Final Report: fluid simulation based on SPH with grid structures

Introduction

In my final project, I primarily focused on the realm of fluid simulation, specifically utilizing Smoothed Particle Hydrodynamics (SPH) with the Navier-Stokes equations. SPH functions by treating a fluid as a collection of discrete particles, with each particle possessing attributes such as position, velocity, and density. As part of this project, I have implemented SPH code by calculating and updating the density, pressure and three main forces from gravity, pressure, and viscosity. In addition to SPH, to handle huge number of particles in the scene, I added a 2d grid structure for each particle to speed up searching neighbor cells. My code is based on a Github project, rewriting and improving its functionality and using its WebGL to show the result.

Key Features

- ### Using Navier-Stokes equations

In my fluid simulation, I use the Navier-Stokes equations to calculate how fluid particles should move in each time step. For each particle, I calculate the pressure gradient and viscous forces that affect it, which are derived from the Navier-Stokes equations. These equations are derived from the fundamental laws of conservation of momentum.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \eta \nabla^2 \mathbf{u} + \rho \mathbf{g}$$

To apply it in the fluid simulation, the formula can be expanded to

$$\rho \frac{\mathbf{Du}}{\mathbf{D}t} = \rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \eta \nabla \cdot \nabla \mathbf{u} + \rho \mathbf{g}$$

The pressure and density are used to calculate the acceleration of each particle, which then helps to update the velocity and position of the particle.

- ### Calculation of density and pressure

For each particle, I loop through the rest of the particles to accumulate their density contributions. The computation of these contributions relies on Müller's Poly6 kernel function, which shapes the influence a particle has on its neighbors. Once having the total density, I compute the pressure for each particle using the ideal gas law. This law serves as our equation of state, which describes the relationship between pressure and the total density. In essence, the pressure depends on the particle's density, and this relationship helps in simulating the behavior of the fluid.

- ### Calculation of force

In a similar process, for each fluid particle, we iterate through all other particles to compute the cumulative contributions towards pressure and viscosity forces. The constant scalar components of the gradient of the Spiky kernel and the Laplacian of the Viscosity kernel are used to compute the forces. The three main formula to calculate the density and forces are shown here.

$$\rho_i = \rho(\mathbf{r}_i) = \sum_j m_j \frac{\rho_j}{\rho_j} W(|\mathbf{r}_i - \mathbf{r}_j|, h) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h)$$

$$\mathbf{F}_i^{\text{pressure}} = -\nabla p(\mathbf{r}_i) = -\sum_j m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W(|\mathbf{r}_i - \mathbf{r}_j|, h)$$

$$\mathbf{F}_i^{\text{viscosity}} = \eta \nabla^2 \mathbf{u}(\mathbf{r}_i) = \eta \sum_j m_j \frac{|\mathbf{u}_j - \mathbf{u}_i|}{\rho_j} \nabla^2 W(|\mathbf{r}_i - \mathbf{r}_j|, h)$$

- ### 2D Grid structure

In professor Zhao's comment, he said "doing SPH is good. If you want to do fancier things, consider how large number of particles can be efficiently handled." Therefore, my solution is using a 2d grid structure to search the neighbor cells instead of the brute force searching as the GitHub's author did. In initializing the particles, I add the grid initialization, which depends on the position of the particles. Then, in calculation of density and force, I could search the particles nearby. When a large number of particles are used, this way could efficiently handled. Lastly, in idle() function, I clear the grid and refill again.

Result and deliverables

My result code is in sph_fluid.js. I rewrite the calculation, idle function without copying the original code, but by referring to the tutorial https://lucasschuermann.com/writing/particle-based-fluid-simulation. I also replace the brute force searching in the original code by using the grid searching. The WebGL is in index.html. I used a large number of particles (800) to illustrate the effect. I also put the original code (called campare) in the zip file. You can compare the speed of two WebGL to see the difference. The GitHub reference is on novalain/sph-fluid-webgl: Fluid Simulation using Smoothed Particle Hydrodynamics (github.com).