# Fluid-dynamic-based Three-dimensional Space Smoke Rendering Method

**Ping Zhang[1], Jie Xiao[2], Honghui Li[1]**

[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
[2] SINOCAN Intellitech Ltd., Beijing 100120, China
11120506@bjtu.edu.cn, hhli@bjtu.edu.cn

**Abstract -** Based the theory of fluid dynamics, an efficient method is proposed for rendering smoke movement in three-dimensional space. Real-time rendering of smoke movement is implemented with coarse grid and larger time interval by solving Navier-Stokes equations. Experimental results show that the method is stable, simple, and can get realistic effects.

Index Terms - real-time rendering, smoke, fluid dynamics, 3D, OpenGL

## I. Introduction

Real-time rendering of smoke can be applied to reproduction of the industrial pollution or accident scene. Simulating the movement of smoke in a virtual environment, instead of practical experiments, we can get rid of many environmental factors, and get a more intuitive result. In addition to having realistic visuals, the method used to render smoke movement in a pollution or accident scene should be stable and fast, and can interact with the environment (such as wind, obstructions, etc.) as well.

In order to achieve real-time, smooth effect, dynamic texture mapping technique is often used to render smoke movement [1]. Its implementation is simple and fast. But with this method, physical properties of smoke movement cannot be described with intuitive parameters. Thus, it is difficult to form a unified description of the action of external force, and to interact with the environment effectively. Particle system was proposed by Reeves for irregular fuzzy object [2]. The particle system fully reflects the dynamic and random characteristics of the smoke movement. However, just due to strong randomness, its controllability and adaptability is no good under different environments and scenarios. Moreover it requires quantities of particles for rendering. Jos Stam of [3] proposed the method based fluid dynamics. The advantages of this method include realistic simulation of the smoke movements, clearly described by physical variables, and interaction with the environment properly. Its disadvantages is heavy computation and difficulty of real-time rendering. For purpose of real-time rendering, Jos Stam put forward an algorithm [4] which also basically meets our requirements.

In this paper, Jos Stam's algorithm of [4] is introduced to three-dimensional space. A real-time OpenGL-based smoke rendering method is presented by studying the fluid-dynamic-based fluid rendering algorithm.

## II. Algorithm Description

The smoke in this paper refers to a cloud of fine industrial dusts suspended in gases and its density and temperature are regarded as constant. Given that the velocity and the pressure are known for some initial time t=0, then the evolution of these quantities over time is given by the Navier-Stokes equations [3]:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

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

where $\rho$ is the density, $p$ is the pressure, $\nu$ is the kinematic viscosity of the fluid, and $\mathbf{f}$ accounts for external forces. "·" is the dot product between two vectors. $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$ in three-dimensions.

Equation (3) can be obtained by combining (1) and (2). The steps that lead to the equation can be found in [3].

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{3}$$

The equation tells us how the velocity will change over an infinitesimal time interval. This velocity field can then be used to animate particles. In the case of smoke, it is difficult to model every dust particle. Hence, the limited space is divided into a series of grid cells of the same size as shown in Fig. 1.

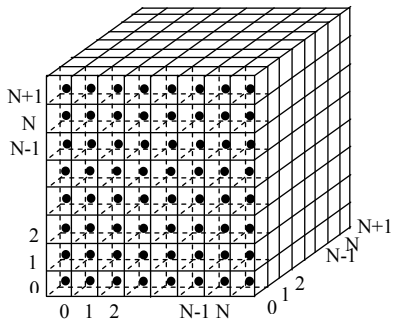


Fig. 1 Computational grids considered in this paper

The dust particles are replaced by smoke density from which we can know the amount of dust particles for each grid cell in space. Jos Stam of [4] proposed an equation which describes the evolution of the density field through the velocity field of the smoke:

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + S \tag{4}$$

Equation (3) and (4) show the process of gradual change over a single time interval in velocity and density respectively.

It is easy to find the similarity between two equations in structure. We roughly consider that density and velocity over a time interval both change due to three causes: the addition of forces, diffusion and self-advection [4].

To explain in detail, the addition of forces in velocity field is the acceleration produced by the wind or other factors. In density field, it is regarded as the addition of smoke. Diffusion is a transport phenomenon in smoke which happens when there is a gradient. Self-advection is the fact that particles are transported by the fluid's velocity. Hence, the density follows the given velocity field, and these particles will transport the fluid's velocity to another location. These are basic compositions of the density solver and velocity solver. As shown in Fig. 2, given an initial value, the density of each grid cell will be changed three times in a single time interval. And the final value after this interval will be the initial density of next time interval.

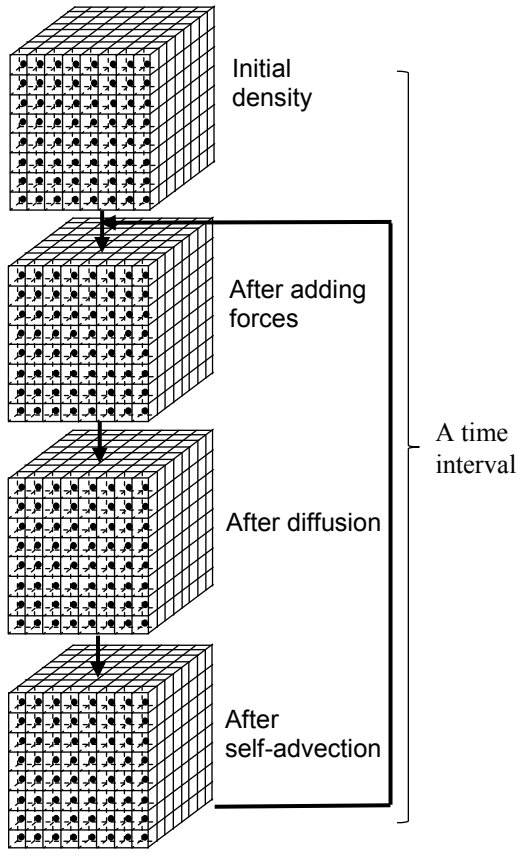


Fig. 2 The fundamental of the density solver

Now, we can get the density of each grid cell. It is used to do the smoke rendering. As the density refers to the quantity of dust particles, the density of each grid cell is in the range of 0 to 1, while 0 indicates that there is no dust particles in this grid cell.

So, in conclusion, the fundamental of the algorithm is: given an initial value, the velocity and density of each grid cell will be updated according to events, such as user's input or reaching the border. The rendering is a series snapshot of the changing density in grid cells.

## III . Design and implementation of Algorithm

The procedure of the core algorithm is as follows:

1. Initialize the velocity field and the density field of each grid cell;
2. Increase the density source or provide acceleration according to user's input;
3. Solve the velocity field;
4. Solve the density field;
5. Render the smoke;
6. If don't quit then return to step 2.

Take a detailed analysis of each step:

The first step is to initialize the velocity field and the density field of each grid cell. Assuming that the velocity and density are defined at the center of each grid cell as shown in Fig. 1. Given that the density and velocity are known for zero. In addition, we assigned an extra layer of grid cells as the borders.

The second step is to increase the density source or provide acceleration according to user's input. The shape and movement locus of the smoke can be changed by the variation of density or velocity. In this paper, users can hold or release I key on the keyboard to start or stop the increase in density. The direction and speed of the velocity are set in advance. The time interval dt between two snapshots is a fixed value given in advance.

The third step is to solve the velocity field. To finish this procedure, there are five terms to calculate: external forces term; diffusion term; projection term; self-advection term; projection term. Each part will be discussed in detail soon after.

The fourth step is to solve the density field. The method to solve density field is similar to solving velocity field. The only difference is that there is no need for projection operation because the density field is a scalar field. So, to finish this procedure, there are three terms to calculate: external forces term; diffusion term; self-advection term.

The fifth step, render the smoke. We use the OpenGL graphics API to draw quadrilaterals at the interface between two grid cells. As shown in Fig. 3, we should draw three quadrilaterals for each coordinates between each two grid cells. Set the color and transparency of points before drawing by using glColor4f () function. The RGB value of the color use the density of the point, and the transparency value set as the fixed value, which is 0.05 in this paper's program. The function "GlBegin (GL_QUADS); glEnd ();" is used to draw a quadrilateral. And the glVertex3f (x, y, z) function for a group of four which means the four vertices of the quadrilateral.

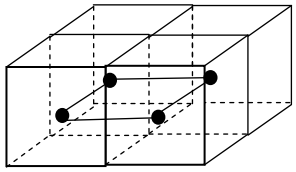

Fig. 3 Draw the planes to render smoke

*A. Data Structure*

The density and velocity of each grid cell is stored in arrays. For the grids in this paper as shown in Fig. 1, the total number of the grid cells is:

size = (N+2)*(N+2)*(N+2);

The density and velocity before and after a time interval stored in the arrays:

float dens[size], dens_pre[size];

float u[size], v[size], w[size], u_pre[size], v_pre[size], w_pre[size];

For efficiency reasons, we use one-dimensional arrays instead of three-dimensional arrays. The array subscript on behalf of the grid cell (i, j, k) is represented by the macro below:

#define IX (i, j, k) ((i) + (N+2)*(j) + (N+2)*(N+2)*(k))

For instance, the density of grid cell (i, j, k) is expressed as dens [IX (i, j, k)].

*B. Addition Of Forces*

It is easy to calculate the addition of external forces term. First of all, we assume that the external forces in a time interval dt is constant, and its value comes from an array source [size]. The original value stored in the array value [size]. Traverse each grid cell, update its value:

value [i] + = dt * source [i];

The external force in this article's scene is regarded as the equipment leaks in a warehouse. The smoke has initial velocity because of the pressures inside the equipment. The released source values controlled by users.

*C. diffusion*

Smoke molecules will transfer from the high density region to the low density region. It means that the density will transfer from one grid cell to others. Consider one of the grid cell first, we assume that the interdiffusion of the grid cells occurred among its six immediate neighbors (see Fig. 4) at a rate diff.

The density of a grid cell will flow to its neighbors, and its neighbors' will lose to the grid cell as well. Diffusion calculating is actually transformed into solving a Poisson equation [5].We use the Gauss-Seidel method for solving. Traverse each grid cell to update its value:

```
a=dt*diff*N*N*N;
value[IX(i,j,k)] = (value_pre[IX(i,j,k)]
                    + a*( value[IX(i-1,j,k)]
                    + value[IX(i+1,j,k)]+ value[IX(i,j-1,k)]
                    + value[IX(i,j+1,k)]+ value [IX(i,j,k-1)]
                    + value [IX(i,j,k+1)]))/(1+6*a);
```

*D. self-advection*

To solve the "advection", we use the "semi-Lagrangian" as Jos Stam did in [6]. Compared to Jos Stam's implementation in two-dimensional space, the calculation in three-dimensional space need to add a new component stand for the additional coordinate axis. The basic idea for this solve is to think of the density as a group of particles, and then we only keep track of the particles. First, we find the particles which end up at the grid cell's centre over a single time interval. Then we trace the particles' position backwards through the velocity field and get a new position. The density and velocity of these particles can be obtained by linearly interpolating the density and velocity of their neighbours. We then assign the density and velocity to the grid cell at the new position.

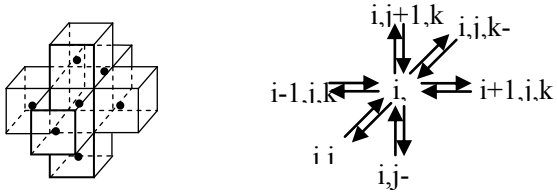


Fig. 4 Density diffusion with direct neighbors

*E. projection calculation*

After the three step above, we can get a velocity field which is the sum of a mass conserving field and a gradient field. In order to show the trend of the smoke movement, we need to add a method to turn the velocity field into a zero divergence velocity field. Compared to the calculation in two-dimensional space, the calculation in three-dimensional space also need to add a new component stand for the additional coordinate axis.

*F. Boundary Treatment*

The simulation of the smoke movement in this paper is considered in a limited space. We can see that there is no boundary treatment involved in the discussion of the external force, diffusion, advection in the front part of the article. But we have assigned an extra layer of grid cells as the border in advance. Boundary grid cells is divided into two cases as shown in Fig. 5: edges and vertices.

We assume the density having continuity when reaches the wall. For velocity field, the "i" or "k" component of the velocity should be zero on the component of the velocity, while the "j" component of the velocity should be zero on the horizontal walls. The "i", "j", and "k" stand for the coordinate axis shown in Fig. 5.

We use the edge IX (i, j, 0) in Fig. 5 as an example. The "j" component of the velocity should be zero, and the "i" component of the velocity should be the same. The "k" component of the velocity should be the opposite number of its medial side edge:

x [IX (i, j, 0)] = -x [IX (i, j, 1)];

For the vertex, such as IX (N +1, N +1, 0) in Fig. 5, its value is the average of the three neighbour grid cell's value of three different coordinate axes.

```
x [IX (N +1, N +1, 0)] = 1.0/3.0 * (x [IX (N, N +1, 0)]
                                    + x [IX (N +1, N, 0)]
                                    + x [IX (N +1, N +1, 1)]);
```

The calculations of the external force, diffusion, advection and projection all have to contain the boundary calculation.

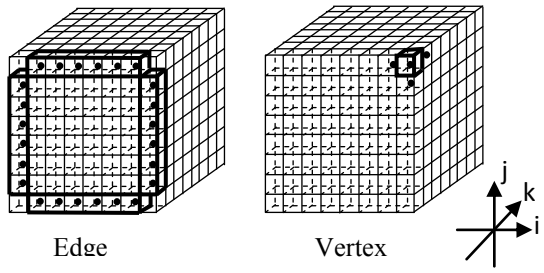


Fig. 5 Boundary grid cells is divided into two cases: edges and vertices

## IV . Simulation results

The application is written in C++. Its development platform is VS2010, and it is run on an ordinary PC, the machine configuration is:

CPU: dual-core i5 2.6GHz;

Memory: 4GB;

Video card's memory 2GB.

In the simulation program, we want to simulate the movement of smoke in an industrial warehouse after a leakage accident. Industrial gas storage container is placed in a limited space. When a leakage occurs, the gas and dust are ejected due to the internal pressure of the container. And then the smoke, as a mixture of gas and dust, moves in the space. After reaching the walls, floors, or roofs for several times, the smoke in the room could be uniformly distributed.

In the implementation of the program, we set a density source at the top of the storage container where the leak happens. At the same place, set a force source, and make its direction the same as the leakage. Also, set an initial velocity facing the ground to simulated gravity.

The simulation results shown in Fig. 6. The size of grid cell is 32 * 32, the time interval dt = 0.1, the average frame rate is 57fps. As shown in the first picture, we can see the ejection of the smoke by the internal pressure. The second shows the cyclotron motion of the smoke after reaching the boundary wall. In the last picture, the smoke evenly fill the limited space.

## V. Conclusion

This article introduces the real-time fluid-dynamic-based fluid simulation algorithm designed by Jos Stam into a three-dimensional space. The paper presents a method which renders the smoke movement in limited three-dimensional space. The simulation results show that this method can show the physical characteristics of the smoke movement nicely and the visual effects is real. Furthermore, it is simple to implement and interact with user's input. However, if we want to get a more sophisticated result, and subdivided the grid cells into large amount, the frames per second will be too low, and the movement is not smooth. Therefore, to further optimize the algorithm, and find the balance of the visual effects and the computation speed is the direction of future research.


## Acknowledgement

This work has been supported by Project NO. 2012AA040912 of the National High-Technology Research & Development Program of China.


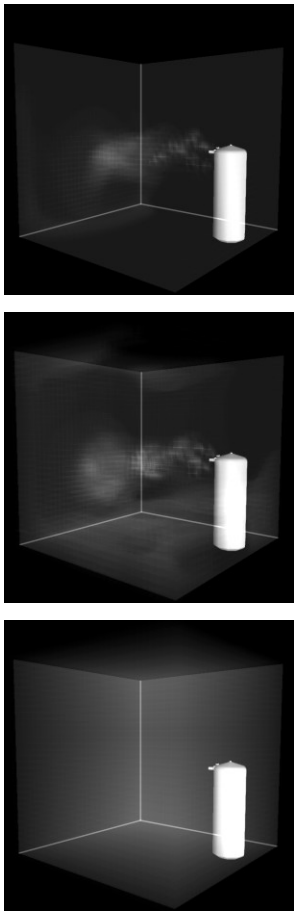

Fig. 6 Simulation results of the smoke movement

## References


[1] NIE Qing, ZHAN Shou-yi, "Real-time fluid-dynamic-based fire modeling and rendering technology," *Computer Engineering and Design*, vol.27, no.21, pp. 3959-3961, November 2006.

[2] Reeves, William T. "Particle systems--a technique for modeling a class of fuzzy objects." ACM SIGGRAPH Computer Graphics. Vol. 17. No. 3. ACM, 1983.

[3] Jos Stam, "Stable Fluids", In SIGGRAPH 99 Conference Proceedings, Annual Conference Series, August 1999, 121-128.

[4] Jos Stam, "Real-Time Fluid Dynamics for Games". Proceedings of the Game Developer Conference, March 2003.

[5] ZHAO Yang, RUAN Yan-ping, YANG Jian-lan, "Research of Chinese Ink Painting Algorithm Based on Fluid Dynamic," *JOURNAL OF LIAONING UNIVERSITY Natural Sciences Edition*, vol.39, no.2, pp.144-148, 2012.

[6] Jos Stam, "A Simple Fluid Solver based on the FFT", Journal of Graphics Tools, Volume 6, Number 2, 2001, 43-52.