A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes

Peter Shirley
Department of Computer Science
University of Illinois
1304 West Springfield Avenue
Urbana, Illinois 61801
USA

Abstract

Several ways of improving the realism of the results of traditional ray tracing are presented. The essential physical quantities of spectral radiant power and spectral radiance and their use in lighting calculations are discussed. Global illumination terms are derived by employing illumination ray tracing for calculation of quickly changing indirect lighting components, and radiosity ray tracing for slowly changing indirect lighting components. Direct lighting is calculated during the viewing phase allowing the use of bump maps. Finally, a method is introduced that reduces the total number of shadow rays to no more than the total number of viewing rays for a given picture.

Keywords: Bump Mapping, Illumination, Radiosity, Radiance, Ray Tracing, Realism, Stratified Sampling, Texture Mapping.

1 Introduction

The quest for accurate lighting models in computer graphics has taken two very different approaches in the 1980s. The first approach is based on ray tracing (point sampling) techniques made popular by Turner Whitted's landmark 1980 paper [Whi80]. The second approach, radiosity methods, is based on zonal techniques borrowed from the heat transfer literature [GTG84]. The results produced by both techniques suffer from discretization errors; ray tracing uses discrete sample points, and zonal methods partition the environment into finite patches.

Recent work has combined ray tracing and zonal techniques [WCG87, SP89]. This combination allows the zonal methods to be used where they perform best, and ray tracing to be used to handle mirrors, where the zonal methods break down. Both Rushmeier and Ward et al. have separated direct and indirect lighting to avoid high levels of discretization associated with pure zonal methods[Rus88, WRC88].

This paper extends the idea of the separate calculation of direct and indirect lighting to the calculation of hard and soft lighting, where hard lighting contains detail and soft lighting changes slowly. This is similar to the idea of patch and element substructuring of Radiosity[CGIB86], but also allows fine detail caused by hard indirect lighting such as the lighting that produces caustics. This is achieved with a three pass method that synthesizes conventional ray tracing, illumination ray tracing, and radiosity. The hard (detail producing) light paths are calculated using illumination ray tracing and an efficient method of view ray tracing, and the soft (slowly changing) indirect diffuse lighting is calculated with radiosity. Most of the discussion is limited to scenes containing only diffuse and specular surfaces, but extensions for imperfect specular surfaces are outlined.

2 Background

Commonly graphics programs calculate direct lighting analytically, and then add an empirically generated ambient term. If we do physically based global lighting calculations, we must be more careful with units because lighting components that are calculated separately must not be improperly scaled relative to each other. In this section some essential terms and formulae are presented that aid in maintaining correct proportionality in lighting calculations.

Since we are interested in the amount of light hitting a surface or film plane during a set time period, *radiant flux* (also called *radiant power*), the radiant energy

per unit time, Φ, is often used. The quantity most often used in graphics is the *radiance*[1], L. Radiance shares many characteristics of the perceptual measure *luminance*; it gives an indication of surface brightness, dependent upon neither the size of the object being viewed, nor the distance to the viewer[2].

Diffuse surfaces can be characterized by a simple equation:

$$L_\lambda = \frac{R_\lambda \Phi_\lambda}{\pi A} \quad (1)$$

where L_λ is the *spectral radiance*[3] of the diffuse surface, R_λ is the reflectance of the surface at wavelength λ, Φ_λ is the *spectral radiant power* incident on the surface from all directions, and A is the area of the surface.

From this definition, the direct lighting component due to a planar diffuse light source is:

$$L_\lambda = \frac{R_\lambda L^l_\lambda \cos\theta_1 \cos\theta_2 A^l}{\pi d^2} \quad (2)$$

where L^l_λ is the spectral radiance of the light source, θ_1 is the angle the direction to the light source makes with the surface normal, θ_2 is the angle the direction from the light source makes with the light source normal, A^l is the area of the light, and d is the distance to the light.

The light that shades a surface can be divided into two types: *hard*, and *soft*. Soft lighting is the highly diffused light that is usually approximated with an ambient term. Hard light has quickly changing components that 'paint' detail onto surfaces, such as the hard edge of a shadow or caustic.

Soft lighting occurs whenever light is bounced off a diffuse reflector. Any fine geometric features of the the light will be lost when diffuse reflection occurs. A mirror does not eliminate detail in the light it reflects, it merely attenuates it. A curved mirror or clear object can even increase detail in the light. This detail will not be evident to the viewer until the light hits a diffuse surface and 'paints' its pattern on the surface.

[1]The radiance of a surface is defined to be $L = d^2\Phi/(d\omega dA\cos\theta)$, where $d^2\Phi$ is the power leaving a surface of area dA, in a direction having angle θ relative to the surface normal, through solid angle $d\omega$.

[2]The radiance will only stay constant along a line of sight if there is no atmosphere between the viewpoint and the surface being viewed. Otherwise there will be filtering effects that will diminish the measured radiance of the object, and scattering effects that will add radiance causing a bleaching effect. This is why mountains in the distance often appear faded.

[3]Spectral radiance is the radiance per unit wavelength at wavelength λ.

This distinction between hard and soft lighting is taken advantage of in the next section to make the rendering algorithm faster.

3 Algorithm

Because of the fine structure of hard light, it must be calculated accurately. For light to produce detail on a surface, its path from the light source cannot have included a diffuse reflector. When a diffuse reflector is encountered detail is lost and one enters the realm of soft lighting. Conventional ray tracing will produce the most important details: shadows on diffuse surfaces, and reflections of shadows. The detail producing light paths not accounted for by conventional ray tracing are those that hit some number of specular surfaces before a diffuse surface. An example of this effect is the fine patterns seen on a surface that is painted by light that has just passed through a bottle or reflected off a wristwatch. This shortcoming can be dealt with by tracing illumination rays from the light source and storing ray hits on diffuse objects in illumination tables[Arv85].

The soft light missing from the ray tracing calculations can be estimated by radiosity. Since this radiosity solution is used for only soft lighting, a very coarse environment partitioning can be used. Usually, radiosity solutions account for all components of the lighting, so care must be taken to remove the detailed components from the radiosity calculation as described in Section 3.2.

The three phases of the algorithm are described in order of their execution. First the illumination ray step and some optimizations is presented in Section 3.1. Second, the radiosity pass is described in Section 3.2. Finally, the viewing stage, along with an optimization for shadow calculation, are presented in Section 3.3.

3.1 Illumination Ray Tracing

The first pass of the algorithm calculates lighting that hits diffuse surfaces after hitting at least one specular surface. This type of lighting is calculated using Arvo's method of simulating light paths by sending illumination rays from the light source[Arv85].

In Arvo's method, each light source emits some large number of rays. Each ray represents a fraction of the outgoing energy that will be transported to other surfaces. Arvo accomplishes this by defining a standard texture map on each diffuse surface. When an energy bundle hits a surface, the energy is divided between the four texture nodes that surround it. To convert the energy maps to radiance maps the area represented

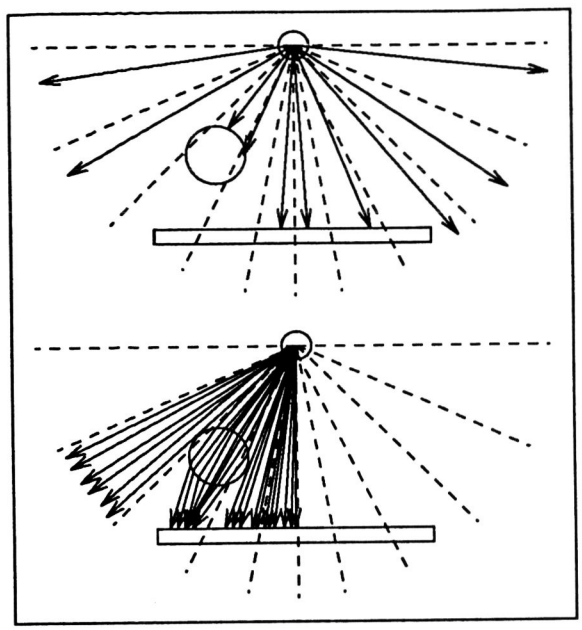

Figure 1: *Two dimensional illumination ray tracing with a specular sphere and a diffuse plane. Top: First pass feeler rays hit the specular sphere in two of the zones. Bottom: Dense energy carrying rays are sent in the two zones where the sphere was hit, and in the two zones adjacent to the hit. Energy rays that hit the ground plane directly are ignored, while those hitting the sphere first are tabulated in the ground plane's illumination map.*

by each texture node must be estimated. When this phase is completed, the lighting will be stored as radiance maps on the diffuse reflectors. Direct lighting is omitted from the maps (by not recording direct hits by energy rays) and calculated in the conventional method during the viewing phase. This calculation is view-independent.

Because direct lighting is calculated in the viewing phase, illumination rays need only be sent toward specular surfaces (which produce indirect lighting not handled by radiosity). To find the specular surfaces where illumination rays should be sent, a set of 'feeler' rays is sent in all directions from the source, and then a dense set of illumination rays is sent in directions where the feeler rays encounter specular surfaces. Since the edges of specular objects should not be ignored by illumination rays, illumination rays are generated in zones where feeler rays hit specular objects, as well in adjacent zones(Figure 1).

Stratified sampling[Coo86] is used to choose both feeler and illumination ray directions. The light sources are assumed to be diffuse and the hemisphere of directions is partitioned into zones in a diffuse distribution. This partitioning is achieved by mapping a partitioned unit square to the hemisphere[WRC88]. The top of Figure 2 shows an initial partition into 36 zones, and a further subdivision of three of these zones. This illustrates the initial partitioning for feeler rays, and a finer partitioning for the actual illumination rays. The bottom of Figure 2 shows how this idea can be used to choose points on a disk [4].

The illumination rays can generate fine patterns such as those inside the metal ring shown in Figure 3.

If there are imperfect mirrors (such as brushed steel) in the scene, illumination rays should be reflected probabilistically. This is similar to the reflection of viewing rays in distributed ray tracing[CPC84], and produces noise in the illumination maps which can only be reduced by sending more illumination rays. Figure 4 shows four plates of variable smoothness and the reflection they cast on a diffuse surface. The same number of rays are shot at each plate. The level of noise on the diffuse surfaces increases with the spread of the reflection functions of the plates. To reduce the noise, a greater number of illumination rays could be sent, or the resolution of the texture map on the diffuse surface could be decreased.

For an area light source, rays should be sent in from all points on the light source. If this is not done, the lighting map will not have soft shadows or 'soft' caustics. To reduce noise in the illumination map, rays can be sent exclusively from the center of the light source. This is similar to the way radiosity hemicubes are fixed to a certain point. If direct lighting is calculated in the viewing phase we can still produce soft shadows, but these soft shadows will be mixed with sharp caustics. For soft caustics, either the light must be subdivided or the illumination rays must originate at random points on the source. Either method will require more rays to avoid aliasing or noise.

3.2 Radiosity for Ray Tracing

The second step of the algorithm uses radiosity to calculate the soft lighting. This lighting involves at least two diffuse reflectors in all light paths. The modified radiosity algorithm used in this paper differs from the standard radiosity method because it omits direct lighting.

Radiosity calculations can be done by ray tracing in a very simple manner. Rays can be sent between all pairs

[4] The partitioning of the disk is required for lens effects of distributed ray tracing. The partitioning of the disk on the right is preferred because the zones have better locality. This is the central idea of designing partioning schemes.

of surfaces[BFP83, NN85, WEH89], or can be sent in a cosine distribution from each surface toward all unobscured surfaces[MBG86, Mal88, SP89]. The second technique has the advantage of allowing indirect transport of specular surfaces, so it is the method of choice for scenes containing mirrors. Recently, progressive refinement techniques have been used to implicitly solve the linear equations of radiance[CCWG88, SP89]. This approach avoids the $O(n^2)$ storage requirement of explicit techniques.

The progressive refinement method can be thought of as straightforward transport simulation. Though we want to find the radiances of all surfaces in an environment, it is easier to formulate the problem in terms of power, and convert to radiance once the solution is obtained. Suppose we are given the emitted power, Φ_i^e, and reflectance, R_i, of each surface, s_i, in the environment, and we wish to find the total power, Φ_i, coming from each surface. If the rays are sent in random directions rather than at specific patches, and there are n_s diffuse surfaces, this yields:

```
for i = 1 to n_s
    Φ_i^unsent = Φ_i = Φ_i^e
while (not converged)
    choose surface s_source to emit unsent power
    choose n_r rays ray_j to send
    for j = 1 to n_r
        find surface s_hit hit by ray_j
        Φ_hit = Φ_hit + (R_hit Φ_source^unsent)/n_r
        Φ_hit^unsent = Φ_hit^unsent + (R_hit Φ_source^unsent)/n_r
    Φ_source^unsent = 0
```

If mirrors are added to the scene, light can bounce directly off a mirror back to the source patch, so the algorithm becomes:

```
for i = 1 to n_s
    Φ_i^unsent = Φ_i = Φ_i^e
while (not converged)
    choose surface s_source to emit unsent power
    choose n_r rays ray_j to send
    Φ = Φ_source^unsent
    Φ_source^unsent = 0
    for j = 1 to n_r
        k = 1.0
        while (not done)
                find surface s_hit hit by ray_j
                if s_hit is diffuse
                  Φ_hit = Φ_hit + kR_hit Φ/n_r
                  Φ_hit^unsent = Φ_hit^unsent + kR_hit Φ/n_r
                  done = True
                else
                  k = kR_hit
                  reflect ray_j off surface s_hit
```

The Φ_i can then be converted to radiances using Equation 1. Accuracy can be improved by making the number of rays n_r proportional to the energy being represented by the particular set of rays. The directions are chosen in a stratified pattern as is done with the illumination feeler rays in Section 3.1.

Since this solution is to be combined with the standard viewing and illumination ray tracing solutions, the direct and direct-specular transport components must be removed. This is done by shooting from each light source in turn and then setting Φ_i to zero. Finally, the patch radiances must be transported to patch vertices as described in [CG85].

3.3 Viewing and Direct Lighting

Direct lighting is calculated in the final, viewing, stage. The viewing rays operate as in a standard distributed ray tracer[Coo86], except that the radiosity and illumination map values are substituted for the conventional ambient term. The only change made in this algorithm is in the treatment of shadow rays, described in the remainder of this section.

One problem with Whitted-style ray tracing is that every viewing ray that hits a diffuse surface generates a shadow ray for every light in the environment. This is computationally expensive if the number of light sources is large. The shadow ray intersection test is unusual in that it does not matter which object is hit by the ray. This observation led to the use of the light-buffer, which optimized shadow testing for point light sources[HG86]. Another way to optimize shadow testing is to reduce the total number of shadow rays as suggested by Kajiya[Kaj86]. This second method is employed in this paper because it preserves the ability to produce soft shadows, and keeps the algorithm simple.

In a realistic ray tracer many viewing rays are sent through each pixel, so a less accurate direct lighting estimate for each viewing ray is acceptable. The number of shadow rays determines the accuracy of the direct lighting component. As a first approximation, there should be about as many shadow rays per pixel as there are viewing rays per pixel. The number of viewing rays for a pixel must be sufficient to provide an adequate statistical estimate of whether the pixel contains an edge between a bright and a dark region. The shadow rays for a particular pixel must provide an adequate statistical estimate for a region partially in shadow and partially not in shadow. Since the shadowed regions will be the dark values for the image, and the unshadowed regions will be the bright values[5] for the image, the shadow

[5]The light sources themselves are the brightest parts of the image. This can safely be ignored because the brightness

rays are averaging values similar to those of the viewing rays. This means each viewing ray that eventually reaches a diffuse surface should generate approximately one shadow ray.

In traditional distributed ray tracing, shadow rays are sent to random points on each light source[CPC84]. Instead one ray can be sent to a random point chosen from all the light source surfaces. First a target selection probability is assigned to each source. This probability is proportional to the energy coming from the light source to the intersection point as calculated by Equation 2. This ensures that rays are likely to be sent to the nearest and brightest light sources. Once the candidate source is chosen, a ray is sent to a random point on the candidate light. If the ray hits an obstruction no direct lighting component is added. Otherwise it is assumed to be illuminated by *all* sources. For example, suppose there are ten lights of equal brightness. If there are twenty viewing rays through the pixel that views the point, then about two rays will be sent to each light. If the point can 'see' four of the lights, about eight rays will not be blocked and each of these will return the lighting for all ten lights. The other twelve shadow rays will be blocked and return zero radiance. The average of the samples will be four times the ten light contributions divided by the twenty samples, or four light contributions, as expected.

The random values used to generate the shadow rays are stratified using uncorrelated jittering as described by Cook[Coo86].

4 Results

The algorithm of Section 3 was implemented in C++ on an Encore Multimax. The primitives used are polygonal meshes. Each mesh made of a diffuse material has an associated illumination map with user set resolution. When a viewing ray hits a polygon, the direct lighting is calculated, a value is retrieved from the illumination map of the parent mesh, and soft radiosity lighting is interpolated from the vertices of the polygon. The direct, illumination map, and radiosity components are then summed and multiplied by the surface reflectance to give the total radiance value seen along the ray. Because the illumination maps are associated with the meshes, and radiosity values are stored at polygon vertices, the resolution of the illumination maps is independent of the level of mesh polygonalization.

If the mesh polygonalization or the illumination map resolution is too coarse, Mach banding appears on diffuse surfaces. Higher resolutions are needed when hard lighting is important. This is more true for illumination maps than for mesh polygonalization because the radiosity values approximate the slowly changing soft lighting components. Very fine illumination maps are usually needed only when light is focused, such as in the caustic in the ring of Figure 3.

of the light sources is usually clamped to a value near the brightest non-light because of the limited dynamic range of display devices.

fig.	*total rays*	*view rays*	*shad. rays*	*illum. rays*	*radios. rays*
5	26,482,766	61%	39%		0.2%
6	15,924,534	55%	45%		0.3%
7	52,842,224	74%	25%	0.9%	0.2%

Table 1: Ray counts for figures

Since most of the work done in ray tracing code is expended in ray intersection tests, ray counts are more informative than machine/implementation dependent timings. Ray totals and the proportion of work done for viewing, shadowing, illumination ray tracing, and radiosity for figures 5, 6, and 7 are shown in Table 1. These three figures have 1024 by 768 resolution.

Figure 5 shows a room with nine lights. It was traced with sixteen viewing rays per pixel. Illumination ray tracing was not used for this figure. Because the radiosity calculations were done on a course grid (32 patches for each wall and each table panel), very little computation was spent on this stage (see Table 1). Fewer shadow rays than viewing rays were required for this picture because there is no direct lighting on the ceiling, so none of the viewing rays hitting the ceiling generated shadow rays.

Figure 6 shows a room with one light. A bump map has been added to the walls to simulate the depressions in the mortar between the blocks[Bli78]. This figure illustrates both the strengths and weaknesses of bump mapping; the back wall looks fairly good, while the left wall is close enough to reveal the painted-on quality of the bump map. The sharp shadows coming from the chair and table legs, usually the bane of radiosity, are achieved easily because the direct lighting is calculated in the viewing stage.

Figure 7 shows a room with two lights and a bottle and a glass. Illumination rays are responsible for the caustics on the table. The banding in the shadow of the bottle results from the inadequate polygonal representation of the bottle; shadows of clear objects reveal imperfections not evident in the appearances of the objects themselves. The large number of viewing rays required for this picture results from the deep viewing ray trees for primary rays that hit the glass objects.

Figure 8 shows six simple rooms. On the top left is a ray traced picture with no ambient lighting. On the top middle is the same picture with an added ambient term (Whitted-style algorithm). On the top right the ambient term is replaced by a radiosity calculation. On the bottom left illumination ray tracing has added a caustic under the ball. On the bottom middle the light source has been shrunk causing sharper shadows. Since the shadows are generated in the viewing stage, the higher frequencies do not lengthen the radiosity calculations. On the bottom right the glass ball has been replaced by a second diffuse block.

5 Conclusion

The ray tracing algorithm presented in this paper is useful for producing pictures of diffuse/specular environments. The detail producing lighting components are calculated using viewing and illumination rays. The lower frequency diffuse lighting is calculated by a radiosity method. This separation follows the spirit of substructuring in conventional radiosity[CGIB86]. Direct lighting calculations allow the use of bump maps and are minimized by limiting the total number of shadow rays.

The conversion of a standard ray tracer to follow this algorithm is fairly easy. The ray tracing and texture map tools needed for the illumination ray tracer already exist in most ray tracing codes. The radiosity calculations are straightforward except for the interpolation to patch vertices. These calculations can be done on polygonal or higher order patches. Since the radiosity calculations model slowly changing lighting components, a very coarse discretization can be used.

The most important conclusion of this paper is that *all* ray tracers that could benefit from global illumination effects should include them. This is demonstrated by the small relative performance cost of global illumination calculations as shown in Table 1. Since the execution time of the global illumination calculations is independent of image resolution, and the viewing time is proportional to image resolution, global illumination calculations are particularly inexpensive for large images.

One problem with the algorithm is that the illumination mapping requires a coordinate system to be applied to all diffuse surfaces. The radiosity calculations requires decomposition of the enironment into patches. The resolution of the illumination maps and size of radiosity patches has an effect on image run time and quality. These parameters are not selected automatically, so some experience is needed to use the algorithm.

Possible extensions to the algorithm include the addition of an isotropically scattering medium[RT87]. The medium could be broken into elements and treated in the radiosity step. Dispersion could also be added as done by Thomas[Tho86].

6 Acknowledgements

Thanks to Bill Kubitz (advisor for this project), Holly Rushmeier, Jean Buckley, Allan Tuchman, Greg Rogers, Kelvin Sung, and Bill Brown for helpful comments on the paper. Thanks also to the reviewers for their frank and detailed suggestions.

References

[Arv85] James Arvo. Backward ray tracing. *Developments in Ray Tracing*, pages 259–263, 1985. ACM Siggraph '85 Course Notes.

[BFP83] William E. Brackett, Wayne L. Fink, and William Pierpoint. Interior point-by-point calculations in obstructed spaces. *Journal of the Illumination Engineering Society*, pages 14–25, October 1983.

[Bli78] James Blinn. Simulation of wrinkled surfaces. *Computer Graphics*, 12(3):286–292, August 1978. ACM Siggraph '78 Conference Proceedings.

[CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988. ACM Siggraph '88 Conference Proceedings.

[CG85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: a radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. ACM Siggraph '85 Conference Proceedings.

[CGIB86] Micheal F. Cohen, Donald P. Greenberg, David S. Immel, and Philip J. Brock. An efficient radioisty approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(2):26–35, 1986.

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *ACMransactions on Graphics*, 5(1):51–72, January 1986.

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics*, 18(4):165–174, July 1984. ACM Siggraph '84 Conference Proceedings.

[GTG84] Cindy M. Goral, Kenneth E. Torrance, and Donald P. Greenberg. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(4):213–222, July 1984. ACM Siggraph '84 Conference Proceedings.

[HG86] Eric A. Haines and Donald P. Greenberg. The light buffer: A ray tracer shadow testing accelerator. *IEEE Computer Graphics and Applications*, 6(9):6–16, 1986.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[Mal88] Thomas J. V. Malley. A shading method for computer generated images. Master's thesis, University of Utah, June 1988.

[MBG86] Gregory M. Maxwell, Michael J. Bailey, and Victor W. Goldschmidt. Calculation of the radiation configuration factor using ray casting. *Computer Aided Design*, 18(7):371–379, September 1986.

[NN85] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer Graphics*, 19(3):23–30, July 1985. ACM Siggraph '85 Conference Proceedings.

[RT87] Holly E. Rushmeier and Kenneth E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics*, 21(4):293–302, July 1987. ACM Siggraph '87 Conference Proceedings.

[Rus88] Holly E. Rushmeier. *Realistic Image Synthesis for Scenes with Radiatively Participating Media.* PhD thesis, Cornell University, May 1988.

[SP89] Francois Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics*, 23(3):335–344, July 1989. ACM Siggraph '89 Conference Proceedings.

[Tho86] Spencer W. Thomas. Dispersive refraction in ray tracing. *Visual Computer*, 2:3–8, 1986.

[WCG87] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: a synthesis of ray tracing and radiosity methods. *Computer Graphics*, 21(4):311–320, July 1987. ACM Siggraph '87 Conference Proceedings.

[WEH89] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):335–344, July 1989. ACM Siggraph '89 Conference Proceedings.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics*, 22(4):85–92, August 1988. ACM Siggraph '88 Conference Proceedings.

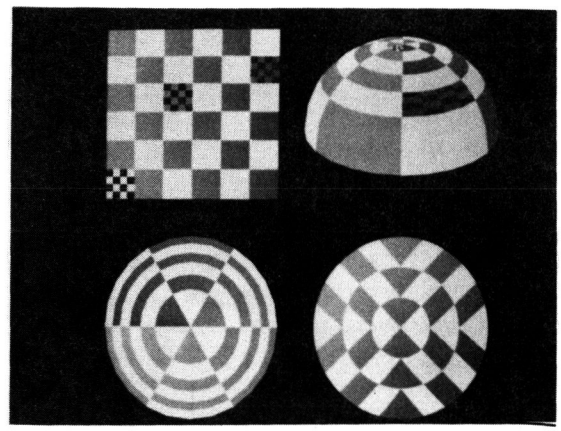

Figure 2: ***Mapping from partitioned square to hemisphere and circle.***

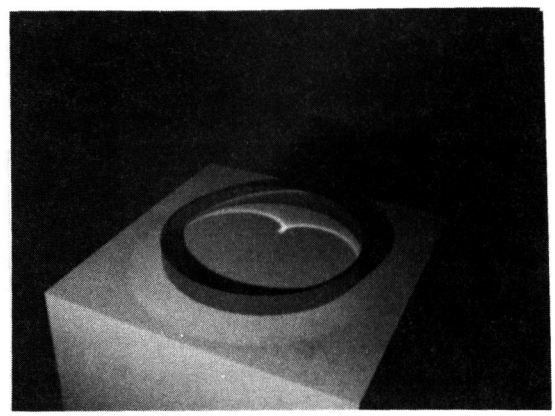

Figure 3: ***Caustic inside ring simulated with illumination rays.***

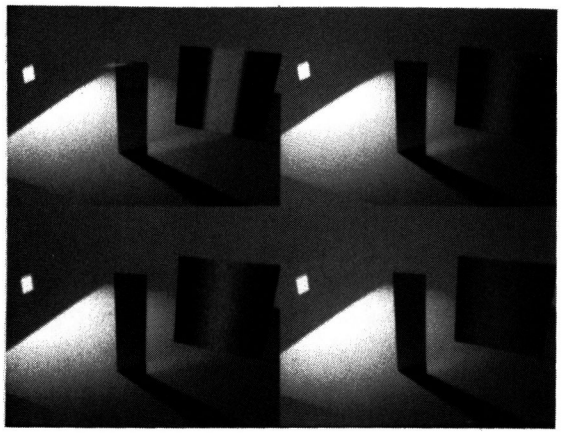

Figure 4: *Reflected light from metal plates of different roughness.*

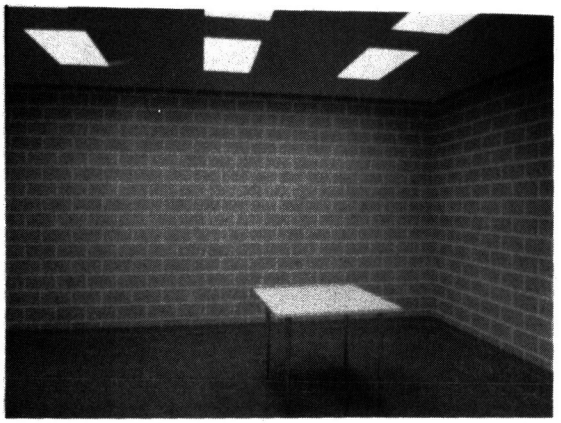

Figure 5: *Room with nine lights, where only one shadow ray is sent for each viewing ray. Sixteen viewing rayys are shot for each pixel.*

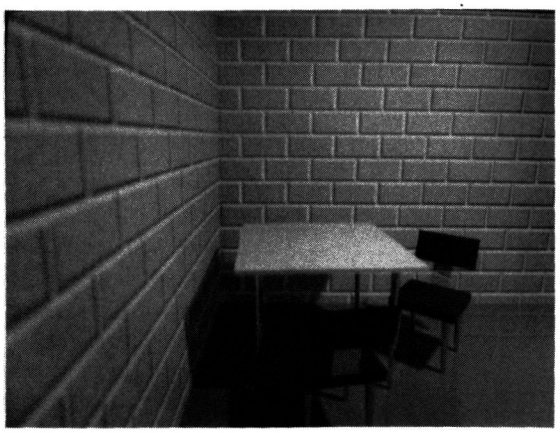

Figure 6: *Room with bump mapped walls. Separate calculation of direct lighting allows directional shading on the bump maps.*

Figure 7: *Wine glass and bottle with hard indirect lighting in the shadows of the bottle and glass calculated using illumination ray tracing.*

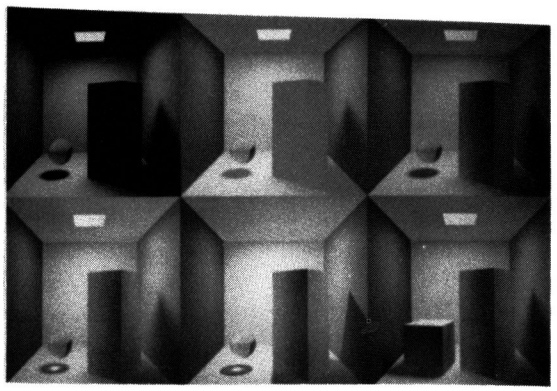

Figure 8: *Importance of lighting components.*