Graphics Hardware (2002), pp. 1–10
Thomas Ertl, Wolfgang Heidrich, and Michael Doggett (Editors)

Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware

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

[†]Hokkaido University [††]The University of Tokyo

Abstract

To create realistic images using computer graphics, an important element to consider is atmospheric scattering, that is, the phenomenon by which light is scattered by small particles in the air. This effect is the cause of the light beams produced by spotlights, shafts of light, foggy scenes, the bluish appearance of the earth's atmosphere, and so on. This paper proposes a fast method for rendering the atmospheric scattering effects based on actual physical phenomena. In the proposed method, look-up tables are prepared to store the intensities of the scattered light, and these are then used as textures. Realistic images are then created at interactive rates by making use of graphics hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Graphics processors; I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and frame buffer operations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, Shading, Shadowing and Texture

1. Introduction

In the field of computer graphics, realistic image synthesis is one of the most important research subjects. When there are small particles such as dust in the air, light is scattered and absorbed by them. This is called atmospheric scattering. Several different methods have been developed to simulate this. These methods make it possible to display various effects, such as the light beams caused by spotlights, shafts of light through clouds, foggy scenes, the sky, and the earth viewed from space[1, 2, 3, 4, 5]. These research results are now widely used for a variety of applications, such as stage lighting designs, driving simulations, and space/flight simulations. However, the computational cost is very expensive, since the intensities of the scattered light must be integrated along with each viewing ray. A fast and precise rendering method must be developed, especially for drive simulators and space/flight simulators.

This paper proposes a method for rendering these atmospheric scattering effects at interactive rates. Our method can handle a wide range of atmospheric scattering effects, including the scattering observed in a smoky room, scattering due to fog, and scattering due to the earth's atmosphere. In the case of a smoky room, the light beams can be produced by point light sources and/or sunlight, and our technique can take this into account. The scattering due to the earth's atmosphere is indispensable for rendering the sky or the earth viewed from space, and we have to take into account the following phenomena. That is, the sky is bluish at daytime and reddish in the evening, and the earth as viewed from space becomes bluish. For the earth viewed from space, clouds also play an important role in creating realistic images. Our method realizes a fast display of these effects. For the earth's atmosphere, we propose a method for displaying the atmospheric effects viewed not only from the ground, but also as seen from space. Note that the rendering of the atmosphere viewed from the ground is the same as rendering the sky. Furthermore, a method for rendering clouds as viewed from space is developed to create realistic images of the earth.

The rendering processes are accelerated by using graphics hardware. The intensities of the scattered light are computed in a pre-process and stored in look-up tables. These tables are used as textures. Next, the images are created by hardware-accelerated volume rendering techniques[6]. Dobashi et al. have already proposed a method for rendering light beams by applying volume rendering techniques[5, 7, 8]. This paper discusses the problems of the approach taken by

these previous methods and addresses them. The differences between this paper and those previous ones are: (1) the accurate sampling of shadows and the use of texture for efficient evaluation of the intensity of scattered light and (2) an extension to the rendering of the sky and the earth as viewed from space. The proposed method improves the image quality significantly without increasing the rendering time. Our method realizes a fast rendering of the light beams produced by point and/or infinite light sources. The proposed method also realizes fast rendering of the earth's atmosphere.

2. Previous Work

Many methods have been developed for generating images by taking into account the scattering and absorption of light due to small particles in the air[1, 2, 3, 4, 9, 10, 11]. Although these methods can create realistic images, most of these employ a ray-tracing algorithm. Therefore, it takes several minutes to create a single image.

To address the problem, hardware-accelerated methods have been developed. Since the rendering of atmospheric effects can be considered as a kind of volume rendering, it is natural to apply the techniques developed for volume rendering. Many of them use hardware texture mapping functions to render volumes defined on voxels[12, 13, 14, 15, 16, 17]. In these methods, the particle density is stored at each voxel. Then the volume data is transferred to texture memories as a 2D/3D texture. These methods have already been applied to rendering smoke and clouds. The volume data, however, consume a large amount of texture memory. Furthermore, for rendering light beams, sharp edges exist in the atmosphere due to the luminous intensity of the light source and shadows of objects. It is very difficult to capture the edges by using a voxel-based approach. To address the problem, methods for rendering shafts of light have been proposed[5, 7, 8, 18, 19]. However, these still have problems in sampling shadows, so artifacts appear in the resulting images. Keller et al. presented a general and elegant solution to reduce the artifacts due to the sampling errors[20]. However, our method achieves a faster image generation for the specific problem treated in this paper, that is, the atmospheric scattering. Mech proposed an altenative method for displaying shafts of light by rendering boundaries of the light beams represented by polygons[21]. However, the method cannot display the shadows of objects in the atmosphere. For the purpose of scientific visualization, hardware-accelerated methods for rendering the earth as viewed from space have been proposed[22, 23, 16]. In these methods, however, realistic images are not generated since the physical properties of the light scattering due to the earth's atmosphere are not taken into account. Our method takes these into account and accelerates the computation of the scattering of light by using two-dimensional textures mapping functions.

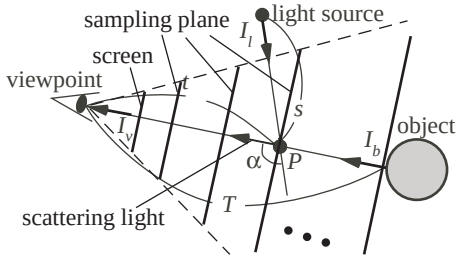


Figure 1: *Calculation of atmospheric scattering of light.*

3. Overview of Our Method

For rendering light beams, we assume the density of atmospheric particles is constant. For the earth's atmosphere, the shape of the earth is assumed to be a sphere and the density of particles decreases exponentially according to their height above the ground. Multiple scattering is important for atmospheric scattering but is time-consuming. Our method approximates multiple scattering as a constant ambient term.

Our method utilizes the hardware-accelerated volume rendering technique[5, 6, 7, 8]. As shown in Fig. 1, planes are placed in front of the screen, and they are subdivided into a mesh. We call the planes sampling planes. The atmospheric scattering effects are rendered by drawing the sampling planes. The intensities of the light scattered at points on the planes are obtained by using the look-up tables computed in a preprocess. The look-up tables are used as textures and are mapped onto the sampling planes. The intensity of scattered light at a point depends on the light source position, the viewing direction, and so on. Therefore high (three- to five-) dimensional look-up tables are required, and it requires a large amount of memory. To address this, we propose the method to store the intensity in two-dimensional textures. Furthermore, sub-planes are introduced for the precise sampling of object shadows and the luminous intensity distribution of the light source.

The above method is extended to the rendering of the sky. To represent the earth as viewed from space, however, it is not appropriate to use the method described in Fig. 1. In this case, the atmosphere is considered as a very thin spherical layer, and therefore a large number of planes are required for accurate sampling of the intensities of the scattered light, which results in increased computation time. Therefore, sample spheres are generated around the earth. The centers of the sample spheres coincide with the earth's center, and we then use the sample spheres instead of the sampling planes. Furthermore, clouds viewed from space are rendered to enhance the reality.

4. Rendering of Light Beams

Applying the method in Fig. 1 to rendering light beams in a straightforward way causes serious problems due to the limitations of the available graphics hardware. In this section,

the problems are firstly described in detail and then solutions to them are proposed. In the following, let us assume that the number of light sources is one for the sake of simplicity. When there are multiple light sources, the intensity of the light reaching the viewpoint is obtained by summing the contribution from each light source.

4.1. Problems in Rendering Light Beams

As shown in Fig. 1, the intensity of light reaching the viewpoint, I_v, is computed by summing the intensity of light reflected from an object and the intensities of light scattered by atmospheric particles. Note that Fig. 1 assumes a point light source. Since we assume the density of the particles is a constant ρ_c, I_v is given by:

$$I_v(\lambda) = I_b(\lambda)g(T,\lambda) + I_s(\lambda) + I_a(\lambda), \quad (1)$$

$$I_s(\lambda) = \int_0^T I_l(t,\lambda)H(t)g(t,\lambda)dt, \quad (2)$$

$$g(t,\lambda) = \rho_c F(\alpha,\lambda)\exp(-\beta\rho_c(s+t))/s^2, \quad (3)$$

where I_b is the intensity of the object, λ is a wavelength, T is the distance between the viewpoint and the object, I_l is the intensity of light emitted from the light source toward point P in Fig. 1 (I_l is obtained by using the luminous intensity distribution of the light source, or light map[25]). Without loss of generality, we can assume I_l is always less than 1.0. s and t are the distances from point P to the light source and the viewpoint respectively, I_a is the ambient term that approximates the multiple scattering, F is a phase function of the particles, and α is the phase angle (see Fig. 1). H is a visibility function that returns the answer 1 if the light source is visible from point P, and otherwise returns 0 if it is not. g is a function of the attenuation ratio of the intensity of light reaching the viewpoint after it is scattered at point P, and β is the extinction coefficient. Note that g is given by $g(t,\lambda) = \gamma\exp(-\beta\rho_c t)$ for an infinite light source, where γ is a constant that represents the attenuation ratio of light. In Eq. 1, the second term, I_s, is directly related to the rendering of the light beams. This subsection discusses the problems in computing I_s. One simple approach to rendering light beams is to apply the traditional texture-based volume rendering technique[5, 6, 7, 8, 17]. First, the intensities of the scattered light are computed at the vertices of the mesh representing each sampling plane and stored as colors of the vertices. Next, the planes are drawn and the intensities are accumulated in the frame buffer by using color-blending functions. This simple approach, however, causes the following problems.

In general, the values of the intensity distribution of the light source and the visibility function H change rapidly. Therefore, a lot of planes are required to achieve accurate sampling of these functions. However, using many planes results in poor images due to color quantization errors. Colors assigned at the vertices are stored with a limited precision (8-bit or 12-bit on high-end machines) in most available graphics hardware. Therefore, quantization errors are accumulated in proportion to the number of the sapling planes used. In our experiment, we found that the attenuation functions, g, are prone to be influenced by the quantization errors. The reason for this is that these functions are very small and quantized to zero in most cases. This problem may be resolved in the future when the graphics hardware with much higher bit depth becomes available. Even so, increasing the number of the planes is not desirable, since the rendering time is proportional to the total number. In addition, the sampling planes have to be subdivided into a finer mesh for the accurate evaluation of the intensity of the scattered light. This also results in significant increase in the rendering time.

4.2. High Quality Rendering

There are two key ideas in our approach. One is to use texture to evaluate the intensity of the scattered light at points on the sampling planes, and the other is to use sub-planes for the accurate sampling of shadows and the luminous intensity distribution of the light source. We first investigate Eq. 2 in detail before explaining the proposed method.

I_s is computed numerically by subdividing the integral of Eq. 2 into n sub-segments. The sub-segments are of equal length, that is, Δt. The kth sub-segment is expressed as $[t_k, t_k + \Delta t]$. Eq. 2 is rewritten as follows.

$$I_s(\lambda) = \sum_{k=1}^{n} \Delta I_s(t_k,\lambda), \quad (4)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda)H(t)g(t,\lambda)dt. \quad (5)$$

The functions on the right-hand side of Eq. 5 are classified into two groups. One group consists of functions that require a high sampling ratio (i.e., I_l and H). The rest of them are included in another group. Eq. 5 is approximated by the product of these groups, that is,

$$\Delta I_s(t_k,\lambda) \approx f_h(t_k,\lambda) \times f_l(t_k,\lambda), \quad (6)$$

$$\begin{cases} f_h(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda)H(t)dt \\ f_l(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} g(t,\lambda)dt \end{cases}. \quad (7)$$

Eq. 5 is approximated very well by Eq. 6 since f_l changes smoothly. Moreover, f_h does not include g, which is sensitive to quantization errors. In particular, the errors do not occur in evaluating the visibility function H since it is a two-valued function (0 or 1). So, f_h can be sampled at a shorter interval than Δt. On the other hand, f_l includes g. f_l is evaluated precisely by using texture mapping. This prevents the quantization errors from accumulating. In the following, the methods for evaluating f_l, f_h are proposed.

4.2.1. Textures for Intensity of Scattering

This subsection describes the creation of textures for f_l. Let us explain our algorithm for the point light source only. It

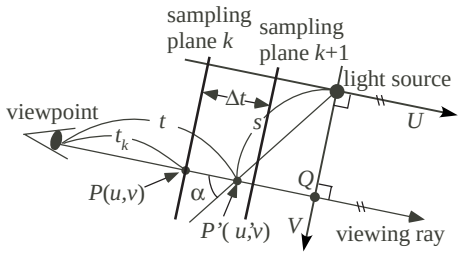


Figure 2: *Creating the textures for light beams.*

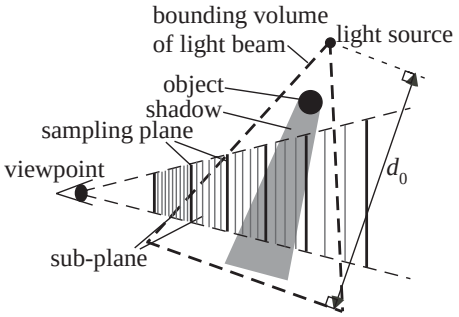


Figure 3: *Rendering light beams using sub-planes.*

is straightforward to extend the method for an infinite light source. f_l is given by the following equation.

$$f_l(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} \rho_c F(\alpha,\lambda) \frac{\exp(-\beta\rho_c(s+t))}{s^2} dt. \quad (8)$$

The basic idea of our method is to store the integrated values expressed by the above equation in a texture. This idea is similar to the pre-integrated volume rendering method proposed by Engel et al[24]. To store the integrated value in Eq. 8, high-dimensional tables are required since f_l is a function of the light source position, the viewpoint, and the direction of the viewing ray. However, in the case where there is a uniform density, a simpler expression exists. As shown in Fig. 2, let us assume a local coordinate system, UV, with its origin at the position of the light source. The U axis is parallel to the viewing ray and it passes through the light source position. The V axis is perpendicular to the U axis, and it passes through a point Q on the viewing ray that is closest to the light source. Using this coordinate system, let us denote the coordinate of point P on sampling plane k as (u,v). The distance from the viewpoint to P is t_k (see Fig. 2). The coordinate of point P' is denoted by (u',v). t is the distance between the viewpoint and point P'. Then, the following relationships are obtained: $t = u' - u + t_k$, $\cos\alpha = -u'/s$, $s^2 = u'^2 + v_k^2$. The exponential term in Eq. (8) is rewritten as:

$$\begin{aligned}\exp(-\beta\rho_c(s+t)) &= \exp(-\beta\rho_c(\sqrt{u'^2+v^2}+u'-u+t_k)) \\ &= \exp(-\beta\rho_c(\sqrt{u'^2+v^2}+u'-u)) \\ &\quad \times \exp(-\beta\rho_c t_k).\end{aligned}$$

As a result, f_l is expressed by:

$$f_l(u,v,\lambda) = c_k q(u,v,\lambda), \quad (9)$$

where,

$$\begin{aligned}c_k &= \exp(-\beta\rho_c t_k), \\ q(u,v,\lambda) &= \int_u^{u+\Delta t} \rho_c F(\cos^{-1}(-u'/\sqrt{u'^2+v^2}),\lambda) \\ &\quad \times \frac{\exp(-\beta\rho_c(\sqrt{u'^2+v^2}+u'-u))}{u'^2+v^2} du'\end{aligned}$$

c_k depends only on the distance from the viewpoint to the points on the sampling plane. c_k is evaluated by assigning it as the color of vertices of the mesh representing the plane. $q(u,v,\lambda)$ is computed in a pre-process step for different values of u and v. The sampling interval Δt is determined by the intersection of the ray with the sampling planes, and so Δt is different from pixel to pixel theoretically. In our implementation, however, Δt is set to a constant, that is, the interval of the sampling planes. This makes it possible to store $q(u,v,\lambda)$ in a two-dimensional look-up table. One advantage of this expression is that f_l is evaluated precisely, since the table stores the integrated values. The tables are used as the texture. λ is sampled at wavelengths that correspond to the RGB components. The remaining issues are how to determine the value of Δt (i.e. the interval of the sampling planes) and the number of the planes.

In most graphics hardware, values stored in the texture are clamped from zero to a certain value I_{max} (usually, I_{max} is 1.0). So, we determine Δt so that the values in the table of q do not exceed I_{max}. In addition, a maximum limit, Δ_{max}, is specified since the use of a too large value of Δt makes the approximation of Eq. 6 poor. First, Δt is set to a small value, Δ_{min}, and the table of q is computed. We extract the maximum value from the table. Then, Δt is incremented by Δ_{min}, and the maximum value is computed again. This process is repeated until Δt reaches Δ_{max} or the maximum value exceeds I_{max}. The resulting value is used as Δt.

The number of the sampling planes is automatically computed at every frame. We first define a bounding volume of the light beam (see Fig. 3). We use a pyramid as the bounding volume. The apex of the pyramid is at the position of the light source and the bottom of the pyramid is placed at a distance d_0 from the light source. d_0 is specified by the user. Then, the nearest and farthest points of the pyramid from the viewpoint are computed. The planes are generated between these two. When the distances from the viewpoint to these points are t_{min} and t_{max}, the number of the planes is $\lceil (t_{max} - t_{min})/\Delta t \rceil + 1$.

4.2.2. Precise Sampling using Sub-planes

This subsection describes the method for computing f_h. As shown in Fig. 3, the basic concept is to insert sub-planes between the sampling planes (or the main planes). Each sub-plane is represented by a single quadrilateral. The sub-planes

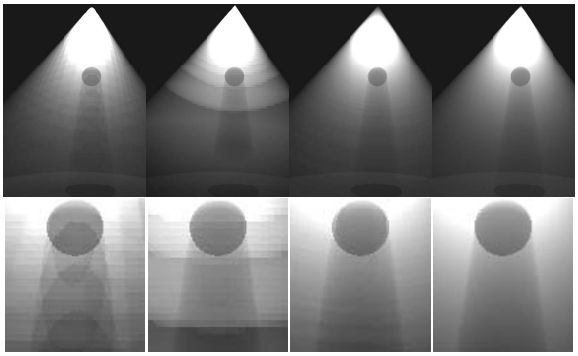


Figure 4: *Examples of artifacts. Bottom column shows the close-up around the sphere. (a) shows sampling errors of shadows, (b) quantization errors of intensity of scattered light, (c) artifacts are removed by our method, (d) a ray-traced solution.*

do not need to be represented by the mesh. As shown in Fig. 3, the appropriate number of the sub-planes is determined adaptively depending on the intensity of the scattered light. Let us assume that m_k sub-planes are generated between plane k and $k+1$. Then, f_h is given by:

$$f_h(t_k,\lambda) = \frac{1}{m_k}\sum_{j=0}^{m_k-1} H(t_k+j\Delta r)I_l(t_k+j\Delta r,\lambda), \quad (10)$$
$$(\Delta r = \Delta t/(m_k-1)).$$

The computation of Eq. 10 is accelerated by the graphics hardware. First, the visibility term H is obtained by computing the shadows on the sub-planes by a hardware shadow mapping technique[25]. Next, I_l is obtained by projecting a light map texture onto the sub-planes by using a projective texture mapping technique[25]. After these processes, the sub-planes are rendered and their colors are accumulated into the frame buffer. Note that only a single quadrilateral is rendered for each sub-plane. The pixel values of the resulting image are equal to f_h. This image is stored as a texture. When creating the images, this texture and the texture for f_l described in Section 4.2.2 are mapped onto the sampling planes using the multiplicative texture mapping function.

The number of the sub-planes, m_k, is determined as follows. From Eqs. (6) and (10), the contribution, ξ_j, of sub-plane j to the pixel intensity is estimated from the following equation.

$$\xi_j = H(t_k+j\Delta r)I_l(t_k+j\Delta r)\,f_l(t_k,\lambda)/m_k \le f_l(t_k,\lambda)/m_k.$$

The inequality in the above equation is due to the fact that $H \le 1$ and $I_l < 1$. If $f_l(t_k,\lambda)/m_k$ is less than a small value ε, we cannot expect sub-plane j to significantly improve the quality of the resulting image. Therefore, it is natural to select the number m_k so that $f_l(t_k,\lambda)/m_k$ is equal to or greater than ε. To implement this idea, we first shoot a ray toward the center of the screen and compute the distance, d_k, from

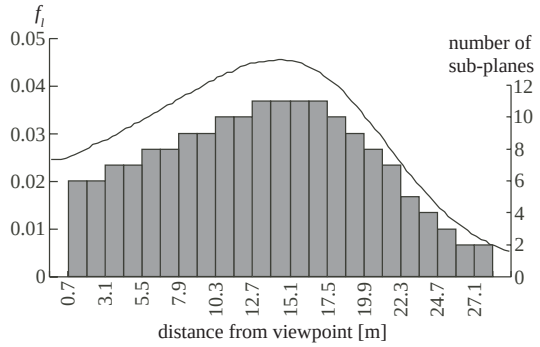


Figure 5: *The distribution of f_l and the number of sub-planes. The curved line shows the distribution of f_l and the bar chart shows the number of sub-planes for each sub-segment.*

the viewpoint to the intersection of the ray with each sampling plane. Then, the minimum number that satisfies the following condition is used as m_k.

$$\max_{\lambda} f_l(d_k,\lambda)/m_k \ge \varepsilon,$$

where ε is a threshold specified by the user. $f_l(d_k,\lambda)$ is obtained by the table for f_l described in the previous section.

4.3. Experimental Result

In this section, the validity of the proposed method is verified by applying it to a simple example where a sphere is illuminated by a single spotlight as shown in Fig. 4. In Fig. 4, the images in the bottom column show the close-up views of the images in the top column. The spotlight is placed at 20 [m] high. Figs. 4(a) and (b) are generated using the previous method[7]. We used a desktop PC (Athlon 1.7 GHz) with a NVIDIA GeForce3. The size of each image is 300×450. In Fig. 4(a), the number of sampling planes is 40. It took 0.13 [sec.] to render this image. It is obvious that the shadows are not sampled precisely. In Fig. 4(b), the number of the planes is increased, 160. However, pseudo-contours now appear due to the quantization errors. The rendering time is also increased to 0.50 [sec]. In these images, each plane is represented by a 60×90 mesh to sample the intensity of the scattered light accurately. This also increases the rendering time. Fig. 4(c) is generated by the proposed method. Clearly, the quality of the image is improved. The rendering time is still fast, 0.08 [sec.]. In this case, the 10×15 mesh is sufficient for each sampling plane since the intensity of the scattered light is evaluated accurately on a pixel basis by a hardware texture mapping function. For comparison, Fig. 4(d) is generated by using the ray tracing method[1]. It took 29 seconds. The average error in intensities between Figs. 4(c) and (d) is 1.5 [%]. Fig. 5 shows the distribution of the function f_l on the ray passing through the center of the screen. The horizontal axis indicates the distance from the viewpoint. The sampling planes are placed at an interval of 1.2[m]. The

threshold ε to insert sub-planes (see Section 4.2.2) is set to 1 [%] of the maximum value I_{max} that can be handled by the graphics hardware (in this experiment, ε = 0.01 since I_{max} = 1.0). The bar chart in Fig. 5 shows the number of sub-planes. In total, the numbers of the main planes and sub-planes are 23 and 174, respectively. As shown in Fig. 5, the number of the sub-planes is increased where the value of f_l is big, that is, the intensity of the scattered light is strong.

Finally, we compare our method with the method proposed by Keller et al[20]. This method can also reduce the artifacts due to the sampling error. The basic idea of Keller's method is to create multiple images with different positions of the planes and accumulate them into the final image. We accumulate four images with forty planes to render the same scene as shown in Fig. 4. These numbers are chosen so that the sampling interval is approximately the same as the one used by our method. Note that the sizes of the four images are same as the one used by our method (i.e. 300×450), to make the number of samples per each pixel equal. Each plane is represented by the 15×10 mesh. Although Keller's method created the image with the same quality, it took 0.12 [sec.]. Our method is 50 % faster. The reason is as follows. Keller's method needs to draw 40 planes represented by the mesh for each of four images. In total, 160 planes represented by the mesh are rendered. In our method, 23 planes represented by the mesh and 174 sub-planes represented by the quadrilateral are rendered. Therefore, the number of polygons rendered is smaller in our method. Furthermore, our method does not generate sub-planes where the intensity of the scattered light is small. This also results in decrease in the rendering time.

The rendering speed of our method depends on the ratio of geometric throughput and pixel fill rate. Especially, for creating high resolution images, our method becomes slow. This is because the fill rate becomes the bottleneck. This problem is resolved by combining our method with the Keller's method. That is, the high resolution image is created by compositing multiple low resolution images with the interleaved sampling technique[20].

5. Rendering the Earth's Atmosphere

This section describes the method for rendering the earth's atmosphere. For the earth's atmosphere, however, we can ignore the shadows of objects. Therefore, we focus on how to create textures for the intensity of the scattered light. In the following, we will first describe the principle of the intensity calculation for the atmospheric scattering, and then our methods for rendering the earth's atmosphere are proposed.

5.1. Intensity Calculation for Atmospheric Scattering

The earth's atmosphere consists of air molecules and aerosols. Aerosols are particles of dusts floating in the air. The scattering properties of the air molecules and

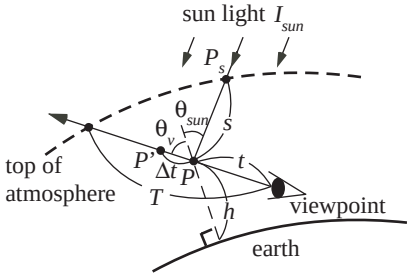


Figure 6: *Rendering of atmosphere viewed from ground.*

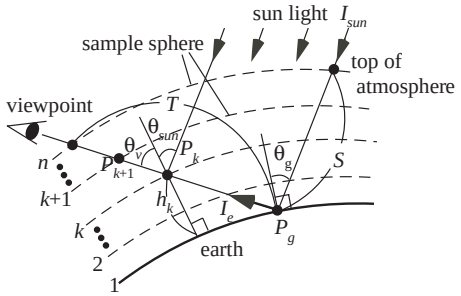


Figure 7: *Rendering of atmosphere viewed from space.*

the aerosols obey Rayleigh and Mie scattering theories, respectively[4]. Figs. 6 and 7 show the principles of the intensity calculation. The viewpoint is on the ground in Fig. 6 or is in space in Fig. 7. The intensity of light reaching the viewpoint I_v is given by Eq. 11[10].

$$I_v(\lambda) = I_e(\lambda)g_v(T,\lambda) + I_{sun}(\lambda)\int_0^T R(\alpha,t,\lambda)g_l(s,\lambda)g_v(t,\lambda)dt, \quad (11)$$

where T is the distance to the top of the atmosphere (see Fig. 6) or to the ground (see Fig. 7). In Eq. 11, subscripts "r" and "m" represent the air molecules (Rayleigh scattering) and the aerosols (Mie scattering), respectively. I_e, R, g_v, and g_l are given by the following equations.

$$\begin{aligned} I_e(\lambda) &= I_{sun}(\lambda)r(\lambda)g_l(S,\lambda)\cos\theta_g, \\ R(\alpha,t,\lambda) &= K_r(\lambda)\rho_r(t)F_r(\alpha) + K_m(\lambda)\rho_m(t)F_m(\alpha), \\ g_v(t,\lambda) &= \exp(-\tau(t,\lambda)), g_l(s,\lambda) = \exp(-\tau(s,\lambda)), \end{aligned}$$

where I_e is the intensity of light reflected on the ground, I_{sun} is the intensity of the sunlight, r is the reflectivity at point P_g on the ground, θ_g is the angle between the normal vector at P_g and the light vector, and S is the distance between P_g and the top of the atmosphere (see Fig. 7). I_e is 0 if the viewing ray does not hit the earth. g_v is a function of the attenuation ratio between the viewpoint and point P, and g_l is a function of the attenuation ratio of the sun light reaching point P. ρ_r and ρ_m are the densities of the atmospheric particles, and t is the optical length. F_r and F_m are phase functions. For more detail, see[4].

5.2. Rendering Sky

In a similar way to the method for the light beams, the integral term in Eq. 11 is evaluated by subdividing it into n sub-segments. The length of each sub-segment is Δt.

First, let us discuss the attenuation ratio of the sunlight reaching point P on the viewing ray (see Fig. 6). Let us assume that the distance between the viewpoint and point P is t, the height of point P is h, and the angle between the sunlight direction and the vertical direction at P is θ_{sun}. Then, the attenuation ratio is expressed as a function of the height h and the angle θ_{sun} since the density of the atmospheric particles depends only on their height from the ground (see Appendix A.1). That is, $g_l(s,\lambda) = g_l(h,\theta_{sun},\lambda)(= \exp(-t(PP_s,\lambda)))$. Note that $g_l(h,\theta_{sun},\lambda) = 0$ if the sun as viewed from point P is behind the earth. $g_l(h,\theta_{sun},\lambda)$ is computed in a pre-process and stored in a texture.

Next, let us consider the ray that passes through sub-segment k, $[t_k, t_k+\Delta t]$. Eq. 11 is rewritten as follows.

$$\begin{aligned} I_v(\lambda) &= I_{sun}(\lambda) g_l(0,\theta_g,\lambda)\cos\theta_g \times \prod_{j=0}^{n-1} \Delta g_v(t_j,\lambda) \\ &+ I_{sun}(\lambda) \sum_{k=0}^{n-1} \Big\{ g_l(h_k,\theta_k,\lambda)[F_r(\alpha,\lambda)\Delta I_r(t_k,\lambda) \\ &+ F_m(\alpha,\lambda)\Delta I_m(t_k,\lambda)] \prod_{j=0}^{k-1} \Delta g_v(t_j,\lambda) \Big\}, \end{aligned} \tag{12}$$

where $\Delta I_r(t_k,\lambda)$ and $\Delta I_m(t_k,\lambda)$ are the intensities of scattering and $\Delta g_v(t_j,\lambda)$ is the attenuation ratio of the segment $[t_j, t_j+\Delta t]$. These are given by the following equations.

$$\begin{aligned} \Delta I_r(t_k,\lambda) &= \int_{t_k}^{t_k+\Delta t} K_r(\lambda)\rho_r(t)\exp(-\tau(t-t_k,\lambda))dt, \\ \Delta I_m(t_k,\lambda) &= \int_{t_k}^{t_k+\Delta t} K_m(\lambda)\rho_m(t)\exp(-\tau(t-t_k,\lambda))dt, \\ \Delta g_v(t_k,\lambda) &= \exp(-\tau(t_k \to t_k+\Delta t,\lambda)), \end{aligned}$$

where $\tau(t_k \to t_k+\Delta t,\lambda)$ indicates the optical length of the segment $[t_k, t_k+\Delta t]$. ΔI_r, ΔI_m, and Δg_v can also be stored in the look-up tables. Let us consider a ray that passes through segment PP' as shown in Fig. 6. The angle between the vertical direction at P and the segment PP' is θ_v. Then, in a similar way to g_l, ΔI_r, ΔI_m, and Δg_v are expressed as functions of the height h and the angle θ_v (see appendix A.2), that is, $\Delta I_r(t,\lambda) = \Delta I_r(h,\theta_v,\lambda)$, $\Delta I_m(t,\lambda) = \Delta I_m(h,\theta_v,\lambda)$, $\Delta g_v = \Delta g_v(h,\theta_v,\lambda)$. These functions are also pre-computed. The interval Δt is determined in a similar way to the method for the light beams (see Section 4.2.1).

The look-up tables for $g_l(h,\theta_{sun},\lambda)$, $\Delta I_r(h,\theta_v,\lambda)$, $\Delta I_m(h,\theta_v,\lambda)$, and $\Delta g_v(h,\theta_v,\lambda)$ are used as textures. The wavelength λ is sampled at RGB. These textures are mapped onto the sampling planes. The phase functions are computed at the vertices of the planes and interpolated by the Gouroud shading function. Then the planes are rendered in back to front order. Only the planes that intersect with the atmosphere are generated at the interval Δt. The images are generated by computing Eq. 12 by hardware color blending functions.

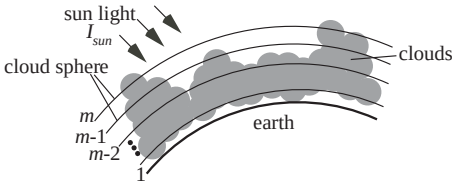


Figure 8: *Rendering of clouds viewed from space.*

5.3. Rendering the Atmosphere as Viewed from Space

As shown in Fig. 7, we use sample spheres for rendering the atmosphere as viewed from space. The integral term in Eq. 11 is evaluated by using the sample spheres. The height of sample sphere k above the ground is h_k and the viewing ray hits sample sphere k at P_k (see Fig. 7). The intensity of the light reaching P_k is obtained by multiplying the intensity of the sunlight and the attenuation function $g_l(h_k,\theta_{sun,k},\lambda)$ as described in Section 5.2. In the following, we will discuss the scattering, $\Delta I'_{r,k}$ and $\Delta I'_{m,k}$ and the attenuation ratio, $\Delta g'_{v,k}$, due to particles in P_kP_{k+1}.

Let us assume that the angle between the vertical direction at P_k and the direction of P_kP_{k+1} is θ_v (see Fig. 7). Then, $\Delta I'_{r,k}$, $\Delta I'_{m,k}$, and $\Delta g'_{v,k}$ are expressed as functions of θ_v (see appendix A.2). Therefore, these are stored as a one-dimensional texture for each sample sphere. Eq. 12 is evaluated by using the textures for $\Delta I'_{r,k}(\theta_v,\lambda)$, $\Delta I'_{m,k}(\theta_v,\lambda)$, and $\Delta g'_{v,k}(\theta_v,\lambda)$ instead of $\Delta I_r(\theta_v,\lambda)$, $\Delta I_m(\theta_v,\lambda)$, and $\Delta g_v(\theta_v,\lambda)$. The images are generated by rendering the sample spheres mapped with these textures. We specify the texture coordinates on a per-vertex basis. Westermann et al. proposed an accurate method[16] for mapping textures onto spheres by specifying the texture coordinates on a per-pixel basis with the pixel **texgen** OpenGL extension. In our case, however, assigning the texture coordinates on a per-vertex basis is sufficient since the intensity of the scattered light is very smooth. The phase functions are computed at the vertices of the mesh of each sphere.

5.4. Rendering of Clouds

For rendering the earth viewed from space, three-dimensional clouds are generated from satellite images. Methods developed in the field of remote sensing enable us to determine the height of a cloud top[26]. Since any estimation of the density inside the clouds and the height of the cloud base are difficult problems to solve, the density and the height are simply assumed to be a constant and specified by the user. Then, as shown in Fig. 8, m cloud spheres are

generated. For each cloud sphere, a texture $T_i (i = 1, \cdots, m)$ is mapped. Let us call T_i cloud textures in the following. The cloud texture represents the density distribution on each cloud sphere. Each element of the cloud texture T_i stores the density obtained from the satellite image. The alpha channels of the cloud texture store the attenuation ratio of the light that passes through the clouds as represented by cloud sphere i.

The rendering method for clouds is as follows. In the following, let us assume for the sake of simplicity that there is no atmosphere. The attenuation due to the atmosphere is computed by using the texture for the attenuation function $g_l(h, \theta_{sun}, \lambda)$. The rendering of clouds consists of two steps. The first step is to compute the attenuation ratio of the sunlight reaching points on the cloud spheres. To do this, the camera position is placed at the position of the sun and cloud spheres are rendered in descending order based on their distances from the center of the earth. After drawing cloud sphere i, the pixel values in the frame buffer represent the attenuation ratio between the sun and points on the cloud sphere. This image is stored as the attenuation texture G_i. These processes are repeated for all cloud spheres.

Next, in the second step, images are generated by using the cloud texture and the attenuation texture. First, cloud texture T_i and attenuation texture G_i are mapped onto cloud sphere i by using the multiple texture mapping function. Next, the cloud spheres are rendered in back to front order, that is, the inmost sphere is rendered first. The phase function of the cloud particles is evaluated at the vertices of the mesh of the cloud spheres.

6. Results

In this section, several images generated by our method are demonstrated†. Fig. 9 shows examples of light beams rendered by the method described in Section 4. In Fig. 9(a), the proposed method is applied to the lighting design of a studio. Fig. 9(b) shows typical shafts of light caused by sunlight. The sunlight passes through the stained glass windows of a church. Next, Figs. 10 and 11 show examples of rendering the earth's atmosphere. Fig. 10(a) is an example of rendering the sky. This image also demonstrates shafts of light due to mountains that shut out parts of the sun's rays. Figs. 10(b), (c), and 11 show the earth viewed from space. In Fig. 10(b), realistic clouds as well as the atmosphere are rendered. In Fig. 10(c), a thin atmospheric layer covers the earth. Fig. 11 shows a sequence of images of the sunrise viewed from space. The atmosphere exhibits beautiful color variations from red to blue. In Fig. 10(b), the method described in Section 5.3 is used, since the viewpoint is far from the earth. In Figs. 10(c) and 11, the method described in Section 5.2 is used, since the viewpoint is near to the surface.

† see color section for Figs. 9(b), 10(c), and 11

Table 1: *Computation time.*

	time [sec.]	No. of planes [sub-planes]
Fig. 9(a)	0.32	9[309], 13[174], 16[162]
Fig. 9(b)	0.12	30[263]
Fig. 10(a)	0.16	100(sky), 30(shaft)[135]
Fig. 10(b)	0.06	10(sample spheres)
Figs. 10(c) & 11	0.10	300

Some of software packages approximate the atmosphere by using spheres with a simple hue shift through the atmosphere (e.g. [27]). However, it is difficult to simulate the color variations shown in Figs. 10 and 11 by using a simple hue shift.

Table 1 shows computation times. We use the same computer as the one described in Section 4.3. The size of each image is 720×480. The numbers of the main planes and sub-planes are also shown in the table. For Fig. 9(a), the numbers for each of three spotlights are shown. For Fig. 10 (c), the number of the sample sphere is shown. In Figs. 9 and 10(a), the threshold to insert the sub-planes is 0.01 (see Section 4.2). In Fig. 10(a), 100 planes are used for computing the sky colors and 30 planes are used for rendering the shafts of light due to the mountains. In Fig. 10(c), the number of cloud spheres used is 5. As shown in Table 1, our method can create the images at interactive rates. The sub-planes and 3D clouds can be omitted for real-time previews. For example, Fig. 9(b) is rendered in 90 fps (frame per second) without the sub-planes and Fig. 10(c) is rendered in 30 fps without 3D clouds.

7. Conclusion

We have proposed a fast rendering method for atmospheric scattering effects. In the method, two-dimensional textures that store the intensities of the scattered light are pre-prepared. The luminous intensity distribution of the light source and the object shadows are sampled precisely by using the sub-planes. We have also realized the efficient rendering of the earth's atmosphere. The sky and the earth as viewed from space are rendered almost in real-time. We have developed a method of rendering realistic clouds by using satellite images.

In future work, we need to discuss the problem posed by the multiple scattering of light. For participating media with a high albedo, the realism of images might be enhanced by taking into account the multiple scattering. An efficient method for the multiple scattering has to be developed.

Acknowledgements

The authors would like to acknowledge Miss Keiko Sotome (Institute of Advanced Media Arts and Sciences) for her help in making the geometric data for the church and the opera house in Fig. 9. This research was partially supported by the Ministry of Education, Japan, Grant-in-Aid for Encouragement of Young Scientists (B) (2) 14750271, 2002.

References

1. T. Nishita, Y. Miyawaki, E. Nakamae, "A Shading Model for Atmospheric Scattering Considering Distribution of Light Sources," *Computer Graphics*, **21**(4):303–310, 1987.
2. N. Max, "Atmospheric Illumination and Shadows," *Computer Graphics*, **20**(4):117–124, 1986.
3. A. Preetham, P. Shirley, B. Smits, "A Practical Analytic Model for Daylight," *Proc. SIGGRAPH'99*, 91–199, 1999.
4. T. Nishita, T. Shirai, K. Tadamura, E. Nakamae, "Display of The Earth Taking into account Atmospheric Scattering," *Proc. SIGGRAPH'93*, 175–182, 1993.
5. Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita, "A Simple, Efficient Method for Realistic Animation of Clouds," *Proc. SIGGRAPH 2000*, 19–28, 2000.
6. B. Cabral, N. Cam, J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," *Proc. 1994 Simposium on Volume Visualization*, 91–98, 1994.
7. Y. Dobashi, T. Yamamot, T. Nishita, "Interactive Rendering Method for Displaying Shafts of Light," *Proc. Pacific Graphics 2000*, 31–37, 2000.
8. Y. Dobashi, T. Yamamot, T. Nishita, "An Accurate, Fast Method Using Graphics Hardware for Rendering Shafts of Light," *The Journal of The Institute of Image Information and Television Engineers*, **55**(7):362–370, 2001.
9. H. W. Jansen, P. H. Christensen "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps," *Proc. SIGGRAPH'98*, 311–320, 1998.
10. T. Nishita, Y. Dobashi, K. Kaneda, H. Yamashita, "Display Method of the Sky Color Taking into Account Multiple Scattering," *Proc. Pacific Graphics'96*, 117–132, 1996.
11. H. E .Rushmeier, K. E. Torrance, "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium," *Computer Graphics*, **21**(4):293–302, 1987.
12. U. Behrens, R. Ratering, "Adding Shadows to a Texture-based Volume Renderer," *Proc. 1998 Simposium on Volume Visualization*, 39–46, 1998.
13. E. Lamar, B. Hamann, K. I. Joy, "Multiresolution Techniques for Interactive Texture-Based Volume Visualization," *Proc. IEEE Visualization'99*, 355–362, 1999.
14. J. Stam "Stable Fluids," *Proc. SIGGRAPH'99*, 121–128, 1999.
15. M. Nulkar, K. Mueller, "Splatting with Sadows," *Proc. Volume Graphics 2001*, 35–49, 2001.
16. R. Westermann, T. Ertl "Efficiently Using Graphics Hardware in Volume Rendering Applications," *Proc. SIGGRAPH'98*, 169–177, 1998.
17. T. J. Cullip, U. Neumann "Accelerating Volume Reconstruction with 3D Texture Hardware," *Technical Report TR93-027, University of North Carolina, Chapel Hill* 1993.
18. M. Kawara, http://www.daionet.gr.jp/ masa/ ishadowmap/index.html
19. C. Everitt, http://www.r3.nu/ cass/shadowsandstuff/
20. A. Keller, W. Heidrich, "Interleaved Sampling," *Proc. Rendering Technique 2001*, 269–276, 2001.
21. R. Mech "Hardware-Accelerated Real-Time Rendering of Gaseous Phenomena," *Journal of Graphics Tools*, **6**(3):1–16, 2001.
22. C. A. Lee, C. Kesselman, "Near-Real-Time Satellite Image Processing: Metacomputing in CC++," *IEEE Computer Graphics and Applications*, **16**(4):79–84, 1996.
23. N. Max, R. Crawfix, D. Williams, "Visualization for Climate Modeling," *IEEE Computer Graphics and Applications*, **13**(4):34–40, 1993.
24. K. Engel, M. Klaus, T. Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading," *Proc. Graphics Hardware 2001*, 9–16, 2001.
25. M. Segal, C. Koroblkin, R. V. Widenfelt, J. Foran, P. E. Haeberli, "Fast Shadows and Lighting Effects Using Texture Mapping," *Computer Graphics*, **26**(2):249–252, 1992.
26. C. D. Rodgers, "Retrieval of Atmospheric Temperature and Composition from Remote Measurements of Thermal Radiation," *Review of Geophysics and Space Physics*, **14**:609–624, 1976.
27. http://ennui.shatters.net/celestia

A. Appendix

Let us assume a coordinate system with its origin at the center of the earth. In this system, the coordinates of point P in

Fig. 6 are assumed to be $(0,0,R_e+h)$, where R_e is the radius of the earth.

A.1 Attenuation of Sunlight

The optical length τ is expressed by the following equation.

$$\tau(t,\lambda) = \int_0^t \beta_r(\lambda)\rho_r(l) + \beta_m(\lambda)\rho_m(l)dl, \qquad (A.1)$$

where β_r and β_m are the extinction coefficients, and ρ_r and ρ_m are the densities of air molecules and aerosols, respectively. These are given by $\rho_r = \exp(-h/H_r)$ and $\rho_m = \exp(-h/H_m)$, where h is the height from the ground, and H_r and H_m are constants $(H_r = 7994[m], H_m = 1.2[km])$.

Assuming the height of the top of the atmosphere is H. A vector in the direction of the sun is given as $(0, \sin\theta_{sun}, \cos\theta_{sun})$. The length s of PP_s in Fig. 6 is:

$$s = -(R_e+h)\cos\theta_{sun} + \sqrt{(R_e+H)^2 - (R_e+h)^2\sin^2\theta_{sun}}. \qquad (A.2)$$

A coordinate of a point on PP_s is given by $(0, s'\sin\theta_{sun}, R_e + h + s'\cos\theta_{sun})$, where s' is the length between point P and the point on PP_s. The height of this point is given by

$$h_s(s') = \sqrt{s'^2\sin^2\theta_{sun} + (R_e+h+s'\cos\theta_{sun})^2} - R_e. \qquad (A.3)$$

Eqs. (A.1), (A.2), and (A.3) indicate that the optical length $\tau(s,\lambda)$ between PP_s is expressed as a function of h and θ_{sun}. So, g_l is also expressed as a function of h and θ_{sun}. That is,

$$g_l(h,\theta_{sun},\lambda) = \exp(-\int_0^s \beta_r(\lambda)\rho_r(h_s(s')) + \beta_m\rho_m(h_s(s'))ds'). \qquad (A.4)$$

A.2 Intensity of Light Scattering and Attenuation Ratio

In Fig. 6, let us consider a point on PP', denoted by Q. The coordinate of point Q is given by $(0, t'\sin\theta_v, R_e + h + t'\cos\theta_v)$, where t' is the length of PQ. The height of point Q, $h_v(t')$, is given by replacing s' and θ_{sun} in Eq. (A.2) with t' and θ_v, respectively. Therefore, $h_v(t')$ is a function of t', h, and θ_v. ΔI_r is equivalent to the integrated value of the scattering intensity between PP' in Fig. 6. The density at point Q and the optical length between PQ are computed by using $h_v(t')$. Therefore, ΔI_r is expressed as a function of h and θ_v. That is,

$$\Delta I_r(h,\theta_v,\lambda) = \int_0^{\Delta t} K_r(\lambda)\rho(h_v(t'))\exp(-\tau(h_v(t'),\lambda))dt'. \qquad (A.5)$$

Similarly, ΔI_m is expressed as a function of h and θ_v. The attenuation ratio Δg_v is obviously a function of h and θ_v since the optical length between PP' is a function of h and θ_v, that is,Similarly, ΔI_m is expressed as a function of h and θ_v. The attenuation ratio Δg_v is obviously a function of h and θ_v since the optical length between PP' is a function of h and θ_v, that is,

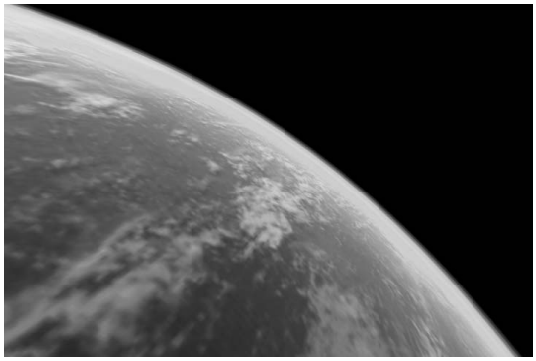
$$\tau(PP',\lambda) = -\int_0^{\Delta t} \beta_r(\lambda)\rho_r(h(t')) + \beta_m\rho_m(h(t'))dt'. \qquad (A.6)$$

In the case of the sample spheres, by replacing P and P' with P_k and P_{k+1} in Fig. 7, it is obvious that $\Delta I'_{r,k}$ $\Delta I'_{m,k}$, and $\Delta g'_{v,k}$ are expressed as functions of h and θ_v. The height from the ground is fixed at h_k for sample sphere k. So $\Delta I'_{r,k}$, $\Delta I'_{m,k}$, and $\Delta g'_{v,k}$ are functions of θ_k.

(a) Light beams caused by spotlights.

Figure 9: *Examples of rendering light beams.*

(a) Sky colors and shafts of light.

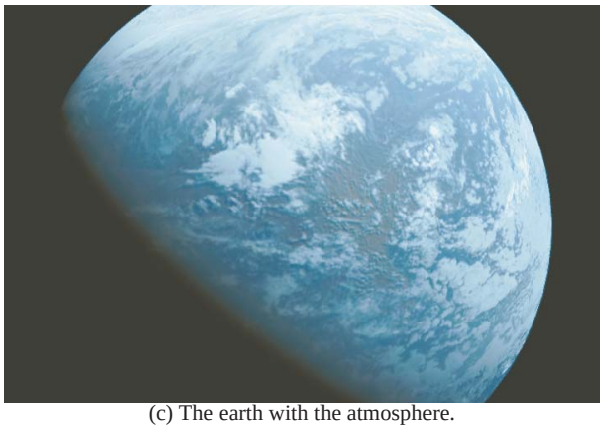


(b) The earth covered by the bluish atmosphere.

Figure 10: *Examples of rendering earth's atmosphere.*

(b) Light beams through stained glass windows.

Figure 9: Examples of rendering light beams.



(c) The earth with the atmosphere.

Figure 10: Examples of rendering earth's atmosphere.

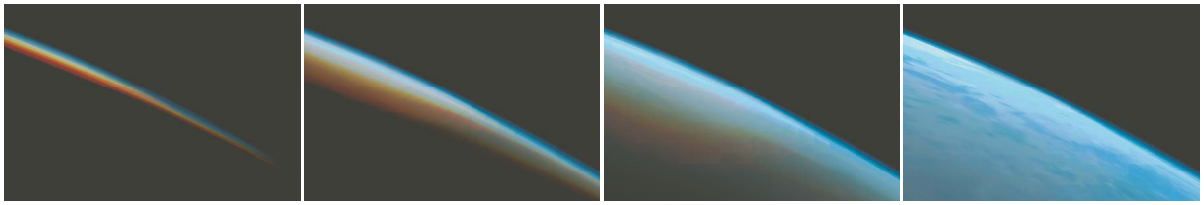


Figure 11: The sunrise viewed from space.