

PCG Environment

Inhoudsopgave

[Process Diary 2](#_Toc129360642)

[Day 1 (07-03-2023) 2](#_Toc129360643)

[Day 2 (08-03-2023) 5](#_Toc129360644)

[Day 3 (09-03-2023) 7](#_Toc129360645)

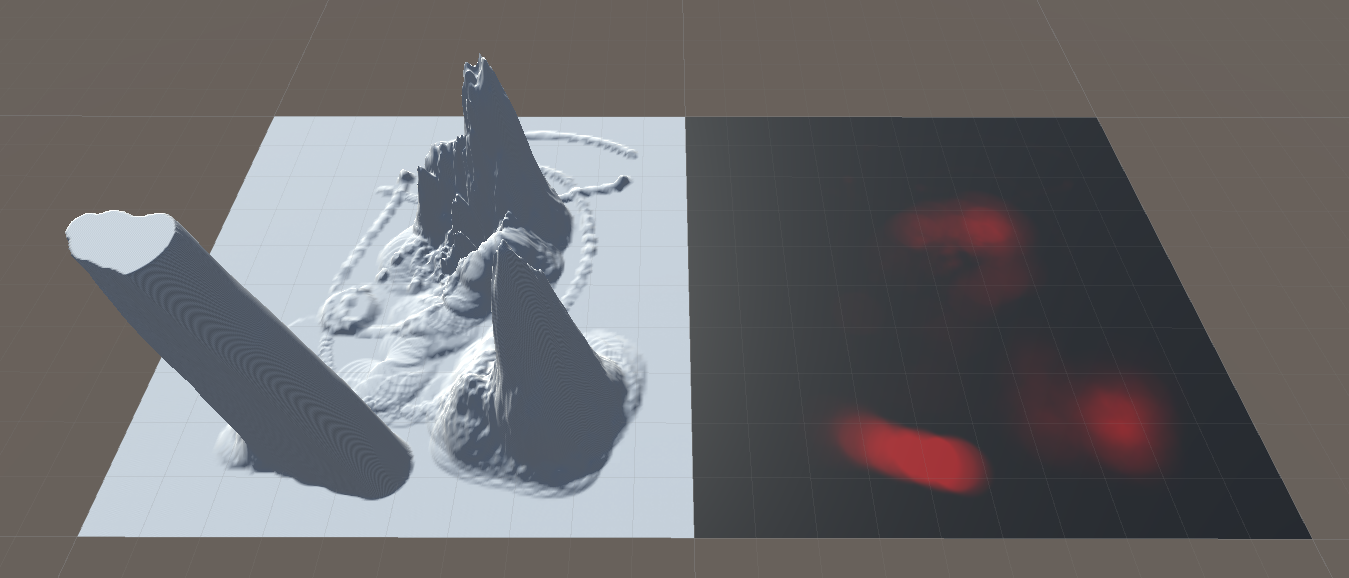
[Day 4 (10-03-2023) 9](#_Toc129360646)

# Process Diary

## Day 1 (07-03-2023)

* Started research into built-in Unity Terrain:

Terrain stores height data in a RenderTexture using the red color as height.



Terrain stores texture data in a Texture2D array. Every Texture2D in the array stores 4 different textures. It stacks the Texture2Ds on top of each other.

The 1st texture is stored as red, the 2nd as green, the 3rd as blue and the 4th as black.

If another Texture2D is stacked on top, it turns every texture saved in it as black on the first Texture2D.

Afbeelding met tekst, binnen

Automatisch gegenereerde beschrijving

* Question: How does it differentiate between the 4th texture and the stacked texture, seeing as it’s both black?

I looked if I could open the Albedo map from the material and I could. After checking I found the 4th texture is stored as the alpha value of the texture.

Afbeelding met tekst, schermafbeelding, monitor, scherm

Automatisch gegenereerde beschrijving

I started experimenting with the tree spawning and grass function of the Terrain. The Terrain can spawn trees (as prefabs) at specific locations or in mass, but you can’t influence them afterwards, only add more or delete. This is the same for the grass/bushes. With the bushes and trees I came across some problems, the LOD component is not supported in the Terrain.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

This is a problem, seeing as this is an important component to keep good performance.

I also took a look at how others fix this problem. Their solution, don’t use the trees/bushes feature, only use the terrain to spawn grass.

I took a look at how the Terrain saves it’s data. This is done in a .asset file, which is not readable. Seeing as I want to store data differently to circumvent the LOD problem it’s probably easiest to make a new (ScriptableObject) data format. I would also need to make a converter (new format -> Terrain data) but I don’t think that will be very difficult.

The Terrain automatically doesn’t spawn trees on steep slopes. Seeing as I won’t be using the Terrain’s tree spawning I will have to imitate it myself. The solution I’ve come up with is to make a new Texture2D map which will store slope data.

With that data I could make sure trees don’t spawn where there is a steep slope.

This data can also be used for texturing. I could say, I want to texture steep slopes with, for example rock.

I thought of combining it with the height map as, for example, the blue color, but I’m not sure if this will interfere with how Terrain reads the Texture2D. I’ll need to investigate.

-End Day 1

## Day 2 (08-03-2023)

* Started Programming

I started out trying to make the new data format I thought of on day 1. It all went well, until it came to generating new Texture2D files. Unity kept giving a ‘type mismatch’ error in the object field. Turns out, scriptable objects don’t like the creation of new instances of Texture2D or something like that. So I had to add the instance to the scriptable objects AssetDatabase file.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

I set the TextureFormat to R16 as that is the format the Terrain data uses. This also means that I can’t store any other colors onto it, so my previous idea of combining the height and slope map won’t work (EDIT: cancel that, explained later in Day 2).

I did source some code from: <https://www.reddit.com/r/Unity3D/comments/7fzmx3/scriptableobject_asset_texture2d_typemismatch/>

The SetTextureToColor() function sets the entire texture to one color.

I sourced the code from:   
<https://answers.unity.com/questions/236740/how-do-i-fill-a-texture-with-one-color.html>

During the process of making this, Unity did crash a couple of times. Tip: Don’t copy past any AssetDatabase related code.

After this I wanted to try loading my newly created height map onto the Terrain. So I converted the pixel data from the height map into a float array and loaded it onto the Terrain.

Screenshot on next page 🡪

Afbeelding met tekst

Automatisch gegenereerde beschrijving

It is a bit hardcoded now with the width, height and layer length, but whatever.

So now it works, and I can load the height map onto the Terrain.

(This example I made in Paint.net, it isn’t procedurally generated)

Afbeelding met versierd

Automatisch gegenereerde beschrijving

The trees are from when I experimented with the tree spawning on Day 1, so pay no mind to those.

So I also now realized that I can actually store the height and slope map into 1 Texture2D. Because I’m not actually setting the Terrain height map to my height map, but rather the red pixel values, I can change the texture format I save to RGB/RGBA and still get the same result.

-End Day 2

## Day 3 (09-03-2023)

* Continued programming

Oh buddy, I continued programming and wanted to clean up the terrain loading so it wouldn’t be as hard coded, so I did. While I was at it, I added some fail-safes and also realized, this is a very performance heavy piece of code.

I reworked part of the loading script to a generic method to convert a pixel list to a pixel grid. Earlier when I was looking back at yesterday’s work, I realized the terrain wasn’t at a high resolution, so I tested the grid conversion with a bigger list to make it a higher resolution. Oh boy this takes a bit. It takes like 30 seconds to convert a 1028 by 1028 texture. (EDIT: it seems it wasn’t the algorithm which was taking long, it was actually the thousands of debug logs it was sending. After disabling them it takes around 0.04 seconds.)

So, what started out relatively simple, has kind off spiraled. I now also have a data format for storing… texture data in a way I can easily work with… overkill I know.

I also moved over the grid conversion to it as static method.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

So now onto the reason I needed the pixel grid, outside the fact the Terrain required it for the height map.

Want I’m going to do is make the algorithm which’ll generate the slope map from the height map.

Well… I didn’t start working on the slope map. I first wanted to fix a problem where if you load a new height map into the serialized field and then generate a new one, the old one gets overwritten (as in, the actual asset get overwritten). Also, when you add a new one to the serialized field, the child asset isn’t updated… This. Was. The. Biggest. Waste. Of. Time. In. History.

I wasted 4 hours on trying to fix it and I swear to god nothing even came close. Unity AssetDatabase absolutely sucks.

Well, beaten and exhausted, I now move to making the slope map, but first, lunch.

SIKE, during lunch I thought up a possible fix, so, smart as I am (not) I returned to pain land, and after half an hour IT WORKED.

This is just the code for getting and setting the height map, I’m not including the making of the child asset in the scriptable object, because that won’t fit here.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

For the creating of the child assets I used this source:  
<https://stackoverflow.com/questions/71250177/trying-to-create-nested-scriptableobject-addassettosamefile-failed-because-the>

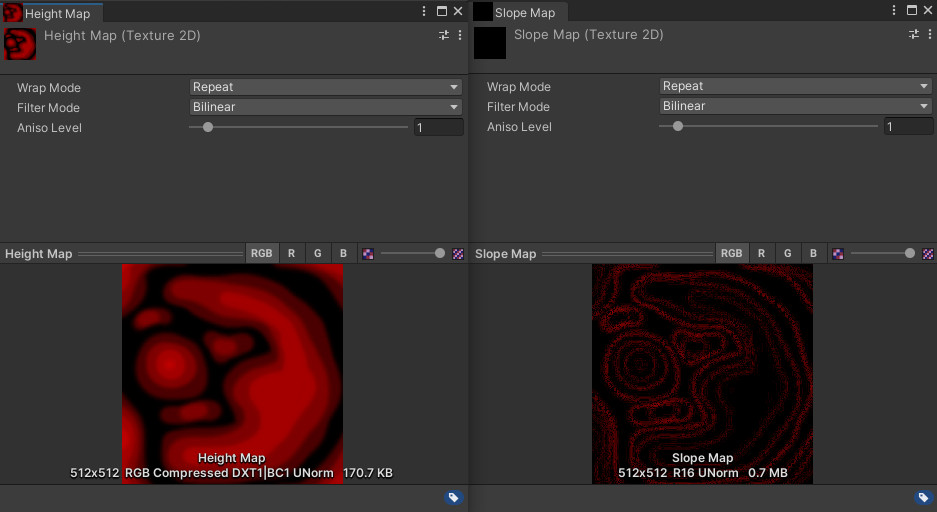
So after this fiasco I’m done for today, I’ll do the slope map tomorrow.

-End Day 3

## Day 4 (10-03-2023)

* Continued programming

I made the slope map by setting the pixel value for each pixel in the slope map, to the highest difference between the height map pixel and its neighbours. This was the result. (Height map left, Slope map right)



The resulting value is also x10 because otherwise you can’t see it.

The algorithms isn’t very precise, you can also see some grain in the result. I might modify it later, but right now, this is good enough.

I tried setting the Terrain alphamapTextures but this isn’t as easy as I’d hoped. It’s already dinner time by now, so I’m going to slide this task to next Monday.

-End Day 4

## Day 5 (13-03-2023)

* Continued programming

First, I tried to fix the grain in the slope map. Currently I’m only checking direct neighbours (left, right, top, bottom) for my calculation, so I tried to also include cross neighbours (left top, right top, left bottom, right bottom).

Adding cross neighbours did not work. It was still grainy.

I tried changing the value I was using from the biggest difference to the average. This resulted in a much better map, but it was still grainy.

I found out that the grain came from the difference in resolution between the Terrain height map and the 512x512 heightmap I was using. The terrain stores its height data in a 513x513 texture but is somehow able to keep way more detail than I currently can. I’m going to investigate a little bit into that.

Unity’s Terrain has a few variables which determine the resolution and detail of the Terrain, but I really don’t understand how this works.

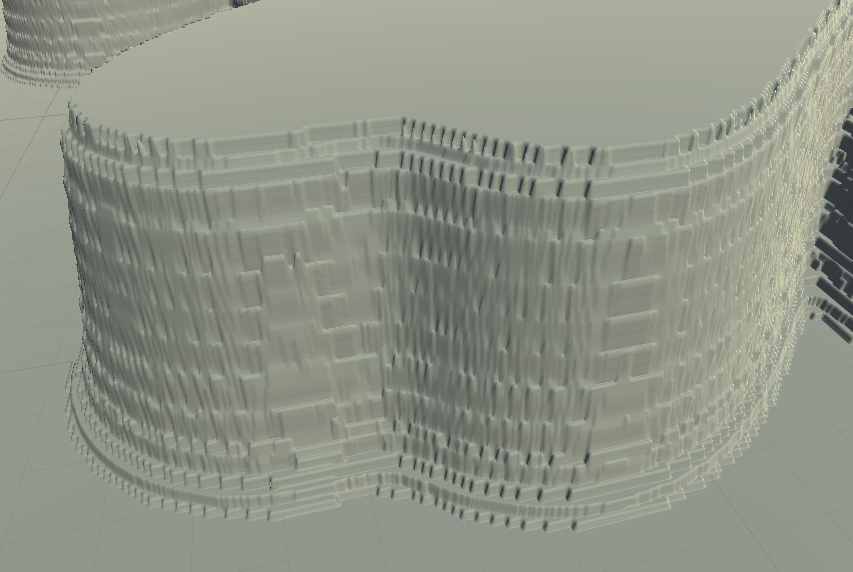
I downloaded the Unity Terrain Tools extension and was able to load my high resolution (4096x4096) onto the Terrain, which, obviously, was way more detailed, not only that, but even my 512x512 map was way more detailed. I’m going to have to look at how they do this.

Ok, I should’ve gone with my other project (multiplayer fighting game). I investigated how the Terrain Tools extension loads a height map (texture) onto a Terrain and it is black magic.

I got it to work somewhat. I took some code from the Terrain Tool and got it to work in my script. Weird thing is, it’s still not right.

Left you can see the Terrain Tool’s result, right you can see the copied code’s result.

Afbeelding met binnen, wit

Automatisch gegenereerde beschrijving

How can this differ so much, as far as I know, I’m doing the exact same as the Terrain Tool. I’m confused.