3D Fluid Dynamics Simulation of Different Smoke Compositions and Containers

ZHENGJIANG HE, University of California, Berkeley, USA
HIROTAKA ISHIHARA, University of California, Berkeley, USA
CHAOMIN LI, University of California, Berkeley, USA
ZHIQI YAN, University of California, Berkeley, USA

There are currently two common approaches to simulate fluids like smoke, a grid-based approach or a particle-based approach. In lecture, we discussed the grid-based approach in detail. However, the particle-based smoke simulation is also a popular choice in many situations due to its simplicity compared to a grid based approach. In this paper, we present a particle-based smoke simulation application. We simulated each particle using Naiver-Stoke equation to ensure a more realistic simulation. We then used OpenGL to accelerate the rendering of particles.

ACM Reference Format:
Zhengjiang He, Hirotaka Ishihara, Chaomin Li, and Zhiqi Yan. 2023. 3D Fluid Dynamics Simulation of Different Smoke Compositions and Containers. 1, 1 (May 2023), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

There are currently two common approaches for simulating fluid, a grid-based approach and a particle based approach. Grid-based approach divides the entire simulation space to small grids, and store information of the fluid (including velocity, density, pressure etc.) in each grid for simulation. We discussed this approach in detail in our lecture. Particle based approach simulates fluid as particles. The simulation involves computing the forces exerted on each particle, and interactions between all particles.

In this project, we will explore a little bit on how the particle based simulation approach works by creating a 3D fluid dynamics simulation that models the behavior of an incompressible, homogeneous fluid under the Navier-Stokes equations. The simulation should include the ability to simulate smoke with different compositions and to model the effects of different containers made of various materials and shapes.

2 PROBLEM

We are going to develop a 3D fluid dynamics simulation that estimates the behavior of an incompressible, homogeneous fluid under the Navier-Stokes equations. The simulation should model the movement of smoke and how they interact with different objects, such as barriers / containers made of different materials and shapes.

Our simulation will also model the behavior of smoke with different compositions, which models the movement of particles within the smoke, and how they interact with each other and with the fluid. To simulate the effects of different

Authors' addresses: Zhengjiang He, University of California, Berkeley, USA, zhengjiang.he@berkeley.edu; Hirotaka Ishihara, University of California, Berkeley, USA, hirotaka.ishihara@berkeley.edu; Chaomin Li, University of California, Berkeley, USA, zhaominl@berkeley.edu; Zhiqi Yan, University of California, Berkeley, USA, zhiqiyan@berkeley.edu.

materials and shapes of the barriers/containers, we will need to implement boundary conditions that represent the surfaces of the containers. These boundary conditions should be able to accurately represent the properties of the materials used to construct the containers and should be able to account for any effects that these materials may have on the fluid and smoke.

3 IMPLEMENTED ALGORITHMS AND TECHNICAL APPROACHES

In order to simulate smoke in three-dimensional space, we have implemented a particle-based smoke simulation [1]. Our approach is a simplified implementation that utilizes Smoothed Particle Hydrodynamics (SPH) [2] to solve the Navier-Stokes equations, which are fundamental to fluid dynamics. We have employed the cubic spline kernel as our smoothing kernel function, providing a balance between accuracy and computational complexity. Additionally, our method incorporates the concept of grid cells to optimize calculation efficiency, resulting in a more performant simulation.

3.1 Framework and Techniques Used

We used NanoGUI and OpenGL to implement this project, similar to Homework 4. NanoGUI is a lightweight user interface library that offers a variety of widgets and tools for creating user interfaces to display our simulation in OpenGL.

In contrast to Homework 4, we did not introduce CGL as our linear algebra library. Instead, we used Eigen, a versatile, fast, and reliable C++ template library for linear algebra[3]. Since NanoGUI already uses Eigen as its linear algebra library, we simplified the dependencies of our project by sharing the same library.

When simulating smoke, calculating new velocity, force, and position for each particle in every draw is always the performance bottleneck. To improve our real-time performance, we used OpenMP for parallelization by dividing the simulation space into grids and parallelizing the inner-grid Navier-Stokes computations using OpenMP. By breaking down the simulation into smaller, more manageable parts, we can optimize the computation process and speed up the simulation. Furthermore, these techniques are essential for ensuring that our system accurately reflects the real-world behavior of smoke particles and produces high-quality visual results. With 8-thread parallelization *ompparallel*, we are able to run the simulation $8x$ faster than the simulation without parallel.

3.2 Simulation Workflow

(1) Initialize the simulation parameters, particle emitter, and spatial data structures.
(2) Generate new particles based on the emitter's properties and the simulation parameters.
(3) Update the simulation in each time step Δt by performing the following operations:
    (a) Build a hash-position-based spatial map to query particles in each grid cell efficiently.
    (b) Calculate the average particle attributes for each grid cell.
    (c) For each particle and the average particles (calculated above) of its neighboring cells, perform the following calculations:
        (i) Compute the temperature change based on the ambient temperature and diffusion coefficient.
        (ii) Calculate pressure forces related to neighbor cells using the kernel gradient and pressure values.
        (iii) Calculate viscosity forces using the dynamic viscosity coefficient and kernel Laplacian.
        (iv) Compute vorticity confinement forces based on the vorticity difference and gradient of the vorticity magnitude.

(v) Calculate air drag forces based on the air drag coefficient and particle velocity.

(d) After updating the forces that are correlated to its neighbors, now update the forces that are only correlated to its own property.

(i) Apply buoyancy forces based on gravity and its new temperature.

(ii) Apply damping forces based on the current vorticity.

(iii) Apply turbulence forces to simulate a more realistic smoke.

(e) Update the particle position and velocity using the calculated forces and remove particles that have reached the end of their lifespan.

(4) Repeat steps 2 and 3 for the desired number of time steps or until a stopping condition is reached.

3.3 Grid Cells

We utilized a hybrid approach for smoke simulation that integrates a particle-based technique for updating properties with a grid-based method for efficient interaction among neighboring particles. Particles are categorized and organized into grid cells based on their positions, using a hash position function in our implementation. The position output relies on screen size parameters and grid size parameters. By adjusting these size parameters, we can regulate the number of particles contained within a cell.

3.4 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian method that discretizes the fluid into a set of N particles ($i = 0, 1, ..., N-1$). Each particle carries fluid properties (in our case, smoke) such as position, velocity, density, viscosity etc. SPH uses smoothing kernel functions to approximate the continuous fluid properties based on neighboring particles. The main purpose of the kernel function is to distribute the influence of each particle over a finite region, defined by the smoothing length h, and to ensure the physical quantities are smooth and differentiable. In this project, we use a cubic spline kernel [4]. The cubic spline kernel is defined as follows:

$$W(r,h) = \frac{1}{\pi h^3} \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3, & 0 \le q < 1 \\ \frac{1}{4}(2-q)^3, & 1 \le q < 2 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $q = \frac{r}{h}$, r is the distance between particles, and h is the smoothing length.

The gradient of the cubic spline kernel is defined as:

$$\nabla W(\mathbf{d},h) = \begin{cases} \frac{d\mathbf{W}}{dr}\frac{\mathbf{d}}{r}, & r \neq 0 \\ \mathbf{0}, & r = 0 \end{cases} \tag{2}$$

where $\mathbf{d}$ is the displacement vector between particles, $r = ||\mathbf{d}||$, and $\frac{d\mathbf{W}}{dr}$ is computed as:

$$\frac{d\mathbf{W}}{dr} = \frac{1}{\pi h^4} \begin{cases} -3q + \frac{9}{4}q^2, & 0 \le q < 1 \\ -\frac{3}{4}(2-q)^2, & 1 \le q < 2 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The Laplacian of the cubic spline kernel is defined as:

$$\nabla^2 W(\mathbf{d}, h) = \frac{1}{\pi h^5} \begin{cases} \frac{12}{4}q - 6, & 0 \le q < 1 \\ -3(2-q), & 1 \le q < 2 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

3.5 Discretization and Solution Strategy

The Navier-Stokes equation is discretized for each particle i as follows:

$$\frac{du_i}{dt} = P_i + V_i + F_{rest} \tag{5}$$

where u_i is the velocity of the i-th smoothed particle, P_i represents the pressure forces, V_i denotes the viscosity forces, and G is the gravity forces acting on all particles. To solve this system of ordinary differential equations (ODEs), we follow a step-by-step approach to update the particles' positions. We will elaborate on each component in the subsections.

3.6 Solving equation

The algorithm we implemented consists of the following steps:

(1) Compute the right-hand side (RHS) for each particle F_i:
   (a) Calculate the distances to the neighbour cell position: $d_{ij} = ||x_i - x_j||_2$
   (b) Compute the density at each particle's position: $\rho_i = M \sum_j W_{ij}(d_{ij})$
   (c) Calculate the pressure at each particle's position using κ, which indicates the resistance to fluid compression, and a base density ρ_0: $p_i = \kappa(\rho_i - \rho_0)$
   (d) Compute the pressure force of each particle: $P_i = -M \sum_j \frac{p_j + p_i}{2\rho_j} \nabla W_{ij}(d_{ij})$
   (e) Calculate the viscosity force of each particle: $V_i = \mu M \sum_j \frac{u_j - u_i}{\rho_j} \nabla^2 W_{ij}(d_{ij})$
   (f) Calculate F_i^{rest}, which consists of buoyancy force
   (g) Sum up the RHS: $F_i = P_i + V_i + F_i^{rest}$

To clarify, L and M are the smooth-particle lengths and the mass of the particle, respectively. We pre-set L to be 1.5 of the particle size, and the mass is set to 1 unit. After we compute the updated force F_i, the velocity of the particle i can also be updated as $v_i^t = v_i^{t-1} + F_i \Delta t$, where v_i^t denotes the new velocity, v_i^{t-1} denotes the velocity of the last time step, and Δt is the time step we choose to run the simulation. F_i^{rest} represents other forces we add to make the simulation more realistic. This includes buoyancy force, air dragging force, vorticity confinement force, damping force, and turbulence force. The vorticity and the vorticity confinement force [5] are calculated in the following way,

Vorticity:

$$\boldsymbol{\omega}_i = \sum_j (\boldsymbol{v}_{ij} \times \boldsymbol{r}_{ij}) \nabla W_{ij} \tag{6}$$

Vorticity Confinement Force:

$$\boldsymbol{F}_{vc_i} = \epsilon L_i \sum_j (\nabla |\boldsymbol{\omega}_{ij}| \times \boldsymbol{\omega}_i) \tag{7}$$

where v_{ij} is the velocity difference between particle i and neighour j and r_{ij} is the position difference.

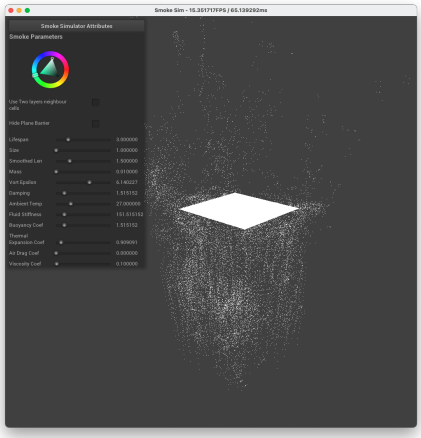

Fig. 1. Particle Collision with Surface 1

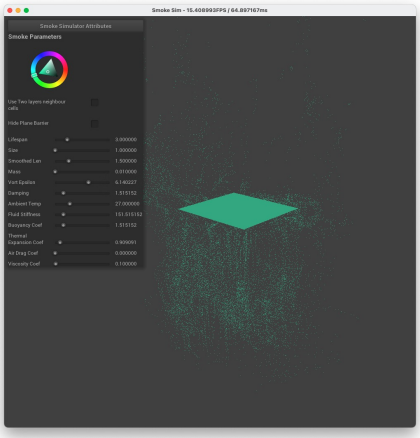

Fig. 2. Particle Collision with Surface 2

3.7 Particle Collision

We first implemented collision detection. When particles hit the surfaces, we adjusted their positions and updated their velocities so that they did not pass through the surfaces. Instead, they diffused on the surface of the barrier, and some of them continued moving to circumvent the barrier before floating upwards.

To improve the performance of collision detection, we used the grid space built for smoke simulation. For each collision surface, we recorded the grids in which they were located (with a small offset in the surface's normal direction). We only tested particle collisions for those particles that entered the grids where the surfaces were located.

We then tested different models for calculating particles' velocities after they hit the surfaces, rather than simply negating their velocities. We experimented with various diffusion algorithms to make the smoke look more realistic. For instance, if a particle hits a face-down surface, one method we are currently using involves first negating the particle's original velocity, reducing its velocity on the y-axis to a small value, and adding a normalized scale to the x-axis and z-axis. This results in smoother smoke flow.

In section 3.1, we outline the simulation workflow. We added the collision detection and implementation after step (3*c*), which corrects the positions and velocities of collided particles before rendering them.

3.8 Rendering

We store all particles in a linked list to enable the efficient adding new particles and removing end-of-life particles. In each frame, we render every particle using the OpenGL GL_POINTS primitives.

In the vertex shader file, we compute the actual screen position of each particle using the global position of the particle as well as the camera setting. The computed position then becomes the final position for render (i.e., the gl_Position). We also set other properties of the rendered points, including the size and color in the shader files using input value from the simulation.

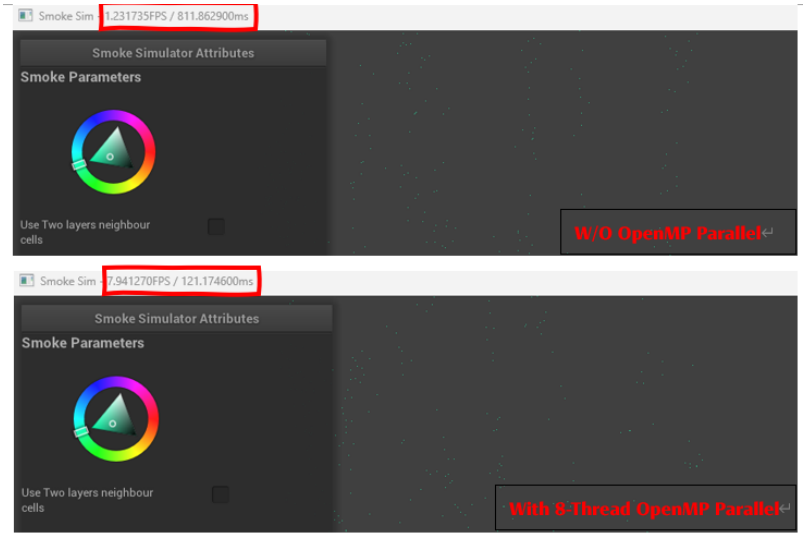

Fig. 3. Serial v.s. Parallel Comparison

4 DISCUSSION

One major issue we encountered in implementing this project is the difficulty in simulating enough particles to resemble real smoke. In the real world, smoke consists of millions of tiny particles, which is not possible to simulate on personal devices. However, the limitation on the number of particles we can simulate implies that the interactions of particles would not be realistic enough. We might underestimate the pressure or viscosity at a given grid.

Another issue is that particle-based simulation is computationally heavy. Setting time steps smaller would make particles flow smoother but at the cost of the slow movement of smoke as viewed in real-time. Setting time steps higher would yield larger error between simulated movement and real world movement of the smoke, as well as larger space gaps of individual particles between timestamps.

Therefore, We believe this is why a grid-based simulation is more popular when it comes to simulating realistic smoke. A grid-based approach computes only the flow of particles (data) between neighbouring grids, eliminating the need to simulate millions of particles and their interactions.

5 RESULTS

5.1 OpenMP Optmization

To improve our real-time performance, we used OpenMP for parallelization by dividing the simulation space into grids and parallelizing the inner-grid particle computations using OpenMP. With 8-thread parallelization *ompparallel*, we are able to run the simulation $8x$ faster than the simulation without parallel.

Here, we render over 13k particles, and we observe the different between the simulation with parallel and w/o parallel. The FPS difference meets our expectation. 3

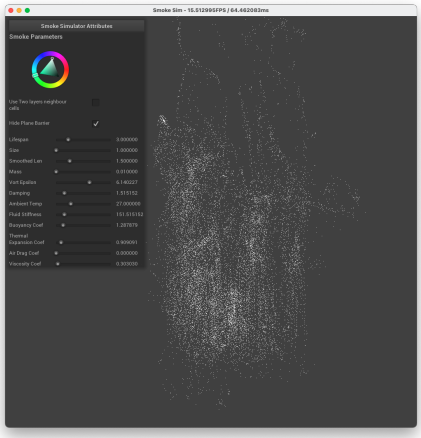

Fig. 4. Particle-based Smoke Simulation 1

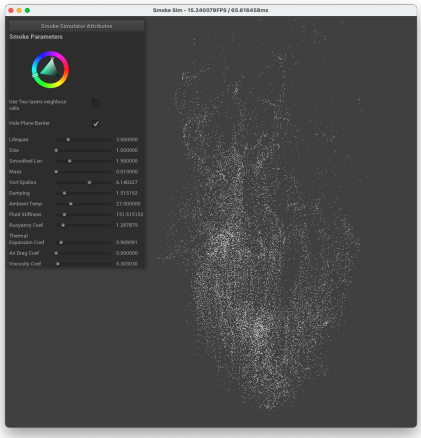

Fig. 5. Particle-based Smoke Simulation 2

5.2 Smoke Simulation Showcase

Here are some screenshots rendered by our smoke simulation application. (See Figure 4) As we can see, we were able to utilize all the techniques we mentioned above to create realistic smoke effects. The smoke appears to flow smoothly and follow natural patterns of dispersion and diffusion. The visualization is enhanced by adding and adjusting properties on the left side, which further add to the realism of the simulation. (See Figure 5) Overall, we are pleased with the quality of the visual results and believe that our simulation is a promising tool for generating realistic smoke effects for various applications.

6 FUTURE WORKS

In future, we have following objectives to achieve in order to make our smoke simulation more realistic and enjoyable. First, we want to simulating different types of smoke, such as the cigarette smoke or the mushroom cloud. Second, we want to simulate smoke collision with surfaces made of different materials. For example, we can change the roughness of the surface and observe how the smoke diffuses differently. Third, even though we achieve real-time simulation speed on Mac book, we still want to have a better real-time rendering speed even with more particles or complex collision calculations.

7 CONTRIBUTION

Zhengjiang He: Basic NanoGUI and OpenGL framework, various bug fixes, and optimizations, report writing
Hirotaka Ishihara: Simulation workflow, smooth kernel, Navier-stokes solver implementation, report writing
Chaomin Li: Particle collision and diffusion, some bug fixes, OpenMP parallelism, report writing
Zhiqi Yan: Testing and debugging, writing code for particle collision, report writing

REFERENCES

[1] 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 (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 15–22. https://doi.org/10.1145/383259.383260

[2] R. A. Gingold and J. J. Monaghan. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 181, 3 (12 1977), 375–389. https://doi.org/10.1093/mnras/181.3.375 arXiv:https://academic.oup.com/mnras/article-pdf/181/3/375/3104055/mnras181-0375.pdf

[3] Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

[4] J. J. Monaghan and J. C. Lattanzio. 1985. A refined particle method for astrophysical problems. *Astronomy and Astrophysics* 149 (1985), 135–143.

[5] John Steinhoff and David Underhill. 1994. Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids* 6, 8 (1994), 2738–2744.