Realistic Multi-Material Interaction Simulation Pipeline

Yiyang Lu
luyy24@mails.tsinghua.edu.cn
IIIS, Tsinghua University
Beijing, China

Abstract

We present a comprehensive pipeline for realistic multi-material interaction simulation in the report. Our pipeline supports the simulation of rigid bodies, fluids, cloth, and smoke, and their interactions. We employ state-of-the-art methods such as Smoothed Particle Hydrodynamics (SPH) for fluid simulation, mass-spring systems for cloth simulation, and Eulerian methods for smoke simulation. Additionally, we implement efficient coupling techniques to handle interactions between different materials, including rigid-fluid and rigid-cloth coupling. Our implementation leverages GPU acceleration and spatial hashing to achieve high performance. The results demonstrate the effectiveness and versatility of our pipeline in producing realistic and visually compelling simulations. Challenges and future directions are also discussed.

Our code is open source and available on Github.

Keywords

Physics-based simulation, SPH, Mass-spring system, Coupling, Rendering

ACM Reference Format:
Yiyang Lu. 2025. Realistic Multi-Material Interaction Simulation Pipeline.

1 Introduction

Physics-based simulation has become increasingly important in computer graphics, especially for creating realistic animations and virtual environments. Among the various materials that can be simulated, rigid bodies, fluids, cloth, and smoke are some of the most common and challenging ones. Each material has its unique physical properties and requires different simulation methods. Moreover, the interactions between different materials add another layer of complexity to the simulation.

In this report, we present a comprehensive simulation pipeline that supports multiple materials and their interactions. Our pipeline implements state-of-the-art methods for each material: Smoothed Particle Hydrodynamics (SPH) for fluid simulation, mass-spring systems for cloth simulation, Eulerian methods for smoke simulation, and rigid body dynamics. We also implement efficient coupling techniques to handle interactions between different materials. The implementation leverages GPU acceleration and spatial hashing to achieve high performance. The results demonstrate the effectiveness and versatility of our pipeline in producing realistic and visually compelling simulations.

The remainder of this report is structured as follows:

- **Methods**: We introduce the simulation methods for rigid bodies, fluids, cloth, and smoke.
- **Implementation**: We describe the implementation details of our simulation pipeline, including configuration, simulation, and rendering.
- **Results**: We present the results of our simulations and demonstrate the effectiveness of our pipeline.
- **Discussion**: We discuss the challenges and future directions of our work.
- **Conclusion**: We conclude the report, summarizing the contributions of our work.

2 Methods

We implement simulation techniques for four types of materials: rigid bodies, fluids, cloth, and smoke. The following sections detail the methods used for each material.

2.1 Rigid

The simulation of rigid body is simple. Rigid body's state is defined by position and orientation. When applying force $\mathbf{f}_i$ to the rigid body at $\mathbf{p}_i$

$$\mathbf{F} = \sum_i \mathbf{f}_i \tag{1}$$

$$\mathbf{M} = \sum_i (\mathbf{p}_i - \mathbf{r}) \times f_i \tag{2}$$

where $\mathbf{r}$ is the center of mass. Linear and angular acceleration can be calculated by

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \mathbf{a} = \frac{\mathbf{F}}{m} \tag{3}$$

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = \alpha = \mathbf{I}^{-1}\mathbf{M} \tag{4}$$

where $\mathbf{I}$ is the intertia tensor.

2.2 Fluid

The main idea of fluid simulation is to solve the Navier-Stokes equation. The Navier-Stokes equation is a set of non-linear partial differential equations that describe the motion of fluid substances. The equation is as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{5}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{6}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, ρ is the density of the fluid, ν is the kinematic viscosity, and $\mathbf{f}$ is the external

Advanced Computer Graphics, Beijing, China

force field. The first equation is the momentum equation and the second equation is the continuity equation. To solve the two equations numerically, Lagrangian method and Eulerian method are two common methods. In this project, we will use Lagrangian particle-based method, specifically **Smoothed Particle Hydrodynamics (SPH)** method, to simulate the fluid.

The main idea of SPH is to discretize the fluid into particles and calculate the physical quantities at each particle. The core formula of SPH is kernel function $W(\mathbf{r}, h)$, which is used to interpolate the physical quantities between particles. The kernel function in our implementation is defined as follows:

$$W(\mathbf{r}, h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6\left(\frac{r}{h}\right)^2 + 6\left(\frac{r}{h}\right)^3 & 0 \le \frac{r}{h} < \frac{1}{2} \\ 2\left(1 - \frac{r}{h}\right)^3 & \frac{1}{2} \le \frac{r}{h} < 1 \\ 0 & \frac{r}{h} \ge 1 \end{cases} \tag{7}$$

where r is the distance between two particles and h is the smoothing length. Define $W_{ij} = W(\mathbf{r}_i - \mathbf{r}_j, h)$. The density at each particle can be estimated by the following equations:

$$\rho_i = \sum_j m_j W_{ij} \tag{8}$$

We implement three SPH solvers: **WCSPH**, **DFSPH** and **PCISPH**. The main difference among them lies in how they handle pressure: WCSPH uses an equation of state for pressure, PCISPH iteratively adjusts pressure to enforce incompressibility, and DFSPH ensures both density and velocity divergence constraints for improved stability and accuracy. The following sections detail the implementation of each SPH solver.

2.2.1 *Weakly Compressible SPH (WCSPH).*

WCSPH is a weakly compressible SPH solver, which is suitable for simulating incompressible fluid[Metaxas and Popovic 2007]. In WCSPH, the pressure field is calculated by:

$$p_i = B\left(\left(\frac{\rho_i}{\rho_0}\right)^\gamma - 1\right) \tag{9}$$

where B is the bulk modulus, γ is the polytropic index, ρ_i is the density at particle i, and ρ_0 is the reference density. The acceleration at particle i consists of four parts: pressure, viscosity, gravity, and surface tension. The pressure is calculated by:

$$\mathbf{a}_{\text{pressure},i} = -\frac{1}{\rho_i}\nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} \tag{10}$$

The viscosity is calculated by[1]:

$$\mathbf{a}_{\text{viscosity},i} = \nu \sum_j m_j \frac{v_{ij}^T r_{ij}}{r_{ij}^2 + \epsilon^2 h^2}\nabla W_{ij} \tag{11}$$

The surface tension is calculated by:

$$\mathbf{a}_{\text{surface tension},i} = -\frac{\kappa}{m_i}\sum_j m_j W_{ij} \tag{12}$$

[1]In practice, for stability and symmetry, we change m_j to $\frac{m_i + m_j}{2}$.

where κ is the surface tension coefficient, and $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$. Gravity is trivial, hence

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \mathbf{a}_{\text{pressure}} + \mathbf{a}_{\text{viscosity}} + \mathbf{a}_{\text{surface tension}} + \mathbf{g} \tag{13}$$

2.2.2 *Divergence-Free SPH (DFSPH).*

DFSPH is a density filter SPH solver, which is suitable for simulating incompressible fluid[Bender and Koschier 2015]. The main idea of DFSPH is to filter the density field to make the density field satisfy the equation of state. We don't need to calculate the pressure field explicitly. After computing non-pressure forces using 11 and 12, correct density and $\nabla \cdot \mathbf{v}$ by iteratively

In our implementation, define

$$\alpha_i = \frac{\rho_i}{\sum_j \|m_j \nabla W_{ij}\|^2 + \|\sum_j m_j \nabla W_{ij}\|^2} \tag{14}$$

$$\kappa_i^v = \frac{1}{\Delta t}\frac{\mathrm{D}\rho}{\mathrm{D}t}\alpha_i = \frac{\rho_i}{\Delta t}\alpha_i \sum_j m_j \left(v_i - v_j\right)\cdot \nabla W_{ij} \tag{15}$$

update velocity by

$$v_i^{n+1} = v_i^n - \Delta t \sum_j m_j \left(\frac{\kappa_i^v}{\rho_i} + \frac{\kappa_j^v}{\rho_j}\right)\nabla W_{ij} \tag{16}$$

until the divergence of velocity field (which is proportional to $\frac{\mathrm{D}\rho}{\mathrm{D}t}$) is small enough. Similarly, define

$$\rho_i^* = \rho_i + \Delta t \sum_j m_j \left(v_i - v_j\right)\cdot \nabla W_{ij} \tag{17}$$

$$\kappa_i = \frac{\rho_i^* - \rho_0}{\Delta t^2}\alpha_i \tag{18}$$

update velocity by

$$v_i^{n+1} = v_i^n - \Delta t \sum_j m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j}\right)\nabla W_{ij} \tag{19}$$

until $\rho_{\text{avg}}^* - \rho_0$ is small enough.

2.2.3 *Predictive-corrective incompressible SPH(PCISPH).*

PCISPH is a pressure-implicit incompressible SPH solver, which is well-suited for scenarios where maintaining incompressibility is critical, such as simulating water splashes, fluid flow in narrow channels, and interactions between liquids and solid boundaries[Solenthaler and Pajarola 2009].

The main idea of PCISPH is to predict the density field and then correct it to satisfy the incompressibility constraint. Derived from 8, the derivation of density at particle i can be estimated by

$$\begin{aligned} \Delta\rho_i &= \sum_j m_j \nabla W_{ij} \cdot (v_i - v_j)\Delta t \\ &= \Delta x_i \sum_j m_j \nabla W_{ij} - \sum_j m_j \Delta x_j \nabla W_{ij} \end{aligned} \tag{20}$$

Δx can be derived from the time integration scheme, neglecting all the forces but pressure, we have

$$\Delta x_i = \frac{\mathbf{F}_i^p}{m_i}\Delta t^2 \tag{21}$$

We assume all neighbors share same pressure $\tilde{p}_i$ and density ρ_0, then

$$\mathbf{F}_i^p = -m_i \sum_j m_j \left(\frac{\tilde{p}_i}{\rho_0^2} + \frac{\tilde{p}_i}{\rho_0^2} \right) \nabla W_{ij} \tag{22}$$

hence

$$\Delta x_i = -\Delta t^2 \frac{2\tilde{p}_i}{\rho_0^2} \sum_j m_j \nabla W_{ij} \tag{23}$$

Due to the pressure p_i of particle i the position of a neighboring particle j changes by $\Delta x_{j|i}$. As the pressure forces are symmetric, particle j gets the contribution from i

$$\mathbf{F}_{j|i}^p = m_i m_j \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij} \tag{24}$$

and the position of j changes by

$$\Delta x_{j|i} = \Delta t^2 m_i \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij} \tag{25}$$

We only consider the effect of the central particle i here, i.e. $\Delta x_j = \Delta x_{j|i}$j, insert 23 and 25 into 20

$$\Delta \rho_i = -\Delta t^2 \frac{2\tilde{p}_i}{\rho_0^2} \left(\| \sum_j m_j \nabla W_{ij} \|^2 + \sum_j m_i m_j \| \nabla W_{ij} \|^2 \right) \tag{26}$$

After solving $\tilde{p}_i$, we have

$$\tilde{p}_i = -\frac{\Delta \rho_i \rho_0^2}{2 \left(\| \sum_j m_j \nabla W_{ij} \|^2 + \sum_j m_i m_j \| \nabla W_{ij} \|^2 \right)} \tag{27}$$

The meaning of 27 is that a pressure $\tilde{p}_i$ is needed to achieve $\Delta \rho_i$. As we know the predicted density error is $\Delta \rho_i$ is $\rho_i - \rho_0$, we update our pressure using $p_i += \tilde{p}_i$ until $\Delta \rho_i < \eta$

2.2.4 *Rigid-Fluid Coupling.*

Rigid-Fluid coupling is a key component in fluid simulation. The main idea is to transfer the force from fluid to rigid body and vice versa[Akinci et al. 2012]. In our implementation, when coupling fluid and rigid body, we convert rigid body from mesh to point clouds. The contribution of j-th particle in rigid body to i-th particle in fluid is calculated by

$$\rho_{ij} = m_j W_{ij} \tag{28}$$

$$\mathbf{a}_{\text{viscosity},ij} = \nu m_j \frac{v_{ij}^T r_{ij}}{r_{ij}^2 + \epsilon^2 h^2} \nabla W_{ij} \tag{29}$$

Surface tension is not considered in rigid-fluid coupling. In WCSPH, the pressure field is calculated by

$$\mathbf{a}_{\text{pressure},ij} = -m_j \frac{p_i}{\rho_i^2} \nabla W_{ij} \tag{30}$$

In DFSPH, we modify the

$$\alpha_i = \frac{\rho_i}{\sum_{j \in \text{fluid}} \| m_j \nabla W_{ij} \|^2 + \| \sum_{j \in \text{fluid and rigid}} m_j \nabla W_{ij} \|^2} \tag{31}$$

and

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = \frac{\rho_i}{\Delta t} \alpha_i \sum_{j \in \text{fluid and rigid}} m_j \left(v_i - v_j \right) \cdot \nabla W_{ij} \tag{32}$$

The rigid body doesn't have attribute κ_i^v and κ_i in physical sense. For consistency, we set $\kappa_i^v = 0$ and $\kappa_i = 0$. The velocity update for divergence correction is

$$v_i^{n+1} = v_i^n - \Delta t \sum_{j \in \text{fluid and rigid}} m_j \left(\frac{\kappa_i^v}{\rho_i} + \frac{\kappa_j^v}{\rho_j} \right) \nabla W_{ij} \tag{33}$$

update 17 and 19 by

$$\rho_i^* = \rho_i + \Delta t \sum_{j \in \text{fluid and rigid}} m_j \left(v_i - v_j \right) \cdot \nabla W_{ij} \tag{34}$$

$$v_i^{n+1} = v_i^n - \Delta t \sum_{j \in \text{fluid and rigid}} m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j} \right) \nabla W_{ij} \tag{35}$$

In PCISPH, we modify pressure computation as

$$\mathbf{a}_{\text{pressure},i} = -\sum_{j \in \text{fluid and rigid}} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} 1_{j \in \text{fluid}} \right) \nabla W_{ij} \tag{36}$$

2.3 Cloth

The main idea of cloth simulation is to discretize the cloth into two-dimensional square grid points, with the mass discretely distributed at each vertex, and the mass points connected by springs. For the sake of simplicity of calculation, and considering that the first-order approximation of many physical systems at small scales is linear, the springs are considered to be linear restoring forces, that is, they satisfy Hooke's law.

2.3.1 *Mass-Spring Model.*

To accurately represent the behavior of the cloth, we employ a **mass-spring system** as demonstrated in Figure 1 with varying properties:

(1) Structural Springs: Connect adjacent mass points, ensuring basic proximity constraints and preventing excessive stretching.
(2) Shear Springs: Connect diagonally adjacent mass points, enhancing the cloth's resistance to shearing forces and improving its isotropic behavior.
(3) Bend Springs: Connect mass points further apart, preventing excessive bending and maintaining the cloth's shape more realistically.

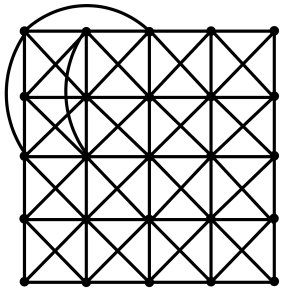

Figure 1: Mass-Spring Model

The internal forces acting on a mass point due to these springs are calculated as:

$$\mathbf{F}_{\text{internal}} = k_{\text{struct}}\Delta_{\text{struct}}\mathbf{x} + k_{\text{shear}}\Delta_{\text{shear}}\mathbf{x} + k_{\text{bend}}\Delta_{\text{bend}}\mathbf{x} \tag{37}$$

The motion of each mass point is governed by Newton's second law:

$$\mathbf{F}_{\text{external}} + \mathbf{F}_{\text{internal}} = m\mathbf{a} \tag{38}$$

2.3.2 *Cloth-Rigid Body Interaction.*

To simulate interactions between cloth and rigid bodies, we employ a collision detection and response system:

(1) Collision Detection: Update the position of each cloth mass point based on the forces acting upon it. Detect collisions between cloth mass points and the rigid body.
(2) Collision Response: if a collision occurs, adjust the position of the colliding mass point to prevent penetration. Calculate the change in velocity of the mass point, and impulse transmitted from the mass point to the rigid body.
(3) Rigid Body Motion Update: Accumulate the impulses from all colliding cloth mass points. Update the position and velocity of the rigid body based on the total impulse and external forces acting upon it:

$$\mathbf{I}_{\text{total}} = -M\mathbf{g}\Delta t + \sum_{i \in \text{all particles in cloth}} \mathbf{I}_i \tag{39}$$

where M is the mass of the rigid.

2.3.3 *Parameter Consideration.*

The accuracy and realism of cloth simulation heavily depend on the appropriate selection of simulation parameters:

- Time Step: A small time step is crucial for accurate integration of the equations of motion. However, excessively small time steps can increase computational cost.
- Spring Stiffness: High stiffness can lead to unrealistic behavior and numerical instability. Low stiffness may result in overly loose and unrealistic cloth deformations.
- Damping: Damping coefficients control energy dissipation within the cloth. Inappropriate damping can lead to excessive energy loss or unrealistic oscillations.

2.4 Smoke

The main idea of smoke simulation is to model the physical behavior and visual appearance of smoke as an incompressible fluid governed by the Navier-Stokes equations. Our goal is to balance physical realism and computational efficiency to produce natural, turbulent, and dynamic smoke effects. We will introduce our implementation following [Fedkiw et al. 2001] in the below.

2.4.1 *Discretization of the Navier-Stokes Equations.*

We model our gases as inviscid, incompressible, constant density fluids. The effects of viscosity are negligible in gases especially on coarse grids where numerical dissipation dominates physical viscosity and molecular diffusion. When the smoke's velocity is well below the speed of sound the compressibility effects are negligible as well, and the assumption of incompressibility greatly simplifies the numerical methods.

Specifically, the simulation of smoke involves solving Euler equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{f} \tag{40}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{41}$$

p is the pressure of the gas and $\mathbf{f}$ represents the external forces. Equations 40 and 41 are solved in three stages.

First, we compute an intermediate velocity field $\mathbf{u}^*$

$$\frac{\mathbf{u}^* - \mathbf{u}}{\Delta t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{f} \tag{42}$$

Then, to enforce the incompressibility condition ($\nabla \cdot \mathbf{u} = 0$), the velocity field is projected onto a divergence-free vector field

$$\nabla^2 p = \frac{\nabla \cdot \mathbf{u}^*}{\Delta t} \tag{43}$$

with pure Neumann boundary condition

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \tag{44}$$

at boundary point with normal vector $\mathbf{n}$. Finally, after computing p, our velocity field is made incompressible by updating $\mathbf{u}$

$$\mathbf{u} = \mathbf{u}^* - \frac{1}{\rho}\nabla p \Delta t \tag{45}$$

position of smoke particles are updated according to $\mathbf{u}$.

2.4.2 *Density and Temperature Evolution.*

The smoke density ρ and temperature T evolve over time along the smoke velocity

$$\frac{\partial T}{\partial t} = -(u \cdot \nabla)T \tag{46}$$

$$\frac{\partial \rho}{\partial t} = -(u \cdot \nabla)\rho \tag{47}$$

Both density and temperature affect the velocity of the fluid. Heavy smoke tends to fall downward because of gravity, while hot gases tend to rise because of buoyancy. We model these effects by defining external forces that are directly proportional to the density and the temperature

$$\mathbf{f}_{\text{buoy}} = -\alpha\rho\mathbf{z} + \beta(T - T_{\text{amb}})\mathbf{z} \tag{48}$$

where $\mathbf{z} = (0, 0, 1)$ is vertical direction, T_{amb} is ambient temperature of air, α and β are constants. We solve terms involving $-(\mathbf{u} \cdot \nabla)$ by semi-Lagrangian method[Staniforth and Côté 1991].

2.4.3 *Vorticity Confinement.*

To improve the visual fidelity of swirling or turbulent motions, especially in simulations of smoke, vorticity confinement is employed. This technique addresses the loss of small-scale vortices (rotating flow structures) that occurs due to numerical dissipation within grid-based simulations.

The vorticity $\boldsymbol{\omega}$ of a velocity field $\mathbf{u}$ is defined as the curl of the velocity:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} \tag{49}$$

which characterizes the rotational nature of the flow. Numerical dissipation weakens these rotational effects. Vorticity confinement aims to counteract this by introducing an artificial force that mimics

the influence of these lost small-scale vortices. Define first normalized vorticity location vectors

$$N = \frac{\eta}{\|\eta\|} \quad \eta = \nabla\|\boldsymbol{\omega}\| \tag{50}$$

that point from lower vorticity concentrations to higher vorticity concentrations are computed. The magnitude and direction of the paddle wheel force is computed as

$$\mathbf{f}_{\text{conf}} = \epsilon h(N \times \boldsymbol{\omega}) \tag{51}$$

The magnitude of this force is carefully controlled by ϵ to ensure that the simulation remains physically accurate as the grid resolution is increased.

3 Implementation

Our implementation is organized into three primary stages: configuration, simulation, and rendering.

3.1 Configuration

Users can configure the scene by specifying the materials, initial state, and external forces. Specifically, for rigid and fluid, user can specify the initial position by loading any **mesh** file or use our built-in geometry; for cloth, user can specify the position of fixed point. Besides, any physical parameters useful for simulation can be set by the user. Camera perspectives are also customizable.

3.2 Simulation

All simulations are implemented in Python, especially Taichi, a high performance acceleration library. We implement **GPU acceleration** by setting `ti.init(arch=ti.gpu)`.

In naive implementation of fluid and cloth simulations, computational efficiency is a critical concern, particularly for operations like neighbor searches and collision detections, which requires an exhaustive search across all particles, resulting in a computational complexity of $O(n^2)$. We apply **Spatial Hash** to overcome this bottleneck, significantly enhance the efficiency of both fluid and cloth simulations.

Spatial hashing divides the simulation domain into a grid of cells. Particles are then assigned to their corresponding cells, and the neighbor search is restricted to particles within the same cell and its immediate neighbors, which reduces the computational complexity to $O(n)$.

Besides, we implement **real-time interaction** for cloth simulation, as well as rigid-cloth coupling. You can adjust the parameters and the initial position of rigid using mouse in real-time.

3.3 Rendering

For all three materials, we store the mesh as simulation results. The simulation of fluid is implemented in point cloud, followed by splashsurf [Löschner et al. 2023] to reconstruct the surface. Simulation result of smoke is represented by the density field, which is visualized by volume rendering. All the simulation results are produced by Blender, a powerful rendering software that supports Python scripting. To speed up surface reconstruction and rendering, we use **multi-threading** to parallelize the whole process.

4 Experiment Results

In this section, we present several compelling results generated using our pipeline. For all simulations, we consistently use the same hyper-parameter settings: particle radius $r = 0.01$, supporting radius $h = 4r$, rest density $\rho_0 = 1000\text{kg/m}^3$, and time step $\Delta t = 0.0001$. The rendering results is done in Blender at 60 frames per second.

4.1 Rigid-Fluid Coupling

In this chapter, we present the results of rigid-fluid coupling, including fluid coupling with both fixed and movable rigid bodies. Specifically, we simulate a water droplet falling on a fixed bunny using both WCSPH (Figure 2)

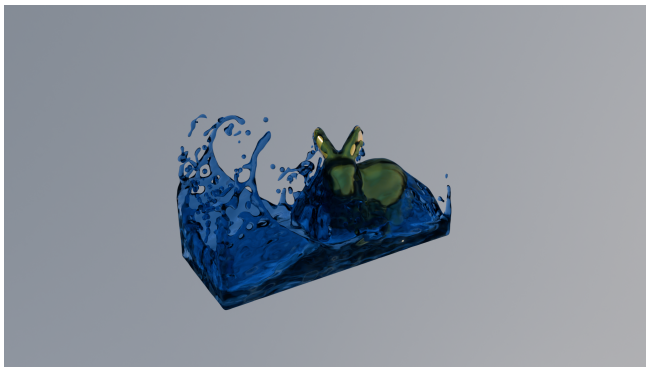

Figure 2: Rigid-Fluid Coupling in WCSPH

and DFSPH (Figure 3).

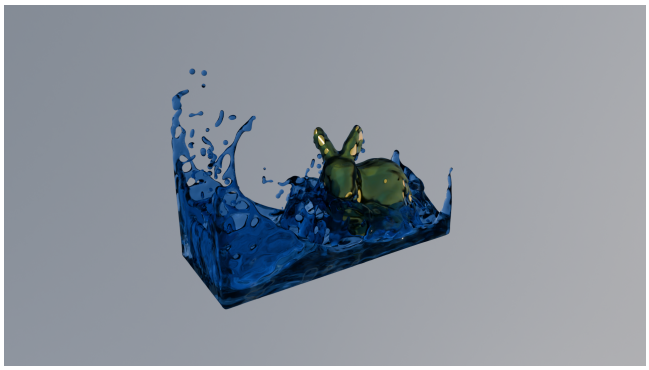

Figure 3: Rigid-Fluid Coupling in DFSPH

The number of fluid particles is 400,000. By checking the simulation results, we can discover that DFSPH has smaller energy dissipation than WCSPH.

Besides, two-way rigid-fluid coupling is presented by simulating a rubber duck falling into a pool of water using PCISPH (Figure 4). The number of fluid particles is 700,000.

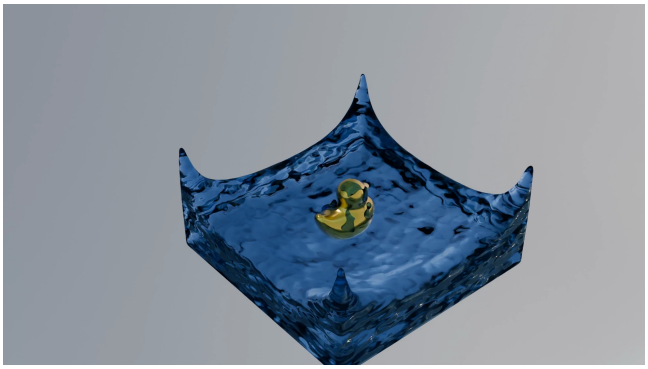

Figure 4: Two-way Rigid-Fluid Coupling

4.2 Rigid-Cloth Coupling

In this chapter, we present the results of rigid-cloth coupling, including cloth coupling with both fixed and movable rigid bodies. Specifically, we simulate a cloth dropping over a fixed rigid body (Figure 5)

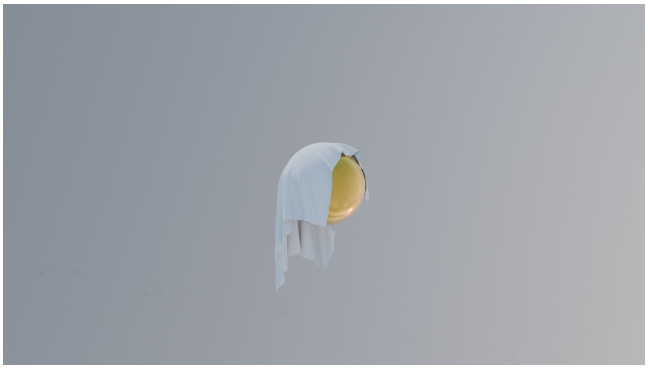

Figure 5: One-way Rigid-Cloth Coupling

and a ball falling into a cloth fixed at four corners (Figure 6) with customized background.

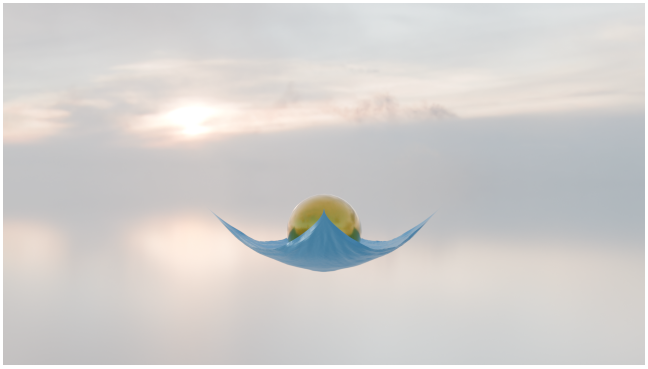

Figure 6: Two-way Rigid-Cloth Coupling

4.3 Smoke

In this chapter, we present the results of smoke simulation. Specifically, we simulate rising smoke restricted in a box (Figure 7).

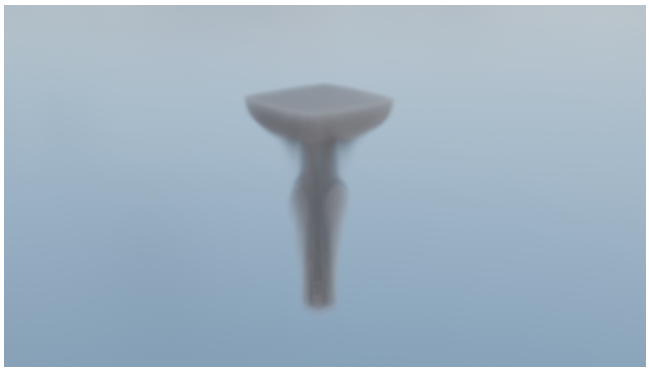

Figure 7: Smoke

4.4 Interactive Real-time Rendering

In this chapter, we present the results of interactive real-time rendering. We simulate a cloth falling into a rigid body. Users can adjust the parameters and initial position of the rigid body using the mouse in real-time (Figure 8).

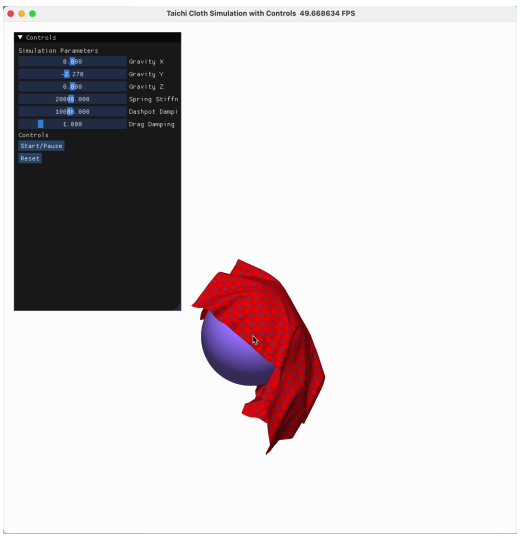

Figure 8: Real-time Rendering

5 Discussion

In this section, we discuss the challenges encountered during the development of our simulation pipeline and propose potential improvements for future work.

5.1 Challenges

During the development of our simulation pipeline, we encountered several challenges that required careful consideration and innovative solutions. Some of the key challenges, along with our approaches to addressing them, are outlined below:

(1) **Numerical Stability**: Ensuring numerical stability, especially in fluid simulations, was a significant challenge. Small time steps were required to maintain stability, which increased computational cost. Implementing advanced numerical methods like DFSPH helped improve stability.
(2) **Performance Optimization**: Achieving real-time performance for complex simulations involving multiple materials and interactions required extensive optimization. Implementing GPU acceleration and spatial hashing helped, but further optimization is needed, especially utilizing Taichi's parallel computing capabilities.
(3) **Collision Handling**: Accurate and efficient collision detection and response, particularly for cloth self-collisions and rigid-fluid interactions, was challenging. Ensuring that collisions were handled correctly without causing instability or unrealistic behavior required careful tuning.
(4) **Parameter Tuning**: Selecting appropriate parameters for different simulation methods (e.g., spring stiffness for cloth, viscosity for fluids) was non-trivial. Incorrect parameters could lead to unrealistic simulations or numerical instability. We relied on empirical testing, simulation visualization, and physical intuition to tune these parameters effectively.
(5) **Rendering Integration**: Integrating the simulation results with rendering tools like Blender to produce visually appealing outputs required additional effort. To ensure that the rendered results accurately representing the simulation data, we search for most realistic materials and lighting settings.

5.2 Potential Improvements

There are several areas where our simulation pipeline can be improved:

(1) **Adaptive Time Stepping**: Implementing adaptive time stepping could improve both the accuracy and performance of the simulations. By dynamically adjusting the time step based on the simulation's current state, we can ensure stability while reducing computational overhead.
(2) **Advanced Collision Handling**: Enhancing the collision detection and response system, particularly for cloth self-collisions, could lead to more realistic simulations. Techniques such as continuous collision detection and more sophisticated response models could be explored.
(3) **Multi-Resolution Methods**: Using multi-resolution methods for fluid and cloth simulations could improve performance without sacrificing detail. By simulating different regions of the material at varying levels of detail, we can focus computational resources on areas of interest.
(4) **Improved Rendering Techniques**: While our current rendering pipeline produces visually appealing results, there is potential for further enhancement. More techniques such as more advanced shading models for fluids could be explored in Blender.
(5) **User Interface and Usability**: Improving the user interface and usability of our simulation pipeline could make it more accessible to a broader audience. More kinds of materials and interactions could be added to the pipeline.

By addressing these challenges and implementing the proposed improvements, we believe our simulation pipeline can become even more powerful and versatile, enabling the creation of highly realistic and interactive simulations for a wide range of applications.

6 Conclusion

In this project, we have developed a comprehensive simulation pipeline for simulating rigid, fluid, cloth, and smoke materials. Our pipeline leverages state-of-the-art numerical methods, including SPH solvers, mass-spring models, and Navier-Stokes simulations, to accurately model the physical behavior of these materials. By implementing advanced techniques such as spatial hashing, GPU acceleration, and multi-threading, we have achieved high performance for complex simulations involving multiple materials and interactions. Our rendering pipeline integrates simulation results with splashsurf and Blender to produce visually appealing outputs. We have demonstrated the effectiveness of our simulation pipeline through compelling results, including rigid-fluid, rigid-cloth coupling, smoke simulations, as well as interactive real-time rendering. By addressing current challenges and proposing potential improvements, we have laid the foundation for future work in this area. We believe that our simulation pipeline has the potential to be a valuable tool for developers working in computer graphics, animation, and visual effects.

Personal Contribution Statement

My personal contribution to this project is as follows:

(1) **Basic Types**: I mainly worked on rigid, fluid simulation for basic types. In fluid simulation, I implemented 3 kinds of different SPH solvers: WCSPH, DFSPH, and PCISPH. Basic rendering is also implemented.
(2) **Coupling**: I implemented rigid-fluid coupling for all 3 kinds of SPH solver, as well as rigid-cloth coupling. In both cases, I implemented two-way coupling, which means both rigid and fluid/cloth can affect each other.
(3) **Acceleration**: To speed up the simulation, I implemented spatial hashing to accelerate the neighbor search in fluid simulation, as well as self-collision detection in cloth simulation. Advanced numerical methods like PCISPH are also implemented to improve stability.
To further speed up the simulation, I implemented GPU acceleration by setting `ti.init(arch=ti.gpu)`. Multi-threading parallelization is also implemented in rendering.

Acknowledgment

I sincerely thank my teammate Xiangchen Tian for his hard work and dedication to this project. I would also like to express my gratitude to the course instructors and TA for their guidance and support throughout the semester. This project would not have been possible without their help.

References

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 31, 4, Article 62 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185558

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

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 15–22.

Fabian Löschner, Timna Böttcher, Stefan Rhys Jeske, and Jan Bender. 2023. Weighted Laplacian Smoothing for Surface Reconstruction of Particle-based Fluids. In *Vision, Modeling, and Visualization*. The Eurographics Association. https://doi.org/10.2312/vmv.20231245

D Metaxas and J Popovic. 2007. Weakly compressible SPH for free surface flows. (2007).

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH 2009 papers*. 1–6.

Andrew Staniforth and Jean Côté. 1991. Semi-Lagrangian integration schemes for atmospheric models—A review. *Monthly weather review* 119, 9 (1991), 2206–2223.