## Sprint 4 – Implementing the Mesh Generator class to display the heightmap 27/02/2019 – 08/03/2019

### Abstract

The goal of this sprint is to create a class to generate a basic mesh that can be generated during run time and in the unity editor. When the class is created the goal is to generate the noise map with Perlin Noise and add the values to the Y component of the vertex vector.

### Research / Implementation

I already had previously knowledge in how to create meshes in DirectX, and this uses the same principle.

#### Mesh Class

##### Implementation

The first thing when creating the Mesh Class, the author created a simple sketch for the class [Figure 1].

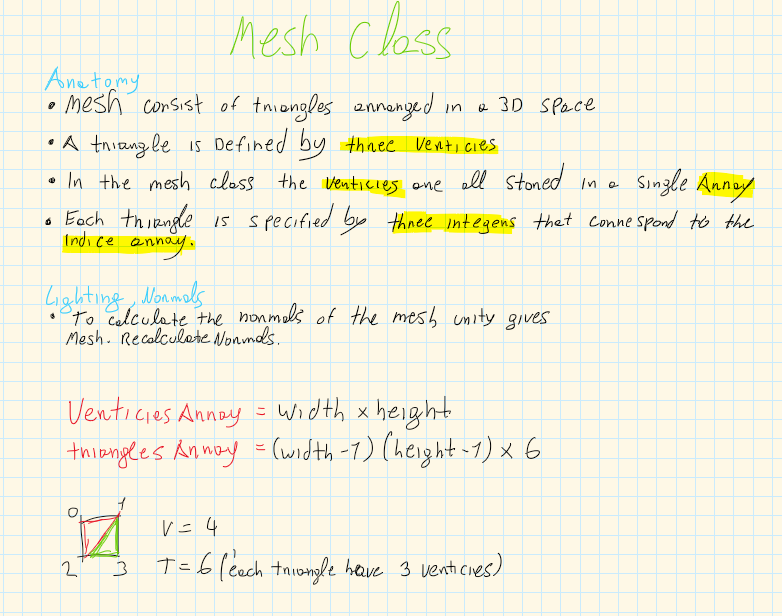


Figure - Mesh Class Sketch

The second thing to make sure is that the empty game object has attached the following two components: Mesh Filter and Mesh Renderer [Figure 2].

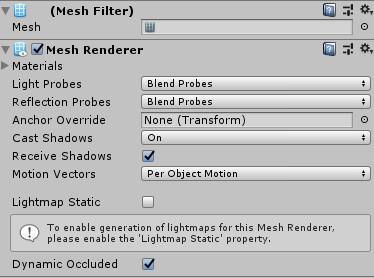


Figure - Requirement of these two components

With the sketch in mind [Figure 1] three initial method were created for the class, SetMeshData(), CreateShape() and UpdateMesh().

###### Set Mesh Data method

This method creates a new mesh, sets the Mesh Filter that is attached to the Empty Game Object to be equals the new mesh and makes sure that the main texture in the Mesh Renderer is set to null. Then it initializes the vertices, triangles and UVs. The initial sketch that the author did to initialize the vertices [Figure 1] was wrong, the fixed result can be seen in [Figure 3], the reason why the improved version has + 1 derives from the fact that the author wanted to have 255 quads when the Map Size was equals to 255.



Figure - Vertices, triangles and UVs initialization

###### Create Shape method

This method is responsible for going through each vertex and UV and initialize them with a defined value and passing the height of the noise map to the vertex [Figure 4].

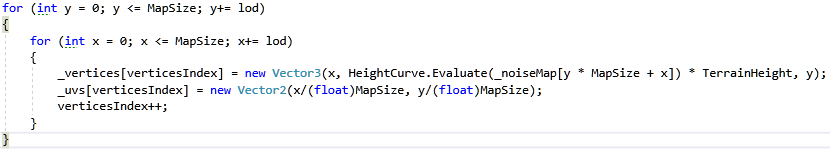


Figure - Vertex and UVs initialization

The other functionality this method is responsible is for creating the triangles for the mesh [Figure 5].

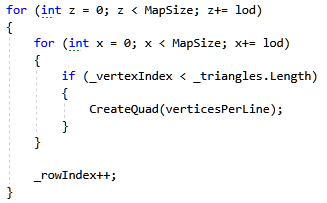


Figure - Triangles generation

#### Problems and Solutions

During the implementation of the class I struggled with different problems. The first problem was found on the initialization of the vertices, for example I created a vector3 named vertices with the size of , let’s imagine that MapWidth and MapHeight have a value of 10, this means that the vector has a size of 100 in total. The only problem is that I wanted to have 10 quads per row, but for this I needed 11 vertices per row. The problem was that the last vertices for the rightest corner quad were present in the other side of the map creating these strange connections [Figure 6].

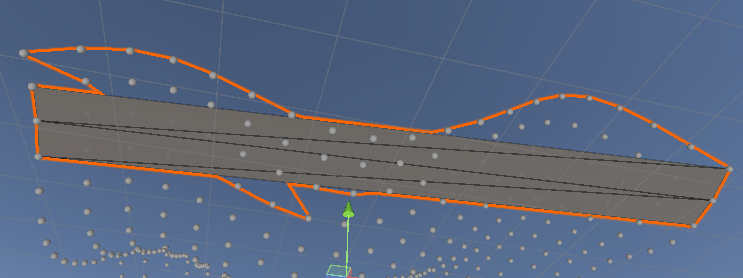


Figure - Generation Mesh Problem

The solution for this was to create the vertices with the size of [Figure 7]

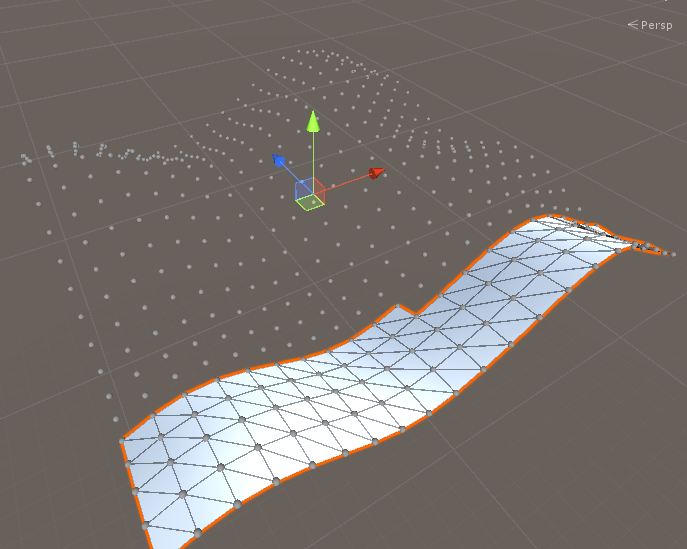


Figure - Mesh Terrain Generation

The terrain generation was looking to soft and far from realistic (Figure 3), to resolve this problem I found that limiting the persistence level between 0 and 1 and increasing the lacunarity to 2.6 fixed the problem (Figure 4).

An animation curve from unity was implemented in the code to limit the height map range (Figure 5). In other words, an animation curve in unity has multiple keys in were the user can control the points in what that curve passes through (Unity Technologies 2018).

### Sprint Review

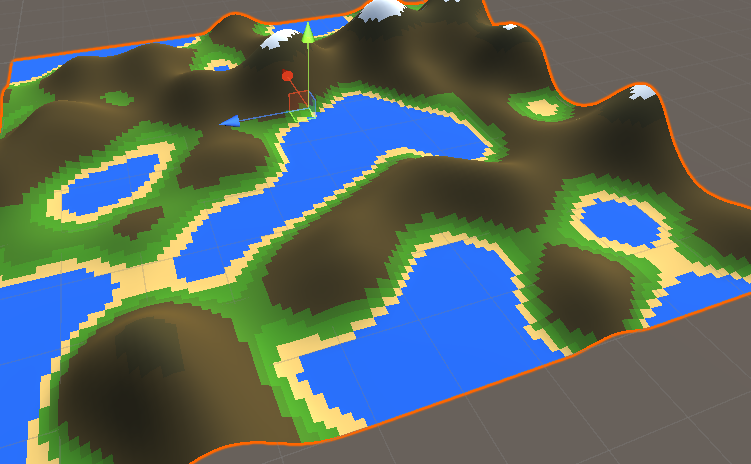


Figure 3 - Terrain Generation with low level of lacunarity

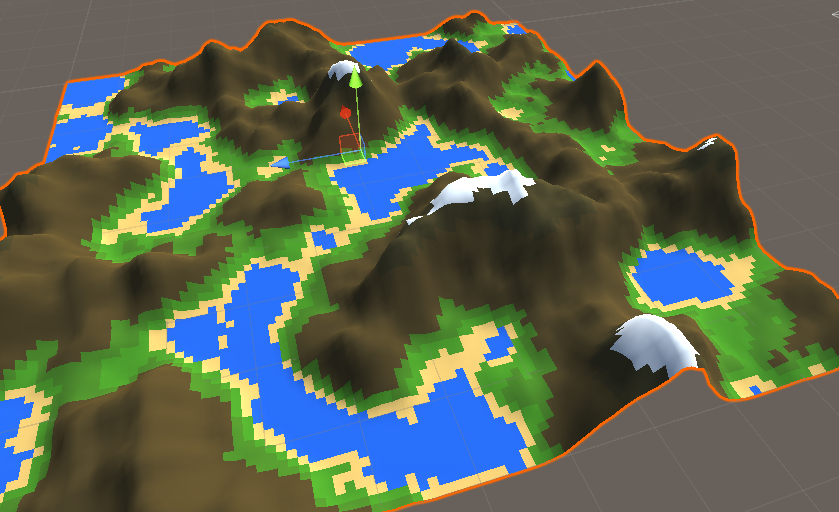


Figure 4 - Terrain Generation with high level of lacunarity



Figure 5 - Evaluate function in Animation Curve

### WBS

1. Research (35%) (8 hours)
2. Create a Mesh Class (50%) (12 hours)
3. Display the terrain using the heightmap inside the mesh class (15%) (3 hours)

### Bibliography

ABHINAV A.K.A DEMKEYS, 2017. *Mathf.PerlinNoise | Smooth Random Values | Unity Quick Tutorial*[viewed 15/03/ 2019]. Available from: <https://www.youtube.com/watch?v=gdSFs0PeBNQ>

BRACKEYS, 2018. *MESH GENERATION in Unity - Basics*[viewed 15/03/ 2019]. Available from: <https://www.youtube.com/watch?v=eJEpeUH1EMg>

BRACKEYS, 2018. *PROCEDURAL TERRAIN in Unity! - Mesh Generation*[viewed 15/03/ 2019]. Available from: <https://youtu.be/64NblGkAabk>

FLICK, J., 2018. *Procedural Grid*[viewed 15/03/ 2019]. Available from: <https://catlikecoding.com/unity/tutorials/procedural-grid/>

UNITY TECHNOLOGIES, 2018. *Using Animation Curves*[viewed 12/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html>

UNITY TECHNOLOGIES, 2019. *Procedural Mesh Geometry*[viewed 15/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/Example-CreatingaBillboardPlane.html>

### References

UNITY TECHNOLOGIES, 2018. Using Animation Curves [viewed 12/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html>