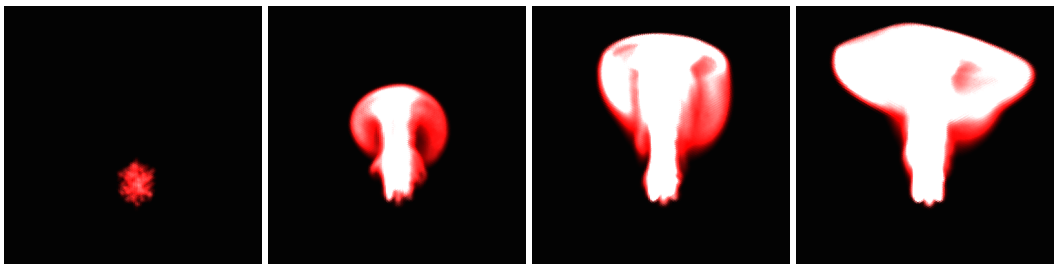
# Realistic Image Synthesis Assignment 3 - Rendering Fluids
# July 31, 2015

Hikaru Ikuta*
Department of Information Physics and Computing, The University of Tokyo



(a) $t = 0$ (b) $t = 4$ (c) $t = 10$ (d) $t = 22$

**Figure 1:** *Fluid simulation results.*

## Abstract

For my final project, I have implemented volume rendering for heterogeneous participating media to render scenes with realistic smokes. To generate physically accurate density distributions of smokes, I have implemented fluid simulations based on the work by J.Stam [2003].

**Keywords:** heterogeneous participating media, fluid simulation, path tracing

## 1 Introduction

For my final project, I wanted to explore on implementing computer-generated models to create sophisticated scenes. To this end, I focused on implementing volume rendering for heterogeneous participating media, to render realistic smokes.

### 1.1 Summary of accomplishments

The major features I have implemented are: (i) Volume rendering for heterogeneous participating media, and (ii) Fluid simulation. These features will be explained in detail in this paper.

I have also implemented the following minor features: (i) Path tracing for Lambertian and specular surfaces, (ii) Importance sampler for image based lighting, and (iii) Texture mapping for Lambertian surfaces.

Until the previous assignment, I have been working with Kento Masui, who has implemented the following features: (i) SIMD ray-triangle intersection, and (ii) Multithread rendering.

### 1.2 Challenges

The density distribution of the smokes were generated by a fluid simulator I have implemented based on the work by J.Stam [2003]. The method introduced in [Stam 2003] simulates two dimensional Newtonian fluids with incompressible flow, as it will be explained later.

*e-mail:hikaru_ikuta@ipc.i.u-tokyo.ac.jp

For this project, we needed a simulator for three dimensional fluids. Therefore, one main challenge of this project was to construct a three dimensional fluid simulator, based on the two dimensional description. A physical understanding of the simulation method was required to construct a working three dimensional fluid simulator based on their work. Since the [Stam 2003] focused on the implementation of the algorithm, this was not as straightforward as mentioned in the paper. In this paper, we will study the physical and mathematical aspects required to extend the method to a three dimensional one.

This paper is organized as follows. First, related works of fluid simulation and volume rendering are introduced. Then, the methods and simulation settings are described in detail. Finally, the results of the implementation and the discussion are given.

## 2 Related Work

### 2.1 Fluid Simulation

The fluid simulation used in this work uses a model where the fluid space is divided into meshes, and the fluid is modeled by vector fields and density fields. Another typical method of fluid simulation is Smoothed Particle Hydrodynamics (SPH), as in works as [Mller et al. 2003]. In SPH, fluids are modeled as particles, and smoothing kernels are used to obtain a continuous representation of the fluid.

One feature of SPH is that it is easy to create a more sophisticated and accurate model, such as surface tensions in [Mller et al. 2003]. On the other hand, the vector field-based approach by [Stam 2003] uses a more simple model and is more straightforward to implement, which is a reason why we used it for our purpose.

### 2.2 Volume Rendering

The volume rendering algorithm is based on ray marching. Ray marching is a general method for evaluating values along a ray, such as line integrals, intersecting implicit surfaces such as fractals [McGuire 2014], etc. The method used in this project was inspired by [Delalandre et al. 2010]. Our method uses ray marching to evaluate the line integral appearing in the transmittance within participating media. Other methods of volume rendering include photon

mapping [Jensen and Christensen 1998], virtual ray lights [Novák et al. 2012]. In these methods, information of emitted lights are first stored, and the radiance in the scene is then estimated using the stored information. Among these methods, ray marching is the most simple and straightforward method, which is why we have chosen these methods for our project.

# 3 Methods

## 3.1 Fluid Simulation

The fluid simulator is based on the Navier-Stokes equations, given as follows:

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

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

where $\mathbf{u}$ is the velocity, $\rho$ is the density, $\nu$ is the viscosity, $\kappa$ is the diffusion coefficient, $\mathbf{f}$ is the external force, and $S$ is the mass source. In this implementation, we consider $\nu$ and $\kappa$ as constants, which describe the properties of the fluid. Fluids where $\nu$ can be considered as a constant is called as Newtonian fluids. Here, $\mathbf{u}$ and $\rho$ fully describes the state of the fluid. The terms $\mathbf{f}$ and $S$ can be used as inputs to control the fluid.

The simulation basically consists of 2 steps: evolution of the density, and evolution of the velocity. We consider a $(N+2)\times(N+2)\times(N+2)$-divided mesh, and calculate the time evolution of the density and velocity field within.

### 3.1.1 Evolution of Density

The time evolution of fluid density is described by Eq. 2. We will first observe the second term, which is the diffusion term. Naively discretizing Eq. 2 with the forward difference operator yields the difference equation

$$\begin{aligned}\rho^{(t+1)}(i,j,k) = \rho^{(t)}(i,j,k) + \frac{\Delta t}{\Delta x^2}\kappa\,(&\\ \rho^{(t)}(i+1,j,k) + \rho^{(t)}(i-1,j,k) +&\\ \rho^{(t)}(i,j+1,k) + \rho^{(t)}(i,j-1,k) +&\\ \rho^{(t)}(i,j,k+1) + \rho^{(t)}(i,j,k-1)&\\ -6\rho^{(t)}(i,j,k)\Big),&\end{aligned} \tag{3}$$

where $\Delta t$ and $\Delta x$ are the time steps and mesh widths, respectively. Eq. 3 provides an iteration for evolving $\rho$ over time, where the density of the next step is an explicit function of the previous steps. However, as it is mentioned in the paper, this algorithm is an unstable algorithm, i.e. $\rho$ can diverge depending on the simulation settings, such as values of $N$ and $\kappa$.

The method avoids this by using a backward difference operator for the discretization, yielding the difference equation

$$\begin{aligned}\rho^{(t)}(i,j,k) = \rho^{(t+1)}(i,j,k) - \frac{\Delta t}{\Delta x^2}\kappa\,(&\\ \rho^{(t+1)}(i+1,j,k) + \rho^{(t+1)}(i-1,j,k) +&\\ \rho^{(t+1)}(i,j+1,k) + \rho^{(t+1)}(i,j-1,k) +&\\ \rho^{(t+1)}(i,j,k+1) + \rho^{(t+1)}(i,j,k-1)&\\ -6\rho^{(t+1)}(i,j,k)\Big).&\end{aligned} \tag{4}$$

This equation can be interpreted as a system of linear equations

$$\frac{\Delta t}{\Delta x^2}\begin{pmatrix}\frac{\Delta x^2}{\Delta t}+6 & 1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}\begin{pmatrix}\rho^{(t+1)}(i,j,k)\\ \rho^{(t+1)}(i+1,j,k)\\ \rho^{(t+1)}(i-1,j,k)\\ \rho^{(t+1)}(i,j+1,k)\\ \rho^{(t+1)}(i,j-1,k)\\ \rho^{(t+1)}(i,j,k+1)\\ \rho^{(t+1)}(i,j,k-1)\end{pmatrix}$$

$$= \rho^{(t)}(i,j,k). \tag{5}$$

By collecting this equation for all $i, j$, and $k$, we obtain the linear equation of the form

$$\mathbf{A}\boldsymbol{\rho}^{(t+1)} = \boldsymbol{\rho}^{(t)}, \tag{6}$$

where $\boldsymbol{\rho}^{(t+1)}$ and $\boldsymbol{\rho}^{(t)}$ are vectors constructed by "straightening out" the $i,j,k$ components of $\rho^{(t+1)}(i,j,k)$ and $\rho^{(t)}(i,j,k)$, respectively, into column vectors. Since $\mathbf{A}$ and $\boldsymbol{\rho}^{(t)}$ are known, calculating the time evolution of diffusion reduces to solving this linear system of equations for $\boldsymbol{\rho}^{(t+1)}$.

The sparse structure of $\mathbf{A}$ can be used for efficient calculation of the solution. In this method, an iterative method known as the Gauss-Seidel Method is used, as it will be explained later.

The third term of Eq. 2, $S$, can be implemented trivially by simply incrementing $\rho$ by $S\Delta t$ in each time step. We also will omit the description of the first term, since it is throughly explained in the original paper.

### 3.1.2 Evolution of Velocity

As inicated in the original paper, due to the similarity of Eqs. 1 and 2, evolution of velocity can be done using the exact same formulations as the evolution of density. The difference in the velocity step is that some transformations are applied to satisfy mass conservation, for more realistic results. This appears as a preprocessing step for $\mathbf{u}$ in the algorithm.

This step can be explained as follows. We have the assumption that the flow of the fluid is incompressible, i.e.

$$\nabla\cdot\mathbf{u} = 0. \tag{7}$$

It is known that combined with the Navier-Stokes equations Eqs. 1-2, this is equivalent to the equation

$$\frac{\partial \rho}{\partial t} + \mathbf{u}\cdot\nabla\rho = 0, \tag{8}$$

which is also known as the continuity equation, an equation that describes mass conservation in a differential form. Therefore, we have that incompressibility is equivalent to mass conservation.

However, the computed values of the velocity field does not necessarily satisfy Eq. 7, due to computational errors, discretization errors, etc. The original paper mentions that the violation of mass conservation hinders the quality of the simulation to a visually recognizable level. Therefore, we must somehow force the simulation results to satisfy mass conservation.

In the algorithm, this is accomplished by decomposing the computed velocity field to a mass-conserving part and non-mass-conserving part, and subtracting the non-mass-conserving part away from the computed velocity field. In the original paper, it is mentioned that a result called Hodge decomposition is used.

Let $\mathbf{u}'$ be the computed velocity field, where $\nabla \cdot \mathbf{u}' := g \neq 0$. It is known that *every* three dimensional vector field can be decomposed using a vector potential and a scalar potential. Therefore, there always exists some vector potential $\boldsymbol{\chi}$ and scalar potential $\phi$, such that $\mathbf{u}$ can be decomposed as

$$\mathbf{u}' = \nabla \times \boldsymbol{\chi} + \nabla\phi. \tag{9}$$

By taking the divergence of Eq. 9, we have

$$\nabla \cdot \mathbf{u}' = \nabla \cdot \nabla \times \boldsymbol{\chi} + \nabla^2\phi, \tag{10}$$

therefore

$$g = \nabla^2\phi. \tag{11}$$

Eq. 11 is known as the Poisson Equation. Suppose that we have found solved Eq. 11 and found $\phi$. We can then construct a new velocity field

$$\mathbf{u}'' := \mathbf{u}' - \nabla\phi, \tag{12}$$

which must satisfy

$$\nabla \cdot \mathbf{u}'' = \nabla \cdot (\mathbf{u}' - \nabla\phi) = \nabla \cdot (\nabla \times \boldsymbol{\chi}) = 0. \tag{13}$$

Therefore, the new velocity field $\mathbf{u}''$ satisfies Eq. 7, ultimately meaning that it satisfies mass conservation. Therefore, we can use this transformation from $\mathbf{u}'$ to $\mathbf{u}''$ to force the computed velocity field to satisfy the mass conservation law.

In the algorithm, this transformation, i.e. solving Eq. 11 and using Eq. 12 is done before computing the diffusion and advection terms for the velocity field. The Poisson Equation Eq. 11 is solved by discretization, which yields a linear system of equations. The Gauss-Seidel method is used for solving the obtained linear system of equations, where the sparse structure is used in this case as well.

#### 3.1.3 Gauss-Seidel Method

The Gauss-Seidel Method is an iterative method for solving linear systems of equations. This method aims to solve the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^n, n \in \mathbb{N}$. For large coefficient matrices $\mathbf{A}$, it is generally computationally expensive to calculate the inverse of $\mathbf{A}$ and multiplying it to $\mathbf{x}$, which has a complexity of $O(n^3)$ with simple methods such as Gauss-Jordan elimination. The Gauss-Seidel Method avoids this computational cost, by exploiting the structure of the matrix, such as sparsity. This method only requires $O(n^2 m)$ time, where $m$ is the number of iterations.

Let $a_{ij}, x_i, b_i$ be the components of $\mathbf{A}, \mathbf{x}$, and $\mathbf{b}$, respectively. Let the $t$th iteration of $x_i$ be $x_i^{(t)}$. Then, starting from $i = 1$, The Gauss-Seidel Method is then described as Alg. 1. A key characteristic for this algorithm is that each component can depend on components of the same time step. Specifically, the first component $x_1^{t+1}$ only depends on the previous steps, $x_i^k$, but the second component $x_2^{t+1}$ depends on $x_1^{t+1}$, which is on the same time step. This is expressed by the summation depending on the index $i$. Thanks to this feature, very few memory space is required for the algorithm, since no copies of $\mathbf{x}$ are required for the iteration.

The convergence criteria can be chosen arbitrarily. In this project, we simply used a fixed number of iterations, specifically 20 iterations, for the convergence criterion.

**Algorithm 1** Gauss-Seidel Method

1: **while** Convergence criteria are not satisfied **do**
2: **for** $i \leftarrow 1, 2, \cdots n$ **do**
3: $x_i^{(t+1)} \leftarrow \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(t+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(t)}\right)/a_{ii}$
4: **end for**
5: **end while**

#### 3.1.4 Simulation Settings

The initial density distribution is shown in Fig. 1 (a). Density was distributed randomly inside a given rectangular region on the bottom of the fluid space. The initial velocity distribution was given so that all the voxels in the region had an almost uniform upward velocity with some fluctuations. The inputs $\mathbf{f}$ and $S$ were set to 0 at all times.

### 3.2 Volume Rendering

#### 3.2.1 Model Construction

The density distrbution $\rho$ was used for constructing the smoke model. The smoke is characterized by the distribution of its extinction coefficient $\sigma_t$ and scattering coefficient $\sigma_s$. The relationship between $\rho$ and $\sigma_t$ was given so that $\rho$ serves as a scaling factor of $\sigma_t$:

$$\sigma_t = \rho/800. \tag{15}$$

Given $\sigma_t$, we gave $\sigma_s$ as

$$\sigma_s = \sigma_t/2. \tag{16}$$

By definition, the relationship $\sigma_t > \sigma_s$ must hold. This relashonship states that the albedo of the medium is 0.5 for all wavelengths, giving the medium a gray color when illuminated with white light.

Note that our medium does not emit any radiance.

#### 3.2.2 Distance sampling

For evaluating the transmittance of a given line segment within the media, our method uses ray marching to evaluate the line integral in the equation of the transmittance.

To take account of scattering, distance sampling is required to determine which point the ray was scattered inside the media. Dinstance sampling was done as shown in Alg. 2. The main idea of this sampling method is to focus on the ratio of radiance that arrives at a given point, and regard that ratio as the probability of the light being able to pass that point. To sample distance, we first sample a random number $prob \in [0, 1]$ from a uniform distribution. Then, we calculate the transmittance $T$ along the ray segment that intersects with the medium. Our distance sampling method then finds the point where $T = prob$. If $T < prob$ at all points, we regard that the ray has passed through the medium.

## 4 Results and Discussion

### 4.1 Fluid Simulation

The density distribution obtained by the fluid simulation is shown in Figs. 1 a-b. The renderer for the simulation was implemented with OpenGL. Each voxel is assigned with grayscale diffuse colors and alpha values proportional to the density of the smoke. The voxels were then rendered with alpha blending, with a red-colored lighting.

**Algorithm 2** Distance Sampling

Intersect medium bounding box and ray
$segment \leftarrow$ intersection line segment of medium bounding box and ray
$prob \leftarrow random([0, 1])$
$S \leftarrow 0$
**for** $p \leftarrow$ points along $segment$ **do**
    $S \leftarrow S + \sigma_t(p)ds$
    **if** $prob < \exp(-S)$ **then**
        **return** $p$ as the scattering point
    **end if**
**end for**
(Ray has passed through medium)

### 4.2 Volume Rendering and Path Tracing

The result of the entire scene is shown in Fig. 3. Notice that the floor color is shaded upon the teapot, indicating that global illumination is accomplished by path tracing. The smoke is also rendered successfully.

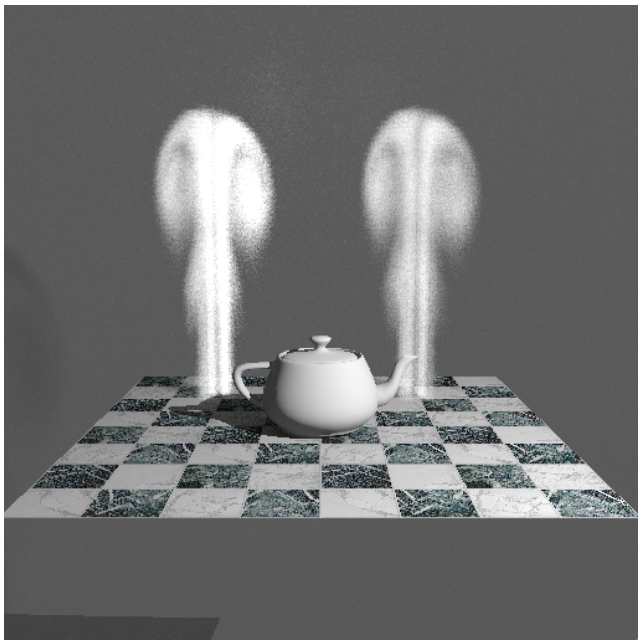


**Figure 2:** *The final image, demonstrating the results of volume rendering.*

## 5 Conclusion

The aim for this final assignment was to implement and use computer-generated models for sophisticated scenes. To this end, I have successfully implemented volume rendering for heterogeneous participating media, based on ray marching and path tracing. I have also successfully implemented a 3 dimensional fluid simulator, based on the work by J.Stam [2003]. Using the simulator, I have generated a density distribution of smoke, and finally rendered a scene containing smokes illuminated with image-based lighting and point lights.

For future works, photon mapping for enhancing volume rendering can be considered.

**Figure 3:** *The final image, demonstrating the results of the IBL importance sampler.*


## Acknowledgements

I would like to thank Kento Masui, for working together on the previous assignments, and sharing the excitement of working on this project. I enjoyed learning C++ techniques, and using them for accomplishments in this project.


## References


DELALANDRE, C., GAUTRON, P., MARVIE, J.-E., AND FRANÇOIS, G. 2010. Single scattering in heterogenous participating media. In *ACM SIGGRAPH 2010 Talks*, ACM, New York, NY, USA, SIGGRAPH '10, 14:1–14:1.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 311–320.

LANDIS, H., 2002. Global illumination in production. ACM SIGGRAPH 2002 Course #16 Notes, July.

MCGUIRE, M., 2014. Numerical methods for ray tracing implicitly defined surfaces. Williams College 2014 Course CS371 Notes, September.

MLLER, M., CHARYPAR, D., AND GROSS, M., 2003. Particle-based fluid simulation for interactive applications.

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph. 31*, 4 (July), 60:1–60:11.

STAM, J. 2003. Real-time fluid dynamics for games. *Proceedings of the Game Developer Conference*.

WIKIPEDIA. Gauss-seidel method.

WIKIPEDIA. Incompressible flow.