

Mesh Workflow

for MicroSplat

Overview

The mesh workflow is a powerful tool allowing you to paint up to 32 textures onto meshes in Unity. It can even blend these textures over an existing shader, allowing you to use whatever shader you like for the main texturing.

Workflow Choice

The Mesh Workflow has two possible ways to store your painting data- as a Texture, or by storing the painting data in the vertices of the mesh. There are advantages and disadvantages to both workflows.

When using the Vertex based workflow, tessellation is not supported and you are limited to 28 paintable textures - 24 if you are using the Wetness, puddles, streams, or lava features. This is due to the maximum amount of data which can be sent from the vertex to pixel stages of a shader. However, when using a vertex based workflow there is no special requirements on UV coordinates, and it is possible for many uniquely painted meshes to share the same material.

Note that your mesh must be marked as Read/Write in it's import settings to paint it- this option keeps the data about the mesh on the CPU so the vertex painter can access it.

When a backing texture is used to store splat data, there are some limitations and requirements as well. If UVs repeat or overlap, the painting will repeat in this section, so ideally everything needs unique UV coordinates. UV's do not have to be in a 0 to 1 space, rather the UV space for the splat maps will be stretched over the UV range. This means that if you leave lots of space between UV islands, you will end up wasting space on your splat map and getting overall lower resolution splat maps. The backing textures store splat weights, and one texture is used for every 4 textures used to paint with. Because each unique paint job requires different splat textures, a unique material is generated for each paint job, which can affect batching performance.

Quick Start

To start, drag a mesh into the scene that you wish to paint on and add the MicroSplatMesh or MicroSplatVertexMesh component, depending on which workflow you want to use.

Before you can start painting, you need to generate a shader and material for your mesh. There are three types of shaders you can generate.

SplatMaps

This is a traditional splat map shader, where you can blend up to 32 textures together to create your texturing. Which texture shows is controlled by a set of control textures, which store weights for each texture, and then get blended at runtime in the shader.

Overlay



An overlay shader can be used to blend with an existing material and shader. This has the advantage of being able to work with the existing shaders you might be using in your scene. In this mode, whatever material/shader you have on the object is drawn, and then the object is drawn a second time with the splat map shader. One texture index is used to represent alpha, allowing the original texturing to show through. Here we see some grime painted on the back of a couch.

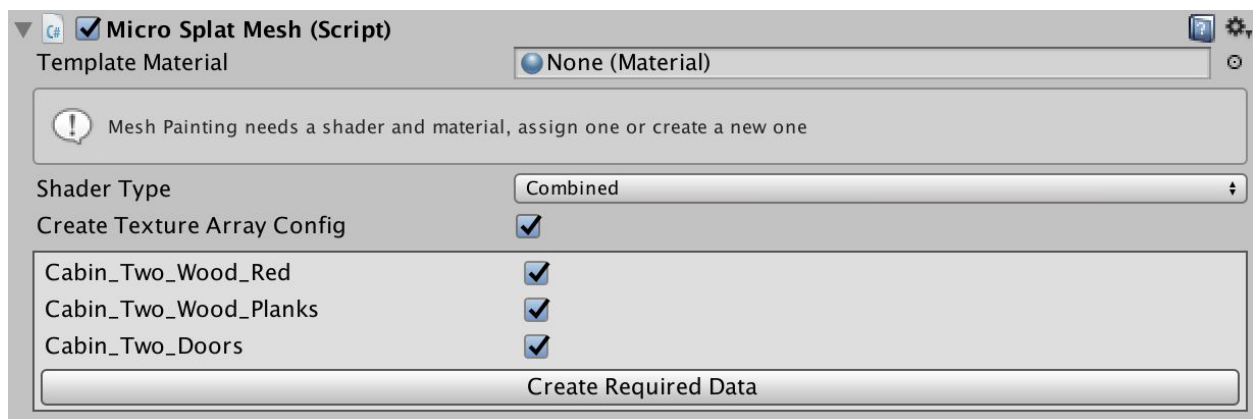
The main disadvantage to this technique is that the object needs to be drawn twice, and that because the shader cannot know about the surface it is drawing on top of, it cannot use height map based blending to blend with the first shader (which might not even have a height map).

Combined

The combined shader includes a regular shader and the splat map shader together. This allows you to blend with a traditional set of textures without needing to render the object twice. There are two disadvantages, one is that it has a fixed set of features and is not whatever

arbitrary shader you might want on your object. The second is that the extra textures will chew up some additional samplers, which depending on what platform your on may limit the number of features you can add to your shader (see MicroSplat Core documentation for more info on Sampler limits).

Once you have selected which type of shader you wish to use from the drop down menu, press the “Create shader and Material” button. This will prompt you for a location to put the data, which will be put inside a MicroSplatData directory. It will then generate a shader and material, and a Texture Array Config where you will pack your textures for use into a set of Texture Arrays, and then finally select the material for you.



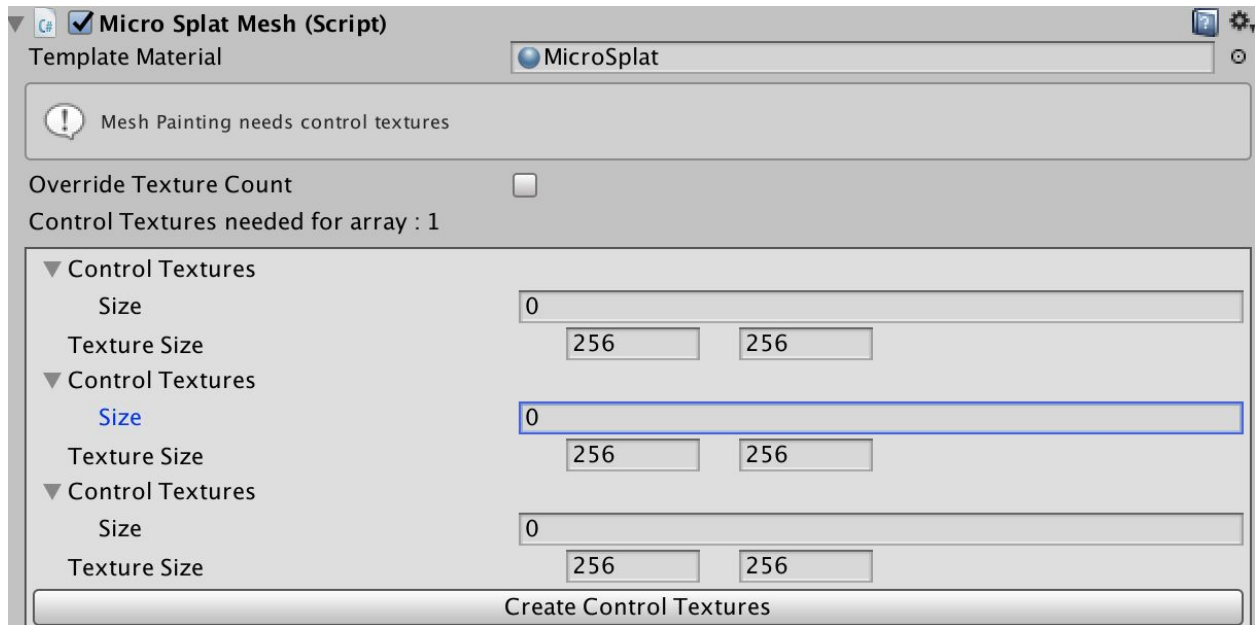
Once you have added the MicroSplatMesh or MicroSplatVertexMesh component, you can select the shader type you wish to generate. You also have the option to generate a Texture Array Config for your textures, or uncheck this if you want to use an existing config.

If you are generating a combined mode shader and your mesh has submeshes, a list of submesh materials is show so you can select which materials will be converted to MicroSplat shaders, and which ones will remain the original shader.

Once you have configured your selections, press Create Required Data and select a location for the MicroSplatData directory to be created in. It will generate a material and shader, along with texture arrays and any other data that is needed.

Once this is done, the material will be selected so you can assign your texture arrays or adjust shader options. If you generated a texture array config, you can find it in the MicroSplatData directory to add textures to it for painting.

If you are using the texture backed workflow, reselecting your mesh the MicroSplatMesh component will direct you to add backing textures for the control maps.



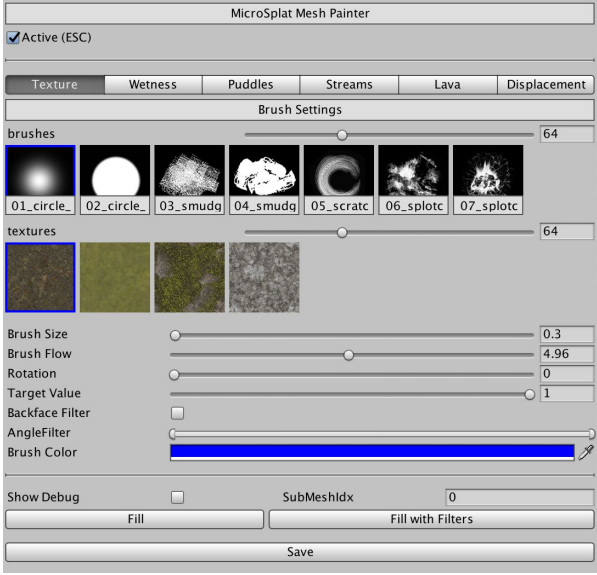
MicroSplat will detect how many textures are in your texture array, and automatically generate a control texture for every 4 textures you use. You can override the count if you want to specify it manually, and adjust the size of the backing texture, then press the “Create Control Textures” button to generate the textures, which will be automatically place in the MicroSplatData directory your material was created in. Note that in the example above we have three submeshes, so we need control textures for each submesh.

A note on working with multiple objects

If you have multiple objects you wish to splat map with the same shader settings and textures, you can add the MicroSplatMesh component to each of them and simply assign the material to the template material field in the MicroSplatMesh component. They will need their own backing textures if they are going to have separate paint jobs, so you can generate them uniquely for each mesh. You can also paint across multiple objects by selecting them all, or a parent object, using the Mesh Painter.

Painting

Go to Window->MicroSplat->MeshPainter to open the Mesh Painting interface for the texture backed workflow, or Window->MicroSplat->Mesh Vertex Painter if using the vertex based workflow. .



Shot is from texture backed workflow

Select your mesh with the painting window open to begin.

The mesh painting interface can be toggled on and off with the escape key or from the menu. Here you can select a brush and a texture to paint. You can control the brush size, flow, and rotation. The brush is represented by an outline material while you paint, and you can control its color with the Brush Color property.

You can fill the entire mesh with a texture with the Fill button.

Please note the **SAVE** button on the bottom. Your painting will update immediately, but will not be saved to disk until you press the save button, so if you restart unity your paint job will be gone.

The vertex based workflow is similar, but the painter does not have texture based brushes. This is because there is rarely enough vertex resolution to make them viable.

Overlay and Combined shader workflows

When using the overlay or combined shader workflows, you specify one of your textures to be the equivalent of the “alpha” value in the material. Painting this texture will reveal the original material underneath (Overlay), or reveal the standard texturing shader (Combined).

Module Restrictions

Most of MicroSplat’s modules will work fine with the mesh workflow, however, there are a few restrictions.

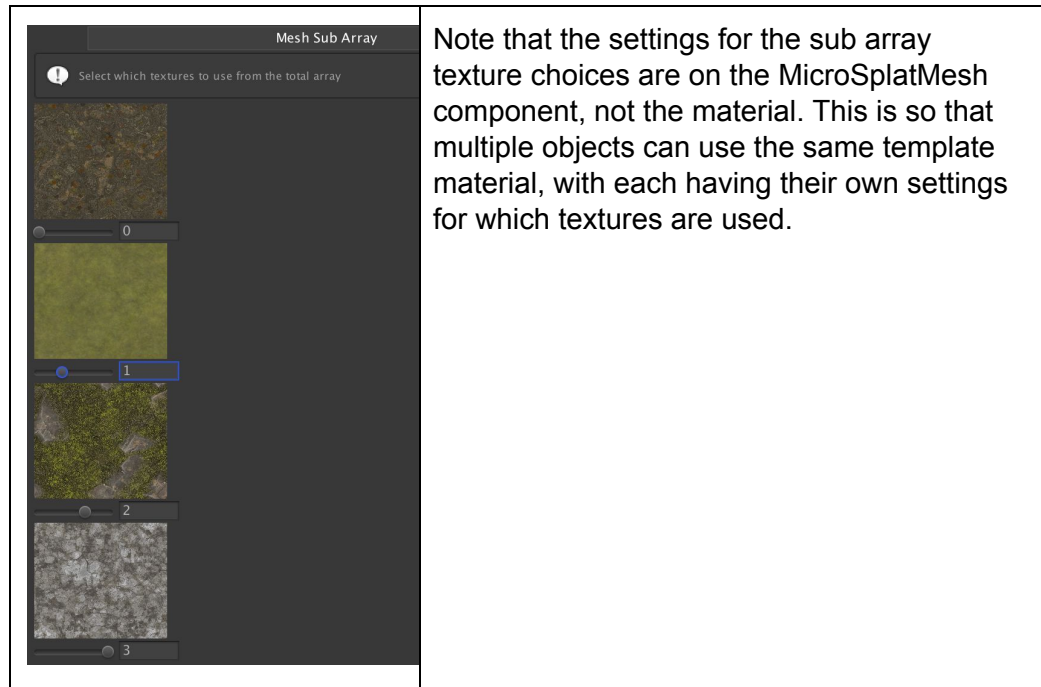
- Streams and Lava are available but you will not always get a correct result, because the UV arrangement can be arbitrary and the direction of flow cannot be defined. Dynamic streams and lava are not available, as they require a height map to work.
- Terrain Blending is also not available, as it requires the mesh topology be readable from a height map, and an arbitrary mesh cannot guarantee’d to follow these restrictions.
- Tessellation is not available in the mesh Overlay mode. This is because it’s being drawn over an existing shader, and there is no way to know what that shader has done to the geometry. Tessellation is only available in a texture based workflow.

- Runtime Procedural Texturing has no way to generate cavity or erosion maps- though you may provide your own maps for these features.

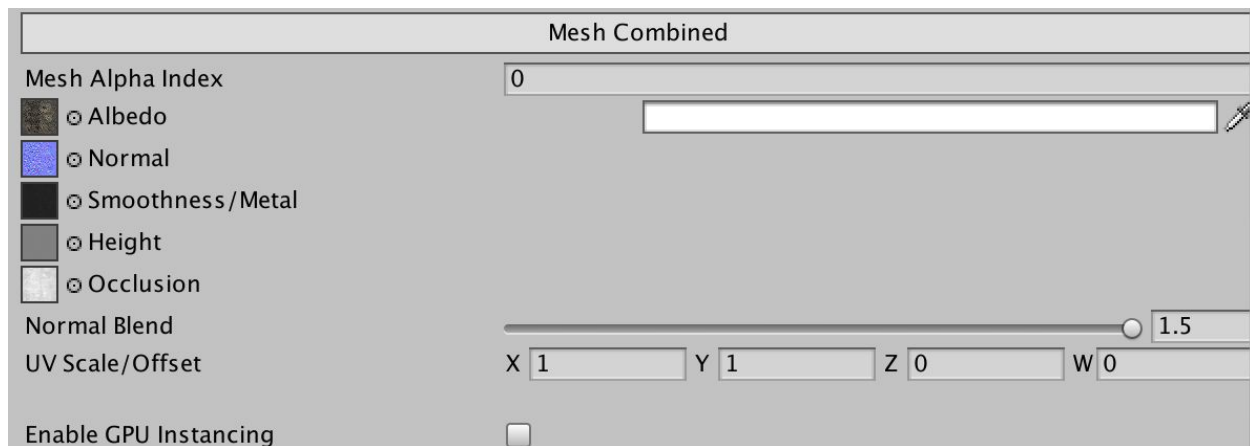
Shader Options

Source UVs	Texture UV
Mesh Shader Mode	Combined
Force Local Space	<input type="checkbox"/>
Packing Mode	Separate
Enabled Smooth/Metal Map	<input checked="" type="checkbox"/>
Enable Height Map	<input checked="" type="checkbox"/>
Enable Occlusion Map	<input checked="" type="checkbox"/>
Enable Emission Map	<input type="checkbox"/>
Disable Splat Maps	<input type="checkbox"/>
Use Sub Array	<input type="checkbox"/>

- Source UVs
 - Which UV channel to derive the UVs from. The UVs should be non-overlapping and in a 0-1 space. If they are not, then using Unity's lightmap UVs (UV2) which can be generated as part of the meshes import options is a possibility.
- Mesh Shader mode
 - This is usually set during the setup, but many options below will be dependent on which mode this is set to.
- In Combined Mode
 - A packing mode is available. In separate mode, textures are packed in the same format Unity uses for the Standard Shader. In packed mode, a more efficient packing of textures is used. The albedo and normal are as usual, but a packed map is available which contains Metallic in the red channel, Smoothness in G, Height in B, and Ambient Occlusion in A.
 - An option to disable splat mapping. This is useful if you want just want to use the shader and apply MicroSplat based effects to it, such as Snow or global texturing. This will produce the same result as erasing all the splat mapping, but without the shader cost of calculating any of it.
- Use Sub Array (Texture Backed Mode Only)
 - When checked, the material will only use 4 textures from the texture array, regardless of how many textures are in the array. This can allow you to have one texture array which is shared between many objects and select which 4 textures are used for each material. If you, for instance, have 32 textures in your array, this means you only need one control map instead of 8, reducing memory in cases when you only need a subsection of the available textures.



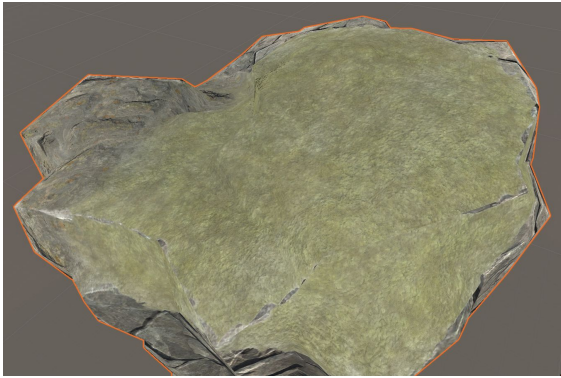
Shader Options



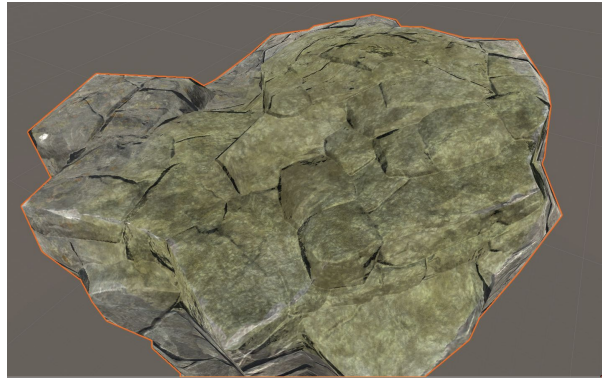
When in Combined or Overlay mode, options are added to the shader as well. In both modes a Mesh Alpha Index is available, which determines which texture is considered to be the transparent texture. Painting this texture will remove splat mapping, while any other will add it back.

The rest of the options shown above are only for Combined mode. Various textures will be available based on your packing mode, along with a tint and UV scale/offset options. The Normal Blend slider controls how the normal are blended between the two texturings. At 0, only the splat map's normal will be used where the splat map is shown. At 1, the splat map normal will be blended equality with the underlying normal map. This slider can go above one to strengthen the underlying normal. Note, a per texture option is called "Mesh Normal Blend" is

also added to the Per-Texture properties, such that you can adjust this blend for each splat map.

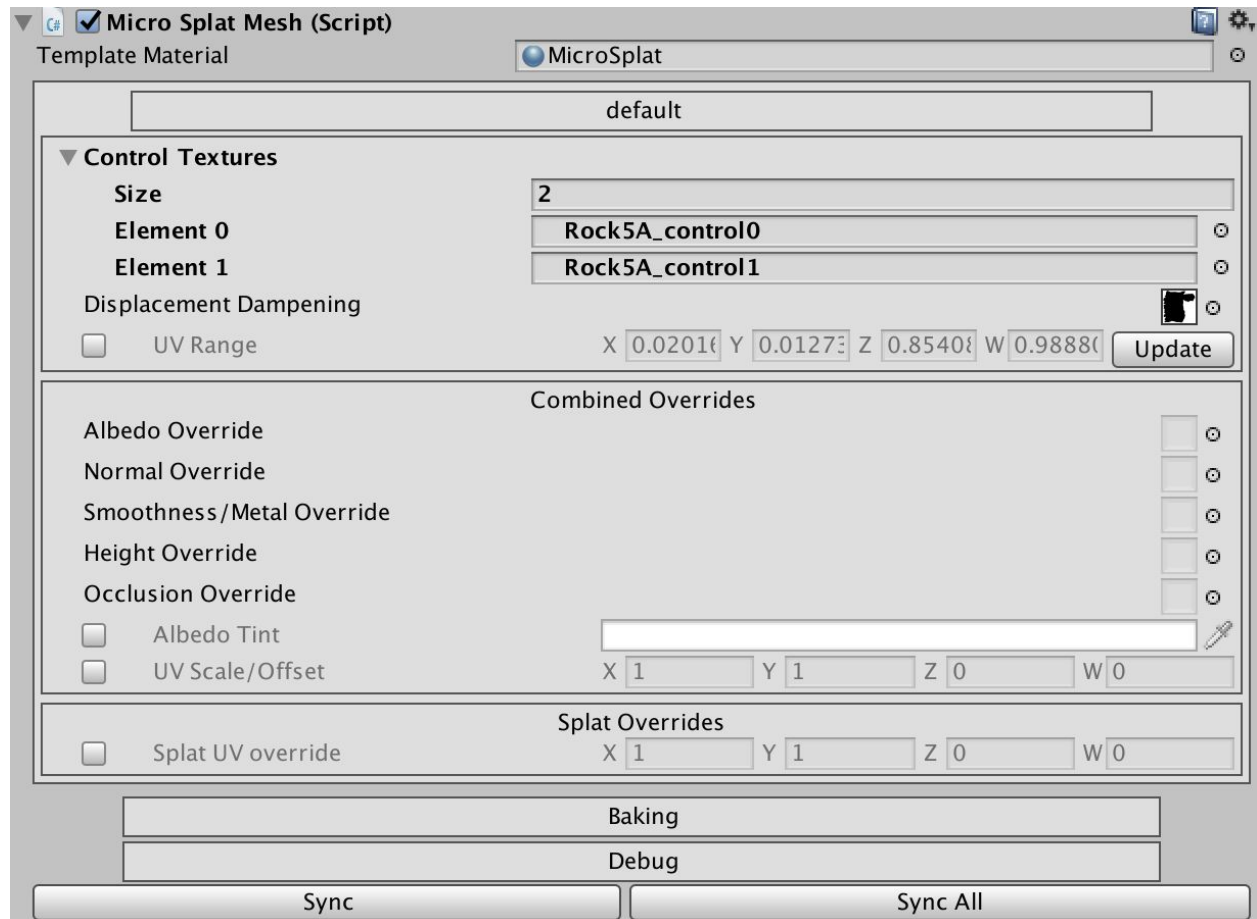


Normal Blend = 0



Normal Blend = 1.5

MicroSplatMesh/MicroSplatVertexMesh Components



Either the MicroSplatMesh or MicroSplatVertexMesh component is required on each object to be splat mapped. It's primary job is to sync the template material to the meshes on your object. Below the template material you will find a group of properties for each submesh in your model. This will contain 3 sections:

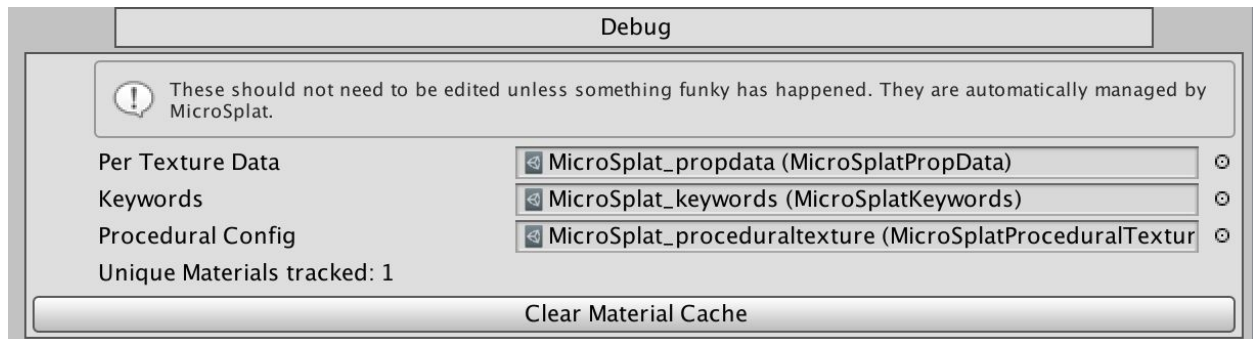
The first section contains the various control maps used for the shader, be that splat maps, displacement dampening, streams and lava, etc. It also contains the UV range data for the mesh, which can be updated if the mesh has changed.

If you're using the combined mode, a series of combined overrides are available, allowing you to override the values off the combined shader on a per-object basis. For instance, you might have different textures for each instance.

Finally, and overrides for the splat maps are also available.

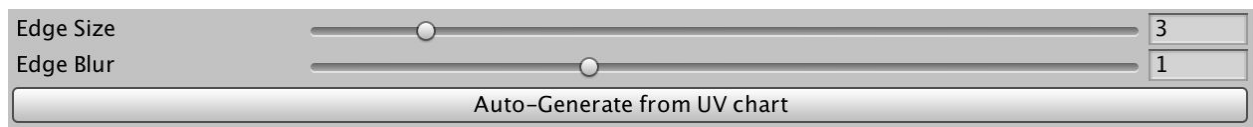
Note that if you set no overrides, then the material will be shared across every instance of the object. If there are overrides provided, a new material will be generated for that instance

and shared by every instance with those same settings. This allows for efficient material usage. Note that you can see how many unique materials are present in your scene in the debug section of the component.



Tessellation

If the tessellation module is installed (sold separately, some restrictions based on SRP used may apply), then tessellation is available for the combined and splat map modes when using a texture backed workflow. A common issue with tessellation on meshes is that cracks form between UV seams. A simple, if somewhat manual, fix is available though - displacement dampening. Once enabled in the shader settings, you can switch to the displacement tab in the painting window and paint the amount of displacement on the mesh just like you paint splat maps. This allows you to adjust the displacement amount to 0 around the cracks. Most commonly these happen around UV seams and hard edges. On that tab you will also find a simple tool which will catch the most common discontinuities in UV seams. To use it, set the edge size and blur (in control texture pixel size) and press "Auto-Generate from UV chart".



Baking Results

When using a texture backed workflow, the resulting data can be baked down into a set of textures to be used with standard shaders. This can reduce the cost of drawing the mesh considerably, and can be useful for LODs. However, because it is a single set of textures from the results of the shader, animated effects, view dependent effects, and world position relative effects cannot be preserved. And obviously the resolution will be based on your output textures, not tiling splat map textures which can maintain a much higher resolution look.

Vertex Instance Stream

When using the vertex painting workflow, a Vertex Instance Stream is added to your object. This object holds the data about the “paint job” for your mesh, and at runtime a feature of Unity called “additionalVertexStreams” is used to add this data to your mesh. This is very convenient for workflow, but has a downside - it prevents the meshes from being statically batches, because the data is dynamic.

To allow for static batching, you must bake the data out to unique meshes. Under the utility panel there is a Save Mesh feature which will allow you to save the selected mesh to a new asset file. Once this is done, you can remove the Vertex Instance Stream and replace the mesh with the new one on the Mesh Renderer component. Note that if you have material overrides on your MicroSplatVertexMesh component, then the object cannot be combined because it will still need a unique material.