CHAPTER 1

Ray Tracing Terminology

Eric Haines and Peter Shirley
NVIDIA

ABSTRACT

This chapter provides background information and definitions for terms used throughout this book.

1.1 HISTORICAL NOTES

Ray tracing has a rich history in disciplines that track the movement of light in an environment, often referred to as *radiative transfer*. Graphics practitioners have imported ideas from fields such as neutron transport [2], heat transfer [6], and illumination engineering [11]. Since so many fields have studied these concepts, terminology evolves and sometimes diverges between and within disciplines. Classic papers may then appear to use terms incorrectly, which can be confusing.

The fundamental quantity of light moving along a ray is the SI unit *spectral radiance*, which remains constant along a ray (in a vacuum) and often behaves intuitively like the perceptual concept *brightness*. Before the term was standardized, spectral radiance was often called "intensity" or "brightness." In computer graphics we usually drop "spectral," as non-spectral radiance, a bulk quantity over all wavelengths, is never used.

Graphics-specific terminology related to rays has evolved over time. Almost all modern ray tracers are recursive and Monte Carlo; few now bother to call their renderer a "recursive Monte Carlo" ray tracer. In 1968, Appel [1] used rays to render images. In 1979, Whitted [16] and Kay and Greenberg [9] developed recursive ray tracing to depict accurate refraction and reflection. In 1982, Roth [13] used inside/outside interval lists along rays, as well as local instancing, to create renderings (and volume estimates) of CSG models.

E. Haines, T. Akenine-Möller (eds.), *Ray Tracing Gems*, https://doi.org/10.1007/978-1-4842-4427-2_1

In 1984, Cook et al. [4] presented *distributed* or *distribution ray tracing*. Elsewhere, this method is often called *stochastic ray tracing*[1] to avoid confusion with distributed processing. The key insight of randomly sampling to capture effects such as depth of field, fuzzy reflections, and soft shadows is used in virtually every modern ray tracer. The next few years after 1984 saw researchers rephrasing rendering using traditional radiative transfer methods. Two important algorithms were introduced in 1986. Kajiya [8] referred to the integral transport equation as the *rendering equation*. He tried various solutions, including a Monte Carlo approach he named *path tracing*. Immel, Cohen, and Greenberg [7] wrote the same transport equation in different units and solved it with a finite element method now called *radiosity*.

Since the rephrasing of the graphics problem using classic transport methods three decades ago, a great deal of work has explored how to numerically solve the problem. Key algorithmic changes include the *bidirectional* [10, 14] and *path-based* [15] methods introduced in the 1990s. Many details, including how to implement these techniques in practice, are discussed in Pharr, Jakob, and Humphreys's book [12].

1.2 DEFINITIONS

We highlight important terms used in this book. No standard set of terms exists, other than terms with standardized units, but our definitions reflect current usage in the field.

Ray casting is the process of finding the closest, or sometimes just any, object along a ray. See Chapter 2 for the definition of a ray. A ray leaves the camera through a pixel and travels until it hits the closest object. As part of shading this hit point, a new ray could be cast toward a light source to determine if the object is shadowed. See Figure 1-1.

[1]The name derives from another paper by Cook [3], where he discusses using nonuniform sampling to avoid aliasing artifacts by turning them into noise.

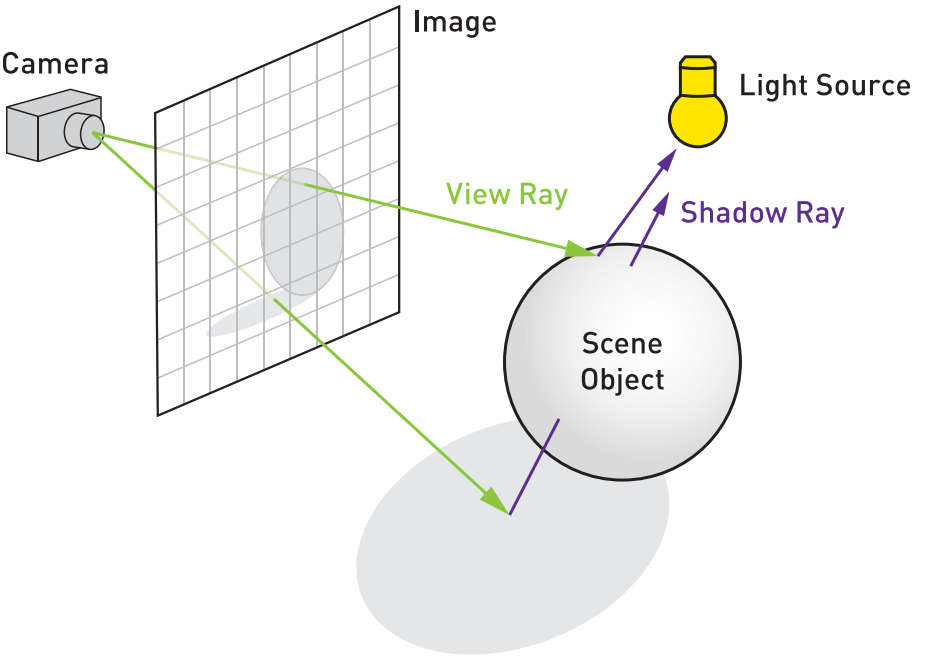

Figure 1-1. *Ray casting. A ray travels from the camera's location through a grid of pixels into the scene. At each location another ray is cast toward the light to determine if the surface is illuminated or in shadow. (Illustration after Henrik, "Ray tracing (graphics)," Wikipedia.)*

Ray tracing uses the ray casting mechanism to recursively gather light contributions from reflective and refractive objects. For example, when a mirror is encountered, a ray is cast from a hit point on the mirror in the reflection direction. Whatever this *reflection ray* intersects affects the final shading of the mirror. Likewise, transparent or glass objects may spawn both a reflection and a *refraction ray*. This process occurs recursively, with each new ray potentially spawning additional reflection and refraction rays. Recursion is usually given some cutoff limit, such as a maximum number of bounces. This tree of rays is evaluated back up its chain to give a color. As before, each intersection point can be queried whether it is shadowed by casting a ray toward each light source. See Figure 1-2.

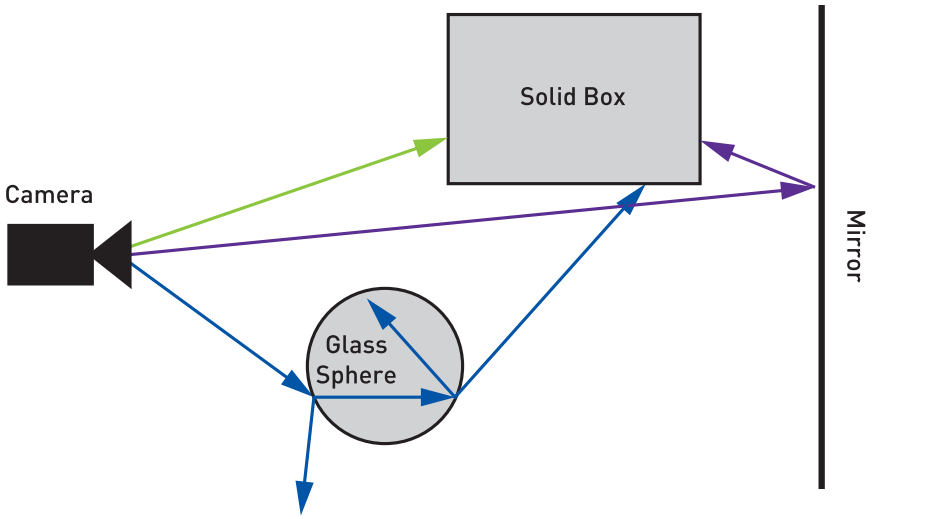

Figure 1-2. *Ray tracing. Three rays travel from the camera into the scene. The top, green ray directly hits a box. The middle, purple ray hits a mirror and reflects to pick up the back of the box. The bottom, blue ray hits a glass sphere, spawning reflection and refraction rays. The refraction ray in turn generates two more child rays, with the one traveling through the glass spawning two more.*

In *Whitted* or *classical ray tracing*, surfaces are treated as perfectly shiny and smooth, and light sources are represented as directions or infinitesimal points. In *Cook* or *stochastic ray tracing*, more rays can be emitted from nodes in the ray tree to produce various effects. For example, imagine a spherical light instead of a point source. Surfaces can now be partially illuminated, so we might shoot numerous rays toward different locations on the sphere to approximate how much illumination arrives. When integrating area light visibility, fully shadowed points lie in the *umbra*; partially lit points are inside the *penumbra*. See Figure 1-3.

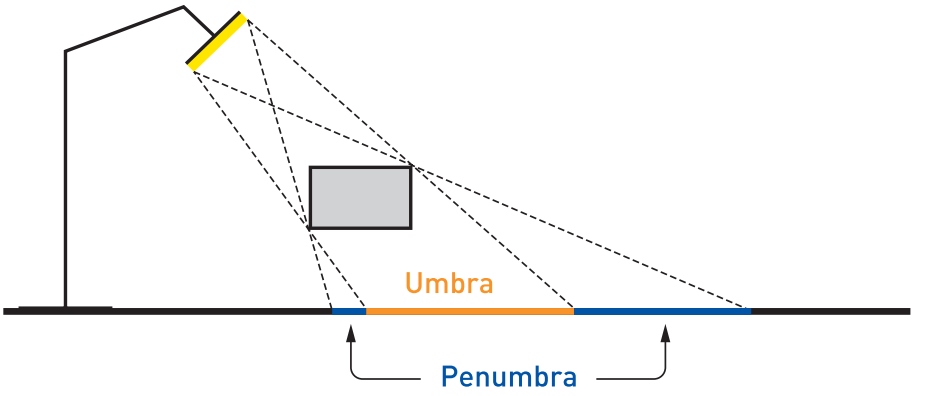

Figure 1-3. *An area light casts a soft penumbra shadow region, with the umbra being fully in shadow.*

By shooting numerous rays in a cone around the reflection direction and blending the results, we get glossy instead of mirrored reflections. See Figure 1-4. This idea of spreading samples can also be used to model translucency, depth of field, and motion blur effects.

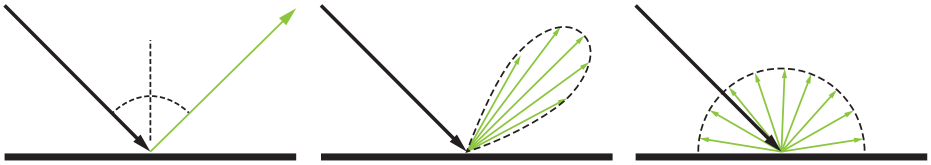

Figure 1-4. *Mirror, glossy, and diffuse reflection rays. Left: the incoming light is reflected in a single direction off a mirrored surface. Middle: the surface is polished, such as brass, reflecting light near the reflection direction and giving a glossy appearance. Right: the material is diffuse or matte, such as plaster, and incoming light scatters in all directions.*

In the real world many sources emit light, which works its way to the eye by various means, including refraction and reflection. *Glossy* surfaces reflect light in many directions, not just along the reflection direction; *diffuse* or *matte* surfaces disperse light in a wider spread still. In path tracing we reverse the light's scattering behavior, using the outgoing direction and the material to help determine the importance of various incoming directions to the surface's shade.

Tracking such complex light transport quickly becomes overwhelming and can easily lead to inefficient rendering. To create an image, we just need the light passing through the camera's lens from a specific set of directions. *Recursive ray tracing* in its various forms reverses the physical process, generating rays from the eye in directions that we know will affect the image.

In *Kajiya-style* or *path tracing* light reflects off matte surfaces in the scene, allowing for all light paths in the real world (except phase effects such as diffraction). Here a *path* refers to a series of light-object interactions that starts at the camera and ends at a light.

Each surface intersection location needs to estimate the contributions of light from all directions surrounding it, combined with its surface's reflective properties. For example, a red wall next to a white ceiling will reflect red light onto the ceiling, and vice versa. Further interreflection between the wall and ceiling occurs, as each further reflects this reflected light, which can then affect the other. By recursively summing up these effects from the eye's view, terminating only when a light is encountered, a true, physically based image can be formed.

The working phrase here is "can be"—if we shoot a set of, say, a thousand rays from an intersection point on a rough surface, then for each of those rays

we recursively send another thousand each, on and on until a light source is encountered for each ray, and we could be computing a single pixel for nearly forever. Instead, when a ray is cast from the eye and hits a visible surface, a path tracer will spawn just one ray in a useful direction from a surface. This ray in turn will spawn a new ray, on and on, with the set of rays forming a path. Blending together a number of paths traced for a pixel gives an estimate of the true radiance for the pixel, a result that improves as more paths are evaluated. Path tracing can, with proper care, give an *unbiased* result, one matching physical reality.

Most modern ray tracers use more than one ray per pixel as part of an underlying *Monte Carlo* (MC) algorithm. Cook-style and Kajiya-style algorithms are examples. These methods all have some understanding of various *probability density functions* (PDFs) over some spaces. For example, in a Cook-style ray tracer we might include a PDF over the lens area. In a path-based method the PDF would be over paths in what is called a *path space*.

Making the sampling PDF for a Monte Carlo algorithm nonuniform in order to reduce error is known as *importance sampling*. Creating random samples using low-discrepancy patterns of samples with number-theoretic methods, rather than conventional pseudo-random number generators, is known as *Quasi-Monte Carlo* (QMC) sampling. To a large extent, computer graphics practitioners use the standard terminology of the fields of MC and QMC. However, this practice can give rise to confusing synonyms. For example, "direct illumination with shadow rays" in graphics are an example of "next event estimation" in MC/QMC.

From a formal perspective, renderers are solving the *transport equation*, also commonly called the *rendering equation* for the graphics-specific problem. This is usually written as an energy-balanced equation at a point on a surface. Notation varies somewhat in the literature, but there is increasing similarity to this form:

$$L_o\left(P, \omega_o\right) = \int_{S^2} f\left(P, \omega_o, \omega_i\right) L_i\left(P, \omega_i\right) |\cos\theta_i| \, d\omega_i. \tag{1}$$

Here, L_o is the radiance leaving the surface at point P in direction ω_o, and the surface property f is the *bidirectional reflectance distribution function* (BRDF). This function is also commonly denoted with f_r or ρ. Also, L_i is the incoming light along direction ω_i, and the angle between the surface normal and the incoming light direction is θ_i, with $|\cos\theta_i|$ accounting for geometric dropoff due to this angle. By integrating the effect of light from all surfaces and objects, not just light sources, in all incoming directions and folding in the effect of the surface's BRDF, we obtain the radiance, essentially the color of the ray. As L_i normally is computed recursively, i.e., all the surfaces visible from point P must in turn have radiance values

calculated for them, path tracing and related methods are used to choose among all the possible paths, with the goal of casting each ray along the path in a direction that is significant in computing a good approximation of the effect of all possible directions.

The location point P is often left out as implicit. Also, the wavelength λ can be added as a function input. There are also more general equations that include *participating media*, such as smoke or fog, and physical optics effects, such as diffraction.

Related to participating media, *ray marching* is the process of marching along a ray by some interval, sampling it along the ray's direction. This method of casting a ray is often used for volume rendering, where there is no specific surface. Instead, at each location the effect of light on the volume is computed by some means. An alternative to ray marching is to simulate the collisions in a volume.

Ray marching, typically under some variant of Hart's *sphere tracing* algorithm [5], is also used to describe the process of intersecting a surface defined by an implicit distance equation or inside/outside test by sampling points along the ray in a search for the surface. The "sphere" in this case is a sphere of equidistant points from the surface; it has nothing to do with intersecting spheres. Following our earlier notation, this process would ideally be called "sphere casting" instead of "sphere tracing." This type of intersection testing is commonly seen in demoscene programs and is popularized online by the Shadertoy website.

We have touched upon just the basics of ray-related rendering techniques and the terminology used. See this book's website http://raytracinggems.com for a guide to further resources.

REFERENCES

[1] Appel, A. Some Techniques for Shading Machine Renderings of Solids. In *AFIPS '68 Spring Joint Computer Conference* (1968), pp. 37–45.

[2] Arvo, J., and Kirk, D. Particle Transport and Image Synthesis. *Computer Graphics (SIGGRAPH) 24*, 4 (1990), 63–66.

[3] Cook, R. L. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics 5*, 1 (Jan. 1986), 51–72.

[4] Cook, R. L., Porter, T., and Carpenter, L. Distributed Ray Tracing. *Computer Graphics (SIGGRAPH) 18*, 3 (1984), 137–145.

[5] Hart, J. C. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer 12*, 10 (Dec 1996), 527–545.

[6] Howell, J. R., Menguc, M. P., and Siegel, R. *Thermal Radiation Heat Transfer*. CRC Press, 2015.

[7] Immel, D. S., Cohen, M. F., and Greenberg, D. P. A Radiosity Method for Non-Diffuse Environments. *Computer Graphics (SIGGRAPH) 20*, 4 (Aug. 1986), 133–142.

[8] Kajiya, J. T. The Rendering Equation. *Computer Graphics (SIGGRAPH)* (1986), 143–150.

[9] Kay, D. S., and Greenberg, D. Transparency for Computer Synthesized Images. *Computer Graphics (SIGGRAPH) 13*, 2 (1979), 158–164.

[10] Lafortune, E. P. Bidirectional Path Tracing. In *Compugraphics* (1993), pp. 145–153.

[11] Larson, G. W., and Shakespeare, R. *Rendering with Radiance: The Art and Science of Lighting Visualization*. Booksurge LLC, 2004.

[12] Pharr, M., Jakob, W., and Humphreys, G. *Physically Based Rendering: From Theory to Implementation*, third ed. Morgan Kaufmann, 2016.

[13] Roth, S. D. Ray Casting for Modeling Solids. *Computer Graphics and Image Processing 18*, 2 (1982), 109–144.

[14] Veach, E., and Guibas, L. Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques* (1995), pp. 145–167.

[15] Veach, E., and Guibas, L. J. Metropolis Light Transport. In *Proceedings of SIGGRAPH* (1997), pp. 65–76.

[16] Whitted, T. An Improved Illumination Model for Shaded Display. *Communications of the ACM 23*, 6 (June 1980), 343–349.