# Sprint 5 – Graphic interface to change the terrain generation on the go 09/03/2019 – 18/03/2019

# Abstract

The goal of this sprint is to create a graphic interface were the user can change the terrain settings during the execution of the program

# Research

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

# Sprint Review

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 to have 10 quads per row, but for this I need to have 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 2).

The solution for this was to create the vertices with the size of (Figure 1)

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).

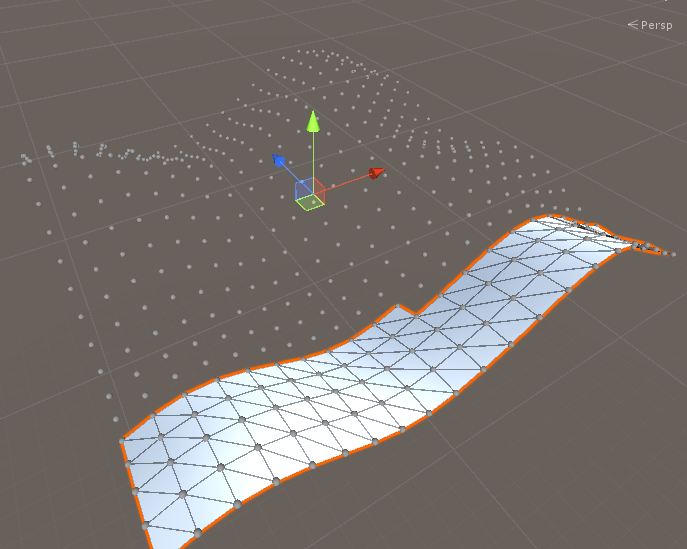


Figure - Mesh Terrain Generation

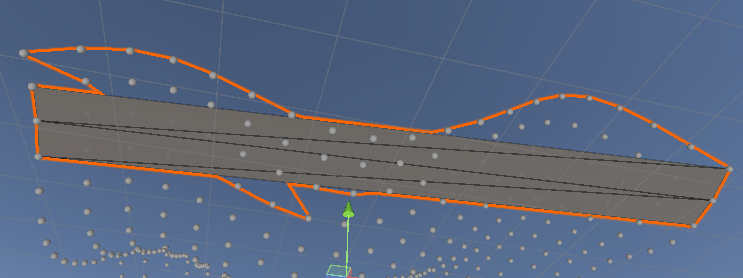


Figure 2 - Generation Mesh Problem

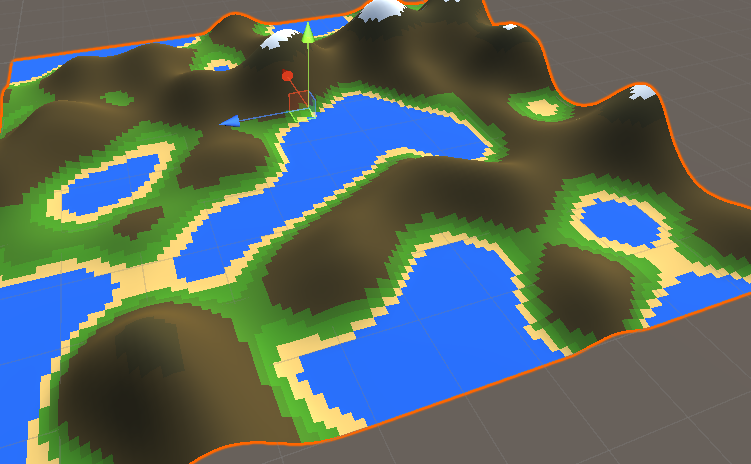


Figure 3 - Terrain Generation with low level of lacunarity

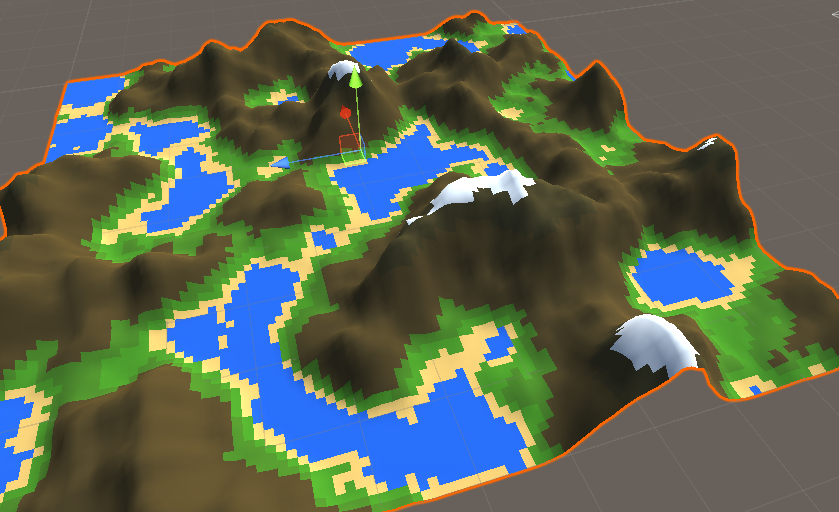


Figure 4 - Terrain Generation with high level of lacunarity



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)

# Reading List

<https://docs.unity3d.com/Manual/Example-CreatingaBillboardPlane.html>

<https://www.youtube.com/watch?v=eJEpeUH1EMg&t=0s>

<https://catlikecoding.com/unity/tutorials/procedural-grid/>

<https://www.youtube.com/watch?v=gdSFs0PeBNQ>

https://www.youtube.com/watch?v=eJEpeUH1EMg

<https://www.youtube.com/watch?v=64NblGkAabk>

https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html

# References

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

2. A **Gantt chart** which captures the sequence in which tasks are expected to be completed during the current timebox, and presents them in an easily understood format. The Gantt should also:

a. Highlight dependencies between tasks.

b. Indicate task progress or completion status.

c. Indicate those tasks which are critical to the phase outputs.

d. Indicate those tasks which were carried forward from the previous timebox.