

Reflected and Transmitted Irradiance from Area Sources using Vertex Tracing

Michael M. Stark and Richard F. Riesenfeld

University of Utah
Salt Lake City, UT
mstark@cs.utah.edu

Abstract. Computing irradiance analytically from polygonal luminaires in polygonal environments has proven effective for direct lighting applications in diffuse radiosity environments. Methods for analytic integration have traditionally used edge-based solutions to the irradiance integral; our previous work presented a vertex-based analytic solution, allowing irradiance to be computed incrementally by ray tracing the apparent vertices of the luminaire. In this work we extend the vertex tracing technique to the analytic computation of irradiance from a polygonal luminaire in other indirect lighting applications: transmission through non-refractive transparent polygons, and reflection off perfectly specular polygons. Furthermore we propose an approximate method for computing transmitted irradiance through refractive polyhedra. The method remains effective in the presence of blockers.

1 Introduction

Indirect lighting, particularly specular and refractive transfer, is an important part of realistic rendering, yet continues to be a difficult problem. The literature is replete with methods for computing caustics and specular effects. For example, Arvo [1] used ray tracing from the light source to generate caustics. Monte Carlo path tracing and other stochastic techniques have also been used, but some common drawbacks are cost and sometimes noise [23]. The theory of catastrophic optics [19] has been applied to caustics from curved surfaces [33], but research has concentrated mostly on point or directional light sources. Recent research has often focused on interactive techniques [10]. More recently, Jensen’s photon map technique has been used for a wide variety of global illumination problems, including specular reflection and caustics from area sources [15–17].

Our previous work in direct lighting and radiosity reconstruction in diffuse environments worked by tracing rays through the source vertices and incrementally computing irradiance [30]. The goal of this paper is to extend this work to include indirect illumination from perfectly reflective facets and through refractive objects. For the reflective effects the resulting illumination remains exact, but we employ a heuristic for refraction and use it to produce plausible caustics. Section 2 contains a brief review of the vertex tracing approach on which this work is based and extends the work to handle non-refractive transparent polygons. Section 3 shows how the approach can be extended to handle reflected irradiance, while Section 4 develops the refraction heuristic. Acceleration and efficiency issues are discussed in Section 5. Results are examined in Section 6 and future work is discussed in Section 7.

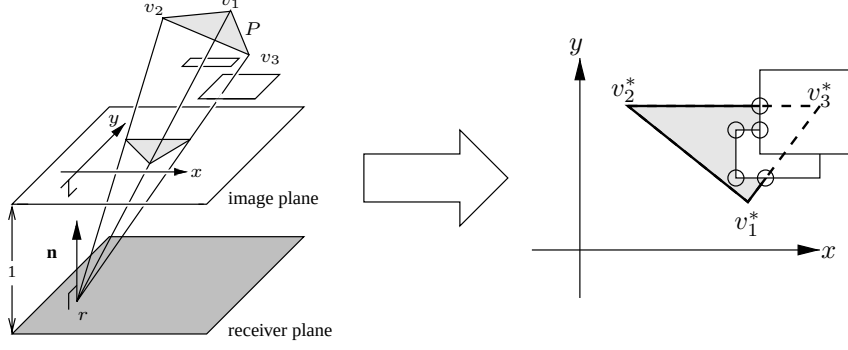


Fig. 1. To apply the vertex-based formula, the source polygon P is projected onto an image plane parallel to the receiver plane. Occluding polygons are also projected, and the apparent source is the projected source clipped against the projected occluders. The vertex tracing algorithm allows for the vertices of the apparent source to be added incrementally without actually projecting any of the polygons, or clipping the source.

2 Computing Irradiance by Tracing Vertices

We recall that the irradiance from a uniformly emitting surface $\mathcal{S}$, which is not self-occluding as viewed from a point r on a receiver, can be computed from the surface integral

$$I(r) = M \int_{\mathcal{S}} \frac{\cos \theta_0 \cos \theta}{d^2} d\mathcal{S}, \quad (1)$$

where d is the distance from r to a point on $\mathcal{S}$, θ_0 and θ are the angles made by the ray joining r and the point with the receiver normal at r and the surface normal at the point, respectively. The constant M is an emission constant of $\mathcal{S}$ [2].

If the surface is a planar polygon P with vertices $v_1, \dots, v_n$, the irradiance may be computed from a formula attributed to Lambert:

$$I(r) = \frac{M}{2\pi} \sum_{i=1}^n \beta_i \cos \gamma_i, \quad (2)$$

where β_i is the angle subtended by v_i, v_{i+1} from r , and γ_i is the angle between the plane containing v_i, v_{i+1} , and r , and the normal to the receiver at r (e.g., [6]).

In polygonal environments where the source polygon can be partially occluded, the irradiance can be computed by integrating over the *apparent* source, which is the source clipped against the scene blocker polygons [7]. Applying Lambert's formula to this clipped source results in the exact irradiance, but a drawback is that the summation is over the edges, and the edges of the apparent source are more difficult to find than the vertices.

In our previous work [30] we developed a vertex-based analogue of Lambert's formula, and showed how it can be used to compute irradiance incrementally by tracing the apparent vertices of the source in the scene. The vertex-based formula is derived by projecting the polygon P through the receiver point r onto a local *image plane*, which is the plane parallel to the surface at r and one unit above as shown in Figure 1. In order for the projected polygon to be a proper bounded polygon on the image plane,

the source must also be clipped against a view frustum [27]. Note this image plane is a device for computing irradiance at r , and has nothing to do with an “image plane” in a ray tracing or camera context.

The irradiance is computed by the following formula:

$$I(r) = M \sum_{v_i^*} F(x_i, y_i, m_{\text{in}}) - F(x_i, y_i, m_{\text{out}}) \quad (3)$$

where each projected vertex v_i^* has coordinates (x_i, y_i) on the image plane, and m_{in} and m_{out} are the slopes of the incoming and outgoing edges, respectively, on the image plane. Terms where the slope is undefined are omitted from the summation. The function F is

$$F(x, y, m) = \frac{Ax}{2} \arctan(Ay) + \frac{C(y - mx)}{2} \arctan[C(x + my)] \quad (4)$$

where

$$A = \frac{1}{\sqrt{1 + x^2}}, \quad C = \frac{1}{\sqrt{1 + m^2 + (y - mx)^2}}. \quad (5)$$

Equations (3), (4) and (5) provide a formula for the irradiance due to a uniformly emitting polygon in terms of the projected vertices and the local behavior of the incident edges. The sum may therefore be evaluated in any order and the formula is thus well suited for an incremental evaluation.

2.1 Vertex Tracing and Angular Spans

The incoming and outgoing edges at each vertex on the image plane can be encapsulated by an *angular span*, consisting of a position, two angles, representing the slopes of the incoming and outgoing edges, a depth, and an emission constant. From an angular span, the vertex contribution to the irradiance in (3) can be immediately computed and added to the sum. A vertex of the clipped source can be either an intrinsic vertex of the source polygon, an intrinsic blocker vertex which appears inside the source, or an *apparent* vertex caused by the intersection of two edges. The form of the angular span depends on the type of vertex.

However, actually projecting the source polygon onto the image plane and clipping defeats the purpose of the vertex-based approach. Rather, our method computes an angular span for each apparent source vertex by tracing the ray through the vertex and collecting angular spans for each polygon vertex, edge, or face the ray intersects and thereby allows the irradiance to be computed without ever projecting or clipping the source. Figure 2 illustrates the idea. The angular span algorithm handles occlusion by inserting a “full” angular span for an interior blocker intersection, and in fact cleanly handles all the “hard” cases where vertices and edges appear to coincide. Furthermore the approach generalizes easily to the situation where all the scene polygons are treated as emitters. Finding all the apparent vertices, which amounts to finding all the apparent edge intersections in the scene, tends to be the bottleneck.

The vertex-based approach provides a method for evaluating irradiance that is fundamentally different from methods based on Lambert’s formula. The vertex tracing algorithm eliminates polygon clipping, an inherently unstable process, and the need to maintain polygon contours. The method continues to work well in situations when polygon clipping methods become impractical, such as when there is a great deal of fine-scale geometry, and when there are many emitting polygons such as in the problem of radiosity reconstruction.

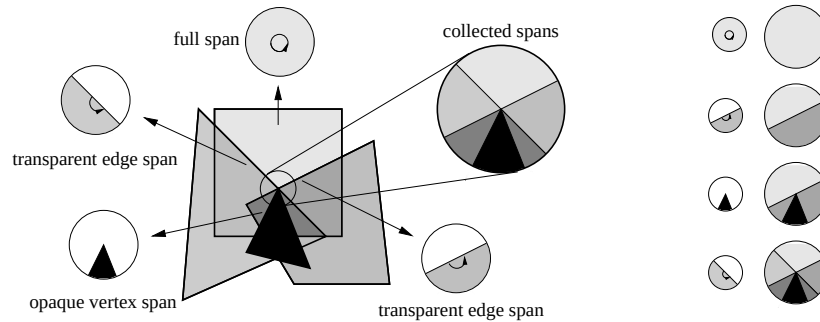


Fig. 2. A “conjunctive” vertex. Two transparent polygons and an opaque blocker appear in front of a square source. When a ray is traced through the vertex of the blocker, angular spans for the objects incident on the ray are collected (left), then sorted by distance and combined into a collection of angular spans that encodes the vertex contribution (right).

2.2 Simple Transparency

Although our previous work was concerned with uniform emission and opaque occlusion, the angular span algorithm can be extended to handle transparent (non-refracting) polygons. An angular span for a transparent polygon must be flagged to indicate it is transparent. The span combination algorithm also requires modification: when a transparent span covers an emissive span the emission is multiplied by the transparency. Furthermore, span insertion in arbitrary depth order is no longer feasible. Rather, the spans are collected as they are found along the vertex ray and then sorted by depth before they are inserted. The spans are inserted from back to front. Insertion of an opaque span involves inserting the start and end of the span, and removing all the span boundaries it covers, while insertion of a transparent span requires the emissions of the covered spans to be multiplied by the transparency of the new span. Figure 2 illustrates this for a “conjunctive” vertex.

3 Reflection

Our previous work concentrated on diffuse environments and considered only direct lighting and radiosity reconstruction. In this section we extend the vertex tracing approach to environments having perfectly reflecting polygonal surfaces.

A perfect mirror creates a *virtual image* of each object it reflects: the virtual image is the object reflected through the plane of the mirror as illustrated in Figure 3(a). Thus the radiance from a source polygon reflected off a mirror polygon is equivalent to the radiance from the virtual image clipped against the complement of the mirror polygon in the mirror plane. Equivalently, the irradiance can be computed by reflecting the receiver point in the mirror plane rather than the source—indeed this is preferable because it does not reverse the orientation of the source polygon, and it eliminates the need to reflect the source at all (Figure 3(b)). The vertex tracing algorithm may also be used provided that angular spans for edges and vertices of “negative” polygons (the complement of the mirror polygon in the mirror plane) are properly handled.

The presence of extra occluders complicates things, because they can appear in two separate places: between the source and the reflector, and between the reflector and the

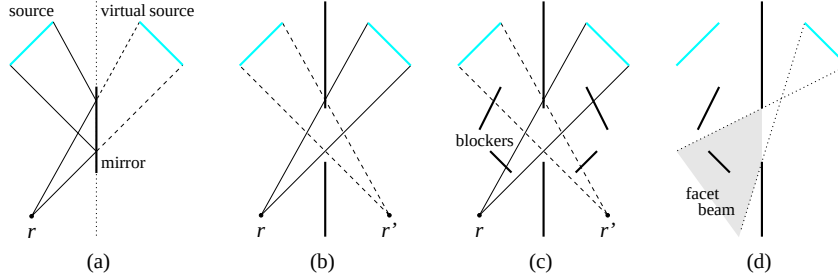


Fig. 3. Reflection in “Flatland”. (a) The radiance coming from the real source reflected off a mirror is the same as the radiance coming from the virtual source, the source reflected through the mirror plane. (b) The irradiance from the virtual source clipped against the complement of the a mirror polygon is equivalent to computing the reflected irradiance, or computing the irradiance at the reflected receiver point r' . (c) Blocker polygons between the source and the mirror are most easily found by tracing from r' ; those between the mirror and r are found by tracing as usual. (d) Beams can be used for efficiency

receiver point. A naive clipping approach would require the occluders to be reflected through the mirror plane, but the vertex tracing approach does not. Rather, each vertex (intrinsic or apparent) is simply traced twice, once from the actual receiver point, and once from the reflected receiver point (Figure 3(c)). From the receiver point, only vertices and intersections in front of the mirror plane are considered; from the reflected point, only those behind are considered. However, pairs of edges from both sets of occluders must be tested for apparent intersection.

Multiple reflection complicates things even more. The virtual scene requires multiple reflections of the scene in the naive approach, and the basic vertex tracing still works, but the multiple reflections must be traced separately, starting from the closest mirror surface, with the source and other blockers reflected through all but this surface, and so the process continues.

4 Refraction

Computing reflected irradiance as in the previous section works in principle by computing the irradiance from a virtual source that cannot be distinguished from the reflected real source. If such a virtual source for a refracted source polygon can be found, then the same technique applies. The refracted irradiance can be computed from the virtual source, occluded by the complement of the refractive facet. The question is, what is the virtual source? That is, what does the source look like when viewed from a receiver point through the refractive interface? This problem is more difficult than the reflective case for several reasons. First of all, lines generally appear curved under refraction, so the virtual source is not generally even a polygon. Consequently we can only approximate refracted irradiance with a polygonal virtual source. Second, accurately approximating the virtual source involves the problem of inverse ray tracing, that is, finding the ray direction which hits a particular point (*i.e.*, a source vertex) through a refractive medium. And finally, while the virtual source of a reflected polygon happens to be independent of the viewing position, this is not the case for a refracted polygon—a different virtual source is needed for each viewpoint. Figure 4 illustrates refracted virtual sources in Flatland.

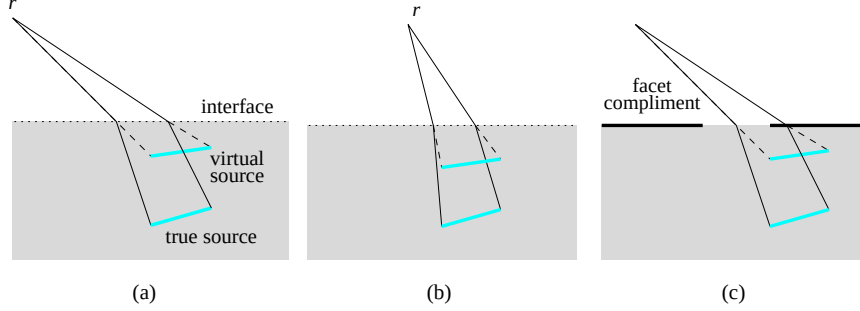


Fig. 4. Refraction in Flatland. (a) A virtual refracted source is more difficult to compute. There are many choices for the virtual source, the one illustrated is the one found by appropriately contracting the depth of the source directly toward the interface plane. (b) The virtual source changes with the viewpoint. (c) The virtual source can be clipped against a facet complement and irradiance computed as if the refractive medium were not there.

It should be made clear that the goal of this section is not to render truly focused caustics, in the sense of catastrophic optics [19]. The focusing of light by its very nature requires curved surface interfaces; indeed even the first-order behavior of refracted (and reflected) light is dependent on the surface curvature. It is therefore not reasonable to expect light transmitted through refractive surfaces to be well approximated by a macro-faceted surface approximation. We are interested in simulating how polyhedral objects transmit light, not how well this approximates a curved surface; the latter is a subject for future research. Figure 6 shows a faceted approximation to a smooth surface.

4.1 Depth Contraction

We now turn to the development of the transformation that approximates how a polygon appears when it is viewed through one, or several, refractive interfaces. This transformation can then be applied to the source, and any blocker polygons on one side of the interface so that the vertex tracing approach can be used to compute the irradiance.

Snell’s law [12] governs the angle θ_2 of refraction given the angle of incidence θ_1 :

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{n_2}{n_1} \quad (6)$$

where n_1 and n_2 are the indices of refraction (Figure 5(b)). The rule is well suited to ordinary ray tracing but by itself does not provide much intuition for how refraction transforms the general appearance of objects. Suppose, for example, one looks into the front face of a fish tank from across the room. The general effect is that the inside of the tank appears compressed toward the front of the tank. A similar phenomenon can be seen from a canoe or kayak in shallow still water. When the bottom is roughly flat, it always looks as if the bottom is shaped like a bowl and the observer is always above the deepest part (Figure 5(a)). Further away it tends to look more shallow. So in fact, the depth contraction is dependent on the viewing angle with respect to the surface of the water.

This apparent depth contraction phenomenon is the basis for the transformation we use for the virtual refracted source. The geometric arrangement is illustrated in Figure 5. A point q at depth d below the (flat) interface appears to be at a depth d'

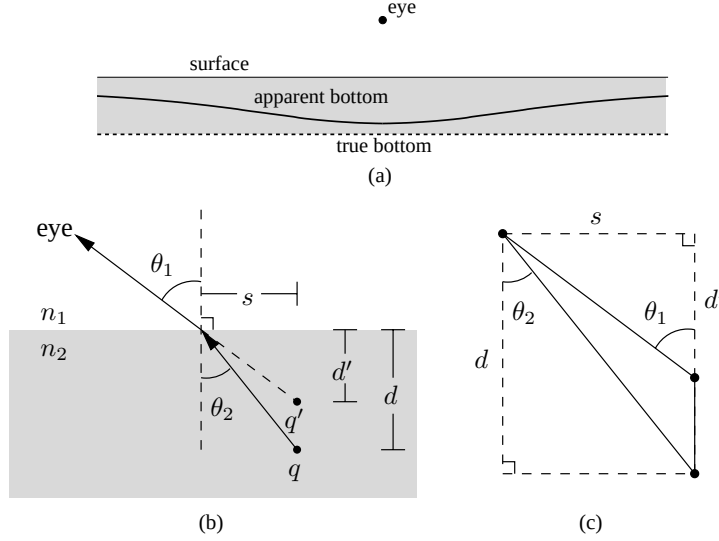


Fig. 5. Apparent depth contraction. (a) A flat bottom of a pond or pool always looks bowl-shaped, and appears deepest directly under the observer. (b) A point q at depth d appears to be at depth d' as viewed from above the interface. (c) The associated trigonometry.

below the interface. The apparent depth of the point is therefore *contracted* by the factor d'/d . From trigonometry we have

$$\tan \theta_1 = \frac{s}{d'}, \quad \tan \theta_2 = \frac{s}{d}$$

so we have for the contraction ratio,

$$\frac{d'}{d} = \frac{\tan \theta_2}{\tan \theta_1}. \quad (7)$$

From Snell's law, the contraction ratio can be given in terms of the incident angle θ_1 or the refracted angle θ_2

$$\frac{d'}{d} = \frac{\cos \theta_1}{\sqrt{\eta^2 - \sin^2 \theta_1}} = \frac{\cos \theta_2}{\sqrt{1/\eta^2 - \sin^2 \theta_2}} \quad (8)$$

where $\eta = n_2/n_1$. Of course, actually computing these angles is equivalent to finding the true ray directions and thus defeats the purpose of the heuristic. But as it happens, the formulation in terms of angles is suited for approximating the virtual source, provided we make a key assumption: the apparent angular size of each refractive facet is small as seen from a receiver point, or another refractive facet (from which it can receive transmitted light). Because of this assumption, the incident and transmitted angles of any ray hitting the facet from a receiver point, or from another facet, have little variation and consequently the depth contraction ratio is nearly constant.

4.2 The Virtual Refraction Transformation

Suppose the facet with centroid c and normal $\hat{\mathbf{n}}$ lies on an interface between two regions of different optical densities and the ratio of the indices of refraction of the back side of the facet to that of the front is η . From a view point r , the cosine of the angle of incidence anywhere on the facet is approximately that of the angle of incidence at the centroid c :

$$\cos \theta_1 = \frac{(r - c) \cdot \hat{\mathbf{n}}}{\|r - c\|}.$$

A point q behind the facet has depth $d = (c - q) \cdot \hat{\mathbf{n}}$; combining this with (8) above yields a formula for the contracted point q' .

$$q' = q + (d - d')\hat{\mathbf{n}} = q + (c - q) \cdot \hat{\mathbf{n}} \left(1 - \frac{\cos \theta_1}{\sqrt{\eta^2 - 1 + \cos^2 \theta_1}} \right) \hat{\mathbf{n}}. \quad (9)$$

If $\eta < 1$, q' will be further from the facet rather than closer, and also there is the possibility of total internal reflection, which occurs when $\cos^2 \theta_1 \leq 1 - \eta^2$.

Application of the transformation given by (9) to each vertex of the source produces a virtual source from which irradiance can be computed as if there is no refraction. Further transformations (reflections or other refractive transformations) can be applied to handle multiple reflections and refractions. The same transformation must be applied to any blocker polygons which lie between the source and the facet on the interface.

In an attenuating medium, the attenuation can be approximated from the centroid-to-centroid path length. Also, reflectance variations such as those caused by Fresnel effects can be approximated in terms of the angles between facets.

Of course, the contracted (or elongated) source is still only an approximation because the true depth contraction is a function of the position on the source and the position of the receiver, and more precise methods could be used (*e.g.*, [3]). The approximation is in the assumption that it does not vary over the source (from a fixed viewpoint). But this method does seem to produce plausible caustics both for single and multiple refractive interfaces, as shown in Figures 6 and 8.

4.3 Dispersion

Dispersion, the separation of different wavelengths of light due to variation of refractive index with wavelength [12, 17], can be simulated using our refraction method. For general dispersion computation, many different wavelengths must be included in order to achieve the smooth spectral colors often visible on the edges of caustics and such. But one advantage of using area sources as in our method is that the caustic edges are not sharp and blending occurs naturally. We have found plausible dispersion effects can be obtained by using only the three RGB channels.

5 Efficiency: Beams and Culling

A brute-force approach to vertex tracing reflective and refractive polygons can be costly, because for a particular evaluation point, every reflective and refractive polygon must be checked, and for each of these polygons, every other reflective and refractive polygon must be checked for secondary reflection and transmission, *etc.*

To reduce the cost, we propose a beam hierarchy for spatial subdivision. While this may sound similar to the beam and cone tracing and hidden surface techniques in the

Algorithm 1 General Beam Construction

```
for each reflective/refractive polygon do
  add a beam for the reflective/refractive region, with depth  $d = 1$ 
end for

for each beam  $\mathcal{B}$  in the beam list do
  for each scene polygon  $P$  which intersects  $\mathcal{B}$  do
    add  $P$  to the occlusion list of  $\mathcal{B}$ 
    if  $P$  is reflective/refractive and the depth  $d$  of  $\mathcal{B}$  is less than the maximum then
      append a beam for  $P$  with depth  $d + 1$ 
    end if
  end for
end for
build a hierarchy for the beams
```

literature (e.g., [9, 13]), the approach is closer to the shaft culling method of Haines and Wallace [11]. In our context a *beam* is a (possibly unbounded) convex polyhedron which bounds the region in space in which a source polygon can potentially appear reflected from or refracted through another polygon, or collection of polygons. Each beam contains the following information:

- bounding planes for the polyhedron (for containment testing);
- a formula for how to create the virtual source (reflection, refraction);
- the reflective/refractive polygon;
- a depth d , indicating the number of reflections/refractions;
- a parent beam (NULL if $d = 1$);
- a list of potential occluding polygons that intersect the beam.

A beam is *primary* if its depth is 1. When a point is found to be inside a primary beam, the source can be vertex-traced along with the polygons in the occlusion list to compute the irradiance. A point inside a non-primary beam requires the multiple-interface vertex tracing described previously. The various interfaces are found by following the parent pointers to the primary beam.

There are several things worthy of note about these culling beams. First, the beam polyhedron is only a bound and nothing more—the vertex tracing algorithm handles the actual visibility. The number of beams in a scene depends on the number of reflective/refractive polygons as well as how they interact. Theoretically there is no upper limit on the number of possible inter-reflections and constructing all the beams could be very expensive. But for simple types of reflective objects, really only primary beams are required, and there will be at most one beam for each reflective polygon. For a refractive object, there will generally be only a few beams for each interior facet.

The general beam construction algorithm is shown in Algorithm 1. In our implementation, the faces and edges are stored in a bounding sphere hierarchy and this accelerates the beam-polygon intersection test. Once the beam list has been constructed, the beams themselves are stored in a hierarchy for faster containment testing. Our implementation uses a bounding volume hierarchy, but other spatial subdivision schemes could be used.

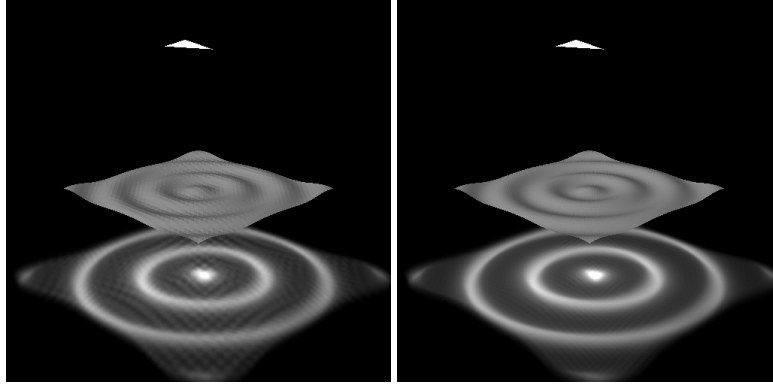


Fig. 6. Approximating caustics from a rippled water surface, using a triangulated grid of 32×32 (left, 12s) and 64×64 (right, 43s). Here the water surface is rendered as a diffuse surface for clarity; note the faceting is less noticeable under diffuse shading than in the caustics. Also note the asymmetry in the caustics due to the shape of the source.

6 Results

We have implemented our methods in the context of a standard ray tracer. When a diffuse surface is hit, the direct lighting is computed (exactly, using vertex tracing) and this is added to the reflected and transmitted light which is computed using the vertex tracing approaches discussed in this paper. The images were run on an SGI workstation using a single 400 MHz IP35 MIPS R12000 processor, and the quoted running times include setup, hierarchy construction, rendering, and output time. The images all have 512 vertical pixels.

Figure 7(a) contains the shadow of three transparent polygons, and illustrates “subtractive absorption” of light. Figure 7(b) and (c) show the shadow of a stained-glass with 171 polygons (122 leads, 49 panes) under different light sources. Note that the shape of the source has a definite effect on appearance of the shadow, and can be seen through the “pinhole” effect in various places. In Figure 7(a) the top edge of the window is coplanar with a source edge and this results in the sharper C^1 discontinuities seen in the shadow.

Figures 8(a) and (b) have purely reflective objects. Figure 8(c) is a rendering of a glass icosahedron, with the surface reflectance exaggerated to more clearly show the reflected light. Figure 8(d) simulates dispersion (with a dimmer source and whiter surfaces) as described in Section 4.3. The three-color split is sufficient for the caustics due to the blurring effect, but results in color discontinuities in ray tracing the object itself. The focus of this paper is on polygonal objects, but Figure 6 shows what happens when a smooth surface is approximated with facets and the polygonal refraction is computed, in this case approximating rippled water.

7 Conclusion and Future Work

In this paper we have extended the vertex tracing approach, a general method of computing irradiance from a uniform polygonal source in a polyhedral environment, first to include transparency, then to compute exact illumination reflected from mirrored

polygons, and finally to approximate transmitted irradiance. Moreover, a shaft culling technique was employed to improve the efficiency.

The caustics produced by our depth contraction heuristic are plausible, but we would like to improve the physical accuracy. In particular more attention should be paid to energy conservation, which is currently only loosely handled by our method. A better approximation to the virtual source would require a non-uniform emission to account for attenuation and reflectance variations such as Fresnel effects. Formulas such as those developed for polynomially-varying luminaires by Chen and Arvo [5] would be required. Finally, the ripple caustics suggest there might be a way to better approximate the surface rather than using flat facets, which would give a smoother irradiance distribution with far fewer surface elements.

Acknowledgments

The authors wish to thank Erik Reinhard, Peter Shirley, and William Martin for many helpful discussions. This work was supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219).

References

1. James Arvo. Backward ray tracing. In *SIGGRAPH '86 Developments in Ray Tracing seminar notes*, volume 12, August 1986.
2. James Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University, 1995.
3. Min Chen. *Mathematical Methods for Image Synthesis*. PhD thesis, California Institute of Technology, 2001.
4. Min Chen and J. Arvo. Perturbation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics*, 6(3):253–264, July–September 2000.
5. Min Chen and James Arvo. A Closed-Form Solution for the Irradiance Due To Linearly Varying Luminaires. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, New York, NY, 2000. Springer Wien. 137–148.
6. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge, MA, 1993.
7. George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 223–230. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
8. Frédo Durand, George Drettakis, and Claude Puech. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 89–100. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
9. H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. volume 14, pages 124–133, July 1980.
10. Xavier Granier, George Drettakis, and Bruce Walter. Fast Global Illumination Including Specular Effects. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 47–58, New York, NY, 2000. Springer Wien.
11. Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.
12. Eugene Hecht. *Optics*. Addison Wesley Longman, third edition, 1998.

13. Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 119–127, July 1984.
14. Wolfgang Heidrich, Hendrik Lensch, Michael F. Cohen, and Hans-Peter Seidel. Light Field Techniques for Reflections and Refractions. In D. Lischinski and G. W. Larson, editors, *Rendering Techniques 1999 (Proceedings of the Tenth Eurographics Workshop on Rendering)*, pages 187–196, New York, NY, 1999. Springer Wien.
15. Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.
16. Henrik Wann Jensen. Rendering caustics on non-lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997. ISSN 0167-7055.
17. Henrik Wann Jensen and Niels Jørgen Christensen. A practical guide to global illumination using photon maps (siggraph 2000 course notes 8), July 2000.
18. Don P. Mitchell and Pat Hanrahan. Illumination from curved reflectors. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 283–291, July 1992.
19. J. F. Nye. *Natural Focusing and Fine Structure of Light*. IOP Publishing Ltd, 1989.
20. H. Plantinga and C.R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.
21. Holly E. Rushmeier and Kenneth E. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Transactions on Graphics*, 9(1):1–27, January 1990.
22. Gernot Schaffler and Henrik Wann Jensen. Ray Tracing Point Sampled Geometry. In B. Perroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 319–328, New York, NY, 2000. Springer Wien.
23. Peter Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, May 1990.
24. Peter Shirley. *Realistic Ray Tracing*. A K Peters, Ltd, 2000.
25. François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.
26. Cyril Soler and François X. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 321–332. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
27. Michael M. Stark. Vertex-based formulations of irradiance from polygonal sources. Technical Report UUCS-00-012, Department of Computer Science, University of Utah, May 2000.
28. Michael M. Stark. *Analytic Illumination in Polyhedral Environments*. PhD thesis, University of Utah, 2001.
29. Michael M. Stark, Elaine Cohen, Tom Lyche, and Richard F. Riesenfeld. Computing exact shadow irradiance using splines. *Proceedings of SIGGRAPH 99*, pages 155–164, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.
30. Michael M. Stark and Richard F. Riesenfeld. Exact Illumination in Polygonal Environments using Vertex Tracing. In B. Perroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 149–160, New York, NY, 2000. Springer Wien.
31. Spencer W. Thomas. Dispersive refraction in ray tracing. *The Visual Computer*, 2(1):3–8, January 1986.
32. Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. ACM Press, 1992.
33. Mark Watt. Light-water interaction using backward beam tracing. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 377–385, August 1990.

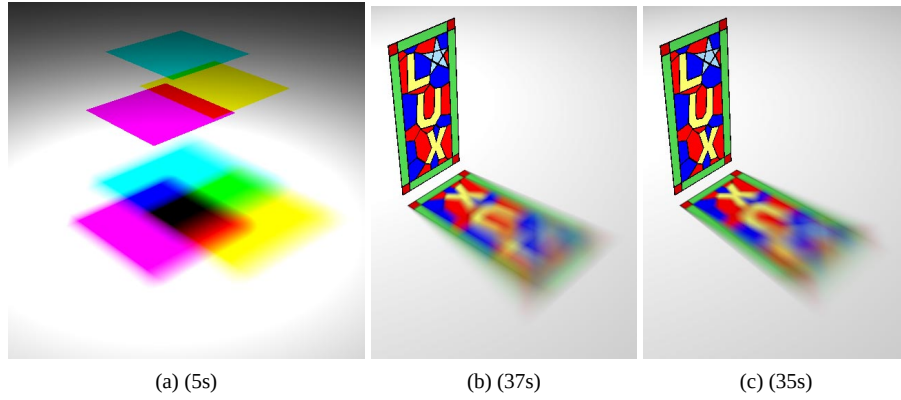


Fig. 7. Shadows of (a) purely transparent polygons, (b) a stained glass window under a large triangular source and (c) the source rotated 45 degrees; note the differences in the fine structure of the shadow.

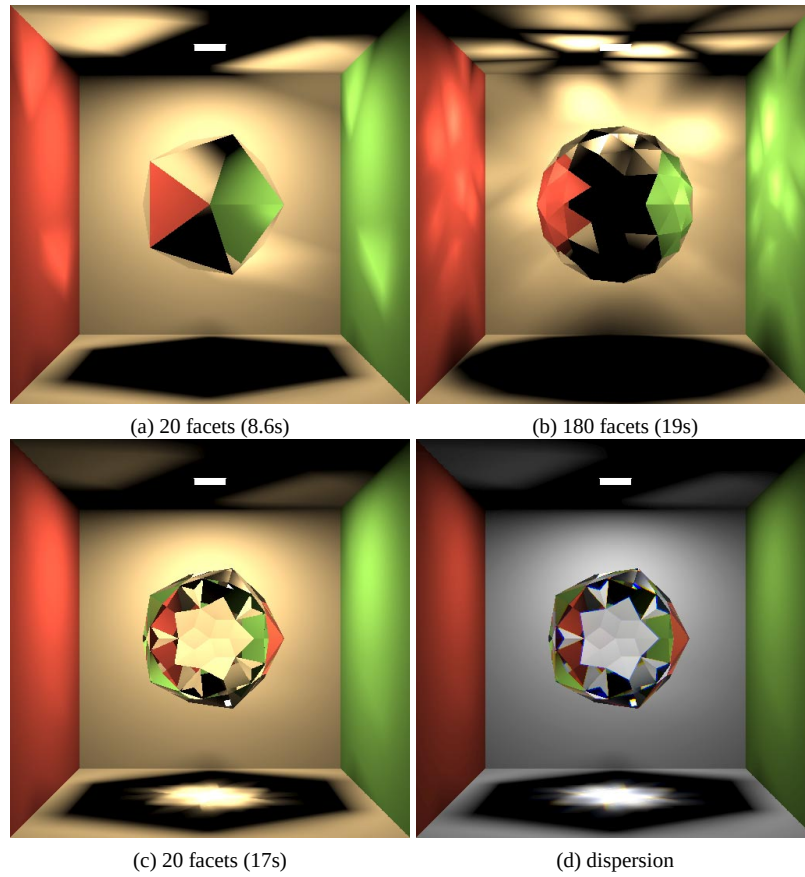


Fig. 8. (a),(b) Purely reflective objects. (c) Transmission through a glass icosahedron. (d) Dispersion.