Octave Gaspard

June 12, 2024

CSE306

Fluid simulator report

1 Introduction

1.1 Using the simulator

Use *make build* to build the *./sim.exe* executable.

The parameters need to be fixed at compile time. They can be found mostly in the main function in the initialization of the point cloud (fluid proportion, number of particles, repartition of the particles) and in the ”animate” function (number of frames, epsilon, dt, mass of the particles).

1.2 Report

The plan this report follows is the same as the order of the TDs, namely we will start by the Voronoi diagram generation, then changing and optimizing the weights with the L-BFGS solver, and finally show the applications of this to make a free-surface liquid simulator.

All render times are on a laptop running a Ryzen 5500U using 16 threads for generating Voronoi diagrams.

2 TD6 Voronoi Diagrams

We start by writing the Sutherland-Hodgman clipping algorithm that will be reused in the last lab, by fixing that the Polygons' vertices are listed in a counter-clockwise order.

I coded the parallelization for the algorithm by splitting the vertices into *N_THREADS* blocks, and running the clipping algorithm for each of them in parallel.

See in Figure 1 the results of the code on a simple example.

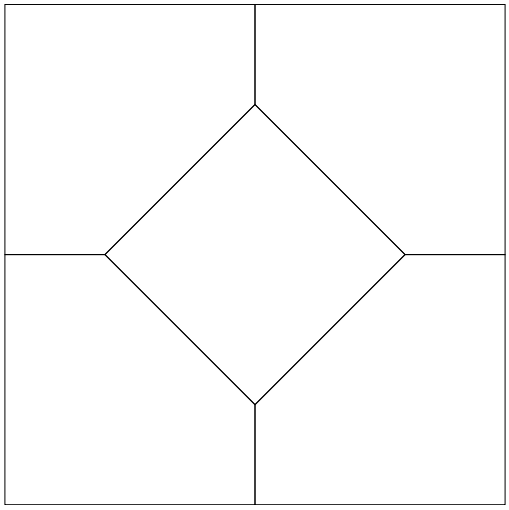

Figure 1: Voronoi diagram of the four vertices of a square and its center

3 TD7 Power Diagrams

We then code Power diagrams, namely the way to alter the size of the Voronoi cells depending on given weights for each cell, as seen in Figure 2

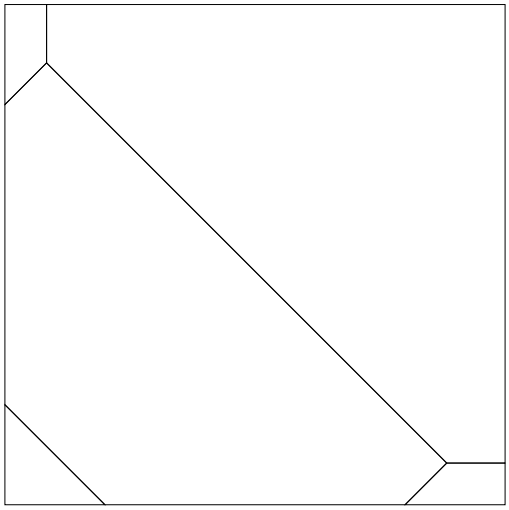

Figure 2: Power diagram of the same vertices as Figure 1 but with different weights.

However, these weights are hard to control and do not represent an area by themselves.

To control the exact area of each cell, we will be using the given lbfgs library, and encoding the objective function and its gradient as defined in the lecture notes. This way, we can use this optimization step on weights to achieve more precision in the wanted Power diagram.

For example, taking again the same example but fixing the area of the lower-left cell to be 50% more than the others, we reach as result Figure 3

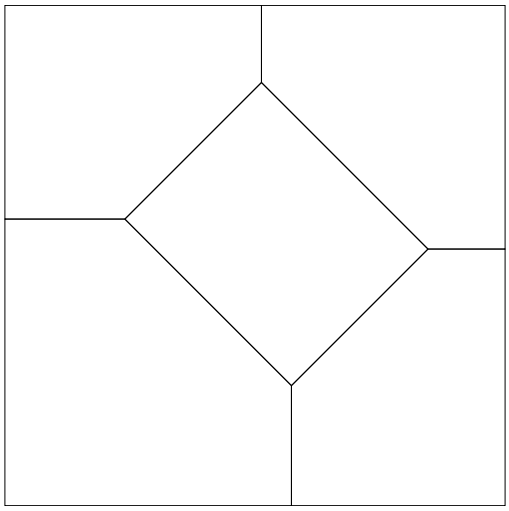

Figure 3: Similar to Figure 2, with optimized weights for an area of 0.3 for the lower left cell and 0.2 for the others

4 TD8 Fluid simulator

In order to simulate free-surface fluids, I followed the lecture notes indication to change the evaluation function for L-BFGS optimization, by adding a weight corresponding to the air volume, and clipping all Voronoi cells by a 30-sided regular polygon as an approximation of a circle, before making the computations.

By additionally adding the spring and gravity force to the cells, and setting the parameters similarly to what is described in the lecture notes ($\epsilon = 0.004$, $dt \simeq 0.002$), I could finally render fluid animations.

5 Additional work and final animations

In addition to parallelizing the Voronoi cell generation, I also parallelized the evaluate function for the L-BFGS optimization (low improvement on performance) and the save_frame function (medium improvement on performance).

I also implemented Lloyd iterations in order to simulate "raining" (see *example_animation_bouncy_rain.gif*). This animation in particular was done with 1000 particles for 100 frames and took 1h32 to render.

Another animation, *example_animation_full_flat.gif*, took 32min to render 100 frames since it was done with only 600 particles.