CSE306 - Fluid Simulator Project 2

Etienne Leroy

June 16, 2024

1 Introduction

In this report, I outline the implementation of my fluid simulator. This simulator implements all the mandatory features, generating videos in 2D. The project consists of a *main.cpp* file which makes use of 5 other header files defining the necessary classes required by the simulator. We also make use of the *lbfgs* library, consisting of *arithmetic_ainsi.h*, *lbfgs.h*, and *lbfgs.c* for L-BFGS optimization. In addition, we use the files *stb_image_write.h* and *stb_image.h* to save generated images.

The project is built by the *Makefile*, and is compiled by running *$ make* in the project directory, and the generated video by running *$ make vid*. For convenience, we can optionally use a shell script to automate clean, make, video generation, and timing, by running *./build.sh*.

All times mentioned in this report were generated from a 2022 M2 chip.

The code for this project can be found at this GitHub.

2 Credits

All files in the *extra* folder (*arithmetic_ainsi.h*, *lbfgs.h*, *lbfgs.c*, *stb_image_write.h*, *stb_image.h*) are not a product of my work. These were provided for L-BFGS optimization and image saving.

My implementation of *PowerDiagram.h* was inspired by, and completed with help from this GitHub.

Lines 22-88 in *FluidSimulation.h* are taken from https://pastebin.com/jVcNAE5Q, as provided on Moodle.

Lines 91-160 in *Poly.h* are taken from https://pastebin.com/bEYVtqYy, as provided on Moodle.

3 Project Structure

3.1 Code Overview

Our program starts by initializing a system with a defined number of points (or particles), which is processed by *FluidSimulation.h*. This header describes the evolution of an array of particles over time with respect to their individual velocities.

The *OptimalTransport.h* header computes the L-BFGS optimization of the initialized number of points and weights (in our case 50, initialized with uniform weights). The solution is then saved to an SVG file (sol_voronoi.svg), with the option of generating the resulting video. Timing is provided regardless.

3.2 File Structure

- **main.cpp:** Initializes and runs the fluid simulator, and sets up and solves an optimal transport problem. Saves the resulting SVG file and generation time.
- **vec3.h:** Defines a vector class to have vector (vec3) objects and their operations (+, -, dot, cross, etc.).
- **FluidSimulation.h:** Defines the *FluidSimulation* class for initializing, simulating, and saving frames of a fluid simulation involving particles. Uses *OptimalTransport.h* to compute particle movements.
- **Poly.h:** Defines the *Poly* class for polygons with methods to compute the area, centroid, and integral of squared distances from a reference point. It also includes functions for saving static and animated SVG files of these polygons.
- **OptimalTransport.h:** Defines the *OptimalTransport* class which manages and solves optimal transport problems using a power diagram. It uses the *lbfgs* library API to implements the methods to evaluate and progress the optimization process, using the L-BFGS algorithm to compute the optimal transport map.
- **PowerDiagram.h:** Defines the *PowerDiagram* class which generates and manages power diagrams by computing power cells for a set of points with associated weights. See section (2) for credits.
- **extra:** Folder containing the following files: *arithmetic_ainsi.h*, *lbfgs.h*, *lbfgs.c*, *stb_image_write.h*, *stb_image.h*.

4 Fluid Simulation

The following images are taken from the generated videos found on my GitHub. The first video holds 100 particles and took 37250 seconds to generate. Both images had a 40% particle content, with a timestep of 0.002, and an epsilon of 0.004, and a particle mass of 50.

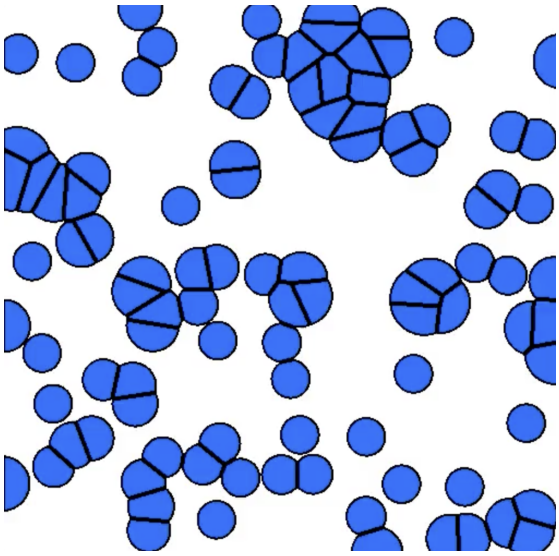

Figure 1: 100 Particles: Frame 1

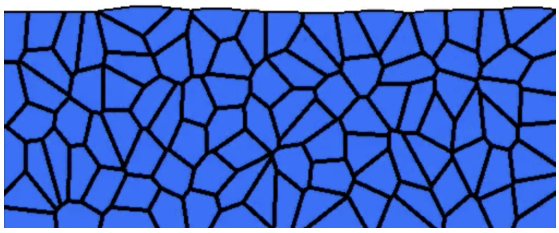

Figure 2: 100 Particles: Frame 18

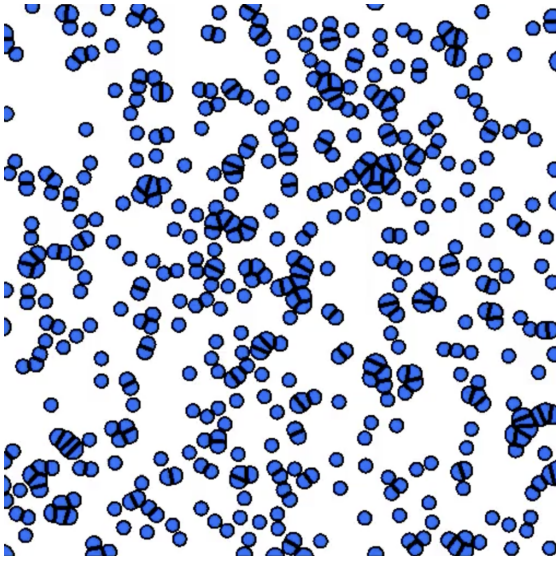

Figure 3: 500 Particles: Frame 1

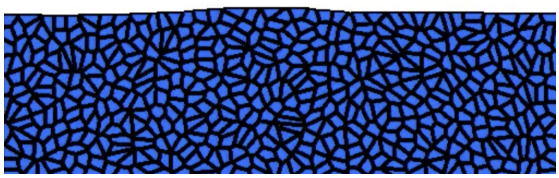

Figure 4: 500 Particles: Frame 18