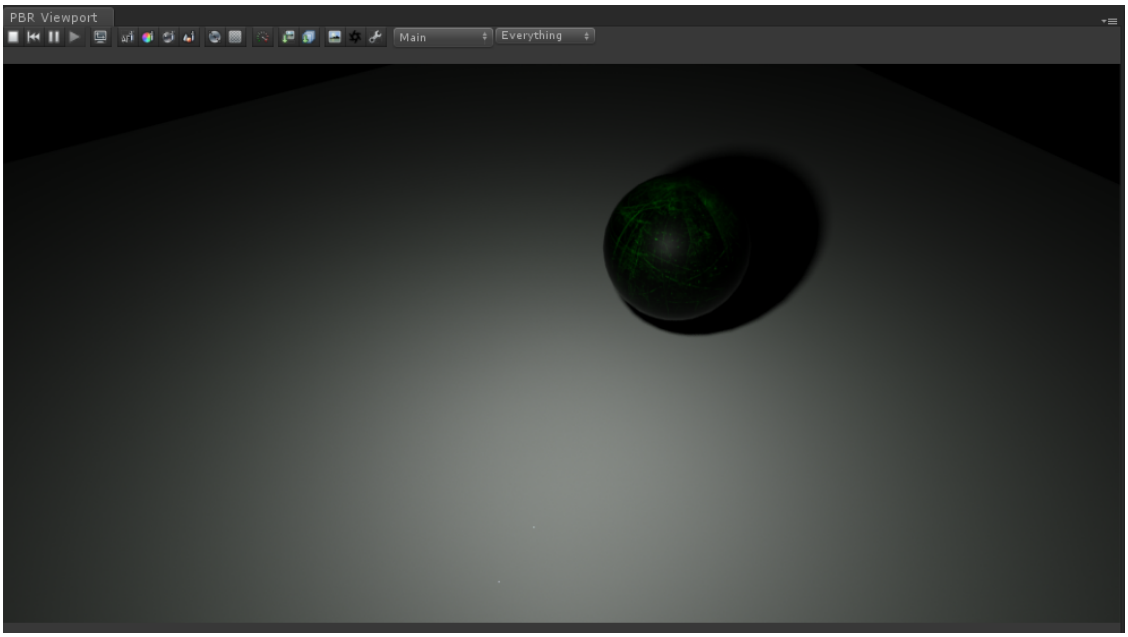


Figure 1: Rendering a Unity Point light with OctaneRender®

The following Point Light parameters are compatible with OctaneRender:

- **Type**
- **Range**
- **Color**
- **Intensity**

Spotlight

The Unity® **Spotlight** emits a light source in a specified direction from a finite location (Figure 1).

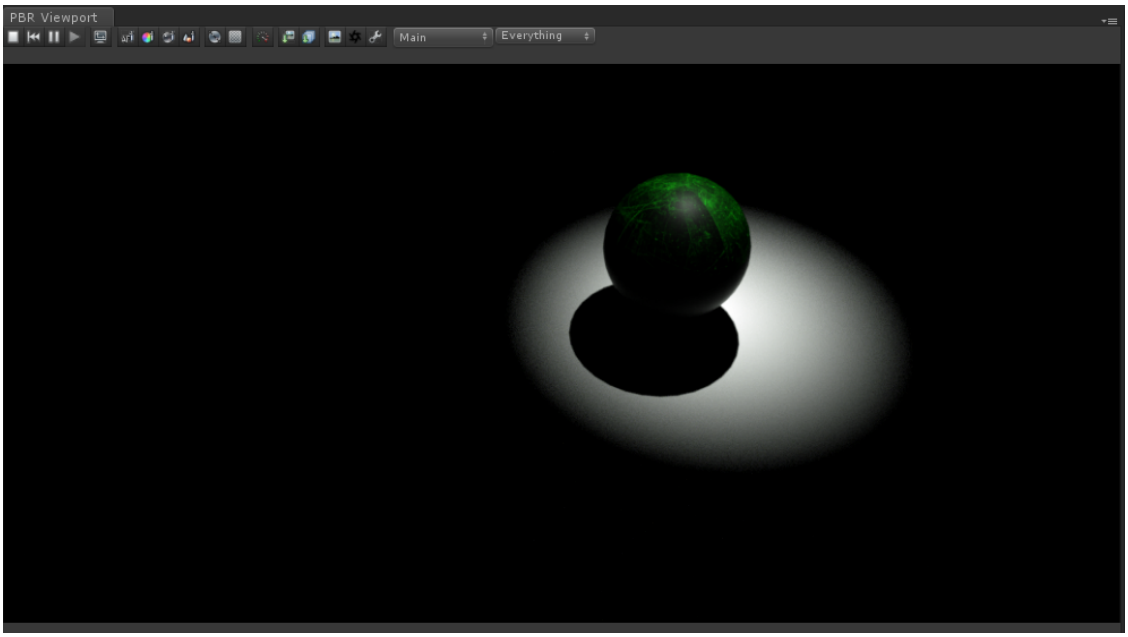


Figure 1: Rendering a Unity Spotlight with OctaneRender®

The following Spotlight parameters are compatible with OctaneRender:

- **Type**
- **Range**
- **Spot Angle**
- **Color**
- **Intensity**

Area Light

The Unity® **Area Light** creates a light source that emits from a specified area and finite location. The larger the emission area, the softer the illumination (Figure 1).

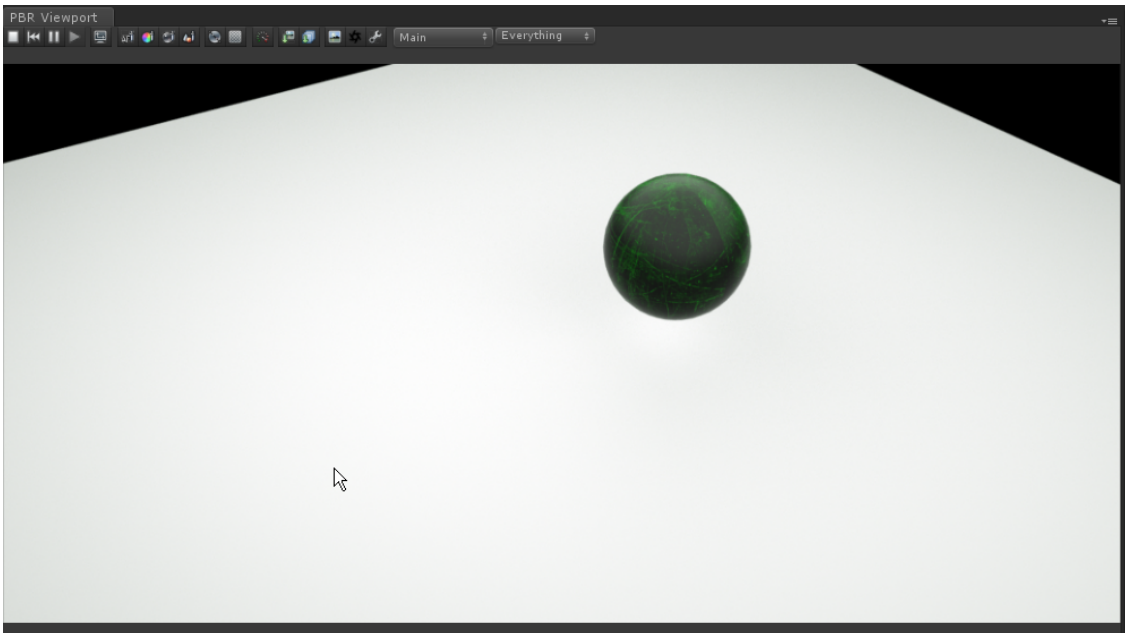


Figure 1: The result of rendering a Unity Area Light with OctaneRender®

The following Area Light parameters are compatible with OctaneRender:

- **Type**
- **Range**
- **Color**
- **Intensity**

Directional Light

The Unity® **Directional** light source cannot function independently like the **Point** light, **Spotlight**, or **Area** light. It works in conjunction with either a **Daylight** or **Texture** environment, which you specify in a **PBR**[1] **Render Target** under the **Environment** rollout (Figure 1).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

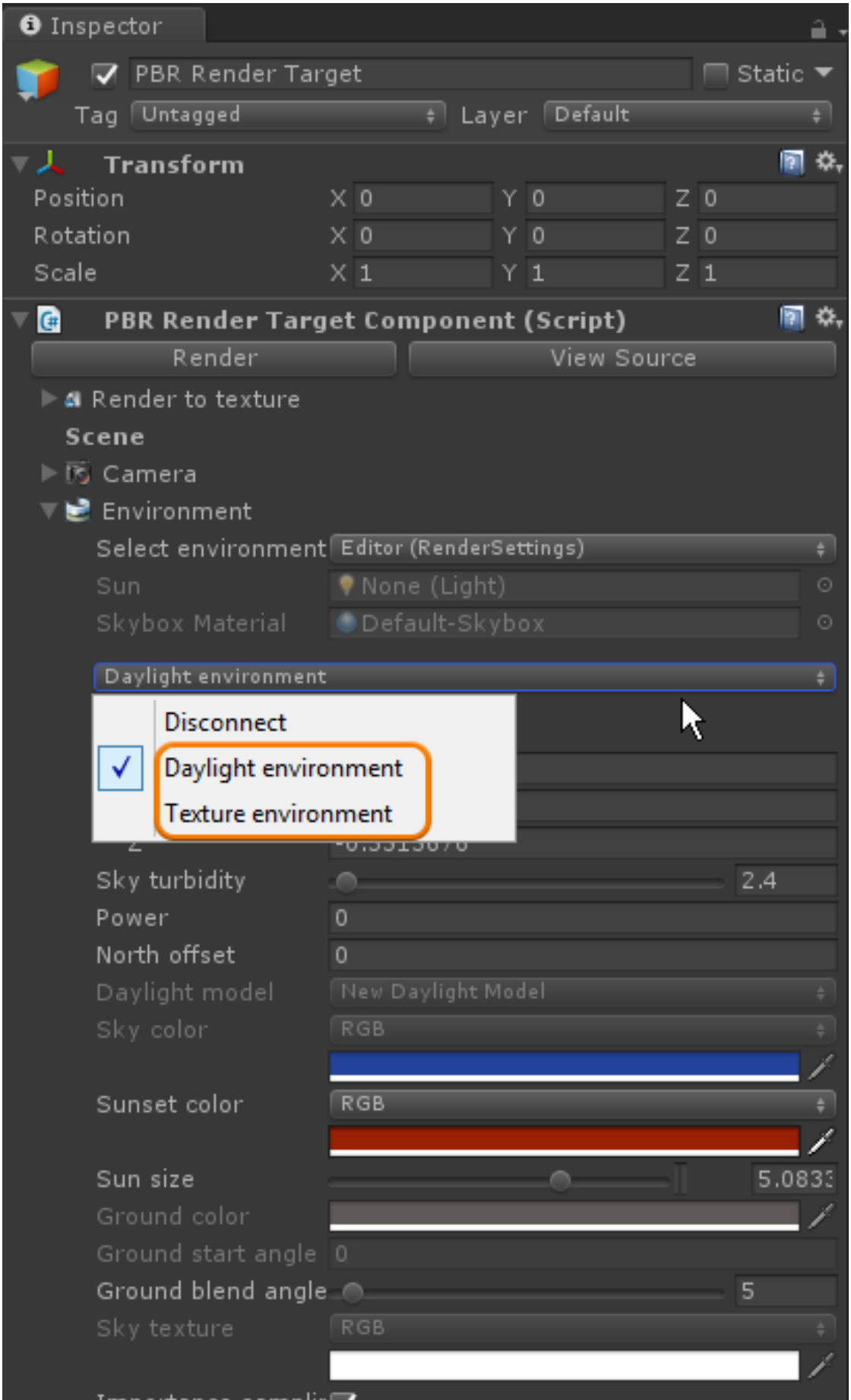

Inspector
PBR Render Target
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
PBR Render Target Component (Script)
Render
View Source
Render to texture
Scene
Camera
Environment
Select environment
Editor (RenderSettings)
Sun
None (Light)
Skybox Material
Default-Skybox
Daylight environment
Disconnect
Daylight environment
Texture environment
Sky turbidity
2.4
Power
North offset
Daylight model
New Daylight Model
Sky color
RGB
Sunset color
RGB
Sun size
Ground color
Ground start angle
Ground blend angle
5
Sky texture
RGB

Figure 1: The Directional light is used in conjunction with either a Daylight or Texture environment

Due to the nature of a Directional light, the only Unity parameter compatible with OctaneRender® is **Intensity**.

PBR Unity Light Component

The **PBR**[1] **Unity Light** component provides PBR control over light sources in Unity® that are rendered using OctaneRender®. When this component is added to a light source, OctaneRender overrides the default Unity **Color**, **Intensity**, and **Range** parameters. You can add a PBR Unity Light component to a light source by clicking on the **Add Component** button in the light's **Inspector** window, and then choosing **PBR**, followed by **Light Override** (Figure 1).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

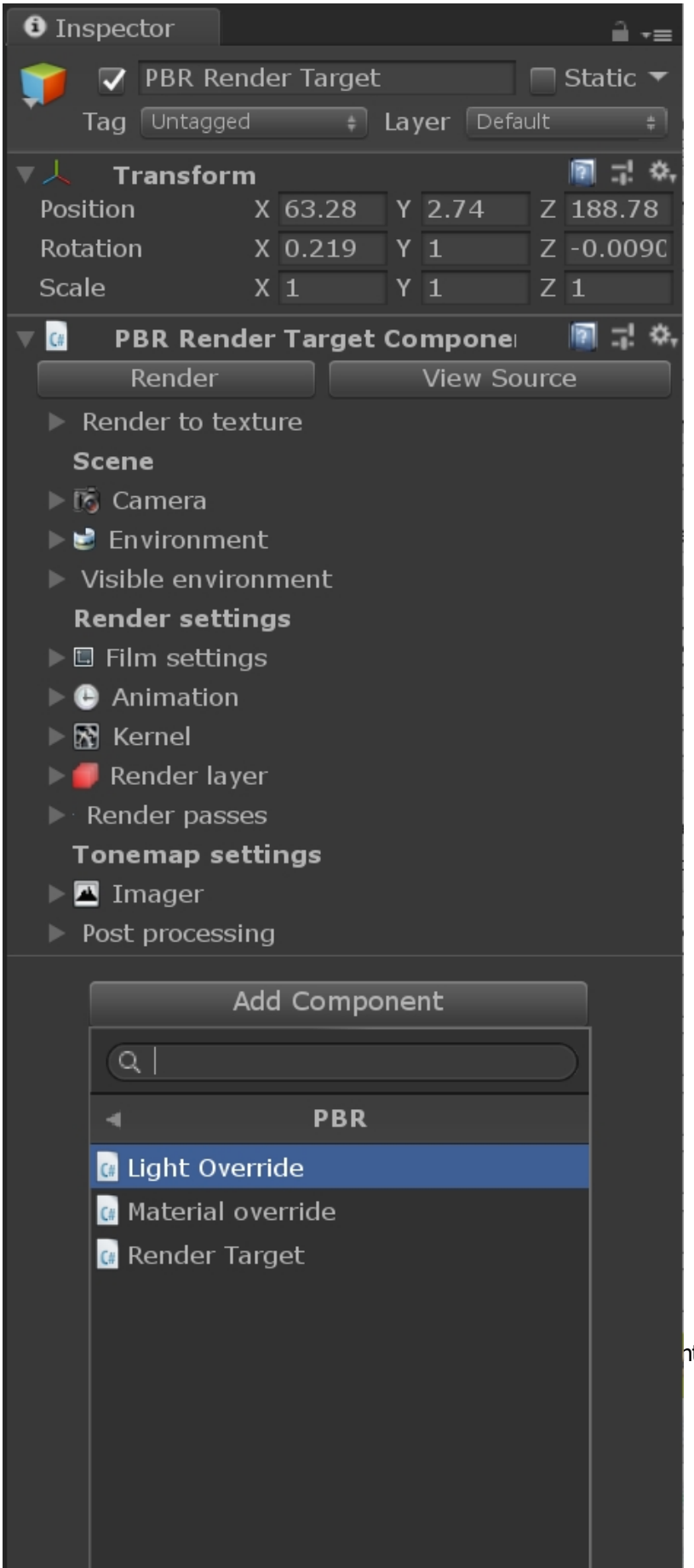

Inspector
PBR Render Target
Static
Tag
Untagged
Layer
Default
Transform
Position
X 63.28
Y 2.74
Z 188.78
Rotation
X 0.219
Y 1
Z -0.0090
Scale
X 1
Y 1
Z 1
PBR Render Target Compone
Render
View Source
Render to texture
Scene
Camera
Environment
Visible environment
Render settings
Film settings
Animation
Kernel
Render layer
Render passes
Tonemap settings
Imager
Post processing
Add Component
PBR
Light Override
Material override
Render Target

Figure 1: Accessing the PBR Unity Light component from the Inspector window

The PBR Unity Light component contains various rollouts. You can ignore the **Material**[1], **Displacement**[2], and **Medium** rollouts, as they do not have any impact on light sources in Unity (Figure 2).

[1]The representation of the surface or volume properties of an object.

[2]The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

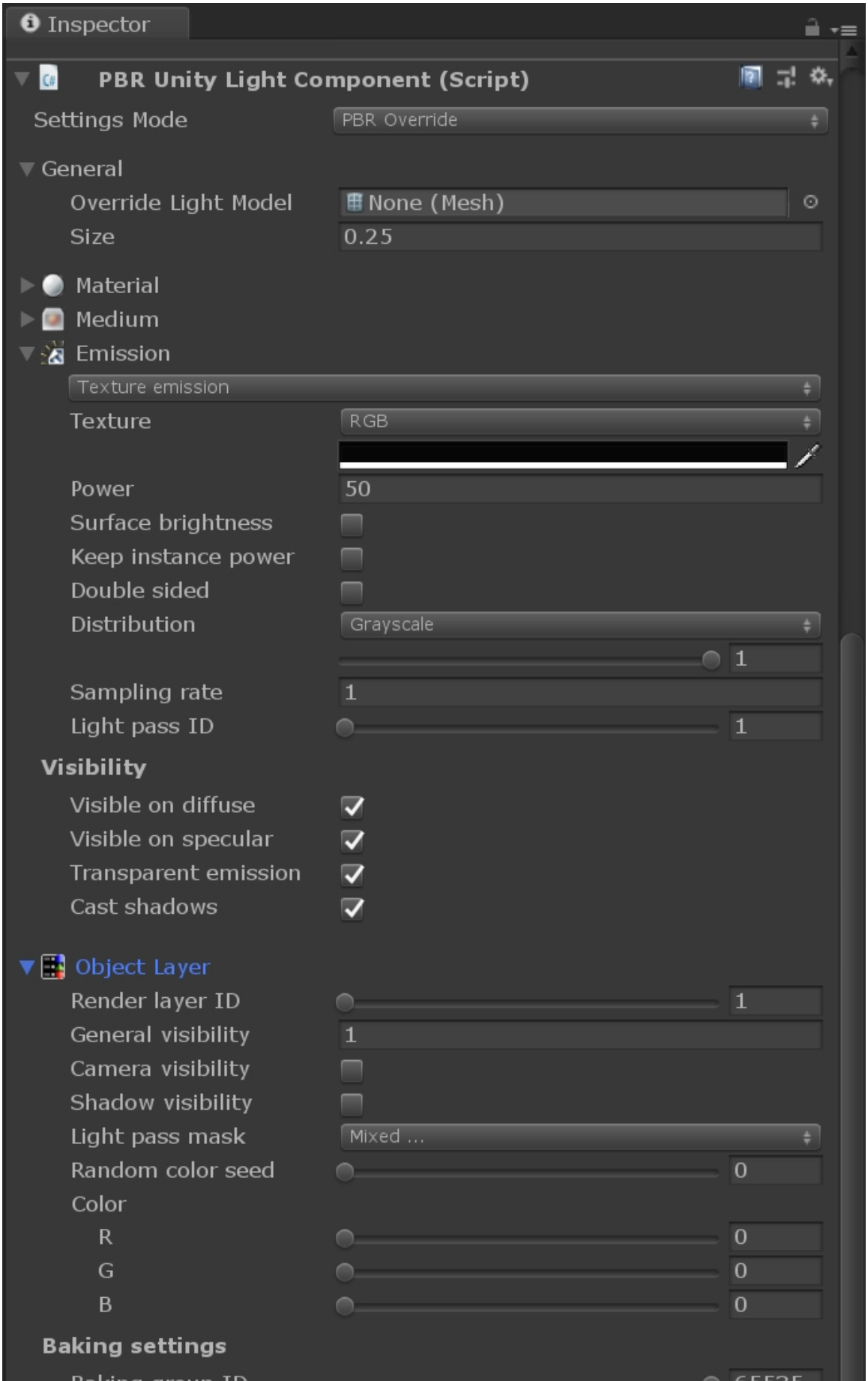

Inspector
PBR Unity Light Component (Script)
Settings Mode
PBR Override
General
Override Light Model
None (Mesh)
Size
0.25
Material
Medium
Emission
Texture emission
Texture
RGB
Power
50
Surface brightness
Keep instance power
Double sided
Distribution
Grayscale
1
Sampling rate
1
Light pass ID
1
Visibility
Visible on diffuse
Visible on specular
Transparent emission
Cast shadows
Object Layer
Render layer ID
1
General visibility
1
Camera visibility
Shadow visibility
Light pass mask
Mixed ...
Random color seed
0
Color
R
0
G
0
B
0
Baking settings

Figure 2: The PBR Unity Light Component parameters

PBR Unity Light Component Parameters

- **General**
  - **Override Light Model** - Uses a mesh shape for the light source emission.
  - **Size** - Determines the overall size of the light source. The larger the light, the brighter the emission.
- **Emission**
  - **Texture** or **Black Body**[1] emission - The primary difference between the Texture and Black Body emission types is that the Black Body emission contains an additional parameter for **Temperature**.
  - **Texture** - Sets the efficiency of the light source. It can be set to a color, value, or a texture.
  - **Power** - The overall brightness or wattage of the light source. You should set each light in the scene to its real-world wattage. For example, set a desk lamp to 25 watts, and a ceiling lamp to 100 watts.
  - **Temperature** (Black Body only) - The emitting light's temperature (in Kelvin). Lower values produce warmer illumination, and higher values produce cooler illumination.
  - **Surface Brightness** - Enabling this option causes emitters to keep the surface brightness constant, independent of the emitter surface area.
  - **Keep Instance Power** - If you enable this option while Surface Brightness is disabled and Uniform Scaling is applied to the object, then the Power remains constant.
  - **Double Sided** - Enables the light source to emit illumination from both sides of its surface.
  - **Distribution** - Controls the pattern of the light. You can set this to **RGB**, **Greyscale**, **Texture**, or an **IES**[2] light profile. The image texture's **Projection** nodes then adjust the orientation or direction of the light.
  - **Sampling Rate** - Choose what light sources will receive more samples.
  - **Light Pass ID** - This is the ID of the light pass that captures the emitter contribution. It works with the **Light Pass ID Render** element.
  - **Visible on Diffuse**[3] - Toggles the surface Albedo illumination on and off. If off, the light only contributes to the shiny aspects of surfaces.

---

[1] An opaque object that emits thermal radiation. In Octane, this is used to designate illumination properties for mesh emitters.
[2] An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.
[3] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

- **Visible on Specular**[1] - Toggles whether the light source contributes to shiny or specular highlights on a surface.
- **Transparent Emission** - Determines whether or not the light source casts illumination even when the light is transparent.
- **Cast Shadows** - Toggles shadow rendering on and off.

- **Object Layer**
  - **Render Layer ID** - The object's render layer ID.
  - **General Visibility** - This controls the visibility level for the object and its shadow.
  - **Camera Visibility** - Specifies if the object is visible to the camera. This is enabled by default.
  - **Shadow Visibility** - Specifies if the shadow cast by the object is visible to the camera. This is enabled by default.
  - **Light Pass Mask** - Enables/disables illumination from Light IDs on the current object layer. This parameter consists of a combination of flags in the LightPassMask enum. You can globally invert any flags in this mask from the **Kernel** settings.
  - **Random Color Seed** - This specifies the start point to initialize the color after which random colors are generated. This is **0** by default when random colors are currently not in use.
  - **Color** - The color used for the assigned object when it is rendered in the object render layer pass.
- **Baking Settings**- Specifies the baking group to bake. By default, all objects belong to baking group number **1**. You can arrange new baking groups by making use of object layers or object layer maps similar to the way render layers work.

[1] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

Daylight Environment

The **Daylight** environment simulates an outdoor lighting setup using real-world parameters. It is the default lighting solution when OctaneRender® for Unity® is invoked (Figure 1).

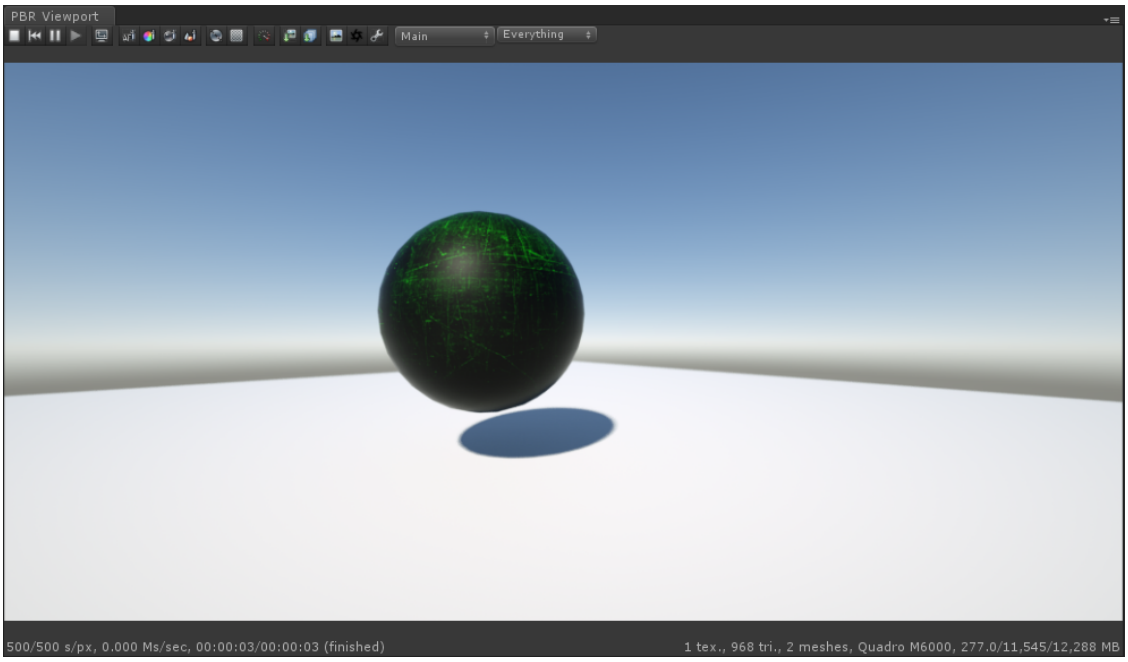


Figure 1: The default Daylight environment in OctaneRender for Unity

You can access the Daylight environment in the **PBR**[1] **Render Target** Inspector window, under the **Environment** rollout (Figure 2).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

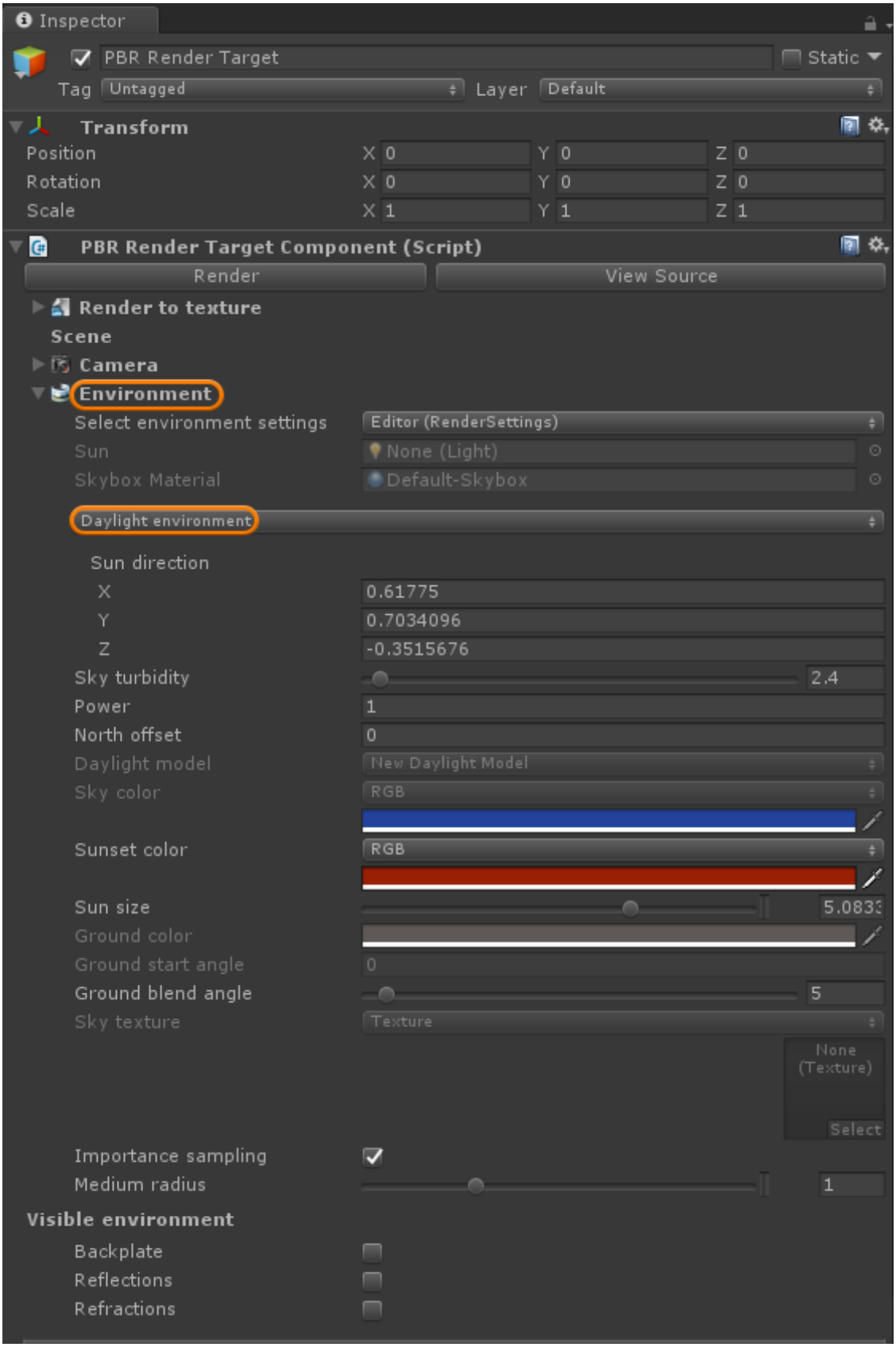

Inspector
PBR Render Target
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
PBR Render Target Component (Script)
Render
View Source
Render to texture
Scene
Camera
Environment
Select environment settings
Editor (RenderSettings)
Sun
None (Light)
Skybox Material
Default-Skybox
Daylight environment
Sun direction
X 0.61775
Y 0.7034096
Z -0.3515676
Sky turbidity 2.4
Power 1
North offset 0
Daylight model New Daylight Model
Sky color RGB
Sunset color RGB
Sun size 5.083
Ground color
Ground start angle 0
Ground blend angle 5
Sky texture Texture
None (Texture)
Select
Importance sampling
Medium radius 1
Visible environment
Backplate
Reflections
Refractions

Figure 2: Accessing the Daylight Environment parameters

The **Environment** rollout provides three parameters that are common to both the Daylight environment and the **Texture** environment:

- **Select Environment Settings**
  - **Render Target** - Allows for selecting different sources as the **Sun** and **Skybox Material**[1].
  - **Editor (RenderSettings)** uses the default **Directional** light and **Default Skybox** settings in a new Unity scene to light the environment.
- **Sun** - Determines which Directional light to use in the scene for the sun's position. If no light is selected, the **Sun Direction** parameters become active and can place the sunlight in the scene.
- **Skybox Material** - Determines the Skybox object to use for the environment colors.

Daylight Environment-Specific Parameters

- **Sky Turbidity** - Adjusts the sharpness of the sunlight's shadows. A low value creates sharp shadows - like on a sunny day - and a higher value diffuses the shadows, similar to a cloudy day.
- **Power** - Adjusts the light strength. This affects the image's overall contrast and exposure level.
- **North Offset** - Adjusts the scene's actual North direction. This is useful for architecture visualization to ensure the sun's direction is accurate to the scene.
- **Daylight Model** - This specifies the daylight model to use as the current environment. The **Old Daylight Model** lights a scene with basic spectral radiance as the sun moves over the horizon at a relative distance from the object. The **New Daylight Model** simulates full spectrum daylight as the sun moves along, and bears shorter rays as the sun moves closer to the normal plane.
- **Sky Color** - The base color of the sky, used by the New Daylight Model to customize the spectral shade of light. This can affect overall mood expressed by the image.
- **Sunset Color** - The color of the sky and sun at sunset, used by the New Daylight Model to customize the spectral shade of light. This affects the overall mood expressed by the image.
- **Sun Size** - Controls the sun radius in the Daylight environment.
- **Ground Color** - The base color of the ground, used by the New Daylight model to customize the spectral shade of light. This affects the overall mood expressed by the image.
- **Ground Start Angle** - The angle below the horizon where the transition to the Ground Color starts, measured in degrees.

---

[1] The representation of the surface or volume properties of an object.

- **Ground Blend Angle** - The angle over which the sky color transitions to the Ground Color.
- **Sky Texture** - Accepts a texture map for the sky environment. This texture is the Daylight environment's sky.
- **Importance Sampling** - This toggles the sky texture's Importance Sampling, similar to the Texture environment's Importance Sampling.
- **Medium Radius** - Controls the radius of the virtual sphere created around the camera when **Volume** mediums are used.

Planetary Environment

This environment type is a flexible Nishita Sky model. It is most useful when rendering scenes as they are seen from outer space. For its effects to be visible, the camera has to have a very high altitude as it moves out into outer space to view the expansive horizon of the planetary body. It takes into account the conditions within and beyond the atmosphere of a planetary body (e.g. planet earth) and its surroundings in space. Instead of a single ground color and a sky/sunset color, there is a planetary surface that reflects and emits light. Most importantly, this node serves to extend the environment's medium (volume rendering and subsurface scattering) with an atmospheric scattering through the planetary body's atmosphere. Here, the atmosphere is perceived as a layer of gas surrounding a planetary mass and it is held in place because of gravity so as the light travels into atmosphere either from the outer layer to the ground or from a light source within the atmosphere, then the atmosphere's density is sampled along the ray at regular intervals resulting in an amount of scattering based on the atmosphere's density. This atmospheric scattering is based on the Nishita Sky Model, a physically based model which displays the variations of color which are optical effects caused by the particles in the atmosphere.

This environment is not connected to the camera and this allows you to zoom the camera view of the objects in the scene in and out while not affecting the position of the environment in the scene. It is a physically based model so it gathers optical depth (transmittance) from the sun position, if the sun position is greater than 0.0f on y axis (upward direction), then it will be colored. If you put it below horizon (i.e. sun position less than 0.0f on the Y axis) then it won't gather transmittance so it will be invisible.

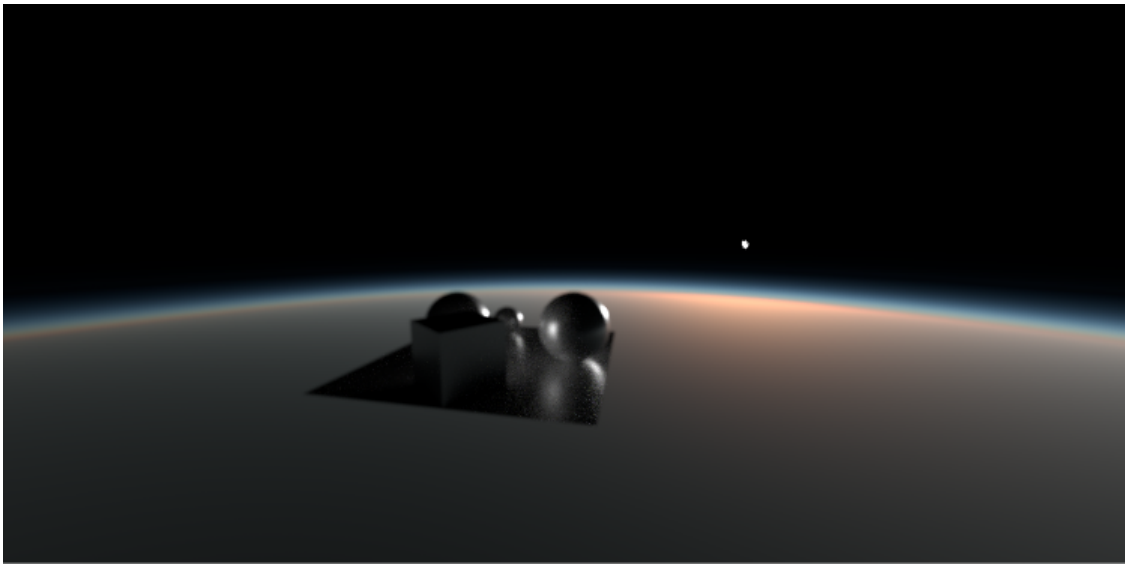


Figure 1: Image rendered using the Planetary Environment where the camera is set at a very high altitude

Altitude

The camera's altitude. This should be set to a very high value in order to view the expansive horizon of the planetary body.

Star Field

Texture to convey star fields behind the planet.

Ground Albedo

surface texture map on the planet.

Ground Reflection

Specular[1] texture map on the planet.

[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

Ground Glossiness

The planetary glossiness.

Ground Emission

surface texture map on the planet at nighttime.

Ground Normal Map

Normal map on the planet.

Ground Elevation

Elevation map on the planet.

Planetary Axis

The rotational axis of the planet running through the North and South Poles.

Planetary Angle

The rotation around the planetary axis, this has a default value of -1.041.

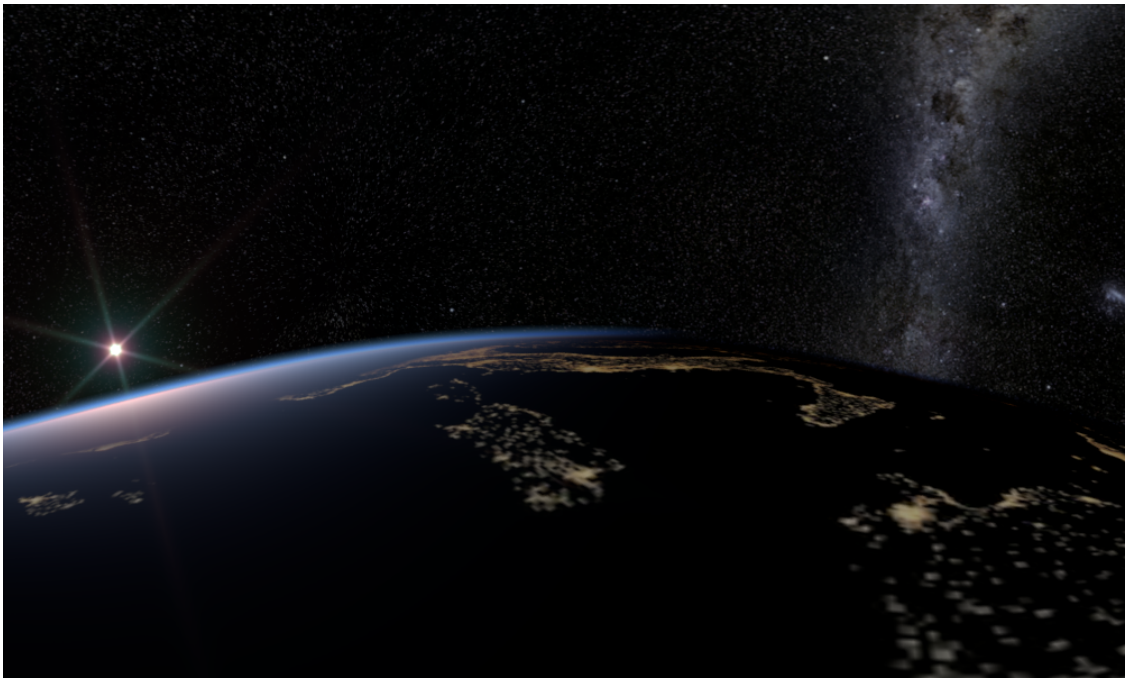


Figure 2: Image rendered using the Planetary Environment with a star field.

Texture Environment

The **Texture** environment affects the environment's illumination and color. You can use it to add an **HDRI**[1] environment texture to a scene for illumination. You can access the Texture environment in the **Inspector** window of the **PBR**[2] **Render Target**, under the **Environment** rollout (Figure 1).

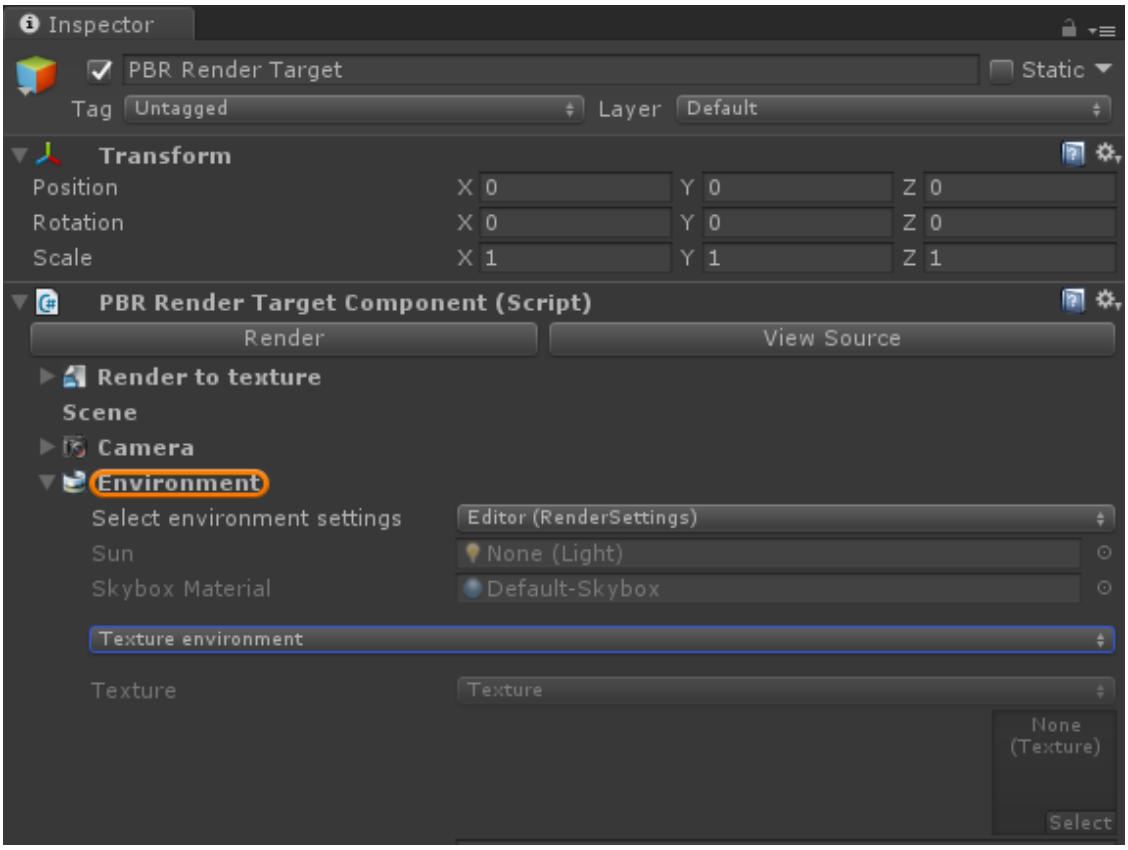


Figure 1: PBR Render Target's Environment rollout

[1] An image which presents more than 8 bit per color channel unlike most common image formats.

[2] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

When you select Texture as the **Environment** type for the first time, it uses the **Default Skybox** from Unity® as its **Skybox Material**[1] (Figure 2).

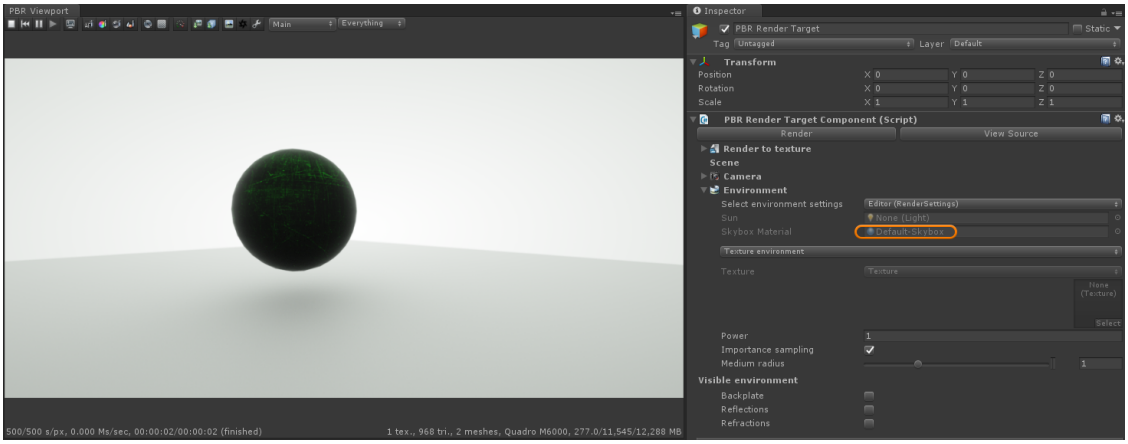


Figure 2: The Default Skybox used for the Texture environment's Skybox Material

To access to the **Texture** parameter, remove the Default Skybox as the Skybox Material (Figure 3).

[1] The representation of the surface or volume properties of an object.

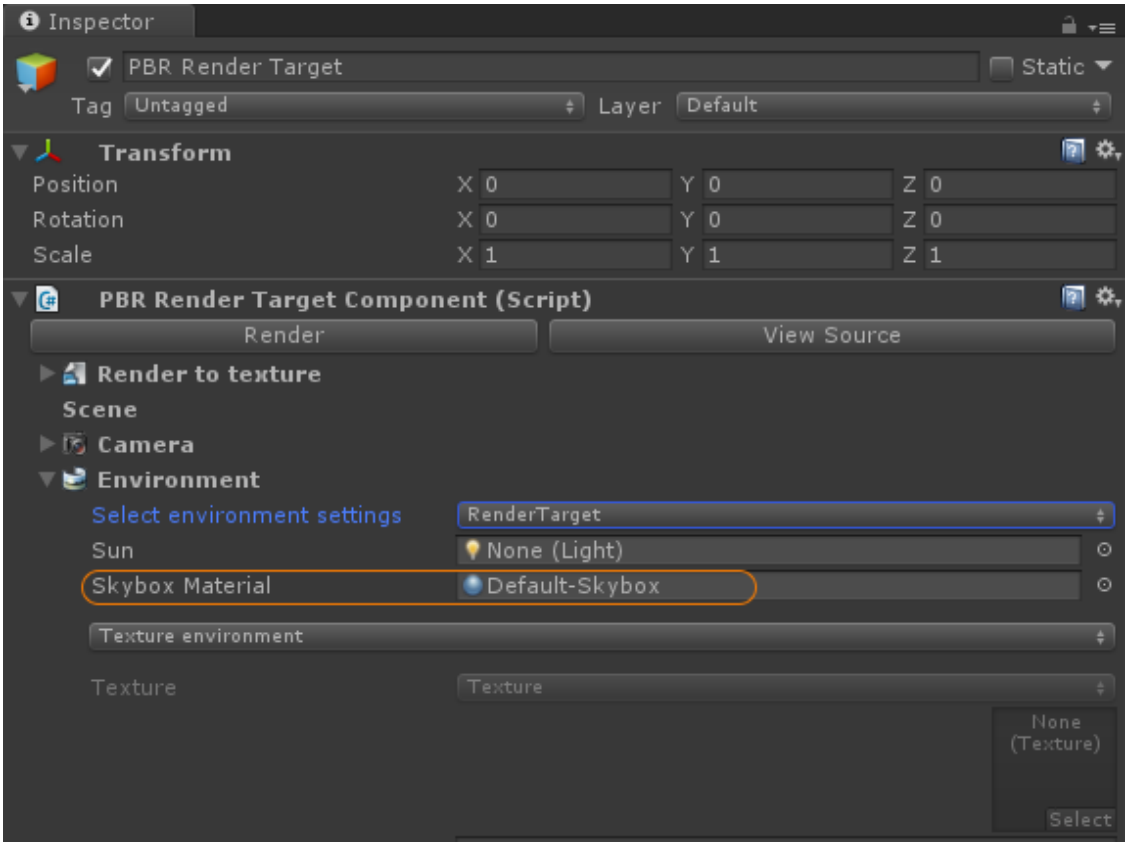


Figure 3: Removing the Default Skybox to access the Texture parameter

Once active, the Texture parameter accepts a color (**RGB**), a value (**Grayscale**), or a **Texture** map.

Other Texture Environment Parameters

- **Power** - Adjusts the light strength. This affects overall image contrast and exposure level.
- **Importance Sampling** - Toggles the sky texture Importance Sampling – similar to the Texture environment's Importance Sampling.
- **Medium Radius** - Controls the camera's virtual sphere radius when you use **Volume** mediums.

Mesh Emitters

A **Mesh Emitter** is a polygon object that emits light into a scene. This is possible by applying a **Diffuse**[1] material to the Mesh object, and then connecting a **Black Body**[2] or **Texture** emission node to the **Diffuse material**[3]'s **Emission** channel (Figure 1).

Figure 1: A light-emitting Diffuse material

---

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2]An opaque object that emits thermal radiation. In Octane, this is used to designate illumination properties for mesh emitters.
[3]Used for dull, non-reflecting materials or mesh emitters.

In order to use a Mesh as a light source, first apply a Diffuse material to the surface, then connect an Emission node to the **Emission** pin. There are two types of **Emissions**[1]:

Black Body Emission

The Black Body emission type uses **Color Temperature** (in Kelvin) and **Power** to control the light's color and intensity, respectively.

Texture Emission

This allows any valid Texture type to set the light intensity. You can use this emission to create interesting effects, such as TV screens, by using an Image texture as the source.

You can access both Emission types by right-clicking in the **Nodegraph Editor** and navigating to the **Emission** category (Figure 2).

[1]The process by which a Black body or Texture is used to emit light from a surface.

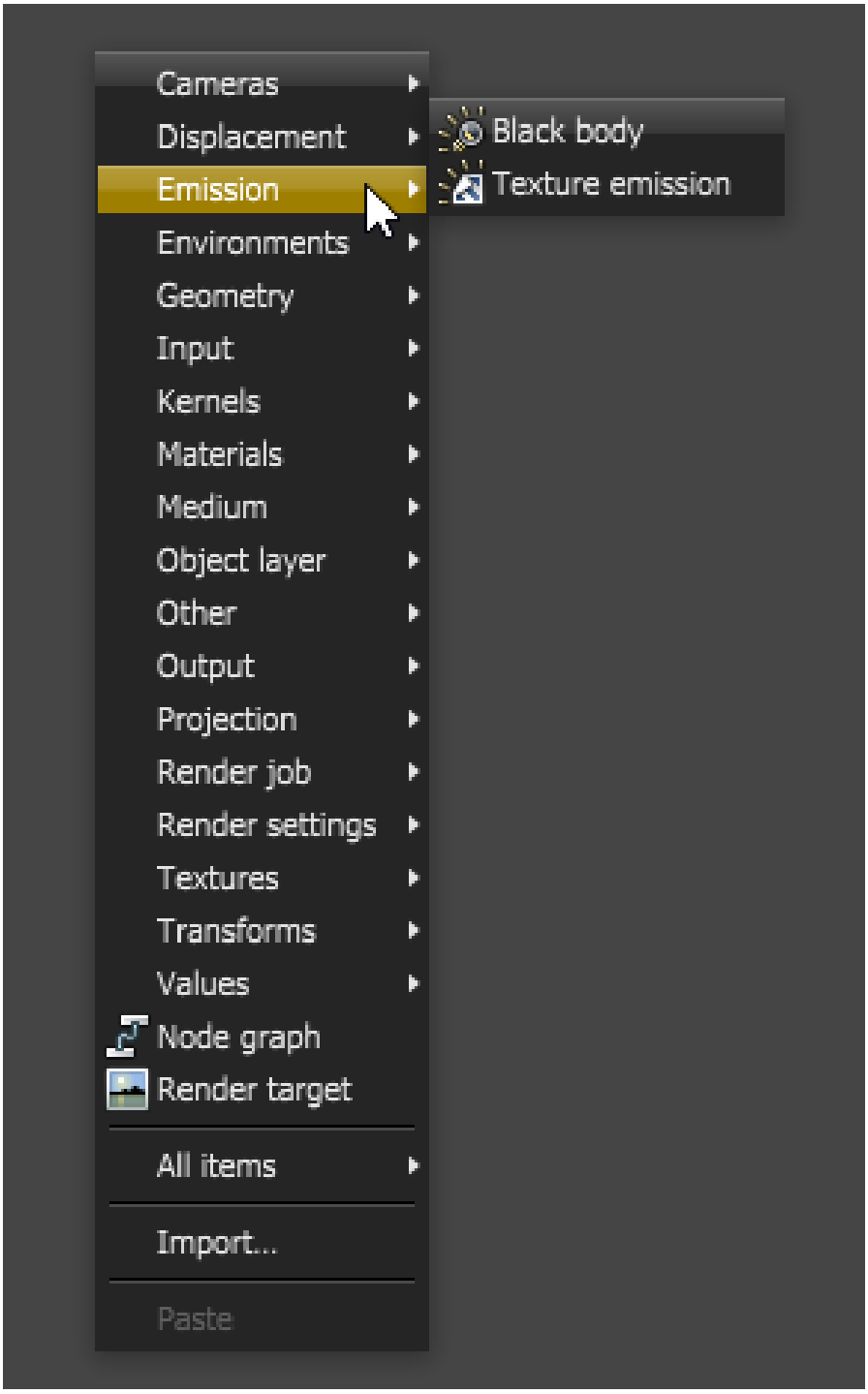


Figure 2: Emission nodes are added to the OctaneRender® scene using the pop-up menu

Note: When opening scenes built with previous versions (e.g. V3.06) in Octane V4, you will need to adjust some values in the emission nodes due to significant changes and improvements built in V4 affecting these nodes.

Black Body

The **Black Body**[1] emission uses **Temperature** (in Kelvin) and **Power** to control the color and intensity of the light, respectively. You can access the Black Body emission by right-clicking in the **Nodegraph Editor** and navigating to the **Emission** category (Figure 1).

[1] An opaque object that emits thermal radiation. In Octane, this is used to designate illumination properties for mesh emitters.

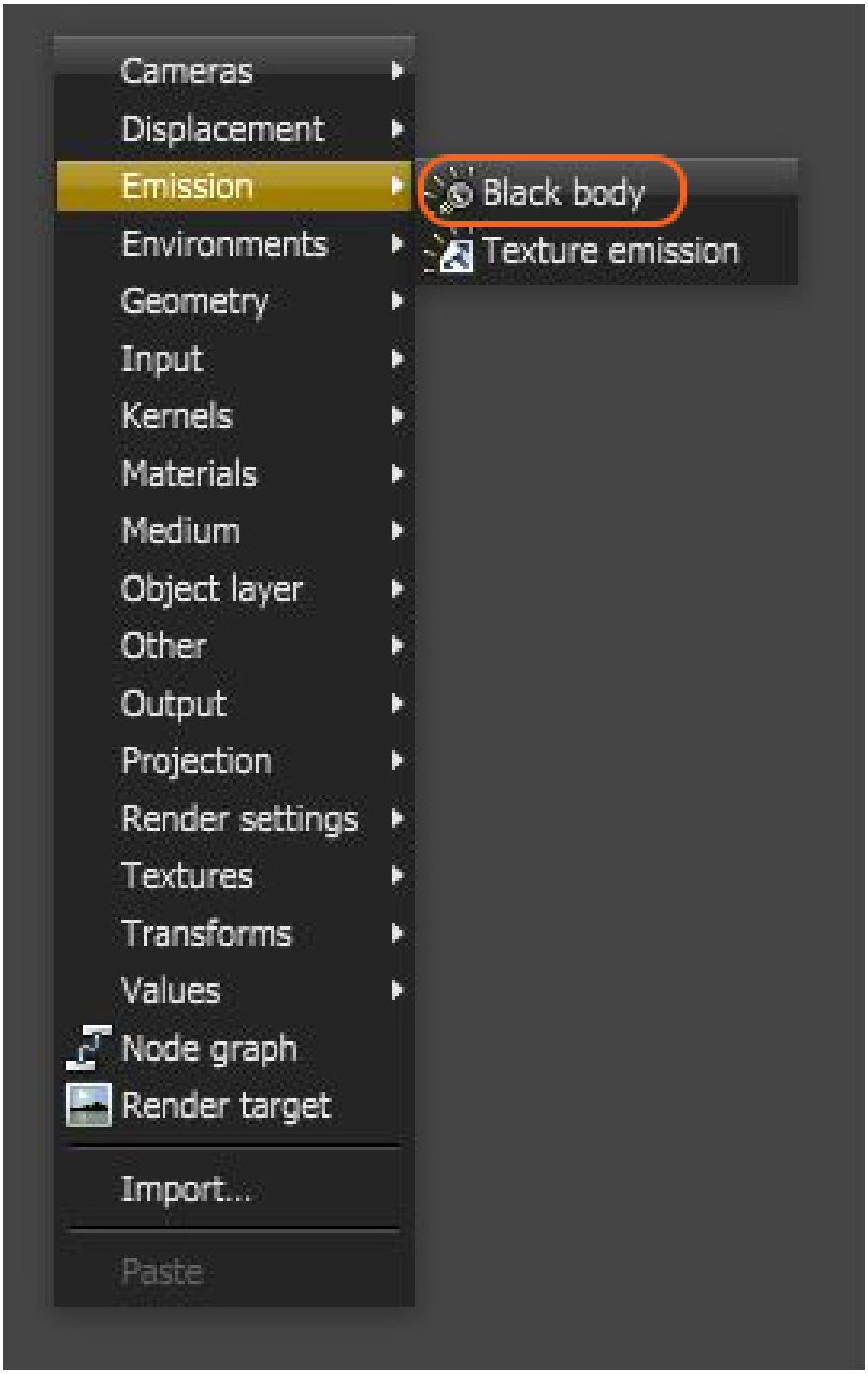


Figure 1: Emission nodes are added to the OctaneRender® scene using the pop-up menu

Black Body Emission Parameters

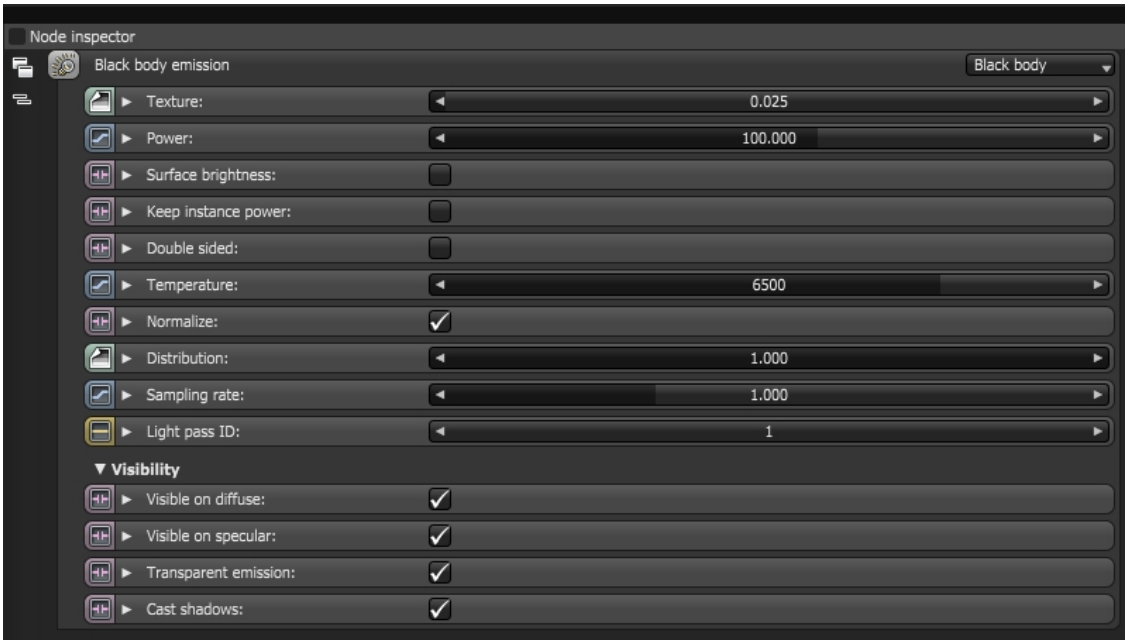
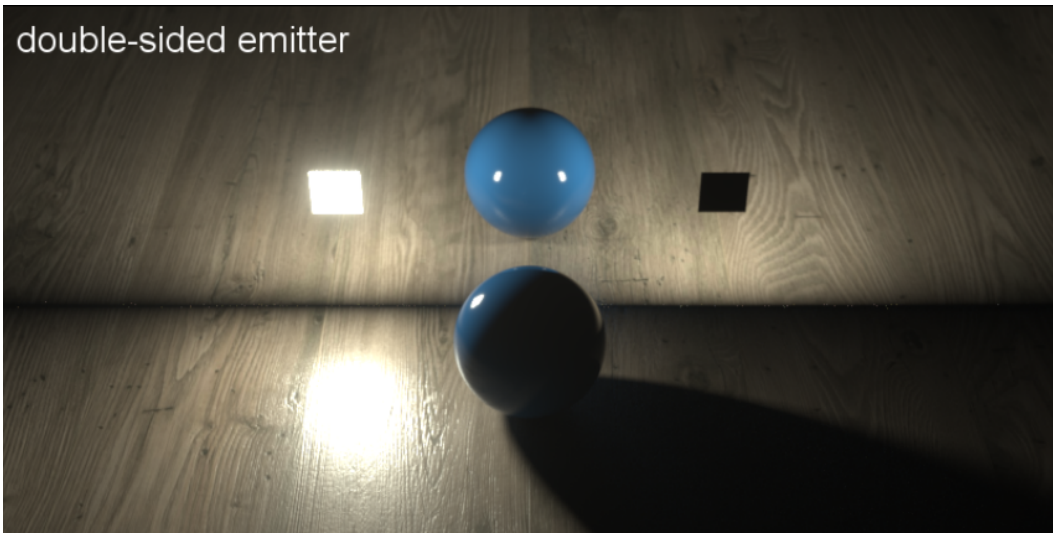


Figure 2: The Blackbody emission parameters

- **Texture** - Sets the light source's efficiency. You can set this to a value or texture. Keep in mind that real-world lights aren't 100% efficient at delivering power at their specified wattage - a 100-watt light bulb doesn't deliver 100 watts of light. This parameter enters the real-world values.
- **Power** - The light source's wattage. You should set each light to their real-world wattage - for example, set a desk lamp to **25** watts, a ceiling lamp to **100** watts, and an LED light to **0.25** watts.
- **Surface Brightness** - Causes emitters to keep a constant Surface Brightness, independent of the emitter surface area.
- **Keep Instance Power** - Enabling this option with Surface Brightness disabled and Uniform Scaling applied to the object causes Power to remain constant.

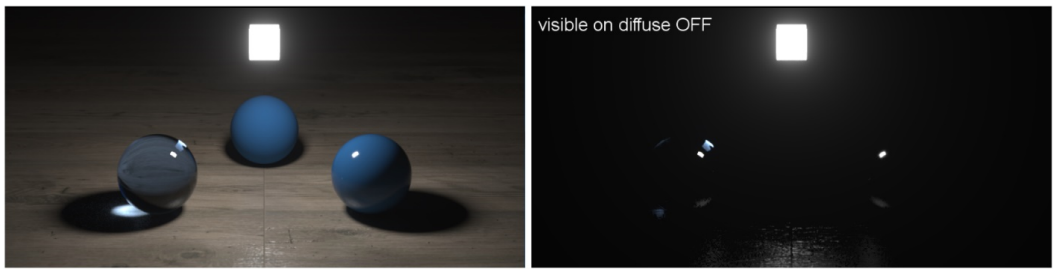
- **Double Sided** - Allows emitters to emit light from the front and back sides.



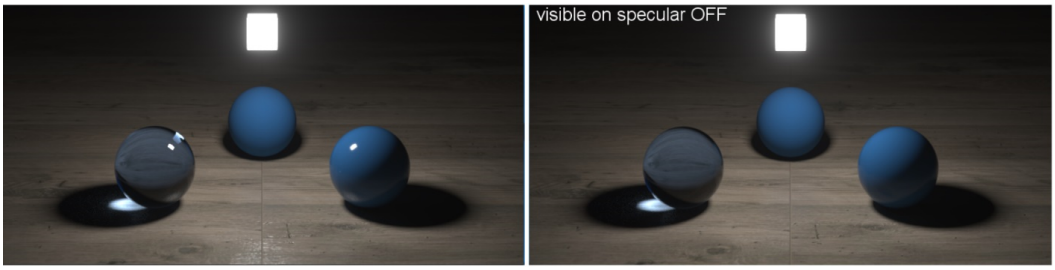
- **Temperature** - The temperature (in Kelvin) of the Black Body emission's light.
- **Normalize** - Ensures all normal vectors have the same length for the Black Body emission - this keeps the emitted light's luminance constant if the temperature varies. this is enabled by default.
- **Distribution** - Controls the light pattern. You can set this to a **Greyscale** or **RGB** image so that you can load an **Image** texture or **IES**[1] file. the Image texture's **Projection** nodes adjust the light's orientation and direction.
- **Sampling Rate** - Choose what light sources receive more samples.
- **Light Pass ID** - The Light Pass ID captures the respective emitter's contribution.
- **Visible On Diffuse**[2] - Enables light source visibility on diffuse surfaces. It allows you to enable or disable the ability for Black Body emission's or Texture emission's light sources to cast illumination or shadows on diffuse objects. Disabling this option disables emission - it's invisible in diffuse reflections, but is still visible on specular reflections. It's also excluded from the **Direct** light calculation. This option is

[1] An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.
[2] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

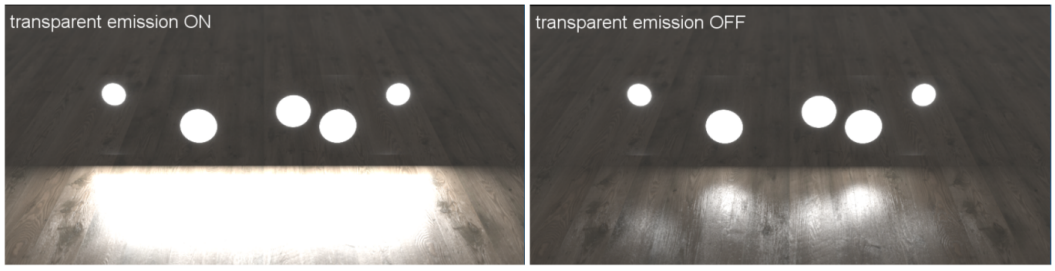
enabled by default.



- **Visible On Specular**[1] - Enables the light source's visibility on specular surfaces, and allows you to hide emitters on specular reflections/refractions. This is enabled by default.



- **Transparent Emission** - Allows light sources to cast illumination on diffuse objects, even if the light source is on transparent material.



[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

- **Cast Shadows** - Enables light sources to cast light and shadows on diffuse surfaces, letting you disable direct light shadows for **Mesh** emitters. To make this option work, the Direct light calculation must include the emitter (the sampling rate must be greater than **0**). This option is enabled by default.

Texture Emission

The **Texture** emission allows any valid **Texture** node to set the light intensity. You can use this to create interesting effects, such as TV screens, by using an **Image** texture as the source. You can access the Texture emission by right-clicking in the **Nodegraph Editor** and navigating to the **Emission** category (Figure 1).

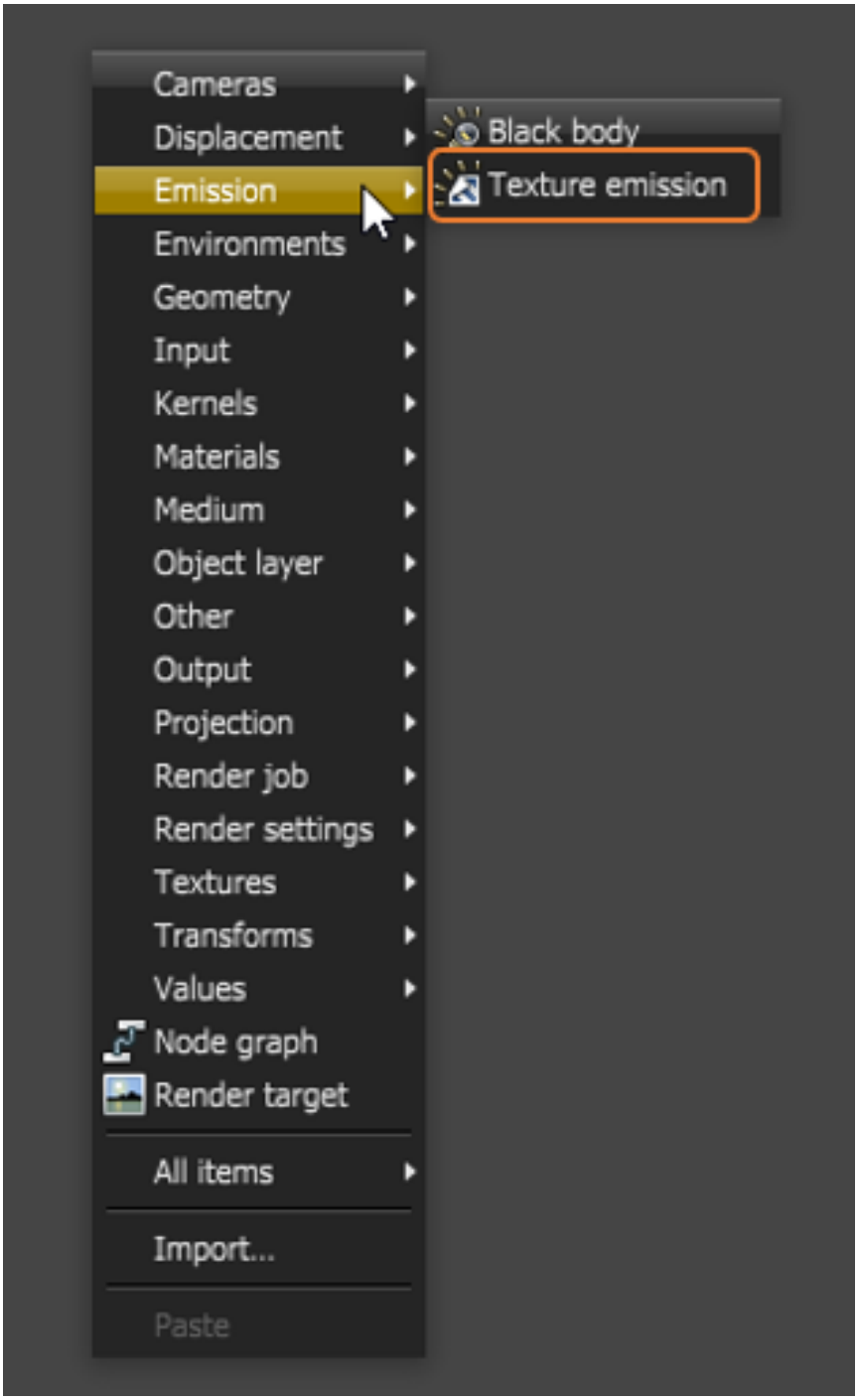


Figure 1: Accessing Emission nodes using the pop-up menu

Texture Emission Parameters

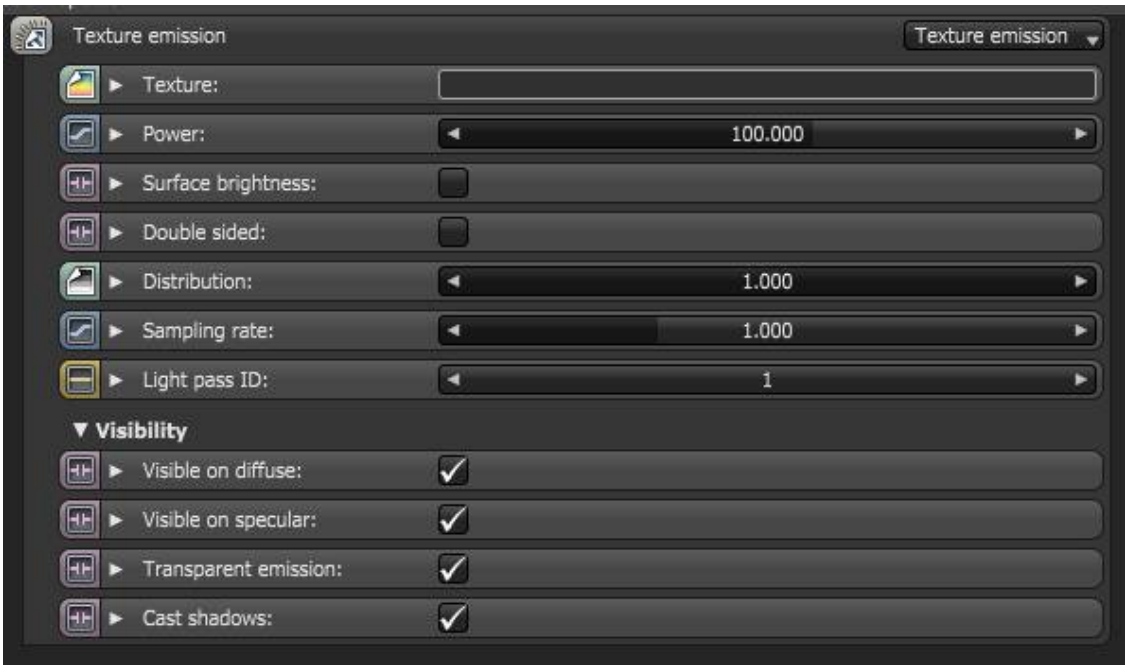
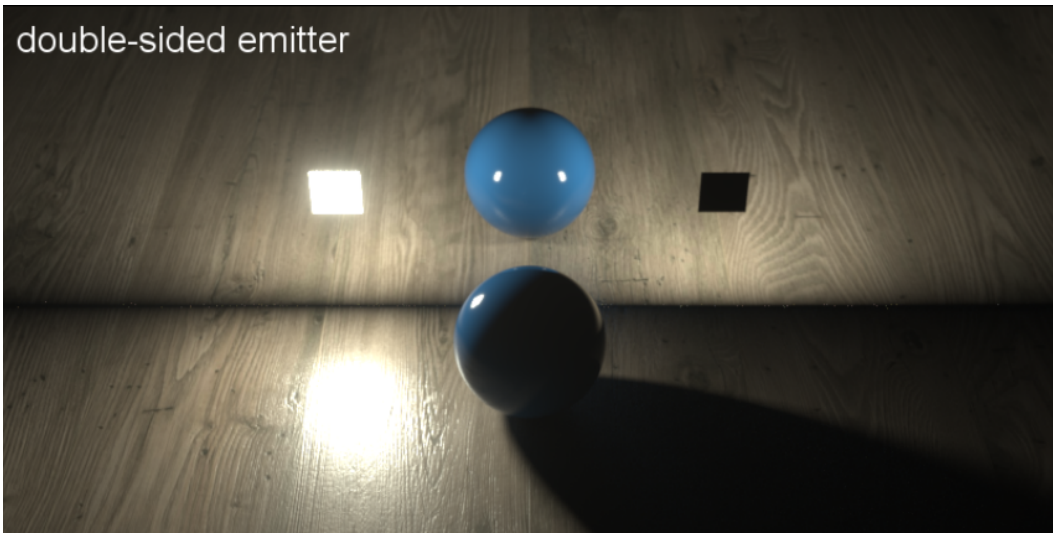


Figure 2: Texture emission parameters

- **Texture** - Sets the light source's efficiency. You can set this to a value or texture. Keep in mind that real-world lights aren't 100% efficient at delivering power at their specified wattage - a 100-watt light bulb doesn't deliver 100 watts of light. This parameter enters the real-world values.
- **Power** - The light source's wattage. You should set each light to their real-world wattage - for example, set a desk lamp to **25** watts, a ceiling lamp to **100** watts, and an LED light to **0.25** watts.
- **Surface Brightness** - Causes emitters to keep a constant Surface Brightness, independent of the emitter surface area.
- **Keep Instance Power** - Enabling this option with Surface Brightness disabled and Uniform Scaling applied to the object causes Power to remain constant.

- **Double Sided** - Allows emitters to emit light from the front and back sides.

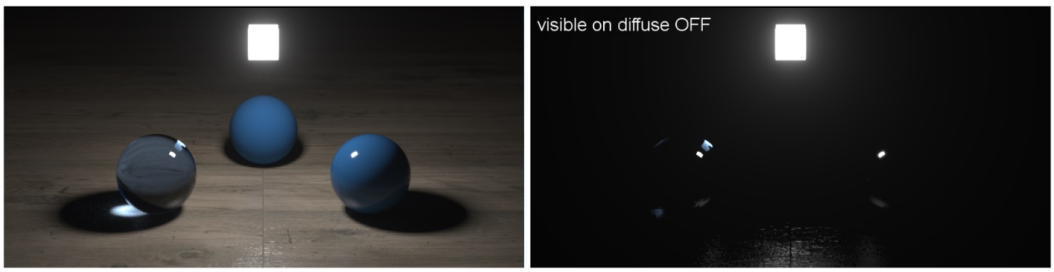


- **Distribution** - Controls the light pattern. You can set this to a **Greyscale** or **RGB** image so that you can load an **Image** texture or **IES**[1] file. the Image texture's **Projection** nodes adjust the light's orientation and direction.
- **Sampling Rate** - Choose what light sources receive more samples.
- **Light Pass ID** - The Light Pass ID captures the respective emitter's contribution.
- **Visible On Diffuse**[2] - Enables light source visibility on diffuse surfaces. It allows you to enable or disable the ability for **Black Body**[3] emission's or Texture emission's light sources to cast illumination or shadows on diffuse objects. Disabling this option disables emission - it's invisible in diffuse reflections, but is still visible on specular reflections. It's also excluded from the **Direct** light calculation. This option

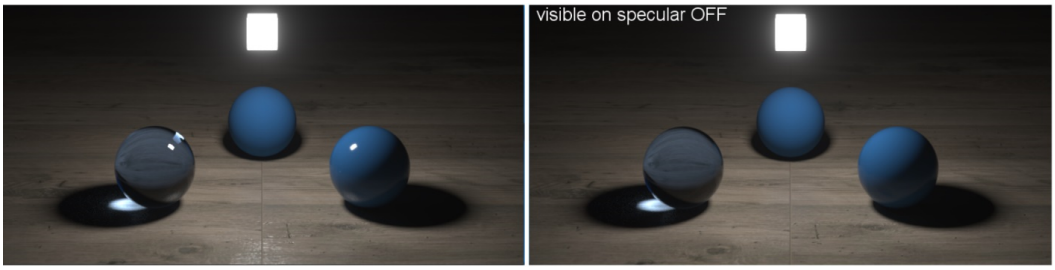
---

[1] An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.
[2] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[3] An opaque object that emits thermal radiation. In Octane, this is used to designate illumination properties for mesh emitters.

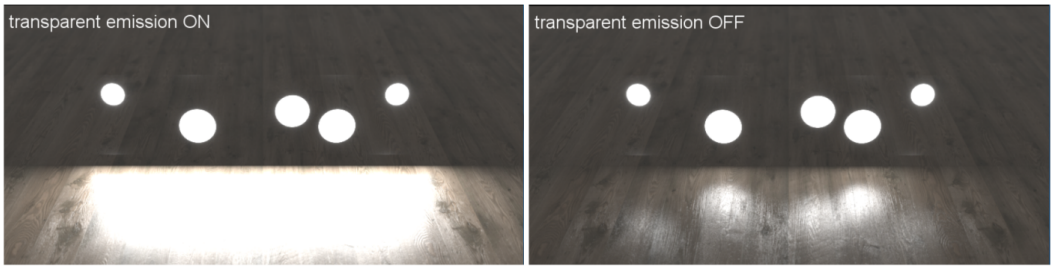
is enabled by default.



- **Visible On Specular**[1] - **Visible On Specular** - Enables the light source's visibility on specular surfaces, and allows you to hide emitters on specular reflections/refractions. This is enabled by default.



- **Transparent Emission** - Allows light sources to cast illumination on diffuse objects, even if the light source is on transparent material.



[1]Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

- **Cast Shadows** - Enables light sources to cast light and shadows on diffuse surfaces, letting you disable direct light shadows for **Mesh** emitters. To make this option work, the Direct light calculation must include the emitter (the sampling rate must be greater than **0**). This option is enabled by default.

Cameras

OctaneRender® for Unity® provides four types of cameras: the **Thin Lens**, **Panoramic**, **Baking Camera**, and **OSL** cameras. The Thin Lens and the Panoramic cameras also contain options for rendering stereoscopic images.

You can access the camera types in the **Inspector** window of the **PBR**[1] **Render Target**, under the **Camera** rollout (Figure 1).

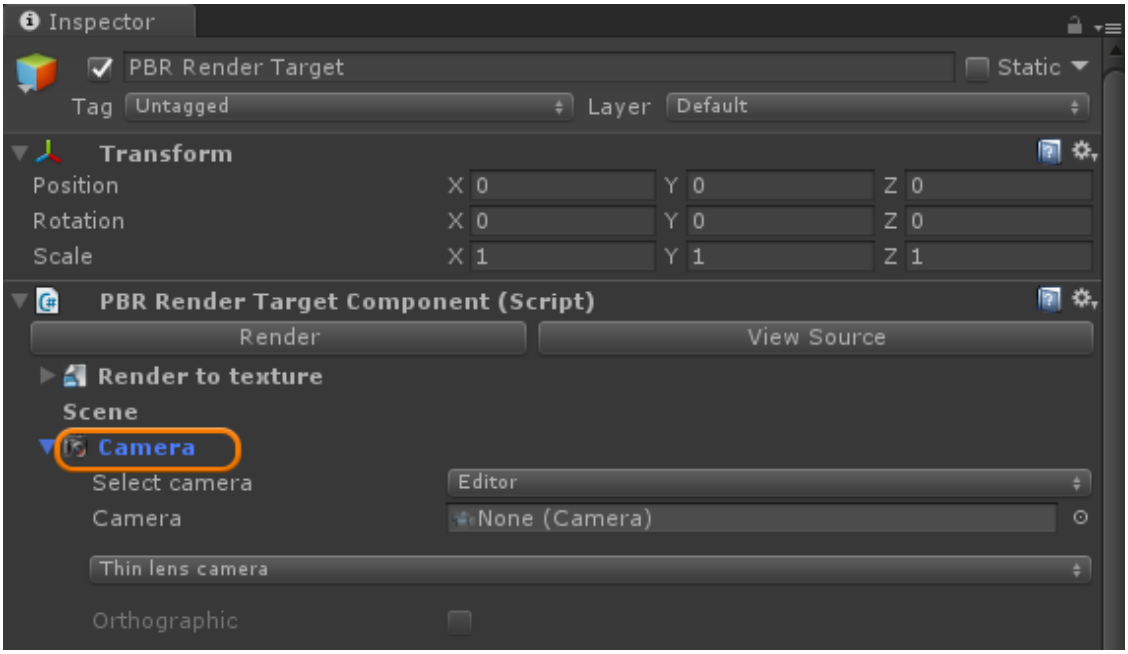


Figure 1: Accessing Camera types in the PBR Render Target

By default, the active camera OctaneRender uses is set to Unity's **Editor** view. This means that when you navigate through a scene in the **Scene** window, the **PBR Viewport** updates to reflect the navigation changes taking place in the Scene window (Figure 2).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

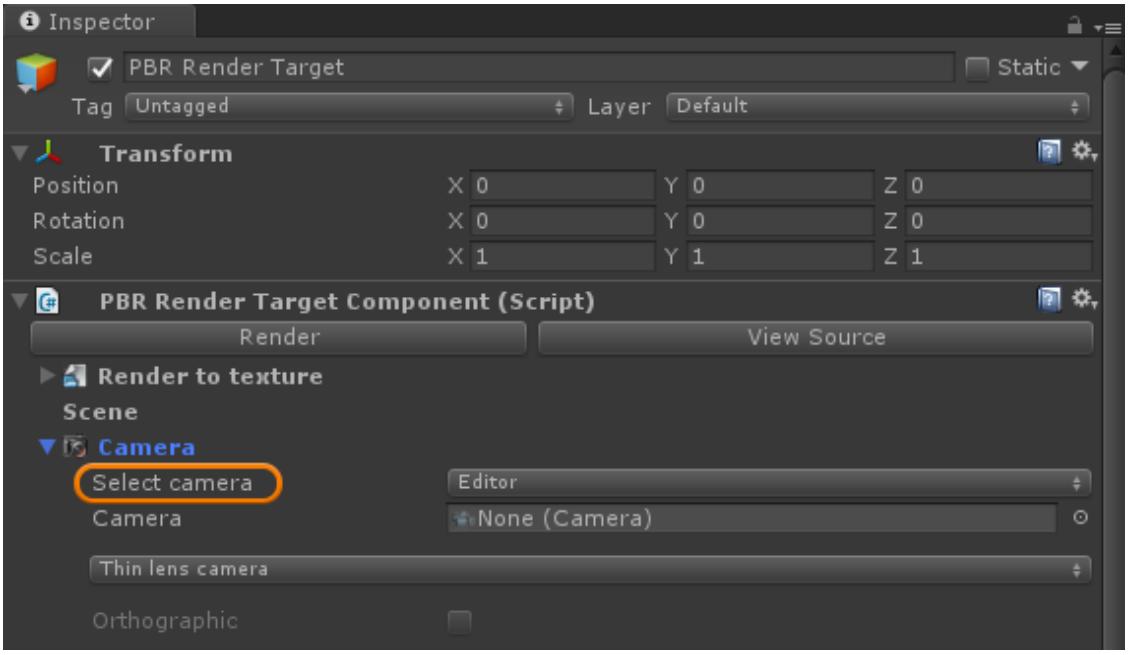


Figure 2: Determining which camera to use for rendering

There are four options for cameras under the **Select Camera** parameter:

- **Render Target** - Uses the camera specified in the **Camera** parameter. This can be any camera placed in the Unity scene.
- **Editor** - Uses the current Scene window view.
- **Game** - Uses the camera specified as the primary game camera.
- **Custom** - Allows you to render using any Unity camera in the scene.

Thin Lens

The **Thin Lens** camera is the most common camera type. It represents a normal camera view (Figure 1).

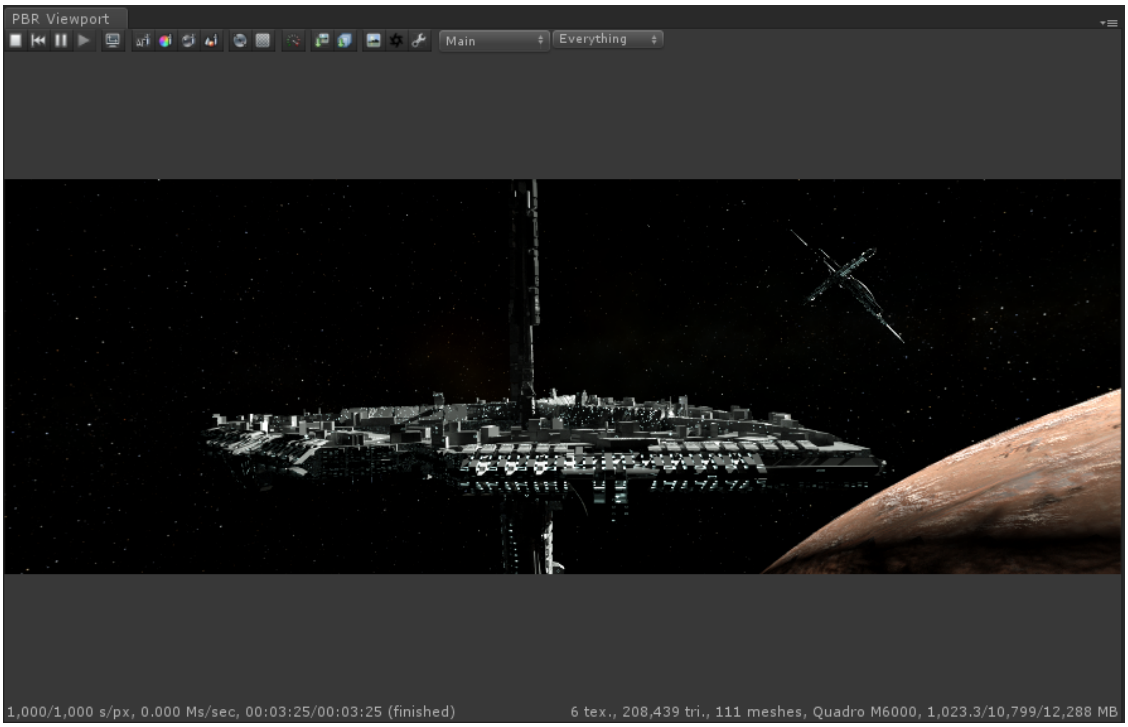


Figure 1: The Thin Lens camera type

Thin Lens Camera Parameters

- **Orthographic** - If enabled, the camera shows an Orthographic view. If disabled, the camera shows a **Perspective** view.
- **Physical Camera Parameters**
  - **Sensor Width** - The width of the sensor or film in millimeters.
  - **Focal Length** - The focal length of the lens in millimeters.
  - **F-Stop** - This is the aperture-to-focal-length ratio.

- **Viewing Angle**
  - **Field of View**[1] - Sets the horizontal field of view for the camera in the scene, measured in degrees. This functions much like a variable zoom camera lens.
  - **Scale of View** - Sets the width of the Orthographic view of the camera, measured in meters.
  - **Distortion** - Adjusts the spherical and cylindrical distortion. The rendered image displays the entire sphere, and uses equidistant cylindrical projection, also known as lat-lon projection.
  - **Lens Shift** - This is useful for architectural rendering, when rendering tall buildings or structures from a similar height as the human eye. This parameter helps keep the vertical lines parallel.
  - **Perspective Correction** - If the up-vector is vertical, enabling this option keeps vertical lines parallel.
  - **Pixel Aspect Ratio** - Allows you to stretch or squash the **Depth of Field**[2] disc and render it to a non-square pixel format (NTSC or PAL).
- **Clipping**
  - **Near Clip Depth** - Distance from the camera to the near clipping plane, measured in meters.
  - **Far Clip Depth** - Distance from the camera to the far clipping plane, measured in meters.
- **Depth of Field**
  - **Auto-Focus** - If enabled, the focus is kept on the closest visible surface at the center of the image, regardless of the **Aperture**[3], the Aperture Edge, and **Focal Depth** values.
  - **Focal Depth** - The depth of the area in focus, measured in meters.
  - **Aperture** - The camera lens opening's radius, measured in centimeters. Choosing a low value creates a wide depth-of-field, where everything is in focus. Choosing a high value creates a shallow depth-of-field, where objects in the foreground and background are out of focus.
  - **Aperture Aspect Ratio** - This allows you to stretch and squash the depth-of-field disc.
  - **Aperture Edge** - Controls aperture edge detection at all points within the aperture, and modifies the bokeh effect. The lower values produce more pronounced edges to out-of-focus objects affected by a shallow depth-of-field, such as objects in the foreground and background. A high value increases the contrast.
  - **Bokeh Side Count** - The number of edges making up the bokeh shape.
  - **Bokeh Rotation** - The bokeh shape orientation.
  - **Bokeh Roundedness** - The roundedness of the bokeh shape's sides.

---

[1] The area that is visible to a camera lens usually measured in millimeters. A wide angle lens provides a larger field of view and a telephoto lens provides a narrow field of view.

[2] The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

[3] Determines how much light enters a camera lens. A large aperture produces a narrow depth of field and a small aperture produces a wide depth of field.

Panoramic Camera

The **Panoramic camera** creates 360-degree, immersive rendering. These can be **Spherical**, **Cylindrical**, or **Cube Mapped** (Figure 1).

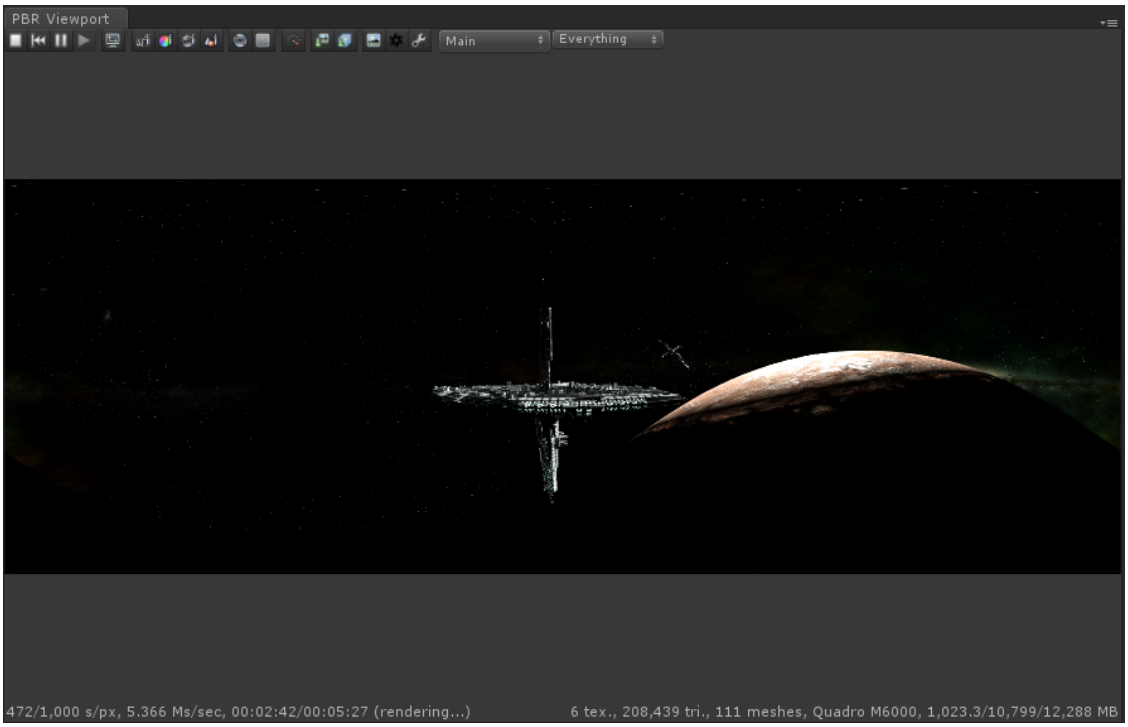


Figure 1: Panoramic camera using the Spherical projection

Panoramic Camera Parameters

- **Projection** - Specifies the panoramic projection to use: **Spherical**, **Cylindrical**, or **Cube Map**. Single-face Cube Map projections are available, and letting you render only one face of the cube. This is useful for animation overlays in stereo panorama renderings.
- **Physical Camera Parameters**
  - **Focal Length** - The lens's focal length in millimeters.
  - **F-Stop** - The aperture-to-focal-length ratio.

- **Viewing Angle**
  - **Horizontal Field of View**[1] - The horizontal field of view in degrees. This sets the x-coordinate for the camera's horizontal field of view in the scene. This is ignored when using Cube Mapping.
  - **Vertical Field of View** - The vertical field of view in degrees. This sets the y-coordinate for the camera's vertical field of view. This is ignored when using Cube Mapping.
  - **Keep Upright** - If enabled, the Panoramic camera is always oriented towards the horizon, and the up-vector stays in its default direction (vertical) at (0,1,0).
- **Clipping**
  - **Near Clip Depth** - Distance from the camera to the near clipping plane, measured in meters.
  - **Far Clip Depth** - Distance from the camera to the far clipping plane, measured in meters.
- **Depth of Field**[2]
  - **Auto-Focus** - If enabled, the focus is kept on the closest visible surface at the center of the image, regardless of the **Aperture**[3], **Aperture Edge**, and **Focal Depth** values.
  - **Focal Depth** - The focal area's depth, measured in meters.
  - **Aperture** - The camera lens opening's radius, measured in centimeters. Choosing a low value creates a wide depth-of- field, where everything is in focus. Choosing a high value creates a shallow depth-of- field, where objects in the foreground and background are out of focus.
  - **Aperture Aspect Ratio** - This allows you to stretch and squash the depth-of-field disc.
  - **Aperture Edge** - Controls aperture edge detection at all points within the aperture, and modifies the bokeh effect. The lower values produce more pronounced edges to out-of-focus objects affected by a shallow depth-of-field, such as objects in the foreground and background. A high value increases the contrast.
  - **Bokeh Side Count** - The number of edges making up the bokeh shape.
  - **Bokeh Rotation** - The bokeh shape orientation.
  - **Bokeh Roundedness** - The roundness of the bokeh shape's sides.

---

[1] The area that is visible to a camera lens usually measured in millimeters. A wide angle lens provides a larger field of view and a telephoto lens provides a narrow field of view.

[2] The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

[3] Determines how much light enters a camera lens. A large aperture produces a narrow depth of field and a small aperture produces a wide depth of field.

Stereo Options

Stereo parameters are found in the **Inspector** window of the **PBR**[1] **Render Target**, towards the bottom of the **Camera** rollout. The **Thin Lens** and **Panoramic** cameras contain identical options for stereo output (Figure 1).

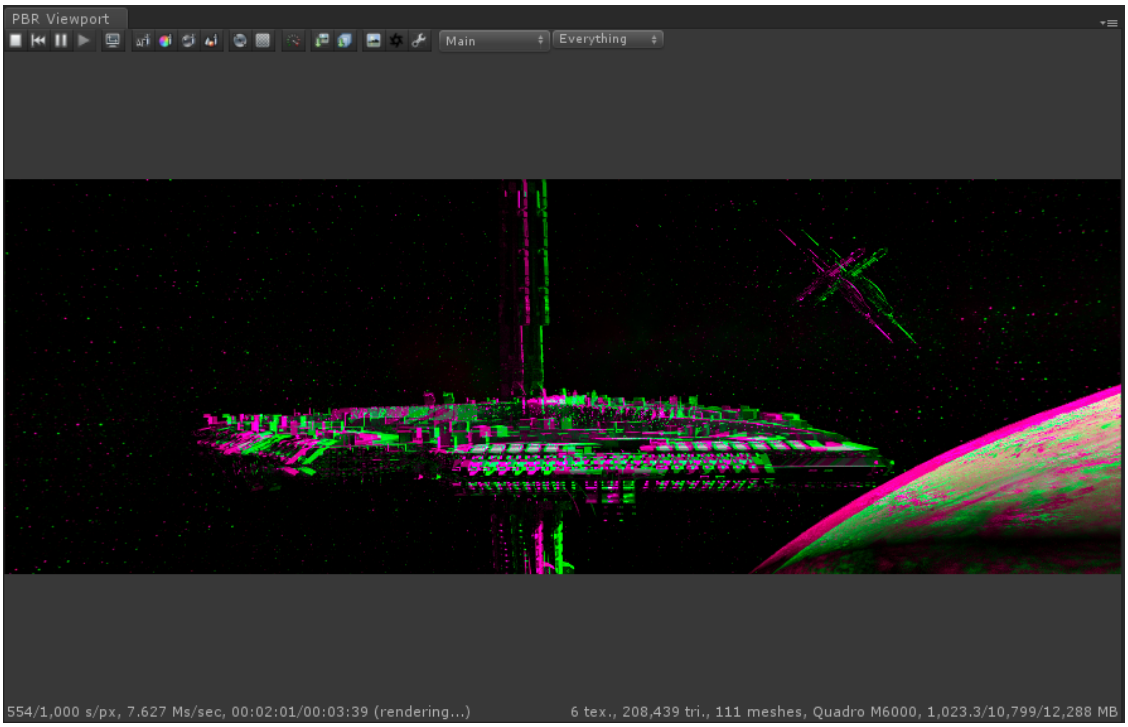


Figure 1: Stereo enabled with Anaglyphic output

Stereo Parameters

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

- **Stereo Output** - This specifies the output rendered in stereo mode.
  - **Left** - Renders only the image for the left eye.
  - **Right** - Renders oly the image for the right eye.
  - **Side-By-Side** - Renders a single image, with the left and right eye renders next to each other.
  - **Anaglyphic** - Renders the image so you can view it with red/blue 3D glasses.
  - **Over-Under** - The left and right eye images are placed one above the other for special viewers.
- **Eye Distance** - The distance between the left and the right eye in stereo mode, measured in meters.
- **Eye Distance Falloff** - Controls how quickly OctaneRender® reduces the eye distance towards the poles. This reduces eye strain at the poles when you view the panorama through a head-mounted display.
- **Pano Blackout Latitude** - The +/- latitude where OctaneRender cuts off the panorama when stereo rendering is enabled. This defines the minimum latitude (in spherical camera coordinates) where the rendering blacks out areas with higher latitudes.
- **Swap Eyes** - This swaps the left and right eye positions when stereo mode shows both.
- **Left Stereo Filter/Right Stereo Filter** - The left and right filters adjust the colors used to create the analglyphic stereo effect in the render.

Baking Camera

The **Baking Camera** is at the heart of the lighting and texture baking process. The texture baking process extracts lighting information from a mesh's surface, then generates a texture that can map back to the mesh at a later time (Figure 1). This process is covered in detail in the **"Lighting" on page 263** and **Baking Texture** topics.

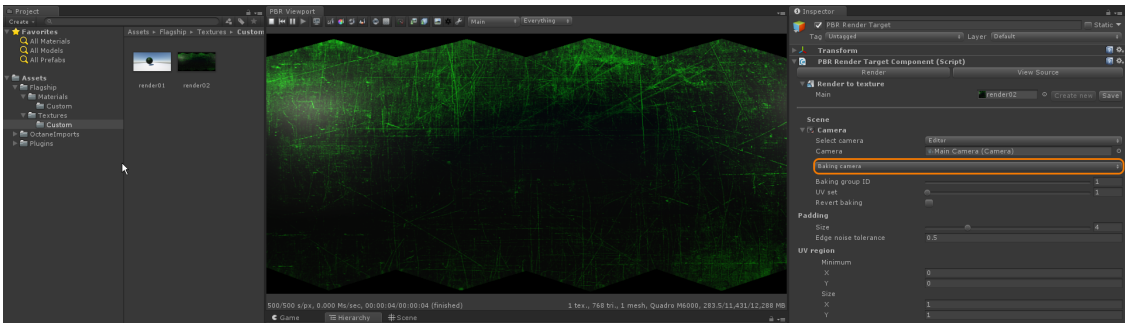


Figure 1: A scene asset baked to texture using the Baking Camera

Rendering

Most of the primary rendering controls for OctaneRender® are located in the **PBR[1] Render Target** (Figure 1). Additional controls for camera and environment are also found here, but are covered in more detail in the **"Cameras" on page 308** and **"Lighting" on page 263** sections of this manual.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

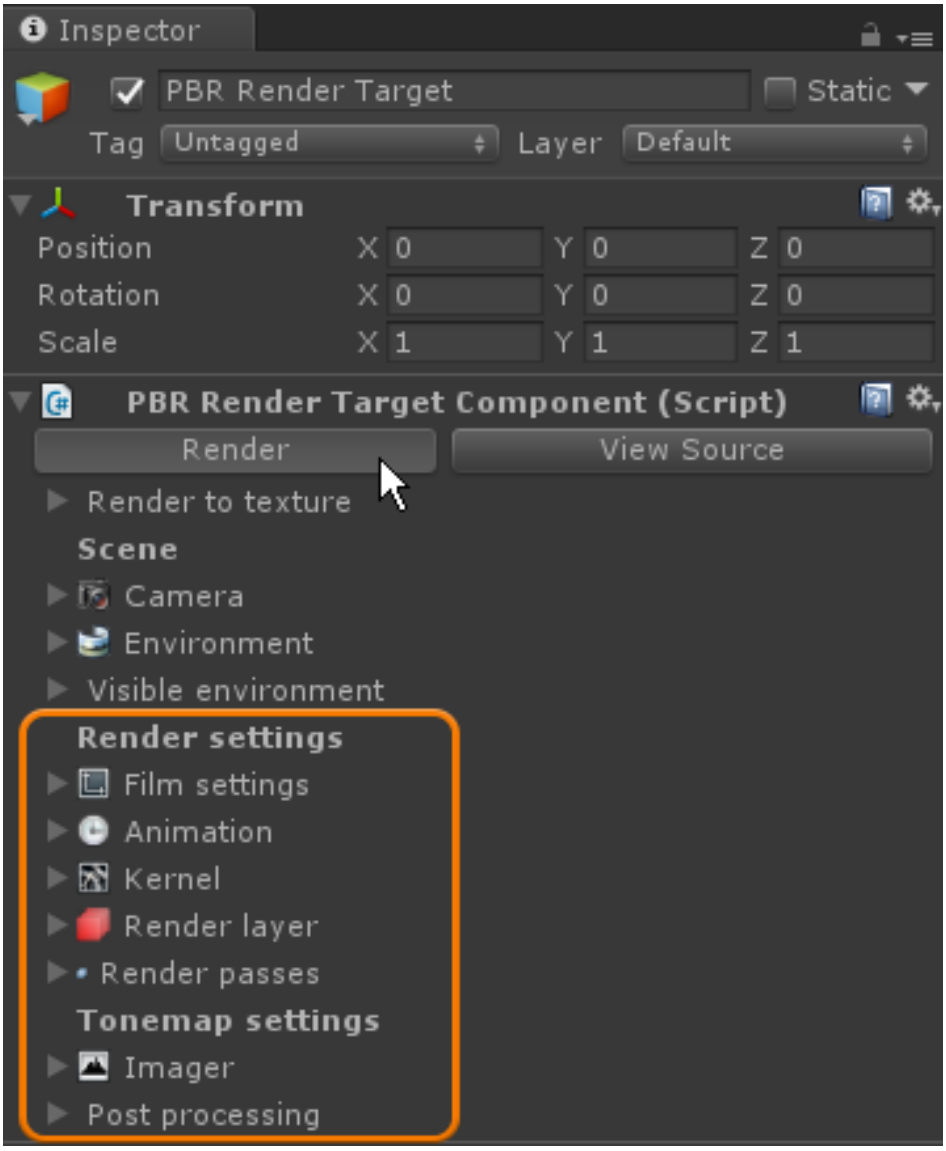


Figure 1: The Render Settings located in the PBR Render Target

PBR Render Target

The **PBR**[1] **Render Target** is the control center for rendering with OctaneRender® for Unity®. A Unity scene can have multiple PBR Render Targets. You can add PBR Render Targets from the **Octane** menu (Figure 1), or by right-clicking in the **Hierarchy** window and choosing **Create PBR Render Target** (Figure 2).

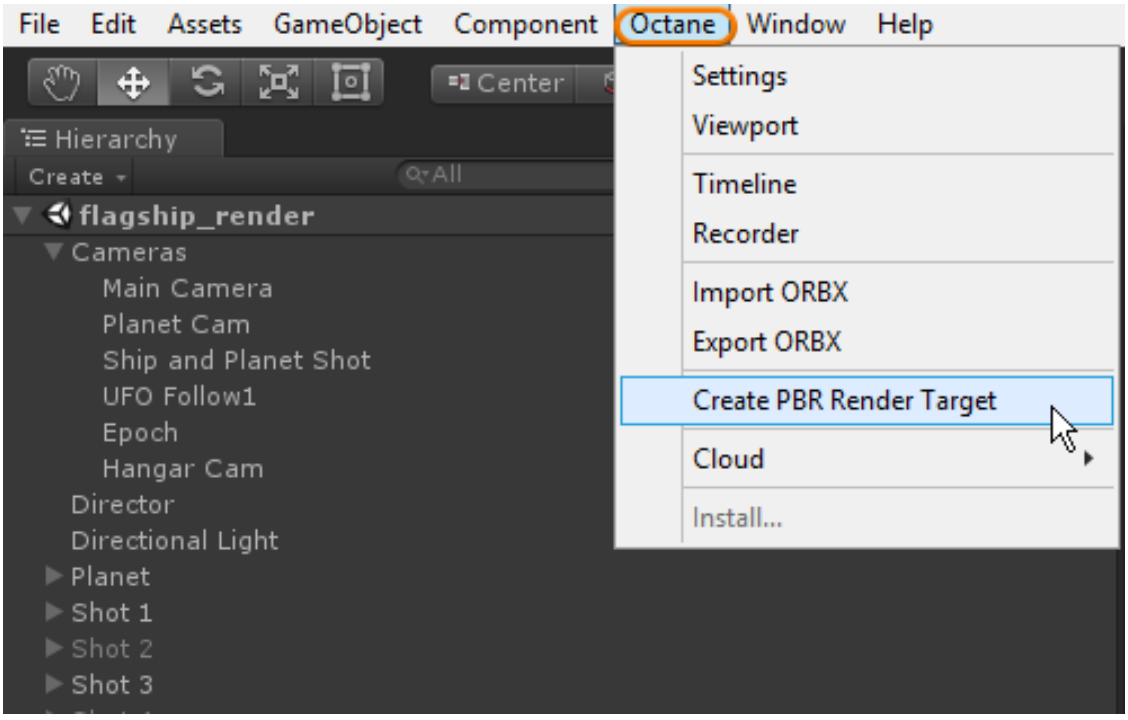


Figure 1: Creating a PBR Render Target from the Octane menu

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

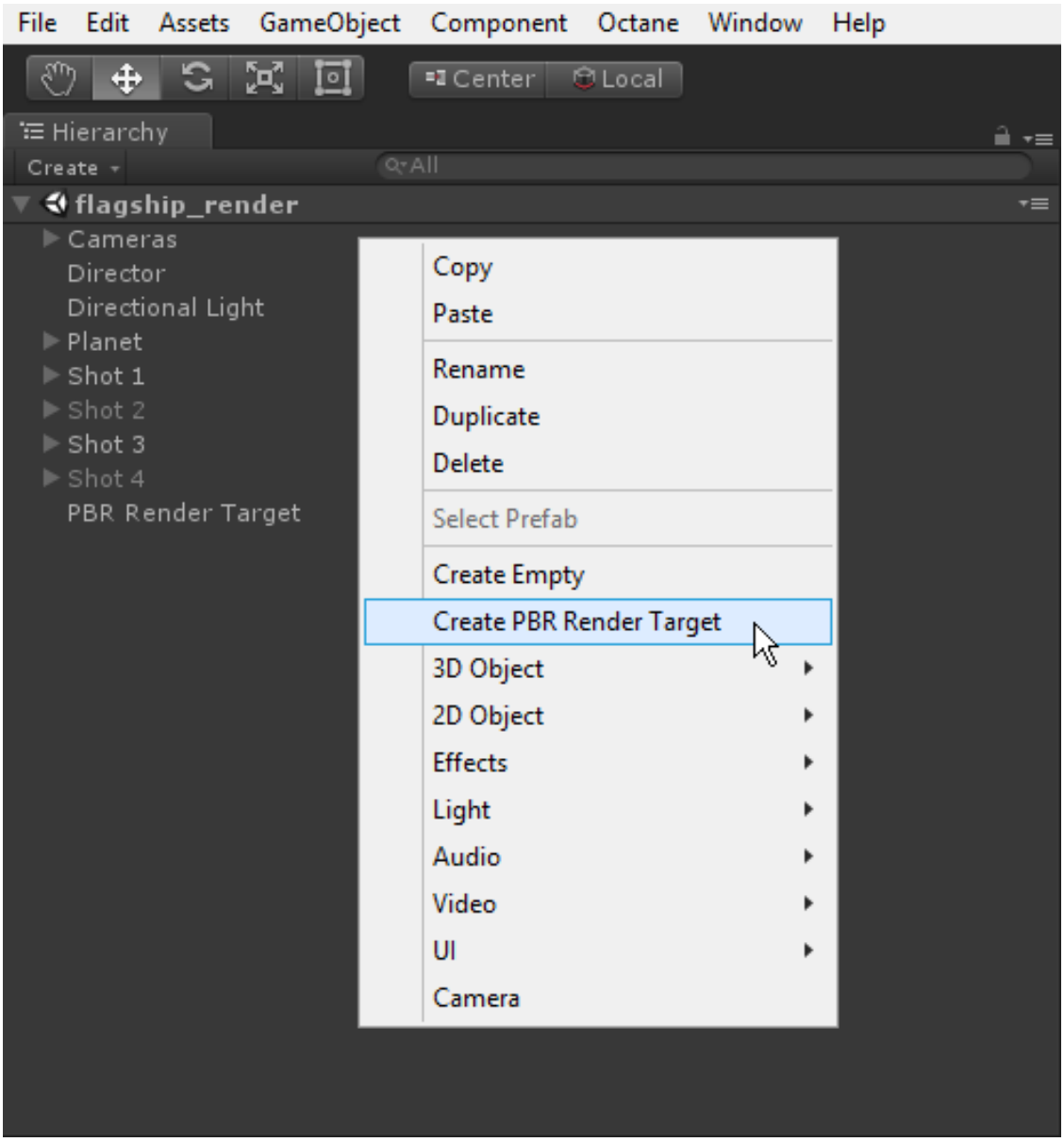


Figure 2: Create a PBR Render Target from right-clicking in the Hierarchy window

Film Settings

The **Film Settings** rollout (Figure 1) provides adjustments to the overall render result resolution, and it also sets interactive or non-interactive region render settings in conjunction with the region render features in OctaneRender®.

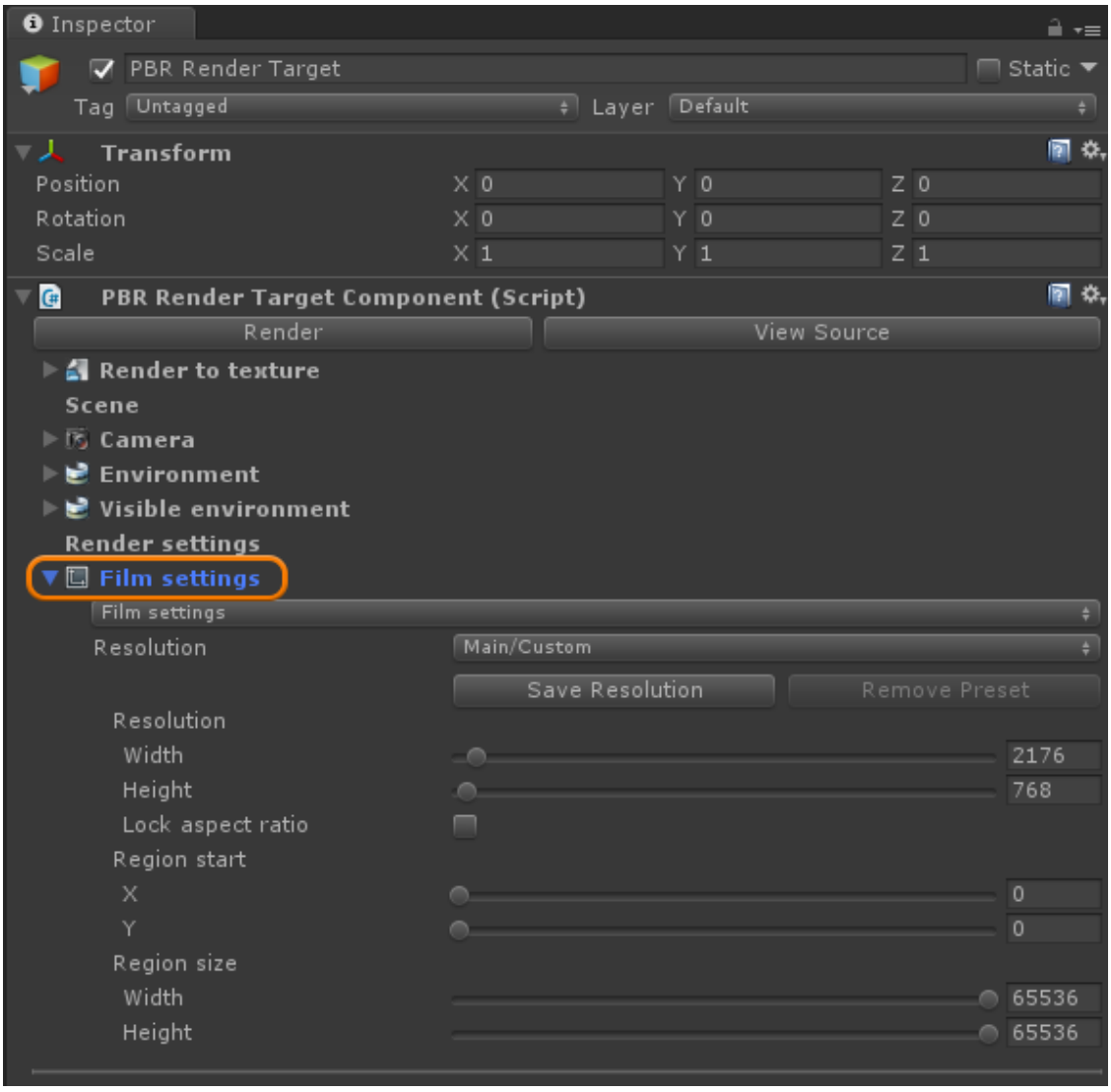


Figure 1: The PBR[1] Render Target's Film Settings

Film Settings Parameters

- **Resolution** - Determines the scene's rendering resolution. There are many presets available from the **Resolution** dropdown menu, otherwise you can set a custom resolution using the **Width** and **Height** parameters.
- **Lock Aspect Ratio** - This locks the aspect ratio of height-to-width. If you change one parameter, the other changes accordingly.
- **Region Start** - Determines where a region render begins in the **PBR Viewport** window. The upper-left of the PBR Viewport is [0, 0].
- **Region Size** - Determines how many pixels are rendered in the PBR Viewport, starting from the values specified under Region Start.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Animation

You can render animations designed in Unity® via OctaneRender® through the batch rendering process discussed in this topic. OctaneRender for Unity provides two tools for rendering Unity animations: **Timeline**, and **Recorder** (Figure 1). Motion blur is also a consideration when rendering animations, and the **Animation Settings** section covers this topic in more detail.

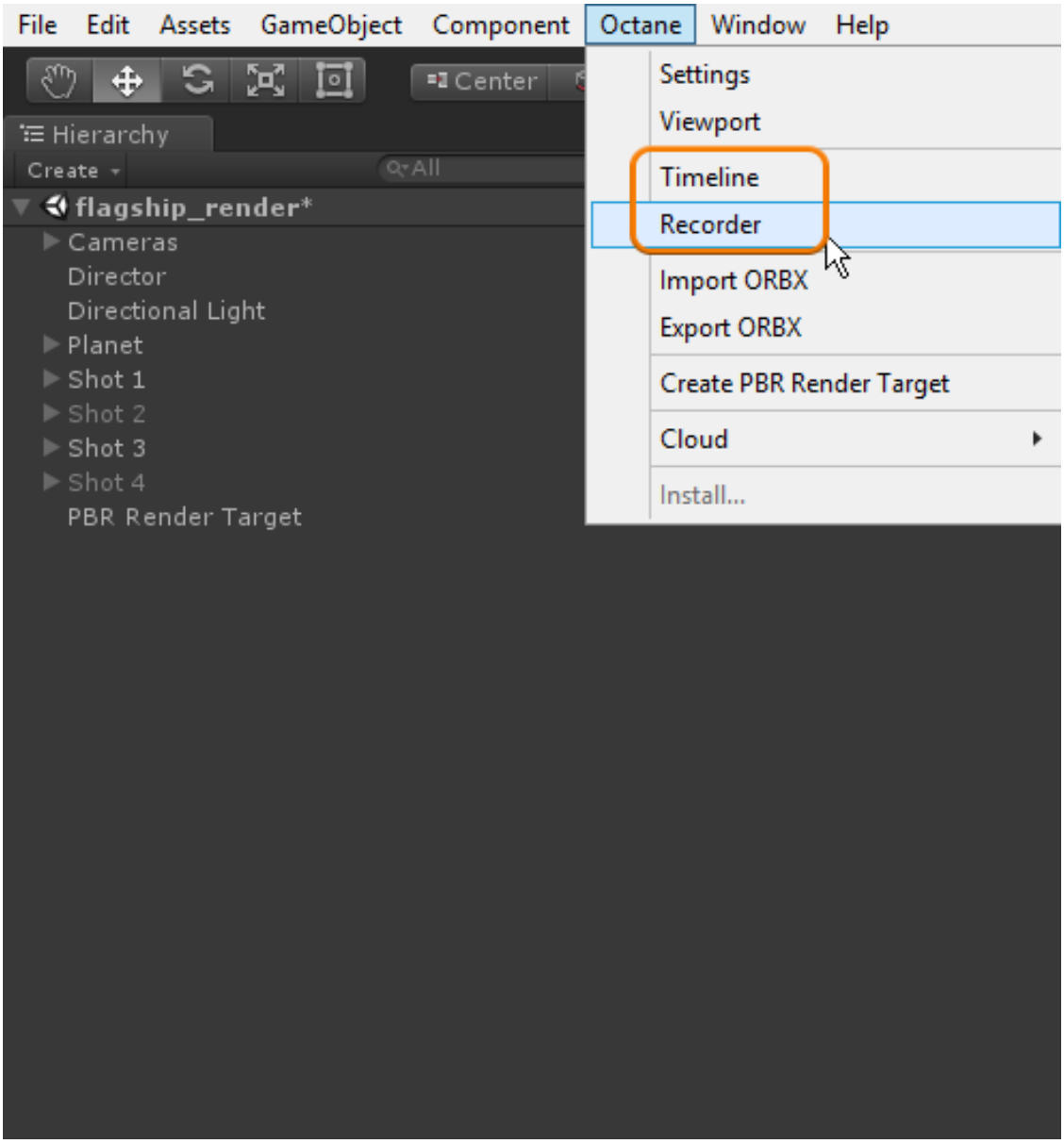


Figure 1: Accessing the OctaneRender Timeline and Recorder windows from the Octane menu

Animation Settings

Motion blur is a common phenomenon in photography, especially when capturing fast-moving objects. It occurs either when: a) the camera has moved, thereby the focal point is no longer the same from the time the camera shutter opens to the time the shutter closes (camera blur); or b) when the object is moving so fast that the camera's shutter speed is not quick enough to capture a sharp image of the object (object motion blur). In OctaneRender®, motion blur effects are applied through the **Animation** rollout in the **PBR**[1] **Render Target**. The scene also needs animated geometry in it (Figure 1).

[1]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

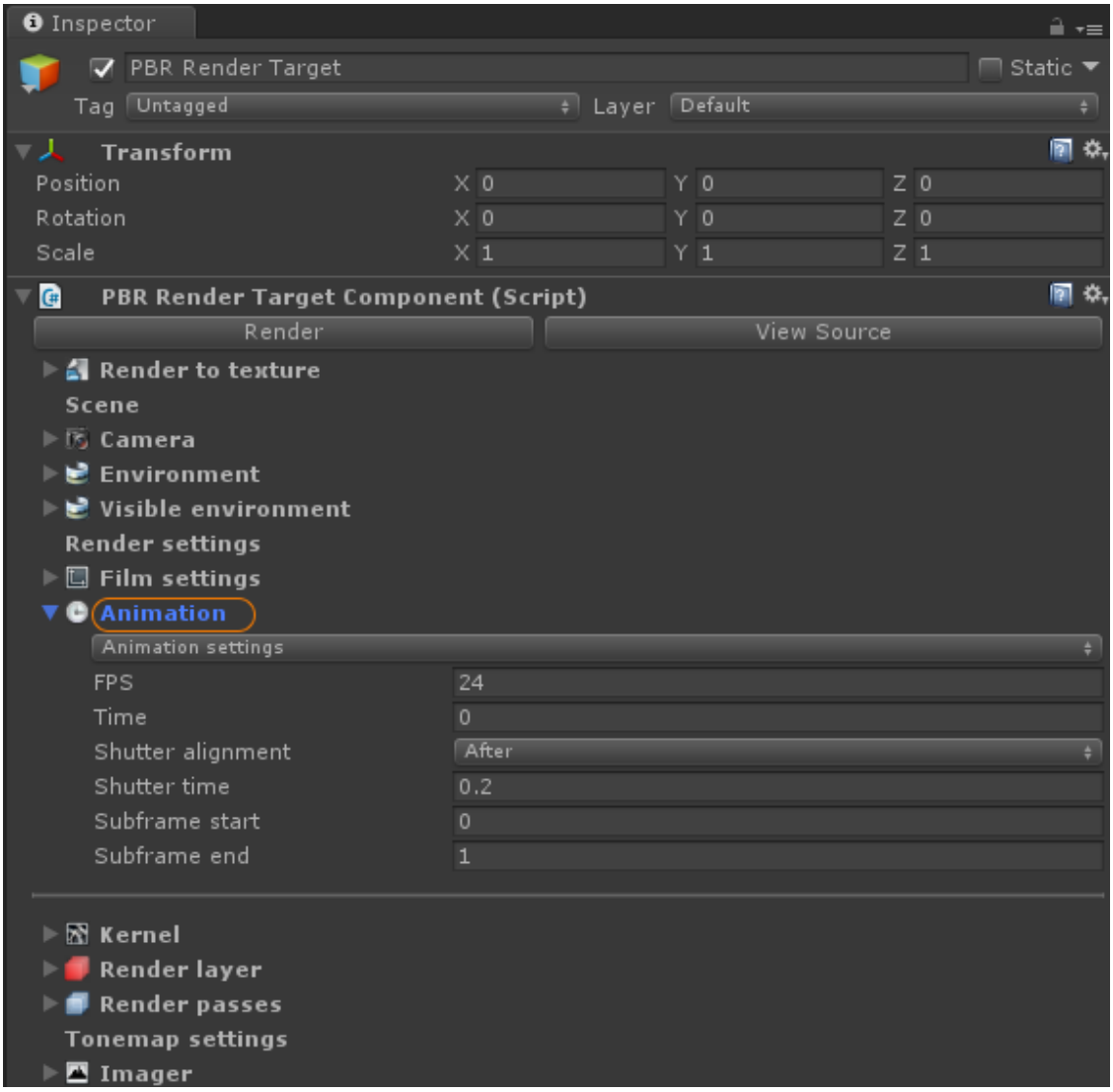


Figure 1: The Animation settings in the PBR Render Target

The Animation Settings Parameters

- **FPS** - Specifies the number of frames per second, where a frame is a different rendered image, each processed and shown consecutively to simulate motion within a scene.
- **Time** - This specifies the shutter time percentage relative to the duration of a single frame, and it controls how much time the shutter stays open. You can set this to any value above 100% manually.

- **Shutter Alignment** - Specifies how the shutter interval is aligned to the current time. This determines when the camera shutter is triggered. The options are **Before**, **Symmetrical**, or **After**, and they apply to each frame thereafter relative to the given frame rate (Figure 2).

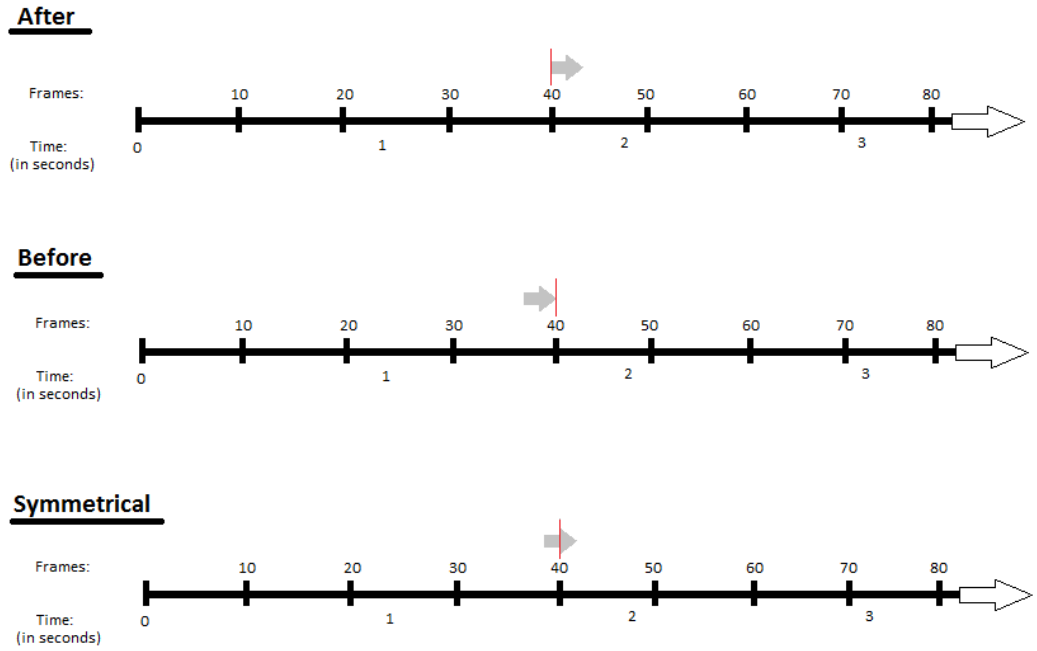


Figure 2: Illustrating After, Before, and Symmetrical Shutter Alignment

- **Shutter Time** - This specifies the shutter time percentage relative to the duration of a single frame. Shutter Time controls how much time the shutter stays open. You can set this parameter to any value above 100%.
- **Subframe Start/Subframe End** - Specifies the approach, in terms of proportion (%) to simulate the camera's shutter speed for that particular frame. OctaneRender uses Subframe Start and End percentages to render only a portion of a particular frame. If the scene has a lot of motion blur, OctaneRender uses these parameters to render only a piece of that motion blur. Values of **0%** and **100%** render the whole frame (default).

Recorder

Recorder combines batch rendering and **Play Mode** to render animations in Play Mode and save the rendered images to disc (Figure 1). This works for animation built with the Unity® **Timeline** as well as animations triggered during gameplay. However, Recorder does not necessarily work in real-time because the scene needs compilation and rendering for OctaneRender®.

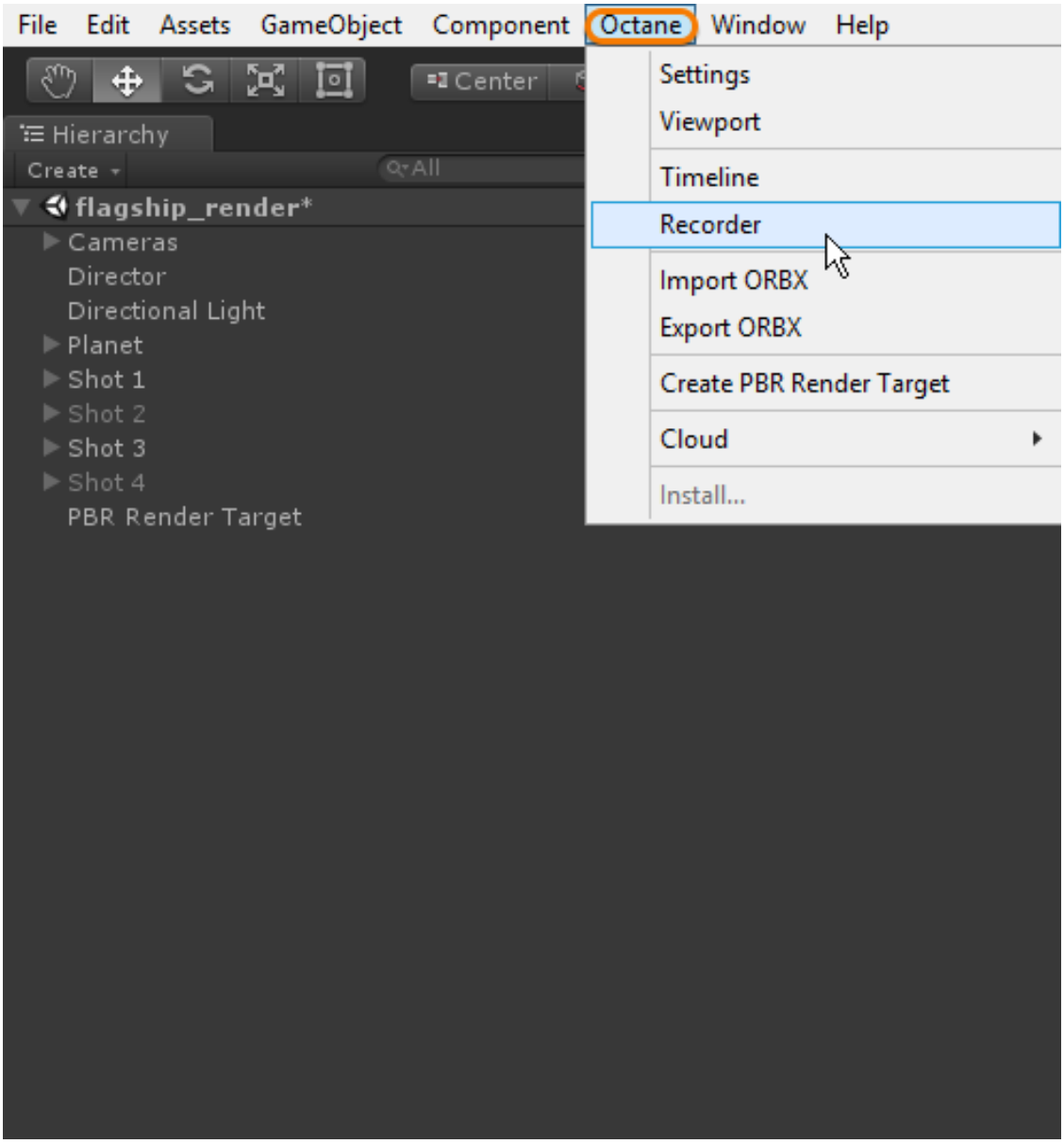


Figure 1: Accessing the Recorder from the Octane menu

To use Recorder with OctaneRender:

1. Start with a scene that has animated elements. Make sure that OctaneRender is installed and loaded.
2. Add a **PBR**[1] **Render Target** and adjust the render settings accordingly.
3. Open the **Recorder** window from the **Octane** menu.
4. Set the appropriate preferences in the Recorder window.
5. Press the **Start** button. The **PBR viewport** opens and OctaneRender compiles the scene. The scene then renders and updates once each render is complete, saving each frame to disc. If an animation is present, OctaneRender plays through the animation and the render similar to a batch render. If the game is playing, OctaneRender renders each frame of the game play. (Note: Real-time rendering may not be possible during game play mode.)

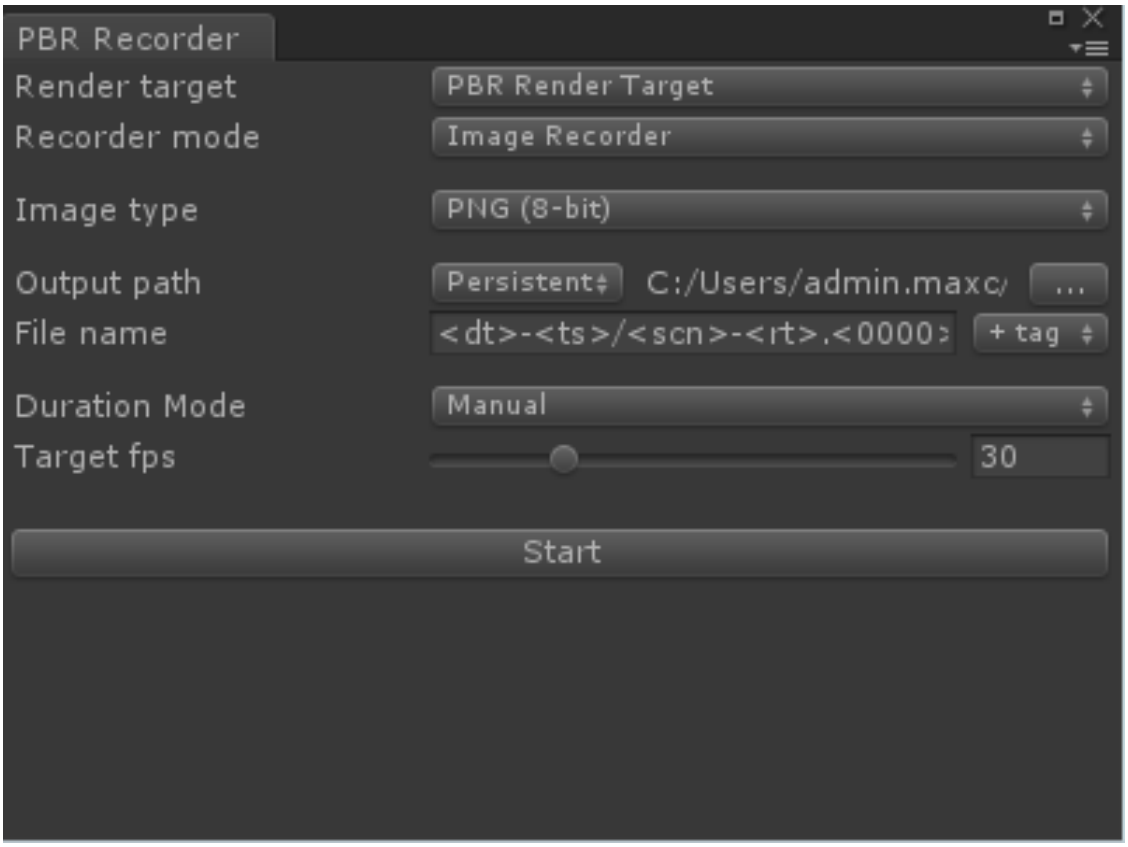


Figure 2: The Recorder window

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Recorder Parameters

- **Render Target** - Choose from the available PBR Render Targets available in the Unity scene.
- **Recorder Mode** - Choose between **Image** and **ORBX**[1]® recording modes. The Image mode records an image sequence to typical image file formats, and the ORBX mode exports the scene for rendering in ORC™ or OctaneRender Standalone.
- **Image Type** - Determines the image format for the rendered sequence.
- **Output Path** - Specifies the location on disk to save the image sequence.
- **File Name** - Specifies the file name for each image in the rendered sequence.
- **Duration Mode** - Provides various options for rendering all frames or frame sequence intervals.
- **Target FPS** - Determines the frame rate that Recorder attempts to maintain during gameplay.
- **Start** - Executes the Recorder rendering process.

[1]The ORBX file format is the best way to transfer scene files from 3D Authoring software programs that use the Octane Plug-in such as Octane for Maya, Octane for Cinema 4D, or OctaneRender Standalone. This format is more efficient than FBX when working with Octane specific data as it provides a flexible, application independent format. ORBX is a container format that includes all animation data, models, textures etc. that is needed to transfer an Octane scene from one application to another.

Timeline

Timeline in OctaneRender® is a batch render feature similar to batch renders in 3D applications such as Maya®. The purpose is to render out a frame sequence at a specific resolution, quality, and format. Each frame renders until they meet specified samples. It is not real-time, and it is best used for creating cinematic renders.

Note that OctaneRender's Timeline is NOT the same as the Unity®**Timeline** window (Figure 1). The Unity Timeline window adds keyframes to animate scene elements. It is not a rendering feature.

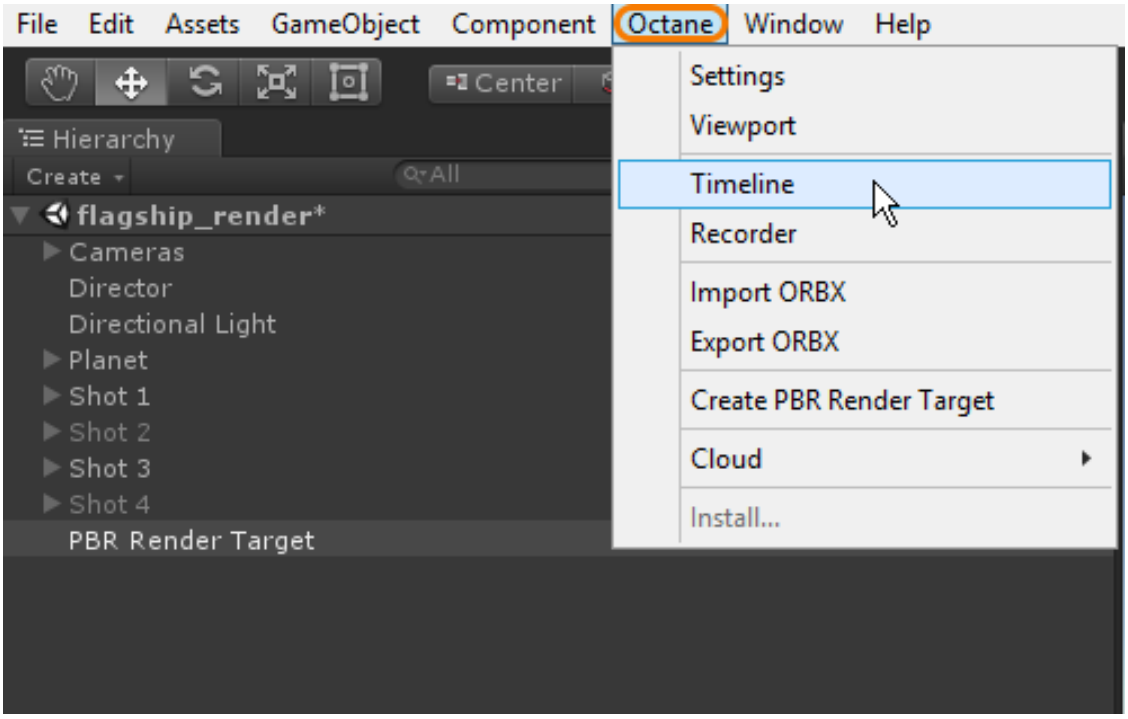


Figure 1: The OctaneRender Timeline window accessed from the Octane menu

To use the OctaneRender **PBR**[1] Timeline:

[1]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

1. Start with a scene that has animated elements. Make sure OctaneRender is installed and loaded.
2. Add a **PBR Render Target** and adjust the render settings accordingly.
3. Go to the **Octane** menu and choose **Timeline** to open up the settings window.
4. Click the **Add Job** button to choose a **Playable Director** (Unity Timeline asset) from the scene. You can add any number of Playable Directors to the render jobs, allowing for batch rendering multiple animations.

Note: You must open the OctaneRender PBR Viewport at least once before clicking the **Render** button in the PBR Timeline window, otherwise an error message displays at the bottom of the PBR Timeline window.

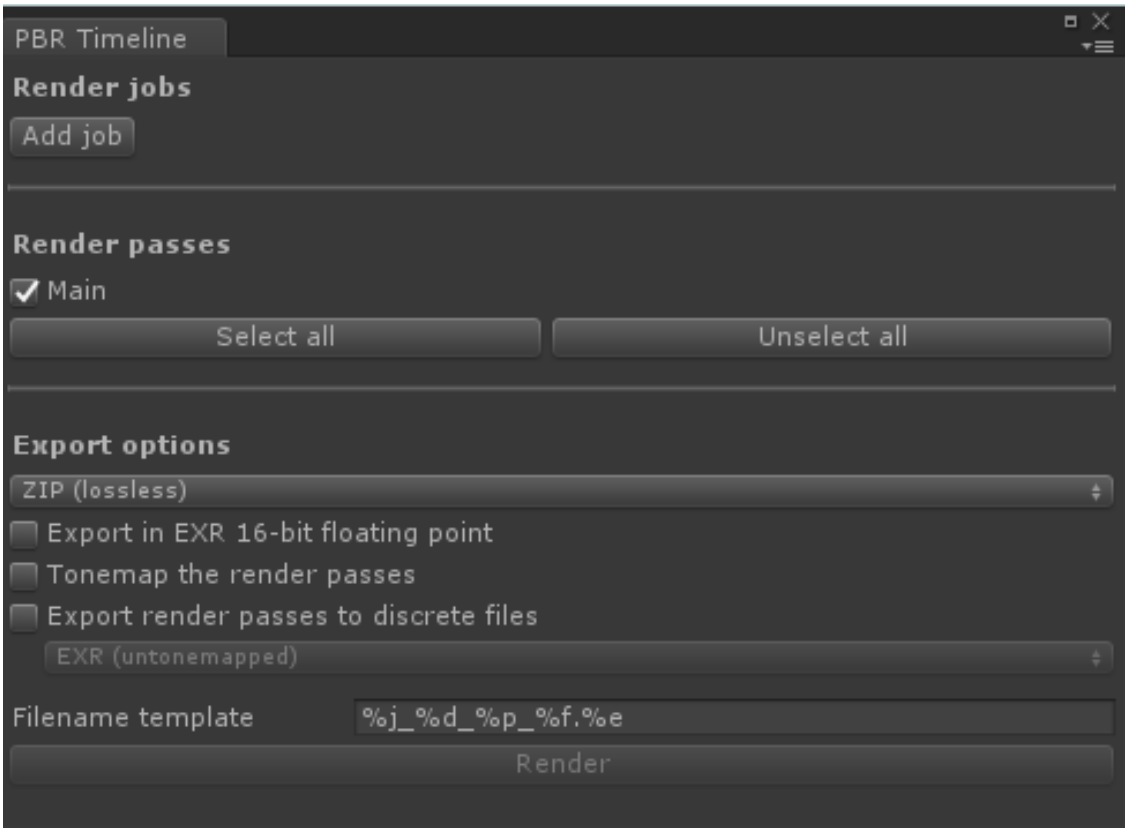


Figure 2: The OctaneRender PBR Timeline window

OctaneRender PBR Timeline Parameters

- **Add Job** - Specifies the Playable Director from the scene to add to the render queue.
- **Render Passes**[1] - If render passes are active in the PBR Render Target, you can select or deselect them here for batch rendering.
- **Export Options** - Provides various options for exporting the image sequence:
  - **Export In EXR**[2] **16-Bit Floating Point** - The EXR files export in 32-bit by default, but this parameter enables exporting in 16- bit instead.
  - **Tonemap The Render Passes** - This enables render passes to export in their native linear format, and applies tonemapping to the exported render pass files.
  - **Export Render Passes To Discrete Files** - EXR files are rendered out with all render passes contained in a single EXR file, but this parameter allows each render pass to render to separate EXR files.
- **Filename Template** - Provides naming conventions for the rendered sequence.
- **Render** - Activates the rendering process.

[1]Render passes allow a rendered frame to be further broken down beyond the capabilities of Render Layers. Render Passes vary among render engines but typically they allow an image to be separated into its fundamental visual components such as diffuse, ambient, specular, etc..
[2]Also known as OpenEXR. This image file format was developed by Industrial Light & Magic and provides a High Dynamic Range image capable of storing deep image data on a frame-by-frame basis.

Kernels

A kernel in OctaneRender® is the central part of the rendering engine that interfaces with the rendering hardware (e.g., graphics card). OctaneRender for Unity® provides four rendering kernel types. The kernels are located in the **PBR**[1] **Render Target**, under the **Kernel** parameter (Figure 1).

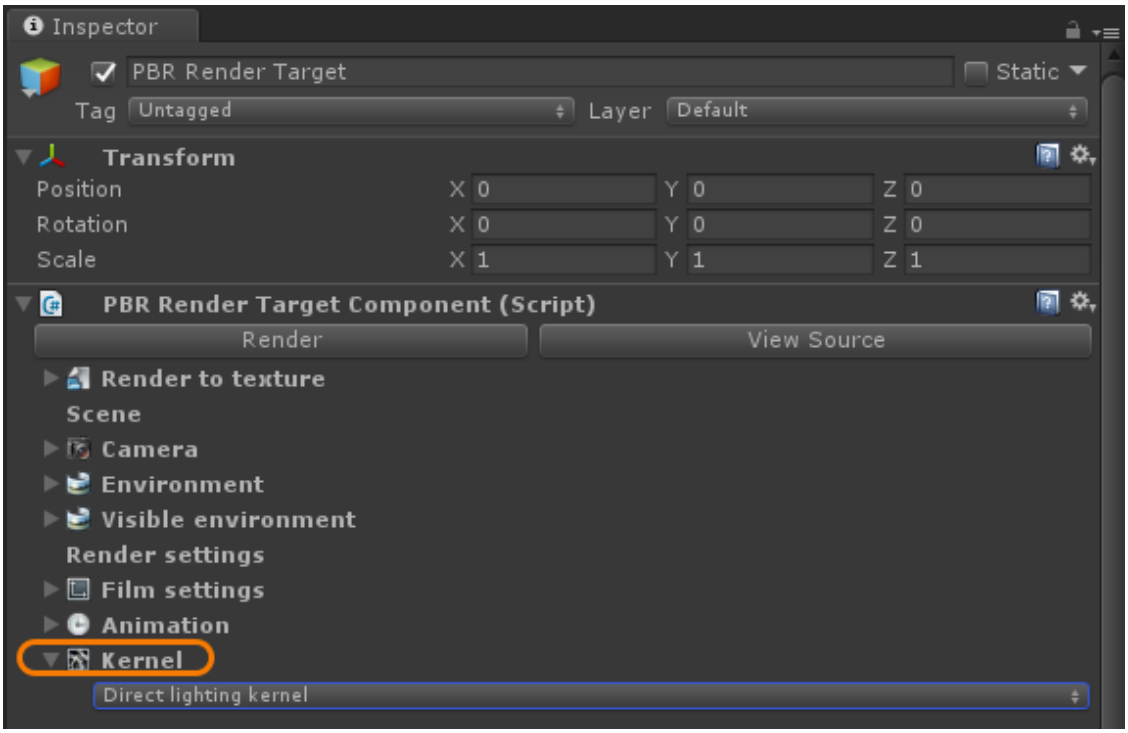
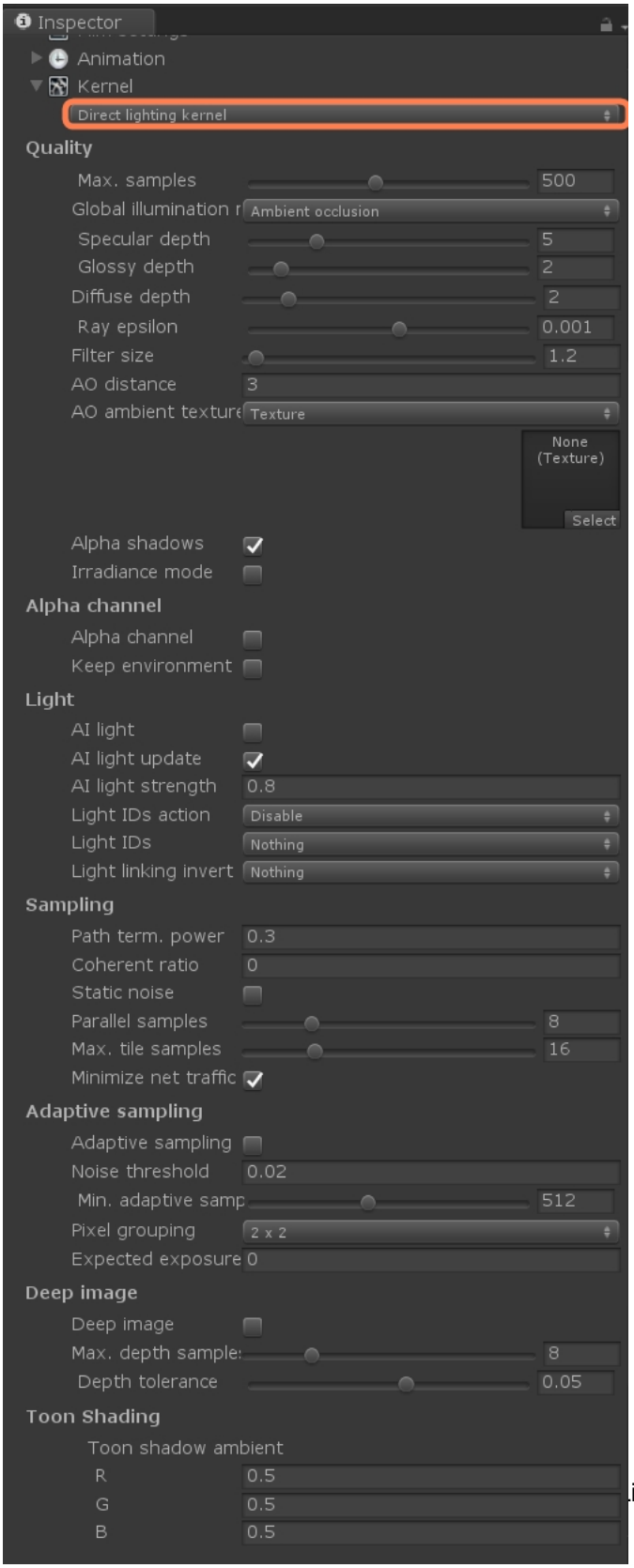


Figure 1: Accessing the OctaneRender kernels

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Direct Lighting

Direct Lighting renders previews faster. Direct Lighting is not unbiased, and shouldn't be used when you are aiming for photorealism. However, it is useful when creating quick animations or renders (Figure 1).


Inspector
Animation
Kernel
Direct lighting kernel
Quality
Max. samples 500
Global illumination r Ambient occlusion
Specular depth 5
Glossy depth 2
Diffuse depth 2
Ray epsilon 0.001
Filter size 1.2
AO distance 3
AO ambient texture Texture
None (Texture)
Select
Alpha shadows
Irradiance mode
Alpha channel
Alpha channel
Keep environment
Light
AI light
AI light update
AI light strength 0.8
Light IDs action Disable
Light IDs Nothing
Light linking invert Nothing
Sampling
Path term. power 0.3
Coherent ratio 0
Static noise
Parallel samples 8
Max. tile samples 16
Minimize net traffic
Adaptive sampling
Adaptive sampling
Noise threshold 0.02
Min. adaptive samp 512
Pixel grouping 2 x 2
Expected exposure 0
Deep image
Deep image
Max. depth sample 8
Depth tolerance 0.05
Toon Shading
Toon shadow ambient
R 0.5
G 0.5
B 0.5

ighting

Figure 1: The Direct Lighting kernel parameters

Direct Lighting Kernel Parameters

Quality

- **Max. Samples** - This sets the maximum number of samples per pixel before the rendering process stops. The higher the number of samples per pixel, the cleaner the render. For quick animations and scenes with predominantly direct lighting, a low amount of samples (500-1000) will suffice. In scenes with lots of indirect lighting and mesh lights, you may need several thousand samples to get a clean render.
- **Global Illumination Mode** - There are three different modes:
  - **None** - Only direct lighting from the sun or area lights is included. Shadowed areas receive no contribution and are black.
  - **Ambient Occlusion** - Standard ambient occlusion. This mode often provides realistic images, but offers no color bleeding.
  - **Diffuse**[1] - Indirect diffuse, with a configuration to set the number of indirect diffuse bounces. This gives a Global Illumination quality that is in between Ambient Occlusion and **Pathtracing**, but without caustics. It is much faster than Pathtracing and **PMC**.
- **Specular**[2] **Depth** - Controls the number of times a ray can refract before dying. Higher numbers mean higher render times, but more color bleeding and more details in transparent materials. Low numbers introduce artifacts or turn some refractions into pure black.
- **Glossy**[3] **Depth** - Controls the number of times a ray can reflect before dying. Higher numbers mean higher render times, and numbers below **4** can introduce artifacts or turn some reflections into pure black.
- **Diffuse Depth** - Gives the maximum number of diffuse reflections when you set Global Illumination Mode to Diffuse.
- **Ray Epsilon** - The distance to offset new rays so they don't intersect with the originating geometry. If a scene's scale is too large, precision artifacts like concentric circles can appear. In that case, increasing Ray Epsilon makes the artifacts disappear.
- **Filter Size** - Sets the pixel size for filtering the render, and improves aliasing artifacts in the render. This can also reduce noise, but if you set the filter too high, then the image becomes blurry.

---

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[3] The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.

- **AO Distance** - Ambient Occlusion distance in units.
- **AO Ambient Texture** - Sets the Ambient Occlusion environment texture for AO rays. If this isn't specified, OctaneRender® uses the environment instead.
- **Alpha Shadows** - Allows any object with transparency (**Specular** materials, materials with opacity settings and alpha channels) to cast a proper shadow instead of behaving as a solid object.
- **Irradiance Mode** - Renders the first surface as a white **Diffuse material**[1]. This mode is similar to **Clay Mode** - however, it only applies to the first bounce. It disables the **Bump** channel and makes samples that are blocked by back faces transparent.

Alpha Channel[2]

- **Alpha Channel** - Removes the background and renders it transparent (zero alpha). This is useful for compositing the render over another image without the background being present.
- **Keep Environment** - Used with the Alpha Channel setting. It allows OctaneRender to render the background with zero alpha, but the background is still visible in the final render. This gives more flexibility to compositing images.

Light

- **AI Light** - Enables AI lighting.
- **AI Light Update** - Enables dynamic AI Light Update.
- **Light IDs Action** - Selects the action taken on selected Light IDs.
- **Light IDs** - Specifies the Light ID.
- **Light Linking Invert** - Inverts the light linking behavior for the selected IDs.

> **Note:** The AI Light feature is only available in OctaneRender v4.

Sampling

- **Path Term. Power** - This parameter provides a system to tweak samples/second vs. convergence (how fast noise vanishes). Increasing this value causes the kernels to keep shorter paths and spend less time on dark areas (meaning they stay noisy longer), but it increases samples/second. Reducing this value causes kernels to trace longer paths on average, and spend more time on dark areas. In short, high values increase the render speed, but leads to higher noise in dark areas.
- **Coherent Ratio** - When enabled for **Path Tracing** or **Direct Lighting** kernels, the render becomes noise-free faster. However, it also causes flickering while rendering animations. For still images and action-heavy animations, this option saves some time.
- **Static Noise** - When enabled, the noise is static (doesn't change between frames). This is disabled by default. The noise is fully static as long as you use the same **GPU**[3] architecture for rendering. Using

---

[1] Used for dull, non-reflecting materials or mesh emitters.

[2] A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.

[3] The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

different architectures produces slightly different numerical errors, which manifest as small differences in noise patterns every time rendering restarts on the frame.

- **Parallel Samples** - Controls how many samples are calculated in parallel. If set to a small value, OctaneRender requires less memory to store the samples state, but renders are slower. If set to a high value, then OctaneRender needs more graphics memory, but renders are faster. The change in performance depends on the scene, the GPU architecture, and the number of shader processors available on the GPU.
- **Max. Tile Samples** - Controls the number of samples per pixel that OctaneRender uses for rendering before it stores the result in the film buffer. Higher numbers mean results arrive less often at the film buffer, but it reduces CPU overhead during rendering, which improves performance.
- **Minimize Net Traffic** - When enabled, OctaneRender distributes only the same tile to the net render slaves until it reaches the max samples/pixel for that tile, and then OctaneRender distributes the next tile to the slaves. This option doesn't affect work done by local GPUs.

Adaptive Sampling[1]

- **Adaptive Sampling** - Provides options to the kernel to disable sampling for pixels that reach a specified noise level.
- **Noise Threshold** - Specifies the smallest relative noise level. When a pixel's noise estimate is less than this value, then the pixel's sampling switches off. You can get good values in the range of **0.01** to **0.03**. The default is **0.02**.
- **Min. Adaptive Samples** - Specifies the minimum number of samples to calculate before adaptive sampling activates. The noise estimate of a pixel is just an estimate with a large initial error, so the higher the noise threshold, the higher you should set this parameter to avoid artifacts.
- **Pixel Grouping** - Specifies the number of pixels that are handled together. If all pixels of a group have reached the specified noise level, sampling stops for these pixels.
- **Expected Exposure** - This value should be close to the same value as the image exposure, otherwise set it to **0** to ignore these settings. The default value is 0. Adaptive sampling uses this parameter to determine the bright and dark pixels, which depends on the **Exposure** setting in the OctaneRender **Imager**. If the value is not 0, Adaptive Sampling tweaks/reduces the noise estimate of very dark areas of the image. It also increases the Min. Adaptive Samples limit for very dark areas, because very dark areas don't always find paths to light sources, resulting in over-optimistic noise estimates.

Deep Image[2]

- **Deep Image** - Enables rendering deep pixel images used for deep image compositing.
- **Maximum Depth Samples** - Sets the maximum number of depth samples per pixel. The **PBR**[3] **Render Target** uses this when you enable Deep Image rendering.

---

[1] A method of sampling that determines if areas of a rendering require more sampling than other areas instead of sampling the entire rendering equally.

[2] Renders frames with multiple depth samples in addition to typical color and opacity channels.

[3] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

- **Depth Tolerance** - OctaneRender merges the depth samples whose relative depth difference falls below this tolerance value. The PBR Render Target uses this when you enable Deep Image rendering.

Toon Shading

- **Toon Shadow Ambient** - Ambient modifier for **Toon Shadowing**.

Path Tracing

Path Tracing is best used for realistic results. The render times are higher than **Direct Lighting**, and working with small light sources and proper caustics can be difficult, but the results are photorealistic (Figure 1).

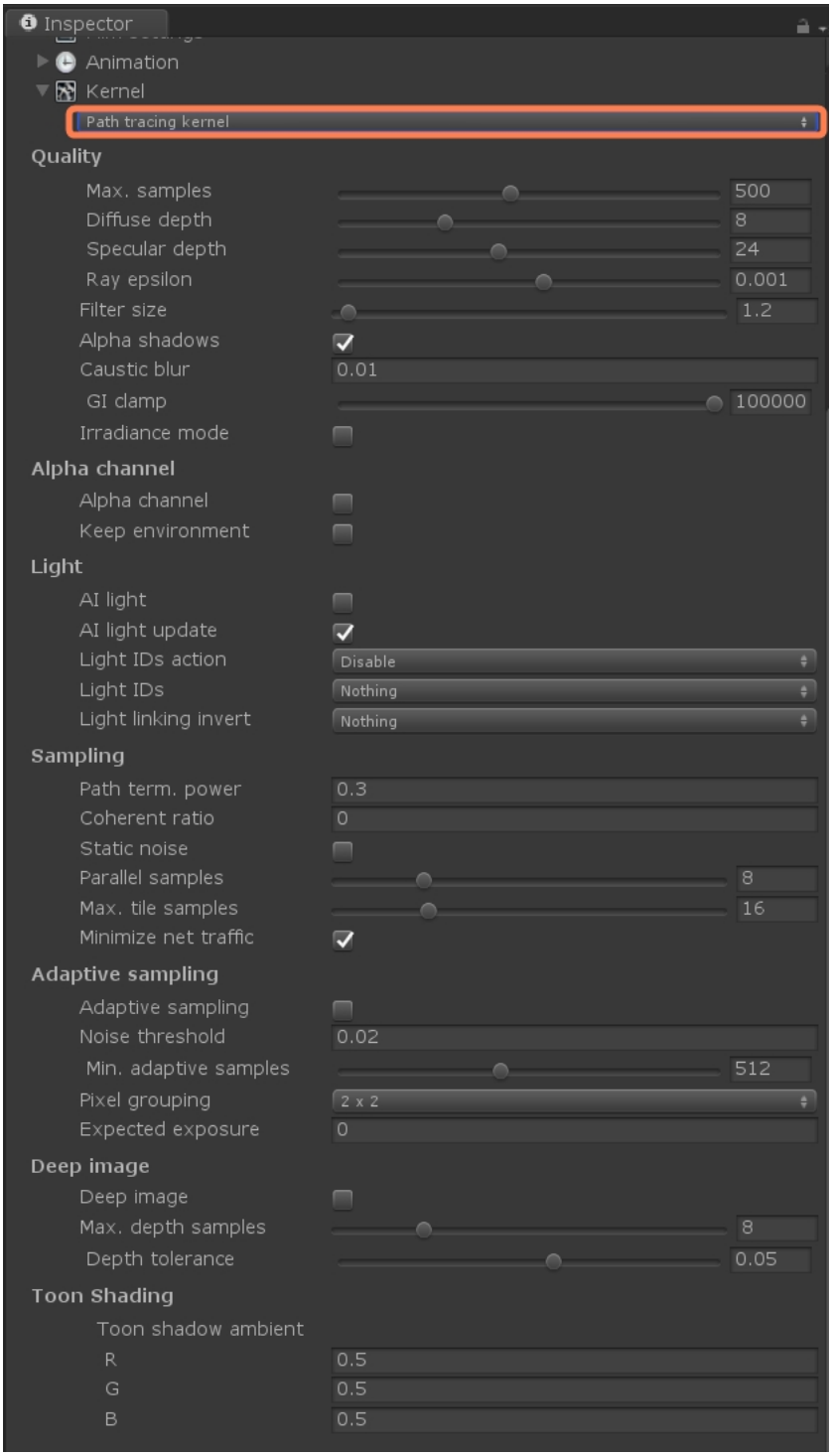

Inspector
Animation
Kernel
Path tracing kernel
Quality
Max. samples
500
Diffuse depth
8
Specular depth
24
Ray epsilon
0.001
Filter size
1.2
Alpha shadows
Caustic blur
0.01
GI clamp
100000
Irradiance mode
Alpha channel
Alpha channel
Keep environment
Light
AI light
AI light update
Light IDs action
Disable
Light IDs
Nothing
Light linking invert
Nothing
Sampling
Path term. power
0.3
Coherent ratio
0
Static noise
Parallel samples
8
Max. tile samples
16
Minimize net traffic
Adaptive sampling
Adaptive sampling
Noise threshold
0.02
Min. adaptive samples
512
Pixel grouping
2 x 2
Expected exposure
0
Deep image
Deep image
Max. depth samples
8
Depth tolerance
0.05
Toon Shading
Toon shadow ambient
R
0.5
G
0.5
B
0.5

Figure 1: The Path Tracing kernel settings

Path Tracing Kernel Parameters

Quality

- **Max. Samples** - Sets the maximum number of samples per pixel before the rendering process stops. Higher numbers of samples per pixel results in cleaner renders. For quick animations and scenes with predominantly direct lighting, a low amount of samples (500-1000) will suffice. In scenes with lots of indirect lighting and mesh lights, OctaneRender® may require several thousand samples to get a clean render.
- **Diffuse**[1] **Depth** - Gives the maximum number of diffuse reflections if you set **GI Mode** to **Diffuse**.
- **Specular**[2] **Depth** - The maximum depth to trace rays (reflections/refractions) passing through specular objects. This prevents transparent objects from causing dark outlines in scenes.
- **Ray Epsilon** - The distance to offset new rays so they don't intersect with the originating geometry. If the scale of a scene is too large, precision artifacts like concentric circles appear. In that case, increasing this parameter makes these artifacts disappear.
- **Filter Size** - Sets the pixel size for filtering the render. This improves artifact aliasing in the render, and it also reduces noise. However, if you set the filter too high, the image becomes blurry.
- **Alpha Shadows** - Allows any object with transparency (**Specular** materials, materials with opacity settings and alpha channels) to cast a shadow instead of behaving like a solid object.
- **Caustic Blur** - Approximates caustics on rough surfaces, and increases or decreases caustic noise sharpness. A **0** value provides the sharpest caustics, and increasing this value increases the blurring effect that makes caustics appear soft.
- **GI Clamp** - Controls indirect light bounces, and provides a facility to balance out global illumination and noise produced by scene caustics. Reducing he GI Clamp lowers noise contributed by strong light passing through clear or specular objects.
- **Irradiance Mode** - Renders the first surface as a white **Diffuse** material. It's similar to **Clay Mode**, but it only applies to the first bounce. It disables the **Bump** channel and turns samples transparent when back faces block them.

Alpha Channel[3]

---

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[3] A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.

- **Alpha Channel** - Removes the background and renders it transparent (zero alpha). This composites the render over another image without the background being present.
- **Keep Environment** - Allows the background to render with zero alpha, but is still visible in the final render. This works with the **Alpha Channel** setting, allowing more flexibility when compositing images.

Light

- **AI Light** - Enables AI lighting.
- **AI Light Update** - Enables dynamic AI Light Update.
- **Light IDs Action** - Selects the action taken on selected **Light IDs**.
- **Light IDs** - Specifies the Light ID.
- **Light Linking Invert** - Inverts the light linking behavior for the selected Light IDs.

Sampling

- **Path Term. Power** - Provides a system for tweaking samples/second vs. convergence (how fast noise vanishes). Increasing this value causes the kernels to keep paths shorter and spend less time on dark areas (which means they stay noisy longer), but increases samples/second. Reducing this value causes kernels to trace longer paths on average, and spend more time on dark areas. In short, high values increases the render speed, but also generates higher noise in dark areas.
- **Coherent Ratio** - When you enable Coherent Ratio for **Path Tracing** or **Direct Lighting** kernels, the render becomes noise-free faster, but the downside is flickering when rendering animations. This ratio saves time for stills and action-heavy animations.
- **Static Noise** - When enabled, the noise is static (doesn't change between frames). This is disabled by default. The noise is fully static as long as you use the same **GPU**[1] architecture for rendering. Using different architectures produces slightly different numerical errors, which manifest as small differences in noise patterns every time rendering restarts on the frame.
- **Parallel Samples** - Controls how many samples are calculated in parallel. If set to a small value, OctaneRender requires less memory to store the samples state, but renders are slower. If set to a high value, then OctaneRender needs more graphics memory, but renders are faster. The change in performance depends on the scene, the GPU architecture, and the number of shader processors available on the GPU.
- **Max. Tile Samples** - Controls the number of samples per pixel that OctaneRender uses for rendering before it stores the result in the film buffer. Higher numbers mean results arrive less often at the film buffer, but it reduces CPU overhead during rendering, which improves performance.
- **Minimize Net Traffic** - When enabled, OctaneRender distributes only the same tile to the net render slaves until it reaches the max samples/pixel for that tile, and then OctaneRender distributes the next tile to the slaves. This option doesn't affect work done by local GPUs.

---

[1]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Adaptive Sampling[1]

- **Adaptive Sampling** - Provides options to the kernel to disable sampling for pixels that reach a specified noise level.
- **Noise Threshold** - Specifies the smallest relative noise level. When a pixel's noise estimate is less than this value, then the pixel's sampling switches off. You can get good values in the range of **0.01** to **0.03**. The default is **0.02**.
- **Min. Adaptive Samples** - Specifies the minimum number of samples to calculate before adaptive sampling activates. The noise estimate of a pixel is just an estimate with a large initial error, so the higher the noise threshold, the higher you should set this parameter to avoid artifacts.
- **Pixel Grouping** - Specifies the number of pixels that are handled together. If all pixels of a group have reached the specified noise level, sampling stops for these pixels.
- **Expected Exposure** - This value should be close to the same value as the image exposure, otherwise set it to **0** to ignore these settings. The default value is 0. Adaptive sampling uses this parameter to determine the bright and dark pixels, which depends on the **Exposure** setting in the OctaneRender **Imager**. If the value is not 0, Adaptive Sampling tweaks/reduces the noise estimate of very dark areas of the image. It also increases the Min. Adaptive Samples limit for very dark areas, because very dark areas don't always find paths to light sources, resulting in over-optimistic noise estimates.

Deep Image[2]

- **Deep Image** - Enables rendering deep pixel images used for deep image compositing.
- **Maximum Depth Samples** - Sets the maximum number of depth samples per pixel. The **PBR**[3] **Render Target** uses this when you enable Deep Image rendering.
- **Depth Tolerance** - OctaneRender merges the depth samples whose relative depth difference falls below this tolerance value. The PBR Render Target uses this when you enable Deep Image rendering.

Toon Shading

- **Toon Shadow Ambient** - Ambient modifier for **Toon Shadowing**.

---

[1] A method of sampling that determines if areas of a rendering require more sampling than other areas instead of sampling the entire rendering equally.

[2] Renders frames with multiple depth samples in addition to typical color and opacity channels.

[3] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

PMC

PMC is a custom mutating unbiased kernel written for GPUs. It helps resolve complex caustics and lighting (Figure 1).


Inspector
PBR Render Target
Static
Tag Untagged
Layer Default
Transform
PBR Render Target Component (Script)
Render
View Source
Render to texture
Scene
Camera
Environment
Visible environment
Render settings
Film settings
Animation
Kernel
PMC kernel
Quality
Max. samples 1000
Diffuse depth 8
Specular depth 24
Ray epsilon 0.001
Filter size 1.2
Alpha shadows
Caustic blur 0.01
GI clamp 1000000
Alpha channel
Alpha channel
Keep environment
Sampling
Path term. power 0.3
Exploration strength 0.7
Direct light importance 0.1
Max. rejects 500
Parallel samples 4
Work chunk size 8
Render layer
Render passes
Tonemap settings
Imager
Post processing
Add Component

Figure 1: The PMC Kernel settings

PMC Kernel Parameters

Quality

- **Max. Samples** - Sets the maximum number of samples per pixel before the rendering process stops. Higher numbers of samples per pixel results in cleaner renders. For quick animations and scenes with predominantly direct lighting, a low amount of samples (500-1000) will suffice. In scenes with lots of indirect lighting and mesh lights, OctaneRender® may require several thousand samples to get a clean render.
- **Diffuse**[1] **Depth** - Gives the maximum number of diffuse reflections if you set **GI Mode** to **Diffuse**.
- **Specular**[2] **Depth** - The maximum depth to trace rays (reflections/refractions) passing through specular objects. This prevents transparent objects from causing dark outlines in scenes.
- **Ray Epsilon** - The distance to offset new rays so they don't intersect with the originating geometry. If the scale of a scene is too large, precision artifacts like concentric circles appear. In that case, increasing this parameter makes these artifacts disappear.
- **Filter Size** - Sets the pixel size for filtering the render. This improves artifact aliasing in the render, and it also reduces noise. However, if you set the filter too high, the image becomes blurry.
- **Alpha Shadows** - Allows any object with transparency (**Specular** materials, materials with opacity settings and alpha channels) to cast a shadow instead of behaving like a solid object.
- **Caustic Blur** - Approximates caustics on rough surfaces, and increases or decreases caustic noise sharpness. A **0** value provides the sharpest caustics, and increasing this value increases the blurring effect that makes caustics appear soft.
- **GI Clamp** - Controls indirect light bounces, and provides a facility to balance out global illumination and noise produced by scene caustics. Reducing he GI Clamp lowers noise contributed by strong light passing through clear or specular objects.
- **Irradiance Mode** - Renders the first surface as a white **Diffuse** material. It's similar to **Clay Mode**, but it only applies to the first bounce. It disables the **Bump** channel and turns samples transparent when back faces block them.

Alpha Channel[3]

---

[1] Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.
[2] Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.
[3] A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.

- **Alpha Channel** - Removes the background and renders it transparent (zero alpha). This composites the render over another image without the background being present.
- **Keep Environment** - Allows the background to render with zero alpha, but is still visible in the final render. This works with the **Alpha Channel** setting, allowing more flexibility when compositing images.

Light

- **AI Light** - Enables AI lighting.
- **AI Light Update** - Enables dynamic AI Light Update.
- **Light IDs Action** - Selects the action taken on selected **Light IDs**.
- **Light IDs** - Specifies the Light ID.
- **Light Linking Invert** - Inverts the light linking behavior for the selected Light IDs.

Sampling

- **Path Term. Power** - Provides a system for tweaking samples/second vs. convergence (how fast noise vanishes). Increasing this value causes the kernels to keep paths shorter and spend less time on dark areas (which means they stay noisy longer), but increases samples/second. Reducing this value causes kernels to trace longer paths on average, and spend more time on dark areas. In short, high values increases the render speed, but also generates higher noise in dark areas.
- **Exploration Strength** - Specifies how long the kernel investigates good paths before it tries to find a new path. Low values create a noisy image, while larger values create a splotchy image.
- **Direct Light Importance** - Makes the kernel focus more on paths with indirect light. For example, imagine sunlight through a window that creates a bright spot on the floor. If Direct Light Importance is **1**, then the kernel samples this area a lot, and it becomes clean fast. If you reduce this parameter, the kernel samples the area less and focuses on areas that are hard to render.
- **Max. Rejects** - Controls the render bias. Lower values produce results with more bias, but shorter render times.
- **Parallel Samples** - Controls how many samples are calculated in parallel. If set to a small value, OctaneRender requires less memory to store the samples state, but renders are slower. If set to a high value, then OctaneRender needs more graphics memory, but renders are faster. The change in performance depends on the scene, the **GPU**[1] architecture, and the number of shader processors available on the GPU.
- **Work Chunk Size** - The number of work blocks (of 512k samples each) done per kernel run. Higher values increase the memory requirements on the system, but it doesn't affect memory usage. Render speed also increases.

Toon Shading

- **Toon Shadow Ambient** - Ambient modifier for **Toon Shadowing**.

---

[1]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Info Channels

The **Info Channel** kernel creates false-color images of the scene according to types that contain various information about the scene (Figure 1).

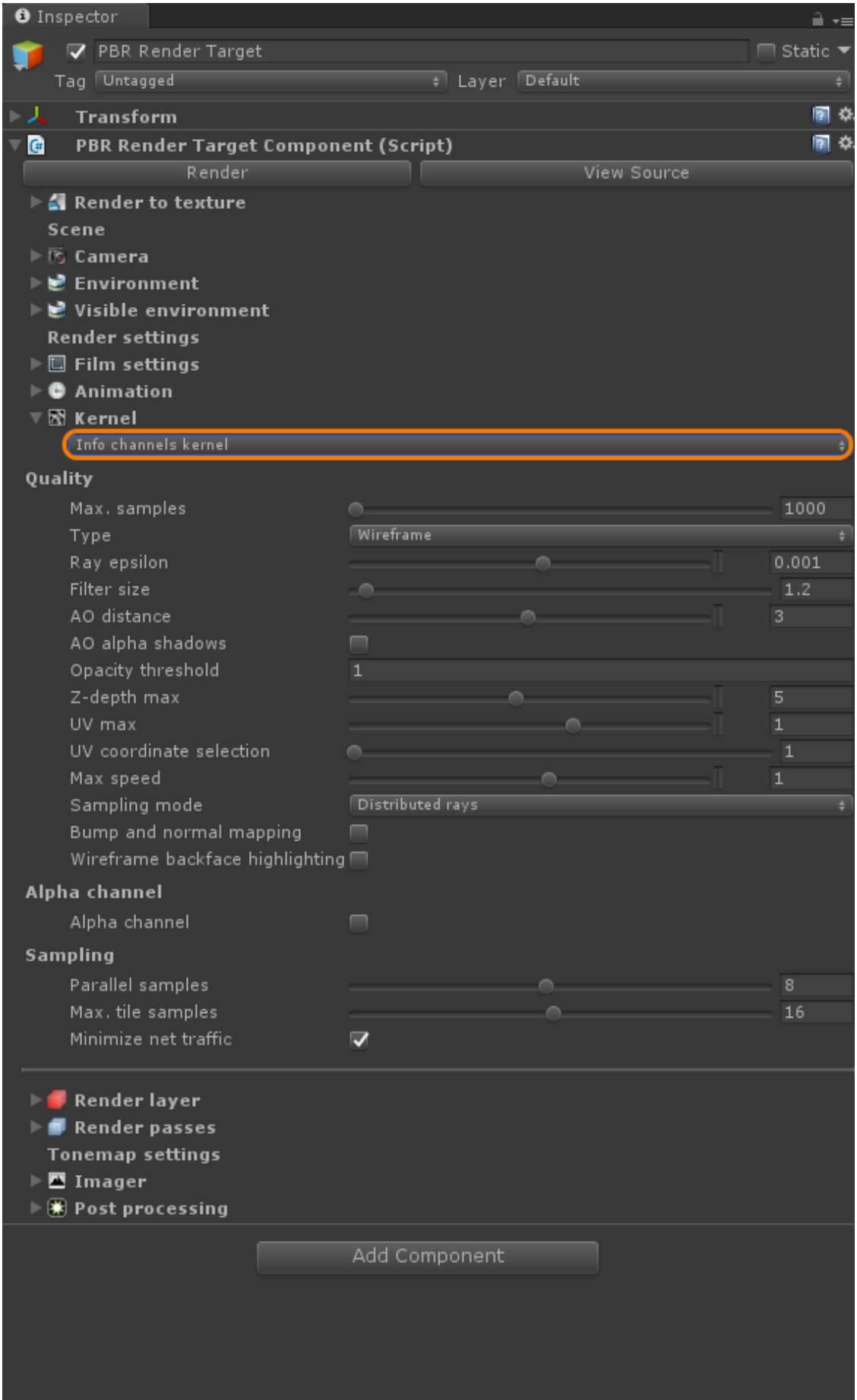

Inspector
PBR Render Target
Static
Tag
Untagged
Layer
Default
Transform
PBR Render Target Component (Script)
Render
View Source
Render to texture
Scene
Camera
Environment
Visible environment
Render settings
Film settings
Animation
Kernel
Info channels kernel
Quality
Max. samples
1000
Type
Wireframe
Ray epsilon
0.001
Filter size
1.2
AO distance
3
AO alpha shadows
Opacity threshold
1
Z-depth max
5
UV max
1
UV coordinate selection
1
Max speed
1
Sampling mode
Distributed rays
Bump and normal mapping
Wireframe backface highlighting
Alpha channel
Alpha channel
Sampling
Parallel samples
8
Max. tile samples
16
Minimize net traffic
Render layer
Render passes
Tonemap settings
Imager
Post processing
Add Component

Figure 1: The Info Channel kernel settings

The following Info Channel **Types** are available:

- **Geometric Normal** - The vectors perpendicular to the triangle faces of the mesh.
- Interpolated Vertex Normals - Similar to **Shading Normal**, but also calculates the shading based on vertex normals provided by actual data in the mesh. This makes flawed normals resulting from stuffed vertices visible in the shading.
- **Shading Normal** - The interpolated normals used for shading by calculations based on the face normals, and discreetly fixes distortion to keep a smooth appearance. This does not take into account the **Bump** map of the object. For objects without smoothing, this is identical to the Geometric normals.
- **Z-Depth**[1] - The distance between the intersection point and the camera, measured parallel to the view vector.
- **Position** - The first intersection point's position.
- **Material**[2] **ID** - Octane represents each **Material** pin as a separate color.
- **Wireframe** - Shows the mesh represented by edges, vertices, and surfaces.

Info Channel Kernel Parameters

Quality

- **Max. Samples** - Sets the maximum number of samples per pixel before the rendering process stops. Higher numbers of samples per pixel results in cleaner renders. For quick animations and scenes with predominantly direct lighting, a low amount of samples (500-1000) will suffice. In scenes with lots of indirect lighting and mesh lights, OctaneRender® may require several thousand samples to get a clean render.
- **Type** - Contains the various Info Channel types.
- **Ray Epsilon** - The distance to offset new rays so they don't intersect with the originating geometry. If the scale of a scene is too large, precision artifacts like concentric circles appear. In that case, increasing this parameter makes these artifacts disappear.
- **Filter Size** - Sets the pixel size for filtering the render. This improves artifact aliasing in the render, and it also reduces noise. However, if you set the filter too high, the image becomes blurry.
- **AO Distance** - Ambient Occlusion distance in units.
- **AO Alpha Shadows** - The Ambient Occlusion calculation takes opacity into account, and render passes use this to determine if shadows cast by Ambient Occlusion render as transparent (zero alpha). This helps composite the render over another image without the AO shadows present.

---

[1] A measure of object distances from the camera typically represented as a grayscale image.

[2] The representation of the surface or volume properties of an object.

- **Opacity Threshold** - Geometry with opacity greater than or equal to this value is treated as totally opaque.
- **Z-Depth Max** - Gives the maximum z-depth that can be shown.
- **UV Max** - Gives the maximum value that can be shown to texture coordinates.
- **UV Coordinate Selection** - Determines the set of UV coordinates to use.
- **Max Speed** - Speed mapped to the maximum intensity in the **Motion Vector** channel. A value of **1** means a maximum movement of 1 screen width in the shutter interval.
- **Sampling Mode** - Includes options for **Distributed Ray Tracing** and **Pixel Filtering** modes. Pixel filtering is like anti-aliasing - OctaneRender applies this in the render passes extracted for third-party compositing, which may require anti-aliased mattes. Without filtering, the jagged edges of objects in the scene are still evident.
  - **Distributed Rays** - Allows **Motion Blur**[1] and **Depth Of Field**[2] samples.
  - **Non-Distributed With Pixel Filtering** - Disables Motion Blur and Depth Of Field, but leaves Pixel Filtering enabled.
  - **Non-Distributed Without Pixel Filtering** - Disables Motion Blur, Depth Of Field, and Pixel Filtering for all render passes, except for **Render Layer Mask** and **Ambient Occlusion**.
- **Bump and Normal Mapping** - Option to calculate or not calculate the **Bump** and **Normal** maps.
- **Wireframe Backface** - Highlights backface in the **Wireframe** channel.

Alpha Channel[3]

- **Alpha Channel** - Removes the background and renders it transparent (zero alpha). This is useful for compositing the render over another image without the background being present.

Sampling

- **Parallel Samples** - Controls how many samples are calculated in parallel. If set to a small value, OctaneRender requires less memory to store the samples state, but renders are slower. If set to a high value, then OctaneRender needs more graphics memory, but renders are faster. The change in performance depends on the scene, the **GPU**[4] architecture, and the number of shader processors available on the GPU.

---

[1] An optical phenomenon that occurs when a camera's shutter opens and closes too slowly to capture movement without recording a blurring of the subject.
[2] The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)
[3] A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.
[4] The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

- **Max. Tile Samples** - Controls the number of samples per pixel that OctaneRender renders before it stores the result in the film buffer. Higher numbers mean results arrive less often at the film buffer, but it reduces CPU overhead during rendering, which improves performance.
- **Minimize Net Traffic** - When enabled, OctaneRender distributes only the same tile to the net render slaves until it reaches the max samples/pixel for that tile, and then OctaneRender distributes the next tile to the slaves. This option doesn't affect work done by local GPUs.

Render to Texture

Render to Texture provides functionality that immediately saves the rendering in the **PBR**[1] **Viewport** to disk (Figure 1).

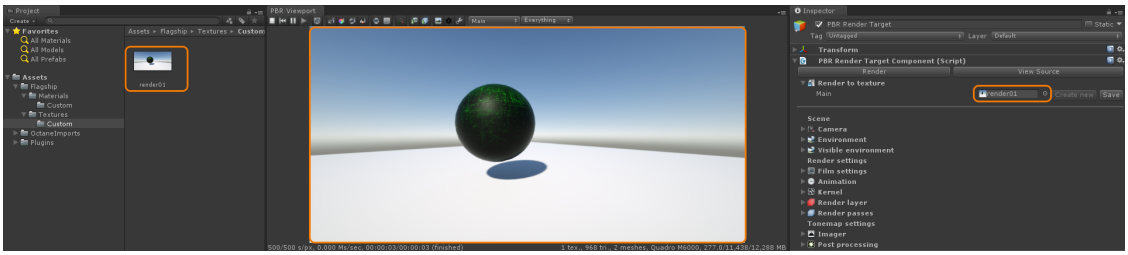


Figure 1: Using Render to Texture as a process to save PBR Viewport renderings directly to disk

You can access Render to Texture from the **PBR Render Target**, in the **Inspector** window (Figure 2).

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

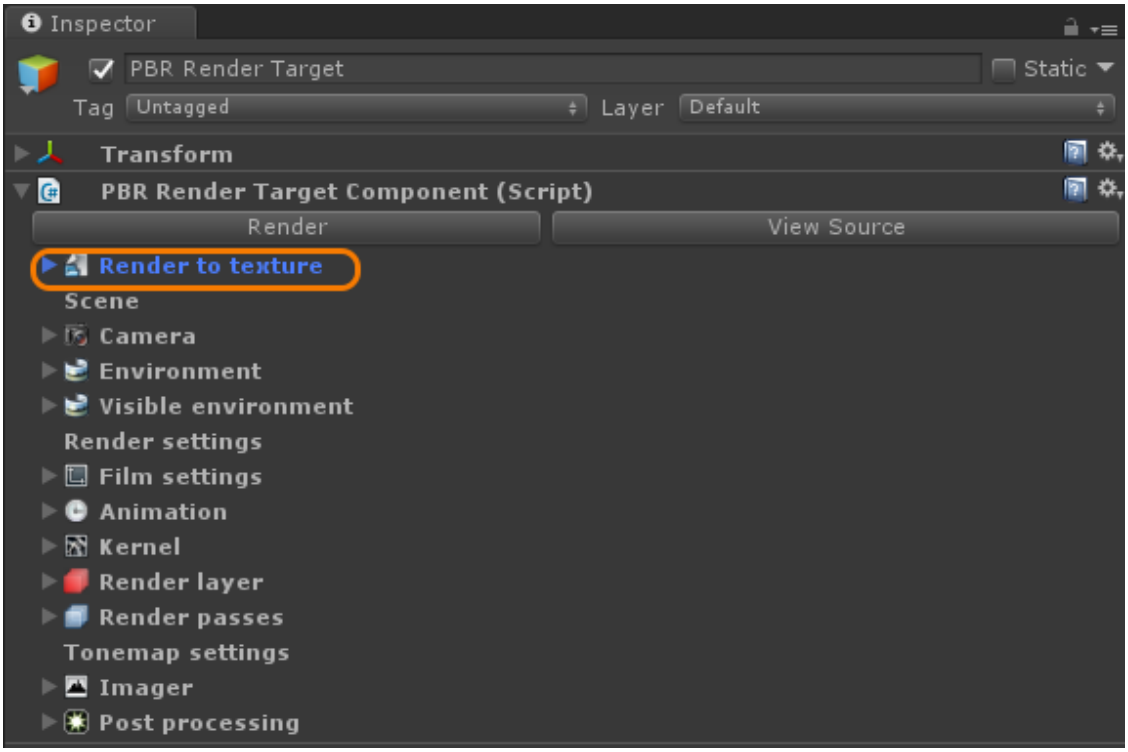


Figure 2: Accessing Render to Texture from the PBR Render Target's Inspector window

To create a new Render to Texture file, click the **Create New** button (Figure 3).

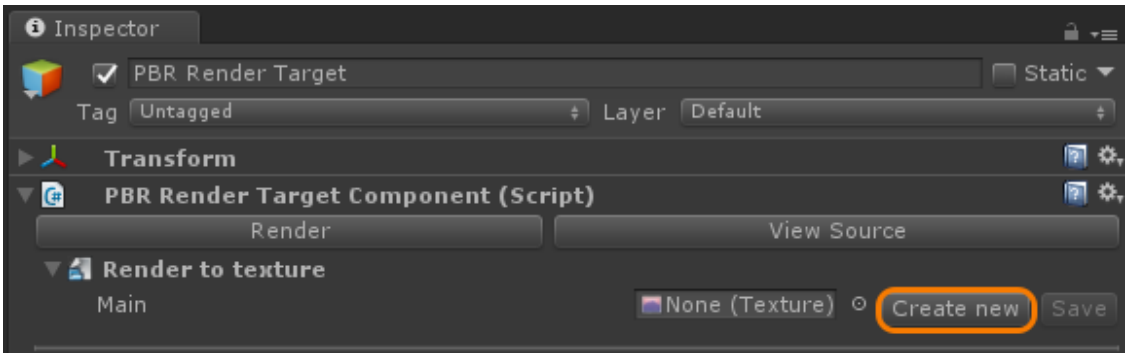


Figure 3: Creating a new Render to Texture file

After you create a new Render to Texture file, a new dynamic link between the rendering in the **PBR Viewport** and the newly saved texture asset in the **Project** window opens. As the PBR Viewport rendering updates, the texture asset in the Project window updates to reflect the PBR Viewport rendering (Figure 4).

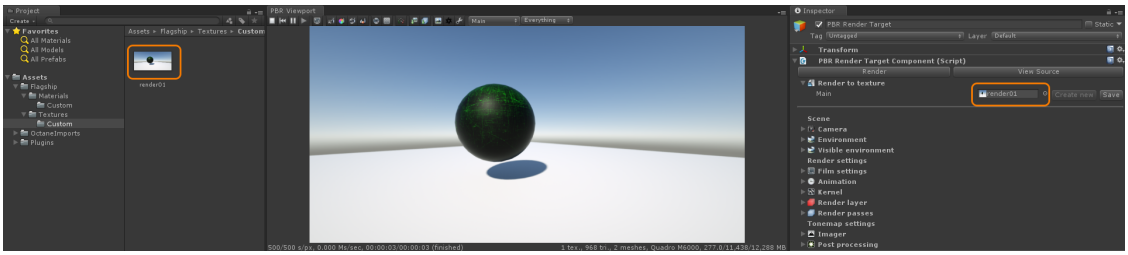


Figure 4: The file specified in Render to Texture is directly linked to the PBR Viewport rendering and the associated asset in the Project window

You can save the rendering in the PBR Viewport by clicking on the **Save** button (Figure 5). Once the Render to Texture file is saved, you can create a new file.

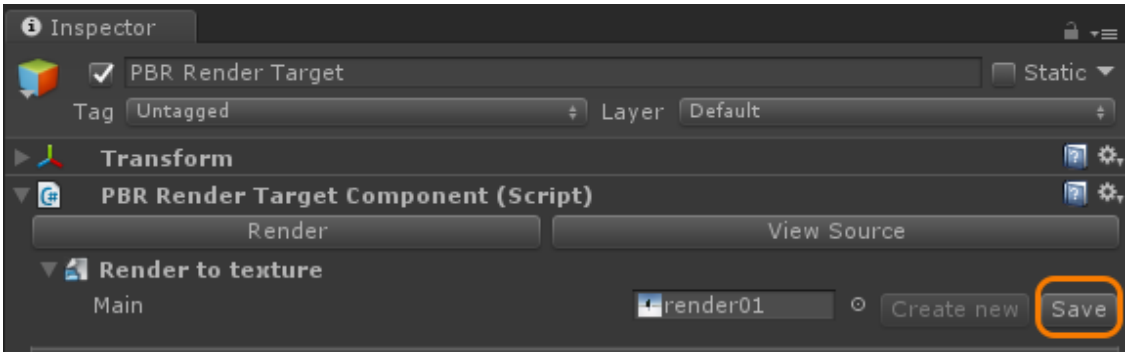


Figure 5: Saving a Render to Texture file

The Render to Texture feature is used for baking object textures, and you can use it with the **Baking Camera** (Figure 6).

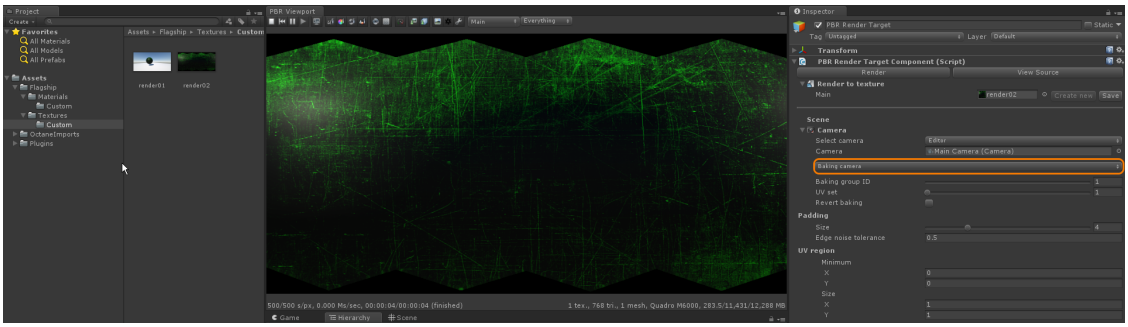


Figure 6: Using the Baking Camera with the Render to Texture feature

Lighting and Texture Baking

In OctaneRender®, texture baking is implemented via the **Baking** camera (Figure 2). In contrast to the **Thin Lens** and **Panoramic** cameras, the Baking camera has one position and direction per sample.

For each sample, the Baking camera calculates the geometry position and normal. Then, using the same direction as the normal, the camera generates a ray that points towards it from the configured kernel's **Ray Epsilon** distance. Once calculated, OctaneRender traces the ray the same way as it would trace with other types of cameras.

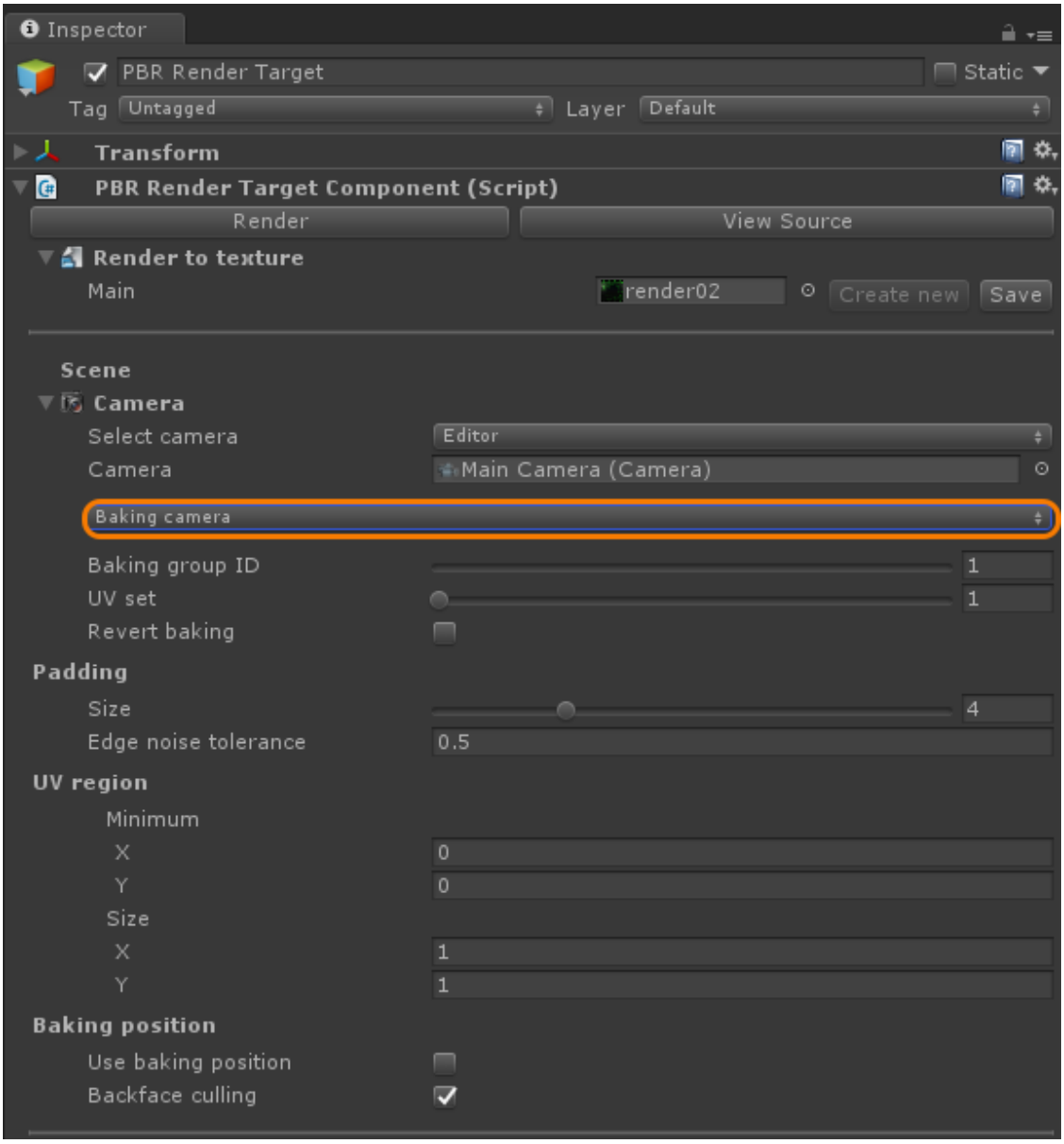


Figure 1: Selecting the Baking Camera in the PBR[1] Render Target's Inspector window

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Baking Camera Parameters

- **Baking Group ID** - Specifies the Group ID to bake. By default, all objects belong to group **1**. You can assign new baking groups to scene assets with this parameter from the scene asset's **PBR Instance Properties (Script)**.
- **UV Set** - Determines the UV coordinates for baking.
- **Revert Baking** - Flips the camera rays. You can use the geometry as a lens, and use the mesh as a camera to render the rest of the scene.

Padding

- **Size** - Number of pixels added to the UV map edges. Due to interpolation when mapping a texture to a mesh, a black edge can appear because the texture is black (no data) beyond the UV mesh. To avoid this, add padding around the baked data's edges.
- **Edge Noise Tolerance** - Removes hot pixels near the edge UV geometry. Values close to **1** don't remove any hot pixels, while values close to **0** try to remove all hot pixels.

UV Region

- **Minimum X And Y** - UV space coordinates for the bounding region's baking origin.
- **Size X And Y** - Size of the bounding region in UV space for baking.

Baking Position

- **Use Baking Position** - Allows baking position-dependent artifacts.
- **Backface Culling** - Bakes the back geometry faces.

Render Layers

Render layers allows you to separate scene geometry into parts, where one part is visible, and the other parts capture the side effects of the visible geometry on it.

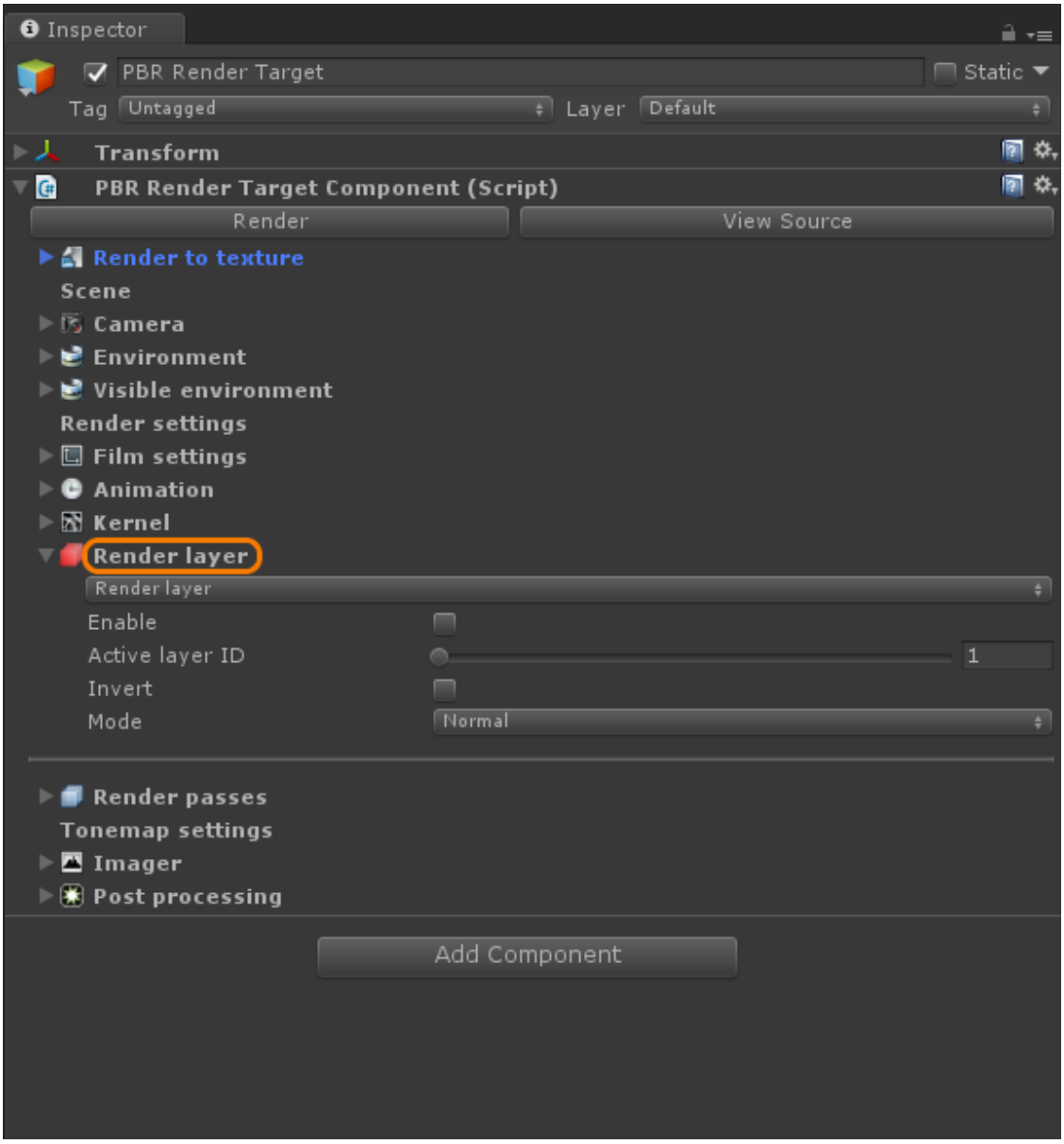

Inspector
PBR Render Target
Static
Tag
Untagged
Layer
Default
Transform
PBR Render Target Component (Script)
Render
View Source
Render to texture
Scene
Camera
Environment
Visible environment
Render settings
Film settings
Animation
Kernel
Render layer
Render layer
Enable
Active layer ID
1
Invert
Mode
Normal
Render passes
Tonemap settings
Imager
Post processing
Add Component

Figure 1: Render Layers[1] are accessible from a PBR[2] Render Target

OctaneRender® assigns objects in the scene with a **Render Layer ID** (Figure 2), and then specifies the **Active Layer** in the **Render Layer** rollout of the **PBR Render Target**. OctaneRender can then render objects into separate images, and then applies some normal render passes.

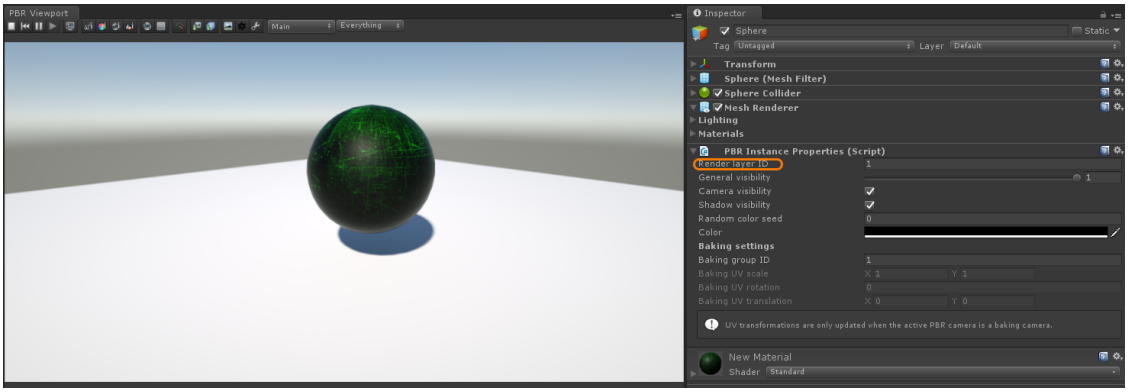


Figure 2: Setting the Render Layer ID for individual scene assets

There are four modes to render layers:

- **Normal** - The beauty passes contain only the active layer, and the render layer passes (shadows, reflections, etc.) record the side effects of the active render layer for those samples/pixels that are not obstructed by the active render layer. Beauty passes are transparent for those pixels covered by the objects on the inactive layers, even if an object is on the active layer behind the foreground.
- **Hide Inactive Layers** - All geometry that is not on an active layer is invisible. No side effects are recorded in the render layer passes — i.e., the render layer passes will be empty.
- **Hide From Camera** - Similar to Hide Inactive Layers, where all geometry not on an active layer is invisible, but side effects (shadows, reflections, etc.) are recorded in the render layer passes.
- **Only Side Effects** - Similar to Normal, except the active layer is invisible to the camera, i.e., the beauty passes are empty. The render layer passes (shadows, reflections, etc.) still record the side effects of the active render layer.

---

[1]Render layers allow users to separate their scene geometry into parts, where one part is meant to be visible and the rest of the other parts "capture" the side effects of the visible geometry. The layers allow different objects to be rendered into separate images where, in turn, some normal render passes may be applied. The Render layers are meant for compositing and not to hide parts of the scene.
[2]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Render Passes

OctaneRender® for Unity® provides **Render Passes**[1] that enable the rendering process to segregate different aspects of the scene, rendering each aspect across multiple images. This is useful in fine-tuning projects, compositing, and creating detailed and photorealistic images. You can access the Render Passes rollout from the **PBR**[2] **Render Target** (Figure 1).

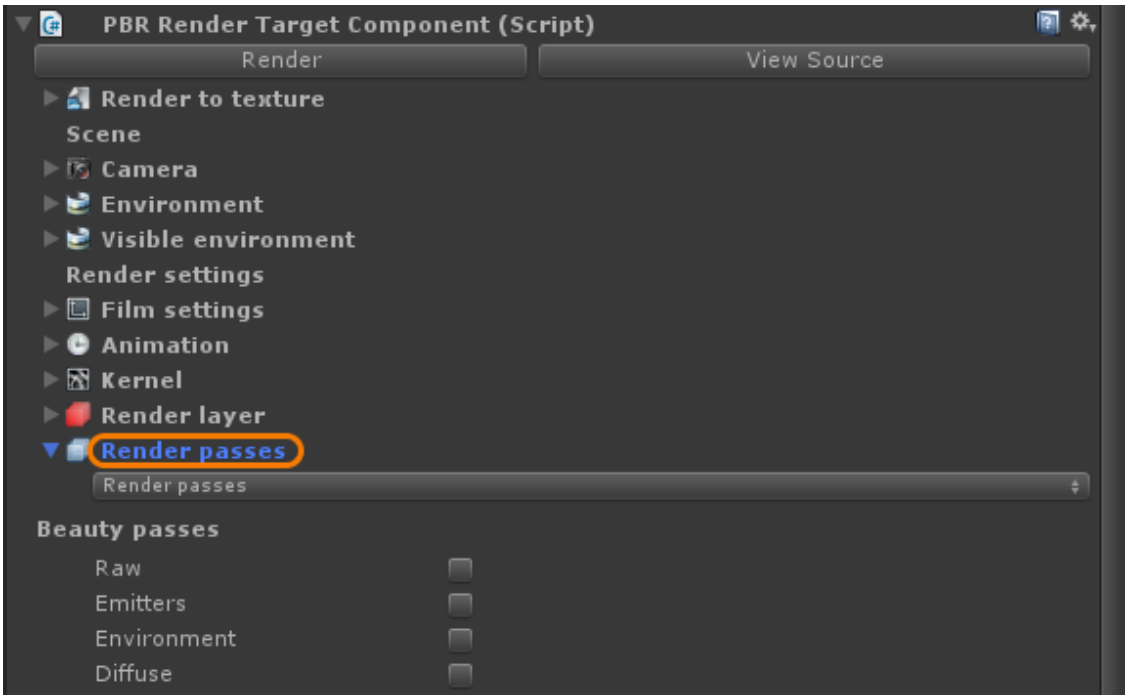


Figure 1: Accessing Render Passes in a PBR Render Target

The passes are classified as:

- **Beauty Passes**- Provides a rendered view of the different aspects lighting the scene, such as shadows, highlights, reflections, illumination, and color. OctaneRender will render beauty passes together

---

[1]Render passes allow a rendered frame to be further broken down beyond the capabilities of Render Layers. Render Passes vary among render engines but typically they allow an image to be separated into its fundamental visual components such as diffuse, ambient, specular, etc..
[2]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

with the main beauty pass (the normal rendering), i.e. each one of these requires its own film buffer to store in addition to the main film buffer on the device.

- **Denoiser Passes** - NEEDS DESCRIPTION
- **Post-Processing Passes**- Provides a view of the post-processing effects on the scene. If the **Post-Processing** node is enabled and a post-processing effect is applied to the scene, the Post-Processing render pass negates these effects and separates itself from the main pass.
- **Render Layer Passes**- These passes allow further separation of scene assets placed on various render layers. Please see the **Render Layers**[1] section for more detail.
- **Lighting Passes**- A light pass isolates individual light sources. Each light pass behaves as if all the other lights in the scene are switched off. OctaneRender adds the individual light passes together to recreate the original render in post, or to further adjust the individual contributions of each light during post.
  To use light passes in OctaneRender, you must identify each light emitter and map them to the desired light pass. You do this by assigning the **Light Pass ID** in each **Emission** node in the scene.

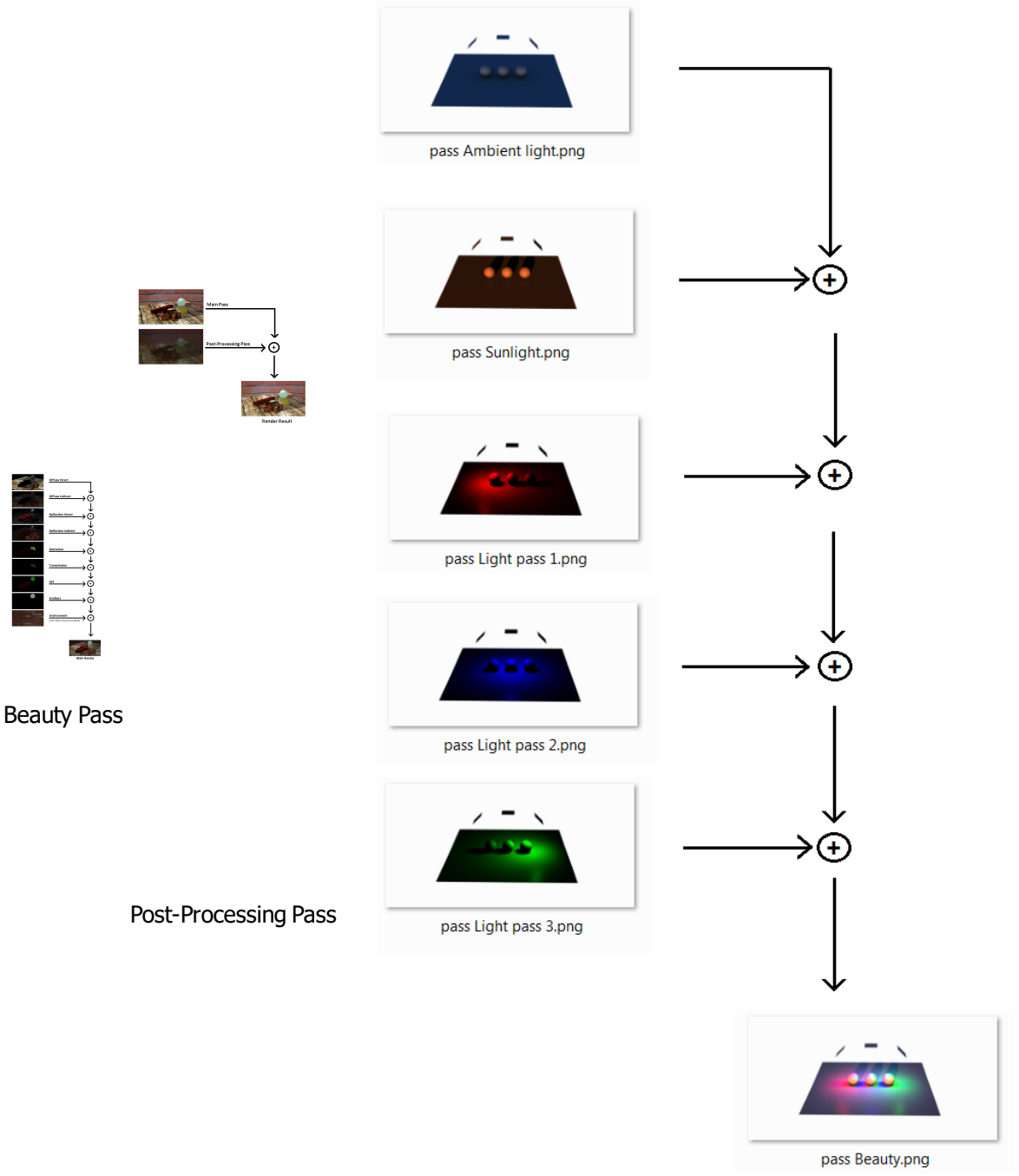
  Note: This can only be done in the **OctaneVR® Nodegraph Editor** window.

- **Info Passes**- Provides views of the effects of the scene's normals, UVs, and geometric data. OctaneRender will render the info passes one-by-one using only one additional film buffer - either at the end, or when needed. This helps save **GPU**[2] memory, and the passes are fast to calculate. You can adjust the **Max Samples** parameter without affecting the maximum samples set toward the final rendered image.
- **Material**[3] **Passes** - Include all characteristics associated with scene materials.

The classifications group similar scene components together to use in the compositing process (Table 1).

---

[1]Render layers allow users to separate their scene geometry into parts, where one part is meant to be visible and the rest of the other parts "capture" the side effects of the visible geometry. The layers allow different objects to be rendered into separate images where, in turn, some normal render passes may be applied. The Render layers are meant for compositing and not to hide parts of the scene.
[2]The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.
[3]The representation of the surface or volume properties of an object.


pass Ambient light.png
pass Sunlight.png
pass Light pass 1.png
pass Light pass 2.png
pass Light pass 3.png
pass Beauty.png
Beauty Pass
Post-Processing Pass

Lighting Pass

Table 1: Classification of Render Passes

Imager

The Camera **Imager** settings provide useful parameters for post-rendering adjustments. These parameters are in the **Imager** roll out of a **PBR**[1] **Render Target** (Figure 1).

[1]A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

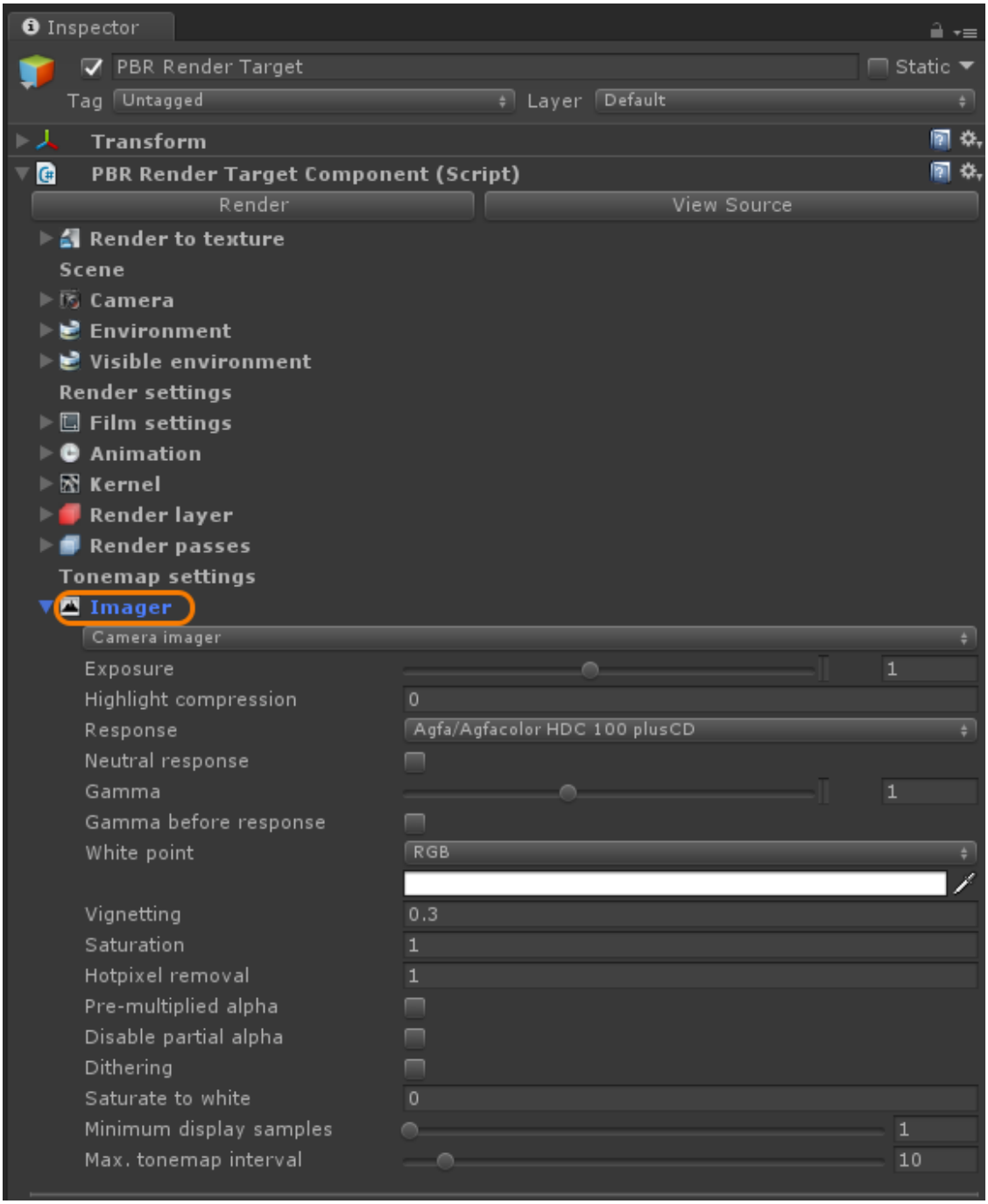


Figure 1: The Camera Imager parameters

Camera Imager Parameters

- **Exposure** - Controls the exposure of the scene. Smaller values darken a scene, while higher values brighten the scene. Exposure has no effect on any render layer passes.
- **Highlight Compression** - Reduces burned-out highlights by compressing them and reducing their contrast.
- **Order** - This defines the order in which the **Response Curve**, the **Gamma**[1], and the **Custom LUT** is applied on the scene. Typically, 3D LUTs are defined for sRGB input values (i.e. you usually want to apply the custom LUT last), but there might also be 3D look-up tables for linear input data, in which case you might want to apply the custom LUT first.
- **Response Curve** - Selects the measured camera response curves. OctaneRender® also has response curves that produce a neutral rendering on a normal display. The **sRGB**, **Gamma 2.2**, and **Gamma 1.8** curves are applicable for most displays that either use sRGB, or simply apply a gamma of 2.2 or 1.8.

  > **Note:** The most common response curve is sRGB, so this is the default setting in the Camera Imager node. Since this option did not exist in earlier versions of OctaneRender, any scene with an sRGB response curve in the Imager settings falls back to **Linear/Off** in older versions.

  For examples of all the camera responses, see the Camera Response Curve topic.
- **Neutral Response** - If enabled, the camera Response Curve won't tint the render result. In the following example (Figure 2), the left material ball is rendered with no Response Curve and Gamma set to **2.2**. The center ball uses the **Agfacolor HDC 200** curve and a Gamma of **1**. The right ball shows the same curve as the center ball, but with Neutral Response enabled.

[1] The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

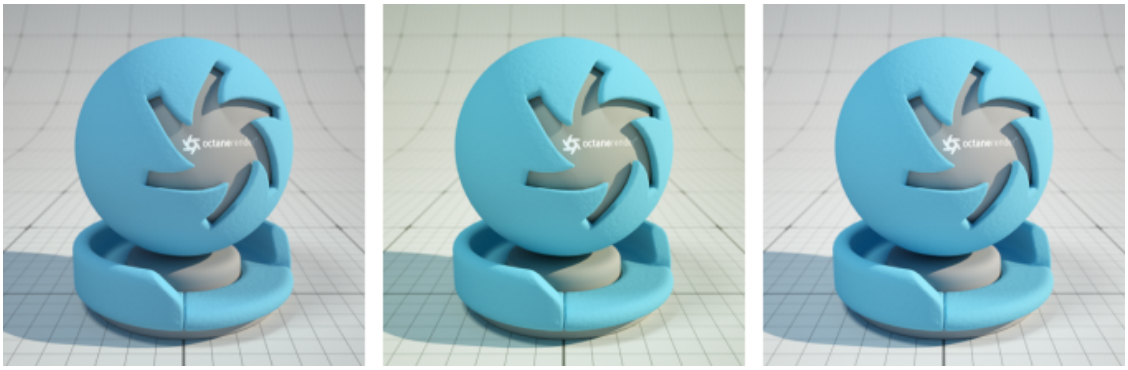


Figure 2: Adjusting the Neutral Response Curve

- **Gamma** - Adjusts the gamma of the render, and controls the overall brightness of an image. Images that are not properly corrected look either bleached out or too dark. Varying the gamma correction amount changes not only the brightness, but also the ratios of red-to-green-to-blue.
- **White Point** - Specifies the color for adjusting the tint that produces and simulates the relative temperature cast throughout the image by different light sources.
- **Vignetting** - Increases the amount of darkening in the corners of the render. With moderate use, it can increase the realism of the render. OctaneRender doesn't apply vignetting to any of the beauty passes except for the main pass.
- **Saturation** - Adjusts the amount of color saturation for the render.
- **Hot Pixel Removal** - Removes bright pixels (fireflies) during the rendering process. While many pixels can disappear if the render progresses, Hot Pixel Removal removes the bright pixels at a much lower sample per pixel.
- **Pre-Multiplied Alpha** - Multiplies any transparency value of the output pixel by the pixel's color.
- **Disable Partial Alpha** - Makes pixels that are partially transparent (Alpha > **0**) fully opaque.
- **Dithering** - Adds random noise, which removes banding in clean images.
- **Saturate To White** - When the sun is too bright, it creates multicolored reflections. Increasing this value changes the colors to white. This applies to all sources of light. You can push saturated parts of the render towards pure white with this option. This helps avoid large patches of saturated colors caused by over-bright light sources, such as bright colored emitters, or reflected sunlight off colored surfaces.
- **Minimum Display Samples** - The minimum amount of samples calculated before the image displays. This feature reduces the noise by a significant amount when navigating, and is useful for real-time walkthroughs. When using multiple GPUs, we recommend setting this value as a multiple of the number of available GPUs for rendering - e.g., if rendering with 4 GPUs, set this value to **4** or **8**.
- **Maximum Tonemap Interval** - Maximum interval between tone maps in seconds.

Spectral AI Denoiser

- **Enable Denoising** - Enables the Spectral AI Denoiser. This denoises some beauty passes, including the main beauty pass, and writes the outputs into separate render passes.
- **Denoise Volumes** - If enabled, the Spectral AI Denoiser denoises volumes in the scene. Otherwise, OctaneRender doesn't denoise volumes by default.
- **Denoise On Completion** - If enabled, OctaneRender denoises beauty passes only once at the end of a render. You should disable this option while rendering with an interactive region.
- **Min. Denoiser Samples** - Minimum number of samples per pixel until the denoiser kicks in. Only valid when the **Denoise Once** option is **False**.
- **Max. Denoiser Interval** - Maximum interval between denoiser runs (in seconds). Only valid when the **Denoise Once** option is **False**.
- **Blend** - Setting to **0** results in a fully denoised image, and setting to **1** results with the original image. An intermediate value produces a blend between the denoised image and the original image.

Post Processing

Post-processing effects add various post-rendered effects to a rendered image. The post effects available in OctaneRender® for Unity® include **Bloom**, **Glare**, and **Spectral Intensity/Shift**.

The post-processing effects are located in the **Post Processing**[1] rollout of a **PBR**[2] **Render Target** (Figure 1).

[1] Effects such as Bloom and Glare that are applied after a scene has been rendered.

[2] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

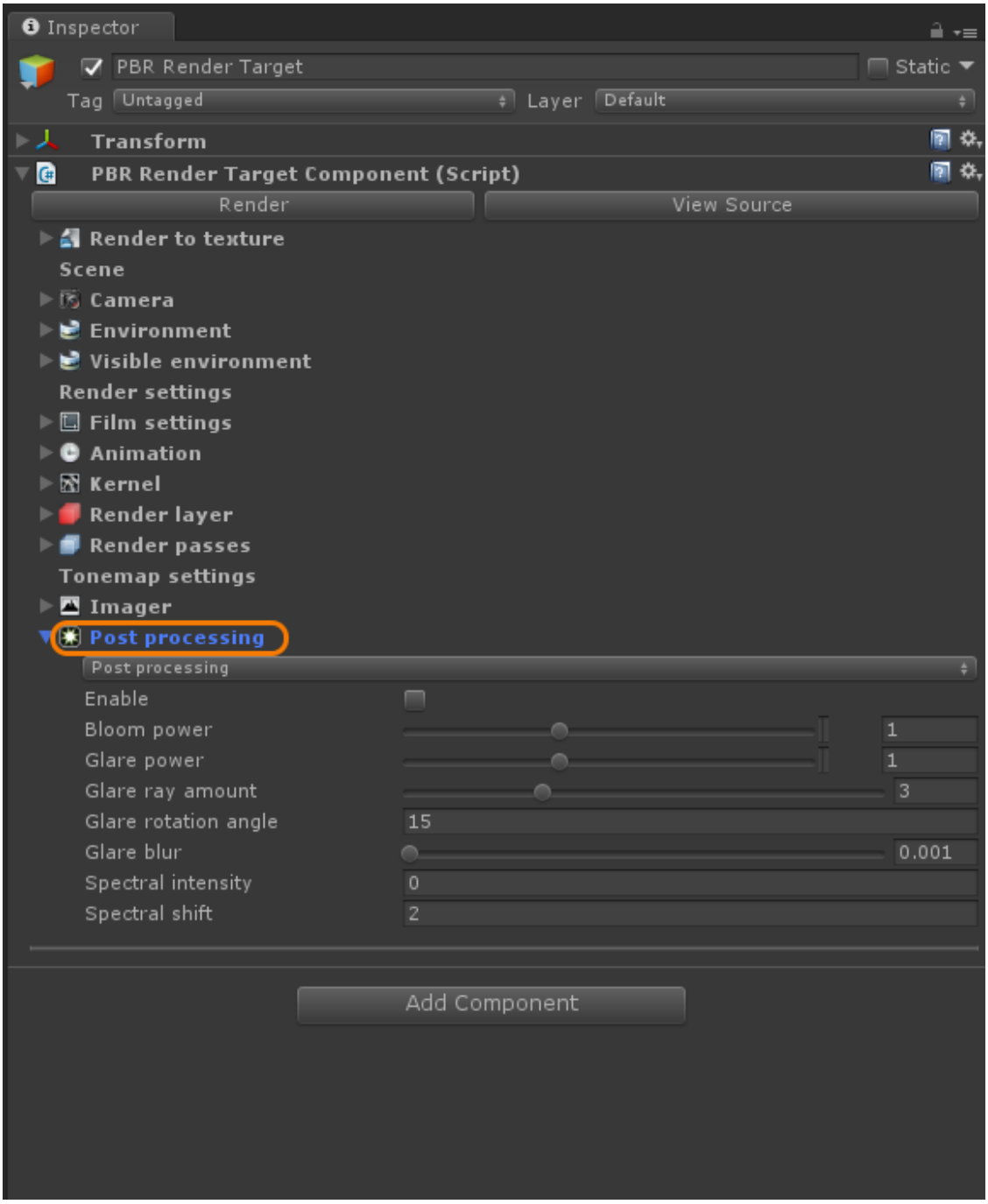


Figure 1: The Post Processing Parameters

Post Processing Parameters

- **Enable** - Enables or disables post-processing effects on the resulting render. Post-processing is disabled by default.
- **Bloom Power** - Controls the size of the glow originating from an emitter, the size of the halo of light originating from the sun, and/or concentrated light on reflective glossy materials in order to add a bloom effect to the rendered image.
- **Glare Power** - Controls the size of the visible rays originating from an emitter, the size of the glare originating from reflective glossy materials at a point where the concentration of light is at its highest in order to add a glare effect to the rendered image.
- **Glare Ray Amount** - Controls the number of radiated/reflected visible rays.
- **Glare Rotation Angle** - Adjusts the rotation of the glare relative to the object. A glare angle of **-90** or **90** results in one main horizontal glare, and a glare angle of **0** results in one main vertical glare.
- **Glare Blur** - Controls the glare sharpness. Smaller values result in a crisp linear glare, and higher values result in softer glare.
- **Spectral Intensity** - Adjusts the intensity distribution of the rays across a source. This affects the brightness of the radiant energy.
- **Spectral Shift** - Adjusts the displacement of the spectrum as the frequency of light emitted from a source changes, affecting the visible spectrum relative to the source on the scene. The shift is evident by a color change, similar to the doppler effect, as the distance traveled by the ray from its source increases or decreases.

Shadow Catcher

The **Shadow Catcher**[1] option can create shadows cast by objects onto the surrounding geometry. The shadows cast are not limited to a ground plane, but can be cast onto other surfaces of varying shapes (Figure 1).

[1]The Shadow Catcher can be used to create shadows cast by objects onto the surrounding background imagery. The shadows cast are not limited to simply a ground plane but can be cast onto other surfaces of varying shapes.

Figure 1: A Model is integrated into an image using the Shadow Catching material

To enable this feature, go to the **OctaneVR**® window and click on the OctaneRender® **Diffuse**[1] material applied to the shadow catching surfaces, then from the **Node Inspector**, click on the **Shadow Catcher** checkbox (Figure 2).

[1]Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh emitters.

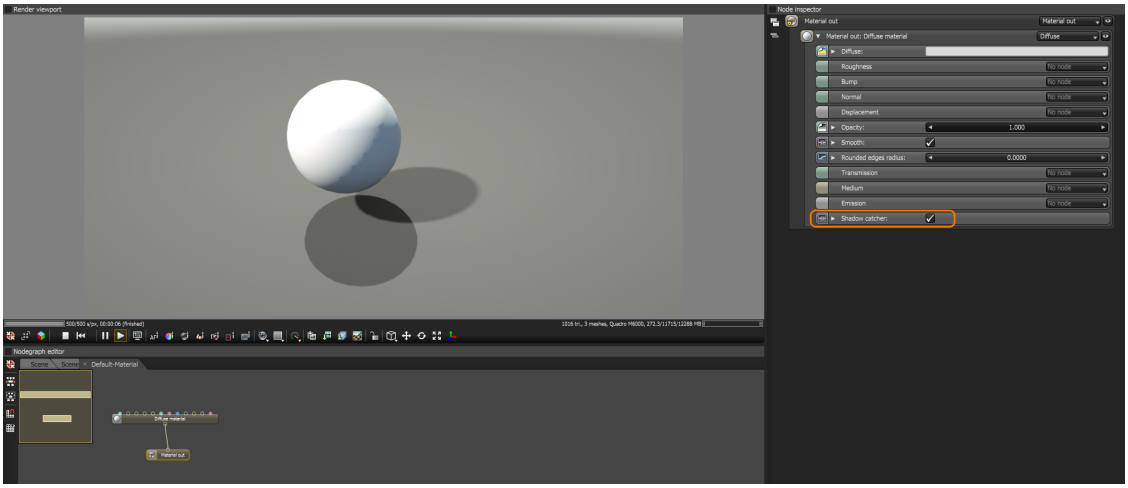


Figure 2: Activating the Shadow Catcher option

In the **PBR**[1] **Render Target**, you need to activate **Alpha Channel**[2] and disable **Keep Environment** (Figure 3). When the image is rendered, the shadows appear over the transparent parts of the surface. You can use this image in a compositing package to merge the object and the shadows into the composition.

[1] A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.
[2] A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.

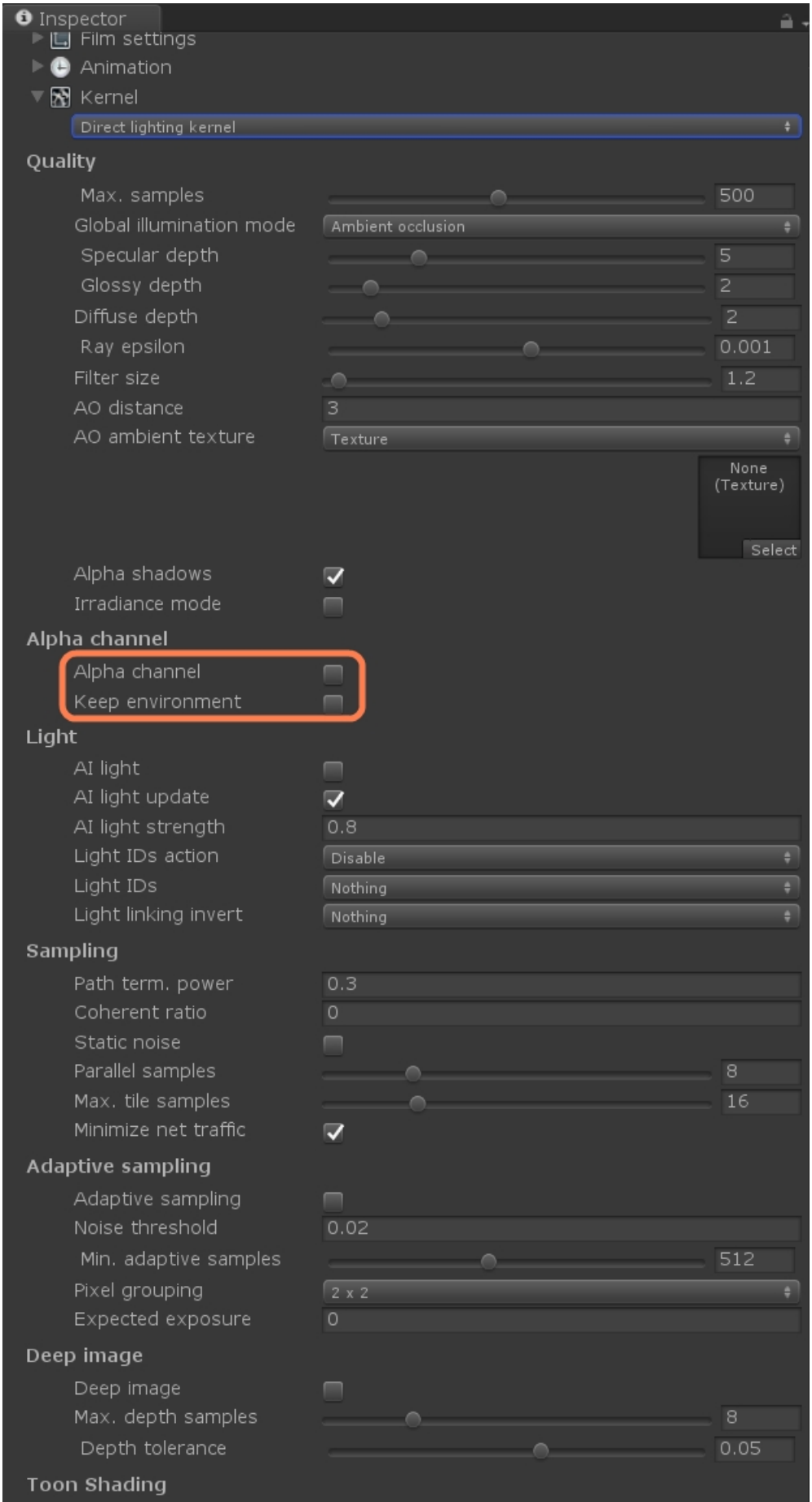

Inspector
Film settings
Animation
Kernel
Direct lighting kernel
Quality
Max. samples
500
Global illumination mode
Ambient occlusion
Specular depth
5
Glossy depth
2
Diffuse depth
2
Ray epsilon
0.001
Filter size
1.2
AO distance
3
AO ambient texture
Texture
None
(Texture)
Select
Alpha shadows
Irradiance mode
Alpha channel
Alpha channel
Keep environment
Light
AI light
AI light update
AI light strength
0.8
Light IDs action
Disable
Light IDs
Nothing
Light linking invert
Nothing
Sampling
Path term. power
0.3
Coherent ratio
0
Static noise
Parallel samples
8
Max. tile samples
16
Minimize net traffic
Adaptive sampling
Adaptive sampling
Noise threshold
0.02
Min. adaptive samples
512
Pixel grouping
2 x 2
Expected exposure
0
Deep image
Deep image
Max. depth samples
8
Depth tolerance
0.05
Toon Shading

Figure 3: Activate Alpha and disable Keep Environment in the Render Kernel settings

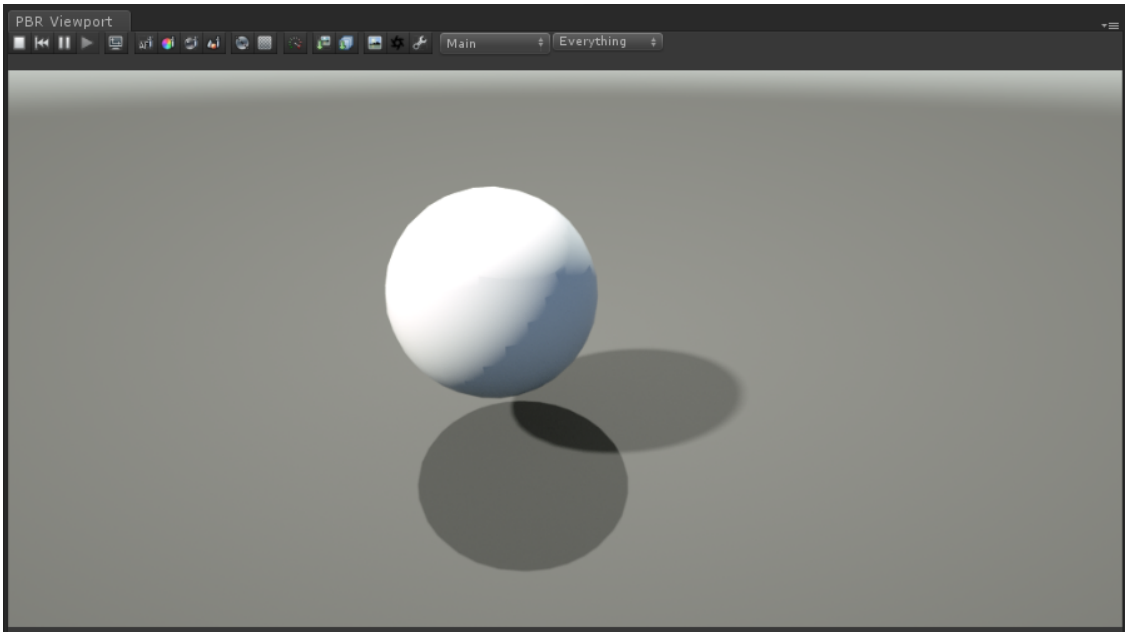


Figure 4: Shadows are cast onto the environment, while the surface receiving the shadows is transparent

Glossary

A

Absorption
Defines how fast light is absorbed while passing through a medium.

Adaptive Sampling
A method of sampling that determines if areas of a rendering require more sampling than other areas instead of sampling the entire rendering equally.

Alembic
An open format used to bake animated scenes for easy transfer between digital content creation tools.

Alpha Channel
A greyscale image used to determine which areas of a texture map are opaque and which areas are transparent.

Anti-Ghosting
The automatic or manual correction involved in the merging a stack of images during the creation of a High Dynamic Range image. The process aims to correct the strange effect when objects that change position in the image set is partially visible (like a ghost) in the final HDR image.

Aperture
Determines how much light enters a camera lens. A large aperture produces a narrow depth of field and a small aperture produces a wide depth of field.

AR
Viewing a conceptual three dimensional scene in context to see how it might look in the real world.

Augmented Reality

Viewing a conceptual three dimensional scene in context to see how it might look in the real world.

B

Batch Rendering

The process of assigning sequential portions of frames to be rendered across multiple systems.

Black Body

An opaque object that emits thermal radiation. In Octane, this is used to designate illumination properties for mesh emitters.

D

Deep Image

Renders frames with multiple depth samples in addition to typical color and opacity channels.

Depth Buffer

A measure of object distances from the camera typically represented as a grayscale image.

Depth of Field

The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

Diffuse

Amount of diffusion, or the reflection of light photons at different angles from an uneven or granular surface. Used for dull, non-reflecting materials or mesh

emitters.

Diffuse material

Used for dull, non-reflecting materials or mesh emitters.

Displacement

The process of utilizing a 2D texture map to generate 3D surface relief. As opposed to bump and normal mapping, Displacement mapping does not only provide the illusion of depth but it effectively displaces the actual geometric position of points over the textured surface.

DoF

The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

Drivers

Files that allow hardware devices to communicate with an operating system. In the case of Octane, the latest Nvidia drivers should be used.

E

Effective Focus Range

The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

Emissions

The process by which a Black body or Texture is used to emit light from a surface.

EXR

Also known as OpenEXR. This image file format was developed by Industrial Light & Magic and provides a High Dynamic Range image capable of storing deep image data on a frame-by-frame basis.

F

FBX

.fbx (Filmbox) is a proprietary file format developed by Kaydara and owned by Autodesk since 2006. It is used to provide interoperability between digital content creation applications. As of Octane 3.07, a scene node will also be available as an FBX file, allowing for quick and easy transport of assets from industry standard DCC applications

Field of View

The area that is visible to a camera lens usually measured in millimeters. A wide angle lens provides a larger field of view and a telephoto lens provides a narrow field of view.

Focus Range

The distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. source: wikipedia (https://en.wikipedia.org/wiki/Depth_of_field)

FoV

The area that is visible to a camera lens usually measured in millimeters. A wide angle lens provides a larger field of view and a telephoto lens provides a narrow field of view.

G

Gamma

The function or attribute used to code or decode luminance for common displays. The computer graphics industry has set a standard gamma setting of 2.2 making it the most common default for 3D modelling and rendering applications.

Glossy

The measure of how well light is reflected from a surface in the specular direction, the amount and way in which the light is spread around the specular direction, and the change in specular reflection as the specular angle changes. Used for shiny materials such as plastics or metals.

Glossy material

Used for shiny materials such as plastics or metals.

GPU

The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

Graphics Card

The GPU is responsible for displaying graphical elements on a computer display. The GPU plays a key role in the Octane rendering process as the CUDA cores are utilized during the rendering process.

H

Hardware

Any physical device present in a computer system. A Nvidia GPU is a required hardware device for using the Octane Render engine.

HDRI

An image which presents more than 8 bit per color channel unlike most common image formats.

High Dynamic Range Image

An image which presents more than 8 bit per color channel unlike most common image formats.

I

IES

An IES light is the lighting information representing the real-world lighting values for specific light fixtures. For more information, visit http://www.ies.org/lighting/.

IFL

(Image File List) file is an ASCII file that constructs an animation by listing single-frame bitmap files to be used for each rendered frame. When you assign an IFL file as a bitmap, rendering steps through each specified frame, resulting in an animated map. (reference: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax/files/GUID-CA63616D-9E87-42FC-8E84-D67E1990EE71-htm.html)

Independent Software Vendor

An individual or business that builds, develops and sells consumer or enterprise software. Although ISV-provided software is consumed by end users, it remains the property of the vendor. An ISV is also known as a software publisher.

Instancing

Instancing an object means taking a single imported mesh object, such as an OBJ or an FBX and making multiple copies, each of which can be placed in different parts of the scene. This saves an enormous amount of computational resources because only a single object is loaded into the scene.

Interactive Photorealistic Rendering

Provides artists a quick preview of the image prior to the final render, and efficiently allows for adjusting some elements in the scene such as lights, shaders and textures interactively. An IPR image contains shading and lighting data including some for visibility, in addition to the software render.

IPR

Provides artists a quick preview of the image prior to the final render, and efficiently allows for adjusting some elements in the scene such as lights, shaders and textures interactively. An IPR image contains shading and lighting data including some for visibility, in addition to the software render.

ISV

An individual or business that builds, develops and sells consumer or enterprise software. Although ISV-provided software is consumed by end users, it remains the property of the vendor. An ISV is also known as a software publisher.

K

Kernels

By definition, this is the central or most important part of something. In Octane, the Kernels are the heart of the render engine.

L

LDR

Image formats that have 8 bits per color channel such as the common image formats JPEG, PNG, GIF among others.

Low Dynamic Range

Image formats that have 8 bits per color channel such as the common image formats JPEG, PNG, GIF among others.

Lua

A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.

Lua Scripting

A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.A scripting language that supports procedural, object-oriented, functional, and data-driven programming. It can be used to extend Octane's functionality.

M

Material

The representation of the surface or volume properties of an object.

Materials

A set of attributes or parameters that describe surface characteristics.

Mediums

The behavior of light inside a surface volume described by scatter, absorption, and transmission characteristics.

Mesh Emitters

The ability for a surface to emit illumination usually described by a Black Body or Texture emission type.

Mix material

Used to mix any two material types.

Mixed

The ratio of diffuse and specular reflection.

Motion Blur

An optical phenomenon that occurs when a camera's shutter opens and closes too slowly to capture movement without recording a blurring of the subject.

N

Network Rendering

The utilization of multiple CPUs or GPUs over a network to complete the rendering process.

NGE

Node Graph Editor

O

Open Shader Language

A shading language developed by Sony Pictures Imageworks. There are multiple render engines that utilize OSL as it is particularly suited for physically-based renderers.

Open SubDiv Surfaces

A set of open source libraries that implement high performance subdivision surface (subdiv) evaluation on massively parallel CPU and GPU architectures. This code path is optimized for drawing deforming surfaces with static topology at interactive framerates. Source: Pixar (http://-graphics.pixar.com/opensubdiv/docs/intro.html).

OpenVDB

Dreamworks' open-source C++ library housing the data structures and tools implementation for storing and manipulating volume data, like smoke and other amorphous materials. The purpose of OpenVDB is mostly to have an efficient way to store volumetric data in memory and on disk. It has evolved into a more general toolkit that also lets you accomplish other things, such as fracturing volumes, converting meshes to volumes and vice versa. However, it does not include a computational fluid dynamics solver, and therefore it cannot procedurally generate smoke or fire. OpenVDB is fully integrated as a library in OctaneRender. For more information about OpenVDB, check at http://www.openvdb.org/.

ORBX

The ORBX file format is the best way to transfer scene files from 3D Authoring software programs that use the Octane Plug-in such as Octane for Maya, Octane for Cinema 4D, or OctaneRender Standalone. This format is more efficient than FBX when working with Octane specific data as it provides a flexible, application independent format. ORBX is a container format that includes all animation data, models, textures etc. that is needed to transfer an Octane scene from one application to another.

Out-of-Core

When scene assets become too large to load completely onto the system's GPU, Out-of-Core technology allows the render engine to utilize the CPU to assist in the rendering process.

P

PBR

A contemporary shading and rendering process that seeks to simplify shading characteristics while providing a more accurate representation of lighting in the real world.

Portal

A technique that assists the render kernel with exterior light sources that illuminate interiors. In interior renderings with windows, it is difficult for the path tracer to find light from the outside environment and optimally render the scene. Portals are planes that are added to the scene with the Portal material applied to them.

Post Processing

Effects such as Bloom and Glare that are applied after a scene has been rendered.

Projections

Methods for orienting 2D texture maps onto 3D surfaces.

Proxy

An object saved as a separate file with the purpose of being reused in larger scenes. This is used to minimize any addition to the total polygon count in the scene, especially if the scene requires the same object to appear several times. If used in conjunction with instancing, Proxies help keep very large scenes from reaching polygon limits and also keeps the relative file size of the main project file manageable.

Proxy Server

A Proxy Server, also known as an application-level gateway, is an intermediary server between the local network and the external servers from which a client is requesting a service. The external servers will only see the network proxy server's IP address thus providing some degree of security and privacy. There are various kinds of proxies, the most common are Web Proxies.

R

RAW

In HDR imaging, this refers to minimally processed HDR image formats. Raw files can have 12 or 14 bits per color channel, although the available dynamic range might be cut down due to noise.

Render Layers

Render layers allow users to separate their scene geometry into parts, where one part is meant to be visible and the rest of the other parts "capture" the side effects of the visible geometry. The layers allow different objects to be rendered into separate images where, in turn, some normal render passes may be applied. The Render layers are meant for compositing and not to hide parts of the scene.

Render Passes

Render passes allow a rendered frame to be further broken down beyond the capabilities of Render Layers. Render Passes vary among render engines but typically they allow an image to be separated into its fundamental visual components such as diffuse, ambient, specular, etc..

S

Scattering

Defines how fast light gets scattered when traveling through the medium.

Shadow Catcher

The Shadow Catcher can be used to create shadows cast by objects onto the surrounding background imagery. The shadows cast are not limited to simply a ground plane but can be cast onto other surfaces of varying shapes.

Spectral Light Transport

A technique in which a scene's light transport is modeled with real wavelengths. Spectral rendering can also simulate light sources and objects more effectively, as the light's emission spectrum can be used to release photons at a particular wavelength in proportion to the spectrum. Source: Wikipedia (https://en.wikipedia.org/wiki/Spectral_rendering).

Specular

Amount of specular reflection, or the mirror-like reflection of light photons at the same angle. Used for transparent materials such as glass and water.

Specular material

Used for transparent materials such as glass and water.

T

Texture Baking

A process in which scene lighting is "baked" into a texture map based on an object's UV texture coordinates. The resulting texture can then be mapped back onto the surface to create realistic lighting in a real-time rendering environment. This technique is frequently used in game engines and virtual reality for creating realistic environments with minimal rendering overhead.

Textures

Textures are used to add details to a surface. Textures can be procedural or imported raster files.

TMO

Maps HDR images to standard displays which have a limited dynamic range. The more prominent TMOs are Mantiuk'06, Reinhard'02, Drago, and Durand.

Tone Mapping

A term referring to various methods of "converting" HDR images into a viewable format.

Tone Mapping Operator

Maps HDR images to standard displays which have a limited dynamic range. The more prominent TMOs are Mantiuk'06, Reinhard'02, Drago, and Durand.

Transformations

Tools used to rotate and position 2D and 3D texture maps onto 3D surfaces.

Transmission

A surface characteristic that determines if light may pass through a surface volume.

U

Unbiased Rendering

Unbiased rendering does not introduce any "errors" or shortcuts into the rendering process. It will calculate all scene data using real-world calculations. This type of rendering is known for producing exceptional render quality.

V

VDB

Dreamworks' open-source C++ library housing the data structures and tools implementation for storing and manipulating volume data, like smoke and other

amorphous materials. The purpose of OpenVDB is mostly to have an efficient way to store volumetric data in memory and on disk. It has evolved into a more general toolkit that also lets you accomplish other things, such as fracturing volumes, converting meshes to volumes and vice versa. However, it does not include a computational fluid dynamics solver, and therefore it cannot procedurally generate smoke or fire. OpenVDB is fully integrated as a library in OctaneRender. For more information about OpenVDB, please see http://www.openvdb.org/.

Virtual Reality

Immersively engaging and experiencing depth perception in a three dimensional scene through stereo vision goggles and head-mounted displays.

Volume Medium

A shading system designed to render volumes such as smoke and fog.

VR

Immersively engaging and experiencing depth perception in a three dimensional scene through stereo vision goggles and head-mounted displays.

Z

Z-Buffer

A measure of object distances from the camera typically represented as a grayscale image.

Z-Depth

A measure of object distances from the camera typically represented as a grayscale image.

Index

I

K

L

M

N

O

P

V

Z