CSE306: Second Assignment Report

Duc Hieu Le

1. Introduction

In this project, I implemented the following geometry processing features:

- Voronoi diagram computation
- Power diagram
- Gallouet-Merigot incompressible Euler scheme

2. Code structure

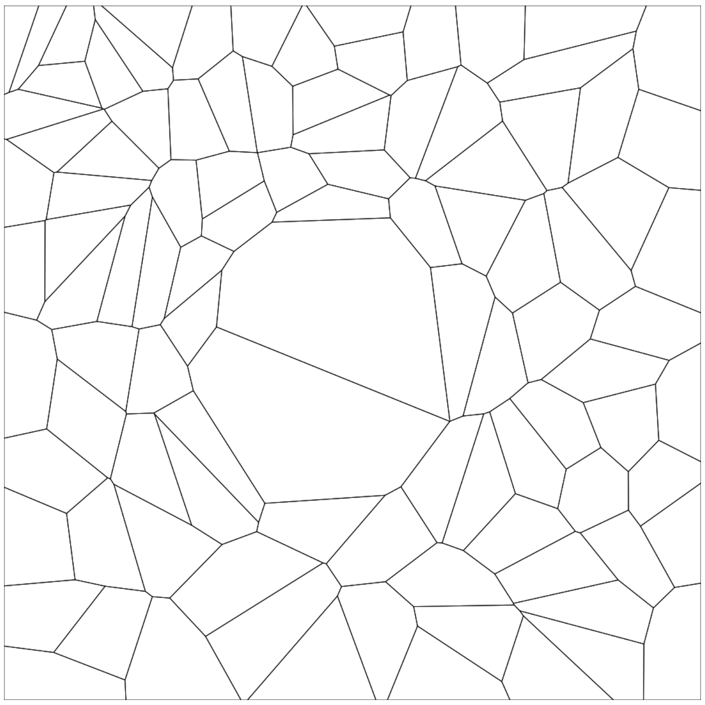
To test the code, simply just run the script run.sh

- **vector.cpp**: contains the class **Vector** with useful vector-related functions, this time it only represents 2D vectors.
- **polygon.cpp**: contains the class **Polygon** and **Triangle** with related methods like **get_area**, **get_centroid**, **clip**, etc.
- **svg.cpp**: contains functions to output a list of polygons in svg format
- **utils.cpp**: contains utility functions such as **compute_vor**, **lbfgs** and **gallouet_method**

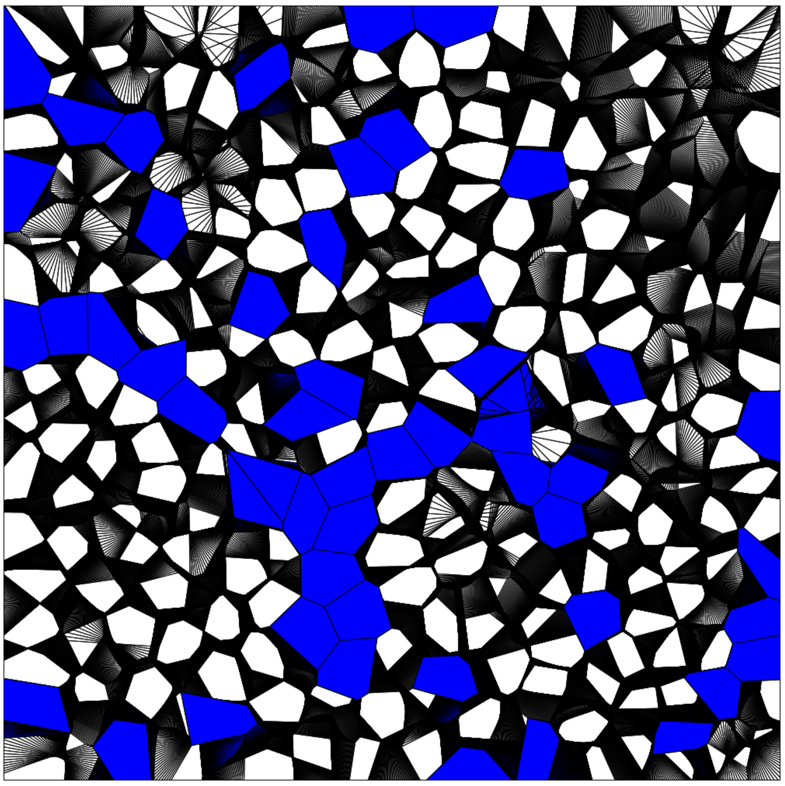
3. Result SVGs

All rendering were done on my laptop, with specs:
- Processor Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz
- 16GB Ram
- Intel(R) UHD Graphics 620

By the instruction of the course, I implemented my Polygon class with the method clip that uses Sutherland-Hodgman polygon clipping algorithm.
Then I implemented Voronoi Parallel Linear Enumeration: for each point, starting with a large square being iteratively clipped by half planes representing the neighboring cells. I create squares and clip the polygon using the Sutherland-Hodgman algorithm.
After that, I implemented LBFGS algorithm to produce a power diagram. I was not able to use the given library correctly, so I implemented a gradient ascent version according to the lecture notes.
Here is the diagram I got, where the mass attributed to each point was following an exponential decay, with the distance from the center. Thus, it will produce larger polygons in the center.



Then, I implemented the Gallouet method to simulate fluid. My function takes a list of points, a list of velocity of these points, and a list of masses of these points and will output a list of new positions and a list of new velocities.
My algorithm considers the area of a water polygon the same as an air polygon. I used a uniform repartition and followed the code given in the lecture notes to simulate the system overtime, integrating forces and velocities step by step. Here is the result svg with 60 frames of simulation.



It looks better in svg so please look at the water.svg file in the repo to see it in a better form