Interactive Rendering Method for Displaying Shafts of Light

Yoshinori Dobashi†, Tsuyoshi Yamamoto†, Tomoyuki Nishita††

†Hokkaido University
Kita 13, Nishi 8, Kita-ku,
Sapporo, 060-8628, Japan
{doba, yamamoto}@nis-ei.eng.hokudai.ac.jp

††University of Tokyo
7-3-1, Hongo, Bunkyo-ku
Tokyo, 113-0033, Japan
nis@is.s.u-tokyo.ac.jp

Abstract

Recently, graphics hardware has increased in capability, and is, moreover, now available even on standard PCs. These advances have encouraged researchers to develop hardware-accelerated methods for rendering realistic images. One of the important elements in enhancing reality is the effect of atmospheric scattering. The scattering of light due to atmospheric particles has to be taken into account in order to display shafts of light produced by studio spotlights and headlights of automobiles, for example. The purpose of this paper is to develop a method for displaying shafts of light at interactive rates by making use of the graphics hardware. The method makes use of hardware-accelerated volume rendering techniques to display the shafts of light.

Keywords and phrases: *shafts of light, graphics hardware, atmospheric scattering, realistic image synthesis*

1. Introduction

In the field of computer graphics, various methods have been developed to display photo-realistic images by taking into account physical phenomena. Among them, the effect of the scattering/absorption of light due to atmospheric particles is one of the most important elements in creating realistic images. This makes it possible to render the shafts of light caused by headlights of automobiles, street lamps, studio spotlights, and light passing through stained glass windows, for example. As a result, there exists a great deal of researches focusing on atmospheric effects [1, 2, 3, 4, 5]. The display of such effects requires the integration of the intensity of light scattered by atmospheric particles along the viewing ray. Previously, the integration was computed by using the ray tracing technique, one of the most time-consuming methods.

Conversely, the processing speed of graphics hardware has been becoming faster and faster recently. In addition, high performance graphics hardware is available even on low-end PCs. Studies of hardware-accelerated rendering is therefore one of the most important research areas for realistic image synthesis [6, 7, 8, 9, 10]. In this paper, we propose a rendering method for shafts of light by making use of graphics hardware. Our method makes use of a hardware-accelerated volume rendering technique to display the shafts of light. Shadows in the atmosphere play a very important part in enhancing the reality of synthesized images. Objects shut out portions of the light produced by light sources, resulting in non-illuminated volumes within illuminated volumes. Our method can handle the shadows by using the concept of a shadow map [11]. Using the proposed method, realistic images are generated at sufficient speed for interactive applications. Since graphics hardware is becoming faster and faster, we believe our method will realize real-time rendering of shafts of light in the near future.

This paper is organized as follows. The methods used previously are discussed in Section 2 and an overview of our method is described in Section 3. In Section 4, a shading model for the shafts of light is briefly explained. Then, in Section 5, the hardware-accelerated method for rendering the shafts of light is proposed. In Section 6, several examples demonstrate the usefulness of our method. Finally, in Section 7, the conclusion and future work are discussed.

2. Previous Work

One of the simplest methods to simulate the scattering/absorption due to atmospheric particles is to

0-7695-0868-5/00 $10.00 © 2000 IEEE

attenuate colors of objects depending on the distance from the viewpoint. This method is often used since it is computationally inexpensive and easy to implement. For example, the method is implemented for one of the standard graphics APIs, OpenGL. The method, however, cannot display the shafts of light. So, a more accurate method to simulate physical phenomena is required to create photo-realistic images. Many different methods have been proposed.

Max proposed a method for displaying the shafts of light through gaps between clouds [1, 2]. The method, however, is limited to parallel light sources. Nishita et al. improved it in order to handle point light sources taking into account their luminous intensity distributions [3]. This method makes it possible to render the shafts of light emanating from spotlights, headlights of automobiles, and so on. Rushmeier et al. extended radiosity methods to volume densities and rendered the shafts of light passing through windows [4]. Recently, Jansen et al. proposed a method using photon maps [5]. Methods for shafts of light not only in the atmosphere but also under water have been proposed [5, 12]. Most of these methods are however, based on ray tracing or the scanline algorithm, and it takes several minutes at least to create a single image. To address the problem, hardware-accelerated methods have been proposed recently. Stam used 3D hardware texture mapping to render gases in real-time [10]. This method, however, focuses on displaying gaseous phenomena and is not suitable for shafts of light.

We proposed a hardware-accelerated method for rendering shafts of light through gaps between clouds [13]. However, the method is limited to shafts of light shining through clouds and parallel light sources. Moreover, it cannot take into account the shadows in the atmosphere. The method proposed in this paper is an extended method, which can handle point light sources and also shadows.

3. Overview of Our Method

Our method utilizes a hardware-accelerated volume rendering technique to display shafts of light. For rendering volume density such as gases, using 3D texture mapping hardware is a more efficient way. However, high-resolution 3D texture is required to create realistic images of the shafts of light without aliasing problems. This results in the need for increased texture memory and a great deal of time is taken up in loading such a large 3D texture. Furthermore, 3D texture mapping hardware is still expensive and is not universally available. Therefore, we propose a method based on 2D texture mapping hardware. The method can handle point light sources and/or parallel light sources. Shadows in the atmosphere are realized by using a shadow buffer [11].

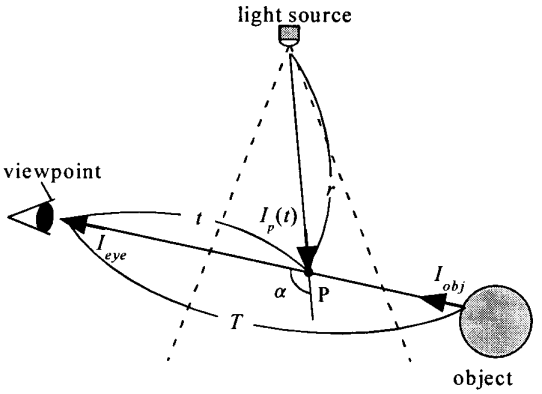


Figure 1: Shading model for atmospheric scattering.

4. Shading Model for Atmospheric Scattering

Several shading models for atmospheric scattering have been proposed [15, 3, 16]. Our method utilizes a model proposed by Nishita et al. [3]. This section describes the shading model briefly. In the following, for the sake of simplicity, let us assume the number of light sources is one. The extension to multiple light sources can be done in a straightforward way, i.e. the intensity reaching the viewpoint due to multiple light sources is obtained by calculating the contribution from each light source and summing them.

Fig. 1 shows the concept of the calculation for light scattering. In Fig. 1, a point light source is assumed. In general, the intensity of light reaching the viewpoint is expressed by the following equation.

$$I_{eye} = I_{obj}\beta(T) + \int_0^T F(\alpha)H(t)I_p(t)\beta(t)dt, \qquad (1)$$

where I_{eye} is the intensity reaching the viewpoint, I_{obj} is the intensity of an object, $\beta(t)$ the attenuation ratio due to atmospheric particles between the viewpoint and a point P on the viewing ray, t the distance between the viewpoint and point P, T the distance between the viewpoint and the object, and $I_p(t)$ the intensity of light from the light source reaching point P. $H(t)$ is a visibility function that returns the value 1 if the light source is visible from point P, or 0 otherwise. $F(\alpha)$ is a phase function of the atmospheric particles and α is the phase angle (see Fig. 1). Under the assumption that the density of the particles is uniform, $\beta(t)$ is given by the following equation.

$$\beta(t) = \rho \exp(-\kappa\rho t), \qquad (2)$$

where κ is the extinction coefficient and ρ the density of the particles. Moreover, $I_p(t)$ is given by the following equation if the light source is a point light source.

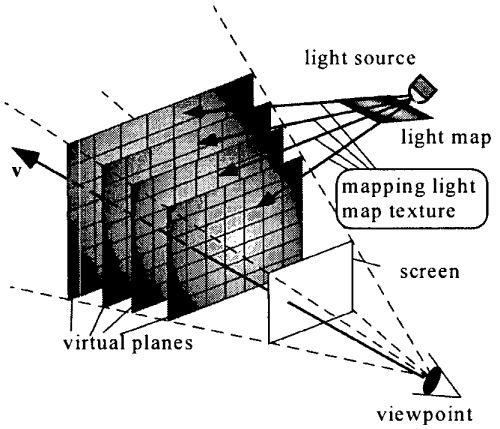


Figure 2: Basic concept of rendering shafts of light.

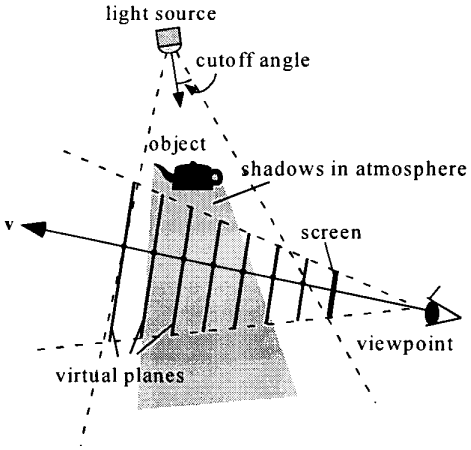


Figure 3: Calculation of shadows in atmosphere.

$$I_p(t) = I(\theta,\varphi)\exp(-\kappa\rho\, r)/r^2, \quad (3)$$

where $I(\theta,\phi)$ is the intensity of light emanating from the light source toward the direction of point P and (θ, ϕ) indicates the direction (see Appendix A). If the light source is a parallel light source, $I_p(t) = I_0$, where I_0 is a constant specified by the user. The phase function is given by the following equation.

$$F(\alpha) = K(1+9\cos^{16}(\alpha/2)), \quad (4)$$

where K is a constant. This function is used for the hazy atmosphere condition [3].

In Eq. 1, the second term indicates the total intensity of scattered light due to atmospheric particles and is directly related to the shafts of light. Most of the graphics hardware can calculate the first term by using a function for simulating fog effects. For simplicity, in the following, let us denote the second term as I_s. The following sections explain our method for calculating the second term, I_s, by making use of graphics hardware.

5. Hardware-accelerated Rendering of Shafts of Light

First, the method for rendering shafts of light without shadows is described. Then, extensions are described that take into account the shadows.

5.1 Rendering Shafts of Light

Fig. 2 shows the idea of rendering shafts of light using graphics hardware. Virtual planes are placed in front of the viewpoint in order to integrate the scattered light. Each virtual plane is parallel to the screen and represented by a n_uxn_v lattice mesh (see Fig. 2). Let us consider the calculation of the total intensity of the scattered light, I_s, in Eq. 1 for a viewing ray **v**. I_s is computed numerically by taking samples at intersections between the ray and the virtual planes. If the light source is a point light source, I_s is obtained by the following equation.

$$I_s = \sum_{k=1}^{n} I(\theta_k,\varphi_k)\xi(t_k), \quad (5)$$

$$\xi(t_k) = \frac{F(\alpha_k)\exp(-r_k)\beta(t_k)}{r_k^2}\Delta t, \quad (6)$$

where (θ_k, ϕ_k) is the direction toward the intersection point of ray **v** with the virtual plane k, n the number of the virtual planes, t_k the distance from the viewpoint to the intersection, r_k the distance between the light source and the intersection, and α_k is the phase angle. If the light source is a parallel light source, $I(\theta_k, \phi_k) = I_0$ and $\xi(t_k) = F(\alpha_k)\beta(t_k)$. The interval between the intersections, Δt, is a constant on the viewing ray **v** since the virtual planes are placed at regular intervals.

In the previous methods, Eq. 5 is calculated for each ray passing through each pixel, resulting in an increased computation time. In Eq. 5, however, a strong coherency exists for ξ, i.e. ξ has similar values for neighboring sampling points. This implies that the coarser sampling interval of ξ is sufficient for the purpose of displaying the shafts of light. Therefore, we calculate and store it at each lattice point of the virtual planes. Values of ξ at arbitrary points on each virtual plane are linearly interpolated by using the stored values. This interpolation is performed using the Gouraud shading function, which all of the graphics hardware packages support. It is possible to store the values of ξ in a look-up table. Then the look-up table can be used as a texture for the virtual planes. In this case, multi-texturing functions are required.

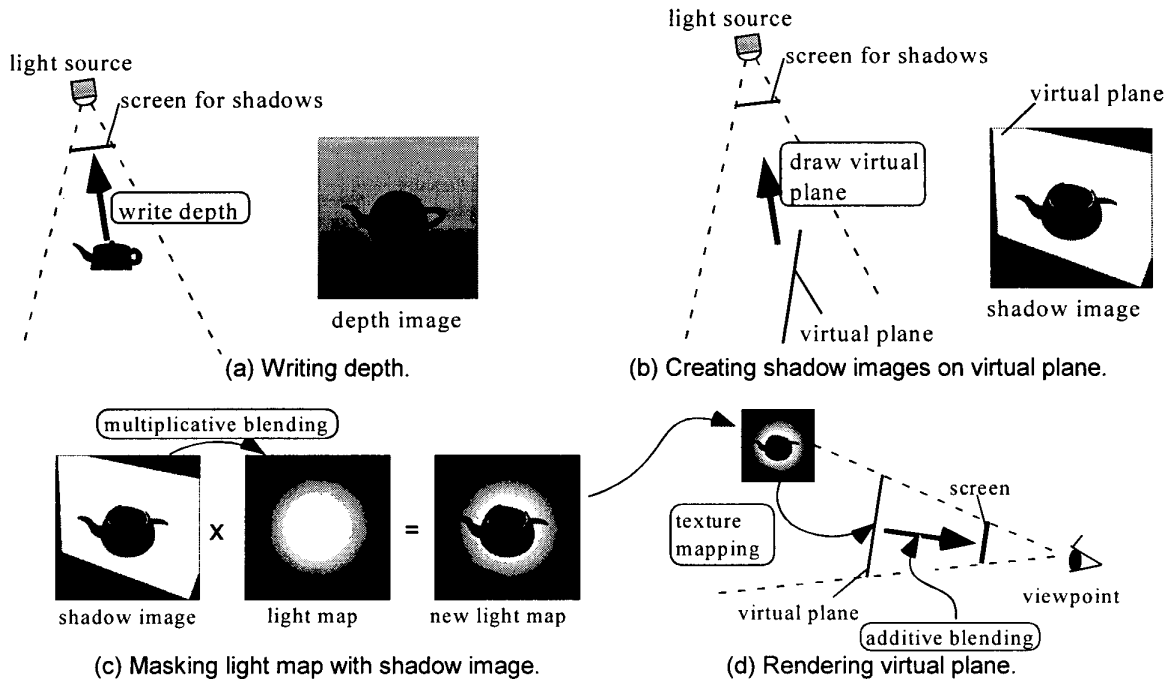


Figure 4: Algorithm for computing shadows.

On the other hand, $I(\theta_k, \phi_k)$ may have a step-change depending on the type of the light source. For example, in the case of spotlights, the intensity suddenly becomes zero at points outside its cutoff angle. It is clear that storing $I(\theta_k, \phi_k)$ at each lattice point introduces undesirable artifacts in the resulting images. Subdividing the virtual planes into a finer lattice may avoid this problem. This, however, increases the computation time to calculate ξ. Our method avoids the problem by using the idea of light map textures [14]. Light maps store the intensity distribution of a light source, $I(\theta_k, \phi_k)$, as textures. This makes it possible for us to independently control the sampling density for ξ and $I(\theta_k, \phi_k)$ since they are stored in different ways. As shown in Fig. 2, the light map texture is projected onto each virtual plane using projective texture mapping techniques [11]. $I(\theta, \phi)$ has to be multiplied by ξ as expressed in Eq. 5. This can easily be achieved by using a multiplicative texture mapping function (in OpenGL, for example, the texture is mapped using GL_MODULATE). Finally, the summation in Eq. 5 is computed by rendering all the virtual planes and creating a composite of their intensities with an additive blending function.

5.2 Shadows in Atmosphere

Shadows cast on particles in the atmosphere are very important when we attempt to create realistic images. When there are objects within an illuminated volume, non-illuminated parts arise within that volume, as shown in Fig. 3. To display this, we have to determine if the intersection points of the viewing ray **v** with the virtual planes are in shadow (see Fig. 3), and therefore non-illuminated parts of the virtual planes must be detected. We use the idea of the shadow map algorithm to achieve this [11]. The shadow map is a depth image viewed from the light source. Shadows can be computed by comparing the depth value of each point on the virtual plane with the corresponding depth value in the shadow map. The implementation of this idea using graphics hardware requires a function that can compare texture elements with pixel values in the frame buffer. Unfortunately, many hardware-rendering engines do not currently support such a function. To address this problem, Heidrich developed a method that stores the shadow map in the alpha plane of the frame buffer and shadows are displayed using alpha-test [9]. Depth values in the shadow map, however, have to be stored in 8-bit memory (or 12-bit memory using a high-end workstation). Therefore, the method might cause aliasing problems, although the technique will be useful in the future if the number of bits for the alpha-plane can be increased. Therefore, we implement the idea of the shadow map by creating shadow

textures for each virtual plane.

Fig. 4 shows the algorithm of calculating the shadow textures. First, as shown in Fig. 4a, a screen for creating the shadow textures is prepared separately from the screen for rendering the final images. For the shadow screen, the camera is placed at the position of the light source. Then, the depth values of all objects viewed from the light source are written in the depth buffer of the shadow screen and then writing to the depth buffer is disabled. An image on the right side of Fig. 4a is the depth image. Next, as shown in Fig. 4b, after initializing the frame buffer of the shadow screen using black, a virtual plane is rendered that is white in color. This creates a shadow image in which visible parts of the virtual plane viewed from the light source are rendered in white (see an image on the right side of Fig. 4b), i.e. white parts of the image are non-illuminated parts of the virtual plane. Then, as shown in Fig. 4c, the light map and the shadow image are blended with the multiplicative blending function. The resulting image is a new light map in which non-illuminated parts are masked. The image is read back from the frame buffer and used as a new light map texture. The new light map texture is projected onto the virtual plane as shown in Fig. 4d. Finally, the virtual plane is rendered with the additive blending function. The shafts of light taking into account the shadows are displayed by repeating these processes for each virtual plane.

5.3 Image Generation

The procedure for rendering the shafts of light is summarized as follows.

For each light source, repeat the following;

For each virtual plane, repeat the following;

a) Compute ξ for each lattice point of the virtual plane.

b) Create the light map texture taking into account the shadows using the method described in the previous section.

c) Project the light map texture onto the virtual plane using a projective texture mapping technique with the multiplicative blending function.

d) Render the virtual plane with an additive blending function.

As shown in the above procedure, the computational cost of the proposed method is proportional to the numbers of light sources, virtual planes, and lattice points.

5.4 Discussion on Accuracy

The proposed method uses multiple virtual planes represented by the lattice mesh. This section discusses the accuracy of the proposed method depending on the number of virtual planes and lattice points.

Figs. 5 and 6 show the comparison of images created by using different numbers of virtual planes and lattice points. In Figs. 5a through c, the numbers of the virtual planes are 15, 30, and 75, respectively. The number of lattice points is fixed, 40x40. On the other hand, in Figs. 6a through 6c, the number of the virtual planes is fixed. The numbers of lattice points are 5x5, 20x20, and 80x80, respectively. The computational times for these images are also shown in the figures. The computation was done on a desktop PC (PentiumIII 733MHz) with NIVIDIA GeForce256. The sizes of images are 640x480.

Fig. 5 shows that serious aliasing problems with shadows are caused when the number of virtual planes is small (see Figs. 5a and 5b). This problem can be avoided by increasing the number of virtual planes although the computational cost increases. Next, in Fig. 6, we can see noticeable differences, especially around the light source. In Fig. 6a, the intensity of the light scattering is much weaker than in Figs. 6b and c. This implies that the 5x5 lattice mesh is not suitable for sampling ξ accurately. Comparing Figs. 6b and 6c, the 20x20 lattice mesh seems to be enough in this example.

As shown in these examples, it is important to specify the appropriate numbers of virtual planes and lattice points. In the current implementation, we determine these numbers experimentally.

Furthermore, we investigated the computational times for the various stages of the algorithm using Fig. 5c. As a result, we found that most of the computation was spent on the shadow texture generation for each virtual plane. That is, approximately 70 percent of the computation is spent on the shadow calculation. This is because the shadow image has to be read back from the frame buffer of the shadow screen and then it has to be sent to the texture memory of the graphics hardware. This will become unnecessary when the graphics hardware will support the shadow map method.

6. Examples

Several images are created to demonstrate the usefulness of the proposed method (please see color pages for Figs. 7 and 8). Fig. 7 shows examples of the shafts of light caused by spotlights. Fig. 7a shows a teapot illuminated by a single spotlight. We acknowledge shadow effects due to the teapot; the shape of the shaft of light is clipped. Fig. 7b shows an example of the lighting design for stages. There are 5 spotlights. Some of the spotlights are special effects lights, polka dots

Fig. 8 shows examples of the shafts of light caused by the sun's rays. In Fig. 8a, mountains shut out parts of the

sun's rays, causing shafts of light along the silhouettes of the mountains. In Fig. 8b, the sunlight passes through stained glass windows in a dusty room. In this image, a pattern of the stained glass is used as the light map texture for the parallel light source (i.e. the sunlight). Shadows on objects such as the floor are realized by mapping shadow textures generated by a similar technique described in section 5.2. Beautiful shafts of light are created.

Table 1 shows the computational times for these images. The computer is the same machine used in section 5.4. The sizes of images are 640x480. The number of virtual planes, the number of lattice points, and the light map resolution are also shown in the table. As shown in Table 1, Fig. 7a was created very quickly since there is only a single light source. Fig. 7b requires a longer computational time because of the multiple light sources. However, it is fast enough for interactive applications. Fig. 8a can be created within half a second since there is only the sunlight and the number of virtual planes is rather small. For Fig. 8b, most of the computation spent on the generation of the shadow texture for the floor. Without the shadows on the floor, the computational time is 0.35 seconds.

Table 1: Computation times.

Figure No.	Virtual planes	Lattice points	Light- map res.	Time [sec.]
Fig. 7a	150	20x20	64x64	0.52
Fig. 7b	150	20x20	64x64	2.27
Fig. 8a	50	20x20	64x64	0.42
Fig. 8b	50	20x20	128x128	1.48

7. Conclusion

In this paper, we have proposed an interactive method for rendering shafts of light. The proposed method makes use of the hardware-accelerated volume rendering technique. Realistic images including the shafts of light are generated using OpenGL. Our method has the following advantages.

(1) Realistic images are created very quickly, at several frames per second if there is only one light source.
(2) Shadows in the atmosphere caused by objects obscuring the light beam are displayed by using the idea of the shadow buffer.
(3) The method can be accelerated by most of the graphics boards supporting standard functions of OpenGL. No extensions are required.
(4) The intensity of light reaching the eye is calculated taking into account the physical phenomena of the atmospheric scatterings.

There remain a few things to be done in the future. The display of shafts of light under non-uniform density distributions of particles is an interesting topic for research. Summing up the contributions of a large number of virtual planes may cause numerical problems since the available precision in the frame buffer is usually 8bit per color channel. This may cause serious aliasing problems. The idea of jittering has to be incorporated. Finally, we are sure that our method will work at real-time frame rates when graphics hardware will support the shadow map method and multi-texturing functions.

Appendix A

As shown in Fig A.1, a local coordinate system with its origin at a position of a light source is assumed. z-axis is assumed to coincide with the direction of the light source called an illumination axis. Then, the intensity distribution of the light source $I(\theta, \phi)$ is expressed by the angle θ from the illumination axis and the revolution angle ϕ from xz plane.

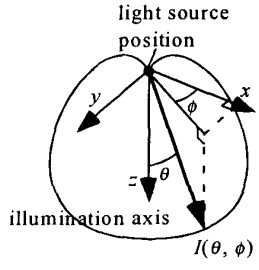
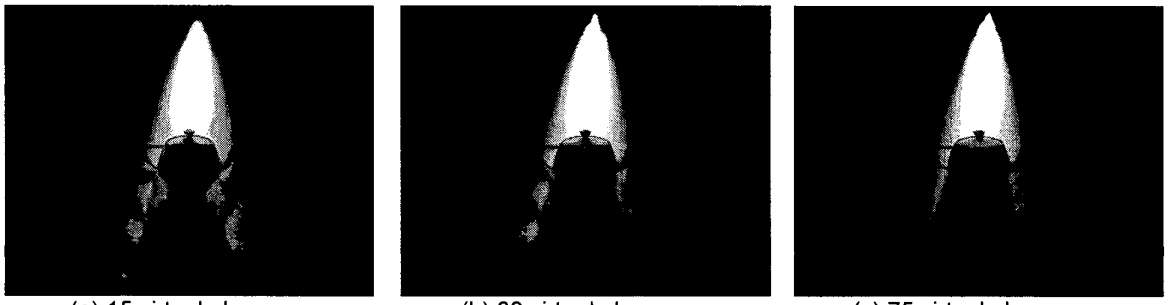


Figure A.1: Intensity distribution of light source.

References

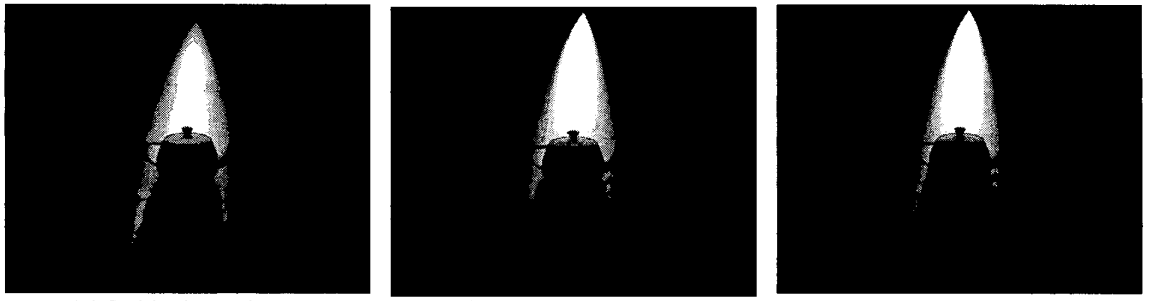
[1] N. Max, "Light Diffusion through Clouds and Haze," Graphics and Image Processing, Vol. 13, No. 3, 1986, pp. 280-292.
[2] N. Max, "Atmospheric Illumination and Shadows," Computer Graphics, Vol. 20, No. 4, 1986, pp. 117-124.
[3] T. Nishita, Y. Miyawaki, E. Nakamae, "A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources," Computer Graphics, Vol. 21, No. 4, 1987, pp. 303-310.
[4] H. E. Rushmeier, K. E. Torrance, "The Zonal Method for Calculating Light Intensities in The Presence of a Participating Medium," Computer Graphics, Vol. 21, No. 4, 1987, pp. 293-302.

[5] H. W. Jansen, P. H. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps," Proc. of SIGGRAPH'98, 1998, pp. 311-320.
[6] E. Ofek, A. Rappoport, "Interactive Reflections on Curved Objects," Proc. of SIGGRAPH'98, 1998, pp. 333-342.
[7] B. Cabral, M. Olano, P. Nemec, "Reflection Space Image Based Rendering," Proc. of SIGGRAPH'99, 1999, pp. 165-170.
[8] W. Heidrich, H. P. Seidel, "Realistic, Hardware-Accelerated Shading and Lighting," Proc. of SIGGRAPH'99, 1999, pp. 171-178.
[9] W. Heidrich, "High-quality Shading and Lighting for Hardware-accelerated Rendering," PhD thesis at University of Erlangen, 1999.
[10] J. Stam, "Stable Fluids," Proc. of SIGGRAPH'99, 1999, pp. 121-128.
[11] M. Segal, C. Korobkin, R. V. Widenfelt, J. Foran, P. E. Haeberli, "Fast Shadows and Lighting Effects Using Texture Mapping," Computer Graphics, Vol. 26, No. 2, 1992, pp. 249-252.
[12] T. Nishita, E. Nakamae, "Method of Displaying Optical Effects within Water using Accumulation Buffer," Proc. of SIGGRAPH'94, 1994, pp. 373-379.
[13] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita, "A Simple, Efficient Method for Realistic Animation of Clouds," Proc. of SIGGRAPH2000, 2000, to appear.
[14] D. Blythe, "Advanced Graphics Programming Techniques Using OpenGL," Course Note #29 of SIGGRAPH 99, 1999.
[15] R. V. Klassen, "Modeling the Effect of the Atmosphere on Light," ACM Trans. on Graphics, Vol. 6, No. 4, 1987, pp. 215-237.
[16] K. Kaneda, T. Okamoto, E. Nakamae, T. Nishita, "Photorealistic Image Synthesis for Outdoor Scenery under Various Atmospheric Conditions," The Visual Computer, Vol. 7, No. 5&6, 1991, pp. 247-258.



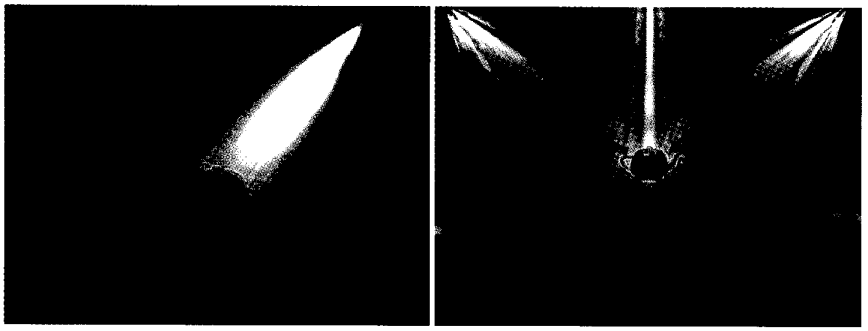
(a) 15 virtual planes 0.23 [sec] (b) 30 virtual planes 0.26 [sec] (c) 75 virtual planes 0.45 [sec]

Figure 5: Comparison of images under different numbers of virtual planes (The number of lattice points is fixed, 40x40).



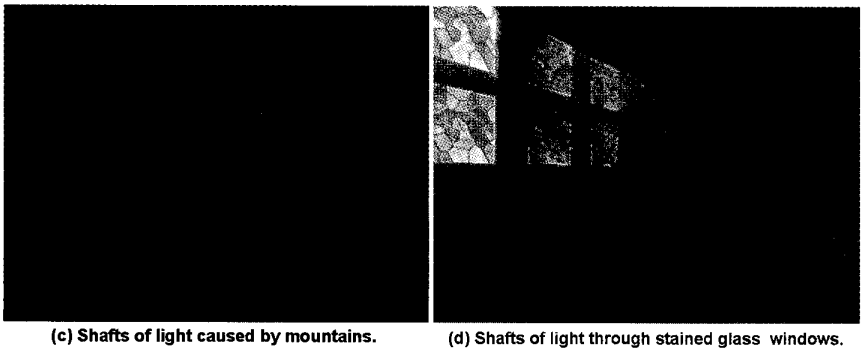
(a) 5x5 lattice points 0.26 [sec] (b) 20x20 lattice points 0.33 [sec] (c) 80x80 lattice points 1.00 [sec]

Figure 6: Comparison of images under different numbers of lattice points (The number of virtual planes is fixed, 75).



(a) A teapot illuminated by a single spotlight. (b) An example of multiple spotlights.

Figure 7: Examples of shafts of light produced by spotlights.



(c) Shafts of light caused by mountains. (d) Shafts of light through stained glass windows.

Figure 8: Examples of shafts of light produced by the sunlight.