Smoothed Particle Hydrodynamics for Fluid Simulation

Abheek Ghosh and Mario Sergio Fuentes Juarez

Abstract

For this final project, we explored and implemented a physical simulation of fluids. We restricted ourselves to in-compressible fluids. We used the Lagrangian approach for the simulation, in particular, we chose to implement the Smoothed Particle Hydrodynamics (SPH) technique adapted to fluids by Müller et al. [2]. The SPH technique was initially proposed as a method for astrophysical simulations.

We implemented 1) pressure, 2) viscosity, and 3) surface tension forces, and 4) some basic UI controls (e.g. mouse dragging) for interacting with the fluid. Other external forces included in the simulation are gravity and bouncing forces from wall collisions.

Description

Smoothed Particle Hydrodynamics (SPH)

In SPH, quantities are interpolated in a continuous space using a finite number of particles. Using radially symmetric smoothing kernels, SPH distributes quantities in a local neighborhood of each particle. A scalar quantity A, interpolated at a location r as:

$$A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h)$$

where j iterates over all the particles. m_j, ρ_j, $\mathbf{r}_j$, A_j are the mass, density, position, field quantity value of particle j, respectively. W is the smoothing kernel and h is the smoothing radius. We only use even, normalized smoothing kernels with finite support. (We will discuss more about W later.)

In SPH, the derivative of field quantities only affect the kernel. It is also necessary to define the gradient of A:

$$\nabla A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h)$$

and Laplacian of A:

$$\nabla^2 A(\mathbf{r}) = \Delta A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \Delta W(\mathbf{r} - \mathbf{r}_j, h)$$

Modelling Fluids using SPH

In Eulerian (grid-based) dynamics that we studied in class, the conservation of mass is given by the equation:

$$\frac{\delta \rho}{\delta t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

while the Hamiltonian Principle gives us the Navier-Stokes equation:

$$\rho(\frac{\delta \mathbf{v}}{\delta t} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}^{ext}$$

where μ is the viscosity coefficient, and $\mathbf{f}^{ext}$ external force.

Using a particle-based representation simplifies this equation. First, we don't have to worry about conservation of mass, because the number of particles and the mass of each particle is fixed. Second, the convective term $\mathbf{v} \cdot \nabla \mathbf{v}$ is not required and the LHS of Navier-Stokes equation can be replaced by ρ times the time derivative of velocity for each particle.

For the RHS, we derive the forces as follows:

Pressure

The pressure force, $\mathbf{f}_{pressure}$, derived from $-\nabla p$ is not symmetric by default. We modify the force to make it symmetric. The formula used is:

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

Viscosity

Similar to the pressure force, the viscosity force derived from $\mu \Delta \mathbf{v}$ is not symmetric. We modify the formula to get:

$$\mathbf{f}_i^{viscosity} = \mu \Delta \mathbf{v}(\mathbf{r}_i) = \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \Delta W(\mathbf{r}_i - \mathbf{r}_j, h)$$

Surface Tension

The surface tension is modelled based on the ideas of Morris [1]. We define a field, called the smoothed color field, that captures whether a particle is surrounded by other particles or not. It is 1 where a particle is present, given by formula:

$$\nabla c(\mathbf{r}) = \sum_j m_j \frac{1}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h)$$

The surface tension force is given by:

$$\mathbf{f}_i^{surface} = -\sigma \Delta c(\mathbf{r}_i) \frac{\nabla c(\mathbf{r}_i)}{\|\nabla c(\mathbf{r}_i)\|}$$

where σ is the coefficient for surface tension.

Smoothing Kernels

Finally, the stability and overall performance of SPH greatly depends on the choice of smoothing kernel. We decided to implement the polynomial kernels described in [2]. We use this kernel for all purposes except for computing $f^{pressure}$ and $f^{viscous}$:

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

For computing the force due to pressure, $f^{pressure}$, we don't use the above kernel because it can cause clustering of particles. If the particles come too close then the gradient vanishes. Therefore, we use this kernel instead:

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

Similarly, for computing the viscous force, $f^{viscous}$, we use a different kernel given below. With the above mentioned kernels, if the particles come too close then they may start gaining velocity rather than losing it.

$$W_{viscosity}(\mathbf{r},h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{\|\mathbf{r}\|^3}{2h^3} + \frac{\|\mathbf{r}\|^2}{h^2} + \frac{h}{2\|\mathbf{r}\|} - 1 & 0 \le \|\mathbf{r}\| \le h \\ 0 & otherwise \end{cases}$$

Implementation

The simulation consists of a 3D scene that is set up using Libigl library in C++. We represented the fluid as a particle system consisting of 1000 particles by default. These particles are located inside of a box-shaped "tank" that provides containment. We defined the position and velocity of each particle in the fluid as the degrees of freedom. We perform Velocity Verlet for time integration using the aforementioned forces.

Besides the fluid-related forces (viscosity, pressure, and surface tension) our simulation also handles three other external forces: (1) gravity, (2) user-controlled force by dragging the right button of the mouse, and (3) bouncing forces from the walls of the tank.

Our simulation supports the following parameters, which can be modified from the GUI:

- A combo box (`numParticles`) controls the number of particles in the simulation.
- The real "Particle Mass" parameter (`particleMass`) is the mass to be used for each particle.
- The real "Time Step" parameter (`timeStep`) is the time step of the numerical integrator.
- A checkbox (`pressureEnabled`) controls whether or not pressure force is enabled in the simulation.
- A checkbox (`viscosityEnabled`) controls whether or not viscosity force is enabled in the simulation.
- A checkbox (`surfaceTensionEnabled`) controls whether or not surface tension is enabled in the simulation.
- A checkbox (`gravityEnabled`) controls whether or not gravity is enabled in the simulation.
- The real "Gravity G" parameter (`gravityG`) is the gravitational constant used to compute the downward acceleration of particles in the simulation.
- The real "Smoothing Length" parameter (`smoothingLength`) is the radial distance from which a particle is affected by neighboring particles.
- The real "Rest Density" parameter (`restDensity`) is the fluid's initial density.

- The real "Viscosity Coefficient" parameter (`viscosityCoefficient`) is a constant that determines the fluid's intrinsic viscosity (the higher, the more viscous).
- The real "Tension Coefficient" parameter (`tensionCoefficient`) is a constant that determines the fluid's intrinsic surface tension (the higher, the more resistant).
- The real "Epsilon Color Normal" parameter (`epsColorNormal`) is the threshold value used to determine if a particle is a surface particle.

In the formulas for calculating forces (like $\mathbf{f}^{pressure}$, etc.) at the position of a particle, we iterate over all the particles in the space. This leads to a quadratic complexity on the number of particles. Nonetheless, the kernel has non-zero value only near a given particle. To improve the efficiency, we use a grid based approach to efficiently find the nearby particles in the space. We divide the space into cubical grids of side h. All the particles within a distance of at most h from a given particle should be in the same grid or in the neighboring grids. This improves the performance of the simulation.

The bulk of the implementation is defined in `FluidsHook.cpp`, which contains the numerical integration and force/acceleration calculation code. Full implementation can be found in our Github repository `https://github.com/msf1013/sph-fluids`.

Limitations

We also report some limitations/future work to be noted for improving the simulation:

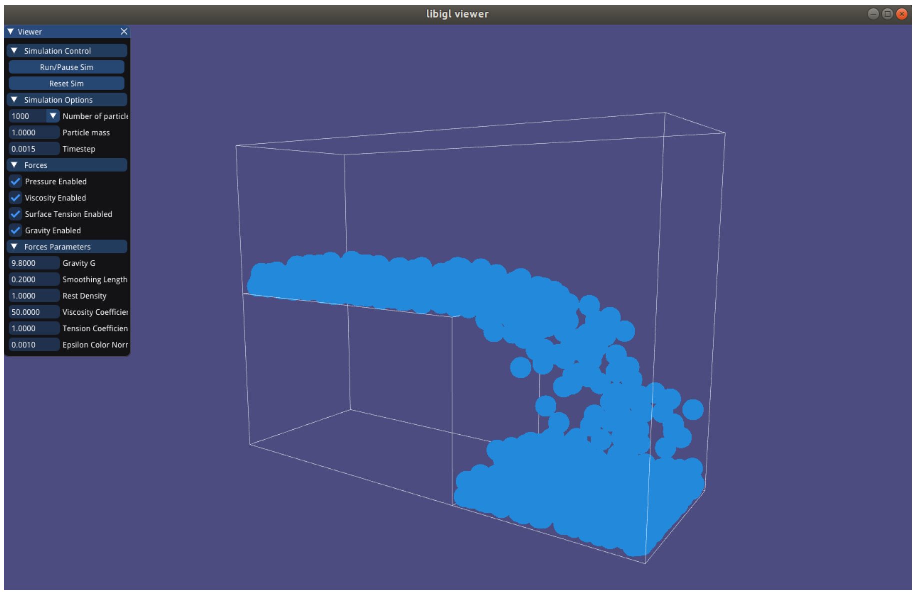
- We were unable to see any outstanding effect from the surface tension force on the particle system. We believe that this might have to do with the fact that we didn't implement a scenario that actually tests this feature. For example, dropping a solid ball on the fluid might exhibit the surface tension effect. Unfortunately we didn't have enough time to implement a more robust collision with generic rigid bodies.
- The particle rendering is not too appealing as we chose to fallback to Libigl/OpenGL's default point-rendering API. We initially tried displaying the particles as sphere meshes, but rendering was slow. We switched to the point-rendering API, but found that Libigl's wrapper API doesn't provide an option to specify any shaders for a rendered point. We chose to stick to the particle-rendering API anyway as we didn't want to compromise runtime performance.
- Due to time constraints, our scene setup is hard-coded, meaning that adding different arrangements of boxes would require some effort. We believe that a more production-ready simulation would require better scene handling.

References

[1] Joseph P Morris. Simulating surface tension with smoothed particle hydrodynamics. *International journal for numerical methods in fluids*, 33(3):333–353, 2000.

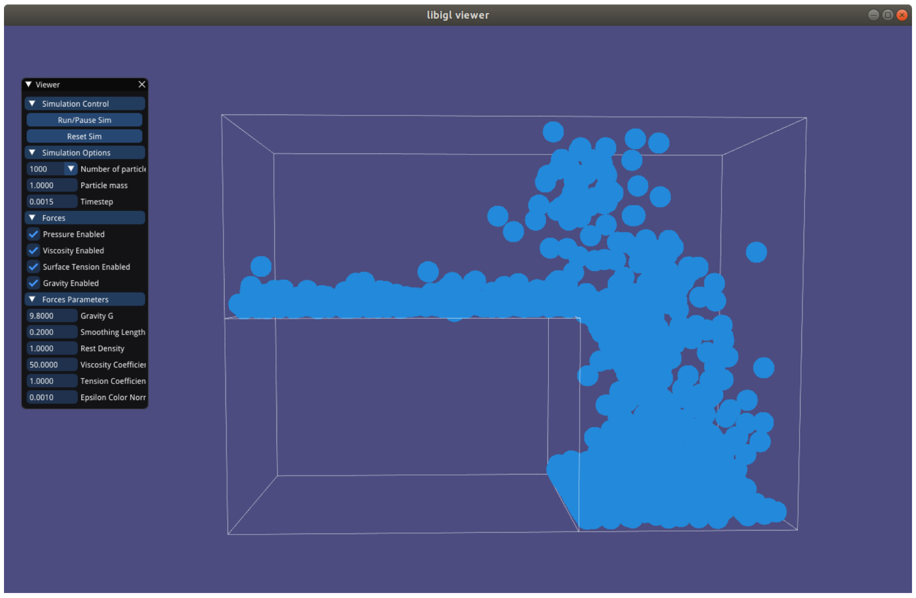
[2] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. *SCA '03 Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, 2003.

Appendix



1000 Number of particle
1.0000 Particle mass
0.0015 Timestep
✓ Pressure Enabled
✓ Viscosity Enabled
✓ Surface Tension Enabled
✓ Gravity Enabled
9.8000 Gravity G
0.2000 Smoothing Length
1.0000 Rest Density
50.0000 Viscosity Coefficier
1.0000 Tension Coefficien
0.0010 Epsilon Color Norr

Figure 1: Running simulation



1000 Number of particle
1.0000 Particle mass
0.0015 Timestep

✓ Pressure Enabled
✓ Viscosity Enabled
✓ Surface Tension Enabled
✓ Gravity Enabled

9.8000 Gravity G
0.2000 Smoothing Length
1.0000 Rest Density
50.0000 Viscosity Coefficier
1.0000 Tension Coefficien
0.0010 Epsilon Color Norr

Figure 2: Mouse-dragging force getting applied to fluid