SPH Fluid Simulation with OpenGL in C++

Borworntat Dendumrongkul[*] Dej Wongwirathorn[†]

May 2025

1 Smooth Particle Hydrodynamics

Smooth Particle Hydrodynamic (SPH) is a computational method for simulating fluid flows. This method is based on the Lagrangian approach, where the fluid is represented by a set of particles that carry properties such as mass, velocity, and density. The particles move with the fluid flow, and their properties are updated based on the interactions with neighboring particles.

In SPH, we use the Navier-Stokes equations to describe the motion of the fluid.

$$\rho \mathbf{a} = -\nabla P + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}$$

2 Project Overview

The primary objective of this project is to implement a fluid simulation using the Largrangian method, specifically the SPH method. The simulation will be visualized using OpenGL, allowing for real-time rendering of the fluid dynamics. The project will be implemented in C++ and will utilize the OpenGL library for rendering.

3 Implementation

Our implementation of the SPH fluid simulation can be accessed at Github. The project is structured as follows:

- `src` - Contains the source code for the simulation.
- `include` - Contains the library which is used in the simulation.
- `shaders` - Contains the shaders used for rendering the simulation.

3.1 Fluid Simulation Solver

The fluid simulation solver is responsible for updating the density (ρ), pressure (P), velocity ($\mathbf{v}$), and position ($\mathbf{x}$) of the particles in each time step. The solver uses the SPH equations to compute the interactions between particles and update their properties accordingly.

The main steps of the solver are as follows:

```
For each particle i:
  Compute density using the kernel function
  Compute pressure using the equation of state

For each particle i:
  Compute forces from neighboring particles

For each particle i:
  Compute acceleration using the forces
```

[*]6632109921, 6632109921@student.chula.ac.th
[†]6432055421, 6432055421@student.chula.ac.th

```
    Update velocity using the computed acceleration
    Update position using the updated velocity
```

3.1.1 Smoothing Kernel

The smoothing kernel is a function that defines the influence of a particle on its neighbors. In SPH, the smoothing kernel is used to compute the density, pressure, and forces acting to the particles.

Our Implementation uses the following kernel functions:

- **Wpoly6 kernel**:

$$W_{poly6}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r_i^2)^3 & \text{if } r < h \\ 0 & \text{otherwise} \end{cases}$$

- **Spiky kernel**:

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r_i)^3 & \text{if } 0 \leq r_i \leq h \\ 0 & \text{otherwise} \end{cases}$$

- **Viscosity kernel**:

$$W_{visc}(\mathbf{r_i}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r_i^3}{2h^3} + \frac{r_i^2}{h^2} + \frac{h}{2r_i} - 1 & \text{if } 0 \leq r_i \leq h \\ 0 & \text{otherwise} \end{cases}$$

Where r is the distance between two particles, and h is the smoothing length.
The smoothing length h determines the range of influence of a particle on its neighbors.

3.1.2 Computing Density and Pressure

The density of a particle is computed using the W_{poly6} kernel. By the definition of the density, we have:

$$\rho_i = \sum_j m_j W_{poly6}(\mathbf{r}_i - \mathbf{r}_j, h)$$

where m_j is the mass of particle j, $\mathbf{r}_i$ is the position of particle i, and $\mathbf{r}_j$ is the position of particle j.
In this implementation, we use the ideal gas equation of state to compute the pressure of the fluid:

$$p = k(\rho - \rho_0)$$

where k is the stiffness coefficient, ρ is the density of the particle, and ρ_0 is the rest density of the fluid.

3.1.3 Computing Forces

The are two main forces acting on the particles in the simulation:

- **Pressure force**: The pressure force is computed using the W_{spiky} kernel. The pressure force acting on particle i is given by:

$$\mathbf{f}_i^{pressure} = -\sum_j m_j \cdot \frac{p_i + p_j}{2\rho_j} \nabla W_{spiky}(\mathbf{r}_i - \mathbf{r}_j, h)$$

- **Viscosity force**: The viscosity force is computed using the W_{visc} kernel. The viscosity force acting on particle i is given by:

$$\mathbf{f}_i^{viscosity} = \mu \cdot \sum_j m_j \cdot \frac{v_j - v_i}{\rho_j} \nabla^2 W_{visc}(\mathbf{r}_i - \mathbf{r}_j, h)$$

Where μ is the viscosity coefficient, v_i is the velocity of particle i, and v_j is the velocity of particle j.
Total force acting on particle i is given by:

$$\mathbf{f}_i = \mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscosity}$$

3.1.4 Computing Velocity and Position

The acceleration of a particle is computed by:

$$\mathbf{a}_i = \frac{\mathbf{f}_i}{\rho_i} + \mathbf{g}$$

The velocity and position of the particle are updated using the following equations:

$$\mathbf{v}_i^* = \mathbf{v}_i + \mathbf{a}_i \Delta t$$

$$\mathbf{x}_i^* = \mathbf{x}_i + \mathbf{v}_i^* \Delta t$$

where v_i^* is the updated velocity, x_i^* is the updated position, and Δt is the time step.

3.1.5 Optimization

The SPH simulation can be computationally expensive, especially for large numbers of particles. To optimize the simulation, we can use the Spartial hashing technique to efficiently find neighboring particles.

Spartial hashing is a technique that divides the simulation space into a grid of cells, where each cell contains list of particles that are located in that cell. When computing the neighboring particles for a give particle, we only need to check the particles in the same cell and the neighboring cells. This reduces the number of particle interactions that need to be computed, resulting in a significant performance improvement.

4 Rendering

We render the simulation using OpenGL, which allows for real-time rendering of the fluid dynamics. In each frame, we assume that the time step is constant, and we update the simulation using the SPH equations.

Our implementation uses double buffering to avoid flickering and tearing in the rendering. In our implementation, the particles are rendered as spheres which created by the shader program. The shader program is responsible for computing the color and lighting of the particles based on their properties.

5 Limitations

Due to the stability of the simulation, the parameters of the simulation must be carefully tuned to achieve stable and realistic results. Currently, our implementation does not support the simulation of complex fluid interactions, such as surface tension and turbulence. These interactions can be added in future work to improve the realism of the simulation. Additionally, the simulation is limited to incompressible fluids, and the equation of state used in the simulation is a simple ideal gas equation.

References

Bindel, Jonathan R. (2011). *Deriving SPH.* Lecture notes for *Applications of Parallel Computers (CS 5220)*, Cornell University, Fall 2011. Accessed: 10 May 2025. URL: `https://www.cs.cornell.edu/~bindel/class/cs5220-f11/code/sph-derive.pdf`.

Bridson, R. (2015). *Fluid Simulation for Computer Graphics, Second Edition.* Taylor & Francis. ISBN: 9781482232837. URL: `https://books.google.co.th/books?id=7MySoAEACAAJ`.

Koschier, Dan et al. (2019). "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids". In: *Eurographics 2019 - Tutorials.* DOI: `10.2312/EGT.20191035`. URL: `https://diglib.eg.org/handle/10.2312/egt20191035`.

Müller, Matthias (n.d.). *Blazing Fast Neighbor Search with Spatial Hashing.* Ten Minute Physics Tutorial 11, `https://matthias-research.github.io/pages/tenMinutePhysics/11-hashing.pdf`. Accessed: 1 May 2025.

Müller, Matthias, David Charypar, and Markus Gross (2003). "Particle-Based Fluid Simulation for Interactive Applications". In: *Symposium on Computer Animation.* Ed. by D. Breen and M. Lin. The Eurographics Association. ISBN: 1-58113-659-5. DOI: `/10.2312/SCA03/154-159`.

Teschner, Matthias et al. (Dec. 2003). "Optimized Spatial Hashing for Collision Detection of Deformable Objects". In: *VMV'03: Proceedings of the Vision, Modeling, Visualization* 3.

Appendix

Our source code is available at Github and the video demo is available at Youtube.