MicroSplat

Overview

MicroSplat is a modular shading system for Unity Terrains. At its core is the fastest 16 texture shader available, which is automatically created based on the settings you choose in order to provide the most optimal shader possible. This unique shader compiling framework allows for MicroSplat to have more options than a traditional shader would provide, and doesn't require you to select from certain combinations of features, only to find that the feature combination you need is not available.

The major focus points of MicroSplat are designed around:

- Performance
- Extremely High Quality rendering
- Ease of Use
- Modularity

Resources

Having an issue? Please use the forums!

[Forum Link](#)

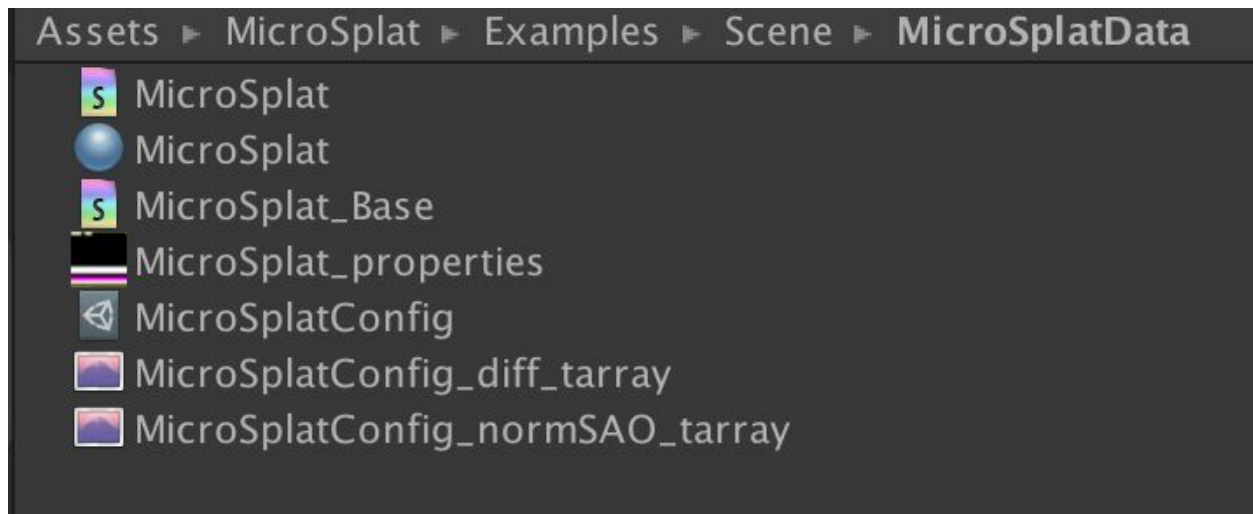Interesting in learning more? Check out my YouTube channel.

[YouTube Channel](#)

Getting Started

If you have an existing Unity terrain, getting started is only a few clicks.

- Select all the terrains you want to treat as a single, continuous terrain.
- Click "Add Component" in the inspector window, and select the "MicroSplatTerrain" component.
- Press the "Convert to MicroSplat" button.

In just a few seconds, this will read the existing textures from your terrain and pack them into custom texture arrays, and create a template material and shaders. A full PBR workflow is generated from the texture maps you have available. For instance, if you only use diffuse maps, the MicroSplat will generate a Normal Map, Height map, Smoothness and Ambient Occlusion map for your textures automatically.

Assets ► MicroSplat ► Examples ► Scene ► **MicroSplatData**

MicroSplat
MicroSplat
MicroSplat_Base
MicroSplat_properties
MicroSplatConfig
MicroSplatConfig_diff_tarray
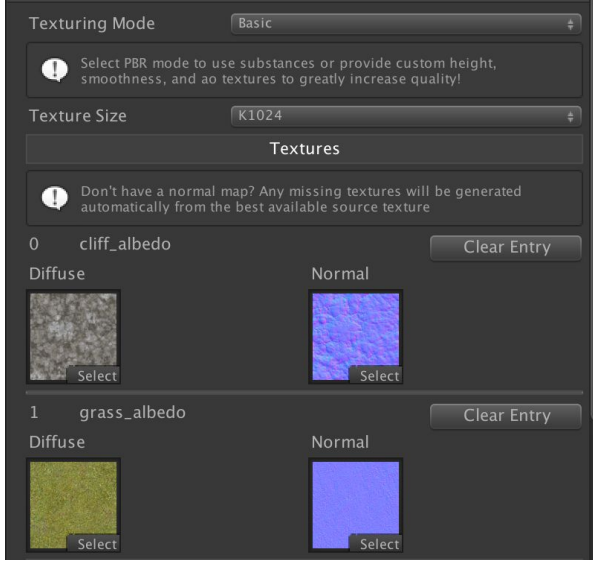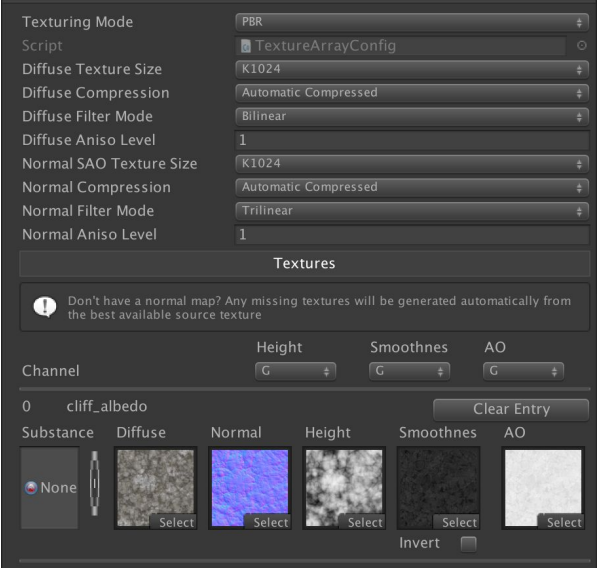MicroSplatConfig_normSAO_tarray

All data created will be organized into a MicroSplatData directory located next to your terrain data.

The Texture Packer

Once the conversion is done, the Texture Array Config will be opened. This is where you will manage textures for your terrain from now on. (You can come back to this screen by re-selecting the TextureArrayConfig object in the MicroSplatData directory.) There are two modes to this interface; a basic version, which only has slots for diffuse and normal maps, and a PBR mode, which allows for substance importing, a full PBR workflow, and more controls over the resulting textures.

Note that no matter which mode you select, any missing textures will be automatically generated from the best available texture as the data is packed into the resulting texture arrays (they don't show up in the interface, or get written to disk). However, the best quality will always be obtained by providing high quality PBR textures, with all texture data defined. High quality normal maps and height maps have the biggest effect on shading quality.

| Basic Mode: A single resolution for all textures is exposed, and only diffuse and normal maps are available. All other channels are generated automatically.	PBR mode: More control over texture formats, substance importing, and the ability to provide all PBR texture channels. You can even choose which channel to pull the data from, or invert the smoothness map if your texture software provides roughness maps instead of smoothness maps.

If you make any edits, press the Update button at the bottom of the interface to regenerate the texture arrays.

The Shader Generator

To quickly access your the shader settings, select the terrain and double click the 'material template' property on the MicroSplatTerrain component. This will open up the material which will be used by your terrains.

At the top of the shader's UI is a list of currently available modules for MicroSplat. Clicking on one of these modules will bring you to the asset store where the module can be purchased. Once you download and install the module (just like any other Unity Asset Store asset), new features will become available in the Shader Generation section of the shader UI.
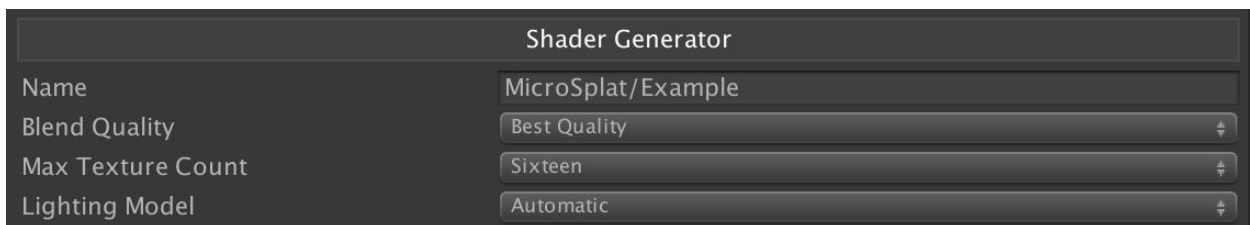

Changing any option in the Shader Generator section causes the shader to be recompiled with the new code included. As you add extension modules, the number of options available in this section will grow substantially. But with only the base package installed, there are only a few options available.

- The Shader Name. This is what the shader is called in Unity's shader selection interface. By default, it names your shader after your terrain.

- **Blend Quality**. This controls how many textures are blended for each pixel on the terrain. While many terrain shaders sample every texture that could potentially be used on a pixel, MicroSplat samples between 2 and 4. When set to "Fastest", only the two most weighted textures are sampled, however blending quality is reduced. On "Balanced", 3 textures are sampled. And at "Best Quality" the top 4 textures are sampled.
- **Max Textures**. This controls how many textures the shader supports on the terrain. In most cases, you can leave it to the default of 16. However, if you are running on OSX or some older (non DX11) platforms and turn too many module features on, you might run out of texture samplers (OSX is limited to 16 due to driver limitations in Apple's drivers, the hardware actually supports 32). Each reduction in this setting saves 1 sampler on these platforms. If you are running DX11, sampler's can be shared, and it is unlikely that you'll hit any limits.
- **Lighting Model**. This allows you to override Unity's default BDRF (specular) function. By default, Unity will usually use a GGX based BDRF which can look too shiny on glancing angles. Switch to 'Simplified' will reduce this. Note that setting this setting will force your terrain to be rendered as a forward renderer object when running in deferred rendering.
- **UV Mode**. By default, the splat maps will use the UVs from the terrain, but when you set this to WorldSpace, it will use a world space projection for the splat map UVs, which can be useful if you want multiple terrains to seem perfectly with oddly scaled UVs.
- **Shader Model**. By default, MicroSplat will choose the minimum shader model possible for your shader. Shader Model 3.5, or 4.6 if Tessellation is enabled. However, you can override the shader model to 4.6 or 5.0 using this control.
- **Disable Height Blending**. This will disable height based blending and use a linear blend, like Unity's included shader. If trying to get the most performance possible, this can reduce calculations slightly as well.
- **Sampler Mode**. This allows you to force the textures to be sampled using the LOD or Gradient sampler modes. For more information on this, see the notes on Per-Texture UV scale near the bottom of this document.

Shader Properties

Below the compiler are various shader property sections.
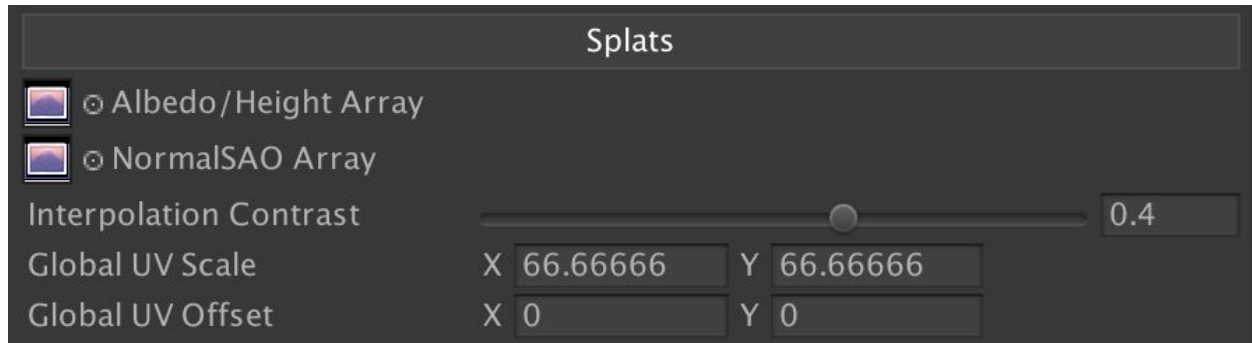

This is the main splat mapping parameters. The Texture Array Config output our full PBR texture set into just 2 texture arrays, which you can see assigned here.

- Interpolation Contrast
    - This controls the blend between textures when painted on the terrain. At 0, it closely mimics a Unity terrain. At 1, the blend is extremely sharp between two textures, and based on the height maps.
- Global UV Scale/Offset
    - These allow you to control the amount of tiling the textures use over the terrain, and offset the textures as well.

Per Texture Properties
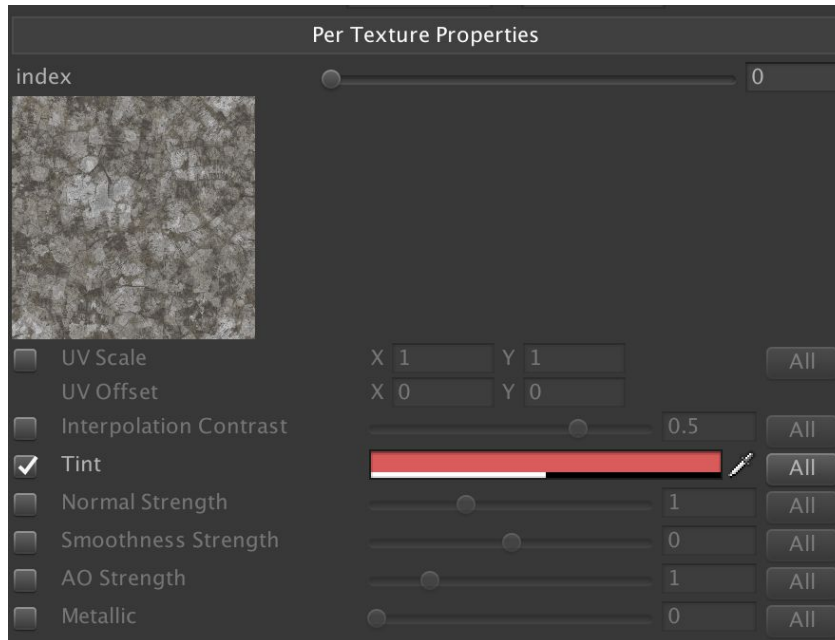
The Per Texture Properties section of the shader UI lets you control various parameters of each texture in the shader. This allows you to give each texture it's own UV scale and offset, tint colors, or adjust PBR parameters like how smooth or metallic the surface is. Additional modules will add additional Per Texture Properties for their various features.

Next to each section of properties is a checkbox. This allows you to toggle the effect on and off. Doing so will recompile the shader with that code removed, ensuring the most optimal shader possible. Once enabled, you can use the "index" slider to choose a texture and adjust the properties. If at any time you wish to set all the textures to the same value, you can press the "All" button and your current settings will be applied to all of the terrain textures.
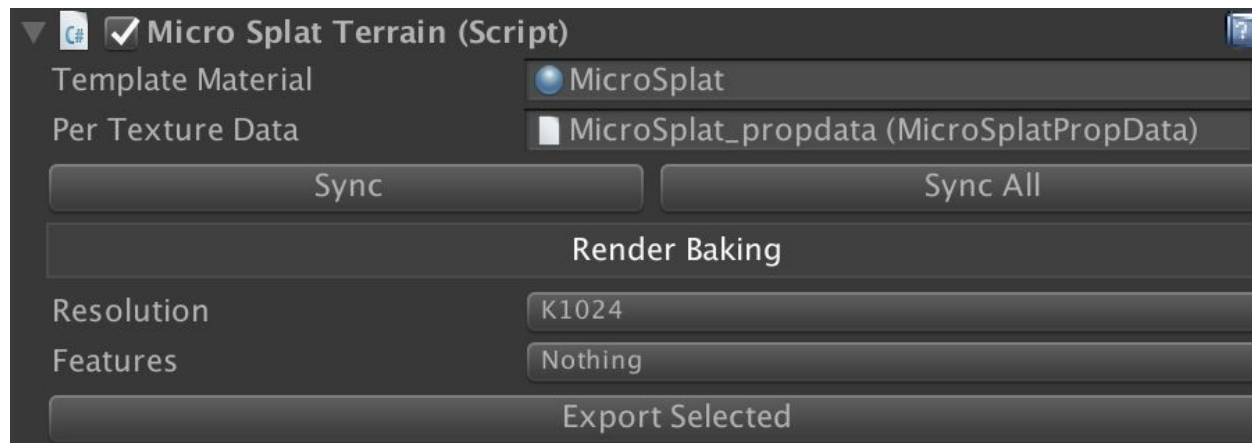
Debug
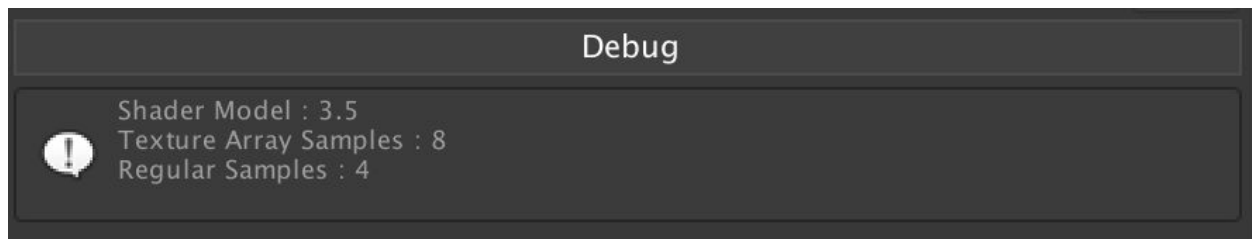
The debug section of the shader UI shows you information about the cost of your shader. While these numbers are somewhat subjective, they can be used to learn about the relative cost of each feature you enable.

MegaSplatTerrain Component

The component placed on each Terrain has a reference to the template material used and the per-texture property data. You can quickly get to the material settings by double clicking on this material. When multiple terrains are setup together, a single template material is used for all of the terrains, and MicroSplat manages syncing changes from the template materials to the actual materials the terrain uses internally. If you change settings on the template material via script, you can call MicroSplatTerrain.SyncAll() to perform the same sync. Sync buttons are

exposed, but you really shouldn't have to use them unless you change the template material used by terrain.

Additional modules may install additional controls into this section. Below them is a rollout titled "Render Baking". You can use this section to render out the resulting texture data of your terrain, which can be used as a vegetation tint mask, for replacing terrains with standard meshes with textures in the distance, or for other uses.

Additional Modules

Additional modules can be downloaded to expand the feature set of MicroSplat. Once downloaded and installed from the asset store, new options will appear in the Shader Generator section. Documentation for all modules is included with this package in case you have any questions about the workflow.

Global Properties

Often you might want to control aspects of MicroSplat from an external control system, like a weather manager such as Enviro. When a "G" button is located next to a property, you can toggle it to be driven by a global shader property. When these are enabled, the G will turn yellow and the property will no longer be editable.

You can then set these values globally via code, using the Shader.SetGlobal function, something like:

Shader.SetGlobalVector("_Global_WetnessParams", new Vector2(minWetness, maxWetness));

A list of all potentially global values:

_Global_WetnessParams	Vector2	Minimum and maximum wetness in the scene. Zero to one values expected.
_Global_PuddleParams	float	Maximum puddles in the scene. Zero to one values expected.
_Global_StreamMax	float	Maximum height of streams. Zero to one values expected.
_Global_RainIntensity	float	Rain drop intensity. Zero to one values expected.
_Global_SnowLevel	float	Overall amount of snow. Zero to one values expected.
_Global_SnowMinMaxHeight	Vector2	Begin and end height of snow in scene in meters.
_Global_WindParticulateRotation	float	Direction of wind particulate flow in radians. Should not be animated at runtime.
_Global_WindParticulateStrength	float	Strength of wind particulate effect. Zero to one values expected.
_Global_SnowParticulateStrength	float	Strength of snow particulate effect. Zero to one values expected.

Vegetation Studio Integration

When Vegetation Studio is installed, a few extra options are exposed. "Vegetation Studio Grass Mask" can be used to add foliage into the terrain texturing at distances beyond which it is feasible to draw actual foliage.

"Vegetation Studio Shadow Map" can be enabled to add raycast shadows for distant trees. This allows tree's and other objects well past Unity's shadow bounds to still cast shadows onto the terrain that correctly move and elongate with the sun position and angle.

Finally, MicroSplat can render out a tint map, which allows the bottom of the grass to be tinted towards the terrain texture using an appropriate shader on your vegitation.

 Please consult the Vegetation Studio Documentation for more information on these features.

Additional Information

Are you interested in creating assets for sale that use this shader system? Do you make an asset which could use a better shader in your demo scene? If so, please contact me and I can help you do this with a special version for inclusion into your asset.

Are you a shader developer who wants to extend this system with new features, even selling an extension module on the store yourself? If so, please contact me and I can help you do this.

Working with Multiple Terrains

First, make sure all your terrains have a unique name. MicroSplat uses the terrain name as a prefix to generate various resources it needs for each terrain, so if multiple terrains have the same name, they will overwrite each other.

When working with multiple terrains, you can setup all the terrains as one "system" by selecting them all and doing the conversion. This will create one MicroSplatData directory for all of their data, with a single shared material, shader and per-texture property object used by all of the terrains. Modules which need to generate additional textures or data will create these based on the terrain name and place them in this same directory.

If you wish to have multiple terrains that do not share settings, it is suggested to put each of the TerrainData assets Unity creates into it's own folder and set them up one at a time. This will give them each their own MicroSplatData directory, with their own shader, material, and per-texture property data object.

Note that you should avoid renaming things in the MicroSplatData directory, as this can cause various issues when objects need to find each other, often causing MicroSplat to regenerate the data it needs from scratch, or potentially finding the wrong data. Also note that textures, like global maps, can be overridden on the MicroSplatTerrain object for each terrain- so you could have one MicroSplat system which is used across multiple terrains, but each terrain can have it's own unique global textures.

Per Texture UV Scale

On some GPUs, when Per-Texture UV Scales are enabled, small, sub-pixel black spots can show up between some texture boundaries. These are a side effect of an optimization that makes MicroSplat run very efficiently. In many games, with high resolutions, Temporal-AA and other post processing effects, these will be very hard to notice. However, if they are bothersome for your setup there are two options which can be used to completely eliminate the issue, but both come with some tradeoffs. The Sampler Mode property can be set to LOD Sampler or Gradient Sampler.

When set to LOD Sampler, mip map selection is handled inside of the shader instead of by the graphics driver, and a single mip map is chosen based on a UV scale of one. This means that any scale applied to textures via Per-Texture UV scale will still use the same mip map as the regular UV's would. As such, it is recommended to keep scales within a reasonable range when using this mode. Additionally, because Mip Map selection is handled by the shader,

anisotropic and trilinear sampling will be disabled. On the plus side, this pathway tends to perform better than the driver based sampling (Default).

Another option is to use the Gradient Sampler mode. This option computes an more accurate mip-map selection, but forces each pixel in a 2x2 shading quad to fully fetch the texture sample, which can slow down the shader significantly. But in this mode, correct mip maps will be chosen for each texture even when scaled.