[-NAME-] TG Brief

# Brief

The [-NAME-] Terrain Generator is a complex modular world generation system that can be used to generate complex terrain, endlessly and randomly. [-NAME-] TG will be completely customizable within the Unity editor. This allows for easier development of and interesting terrain possibilities. Through out this brief I will be using the acronym *TG* to refer to *Terrain Generator*.

# Goals

The [-NAME-] TG aims to provide a quick and easy tool for developers to create procedural worlds for their games. The tools provided in [-NAME-] TG should make world creation a two step process, allowing for quick implementation of other features. [-NAME-] TG should also allow for fine refinement of the terrain parameters to create many different types and styles of terrain.

# Third-Party Libraries

The [-NAME-] TG uses the Unity built-in libraries and the C# Random library. The Unity Math library and Unity Engine library are used to interface with the Unity editor. The C# Random library is used to generate random numbers within a range.

# Maths and Algorithms

* **Noise**

I will be using the Perlin Noise function provided by Unity’s Math library. This will be used to generate the noise maps that form the height map. The noise shape will primarily be controlled by 3 variables; Octaves, Persistence, and Lacunarity.

* **Octaves:** The octaves is the number of layers of Perlin Noise that will be applied to create the height map. Each layer (octave) will contribute to the height map less and will have smaller and finer details than the last octave.
* **Persistence:** The persistence is the amount by which each octave decreases its contribution. Persistence is applied as exponential decay, meaning that with a persistence of 0.5 each octave will contribute half as much to the final height as the last octave.
* **Lacunarity:** The lacunarity is how much of an increase in detail each octave will have. Lacunarity is applied as exponential growth, meaning that with a lacunarity of 2 each octave will have twice as much detail as the previous octave.
* **LODs & Interpolation:**

The height map that is generated by the Perlin Noise does a good job of creating terrain that looks realistic from a distance. However, if we were to add a player straight onto this terrain the effect will likely be lost due to things like polygon count limitations and unrealistic scales. To solve this problem I will be implementing a system of LODs that allow for higher amounts of resolution close to the player but lower resolution as the chunk gets further from the player.

* **LODs:** The main idea behind LODs is skipping every i vertex, i being the number of vertices to skip. The problem with simply skipping a number of vertices is that depending on the number of vertices and the value of i, it is possible that an “out of bounds Exception”. The solution to this problem is to use LOD levels that have values for i that are a factor of the edge length of each chunk, defaulted at 255 which is the maximum length that supports Unity’s cap of 65,535 vertices per mesh.
* **Interpolation:** Interpolation can be very useful for smoothing out terrain as the player get closer. The main use of interpolation in this project will be to create higher resolution textures for the terrain.

The

* Editor modifiable noise settings
* Editor modifiable noise curves

BIBLIOG

<https://gamedevacademy.org/complete-guide-to-procedural-level-generation-in-unity-part-1/>

<https://www.youtube.com/watch?v=CSa5O6knuwI>

<https://www.youtube.com/watch?v=417kJGPKwDg&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3&index=6>

<https://www.youtube.com/watch?v=xlSkYjiE-Ck&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3&index=8>