Computer Graphics Fluids

Bogdan Budura

June 2024

1 Introduction

This report summarizes my work over the second half of the semester, which was dedicated to building a fluid simulation program. I will introduce each feature I have implemented step by step, accompanied by images and execution times. Additionally, I will discuss the challenges I faced while implementing these features and the lessons I learned along the way.

2 Voronoi Diagrams

In the initial phase of my fluid simulation program, I set up the foundational scene using Voronoi diagrams. Formally, a Voronoi diagram for a collection of n points (referred to as sites) divide the plane into n regions. Each region is associated with one of these points and encompasses all locations that are closer to it than to any other point in the set.

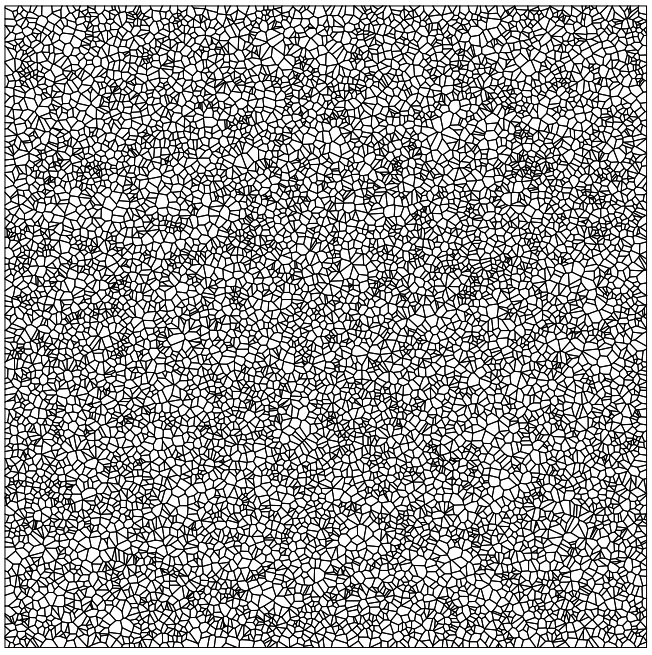


Figure 1: Voronoi Diagram with 10000 points. The time it took to generate this image was $3.08s$, running the code in parallel with no extra optimizations.

3 Power Diagrams

For the next part of my project I implemented Power diagrams. Formally we can define power diagrams in the following way. Given a set of points $\{P_i\}$ in a plane, where each point P_i has an associated weight w_i, the power distance from any point x in the plane to P_i is defined as:

$$\pi_i(x) = \|x - P_i\|^2 - w_i$$

where $\|x - P_i\|$ is the Euclidean distance between x and P_i, and w_i is the weight of the point P_i. The power diagram is then a partition of the plane into cells, each associated with one of the generator points P_i. A point x belongs to the cell associated with P_i if:

$$\pi_i(x) \leq \pi_j(x) \quad \text{for all} \quad j \neq i$$

This means that the cell associated with P_i consists of all points in the plane for which P_i is the closest generator point according to the power distance. The reason why this characterization is really wonderful is that in this way we can impose special conditions on the partitions formed by the sites. One particularly interesting type of partition occurs when we require that all the partitions have the same area. We shall see in a bit why this is useful and how we can use it to achieve our goal of creating a fluid simulator in 2d. The way I implemented this in my project was using gradient **ascent** or **descent**, depends how one likes to see it, using LBFGS. I have implemented this in my fluid simulation project by using *LBFGS*.

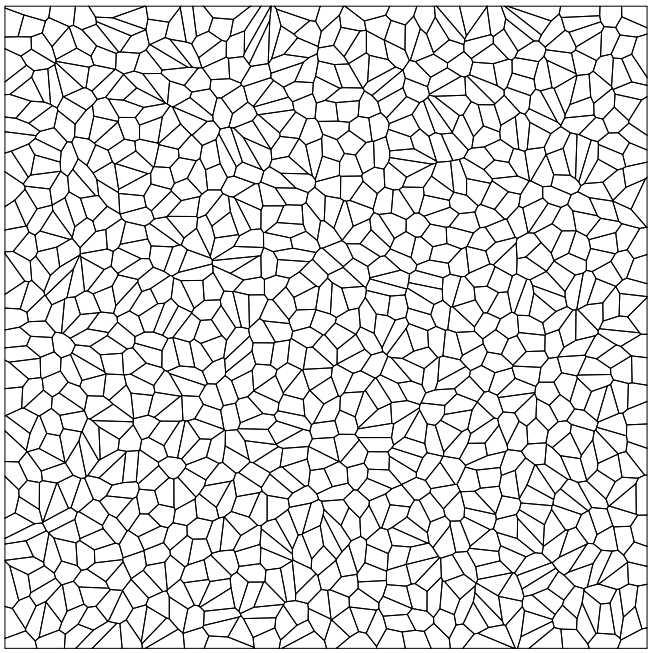


Figure 2: Power Diagram with 10000 points. The time it took to generate this image was $8.04s$ running the code in parallel with not extra optimizations.

4 Fluid Simulation

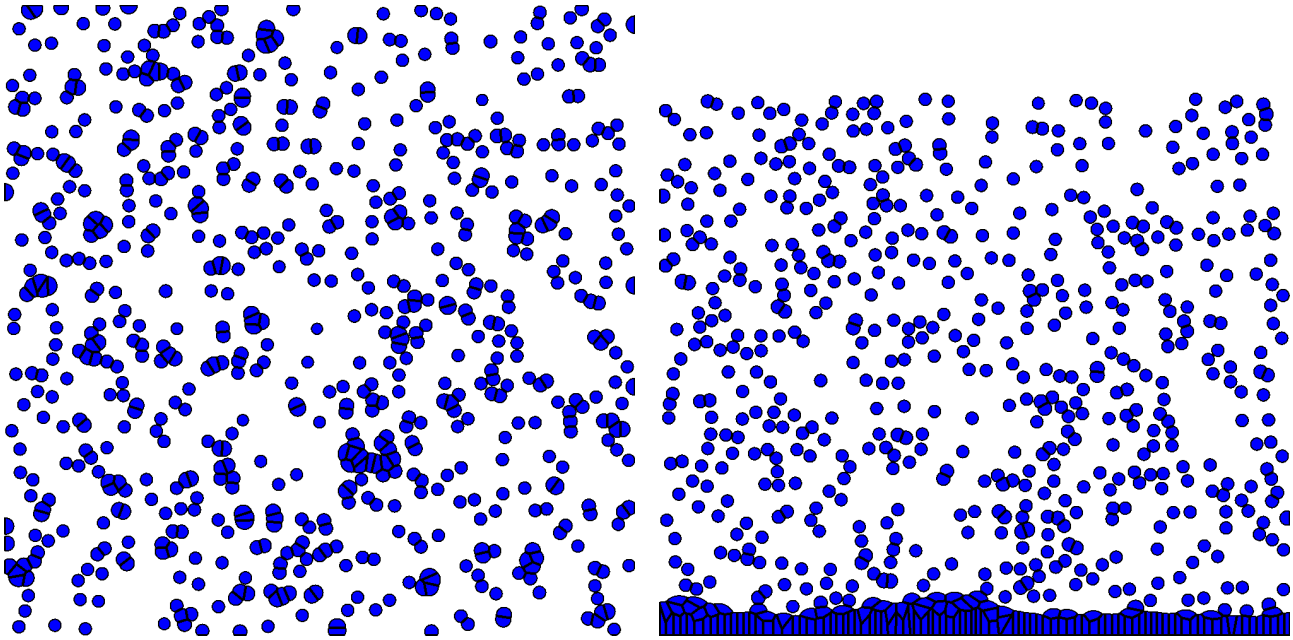
The objective of my fluid simulation project is to simulate droplets and their movement within a two-dimensional space. For this part, I built upon my previous work on Voronoi diagrams and power diagrams. Specifically, we approximate our droplets as polygons with a high number of sides. For this project, I chose to use polygons with 100 sides, which effectively resemble circles. For this part of the project, we are heavily reusing the previous implementation of the power diagram, however, with one important change. Instead of creating the power diagrams with respect to bounding square with unit coordinates, we are now creating the power diagrams with respect to a circle centered at the sites and with a radius of the form $R = \sqrt{w_i - w_{air}}$. As before, I run the LBFGS algorithm on this input to generate droplets of equal size. After each generation, I update the position of my points by modifying their velocity according to spring forces. The equations I used for this part are as follows:

$$\mathbf{F}_i^{\text{spring}} = \frac{1}{\epsilon^2}(\text{Centroid}(\mathbf{V}_i^W) - \mathbf{X}_i)$$

$$\mathbf{F}_i = \mathbf{F}_i^{\text{spring}} + m_i\mathbf{g}$$

$$\mathbf{v}_i' = \mathbf{v}_i + \frac{dt}{m_i}\mathbf{F}_i$$

$$\mathbf{X}_i' = \mathbf{X}_i + dt\,\mathbf{v}_i$$



(a) Inital Setup (b) Evolution after some δt

Figure 3: A snippet of the fluid simulation's runtime. The configuration I am using for the main fluid simulation of my project is $m_i = 200\ \forall i\ \in P,\ \epsilon = 0.004$, $\delta t = 0.002$ on 700 points and 500 frames. I ran this simulation in parallel on 8 cores and the total execution time was 1 hour 22 minutes. The link to the fluid simulation can be found here FluidSimulation.

5 Feedback

I think this class was great and encompassed many new topics that I had never had the pleasure of working with before. The pace was good, and the materials were of excellent quality. I do not have anything to complain about.

6 Conclusion

This project was quite enjoyable and generally straightforward. However, there were specific instances where extensive debugging was necessary. One notable challenge involved the LBFGS algorithm for the Power Diagram implementation. Initially, I generated my clipping polygon in a clockwise direction instead of counterclockwise, causing my formulas to malfunction and preventing LBFGS from converging. Besides this error, the most significant issue I encountered was with LBFGS in the final part of the project, related to fluid generation. Fortunately, this was due to a variable naming mismatch, which was easily resolved. Nevertheless, the satisfaction at the end was worth the effort.