Terrain Scanner with object detection

Table of contents

Overview

Discover the power of hidden object detection with our Terrain Scanner Shader, available now on the Unity Asset Store! Unveil the secrets of your virtual environment by scanning and revealing concealed objects in a stunning glow, ensuring nothing remains hidden. With our shader, detected objects remain visible for a defined amount of time even when obstructed by other elements in your scene.Get the Terrain Scanner Shader and illuminate the unseen, bringing a new dimension to your creations.

This package contains 2 easy to use and fully customizable URP shaders for the terrain scanner effect and for the glow effect of the hidden objects.

Fully customizable shaders.

Terrain Scanner:

- Intersection depth
- Intersection color
- Base Texture (create a dome like scanner)
- Alpha value of the base texture
- Combined alpha value of base and intersection color

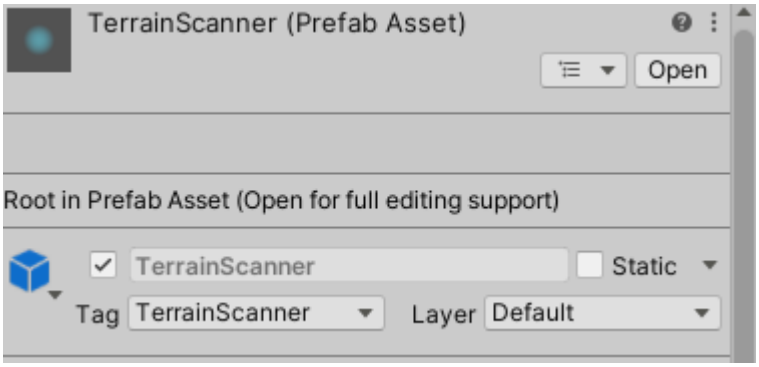
Glow Shader:

- Glow color
- Tiling
- NoiseScale
- ScrollSpeed
- Alpha value

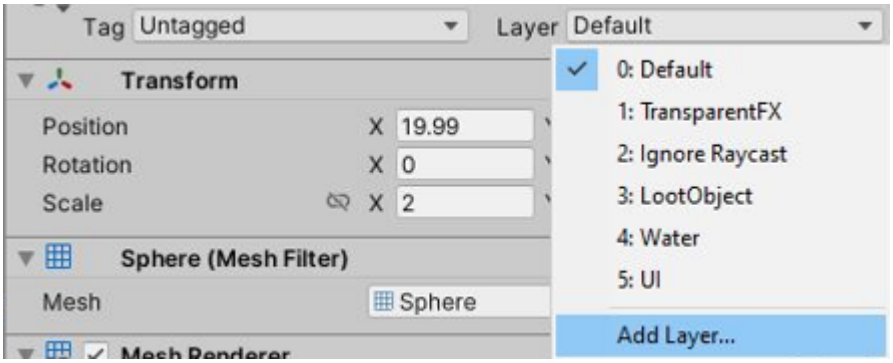
Setting up the asset

Since the Asset uses some URP render pipeline features you will need to make sure everything is set up properly before you use the asset.
Please follow these step by step instructions.

(1) Terrain Scanner tag:
First we need a Tag for the TerrainScanner. If it is not already added automatically, add a new tag named “TerrainScanner”. Locate the TerrainScanner Prefab in the prefabs folder and assign the newly created tag to the prefab.



(2) LootObject Layer
Go to the layer option and check if the *LootObject* layer exists. It not add a new layer named *LootObject*



Not adding this layer to your project will result into a Runtime Error:
“A game object can only be in one layer. The layer needs to be in the range [0...31]”

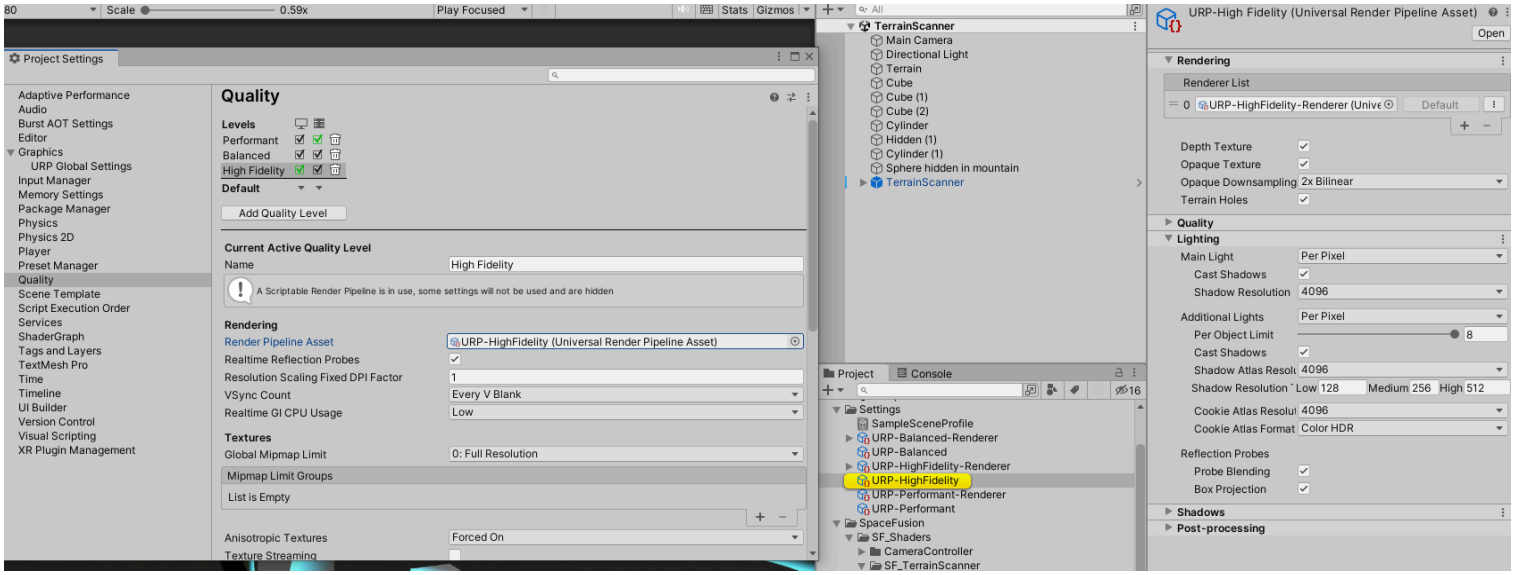
If you want to use your own existing layers for the hidden layer just go to the LayerSwitcher.cs script and insert your layers in the Start method:

```
private void Start() {
    defaultLayerNumber = LayerMask.NameToLayer("Default");
    hiddenLayerNumber =LayerMask.NameToLayer("LootObject");
}
```

If you do so please keep in mind to use your layer in the LayerMask of the Render Objects in step (5) and not the “LootObject” layer.

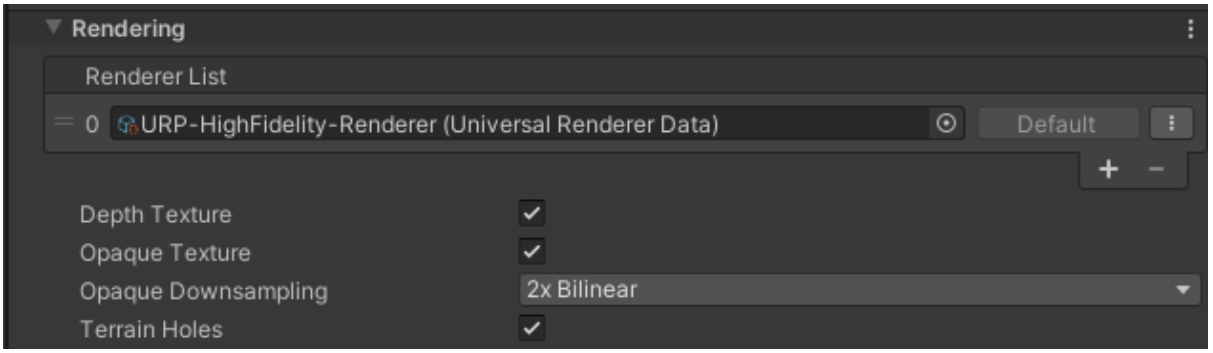
(3) Locate the used RenderPipeline

Go to Edit → Project Settings → Quality.
Under Rendering double click on the Render Pipeline Asset.
This will open the render pipeline in the inspector and lead you to the correct setting in your project folder.



(4) Enabling Depth and Opaque Texture

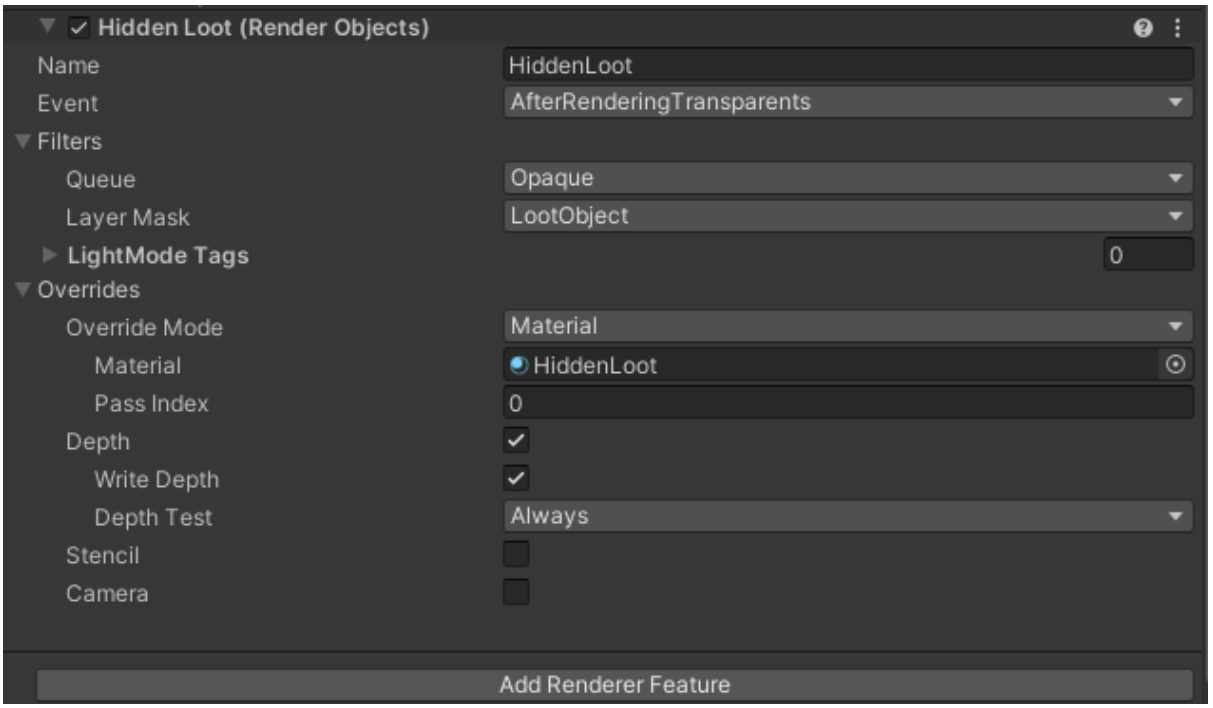
After selecting your used render pipeline, in the Inspector and make sure the *Depth Texture* and *Opaque Texture* settings are enabled



Double click on the Renderer Element of the Renderer list, this will open the inspector for this renderer.

(5) Adding Render Objects

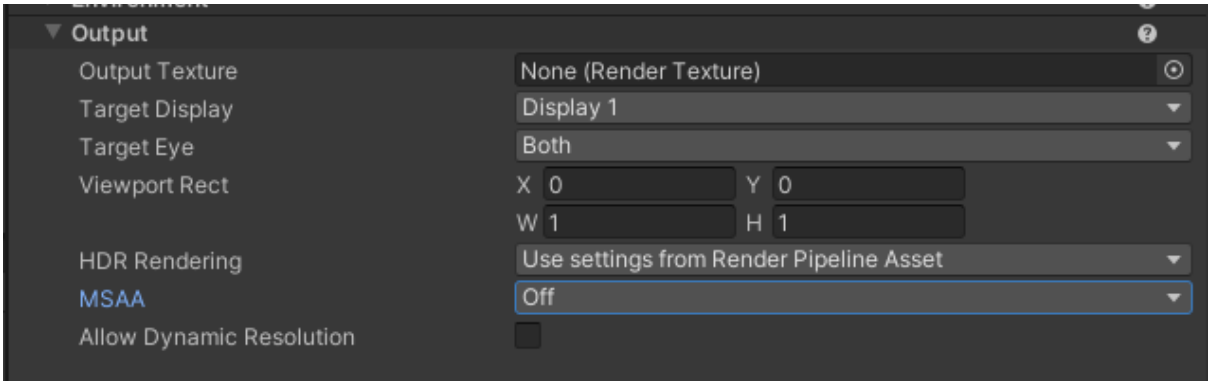
On the bottom click Add Render Feature and add a new Render Object with the settings that you see in the following image:



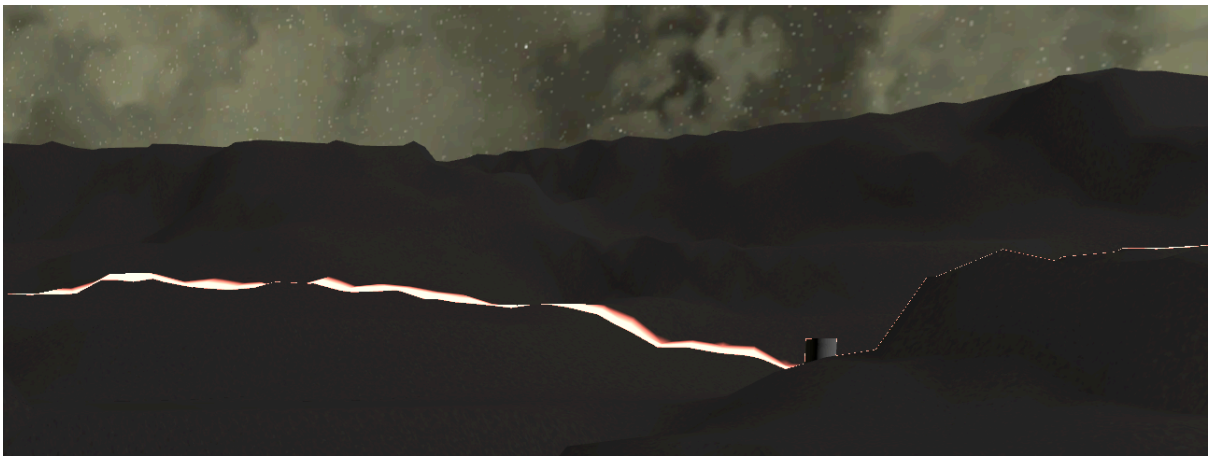
The HiddenLoot material has the glow shader attached which will make the scanned objects appear to be glowing

Now your setup is finished and should work properly.

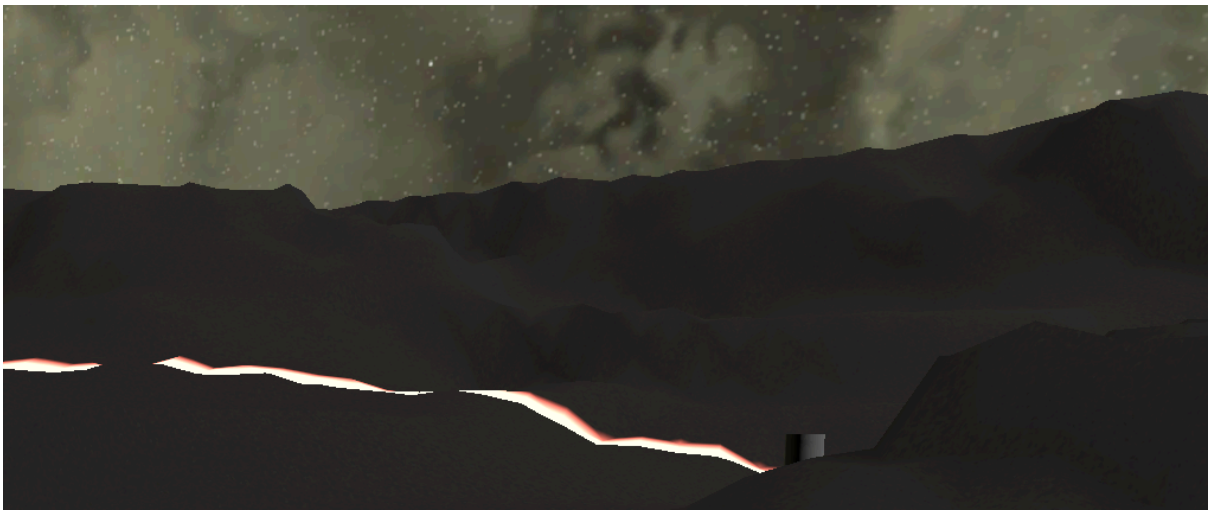
Some side note about the camera settings:
The camera has MSAA turned off to avoid some artifacts from the intersection shader



With MSAA turned on you are seeing some outlines, even if the scanner is already behind the terrain and should not be visible to the camera (see the right part of the image)



Turning MSAA off will remove those lines:



Depending on which behavior you want, feel free to turn this setting on again.

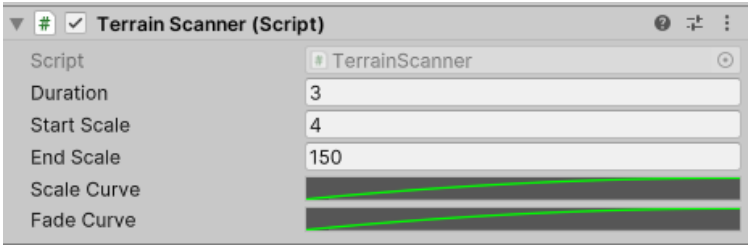
Start demo scene

You can simply start the demo scene and by pressing Space the scanner should be activated. You can also move around the camera with following keys:

- W: move forward
- A: move left
- S: move back
- D: move right
- Q: move down
- E: move up
- Right click mouse and hold: rotate camera

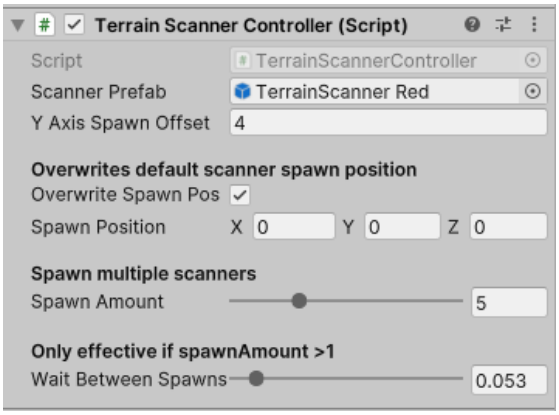
Configure the Terrain Scanner

There is a *Terrain Scanner* script attached
To the TerrainScanner prefab.



Use this script to modify the properties of the scanner.
You can define the duration/ lifetime of the scanner, the start and the end scale.
Since the scanner uses a sphere mesh, the x, y and z scale will always be the same.
You can also define a scale curve to manipulate the scale speed over time.
With the current curve, the scanner scale increases fast in the beginning and slows down towards the end until it reaches the defined endscale.
You can also specify a Fade curve which will make the scanner effect fade out near the end of the lifetime.

On the Main Camera gameobject you have a *Terrain Scanner Controller* script attached:



The **y spawn offset** of 4 means that the scanner spawn position will be the camera position but minus 4 in the y axis.
So if the camera position is e.g. (2,8,2) the spawn position of the scanner will be (2,4,2).
By checking **overwrite spawn pos**, the scanner will always spawn on the overwritten position, independent of the camera position.
You can also **spawn multiple scanners** at one by configuring the spawn amount. If you spawn more than 1 scanner you can define the wait time in between the spawns.

In the Prefabs folder you can find a couple of preconfigured Terrain scanners.
Just place them in the Scanner prefab property of the Terrain Scanner controller to use them!

Configure Object detection

For every object in the scene that you want to be detected by the terrain scanner, simply attach the *LayerSwitcher* script to it.
Please also make sure that the gameobject has a Collider attached to it, and that's it!:)

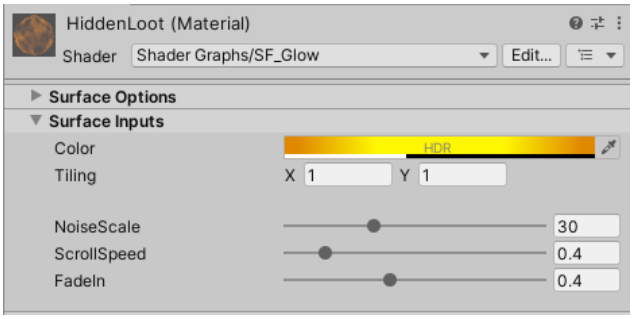
How the script works:
Script will fire an OnTriggerEnter event when the terrainScanner hits the object.
Then it will change the current layer of the object to LootObject.
This will activate the *Render Object* that you set up previously which is always visible, this means all scanned objects will be visible to your camera with a glow effect, even if they are hidden behind other objects.
After the default duration of 8 seconds the layers will be switched back to the previous one, disabling the glow effect again.

Shaders

Under SpaceFusion/SF_Shaders/SF_TerrainScanner/ShaderGraphs you can find the two shadergraphs used by this asset:

Glow Shader

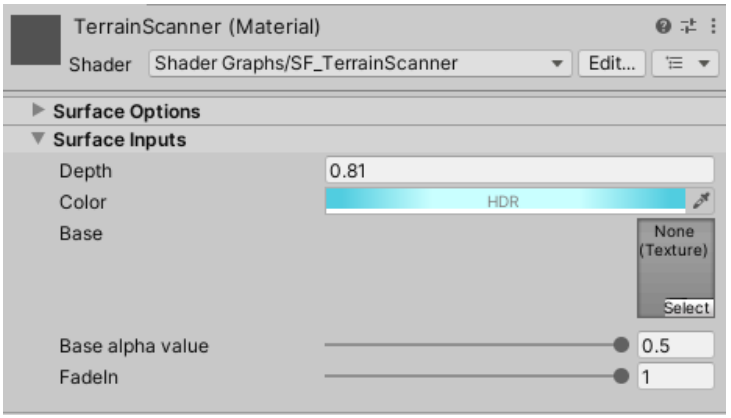
There is a *HiddenLoot* material in the materials folder that is using the Glow shader.
You already used this material when setting up the asset. You attached it to the Render Object for the *HiddenLoot* layer. This material will be used for every object that has set the layer to *HiddenLoot*



Feel free to play around with the settings and create a glow effect that is appealing to you.

Terrain Scanner Shader

The TerrainScanner material is using the TerrainScanner shader.



Feel free to play around with the settings and create a scanner effect that is appealing to you.

- Depth: changes the intersection effect size
- Color: changes the intersection color
- Base texture: adds a texture to the scanner sphere, so you can potentially create a dome scanner effect
- Base alpha value controls the transparency of the base texture
- FadeIn controls the combined transparency of the texture and the intersection color.
    - This value is modified in the TerrainScanner script with the fade curve

Hope you enjoy this asset!
For any questions/ complaints/suggestions feel free to contact me under gamedevibk@gmail.com

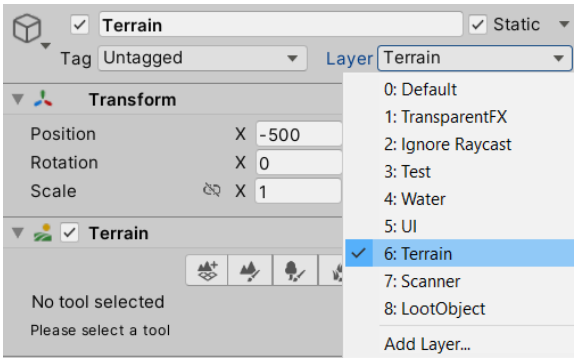
Appendix 1 - Apply TerrainScanner only on terrain

Since this was already asked by a customer, maybe it can also help others if they are struggling with the same issue.

The shader itself has no detailed info about other objects in the scene, so we can not achieve this by modifying the shader.
But what we can do is change the order in which things are rendered.

- First render the Terrain
- Then render the TerrainScanner
- And afterwards render all the other objects in the scene

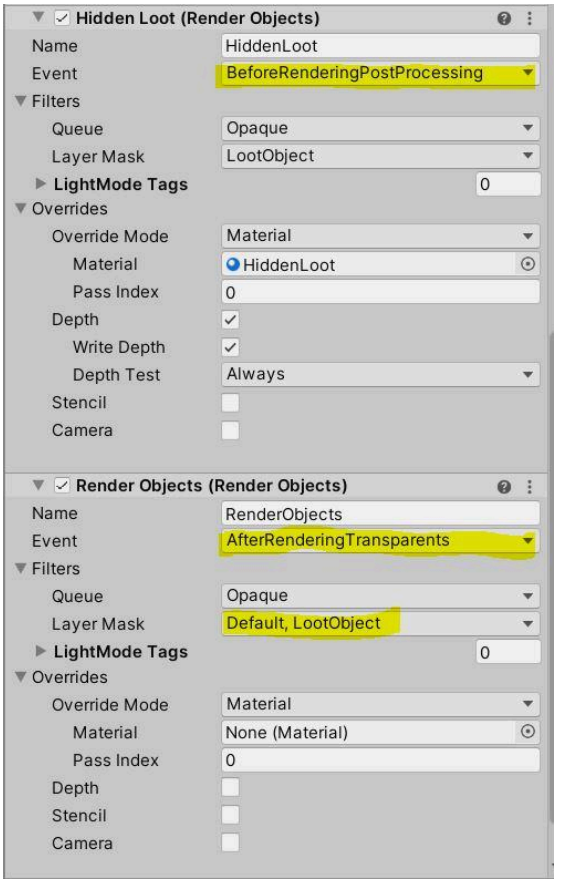
That's at least the method I came up with. If someone has an even better solution, and wants to share it, feel free to contact me. So to achieve this we need to create a new Layer named “Terrain” and attach it to the terrain:



Then we go to the Render Pipeline where you have already added the HiddenLoot RenderObject. This is described in “Setting up the asset”.

Apply following configuration:

- Change Hidden Loot Render to BeforeRenderingPostProcessing
- Create new render object with Event “AfterRenderingTransparents”
- Select all Layers that you want to be rendered after the Scanner
  (In our case every object is either Default or LootObject)



With this order you will make sure that the terrain layer is Rendered in the beginning. Then the scanner is applied. Since scanner is of type “transparent” it will be rendered Before all the other objects.
Then we apply “AfterRenderingTransparents” for all Other layers. And in the end we render the specific HiddenLoot material with “BeforeRenderingPostProcessing” To ensure the object detection is still working.