Volume xx (200y), Number z, pp. 1–11

An Efficient Method for Rendering Underwater Optical Effects Using Graphics Hardware

Kei Iwasaki[†], Yoshinori Dobashi[‡] and Tomoyuki Nishita [†]

[†] Department of Complexity Science and Engineering, The University of Tokyo, Tokyo, Japan
[‡] Department of Super Integrated Computer Systems, Hokkaido University, Sapporo, Japan
kei-i@is.s.u-tokyo.ac.jp, doba@nis-ei.eng.hokudai.ac.jp, nis@is.s.u-tokyo.ac.jp

Abstract

The display of realistic natural scenes is one of the most important research areas in computer graphics. The rendering of water is one of the essential components. This paper proposes an efficient method for rendering images of scenes within water. For underwater scenery, the shafts of light and caustics are attractive and important elements. However, computing these effects is difficult and time-consuming since light refracts when passing through waves. To address the problem, our method makes use of graphics hardware to accelerate the computation. Our method displays the shafts of light by accumulating the intensities of streaks of light by using hardware color blending functions. Making use of a Z-buffer and a stencil buffer accelerates the rendering of caustics. Moreover, by using a shadow mapping technique, our method can display shafts of light and caustics taking account of shadows due to objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Creating realistic images of natural scenes is one of the most important subjects in computer graphics. Accurate simulation of natural phenomena is required to generate realistic images. Of all the natural phenomena, the realistic rendering of water, such as the sea and lakes, is one of the most essential components, so many techniques have been developed to represent water accurately. The generation of realistic images relating to water can be split into two categories; the rendering of water as viewed from above the surface and rendering underwater scenes.

Compared with those methods designed for rendering water as viewed from above, there has been very little work focusing on rendering underwater images. There are several optical effects that are important in water, such as shafts of light, caustics, and shadows. Caustics are the light patterns that are formed on surfaces. Refracted light from waves in the water can converge and diverge, and this effect creates caustics and shafts of light due to light scattering by suspended particles. Scattering or absorption due to water molecules and suspended matter determine the color of the water. The representation of shadows due to objects within the water is necessary in order to increase realism.

This paper proposes a new method for fast rendering of realistic images of scenes within water. Recently, highly efficient graphics hardware has become available and therefore several different methods have been developed to generate realistic images of natural phenomena[2, 3, 6, 20, 22]. Our method makes use of this advanced graphics hardware.

Our method is based on Nishita's method[12]. We have extended his method in order to accelerate the computation by using graphics hardware. Our method can display shafts of light, caustics on objects and the shadows due to objects. The shafts of light are rendered by using hardware color blending functions. Caustics are calculated by making use of a stencil

buffer and a Z-buffer. A shadow map technique is used to create the shadows of the objects.

In this paper, previous work is discussed in Section 2. In Section 3 we describe our rendering method for shafts of light using graphics hardware. Section 4 deals with caustics, not only on planes but also on complex objects. In Section 5 we discuss the displaying of shadows using graphics hardware. Finally, examples and a conclusion are presented in Sections 6 and 7, respectively.

2. Previous Work

Research concerning the creation of realistic images related to water can be categorized into two types. First, we shall review the previous methods used for rendering water as viewed from above, and then various methods for rendering underwater scenes are discussed.

There are two main components used to create realistic images of water. One of these is the modeling of waves. Fournier and Reeves displayed coastal scenes[4] using Gerstner's wave model[5]. Peachey[14] presented Stokes' wave model, which could simulate the behavior of water approaching a sloping beach. Ts'o and Barsky proposed a "Wave-Tracing" method and modeled waves using Beta-spline interpolation[23]. Tessendorf proposed a statistical wave model, which synthesizes sine and cosine waves[21]. The model that we adopted is a statistical wave model, since this type of model makes it easy to handle the shapes of the waves (see Appendix).

The other component required for rendering images of water is the calculation of the color of the water. Kaneda et al.[8] proposed a method for rendering realistic water surfaces by taking into account the radiative transfer equation of light in water. Nishita et al.[11] obtained an analytical expression for calculating ocean color taking into account the single scattering of light. Premoze and Ashikhmin[15] recently presented a light transport approach. They simulated different ocean depths near islands and various types of ocean.

Methods for rendering caustics, shafts of light, and the color of the water itself have already been developed for underwater images. Shinya et al.[18] proposed an algorithm for displaying caustics. They calculated the illumination distribution on the surface in advance by using grid-pencil tracing. Watt[24] developed backward beam tracing. Recently, Brière and Poulin[1] displayed caustics by using a hierarchical light beam structure. These methods use a kind of ray tracing algorithm, so it takes too much time to generate images. Stam proposed a method for generating the texture of caustics[19]. His method, however, cannot generate caustics on any object directly. Kunimatsu et al.[9] presented a method of rendering caustics with a short turn-round time. They primarily describe a method for rendering caustics as viewed from above the water surface. Trendall and Stewart[22] displayed caustics by using graphics hardware. This method, however, could not deal with caustics on objects with complex shapes.

Displaying shafts of light is one of the most important elements of simulating the scattering of light, so many methods have been proposed to achieve this. Regarding shafts of light in the atmosphere, Nishita et al.[10] proposed a shading model for the scattering and absorption of light. Recently, Dobashi et al. presented a method for rendering shafts of light through gaps between clouds[2]. They also improved the method so that it can not only deal with parallel light sources, but also with point light sources[3]. As for shafts of light within water, Jensen et al.[7] displayed caustics and shafts of light using photon maps. In general, however, the method using photon maps takes too much time.

In 1994, Nishita and Nakamae[12] presented a method for displaying caustics, shafts of light, and the color of the water using an accumulation buffer. They subdivided the water surface into meshes and defined illumination volumes. The illumination volume is calculated by sweeping refracted vectors at each lattice point of the mesh. Then they used the scan line algorithm to calculate the intensities of the illumination volumes and stored them in the accumulation buffer. One of the advantages of this method is the ability to handle free-form surfaces, but it takes several minutes to create an image since the accumulation buffer used was not a graphics hardware buffer and the intensities of the illumination volumes at each scan plane have to be calculated.

We propose a fast method for rendering optical effects within water using advanced graphics hardware. In our method, we also use illumination volumes for displaying shafts of light. The illumination volume is divided into several volumes so that it is suitable for graphics hardware and the intensities of the shafts of light can be calculated by using hardware color blending functions. Caustics are also rendered by making use of illumination volumes. We determine the intersection area between the illumination volume and the objects by using a stencil buffer. Our method can handle the shadows due to objects by using the hardware-accelerated shadow map method.

3. Displaying Shafts of Light

The phenomenon of shafts of light within water, which arises from refracted light due to waves, is one of the

optical effects found within water. The incident light reaching the water's surface consists of two components: direct sunlight and skylight. In our method, we take sunlight into account and regard skylight as an ambient light.

3.1. Basic idea of the intensity calculation

Here we describe our method for calculating the intensity of light reaching the viewpoint. The scattering of light due to water particles has to be taken into consideration to display shafts of light.

Our method is an improved version of Nishita's method[12], so we explain this method briefly. Moreover, to simplify the explanation, we assume in this section that there are no objects within the water.

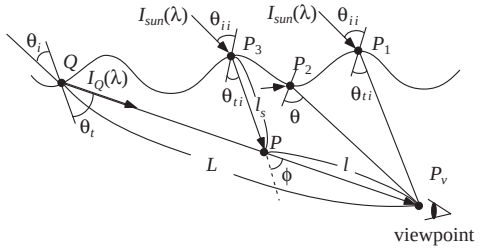


Figure 1: *Calculation of intensity reaching viewpoint.*

As shown in Fig. 1, when a viewpoint is located within the water, the intensity of the light $I_v(\lambda)$ originating from point Q on the water surface and reaching viewpoint P_v is calculated using the following equation.

$$I_v(\lambda) = I_Q(\lambda)\exp(-c(\lambda)L) + \int_0^L I_P(\lambda)\exp(-c(\lambda)l)dl, \quad (1)$$

where λ is the wavelength of light, L is the distance between P_vQ, l is the distance between PP_v, $c(\lambda)$ is the attenuation coefficient of light within water and $I_P(\lambda)$ represents the intensity of light scattered at point P. If the angle between the viewing ray and the normal to the water surface is less than the critical angle, the intensity I_Q is expressed by the following equation.

$$I_Q(\lambda) = T(\theta_i, \theta_t)(I_{sun}(\lambda)\delta(\theta_i, \theta_t) + I_{sky}(\lambda, \theta_i)), \quad (2)$$

where $T(\theta_i, \theta_t)$ is the Fresnel transmittance from an angle of θ_i to θ_t and $\delta(\theta_i, \theta_t)$ is a function that takes the value 1 when the direction of the refracted light is coincident with the direction of P_vQ, and is 0 when it is not.

If the angle between the viewing ray and the normal of the water surface is greater than the critical angle ($48.6[deg]$), the incident light is the reflected light at a point on the water surface (P_2 in Fig. 1). Therefore, the light that reflects at point P_2 reaches the viewpoint. We consider this reflected light as an ambient light. Then we multiply the ambient light and the attenuation ratio due to water particles between P_2P_v.

The intensities of the shafts of light are calculated by the integration term of Eq. (1). Let us denote the integration term as I_{shaft} for simplicity. To calculate I_{shaft}, we make use of the idea of *illumination volumes*[12].

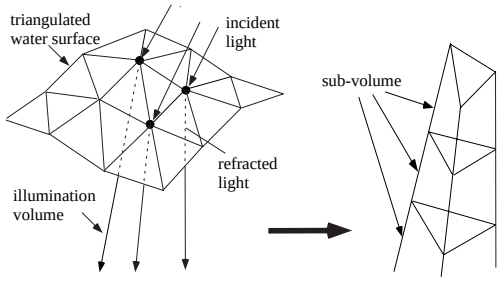


Figure 2: *Illumination volume and sub-volumes.*

As shown in Fig. 2, the water surface is subdivided into triangular meshes, and then the refracted vector at each mesh point is calculated. The volume that is defined by sweeping the refracted vector at each point of the mesh is called the *illumination volume* (see Fig. 2).

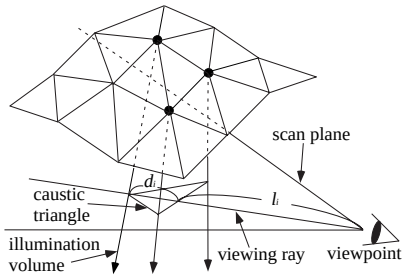


Figure 3: *scan plane and caustic triangle.*

In the previous method[12], illumination volumes were sliced by each scan plane and the intersection area between an illumination volume and the scan plane was defined as the *caustic triangle* (see Fig. 3). The integral of the scattered light is equal to the sum of the intensities of the caustic triangles that the viewing ray intersects. The second term of Eq. (1), I_{shaft}, is approximated by the following equation.

$$\int_0^L I_P(\lambda)\exp(-c(\lambda)l)dl = \sum_i I_{p_i}(\lambda)d_i\exp(-c(\lambda)l_i), \quad (3)$$

where $I_{pi}(\lambda)$ is the average intensity of the ith caustic triangle, d_i the intersected length of the ith triangle and the viewing ray and l_i is the distance between the ith caustic triangle and the viewpoint (see Fig. 3). The scattered intensity $I_P(\lambda)$ at point P is calculated

using the following equation.

$$I_P(\lambda) = (I_{sun}(\lambda)T(\theta_{ii},\theta_{ti})F_p\beta(\lambda,\phi) \times \exp(-c(\lambda)l_s) + I_a)\rho, \quad (4)$$

where $I_{sun}(\lambda)$ is the intensity of the light that is incident onto the water surface, T is the transmittance at point P_3 in Fig. 1 and θ_i and θ_t are the incident and transmitted angles, respectively. $\beta(\lambda,\phi)$ is the phase function, ρ the density, I_a the ambient light. F_p is the flux ratio between the intensity just beneath the water surface and at point on the caustic triangle, since the energy of light in a caustic triangle is inversely proportional to its area. l_s is the distance between P_3P.

Nishita et al. calculate I_{shaft} on a scan line basis. That is, I_{shaft} was calculated by slicing the illumination volumes by each scan plane and summing the intensities of the caustic triangles. A large number of illumination volumes are required to create realistic images. Therefore, their method is computationally very expensive. In the following subsections, we will describe our method of accelerating the computation of I_{shaft} by using graphics hardware.

3.2. Proposed method for shafts of light

The basic idea to render shafts of light is to display the shaded front faces of the illumination volume. The intensity of the illumination volume is calculated by integrating the scattered light along the intersection segment of the viewing ray with the illumination volume. However, the intersection test for each pixel requires a great deal of computational cost. So, as shown in Fig. 2, we subdivide each illumination volume into several volumes (we call these volumes *sub-volumes*) by horizontal planes. Since the intensity of sunlight, that is refracted at the water surface, is exponentially attenuated, the illumination volumes are subdivided so that the intensities of scattered light along the sub-volumes can be approximated linearly. This makes it possible to display the illumination volumes by using the Gouraud shading function. The sub-volume is further subdivided into three tetrahedra to integrate intensities of scattered light along the viewing ray within the sub-volume. We render the shafts of light by drawing these tetrahedra. We can reduce the computational time by the approximate calculation of the tetrahedron intensities as follows.

As shown in Fig. 4, let us consider the sub-volume which consists of six points $P_i (i = 1, ..., 6)$. Here we focus on the tetrahedron consisting of four points P_1, P_2, P_3 and P_4. Geometrically, the tetrahedra projected onto the screen can be classified into two cases (see Figs. 5 and 6). Let P_1', P_2', P_3' and P_4' be the projected vertices of P_1, P_2, P_3 and P_4, respectively. In

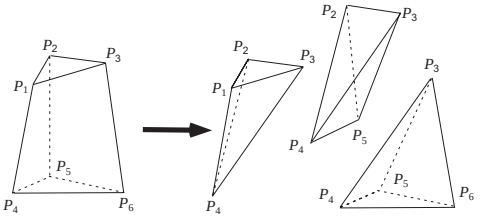


Figure 4: *Subdivide sub-volume into three tetrahedra.*

case I the point P_1' is inside the triangle $\triangle P_2'P_3'P_4'$ and in case II the point P_1' is outside the triangle.

In this section, we explain first the idea behind the proposed method for case I. Case II is described later. In the proposed method, shafts of light are rendered by displaying three triangles $\triangle P_1'P_2'P_3'$, $\triangle P_1'P_2'P_4'$ and $\triangle P_1'P_3'P_4'$ and accumulating their intensities into the frame buffer. The intensities of pixels inside these triangles should be calculated by integrating the intensities of light scattered on the intersection segments of the viewing ray with the tetrahedron. However, the intersection test for all pixels is very time consuming. Therefore we assume that the intensities of neighboring pixels are almost the same. This assumption is convincing if the illumination volume is sufficiently subdivided into small sub-volumes. Then, we calculate the intensities of four vertices, P_1', P_2', P_3' and P_4' only and the intensities of pixels inside the triangles are interpolated by using Gouraud shading function hardware. The intensities of these vertices are calculated as follows.

In case I, the length of the intersection segment between the viewing ray and the tetrahedron is the longest when the viewing ray passes point P_1 (see Fig. 5). We calculate I_C which is the integration of the scattered light arriving at the viewpoint for this longest case. On the other hand, since the intersection segment between the tetrahedron and the viewing rays for P_2', P_3' and P_4' is equal to 0, integration of the scattered light also gives 0. Therefore, we set the intensity of the vertex P_1' to I_C described before, and the intensities of the other vertices are set to 0. This method produces visually convincing results if the illumination volume is subdivided into small sub-volumes.

In case II, as shown in Fig. 6, let point P_7' be the intersection point between segments $P_1'P_4'$ and $P_2'P_3'$. The length of the intersection segment is longest when the viewing ray passes point P_7'. So, we calculate the integrated value I_C along that segment. The intensity of point P_7' is set to I_C and the intensities of other vertices are set to 0. Then these four triangles $\triangle P_7'P_1'P_2'$, $\triangle P_7'P_1'P_3'$, $\triangle P_7'P_2'P_4'$ and $\triangle P_7'P_3'P_4'$ are displayed.

In this way, we can reduce the computational time since our method does not require the intersection test

between the viewing rays (or scan planes) and the illumination volumes used in the previous methods[12, 24]. Moreover, since our method renders shafts of light by displaying triangles, we can make use of graphics hardware to accelerate the computation.

The outline of the process for rendering shafts of light is as follows.

1. Initialize the frame buffer.
2. Repeat the following process for each illumination volume.
   a. Subdivide the illumination volume into sub-volumes. Remove self-intersecting sub-volumes.
   b. Repeat the following process for each sub-volume.
      i. Subdivide each sub-volume into three tetrahedra.
      ii. Project each tetrahedron onto the screen.
      iii. Classify the tetrahedron into the two cases.
      iv. Calculate the intensities of the vertices of the triangles.
      v. Display the triangles and accumulate the intensities in the frame buffer.

The classification of the tetrahedron is performed by investigating the geometric relation between the projected vertices of the tetrahedron. In the next section, we describe the calculation of I_C.

3.2.1. Intensity calculation of tetrahedra of sub-volume

The value of the integration, I_C, described above is calculated as follows. First, we calculate the intersection points between the tetrahedron and the viewing ray passing through P_1' (case I) or P_7' (case II). There are two intersection points. Let these two intersection points be P_a and P_b, respectively. Let us denote the intensities at points P_a and P_b as I_a and I_b, respectively. In general, the length of the segment P_aP_b is very short, so we assume that the intensity in this segment P_aP_b varies linearly. Then, I_C can be expressed by the following equation.

$$I_C = \int_0^T \frac{I_b - I_a}{T} t + I_a dt = \frac{I_a + I_b}{2} T, \qquad (5)$$

where T is the distance between P_aP_b and t the distance between P_a and a point on the segment P_aP_b. I_a and I_b can be calculated by interpolating the intensities of the vertices of the sub-volume. We explain the calculation method for I_a and I_b in the following sections. First, we describe the calculation of the intensity at each sub-volume vertex. Then we present the calculation of the intensities I_a and I_b for case I and case II, respectively.

3.2.2. Intensity calculation of sub-volume

Let's consider a sub-volume shown in Fig. 4. First we calculate the scattered intensity $I^s_{P_i}$ at sub-volume vertex P_i $(i = 1, ..., 3)$ by Eq. (4). The flux ratio in Eq. (4) is calculated by $F_p = S_u/S_s$, where S_s is the area of triangle $\triangle P_1P_2P_3$ (see Fig. 4) and S_u is the corresponding area of the illumination volume at the water surface. Then intensity I_{P_i} at point P_i of the sub-volume is computed by the following equation.

$$I_{P_i}(\lambda) = I^s_{P_i}(\lambda) \exp(-c(\lambda) l_i), \qquad (6)$$

where l_i is the distance between P_i and the viewpoint. The intensities of P_4, P_5 and P_6 can be calculated in the same way.

3.2.3. Intensity calculation in case I

First, the two intersection points, P_a and P_b, between the viewing ray and the tetrahedron are calculated. In case I, P_a is equal to P_1. P_b is obtained by calculating the intersection of the ray with the triangle $\triangle P_2P_3P_4$ (see Fig. 5). The intensity, I_a, of point P_a is given by Eq. (6). The intensity, I_b, of point P_b is calculated by interpolating the intensities of vertices P_2, P_3 and P_4. The intensities of these vertices are also given by Eq. (6).

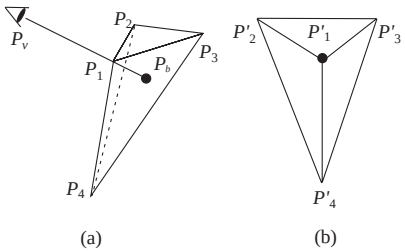


Figure 5: *Geometric relation of tetrahedron in case I.*

3.2.4. Intensity calculation in case II

In case II, the two intersection points, P_a and P_b, between the viewing ray and the tetrahedron are calculated as follows. Clearly, P_a is on segment P_1P_4 and P_b on P_2P_3 (see Fig. 6). We calculate the ratio between the segments P_1P_a and P_aP_4. Similarly, we calculate the ratio between the segments P_2P_b and P_bP_3. Then, the intensity, I_a, of point P_a is obtained by interpolating the intensities of vertices P_1 and P_4. The intensity, I_b, of point P_b is obtained from the intensities of vertices P_2 and P_3.

4. Displaying Caustics

4.1. Intensity calculation for caustics

To render caustics, we make use of illumination volumes again. Caustics patterns on objects are displayed

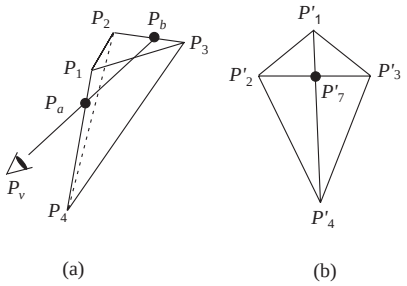


Figure 6: *Geometric relation of tetrahedron in case II.*

by drawing intersection triangles (caustic triangles) between illumination volumes and the objects. The intensity at each vertex of the caustic triangle, I_{P_i}, is calculated using the following equation.

$$I_{P_i}(\lambda) = I_{sun}(\lambda)T(\theta_{ii},\theta_{ti})\exp(-c(\lambda)(l_i+l_v)) \times F_p K_{obj}\cos\gamma, \quad (7)$$

where $I_{sun}(\lambda)$ is the intensity of the incident light, $T(\theta_{ii},\theta_{ti})$ is the transmittance at point P_i^s (see Fig. 7) on the water surface which corresponds to P_i, l_i is the distance between P_i^s and P_i (see Fig. 7), F_p is the flux ratio which can be calculated by the ratio of the area of the caustic triangle to the corresponding area of the water surface, K_{obj} is the reflectance of the object, γ is the angle between the object normal and the incident ray, and l_v is the distance between P_i and the viewpoint. In the next section, we propose two methods for rendering caustics: caustics on planes and caustics on objects. Both methods can be accelerated by graphics hardware.

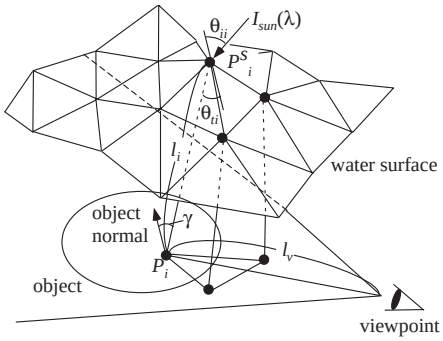


Figure 7: *Intensity calculation of caustics on object.*

4.2. Caustics on flat planes

Let's consider that the bottom of the water is flat as shown in Fig. 8. Caustics can be displayed by the following steps.

1. Calculate the intersection points P_1, P_2 and P_3 between each illumination volume and the plane.
2. Calculate the intensity of each point of each intersection triangle by Eq. (7).
3. Draw the intersection triangle $\triangle P_1P_2P_3$ and accumulate its intensity into the frame buffer by using hardware color blending functions.

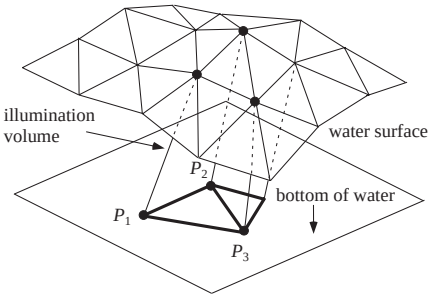


Figure 8: *Caustics on flat plane.*

4.3. Caustics on objects

The method for caustics on objects is not so easy as that for caustics on a plane. In this case, we have to investigate which illumination volume intersects the object. In the previous method[12], the caustic triangle between the illumination volume and the scan plane is calculated first. Then each point on the object is checked to see whether the point is located within the caustic triangle or not. To detect the points included in the caustic triangle, the depth of the triangle is compared with the depth of the objects. In the proposed method, the intersection test is accelerated by using the graphics hardware Z-buffer and stencil buffer.

Before rendering caustics, the bounding box of the object is calculated. We calculate the illumination volumes which intersect the bounding box in advance. In the rendering process for caustics described in subsection 4.3.1, we take into consideration only these illumination volumes which intersect the bounding box.

4.3.1. Rendering caustics by using sub-volumes

First, we make sub-volumes by slicing the illumination volumes with n scanplanes at a certain interval, Δs. We call these scanplanes *sample planes*. That is, sample plane i $(i = 1, ..., n)$ corresponds to scanline $i\Delta s$. Fig. 9(a) shows the geometric relation between the sub-volume and the object. To explain the proposed idea clearly, we show the rendering process on a scanplane (see Figs. 9(b), (c) and (d)). The drawing process of caustics is as follows.

1. Initialize the frame buffer and store the z values of objects in the Z-buffer (see Fig. 9(b)).
2. Draw the back faces of the sub-volume as viewed from the viewpoint (see Fig. 9(c)). We do not draw

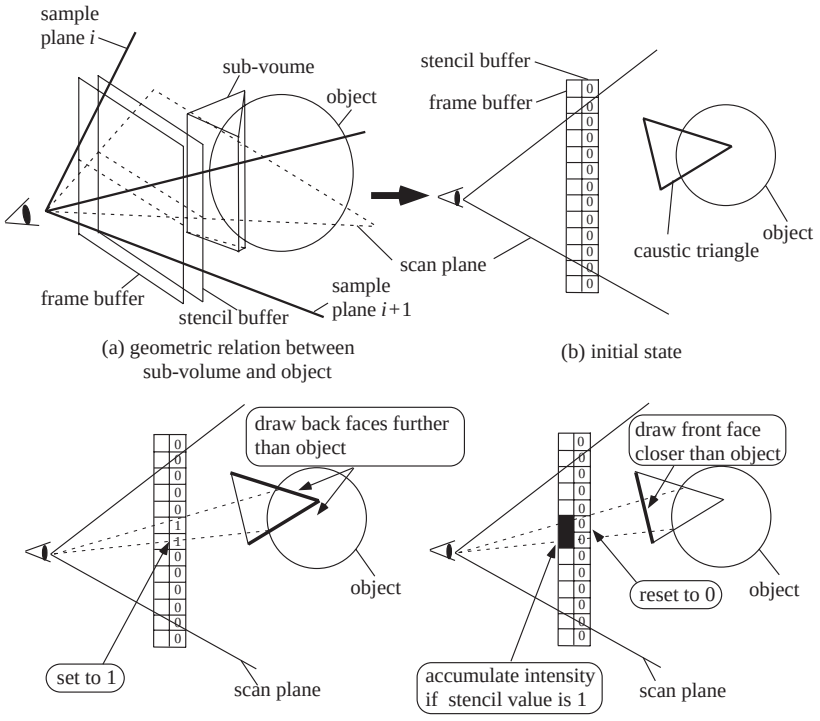


Figure 9: *Rendering caustics by using sub-volumes.*

the faces of the sub-volume in the frame buffer. We write 1 in the stencil buffer where the z values of the sub-volume are larger than the values of Z-buffer. In this step, we do not update the Z-buffer.

3. Draw the front faces of the sub-volume (see Fig. 9(d)). We accumulate the intensity in the frame buffer if the value of the stencil buffer equals 1 and the z values of the sub-volume are smaller than the values of the Z-buffer. In this step, we do not update the Z-buffer either.

Step 3 of the above procedure is as follows. First, we calculate the intensities of the two caustic triangles which are the intersection areas between the illumination volume and sample planes i and $(i+1)$ (see Fig. 9(a)). Next, we interpolate the intensities of the intersection area by using the intensities of the two caustic triangles. We render the shaded front faces of the sub-volume by setting the intensities of the six verticies, that is, the intensities of the two caustic triangles. Then the intensities of the intersection area are interpolated by using Gouraud shading function which can be accelerated by graphics hardware. In this step, we reset the stencil buffer to 0 when we draw front faces in order to return the stencil buffer to the initial state.

The above process creates an image of caustics on the objects. The value of each pixel represents the intensity of incident light on the objects. We call this image a caustic image. The final image is created by multiplying the pixel value by the surface reflectance and the cosine term ($K_{obj}\cos\gamma$ in Eq. (7)). To take into account this approximately, we calculate the refracted sunlight by assuming that the water surface is horizontal. The refracted sunlight is considered as a parallel light source. Then, an image of objects illuminated by this parallel source is created by using graphics hardware.

The final image is obtained by multiplying this image and the caustic image. The multiplication of these images is done by using graphics hardware.

5. Shadows within Water

In this section, we propose a method for rendering shadows due to objects within water. There are two types of shadows: shadows on the objects such as the ocean floor and shadows of shafts of light. In the proposed method, we display both of two types of shadows by the shadow map method[16].

First, we assume that the water surface is horizon-

tal. We calculate the refraction vector of sunlight. Secondly we generate a texture which is a depth image from the light's point-of-view (i.e. a depth image as viewed from the light source). The direction of the light is that of the refracted sunlight. Thirdly we render a scene from the eye's point-of-view and store the depth values as a texture. Finally, we compare the values of the two textures and determine whether the corresponding pixel is shadowed or not.

Applying the shadow map to displaying shafts of light and caustics can render shadows on objects and on shafts of light. The shadow map method can be accelerated by graphics hardware. However, the shadow map method has several disadvantages. The precision of the depth value stored in the shadow map is only 8-bit on a standard PC. It is not enough to render complex scenes. So we have adopted a 16-bit precision shadow map method by using a multi-digit comparison[13]. Although this method works only on some graphics hardware, we are sure that this extension will in future be supported by much more graphics hardwares.

6. Examples

Fig. 10 shows simple examples of a teapot within water. Figs. 10(a) and (b) show the teapot without and with shadows, respectively. Compared with Fig. 10(a), the shadows on the bottom of the water add reality in Fig. 10(b). Fig. 11 shows several examples of underwater optical effects. As shown in Fig. 11(a), shafts of light are obstructed by the teapot and shadows on the bottom of the water can be seen. Fig. 11(b) shows caustics on a submarine. This image indicates that our method can calculate caustics on objects with complex shapes. Figs. 11(c) and (d) are examples of a dolphin. Fig. 11(d) shows the dolphin as viewed from the bottom of the water. Figs. 11(e) and (f) are stills from the animation of two dolphins.

We created these images on a desktop PC (PentiumIII 1GHz) with Geforce2ULTRA. The image sizes of these figures are all the same, 640x480. The mesh size of the water surface is also all the same, 512x512. The intensities of shafts of light reaching the viewpoint are attenuated exponentially. Therefore we can ignore the illumination volumes that are far from the viewpoint. Moreover, we take into account only the illumination volumes that are in the viewing frustum. In our experiment, the 18,000 illumination volumes seem to be enough to generate these images. The quality of image depends on the number of sub-volumes per illumination volume. The 20 sub-volumes per illumination volume seem to be enough to generate convincing results. The computational time for each figure is shown in Table 1. The computational time of Fig. 10(a) using software is 64 seconds. That is, the method using hardware is 22 times faster than the method using software. The motion of the dolphins in the animation is calculated by Free-Form Deformation[17]. These examples demonstrate that the proposed method can create realistic underwater images efficiently.

Table 1: *Computation time*

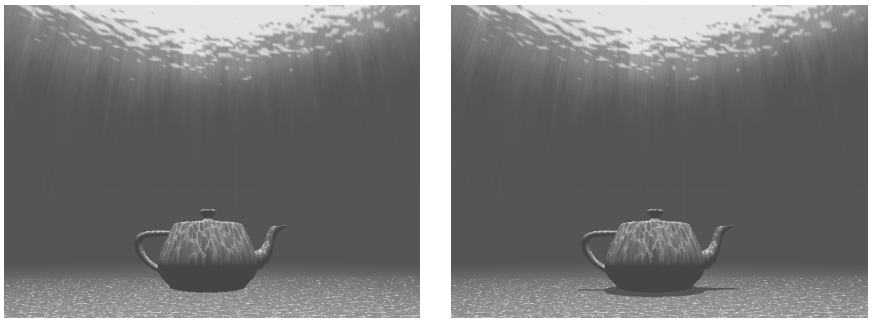
Figure No.	9(a)	9(b)	10(a)	10(b)
Time[sec]	2.92	3.85	4.24	7.89
Figure No.	10(c)	10(d)	10(e)	10(f)
Time[sec]	4.12	4.34	4.88	5.22

7. Conclusion

In this paper, we have proposed a method for rendering optical effects within water such as shafts of light, caustics on objects and shadows due to objects. The proposed method utilizes graphics hardware that has recently become highly efficient. We adopted OpenGL as a graphics library. The advantages of the proposed method are as follows.

1. Shafts of light are efficiently displayed by rendering illumination volumes which are subdivided into sub-volumes. Sub-volumes are further divided into tetrahedra. Then, the shafts of light are displayed by drawing the visible triangles of the tetrahedra and accumulating their intensities. This process is accelerated by hardware color blending functions.
2. Our method can display caustics not only on flat planes but also on any object. To render caustics on objects, the intersection test between illumination volumes and the object is required. We detect the intersection regions by using a Z-buffer and a stencil buffer.
3. Shadows due to objects within water are also taken into consideration. There are two types of shadows: shadows on the objects and shadows due to shafts of light. We display both types of shadow by using the shadow map technique which can be accelerated by graphics hardware.

In future work, we will improve the following subjects. First, we want to accelerate the proposed method in order to achieve real-time animations. Next, our shadowing method still has a room for improvement. Since we assume that the water surface is horizontal when calculating shadows, the boundaries of the shadows are sharp. Objects within water, however, are illuminated from various directions, so the boundaries of the shadows are not always sharp. Therefore we plan to render soft shadows.



(a) teapot without shadows (b) teapot with shadows

Figure 10: *Examples of a teapot within water.*

References

1. N. Brière, P. Poulin, "Adaptive Representation of Specular Light Flux," *Computer Graphics Forum*, Vol.20, No.2, 2001, pp.149-159.
2. Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita, "A Simple, Efficient Method for Realistic Animation of Clouds," *Proc. SIGGRAPH2000*, 2000, pp.19-28.
3. Y. Dobashi, T. Yamamoto, T. Nishita, "Interactive Rendering Method for Displaying Shafts of Light," *Proc. Pacific Graphics2000*, 2000, pp.31-37.
4. A. Fournier, W.T. Reeves, "A Simple Model of Ocean Waves," *Proc. SIGGRAPH'86*, 1986, pp.75-84.
5. F.J.v. Gerstner, *"Theorie der Wellen,"* Ann. der Physik, 32, 1809, pp.412-440.
6. W. Heidrich, H. P. Seidel, "Realistic, Hardware-Accelerated Shading and Lighting," *Proc. SIGGRAPH'99*, 1999, pp.171-178.
7. H.W. Jensen, P.H. Christensen, "Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps," *Proc. SIGGRAPH'98*, 1998, pp.311-320.
8. K. Kaneda, G. Yuan, Y. Tomoda, M. Baba, E. Nakamae, T. Nishita, "Realistic Visual Simulation of Water Surfaces Taking into Account Radiative Transfer," *Proc. CAD/Graphics'91*, 1991, pp.25-30.
9. A. Kunimatsu, Y. Watanabe, H. Fujii, T. Saito, K. Hiwada, T. Takahashi, H. Ueki, "Fast Simulation and Rendering Techniques for Fluid Objects," *Eurographics 2001*, 2001, pp.57-66.
10. T. Nishita, Y. Miyazaki, E. Nakamae, "Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources," *Computer Graphics*, Vol. 21, No. 4, 1987, pp.303-310.
11. T. Nishita, T. Shirai, K. Tadamura, E. Nakamae, "Display of The Earth Taking into account Atmospheric Scattering," *Proc. SIGGRAPH'93*, 1993, pp.175-182.
12. T. Nishita, E. Nakamae, "Method of Displaying Optical Effects within Water using Accumulation-Buffer," *Proc. SIGGRAPH'94*, 1994, pp.373-380.
13. "Shadow Mapping with Today's OpenGL Hardware," GDC 2000, 2000, http://www.nvidia.com/developer.nsf/.
14. D. Peachey, "Modeling Waves and Surf," *Proc. SIGGRAPH'86*, 1986, pp.65-74.
15. S. Premoze, M. Ashikhmin, "Rendering Natural Waters," *Proc. Pacific Graphics2000*, 2000, pp.23-30.
16. M. Segal, C. Korobkin, R. Widenfelt, J. Foran, P. Haeberli, "Fast Shadows and Lighting Effects Using Texture Mapping," *Computer Graphics*, Vol. 26, No. 2, 1992, pp.249-252.
17. T.W. Sederberg, S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Computer Graphics*, Vol. 20, No. 4, 1986, pp.151-160.
18. M. Shinya, T. Saito, T. Takahashi, "Rendering Techniques for Transparent Objects," *Proc. Graphics Interface'89*, 1989, pp.173-181.
19. J. Stam, "Random Caustics: Natural Textures and Wave Theory Revisited," *Technical Sketch SIGGRAPH'96*, 1996, p.151.
20. J. Stam, "Stable Fluids," *Proc. SIGGRAPH'99*, 1999, pp.121-128.
21. J. Tessendorf, "Simulating Ocean Water," *SIGGRAPH'99 Course Note*, Simulating Natural Phenomena, 1999, pp.1-18.
22. C. Trendall, A.J. Stewart, "General Calculations Using Graphics Hardware, with Application to Interactive Caustics," *Proc. Eurographics Rendering Workshop*, 2000, pp.287-298.
23. P.Y. Ts'o, B.A. Basky, "Modeling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping," *ACM Trans. on Graphics*, 1987, pp.191-214.
24. M. Watt, "Light-Water Interaction using Backward Beam Tracing," *Proc. SIGGRAPH'90*, 1990, pp.377-385.

Appendix A: Modeling of Waves

Modeling of waves is very important since shafts of light results from the convergence and divergence of refracted light

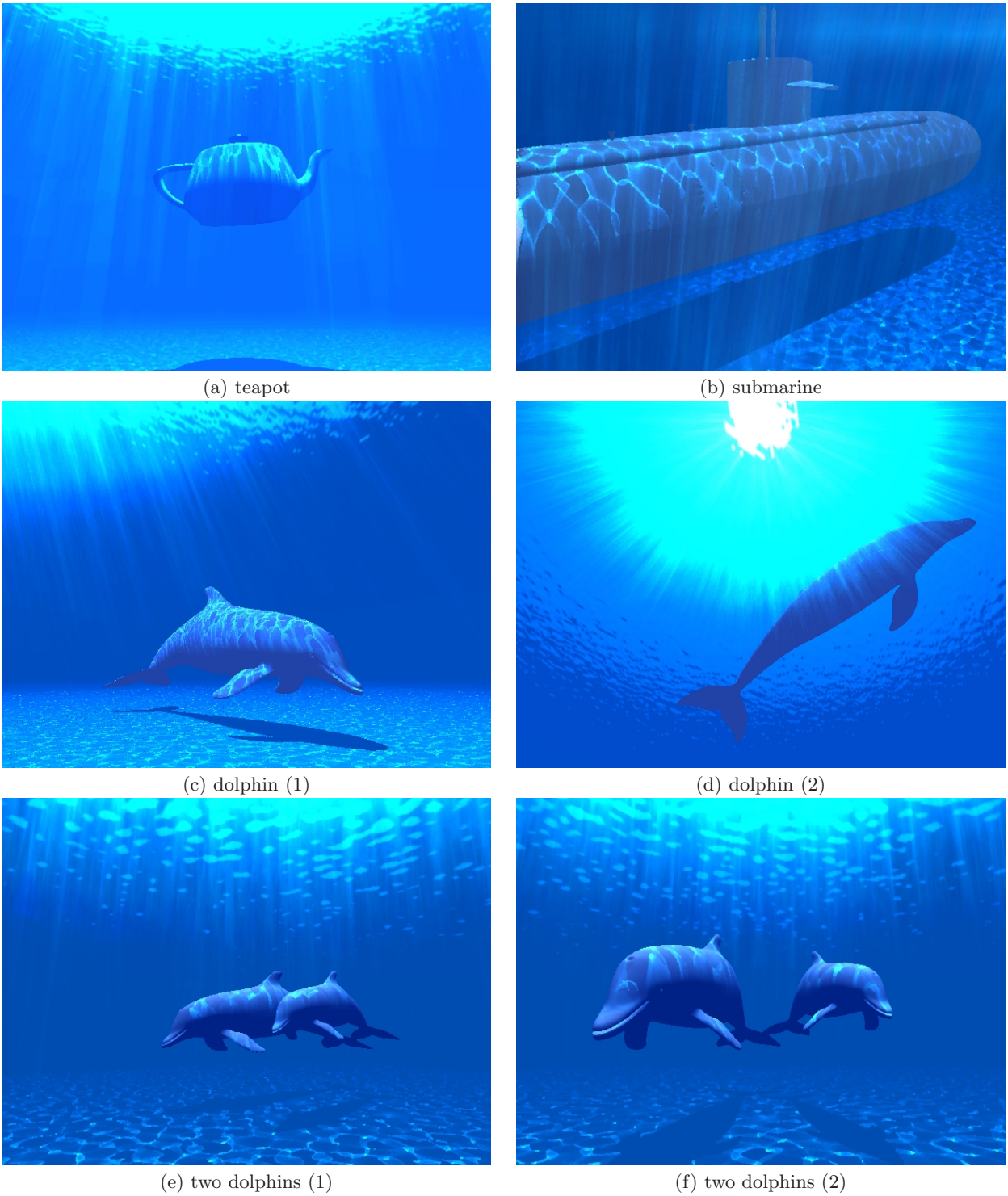
at the water surface. Caustics patterns are also determined by the shape of waves.

The wave model which we have adopted is a statistical wave model[21]. The statistical wave model represents the wave height as the decomposition of sine and cosine waves. The wave height $h(\vec{\mathbf{x}}, t)$ at time t can be calculated by the following equation.

$$h(\vec{\mathbf{x}}, t) = \sum_{\vec{\mathbf{k}}} \tilde{h}(\vec{\mathbf{k}}, t) \exp(i\vec{\mathbf{k}} \cdot \vec{\mathbf{x}}), \tag{8}$$

where $\vec{\mathbf{x}}$ is the horizontal position and $\vec{\mathbf{k}}$ is the wave number vector. This equation can be calculated by Fast Fourier Transforms. The Fourier amplitudes $\tilde{h}(\vec{\mathbf{k}}, t)$ determine the shape of the wave. $\tilde{h}(\vec{\mathbf{k}}, t)$ is based on Phillips spectrum, which is calculated from the wind speed and direction.

One of the advantages of this model is that wind speed can be handled by users. Although this model cannot represent plunging waves because of the limitation of the height field, the statistical wave model can generate various waves from calm waves on a sunny day to rough waves by specifying the wind speed.



(a) teapot (b) submarine

(c) dolphin (1) (d) dolphin (2)

(e) two dolphins (1) (f) two dolphins (2)

Figure 11: *Examples of underwater scene.*