A Fast Rendering Technique of Transparent Objects and Caustics

Kei Iwasaki
Wakayama University
iwasaki@sys.wakayama-u.ac.jp

Fujiichi Yoshimoto
Wakayama University
fuji@sys.wakayama-u.ac.jp

Yoshinori Dobashi
Hokkaido University
doba@nis-ei.eng.hokudai.ac.jp

Tomoyuki Nishita
The University of Tokyo
nis@is.s.u-tokyo.ac.jp

Abstract

Rendering refractive caustics from transparent objects on opaque objects is computationally intensive. This paper presents a fast rendering technique for transparent objects and refractive caustics due to transparent objects on the opaque object. To calculate the intensities of caustics, we set virtual planes around the opaque object and store the intensities of caustics on the virtual planes as textures. Caustics are then rendered by using these textures. Our implementation, which additionally uses a GPU accelerator, enables us to render refractive caustics at interactive frame rates. We demonstrate this interactive rendering additionally for translation, rotation of both transparent objects and opaque objects, as well as for changing light and viewing directions.

Keywords: caustics, transparent, refraction, global illumination, graphics hardware

1 Introduction

Transparent materials like glass exist widely in real world. If the transparent object is illuminated, caustics are formed by transmitted light through the transparent object (see Fig. 1). Research into rendering caustics is a challenging task and various methods have been developed to achieve this purpose. Due to the complexity of the caustics calculation, the computational cost for rendering caustics is high. In this paper, we present an efficient rendering method for transparent objects and caustics using a programmable graphics processing unit (GPU).

To render refractive caustics formed by transmitted light, we calculate the illumination distribution on the surface illuminated by the transmitted rays. In this paper, we call an opaque object that is illuminated by the transparent objects *a target object*. We assume that the surfaces of the target objects are diffuse surfaces.

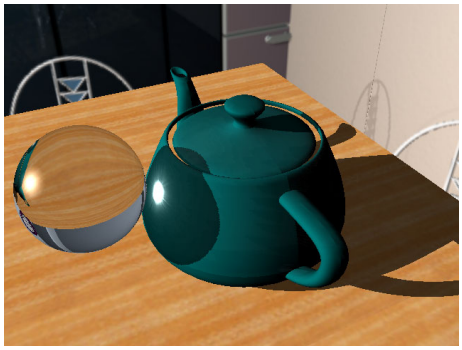


Figure 1: A glass ball casts caustics and shadows on a teapot. This image can be rendered at 18.0 fps on a desktop PC.

To calculate the illumination distribution of the target object, we set virtual planes around the target object. The illumination distribution of the caustics on the surface of the target object is calculated by using the illumination distribution on the virtual planes. The illumination distributions of caustics on virtual planes are efficiently calculated by using the GPU and are stored as textures. Caustics on the target object are rendered by using the textures, the normal information and depth information of the target object. This process is accelerated by using the GPU.

Our method has the following features.

- Refractive caustics due to transparent objects can be rendered efficiently using the GPU.

- Our method allows the rapid rendering of refractive caustics under rigid body transformations (rotation and translation) and with changing light directions.

- Fast rendering of refractive caustics on target objects with rigid body transformations.

2 Previous Work

Many methods have been proposed to render caustics due to transparent objects [1, 2, 3, 4, 5]. Although realistic images that include caustics can be rendered by using these methods, they are all computationally intensive. Fast rendering methods for caustics using GPUs have also been developed. Purcell et al. [6] proposed a photon mapping method for GPUs. This method, however, does not achieve real-time rendering. Sloan et al. [7] presented precomputed radiance transfer to achieve real-time rendering of interreflections, soft shadows and caustics. Wand and Strasser [8] proposed a hardware acceleration method for rendering reflective caustics due to specular objects. These methods, however, do not deal with refractive caustics from transparent objects.

Several methods have been developed to render refractive caustics at interactive frame rates [9, 10, 11]. However, these methods require several CPUs to render caustics at interactive frame rates. Therefore, these methods are much slower than our method that can achieve interactive frame rates on a current standard PC (and 1CPU). Iwasaki et al. [12] proposed an efficient rendering method for caustics due to transparent object. This method, however, is limited to caustics on simple shape surfaces. We extend this method to handling the complex shape of the target object.

3 Overview

We focus on the refractive caustics on the diffuse surface due to transparent objects, and refractions and transmissions of light due to the transparent objects. Though our method deals with the refractive caustics, reflective caustics can be rendered by extending our method to take into account reflected light from transparent object. Our method can deal with the geometry of transparent and diffuse materials represented by triangle meshes. The light sources we can deal with are a parallel light source, a point light source. We handle a linear light source approximated by a set of point light sources.

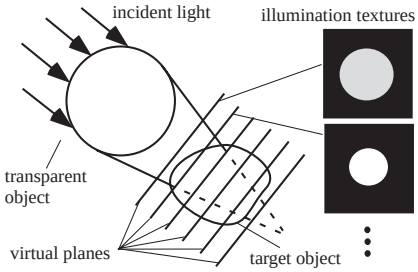


Figure 2: Virtual planes and illumination textures.

To render refractive caustics on a diffuse surface due to a transparent object, the refraction of the incident light at the transparent object surface and its transmission through the transparent object must be taken into consideration. In our method, the transmitted rays can be calculated by referring to a light transport table [12], which stores the information of the outgoing rays for each incident rays onto each vertex. The light tranport table is calculated in advance. The light transport table allows us to rapidly render refractive caustics from moving and rotating transparent objects with changes of the light direction.

Next the illumination distribution on the target object illuminated by the converged and diverged transmitted light must be calculated. This calculation requires intersection tests between the transmitted rays and the surfaces of the target objects. The computational cost of rendering refractive caustics is therefore quite expensive.

To address this problem, we set virtual planes around the target object and calculate the illumination distribution of the virtual planes (see Fig. 2). This reduces the computational time since the intersection calculation between the transmitted rays and many polygons representing the target object surface is replaced with that between the rays and the plane. Then the illumination distribution of the target object is approximated by using the illumination distribution of the virtual planes. We call the illumination distriution on each virtual plane an *illumi-*

nation texture.

In the previous method [13], the virtual planes are also used to calculate the intensities of caustics. However, these virtual planes are set perpendicular to the average directions of the refracted viewing rays. Therefore, the virtual planes depend on the viewpoint and this indicates that the caustics intensities are calculated when the viewpoint changes. On the other hand, our new method does not depend on the viewpoint, and therefore we can render caustics without re-calculation of the caustics intensities when the viewpoint changes.

To render the images of refracted objects through the transparent objects, we use the sliced object images [13]. The calculation of the transmitted viewing rays are accelerated by using the light transport table [12]. To create sliced object images, virtual planes are set perpendicular to the transmitted viewing rays. Then the sliced object images are created by projecting a part of the target object surface between two adjacent virtual planes onto the virtual plane. The illumination textures are mapped onto the target object when creating the sliced object images. Next, by mapping the sliced object images onto the transparent object surface, the images of refracted objects taking into account refractive caustics are rendered. (See the previous method [13] for more details.)

4 Rendering Caustics

4.1 Basic Concept

To render refractive caustics, we employ the idea of illumination volumes proposed by Nishita et al. [14]. We calculate the transmitted light to create these illumination volumes. The incident direction of the light at each vertex of the transparent object is computed first to calculate the transmitted light. Then, the outgoing direction and position after passing through the transparent object are obtained by simply referring to the light transport table using the incident light direction at each vertex.

As shown in Fig. 3, by sweeping the refracted outgoing rays, illumination volumes are created. The areas of intersection between the illumination volumes and the object surface are calculated and the intensities of these are accumulated

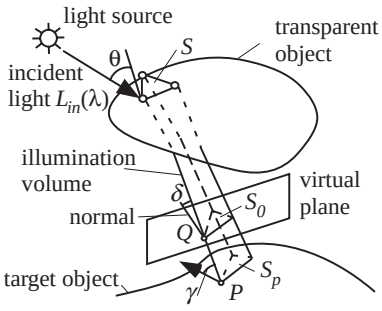


Figure 3: Illumination volume.

to render refractive caustics due to the transparent object. The intensity L_P at point P of the intersection area is calculated by using the following equation (see Fig. 3).

$$L_P(\lambda) = L_{in}(\lambda)\cos\theta F_t(\lambda)F_p\rho_d(\lambda), \qquad (1)$$

where λ is the wavelength, which is sampled for RGB components, $L_{in}(\lambda)\cos\theta$ is the intensity of the incident light onto the triangular mesh of the transparent object surface, and $F_t(\lambda)$ is the total Fresnel transmittance. F_p is the flux ratio and is calculated from the following equation $F_p = S/S_p$, where S_p is the area of the cross section between the illumination volume and the surface of the target object, and S is the area of the triangular mesh on the surface of the transparent object. Also, $\rho_d(\lambda)$ is the diffuse reflectance of the surface of the target object. The calculation of the intersected areas are time-consuming since it requires many intersection tests. In our method, the area of the intersection area S_p is approximated by using the area of the intersection triangle S_0 between the virtual plane and the illumination volume (see Fig. 3). The area S_p is approximated by $S_0\frac{cos\delta}{cos\gamma}$, where δ and γ are angles between the transmitted ray and the normal of the virtual plane, and between the transmitted rays and the normal of the target object surface, respectively.

4.2 Creation of Illumination Textures

Caustics on the target object surface are calculated by using the illumination textures. By using the area S_0, Eq. (1) is rewritten as the following equations,

$$L_P(\lambda) = I_Q(\lambda)\rho_d(\lambda), \qquad (2)$$

$$I_Q(\lambda) = L_{in}(\lambda)\cos\theta F_t(\lambda)\frac{S\cos\gamma}{S_0\cos\delta}. \qquad (3)$$

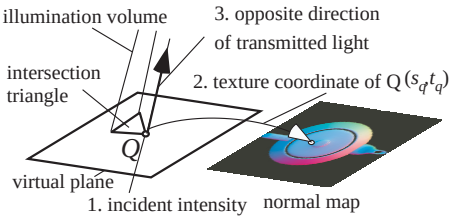


Figure 4: Creation of illumination textures using vertex program. Three parameters are set to each vertex of the intersection triangle.

The illumination texture stores the incident intensity I_Q at point Q of the intersection triangle. The caustics intensity at point P on the target object is calculated by multiplying the diffuse reflectance $\rho_d(\lambda)$. Illumination textures are created by drawing the intersection triangles and accumulating the intensities expressed by Eq. (3). We set a virtual camera so that the viewing direction is parallel to the average direction of the transmitted lights.

To calculate the incident intensity $I_Q(\lambda)$, the dot product of the direction vector of the transmitted light and the normal vector of the target object surface, $\cos\gamma$, must be calculated. In the previous work on the caustics due to water surfaces [13], this term is only calculated approximately. In our method, we calculate the cosine term with per pixel accuracy by using vertex and fragment programs. The information of the normal of the target object surface is stored in the normal map. The information of the depth values from the virtual camera is also calculated. The depth information is used to calculate the shadows.

Fig. 4 provides a visual overview of the vertex program. First, we calculate the intersection triangle between the illumination volume and the virtual plane. Then three parameters are set to vertex Q of the intersection triangle. One is the intensity $L_{in}(\lambda)\cos\theta F_t(\lambda)\frac{S}{S_0\cos\delta}$ that is the incident intensity I_Q except for $\cos\gamma$. A second parameter is the texture coordinate of vertex Q to refer to the normal map. The texture coordinate of vertex Q is calculated by using the idea of projection texturing [15]. Our third parameter is the opposite direction of the transmitted light onto vertex Q.

In the fragment program of the GPU, we calculate the term $\cos\gamma$ per pixel of the illumination texture. The normal information is obtained by addressing the normal map texture. The texture coordinate for each pixel is interpolated by using the texture coordinates set in the vertex program. Each pixel is assigned the opposite direction of the transmitted light, the cosine term is calculated by the dot product of the normal vector and the opposite direction vector of the transmitted light. The intensity I_Q is calculated by multiplying the first parameter and the cosine term per pixel. The illumination textures are created by drawing the pixel with the intensity I_Q and accumulating the intensities.

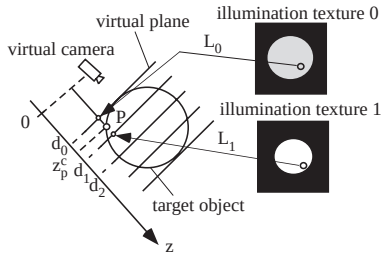


Figure 5: Rendering caustics using illumination textures.

4.3 *Rendering Caustics Using Illumination Textures*

Caustics on the target object are rendered by using illumination textures. The rendering of caustics on the target object surfaces is performed by using vertex and fragment programs. The intensity at point P on the target object is calculated by multiplying the incident intensity at P, that is stored in illumination textures, by the diffuse reflectance $\rho_d(\lambda)$. Therefore, to calculate the incident intensity at point P, the texture coordinate of point P in texture space of the illumination texture are calculated. This calculation is also based on the projection texturing method [15].

In a fragment program, the incident intensity at point P is calculated as follows. We first determine whether point P is visible from the virtual camera by using the depth information of the normal map texture. If point P is not visible from the virtual camera, point P is occluded by the target object surface and is considered as shadowed regions. Thus the color of the pixel corresponding to point P is assigned the intensity of the ambient light. If point P is visible from the virtual camera, we consider that point P is illuminated by transmitted light. The color

of the pixel corresponding to illuminated point P is calculated as follows.

Let the number of virtual planes be n and the depth value of the ith virtual plane from the virtual camera be d_i $(d_0 < d_1 < \cdots < d_{n-1})$. Two adjacent virtual planes of point P are determined by finding the depth values that satisfy $d_i \leq z_p^c < d_{i+1}$, where z_p^c is the depth of P from the virtual camera. Next, the incident intensity at point P is calculated by using the illumination textures corresponding to the ith virtual plane and $(i+1)$th virtual plane. The incident intensity at point P is interpolated by the intensities of the illumination textures i and $i+1$. Finally, the intensity at point P is calculated by multiplying the incident intensity by the diffuse reflectance $\rho_d(\lambda)$.

5 Results

Figs. 6(a) and (b) show refractive caustics on a buddha model due to a glass ball. In these figures, refracted buddha model with caustics can be seen through the glass ball. The rendering frame rates for Figs. 6(a) and (b) are 7.8fps. Fig. 6(c) shows caustics on a buddha model due to a glass teapot. This figure demonstrates that our method can create caustics on complex target object surfaces from complex shape transparent object. Fig. 6(d) shows refractive caustics from the glass dolphin due to a linear light source. The linear light source is approximated by 8 point light sources. We calculate caustics and shadows for each point light source and accumulate the results. The rendering frame rates for Figs. 6(c) and (d) are 6.6fps and 3.6fps, respectively.

These images were created on a desktop PC (Pentium4 3.4GHz, 2GB main memory) with a nVidia GeForce 6800 GTO. The image size of these figures is 640×480. The sizes of the light transport tables used in these figures are less than 20MB. The number of the sliced object images in all figures except for Fig. 1 is 6. In Fig. 1, the number of the sliced object image is 1. The number of the virtual planes is 4. The size of the sliced object images and that of the illumination textures are 512×512. These parameters are determined experimentally. Since our method can render transparent objects and refractive caustics at interactive frame rates, the parameters can be changed interactively.

The examples shown in this section demonstrate that our method can render refractive caustics and transparent objects at interactive frame rates.

6 Conclusion and Future Work

We have presented a fast rendering technique for transparent objects and caustics. We calculate the intensity of caustics on the diffuse object surfaces by using the illumination textures that store the illumination distribution on virtual planes around the object. This indicates that our method can render caustics on the diffuse surfaces without the intersection calculations between the transmitted lights and the surface, whose computational costs are expensive. The illumination textures are created efficiently by using a GPU. The rendering of caustics on the surface is performed by mapping the illumination textures. In our method, caustics and the shadows are rendered in the same process using programmable graphics hardware.

The limitation of this technique is that it takes into account only refraction of light, and does not consider the rays which are refracted out after being reflected inside the object. This limitation can be solved by taking the reflection of light into account to calculate the light transport table. However, this results in the huge memory size for the light transport table and the computational costs become very expensive. We consider that the refracted rays are the most important components to render caustics and therefore we neglect other paths.

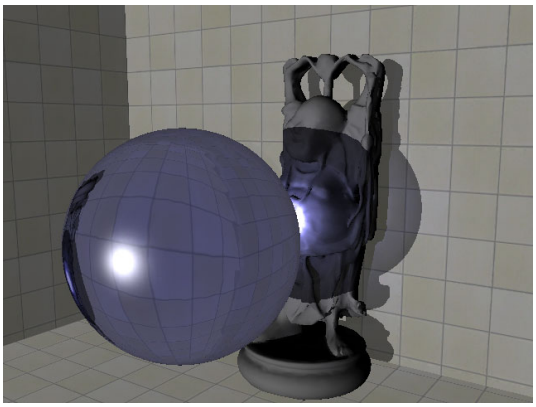
In future work, we plan to render caustics under area light sources and render caustics under distant lighting expressed by an environment map.

References

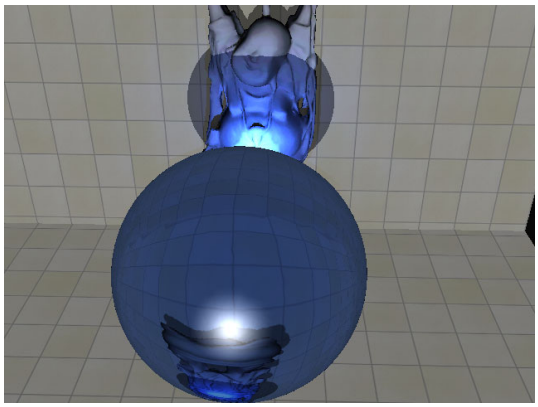
[1] J Arvo. Backward ray tracing. In *Course Note #12 of SIGGRAPH'86*, pages 259–263, 1986.

[2] M Shinya, T Saito, and T Takahashi. Rendering techniques for transparent objects. In *Proceedings of Graphics Interface'89*, pages 173–181, 1989.

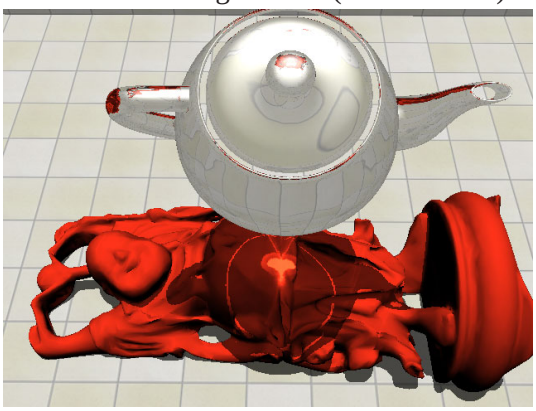
[3] H. W. Jensen. Rendering caustics on non-lambertian surfaces. In *Proceedings of Graphics Interface'96*, pages 116–121, 1996.



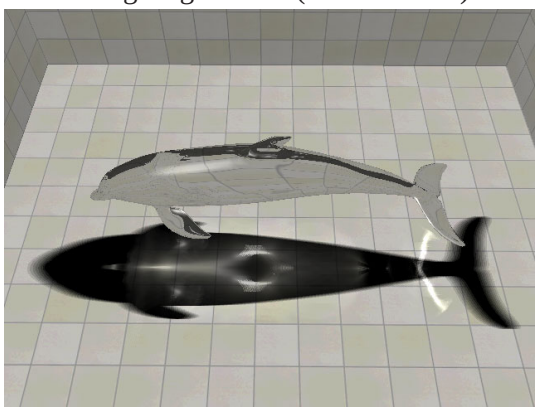
(a) Refractive caustics on a buddha due to a colored glass ball (6.4K vertices).



(b) Refracted buddha and walls through a glass ball (6.4K vertices).



(c) Refractive caustics on buddha due to a glass teapot (13.0K vertices).



(d) Refractive caustics due to a glass dolphin (8.9K vertices) under linear light.

Figure 6: Examples of refractive caustics.

[4] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH'98*, Computer Graphics Proceedings, Annual Conference Series, pages 311–320. ACM, 1998.

[5] N. Briere and P Poulin. Adaptive representation of specular light flux. *Computer Graphics Forum*, 20(2):145–159, 2001.

[6] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of Graphics Hardware 2003*, pages 41–50, 2003.

[7] P Sloan, J Kautz, and J Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 527–536. ACM, 2002.

[8] M Wand and W Straβer. Real-time caustics. *Computer Graphics Forum*, 22(3):610–619, 2003.

[9] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination using fast ray tracing. In *Proceedings of Eurographics Workshop on Rendering*, pages 15–24, 2002.

[10] J. Gunther, I. Wald, and P. Slusallek. Realtime caustics using distributed photon mapping. In *Proceedings of Eurographics Symposium on Rendering*, pages 111–121, 2004.

[11] C Wyman, C Hansen, and P Shirley. Interactive caustics using local precomputed irradiance. In *Proceedings of Pacific Graphics*, pages 143–151, 2004.

[12] K. Iwasaki, F. Yoshimoto, Y. Dobashi, and T. Nishita. A rapid rendering method for caustics arising from refraction by transparent object. In *Proceedings of Cyberworld 2004*, pages 39–44, 2004.

[13] K. Iwasaki, Y. Dobashi, and T. Nishita. A fast rendering method for refractive and reflective caustics due to water surfaces. *Computer Graphics Forum*, 22(3):601–609, 2003.

[14] T Nishita and E Nakamae. Method of displaying optical effects within water using accumulation-buffer. In *Proceedings of SIGGRAPH'94*, Computer Graphics Proceedings, Annual Conference Series, pages 373–380. ACM, 1994.

[15] M Segal, D Korobkin, R Widenfelr, J Foran, and P Haeberli. Fast shadows and lighting effects using texture mapping. In *Proceedings of SIGGRAPH 1992*, Computer Graphics Proceedings, Annual Conference Series, pages 249–252. ACM, 1992.