Cmpt 406 Final Project: Voronoi Diagrams

Rafiq Dandoo

Project Overview:

In this project I was attempting to create a Voronoi map generator that would create maps based from randomly placed nodes. The original idea was to take the centroid of the polygons generated by the Voronoi diagram and create a new Voronoi diagram based from those points, iterating over and over to normalize the shapes. After that the plan was to use some form of graph searching algorithm, most likely a Breath First Search to create terrain elevations in the Voronoi regions.

This overall project had too large of a scope after realizing how much work was involved with just getting fortunes algorithm to work, so I had to scale back the project to just implementing a version of Fortune's algorithm.

What I did manage to accomplish was a successful implementation of Fortunes algorithm that does generate Voronoi diagrams from randomly placed points and it works for very large numbers of points quickly.

Challenges:

To start off, the biggest challenge that made a lot of the other things harder is the whole debugging process to figure out where something is wrong and being able to visualize it. When you get an error, tracking down what that error was caused by can be very time consuming because of the nature of the algorithm and having a visual output means that stepping through the code and reading the numbers makes it very hard to visualize if something is wrong or right.

There were a lot of challenges to complete this, one of the big ones was that I couldn't collect the faces of the Voronoi regions property which prevented me from moving on. I had a data structure of "line segments" that were meant to be like the doubly connected edge lists the book went over but it turns out that implementing it can be much trickier. I think if I had another week I could get it working but I think I set my scope a bit too large for this. One of the problems with collecting the regions is also the unbounded ones, setting a bounding box around the area to close off points is harder than it seems on paper.

There were a few edge cases that caused a bunch of problems that either required some ugly fixes or minor inaccuracies added, such as having division by zero errors because of how two or three points are positioned, so to fix that I would just slightly tweak the positioning by adding a very small error.

What I Learned:

I learned that Implementing visual geometric problems can be much harder than one would think going into this. I did learn how to implement the sweep line / beach line using heaps and using an almost doubly connected edge list I stored the line segments, so many of the data structures that we covered in class I got to implement and use which was interesting.

Things to Work on:

I'd want to continue working on this and progress it, it's a fun hard problem to do. Knowing what I do now I would most likely go back and refactor some of how I setup my data structures to make gathering regions a bit easier. Plus, there are a few ugly sections of that I think with some more time could be cleaned up a bit better.

Conclusion:

Overall, I enjoyed the project and I think it was a good learning experience on implementing something we learned the theory for and it was a challenging and rewarding task.