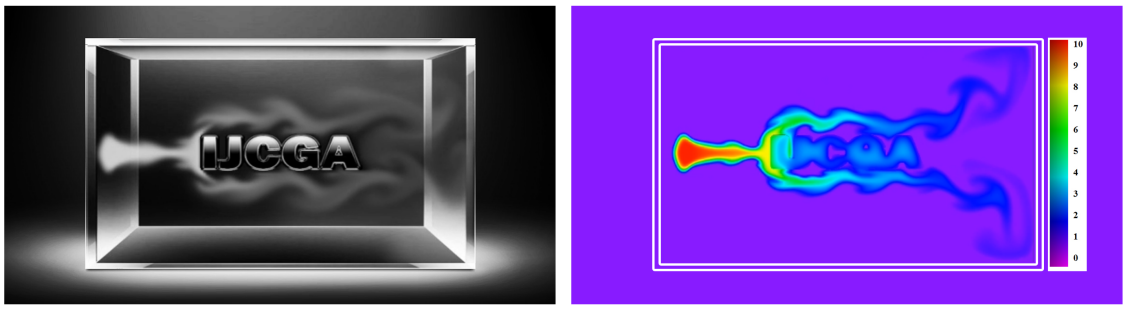
Fast Fluid Thermodynamics Simulation By Solving Heat Diffusion Equation

Wanwan Li

George Mason University



(a) Real-time Fluid Rendering (b) Fluid Heat Map Visualization

Figure 1: A demo of GPU-accelerated fluid thermodynamics simulation which is solved by Navier-Stokes equations coupled with Heat Diffusion Equation: (a) Photo-realistic fluid rendering. (b) Fluid heat map visualization. Temperature is varying between 0°C(purple) and 10°C(red).

Abstract

In mechanical engineering educations, simulating fluid thermodynamics is rather helpful for students to understand the fluid's natural behaviors. However, rendering both high-quality and real-time simulations for fluid dynamics are rather challenging tasks due to their intensive computations. So, in order to speed up the simulations, we have taken advantage of GPU acceleration techniques to simulate interactive fluid thermodynamics in real-time. In this paper, we present an elegant, basic, but practical OpenGL/SL framework for fluid simulation with a heat map rendering. By solving Navier-Stokes equations coupled with the heat diffusion equation, we validate our framework through some real-case studies of the smoke-like fluid rendering such as their interactions with moving obstacles and their heat diffusion effects. As shown in Fig. 1, a group of experimental results demonstrates that our GPU-accelerated solver of Navier-Stokes equations with heat transfer could give the observers impressive real-time and realistic rendering results.

Keywords

Fluid Simulation; Navier-Stokes Equations; OpenGL/SL; Thermodynamics;

1. Introduction

As a very useful tool to simulate the behavior of fluid dynamics through computers, fluid simulation technologies have several important applications range from the extremely time-consuming high-quality animations for film, to simple real-time particle systems[1][2][3], from the Laplacian Eigenfunctions based simulation[4] to Fourier synthesis of water surface wave[5] used in the games. However, open-source codes for the fluid simulation via solving Navier-Stokes equations[6] such as OpenFOAM[7], LifeV[8], FEAT-FLOW[**?**] are based on the CPU computation processors which are very slow to deal with the computation-intensive task. Under the need for speeding up the fluid simulation, GPU-accelerated real-time fluid rendering has already been applied for 3D interactive simulation[9][10]. For

DOI: 10.5121/ijcga.2021.11401

comparison, CPU-based fluid simulation takes additional time to render every time step and then compile all the results into a video file. For high spatial and temporal resolution simulations, this complex process will definitely slow down its process and speed. In order to increase the simulation efficiency, one powerful method is to use the GPU devices to simulate the fluid dynamics (and corresponding heat and mass transfer therein) using multi-threads parallel programming technology.

GPU-based fluid simulation has been well studied in the computer graphics area. For example, Harris et al.[11] has previously described the basic concepts, programming methodology, and numerical algorithms about how to simulate the dynamic fluid flow-field using the texture which is a special data structure used for the GPU devices and using the shader language which is used to instruct the GPU about how to render each pixel on the screen. However, the GPU-based thermal-fluid simulation has not yet reached its full potential been widely used in the heat and mass transfer community due to several conceptual challenges with GPU programming and the limited availability of the open-source GPU code for specific simulations of a complex physical phenomenon – such as multi-component species diffusion, coupled heat and mass transport, fluid-obstacle interactions, boiling, condensation, evaporation, fluid droplet formation, break-up, and contact-line motion on solid surfaces coupled with solid-fluid and fluid-air interface forces[12][13].

Among those recent works, Nuli et al. [14] used CUDA-enabled GPU to speed up the Sph-based fluid animation. Clausen et al. [15] proposed a Lagrangian finite element method to simulates the interactions between liquids and solids. De et al. [16] introduced a power diagrams-based approach for incompressible fluid simulations. Considering the particle-strength exchange phenomenon, Zhang et al. [17] resolved the fluid boundary layers issues in fluid simulations. Koster et al. [18] improved the performance of real-time fluid simulations with an adaptive position-based method. Chu et al. [19] has used the CNN-based feature descriptors to implement an efficient data-driven approach for smoke flow simulations. Akbay et al. [20] presented a novel extended partitioned method for two-way solid-fluid coupling. Nagasawa et al. [21] proposed a nonlinear blending model that can simulate shear-thinning fluids. Though constraint-based bubbles and affine fluid regions, Goldade et al. [22] proposed a novel model for efficient immersed bubbles and flexible spatial coarsening fluid simulation approach. Fang et al. [23] proposed an interface called IQ-MPM to apply a quadrature material point method for simulating non-sticky strongly two-way coupled nonlinear solids and fluids interactions. Yang et al. [24] proposed a solver of Clebsch wave functions for gauge fluid simulations. Xiong et al. [25] proposed a vortex segment cloud-based approach for incompressible flow simulation. Ruan et al. [26] proposed a novel method to model the contact interaction between solid and fluid where exists strong surface tension. Wang et al. [27] devised an approach for thin-film smoothed particle hydrodynamics fluid simulations.

Our work extends the GPU fluid simulation implemented by Philip Rideout et al.[28] by adding the interaction between fluid and moving obstacle and adding the diffusion property of the smoke-like fluid. So at here, in order to demonstrate a more practical open-source framework for GPU accelerated fluid simulations with well-explained implementation details, and a couple of numerical simulations are implemented and discussed, we also give detailed algorithms, proof, and mathematical models used for a specific GPU simulation such as the two-phase smoke-obstacle rendering. To support advances in both the computer graphics community and the heat mass transfer community, we also provide some supplemental materials including our source code, implementation instructions, and videos of the simulation results which can be accessed through this link: `https://youtu.be/zfczUyBE3ds`.

2. Numerical Methods

To simulate the heat and mass transport of an incompressible fluid, an implicit volume of fluid (VOF) GPU model is used. A fluid is incompressible if the volume of any sub-region is constant over time. Specifically, we simulate the spatial and temporal dynamics of smoke in the air on a regular Cartesian grid with the 2D spatial coordinates $\mathbf{r} = (x, y)$. The fluid is represented by its velocity field $\mathbf{u}(\mathbf{r}, t)$ and a scalar pressure field $p(\mathbf{r}, t)$. If the velocity and pressure are known at some initial time $t = 0$, then the velocity of the fluid changing over time can be described by the Navier-Stokes equations[10]:

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

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

where ρ is the fluid density, ν is the kinematic viscosity, and $\mathbf{F}$ represents any external forces per unit volume that act on the fluid. Equations of the form $\nabla^2 x = b$ are known as Poisson equations. The case where $b = 0$ is Laplace's equation, which is the origin of the Laplacian operator.

2.1. Advection Equation

Advection is the process by which a fluid's velocity transports itself or other fluid properties $\mathbf{q}(\mathbf{r}, t)$ such as its concentration, temperature, or any quantity carried by the fluid at any position $\mathbf{r}$ and time t in the fluid. This property is described as the first term in Eq. 1 which is shown in Eq. 3:

$$\frac{\partial \mathbf{q}(\mathbf{r}, t)}{\partial t} = -\mathbf{u}(\mathbf{r}, t) \cdot \nabla \mathbf{q}(\mathbf{r}, t) \tag{3}$$

However, in the multi-threads-based parallel GPU computation, it is not a good idea to simply move the position $\mathbf{r}$ of each pixel forward along the velocity field at the corresponding distance that pixel would travel in Δt time by $\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{u}(\mathbf{r}, t)\Delta t$. The reason is the locations where the GPU's multi-threads are writing cannot be moved. The solution is to invert the problem and use an implicit method[29] to advect quantities by copying a pixel from the previous time step. According to the advection equation: $q(\mathbf{r} + \mathbf{u}(\mathbf{r}, \Delta t), t + \Delta t) = q(\mathbf{r}, t)$, we can trace the trajectory of the pixel from each grid cell back in time to its former position, and we copy the quantities at that position to the new grid cell. To update a quantity q, we use Eq. 4:

$$q(\mathbf{r}, t + \Delta t) = q(\mathbf{r} - \mathbf{u}(\mathbf{r}, t)\Delta t, t) \tag{4}$$

In order to prove the correctness of Eq. 4, we need to assume that the Δt is sufficiently small to allow the Taylor expansion on both sides of Eq. 4, then it gives Eq. 5:

$$q(\mathbf{r}, t) + \frac{\partial q(\mathbf{r}, t)}{\partial t}\Delta t = q(\mathbf{r}, t) - \frac{\partial q(\mathbf{r}, t)}{\partial \mathbf{r}}\mathbf{u}(\mathbf{r}, t)\Delta t \tag{5}$$

We assume that any 2D position $\mathbf{r}$ can be described as a 2D coordinate (x, y), then by subtracting $q(\mathbf{r}, t)$ and dividing Δt on both sides of Eq. 5 yielding:

$$\frac{\partial q(x, y, t)}{\partial t} = -\mathbf{u}(x, y, t)\left(\frac{\partial q(x, y, t)}{\partial x} + \frac{\partial q(x, y, t)}{\partial y}\right) \tag{6}$$

Finally, we introduce the divergence operator into the Eq. 6 to get:

$$\frac{\partial q(x, y, t)}{\partial t} = -\mathbf{u}(x, y, t) \cdot \nabla q(x, y, t) \tag{7}$$

From the proof shown here, we could prove that Eq. 7 is equivalent to Eq. 3, so the implicit method can give us a vivid animation of fluid as it leads to the same result to the methods that directly advecting quantities by computing where a particle is moving towards over the next time step.

2.2. Diffusion Equation

In our GPU-based real-time fluid rendering, heat and mass diffusion are important transport mechanisms to account for. Due to the molecular interaction caused by the molecular heat motion, there is the diffusion phenomenon for both its temperature T and concentration C. The heat and mass diffusion property can be expressed by heat diffusion equation:

$$\frac{\partial \Psi(\mathbf{r},t)}{\partial t} + \mathbf{u}(\mathbf{r},t)\cdot\nabla\Psi(\mathbf{r},t) = \nabla\cdot(D_\Psi(\mathbf{r},t)\nabla\cdot\Psi(\mathbf{r},t)) \tag{8}$$

Assume that the diffusion coefficient $D_\Psi(\mathbf{r},t)$ does not change over time and space, For simplicity, hence it can be replaced into a constant D_Ψ, then Eq. 8 is in this form:

$$\frac{\partial \Psi(\mathbf{r},t)}{\partial t} + \mathbf{u}(\mathbf{r},t)\cdot\nabla\Psi(\mathbf{r},t) = D_\Psi\nabla^2\Psi(\mathbf{r},t) \tag{9}$$

For the mass diffusion equation of fluid, we can simply replace q into C. But for the heat diffusion of fluid, by replacing q into into T and adding the heat transfer term, we would get the famous heat equation:

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = D_T\nabla^2 T + \kappa(\dot{T_S} - T) \tag{10}$$

Where $\dot{T_S}$ is the temperature of the surface on the obstacle, and k is the thermal conductivity of the fluid. At the same time, this equation equally applies to the obstacle temperature. Eq. 10 can be decomposed into three terms:

(1) Advection Term:

$$\frac{\partial T'}{\partial t} = -\mathbf{u}\cdot\nabla T' \tag{11}$$

(2) Diffusion Term:

$$\frac{\partial T''}{\partial t} = D_T\left(\frac{\partial^2 T''}{\partial x^2} + \frac{\partial^2 T''}{\partial y^2}\right) \tag{12}$$

(3) Heat Transfer Term:

$$\nabla T_{\mathbf{r}} = \frac{1}{N}\sum_{\forall \mathbf{d}\in D_{\mathbf{r}}} \dot{T_{\mathbf{d}}} - T_{\mathbf{r}} \tag{13}$$

Where $D_{\mathbf{r}}$ is a 2D domain both near the position $\mathbf{r}$ and within the obstacle, N is the number of pixels on such domain. Then we can get the final heat equation as $T = T' + T'' + k\nabla T_{\mathbf{r}}$ So according to this, the GPU-based solver for the diffusion equation is shown below:

$$T(\mathbf{r}, t+\Delta t) = T(\mathbf{r} - \mathbf{u}(\mathbf{r},t)\Delta t, t) + D_T\nabla^2 T(\mathbf{r},t)\Delta t + k\nabla T_{\mathbf{r}}(\mathbf{r},t) \tag{14}$$

Where the Laplacian operator $\nabla^2 T(\mathbf{r},t)$ in GPU solver can be expressed as:

$$\nabla^2 T_{x,y} = \lim_{\Delta x,\Delta y\to 0}\left(\frac{T_{x-\Delta x,y} - 2T_{x,y} + T_{x+\Delta x,y}}{\Delta x^2}\right) + \left(\frac{T_{x,y-\Delta y} - 2T_{x,y} + T_{y,y+\Delta y}}{\Delta y^2}\right) \tag{15}$$

Diffusion term actually smoothes the value of concentration or temperature of the smoke among the space.

2.3. Poisson Equation

Let domain D be the region in which the fluid is moving, then there is the Helmholtz - Hodge Decomposition Theorem[30]: Any vector field $\mathbf{W}$ on a domain D can be uniquely decomposed in the form described by equation:$\mathbf{w} = \mathbf{u}+\nabla p$. Where $\mathbf{u}$ has zero divergences. The continuity equation 2 requires that for each time step fluid simulation is constrained with a divergence-free velocity. According to the Helmholtz-Hodge Decomposition Theorem, divergence of the velocity can be corrected by subtracting the gradient of pressure field: $\mathbf{u} = \mathbf{w} - \nabla p$. When we apply the divergence operator to both sides of the above equation, we obtain: $\nabla \cdot \mathbf{u} = \nabla \cdot (\mathbf{w} - \nabla p)$, so that we have $\nabla \cdot \mathbf{u} = \nabla \cdot \mathbf{w} - \nabla^2 p$. But since Eq. 2 enforces that $\nabla \cdot \mathbf{u} = 0$, then this simplifies to the Poisson-pressure equation:

$$\nabla^2 p = \nabla \cdot \mathbf{w} \tag{16}$$

In our fluid simulation, we need to solve the Poisson-pressure equation. Poisson equations are common in physics and well understood in numerical analysis[31]. We use an iterative solution technique called Jacobi iteration[32] that starts with an approximate solution such as a matrix of zeros and improves it through every iteration with the form:

$$\nabla^2 p_{x,y}^{t+\Delta t} = \lim_{\Delta x, \Delta y \to 0} \frac{1}{4} \left(p_{x-\Delta x,y}^{t} + p_{x+\Delta x,y}^{t} + p_{x,y-\Delta y}^{t} + p_{x,y+\Delta y}^{t} - \Delta x \Delta y \nabla \cdot \mathbf{w}_{x,y} \right) \tag{17}$$

3. Implementation

3.1. Smoke Physics

Smoke is a collection of airborne solid and liquid particulates and gases emitted when a material undergoes combustion [33]. According to this definition, the smoke itself is not fluid but a particle system. The reason that why smoke propagates like fluid is the smoke particles are advected by a translucent media, that is, the airflow, at the same time diffused into the air. So in order to simulate the behavior of smoke is actually to consider the fluid property of air plus the diffusion property of smoke.

Recall the process of the generation of smoke: due to combustion, heat is generated and diffused into the air which causes the particular region of air to rise up due to the heat buoyancy. At the same time, the burned tiny solid particles are advected by the uprising air and cause the distribution of smoke changes among the space like the flow of fluid. So the basic thermodynamics in this process is the combined external force of buoyancy, caused by the heat source, and gravity together with the heat diffusion.

3.2. External Force

Apply external force $\mathbf{F}$ such as buoyancy and gravity. For smoke simulation, temperature T can influence velocity by making it rise. In our implementation, we also apply the weight of the smoke in this stage; high concentration C in cool regions will sink. The velocity $\mathbf{u}$ of air caused by heat buoyancy B, density of smoke D_{smoke} and density of air D_{air} at region $\mathbf{r}(x, y)$ is:

$$\mathbf{F} = (C(\mathbf{r}, t)D_{\text{smoke}} + (1 - C(\mathbf{r}, t))D_{\text{air}} + T(\mathbf{r}, t)B)\mathbf{y} \tag{18}$$

This external forces model consists of buoyancy and gravity that can affect the velocity of smoke only along the $\mathbf{y} = (0, 1)$ direction.

3.3. Fluid-Obstacle Interactions

To account for the fluid simulation of a solid-fluid interface, in this part we will take a close look at the mechanism of the interaction between the smoke and moving solid object.

In our experiments, we try different motion equations for the moving obstacle. For implementation on GPU, we are using three different texture buffers to store the information of obstacle's mass, obstacle's velocity, and obstacle's displacement. We send this information as framebuffers into the GPU for high-performance computation. However, in order to expand this obstacle-fluid simulation to the two-phase fluid simulation easily, we use the advection-based model to render the moving object on the texture space. Here for any solid object, there is the advection equation of the phase function of obstacle ϕ_s:

$$\frac{\partial \phi_s}{\partial t} = -\mathbf{u} \cdot \nabla \phi_s \tag{19}$$

And the phase function Φs is defined as a property to describe the distribution of the obstacle, when Φs is equal to 1, it means this position is fully filled with an obstacle. However, due to the fact that GPU texture has discrete texels, or say, grids. So it loses the precision after a large number of iterations according to the advection calculations, this causes the blurring of the edge of the obstacle. So we propose a constrained advection model applied during the advection phase, which is:

$$\phi_s(\mathbf{r}, t + \Delta t) = \begin{cases} 1 & \phi_s(\mathbf{r} - \mathbf{u}\Delta t, t) > 1 - \phi_0 \\ 0 & \phi_s(\mathbf{r} - \mathbf{u}\Delta t, t) \leq \phi_0 \end{cases} \tag{20}$$

Where ϕ_0 is a pre-defined number, in our implementation, we empirically set $\phi_0 = 0.1$ which yields some pretty good effects even under the effects of the loss of mass during the advection.

The correction of velocity is to subtract the new velocity by the gradient of pressure solved from the Poisson equation. However, when we consider the interaction between fluid and obstacle, as the obstacle pushes the fluid around itself moving forward together, we force the velocity of fluid $\mathbf{u}$ equal to the velocity of obstacle $\mathbf{u}_s$ where the obstacle phase function equals to one. The corrected velocity can be calculated as:

$$\mathbf{u}(\mathbf{r}, t + \Delta t) = \mathbf{u}_s(\mathbf{r}, t) \mid_{\phi_s(\mathbf{r},t)=1} \tag{21}$$

From experiments, we have found that our proposed model helps us get rid of applying the "no-slip boundary condition"[34] by directly forcing the advected properties q into 0 where the obstacle phase function ϕ_s is equal to one:

$$q(\mathbf{r}, t + \Delta t) = \begin{cases} q(\mathbf{r} - \mathbf{u}\Delta t, t) & \phi_s(\mathbf{r}, t) = 0 \\ 0 & \phi_s(\mathbf{r}, t) = 1 \end{cases} \tag{22}$$

3.4. Simulation Steps

As discussed in the above section, our GPU simulation methods follow the idea of the Helmholtz-Hodge decomposition approach. We decompose our uniform grid into sub-blocks of similar size. In a simulation, each parallel process stores the simulation data of its own sub-block into the texture rendered as a buffer for GPU computation. For rendering operations, we add several layers or surfaces into one slab and let these two "play ping-pong" with each other to read and write the buffer. Each step of our algorithm is shown in Fig. 2. This process is repeated for every time step, so this figure plots a single time step.

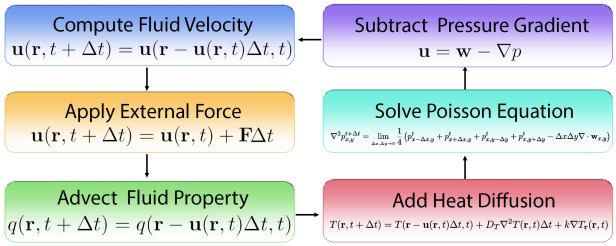


Figure 2: Fluid Simulation Steps.

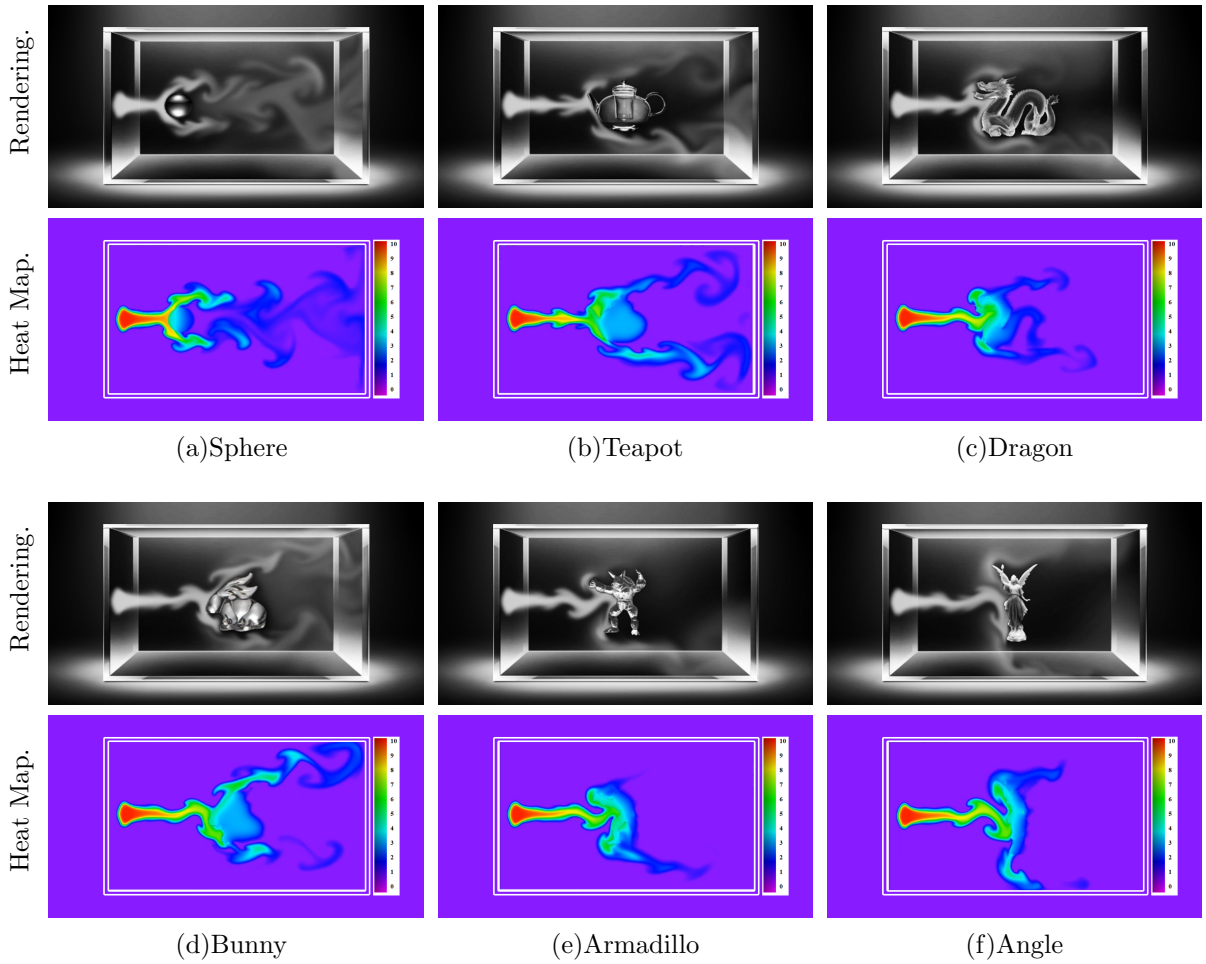


Figure 3: Experiments Results of GPU-Accelerated Smoke Thermodynamics Simulation.

The variables $\mathbf{u}$ and p hold the velocity and pressure field data. In our implementation on OpenGL/SL platform which is widely used for high-performance rendering[35]. We have implemented several different shader functions for each rendering phase.

4. Results And Discussion

In order to demonstrate the properties of fluid through simulations, we have tried the smoke rendering which has diffusion properties making itself spreading and propagating in another media fluid, say, air. At the same time, we consider the interaction between smoke and a moving solid obstacle. As shown in Fig. 3, we have simulated the interaction between the obstacle and smoke on GPUs. In the sphere example(a), the motion equation for the osculating sphere is using the spring oscillation model interpreted by the famous Hook's law which is shown in Eq. 23:

$$\frac{\partial \mathbf{v}(\mathbf{r},t)}{\partial t} = -\frac{k}{M}\int_0^t \mathbf{v}(\mathbf{r},t)\mathrm{d}t \tag{23}$$

Where the initial velocity of the moving sphere is $\mathbf{v} = (-100.0, 0)$, elastic coefficient $k = 5$, and mass $M = 10$. During the rendering of each frame, we use the blending function to mix up three different texture frame buffers: (i) Background image (a glass box), (ii) Obstacle (a glass sphere), and (iii) Fluid Density Image (the smoke). All graphics contents are rendered by GPU shader language OpenGL/SL. In order to validate our code on the case

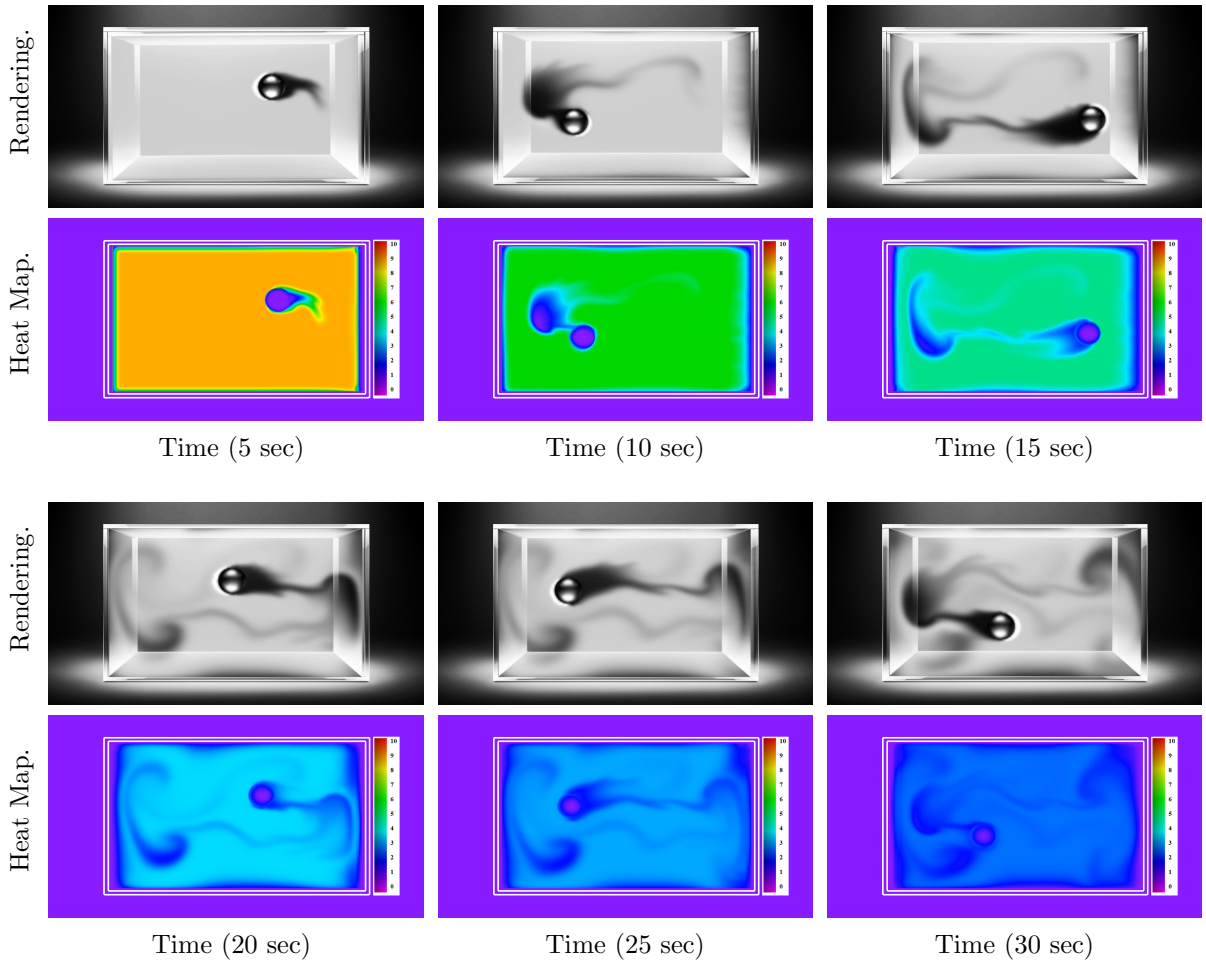


Figure 4: Experiments Results of GPU-Accelerated Smoke Thermodynamics Simulation.

where the obstacle is a none-sphere object, such as the object with a non-irregular shape, we have shown the simulation of the interaction between the smoke and: a glass teapot(b), a glass dragon(c), a glass bunny(d) a glass armadillo(e) and a glass angel(f). Besides the oscillating sphere example(a), other cases(b-f) are not moving.

As shown in the second and fourth row of Fig. 3, presented are the distributions of the heat among a cold space with ambient temperature as 0°C. As an impulse of the temperature of a 10°C at the center of the smoke, the temperature is advected by the smoke and diffused into the space. We visualize the heat map with colors where the temperature is varying between 0°C(purple) and 10°C(red). In order to plot the heat value with realistic visual effects, we are using the heat mapping algorithm by transforming the HSB color space into the RGB color space[36]. The correlation between the color and the heat value is represented as the linear interpolation of the hue value from 0 to 4.5 between red and purple, which is transforming into the heat map color space from min value of 0°C and max heat value of 10°C, the correspondence between temperature(number in °C) and the color is shown at the right side label next to the white boxes in the simulation. During the simulation, we consider the heat transfer between the hot smoke and the cold sphere, so, there is heat distribution colored on the sphere. Similarly, the heat maps are rendered for the simulations on other obstacles such as teapot and dragon, etc. as well.

As Shown in Fig. 4, another experiment is conducted to simulate a sphere that is swimming

in a glass box filled with smoke. The initial velocity of the sphere is vertical to the gravity diffraction. Therefore, according to Hook's law shown in Eq. 23, the sphere will rotate periodically around the center. In this figure, different columns represent different time slots during the simulation. As the rotating sphere push the smoke forward, the moving-forward sphere left an empty area with a low concentration of smoke behind itself. On the other hand, the smoke in front of the sphere will become denser due to being "pushed". The concentration maps of the smoke which is changing with the time t from 1sec to 30sec are shown in the first and third row in Fig. 4.

The second and fourth row in Fig. 4 visualizes the heat map of the fluid simulation. Noted that the sphere and the ambient temperature of the environment outside the glass box is a low temperature of 0°C. At the initialization step, the smoke in the glass box is set with a high temperature 10°C. During the simulation, as time goes by, the smoke is becoming cooler as the heat is attracted to the environment at the same time the periphery area of the sphere is becoming hotter as it is heated by the smoke. However, the heating speed of the sphere is much slower than the cooling speed of the smoke as the environment is an open space and has a constant low temperature.

5. Conclusions

During this experiment, we have implemented the GPU-based fluid simulation which takes into account the interaction between object and smoke-like fluid and has some realistic results shown in the figures above. However, there are still some problems that are waiting for us to deal with, such as the ones that happen to the simple advection model of the solid obstacle. Actually, this model could only deal with the translation of solid objects rather than any other complex motion such as rotation or deformation which may always happen on the common simulations such as swimming animals or rotating gears. The reason why this model is limited for such cases is that the simple advection model works only if the velocity is exactly the same for every tiny part of the obstacle, and also the motion equation of the object is known as the input parameters. But for complex motion like deformation or rotation, some more advanced techniques are needed in order to estimate the velocity of the edges along with the obstacle, this part can be the next step for our research. Besides this, as we mentioned in the introduction, we need to continue our work by taking the multi-phase fluid simulation into account, and validate this further work on the simulation of the formation of the droplet from a square and finally apply this to the GPU-based simulation of the droplet evaporating on a heating surface. However, our presented methods are focusing on the 2D fluid thermodynamics simulation, so the possible expansion from the 2D simulation into the realistic 3D fluid rendering is a both challenging and promising task.

6. References

[1] M. Müller, D. Charypar, and M. H. Gross, "Particle-based fluid simulation for interactive applications.," in *Symposium on Computer animation*, pp. 154–159, 2003.

[2] S. Premžoe, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker, "Particle-based simulation of fluids," in *Computer Graphics Forum*, vol. 22, pp. 401–410, Wiley Online Library, 2003.

[3] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–6, 2008.

[4] T. De Witt, C. Lessig, and E. Fiume, "Fluid simulation using laplacian eigenfunctions," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 1, pp. 1–11, 2012.

[5] G. A. Mastin, P. A. Watterberg, and J. F. Mareda, "Fourier synthesis of ocean scenes," *IEEE Computer graphics and Applications*, vol. 7, no. 3, pp. 16–23, 1987.

[6] K. M. Kalland, "A navier-stokes solver for single-and two-phase flow," Master's thesis, 2008.

[7] H. Jasak, A. Jemcov, Z. Tukovic, *et al.*, "Openfoam: A c++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000, pp. 1–20, IUC Dubrovnik Croatia, 2007.

[8] L. Bertagna, S. Deparis, L. Formaggia, D. Forti, and A. Veneziani, "The lifev library: engineering mathematics beyond the proof of concept," *arXiv preprint arXiv:1710.06596*, 2017.

[9] A. Kolb and N. Cuntz, "Dynamic particle coupling for gpu-based fluid simulation," in *Proc. 18th Symposium on Simulation Technique*, pp. 722–727, Citeseer, 2005.

[10] K. Crane, I. Llamas, and S. Tariq, "Real-time simulation and rendering of 3d fluids," *GPU gems*, vol. 3, no. 1, 2007.

[11] M. J. Harris, "Fast fluid dynamics simulation on the gpu.," *SIGGRAPH Courses*, vol. 220, no. 10.1145, pp. 1198555–1198790, 2005.

[12] M. Griebel and P. Zaspel, "A multi-gpu accelerated solver for the three-dimensional two-phase incompressible navier-stokes equations," *Computer Science-Research and Development*, vol. 25, no. 1, pp. 65–73, 2010.

[13] S. A. Putnam, A. M. Briones, L. W. Byrd, J. S. Ervin, M. S. Hanchak, A. White, and J. G. Jones, "Microdroplet evaporation on superheated surfaces," *International journal of heat and mass transfer*, vol. 55, no. 21-22, pp. 5793–5807, 2012.

[14] U. A. Nuli and P. Kulkarni, "Sph based fluid animation using cuda enabled gpu," *International Journal of Computer Graphics & Animation*, vol. 2, no. 4, p. 45, 2012.

[15] P. Clausen, M. Wicke, J. R. Shewchuk, and J. F. O'brien, "Simulating liquids and solid-liquid interactions with lagrangian meshes," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, pp. 1–15, 2013.

[16] F. De Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun, "Power particles: an incompressible fluid solver based on power diagrams.," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 50–1, 2015.

[17] X. Zhang, M. Li, and R. Bridson, "Resolving fluid boundary layers with particle strength exchange and weak adaptivity," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–8, 2016.

[18] M. Köster and A. Krüger, "Adaptive position-based fluids: improving performance of fluid simulations for real-time applications," *arXiv preprint arXiv:1608.04721*, 2016.

[19] M. Chu and N. Thuerey, "Data-driven synthesis of smoke flows with cnn-based feature descriptors," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[20] M. Akbay, N. Nobles, V. Zordan, and T. Shinar, "An extended partitioned method for conservative solid-fluid coupling," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[21] K. Nagasawa, T. Suzuki, R. Seto, M. Okada, and Y. Yue, "Mixing sauces: a viscosity blending model for shear thinning fluids," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–17, 2019.

[22] R. Goldade, M. Aanjaneya, and C. Batty, "Constraint bubbles and affine regions:

reduced fluid models for efficient immersed bubbles and flexible spatial coarsening," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 43–1, 2020.

[23] Y. Fang, Z. Qu, M. Li, X. Zhang, Y. Zhu, M. Aanjaneya, and C. Jiang, "Iq-mpm: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 51–1, 2020.

[24] S. Yang, S. Xiong, Y. Zhang, F. Feng, J. Liu, and B. Zhu, "Clebsch gauge fluid," *ACM Transactions on Graphics (TOG)*, vol. 40, pp. 1–11, 8 2021.

[25] S. Xiong, R. Tao, Y. Zhang, F. Feng, and B. Zhu, "Incompressible flow simulation on vortex segment clouds," *ACM Trans. Graph.*, vol. 40, no. 4, 2021.

[26] L. Ruan, J. Liu, B. Zhu, S. Sueda, B. Wang, and B. Chen, "Solid-fluid interaction with surface-tension-dominant contact," *ACM Trans. Graph.*, vol. 40, July 2021.

[27] M. Wang, Y. Deng, X. Kong, A. H. Prasad, S. Xiong, and B. Zhu, "Thin-film smoothed particle hydrodynamics fluid," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–16, 2021.

[28] P. Rideout, "Fluid sim." `https://prideout.net/`, May 2012.

[29] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 121–128, 1999.

[30] H. Bhatia, G. Norgard, V. Pascucci, and P.-T. Bremer, "The helmholtz-hodge decomposition—a survey," *IEEE Transactions on visualization and computer graphics*, vol. 19, no. 8, pp. 1386–1404, 2012.

[31] T. Simchony, R. Chellappa, and M. Shao, "Direct analytical methods for solving poisson equations in computer vision problems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 5, pp. 435–446, 1990.

[32] R. Amorim, G. Haase, M. Liebmann, and R. W. Dos Santos, "Comparing cuda and opengl implementations for a jacobi iteration," in *2009 International Conference on High Performance Computing & Simulation*, pp. 22–32, IEEE, 2009.

[33] G. W. Mulholland, "Smoke production and properties," *SFPE handbook of fire protection engineering*, vol. 3, 2002.

[34] S. Richardson, "On the no-slip boundary condition," *Journal of Fluid Mechanics*, vol. 59, no. 4, pp. 707–719, 1973.

[35] Y. Feiniu, L. Guangxuan, F. Weicheng, *et al.*, "High quality interactive volume rendering based on 3d texture mapping using opengl sl," *CJK-MI*, pp. 134–138, 2004.

[36] J. D. Foley, F. D. Van, A. Van Dam, S. K. Feiner, J. F. Hughes, and J. Hughes, *Computer graphics: principles and practice*, vol. 12110. Addison-Wesley Professional, 1996.