CSE306 Computer Graphics

Report on Lab 6 - Lab 8

Introduction

This program implements a basic 2D fluid simulation using centroidal Voronoi and optimal transport methods. Particles are initialised randomly within a unit square and are iteratively guided toward equilibrium positions using a simulated spring force model. Each frame of the evolving fluid distribution is saved as a PNG image, allowing the simulation to be visualised as an animation. The code for the labs is available in my git-hub under the folder `FluidLab`.

I used the following gitbub to check for bugs whenever I got stuck: https://github.com/salmazainana/Computer_graphics-Project2/tree/main

Further, I want to thank Andreea Patarlageanu for taking a lot of time to help me implement the fluid simulation.

Voronoi Diagram

A Voronoi diagram partitions space into regions where each region contains all points closest to a specific generator point. In the code, fluid particles act as these generator points, and a weighted Voronoi diagram is computed using their positions and weights. The `compute()` function constructs these regions as polygons, and the `optimise()` function updates the weights using L-BFGS to simulate fluid behaviour. The resulting diagram is rendered to image files to visualise the simulation.

The figures below depict the examples of the basic implementation of the Voronoi diagram, before the addition of the fluid simulation. The image on the left is from TD 7 and the image on the right from TD 6.

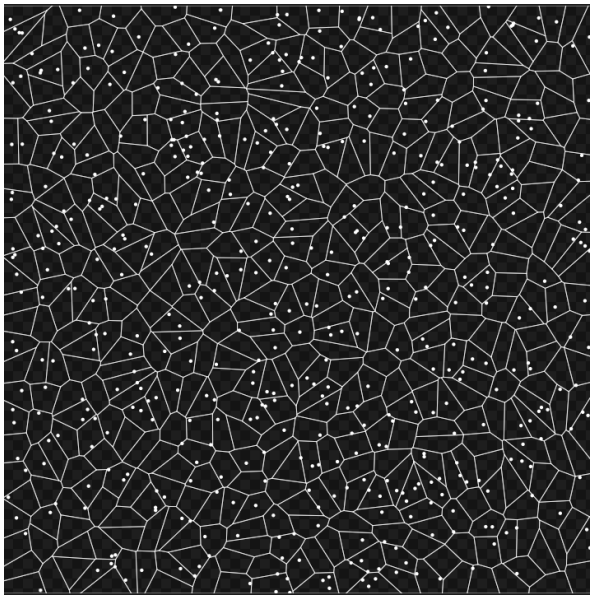

Figure 1: 500 particles, time: 10.67 seconds

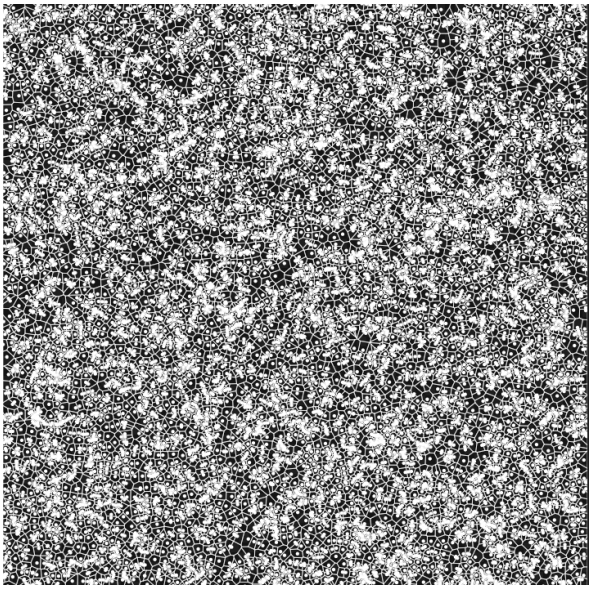

Figure 2: 10,000 particles, time: 97.50 seconds

For these two TDs only I implemented a new variable called `target_areas`. The variable `target_areas` in the code defines the desired area for each Voronoi cell. During optimisation, the algorithm adjusts the weights

of the cells so that their actual areas match these target values as closely as possible. This allows more unique Voronoi diagrams as seen in the lecture notes, e.g. figure 4.15 in the lecture notes.

Fluid Simulation

Fluid simulation models the motion and behaviour of liquids or gases by solving physical equations that govern flow, such as the Navier–Stokes equations. In practice, this involves discretising space into particles or grid cells and updating their positions, velocities, and pressures over time. In the context of semi-discrete optimal transport, fluid is often represented as mass distributed over a domain, and its evolution is simulated by repeatedly solving for optimal transport maps that redistribute mass according to physical forces like pressure or gravity, while preserving incompressibility and volume constraints.

I tested different values of spring stiffness and observed that higher stiffness results in slower visual movement in the simulation. This occurs because stiffer springs cause particles to move less between frames, making the video appear slower. This effect is evident in the two videos in the repository: one with a stiffness of `0.004`, and another with `0.0001`. The latter appears choppier due to larger particle displacements. Based on these tests, I found that a stiffness range between `0.001` and `0.1` offers the best visual quality and simulation stability. Of course decreasing the stiffness also effects the balls as expected, ie. they become squishier if you will.

Below is a table outlining the compilation time:

No. Particles	**Mass**	**Stiffness**	**Frames**	**Time (seconds)**
50	200	0.001	350	39.98
100	200	0.001	350	85.88
150	200	0.001	350	154.12

Table 1: Fluid simulation time for different spring stiffness values

The videos in the repository have the following parameters:

- **Mass**: 200
- **Stiffness**: 0.001
- **Frames**: 350

The name of the file indicates the number of particles in the video. The lengths of the videos vary although the contain the same number of frames since I sped them up at different rates.