Better Lit Shader

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, Triplanar, Stochastic Sampling, multiple layer blending, and tighter texture packing modes for increased performance.

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.

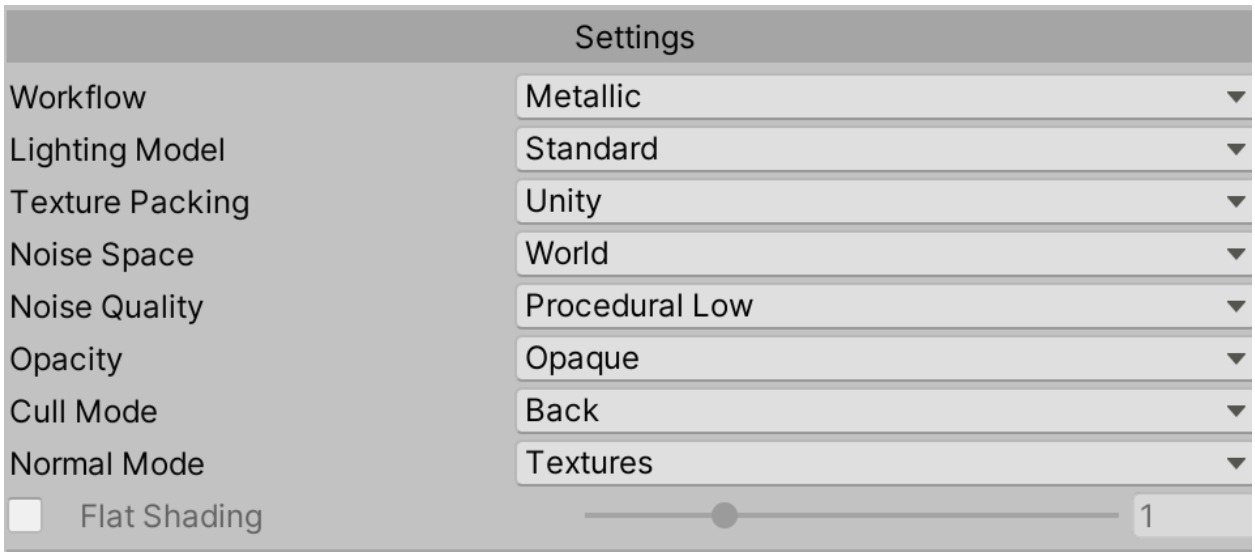
Shader Options

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. This setting is ignored in HDRP.

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

Noise Space - This selects how noise is computed for all noises used in the system. 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 therefor cheaper than the 3d noise computed in Local or World space.

Noise Quality - There are four noise qualities available. Texture based noise can be the cheapest, and uses a noise texture you supply. The noise will get triplanar sampled when 3d noise is needed. 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.

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.

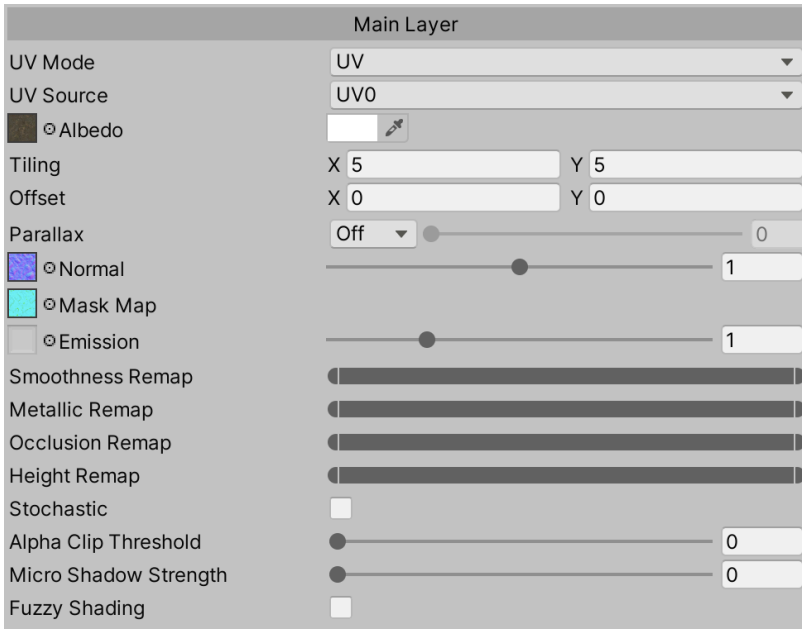
Cull Mode - This lets you set the cull mode of the object, drawing it as single sided, reversed, or double sided. When it's set to something other than Back, a control is available to determine how normals and lighting should be handled on the back side.

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 for the most part.



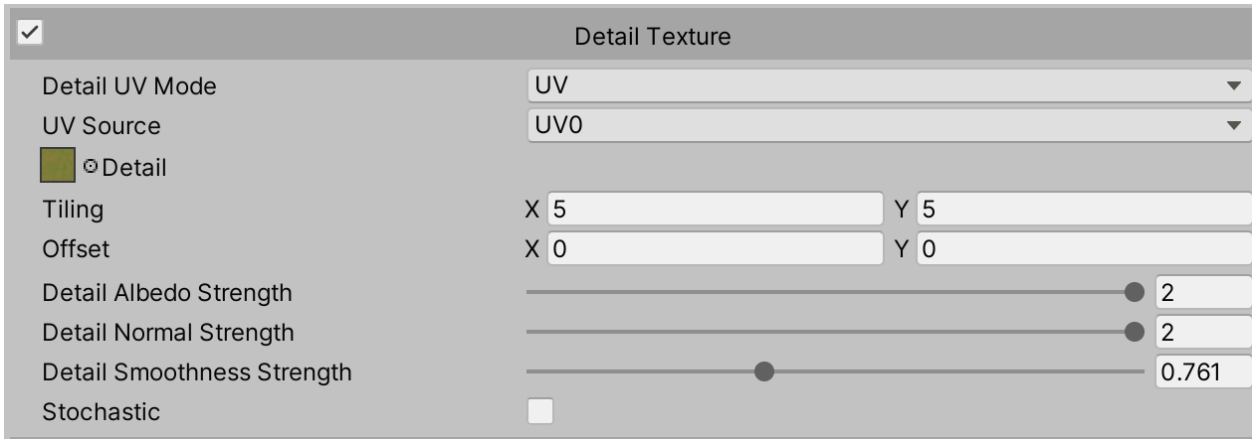
UV Mode - You can select between UV or Triplanar texturing. When Triplanar is active, you can select between local and world space, and adjust the contrast of the triplanar texturing. When UV Mode is set to UV, you can select between UV0, UV1, or a projection on any of the axis's in world space.

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 **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.

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.



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:

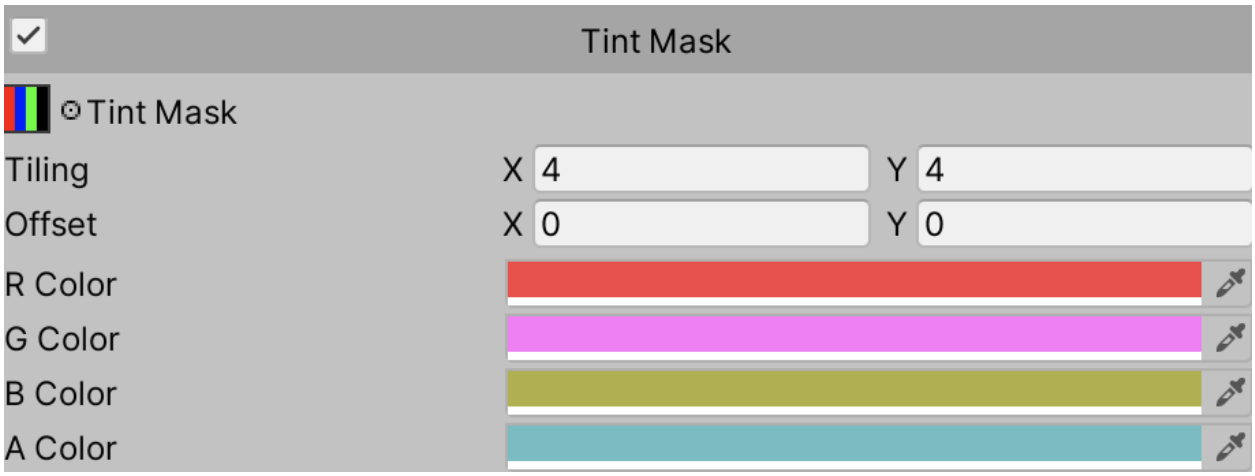
R - Luminosity which modified diffuse (0.5 is no modification)
G - Normal Y
B - Smoothness
A - Normal X

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

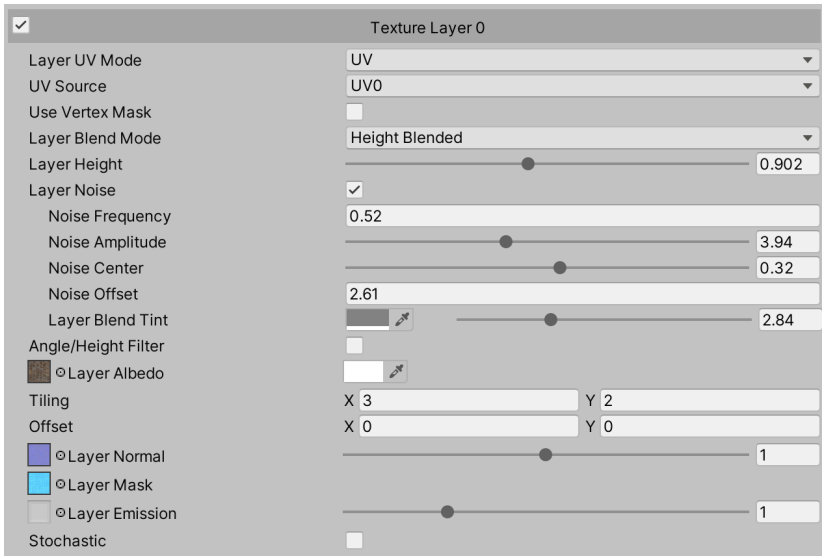
Tint Mask

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.



Texture Layers



Texture layers are incredibly powerful additions, allowing you to blend multiple materials together. The shader supports up to 3 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.

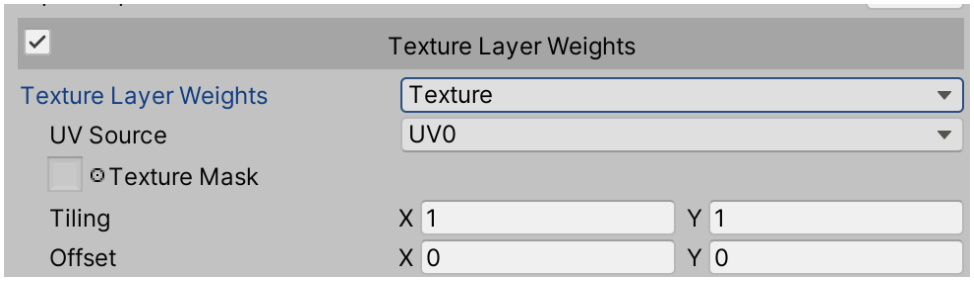
Layer Blend Mode allows you to control how the layer is blended with the previous. The options are:

- **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

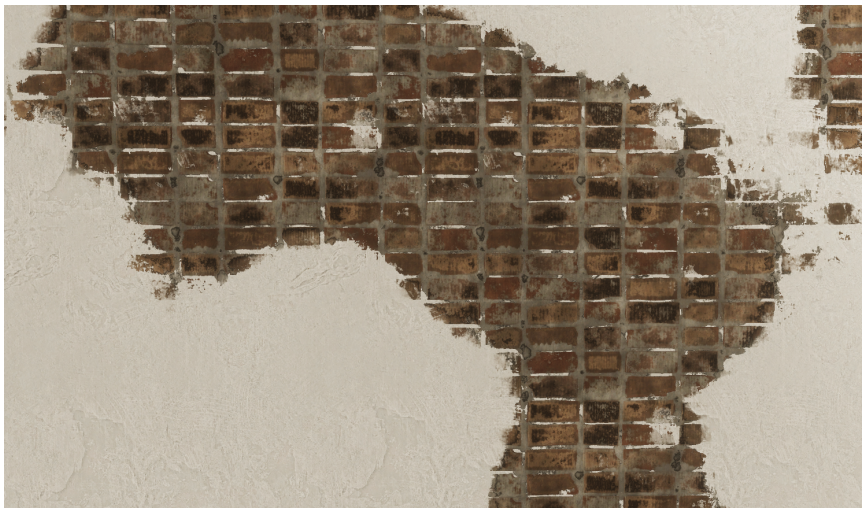
There are several ways to determine where a texture layer has weight, and they can all be used together.



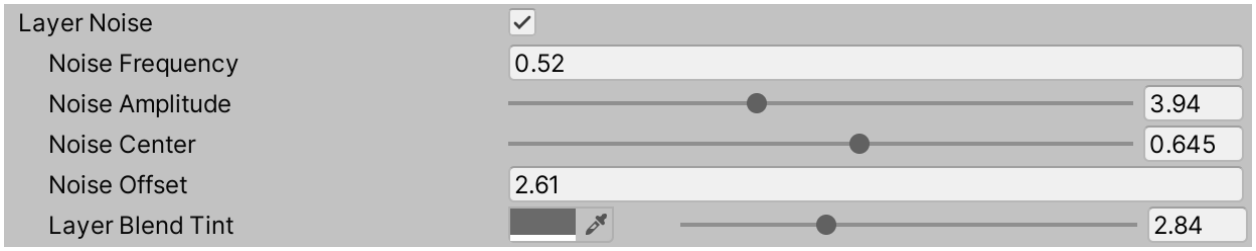
The first way is via the Texture Layer Weights. This can be set to vertex or texture mode. In Vertex Mode, the R channel of the vertex color will control the weight of the first texture layer, the G the second, and the B for the third.

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.

Noise



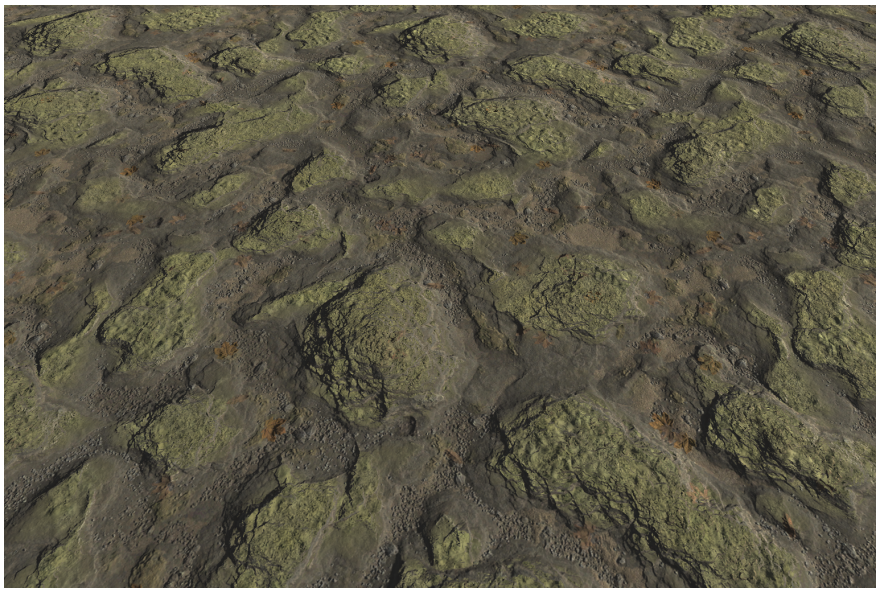
Each Layer has its own noise function, which lets you modify this weight of the texture layer with the type of noise defined in the settings. The example pictured above blends in a damaged texture on a wall based on this noise function, and it varies based on where the wall is placed in the level, causing 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.



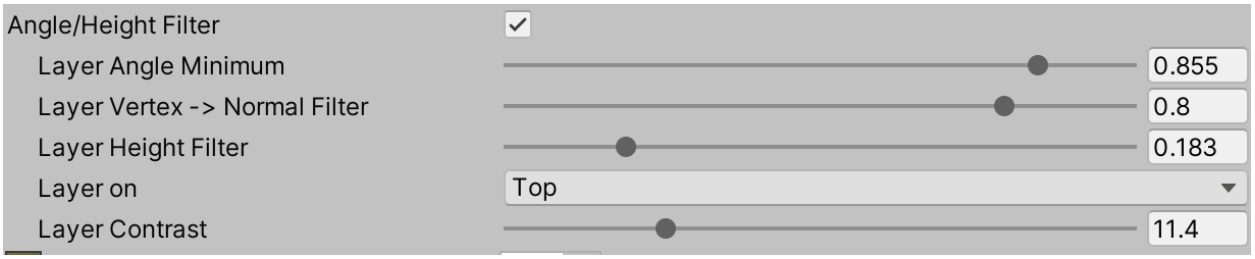
When Layer Noise is enabled, you have settings for its frequency, amplitude, and offset. The noise center value can be used to boost or reduce the value of the noise, and is extremely useful when you want to use low amplitudes of noise for wide variations, as you'll want to use the noise center to control 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.

Layer Angle Minimum - This controls the filter based on slope

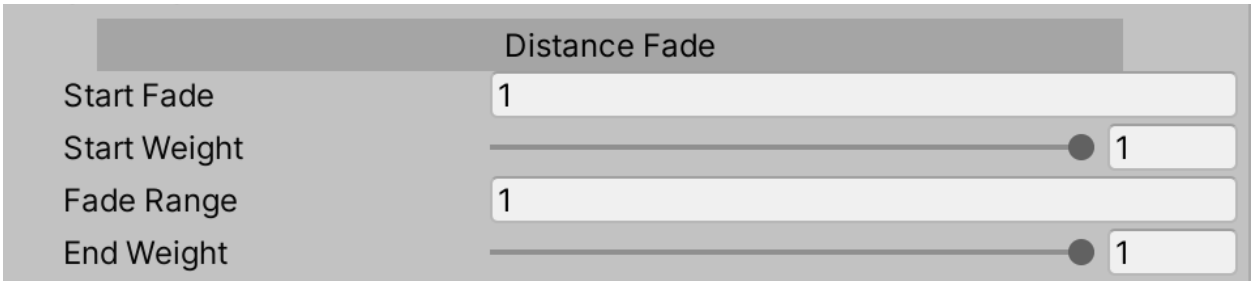
Layer Vertex -> Normal Filter - This controls how much the slope is based on the vertex normal vs. the per pixel normal from the layers below it. At 0, it will only use the smooth

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 3 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.

Effects

Wetness

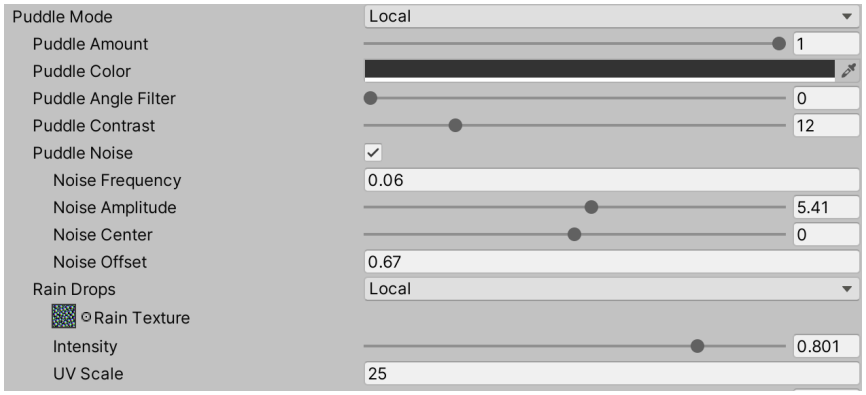
Wetness mode can be set to off, local, or global. When set to global, no wetness amount slider is exposed. Rather, these are set via global shader properties from Eniro, Weather Maker, or your own code

```
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:

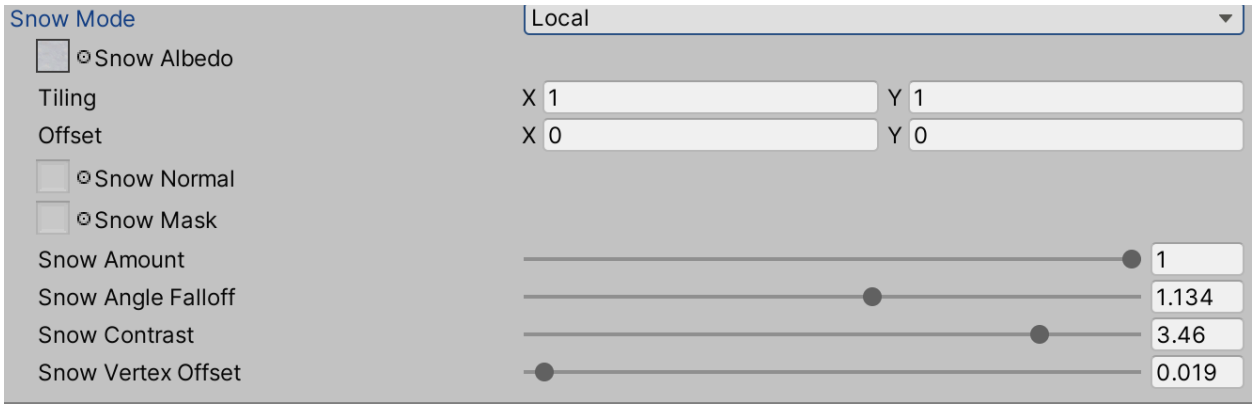
```
Shader.SetGlobalVector("_Global_PuddleParams", new Vector2(amount, 0));
```

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

Rain drops can also be enabled, and also respond to a shader global when set to global.

```
Shader.SetGlobalVector("_RainIntensityScale", new Vector2(intensity,
size));
```

Snow

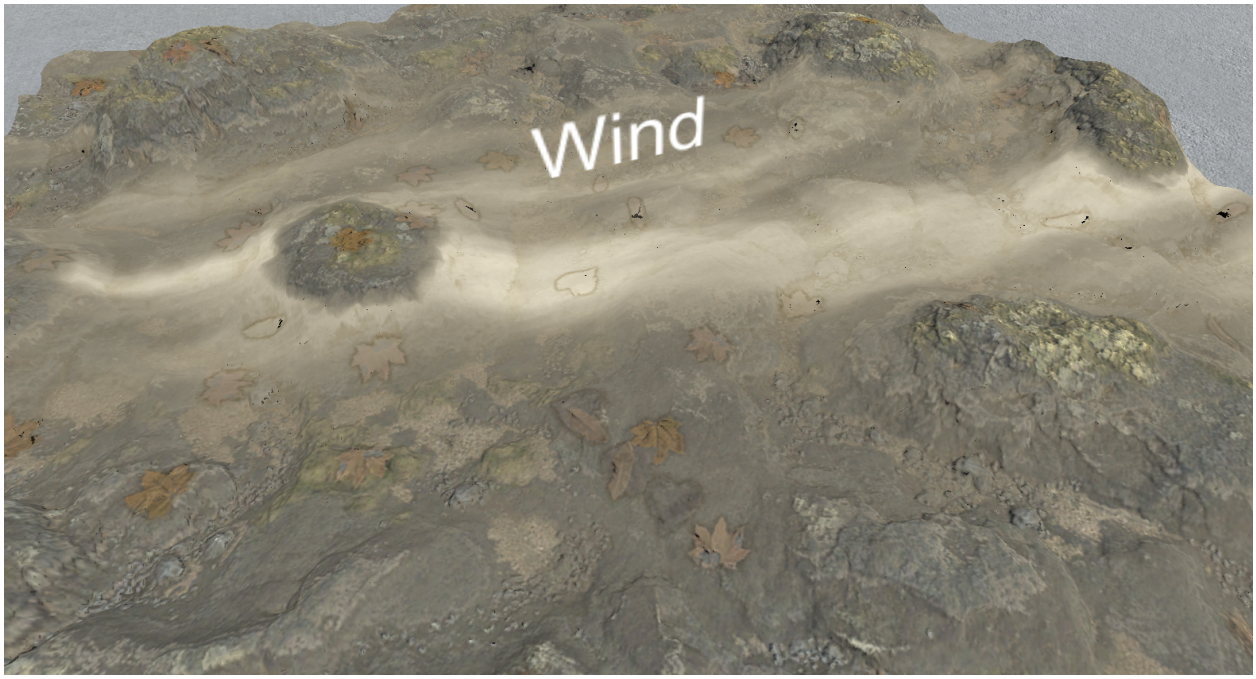


Like Wetness, snow is available in both local and global modes. When in the global mode, snow amount is controlled by Enviro or Weather maker, of via a global shader property:

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

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.

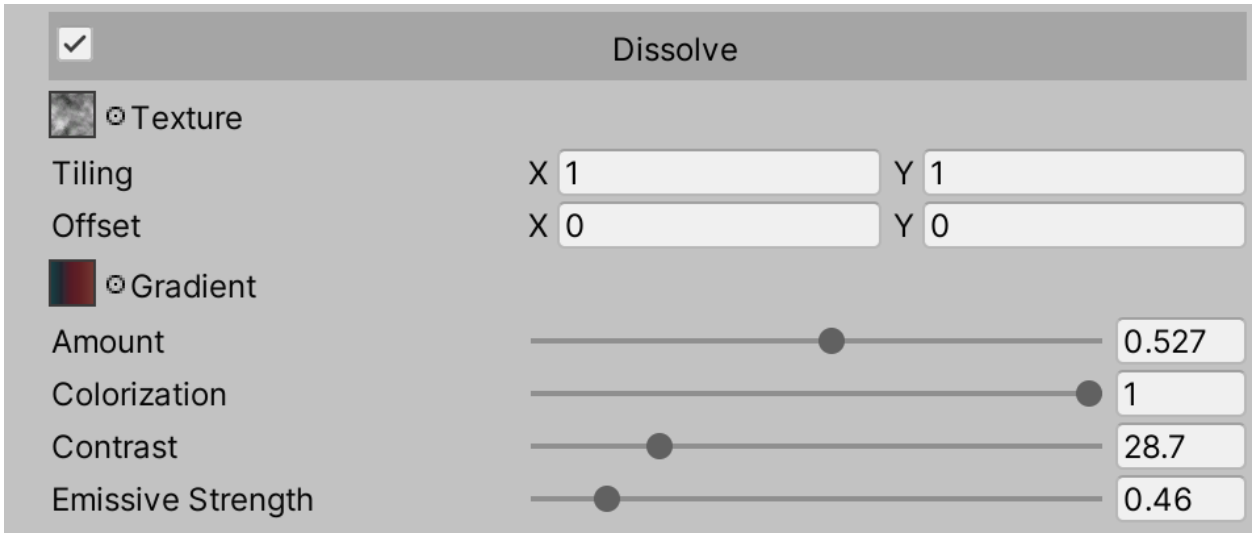
Trax

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.

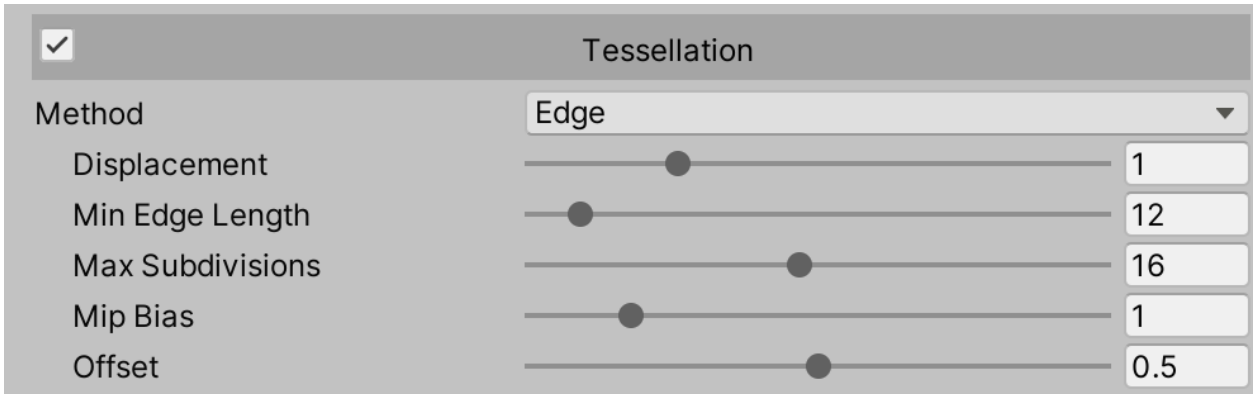
Dissolve

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



A dissolve texture provides noise for how the dissolve should occur; lower values will dissolve before 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.

Tessellation

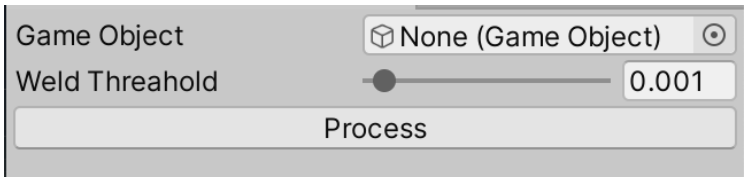


When Tessellation is used, the height map stored in the alpha channel of the albedo texture is used to determine how the surface is displaced. You can increase or decrease this effect with the displacement amount, or offset it up or down with the offset property. You can set the tessellation to be based 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.



To process a mesh, open the Window/BetterLitShader/Crack Free Tessellation Mesh Processor window. Drag a game object from the project window into the game object setting, and press process. This will save a new asset next to your original game object with the extension _crackfree, 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.

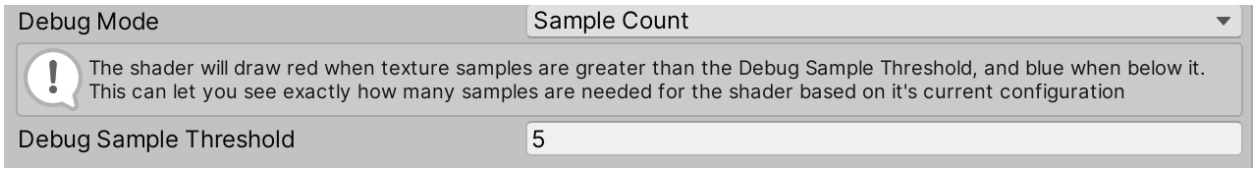
Note that this merely dampens the displacement on those areas, which may or may not suit your particular use case. For instance, this would not work well on an unsubdivided box, since there are no vertices between the unshared vertices to displace. But for organic shapes like rocks and such, it works very well.

Bakery

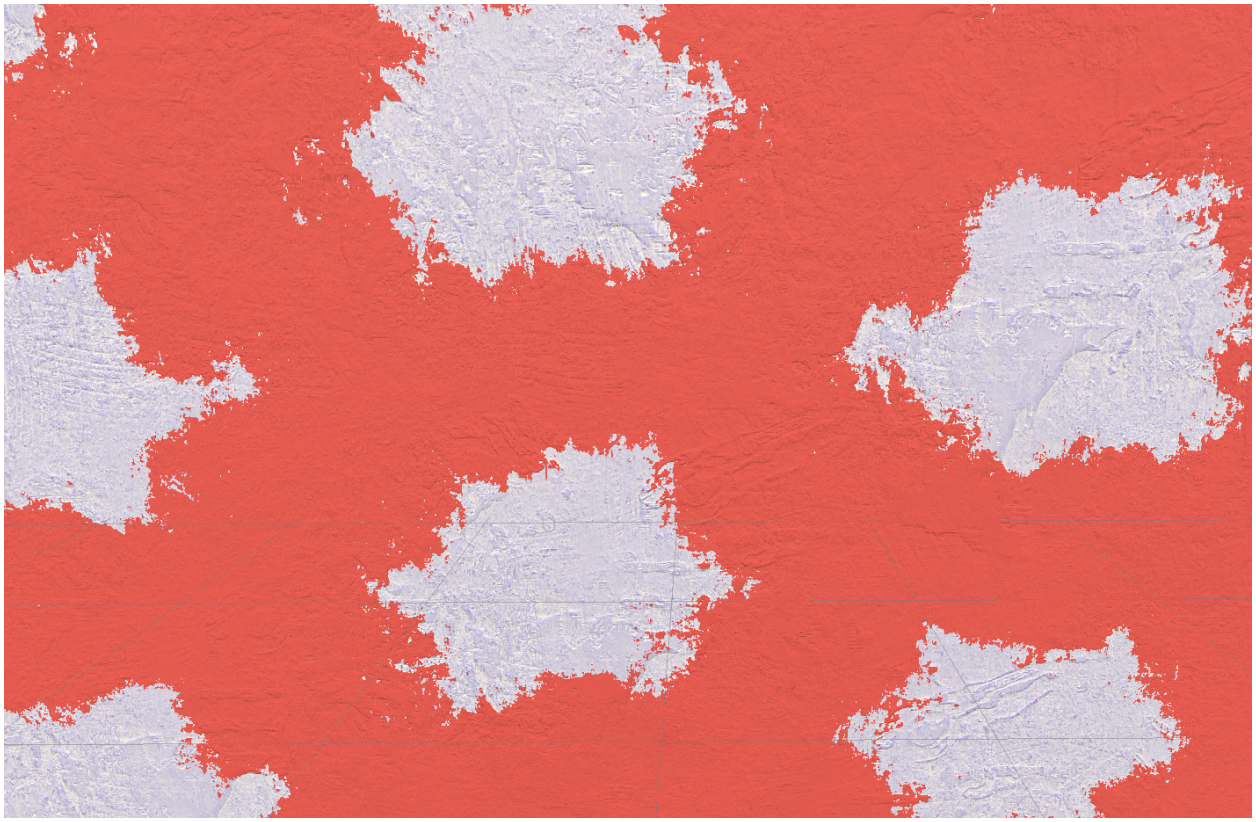
The Better Lit shader has full support for Bakery, a GPU lightmapping tool for unity with increased lightmapping speed and quality. Consult the Bakery documentation for more information.

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 Shaders were written using Better Shaders. 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 URP2019, URP2020, Standard, HDRP2019, and HDRP2020, 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.

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 five incompatible rendering engines in Unity (URP2019, URP2020, HDRP2019, HDRP2020, and the Built in Rendering Pipeline).

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 3 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.