COMP559 Final Project Report
Particle-based Fluid Simulation

Yue Wang

Abstract

This report covers the implementation of particle-based fluid simulation using a Smoothed Particle Hydrodynamics(SPH). The method used is mostly based on the paper of Matthias Muller [] and is commonly referred to as the PIC/FLIP method. In the method, the fluid surface is tracked using particles and its mass conserved by enforcing zero divergence on the deforming velocity field. We store the velocity field on a staggered grid as described in [2], which greatly helps to fulfill the mass-conserving criteria. We also reconstruct the actual fluid surface by evaluating the Improved Blob- bies signed distance function introduced in [3] on a regular grid followed by a standard implementation of Marching Cubes. We have tested our implementation using five different simulation cases and obtain great results, all of which are shown in the report together with benchmark data.

1 Introduction

Water, air, and smoke, these fluids are very common things in every day life. Simulation of fluids can be very interesting, which allows us to design creative scenes that will never happen, or simulate realistic scenes that people can't tell whether it's simulation or reality. Simulation of fluids can also be very useful, which can be applied in modern movies and video games.

2 Related Work

Early works such as [**?**] and [**?**] defines the water in staggered MAC grids, uses marker particles and level set to define the free surfaces. [**?**] gives a thorough introduction on fluid simulation methods, which covered different types of fluids, and many state of art methods. Chapter 9 introduced Smoothed Particle Hydrodynamics(SPH), which is based on [**?**], uses particles instead of a stationary grid simplifies the simulation substantially. in contrast to Eulerian grid-based approaches, the particle-based approach makes mass conservation equations and convection terms dispensable which reduces the complexity of the simulation. Therefore, in this project, we will mainly implement the SPH method in [**?**].

3 Technical Details

3.1 SPH

SPH is an interpolation method for particle systems. It defines a smoothing kernel function to interpolate(smooth) between discrete particles. The kernel works like a KNN algorithm, which gives different weights to particles in the vicinity of the position to be computed.

3.2 Modeling Fluids using SPH

Different smoothing kernels are provided for pressure and viscosity, to satisfy the physical properties. Using the Navier Stokes Equation,

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{g} + v\nabla^2\mathbf{u} - \mathbf{u}\nabla\mathbf{u} - \frac{1}{\rho}\nabla p \tag{1}$$

where $\mathbf{g}$ is gravity and other external forces, v is the viscosity of the fluid, $\mathbf{g}$, $v\nabla^2\mathbf{u}$, $\frac{1}{\rho}\nabla p$ are updated according to SPH rule, i.e. interpolated according to different smoothing kernel. Surface tension is treated as forces acting near the surface, added to particles near surface.

4 Results

4.1 Libraries

We'll need OpenGL libraries for graphical support, and a math library such as vecmath for data structure and computational support. For the rendering part, library available is not clear, since water rendering is complex. If time allowed, we'll explore some code base or implement a photon mapping.

4.2 Solvers

The idea behind [**?**] is easy to understand. The terms in 1 are updated in a relatively independent way, and the stability, accuracy, and speed mainly depend on the smoothing kernels. We didn't see any needs to use specific solvers for now. We would like to use simpletic integration to update the position of particles.

4.3 Development Plan

Based on the time budget we have, the development plan is given below. The easy part is update pressure and velocity, because we are familiar with them from assignments. The difficult part would be surface tension computation and color field, because it is not fully elaborated in [**?**]. Another difficult part would be rendering, since water volume cannot be well rendered by simple ray tracer.

Table 1: *Project Time Schedule*

✓	design data structure, implement kernel functions, position and velocity update
Mar. 25	implement pressure and viscosity update, color field and surface tension
Apr. 1	add mouse interaction and external forces, test new scenes
Apr. 8	write report, prepare for presentation, demo recording

5 Conclusion

We will use dam break and faucet scenes for testing and recording a demo. If there's enough time left, we would consider complex scenes like pouring water into a glass bottle. Then we consider rendering and adding interactions.