Better Lit Shader

HDRP/URP Install

Sometimes HDRP or URP won't be detected on the first install and the shader will display as pink. If this happens, right click on the Better Lit Shader folder in Packages and select "Reimport".

Introduction

Thanks for purchasing the Better Lit Shader. The Better Lit shader is a replacement for the Standard/Lit shaders in URP, Standard, or HDRP pipelines. It aims to take the best of each of those shaders and standardize them, so features like tessellation, detail texturing, and layering are available regardless of which render pipeline you use. It also provides a lot of other features those shaders do not provide, such as Flat Shading, TriplanarUV's, Triplanar Texturing, Stochastic Sampling, multiple layer blending, and tighter texture packing modes for increased performance.

Unlike many other shader packages you might have worked with, there are only two shaders in Better Lit, the Lit shader, and the Unlit shader. All features can be combined using whatever combinations you want, without having to select specific shaders, only to find that the combination you want is not available.

Upgrading from 2019, 2020

In Unity 2021, the limit to how many shader keywords a shader can use was effectively removed. This has allowed for new optimizations and features, and some of those optimizations changed the way the material data is used. To automatically update your materials from 2019 or 2020, open the upgrading tool from the Window/Better Lit Shader/Upgrade Materials to 2021 and press "Do It".

Note that the Better Lit Shader is now delivered as a package, so you should delete any copy found in the Assets folder.

What's new in 2021

- **Wireframe** Rendering
- **Triplanar Texturing**
    - This allows you to project textures from each axis and blend them together. It's available on the main layer and all texture layers as well.
- **Shape Effector** System
    - This allows you to place shapes in the world which affect shader parameters, such as the weights of texture layers, dissolve effects, and wireframe rendering.
- **Mat Cap** lighting
    - Mat Cap lighting is a technique often seen in modeling software, such as zbrush, in which the object is lit by a special texture that describes the lighting. Better Lit can use matcap lighting on the whole object, or apply a different mat cap or traditional lighting to each texture layer.
- **Vertex Coloring**
- **Parallax Occlusion Mapping**
- **Subsurface Scattering**
- An **extra texture layer**
    - There are now 4 texture layers in addition to the main layer
- Texture Layer **Angle Filter** now supports any direction, as well as local or world space, and mirroring for the effect.
- **Noise Quality** and **Noise Space** are separately available in each use case instead of being global. So if you want to use a 3d noise for one texture layer, and a 2d worley noise for another, you can do that now.
- **6 sided gradient** projection for stylized scenes
- **Fresnel** Effects on all texture layers and snow
- **Sparkle** Effects on all texture layers and snow
- A new **Fast Metallic** option is available which packs a metallic map instead of an AO map

- **Dissolve** effect can now use noise functions as well as textures
- **Unlit** shader versions
- **Triplanar Blend Mode** allows you to blend between the vertex normal (smoothed) and face normal (hard) when choosing triplanar projections in world space.
- Better Lit Shader is now delivered as a **package**
- Some features moved from branches to shader keywords for better optimization
- **Baked Lit** Lighting model for URP
- **Geometric Specular AA** for HDRP
- **Coat Mask** support for URP/HDRP

Texture Formats

In the built in render pipeline's Standard Shader, Unity does not pack textures- you have a texture for height, another for ambient occlusion, etc. This is extremely wasteful, because you have to sample more textures and store more textures in Memory. Rather, it is much more efficient to pack data together, sampling fewer textures and saving memory. The default texture packing format is based on HDRP texture packing format.

Albedo Texture:

RGB = base color

A = height or alpha map

Normal Map

Mask Map:

R = Metallic

G = Occlusion

B = Detail Mask

A = Smoothness

This texture packing format provides a full metallic PBR specification in 3 textures. The same data in URP's Lit shader or the built in pipeline's Standard shader would take 5 textures, so this is a considerable improvement. Note that when using an Opaque shader, the Better Lit Shader stores height in the alpha channel for height based effects.

Alternatively, there is a Fast Packing option. When this option is used, textures are packed into an even tighter format, saving an extra texture, but at the loss of having a

Metallic map. Diffuse is the same, but the normal map now contains the smoothness and AO channels.

Packed Map:

R = Smoothness

G = Normal Y

B = Occlusion

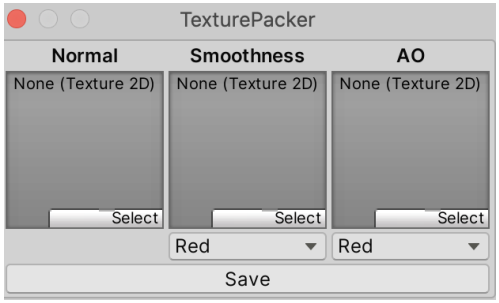
A = Normal X

When packing mode is set to fastest, all textures are expected to be in packed format. Make sure to turn off sRGB on a packed texture, as the data should be stored as linear- the UI will warn of this and provide a 'fix' button if you forget.

Packing textures this way saves both memory and GPU performance, at a slight loss of normal quality. In general, on organic surfaces such as rocks, this is not noticable, while on smoother surfaces like a car hood, banding might be present due to the decreased quality of gradient data.

Finally, if you'd prefer you can pack a metallic map instead of an AO map and set the shader packing mode to Fast Metal. An analytical AO value will be calculated based on the normal map instead, and you get a metallic map while still being in the fastest packing mode.

Texture Packer



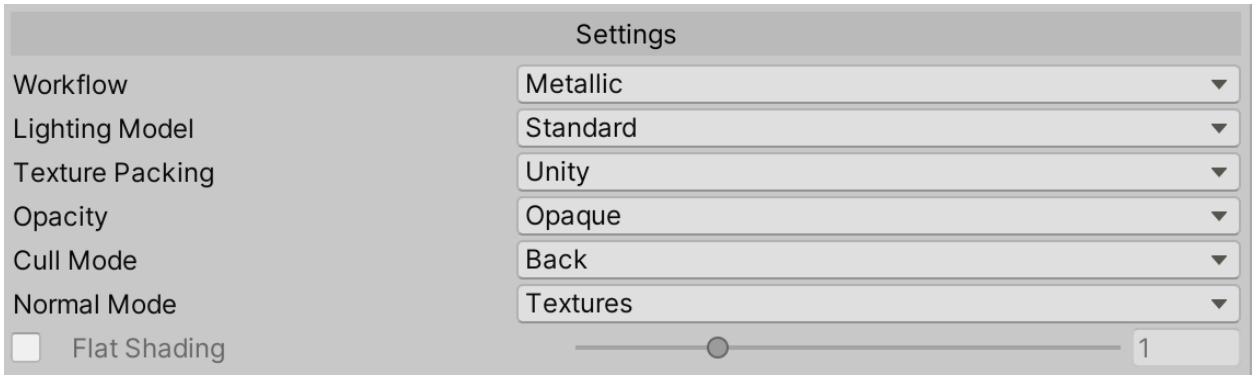
There's a small texture packing utility included to help with this task. You can select the packing operation from the Windows/Better Lit Shader menu. Each will contain a number of texture inputs, and in some cases a channel to pull the data from.

Shader Options

The shader supports a number of features and options, which can be used to produce much more complex effects than the existing shaders Unity provides. All sections have a rollout which can be clicked on to show or hide the options, and most have a checkbox to turn the feature on or off.

To texture a boulder, you might want a full scale normal map from your zbrush model, while still using triplanar texturing and stochastic sampling for the overall surface of the rock. You might also decide to project moss onto the top of this rock, and want it to get wet when it rains, or have snow accumulate on it when it snows. This would allow you to preserve the quality of your sculpted rock, take advantage of the tiled detail textures for the surfacing, and have each uniquely rotated rock vary in where it's moss grows, or how snow accumulates on it. All of these options are simple with the Better Lit Shader, and can automatically integrate with weathering systems like Enviro and Weather Maker.

Settings



This is the main configuration settings of the shader. The options are:

Workflow - Specular or Metallic workflows are supported. If you are unfamiliar with these workflows please consult the Unity documentation.

Lighting Model - When rendering in the Built In or URP rendering pipelines, a Simplified lighting model is available. A standard and unlit model are available in all pipelines.

Texture Packing - This is where you set the texture packing for the shader. All textures will be expected in the formats specified here.

Opacity - You can select between Opaque and Alpha shaders. Note that many effects such as tessellation and height blending use the alpha channel of the diffuse as a height map, however it is used for opacity when in Alpha mode.

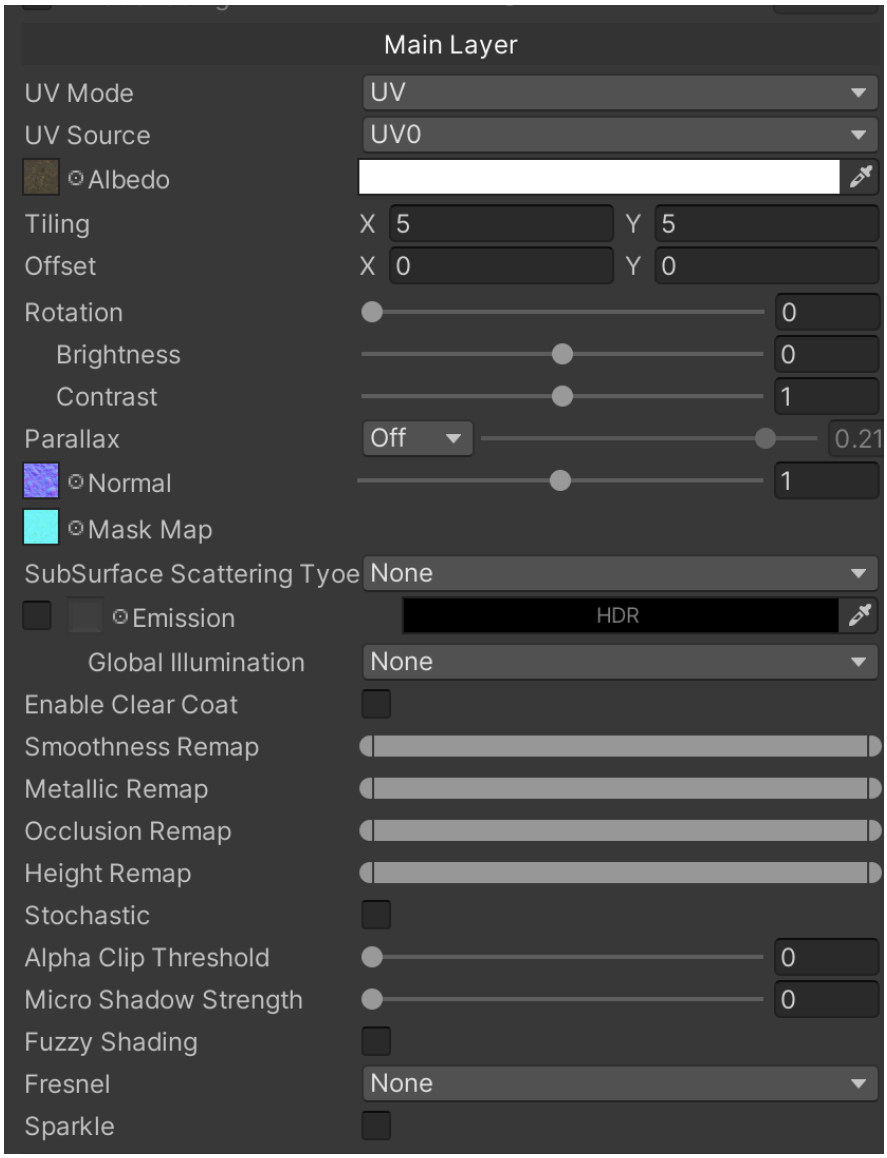
Cull Mode - This lets you set the cull mode of the object, drawing it as single sided, reversed, or double sided. When it's set to something other than Back, a control is available to determine how normals and lighting should be handled on the back side.

Normal Mode - By default this is set to texture, and you use tangent space normal maps for the shader. However, you can set it to use AutoNormal, which produces normals from the height maps stored in the alpha channel. This saves having to sample normal maps, which can be faster on low end systems. Note, however, that this is lesser quality than texture based normals, and can get very blocky when close up. Finally, there's an option to use the new Surface Gradient framework, which produces more accurate normals when heavily mixing lots of normal maps together.

Flat Shading - Flat shading allows you to interpolate between the smoothed vertex normals and the triangle face normals, giving you a flat polygon look.

The Main Layer

The main layer of the shader will be mostly familiar- it's similar to the lit or standard shaders, but with a lot of extra options.



UV Mode

There are three UV modes. The default is to use the UV's of the model to texture the object. When in UV mode, you can select between the first or second UV set of the model, or project the texture from any axis.

When set to Triplanar UVs, UV's will be generated from three projections in either local or world space, and the textures will be sampled and blended. This is a technique commonly used when UV's are not available, or when you want an equal UV distribution across a surface.

When set to Triplanar Texturing, each projection gets its own set of textures, allowing you to apply different textures from the front, side, and top projections.

When using world space Triplanar, the Projection Mode option is available to control which normal is used to choose the triplanar projection. In Flat Blend mode, you can blend between using the smoothed vertex normal and the face normal. Note that this can create discontinuities between the textures as it switches between projections on different faces. In Barycentric Blend mode, the barycentric coordinates are used to blend the projection across edges, such that the normal is the face normal across the surface, but the smoothed vertex normal along the edges of faces. Please note that you must process your mesh with the Mesh Converter to have barycentric coordinates baked into the mesh otherwise this mode will not work. For more information on this feature, and its use, read this link:

https://medium.com/@jasonbooth_86226/improved-triplanar-projections-b990a49637f9

Depending on the **Packing Mode**, it may show a normal map, or normal and mask map. All maps are optional, and if unassigned are not sampled.

The **Parallax** option lets you use either Parallax (Offset Mapping), or Parallax Occlusion mapping (POM). Parallax Offset Mapping is a cheap effect that increases the perceived depth of the texture. Parallax Occlusion Mapping is a more expensive effect which takes many samples of the albedo texture to create a much more convincing depth effect. When POM is enabled, additional properties to control the number of samples taken and how fast the effect should fade in the distance are available.

Note that POM is only performed on the main texture layer. Additional texture layers will conform to the POM surface. Note that POM does not work with world space mappings like UV projections or Triplanar, and any subsequent effect which uses these mappings will appear to float above the POM surface.

The **Stochastic** option will enable stochastic sampling on the texture, which completely eliminates all tiling. Stochastic has a scale and contrast settings, for controlling how the samples are blended and how large the stochastic patches are. The textures are height map blended based on the data in the Albedo's alpha channel, and there are sliders to control the contrast of the blend, and how large the texture patches are. Note that

performance is improved by having a sharper contrast (lower value), and larger scale areas, as this will allow the shader to sample fewer textures in most areas. (See the performance section for more)

The **apha clip** threshold will let you clip away parts of the shader who's alpha channel is below a certain value.

Clear Coat is available if you are running in HDRP or URP. When enabled, you can set the mask and smoothness amount or provide a texture which packs the mask and smoothness into the R and G channels respectively.

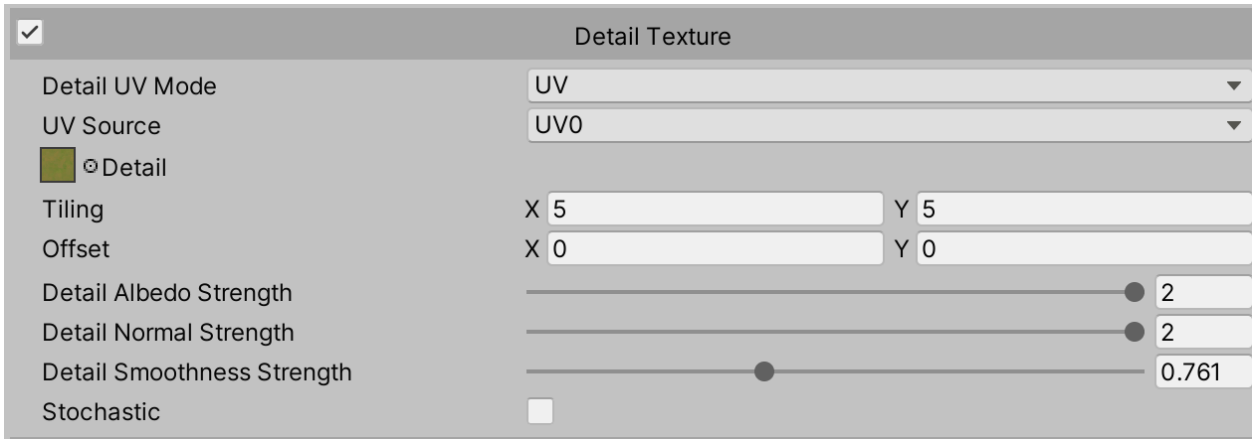
When **Sub Surface Scattering** is set to Approximate, a sub surface scattering effect is available which is compatible across all render pipelines. This effect only simulates subsurface scattering from the main directional light. Note that HDRP sub surface scattering is currently not supported, because Unity has made it impossible to have a diffusion profile set on a shader not created with a shader graph or their material system (the actual scriptable object class type is internal).

MicroShadows simulate small scale surface shadows.

Fuzzy shading adds a kind of fresnel color change useful for simulating velvet and moss style surfaces. It gives you a tint color, along with a strength multiplier for the inner and outer edge of the fresnel, and a power slider to control the falloff angle.

Fresnel allows you to add a rim light effect, as if the object is being back lit by a color. You can use the textured or vertex normal to compute the fresnel.

Sparkle adds a sparkle effect, useful for sparkly materials, snow and sand. You can control the tiling of the texture, and use the cutoff to control the density of the sparkles. The intensity controls how much of the effect is used, and the emission allows you to add the effect to the emission channels as well as albedo.



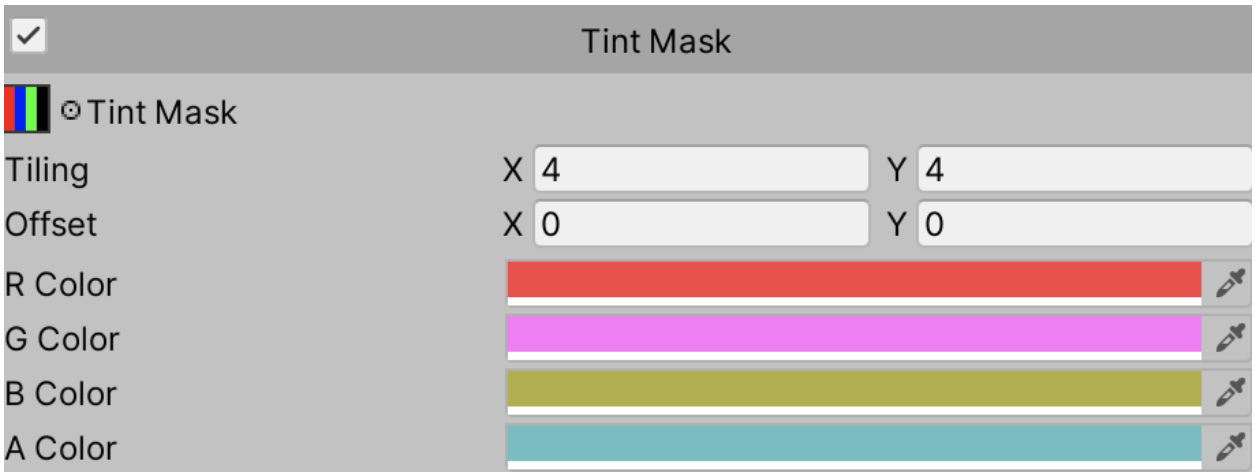
The **Detail Texturing** section of the shader is exactly like the detail texturing in the HDRP Lit shader, with the addition of things like triplanar texturing and stochastic sampling. The packing on the detail texture is exactly like the HDRP Lit shader, so I suggest reading those docs if you are unfamiliar. The Texture packing is as so:

R - Luminosity which modified diffuse (0.5 is no modification)
G - Normal Y
B - Smoothness
A - Normal X

This packing allows for decent detail texturing in just one texture.

Tint Mask

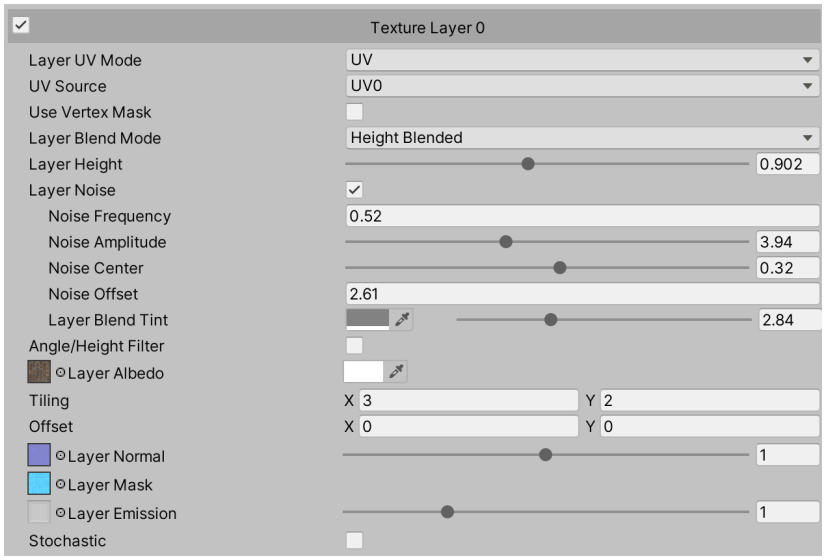
The tint mask is a way to selectively tint part of the main layer based on a mask texture. You can use 4 tint colors, corresponding to the value in the R, G, B, A channels of a texture. This can be useful for creating cheap variations using a single texture. For instance, you might paint the mask so that the metal areas are red, the leather green, and cloth blue, and anything else has the white in the alpha channel. Then, you can use the 4 color controls to tint each area separately.



Vertex Coloring

You can also apply the vertex colors to the albedo or alpha of the object. These values can be multiplied (tint), multiplied 2x (0.5 has no effect), or overlay blended with the color of the main texture layer.

Texture Layers



Texture layers are incredibly powerful additions, allowing you to blend multiple materials together. The shader supports up to 4 additional layers, which like the main layer, you can select between various UV modes such as using a specific UV, triplanar or world projection, and assign various textures as you would the main section.

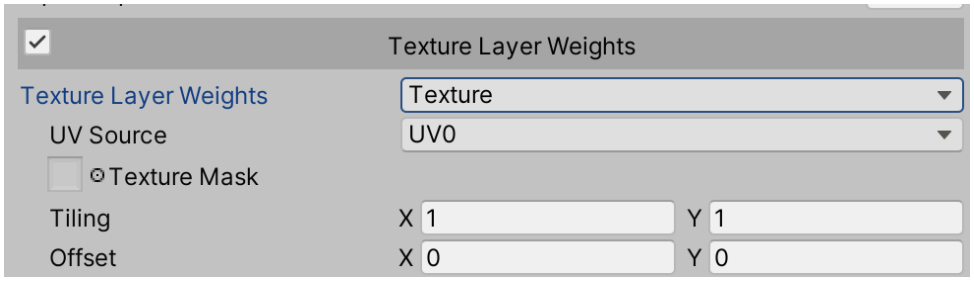
Layer Blend Mode allows you to control how the layer is blended with the previous. The options are:

- **Multiply2X**. This mode acts like a traditional detail texture, where a grey albedo value has no effect and lower or higher values darken or lighten the texture. The normal is also blended.
- **Alpha Blend.** This mode does traditional alpha blending between the layers.
- **Height Blend.** When height blending, the height map from the alpha channel of the albedo textures are used to resolve the blend, such that lower surfaces are covered before higher ones. This can give the physical effect of sand creeping between the rocks before covering them.

Once you have chosen how you want to blend your texture layer, you can use various techniques to determine where the texture layer will show through.

Filtering Texture Layers

There are several ways to determine where a texture layer has weight, and they can all be used together.



The first way is via the Texture Layer Weights. This can be set to vertex or texture mode. In Vertex Mode, the R channel of the vertex color will control the weight of the first texture layer, the G the second, and the B for the third, A for the fourth.

In Texture mode, a texture is used to perform this same function. The texture can be mapped to the UV coordinates, but can also be projected in world space. This is extremely powerful! For instance, you might use the Y projected mode to do a top down projection of a noise texture and size it to cover 100 meters in the world. Then, blend

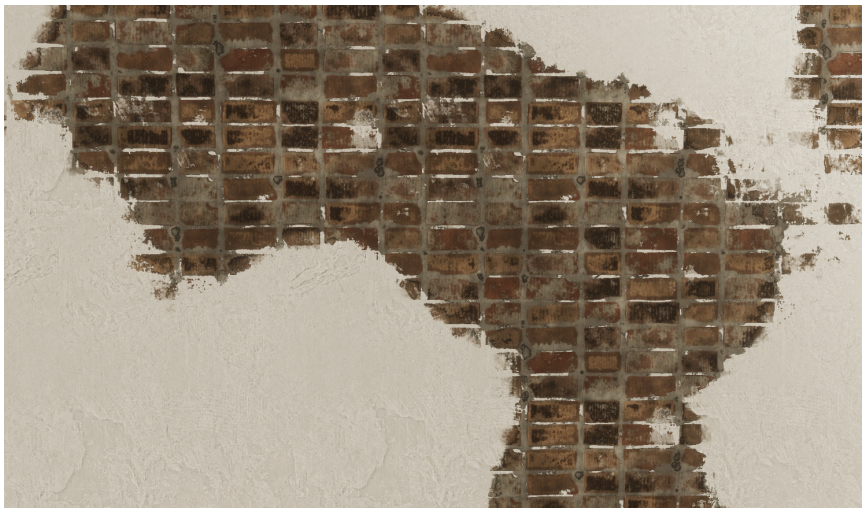
between several textures based on this map, such that every rock using the material gets a unique texturing over that 100 meter area. This would only add one texture sample for the weights.

The curve weights option can be used to tighten the blending area of the weights. This is useful when you want spline-like divisions between surfaces instead of alpha (blurry) or height map based blends. For a demonstration of this technique check out this MicroSplat video:

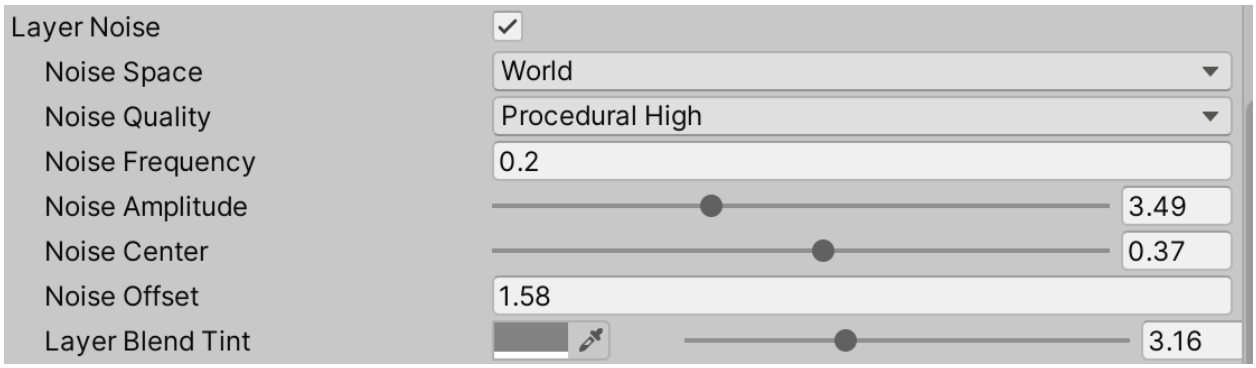
MicroSplat - how to make a golf course or road

The amount of curve to provide can be individually set per channel.

Noise



Each Layer has its own noise function, which lets you modify the weight of the texture layer with the type of noise defined in the settings. The example pictured above blends in a damaged texture on a wall based on this noise function, and it varies based on where the wall is placed in the level, causing every wall to have unique damage markings on it. Here, two textures are height blended based on a noise function, and the tint functionality is used to blacken bricks around the transition area. Anywhere this wall gets used in a level will have unique damage applied based on this noise function.



When Layer Noise is enabled, you have settings for its quality, space, frequency, amplitude, and offset.

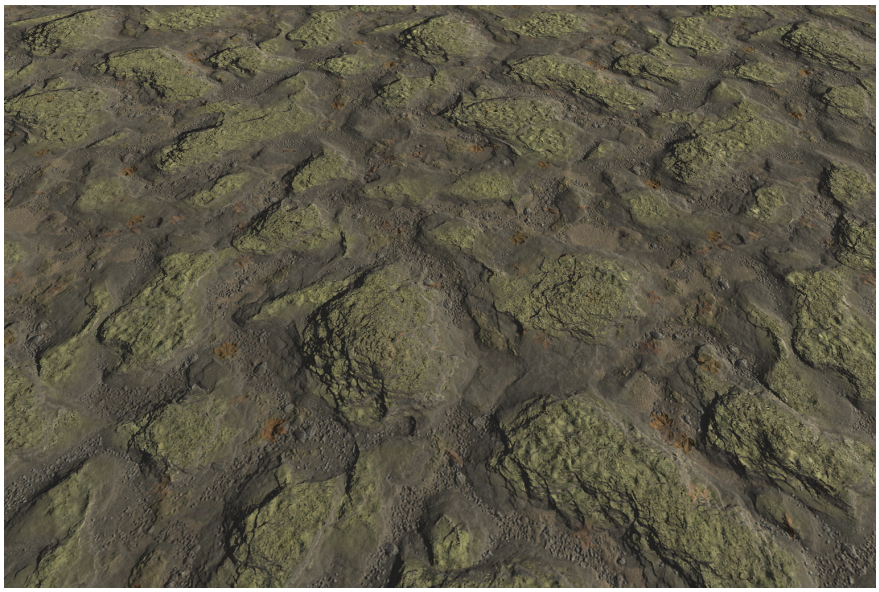
The Noise Space property controls the space of the noise's UVs. It can be computed based on the object's UVs, Local Space, or World Space coordinates. World space is particularly useful for static objects, so that every object placed gets a different noise treatment. Note that noise in UV space is computed in 2 dimensions and therefore cheaper than the 3d noise computed in Local or World space.

There are four qualities of noise available. Texture based noise can be the cheapest, and uses a noise texture you supply. The noise will get triplanar sampled when in local or world space, which makes it more expensive in those modes. You may also use Procedural noise in Low or High quality. In low quality, a single octave of Value noise is computed in 2d or 3d, while in High quality three octaves are computed and mixed in a FBM function. Finally there's worley noise, which produces a more cellular shape than the High quality noise, which is close to perlin.

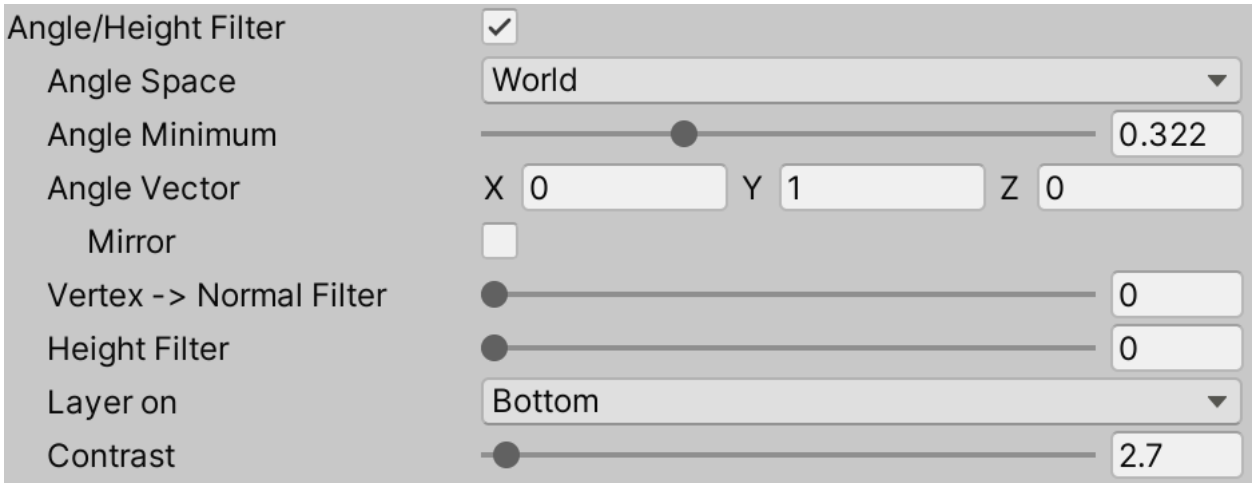
The noise center value can be used to boost or reduce the value of the noise, and is extremely useful when you want to use low amplitudes of noise for wide variations, as you'll want to use the noise center to control the overall amount of noise.

The Layer Blend Tint will apply a tint to the layer in areas where the noise is closest to 0.5 - in the middle of the transition area. In the example above, this causes many of the bricks to darken, providing a more damaged look and unique blend between the textures.

Angle Height filters



Here we see a moss layer appearing on the top of the rocks. This is being done by enabling the Angle/Height filter and using the alpha blend mode for the layer.



When this is enabled, you can filter where the texture layer appears based on the height field, vertex normal, and pixel normal of the previous layers.

Angle Space - Is the angle computed in world or local space

Layer Angle Minimum - This controls the filter based on slope

Angle Vector - This is the vector of comparison for the angle filter.

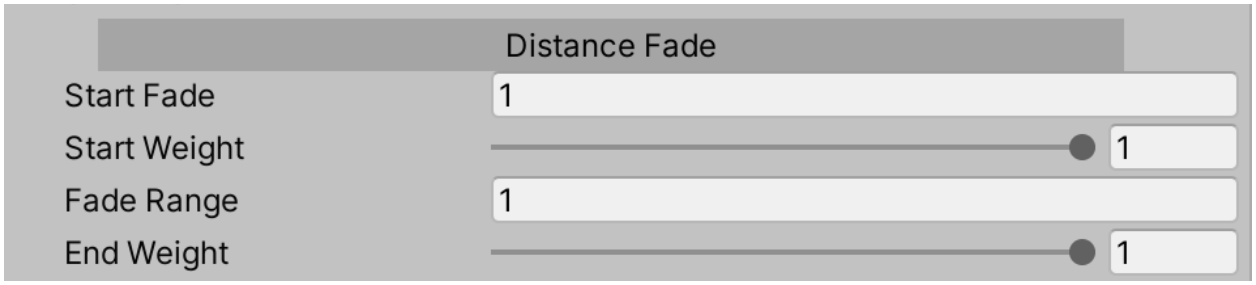
Mirror - Mirror the effect on the opposite side. For instance, if you use an angle filter of 1, 0, 0 then the texture will show on the right world space of the object. If you turn on mirror, it would be on the left side as well.

Layer Vertex -> Normal Filter - This controls how much the slope is based on the vertex normal vs. the per pixel normal from the layers below it. At 0, it will only use the smooth vertex normal, but at one it will use the final world normal of the layer below it, which will give a much finer detailed control over where the layer appears.

Layer Height Filter - This lets you filter the layers opacity by the height of the previous layers. In this example, we're filtering areas that are low in the height field.

Layer on - We can flip the filter to only show areas above or below our Height Filter value.

Layer Contrast - The layer contrast controls how quickly the filter interpolates between no and maximum weight.



The distance fade can be used to fade the layer in or out based on distance from the camera. This can be used for a number of effects such as Distance Resampling (AKA Mix Mapping), or bringing in global details at far distances.

Start Fade - This is the distance at which the Start Weight will begin cross fading into the End Weight

Start Weight - Weight at which the layer should be closer than the Start Fade distance

Fade Range - How many units to crossfade to the End Weight over

End Weight - the weight the layer should have beyond the fade range

Combining it all together

Note that you can combine all filtering methods on a surface - noise, weight maps, angle or height filtering, distance fades. Together with the blend modes and up to 4 layers of additional texturing, this provides a powerful tool set for procedural texturing and effects.

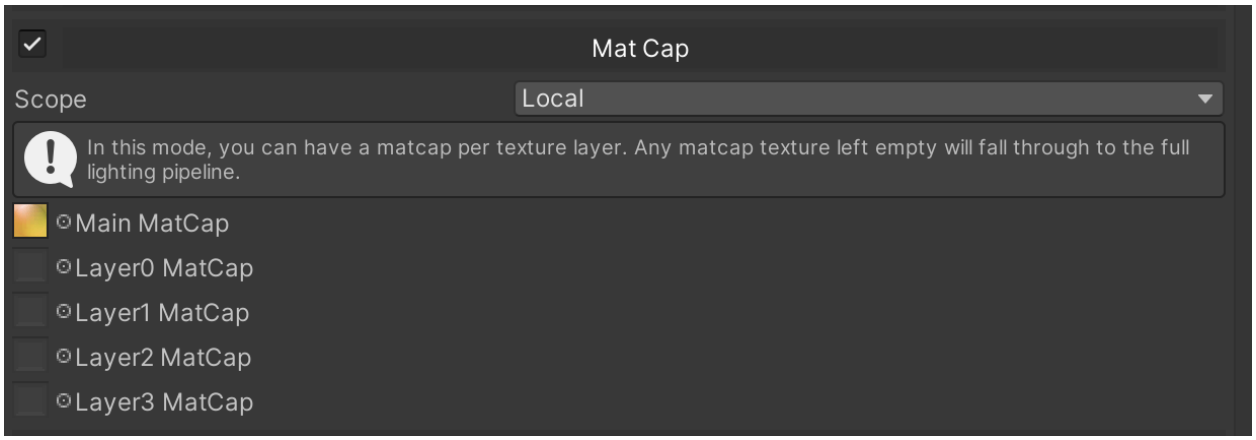
Stochastic

Like the main layer, each texture layer can be stochastic sampled to prevent tiling patterns.

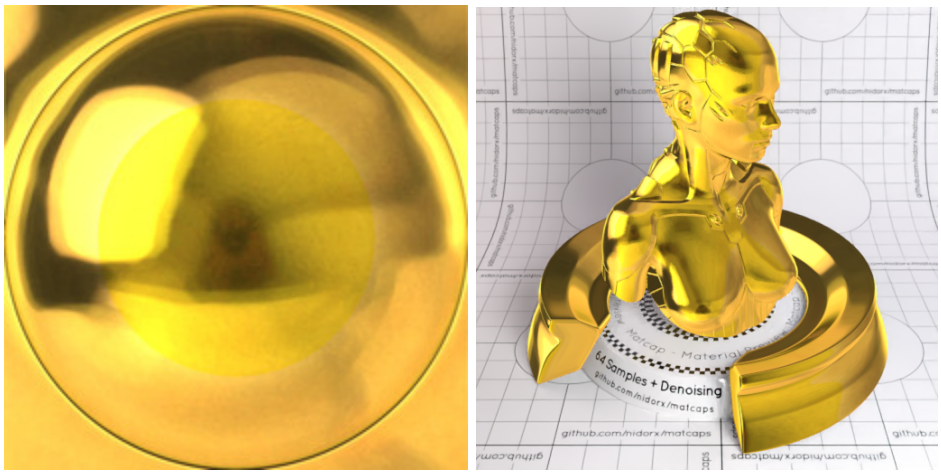
Displacement Amount

When Tessellation is enabled, each layer can also have a displacement amount. For instance, if you are height blending rocks and stone, you want both height fields displaced the same, creating an interactive surface as the sand comes up over the rocks. However, if you are putting some kind of procedural stains onto a wall, or moss onto rocks, you might not want that to adjust the displacement of the surface.

Mat Cap



A MatCap lighting model uses a special texture for lighting instead of a traditional lighting model. It is often used in modeling programs, like zbrush, and is quite a powerful technique for abstract or non-traditional lighting.



An example MatCap texture and the resulting lighting on a model.

Note that there are large libraries of free to use MatCap textures on the web, and changing the texture can radically change the look of the object.

The Better Lit Shader takes this idea even further, allowing for not only the traditional mat cap technique, but also applying one mat cap texture per texture layer, and allowing pass through for layers to use the Unity lighting model. Which options are available will depend on if your shader is set to Unlit one of the Lit lighting models.

When in Unlit mode, you can use the Single or Layered MatCap mode. In Single mode, a single matcap texture will be used for the entire object. In Layered mode, you will get 5 mat cap textures, one for the main texture layer, and one for each of the additional 4 texture layers. This lets you customize the lighting model for each texture layer. Note that this will enable normal maps on the Unlit shader, since a MatCap uses them to perturb the lighting lookups and create a more detailed surface.

When using a lit shader, only the Layered mode is available. If a texture layer is used and no Mat Cap texture is assigned, it will use the Unity lighting model for that layer. This lets you combine lighting models within a single surface, which can be very useful for special effects.

Finally, you can switch the matcap from local to global mode. In global mode, all textures are taken from global shader variables instead of the material. Set them with

Shader.SetGlobalTexture("_GMainMatcap", matCapTexture). The names for the layer properties are _Layer0Matcap, _Layer1Matcap, etc.

Effects

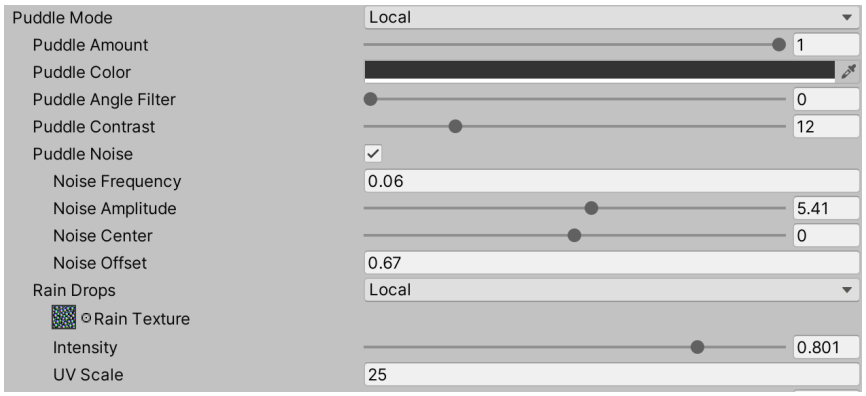
Wetness

Wetness mode can be set to off, local, or global. When set to global, no wetness amount slider is exposed. Rather, these are set via global shader properties from Eniro, Weather Maker, or your own code

```
Shader.SetGlobalVector("_Global_WetnessParams", new Vector2(wetnessMin, wetnessMax));
```

In local mode, everything is controlled via the material settings. The porosity control affects the reflectivity of the water. Consider how a rocky surface looks when wet- it gets quite dark, and reflective at the right angle, because small water droplets form little mirrors in the rough surface. However, a slick surface, like plastic, does not darken nearly as much because water rolls right off the surface. The min and max wetness can be used to clamp the minimum or maximum amount of wetness that the surface receives. Finally, the wetness falloff and angle let you filter the object so it doesn't get wet on the underside.

Puddles

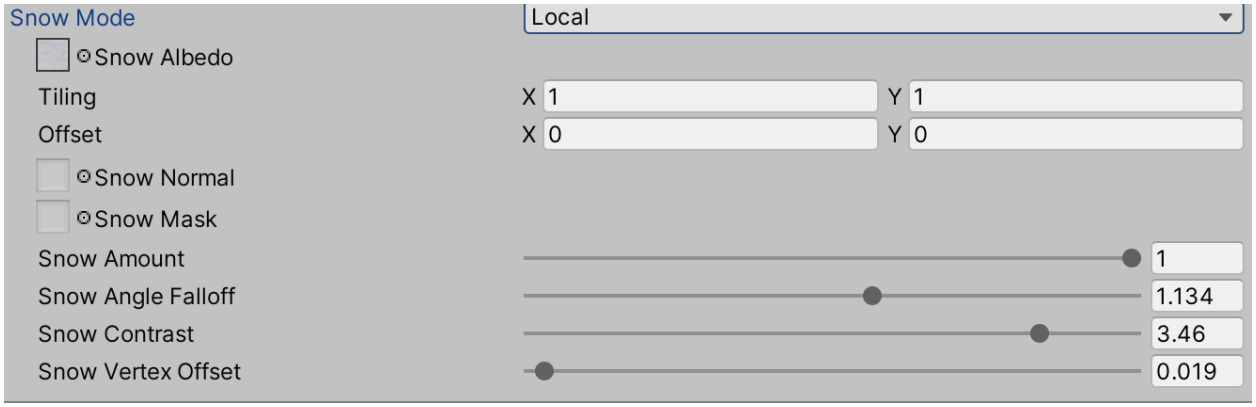


Puddle mode also offers local or global controls, allowing weather systems to modify the puddles via a shader global property:

```
Shader.SetGlobalVector("_Global_PuddleParams", new Vector2(amount, 0));
```

Puddles can be filtered based on angle, but also placed via noise functions.

Snow



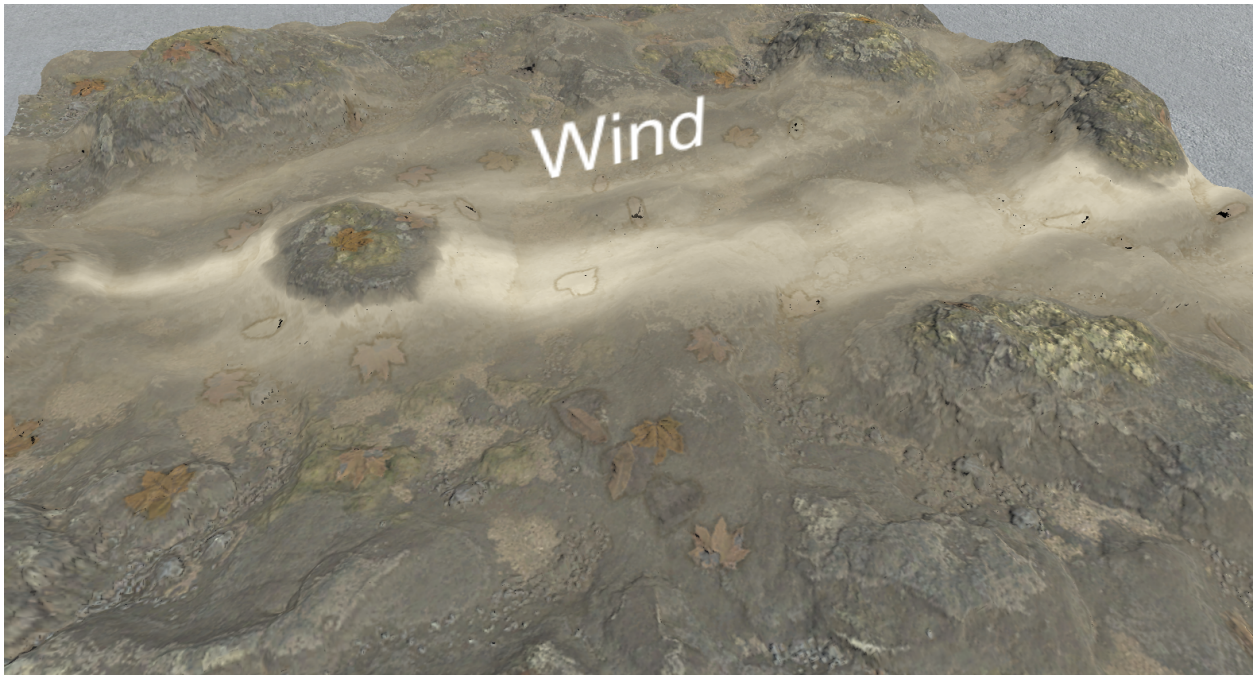
Like Wetness, snow is available in both local and global modes. When in the global mode, snow amount is controlled by Enviro or Weather maker, of via a global shader property:

```
Shader.SetGlobalFloat("_Global_SnowLevel", snowAmount);
```

Note if you are using the vertex or texture based masks, the alpha channel will get multiplied with the snow amount. Thus, painting 0 into the alpha will prevent snow from accumulating on an area.

Wind

The wind effect can create the illusion of particles blowing over and around your surface. It can interact with the height field, or be filtered based on world height or angle.



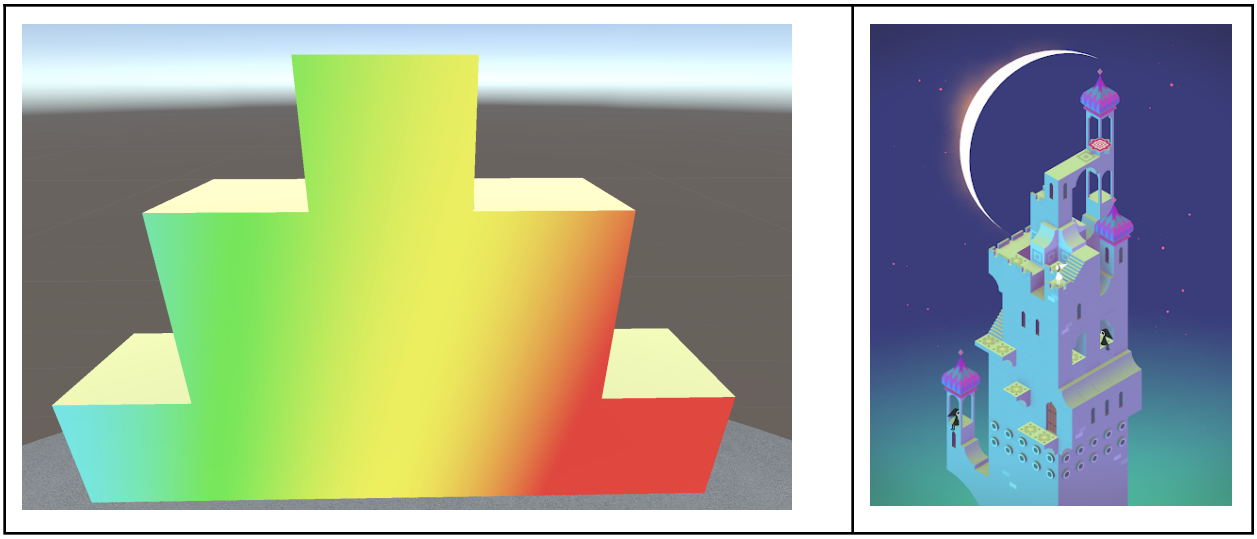
On this tessellated shader, we see the wind being filtered from the higher elevations of the height map.

Trax

The Trax system is sold separately, but an integration is included to allow you to make your shader compatible with Trax with a single click. Once enabled, any objects rendered into the trax buffer that collide with the surface will blend to a texture layer you provide. The texturing is projected top down in world space, along with the Trax Buffer. This also works with tessellation, allowing you to carve physical trails into surfaces.

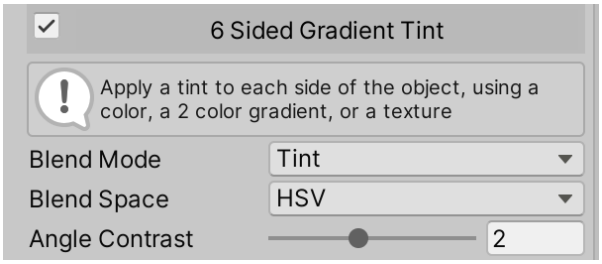
While Trax was originally designed as a MicroSplat plugin, this allows you to use Trax on any surface with the Better Lit shader, and the stackable can be used with Better Shaders to author your own shaders with Trax support. MicroSplat is not required to be used, or even installed, only the Trax module is needed.

6 Sided Gradient Tint



6 Sided Gradient Tint lets you project color, gradients, or texture data from each of the 6 word space axis's. This type of shading was made popular by games like Monument Valley (pictured on the right), and can be used in lue of lighting.

The version of this offered in better lit is very configurable, and is blended on the base shader, allowing it to act as a kind of post processing effect on existing shaders over the scene, a replacement for lighting, or fogging effects.



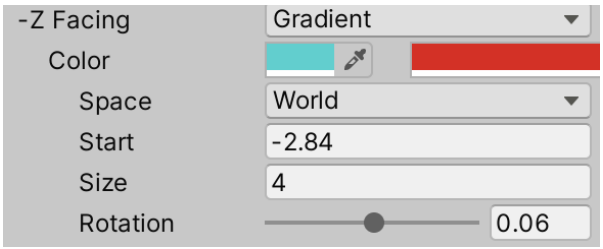
The effect can be blended as a Tint, Multiply2x or overlay onto the existing data.

The blend space can be in RGB or HSV color space.

Angle contrast controls how fast the projections transition from one to the other on a non-axis alignment.

Each axis has a set of controls for its projection. If set to none, no effect is applied on that axis and the regular shading is unaffected. Each projection is labels from its axis (X, Y, Z) and if it's the positive or negative facing of that axis.

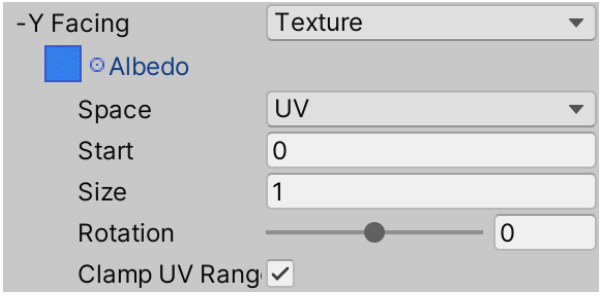
In color mode, a single color is used for that projection.



Gradient mode gives you a two color linear gradient. The gradient can be in UV space, local space, or world space of the object. You can set the overall size of the gradient in that space, and a start point to adjust where it starts. Finally, you can rotate the gradient effect around that axis.

In texture mode, the 2 color gradient is replaced by the texture, allowing you to specify complex gradients, or even project regular textures. The gradient should be laid out horizontally.

You can decide to clamp the texture, such that anything outside of the range gets the final values on the edge of the texture, or let it wrap, repeating the texture.



Understanding the space, start and size properties can take a moment. If you look at the example included and look at the -Z facing entry (pictured 2 above), you'll see that it is

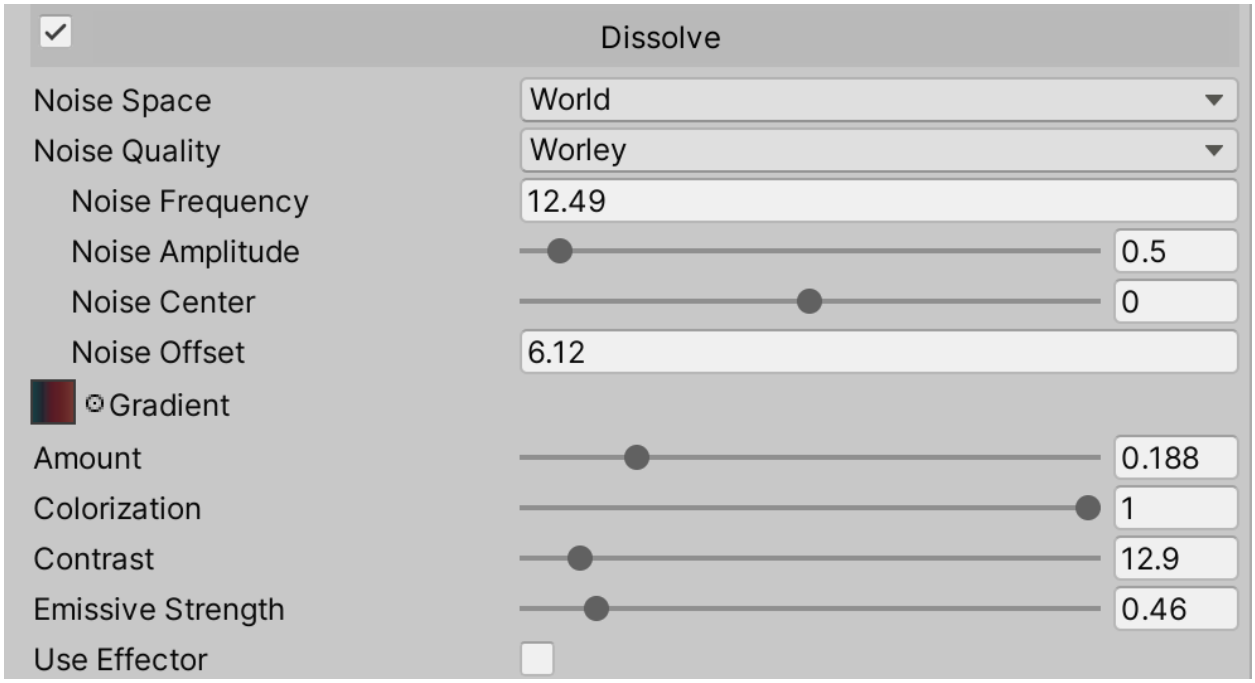
in world space and the Start value is -2.84 and the size is 4. This means the gradient will start at -2.84 units from 0 and extend 4 meters, to 1.84 meters on the X axis. If we adjust the start value, you will see the gradient move left or right along the X axis. If we adjust the range, the gradient will get larger or smaller. And if we move the objects from left to right, you'll see that the gradient remains stationary and the effect is adjusted on the object. You will also notice you can rotate the gradient, giving it a different direction from a simple left-right gradient.

When space is set to UV, these same controls work based on the UV values of the object. And in local space mode, they are based on the original positions of the vertices, and the effect moves with the object.

	A game like Monument Valley, on the left, used these types of effects for it's lighting and fogging to great effect. In this example a simple color is used for each side of the building, tinting whatever color of texture is below it to create the effect of lighting.

Dissolve

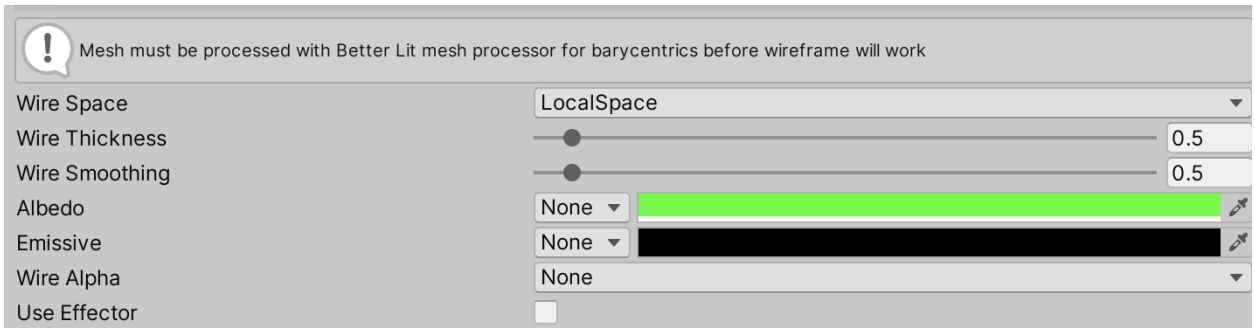
The dissolve effect lets you dissolve the object, complete with a colorized or emissive edge.



Dissolve uses a noise function to control where the dissolve happens on the object. It can be in world, local, or UV space, and based on a texture or 3 procedural functions. Lower values dissolve quicker than higher values.

The gradient controls the colorization along the edge of the dissolve. Animating the Dissolve Amount from 0 to 1 will cause the object to animate from solid to fully dissolved. The colorization controls how much of the gradient texture is used as part of the effect. The contrast controls the width of the edge, and the emissive strength controls how much of the color is added to the objects emissive output.

Wireframe

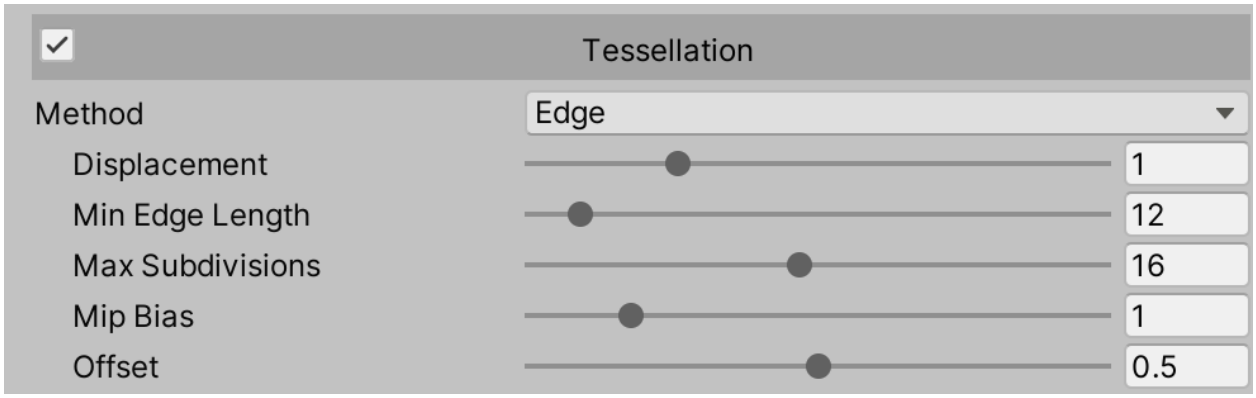


Wireframe rendering requires that you process the mesh to add barycentric coordinates using the Better Lit Shader's Mesh Processor, described below. Once processed, efficient wireframe rendering is available on all platforms, without use of Geometry Shaders, line rendering, or other inefficient techniques.

The wireframe size can be in local or world space. When in world space a consistent pixel size is maintained, whereas in local space it will scale up or down with the objects size on screen.

The thickness parameter controls the size of the line, with the smoothing softening the edge to provide anti-aliasing. The Wireframe can be drawn into the albedo or emissive color of the shader, using a linear interpolation blend mode or multiply 2x blend mode. You can also blend the wire into the alpha channel, or use cutout mode to make the object clip non-wireframe areas.

Tessellation



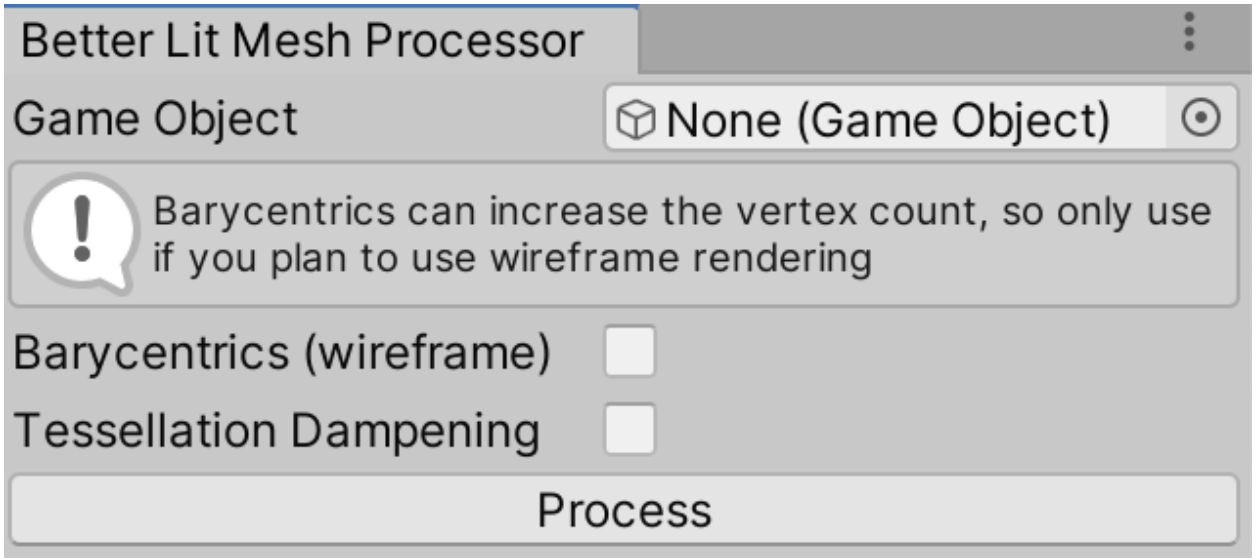
When Tessellation is used, the height map stored in the alpha channel of the albedo texture is used to determine how the surface is displaced. You can increase or decrease this effect with the displacement amount, or offset it up or down with the offset property. You can set the tessellation to be based on distance or edge length. When set to edge, the Min Edge Length controls how small triangle edges should be in pixels before it gets subdivided. In distance mode you get a distance at which the tessellation should be at it's maximum, and a distance after that in which it should not tessellate anymore. Max Subdivisions control the maximum number of times a triangle can be subdivided. The Mip Bias allows you to look into lower mip map levels for the displacement- this is not only faster, but often produces less jittery results than using the highest mip level, which might contain small details much smaller than the tessellation can represent.

In general, tessellation becomes very expensive when it creates lots of tiny triangles-

triangles smaller than about 10 pixels in size become exponentially more expensive for a GPU to draw, so tessellation or not it's best to keep triangle size above that.

Note that tessellation will pull a mesh apart if the normals and vertices are not merged. This can be mitigated by dampening the tessellation in areas where vertex data is not shared. A preprocessor is available to process meshes, and store dampening data in the z channel if the UV coordinates.

Better Lit Mesh Processor



To process a mesh, open the Window/BetterLitShader/Mesh Processor window. Drag a game object from the project window into the game object setting, select which data you want to burn into the mesh, and press process. This will save a new asset next to your original game object with the extension _betterlit, and any meshes found in that game object inside of it. You will need to replace the meshes in the scene or on prefabs with the new meshes.

Barycentrics are used for wireframe rendering, and should not be enabled if you don't plan to use it. The barycentric process can add extra vertices.

When tessellation dampening is enabled, any area of the mesh which would get ripped apart gets a dampening factor added to it to prevent this. This would cause a simple box to be undisplaced, but for organic surfaces this works pretty well to prevent tessellation cracks.

Bakery

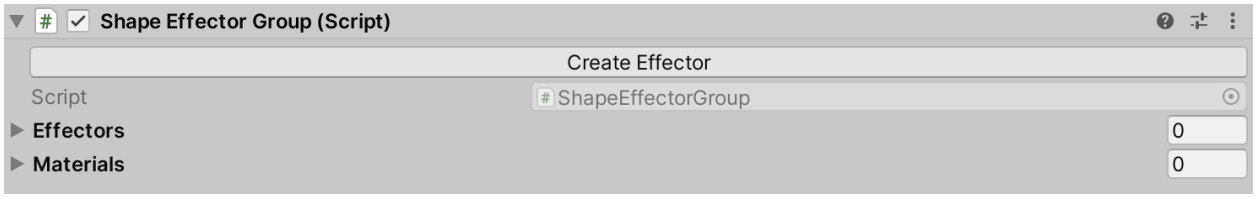
The Better Lit shader has full support for Bakery, a GPU lightmapping tool for unity with increased lightmapping speed and quality. Consult the Bakery documentation for

more information, and direct any questions about Bakery to their support channels as I did not write the code.

Shape Effectors

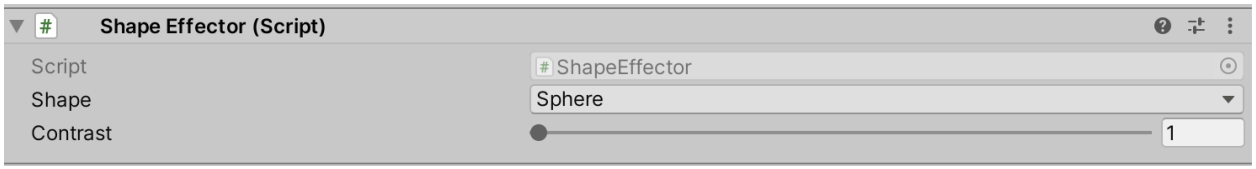
Shape effectors are a powerful system that allows you to adjust various parameters of the shader based on the location of shapes in the scene. For instance, you can have a sphere that causes the shader to turn into a wireframe rendering in the sphere's radius, dissolve away, or modify the weight of a texture layer.

To start with the shape effector system, create a shape effector group from the Windows/Better Lit Shader/Create Shape Effector Group.



You can easily create effectors with the create effector button. Each group can have up to 4 effectors. Add any materials you want to be affected by the group to the materials list.

Once you create a effector, you can select it from the hierarchy, under the newly created group.

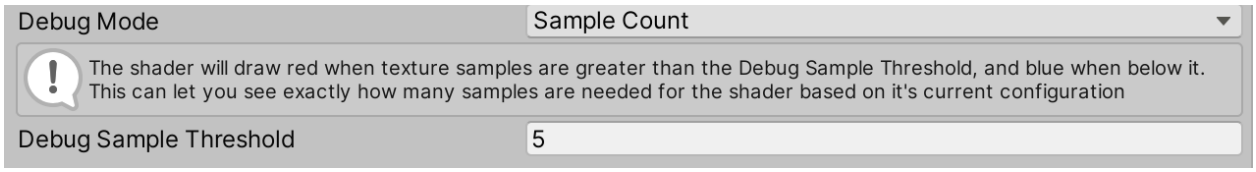


You can change the shape from sphere to plane. In sphere mode, the effect happens when the sphere intersects any objects with the material's specified in the groups material list. When set to plane mode, the effect happens on one side of the plane and not the other. You can control how soft the falloff is with the contrast setting. You can scale and position the sphere, position or rotate the plane.

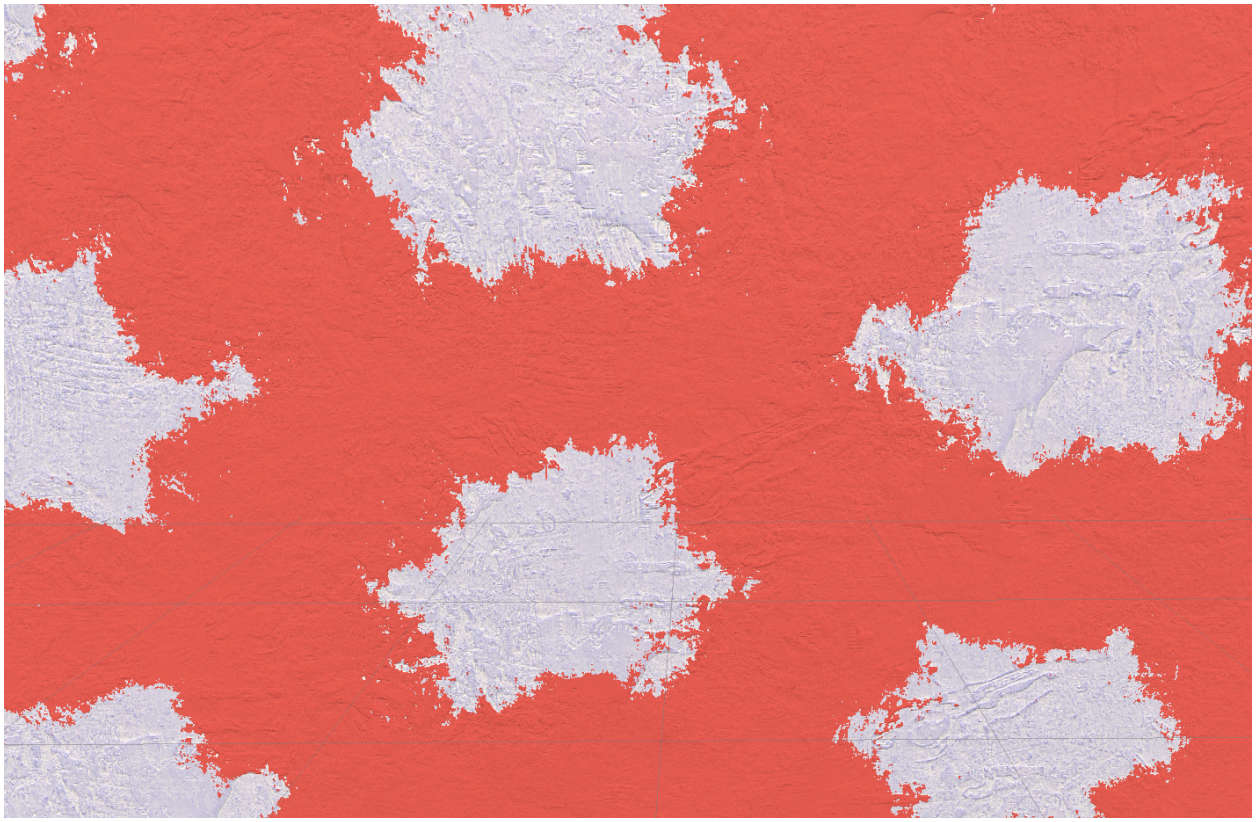
To control what the effector affects, turn on one of the use effector options in the material. (They are in many places, for instance, on each texture layer, on puddles, snow, wetness, dissolve, etc). For instance:

Performance

The performance of the shader is highly dependent on which features are active. Enabling triplanar or stochastic are the two most expensive effects, as they require sampling each texture up to 3 times. Note that you can interactively visualize how many samples the shader is actually taking.



At the bottom of the interface you can find the debug mode. When this is set to Sample Count, the shader will tint the object based on how many samples are being taken. When the shader is taking more samples than the Debug Sample Threshold, it will draw with a red tint. If it's under the Debug Sample Threshold, it will draw with a blue tint. This lets you not only see exactly how many samples are taken, but also visualize where certain optimizations are happening.



Here we see a wall with Stochastic Sampling enabled. In the red areas the shader is doing 6 or more samples per pixel, while in the blue area it is doing 5 or less. As you adjust the stochastic scale and contrast settings, you can see this adjust based on the current culling.

Note that currently, only triplanar, stochastic, and snow areas are culled when not in use. This is because these tend to have wide areas of savings, making the dynamic flow control faster than sampling the textures.

Modifying the shaders

The Better Lit Shader 2021 was written using Better Shaders 2021. Better Shaders allows you to write complex shaders in a very simple manner, then use them in any render pipeline. It also lets you stack or inherit various effects into existing shaders, increasing the modularity of writing shaders. For instance, the wetness, moss, and snow effects included were just stacked on top of the base shader, and can be easily applied to any Better Shader's shader, without even needing to modify source code. Further, the texture layers are the same shader stacked on top of the lit shader three times.

Additionally, code written in better shaders is a fraction the size of a vertex/fragment shader written by hand, and would have to be almost entirely rewritten to run on another render pipeline, and would also need to be rewritten for different versions of URP/HDRP as they often require a large rewrite to shaders for compatibility. Better Shaders not only handles all of this for you, but also includes a packaging system so you can install the resulting shaders on a system and have it just work, regardless of which render pipeline they have installed. This package is using that system to deliver this set of shaders to you. (Internally, as of this writing, each shader is compiled for URP2021, Standard, and HDRP2021, which is about 90,000 lines of shader code, produced from under 2000 lines of Better Shaders code).

Source code for these shaders is provided in Better Shaders format, because editing them any other way would just be crazy.

If you install Better Shaders and wish to modify the source, realize that the shaders in the main shader directory that you have been using are packed versions for all platforms. This adds a few more steps in the process because it allows you to pack these shaders for use without Better Shaders installed. To repack the shaders, you must do the following:

- Put the Better Lit Shader into development mode. You can do this from the Windows/Better Lit Shader/Development Mode menu items.
- After making any change to the source code shaders, they will be compiled automatically - but the packed shaders in the Shaders directory must be packed manually (this compiles all the versions into one file). To do this, select the shader and press the "Pack" or "Pack all in project" button.

A note about the source

The code contained in these shaders represents pretty cutting edge use of Better Shaders, as this system was originally written to 'dogfood' the Better Shaders system. It features a custom GUI for each shader stack, a traditional custom material editor for the resulting exported shaders (sharing the same code), and uses the packaging system to deliver this as a single shader which runs on multiple incompatible rendering engines in Unity.

I have heavily documented the source so that it's not only easy to understand, but acts as a guide for people who want to learn to write shaders in the Better Shaders system. There are quite a few tricks using stackables that can allow you to write significantly less code - for instance, the Texture Layers are all the same code stacked 4 times.

Most of the stackables can be used in your own code without any modification, though there is a define and a few pieces of code which you will need to add if you want to support tessellation (this is all in the LitTessellationBase-Dev surfshader file.

The custom editor code is mostly in EditorStub.cs, and called from the custom material editor, or called from the individual material editors assigned to each stackable. See the Better Shaders documentation for how this works.