Rendering Techniques for Transparent Objects

Mikio Shinya* Takafumi Saito Tokiichiro Takahashi

NTT Human Interface Laboratories
1-2356, Take, Yokosuka-shi, Kanagawa, Japan

Abstract

Three techniques are proposed for more realistic image synthesis of transparent objects. The techniques simulate important phenomena due to refraction and/or reflection, such as focuses and dispersion. For efficient simulation of focuses and caustics, a method called *grid-pencil tracing* is proposed, based on the ideas of pencil tracing and backward ray tracing. An anti-aliasing filter is designed for grid-pencil tracing in area-source illumination environments. The theory of pencil tracing is also applied to dispersion simulation. An extension of the theory provides a basis for dispersive pencil tracing, where both spatial coherency and 'wavelength coherency' is effectively used for fast image synthesis of dispersive objects.

Furthermore, a linear filtering technique is applied to effectively express extremely bright spots such as sunshine reflection. The technique is based on an analysis of a special camera filter called a 'cross-screen filter,' which emphasizes bright spots by producing star-like lines around the spots on the image.

With these techniques, the realism of transparent object images can be much enhanced in an inexpensive way.

KEYWORDS: Ray Tracing, Lighting Model, Dispersion, Filtering, Realistic Image Synthesis

1 Introduction

Photo-realistic rendering based on optical models is one of the most important areas in the field of computer graphics. Realization of realistic rendering of transparent objects, in particular, has been the focus of much interest, because of their attractive properties due to refraction and reflection.

The most noticeable phenomenon due to refraction is image distortion through transparent objects, which can be simulated by the ray tracing algorithm [1]. In many situations, however, simple application of ray tracing is not adequate to render photo-realistic images of transparent objects, because other important phenomena occur due to refraction and/or reflection. For example:

- **caustics and focuses;** Refraction and reflection causes light concentration, resulting in a complex pattern of caustics and focuses. This phenomenon is often observed in water scenes (swimming pools, seaside, and so on) and can even be produced by a glass of water.
- **dispersion;** Due to dispersion, a rainbow spectrum is often produced by transparent objects, such as prisms, gemstones, and cut-glass.
- **extremely bright spots;** Reflection of primary light sources are manifested as extremely bright spots. The glint off a knife or shimmering light reflected off a body of water, for example.

In this paper, we propose three related techniques for effectively simulating these various phenomena. These techniques take advantage of pencil tracing [2] and filtering techniques to allow their fast execution.

2 Shadowing Techniques for Transparent Objects

Transparent objects often make caustics and focuses of light by refracting and/or reflecting rays emitted from light sources. Although this phenomenon is important for creating a realistic picture of transparent objects, it has been generally neglected because of the large amount of computation required for its simulation. The high computation cost is due to the following characteristics of caustics and focuses.

- The simulation requires expensive refraction/reflection calculation, which means that simple projection is inapplicable to either illuminance calculation or viewing transformation.
- Illuminance around caustics changes much more rapidly than in other areas. A high sampling rate is thus required to avoid aliasing artifacts.

The most important problem is, therefore, to find a way of reducing the number of sample rays without aliasing artifacts.

*Current address: Dynamic Graphics Project, Computer Systems Research Institute, University of Toronto, 10 King's College Road, Toronto, Ontario M5S 1A4

In this section, we provide an inexpensive technique for simulating caustics and focuses of transparent objects in area-source illumination. This technique is based on the ideas of pencil tracing and backward ray-tracing.

2.1 Background

There are two general approaches to the global lighting model, namely, radiosity [3] and ray tracing [1]. Both approaches are theoretically capable of simulating the phenomenon, however, their direct application is not practical because of their computation cost and the aliasing problem.

Cohen, et al., pointed out [4] that the radiosity model is applicable to caustic simulation by using ray tracing in the form-factor calculation phase instead of simple projection and z-buffer algorithm. However, this straight-forward extension would be too costly, because it would require division into too many patches to avoid the aliasing artifacts due to the fine structure of caustics.

Ray tracing, on the other hand, can be more easily applied to the simulation. Kajiya succeeded in simulating caustics by using distributed ray tracing with his 'interest search' strategy [5]. This technique too, however, is impractical in terms of computation time, because a tremendous amount of rays have to be traced.

When all light sources are point sources, it is possible to reduce the number of rays by tracing light energy transmission from light sources. Arvo proposed backward ray tracing [6], where, to make up an illuminance map, rays are traced from a point source in a pre-processing phase.[1] There, each ray is regarded as a photon carrying a certain amount of energy, and then illuminance is calculated by photon counting. The point sampling, however, results in a serious aliasing problem, as will be shown later. Shinya, et al., applied pencil tracing to the simulation [2]. In their method, light pencils are traced from a point light source, and intersections of pencils and objects are stored as illuminance polygons, which are considered to be extensions of shadow polygons with finite illuminance. The advantages of the method are elimination of moire artifacts and reduced computation cost, but it can only applicable to a limited range of objects.

In this paper, we outline a practical method for simulating caustics. First, the idea of pencil tracing is introduced into backward ray tracing to solve the moire problem. Second, a point source method is extended to deal with area-light sources by applying a space-variant filter. Finally, the proposed method is applied to a more general renderer.

2.2 Point source illumination

The problem addressed here is to simulate diffuse reflection of specularly reflected and/or refracted light emitted from a point source. If light is neither refracted nor reflected, illuminance can be easily calculated by the inverse square law. Unfortunately, this law can not be applied to the simulation of caustics because of refraction and reflection. In this section, we first introduce an illuminance formula for refracted and/or reflected light [2]. Second, two previous methods for illuminance calculation are briefly reviewed, and their problems are pointed out. Finally, the two methods are combined into a new method we call *grid-pencil tracing*, which solves the problems of the previous methods.

[1] Prior to this, Nelson Max applied a similar technique to his animation 'Carla's Island'[7].

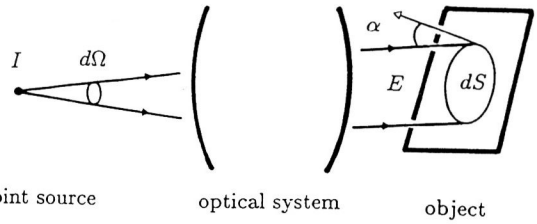


Figure 1 Illuminance formula

2.2.1 Illuminance formula

Consider a light beam emitted from a point source and passing through an optical system onto an object surface, as shown in Figure 1. Let the luminous intensity of the source be I, the illuminance on the surface E, and the transmittance of the system t_r. When the source and the object are surrounded by media of an optical index n_1 and n_2, respectively, energy conservation is represented by

$$n_1 t_r I d\Omega = n_2 E dS,$$

or

$$E = (n_1/n_2) t_r I (d\Omega/dS), \tag{1}$$

where $d\Omega$ is the emitted beam solid angle, and dS is the illuminated area on the surface. Note that $d\Omega/dS$ is quite similar to a form-factor in the radiosity model.

When a light beam transmits through the system with no refraction or reflection, $d\Omega/dS$ is calculated by

$$d\Omega/dS = cos\alpha/r^2, \tag{2}$$

which indicates the inverse square law, where r is the distance between the source and the surface, and α is the angle between the surface normal and the beam direction. However, because of light concentration due to refraction and/or reflection, Eq. (2) is not valid in general situations, thus requiring that $d\Omega/dS$ be calculated using other methods.

2.2.2 Previous methods

For calculating the $d\Omega/dS$ factor, two methods have been proposed: backward ray tracing and pencil tracing.

Backward ray tracing [6]

In backward ray tracing, Eq.(1) is calculated by tracing many rays from a point light source, where a ray is regarded as an energy quantum, or a photon. Consider the model shown in Figure 2. Rays, or photons, emitted from a point source at equal intervals, $\delta\omega_0$, go through an optical system, and arrive at an object surface. Since illuminance E is defined by incident energy per unit area, it can be given by counting photons coming into a given area and calculating the density of photons.

$$E \simeq e_0 m/\delta S, \tag{3}$$

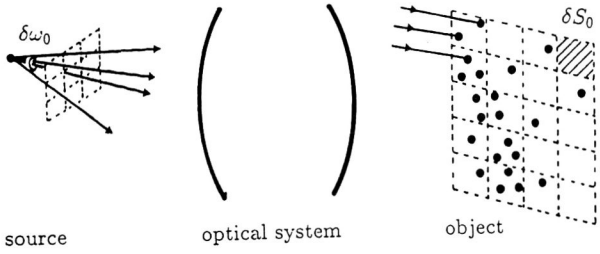


Figure 2 Backward ray tracing

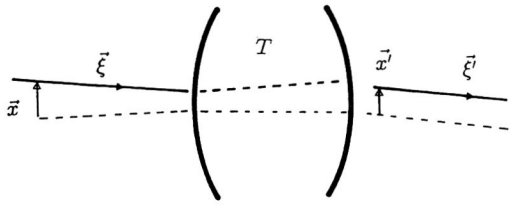


Figure 4 System matrix

where m is the number of photons coming to the area δS, and e_0 is the amount of energy carried by a photon, given by

$$e_0 = I(n_1/n_2)t_r\delta\omega_0.$$

In the sense of Eq. (1), this calculation of Eq. (3) can be understood as an approximation of $d\Omega/dS$,

$$d\Omega/dS \simeq (m\delta\omega_0)/\delta S. \tag{4}$$

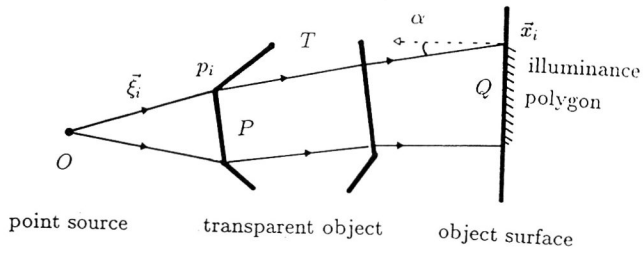
To evaluate Eq. (3), the surface is divided into a cellular array (Figure 2), and the photons in each cell are counted. This results in an illuminance map, which is used in the rendering process.

Unfortunately, however, Eq. (4) does not produce a good approximation unless the number of incoming rays m is sufficiently large, as can be seen in Figure 2. In low illuminance areas, a serious aliasing problem occurs. In fact, the number of incoming photons m assumes only one of two values, either 0 or 1, in a typical low illuminated region, which results in moire artifacts.

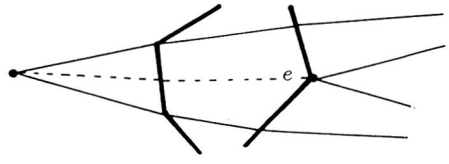
Figure 3 shows an example of light concentration simulated by backward ray tracing. Although 1024×1024 rays are traced from a point source in the pre-processing for creating a 512×512 illuminance map, a moire pattern is still noticeable in the low illuminance areas. Since the required CPU time for the pre-processing phase is 6.6 hours on VAX11/780, it is impractical to reduce the moire artifacts by increasing the number of sample rays.

Pencil tracing [2]

Pencil tracing is an application of the paraxial theory to computer graphics, where a pencil, or a bundle of rays, is traced using linear approximation. The approximation is formulated by a 4×4 matrix called a system matrix, which describes changes of output ray direction $\vec{\xi}'$ and position $\vec{x}'$ with respect to the changes of the input ray's $\vec{x}$ and $\vec{\xi}$



(a) Pencil tracing and illuminance polygon



(b) Pencil division

Figure 5 Pencil tracing

through an optical system, as shown in Figure 4. Mathematically, this is expressed by

$$\begin{pmatrix} \vec{x}' \\ \vec{\xi}' \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{\xi} \end{pmatrix}, \tag{5}$$

where the 2×2 matrices A, B, C, D can be analytically calculated from the parameters of a given optical system.

Illuminance patterns can be obtained by tracing all refracted and/or reflected pencils and calculating $d\Omega/dS$. For polygonal objects, this calculation can be efficiently conducted by using a system matrix. The procedure is briefly explained here referring to simple examples. For a more detailed discussion, see [2].

Consider the situation shown in Figure 5-a. An emitted pencil passing through a polygon P of a transparent object is refracted by the object, and then illuminates another object surface. A vertex of the illuminated area, $\vec{x}'_i$, is calculated using the system matrix as follows:

$$\vec{x}'_i = B\vec{\xi}_i,$$

where B is a 2×2 sub-matrix of the system matrix in Eq.(5), and $\vec{\xi}_i$ is the direction from the point source to the corresponding vertex, p_i, of the polygon P. The illuminance is calculated from

$$d\Omega/dS = cos\alpha|d\vec{\xi}/d\vec{x}'| = cos\alpha|1/det(B)|. \tag{6}$$

The illuminated area Q and the calculated illuminance is stored as an illuminance polygon, which is referred to in the rendering phase.

In the case of Figure 5-b, where a pencil intersects several polygons, the pencil must be divided. This can be conducted by transforming the polygon edge e by the matrix B^{-1} and dividing the pencil in $\vec{\xi}$-space. By repeating the procedure throughout all the directly visible polygons of

the transparent object, the illuminance calculation is completed.

The pencil tracing approach has the following three advantages:

- **Reduction of moire artifacts:** Using illuminance polygons with their area sampling feature, moire artifacts are reduced.
- **Computation cost reduction:** An expensive raytracing calculation is replaced by a simple matrix-vector production, which results in computation cost reduction.
- **Analytical calculation of $d\Omega/dS$:** The $d\Omega/dS$ factor is calculated from a system matrix, as shown in Eq.(6).

Figure 6 shows the same scene as Figure 3, produced by pencil tracing. Moire patterns observed in Figure 3 have been eliminated here. The computation time required for the illuminance calculation was only 200 seconds, which is about 120 times faster than backward ray tracing.

For polygonal objects, this method is very efficient. It is theoretically applicable to curved objects by adaptively dividing them into small polygons. However, this approach would be too expensive because of the increased overhead involved in dividing objects as well as the increased number of pencils that would have to be traced. Thus, there is still a requirement for an efficient technique applicable to curved transparent objects.

2.2.3 Grid-pencil tracing

By combining both pencil tracing and backward ray tracing, an efficient method can be realized that is applicable to any raytraceable object. The basic idea is to reduce moire artifacts by area sampling, or pencil tracing. Consider the situation shown in Figure 7. In backward ray tracing, each light ray, a, b, c, d is regarded as a photon, and the point sampling feature causes the moire problem. In order to solve the problem, we regard a ray as an edge of a pyramidal pencil Π. Then, a light pencil, Π, is formed with four rays, a, b, c, d, as its edges. The intersection between the pencil and an object surface is regarded as an illuminance polygon, and the illuminance is simply calculated by Eq. (1) and

$$d\Omega/dS = \delta\omega_0/\delta S, \tag{7}$$

where $\delta\omega_0$ is the pencil spread angle, or ray sampling interval, and δS is the illuminated area.

Tracing a grid of rays, or a grid of pencil edges, from a point source and calculating Eq. (7) results in all the necessary illuminance polygons. However, it is not practical to refer to a great number of illuminance polygons in the rendering process. Therefore, they are scan-converted into an illuminance map for easier access. In this scan-conversion process, linear interpolation of illuminance, or Gouraud shading, is applied to improve the smoothness. Since this method involves tracing a grid of pencils, we will refer to the method as 'grid-pencil tracing'.

Figure 8 shows an example image, where ripples on a water surfaces cause light concentration on the bottom of a swimming pool. The light source is a parallel source, and the wave pattern was represented by a periodic bump map, generated by Fourier synthesis [8]. In the pre-processing, 128×128 pencils were traced, which required only 7 seconds of CPU time on the VAX8840, (approximately 4 or 5 times faster than the VAX11/780). As shown in the figure, moire artifacts are not noticeable.

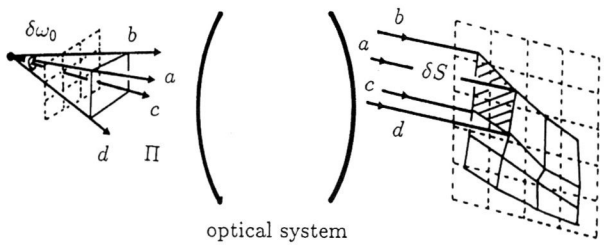


Figure 7 Grid-pencil tracing

2.3 Area source illumination

Although grid-pencil tracing works well for point light sources, the point source model is rather a poor lighting model for a photo-realistic simulation of indoor scenes. In this section, space-variant filtering is applied to grid-pencil tracing, so that it can deal with area sources.

2.3.1 Filter design

A straight-forward application of the grid-pencil tracing is conceivable as follows: first, sample points over an area source, then calculate an illuminance map for each point source by grid-pencil tracing, and finally sum up the calculated illuminance maps. However, this point sampling again would result in an aliasing problems: if the sampling rate in the area source is insufficient, the summation of the illuminance maps would look jagged, like a collection of discrete focuses, while increasing the sampling number is too expensive. We propose solving the aliasing problem by pre-filtering, maintaining the number of sample points at a reasonable level.

An effective filter can be designed in the following way. For simplicity, we will initially discuss a linear source, and then extend the discussion to area source illumination. Consider the situation shown in Figure 9, where a small linear

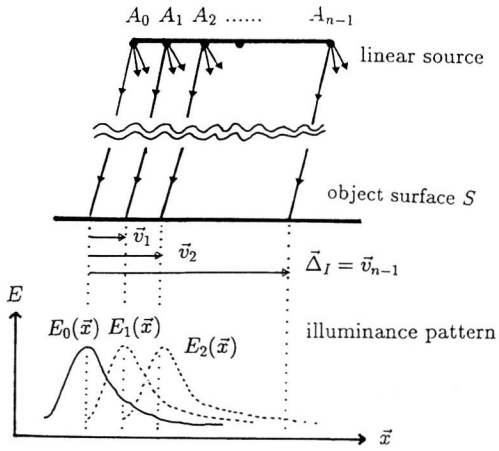


Figure 9 Linear light source and illuminance pattern

source A_0A_{n-1} illuminates the object surface S. The linear source is sampled into n point sources, $\{A_i\}_{i=0,n-1}$, where n is assumed to be large enough to approximate the linear source. Let the illuminance pattern of a sample point source A_i be E_i. The illuminance pattern of the linear source E_{line} can be calculated by

$$E_{line}(\vec{x}) = \sum_{i=0}^{n-1} E_i(\vec{x}). \tag{8}$$

When the length of the line A_0A_{n-1} is small, an illuminance pattern E_i can be approximated by shifting E_0 in the vicinity of a given position $\vec{x}$, represented as

$$E_i(\vec{x}) \simeq E_0(\vec{x} - \vec{v}_i). \tag{9}$$

The shift vector $\vec{v}_i$ is obtained as the displacement of a ray-surface intersection when the source position changes from A_0 to A_i while the emitted ray direction remains fixed (Figure 9). The vector $\vec{v}_i$ can be also approximated by

$$\vec{v}_i \simeq (i/n)\vec{v}_{n-1} = (i/n)\vec{\Delta}_I. \tag{10}$$

From Eqs. (8), (9), and (10), the total illuminance E_{line} can be approximated by

$$\begin{aligned} E_{line}(\vec{x}) &= \sum_i E_i(\vec{x}) \\ &\simeq \sum_i E_0(\vec{x} - (i/n)\vec{\Delta}_I) \\ &\simeq \sum_{\vec{x}'} nE_0(\vec{x} - \vec{x}')f(\vec{x}';\vec{x}), \end{aligned} \tag{11}$$

where the factor n is a factor from the summation variable substitution. The function $f(\vec{x}';\vec{x})$ is a normalized space-variant filter with a linear kernel $\vec{\Delta}_I$, defined by

$$f(\vec{x}';\vec{x}) = \begin{cases} 1/|\vec{\Delta}_I| & \text{if } \vec{x}' \text{ is on the vector } \vec{\Delta}_I(\vec{x}) \\ 0 & \text{otherwise} \end{cases}.$$

Equation (11) indicates that the total illuminance E_{line} is obtained simply by filtering an illuminance pattern of a sample point source, E_0. In other words, n sample points are reduced to one sample with the continuity of the illuminance pattern being maintained.

The discussion for a linear source can be readily extended to area sources by regarding a small area source as a collection of linear sources. Quite similarly, the summation is approximated by filtering an illuminance pattern of a single linear source, represented by

$$\begin{aligned} E_{area}(\vec{x}) &= \sum_j E_{line(j)}(\vec{x}) \\ &\simeq \sum_{\vec{x}'} E_{line(0)}(\vec{x} - \vec{x}')g(\vec{x}';\vec{x}) \\ &\simeq \sum_{\vec{x}'} g(\vec{x}';\vec{x})(\sum_{\vec{x}''} E_0(\vec{x} - \vec{x}' - \vec{x}'')f(\vec{x}'';\vec{x}')) \end{aligned} \tag{12}$$

where

$$g(\vec{x}';\vec{x}) = \begin{cases} 1/|\vec{\Delta}_J| & \text{if } \vec{x}' \text{ is on the vector } \vec{\Delta}_J(\vec{x}) \\ 0 & \text{otherwise} \end{cases}.$$

The resulting anti-aliasing filter is a pair of one-dimensional space-variant filters, $f(\vec{x}';\vec{x})$ and $g(\vec{x}';\vec{x})$, specified by displacement of ray-object intersections, $\vec{\Delta}_I$ and $\vec{\Delta}_J$. Note that Eq. (12) shows that the filtering can be achieved by separable convolution, which is computationally much cheaper than non-separable two-dimensional convolution.

2.3.2 Procedure

Illuminance calculation for an area source in a refracting environment is performed by dividing the source into a mesh, applying grid-pencil tracing and space-variant filtering. The outline of the procedure is as follows:

1) Initialize $E_{total} \leftarrow 0$.
2) Divide an area source into a mesh.
3) For each cell (i,j) in the mesh, do the following:
   a) Do grid-pencil tracing, to produce illuminance map $E_0^{i,j}$ and ray-object intersection points $\vec{p}\,_{k,l}^{\,i,j}$, where (k,l) specifies an original ray direction.
   b) Calculate the filter kernels $\vec{\Delta}_I$ and $\vec{\Delta}_J$ by subtraction, such that
   $$\begin{aligned} \vec{\Delta}_I(\vec{p}\,_{k,l}^{\,i,j}) &= \vec{p}\,_{k,l}^{\,i,j} - \vec{p}\,_{k,l}^{\,i+1,j}, \\ \vec{\Delta}_J(\vec{p}\,_{k,l}^{\,i,j}) &= \vec{p}\,_{k,l}^{\,i,j} - \vec{p}\,_{k,l}^{\,i,j+1}. \end{aligned}$$
   c) Do convolution of Eq. (12), resulting in $E_{area}^{i,j}(x)$
   d) $E_{total} += E_{area}^{i,j}$

2.3.3 Result

Photo-realistic rendering is achieved in a two-path process: in the pre-processing phase, an illuminance map is calculated by grid-pencil tracing and space-variant filtering. Then, distributed ray tracing is applied, referring to the prepared illuminance map. In the distributed ray tracing phase, illuminance for a direct (non-refracted/reflected) light component is calculated by a conventional stochastic sampling method, and illuminance for a refracted/reflected light component is obtained by referring to the prepared illuminance map.

Figure 10 shows an example image, simulating light concentration through a glass of water. The glass is illuminated by a rectangular light source of approximately 0.3(rad.)×0.1(rad.) in visual angle. The source was divided into a 7×7 mesh, for each sample of which 64×64 rays were traced. As seen in the picture, a light focus was realistically simulated. In the pre-processing, 1.9 minutes of CPU time was required on the VAX 8840, while the rendering phase required 67 minutes for 1024×682 pixels with 3×3 sub-pixel sampling. The wood texture was taken from a photograph.

2.4 Summary of the section

In this section, we have proposed effective methods for calculating illuminance patterns of light through transparent objects. The methods have been developed based on the ideas of pencil tracing and backward ray tracing, and have resulted in reduced aliasing artifacts with small computation cost. Depending on the light source - object shape mix, an appropriate method could be selected, as shown in Table 1.

Table 1: Appropriate methods for various situations

Type of light sources	Type of object shapes	Appropriate method
Point sources	Polygons	Pencil tracing
Point sources	Curved surfaces	Grid-pencil tracing
Area sources	Curved surfaces	Grid-pencil tracing with space-variant filtering

All of these methods can be used as pre-processing, interfaced by an illuminance map. Combining the methods with distributed ray tracing, realistic images of transparent objects can be synthesized. Since the calculation of an illuminance map is almost equivalent to a form factor calculation, the methods could be potentially applied to the radiosity model.

3 Pencil Tracing Dispersive Objects

A rainbow spectrum of light can be seen through transparent objects, such as a prism. The phenomenon is called 'dispersion,' and is one of the most lovely features of transparent objects. Simulation of dispersion not only provides colorful accents in pictures of transparent objects, but is even a critical process in some applications, such as the rendering of gemstones [9]. The chief obstacle to the simulation of dispersion is the large amount of computation required, considering that even the normal non-dispersive ray tracing calculation is expensive.

In this section, the theory of pencil tracing is extended to describe the dispersion effect, and a fast rendering method is proposed for dispersive objects, based on the theory.

3.1 Previous works

Dispersion occurs due to the fact that an optical index n of objects generally depends on wavelength λ of incident ray, causing a refracted ray angle change with respect to λ (Figure 11). This means that an image through a transparent object changes with wavelength, or color. Therefore, the simulation requires image calculation throughout the visible wavelength range from violet to red.

Takagi, et al, applied ray tracing to the simulation by super-sampling in the wavelength domain [9]. In their straightforward method, several (the suggested value is nine) monochromatic rays per pixel are sampled and individually traced, resulting in monochromatic images at the sampled wavelength. The calculated images are then converted into a normal (R,G,B) color image using color functions. The obvious disadvantage is that the computation cost is increased several (typically nine) times compared to conventional non-dispersive ray tracing.

Yuan, et al., tried to reduce the computation time by using 'wavelength coherency' for polygonal objects [10]. In this method, only three rays/pixel are traced, and other necessary information for various wavelength are interpolated. Although the use of 'wavelength coherency' is a good strategy, the computation even for this method is still far from practical; its limited efficiency lies in its usage of the conventional ray tracing calculation.

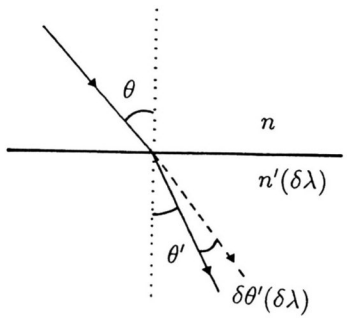


Figure 11 Dispersion

Since pencil tracing synthesizes non-dispersive transparent images several tens of times faster than conventional ray tracing, it is a promising simulation approach for reducing computational cost.

3.2 Theories and implementation

For the application, we first extend the system matrix so that it can describe the dispersion effect. Second, we introduce the concept of a pseudo-3D screen, quite similar to the spatio-temporal screen proposed by Grant [11] for spatio-temporal anti-aliasing. Using an extended system matrix and a pseudo-3D screen, pencil tracing can be easily applied to the simulation of dispersion.

3.2.1 Extended system matrix

As discussed in Section 2.2.2, a system matrix describes changes in a ray's direction $\vec{\xi}$ and intersection with objects $\vec{x}$ due to an optical system. Adding terms for changes with respect to wavelength, a system matrix can express the dispersion effect, represented by

$$\begin{pmatrix} \vec{x}' \\ \vec{\xi}' \\ \delta\lambda \end{pmatrix} = \begin{pmatrix} A & B & \vec{x}_\lambda \\ C & D & \vec{\xi}_\lambda \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{\xi} \\ \delta\lambda \end{pmatrix}, \qquad (13)$$

where A, B, C, D are 2×2 submatrices defined in Eq. (5), and $\delta\lambda$ is the difference of wavelength between a given ray and a reference ray (or an axial ray). The vectors $\vec{x}_\lambda$ and $\vec{\xi}_\lambda$ denote the changes of $\vec{x}'$ and $\vec{\xi}'$ with respect to wavelength λ, defined by

$$\vec{x}_\lambda = \partial\vec{x}'/\partial\lambda,$$
$$\vec{\xi}_\lambda = \partial\vec{\xi}'/\partial\lambda.$$

For example, in the case of a single refraction, as shown in Figure 11, $\partial\vec{x}/\partial\lambda$ is equal to 0 and $\partial\vec{\xi}/\partial\lambda$ is calculated from Snell's law, expressed by

$$nsin\theta = n'(\delta\lambda)sin(\theta' + \delta\theta'), \qquad (14)$$

or

$$sin(\theta' + \delta\theta') = (n/n'(\delta\lambda))sin\theta.$$

For a small $\delta\theta'$, Eq. (14) is approximated by

$$\sin\theta' + \cos\theta' \cdot \delta\theta' \simeq n/n'_0(1 - (dn'/d\lambda)\delta\lambda)\sin\theta,$$

or

$$\delta\theta' = (n/n'_0)(\sin\theta/\cos\theta')(dn'/d\lambda)\delta\lambda,$$

where $n'_0 = n'(0)$. On the other hand, both $\partial\vec{x}/\partial\lambda$ and $\partial\vec{\xi}/\partial\lambda$ are equal to 0 in the case of a single reflection.

3.2.2 Dispersion image solid

As was previously mentioned, the simulation of dispersion is an image creation process throughout the visible wavelength range. Hence, consider a pseudo-3D screen where the wavelength (λ) axis is added to a normal two-dimensional x-y screen (Figure 12). In the pseudo-3D screen, a dispersive image of a polygon is represented by a polygonal solid, as shown in Figure 12, where any slice of the solid parallel to the x-y plane is a monochromatic image of the polygon at a given wavelength. Let us call the solid a *dispersion image solid.*

A dispersion image solid can be easily calculated by an extended system matrix. Let the visible wavelength range be $[\lambda_0, \lambda_1]$, vertices of a given object polygon be $\{\vec{x'}_i\}$, and the system matrix be defined as in Eq. (13). A vertex of the dispersion image solid, $(\vec{X}_i, \lambda_j)$ is obtained by

$$\vec{X}_i \simeq S\vec{\xi}_i = SB^{-1}(\vec{x'}_i - \lambda_j\vec{x}_\lambda), \qquad (15)$$

where the 2×2 matrix S denotes the approximated relationship between an initial ray direction angle $\vec{\xi}_i$ and its corresponding (two-dimensional) screen point $\vec{X}_i$.

In the non-dispersive pencil tracing [2], visible polygons are calculated by two dimensional clipping on the screen using a priority list. Replacing two dimensional clipping by three dimensional clipping, the pencil tracing procedure can be easily extended to apply to dispersion image solids.

3.2.3 Transformation into an R-G-B image

Completing the extended process of pencil tracing, all visible dispersion image solids are obtained. However, since our goal is to present an image on an R-G-B color display, the solids must be transformed into an R-G-B image.

Figure 13 shows a cross section of a dispersion image solid on a scanline. First, we have to calculate the wavelength range $[\lambda_s, \lambda_e]$ of incoming light to each pixel x_i. This range can be incrementally calculated by referring to increments of wavelength already prepared for each polygon of the solid, much same as the scan-line algorithm. The R-G-B values at each pixel can be obtained by the following integrals:

$$R = \int_{\lambda_s}^{\lambda_e} t(\lambda)r(\lambda)L(\lambda)d\lambda,$$

$$G = \int_{\lambda_s}^{\lambda_e} t(\lambda)g(\lambda)L(\lambda)d\lambda,$$

$$B = \int_{\lambda_s}^{\lambda_e} t(\lambda)b(\lambda)L(\lambda)d\lambda,$$

where $L(\lambda)$ is a spectra of object light, r, g, b are proper color mapping functions, something like color matching functions of CIE, and $t(\lambda)$ is a spectral transparency. Note that

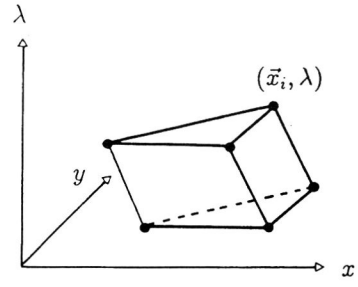


Figure 12 Dispersion image solid and pseudo-3D screen

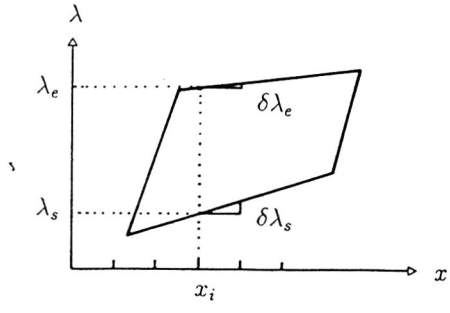


Figure 13 Scan-conversion of dispersion image solid

the integrals can be efficiently conducted by referring to integral tables when $L(\lambda)$ and $t(\lambda)$ are described by a linear combination of some function basis.

3.2.4 Selective dispersive pencil tracing

Since the factor $\vec{x}_\lambda$ in Eq. (15) denotes the amount of dispersion effect, we can estimate, in the system matrix calculation phase, whether dispersion is noticeable or not, as

$$\delta\vec{X}_{dis} = \lambda_1 SB^{-1}\vec{x}_\lambda.$$

When $|\delta\vec{X}_{dis}|$ is less than 1 pixel width, the normal non-dispersive pencil tracing is applied to the pencil for the reduction of overhead due to the extension of the process. This selective process further accelerates the execution without loss of created pictures.

3.3 Result and discussion

Figure 14 shows a sample image of a dispersive polyhedra with 120 polygons. The required CPU time to create the image was just 1.2 minutes on the VAX 8840. Since non-dispersive pencil tracing requires 1.0 minute for the same scene, the overhead for the dispersion simulation is very small. For simple scenes, non-dispersive pencil tracing is several tens times faster than conventional ray tracing, and thus the dispersive version is even far faster than conventional non-dispersive ray tracing. In fact, a conventional non-dispersive ray tracer took 57.9 minutes to simulate the same scene as Figure 14 without the dispersion effect.

Another advantage of the method is continuous representation of dispersion effects involved in the dispersive image solids. This allows reproduction of smooth color changes, while other super-sampling methods may cause step-function-like color changes.

The drawback of the method is the assumption of the linearity of dispersion, $dn/d\lambda \simeq$ constant, which is, in general, a rather poor approximation. However, we suppose that the resultant images would not look unnatural in most situations, just like the case of Gouraud shading, because it is difficult to detect the errors based on daily experiences. When all dispersive objects in a scene have the same optical index, however, we don't have to assume the linearity. In this case, it is possible to introduce a proper nonlinear scale of wavelength $\Lambda(\lambda)$, such that $dn/d\Lambda =$ constant. By using Λ instead of λ as the wavelength-axis in the pseudo-3D screen, the linear transformation can be applied to the projection.

The extended system matrix can also be applied to the illuminance calculation method discussed in Section 2.2.2, which enables the method to simulate colorful shadows of dispersive objects.

The extension discussed here is also applicable to beam tracing [12]. In that case, however, there is a problem with approximation accuracy especially for beams refracted nearly at a critical reflection angle, while such critical beams often cause significant dispersion.

4 Cross-Screen Filter

An important feature of transparent objects is their high reflectance. When a very bright light source is reflected, very bright spots (such as shimmer on water surfaces) are observed on the surface of transparent objects. However, it is difficult to effectively express such bright spots on a display with a limited brightness dynamic range (typically 256), because contrast between the reflection and other objects is usually extremely high. Figure 8 illustrates the difficulty. White spots on the water surface represent reflections of the sun. The calculated contrast between the sunshine reflection and the average illuminance in other areas is more than 3,200. In the figure, however, due to the poor image contrast, they just look like noise.

In such contexts, photographers attach a special-purpose filter called a 'cross-screen filter' to their camera, which emphasizes bright spots by producing star-like lines around the spot called asterism. Although a few computer-generated pictures have exploited this technique, no systematic methodology has been established. In this section, we analyze the optics in a cross-screen filter, and show that the phenomenon can be simulated by simple linear filtering.

4.1 Analysis

A cross-screen filter is a parallel glass plate with grid-like scratches, and refracted rays on the scratches cause asterism. Figure 15 illustrates a cross-screen filter attached to an imaging system, where it is assumed that the system is an ideal paraxial system[2]. For simplicity, we will assume the image plane is the focal plane.

First, consider the image of parallel incident rays in the direction of the system axis (z-axis). These rays intersect on the image plane as follows. The rays are

[2]An ideal paraxial system is regard as a linear system.

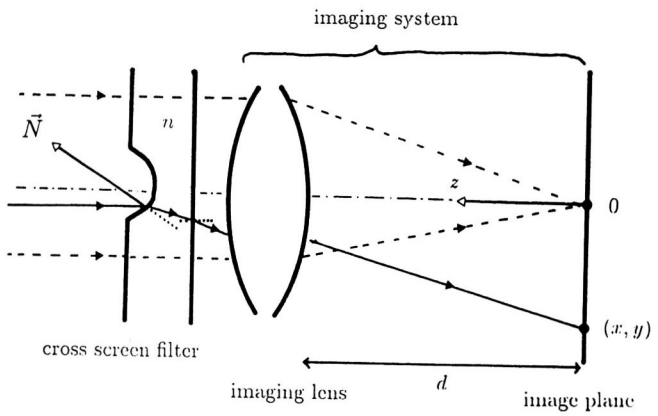


Figure 15 Cross-screen filter and imaging system

1) refracted on the front surface of the filter, of which normal is $\vec{N}$;
2) refracted on the back surface of the filter, of which normal is in the z-axis direction;
3) imaged at an image point (x, y) on the image plane through the imaging system.

The image point (x, y) is derived from a linear approximation of Snell's law, as

$$\begin{aligned} x &= (n-1)d \cdot N_x, \\ y &= (n-1)d \cdot N_y, \end{aligned}$$

where N_x and N_y are x- and y-components of $\vec{N}$, d is the distance between the image plane and the lens, and $n > 1$ is the optical index of the filter. Using a distribution function of normal vectors on the front filter surface $f(\vec{N})$, the illuminance E at the image point (x, y) is calculated by

$$\begin{aligned} E(x,y) &= f(\vec{N})(dN_x dN_y/dxdy) \\ &= f(\vec{N})/((n-1)d)^2. \end{aligned} \tag{16}$$

When the direction of the incident rays is not the z-axis direction, their illuminance pattern is simply given as a parallelly shifted pattern of Eq. (16) centered at their original focal point. This means that the simulation of a cross-screen filter can be completed by linearly filtering an original image with the function shown as Eq. (16).

4.2 Implementation

Procedure According to the analysis, we realize the simulation as a post-processing, as follows:

1) **rendering:** An original image $I_0(x, y)$ is created by a conventional renderer.
2) **linear filtering:** The original image is linearly filtered by Eq. (16), represented by

$$I_1(x_0, y_0) = \int I_0(x_0, y_0) E(x - x_0, y - y_0) dxdy.$$

3) **non-linear transformation:** Because the dynamic range of the final image is limited by a given display,

non-linear transformation of brightness is applied by

$$I_2(x,y) = \phi(I_1(x,y)).$$

Filter design The distribution function of normal vectors $f(\vec{N})$ can be modeled by considering the directions of scratch patterns, in a way quite similar to the isotropic reflection model [13]. Since most parts of the filter surface are flat with the surface normal in the z-direction, the distribution can be represented by

$$\begin{aligned} f(N_x,N_y) &= A\delta(N_x,N_y) + Bf_0(N_x,N_y), \\ \int f_0 dN_x dN_y &= 1, \\ A+B &= 1, \end{aligned}$$

where $\delta(N_x,N_y)$ is a two-dimensional delta function. Function f_0 presents the normal distribution in the scratched area, modeled by an appropriate function, such as a Gaussian function. Factor B represents the ratio of the scratched area.

4.3 Result

Figure 16 shows an example, applying the filtering technique to the original image of Figure 8. In this figure, reflection of the sunshine is effectively emphasized. The post-processing for this 512×512 image required only 8 seconds on the VAX8840.

The distribution function used here is a sum of two Gaussian functions, with $B = 0.07$.

5 Summary

We have proposed three methods for photo-realistic rendering of transparent objects. Each method simulates one of the following important phenomena due to refraction and reflection:

- **Caustics and focuses:** By combining pencil tracing and backward ray tracing, a method called *grid-pencil tracing* was developed for more efficient simulation of caustics and focuses of point sources. Furthermore, a space-variant filtering technique made grid-pencil tracing applicable to area-source illumination at reasonable computation cost.
- **Dispersion:** A system matrix was extended to describe the dispersion effect, allowing pencil tracing to efficiently simulate the effect.
- **Sunshine reflection:** Extremely bright spots are emphasized by the simulation of a cross-screen filter, realized by linear filtering.

Adopting these various techniques, realistic representation of transparent objects is very much enhanced at low computer cost.

Acknowledgments

We would like to thank Hiroshi Yasuda and Kei Takikawa for their continuous support. We also wish to thank Hitoshi Takakuwa for his assistance with the analysis of cross screen filters, Tadashi Naruse, Masaharu Yoshida, and Toshimitsu Tanaka for helpful discussion.

References

[1] Whitted, T., 'An Improved Illumination Model for Shaded Display,' Comm. ACM, **23**, No.6, pp.343-349, 1980.

[2] Shinya, M., Takahashi, T., and Naito, S., 'Principles and Applications of Pencil Tracing,' Computer Graphics, **21**, No.4, pp. 45-54, 1987.

[3] Goral, C.M., Torrance, K.E., Greenberg, D.P., Battaile, B., 'Modeling the Interaction of Light Between Diffuse Surfaces,', Computer Graphics, **18**, No.3, pp. 213-222, 1984.

[4] Wallace, J.R., Cohen, M.F., and Greenberg, D.P., 'A Two-Pass Solution to the Rendering Equation: A study of ray tracing and radiosity methods,' Computer Graphics, **21**, No.4, pp. 311-320, 1987.

[5] Kajiya, J.T., 'The Rendering Equation,' Computer Graphics, **20**, No.4, pp. 143-150, 1986.

[6] Arvo, J., 'Backward Ray Tracing,' Developments in Ray Tracing, SIGGRAPH Course Notes, **12**, 1986.

[7] Max, N. 'Carla's Island,' SIGGRAPH VIDEO REVIEW Issue #5, 1982

[8] Mastin, G.A., Watterberg, P.A., and Mareda, J.F., 'Fourier Synthesis of Ocean Scenes', IEEE CG&A, **7**, No.3, pp. 16-23, March 1987.

[9] Takagi, J., Yokoi, S., Tsuruoka, S., Kimura, F., and Miyake, Y., 'A Ray Tracing Model Considering Color Dispersion of Light', Proc. Graphics and CAD Symposium, pp. 81-87, Information Processing Society of Japan, 1984 (in Japanese).

[10] Yuan, Y., Kunii, T.L., and Sun, L., 'GemstoneFire: Adaptively Dispersive Ray Tracing of Polyhedrons', Visual Computer, **4**, No.5, pp.259-270, 1988.

[11] Grant, C.W., 'Integrated Analytic Spatial and Temporal Anti-aliasing for Polyhedra in 4-Space', Computer Graphics, **19**, No.3, pp.79-84, 1985.

[12] Heckbert, P.S., and Hanrahan, P., 'Beam Tracing Polygonal Objects,' Computer Graphics, **18**, No.3, pp.119-128, 1984.

[13] Takagi, J., Yokoi, S., Tsuruoka, S., Kimura, F., and Miyake, Y., 'Anisotropic Reflection Models,' IPSJ Tech. Report CAD-11-1, pp. 1-10, 1983 (in Japanese).

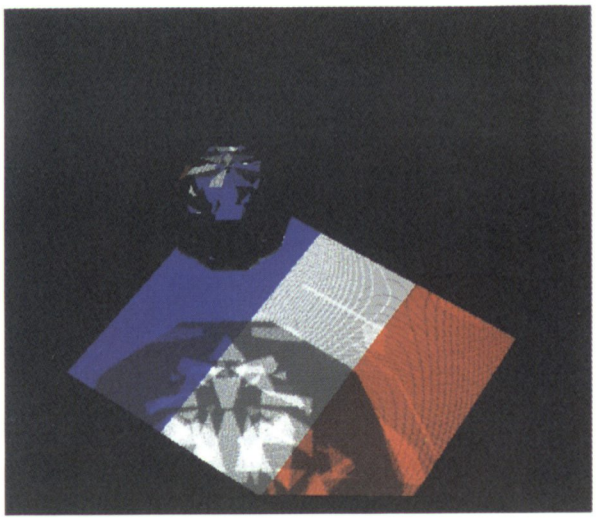


Figure 3

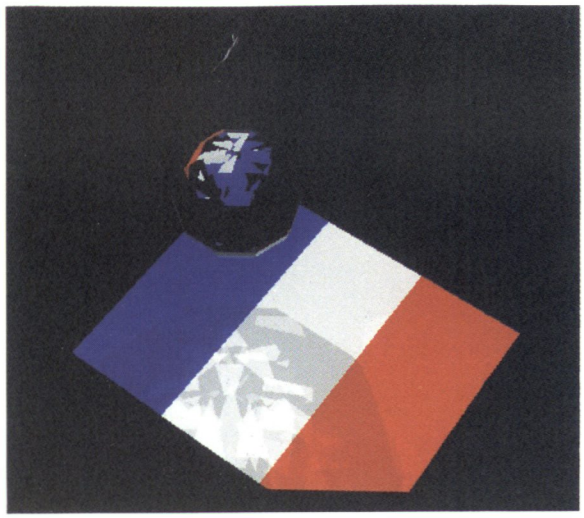


Figure 6

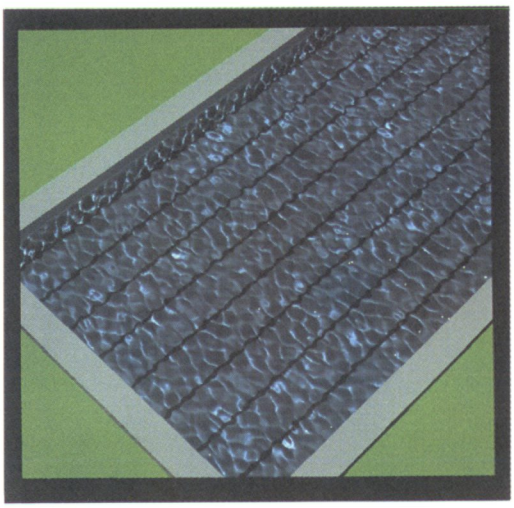


Figure 8

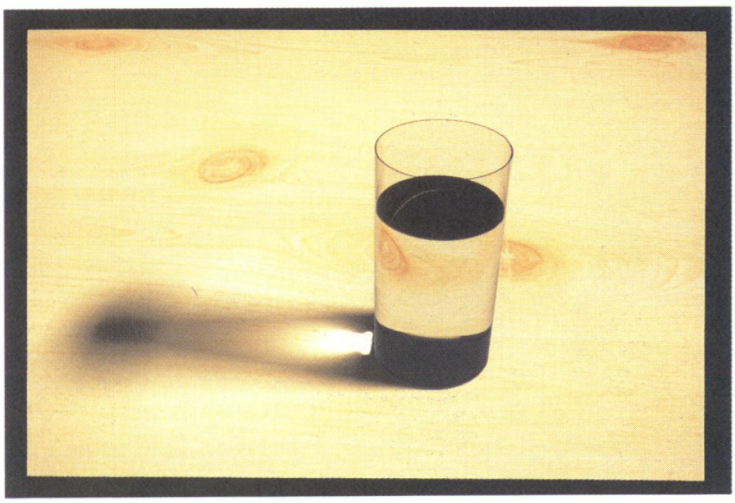


Figure 10

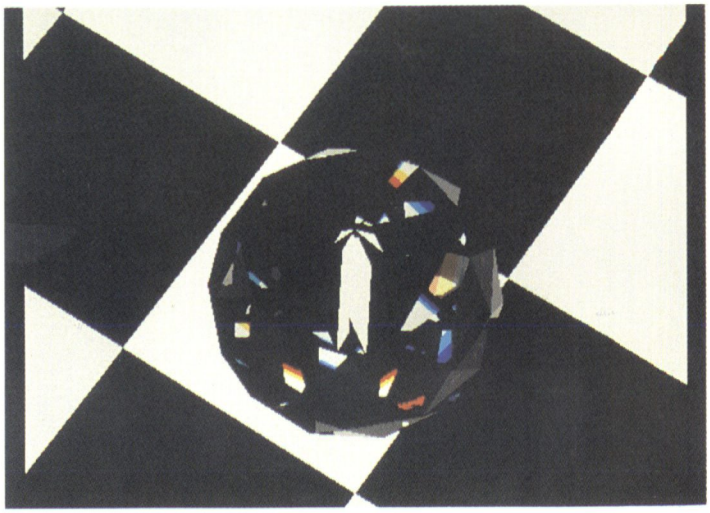


Figure 14

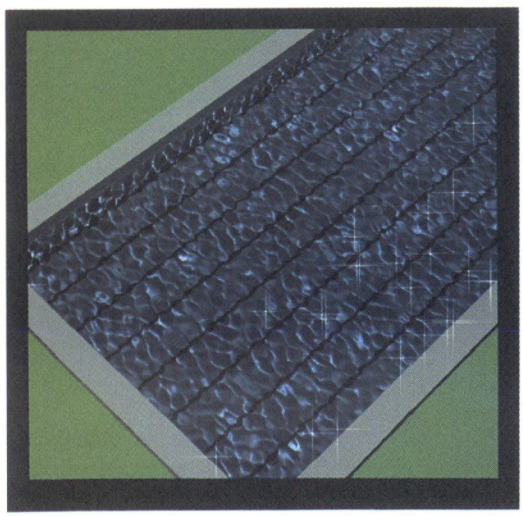


Figure 16