Brute Force - Ground Shader Documentation

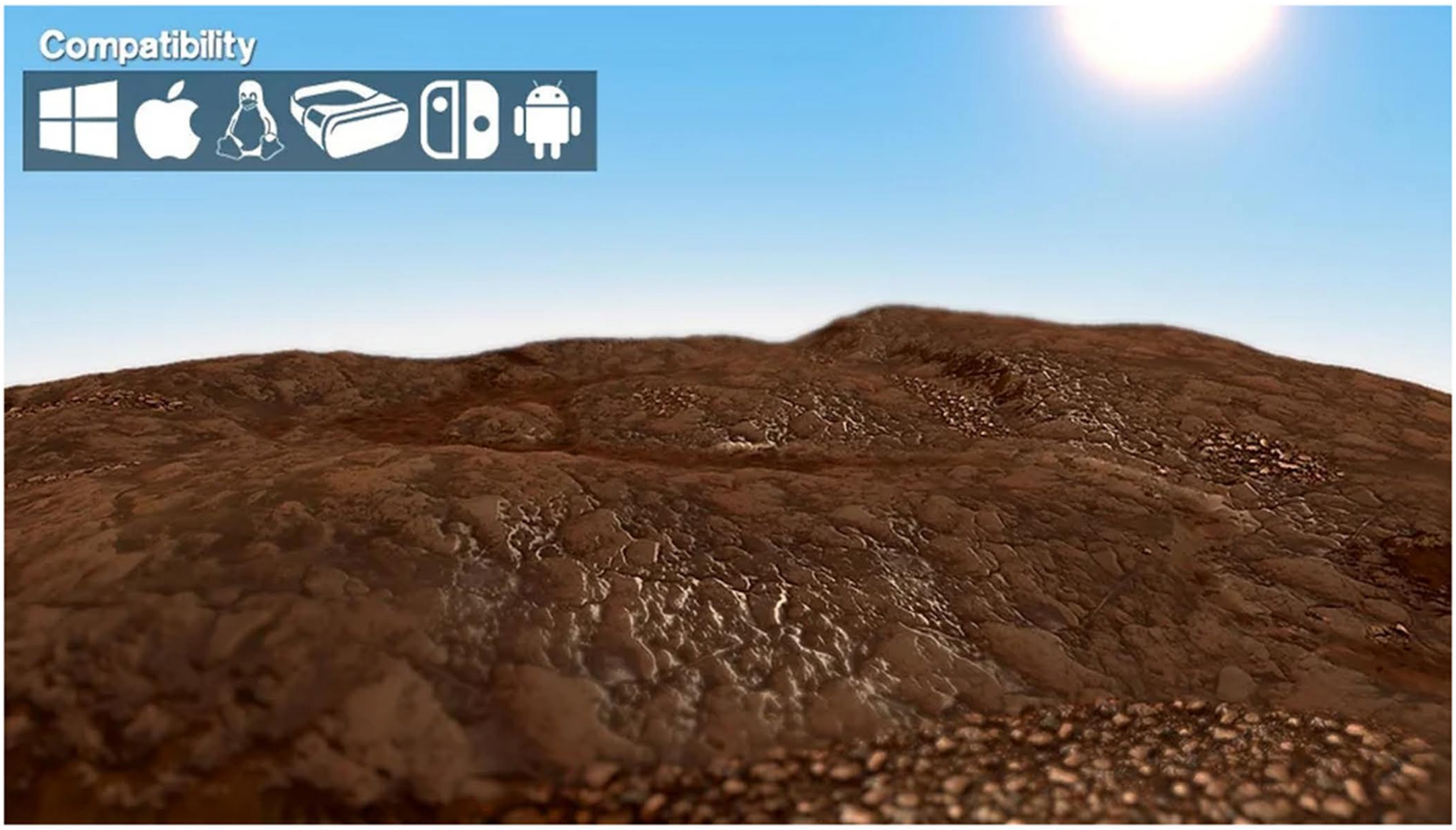
1. Ground Shader
2. Overview
3. Compatibility and Features
4. Getting started
5. Paint Ground
6. Ground Properties
7. WebGl and Mobile
8. Terrain Feature
9. BRUTE FORCE compatibility
10. Performance Tips
11. FAQ & Troubleshooting

1. Brute Force - Ground Shader

Brute Force - Ground Shader Unity Asset Store

This is the **Brute Force - Ground Shader** visual and online documentation.
Consider leaving a review on the asset store to support me c:, it helps a lot!
And if you have questions email me at: **bruteforcegamesstudio@gmail.com**

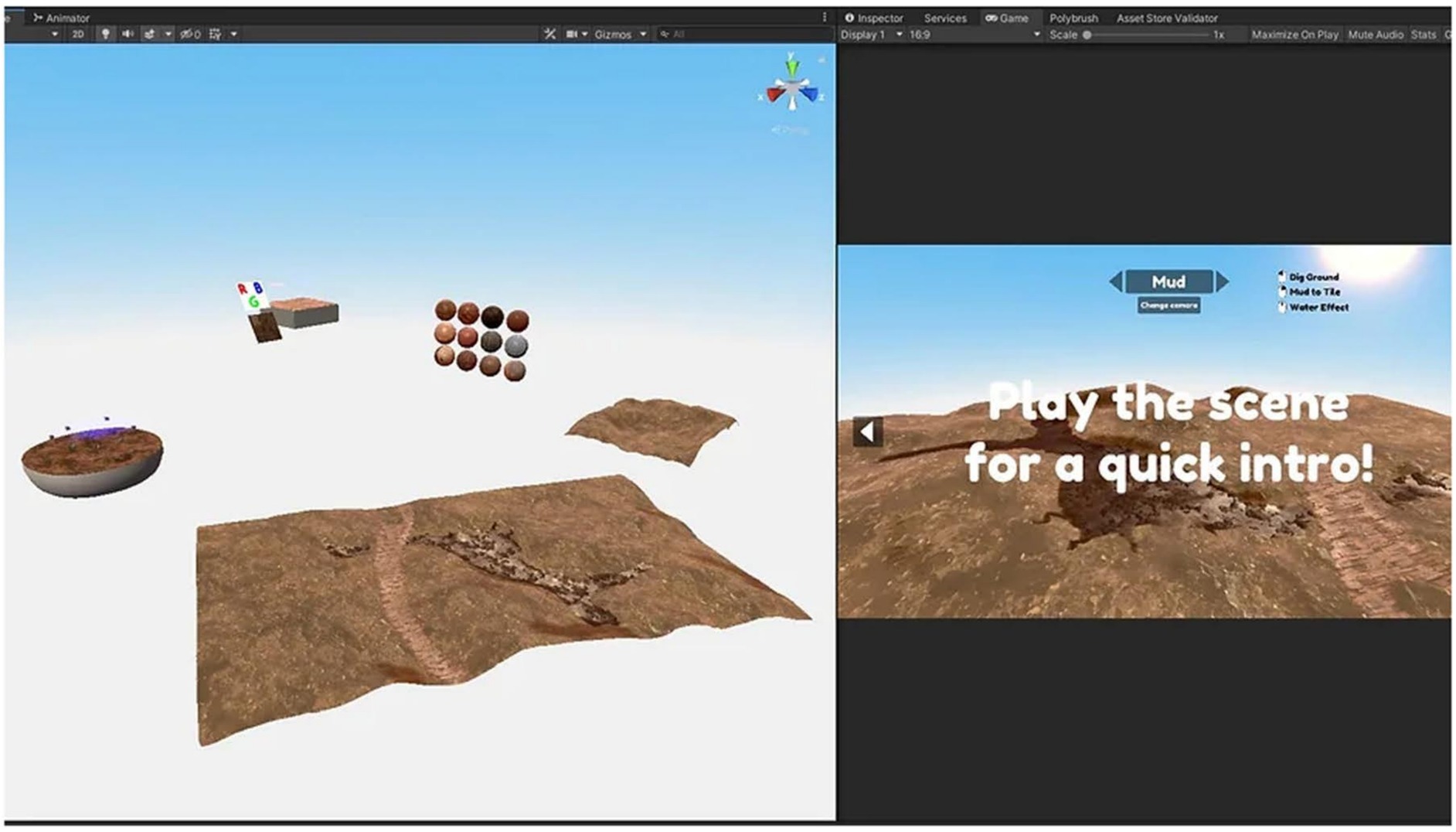
3. Compatibility and Features



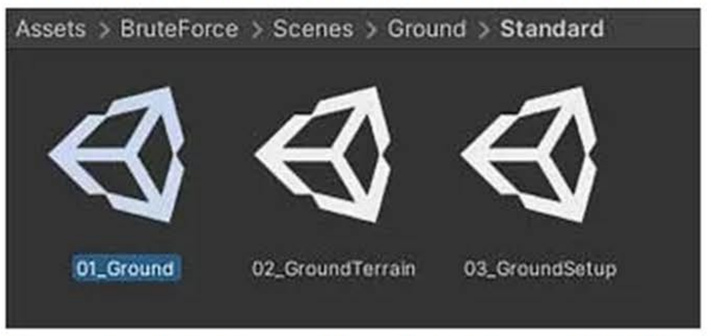
The **Brute Force Ground Shader** is a unique solution for general grounds and tilings such as **soil** or **walls**. It offers dynamic interactivity at minimal performance cost, create **streams of water** or **footprints** in wet grounds.
Works on **PC/Mac/Linux/Nintendo Switch** and any **mobile** devices.
Compatibility with **Unity Standard** and **URP**

Features:
-**Mutlicompatibility**: works on PC/Mac/Linux/Nintendo Switch/WebGl and mobile devices
-**Simplicity**: Drag and drop materials
-**Interactivity**: With a simple particle system you can create interactive effects like a trail on mouse/player position
-**Water Effect**: Based on the interactivity feature create water streaming effect flowing based on mesh normals, instantiate footprints or hit effects
-**Tessellation**: Custom tessellation based on camera distance
-**Dynamic Custom Shadows**: Sand featuring custom colored shadows and shadow casting
-**Customization**: Fully customizable look, render the ground with a single material
-**Lights Support**: 4x point lights and spot lights support (with shadow casting)
-**Terrain**: Use texture Splat Map to draw ground terrains, the asset features a compatibility tool to work with other terrain assets.
-**VR Support**: Works with Multi and Single Pass Stereo rendering
-**Compatibility:** Great Compatibility with my other assets

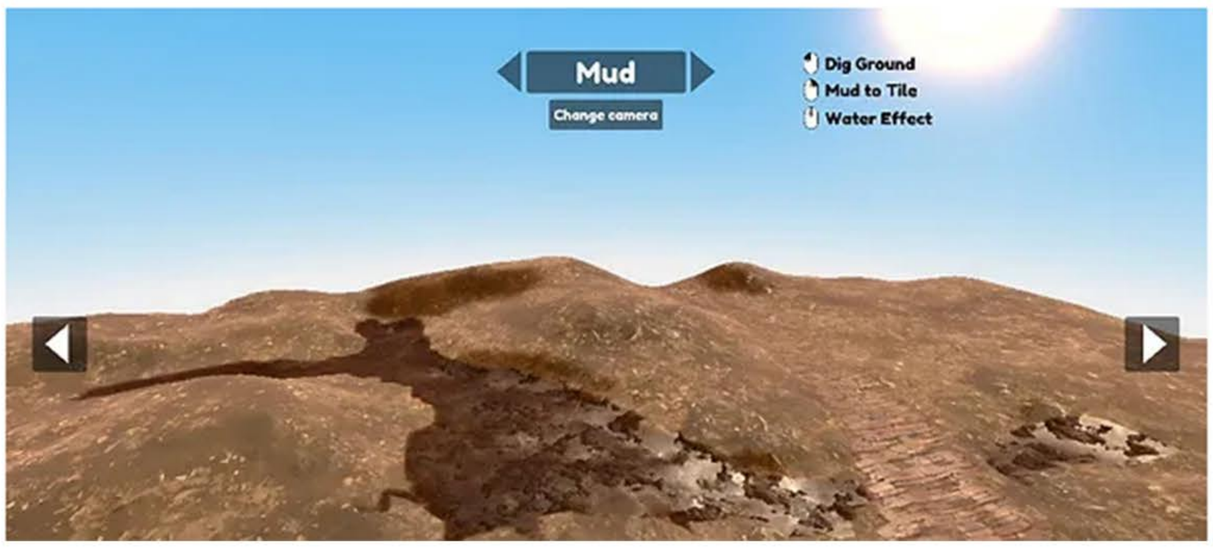
4. Getting started



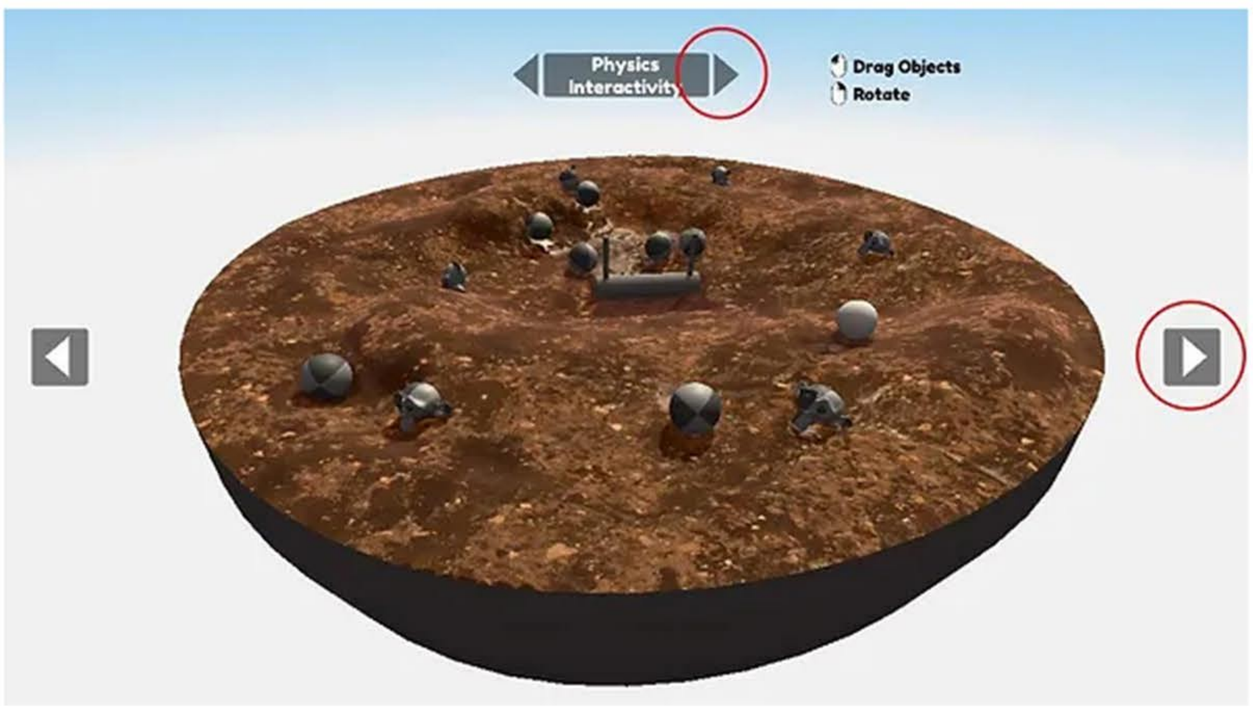
First thing you should do is load up the ground demo scene inside the asset package
"**Assets\BruteForce\Scenes\Ground\...**"
You will see that some files have a Standard and URP version, choose accordingly.



Once the "**01_Ground**" or "**01_GroundURP**" is loaded you will be prompted to play the scene for a quick introduction. This will help you understand the asset better and showcase all its basic features in a complete environment.
In the first showcase you can use **Left Click** to dig the ground, **Right Click** to change mud to tile, **Middle Click** to create water effects



You can click the top arrows on the screen to switch sub-showcases and the left and right arrows to switch showcases.

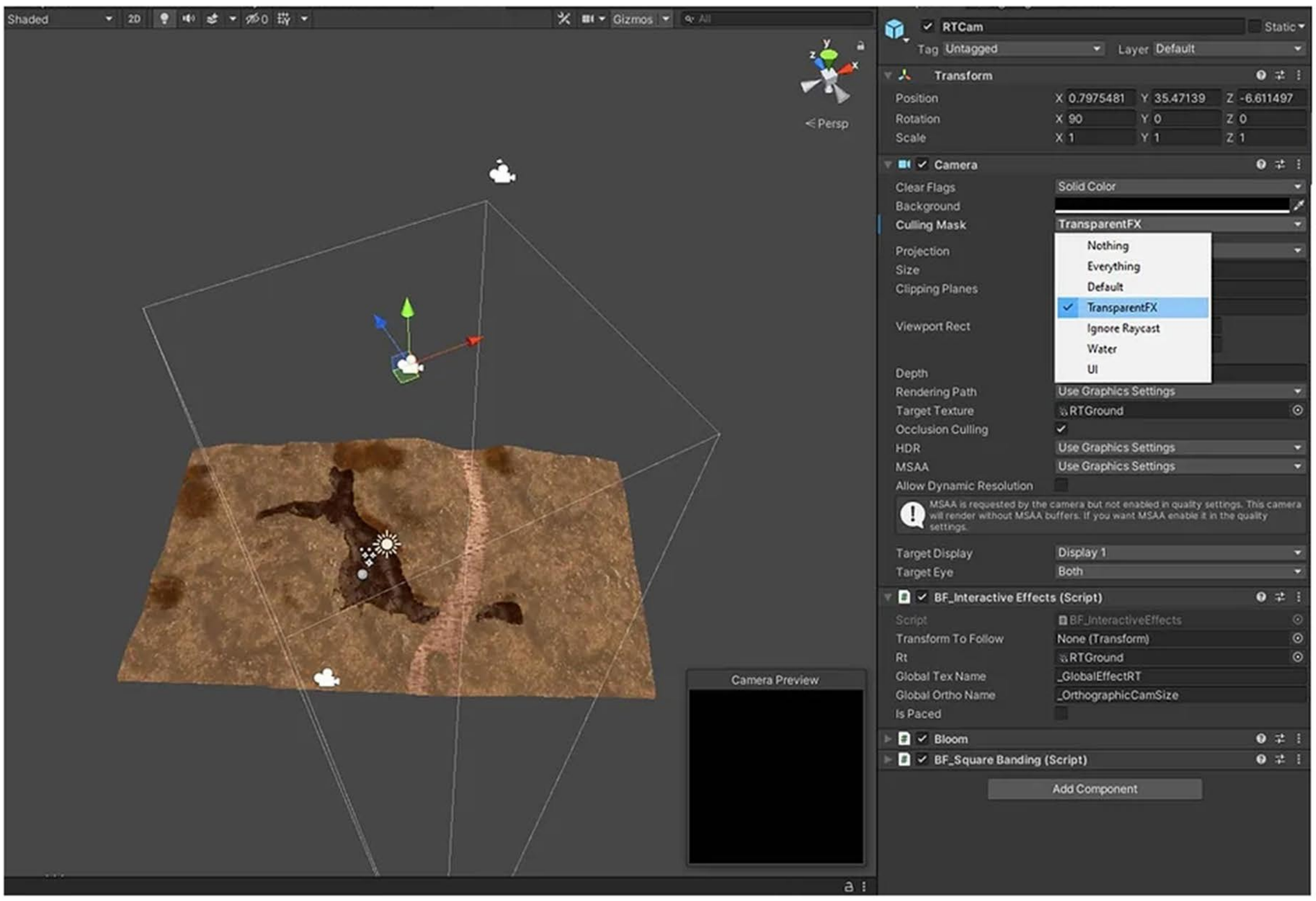


There is a player template that leaves footprint as well as other dynamic objects leaving traces in the mud that you can use to easily replicate the effects on your player.

Importing it in your project and Components

To append the ground assets in your projects you can either copy and import one of the working **showcase** as a starting template or you can use the **setup prefab**. Keep in mind that if you need the interactive feature you have to append the **CameraEffects** in your scene. I will go into details on what components you need in the prefab for the ground to work properly:

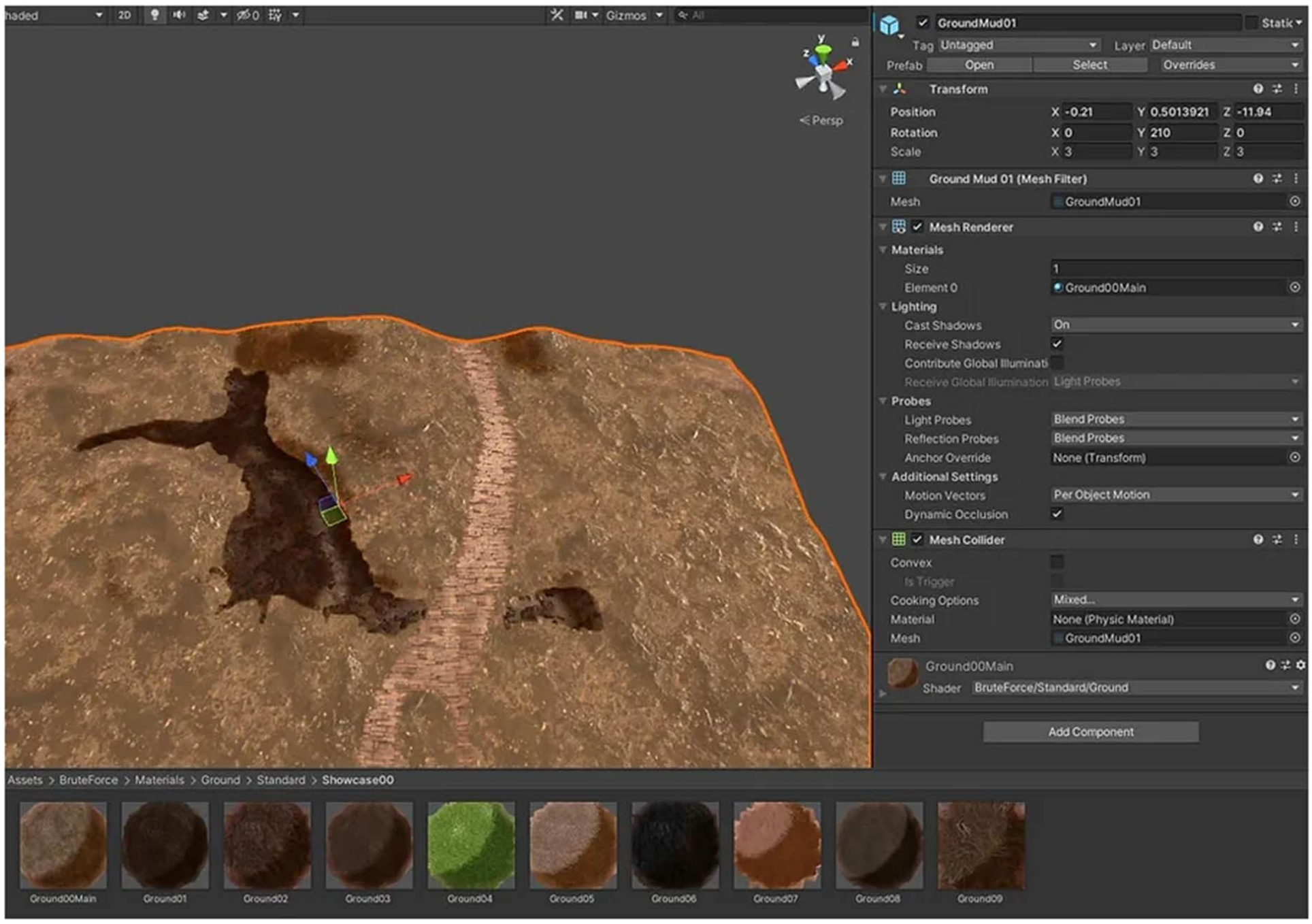
Camera Effect



The first main component you'll need if you want **interactivity** on your ground are the "**Camera Effects**" like a regular camera it will be used for rendering, but instead this one will render to a target texture namely: "**RTGround**" and "**RTGroundAdditional**" and will have only ONE culling mask set to "**TransparentFX**", we only need to render the effects.
It needs to have a target texture set to a render texture and it also needs the script "**BF_InteractiveEffects.cs**" for the first one and "**BF_InteractiveEffectsAdditional**" for the second one (which is optional, more details below). This script will make sure your cameras will render to the render textures.
If you have a moving camera you'll need to set a transform to follow (preferably the camera). RTCamAdd MUST have your main camera as transform value.
If you don't know what the global names are do not change them.
If you do not need the interactive part of the shader you can disable the CameraEffects and the "**Use RT**" in the material pragmas for a small performance boost.

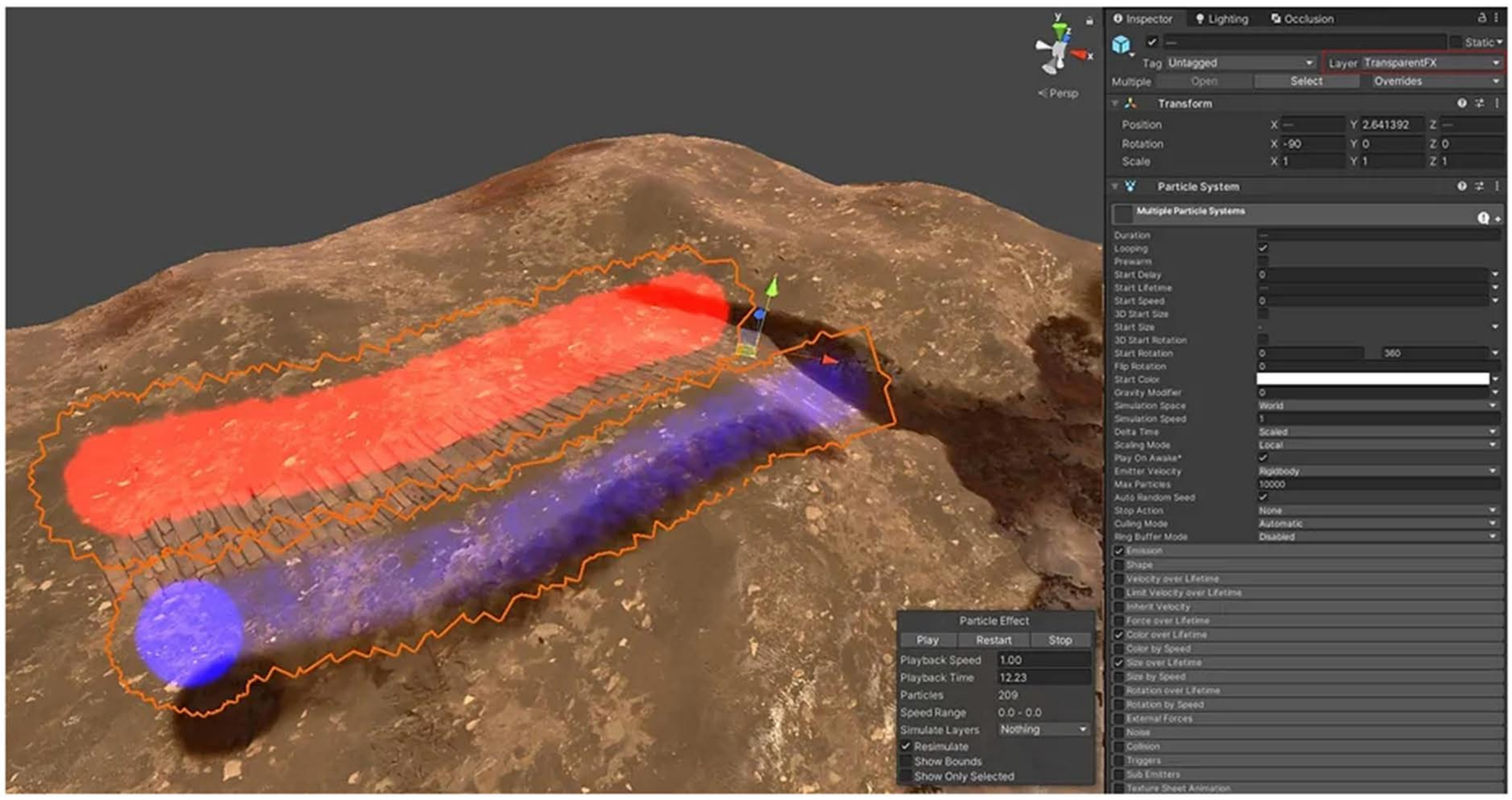
You can increase the size of the camera projection to increase the radius of the rendered effect at a precision cost (more details below), do not rotate the RTcamera.

Ground Material



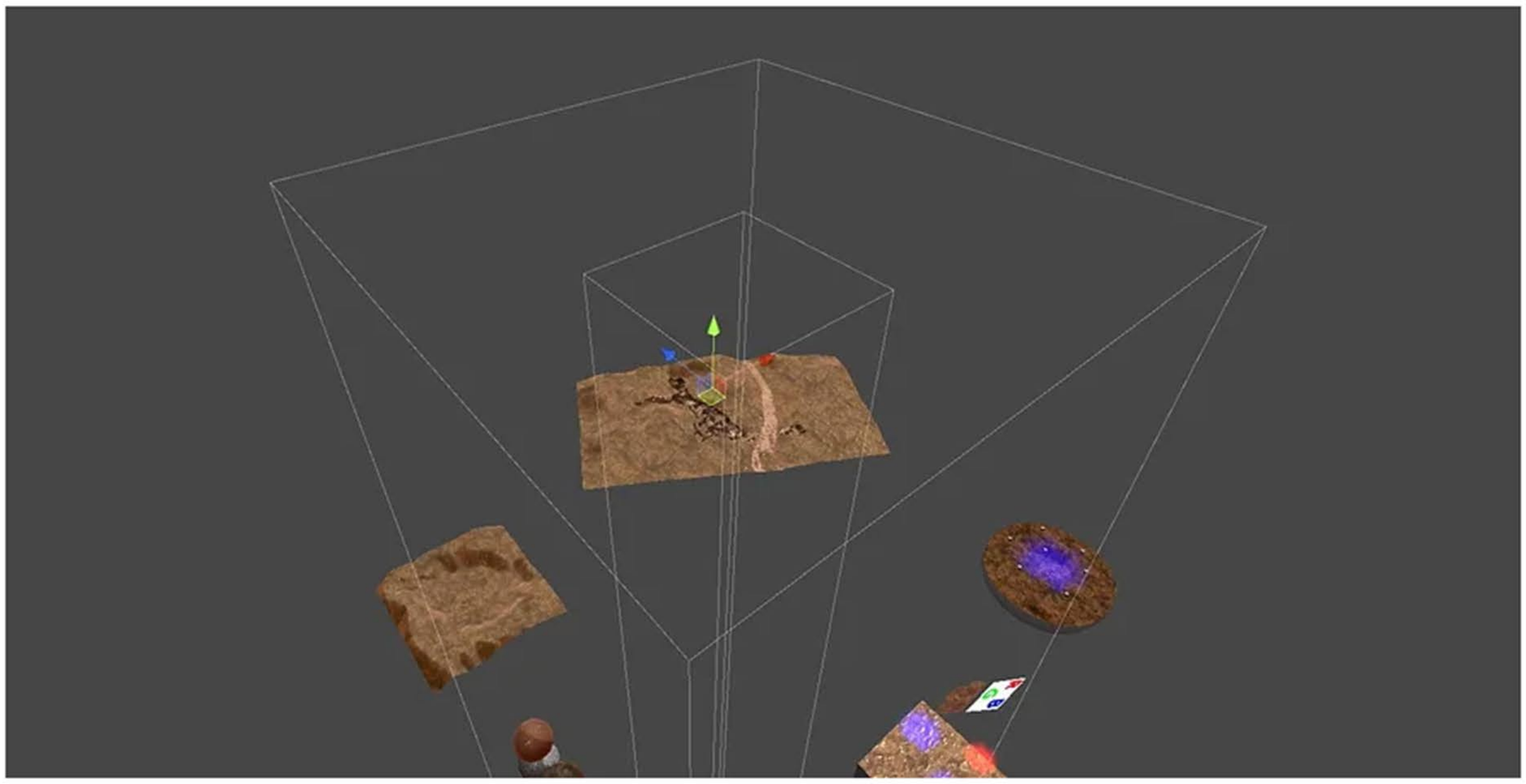
You can drag and drop any ground materials to a mesh or any assets you want to use as ground, I will explain in more depth how the [Ground Shader]() works in the next page so you can be able to customize or create a ground material for your projects.

Particle System Effects

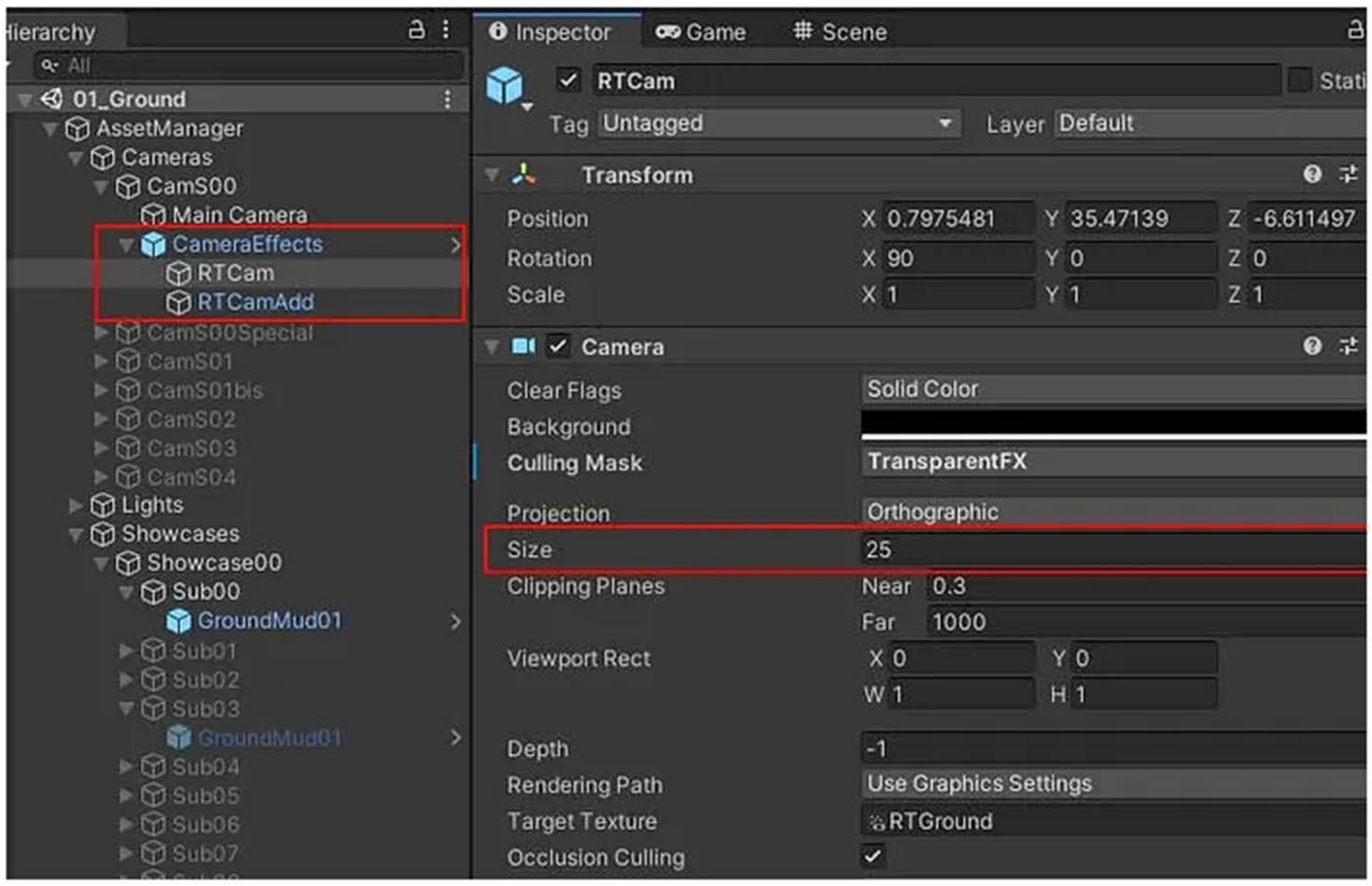


To control the interactive effect of the ground you can use a particle system set on the **TransparentFX layer**, the important thing to note is that only the rendered color matters:
Red color will turn Mud to Tile, **Blue** color will dig the sand and create water, **Green** color will disable vertex displacement
You can use anything that can be rendered to achieve the desired effect: meshes, sprites, particle systems etc...
This means you can place a **colored quad** in the **TransparentFX** layer to have a **permanent ground effect** in your scene.
Alternatively you can edit the "Duration" and "Start Lifetime" parameters of the particle system to either shorten or extend the effect duration.

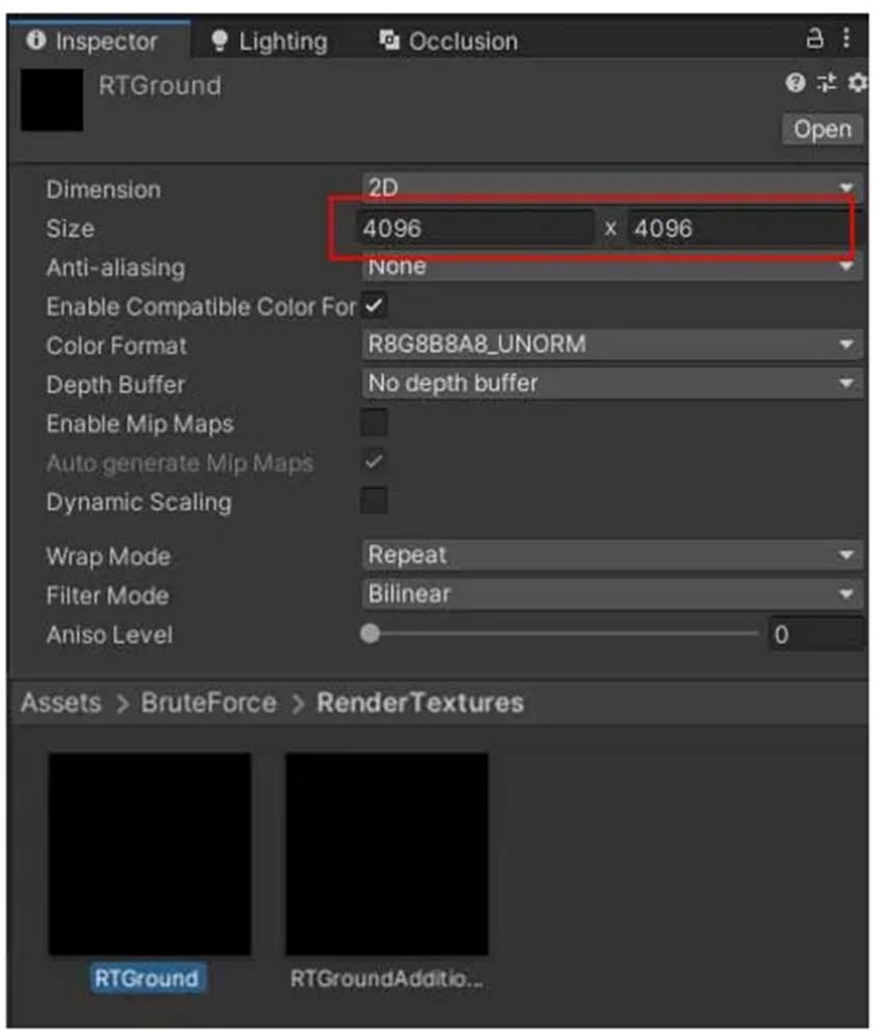
Increasing the Render Texture size



Depending on your project, you can increase the size of the Render Texture field so you would be able to see sand effects from much further away.
To do so you'll need to select your **RTCam** in your scene and increase the "Size" Camera variable, keep in mind that increasing this value will decrease the effect resolution.

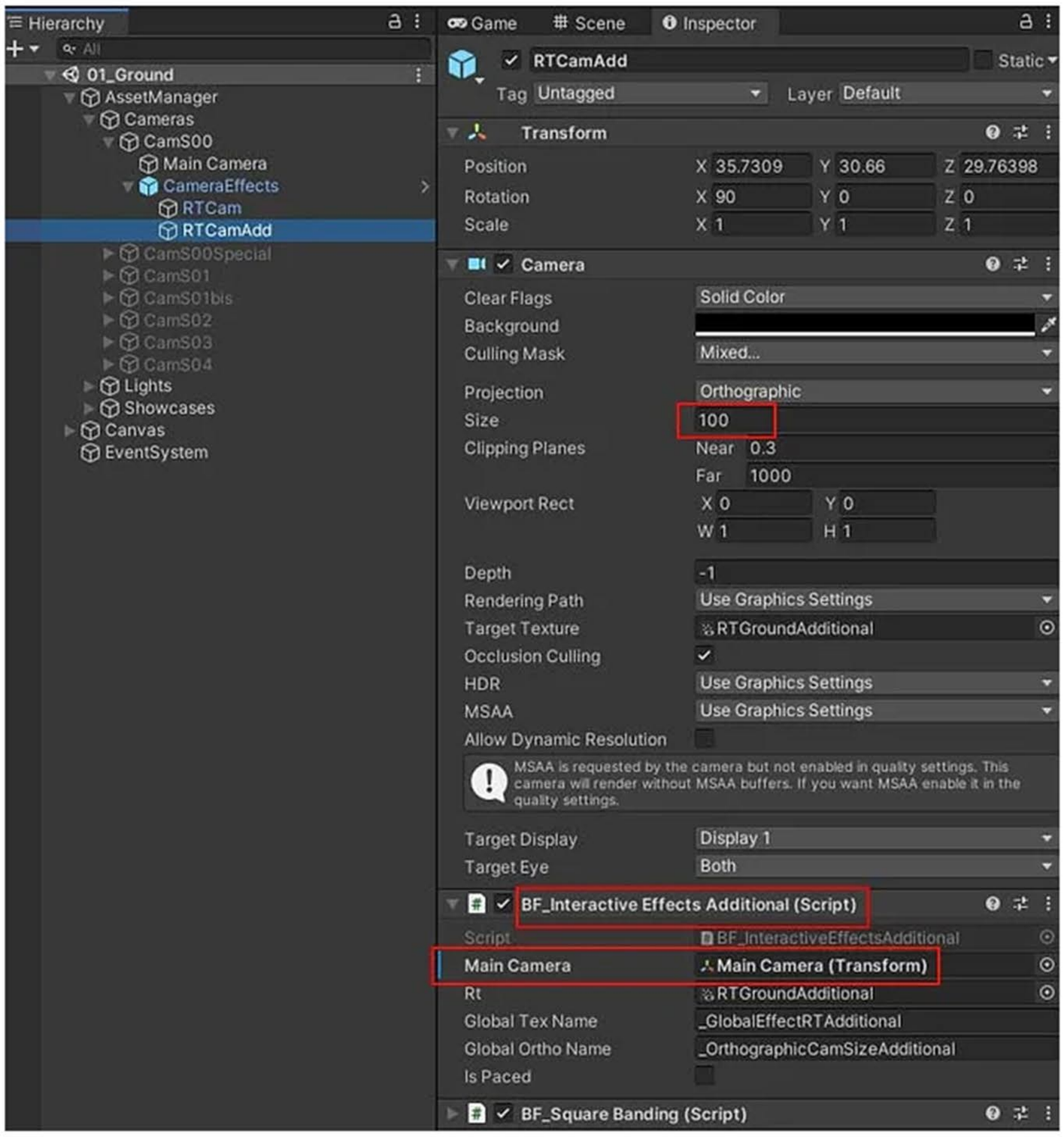


To increase the effect resolution you can change the "**Size**" of the **RenderTexture** to 2048x2048 or 4096x4096 by selecting the "**RTGround**" in your project. Changing this value will greatly increase memory consumption.



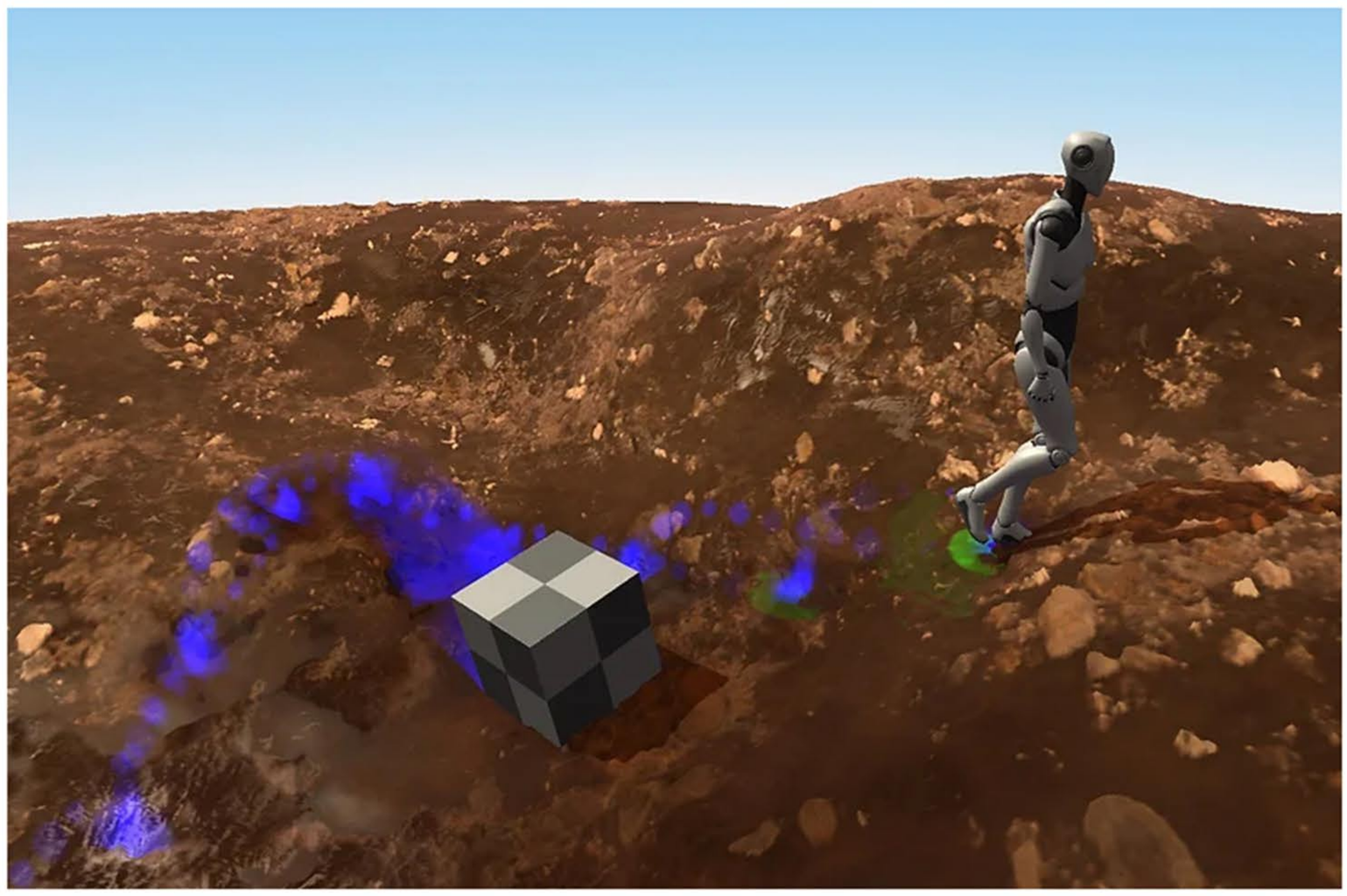
Using the RTCamAdd to expand the Ground effect area

In addition to the RTCam I created an experimental additive RenderTexture effect which is much larger. This camera is optional but its use is highly recommended if you intend to use the effects on a large area.



When you assign the main camera to the "Main Camera" field in the **BF_InteractiveEffectsAdditional** script; the RTcamAdd will automatically change its position based on the main camera FOV.

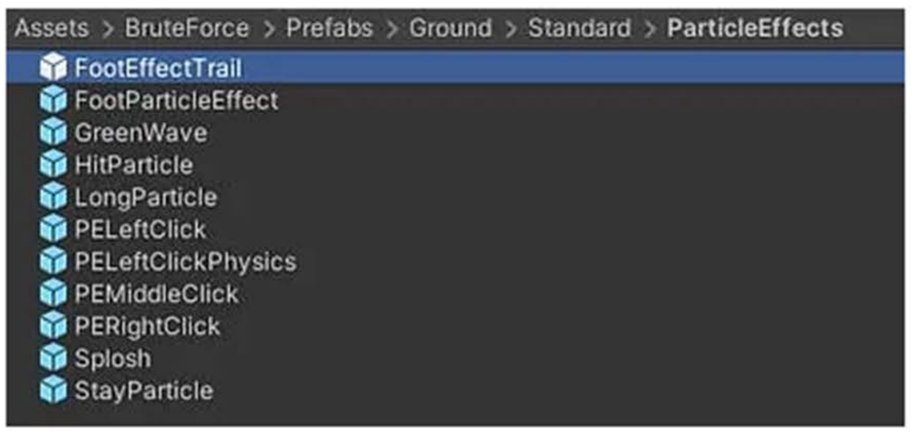
Import footstep effects on your character



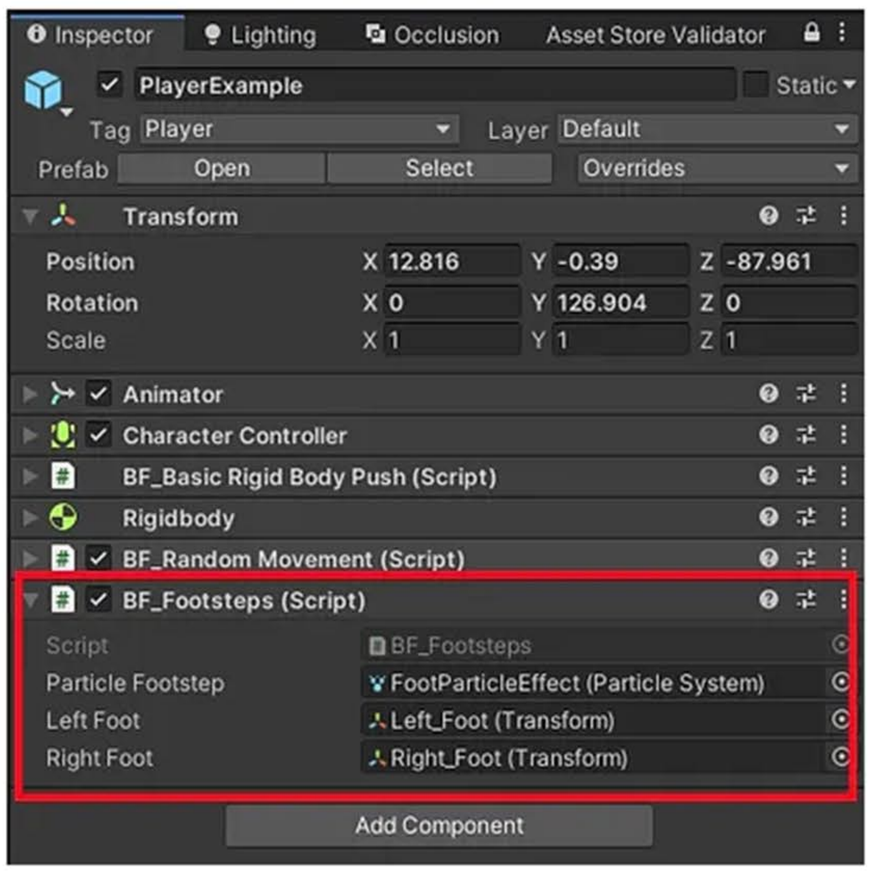
There are 2 main ground effect templates you can add on your character that both have their unique advantages:

-The FootEffectTrail:

This effect will create continuous trail on the ground as if the foot was in the mud, it doesn't need any scripts to work and will work even without any ground contact.
You can simply drag and drop this one from the prefabs to the foot transforms of your character, then reset their local position to 0 so that they math the foot position. The FootEffectTrail prefab is located here:

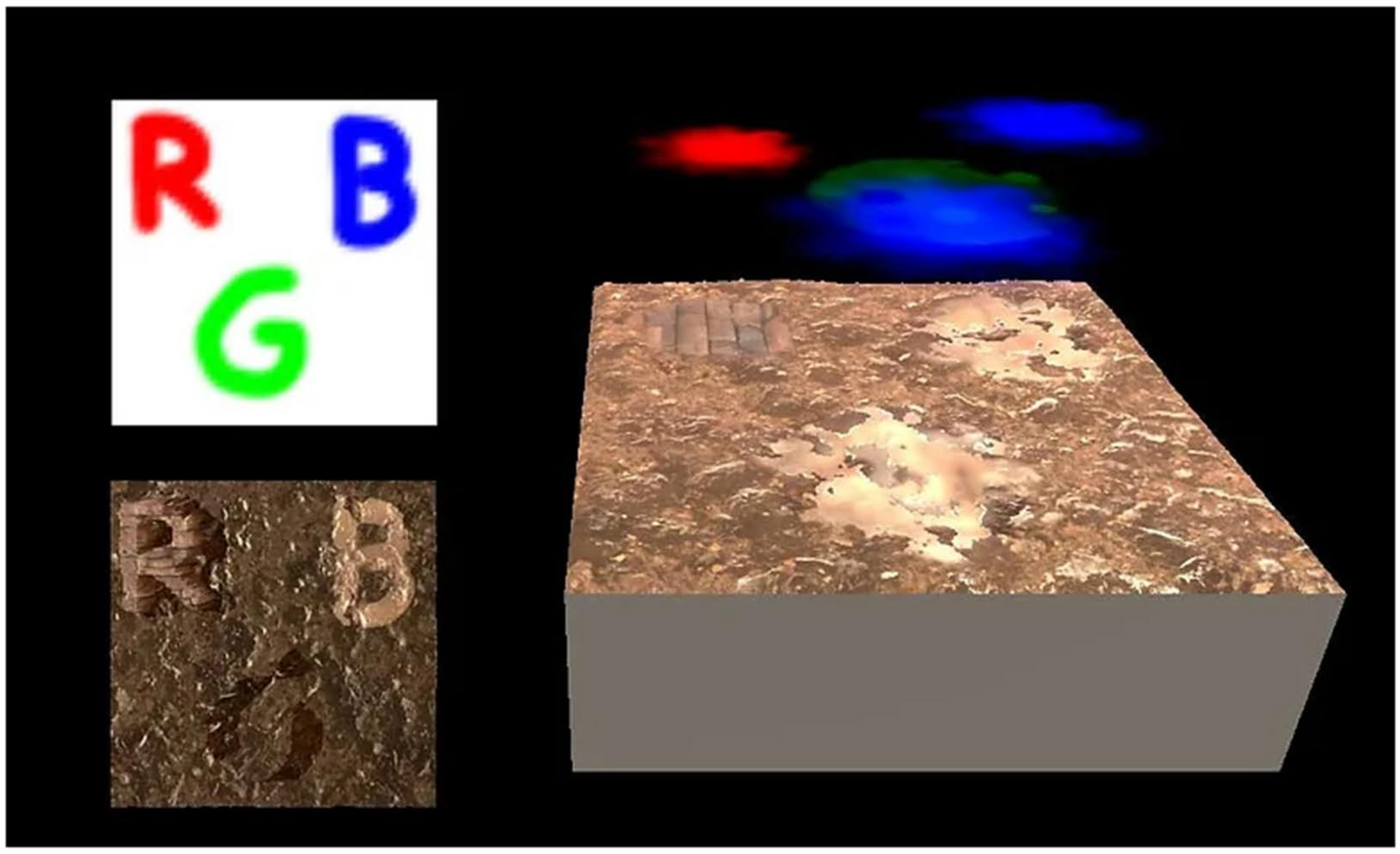


-The FootParticleEffect:



This one will instantiate a particle effect right where the foot hits the ground, for it to work you will need to add the "BF_Footsteps.cs" component to your player, then assign the "FootParticleEffect" prefab that's located in the prefabs folder or any custom particle system that will emulate footsteps. Then Assign **Left foot** to the left foot transform of your player and **Right foot** to the right foot, if your player has only 1 leg then I guess set both feet to the same one.

5. Paint Ground

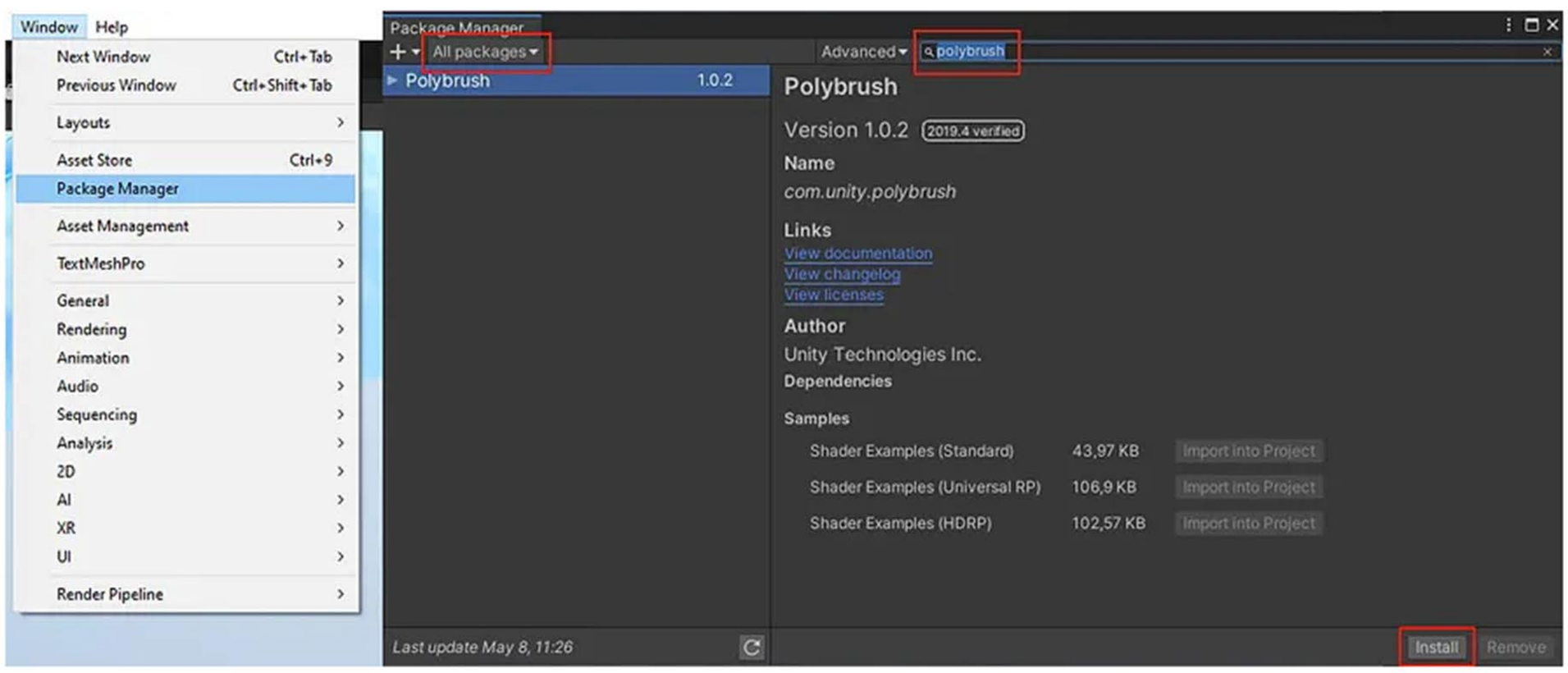


Using the Brute Force - Ground shader you can paint the vertex color of your assets to dynamically draw mud or tilings.
The default value and state of the shader is ground.
With that in mind let's see in details what each vertex color channel does:

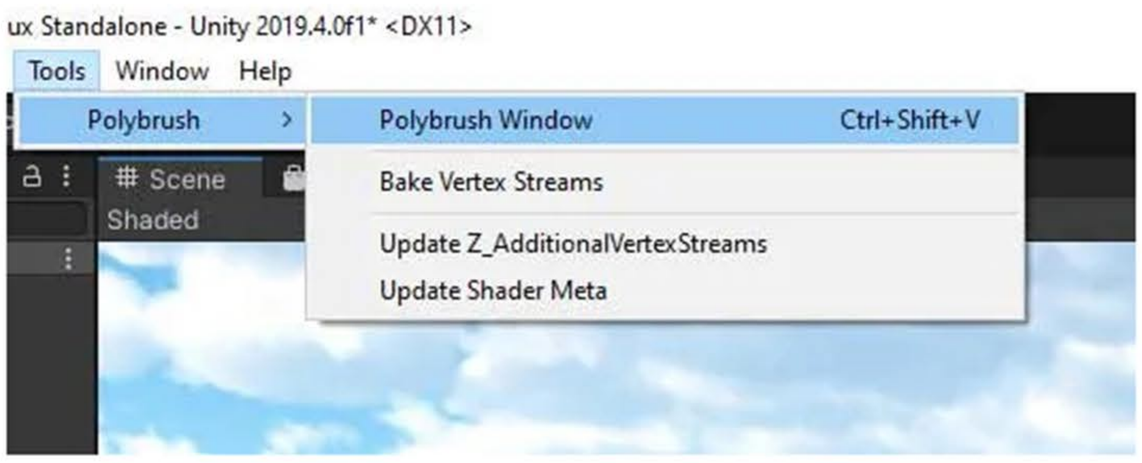
- **White**: Renders Ground
- **Red**: Renders Tiling
- **Green**: Renders Ground without displacement
- **Blue**: Renders Water effect (dug ground)

How to paint Vertex Color on your mesh

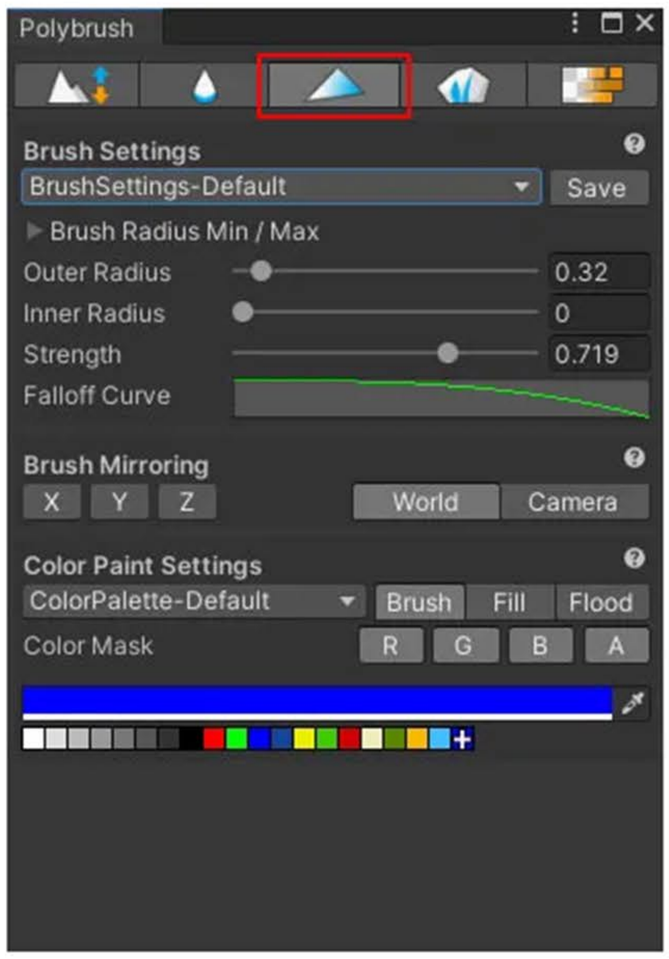
I strongly recommend you download the Polybrush Asset from the Package manager (see documentation). It's an official Unity asset that lets you draw colors on vertex. You can also use your favorite 3D software to draw vertex color and it will work fine.
For the rest of this page I will assume you are using the Polybrush asset.



Now that you've downloaded and installed **Polybrush**:

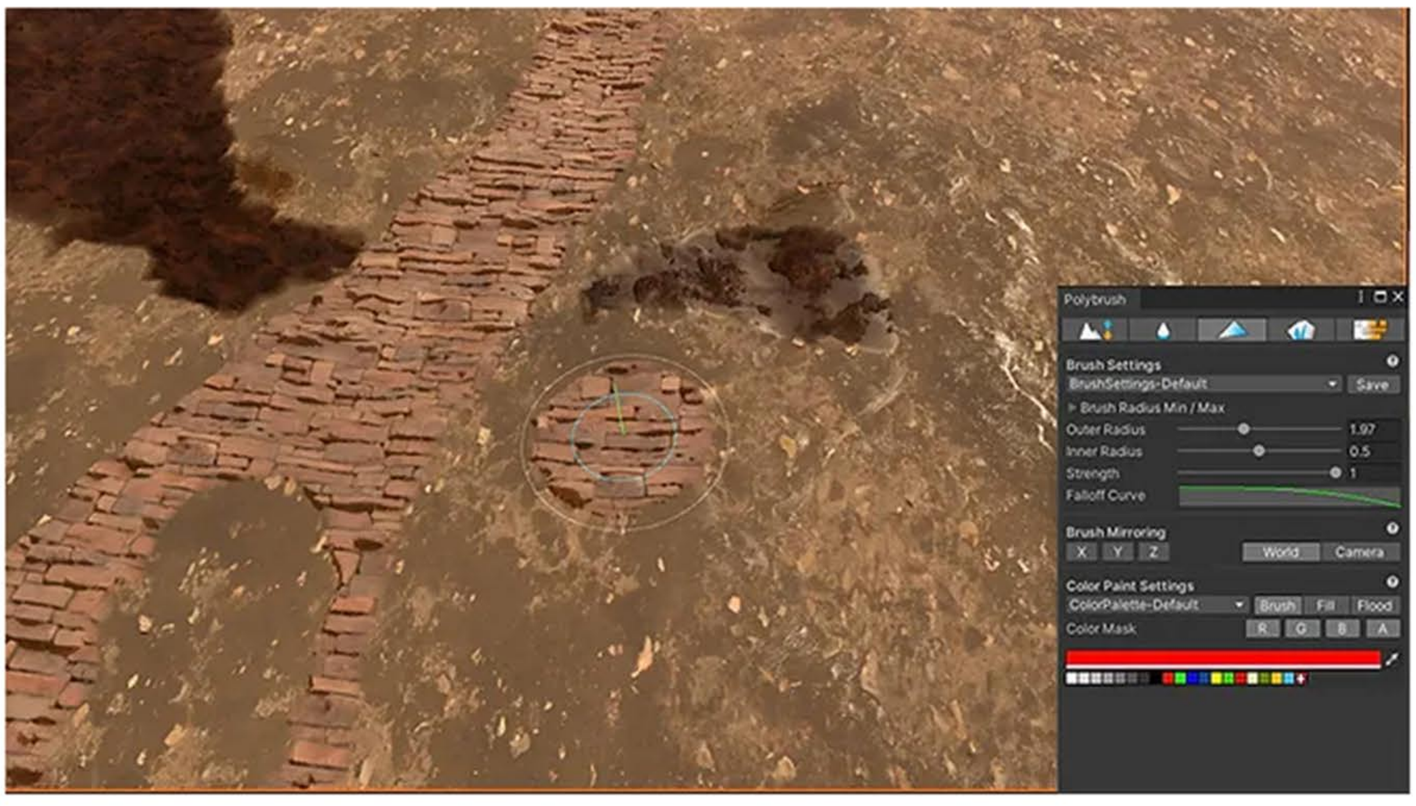


Open the Polybrush Window by going to: **Tools** > **Polybrush** > **Polybrush Window**
You will have this window opened:

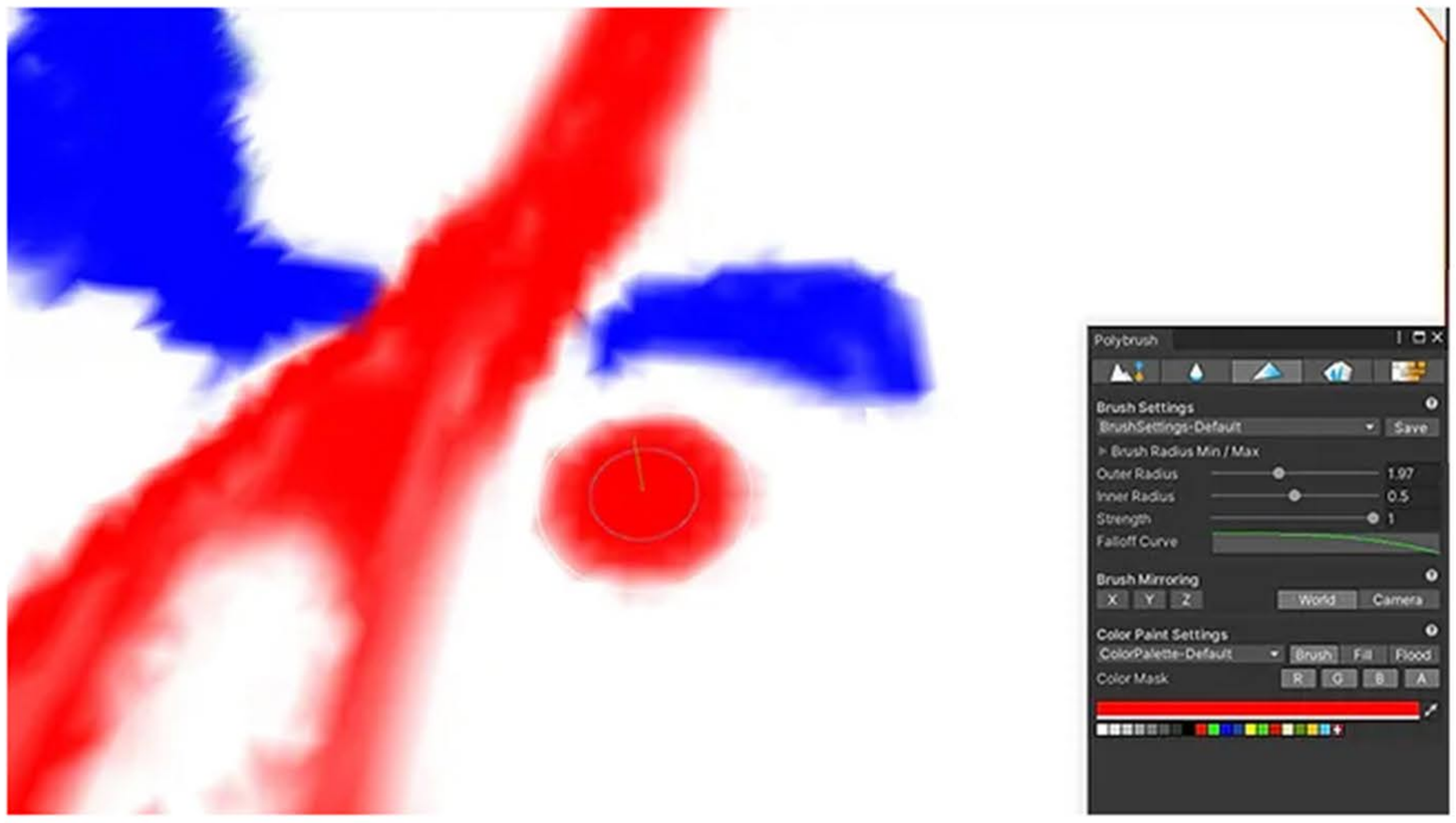


The tab that interests us is the "**Paint vertex colors on meshes**" click on that.
You can see there is a "Outer Radius" value that controls the radius of the brush size and a Color Paint Settings at the bottom.
You can choose which color you want to paint by clicking the big color field below color mask.
When choosing between White, Red, Green or Blue colors make sure it is the absolute color; meaning white is fully white (1,1,1) and red is fully red (1,0,0). Again, tead the polybrush documentation for more details.
Select red color:

Now click on the mesh you want to paint on with the ground material already placed, you can start painting Tilings:

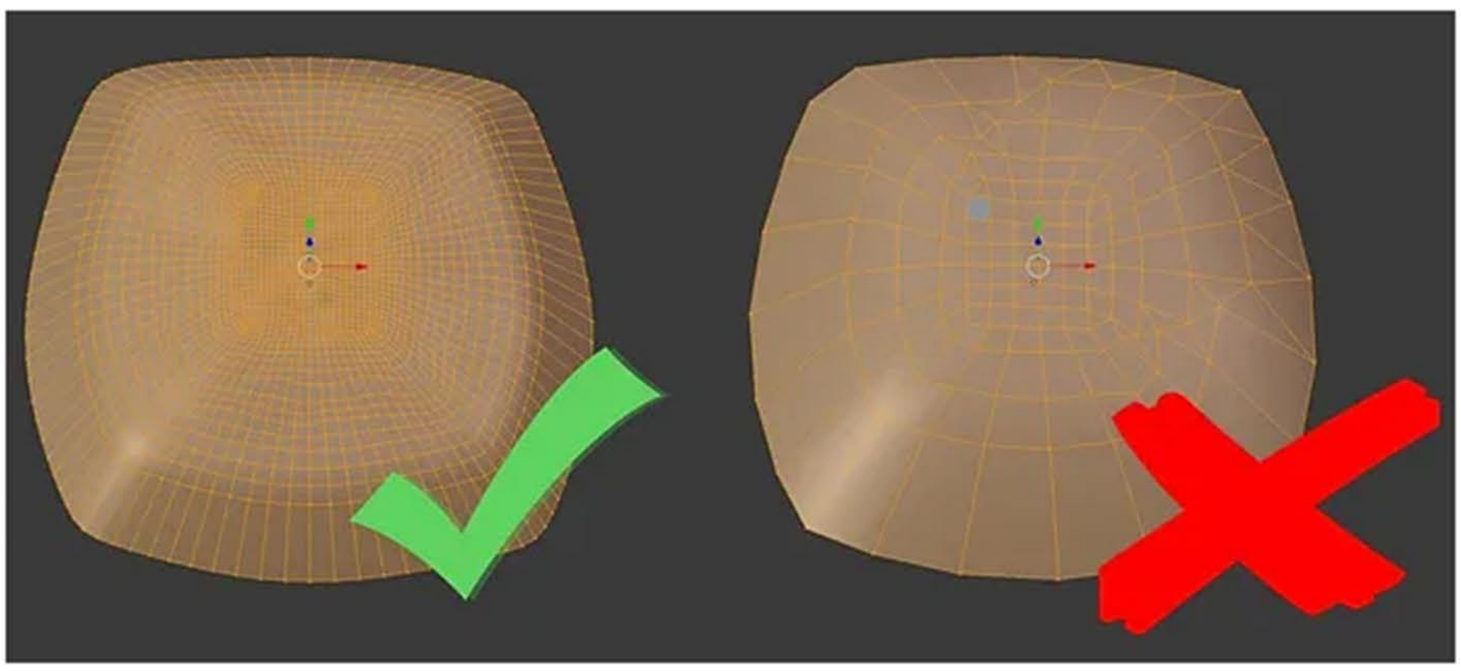


This is what you are actually doing by painting:

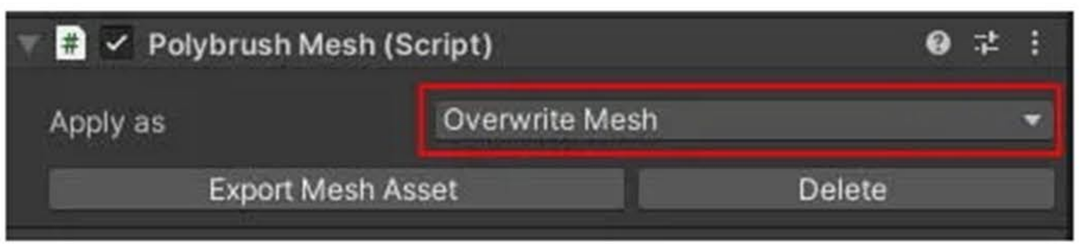


If you want to paint water you need to paint the vertex blue:

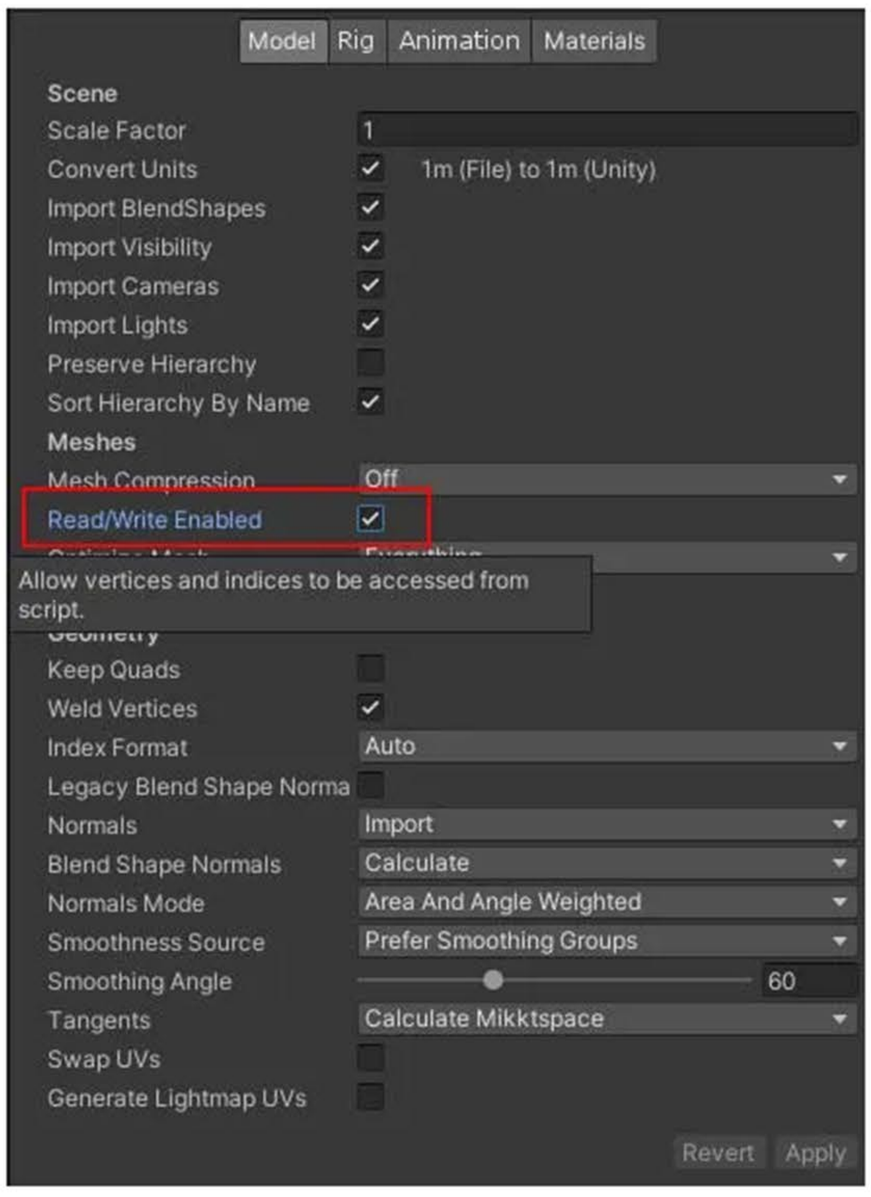
Keep in mind you need a certain **number of vertices** to have a higher effect definition, here's a good guideline:



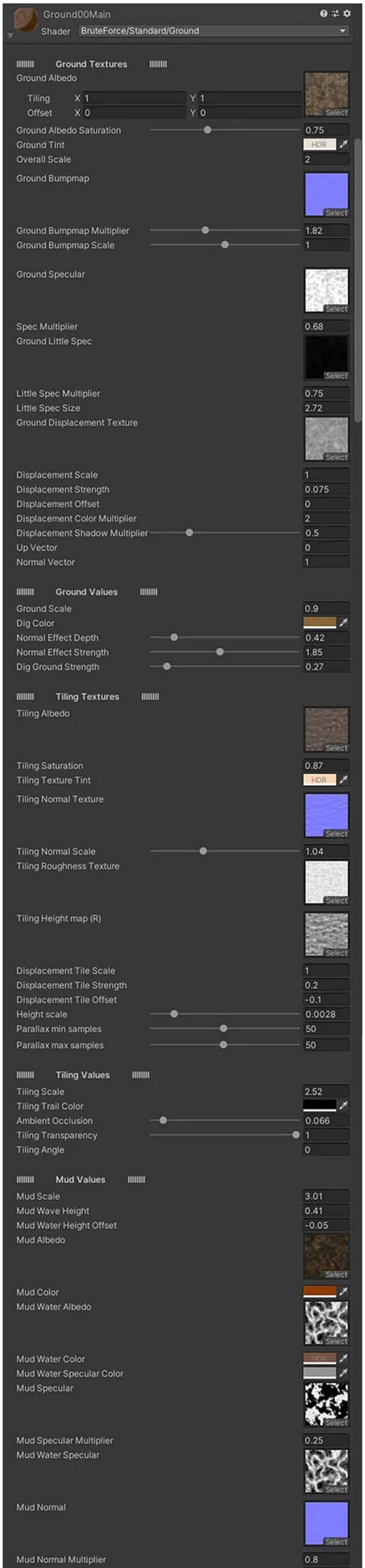
Finally when you are happy with your mesh make sure the "Apply As" is set to **Overwrite Mesh**

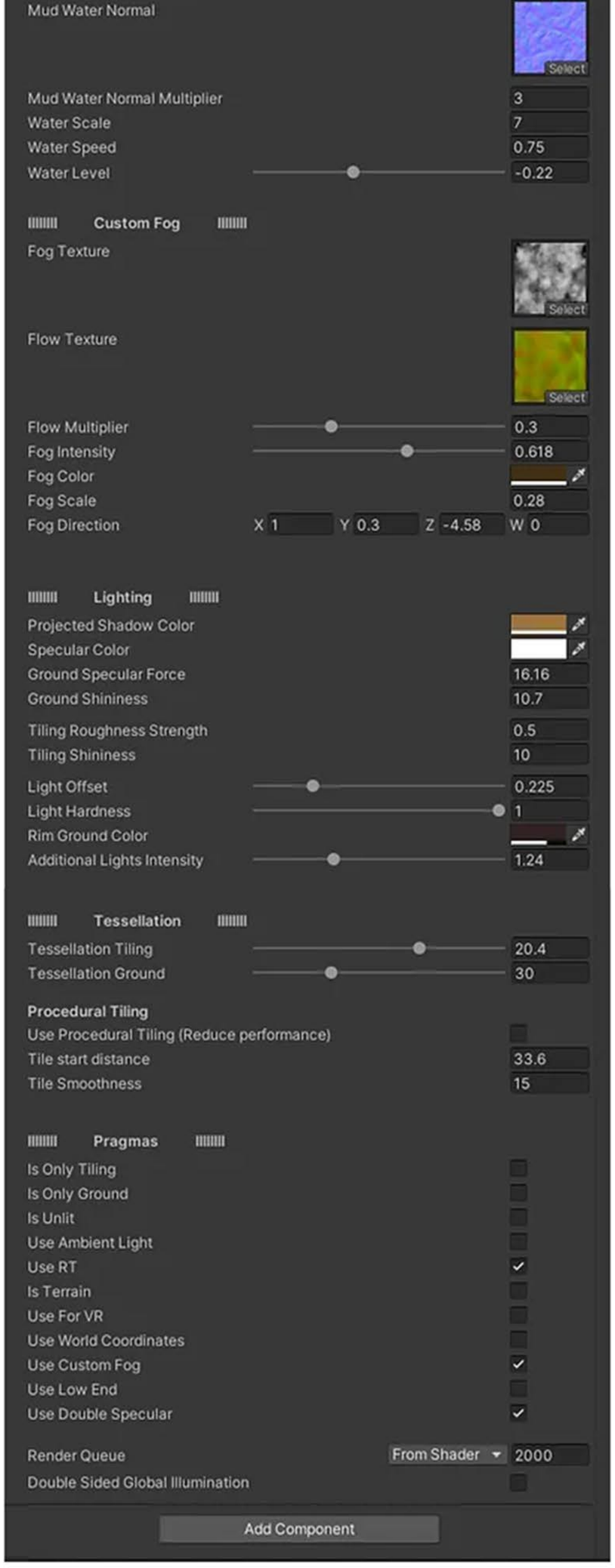


When importing a custom mesh inside Unity **you are required to enable Read/write** in the import settings of said mesh.



6. Ground Properties





The Ground asset is controlled entirely through the material property block, I suggest toying with the values to familiarize yourself with them. The material being very long I categorized the fields into:

Ground Textures || Ground Values || Tiling Textures || Tiling Values || Mud Values || Custom Fog || Lighting || Tessellation || Pragmas

I will describe the important parameters in order:

|| Ground Textures ||
- **Ground Albedo**: A colored texture to control the ground color
- **Ground Albedo Saturation**: The saturation value of the ground texture
- **Overall Scale**: The scale parameter controlling every texture coordinates except the ground displacement
- **Ground Little Spec**: The B&W texture controlling the specular intensity where the light reflects on the ground surface
- **Ground Displacement Texture**: The texture controlling the displacement of the ground vertex
- **Displacement Offset**: The value differentiating the height between ground and tiling, 0 = no differences in height
- **Displacement Color Multiplier**: The parameter multiplying the value(color) difference based on displacement height
- **Displacement Shadow Multiplier**: The parameter multiplying the shadow value based on displacement height
- **Up Vector**: The higher this value is the more the displacement will move towards the world upward vector
- **Normal Vector**: The higher this value is the more the displacement will move towards the vertex normal

|| Ground Values ||
- **Ground Scale:** The scale parameter controlling the ground texture coordinates
- **Dig Color**: The color tint of the dug up ground based on interactive effect (blue)
- **Normal Effect Depth**: The multiplier offset of the interactive effect, does not scale with the RenderTexture dimension if you do not understand what it does leave it between 0.10-0.20
- **Normal Effect Strength**: The value multiplying the overall normal interactive effect (Visual only)
- **Dig Ground Strength**: The vertex displacement multiplier (Vertex position only)

|| Tiling Textures ||
- **Tiling Albedo**: A colored texture to control the tiling color
- **Height Map**: The texture controlling the parallax occlusion effect, white = no height, black = deep height
- **Height Scale**: The multiplier of the parallax effect
- **Parallax min/max samples**: The sample definition of the height effect, lower it for better performances

|| Tiling Values ||
- **Tiling Scale:** The scale parameter controlling the tiling texture coordinates
- **Tiling Trail Color:** The tint of the interactive effect on the tiling (color is reversed; white = very dark tint trail)
- **Tiling Transparency:** Parameter used by the additive component and the terrain script, sets the rock alpha, in most cases you do not want to touch this

|| Mud Values ||
- **Mud Scale:** The scale parameter controlling the mud texture coordinates, here "mud" refers to the dug ground effect, when you interact with the ground with a blue colored effect you get mud.
- **Mud Wave Height:** The max height a water wave can reach (based on local position)
- **Mud Water Height Offset:** The offset value to offset the water height (based on local position)
- **Mud Albedo:** The color of the ground mud
- **Mud Water Albedo:** The color of the water in the mud
- **Water Scale:** The scale of the water textures
- **Water Speed:** The speed of water scrolling
- **Water Level:** The value at which the water is supposed to be levelled, -1 = no water / 0 = water barely below ground surface / 1 = water flood

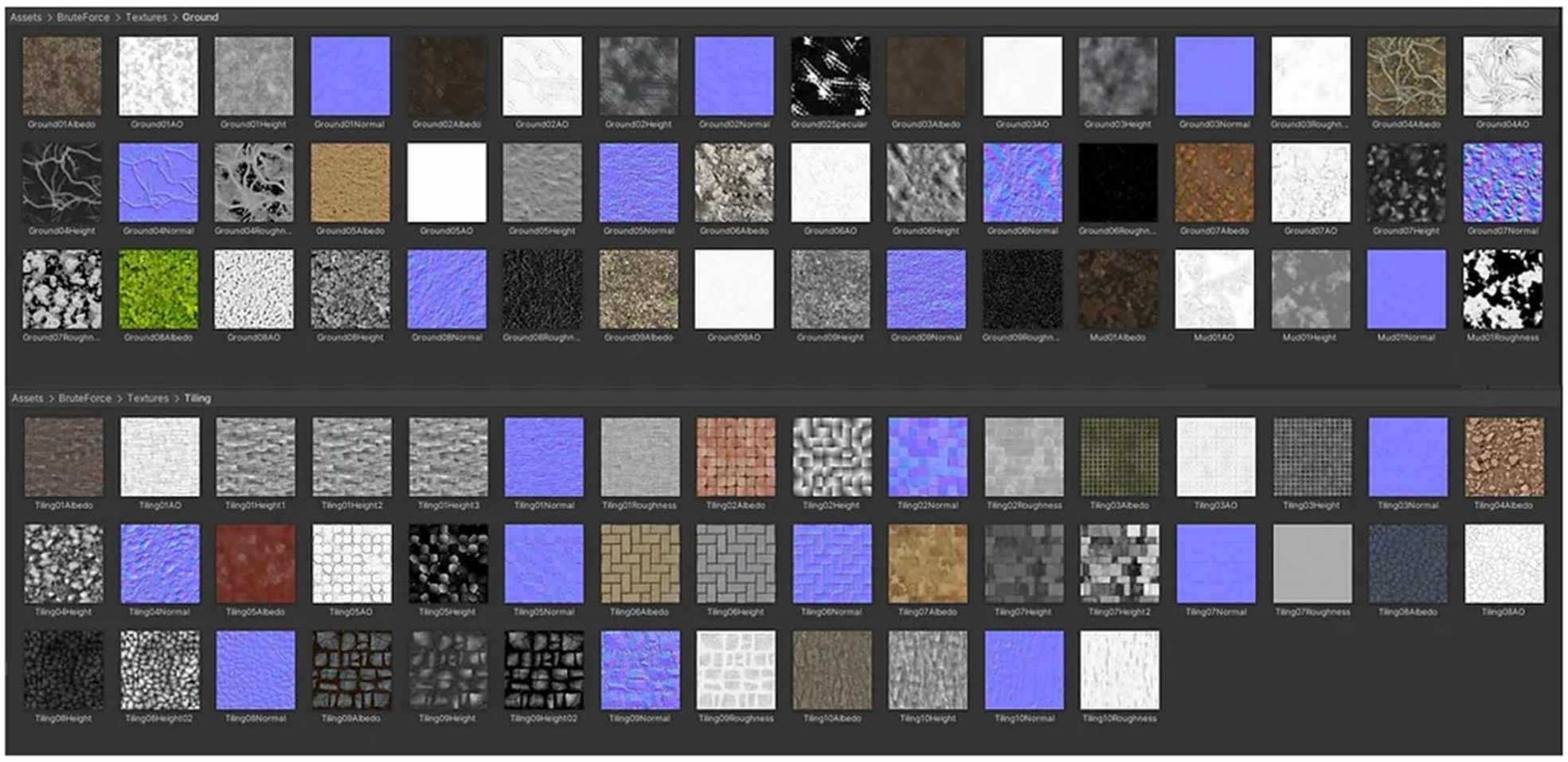
|| Custom Fog ||
- **Fog Texture:** The fog pattern texture that will render above the ground (if you don't require custom fog then disable it in the pragmas)
- **Flow Texture:** The flow texture of the fog (how it will move)
- **Flow Direction:** The overall movement direction of the fog

|| Lighting ||
- **Projected Shadow Color:** The color of the ground & tiling shadows
- **Light Offset/Hardness:** Range values to control how the light gets rendered on the material
- **Rim Ground Color:** The tint color of the ground rim light (alpha controls the scale)
- **Additional Lights Intensity:** The additional lights (points, spots) intensity multiplier, only works with additional lights

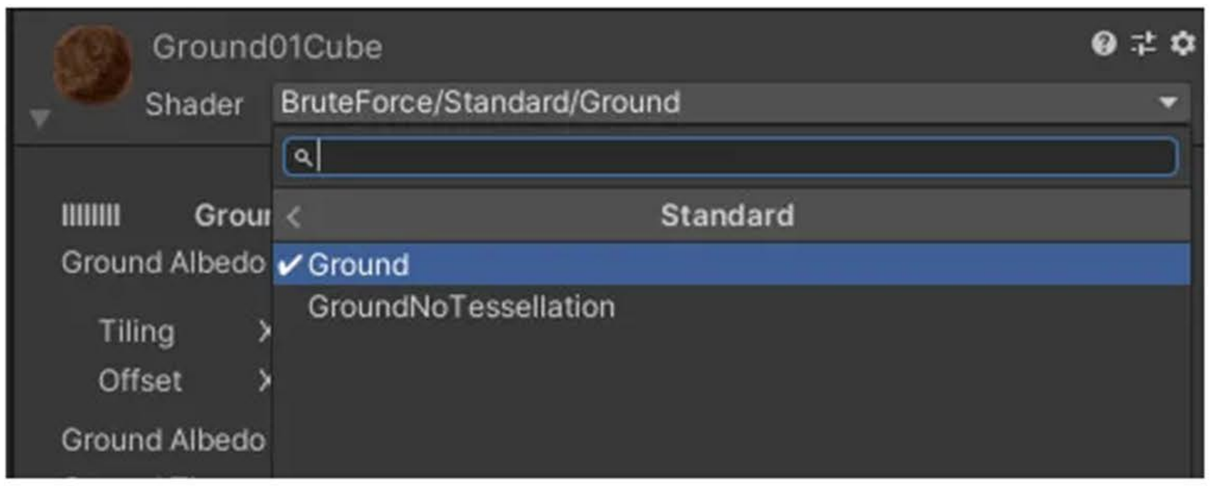
|| Tessellation ||
- **Tessellation Tiling:** The tessellation control value of the tiling, 1 = No Tessellation | 30 = maximum Tessellation, in the majority of uses you SHOULD leave it below 20
- **Tessellation Ground:** The tessellation control value of the ground, a good optimized value should be between 20 and 50

|| Pragmas ||
- **Is Only Tiling:** If enabled, will turn the whole mesh into tilings independently from vertex color
- **Is Only Ground:** If enabled, will turn the whole mesh into ground independently from vertex color
- **Is Unlit:** If enabled, will disable all lighting calculation from the material
- **Use Ambient Light:** Enables the use of ambient scene lighting
- **Use RT:** If enabled, will activate the interactive feature, disable it if you don't intend to use the interactive part
- **Is Terrain:** If enabled, will activate the Terrain feature, do not enable this feature by itself instead you should use the Ground Terrain component
- **Use For VR:** Enables the compatibility with single pass rendering for VR
- **Use World Coordinates:** If enabled, will project the ground textures in world coordinates independent from mesh uv, turn it off if you want to use your mesh uvs
- **Use Custom Fog:** If enabled, will enable the custom animated fog overlaying the ground
- **Use Low End:** If enabled, will disable every process intensive shader features for low-end hardwares. Enable this feature to get a performance boost
- **Use Double Specular:** If enabled, will project 2 opposite suns onto the specular shading, disable it to render only 1 sun specular

You can switch textures around or try different materials from the large collection in the package:
You can create your own textures for any texture slots

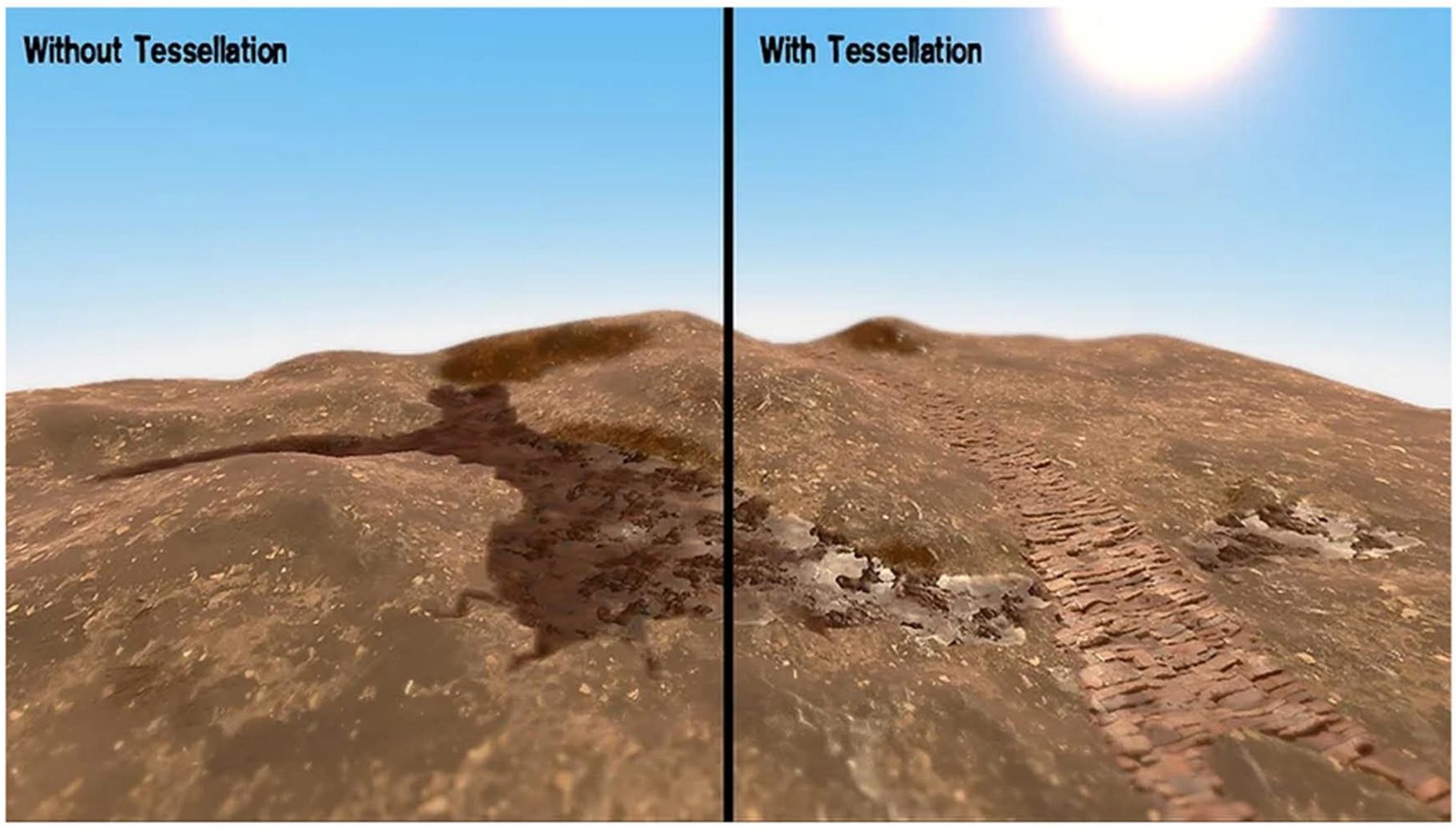


If you want to create your own materials here is the list of the Ground shaders:

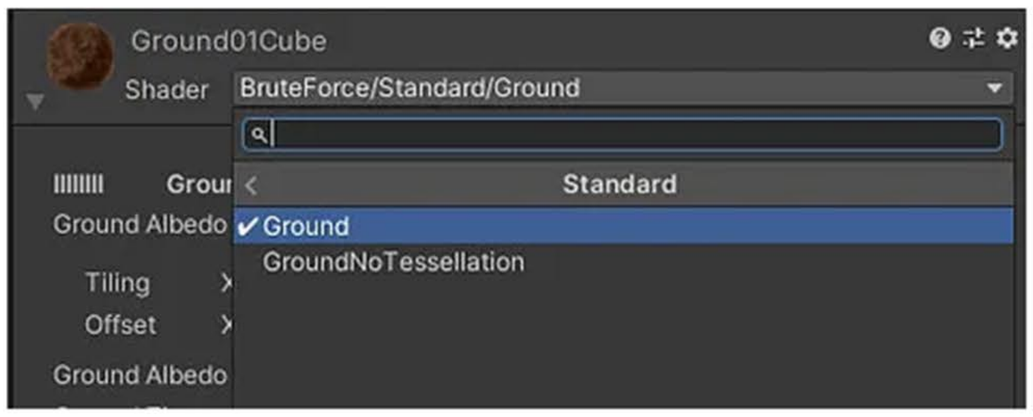


Ground: The standard Ground shader
GroundNoTessellation: The same as the above but without the vertex tessellation feature, use this variant to build for WebGl or low-end mobile; [see more about WebGl and Mobile Build]

7. WebGl and Mobile Build

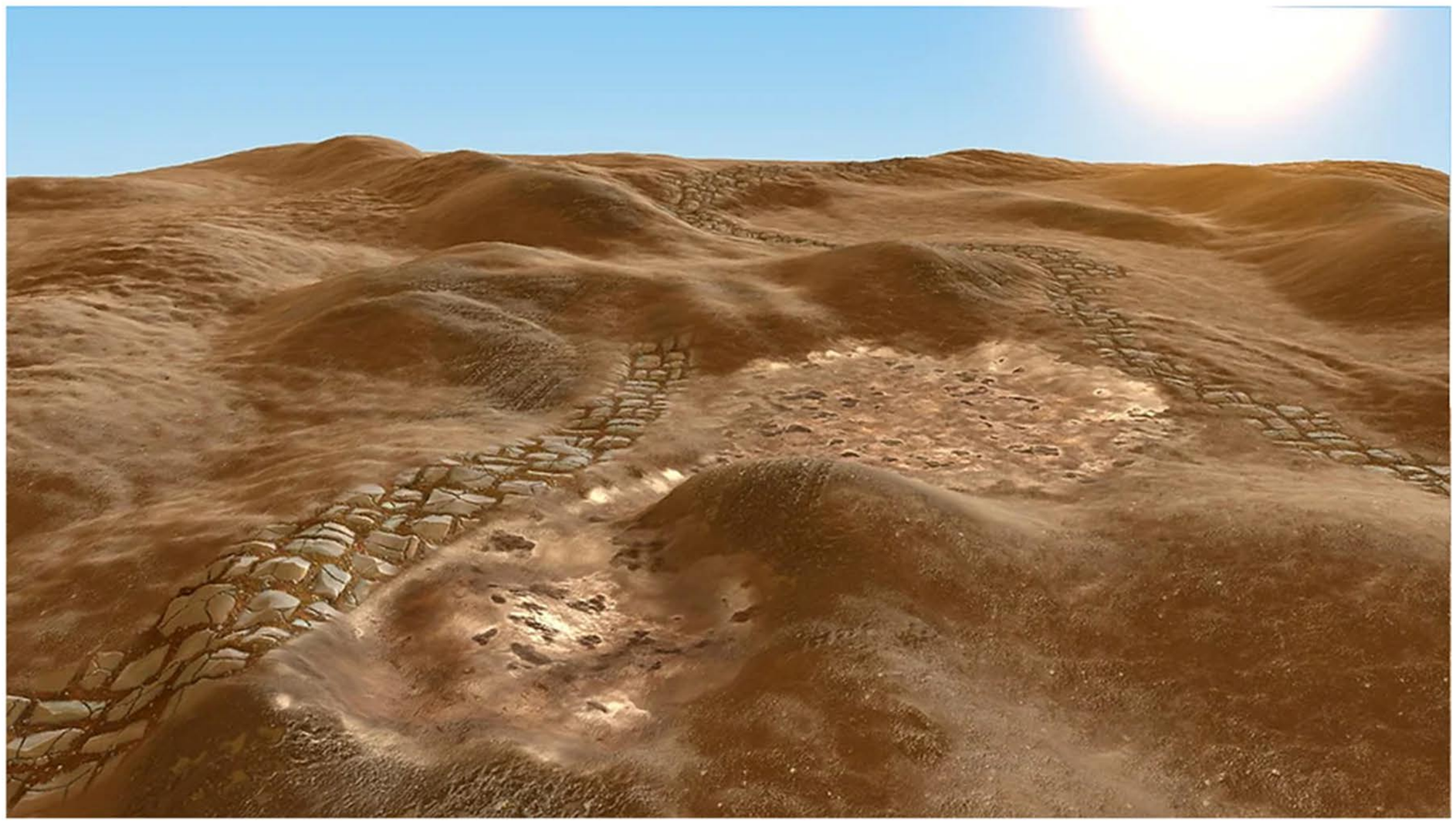


If you want to use the Ground shader for your WebGl or low-end mobile projects the asset features a tessellation-less variant of the shader that is compatible with low render target APIs. Simply switch any ground materials to the variant:



As you can see in the image above you will not lose much fidelity and you will gain compatibility and performance.

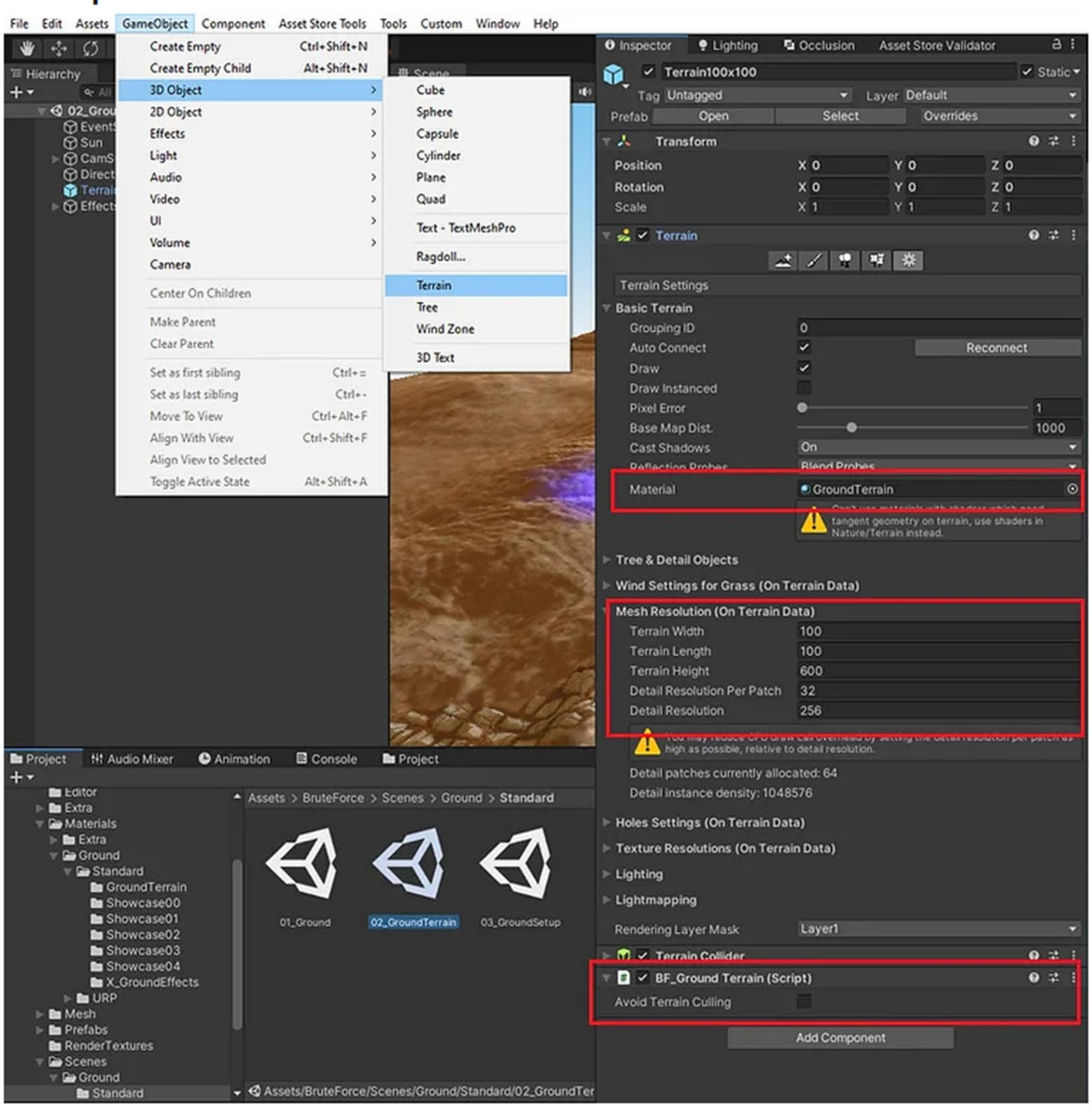
8. Terrain Feature



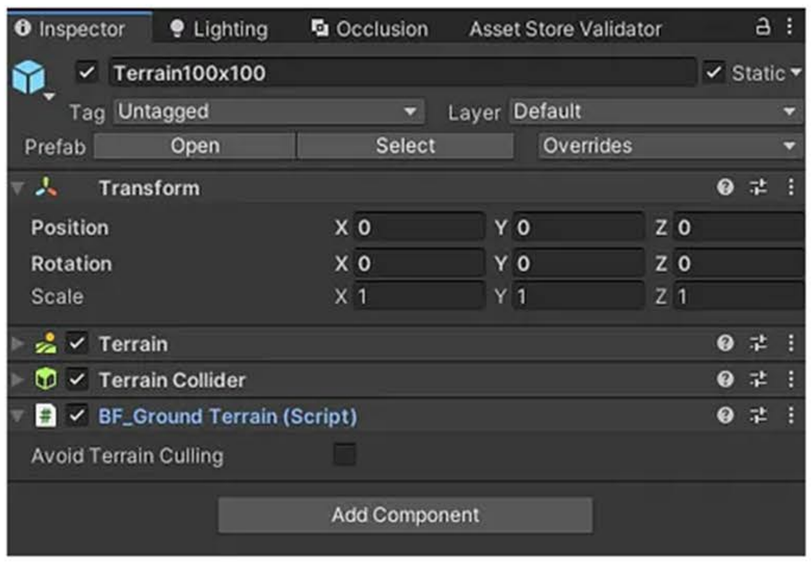
There is a template scene for terrain environment in the asset package (both built-in and URP) located here:

If you want to create your own terrain and implement the ground shader here's the important steps to follow:

1. Setup the terrain



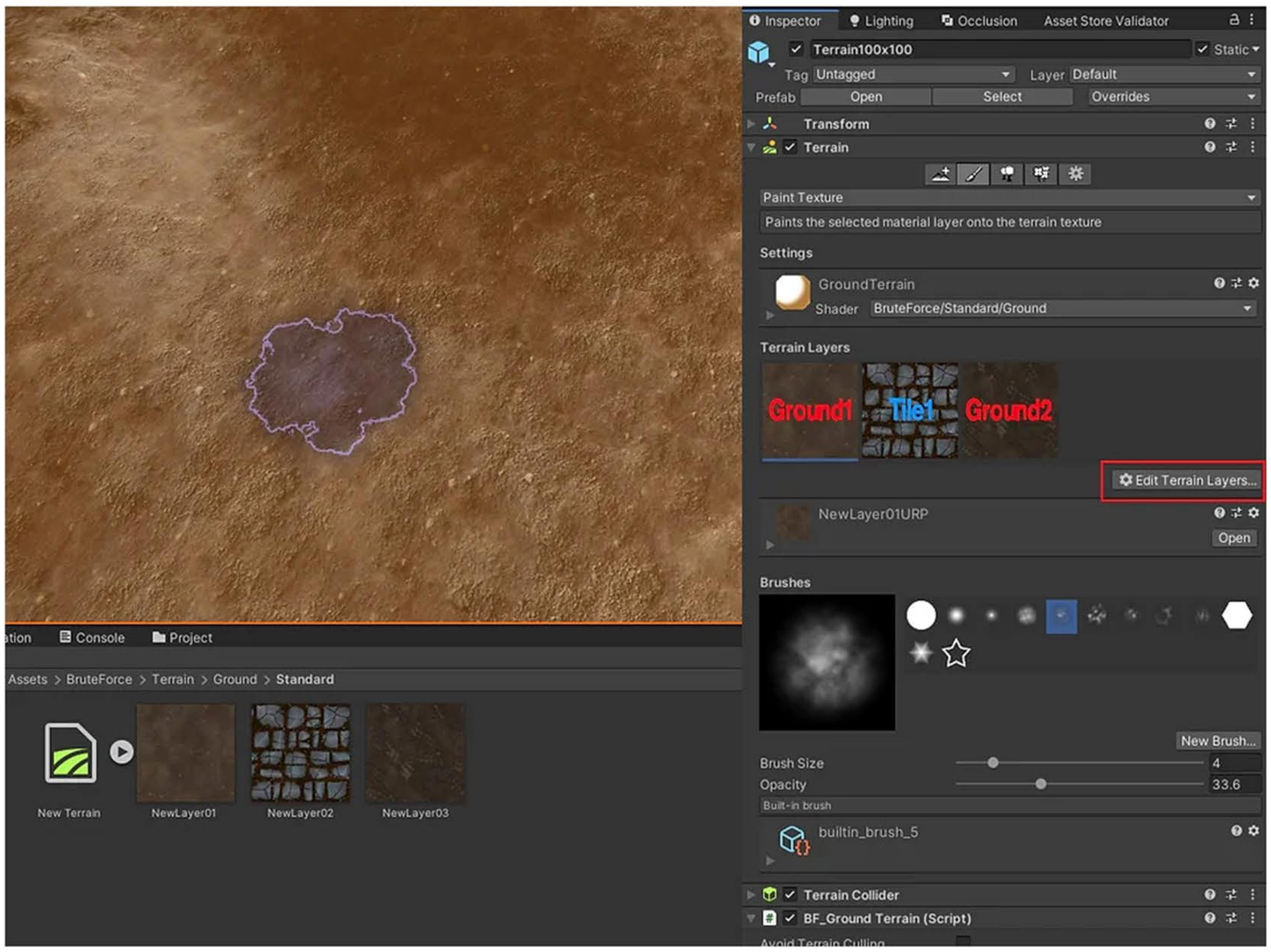
Create a new terrain with a 100x100 dimension (works best out of the box rather than the HUGE 1000x1000 default terrain)
Then add the "GroundTerrain01" material or its URP equivalent or your own Ground material



IMPORTANT: You need to add the "**BF_GroundTerrain**" script to your terrain for it to work properly

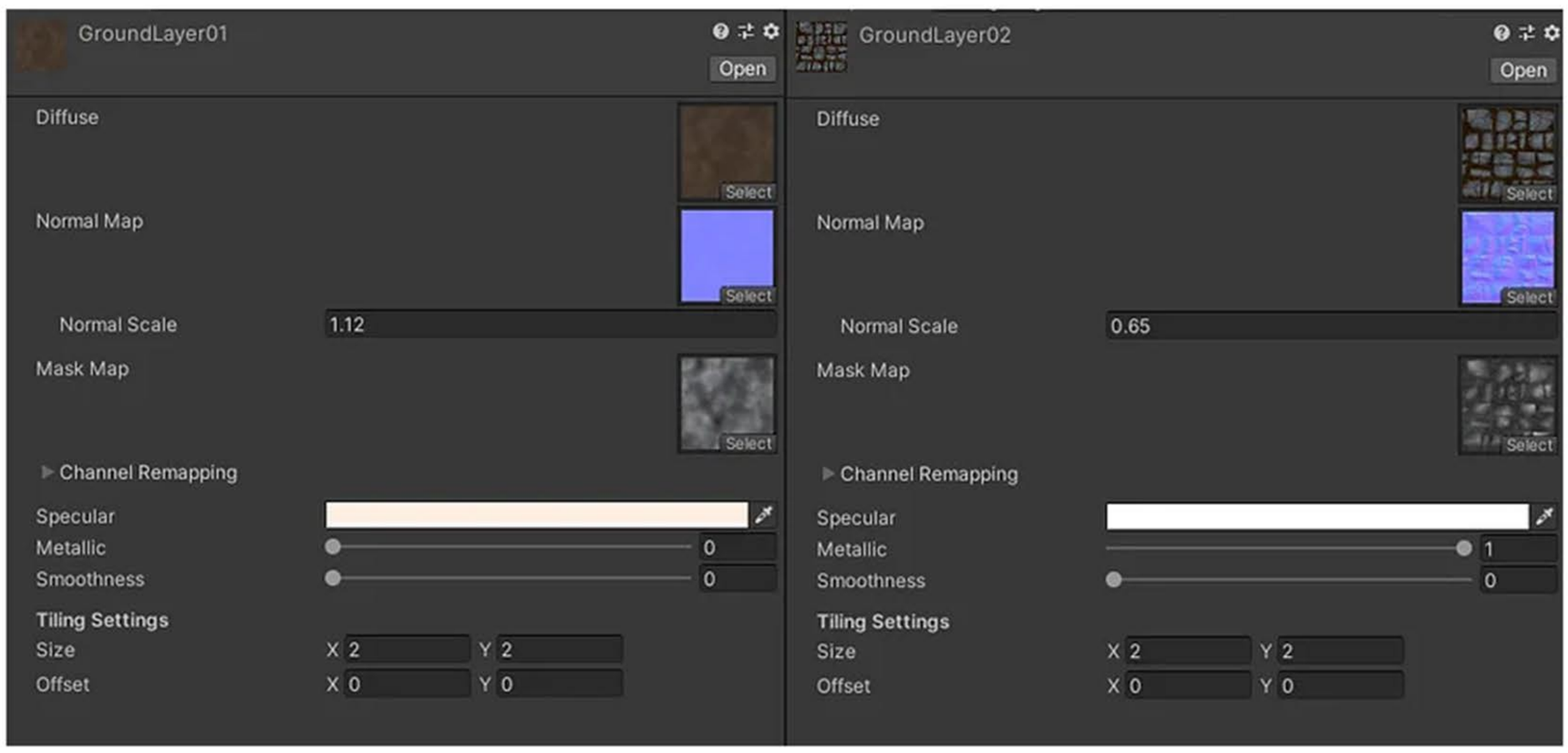
2. Edit terrain layers

After you've set the right shader/material to your terrain get over the paint terrain tool in the terrain inspector:



To draw ground on the terrain use the **GroundLayer01**, the position in the terrain layers arrangement is very important, the first one SHOULD ALWAYS be the ground layer.
To draw tiling on the terrain use the other **GroundLayer** (02,03), if you don't want/need other ground/tiling layers you can delete them via the "Edit Terrain Layers" button.

If you want to edit the layers to have a new ground/tiling layer you should head over to the corresponding "**NewLayerXX**" in the asset project:



If you want to have a Ground Layer:
Diffuse is the Albedo Texture,
Normal Map is the bump map,
Normal Scale is the normal multiplier,
Mask Map is the displacement map,
Specular is the overall color of the ground,
Metalic should be set to **0**,
Size.XY is the Tiling size

If you want to have a Tiling Layer:
Diffuse is the Albedo Texture,
Normal Map is the bump map,
Normal Scale is the normal multiplier,
Mask Map is the Height Map,
Specular is the overall color of the tiling,
Metalic should be set to **1**,
Size.XY is the Tiling size

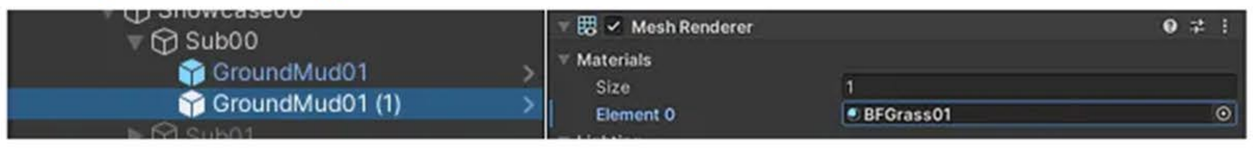
The only difference between a Ground Layer and an Tiling Layer is the Metallic slider.

9. BRUTE FORCE compatibility

The ground asset has a special connection with my other assets, it can serve as a polished base for the grass, the snow or the sand assets. They will share interactive effects and vertex color properties.

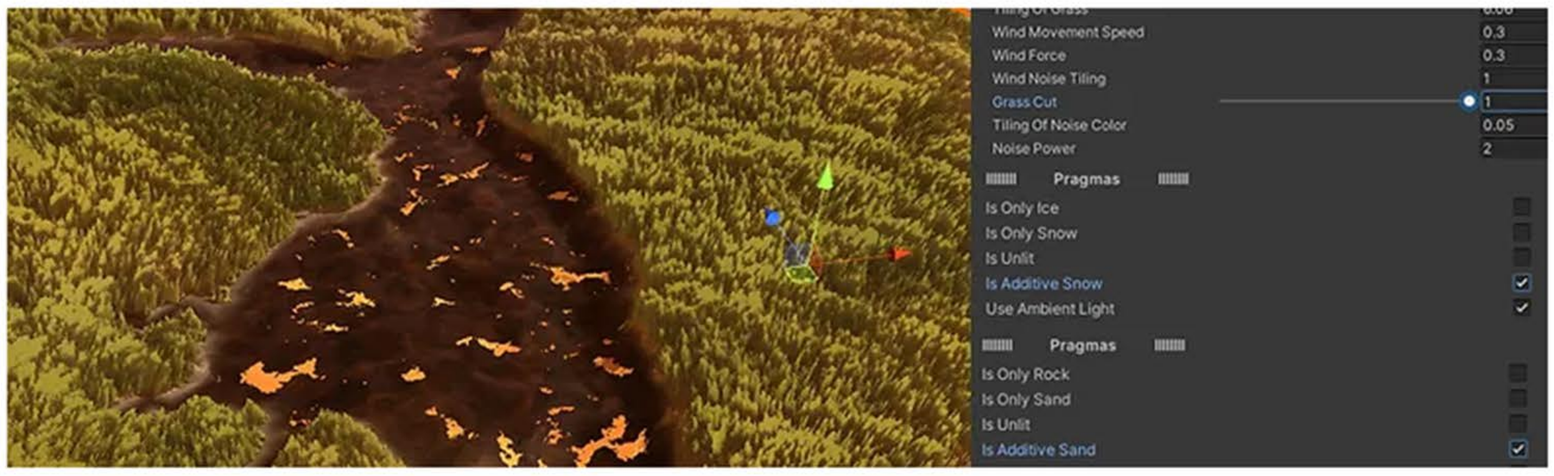
Here are the steps to have both the ground asset and any other Brute Force assets on top:

1. Duplicate the ground mesh



Duplicate the ground you want to add another asset on, then rename it to something else.
Then assign the material and the copied gameobject.

2. Avoid Culling



You should then have something like the screen above with the ground doing z-fighting with the other asset, to fix this you only need to either set "Grass Cut" to 1 if you are using the grass shader or "Is Additive Snow/Sand" if you are using the snow or sand shaders.

3. Enjoy

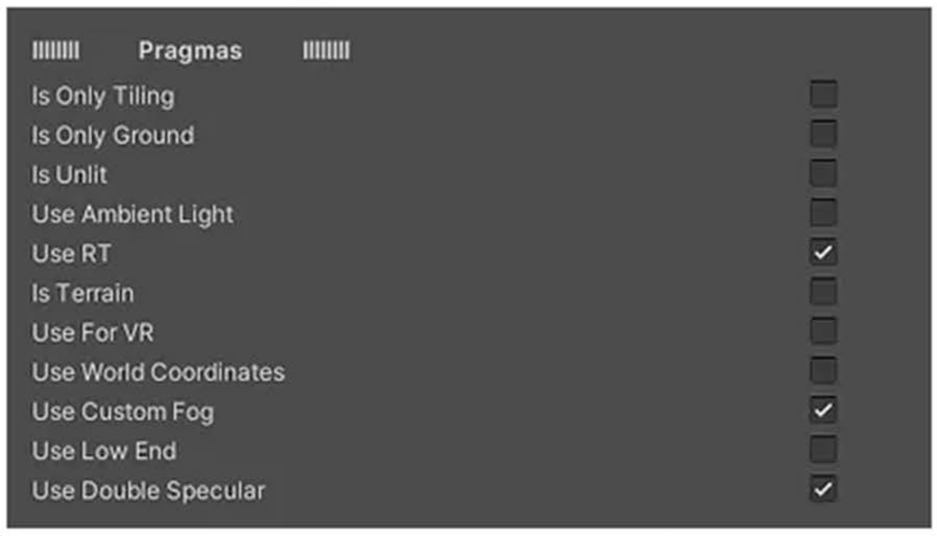
That's it you're done! Enjoy interactive effects and seamless texturing.

10. Performance Tips

Tessellation Factor

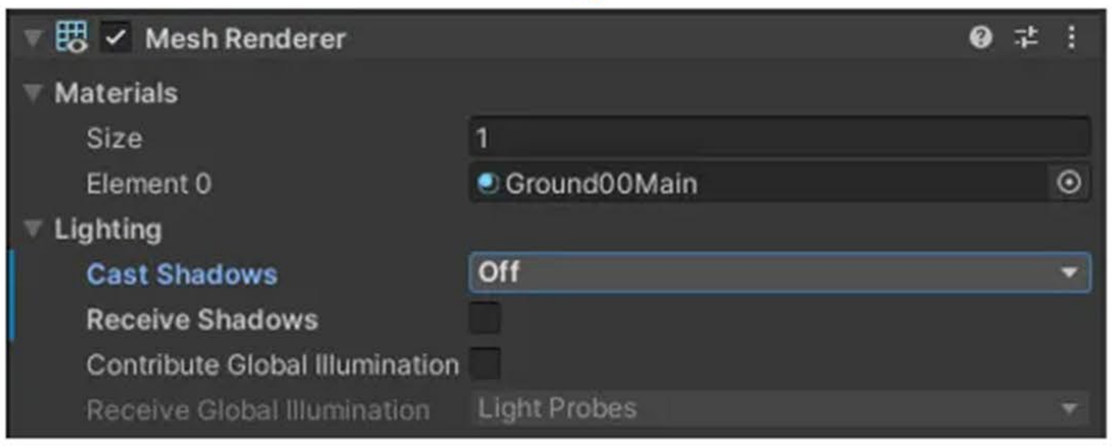
Probably the biggest performance factor of the ground shader is the tessellation factor.
You should keep the **Tessellation Tiling below 10**
For PC/Linux I recommend setting **Tessellation Ground between 20 and 50** and for mobile/VR/AR you should use the NoTessellation shader variant.
The performance hit is relative to the view distance between the ground and the camera.

Material Pragmas



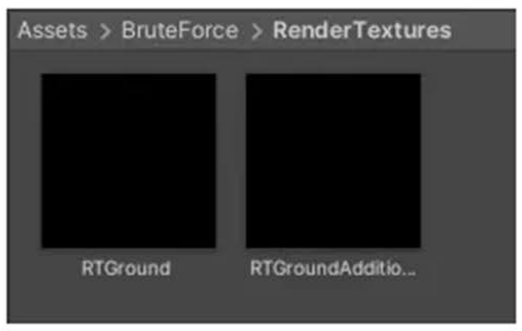
Check or uncheck the features you don't need in the material, if you don't need Interactive effect you should disable "Use RT". You can enable "**Use Low End**" when you need an extra performance boost.

Disable Shadow Casting

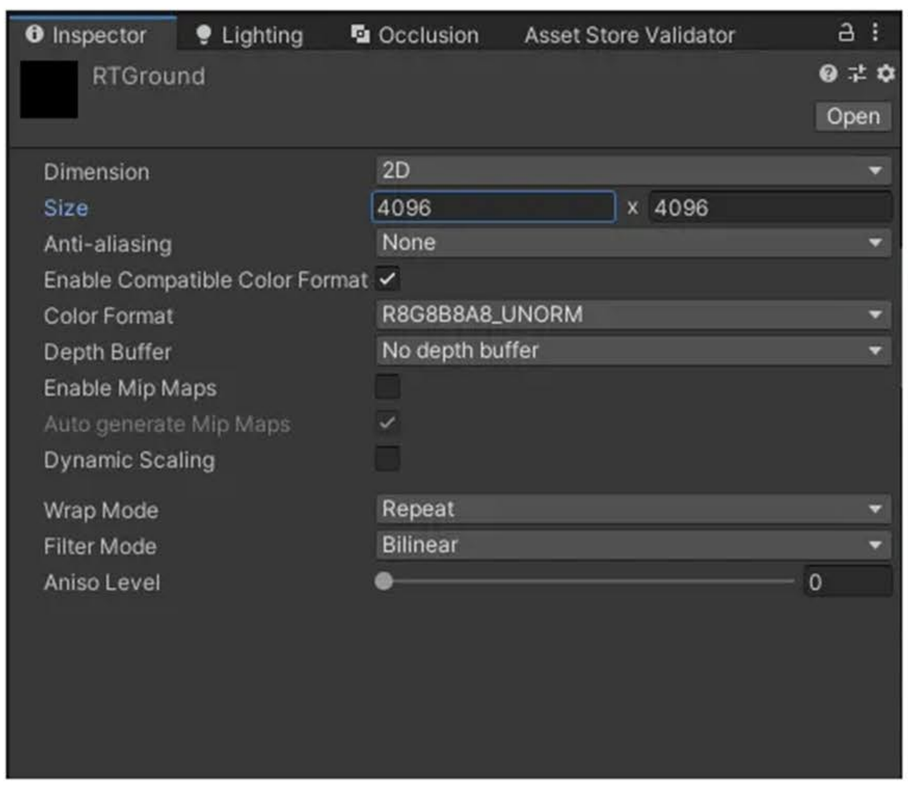


You might not need the shadow casting feature of the mesh renderer in which case this could save you a good performance chunk. This is especially true for terrains which have a Double shadow casting enabled by default!

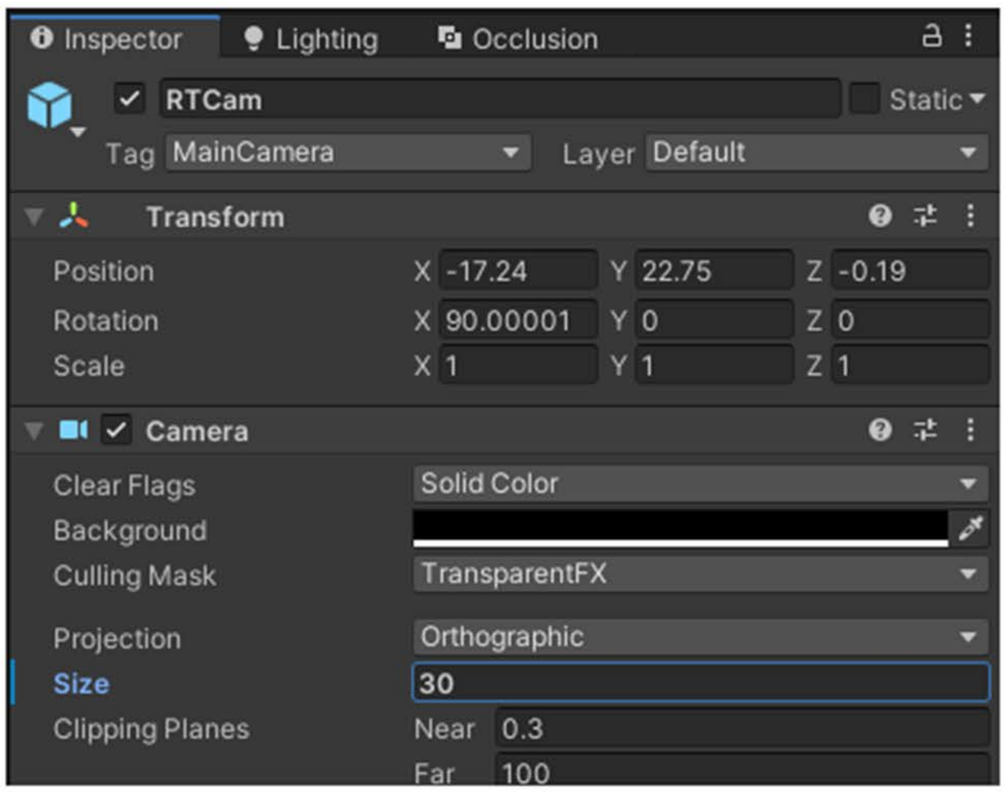
Decrease Render Textures Size



If you notice some fps drops when using the RT effects, then you might consider reducing the Render Texture size, by default the size value of the main RT is 4096 but you can scale it down to 2048 or even 1024 if you don't need high precision effects. The Additional RT is set to 1024 by default and is reasonably small enough:

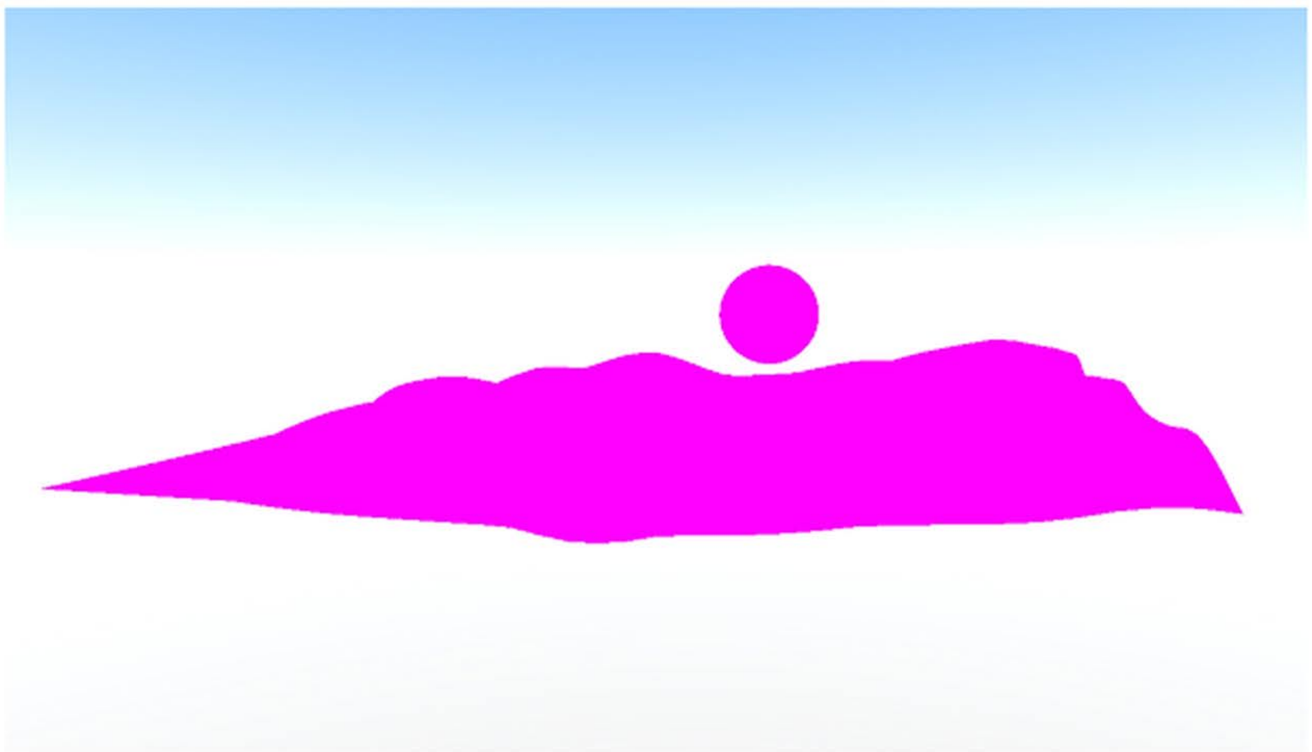


Keep in mind that decreasing the RT size will also reduce effects definition, so you might want to also scale down the main RTCam Size value:

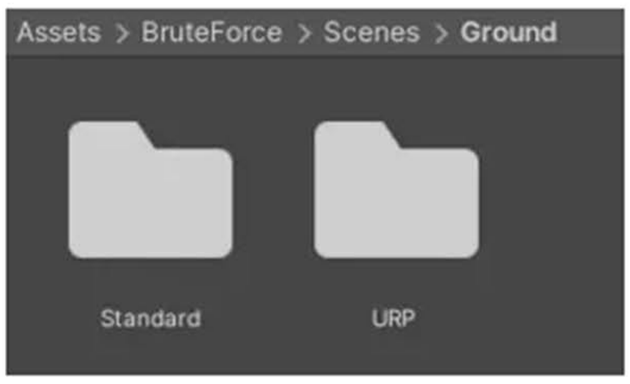


11. FAQ & Troubleshooting

I opened the package scene and all I see is pink!



You opened the wrong scene, there is 2 variations of the Ground Shader: one for Unity Standard and one for URP.
Use the folders corresponding to your render pipeline:

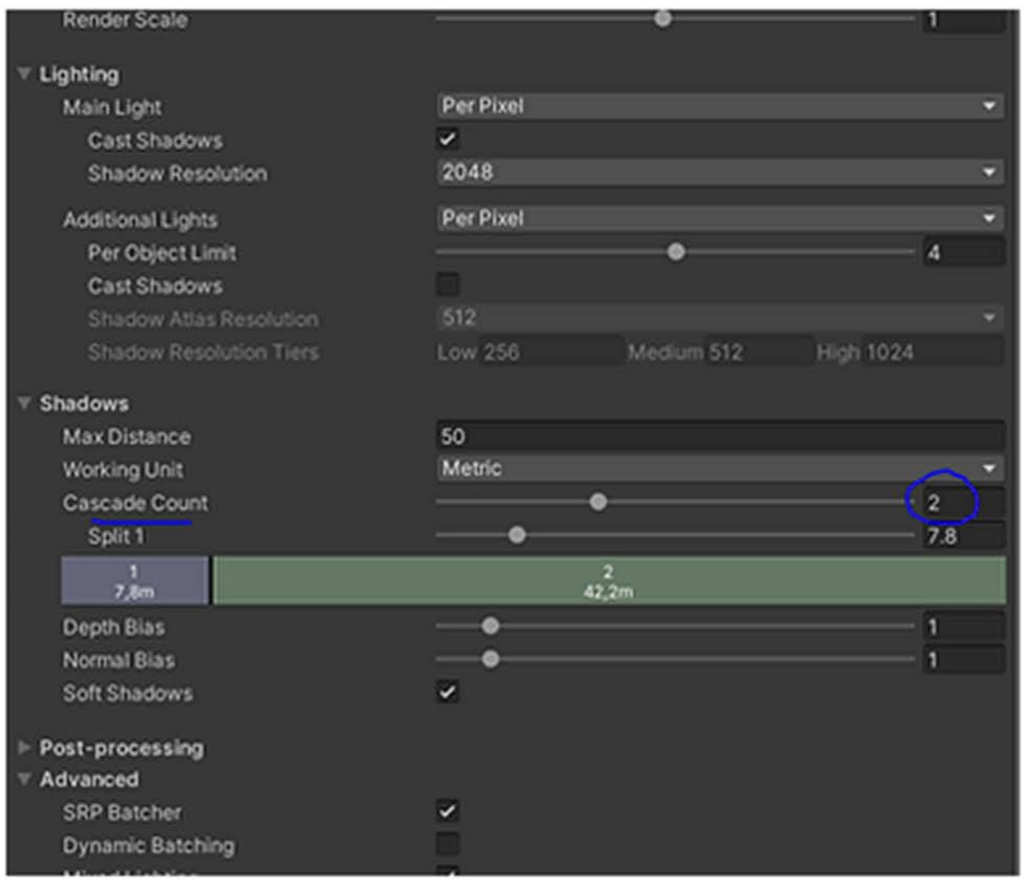


I can't find the URP folders and when I import the package it says only v1.0

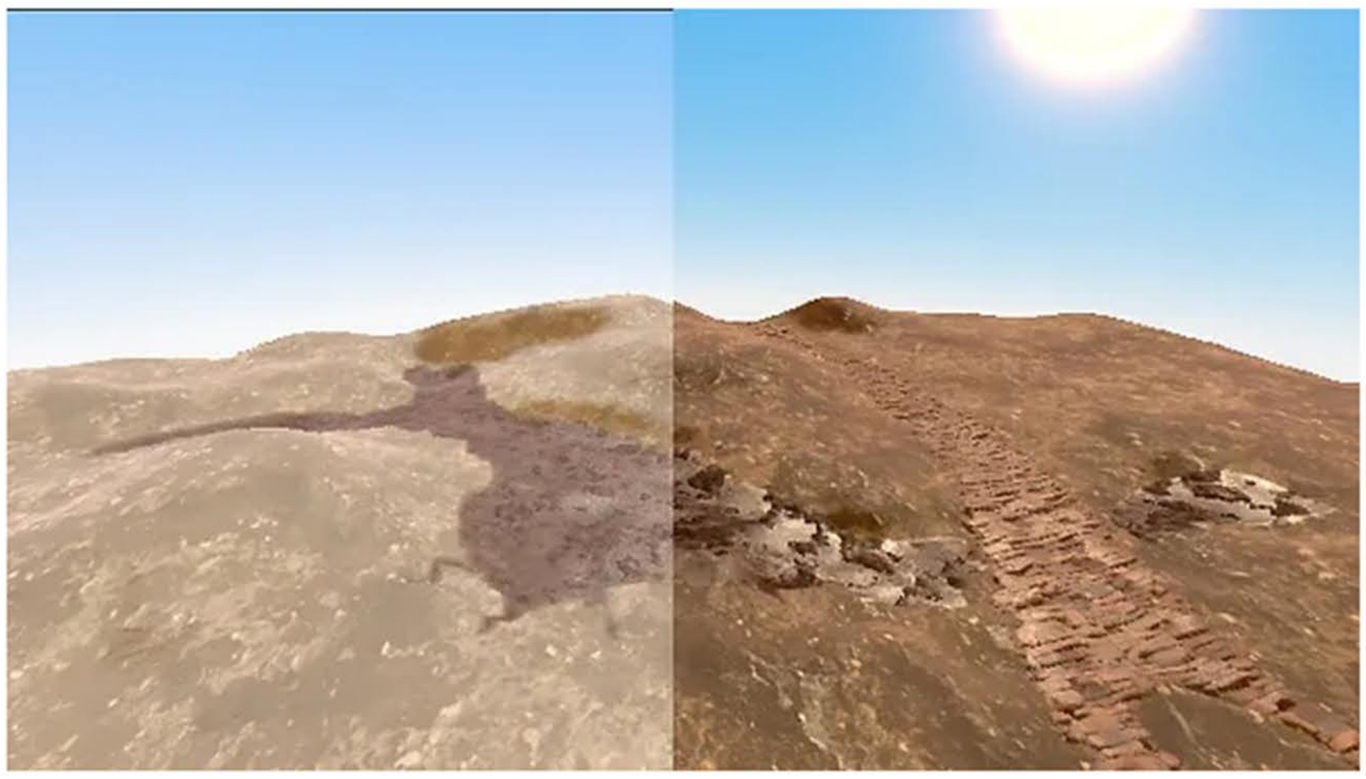
Contact me at bruteforcegamesstudio@gmail.com with your unity asset store order and I will give it directly to you, chances are the unity package manager is forcing the wrong one on your unity version.
You can solve this issue without my help by installing any unity above 2019.4 and importing the package, then you can export the unity package in your projects.

I have shadow artefacts on Mobile URP

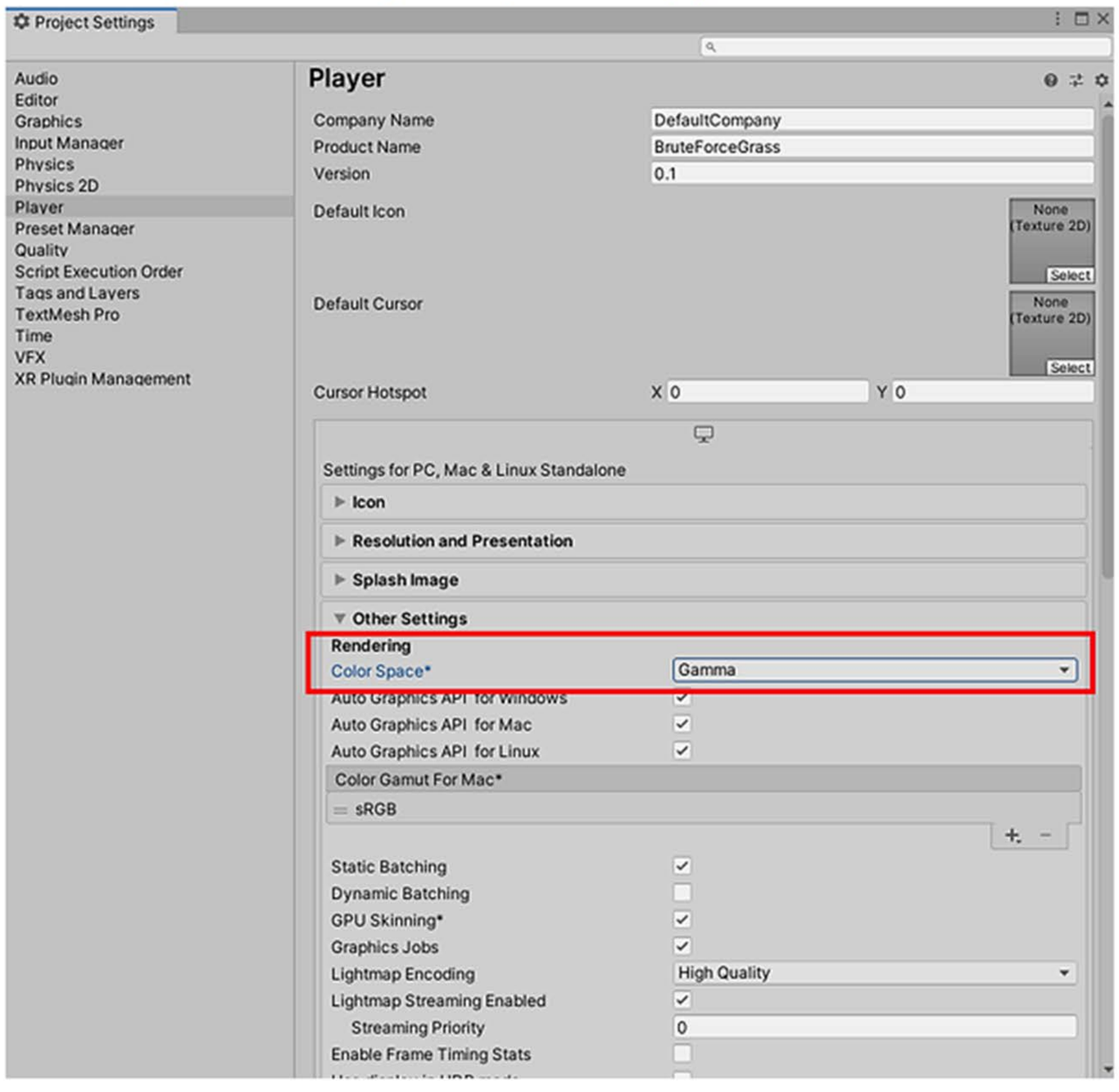
The problem comes from the URP pipeline itself, to get around this issue you need to set your shadow cascade count to 2 or more:



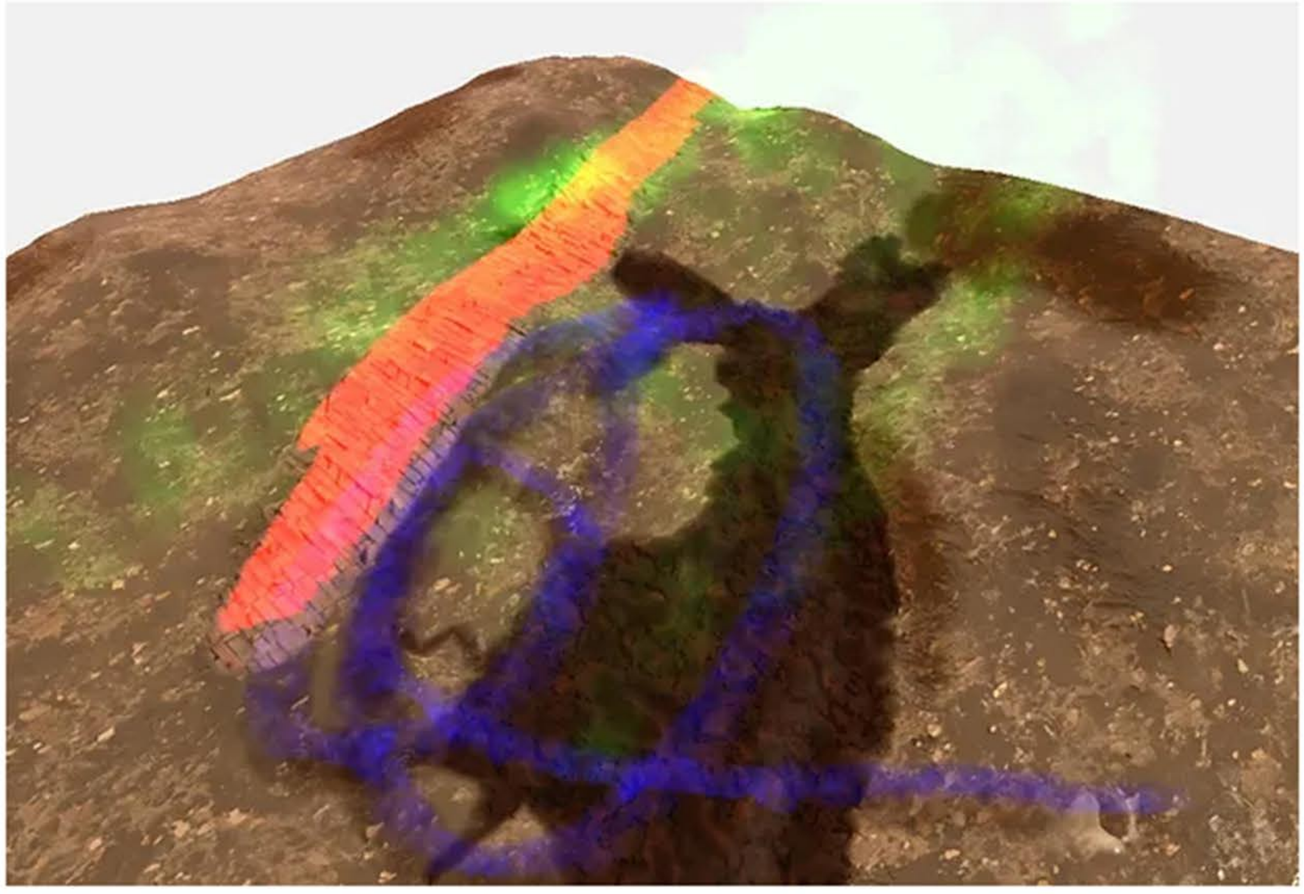
The ground looks so pale/bright in my project



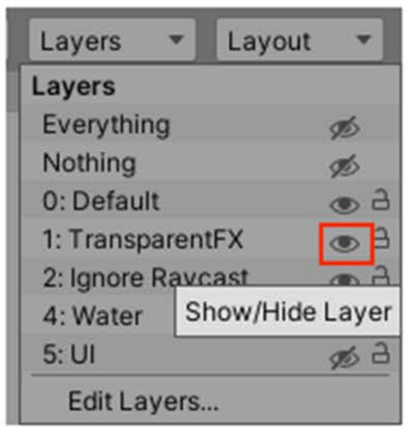
Color differences may occur when you're not using the default unity color space depending on your Unity version and Render Pipeline, (Gamme for Built-in and Linear for URP)
Either change the rendering color space or simply tune the materials to look the way you want



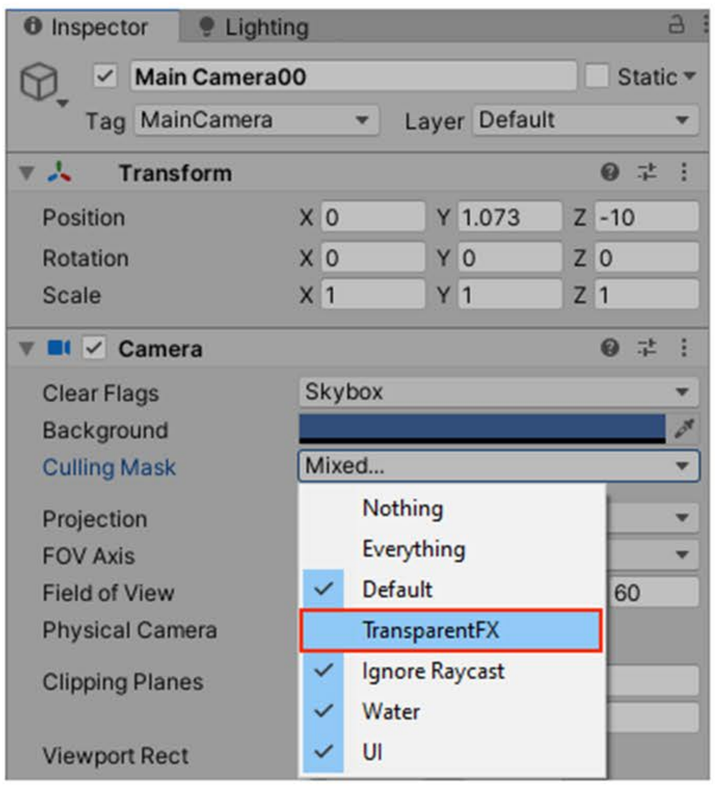
The Particle Effect is visible in the editor or at runtime



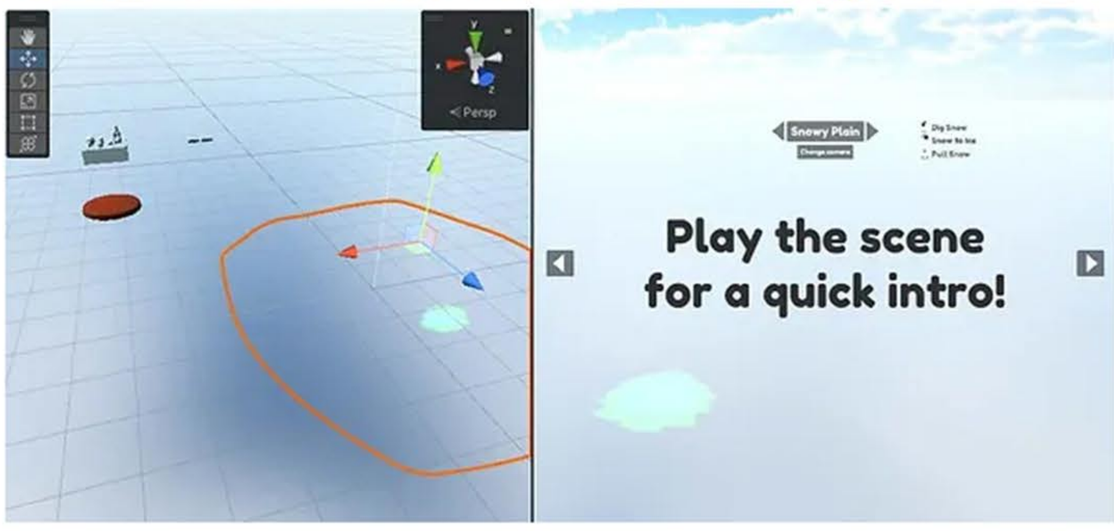
If you want to not see the particle effect rendering inside the editor, you have to disable the same layer it's in by using the layers visibility option, here the layer used is **TransparentFX**:



And if you need to disable it's visibility inside the game window or on build you need to do the same for the camera culling masks of your main camera:

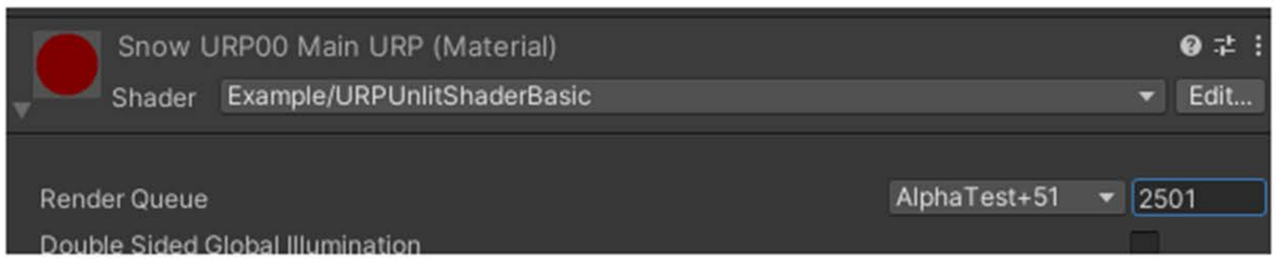


The mesh is invisible in the editor using URP



This is an annoying Unity bug that's been plaguing the latest version of Unity overall, some versions are just straight up cursed, for instance you could replace my shader with the most basic URP shader that is from the official Unity URP documentation and it still wouldn't work.

There is 2 solutions for this:
-The simplest one but it's not the best for the long run: set the render queue over every ground materials to more than 2501 like so:



And because my ground shader uses custom shadow technique you will still have shadows.

-The better solution is to switch to a better Unity version (every LTS below 2021.3.13 unity version are the best because they get frequent updates and are generally better supported by everyone), I use 2021.13 frequently and it's the most reliable.
2022.2 even though it's in beta doesn't seem to have this issue.