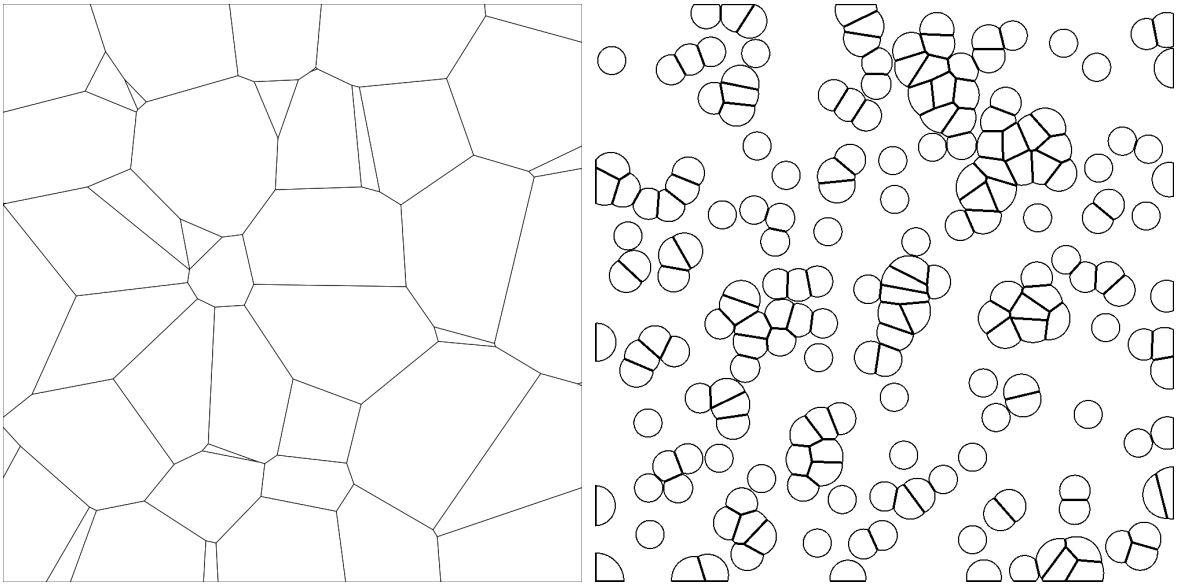
CSE306 Assignment 2
Fluids

Victor Lasocki

June 16, 2024



Contents

1 Introduction

In this project, we explore two concepts in computer graphics: plotting Voronoi diagrams and simulating ball drops under the influence of gravity. They are concepts important in the understanding of geometric structures and of physical simulations, and also have wide-ranging applications in various fields of computer graphics.

Geometry Processing: Clipping and Voronoi Diagrams

A Voronoi diagram is a partitioning of a plane into regions based on the distance to a specified set of points. Each region corresponds to one of the points, and consists of all locations that are closer to that point than to any other point, with the limits begin points equidistant from too specified points. This technique is used in fields like meteorology, geography, and network planning. In our case, we generated a Voronoi diagram by first plotting a set of points and then determining the boundaries that equidistantly separate each point from its neighbors.

Computational Fluid Mechanics: Simulating Ball Drops with Gravity

The second part of the project focuses on simulating fluid mechanics, and in our case we implemented balls dropping under the influence of gravity that act like water. This simulation is a classical problem in physics-based animation and serves as a basic example of how gravity affects objects in motion. The simulation accounts for the initial position, velocity, and the gravitational force acting on each ball, updating their positions over time to reflect realistic physical behavior. In our case of simulating water, we have to add to accumulation of the balls at the bottom, much like water accumulating.

2 Clipping and Voronoï Diagrams

The Voronoi diagram is generated using a set of seed points, each defining a region of the plane. The algorithm implemented follows these steps:

1. **Initialization**: A set of seed points is randomly generated or provided as input.
2. **Distance Calculation**: For each pixel in the plane, the distance to each seed point is calculated.
3. **Region Assignment**: Each pixel is assigned to the region of the closest seed point, forming distinct cells.
4. **Edge Drawing**: The boundaries between regions are determined and drawn to visually represent the Voronoi diagram.

This implementation efficiently partitions the space, creating a visually appealing and mathematically accurate Voronoi diagram. The clipping process ensures that only the parts of the diagram within the viewing window are rendered, optimizing the performance and focusing the viewer's attention on the relevant areas. Below is a simplified pseudo code:

The resulting Voronoi diagram provides a clear and informative partitioning of the space, which can be used in various applications such as spatial analysis and graphical representations. The integration of clipping ensures that the diagram is displayed effectively within the defined region, enhancing both performance and visual appeal.

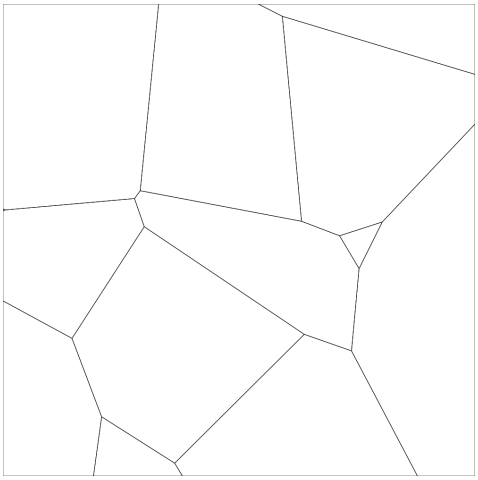


Figure 1: Voronoi pattern generated for $N = 200$

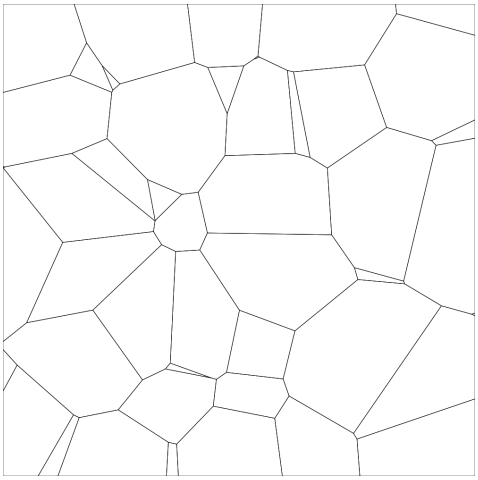


Figure 2: Voronoi pattern generated for $N = 2000$

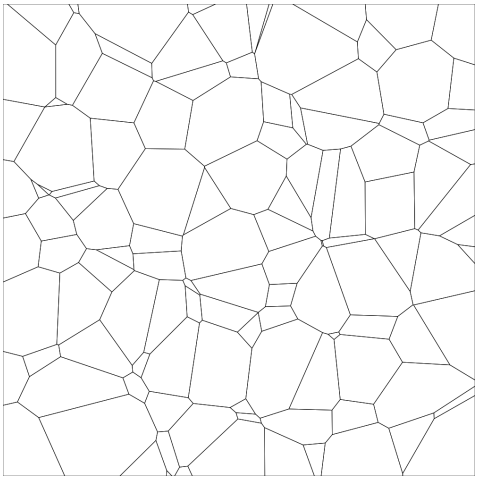


Figure 3: Voronoi pattern generated for $N = 10000$

3 Computational Fluid Mechanics

Our implementation of ball drops with gravity follows a straightforward approach, using basic kinematic equations to update the position and velocity of each ball over time. The key steps in the implementation are as follows:

1. **Initialization**: Define the initial positions and velocities of the balls, and set the gravitational constant.
2. **Position and Velocity Update**: For each time step, update the velocity of the balls by adding the gravitational acceleration, and then update the position by adding the velocity.
3. **Collision Detection**: Detect collisions with the ground or other objects, and update the velocity and position accordingly to simulate realistic interactions.
4. **Rendering**: Render the updated positions of the balls on the screen, creating an animation of the falling balls.

The resulting simulation accurately represents the behavior of balls under the influence of gravity, including realistic collisions with the ground and between balls. By adjusting parameters such as the gravitational constant and the restitution coefficient, we can simulate different physical environments and interactions, demonstrating the versatility and educational value of this simple yet powerful simulation.

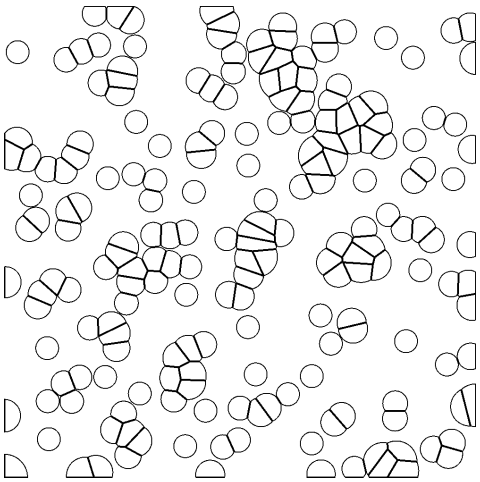


Figure 4: First frame of the animation generated for $N = 200$

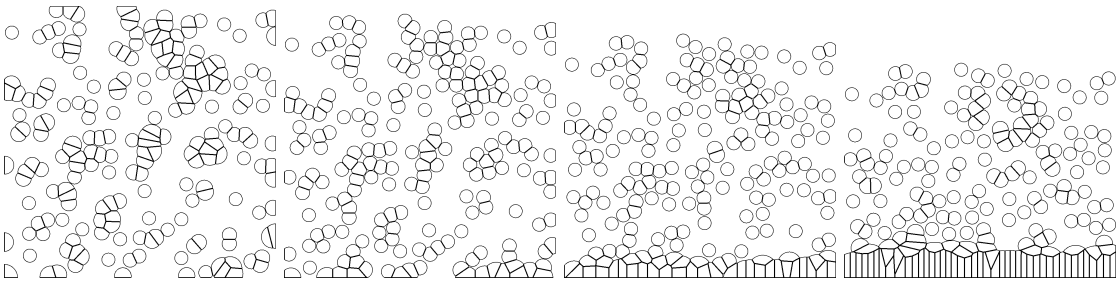


Figure 5: Progression of animation of ball dropping simulation, selecting 1 in 3 generated frames, with $N = 200$ and `time_step(0.01)`

And, with a bit of magic (adding blue color through the `isInside` method), we can make it look like "real" water! Or at least close enough to resemble a fluid moving.

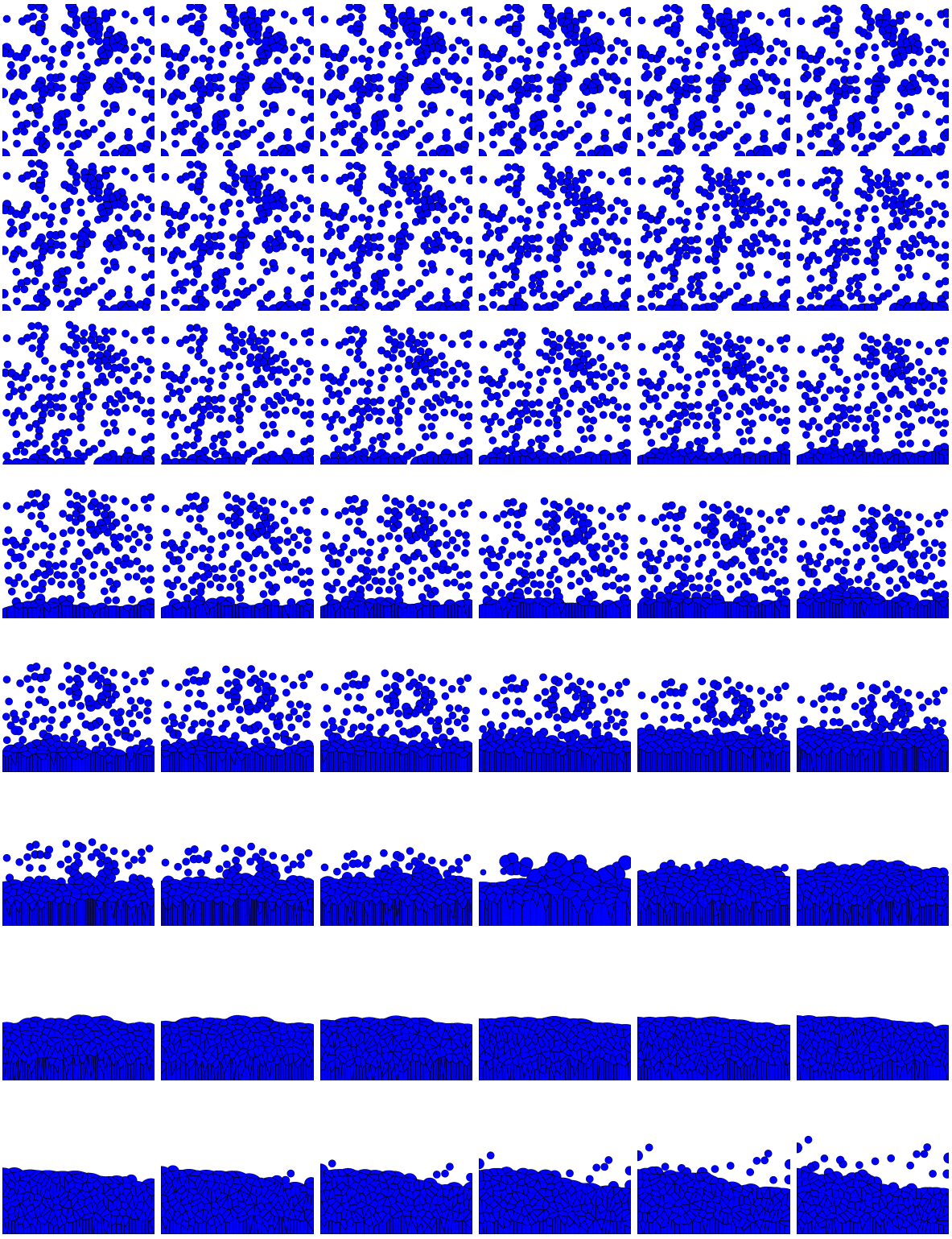


Figure 6: Animated water drops (blue cells) dropping, with $N = 200$ and `time_step(0.01)`