WHAT IS "TOON SHADERS PRO FOR URP"?

Toon Shaders Pro for URP is a collection of shaders for creating toon-styled games, in the cel-shaded style made popular by games like *Jet Set Radio* and *The Legend of Zelda: Breath of the Wild*. The main toon shader gives you control over the different kinds of lighting applied to objects, including terrains, and the outline shader gives you several options for rendering lines along the edges of objects.

The latest version of this asset has been tested with the following Unity versions:

- **Unity 2022.3, 6.0, 6.1, 6.2, 6.3, 6.4**

SUPPORT

Sometimes things break! If you've encountered an error and this README doesn't have the answers (or if you have feature requests), then pop me an email at **danielilett+toonshaders@gmail.com** and I'll try to sort you out. Please:

- **Clearly** describe the problem you are having and what steps I can take to reproduce the error.
- Include the **Unity version you are using.**
- Also include the **package version** you are using (you can find this in the Package Manager).
- Include the **render path** (i.e. Forward, Forward+, Deferred) – this option is on your URP settings asset.
- A**ttach images or short videos** where necessary to describe your problem.

Following these steps will help me fix your issue as quickly as possible!

SETUP

Folder Structure: Upon installing the pack, all the assets will be contained in the "Toon Shaders Pro" folder. All demo scenes are included under "Toon Shaders Pro/Demo", containing an example scene with materials and post processing volumes configured to work with this asset pack.

Most of the shaders are included inside "Toon Shaders Pro/Shaders". The post processing assets are included in the "Retro Shaders Pro/Shaders/Outline" folder.

Warning: There is a chance that you will encounter problems when you import the pack due to the Shader Graph version of the toon shader. This problem relates to the shader keywords used for lighting. If you have this issue, please go into *Preferences -> Shader Graph* (before Unity 6) or *Project Settings -> Shader Graph* (Unity 6 and after) to increase the shader variant limit – this might need to be at least 512.

TOON SHADER FOR MESHES

The main Toon shader is intended to be used for regular objects in your game, and there are two versions of it. For most objects, you should just create a material with the **Toon Shaders Pro/URP/Toon** shader.

If you want to make your own shaders which have toon capabilities, you might find it helpful to duplicate the **Toon Shaders Pro/URP/Toon (Graph)** Shader Graph and modify the copy. However, this version doesn't support every feature of the code-based Toon shader.

TOON SHADER FOR TERRAINS

There is also a version of the toon shader specifically for terrains. This version should only be attached to terrains, and it will use the splatmaps applied to the terrain to determine its albedo color. The "**Toon Shaders Pro/URP/Toon (Terrain)**" shader should be used for the material, which will let you modify the lighting, just like regular objects.

OUTLINE POST PROCESS

This shader pack uses **Universal Render Pipeline**'s **ScriptableRenderFeature** functionality for the custom Outline post processing effect. The Unity documentation will outline the basics of URP if you're not familiar with how to create custom renderers.

Please follow these steps to enable an effect in your scene:

- Find your **URP Renderer Asset** and add the effect(s) you wish to use in the **Renderer Features** section at the bottom.
    - This is most commonly found in the Assets/Settings folder if you created a new project using the URP template from the Unity Hub.
    - This asset will be named something like "UniversalRP-HighQuality" (Unity versions 2022.3 and prior) or "PC_RPAsset" (Unity 6) by default.
    - *Toon Shaders Pro* also includes a ready-made asset named "**URP-Toon-Renderer**" in the Demo folder, which has the Outline effect pre-added.

- Create a volume profile asset via **Create -> Volume Profile** and add the Outline effect (and any other effects that you want to use) to the profile.
- Add a volume to your scene via *GameObject -> Volume* and attach the volume profile.
- Tweak the settings on your volume profile as desired.

RENDER GRAPH SUPPORT (UNITY 6+)

Render Graph is a system for setting up and optimizing resources for render passes. In Unity 2022.3, *Toon Shaders Pro* uses a fallback since Render Graph wasn't yet implemented in URP.

In Unity 6.0 - 6.3, the Outline effect supports RG and non-RG workflows. It is only necessary to use the non-RG Compatibility Mode if your project contains other effects which don't support RG.

Compatibility Mode was removed in Unity 6.4, and the Outline effect will always use the RG workflow in that version and beyond.

ADDITIONAL WARNINGS

These shaders are designed for **linear color space**, so you may encounter issues in gamma space. To swap between color spaces, go to *Project Settings->Player->Other Settings* and find the **Color Space** dropdown option.

ASSETS INCLUDED

The following assets are included in the pack.

TOON SHADER

The main Toon shader lets you separate the diffuse, specular, and rim lighting and the shadows shown on the object and apply cutoff thresholds for each. You can also specify tint values for the areas of the mesh in light and darkness, or apply a normal map for even more interesting lighting.

SURFACE OPTIONS

- **Surface Type** – Toggle between *Opaque* and *Transparent* rendering.
- **Render Face** – Choose whether to render *Front*, *Back*, or *Both* faces.
- **Alpha Clip** – Toggle whether the shader should apply the *Alpha Clip Threshold*.
- **Alpha Clip Threshold** – Pixels with final base color alpha values below this threshold will be culled if *Alpha Clip* is enabled.
- **Receive Shadows** – Toggle whether realtime shadows should be applied to the object.
- **Shadow Thresholds** – Provide a toon-style cutoff to the realtime shadows.

TOON PROPERTIES

DIFFUSE SETTINGS

- **Use Second Threshold** – When ticked, the shader introduces a third lighting tint value for diffuse lighting and lets you specify two thresholds: one for dark -> midtones, and another for midtones -> light.
- **Base Color** – The albedo color of the object.

- **Base Texture** – An albedo texture to apply more color detail than *Base Color* alone.
- **Base Texture Tiling and Offset** – Lets you scale and translate the texture across the object surface. Note that these values are also used to scale and translate *Specular Map* and *Normal Map*.
- **Light Tint** – Color multiplier applied to fully-lit areas of the object.
- **Middle Tint** – Only active if *Use Second Threshold* is ticked. Color multiplier applied to midtones on the object.
- **Shadow Tint** – Color multiplier applied to shadowed regions of the object.
- **Ambient Light Strength** – How strongly Unity's default ambient lighting is applied to the object.
- **Diffuse Thresholds** – Provide a toon-style cutoff to the diffuse lighting. Diffuse lighting comes from the angle between the light and the normal vector of the object surface.

SPECULAR SETTINGS

- **Specular Map** – Control the specular strength at different parts of the object using a texture.
- **Specular Strength** – Control the global specular strength of the object.
- **Specular Offset Noise Map** – Use a noise texture, such as Perlin noise, to modify the shape of the specular highlights.
- **Specular Offset Noise Strength** – Controls the strength of the offset applied to specular light values.
- **Specular Color** – Multiplier for specular light.
- **Glossiness** – Higher glossiness makes the object look shiny and results in smaller highlights.
- **Specular Thresholds** – Provide a toon-style cutoff to the specular lighting. Specular lighting arises due to the angle between the viewer and the light rays reflected from the light source in the normal vector of the object's surface.

RIM LIGHTING SETTINGS

- **Rim Color** – Multiplier for rim light.
- **Rim Thresholds** – Provide a toon-style cutoff to the rim lighting. Rim lighting (also called Fresnel) comes from the angle between the viewing angle and the normal vector of the surface. Typically, looking at the object at shallow angles results in high rim lighting.
- **Rim Extension** – Lets you extend rim lighting across the surface further than usual.

NORMAL MAPPING SETTINGS

- **Normal Map** – Normal maps modify the surface normals for finer details that are difficult to capture with mesh geometry alone.
- **Normal Strength** – Larger values have a more pronounced effect on the surface normals.

USAGE TIPS

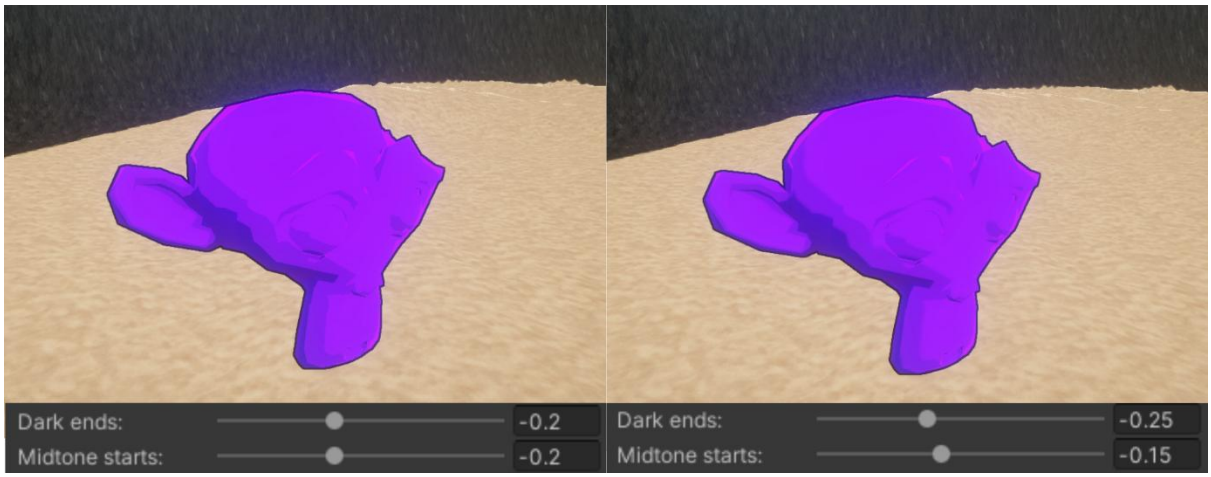
Here are some quick tips for using the toon shader effectively.

DIFFUSE THRESHOLD SMOOTHING

Each of the diffuse lighting thresholds uses a Smoothstep function under the hood, which means you can change how smooth the transition from light to dark appears. If you want a hard cutoff, then use

the same value for the upper and lower ends of the threshold (e.g. Dark ends and Midtone starts). If you want a small blending region between light and dark, then make the upper value a little higher than the lower value.

Internal lighting values run from -1 to +1, so that is the range of possible values you can use for the thresholds.



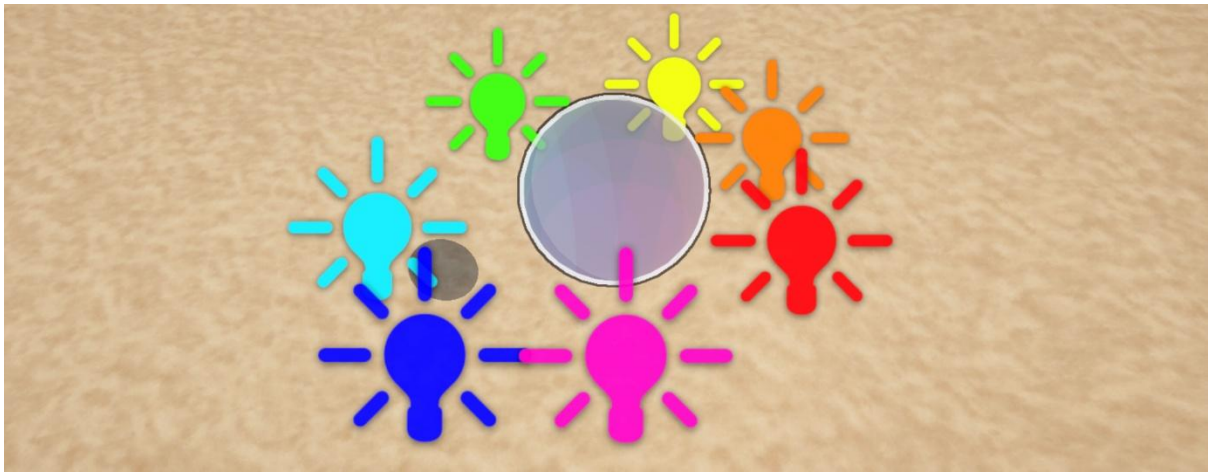
MULTIPLE THRESHOLDS

You can choose to use one cutoff threshold or two. When you use one, your object will have lit and dark areas. When you use two, your object will additionally have a midtone region. You may configure the amount of lighting assigned to each region.

MULTIPLE LIGHTS

Lighting is additive, so the individual diffuse, specular, and rim lighting values from each light get added together and applied to the object.

All lighting and thresholding is applied on a per-light basis. If you have a main direction light and several point lights near your object, every one will apply toon-style lighting to the object. This is also additive, so having several lights around an object may make it extremely bright.



NORMAL MAPPING

If you would like some extra detail on your object without resorting to using the Base Texture for details, try adding a normal map and letting the shader calculate lighting using the new normal vectors. Even while using block colors or a simple base texture, a good quality normal map can help to achieve striking visuals.

TOON (SHADER GRAPH)

This is largely a copy of the base Toon shader, but made with Shader Graph. Most of the toon lighting logic is implemented in a subgraph named "**CalculateToonLighting**". I've included this to make it simpler to make your own effects.

All you need to do is duplicate this graph and add your own nodes however you want. Or, you can drag the CalculateToonLighting subgraph into any graph. Be warned, though: it has a lot of inputs! I've tried to include sensible defaults where possible.

TOON TERRAIN SHADER

The terrain version of the shader has exclusive additional features that only make sense with terrains, and is missing the Base Map option, as the base/albedo texture now comes from the terrain splatmaps and layers.

- **Use Stochastic Texturing** – When enabled, the shader uses UV offsets to prevent the input textures from looking tiled, which is a common problem with terrain textures. This method uses three times as many texture samples for each terrain base layer.

OUTLINE POST PROCESS

The Outline effect is a collection of multiple outline rendering algorithms which each have different settings and use cases.

The options provided with this feature change quite drastically based on the *Outline Type*, but I will list them all here for convenience.

RENDERING OPTIONS

- **Render Pass Event** – Choose whether to render the outlines before or after URP's internal post processing effects. That includes effects such as color correction and bloom.
- **Outline Type** – Choose which outline algorithm to use. There are six choices:
    - **No Outlines** – Don't render any outlines.
    - **Depth Normal Outlines** – Detect small gradients in the color, depth, and normal vector information across each pixel to find edges.
    - **High Quality Masked Object Outlines** – Render specific object layers to a separate mask texture and find edges in that mask, allowing for varying thickness and a greater level of customization than other algorithms.
    - **Pixel Width Masked Object Outlines** – Use the same object mask as above, but use a cheaper algorithm for pixel-width outlines.

- **Hull Outlines** – Render each mesh a second time while inverting and inflating the mesh hull.
- **Debug Outline Mask** – Render only the object mask used for the mask-based effects to the screen. Helpful for finding objects which may have been incorrectly assigned to the wrong layer.

OUTLINE OPTIONS

- **Outline Color** – Color applied to the outlines. Some effects may use the transparency value.
- **Use Noise Offsets** - Should the outline effect use a Perlin noise offset? Warning: this might be a bit expensive.
- **Noise Scale** - Scale to use for the Perlin noise generator. Higher values result in more noise variation over UV space.
- **Noise Offset** - Offset value to use for the Perlin noise generator. You can manually change this to tweak the appearance of the noise pattern.
- **Noise Strength** - How strongly the noise values offset the outline UV calculations.

DEPTH-NORMALS OPTIONS

- **Color Sensitivity** – Color gradients greater than this threshold will be detected as edges.
- **Color Strength** – How strongly a detected color-edge should contribute to the final edge-detection value.
- **Depth Sensitivity** – Depth gradients greater than this threshold will be detected as edges.
- **Depth Strength** – How strongly a detected depth-edge should contribute to the final edge-detection value.
- **Normal Sensitivity** – Normal vector gradients greater than this threshold will be detected as edges.
- **Normal Strength** – How strongly a detected normal-edge should contribute to the final edge-detection value.
- **Depth Threshold** – Pixels with a depth higher than this threshold are never detected as edges.

OBJECT MASK OPTIONS

- **Object Mask** – Set a layer mask to determine which object layers get rendered into the mask texture.
- **Mask Ignore Depth** – If enabled, objects get rendered into the mask texture without checking for depth. If rendering many objects, this likely means objects will be rendered out of order. Useful if you render only one object and you want it to be seen through walls.
- **Mask Drawing Mode** – Choose how to draw the outlines.
    - **Per Object**: draw an outline around each individual object. A distinct object should have a unique object pivot position in world space.
    - **Per Triangle**: draw an outline around each triangle of each mesh.
    - **Merge All Masked Objects**: draw an outline around the combined shape of everything that is in the mask.

    - **Vertex Colors**: draw an outline around each unique vertex color. Attach unique vertex colors to each object to draw outlines around distinct objects, or group objects by using the same vertex color.
    - **Renderer Shader User Value**: in Unity 6.3+ only. Draw an outline around each distinct RSUV. Set a unique RSUV to each object to draw outlines around distinct objects, or group objects by using the same RSUV.
- **Light Modes** – Determines which objects get drawn into the mask, based on which shader they are using originally. Hovering over this option in the Inspector will bring up a tooltip with helpful context about this setting.
- **Render Queue** – Choose whether to render only opaques, only transparents, or all objects into the mask. However, only opaque objects are properly supported.

MASK OUTLINE OPTIONS

- **Masked Outline Thickness** – Set the thickness of the outlines. Larger values get more expensive, so try to keep this as low as possible.
- **Masked Outline Smoothing** – Decreases the outline contribution of further away pixels for a false smoothing effect.
- **Outline Draw Sides** – Choose to draw objects 'inside' the mask, 'outside' it, or on both 'sides'. Drawing on both sides results in a thicker outline.
- **Outline Fade Start** – Start to fade out the outlines when the object is this many Unity units from the camera.
- **Outline Fade End** – When the object is at this distance from the camera, the objects become totally invisible.

NORMALS OPTIONS

- **Use Depth Normals** – When using mask-based outlines, choose whether to layer normal-based edge detection for details 'inside' the mask.

HULL OUTLINE OPTIONS

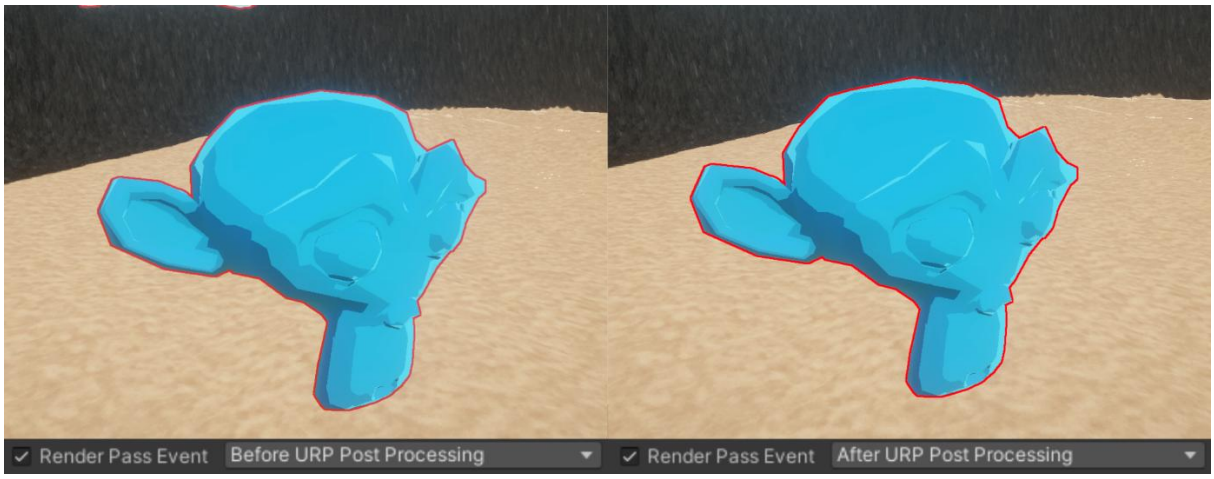
- **Outline Thickness** – How far the outlines extend away from the object.
- **Outline Transparency** – Choose whether to render the outlines with transparency. Since this method uses actual mesh geometry, this has an impact on rendering order.
- **Outline Lighting** – Should the outlines use simple diffuse lighting based only on the main directional light?
- **Flip Outline Direction** – When lighting is being used, should the direction of the normal vector be inverted?
- **Outline Min Lighting** – Set a minimum lighting value to use for the diffuse outline lighting.

USAGE TIPS

RENDERING ORDER

Rendering outlines *before* other post processing effects means that they are subject to those effects. For example, color correction and bloom effects will apply to the outline color alongside everything

else. This may result in a soft, stylized look for your outlines. Otherwise, if you want the outlines to use exactly the color you specify, you should render them *after*.



In this example, the same red color is used in both effects, but the left image (Before URP Post Processing) is subject to color grading, whereas the right image (After URP Post Processing) is not.

LIGHTMODE SETTINGS

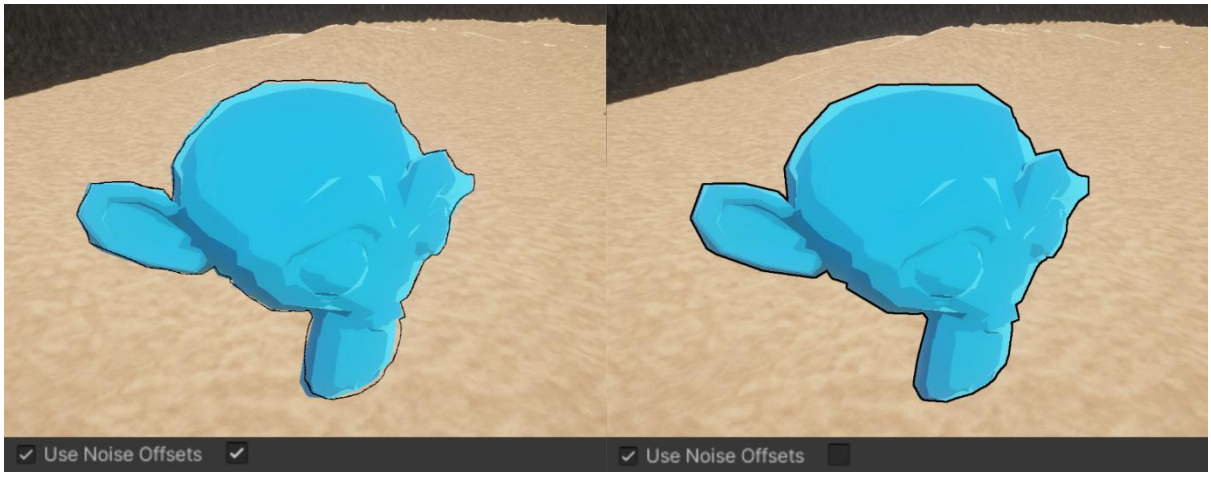
The LightMode setting restricts the choice of outline objects to only those which currently use a shader which uses a LightMode inside this list. Here's a non-exhaustive list of the kinds of shaders which use specific LightMode tags:

- Most Lit shaders, including the URP Lit effect, use the **UniversalForward** tag for their main shader pass, so include UniversalForward in the list to outline Lit objects.
- Most Unlit shaders don't use a tag for their main pass, and under the hood, Unity injects a tag called **SRPDefaultUnlit** into those passes. Include SRPDefaultUnlit to outline Unlit objects.
- Some shaders are special, such as hand-written shaders, so they may include the **UniversalForwardOnly** tag. The Toon shader in this pack is an example, so include the UniversalForwardOnly tag to outline Toon objects.
- Shader passes which support deferred rendering use the **UniversalGBuffer** tag.
- Any shader that supports shadow casting includes a **ShadowCaster** tag in one of its passes.

Using this setting won't necessarily mean that your object receives outlines – it still needs to match the other settings like Object Mask.

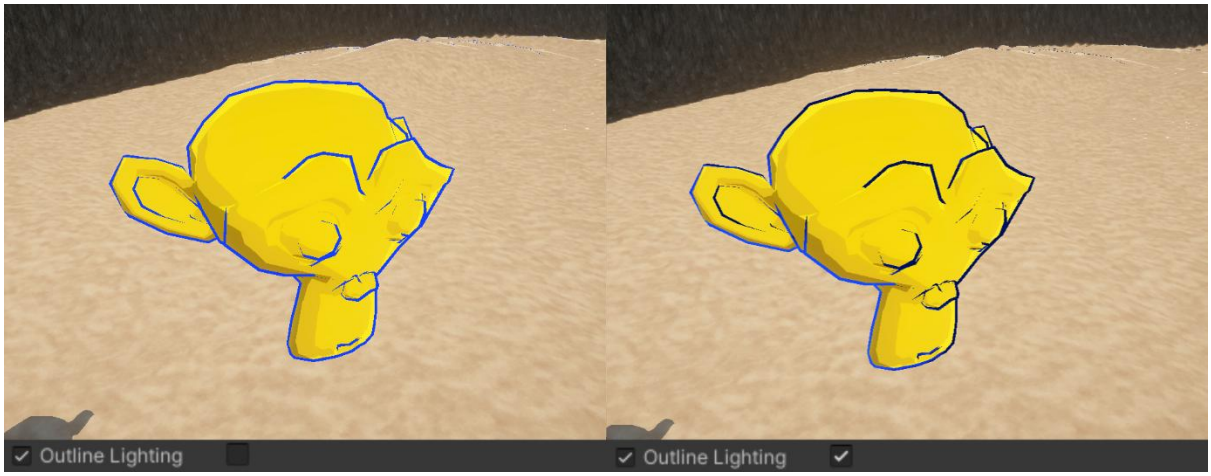
PERLIN NOISE

You can apply a noise offset to the outline UV calculations so that the outline appears like it was sketched onto the object.



HULL OUTLINE LIGHTING

Since the Hull Outline effect uses real object geometry, it's possible to easily calculate diffuse lighting from the main light and apply it to the outline itself for a stylized look. In the following example, the right-hand image uses outline shading, which makes the upper-right parts of the outline appear darkened.



SPECIAL THANKS

Many thanks to:

- ambientCG for many of the CC0 licensed textures used in the demo
- OpenGameArt for some CC0 licensed audio used in the demo
- Kenney for some smoke particle textures used in the demo
- Blender Foundation for the Suzanne monkey model used in the demo