Terrain Mesh Blending

For URP 14.0.10 and URP 17.0.3. I had another look into the old, experimental terrain mesh blending feature as it did not work with “Depth Priming”. I addressed this issue but also added other and more common solutions.

For those who are not familiar with terrain mesh blending:

> Terrain mesh blending allows you to smoothly blend meshes like cliffs or overhangs with your terrain. There are different techniques to accomplish such blending – like probably the most common one which draws the mesh object twice using the terrain material for the first draw and projecting it on top of your mesh.
>
> The original *terrain blend shader* worked differently and added a soft intersection between terrain and mesh using alpha blending and slightly tweaked depth values. Latter broke when “Depth Priming” was enabled as the URP makes all materials use ZTest Equal during the opaque pass whereas this technique needs ZTest LEqual.

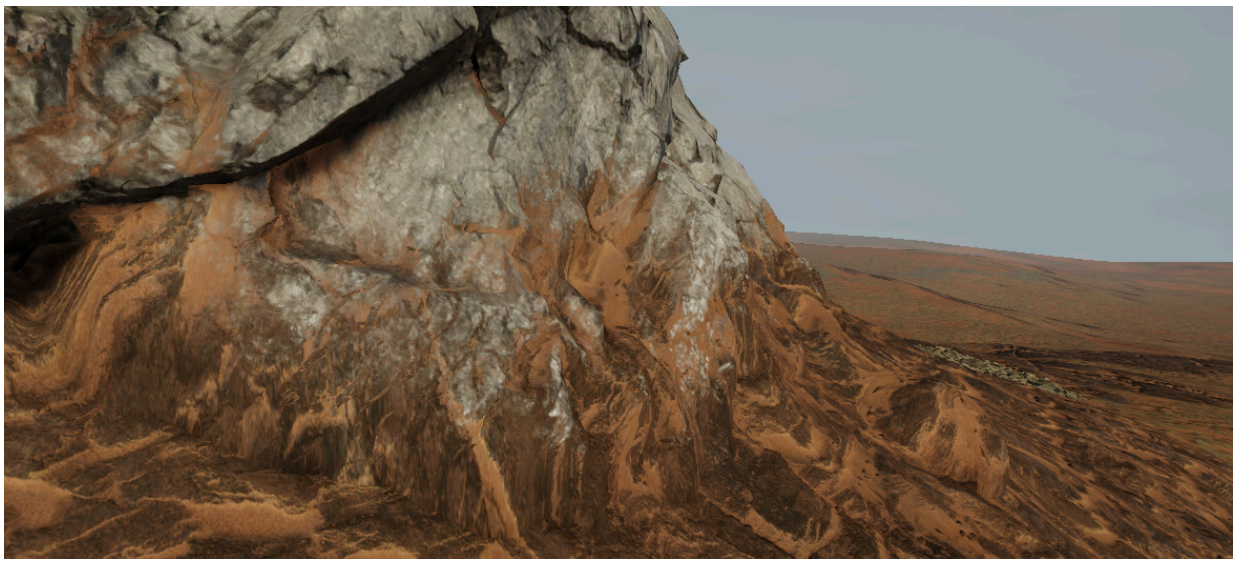
While I still like the idea of original *terrain blend shader* I also added a shader graph based upon the first method described above as it is more accessible by reducing the amount of dependencies and works under all conditions as well as a shader graph which adapts the vertices to the surface of the terrain.

All three shaders support detail texturing which is implemented as a custom function and can be used in other shaders as well.

Blend With Terrain Shader Graph

This shader graph picks up the most common technique of terrain mesh blending and projects the terrain textures and vertex normals onto the mesh around the intersection.

Apart from this it does not do anything fancy and is straightforward to use.

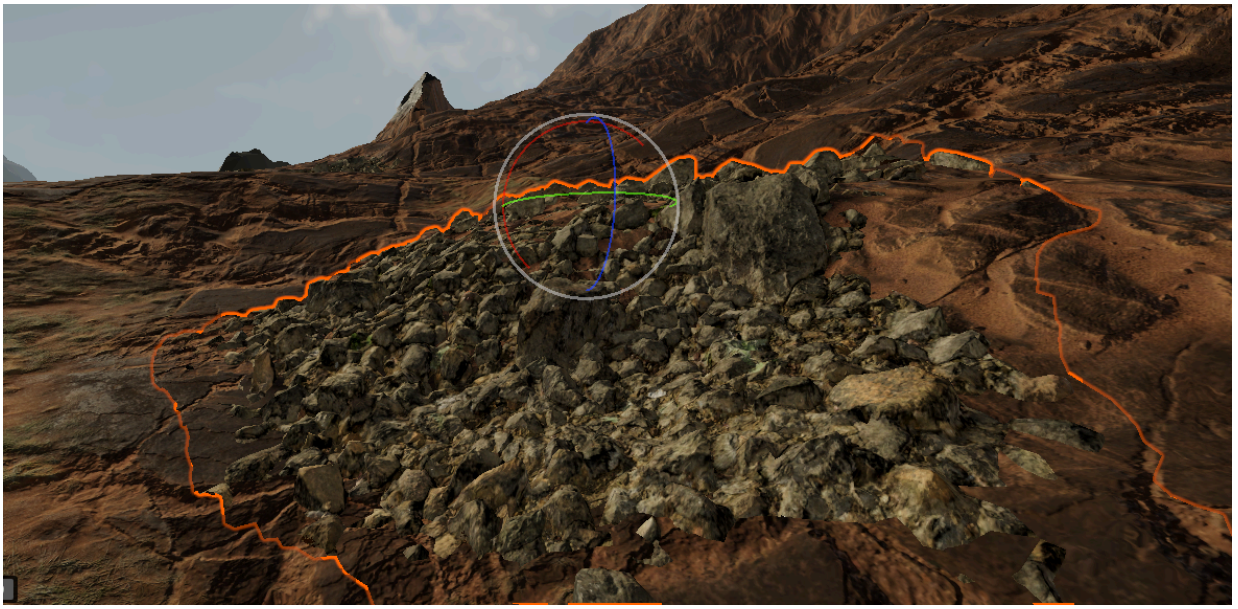


Pros and Cons

- In order to make it really fast the shader only projects **one single terrain layer** onto the mesh which is specified in the material. So you need to paint your terrain accordingly.
- As the terrain uses a simple top down projection you will get some **texture stretching** on really steep slopes of your mesh.
- It adds additional texture lookups but only around the blend zone.
- It does not need additional draw calls or renderer features.
- It is a shader graph.

Adapt To Terrain Shader Graph

This shader actually does not blend with the terrain but will adapt the terrain's height and apply it to the vertices of your mesh so you will get some kind of "terrain carpet".



Here the originally flat rock cluster got deformed to match the terrain surface.

It is a nice and handy shader to add some decoration to your terrain. However it is a purely visual effect. So meshes which need accurate collisions are not supported as collision geometry does not follow the terrain.

Pros and Cons

- The shader does not handle colliders of course.
- The shader does not handle LOD changes of the terrain.
- The shader will displace vertices along the local y axis so the mesh needs the right orientation in object space and must be kind of "parallel" to the local xz axes.

- As the shader may heavily shift vertices up and down you have to **adjust the bounds** of the mesh to prevent it from being culled altho it should be visible. *Use the provided "AdjustMeshBounds" script to do so.*
- Meshes should be scaled uniformly as the shader calculates scale only along the y axis.
- Meshes will be stretched on steep slopes.

Terrain Blend Shader V2

Version 2 addresses two shortcomings of the original version: the lack of support when "Depth Priming" is enabled and blended meshes poking through closeby geometry.

Fixing Depth Priming issues

In order to support "Depth Priming" we have to control depth testing in the pass that renders the blend. So I added a **custom renderer feature** which only draws the overlapping parts of the mesh and let us set depth testing to our needs.

The render feature which handles this is the "LuxURP_TerrainBlendingRendererFeature". You have to add it to your "URP Renderer Data" like all other renderer features such as "Screen Space Ambient Occlusion".

When it comes to the shader itself it now contains 2 passes:

- A regular "UniversalForward" pass which draws the mesh as usual during the opaque pass.
- A custom "LuxTerrainBlending" named pass which gets rendered by our custom renderer feature only.

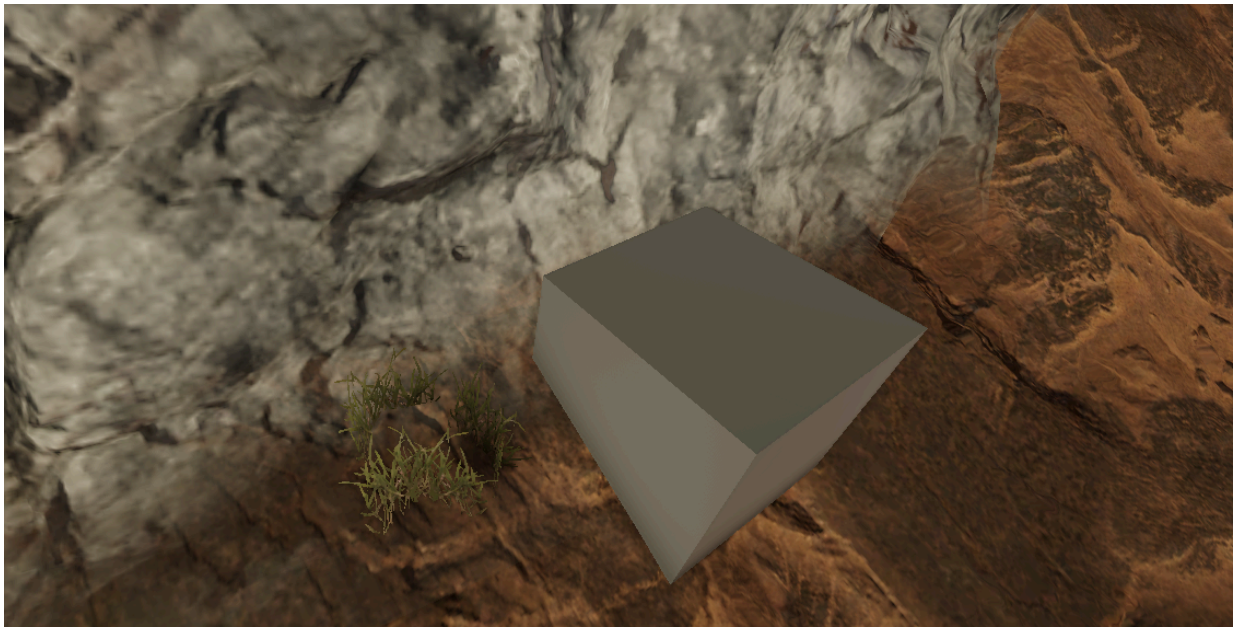
Fixing overlapping objects in the foreground

In order to fix this issue the shader now relies on the stencil buffer. The terrain has to mark the terrain pixels properly (so you have to use the Lux URP terrain shader or have to tweak the one you use) then the "LuxTerrainBlending" pass which draws the overlap will only render if the stencil buffer marks the pixel as terrain pixel.

Please note that using the stencil buffer to mark terrain pixels only works if "Depth Priming" is active. In case you do not use "Depth Priming" you have to stick with the old limitation and use rather conservative "Depth Shift" values as described in the original documentation.

It would work with "Depth Priming" disabled if all your shaders used by objects in the foreground let you tweak the stencil buffer – which unfortunately will not be the case if you use shader graph shaders.

*In case you use **deferred lighting,** using the **stencil buffer is not supported.***



In V1 such a smooth blending between mesh and terrain would override parts of the cube and most likely the entire grass prefab. Using the stencil now these problems are gone and we can use way higher "Depth Shift" values to create pretty soft blends.

Pros and Cons

- Due to the technique used here you will still see some strange parallax effects especially when the camera gets close.
- Needs you to use the Lux URP Terrain shader in case you want to benefit from stencil usage.
- Especially when you are CPU bound: We have to add an additional draw using the renderer feature. So I would not recommend adding hundreds of meshes using this technique – unless you use the resident drawer :)
- It is an HLSL shader.
- It needs some time to set it up and test the results.
- It blends just with any terrain layer without adding more and more texture lookups.

Getting started

Preparation

All shaders need you to create a combined **terrain height normal map** first. This is needed to calculate the distance to the terrain as well as smoothing the normals towards the normals of the terrain or tweaking the vertices.

> This being said you may have already guessed: Each terrain needs its own blend materials. And geometry overlapping 2 or even more terrains is not supported.

In order to do so, assign the provided *GetTerrainHeightNormal* script to your terrain and hit *Get Height Normal Map*. This will let you save a combined height and normal texture to disc.

> *The texture actually would only need 8 bit per channel but as GPUs which support half next to float may apply an 8 bit interpolation which results in a quite quantized result I recommend checking **"Use TextureFormat Half"** when addressing Metal (tested on M1 Mac) or Vulkan on Android. If you address desktop GPUs on PC or consoles you should keep it unchecked to save some memory and bandwidth.*
>
> *You can always switch the texture format in the import settings: Use RGBA 32 bit for desktop and consoles or RGBA Half for Metal and mobile devices.*

In case you want to use the ***Adapt To Terrain* shader** you also have to **tweak the bounds** of the mesh: As the shader may heavily shift vertices up and down the original mesh bounds will be too small and Unity might cull the mesh renderer way too early.

Do so by opening the **Windows → Lux URP Essentials → Scale Mesh Bounds** window.

- Add your meshes (LODs) to the array "Meshes".
- As soon as the first mesh is assigned the window will show the "Current Bounds" of the mesh which should give you an idea how to scale the bounds.
- Set "Extent Bounds" to an appropriate value and hit "Adjust Bounds".

The script then will create and save new meshes (as .asset) next to the original ones in the asset folder.

> *The new bounds will be calculated using "Current Bounds" + "Extent Bounds". As the shader will apply most of the displacement along the local y axis I would focus on adjusting the y extent. The actual bounds needed depend on the slopes of your terrain. Choosing bounds which are far too large will make culling less efficient.*

The last step in case you use the **Terrain Blend V2 shader**: You have to add the **"LuxURP_TerrainBlendingRendererFeature"** to your Universal Renderer Data.

Setting up the materials

Once you have the terrain height normal map, eventually adjusted the mesh bounds and added the custom renderer feature you are ready to go:

1. Create a new material and assign the desired shader.
2. Assign the created combined height and normal texture to the material and set *Terrain Position (in world space)* and *Terrain Size (width, height, length)*.
3. Now you can assign the material to your object and tweak it using the parameters described below.

Blend With Terrain Shader Graph

Terrain Inputs

> **Terrain Height Normal** The terrain height normal (as generated with the provided script)
>
> **Terrain Position** Position of the terrain in world space.

Terrain Size Length, height, width of the terrain like in the terrain settings. *Please mind the order!*

Terrain Layer

Terrain Albedo Albedo (Albedo/Smoothness) texture of the terrain layer you want to blend with. *In case you do not enable the Terrain Mask Map smoothness is sampled from the alpha channel of the albedo texture.*

Terrain Normal Normal texture of the terrain layer you want to blend with.

Terrain Normal Scale Scale of the terrain normal

Enable Terrain Mask Map Check this if your terrain material uses mask maps.

Terrain Mask Map The mask map which stores meteallness in the red color channel, ambient occlusion in the green color channel and smoothness in alpha.

Terrain Tiling Tiling value for the layer in world space. *Tiling in the terrain Layers is in meters. So you can calculate the proper tiling value using 1/10 = 0.1.*

Terrain Blending

Blend Per Pixel Depending on the vertex density of your mesh you may want to calculate the blending per pixel instead of per vertex. *Usually using per pixel blending should be more expensive. Note: If checked "Extrude Along Normal" will not have any effect.*

Texture Blend Height Height or range around the intersection over which the terrain texture should be blended.

Texture Blend Steepness Small or negative values will blend in the terrain texture only if the normals of the terrain and the object more or less match.

Texture Blend Contraction Lets you smoothen or sharpen the texture blending.

Normal Blend Height Height or range around the intersection over which the terrain normal should be blended.

Normal Blend Steepness Small or negative values will blend in the terrain normal only if the normals of the terrain and the object more or less match.

Normal Blend Contraction Lets you smoothen or sharpen the normal blending.

Extrude Along Normal Lets you extrude the mesh along its normals within the normal blend range which helps to ground it a bit and mitigates texture stretching.

Texture Break Up Blending If checked the shader will take the luminance and ambient occlusion of the underlying textures to break up the texture blending.

Occlusion Influence Specifies the influence of the sampled ambient occlusion

Blend Sharpness Lets you sharpen the break up.

Break Up Texture Stretching Values > 0 will distort the sampling UVs used to sample the terrain textures using the normal of the underlying material.

Adapt To Terrain Shader Graph

Terrain Inputs

Terrain Height Normal The terrain height normal (as generated with the provided script)

Terrain Position Position of the terrain in world space.

Terrain Size Length, height, width of the terrain like in the terrain settings. *Please mind the order!*

Terrain Adaption

Enable Adapt to Terrain Must be checked to make it all work

Terrain Offset Lets you raise or lower the vertices

Accurate Normals If checked the shader performs an accurate but more expensive normal calculation

Terrain Blend V2 shader – still experimental

Surface Blending

Depth Shift Amount the vertices get shifted towards the camera. *Keep this as small as possible to avoid shading artifacts in case Depth Priming is disabled and you can not use the stencil buffer.*

Terrain Height Normal The terrain height normal (as generated with the provided script)

Terrain Position Position of the terrain in world space.

Terrain Size Length, height, width of the terrain like in the terrain settings. *Please mind the order!*

Alpha Shift Lets you shift the start of the blending range above or below the terrain surface.

Alpha Contraction Higher values will shrink the range over which the alpha gets blended resulting in a sharper transition.

Shadow Shift Threshold In order to prevent the terrain from shadowing the parts of the blended mesh below the terrain we have to lift the position in world space the shadows are sampled at.

Shadow Shift Threshold defines the distance to the terrain surface in cm where this lifting kicks in. As we can not 100% precisely reconstruct the terrain's height values around 0.05 (= 5cm) should be fine in most cases.

Shadow Shift Factor which determines the final lift of the shadow sampling position. Usually a value of 1.0 should be fine.

Shadow Shift View Pulls shadow sampling position towards the camera.

Normal Shift The shader will blend the geometry normals towards the normals from the terrain to smooth the transition. Normal Shift lets you define if the blending shall start above or below the terrain surface.

Normal Contraction Higher values will shrink the range over which the normals get blended.

Normal Threshold Lets you reduce the influence of the terrain normal according to the angle between geometry and terrain normal.

Vertex Blend Threshold As we want to render only a small ring like part of the geometry around the intersection and not the entire mesh this param lets you tweak the height of this ring. Vertices further away from the intersection will be degenerated and skipped.

Pixel Blend Threshold Like above but on a per pixel basis.

More information can be found in the original documentation.

Surface Options and Surface Inputs

More or less common inputs. So I will not explain these in detail here except for:

ZWrite Blend Pass Lets you define if the blend pass shall write to depth. Set it to Off to speed up rendering and only enable it in case you encounter rendering artifacts in the blend pass.

Stencil Settings

In case your terrain uses the Lux URP terrain shader and "Depth Priming" is active the terrain may prepare the stencil buffer so that the blending parts of the mesh will only be rendered on top of the terrain and the mesh itself.

Select your **terrain material** and set:

- **Stencil Reference** to e.g. 127
- **Read and Write Mask** to 255 unless you need something fancy
- **Stencil Comparison** always
- **Stencil Pass Op** replace
- **Stencil Fail Op** keep

In the **blend material** set:

- **Stencil Reference Opaque** Stencil Ref set by the opaque pass - e.g. 127.
  In case you want to render the blended geometry on top of the opaque parts as well and not only on top of the terrain this has to match the value used in the following "Stencil Reference" param. If you do not render the blended parts on top of the opaque geometry you will get lighting discontinuities as only the blended parts will tweak the normals.
- **Stencil Reference** to e.g. 127 (must match the ref value from the terrain)
- **Read and Write Mask** to 255 unless you need something fancy
- **Stencil Comparison** equal
- **Stencil Pass Op** keep
- **Stencil Fail Op** keep

Detail Texturing

Enable Detail Texturing Check this to enable detail texturing

Detail Map The detail texture

Detail Tiling Tiling relative to the tiling defined by the UV you feed into the custom function

Detail Albedo Strength Lets you tweak the influence of the detail albedo

Detail Normal Strength Lets you tweak the influence of the detail normal

Detail Smoothness Strength Lets you tweak the influence of the detail smoothness

The layout of detail map is taken from HDRP:

- Red contains the albedo as grayscale
- Green contains the green component of the normal map
- Blue contains smoothness
- Alpha contains the red component of the normal map

The way the shader combines detail and regular sample however is taken from the BIRP standard shader as this is faster and more intuitive.

The detail texture can be masked by the Mask Map → Blue of the base texture set.