## Sprint 5 – Graphic interface to change the terrain generation on the go and Level of detail implementation 09/03/2019 – 18/03/2019

### Abstract

This sprint is going to be split in two different sections. The section one is were the author is going to focus on the development of user interface, the interface is going to be simple but effective. The goal is to have a user interface easy to use were the user can contemplate all the changes that the terrain goes through.

The second goal of this sprint is to implement a method to change the level of detail on the terrain, this is needed in case the author decides to implement an endless terrain generation.

### Research / Implementation

The first thing that the author did in the research for creating a user interface was going through the unity documentation (Unity Technologies 2019).

#### User Interface

##### Implementation

With previous knowledge in user interface development for unity and a good look at unity documentation a simple user interface was implemented [Figure 1].

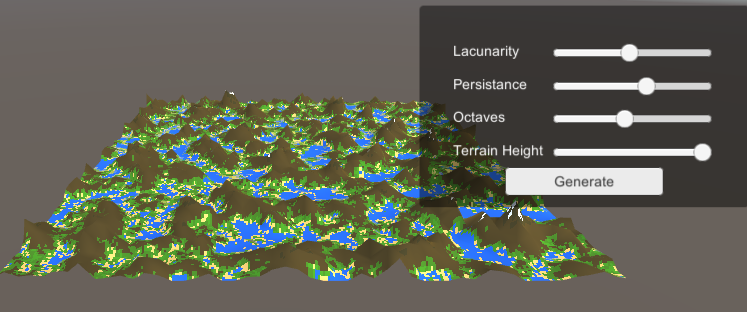


Figure 1 - User Interface

##### Process

The process for implementation of the UI was straight forward, after having the design ready was time to work in the code.

The author wanted to update the lacunarity, persistence, octaves and terrain height variables only when the slider values changed, instead of updating them every frame. To implement this solution, the first step was to create an array of Sliders, this variable is public of type UnityEngine.UI.Slider and is implemented inside the script MeshGeneration [Figure 2].

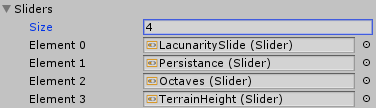


Figure - Sliders Array in Unity

After this the author created inside the start function a set of listeners for each slider that checks when the slider value changes [Figure 3].

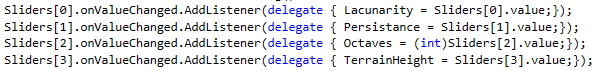


Figure - Listeners

#### Level of Detail (LOD)

When researching in how to implement the level of detail, the author found that unity already had this technique pre-build in the engine (Unity Technologies 2019).

Practically it works by checking the distance each object is from the camera. The farther an object is from the camera the smaller the number of triangles is needed to be rendered.

To use this in unity is necessary to go to a 3D program like Maya or Blender and when creating a mesh set the mesh name for something like meshName\_LOD0 (Brackeys 2018). Each of this meshes have less triangles on them, being the LOD0 the one with the highest triangles count [Figure 4].

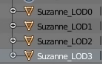


Figure - Name convention in Blender

After the mesh is created an imported into Unity, the engine automatically detects the mesh name and creates something named LOD Group in the game object (Arndt 2014) [Figure 5].

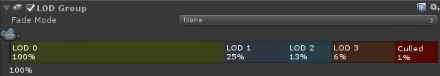


Figure - LOD Group in Unity

This pre-build functionality was good but far away from what the author was looking for, so with the idea of level of detail in mind, he started to implement this technique directly on the mesh generator class.

##### Implementation

The idea was simple, reduce the quantity of triangles to be created in the mesh by a certain amount. When trying to implement the method to reduce the level of detail of the mesh, few problems arise.

##### Problem and Solutions

The first problem took some time to understand how to solve it, every time the author tried to decrease the level of detail of the mesh, the render was totally wrong [Figure 6].

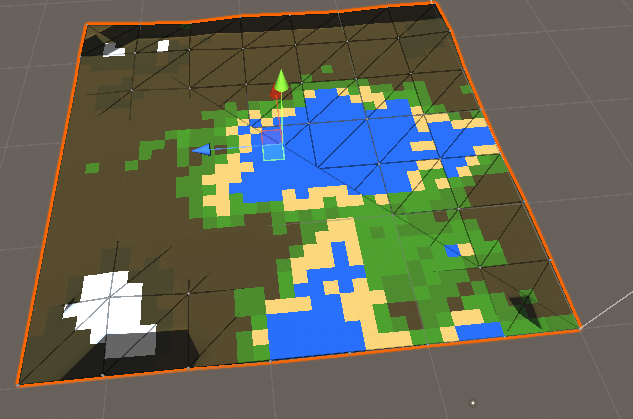


Figure - Mesh Rendering Problem

To resolve this issue, the author created a new method named Update Mesh Data, this method takes an int as a parameter and updates the mesh data by passing the vertices per line of the mesh. The value of the vertices per line is just the Size of the map divided by the level of detail [Figure 7].

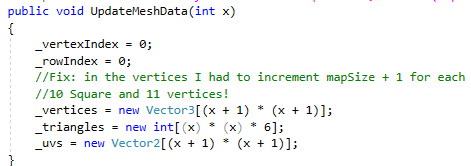


Figure – Update Mesh Data method

The second problem arise just after the first one was solved, now the level of detail of terrain got decreased, it was noticeable to the eye, even because the number of triangles were reduced in relation of the LOD (the higher the less triangles), but there was a problem, some of the triangles on the mesh were not been rendered properly [Figure 8].

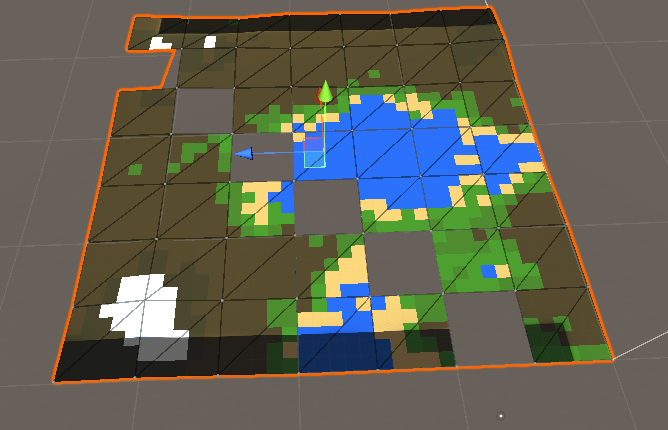


Figure - Terrain Rendering, invisible triangles

To solve this problem the first thing that came to the author mind was googling it and searching for: “how to implement level of detail of a mesh in code” but nothing helpful was found.

The next step was debugging the code and check step by step what was happening. The author then decided to use a sketchbook to draw some programming logic to think through in how the terrain generation was been processed [Figure 9].

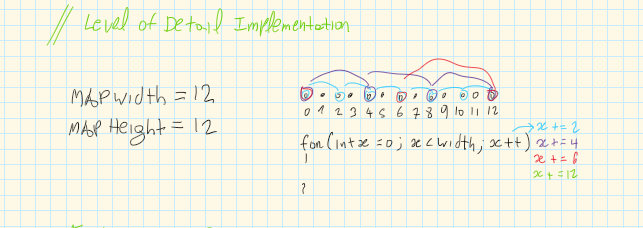


Figure - LOD

Sketching the problem not only helped to figure out the solution but also created a clear view in how to manipulate the triangles in the terrain. So, at this point the author discovered that the formula for the number of vertices needed were:

The author then decided to create a terrain with a fixed size of , to do this it was necessary to know what numbers are divisible by 255 with a result of the division been an integral value. The first thing that came to the author mind was searching for the factors of 255. The divisors of 255 were the following: 1,3,5,15,17,61,85,255 [Figure 10].

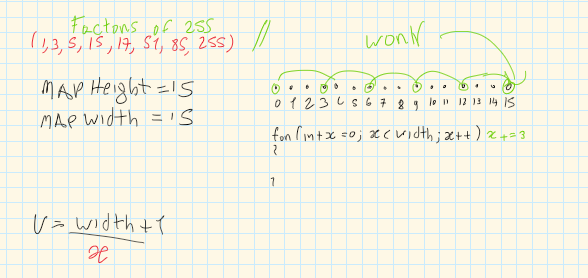


Figure - Factors of 255

Now with the vertex’s formula created and knowing all the factors of 255 with a fixed map size, implementing the final solution was easy. It is possible to see when the author tried to create a mesh with a map size of 15 [Figure 11] that for 15 quads there are 16 vertices and when the level of detail is 5, 4 vertices are created and only 3 quads .

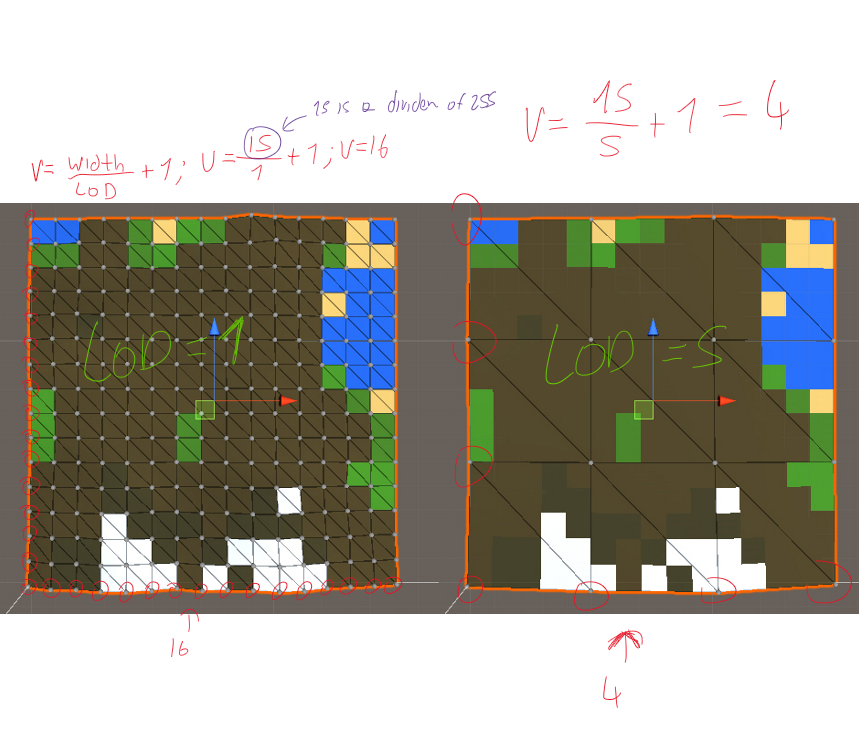


Figure - Level of Detail fully implemented

### Sprint Review

The sprint was a success, the author was able to create a simple user interface to be improved in future sprints and a method to reduce the level of detail of a mesh. This sprint gave the logic needed to the author for creating better mesh manipulation in future sprints.

### WBS

1. Research (30%) (24 hours)
2. Create a user interface (8%) (6 hours)
3. Implement the level of detail functionality (62%) (48 hours)

### Reading List

ARNDT, J., 2014. *Setting up an LOD System - Unity Game Engine* [viewed 11/03/ 2019]. Available from: <https://www.youtube.com/watch?v=IzlU_xvTK3Y>

BRACKEYS, 2018. *MAKE YOUR GAME RUN SMOOTH - Unity LOD Tutorial*[viewed 10/03/ 2019]. Available from: <https://www.youtube.com/watch?v=ifNyVS2_6f8>

UNITY TECHNOLOGIES, 2019. *Level of Detail (LOD)*[viewed 14/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/LevelOfDetail.html>

UNITY TECHNOLOGIES, 2019. *Unity User Manual (2018.3)/UI/Basic Layout*[viewed 10/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/UIBasicLayout.html>

Top of Form

Bottom of Form

### References

UNITY TECHNOLOGIES, 2019. *Level of Detail (LOD)*[viewed 14/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/LevelOfDetail.html>

UNITY TECHNOLOGIES, 2019. *Unity User Manual (2018.3)/UI/Basic Layout*[viewed 10/03/ 2019]. Available from: <https://docs.unity3d.com/Manual/UIBasicLayout.html>

ARNDT, J., 2014. *Setting up an LOD System - Unity Game Engine* [viewed 11/03/ 2019]. Available from: <https://www.youtube.com/watch?v=IzlU_xvTK3Y>

BRACKEYS, 2018. *MAKE YOUR GAME RUN SMOOTH - Unity LOD Tutorial*[viewed 10/03/ 2019]. Available from: <https://www.youtube.com/watch?v=ifNyVS2_6f8>