CSE306 Geometry Processing Project Report

Jasmine Watissee (BX24)

June 2024

1 Introduction

This project is aimed at implementing an optimized Voronoï diagram as well as a Fluid simulator in C++. This report contains a longer description of my work which is present on my GitHub (https://github.com/jasminewatissee/CSE306-Computer-Graphics). I will describe in more details the steps and features of my project.

2 Code Organisation

All of my code is present in main.cpp. It is organised in classes: one for the Polygons, one for the Voronoi diagram, one for the optimisation and one for the fluid. You can use make to compile the project and then use ./geo ¡opt¿ to run it. ¡opt¿ corresponds to a umber between 0 and 3: 0 being a Voronoï diagram, 1 being a power diagram with optimised transport, 2 again an optimised diagram but this time with circles replacing the cells and 3 for the fluid simulator. The number of cells or other parameters like this can be modified directly in the code.

I also want to mention that I kept multiple versions of the same functions to be able to run these four options. Hence, I have a function compute() and a function compute_circles(), first one only clips the polygons and the second clips to an approximation of a circle. I also have two functions optimize, one simple to optimise the power diagram and one for the fluid taking in consideration the weight of the air.

As a side note, my code is highly inspired from what was coded in class by the professor.

3 Voronoi

Voronoï Diagram was implemented during TD6. A Voronoï diagram is the partitioning of a plane with points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other. We created a class Voronoï with a function compute to create such a Voronoï Diagram based on a set of points (generated randomly). It uses the Voronoï Parallel Linear Enumeration algorithm and the Sutherland-Hodgman algorithm which allows us to clip polygons together.

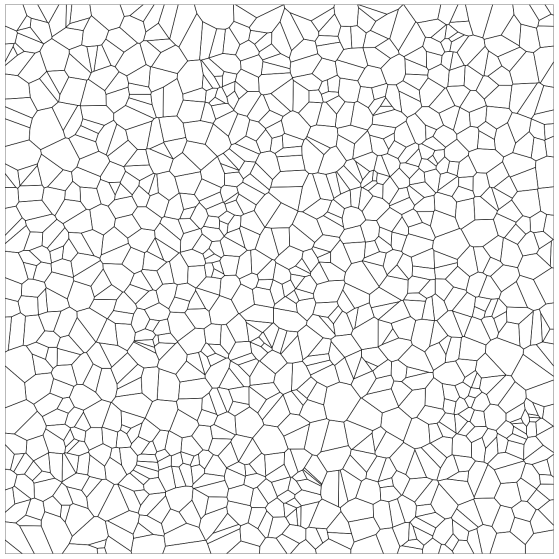

Figure 1: Voronoï Diagram for 1000 points, execution time: ¡1s.

4 Semi Discrete Optimisation Transport

During the second lab, we implemented a Semi Discrete Optimisation Transport with Power Diagrams, which gives us a way to change the size of Voronoï cells based on given weights for each cell. To optimise the weights of the power diagram we use the lbfgs library and followed the gradient descent as described in the lecture notes. You can see an example in Figure 2 of a Power Diagram with areas proportional to 1 for all cells.

5 Fluid Simulator

Finally, in the last lab, we implemented a fluid simulator. First, we transformed the cells into approximation of circles to make particles. In Figure 3, we can see a simple optimised diagram of circles instead of cells. Then, to animate it, we followed the Gallouet Merigot scheme described in the lecture notes. At each step, we update the position of the fluid particle based on the two forces working on it: gravity and the string force. After each time step, we save the

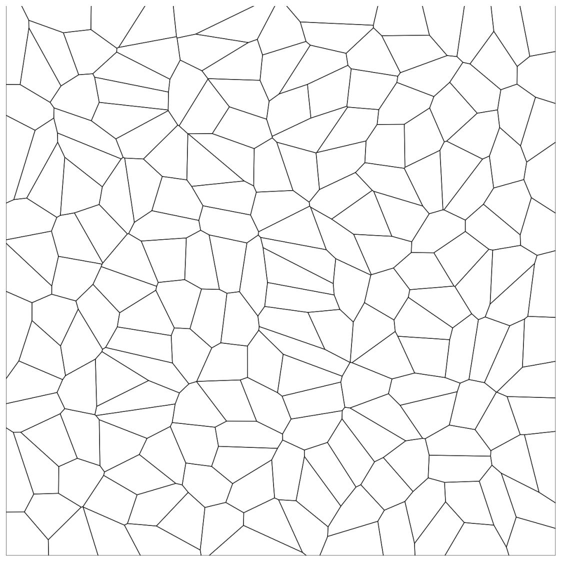

Figure 2: Optimised Power Diagram with 200 points with weights of 1, execution time: 4s.

frame and from this, I was able to create an animation present in my repository like rain drops falling. Here is in Figure 4 an example of a frame.

6 Feedback on the class

I found the course really interesting. It was sometimes a bit difficult as I don't have strong notions in physics, but it was really interesting to see how it can be applied in Computer Science to create graphics.

I liked working on the first project, I mainly followed what the professor was coding in TD but then worked in my own time on optional parts. The subject of the first project was broad and there was a lot of place for creativity and creation with the optional parts which I liked.

I found the second project a bit harder, I found the lecture notes less helpful to implement it and relied strongly on the code of the professor to implement my fluid simulator.

One last comment I want to make is that the TAs weren't very helpful anytime I had a problem.

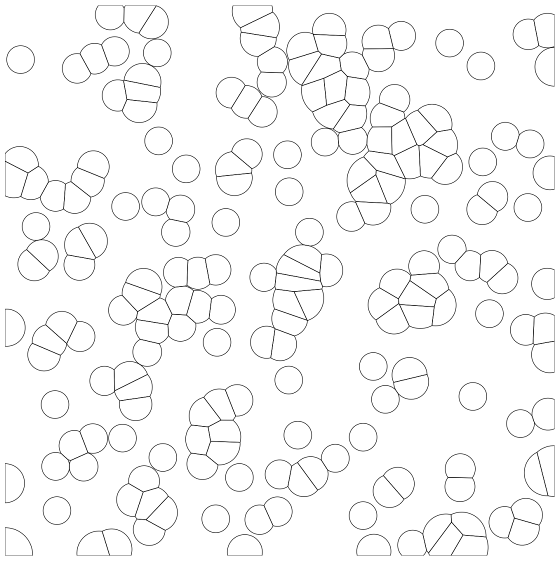
Figure 3: Optimised Power Diagram with circles, execution time: 9s.

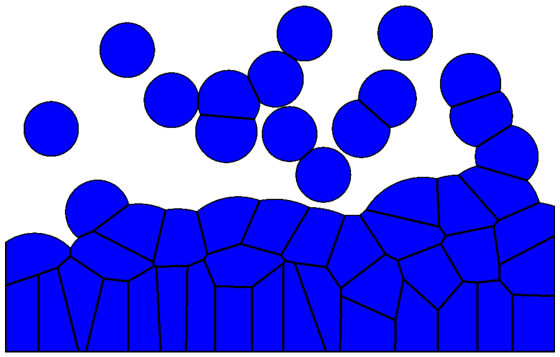
Figure 4: Frame of the Fluid Simulator