Matthew Emerson
Graphics Final Project Reflection

The goal of my project was to create a random map generator. Some of my goals for the end product were to be able to generate water, rock and grass as well as placing trees and creating fires that could burn the trees. Part way through the development process I decided I would work towards other goals rather than add fire. Instead of just adding random extra features, I decided to try and see in what ways I could make the environment more lifelike. Some of the ways i improved the environment was through making variable grass tiles, by generating faded edged tiles for bodies of water and creating lighting effects for the mountains. I also animated the water and created a day/night cycle.

I decided I wanted to make a random map generator because I enjoy gaming and similar systems are used in strategy and sandbox type games quite often. I have an interest in developing games and so I figured creating something I could extend in the future would be a good choice. There is no interaction in the environment as I instead focused on trying to make the environment less flat.

When beginning the project, I started working on an algorithm that I had thought of myself. I thought it would be a good way to generate coherent noise for generating maps, but I soon realized that the geometric properties of the equations I used created to many repeating patterns in the terrain. Now, almost everything generates using some form of cellular automata.

When i begin making the board, I create a 2d vector of integers. I then generate a random value for each entry in the vector between 0 and 255. After all of the number have been generated, I smooth the numbers by averaging them with the cells directly adjacent to them. This creates the land and the water. Initially I also had this step create the mountains, but they came out looking very fragmented and small. To create the mountains I create a random number of drunken walkers that pick a random direction to move each turn and deposit a large amount of sediment (increases number of heightmap) in the new space. After all of the drunk walkers have ran out of soil, then board is averaged out to smooth the resulting patterns. This proved to be a fast and efficient method for generating large connected areas of mountain. I then made an algorithm that would go through the board and remove sections of water that were below a certain size and since there were too many lakes around the map I also delete every other lake.

The next thing I implemented was a day night cycle for the game. I did this by looping through all of the vertex arrays and changing their colors towards black, but leaving more blue in the rgb values. After creating the day night cycle, I also made certain grass tiles have a different texture to give some variation. Next, the trees were added. I generate a random number of forests in the game map. For each one, I start with a single tree and simulate a certain amount of time. Each tree has the possibility of creating a new tree in the outside ring of a 15x15 square. I did this so the trees would have more of a tendency to spread out since letting them spawn new trees closer to themselves created very dense forests that covered small areas. When the map is fully generated, the trees can still reproduce, although with a slightly lower fertility rate.

One of the last things I was able to implement was different colored edges for features. For the mountains I made edge pieces either lighter or darker to simulate lighting on the mountain edge. For the water, if it was next to grass, I made the vertices on the grass side transparent and drew grass underneath the water tile to create a smooth transition effect between tiles. I also made a function that changes the texture of the water tiles periodically to make it animated.

Testing took the longest time because a lot of the program was trial and error. There were few times that the code didn't work at least close to how I expected it to, although there were many times that the generated results were not desireable. One thing that could use improvement is the blending on the water tiles. Certain patterns create jagged edges. With the method I chose, I don't believe there is anything I could have done to rectify the situation. I think if i were to try and make a real game I would either manually draw transition tiles or I would have to create four triangles instead of two for each tile so I have more control over the color gradients. Another thing that does not work as expected is the floodFill algorithm [1]. I modified an algorithm I found online so that it would return the number of nodes that had been visited. The algorithm tends to return too many nodes and so the number is incorrect. As you increase the threshold, the size of lake which is deleted is also increased still so even though the numbers aren't exact, i was able to achieve the desired result through some trial and error.

While programming this project I learned that true randomness often creates the most organic looking results. I also learned that you can create a lot of effects through creative use of the modulus operator. I learned a bit about how to blend textures together in code rather than manually drawing them. I do think however that if I took time to create a fragment shader I could create even better blending effects. I didn't use any objects really since I thought it would be unnecessary bloat and we needed to handle a lot of data. Objects are tracked through integers in 2d arrays. Had I been even more serious about optimization, I could have used a short int to conserve memory since I didn't have many objects.

References

[1] Flood Fill Algorithm. Darshan Gajara. Pracspedia.
http://www.pracspedia.com/CG/floodfill.html