Advanced Computer Graphics Final Written Report

XUANYI XIE, Yao Class 33

ACM Reference Format:
Xuanyi Xie. 2025. Advanced Computer Graphics Final Written Report. *ACM Trans. Graph.* 1, 1 (January 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Our project topic is simulation for rigid bodies and fluids with SPH methods. Our code is available at **https://github.com/Yao-class-graphics-studio/project-songbo-hu-and-xuanyi-xie**. We have implemented a full pipeline containing importing objects, running simulation, and rendering. We have implemented solid-solid coupling, solid-fluid coupling and fluid-fluid coupling based on WCSPH and DFSPH methods, and have used algorithm acceleration and GPU acceleration methods to get a nearly in time simulation. The visual result of our simulation is very realistic.

2 Technical points and results

Beside all basic requirements, we have implemented nearly all the technical points we can include in this project. In the following we use alphabets in brackets to show the main leader in one technical point. X stands for Xuanyi Xie (me) and H stands for Songbo Hu (my teammate).

2.1 Coupling:

Fluid-solid (H), solid-solid (H) and fluid-fluid (X).

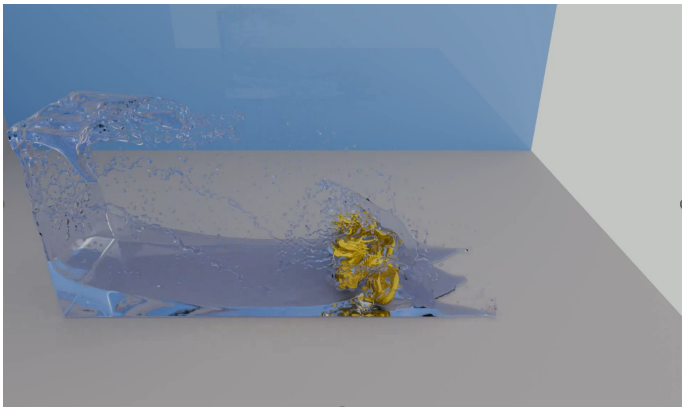

Fig. 1. Large-scale fluid-solid coupling

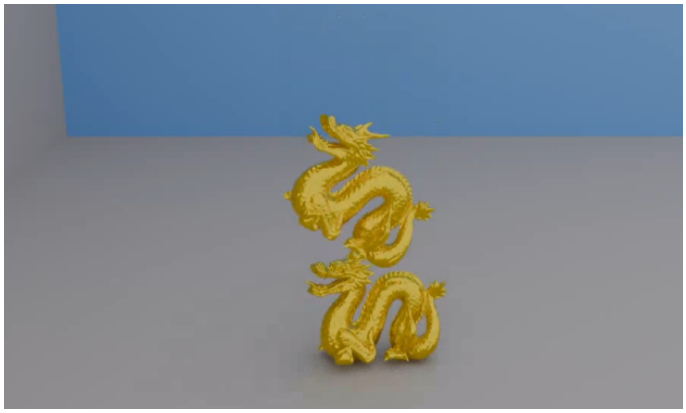

Fig. 2. Solid-solid coupling between two complex objects

Fig. 3. Large-scale fluid-fluid coupling, different density, based on DFSPH

2.2 Geometry:

Complex mesh based geometry (H).

Author's Contact Information: Xuanyi Xie, Yao Class 33, xie-xy23@mails.tsinghua.edu.cn.

ACM 1557-7368/2025/1-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Fig. 4. Large-scale fluid-fluid coupling with shape of alphabets "ACG"

Fig. 5

2.3 Acceleration:

Algorithm acceleration (X), GPU and multi-thread acceleration (H).

2.4 Control:

Customized scene configuration (H) and interactive scene.(H)

Fig. 6. Initial scene

Fig. 7. Beautiful ripples created when the fan rotates at a constant speed

2.5 Rendering:

Use Blender to generate a nice video.(X)

3 Methods

I will introduce the methods I have used for the technical points I have implemented in this section. I am not very familiar with my teammate's work so I will not discuss about his work here.

3.1 Smoothed Particle Hydrodynamics (SPH)

Smoothed Particle Hydrodynamics (SPH) is a widely-used computational method to simulate the behavior of fluids. It is a mesh-free Lagrangian method, meaning that it represents a medium (such as fluid or gas) as a collection of discrete particles that move with the flow, rather than relying on fixed grids or meshes. Each particle carries properties such as mass, position, velocity and density.

Particles interact with each other using smoothing kernels. A smoothing kernel is a mathematical function used to estimate and interpolate physical quantities (e.g., density, pressure) at any point in the simulation domain. A smoothing kernel defines the influence range of a particle in its surrounding area. A common kernel shape would have maximum influence at a particle's position and taper to zero at a certain distance.

SPH solves the Navier-Stokes equations using a particle-based approach. It derives quantities like pressure and velocity using interpolation and interaction between particles.

SPH has the following advantages: Firstly, unlike grid-based methods, SPH does not require a fixed spatial grid, making it highly adaptable to complex geometries, free-surface flows and large deformations. Secondly, different variants of SPH can simulate various physical systems, including incompressible and compressible fluids, high-viscosity fluids, multi-phase flows, and even elastic solids.

I have implemented two variants of SPH for fluid-fluid coupling: WCSPH and DFSPH. I have also implemented a density correction method for interaction boundary in WCSPH for fluid-fluid coupling of different densities. I have referred to [2] for basic WCSPH and

density error correction implementation and [1] for DFSPH implementation. Here are some important pseudocode and formulas taken out from the two papers:

Algorithm 1 basic WCSPH

for all particle i **do**
  find neighbors j
end for
for all particle i **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i using ρ_i
end for
for all particle i **do**
  $\mathbf{F}_{i,\text{pressure}} = -\frac{m_i}{\rho_i}\nabla p_i$
  $\mathbf{F}_{i,\text{viscosity}} = m_i \nu \nabla^2 \mathbf{v}_i$
  $\mathbf{F}_{i,\text{other}} = m_i g$
  $\mathbf{F}_i(t) = \mathbf{F}_{i,\text{pressure}} + \mathbf{F}_{i,\text{viscosity}} + \mathbf{F}_{i,\text{other}}$
end for
for all particle i **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$
end for

For density correction, do the following in pressure calculation:

$$\delta_i = \sum_j W_{ij}$$

$$\tilde{p}_i = m_i \delta_i$$

$$\mathbf{F}_i^{\text{pressure}} = -\sum_j \left(\frac{\tilde{p}_j}{\delta_j^2} + \frac{\tilde{p}_i}{\delta_i^2} \right) \nabla W_{ij}$$

DFSPH improves incompressibility by directly solving both constant density and velocity divergence free constraints, leading to more stable and realistic fluid simulations.

Algorithm 2 Divergence-free solver

function CorrectDivergenceError($\alpha, \mathbf{v}^*$)
  while $\left(\left(\frac{D\rho}{Dt} \right)_{\text{avg}} > \eta^v \right) \vee (\text{iter} < 1)$ **do**
    for all particles i **do** ▷ compute $\frac{D\rho}{Dt}$
      $\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \mathbf{v}_i^*$
    end for
    for all particles i **do** ▷ adapt velocities
      $\kappa_i^v = \frac{1}{\Delta t} \frac{D\rho_i}{Dt} \alpha_i, \quad \kappa_j^v = \frac{1}{\Delta t} \frac{D\rho_j}{Dt} \alpha_j$
      $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{\kappa_i^v}{\rho_i} + \frac{\kappa_j^v}{\rho_j} \right) \nabla W_{ij}$
    end for
  end while
end function

Algorithm 3 Constant density solver

function CorrectDensityError($\alpha, \mathbf{v}^*$)
  while $(\rho_{\text{avg}} - \rho_0 > \eta) \vee (\text{iter} < 2)$ **do**
    for all particles i **do** ▷ predict density
      compute ρ_i^*
    end for
    for all particles i **do** ▷ adapt velocities
      $\kappa_i = \frac{\rho_i^* - \rho_0}{\Delta t^2} \alpha_i, \quad \kappa_j = \frac{\rho_j^* - \rho_0}{\Delta t^2} \alpha_j$
      $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j} \right) \nabla W_{ij}$
    end for
  end while
end function

3.2 Uniform grid and hashing

In SPH algorithms, in every timestep we need to compute the neighboring particles for all particles, which is one of the main bottlenecks in simulation speed. The uniform grid method works by dividing the simulation space into regular, fixed-size cells. Each particle is assigned to a specific cell based on its position, and when searching for neighbors, only the particle's cell and its adjacent cells are checked. This dramatically reduces the number of distance calculations required, making neighbor search operations significantly faster. We can make grid search even faster by mapping grids to hash lists.[2] In our implementation this algorithm can give a 10 times acceleration when there are 10000 particles.

3.3 Rendering

I have used Blender GUI to create a background environment for rendering, including light sources, cameras, materials and textures. Because we use particles in SPH method, I convert them into meshes using Splashsurf before importing frames into Blender.

3.4 External packages

Taichi
Taichi is a high-performance, parallel programming framework designed for scientific computing, especially in the context of GPU computing. It allows us to efficiently leverage the power of GPUs for computations in simulation tasks, offering fast and flexible performance for physical simulations and rendering.
Blender
Blender is an open-source 3D creation suite that supports the entire pipeline of 3D production. We have used Blender for rendering, texturing, our 3D scenes. It provides advanced features such as ray tracing, shading, which are essential for creating high-quality visuals and simulations. We haven't used it for any simulation purpose.
Splashsurf
Splashsurf is an open source tool for surface reconstruction of the .obj files. Since the output of our simulation is the color and position of the rigid and fluid particles, we need to use this tool to reconstruct the surface for future rendering.

4 Discussion

Through this project I get a better understanding on computer graphics, especially simulation. I have also gained much practical skills in programming and in the use of Blender. I think the simulation part can be more complete by implementing fluid-solid interaction in DFSPH.

5 Personal Contribution Statement

My personal contributions are mainly on fluid-fluid coupling, algorithm acceleration and rendering. The details have already been mentioned.

6 Ackowledgement

Thanks to my best teammate who have done his work pretty well! Thanks to Professor *Li Yi* for his excellent course and TAs for their hard work. Thanks to the helpful classmates who have helped us a lot.

References

[1] Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 147–155, 2015.

[2] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. Sph fluids in computer graphics. 2014.