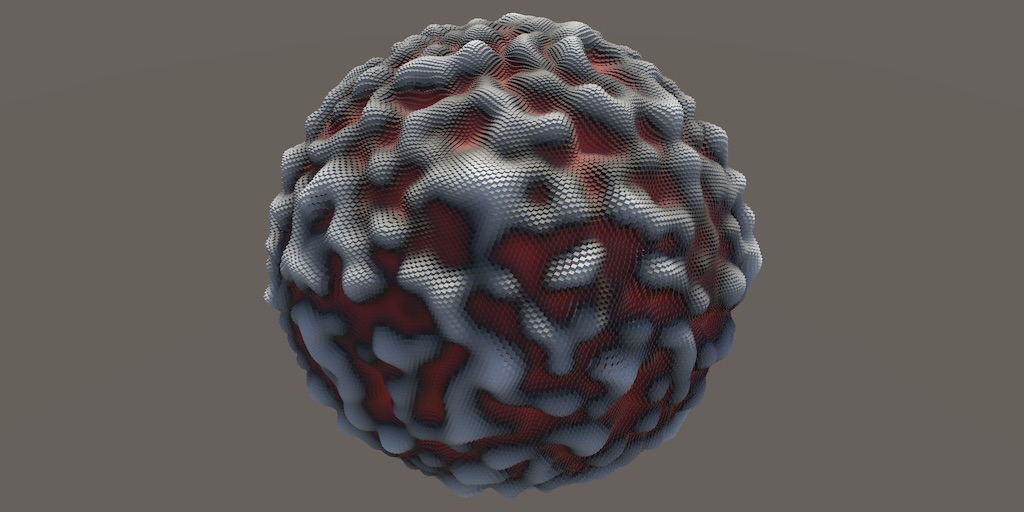
Alec Walker

Complex Games Assessment Task 1

**Goal of System:**

The goal of this system will generate generic mesh data based on a modified Simplex Perlin noise algorithm. This demonstration will focus on generating a bumpy / rough sphere as shown in figure 1.0 with the intent of saving the data in the .fbx file format.

**Dependencies:**

The project will rely on Unity libraries and some system libraries, specifically:

Figure 1.0

* UnityEngine
* UnityEditor
* UnityEditor.Formats.Fbx.Exporter (for mesh exportation)
* System

Even though the project will rely on Unity, the result shouldn’t. This is because as previously mentioned the end result will be in a . fbx format which can be displayed by any . fbx supported software.

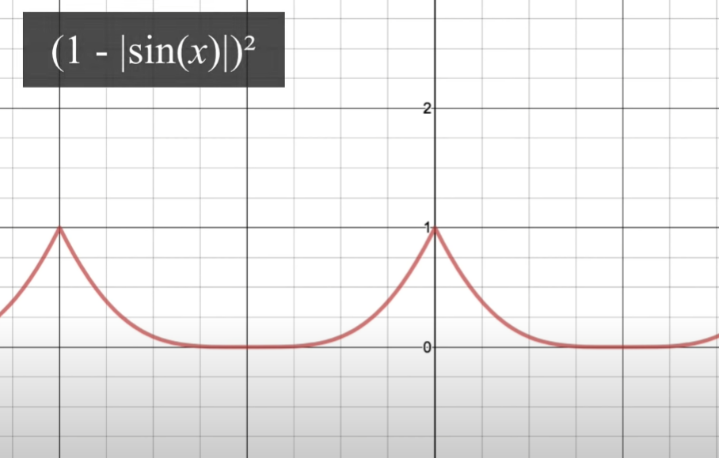
**Algorithm Being Used:**

The primary algorithm used in this system is a 3-dimensional Simplex Perlin noise algorithm. This algorithm outputs values in a range of -1 to 1 where I will linearly interpolate the results to be within the values 0 to 1. There is also no guarantee that results of this module will land exactly between -1 to 1 so before “lerping” them they are clamped to a minimum of -1 and a maximum of 1. The final result will be a point at a given distance away from the centre point of the sphere, neighbouring points will follow the Simplex Perlin noise algorithm.

**Mathematical Operations:**

Once I have received point data from the Simplex noise algorithm I’ll create a few noise filters out of them. I plan on creating two for this demonstration but will keep the system modular in such a way more types of noise and different implementations of Simplex noise can be added. There will be a “SimpleNoise” filter and an “InverseNoise” filter which will both inherit from an “AbstractNoiseFilter” class. The only difference between these two classes is how they evaluate a points location.

**SimpleNoise:**

This will take a given point, run it through the Simplex Noise script. Then to allow for more control of the results I plan on adding an “Origin” vector 3 variable which can be added after the calculation to displace the noise. There will then be additional variable like “Strength”, “Roughness”, etc. which will be multiplied by the initial point to create different results. I may add more variables when implementing this function to increase the range and difference of results.

**InverseNoise**

This will act in the exact same way as **SimpleNoise** however when running the initial point through the Simplex Noise script, I will attempt to modify the result before tampering with it. The goal is to generate data as seen in figure 2.0, where “sin(x)” is the result of a point being ran through the Simplex Noise algorithm.

Figure 2.0

**Modular Design:**

The project will be made modular by allowing users to simply install a unity package to a given project and start creating planets within their scenes. These planets can then be saved and sent anywhere with the exporting feature to be given custom materials or more finely edit the results. I believe this is a modular approach as a majority of 3D art software supports the reading and writing of .fbx files.

**Integration:**

I’ll integrate this system into unity and the results it’ll produce into my previous OpenGL application as a proof of concept. If I have enough time I’ll also experiment with handing off my assets to an artist/designer to test the painting / creating of materials in third party software such as Autodesk’s Maya, Epic Game’s Unreal Engine and / or the open-source program Blender.