An Evaluation of Two Low-Cost Approximate Models of Refractive Objects

Gustavo A. Patow 1, José Luis Castiglioni 2 and Claudio A. Delrieux 3

Keywords: COMPUTER GRAPHICS — THREE-DIMENSIONAL GRAPHICS AND REALISM — REFRACTION MODELS — SCAN-LINE RENDERING — RAY TRACING

Abstract

Efficient rendering of refractive objects is of importance in Computer Graphics. However, state-of-the-art solutions are based on computationally prohibitive rendering schemata like Ray Tracing or multipass scan-line techniques. This paper is aimed to model approximations of refractive objects that are geometrically accurate and of low computational cost. In particular, we present a comparison of two geometric approximations that are adequate to compute refractions in scan-line rendering or accelerated ray tracing algorithms. The first model consists on approximating a translucent object by a spherical lens. The geometrical parameters of the lens are easily derived from any given description of the refractive object. This model can be further simplified using paraxial, thin lens, and Blinn-Newell approximations, further accelerated by environment map preprocessing, resulting in a very low cost and overall accurate geometric model that can be easily rendered within any scan-line algorithm in times similar to standard reflection computation. The second approximation model is based on Grid Tracing. The refractive object is approximated by a lattice of rectangular prisms. Refracted rays are computed with Snell’s law on a bilinear interpolated normal vector, and within the refractive object are computed through the grid using a DDA-like traversal. Then, implementations of the presented approximation models are evaluated and compared with respect to ray tracing and other rendering methods. The lens model is shown to produce adequate results and to be quite fast, but is of very limited representation flexibility. The grid model, on the other hand, can represent objects of an arbitrary geometry, but at the expense of being slower.

1 Refraction Models

The optical problem of refraction can be presented in the following terms: consider a light ray \( \mathbf{i} \) passing through the interface of two media \( m_1 \) and \( m_2 \) at point \( p_1 \). Let \( \mathbf{n}_p \) be the (inner) normal of the surface at \( p_1 \). \( \mathbf{r}_1 \) will be the ray in the refracted direction, obeying the following law [7]:

\[
\mathbf{r}_1 = \mathbf{i} + \left( \sqrt{\nu_2^2 - (\nu_1)^2} \right) \frac{(\mathbf{n}_p \cdot \mathbf{i})}{(\mathbf{n}_p \cdot \mathbf{n}_p)} - \mathbf{n}_p \cdot \mathbf{i} \mathbf{n}_p,
\]

\[ (1) \]

1Contact author. LIFIA – Laboratorio de Investigación y Formación en Informática Avanzada. Universidad Nacional de La Plata – c.c.11 – La Plata – ARGENTINA. dagush@sol.info.unlp.edu.ar. Supported by a grant of Fundación Antorchas

2Departamento de Matemáticas. Universidad Nacional de La Plata – (1900) La Plata – ARGENTINA

3Departamento de Ingeniería Eléctrica. ICIC – Instituto de Ciencias e Ingeniería de Computación. Universidad Nacional del Sur – Alem 1253 – (8000) Bahía Blanca – ARGENTINA.
where $n_1$ and $n_2$ are the refraction indices of the two media $m_1$ and $m_2$. Equation 1 can be used at $p_1$ to compute the refracted ray $r_1$ that comes from the incident ray $i$ from medium $m_1$ (usually air) through the translucent object of medium $m_2$. If we can find the point $p_2$ where the light ray is again refracted and leaves the object, then we can apply again equation 1 to compute the emerging light ray $r_2$ as the refraction of $r_1$ (see figure 1).

Most widespread refraction models are generally based on standard ray tracing implementations [4]. For example, beam tracing [6] is based on the tangent law approximation. This model is only adequate for ray-trace polygonal models and has several disadvantages. Partially intersecting beams must be chopped into separate beams of complex cross sections, leading to potentially explosive computing times. Another approach, pencil tracing [13], considers a paraxial approximation for efficient ray tracing. The proposed paraxial approximation is formulated from a $4 \times 4$ equation system, which provides a basis for pencil tracing techniques. A $4 \times 4$ equation system is determined for every principal (axial) ray, and every paraxial ray must be multiplied by this matrix. This results in an overall speed up factor. However, the computational cost, the limitations of the acceleration techniques, and some unavoidable inadequacies in the illumination model, have made many claim that ray tracing is not a general approach to photorealistic rendering [17].

The other possible alternative is to consider scan line algorithms [3] to provide an adequate solution in terms of cost, efficiency and accuracy. The simplest rendering scheme is to ignore refraction at all, so that the light rays are not refracted, but only a color bleeding or transparency occurs. This simplification is further enhanced by Kay and Greenberg illumination model [8], where the transmission coefficient is $k = k_{\min} + [k_{\max} - k_{\min}] [1 - (1 - z_n)^m]$, where $k_{\min}$ and $k_{\max}$ are the minimal and maximal transparencies of the object, $z_n$ is the $z$ component of the surface normal, and $m$ is an arbitrary coefficient, normally between 2 and 3, where a greater $m$ represents a thinner object.

More accurate scan-line rendering schemata for refractive objects are based on $z$-buffer techniques. The main geometrical problem is that in scan-line algorithms it is quite easy in general to find $p_1$, but usually finding $p_2$ can be hard, if possible at all. For this reason, this geometrical problem is avoided or not addressed at all in $z$-buffered scan-line rendering schemata. Mammen, for example [9], considers a back-to-front ordering schema with a multipass rendering and auxiliary data structures. First, all opaque objects are conventionally scan converted and $z$-buffered,
Then, translucent objects are back-to-front processed with a transparency z-buffer, storing information about every farthest object whose pixels are visible. Information of these pixels is combined with the opaque z-buffer and frame buffer. This process is repeated back-to-front with all translucent objects. It is easy to see that this schema may produce gross geometric errors in some cases. Another z-buffered refraction model was presented by Diefenbach and Badler [2]. This technique computes the refracted image of a planar surface by means of a “virtual camera” positioned pointing along the refracted ray (in a similar way as reflections are computed). This image is then texture-mapped to the planar surface. It is, however, limited to compute refraction of planar surfaces.

In this paper we present a comparison of two geometric approximations that are adequate to compute refractions in scan-line rendering or accelerated ray tracing algorithms. The first model (already presented in [12]) consists on approximating a translucent object by a spherical lens. In the next section we give a brief overview of the model, the derivation of the geometrical parameters of the lens from any given description of the refractive object, and some useful strategies using paraxial, thin lens, and Blinn-Newell approximations. We then present in section 3 a new approximation model, based on Musgrave’s Grid Tracing [11]. The refractive object is approximated by two grids, resulting in a lattice of rectangular prisms. Refracted rays are computed with Snell’s law on a bilinear interpolated normal vector, and within the refractive object are computed through the grid using a DDA-like traversal. Then, in section 4, implementations of the presented approximation models are evaluated and compared with respect to ray tracing and other rendering methods.

2 Model I: Lens Approximation

Consider a situation where the shading value of a pixel p that intersects the projection of two polygons $P_1$ and $P_2$ of a translucent object. The shape of the “generic object” lying between $P_1$ and $P_2$ can be represented—in a rough approximation—as a pair of parallel plane faces. This is exactly what is done in scanline refraction models considered in the previous section. A better approximation can be obtained considering the object as a lens. Now the object lying between $P_1$ and $P_2$ can be described as an object limited by the two sphere segments: the “input” surface is $P_1$ (with curvature $\rho_1 = \frac{1}{r_1}$), the “output” surface is $P_2$ (with curvature $\rho_2 = \frac{1}{r_2}$) (see figure 2). This optical system possesses axial symmetry with respect to an axis that intersects the centers of the spheres, called the optical axis. The intersection of the spheres with the optical axis determines the vertices $v_i$ of the lens. The sign of the curvatures $\rho_i$ will be taken as positive if the surface is convex with respect to the incidence direction of the optical ray and negative if it is concave. The thickness of the lens is the distance $d$ between the vertices. This section is a brief overview of [12] included for completeness and because here we discuss many considerations of importance for the rest of the paper.

2.1 Object shape approximation

The surface $S$ of any object can be subdivided as a set of $n$ domains (i.e., $S = \bigcup_{i=1}^{n} S_i$), in a way such in every domain of the subdivision, the surface $S_i$ can be uniquely characterized as a function $z = f(x, y)$. 
DEFINITION 1 The Weingarten map of surface $S$ at point $p$ is the linear map $\nabla_p : S_p \to \mathbb{R}^3_p$ (i.e., the tangent space of $S$ at $p$) is given by $\nabla_p(v) = -\nabla_v n$. □

This expression can be recast as $\nabla_v n = (\nabla_v \bar{n})(t_0)$, where $\bar{n} : [0, 1] \to S$ is any curve parametrized in $S$ such that $\bar{n}(t_0) = v$. It is easy to see that $\nabla_p(v)$ is a measure of the variation rate of $n$ (which, by definition, is of constant length) in passing through $p$ following an arbitrary curve $\bar{n}$. Consider for instance the case when $S$ is a (3D) sphere $x^2 + y^2 + z^2 = r^2$, with orientation $n_p = \frac{r \bar{n}}{|\bar{n}|}$. Then $\nabla_p$ is simply the multiplication by $\frac{1}{r}$. Moreover, this curvature assumes the same value following every direction $v$ and at every point $p \in S$. This suggests to consider a spherical approximation as a higher order approximation with respect to parallel plane approximation, and lays the necessary foundation for future work. (Further details can be seen in [14].) In this example there is no need to go into differential topology, but it may be needed if, for instance, third-order (bicubic) approximations are considered. Now we will concentrate on second-order surfaces, in particular, on how to extract the data needed to characterize the lens from the surface data. The first step is to find the principal plane $P$ of the lens, in a way such that it contains the two axes of greatest geometrical dispersion (see figure 2). A simple way to do this is to consider the coefficients of the plane equation $ax + by + cz + d = 0$ with the additional constraint $a^2 + b^2 + c^2 = 1$, and adjust $a, b, c, d$ with a least square regression. The following step is to transform the object to a space in which the $xy$ plane coincides with $P$, and the origin coincides with the center of the object. The final step is to find the mean curvature of the points above and below $P$ [10]. This also can be done with a least square regression, with respect to the functional form

$$F(u, z) = (z - z_0) - \rho(u + \frac{(z - z_0)^2}{2}) = 0,$$

where $2u = x^2 + y^2$ is twice the squared distance to the origin in cylindrical coordinates, $\rho$ is the curvature of the lens, and $z_0$ is a vertex of the lens. The distance $d$ between the vertices is the thickness of the lens. $F$ is biparametric, and so its regression is straightforward to determine the parameters $\rho$ and $z_0$.

2.2 Refracting Rays Through a Lens

Now consider again an incident light ray $i$ passing through the interface of two media $m_1$ and $m_2$ at point $p_1$ as in figure 1. The first refracted ray $r_1$ can be computed with equation 1. It can be shown [7] that the two dimensional case is equivalent, since the surface is of revolution and possesses axial symmetry. For simplicity, we assume that the input surface has its vertex at the origin, so that axis $z$ can be disregarded, and work carried out with the entities projected on the $xy$ plane. We will also follow the convention that designates the projected entities with the corresponding uppercase letter, (i.e., if $s = (\eta, \xi, \zeta)$, then $S = (\eta, \xi, 0)$). Every ray $i$ can be regarded as an origin $a$ plus a parametric direction $s$, i.e.: $i = a + l \cdot s$. To simplify things, we can choose $a$ on the original ray and in the plane that contains the surface vertex. That means that $a = (x, y, 0)$. In the two-dimensional case, the refracted ray $R_1 = (A', S')$ must be a linear combination of the incident ray $I = (A, S)$:

$$A' = \alpha A + \beta S$$

$$S' = \gamma A + \delta S,$$
Figure 3: Refraction geometry of the different vectors arising in the refraction of a ray \((\mathbf{a}, \mathbf{s})\). Here \(\mathbf{s}'_2(\mathbf{a}, \mathbf{s})\) denotes the ray \(\mathbf{s}'_2\) referred to the local coordinate system of \((\mathbf{a}, \mathbf{s})\).

where \(\alpha, \beta, \gamma, \delta\) are real scalars such that \(\alpha \delta + \beta \gamma = 1\).

So far, we can compute the first refracted ray \(\mathbf{r}_1 = (\mathbf{a}', \mathbf{s}')\) using equation 1. To find the second (emerging) ray \(\mathbf{r}_2\) we must consider that we are approximating translucent objects with spherical surfaces. These surfaces were described in the previous subsection with the equation 2, where the vertex of the lens \(i.e.,\) the intersection of the sphere with the \(z\) axis) is positioned at the origin of the coordinate system. To simplify the results, we introduce new variables \(q\) and \(\Psi\) that allow a compact expression for the final rays outgoing from the optical system.

**Procedure 1** The path of the incident ray \(\mathbf{I} = (\mathbf{A}, \mathbf{S})\) through the first surface of the lens can be computed with the following evaluations,

1. \((\zeta_1)^2 \leftarrow (v_1)^2 - S^2\)
2. \((q_1)^2 \leftarrow (\rho_1 \mathbf{A} \cdot \mathbf{S} - (\zeta_1)^2)^2 - (v_1)^2(\rho_1)^2 A^2\)
3. \(z_1 \leftarrow \frac{\rho_1 \mathbf{A}^2}{q_1 - \rho_1 \mathbf{A} \cdot \mathbf{S} + (\zeta_1)^2}\)
4. \((q_1')^2 \leftarrow ((v_2)^2 - (v_1)^2)(\zeta_1)^2 + (q_1)^2\)
5. \(\Psi_1 \leftarrow \frac{q_1' - q_1}{(\zeta_1)^2}\)

where \(v_1\) is the refraction index of the initial medium and \(v_2\) the index of the lens. In step 2, take \(q_1 > 0\), and in step 4 take \(q_1' > 0\) for refractions, and \(q_1' = -q_1\) for reflections. The refracted ray \(\mathbf{R}_1 = (\mathbf{A}_1', \mathbf{S}_1')\) that passes through the first surface can be found with

\[
\mathbf{A}_1' - \mathbf{A} = \Psi_1 (\mathbf{A} + z_1 \mathbf{S}),
\]
\[
\mathbf{S}_1' - \mathbf{S} = -\rho_1 \Psi_1 (\mathbf{A} + z_1 \mathbf{S}).
\]

This ray will hit the second surface, refracting again, and emerging as a new
The corresponding expression for this ray is obtained from the previous expression, replacing $A_2$ for $A$, $S_2$ for $S$, and replacing every subscript 1 for a subscript 2 and vice-versa. An adequate justification of this procedure is too extensive to be given here, but readers can consult [12].

Light refracted by a refractive object can be combined with the diffuse and specular illumination components (as in the standard Phong model):

$$rc = dc.D + sc.S,$$

where $rc$ is the refracted color of the pixel, $dc$ is the diffuse coefficient, $D$ is the diffuse illumination, $sc$ is the Snell refraction coefficient, and $S$ is the illumination at the refracted ray. Note that ambient light is accounted for in the diffuse illumination component.

### 2.3 Some Approximations and Strategies

We will consider in this subsection how equations 3 can be simplified if we consider some reasonable approximations, namely, paraxial (Gaussian) approximation, Blinn-Newell approximation, and thin lens approximation. In a paraxial approximation, light paths are supposed to be close to the optical axis of the system, so that second order quantities can be neglected. Using paraxial approximation, and if we suppose that the first medium is air ($i.e., \nu_1 = 1$) and replacing $\nu$ for $\nu_2$, we find that equations 3 can be recast with the following expressions (see [7, 12]):

$$A' = \left[\frac{d}{\nu}(1 - \nu)\rho_1 + 1\right] A + \left[\frac{d}{\nu}\right] S$$

$$S' = \left[(1 - \nu)(\rho_1 - \rho_2) - (\nu - 1)^2\rho_1\rho_2\frac{d}{\nu}\right] A + \left[1 + \frac{d}{\nu}(\nu - 1)\rho_2\right] S. \quad (4)$$

Note that all the quantities in the square brackets depend only on the geometrical properties of the lens, and consequently can be precomputed. Moreover, if the approximations correspond to small objects or large distances (as in Blinn and Newell[1]), then $S'$ suffices for finding the refracted ray, neglecting the value of $A'$. Further simplification can be done in the case of thin lenses. In lenses where $d$ is tiny or, more generally, where $\frac{d}{\nu}(\nu - 1)\rho_2 \simeq 0$, the expression for $S'$ in equation 4 reduces to

$$S' = [(1 - \nu)(\rho_1 - \rho_2)] A + S. \quad (5)$$

Equation 5 is quite inexpensive and can be a good first approximation of equations 4 in most cases.

Equations 4, show a simple relation between the surface normal and the reflected and refracted rays. This suggests to compute the refracted rays as $\mathbf{s}' \sim \mathbf{M}_x y z \mathbf{s}_r'$, where $\mathbf{s}_r'$ is the reflected ray at the surface, and $\mathbf{M}_x y z$ is a mirror operationon wrt the $x y$ plane (i.e., just change the sign of the $z$ component). The corresponding approximation that leads to equation 5 but for reflection from a single spherical surface gives

$$\mathbf{s}_r' = [2\rho_1] A + S. \quad (6)$$

(The proof can be found in [7].) Now, equations 5 and 6 are comparable when $\rho_2 \sim \frac{\rho_1}{\nu^2}$. This suggests the possibility of preprocessing the environment map in a way such that we can compute refractions with the same algorithmic machinery for
for different resolutions of the environment map (simulating the glossiness of the lens). A mirror operation $\mathbf{M}_\theta$ about a generic plane (specified by its unit normal $\vec{u}$), is equivalent to determining a rotation about the Euler angles of the plane [5]. This suggests a way to approximate complex objects, superimposing a series of lenses, because the associated refraction calculations can be easily computed as a composition of several $\mathbf{M}_\theta$ operations (see figure 5). Finally, it is worth to note that the preprocessing procedure is invariant with respect to viewing transformations. Then, for static scenes, animation can be effortlessly computed (see figure 6).

Figure 4: Computing refractions with procedure 2 and environment maps of $400 \times 400, 200 \times 200, 100 \times 100$ and $50 \times 50$.

Figure 5: A complex object represented as a composition of several lenses, and Figure 6: some frames of an animation.

3 Model II: Grid Approximation

Now we will consider a radically different approach to the problem, based on the following observations:

1. Translucid objects can be represented as a grid or lattice of rectangular prisms. This representation can be considered as a height field (or more accurately, two height fields: a top one and a bottom one). Moreover, standard object
models (polygonal, Splines, etc.) can be easily recast to a grid representation (as we will see below).

2. Refraction in the interface between the object and the exterior medium (usually air) can be computed by locally approximating the normal in equation 1 with a bilinear interpolation from the local data of a given prism. This applies both to the incident light ray penetrating the object (where the incident ray and the normal are computed in the rendering process) and to the emergent ray (where the normal is computed with a Phong interpolation applied to the prism data).

3. Within the object, the light ray is always straight. Then, the path it traverses can be computed by means of any incremental algorithm like a DDA traversal for lines [3], keeping track of the “altitude” (as is suggested in [11]). This allows to straightforwardly search the cell where the light ray emerges.

Procedures required in step 2 are a natural by-product of any scan-line rendering algorithm with a Phong interpolation. In the incident ray, the interpolated normal required in the shading equation is reused to compute the refraction of the incident ray. The same applies to the emergent ray, except for the fact that the interpolation is performed after the cell where the ray emerges was searched. Then, most of this section will be concerned with step 1, how to build a grid of prisms from a given object model, and with step 3, how to traverse it.

3.1 Grid construction

Basically, a grid or lattice of prisms is a matrix of \( m \times n \) positions, each with a height and a normal. Then, we can consider the grid structure to be equivalent to a pair of \textit{height fields} [11], a top one with normals pointing upwards, and a bottom one, with normals pointing downwards. Moreover, since every prism can be accessed from the top or from the bottom, we must consider the two interfaces. Translucid objects can be easily represented with this schema. But normally we will want to render objects with an arbitrary representation (polygonal, Spline interpolated, CSG or libraries). Then we must consider a means to transform an arbitrarily represented object to a grid representation.

**Procedure 2** A \textit{transformation to a grid representation can be achieved with the following steps:}

1. \textit{Transform the object space to a “grid space”, where the object is centered with respect to the grid, and the extent of the object matches the size of the grid.}

2. \textit{Then traverse every element (polygon, patch, etc.) of the object in its native representation, where the “scan lines” are now lines in the grid, and the “z-buffer” are the records for the heights and normals of the top and bottom interfaces of every cell, replacing the standard z-buffer verification.}

The above procedure can be considered as a parallel scan-line projection onto the grid, which is positioned in the \( xy \) plane. Layers are indexed with respect to the sign
than zero (the normal points upwards), then it is the top interface of the prism, and
if it is less than zero (the normal points downwards), then it is the bottom interface.
It is easy to see that any object in any representation that can be rendered with a
scan-line algorithm, can also be converted to a grid with the above procedure.

### 3.2 Refraction through a grid of cells

The original grid tracer considers a height and normal for each cell [11]. It is better
if we consider the heights as samples over a height function instead of constant
heights. Then, with the data that we already had (the height and normal of every
cell), positioned in the corners of the cells, we obtain a smoother approximation
that is, also, easier to traverse (see figure 7). The borders of the top and the
bottom layers must be joined to "close" the object to avoid the unlikely case that
a ray escapes between the two layers. Now we discuss the algorithm to compute
refractions traversing the grid of cells. The emergent ray is computed from the
incident ray by means of a procedure that receives an input incidence direction, a
normal and the refraction indices of the media, and returns the refracted vector, or
null if a total internal reflection arose.

As was stated above, the first refraction is computed with the position and nor-
mal values that are provided by the scan-line rendering. Then the grid is traversed
untill the emerging ray is computed traversing both layers simultaneously with an
incremental algorithm (like DDA). At every step, the height of the ray in the cell
must be compared with the four heights of the corners. Consider for instance that
we are considering the bottom height field. Then, if the ray is below the four corners,
then the ray has not emerged, and then iterate (go to the next cell in the DDA). If it
is above the four, then the ray emerges in this cell, and we are done. Otherwise, the
height of the ray is among the heights of the corners, and then we organize the four
corners in two triangles and perform the two triangle-ray intersection algorithms to
see if the ray has emerged or not. Had we considered the top height field, then the
gometry of the comparisons should be inverted.

**Procedure 3** The steps to render refractive objects with the grid schema are:

1. Convert every translucent object in the scene to a grid representation.
2. Render the scene with a scan-line algorithm of your favorite flavor.

3. When scan-converting a grid object, for every pixel and interpolated normal
   
   (a) Compute the incident refraction.
   
   (b) Grid-trace the refracted ray until it emerges the grid object.
   
   (c) Compute the emerging ray.
   
   (d) The color of the pixel is a combination of the refracted ray and a diffuse refraction coefficient, and of the specular and diffuse reflection.

In figure 8 we show a translucent object modelled with 864 polygons on a background of 512 polygons. We chose high refraction indices ($n = 1.6$ and $3.4$) to illustrate the behavior of the algorithm, and in particular, the total internal reflection phenomenon. Experimental results show a time complexity lower than $O(\sqrt{N})$, where $N$ is the number of grids.

4 Results and Comparison

The testbed used to establish a comparison between the different approximations corresponds to a translucent sphere on a checkerboard background. The environment maps are of $300 \times 300$. We used a standard PC 486 clone with 8M RAM and 1M SVGA. The timings correspond to the effective computation of the refracted rays, omitting generation and handling of the scene. In brackets we included the relative time with respect to the preprocessed environment map computation. Timings with different objects produced similar relative times.

<table>
<thead>
<tr>
<th>Plain Lens (procedure 1)</th>
<th>Paraxial (equation 4)</th>
<th>Preproc. Env. Map (equation 6)</th>
<th>Grid (procedure 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 sec. (2.2)</td>
<td>5 sec. (1.25)</td>
<td>4 sec.</td>
<td>297 sec. (74.25)</td>
</tr>
</tbody>
</table>

In establishing a comparison between these two approaches, we must keep in mind that the lens approximation, however fast, is of little geometrical flexibility. Thus, only very simple objects can be easily modelled. More complex objects, like the one shown in Figure 5, can be quite cumbersome to handle with this model. On the other hand, the grid model offers the greatest flexibility in geometry, but the rendering time is more than an order of magnitude higher. However, scan-line and ray-tracing rendering algorithms will benefit with this schema. Scan-line algorithms can enhance its realism capabilities in representing translucent objects of an arbitrary geometry, and ray-tracing will be strongly accelerated since rendering a translucent object of an arbitrary geometry may result in prohibitive times. In figure 9 we show the results of rendering a $1024 \times 1024$ picture with POV-Ray (persistence of vision), our preprocessed environment map method (PPEM), and the Kay and Greenberg model rendered with the 3D Studio. The timings are 21423 sec. (computing one ray per pixel), 765 sec., and 831 sec., respectively.
5 Conclusions and further work

We presented two shape approximation models adequate to compute low-cost refractions. In the first method, translucent objects are approximated as spherical lenses, and a formal method that justifies the approximation is proposed. The geometrical parameters of the lenses are easily derived from the polygonal description of the objects. A computationally inexpensive refraction model was then developed. Given the representation of objects as lenses, we presented a simplification of the geometrical refraction model that is accurate in most cases and of low computational cost. The simplification is based on paraxial approximation, which in turn can be accurately computed with any environment map implementation. We presented examples of the utilization of multiple lenses as an approximation of complex objects. In the second method, translucent objects were approximated as grid of cells, where the geometrical parameters of the cells were easily derived from the description of the objects. Then, we developed a fast and precise algorithm that traverses the grid, following the refracted ray path. The overall performance of the algorithm is slower.
than the lens model, computing high resolution pictures in several minutes, but the model is more flexible in geometry.

Several improvements can be done upon this work. Third-order (bicubic) approximation seems to be a good target, since most complex computer graphic objects are modeled with B-Spline bicubic patch interpolation. We are also currently investigating a hybrid model, i.e., representing the objects as grids of third order (bicubic) patches. This model is quite advantageous in two senses: it combines the best of the lens model (speed) and of the grid model (geometry).

Acknowledgment: The authors are grateful to an anonymous referee for her/his helpful comments on an earlier draft of this paper, and for improving the quality of its presentation.

References