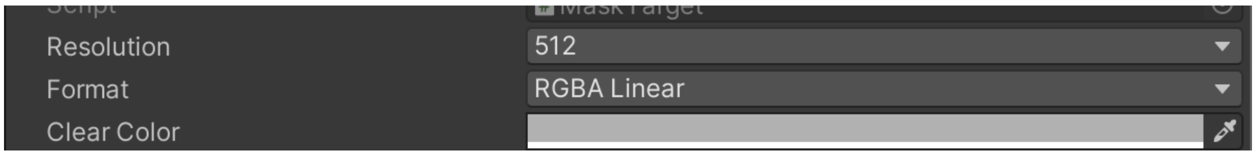
MicroVerse - Masks module

With the MicroVerse masks module installed, you can use MicroVerse to generate additional textures you might need. As an example, MicroSplat's shader supports things like global tint textures, masks to control where snow, wetness, puddles, streams, or laval appear, and other such effects. Systems which can be used to generate terrain stamps might output some of these textures as well, allowing you to use that data to compose a combined image, like any other stamp data in the system. Further, you can use existing MicroVerse filters, like slope or curvature maps, to filter these stamps as well.

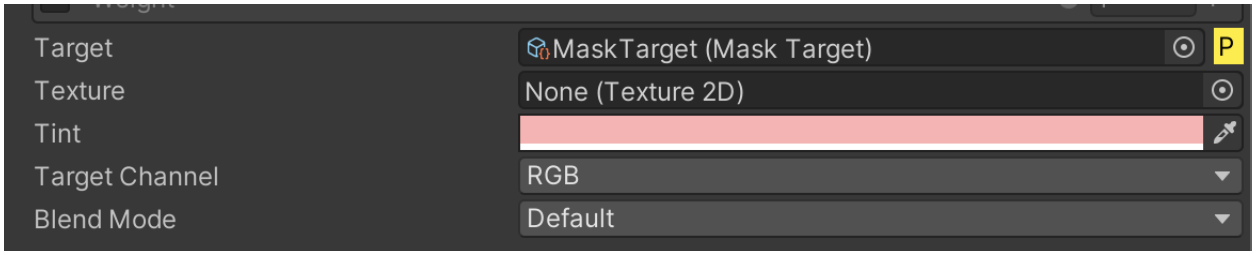


Resolution	512
Format	RGBA Linear
Clear Color	

To use the masking system, you first create a Mask Target scriptable object from the right click context menu (MicroVerse/Mask Texture). Once created, this will act as the container for all of the generated textures, which will be named based on the name of the terrain and mask. You can have as many Mask Targets as you need.

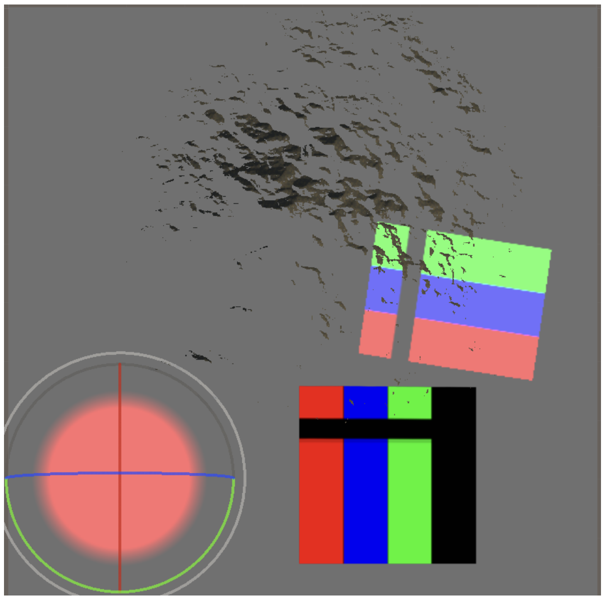
The mask target also lets you set the parameters for your masks. Resolution, texture format (RGBA in linear or sRGB color, or an R8 texture). You can also specify a clear color for the initial color of the texture.

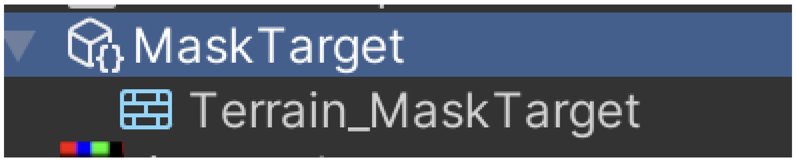
Next you can create a Mask Stamp from the MicroVerse stamp menu.



Target	MaskTarget (Mask Target)
Texture	None (Texture 2D)
Tint	
Target Channel	RGB
Blend Mode	Default

The stamp needs to target a specific Mask Target so it knows where to write textures. You can assign a texture for the data it will project into the target texture, which can also be tinted. The Target Channel lets you decide which channels to write data into- for instance, an external system might have multiple things packed into one texture. Finally, you can set the blend mode for how this stamp should be blended with the existing stamp data.

	In many cases, the data you might be rendering is not directly visible on the terrain. Note the small P button next to the target - if this is pressed, the resulting data will be rendered on the terrain using the preview system. This is an example of several mask stamps on a color texture.

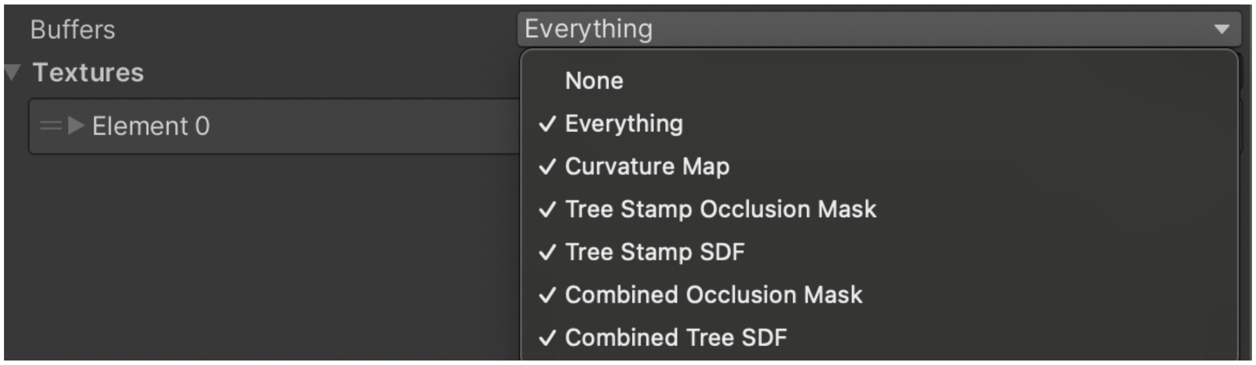


After saving the scene, a texture will appear under the mask target scriptable object. This can be assigned for use elsewhere in your project, and will be updated when your stamps change. Note however if you change the format of the mask texture it will replace this texture with a new one, breaking any references to it.

Extracting Internal Buffers

The masks module also allows you to extract various maps that MicroVerse internally generates, such as curvature maps for the terrain, the occlusion mask, or SDF data for tree stamps or all trees, which can be used for rapid lookups for gameplay operations.

To extract this type of data use the right click context menu to create a BufferCaptureTarget scriptable object. Then assign this object to the MicroVerse component.


Buffers

Everything

Textures

Element 0

None

✓ Everything

✓ Curvature Map

✓ Tree Stamp Occlusion Mask

✓ Tree Stamp SDF

✓ Combined Occlusion Mask

✓ Combined Tree SDF

Curvature Map - This is a single channel texture representing the curvature of the terrain, 0-1 from concave to convex. This is used by the curvature filter on stamps.

Tree Stamp Occlusion Mask - This is a single channel texture with a pixel set to the weight of the current tree, and black where there are no trees. The stamp will be named with the tree stamp's name on the end of it, so if you have non-uniquely named tree stamps only the last one will be exported.

Tree Stamp SDF - This is an SDF version of the Tree Stamp Occlusion Mask, where each pixel stores the distance to the nearest tree from that stamp. To get the distance in pixels, multiply the value by 256. To convert that into meters, you'll have to compute the pixel per meter of this map using the terrain's size and texture size. The stamp will be named with the tree stamp's name on the end of it, so if you have non-uniquely named tree stamps only the last one will be exported.

Combined Occlusion Mask - This is the occlusion mask for the whole terrain. **R** stores height stamp occlusion, **G** stores textures, **B** stores trees, and **A** stores details.

Combined Tree SDF - This is a combined SDF of all tree stamps.

Note that the first time MicroVerse updates all the textures for the output will be generated, which can take some time to do. After that, updates when making any edit are fast.

Notes

MicroVerse stores the images in the scriptable objects you create as sub assets. This allows them to be stored and referenced by other assets in Unity, but avoids needing a lengthy import process that would make real-time updates of MicroVerse impossible. So while this has the upside of being automatically updated with any change while not breaking real-time updating, it has some downsides as well.

First, because the textures don't exist as a separate asset with a meta file, there's no way to change the format of the textures without breaking any references to the texture. In other words, if you change a MaskTarget's texture resolution or format, any material or script using that texture will have a missing reference and will need to have it reassigned to the new version.

Second, because the textures are saved internally in uncompressed format, there's no easy way to easily compress them. That said, you can export the textures from these scriptable objects and have them written into the project as TGA files, which are imported like regular textures, compressed, etc. But these versions won't be automatically updated with every change.