EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Interactive Simulation of the Human Eye Depth of Field and Its Correction by Spectacle Lenses

Masanori Kakimoto[1] Tomoaki Tatsukawa[1] Yukiteru Mukai[1] and Tomoyuki Nishita[2]

[1]SGI Japan, Ltd., Japan [2]The University of Tokyo, Japan

Abstract

This paper describes a fast rendering algorithm for verification of spectacle lens design. Our method simulates refraction corrections of astigmatism as well as myopia or presbyopia. Refraction and defocus are the main issues in the simulation. For refraction, our proposed method uses per-vertex basis ray tracing which warps the environment map and produces a real-time refracted image which is subjectively as good as ray tracing. Conventional defocus simulation was previously done by distribution ray tracing and a real-time solution was impossible. We introduce the concept of a blur field, which we use to displace every vertex according to its position. The blurring information is precomputed as a set of field values distributed to voxels which are formed by evenly subdividing the perspective projected space. The field values can be determined by tracing a wavefront from each voxel through the lens and the eye, and by evaluating the spread of light at the retina considering the best human accommodation effort. The blur field is stored as texture data and referred to by the vertex shader that displaces each vertex. With an interactive frame rate, blending the multiple rendering results produces a blurred image comparable to distribution ray tracing output.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1. Introduction

Reflection and refraction are fundamental themes in computer graphics. Ray tracing is a common solution to these subjects. Considering today's boost in computing power, ray tracing is still not fast enough to achieve a real-time result for practical applications. Meanwhile, approximations for reflection and refraction based on environment mapping with a GPU are used for major applications, especially for games. For applications of lens system design, such approximation methods are not used since they lack the accuracy that is required for the design processes. Ray tracing is still a major tool to simulate refraction for this purpose.

Spectacle lens designers, like most designers of other industrial products, use CAD applications to form the surface shape of their product. Their design task is a repeated processes of shape improvement and verification. The turnaround speed is getting better because of more sophisticated interactive tools enabling intuitive design of surface shapes. Unlike other industrial design areas, the turnaround of lens design is limited because ray tracing is in its verification stage.

A naked eye view (30.6fps) | A view through a concave lens (20.1fps) | A view through a convex lens (20.1fps)

Figure 1: *Simulation results of a patient model with both myopia and presbyopia.*

In spectacle lens verification this is especially noticeable because distribution ray tracing, which is several times or an order of magnitude slower than regular ray tracing, is mandatory for the simulation of a defocused image. Our

interactive display of human vision corrected by spectacle lenses dramatically changes the design process.

This paper presents a method of human eyesight simulation targeted especially for refraction correction with spectacle lenses. We introduce an interactive algorithm for reflection and refraction, which we call Per-vertex Ray Tracing. Recently, there are several other research works pursuing real-time plausible reflection and/or refraction. An advantage of ours is that the image is accurate at every vertex on the reflective or refractive object. Our algorithm enables the user to display 3D scenes through the lens in real-time with a sufficient refraction accuracy to simulate the distortion and color aberration of the lens.

Defocusing is a more critical issue for spectacle lens simulation. Many interactive applications employ a defocus effect that is often referred to as depth of field (DOF). Much work has been done for defocus effect with various camera models or image based models. However, these models lack human eye characteristics and are not applicable for DOF correction by spectacle lenses. There have been a small number of research activities on such correction. No previous work has been found which successfully realized real-time or interactive simulation of the human visual DOF phenomenon and its correction by lenses.

Two major obstacles are the performance of distribution ray tracing and the performance of human eye simulation. We solved the first issue with Per-vertex Ray Tracing and the second issue by precomputing the result of human eye and spectacle lens simulation. The result is stored as spatially distributed, volumetric information, which we call a *blur field*. In the rendering runtime each vertex of the objects in the scene is displaced according to its location in the blur field.

The blur field concept alone can be combined with distribution ray tracing to simulate eyesight or even other DOF models, provided that the primitives are polygon based and a routine is defined to displace each vertex instead of jittering the viewpoint.

This paper is organized as follows: In Section 2 we review some previous work on interactive refraction and human visual simulation. Section 3 describes a real-time refraction technique which we call per-vertex ray tracing. Section 4 introduces our human visual simulation model and the concept of blur field. Section 5 summarizes results and performance and Section 6 gives our conclusion with some discussion.

2. Previous Work

Since our proposed method uses real-time refraction, we first introduce related research on this topic. The second half of this section presents some previous work on the simulation of the human visual system and refraction correction.

2.1. Real-time Rendering with Refraction

Environment mapping [BN76] is a common technique to approximate reflection. Since it can easily be hardware-accelerated [HS93] [VF94], environment mapping became widely accepted for reflection effects. Real-time refraction mapping techniques were derived [LKM01] [Wym05] but they are limited to refraction for distant objects, which is a critical limitation of environment mapping. Hakura [HSL01] used a layered map to support local objects. Ohbuchi [Ohb03] realized a real-time method using refraction rays. Both methods map the environment objects to individual image planes and the accuracy is limited. To improve the accuracy of the local environment mapping, distance impostors [SKALP05] were introduced. Using the depth channel of the environment map, they approximated the ray hit points by iterative computing. Their method is fast and accurate for large-planar objects but the accuracy decreases for complex environment objects.

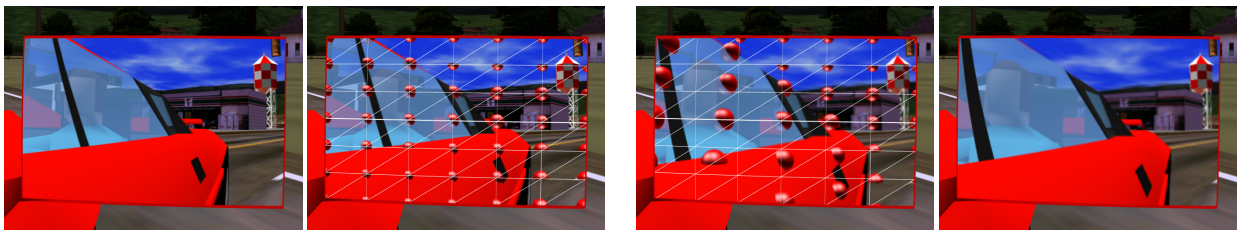
Unlike other real-time refraction methods, ours computes per vertex ray-scene intersections to obtain accurate 3D coordinates. Although the performance penalty is higher than distance impostors, it still runs in 30fps for practical scenes. More importantly for distortion evaluation, except for a case of occlusion discrepancy, which virtually does not occur in lens simulation, our method guarantees the refraction accuracy at each mesh point of the refractive surface.

2.2. Simulation of Correction of Human Visual Systems

A number of researchers have proposed DOF models [PC81] [Shi94] [KMH95]. Distribution ray tracing [CPC84] is one of the most popular DOF techniques for off-line rendering. Accumulation buffer [HA90] is a hardware mechanism which realizes real-time DOF.

Some research work has been done on DOF taking the human eye model into account. Santamaria et al. [SAB87] measured the human eye as a lens system to acquire a point spread function and tried to reconstruct the image on the retina. Mostafawy [MKL97] introduced a thick lens model of a human eye and simulated refractive surgery effects. Loos et al. [LSS98] used wavefront tracing to evaluate the human eye accommodation and visualized refractive correction by progressive lenses. Both of the above use distribution ray tracing and real-time processing is impossible. Barsky [Bar04] advocated the concept of Vision Realistic Rendering and introduced a DOF technique using aberration information measured from a patient's optical system.

Our proposed method produces a similar type of result to Loos' or Barsky's but the performance is enhanced by several orders of magnitude, for example 20fps vs 0.002fps.



Per-vertex ray tracing Cube mapping

Figure 3: *An example of reflection in a car door mirror model. In the central two images a wireframe model of the mirror is overlaid to show the vertex positions. Each red sphere has been located at the 3D intersection of each reflection vector and the scene, and is reflected in the mirror. The intersections are reflected exactly at the corresponding mirror vertices in the images of per-vertex ray tracing while they are not in the cube mapping images.*

3. Per-vertex Ray Tracing

The basic idea is to warp the environment map so that each vertex reflects the accurate scene intersection point of the extension of its reflection vector.

3.1. Algorithm for Each Frame

The processing step in each frame is as follows: The item numbers correspond to those appearing in Figure 2.

1. Compute reflection (refraction) vector at each vertex of the reflective (refractive) objects sharing an environment map.
2. Compute a virtual viewpoint, form a view volume, and render an intermediate image to be used as an environment map.
3. For each vertex of the reflective (refractive) objects:
   a. Trace the extension of the reflection (refraction) vector to find the nearest intersection with the scene.
   b. Find the line that connects the intersecting point and the virtual viewpoint.
   c. Find the point where the line intersects with the projection plane for the environment map.
   d. Assign the coordinates of the point in the environment map to the texture coordinates of the relevant vertex.
4. Render the scene and the reflective (refractive) objects from the original viewpoint.

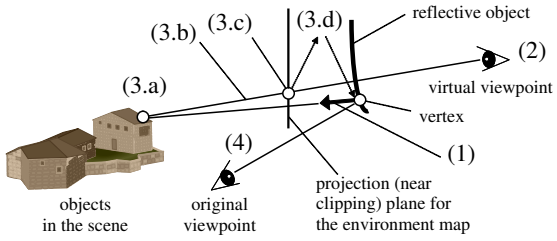


Figure 2: *Processing steps of the per-vertex ray tracing.*

In step 2 a least-squares approach is used to quickly find the virtual viewpoint that best represents the reflection (refraction) rays from the vertices.

With this algorithm the accurate intersection point is reflected at each vertex of the reflective object. The only exception is occlusion inconsistency potentially caused by the difference between the viewing ray from each vertex and that from the virtual viewpoint. Its possibility should, however, be minimized by the least-squares method. In addition, the possibility is even more negligible for spectacle lens simulation, in which case the distributions of the vertex positions and the refraction vector directions are moderate and limited.

For refractive lens objects, the process is identical to that in Figure 2 except that each refraction vector is computed with double refraction at the back and front faces of the lens.

3.2. Results of reflection and refraction

Figure 3 presents a sample result of reflection using Per-vertex Ray Tracing with a cube mapping reference. While local reflections yield apparent errors in the cube mapping example, the per-vertex ray tracing results in an exact matching at each vertex.

A rendering example of double-refraction through a lens is shown in Figure 4.

This is also an example of color aberration of the lens. The right image of Figure 4 is a close-up of the edge of the lens, where the color difference is apparent. The color aberration simulation is accomplished by rendering the lens three times, each with one of the three separate sets of texture coordinates computed by iterating step 3 described in Section 3.1 using different refraction indices (and thus different sets of refraction rays) for red, green, or blue. Each of the above lens rendering tasks is carried out with appropriate color mask settings to make a color-shifted refraction image over the lens surface. Note that all three tasks share the first pass result, i.e., the environment map, and thus the extra computation for color aberration is minimized.

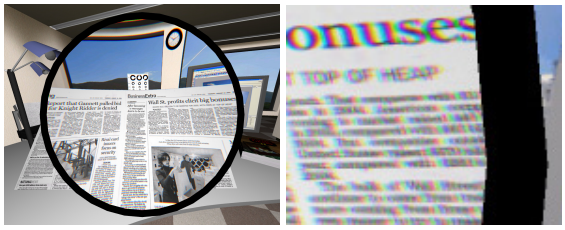


Figure 4: *Color aberration exhibited near the edge of a convex lens. RGB's are separately rendered for the lens surface using different refractive indices (R:1.60, G:1.61, B:1.62).*

4. Defocus Using Blur Fields

The basic idea for defocus is to displace the objects on a per vertex basis, according to the vertex location relative to the lens and the eye. We introduce a concept of *blur field*, a precomputed 3D spatial distribution of shape information which defines how a specific place should be blurred.

We define the blur field in the normalized device coordinate space, by subdividing in x, y, and z directions. Each subdivided box, which we call a *voxel*, contains a blur field value as a result of the precomputation.

We assume that the scene objects consist of polygon mesh models. Since the defocus operation is carried out per vertex, each model needs to be tessellated with sufficient granularity to produce a desired quality.

This section describes both the precomputation process (Sections 4.1-4.4) and the displacement operation (Section 4.5) in the rendering loop. The outline of the process is described in Figure 5 and Figure 6.

4.1. Blur Field

4.1.1. Use of the Normalized Device Coordinate System

The view volume for the refraction map is used as the space for the blur field. After the projection transformation and perspective division, the view volume is mapped to the normalized device coordinate system (NDC). The blur field is defined in the NDC and each sampled value is stored in a voxel. The voxels are formed by evenly subdividing the whole NDC space.

An advantage of using NDC is its compatibility with the perspective projection. Places closer to the viewpoint in the world coordinate space are more precisely subdivided than places farther from the viewpoint. This is a similar concept to that of perspective shadow maps [SD02].

Another major advantage is rendering performance. Since the vertex shader always transforms all vertices to the clipping space, which is very close to the NDC, the determination of the voxel in which each vertex resides will require a simple arithmetic rather than a 4×4 matrix transformation. Section 4.5 describes the details.

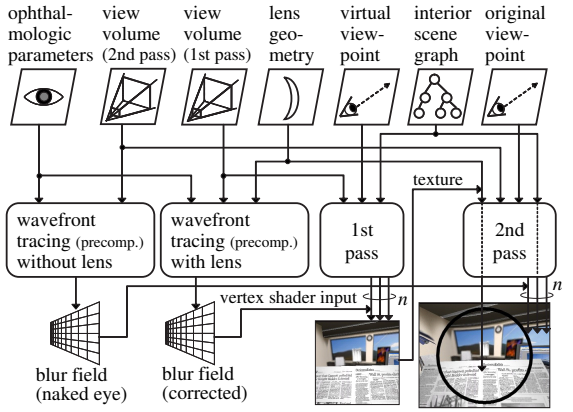


Figure 5: *The whole process of the method. The n is the number of jittered images to blend.*

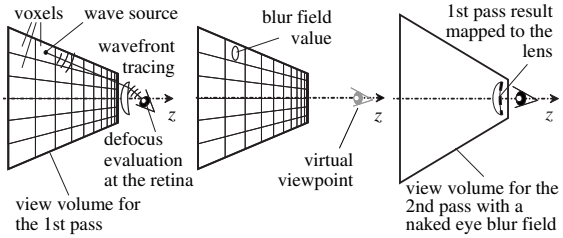


Figure 6: *The outline of the defocus processing flow. As a result of the precomputation, an ellipsoid shape information is stored for each voxel. The 1st pass consists of several separate rendering passes applying, in turn, a displacement vector within the ellipsoid to every vertex in the scene. Their results are blended onto the accumulation or auxiliary buffer. The blurred image is mapped to the lens surface in the 2nd pass using the per-vertex ray tracing technique. Note that we assume that the lens position relative to the viewpoint does not change during the rendering loop.*

4.1.2. Voxel information

Each voxel contains region information which defines the limit of the displacement of the vertices residing in the voxel. The region is determined by an ellipsoid whose size and orientation are computed by wavefront tracing. The wave emanates from a point in the voxel through the spectacle lens and the eye lens, and finally onto the retina. Details are presented in Sections 4.3 and 4.4.

4.2. Finding Light Ray Paths

To trace a wavefront, we need to know the path along which the ray from the center of the voxel travels through the spectacle lens into the eye, as indicated in Figure 6(a). It is not a trivial problem to find the light paths often described as Fermat's Principle. We solved this using an image processing approach.

First, a 2D (x-y) array of voxels at a certain depth (z) is rendered as a 2D array of rectangles through the lens with per-vertex ray tracing, using the same viewpoint position as in rendering the current frame (2nd pass). The background is set to black and the color of each rectangle is made unique.

A simple image processing of the rendered result can identify the pixels corresponding to the centers of the voxels. Each of these pixels corresponds to a point on the projection plane, in the world coordinate system. A ray from the viewpoint to such a point should travel through the lens being refracted twice, and then reach to the center of the voxel. This double-refracted ray is a very good approximation of the path obeying Fermat's Principle, connecting the viewpoint and the center of the voxel. In addition, in most cases for lenses, the double-refracted ray strikes through all voxels of different depths but at the same x-y position. Thus, the refracted light ray path can be shared among such voxels as line up along the z direction. To compute a blur field value for a voxel, a wave along this path is traced back from an appropriate passing point in the voxel. In this paper, we call this starting point for each voxel a *wave source*.

4.3. Wavefront Tracing

Wavefront tracing [Kne64] [Sta72] is a general tool to simulate traveling light rays. It was applied to produce caustics from reflective surfaces [MH92] [Elb94]. Loos et al. [LSS98] used it to evaluate the human eye accommodation for each pixel. We used wavefront tracing more directly than Loos et al. in order not only to evaluate accommodation but also to make a defocus spread pattern on the retina by deriving new wavefront operations, which are described in Section 4.4.

The wavefront at a point can be represented by the direction of the wave movement, i.e., a normal vector $\mathbf{N}$, two principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$ which are perpendicular with each other, and corresponding principal curvatures κ_1 and κ_2. Note that a curvature is negative when the center of curvature is in the opposite direction of the wave movement.

In spreading or transferring by a distance d, a wavefront is simply transformed as follows.

$$\kappa_1' = \frac{\kappa_1}{1 - d\,\kappa_1}, \quad \kappa_2' = \frac{\kappa_2}{1 - d\,\kappa_2}. \tag{1}$$

Here, the vectors $\mathbf{N}$, $\mathbf{e}_1$, and $\mathbf{e}_2$ do not change.

The transformation for refraction is derived from Snell's law and is fully described in the references [Kne64] [Sta72]. We present only the result here as Mitchell [MH92] and Loos [LSS98] described. The direction vector $\mathbf{N}$ is transformed as $\mathbf{N}' = \mu\mathbf{N} + \gamma\mathbf{N}^{(s)}$, where $\mathbf{N}^{(s)}$ is the unit normal vector of the refractive surface and $\mu = \frac{n_1}{n_2}, \gamma = -\mu\cos\varphi + \cos\varphi'$. n_1 and n_2 are the indices of refraction of the media before and behind the surface, respectively. φ and φ' denote the angle of incidence and the angle of refraction, respectively. When a unit vector $\mathbf{u} \propto \mathbf{N} \times \mathbf{N}^{(s)}$ is given, which is tangent both to the incident wavefront and to the surface, curvatures with respect to $\mathbf{u}$ can be obtained as follows. Using another unit vector $\mathbf{v} = \mathbf{N} \times \mathbf{u}$ and the angle ω between $\mathbf{u}$ and the first principal direction $\mathbf{e}_1$, i.e.,

$$\mathbf{u} = \cos\omega\mathbf{e}_1 - \sin\omega\mathbf{e}_2, \quad \mathbf{v} = \sin\omega\mathbf{e}_1 + \cos\omega\mathbf{e}_2, \tag{2}$$

Euler's Formulas

$$\begin{aligned} \kappa_u &= \kappa_1\cos^2\omega + \kappa_2\sin^2\omega \\ \kappa_v &= \kappa_1\sin^2\omega + \kappa_2\cos^2\omega \\ \kappa_{uv} &= (\kappa_1 - \kappa_2)\cos\omega\sin\omega \end{aligned} \tag{3}$$

determine the directional curvatures κ_u, κ_v and the torsion κ_{uv} on the incident wave. By refraction, curvatures of the wavefront are transformed as:

$$\begin{aligned} \kappa_u' &= \mu\kappa_u + \gamma\kappa_u^{(s)} \\ \kappa_v' &= \mu\kappa_v\frac{\cos^2\varphi}{\cos^2\varphi'} + \gamma\kappa_u^{(s)}\frac{1}{\cos^2\varphi'} \\ \kappa_{uv}' &= \mu\kappa_{uv}\frac{\cos\varphi}{\cos\varphi'} + \gamma\kappa_{uv}^{(s)}\frac{1}{\cos\varphi'}, \end{aligned} \tag{4}$$

where $\kappa_u^{(s)}$, $\kappa_v^{(s)}$, and $\kappa_{uv}^{(s)}$ are, respectively, the directional curvatures and the torsion of the refractive surface at the incident point. They can be computed using Equation 3 similarly to the wavefront. To represent the wavefront after the transformation, we need to find the principal curvatures and directions by inverting Equation 3.

$$\begin{aligned} \kappa_1 &= \kappa_u\cos^2\omega + 2\kappa_{uv}\cos\omega\sin\omega + \kappa_v\sin^2\omega \\ \kappa_2 &= \kappa_u\sin^2\omega - 2\kappa_{uv}\cos\omega\sin\omega + \kappa_v\cos^2\omega \\ \tan 2\omega &= \frac{2\kappa_{uv}}{\kappa_u - \kappa_v}. \end{aligned} \tag{5}$$

Table 1 presents the usage of wavefront tracing operations in the proposed method, showing which of the operations takes place in which turn during the computation of the blur field for each voxel.

Table 1: *The process of precomputation for each voxel. These operations take place along the double-refracted ray from the wave source of each voxel to the viewpoint.*

place or media		operation	equation
wave source		wavefront creation	
air		transfer	(1)
spectacle lens	front	refraction	(3) (4) (5)
	glass	transfer	(1)
	back	refraction	(3) (4) (5)
air		transfer	(1)
eye	pupil	accommodation	(13) (15)
		vergence	(3) (7) (5)
	retina	conoid sectioning	(8)
voxel		scaling to NDC	(9)

4.4. Eye Model

To simulate vision with astigmatic eyes, Loos et al. [LSS98] used distribution ray tracing. We continued to use wavefront tracing for the region within the eye by introducing new wavefront operations which are a transformation by a thin lens and a sectioning. Our model transforms the wavefront at the eye lens to form a revolved conoid shape and sections the shape at the retina to acquire correct defocus or a point spread shape which will be stored as a value of the blur field. We assume that the eye as a lens system is a thin toric lens, i.e., having orientation dependent refractive powers which are uniform over the lens surface. From an ophthalmological perspective, our eye model can treat regular or oblique astigmatism in addition to myopia, hyperopia, and presbyopia.

4.4.1. Wavefront Transformation by a Thin Toric Lens

Wavefront transformation in the eye is a complex phenomenon. In addition to approximating the eye as a thin toric lens, we assumed that the incident light ray direction is perpendicular to the lens. This assumption makes sense since the eyeball rotates to the region of interest so that the ray converges at the central retinal fovea. This subsection introduces a simple transformation of the incident wavefront and derives a refracted wavefront which will converge at the retina.

After refractions by the spectacle lens and a translation to the surface of the cornea, we obtain a wavefront with principal curvatures κ_1 and κ_2 and the principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively. As illustrated in Figure 7, we define an eye lens coordinate system as a right-hand coordinate system with its origin at the incident point and the z axis aligned with the wavefront normal vector. The x axis is chosen to be both horizontal and tangent to the wavefront and the lens. For convenience the direction of positive x is determined such that the y axis points upward. Since any tangent vector on the incident wavefront will be a common tangent vector with the lens, we can choose the x axis as a common tangent, which was denoted as $\mathbf{u}$ in Section 4.3, in order to represent the wavefront for the transformation by the thin toric lens.

Let ω_1 be the angle between the x axis and the first principal direction $\mathbf{e}_1$, and we can apply Equation 3, substituting $\omega = -\omega_1$, to obtain the directional curvatures κ_x, κ_y, and κ_{xy}.

For the wavefront transformation at the surface of the eye lens, we employ the vergence formula instead of Snell's Law. Here, we introduce directional refractive powers of the toric lens as a set of input parameters of the astigmatic eye.

Given the principal refractive powers P_1 and P_2, and the angle between the x axis and the first principal direction $\omega_1^{(e)}$, we find the directional refractive powers P_x (in the x-z plane), P_y (in the y-z plane), and P_{xy} by applying Equation 3 substituting $\kappa_1 = P_1$, $\kappa_2 = P_2$, and $\omega = -\omega_1^{(e)}$.

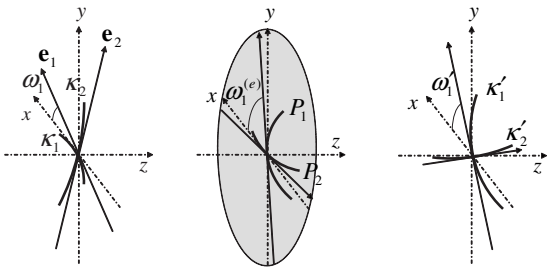


Figure 7: *Wavefront transformation by an astigmatic eye with principal refractive powers P_1 and P_2.*

In optical science, the basic vergence formula is

$$V + P = V', \tag{6}$$

where V is the vergence of the incident light wave and can be denoted as $V = n\kappa$ by definition, with the refractive index n of the media of the incident side and the curvature κ at the incident point, P is the refractive power of the lens, and V' is the vergence of the outgoing light. We extended this equation for directional curvature and derived formulas for the wavefront transformation by an eye as a toric lens, yielding

$$\kappa_x' = \mu\kappa_x + \frac{P_x}{n_2}, \quad \kappa_y' = \mu\kappa_y + \frac{P_y}{n_2}, \quad \kappa_{xy}' = \mu\kappa_{xy} + \frac{P_{xy}}{n_2}, \tag{7}$$

where $\mu = \frac{n_1}{n_2}$, n_1 is the refractive index of the air, and n_2 is that of vitreous humor of the eye.

We find the final wavefront to converge at the retina by using Equation 5, with $\kappa_u = \kappa_x'$, $\kappa_v = \kappa_y'$, and $\kappa_{uv} = \kappa_{xy}'$, obtaining the principal curvatures κ_1' and κ_2', and the angle of the first principal direction $\omega_1' = -\omega$.

4.4.2. Evaluation of Light Spread at the Retina

In general the light wave as derived above, when refracted by a toric lens, does not converge to a point. After incoming through a circular pupil of a certain size, the transferring light forms a conic-like shape known as Sturm's conoid, yielding two focal lines being perpendicular with each other as illustrated in Figure 8. Two main issues here are (1) the analysis of Sturm's Conoid and (2) the human eye accommodation. This subsection describes how to find, given a pupil and a static light wave forming a conoid, the shape of the light spread projected onto the retina, whereas the next subsection discusses the accommodation.

For simpler observation of the Sturm's conoid, we ignore for a while the rotation of the principal directions about z-axis, i.e., we assume $\omega_1' = 0$. To simulate the spread of the projected light spot, we take a cross-section of the conoid with a plane perpendicular to z-axis. By setting z as a constant parameter z_e for the distance between the cornea and the retina, we obtain an implicit equation of an ellipsoid

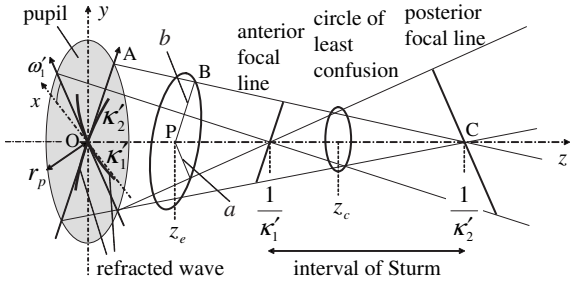


Figure 8: *Sturm's conoid formed by a toric lens.*

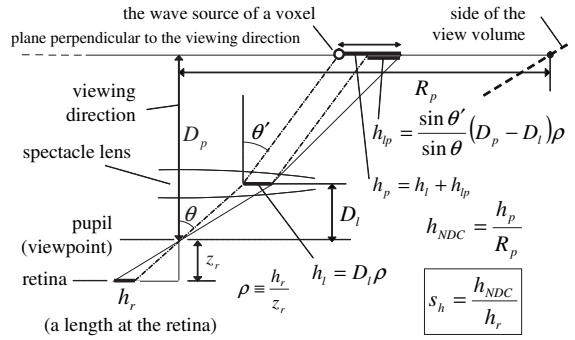


Figure 9: *Scaling factor to convert a horizontal length h_r at the retina to h_p at the wave source of a voxel and to h_{NDC} in the normalized device coordinate. The refraction at the spectacle lens is taken into account. The lens is approximated as a thin lens in this case.*

$S(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, where

$$a = r_p(1-\kappa_1' z_e), \quad b = r_p(1-\kappa_2' z_e), \tag{8}$$

and r_p is the radius of the pupil. The above a and b can be derived from the proportional relationship among the sides of several of the right-angled triangles along z-axis (e.g. $\triangle$AOC and $\triangle$BPC in Figure 8). The parameters κ_1' and κ_2' should be what the system had determined taking into account the accommodation as will be explained in Subsection 4.4.3.

At the retina, where $z_e = z_r$, the ellipse shape described by a, b, and ω_1' represents the point spread of the light source. When the light wave from the wave source of a voxel is traced and eventually results in an ellipsoid at the retina, we treat the shape as an approximated defocusing pattern which any point light source in the voxel should produce, and register the information for the voxel. Instead of registering a, b, and ω_1', an equivalent 2×2 matrix which should transform a unit circle into the oriented ellipsoid is stored as the voxel value. This precomputed matrix for each voxel is the entity of the blur field.

To determine the scaling factor of the matrix, we need to consider a geometrical relationship between the retina and the normalized device coordinate system (NDC), the scale of which is equivalent to that of the voxel space.

Let s_h and s_v be the scaling factors from the retina to the NDC for horizontal and vertical directions respectively, and we obtain the final blur field value $\mathbf{B}_{v_x,v_y,v_z}$ at a voxel (v_x, v_y, v_z) as:

$$\mathbf{B}_{v_x,v_y,v_z} = \begin{pmatrix} s_h & 0 \\ 0 & s_v \end{pmatrix} \begin{pmatrix} \cos\omega_1' & -\sin\omega_1' \\ \sin\omega_1' & \cos\omega_1' \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}. \tag{9}$$

Figure 9 illustrates how s_h is computed. A small horizontal length h_r at the retina should be mapped to a length in the normalized device coordinate system (NDC) h_{NDC}. The scaling factor s_h can be computed by h_{NDC}/h_r. The vertical scaling factor s_v is computed similarly.

Note that every variable in the right-hand side of Equation 9 varies by the individual voxel. The result of the multiplication of the three 2×2 matrices is stored in each voxel.

4.4.3. Accommodation of the Eye

It is the common assumption that the eye tries to accommodate so as to align the circle of least confusion on the retina. Let z_r be the distance between the thin lens of our eye model and the retina, and the above condition is satisfied by letting $a = -b$ in Equation 8. Thus we obtain

$$z_r = \frac{2}{\kappa_1' + \kappa_2'}. \tag{10}$$

We can expand the above formula by using Equations 3, 7, 4, and 5 in turn and using the fact $n_1 = 1.0$, to yield

$$z_r = \frac{n_2}{\bar{\kappa} + \bar{P}}, \tag{11}$$

where $\bar{\kappa} = \frac{\kappa_1+\kappa_2}{2}$ and $\bar{P} = \frac{P_x+P_y}{2}$. Optically $\bar{\kappa}$ is the average curvature of the incident wavefront to the eye and $\bar{P}$ is the average refractive power of the eye.

Let $\bar{P}_{min}$ be the standard non-astigmatic refractive power of the eye in minimum (relaxed) accommodation, and let ΔP be the accommodation refractive power limited by the maximum accommodation ability Δ_{max}, i.e., $\Delta P \in [0, \Delta_{max}]$, then

$$\bar{P} = \bar{P}_{min} + \Delta P. \tag{12}$$

From Equations 11 and 12, we obtain

$$\Delta P = \max\left(0,\ \min\left(\frac{n_2}{z_r} - \bar{\kappa} - \bar{P}_{min},\ \Delta_{max}\right)\right). \tag{13}$$

The meaning of the above accommodation formula is as follows. When the mean curvature of the incident light $\bar{\kappa}$, which is usually negative, is smaller ($-\bar{\kappa}$ is greater), i.e., the light wave comes from a closer location to the eye, the accommodation effort ΔP becomes greater, and vice versa. A standard value for Δ_{max} is presented in Table 2 but it declines throughout life.

Let A be the user-defined astigmatism, i.e., $A = P_{1,min} - P_{2,min}$, where $P_{1,min}$ and $P_{2,min}$ are the principal refractive

powers of the eye in minimum accommodation and it is natural to assume $\bar{P}_{min} = \frac{P_{1,min}+P_{2,min}}{2}$, then

$$P_{1,min} = \bar{P}_{min} + \frac{A}{2}, \quad P_{2,min} = \bar{P}_{min} - \frac{A}{2}, \tag{14}$$

and, by definition, we obtain the principal refractive powers of the eye taking account of accommodation,

$$P_1 = P_{1,min} + \Delta P, \quad P_2 = P_{2,min} + \Delta P. \tag{15}$$

The wavefront tracer will receive the above two values with the user-defined parameter $\omega_1^{(e)}$ in order to carry on tracing as was described in the second half of Subsection 4.4.1.

Table 2 shows the standard parameters of the Gullstrand and Helmholtz's schematic eye [GvH09] which we used in the implementation. Table 3 indicates which of the parameters should be controlled to simulate which of the the human vision symptoms.

Table 2: *Values of the Gullstrand and Helmholtz's schematic eye. The refractive power of maximum accommodation should be reduced according to the age.*

parameter		symbol	value
refractive power	minimal accommodation	$\bar{P}_{min}$	58.64 (D)
	maximal accommodation ability	Δ_{max}	11.93 (D)
distance between front cornea and retina		z_r	22.785 (mm)
refractive index	air	n_1	1.000
	vitreous humor	n_2	1.336

The unit D is diopter (m^{-1}).

Table 3: *Simulation targets and their control parameters.*

target symptom		parameter control
myopia	refractive	$\bar{P}_{min} > 58.64$ (D)
	axial	$z_r > 22.785$ (mm)
hyperopia	refractive	$\bar{P}_{min} < 58.64$ (D)
	axial	$z_r < 22.785$ (mm)
presbyopia		$\Delta_{max} < 11.93$ (D)
astigmatism	regular	$A > 0,\ \omega_1^{(e)} = 0$
	oblique	$A > 0,\ \omega_1^{(e)} \neq 0$
daylight or night vision		$0.75 < r_p < 4$ (mm)

4.5. GPU Implementation

In the rendering main loop, the scene is iteratively drawn as many times as the user-defined number of samplings for defocus and the results are blended to produce a blurred image. The trick to apply the blur field to each vertex is a simple process.

For each sampling, a *displacement seed vector* is generated and fed into the vertex shader. A displacement seed vector is a 2D vector from the center of a unit circle to a randomly generated point within the circle. In the vertex shader, the seed is transformed into a 2D displacement vector using the blur field value $\mathbf{B}_{v_x,v_y,v_z}$ (Equation 9) which has been looked up with the NDC coordinates of the vertex. The NDC can be obtained by dividing the x, y, and z by w coordinate. The shader adds the 2D displacement vector to the NDC x and y values and gets them back to the homogeneous coordinates just before exiting.

The blur field $\mathbf{B}_{v_x,v_y,v_z}$ is supplied to the vertex shader as either a 3D texture or a 2D texture tiled with sub-textures. A typical size of a blur field is 32×32 voxels for x and y directions and 128 for z (depth) direction. Each field value is a 2×2 matrix and can be represented as a 4-component texel.

A vertex shader code for the displacement using OpenGL Shading Language [RK06] is shown below.

```
uniform sampler3D blurField;
uniform vec2 seed;
void main(void) {
  // emulate fixed function lighting
  ......
  ......
  gl_Position = ftransform();
  // displace the vertex in NDC
  const float w = gl_Position.w;
  vec3 ndc = vec3(gl_Position.x/w,
                  gl_Position.y/w, gl_Position.z/w);
  vec3 texCoord = vec3(ndc.x/2.0 + 0.5,
                       ndc.y/2.0 + 0.5, ndc.z/2.0 + 0.5);
  vec4 mat = texture3D(blurField, texCoord);
  vec3 disp = vec3(mat.x*seed.x + mat.y*seed.y,
                   mat.z*seed.x + mat.w*seed.y, 0.0);
  ndc += disp;
  gl_Position = vec4(ndc.x*w, ndc.y*w, ndc.z*w, w);
}
```

5. Results and Performance

The performance is affected by the overall number of polygons in the scene. Especially, the mesh complexity of the refractive lens has an impact because the rays are traced on a per-vertex basis. We used lens models of about 1,000-vertex meshed polygons, which yielded images practically as accurate as ray tracing. All performance numbers were measured using 3.2GHz Pentium4 CPU with NVIDIA GPU GeForce 7800GTX graphics, with the 1024×768 pixel output image size. Please see the attached movies for the real-time captured interactive operations.

Rendering double-refraction through a lens object achieves 30-40fps for scenes of approximately 10,000 polygons. With color aberration, the throughput declines to 15fps (Figure 4).

The depth of field processing consists of multiple rendering and the performance is affected by the number of samplings. We used ten-sampling blurred image, which is sufficient for lens design verification in our experience. Without lens refraction, the frame rate for a blurred image in Figure 1

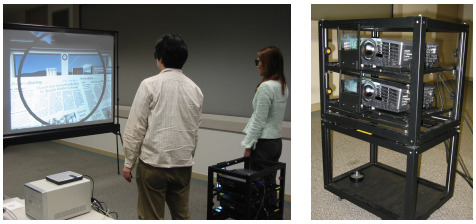


Figure 10: *A virtual eyeglass simulator using polarized stereo eyeglasses and projectors with polarized filters.*

left is around 30fps. Rendering of refraction correction images by a spectacle lens took 7fps to over 20fps depending on the scene complexity, window size, and the number of samplings. Each of the middle and right images in Figure 1 took 20.1fps at 10-sampling with 1024×768 window. For the Figure 1 images, the pupil diameter was set to be 2cm intentionally to exaggerate the defocusing results. The blur field values are in proportion to the pupil size r_p, which is obvious from Equations 8 and 9.

Figure 10 shows snapshots of a binocular, stereoscopic visualization system using this method. The rendering takes just twice the normal application since the dual images are rendered independently with different input conditions.

Figure 11 is a result of the verification of a progressive lens under design. While the upper half of the lens is pretty good at adjusting the myopia, a major portion of the lower half except the center fails to correct the presbyopia, showing the difficulty of the design of progressive lenses.

An astigmatism example is presented in Figure 12 with a partial blur field voxel information visualized as a set of ellipsoids. An observation of a whole blur field is presented in Figure 13.

The precomputation to generate a blur field using the wavefront tracing takes 300 milliseconds for a $8 \times 8 \times 128$-voxel blur field, 1 second for $16 \times 16 \times 128$, and 7 seconds for a $32 \times 32 \times 256$ blur field.

6. Conclusion

We have proposed a fast depth of field rendering method taking into account the properties of the human eyesight. Our method has the following characteristics.

- Near real-time frame rate with refraction and defocusing
- A precise human-eye model capable of simulating various eyesight syndrome and their correction by spectacle lenses
- Precomputed spatially distributed defocus information stored as a blur field

The blur field is a key and novel idea for the fast rendering. It is easy to implement on programmable GPUs and the data set is simple and compact. It could be utilized by other applications which require flexible, space dependent displacement of objects in the scene. Blur fields can be combined even with distribution ray tracing if the programmers add a vertex displacement routine.

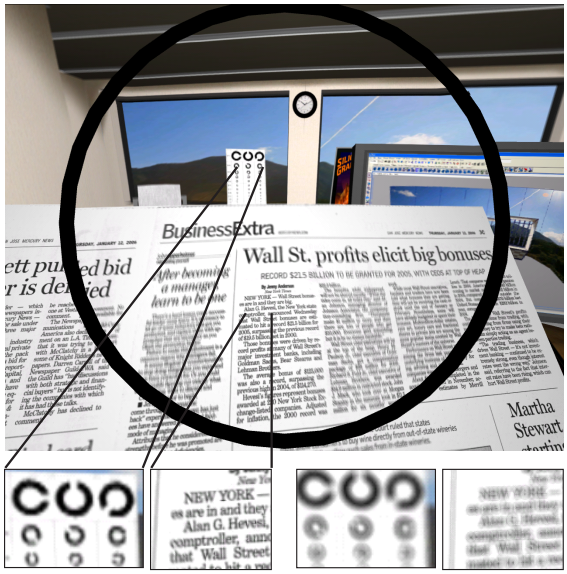
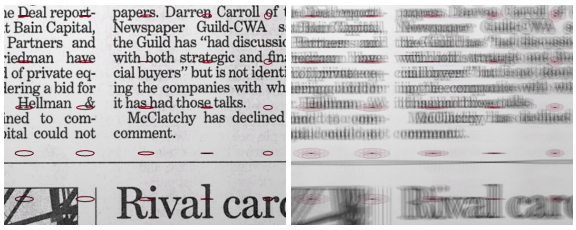


Figure 11: *Correction by a prototype progressive lens.*



(a) Without DOF (b) DOF with a blur field

Figure 12: *An astigmatic naked eye simulation result. The newspaper object is placed so close (6.1cm at the lower-left corners and 10.7cm at the upper-right) that the eye is unable to accommodate. A partial blur field (voxels of almost the same depth as the newspaper) is visualized as an ellipsoid for each voxel. Note that every blur field ellipsoid is also blurred exactly by its own spread range. Also note that the ellipse shapes differ by their distances from the viewpoint, exhibiting the sections of the Sturm's conoid.*

A limitation of the blur field technique is that the objects need be tessellated into meshed polygons with some sufficient granularity comparable to the blur field voxel size. A solution to this might be to use the blur field in the fragment shader, which is a future work.

Although our method is currently applied in a spectacle lens design company, applications to the image content generation industry are also possible, for example, as a special effect for realistic human vision.

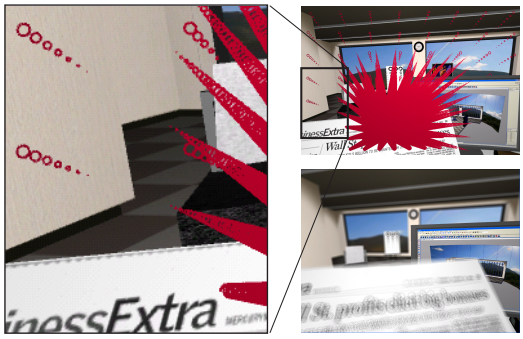


Figure 13: *An observation of an* $8 \times 8 \times 128$ *voxel blur field (top-right) and its applied result (bottom-right). Each of the small red circles represents a blur field value of a voxel. The circles are so closely formed near the viewpoint that they appear to be a block. The size of a circle is equal to the size of the blurring at the place, as shown in Figure 12. Strong myopia and presbyopia are simulated, and the focus is only on the object where the blur field values are zeros.*

Acknowledgments

We thank Naoya Hirai, Masaki Sawai, and Tomofumi Teratani of SGI Japan for making the interior scenes and the demo movies, and Takeshi Haga for the advice on the shader program implementation. Thanks also go to Gordon Jolley for proofreading the drafts.

References

[Bar04] BARSKY B. A.: Vision-realistic rendering: simulation of the scanned foveal image from wavefront data of human subjects. In *Proc. APGV '04* (2004), pp. 73–81.

[BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Communications of the ACM 19*, 10 (1976), 542–547.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed Ray Tracing. In *Proc. SIGGRAPH '84* (1984), pp. 137–146.

[Elb94] ELBER G.: Low cost illumination computation using an approximation of light wavefronts. In *Proc. SIGGRAPH '94* (1994), pp. 335–342.

[GvH09] GULLSTRAND A., VON HELMHOLTZ H.: *Handbuch der Physiologischen Optik*. 1909.

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: hardware support for high-quality rendering. In *Proc. SIGGRAPH '90* (1990), pp. 309–318.

[HS93] HAEBERLI P., SEGAL M.: Texture mapping as A fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering* (1993), pp. 259–266.

[HSL01] HAKURA Z. S., SNYDER J. M., LENGYEL J. E.: Parameterized environment maps. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), pp. 203–208.

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A Realistic Camera Model for Computer Graphics. In *Proc. SIGGRAPH '95* (1995), pp. 317–324.

[Kne64] KNEISLY J. A.: Local curvature of wavefronts in an optical system. *Journal of the Optical Society of America 54*, 2 (1964), 229–235.

[LKM01] LINDHOLM E., KILGARD M. J., MORETON H.: A user programmable vertex engine. In *Proc. SIGGRAPH 2001* (2001), pp. 149–158.

[LSS98] LOOS J., SLUSALLEK P., SEIDEL H.-P.: Using wavefront tracing for the visualization and optimization of progressive lenses. *Computer Graphics Forum (Proc. Eurographics 1998) 17*, 3 (1998), 255–263.

[MH92] MITCHELL D., HANRAHAN P.: Illumination from curved reflectors. In *Proc. SIGGRAPH 1992* (1992), pp. 283–291.

[MKL97] MOSTAFAWY S., KERMANI O., LUBATSCHOWSKI H.: Virtual Eye: Retinal Image Visualization of the Human Eye. *IEEE CG&A 17*, 1 (January-February 1997), 8–12.

[Ohb03] OHBUCHI E.: A real-time refraction renderer for volume objects using a polygon-rendering scheme. In *Proc. Computer Graphics International 2003 (CGI'03)* (2003), pp. 190–195.

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *Proc. SIGGRAPH '81* (1981), pp. 297–305.

[RK06] ROST R., KESSENICH J. M.: *OpenGL Shading Language, 2nd Edition*. Addison-Wesley, 2006.

[SAB87] SANTAMARIA J., ARTAL P., BESCOS J.: Determination of the point-spread function of human eyes using a hybrid optical-digital method. *Optical Society of America A 4*, 6 (1987), 1109–1114.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proc. SIGGRAPH '02* (2002), pp. 557–562.

[Shi94] SHINYA M.: Post-filtering for depth of field simulation with ray distribution buffer. In *Graphics Interface '94* (1994), pp. 59–66.

[SKALP05] SZIRMAY-KALOS L., ASZODI B., LAZANYI I., PREMECZ M.: Approximate Ray-Tracing on the GPU with Distance Impostors. *Computer Graphics Forum (Proc. Eurographics 2005) 24*, 3 (2005), 685–704.

[Sta72] STAVROUDIS O. N.: *The Optics of Rays, Wavefronts, and Caustics*. Academic Press, 1972.

[VF94] VOORHIES D., FORAN J.: Reflection vector shading hardware. In *Proc. SIGGRAPH '94* (1994), pp. 163–166.

[Wym05] WYMAN C.: An approximate image-space approach for interactive refraction. In *Proc. SIGGRAPH 2005* (2005), pp. 1050–1053.