Implementation of weakly compressible SPH

Huiting Chen
ht.chen@mail.utoronto.ca
University of Toronto

Figure 1: Water block breaks and hits wall

ABSTRACT

Simulating fluid has been an interesting research area in computer graphics and it can generate visually satisfying results. In my project, I implemented weakly compressible smoothed particle hydrodynamics (SPH) to simulate fluid based on paper by Markus Becker et al[1] and calculated isosurface to reconstruct surface based on paper by Jihun Yu et al [4].

KEYWORDS

Smoothed Particle Hydrodynamics, fluid simulation, surface tension

1 INTRODUCTION

Simulating fluid is an interesting research topic in computer graphics and many algorithms have been proposed for fluid simulation. Most of the algorithms can be roughly divided into 2 approaches: the Lagrangian viewpoint and the Eulerian viewpoint. The Lagrangian keeps track of physical quantities in each particles and the fluid is the particle system. The Eulerian viewpoint keeps track of physical quantities at certain locations in the space. [2] Smoothed particle hydrodynamics (SPH) simulates fluid with particles. In my implementation, I will use the Lagrangian particle system and implement weakly compressible smoothed particle hydrodynamics (WCSPH). I have also attempted to implement surface reconstruction using anisotropic kernel based on paper by J. Yu et al [4], but I do not have enough time for debugging and I end up using isotropic kernel for surface reconstruction.

2 MODEL OVERVIEW

2.1 weakly compressible SPH model

The formulas in this subsection come from paper by Markus Becker et al[1].

As noted in the introduction, the smoothed particle hydrodynamics can be used to simulate fluid with a particle system. In SPH, a continuum function $A(x)$ can be interpolated as:

$$A(x) = \sum_j m_j \frac{A_j}{\rho_j} W(x - x_j, h)$$

with subscript j denoting the particles within 2 times the smoothing length h. Therefore, we use the A to model physical quantities velocity, density, pressure etc. over the entire simulation domain. We derive the derivatives of physical quantities and use forward Euler method to update the physical quantities. W is a cubic kernel function that assigns higher weight to neighboring particles closer to x.

In smoothed particle hydrodynamics the Euler equations are given as

$$\frac{d\rho}{dt} = -\rho div(v)$$

$$\frac{d\boldsymbol{v}}{dt} = -\frac{1}{\rho}\nabla P + g$$

The momentum term is given as

$$\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2}\right) \nabla_a W_{ab} + \mathbf{g}$$

with

$$P = B\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right)$$

The above equation is known as Tai equation in the original paper [1].

∇W_{ab} can be derived using the chain rule:

Let $\boldsymbol{dx} = x_a - x_b$, $r = ||\boldsymbol{dx}|| = \sqrt{\boldsymbol{dx}.x^2 + \boldsymbol{dx}.y^2}$. $\boldsymbol{dx}.x$ represents the x coordinate of $\boldsymbol{dx}$ and $\boldsymbol{dx}.y$ represents the y coordinate of $\boldsymbol{dx}$.

$$\frac{dW}{\boldsymbol{dx}} = \frac{dW}{dr} * \frac{dr}{\boldsymbol{dx}}$$

$$\frac{dr}{dx.x} = \frac{1}{2\sqrt{dx.x^2 + dx.y^2}} * 2 * dx.x$$
$$= \frac{dx.x}{\sqrt{dx.x^2 + dx.y^2}}$$

$$\frac{dr}{dx.y} = \frac{dx.y}{\sqrt{dx.x^2 + dx.y^2}}$$

So

$$\frac{dr}{dx} = \frac{dx}{\sqrt{dx.x^2 + dx.y^2}} = \frac{dx}{\sqrt{dx.x^2 + dx.y^2}} = \frac{dx}{||dx||}$$

Artificial viscosity is added to the model to increase numerical stability and it is given as:

$$\frac{d\mathbf{v}_a}{dt} = \begin{cases} -\sum_b m_b \Pi_{ab} \nabla_a W_{ab} & \mathbf{v}_{ab}^T \mathbf{x}_{ab} < 0 \\ 0 & \mathbf{v}_{ab}^T \mathbf{x}_{ab} \geq 0 \end{cases}$$

$$\Pi_{ab} = -\nu \left(\frac{\mathbf{v}_{ab}^T \mathbf{x}_{ab}}{|\mathbf{x}_{ab}|^2 + \varepsilon h^2} \right)$$

ϵh^2 is used to prevent division by zero in case $|X_{ab}|$ is 0.

Surface tension is also added to the model and it is given as:

$$\frac{d\mathbf{v}_a}{dt} = -\frac{\kappa}{m_a} \sum_b m_b W(\mathbf{x}_a - \mathbf{x}_b) \ (\mathbf{x}_a - \mathbf{x}_b)$$

for a particle system.

2.2 surface construction

For surface reconstruction, I tried to implement anisotropic kernel but have not finished debugging it due to time constraint, and I end up using isotropic kernels, and the formula for constructing the isotropic kernels is [4]:

$$\phi(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h_j), \quad \text{with}$$

$$W(\mathbf{r}, h) = \frac{\sigma}{h^d} P\left(\frac{||\mathbf{r}||}{h} \right)$$

3 IMPLEMENTATION DETAILS

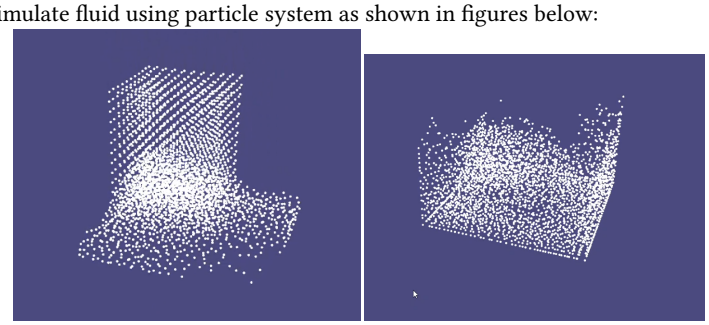
I first implemented a weakly compressible SPH in 2 dimension using Taichi, and implemented the same algorithm in 3 dimension using C++.

To calculate $\frac{dv}{dt}$ and $\frac{d\rho}{dt}$, I need to find the neighbors of a particle (due to $\sum_b$ in the formulas) efficiently. If I loop over all other particles b to search for the neighbors of a certain particle a and calculate the distance between a and b, the time complexity would be as high as $O(n^3)$ for only a single frame of simulation, with n being the number of particles in the system, and n can be nearly 30000 for a normal scene in my simulation. This naive implementation is too slow to simulate fluid. Instead, I use the strategy used by erizmr [3] (original code written in Taichi) and iterate over all particles and store them in `cells` based on their location in each time frame. When searching for neighbors, I only need to loop over the adjacent cells of a certain particle a and check whether the distance between a and other particles b in adjacent cells is below 2 times the smoothing length. Since the number of particles in each cell is no more than 200, this implementation makes the algorithm more efficiently.

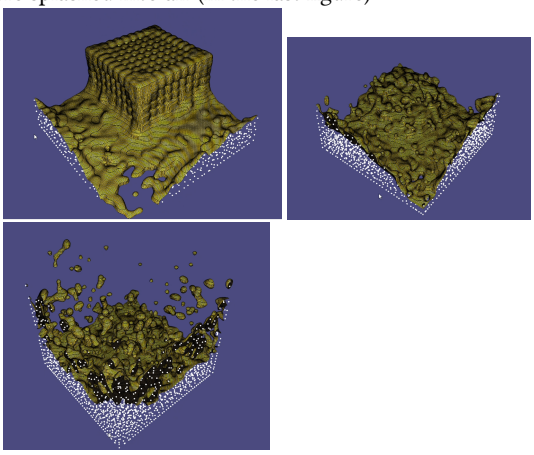
To further speed up the program, I use `parallel_for` in tbb to make the program multi-threaded, and the calculation for particles and cells can be run in parallel.

4 RESULTS

After implementing the weakly compressible SPH, I am able to simulate fluid using particle system as shown in figures below:



The reconstructed surface based on isotropic kernel looks fine in the first 2 pictures, but it looks a bit weird when the particles were splashed into air (in the last figure):



ACKNOWLEDGMENTS

Taichi was used for the initial 2 dimensional implementation.

Tbb is used to parallelize the code, mainly the for loops.

Libigl is used for displaying particles.

Eigen is used for vector and matrix calculation.

REFERENCES

[1] Markus Becker and Matthias Teschner. 2007. Weakly Compressible SPH for Free Surface Flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 209–217.

[2] Robert Bridson. 2016. *Fluid Simulation for Computer Graphics (2nd edition)*. CRC Press.

[3] erizmr. [n.d.]. *Knuth: Computers and Typesetting*. https://github.com/erizmr/SPH_Taichi

[4] Jihun Yu and Greg Turk. 2010. Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels. *ACM Transactions on Graphics* 32, 217–225. https://doi.org/10.2312/SCA/SCA10/217-225