Grid based 2D smoke simulation

HAN QI, University of California at Berkeley, USA
YUHAN YANG, University of California at Berkeley, USA
RUINAN XU, University of California at Berkeley, USA
ANTHONY LING, University of California at Berkeley, USA

In this project, we implemented an interactive 2D smoke simulation with multiple novel features. Our approach is mainly based on the famous Navier-Stokes equations, and we utilized the data structure called MAC Grid to simulate smoke in a 2D grid system. In addition to regular smoke dynamics simulations, we also simulated the collision between the smoke and three types of sphere: absorptive, reflective and attractive sphere. Besides, we pre-defined two simulation scenes to enrich our simulation.

CCS Concepts: • **Computer graphics**;

Additional Key Words and Phrases: smoke simulator, Navier-Stokes equations, MAC Grid

ACM Reference Format:
Han Qi, Yuhan Yang, Ruinan Xu, and Anthony Ling. 2021. Grid based 2D smoke simulation. 1, 1 (May 2021), 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Smoke is one type of fluid and can be used for simulating collections of airborne solids, liquid particulates and gases, such as those that make up smoke. A smoke simulator simulates the smoke movement and generates animated pixel textures representing the density, heat, and velocity of smoke particles which can be used for rendering.

In this project, we implemented an interactive 2D smoke simulation with multiple novel features. The simulation is mainly based on the famous Navier-Stokes equations. We used MAC Grid (Maker-and-Cell Method) data structure to store all the physical properties related to smoke dynamics. The code of this project was written in C++ and was built on top of an existing project [3]. We created an interactive interface in which users can create sources simply by mouse clicking. Besides that, users can apply external forces by dragging the mouse on the screen. We also created two modes which specifies the environment of simulated smoke. As a novelty of our project, we simulated the interaction between the smoke and other objects, such as a sphere. We simulated the collision between the smoke and three types of sphere: absorptive, attractive and reflective. These features enriches our simulator and makes it more realistic.

Authors' addresses: Han Qi, University of California at Berkeley, USA, han2019@berkeley.edu; Yuhan Yang, University of California at Berkeley, USA, yuhany@berkeley.edu; Ruinan Xu, University of California at Berkeley, USA, ruinanxu@berkeley.edu; Anthony Ling, University of California at Berkeley, USA.

2 APPROACH

2.1 Navier-Stokes equations

Smoke is a type of incompressible fluid whose behaviors can be described and predicted by Navier-Stokes equations. The equations are[1]:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{F} + \nu \nabla \cdot \nabla \vec{u} \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

The letter $\vec{u}$ is the velocity of the fluid which has two components in our 2D simulation. ρ is the density of the fluid. p is the pressure. $\vec{F}$ is the external force applied to the fluid. ν is kinematic viscosity, measuring how much the fluid resists deforming while it flows. This equation is nothing but the Newton's second law, conservation of momenutum, $\vec{F} = m\vec{a}$. It basically describes how the fluid flows when internal and external forces are applied on it.

2.2 MAC Grid

Typically, there are two ways to track the movement of a flow, Langrangian and Eulerian. In our simulation, we chose the Eulerian view-points in which fluids are simulated in a grid system, rather than a particle system. Particularly, we used MAC Grid [2] structure to store the scalar and vector fields related to fluids. We discretized the 2D spatial space where the different variables were stored at different locations. All scalar fields, such as pressure, density and temperature, were stored in the center of each cell. Vector field, i.e. the velocity $\vec{u}$, were stored at faces. Specifically, The horizontal u component is sampled at the centers of the vertical cell faces. The vertical $\vec{v}$ component is sampled at the centers of the horizontal cell faces. This data structure benefits us in accurately estimating the derivative of velocity field at the center of cells, see 1.

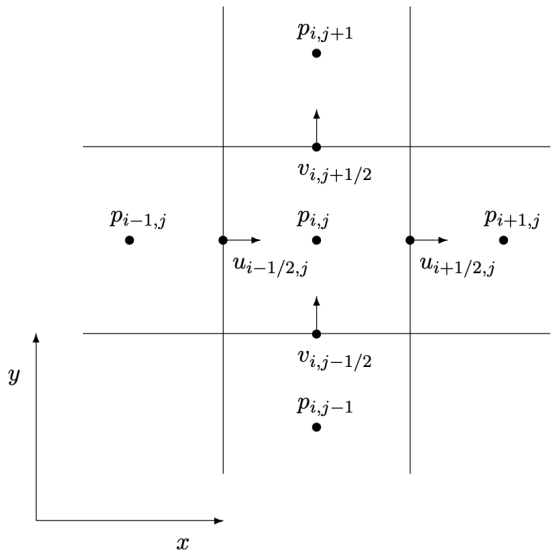

Fig. 1. MAC grid data structure illustration

2.3 Simulation

In our simulation, we firstly track the user input to initialize the density/temperature/velocity field. The user can specify the source of smoke by clicking the mouse on the screen. On a click, the program adds temperature, density and velocity to the cells around the mouse position.

Next, we advect and update the velocity. We use Langrangian method to do an Eulerian calculation (so-called semi-Langrangian advection). We go through each cell in the grid and traceback to fetch the velocity that will be at our current location next step $\vec{u}_P^{n+1}$. It should be equal to the velocity at the old location at the current step $\vec{u}_G^n$. In other words, $\vec{u}_P^{n+1} = \vec{u}_G^n$. The current location $\vec{x}_P$ and the old location $\vec{x}_G$ have the relationship $\vec{x}_G = \vec{x}_P - dt * \vec{u}_P$.

Then we add external forces to the smoke to drive the smoke motion. The external forces contains three portions:

(1) We allow users to input an external force by dragging the mouse along a specific direction.

(2) Thermal Buoyancy force which drives the smoke flowing upwards caused by fluid pressure.

$$F_{buoy} = (\alpha d - \beta(T - T_{amb})) * g \tag{3}$$

where g is the gravity. The constant α and β were properly chose. T is the temperature at the current cell. T_{amb} is the average temperature of the fluid grid. Thie Buoyancy force is along y direction to push the smoke moving upwards.

(3) Vorticity force which is used to capture the swirly motion characteristic of turbulence. The vorticity force can be calculated as follows:

$$F_{vort} = \vec{N} \times \vec{\omega} \tag{4}$$

where $\vec{\omega} = \nabla \times \vec{u}$ and $\vec{u}$ is the velocity. $\vec{N}$ is the vector pointing to the vortex center $\vec{N} = \frac{\nabla|\vec{w}|}{|\nabla|\vec{w}||}$

Once we computed all forces, we can update the velocity using Newton's second law:

$$Force = mass * acceleration \tag{5}$$

$$\vec{u}_{n+1} = \vec{u}_n + dt * acceleration \tag{6}$$

Next, we projected our velocity field onto a divergence-free space to satisfy eq (2). Specifically, we compute and update pressure so that when we apply them to the field the divergence-free property still holds. The values of THE fluid at the next time step can be described in terms of our current (divergent, unstable) velocity field and the pressure:

$$\vec{u}_{n+1} = \vec{u}_* - dt \cdot \frac{1}{\rho} \nabla p \tag{7}$$

The divergence-free velocity field must satisfy

$$\nabla \cdot \vec{u} = \frac{du}{dx} + \frac{dv}{dy} = 0 \tag{8}$$

We can get the pressure with the above two equations.

Lastly, we advect the temperature and density just as what we did for velocity advection.

2.4 Collision with the sphere

1. Totally reflecting sphere. We imitate the collision of the smoke with a totally reflecting sphere. In the function of advectRenderingParticles, we will tell whether the particle's next position is inside the sphere or not. If its next position is inside the sphere, we calculated a correction position by pushing it back to the surface of the sphere along the line of sphere center and the old next position. In the function of getVelocity, we recalculated the velocity of the particle if it

Table 1. Pseudo code

Smoke simulation pseudo code
1. **for** each frame **do**: 2. get smoke source from user. Initialize velocity, density and temperature. 3. Compute the velocity: Advect the velocity. Compute the external force Projection. 4. Advect the temperature and density.

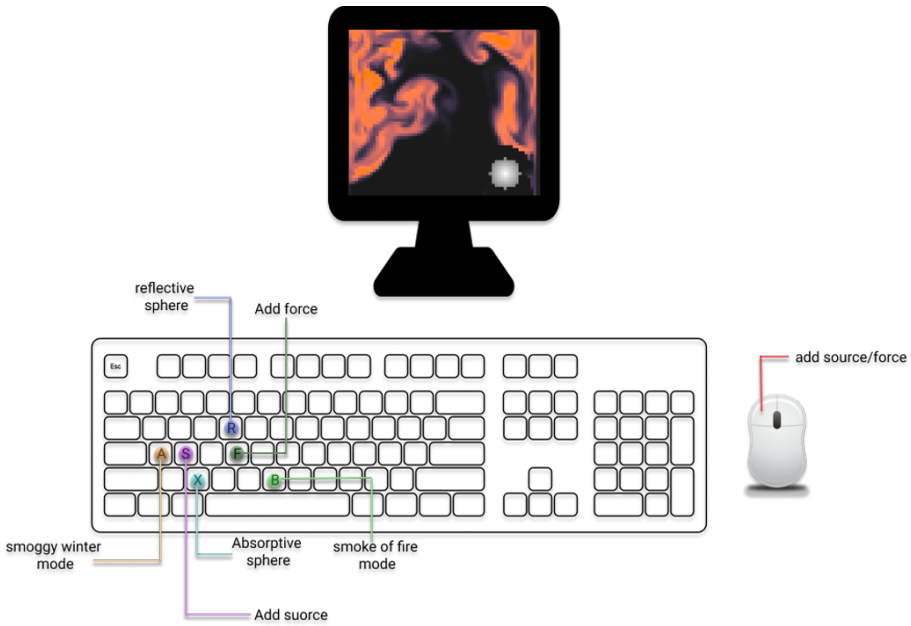

Fig. 2. The schematic of user interface

inside or on the sphere. We make the new velocity to be tangent to the sphere surface. 2. Totally absorptive sphere. We imitate the collision of the smoke with a totally absorptive sphere. This means that all particles will flow in the air but when it hits the sphere, the particle will be absorbed. 3. Attractive sphere. It can attract the smoke and absorb the smoke. We add an external force applied by the sphere on smoke particles. The force imitates Newton's law of universal gravitation. This makes the sphere attract all the smoke flowing in the air and then absorb all the smoke.

3 IMPLEMENTATION

We created a simulation of smoke based on physical properties like velocity, temperature and density.

Firstly, in our simulation, we enable users to create multiple sources with mouse clicking on the screen. This gives users the flexibility of simulation and they can simulate the flowing smoke starting from anywhere on the screen. Also,

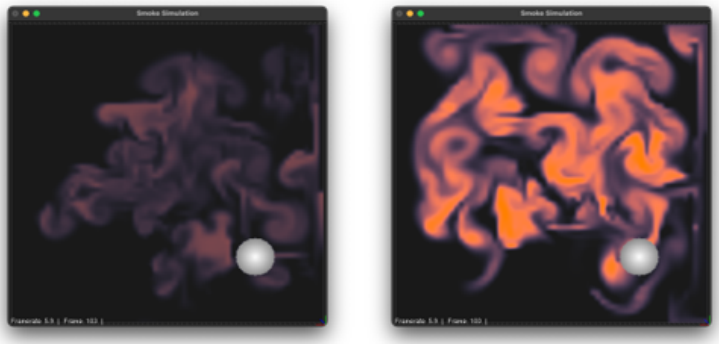

Fig. 3. Both cold and hot temperature smoke examples

based on this function, one improvement possible for the future is that we can simulate the collision of two flow of smokes.

Secondly, we add the features that users can drag on the screen to add external forces to smoke and control the direction of the forces. This feature provides the ability to simulate the smoke in the real world. For example, the users can add the force of wind to the smoke and see how the smoke flows under the external forces.

Thirdly, we create two mode simulating the smoke movement in two different situations: smoggy winter morning and smoke of fire in a small room. In a winter morning, the temperature of smoke is low and usually the smog is not very dense. So we set low temperature value and low density value for the smoke in this situation. However, for the smoke of fire in a small room, the temperature is high and the smoke is very dense since the space is limited. So we set high temperature value and high density value for the smoke in this situation.

Finally, we make the simulation of the collision between the smoke and a sphere. When the smoke hits the sphere, it cannot enter the sphere since the sphere is solid. Also, its velocity will be redirected because of the collision. So the effect is that the smoke will flow around the sphere, which is in the space of simulation. We can set the mode with absorptive sphere. So if the smoke hit the sphere, the smoke will be absorptive. We can also the set the mode with attractive sphere. Besides the attractive sphere can absorb the smoke, it can also attract the smoke in the air.

In conclusion, we make a simulation of smoke based on physical properties and formulas. This simulation is a simplified but realistic imitation about how the smoke would flow in real world. We add different functions to diversify the simulation scenes, like different temperatures and collision with a sphere. We also designed a user interface which allows the user to interact with the smoke and change the simulation modes by mouse and keyboard control.

4 RESULTS

The GitHub repository link is here: https://github.com/yuhany1024/cs184-final-project. In this section, we present some simulation effects that we implemented, see Figure 4.

- Users can add source or forces on the screen by clicking and dragging the mouse.
- Users can simulate smoggy whether in colder winter.
- Users can simulate smoke of fire.
- Users can simulate the collision between smoke and reflective sphere.

Fig. 4. Examples of the smoke simulation effect

- Users can simulate the collision between smoke and absorptive sphere.
- Users ca simulate the collision between smoke and attractive sphere.

5 DISCUSSION

Using the Navier-Stokes equations, we were able to produce an accurate smoke simulator with smoke-object interaction and with different smoke temperatures. Our current implementation makes use of GLUT's GUI and utilizes its 2D drawing, as we place an emphasis on only displaying 2D simulations. We added multiple novel features on top of the regular smoke simulator.

First, users are allowed to control the direction of the external forces applied to the smoke by dragging the mouse on the screen. Compared to other smoke simulations which can only apply constant forces such as gravitational force, our implementation greatly enriches the scenes that our simulations can be applied to.

Second, we simulated the interaction of the smoke with other objects. Although we only simulated the collision between smokes and spheres in this project, our approach can be also utilized to more complex objects to make the scene more realistic.

Third, we defined three types of sphere based on their interactions with the smoke. Absorptive sphere absorbs any smoke particles that hit on its surface. Reflective sphere changes the moving direction of the smoke. Attractive sphere is just like a black hole that attracts all smoke particles.

6 FUTURE WORK

One limitation of our simulation is that it is computed on CPU which has limited speed compared to using a GPU's powerful shading program. In order to render the smoke in real-time, we have to sacrifice the rendering resolution to speed up the computation. One potential future work improvement is by changing the current implementation to utilize fragment shaders.

Another possible future work implementation is the implementation of dynamic object interaction. If the circumstances were different (i.e. debugging, more time), we would've attempted to implement dynamic objects, such as a moving sphere or a pinwheel rotation.

7 CONTRIBUTIONS

(1) **Han Qi**: implemented the collision between a sphere and the smoke, implemented two simulation modes: smoggy winter morning and smoke of fire, implemented reflective sphere and absorptive sphere, wrote the result part in the paper, make the video.
(2) **Yuhan Yang**: implemented user interactive input, allow users to add smoke source and change the direction of the external force in the interface, display the sphere, wrote the approach and discussion part in in the paper, make the slides.
(3) **Ruinan Xu**: wrote the abstraction and introduction part in the paper.
(4) **Anthony Ling**: worked on writeup and the Future Work section, as well as converting the previous implementation from 3D space into 2D space and optimization.

REFERENCES

[1] Robert Bridson and Matthias Muller-Fischer. 2007. Fluid Simulation, SIGGRAPH 2007 Course Notes. https://www.cs.ubc.ca/~rbridson/fluidsimulation/fluids_notes.pdf
[2] N. Foster and D. Metaxas. 1997. Modeling the motion of a hot, turbulent gas.
[3] Jon Lee. 2018. Fluid Simulation. https://github.com/AgentLee/FluidSim