Taylor Cronk
EECS 494, Project 2, Prototype 1
Tool: Random Terrain Generator

Random Terrain Generator

I'm a huge fan of randomly and procedurally generated content, so I figured that I would make some sort of tool that randomly generated content. Eventually I settled on the idea of a random terrain generator. The main idea came from Minecraft, and all of the Minecraft-esque games out there that use random terrain generation. It was an idea that I always wanted to play around with, but never had a graphical viewpoint of what I was generating until I started using Unity.

Starting out, my first thought was to make a script that you would attach to a terrain object. This idea was quickly scrapped when I found out you could create your own Unity windows and I began to use that instead. I found it to feel more like a Utility, something you would find packaged with Unity. The first version contained only the Random Generator, in which I generated random points for each of the map points. The first version showed me that what I wanted to do was possible, as I learned how to change the terrain height map from code and how to add set-able parameters to the window that I could pass into the algorithms.

Once I got together my system for loading in the algorithms I decided that I should add more types of terrain algorithms. For this I did some research into terrain generation and noise algorithms. The first one I implemented was the Diamond-Square Algorithm. I chose this for the general simplicity of implementing the algorithm and the practical results it gave. However the algorithm did not feel very customizable. I was able to add two parameters, size of the starting squares and the rate at which the scale changes. In the end I was not happy with the overall customizability. This led me to search out another algorithm that had more control to it.

The next algorithm I implemented was the classic Perlin Noise. The algorithm was more complicated than Diamond-Square, but very well documented. The algorithm gave a lot more expression to the generation, allowing for some more interesting results.

I felt like the generator was missing something and had my roommate look at project. He noticed that there was a lack of cliffs, at this point everything was smooth slopes. Upon researching this idea, I found the idea of Erosion Algorithms described in the paper linked in the C# files for the erosion algorithms. These algorithms would allow me to give the user the ability to add in cliffs if she wished too. I proceeded to add two types, Thermal and Improved Thermal. Both give different and interesting results that users should find fun to play with. Next I added the ability to smooth the landscape on command, which helped with unwanted jaggedness from the erosion algorithms.

During my roommates testing of the tool, I noticed that he would make something that he liked the look of, but then add too much erosion. This would annoy him, as it just destroyed what he was working on. After seeing this I added an undo button to help retrieve some of you work if you find that you no longer like the look of that last erosion, or maybe you hit a stray button. Adding this functionality helped my roommate enjoy playing around with the tool more.