Computer Graphics Report

Ziyue Qiu (BX25)

1 Lecture 6 & 7: Voronoï, Power Diagrams & Optimal Transport

In Labs 6 and 7 we built the geometric backbone for our fluid solver:

- **Lab 6 (Voronoï & Clipping).** We implemented the naïve $O(N^2)$ Parallel Linear Enumeration algorithm for 2D Voronoï diagrams, relying on the Sutherland–Hodgman polygon-clipping routine. This gave us a robust Voronoï cell generator in `Voronoi.cpp/h` and `Polygon.cpp/h`.
- **Lab 7 (Power Diagrams & L-BFGS).** We extended the Voronoï code to support weighted cells (Power Diagram) and used libLBFGS to optimize the weights so each cell has equal area. The core is in `PowerDiagram.cpp/h` and `OptimalTransport.cpp/h`, with callbacks that compute the semi-discrete OT objective and its gradient.

These two labs established our ability to compute cell decompositions and solve for weights efficiently, which we then leverage in the fluid simulation.

2 Lecture 8: Free-Surface Fluid Simulation

Implementation Overview

Our Lab 8 solver (`Fluid.cpp/h`) runs 50 particles through 400 frames of a free-surface fluid simulation under incompressible Euler:

1. *Initialization:* place $N = 50$ particles uniformly at random, zero their velocities, and create an initial weight vector of all 1's.
2. *Equal-area decomposition:* at each frame, call `solver.compute_fluid()` on the current particle positions and initial weights to solve an L-BFGS problem for $N + 1$ weights (one per particle plus air), enforcing each Laguerre cell's area; then retrieve the power diagram cells.
3. *Output frame:* write the current diagram to a PNG file via `save_frame()`, producing 400 frames at 25 fps.
4. *Forces & integration:* for each particle i:
   - Compute spring force

$$F_s = \frac{c_i - p_i}{\varepsilon} \quad \text{where } \varepsilon = 0.004^2.$$

   - Add gravity $F_g = m\,\mathbf{g}$, with mass $m = 200$ and $\mathbf{g} = (0, -9.81)$.
   - Total force $F = F_s + F_g$.

- Explicit Euler update
$$v \leftarrow v + \frac{F}{m}\Delta t, \quad p \leftarrow p + v\,\Delta t,$$
with $\Delta t = 0.003$.

5. *Boundary handling:* reflect any out-of-bounds particle by mirroring its position and zeroing its velocity component against the wall.

Key constants:
$$m = 200, \quad \varepsilon = 0.004^2, \quad \Delta t = 0.003, \quad \text{frames} = 400, \quad \text{fps} = 25.$$

Sample Frame

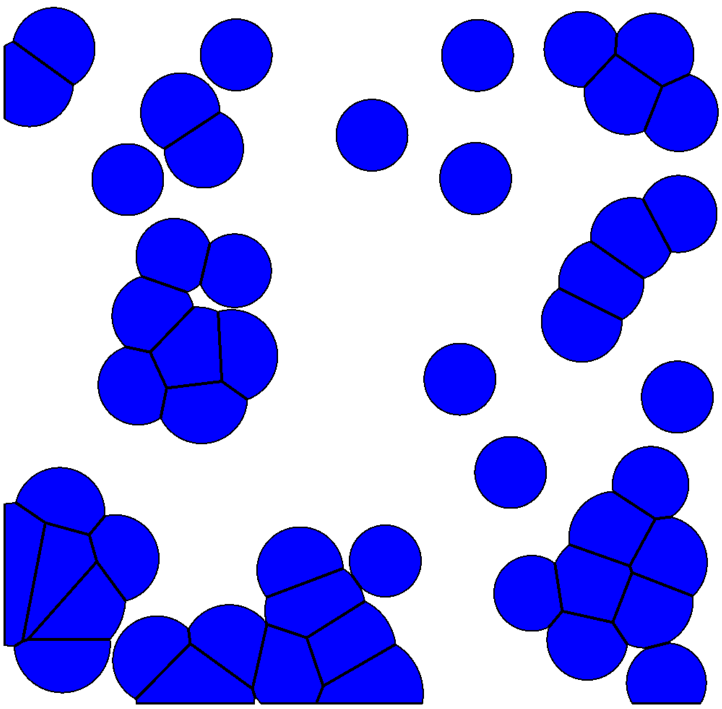

Figure 1: Frame 22 of the simulation: the fluid drop strikes the surface, forming a crown.

Video Access

The full 16s animation (400 frames at 25 fps) can be viewed on GitHub:

View the Fluid Simulation Video on GitHub

Feedback

These labs deepened my understanding of geometric clipping, optimal transport, and free-surface fluid dynamics. Implementing the spring-damping and restitution terms required careful CFL-style checks, and tuning the L-BFGS convergence parameters was critical to avoid flicker. For future offerings, more guidance on parameter ranges and an automated profiler for per-frame timings would be very helpful.