Radiometry, Radiosity and Photon Mapping

When images produced using the rendering techniques described in Chapter 3 just are not giving you the realism you require probably the first technique that comes to mind to improve things is *ray tracing*. There are other techniques for generating high quality synthetic images but they will not be discussed here. Ray tracing may be slow but it does give superb, almost photographic quality, images and better algorithms are being developed all the time, so that under certain conditions it is even possible to achieve real-time ray tracing. There are many texts that describe the theory of ray tracing in detail, Watt [9] in chapters 7 and 8 covers all the basics and in Glassner [3] the story is continued. Shirley [8] specializes in realistic ray tracing. A full description of a complete package is given by Lindley [5] and there are also quite a few good publicly available freeware and shareware ray tracing programs with code that can be retrieved from many FTP sites and open source repositories.

Despite the popularity of the ray tracing algorithm some would argue that it does not implement a true model of the illumination process and cannot easily image things like the penumbra of soft shadows, particulate light scattering or the caustics due to light passing through refractive glass A lot of work has been done on an alternative approach know as radiosity, and this has been used to produce some of the most realistic images ever synthesized, however radiosity is not perfect, it cannot account for specular reflections for example. We will take a brief look at the principle on which radiosity is based and explain why, even with the realism it offers, it has not come to dominate rendering software. Radiosity tends to be explained in abstract mathematical terms in most texts, for a more practically orientated introduction the book by Ashdown [2], although quite old offers a programmer's perceptive.

0.1 Radiometric lighting and shading

Light is an electromagnetic (EM) radiation and physicists assign it a wave-particle duality, which means that they can model its behavior using the either the behavior of waves, or ballistic particles. As observers we mostly perceive light as a reflection from the world around us. This can be the scattered light in

a blue sky or the yellow light reflected from a white object lying in the shadow of a yellow object that is in bright sunlight. Accounting for multi-path illumination realistically is one of the major challenges in computer graphics. The light that we perceive has a wavelength lying in that part of the EM spectrum between about 390nm and 700nm, and what we see when a scene is illuminate is the refection due to the way the light interacts with surface materials.

The way light or any other electromagnetic radiation interacts with a surface is usually modeled using the wave approach and is based on the assumption of conservation of energy in which the incident light is either absorbed, transmitted, or reflected, there can also be an emissive component, this model is know as the *radiometric* model as it valid across the EM spectrum.

A very detailed study of all aspects or the radiometric model of light transport in the context of computer graphics and all other aspects of physical rendering may be found in Pharr and Hymphreys [7], which includes a detailed description of a C++ implementation and some beautiful examples of rendered scenes.

In reality, light enters the scene from the light sources, it bounces around the scene, passes through transparent materials or is absorbed, sometimes to be re-emitted. Once an equilibrium has been established, every point on every surface is acting as a small radiator and will be emitting some light, in addition to the active sources (*what we call the lights*). With the exception of the active emitters, most of the emission will be due to reflection, sampling the light emanating form at a point on a surface gives the color that we perceive with or eyes or recored in a photograph. Trying to calculate the light emerging from a every point on a surface due to the illumination from light emerging from every other point on the surface the essence of the radiosity algorithm.

The radiometric model says that at a point on a surface, the light output T_o, may be obtained by considering a function called the Bi-directional Reflection Function (BRDF) which relates the light reflected in a particular direction to the light incident from any direction. For many materials the BRDF is determined experimentally but a number of theoretical models and simple approximations have been proposed that work well. Making use of a BRDF in a practical program adds to the complexity because at each surface point P , the output of the BRDF must be integrated over all possible incident directions.

For a surface that has some fraction of transparency and mirror specular reflection [1] it is possible to consider these terms separately and only apply the BRDF to the ambient illumination fraction. Thus the light output seen at any point on a surface may be expressed as:

$$T_o = f_a T_{av} + (1 - f_a) T_s \tag{1}$$

where f_a is the fraction of the output due to ambient illumination and subject to determination by a BRDF. T_{av} is the ambient illumination due to the environment , including any direct illumination from light sources. In this model,

[1] The term specular here is used in its correct sense meaning true reflection as determined by Fresnel's laws of reflection for EM waves. In chapter **??** the term specular was used to describe an approximation of a BRDF for near mirror surfaces.

that fraction of radiometric output not accounted for by ambient illumination is assumed to arise from specular reflection, transmission and emission (this is represented in equation 1 by T_s.

T_s in equation 1 is obtained from:

$$T_s = f_r T_r + (1 - f_r) T_{ta} \tag{2}$$

f_r is a measure of the reflectivity and is determined using the well know Fresnel equations, see section 0.1.2. T_r is determined by ray-tracing along the reflected direction and obtaining the illumination coming from that direction (possibly from the sky, ground or other objects) T_{ta} is made up by calculating that fraction of the incident light that is absorbed or transmitted:

$$T_{ta} = (1 - f_e) T_t + f_e T_a \tag{3}$$

The fraction $f_e T_a$ is the emission from the surface. T_t is determined by ray-tracing along the refracted direction to obtain the illumination coming from that direction.

The fraction f_a in equation 1 can be determined for each material before any actual imaging begins. This is the fraction of incident radiation not arriving from any reflections or refractions, it can be calculated by integrating the BRDF intensity function over a hemisphere of all reflections surrounding the point of illumination on the outside of the surface.

0.1.1 The BRDF

Normally a BRDF is a function of four angular parameters and some other material dependent properties which are not angular dependent $B(\phi_o, \theta_o, \phi_i, \theta_i, ...)$ Thus :

$$f_a = \frac{1}{2\pi} \int_0^{2\pi} \int_0^{pi/2} \int_0^{2\pi} \int_0^{pi/2} B(\phi_o, \theta_o, \phi_i, \theta_i, ...) \sin\theta_o \sin\theta_i \cos\theta_i d\theta_i d\phi_i d\theta_o d\phi_o \tag{4}$$

The factor $\frac{1}{2\pi}$ is required to normalize for the surface area of the hemisphere. The $\cos\theta_i$ term arises because the BRDF function is defined in a way that makes it independent of the orientation of the surface normal relative to the incident direction, and so it must be included to calculate the actual incident energy falling on the surface. When a BRDF function is isotropic the integral in equation 4 can be simplified.

The reflected (diffuse) intensity measured at any angle θ_o may be obtained from:

$$I(\theta_o) = 2\pi \int_0^{pi/2} B(\theta_o, \theta_i) \sin\theta_i \cos\theta_i d\theta_i \tag{5}$$

and the complete f_a from:

$$f_a = \int_0^{pi/2} \int_0^{pi/2} B(\theta_o, \theta_i) \sin\theta_o \sin\theta_i \cos\theta_i d\theta_i d\theta_o \tag{6}$$

When rendering an individual pixel, T_{av} is determined by evaluating the discrete equivalent of equation 5. Weighting it by the color detected by a number of feeler rays which originates on the polygon of interest and sense in a direction specified by θ_i, ϕ_i. (The isotropy of the BRDF makes it independent of ϕ_i but that does not mean that the illumination is isotropic and so it is still necessary to feel in all directions over the whole hemisphere.) If the number of feeler rays equals the number of lights and they are directed at the lights then then T_{av} will determine the direct illumination. For a constant BRDF $f(...) = \frac{1}{\pi}$ the resulting illumination reduces to the basic Lambertian model.

If N feelers are sent out to sense the illumination in the environment.

$$w_k = 2\pi B(\theta_o, \theta_{i_k} sin\theta_{i_k} \cos\theta_{i_k})$$

k is the feeler ray identifier. To determine the average illumination for a specific θ_o :

$$T_{av} = \sum_{k=0}^{N} \frac{T(\theta_{i_k}) w_k}{W}$$

where $W = \sum_{k=0}^{N} w_k$

$T(\theta_{i_k})$ is the angle between the surface normal and the feeler ray, and in this context, θ_o is the angle between the input ray and the surface normal.

If a Lambertian diffuse model is employed the equivalent BRDF is independent of any directional bias and its BRDF is given by $B(\theta_o, \theta_i) = \frac{1}{\pi}$. In such cases $I(\theta_o) = 1$ and $f_a = 1$. A better BRDF model that is sometimes used is :

$$B(\theta_o, \theta_i) = \frac{A}{(B + (sin\theta_o - sin\theta_i)^g)} \tag{7}$$

Values of $g = 0$, $A = 0.63661$ and $B = 1$ in this model are equivalent to a Lambertian diffuse model with no specular component.

0.1.2 Specular reflection and transmission

For a material in which, t is the layer thickness, n_i is the imaginary component of the refractive index and θ is the angle of incidence, the attenuation α is given by:

$$\alpha = \exp(\frac{2\omega t}{c \cos\theta} n_i) \tag{8}$$

ω is the frequency and c the speed of light.

When α exceeds 1 the attenuation is complete, thus the fraction of observed light used in equation **??** is limited to give:

$$f_a = \begin{cases} \alpha & \text{if } \alpha < 1 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

The specular reflection component is determined using the material properties of its thickness and complex refractive index. To determine the reflected fraction Fresnel's equations are used. Equation **??** determines the absorption

in the refracted component; The absorbed fraction is used to determine the fraction of the observed illumination due to emission from the surface.

The reflected component is given by:

$$f_r = rr^* \tag{10}$$

Where r is the complex reflected amplitude ratio given by the Fresnel equations and, r^* is its complex conjugate.

For the TE component of the specular wave r is given by:

$$r_{TE} = \left(\frac{\cos\theta - \sqrt{n^2 - \sin^2\theta}}{\cos\theta + \sqrt{n^2 - \sin^2\theta}} \right)^2 \tag{11}$$

n is the complex refractive index of the material into which the ray is transiting as it passed through the polygon. θ is the angle of incidence. This assumes that a ray always enters a polygon from air and that: where a layer is thin (e.g. clothing) no modeling takes place inside the layer itself so that the transmitted ray passes through into the next *air gap* before the ray tracing continues. For thick and/or absorbent layers or material with a large dielectric constant ($n = \sqrt{\epsilon}$) the model assumes that there is no transmitted component.

And for the TM component::

$$r_{TM} = \left(\frac{-n^2\cos\theta + \sqrt{n^2 - \sin^2\theta}}{n^2\cos\theta + \sqrt{n^2 - \sin^2\theta}} \right)^2 \tag{12}$$

When a ray strikes a polygon in the model, its incident direction is resolved into two components TE and TM relative to the surface polygon's orientation, r computed for each component and then f_r determined.

This short section has introduced some of the enhancements that can be made to the basic lighting model to take advantage of the ability of the ray tracing algorithm to render more realistic surface materials, for example denim cloth has a very different BRDF functions than that of billiard balls, and hair can have an anisotropic translucency.

0.2 Radiosity and Photon Mapping

Section 0.1 delved a little deeper into the physics on which the rendering process is based. It was highlighted that the light we see coming from any point $\mathbf{p}$ in the scene (thinking of it in a different way - the surface color of the object we see) is due to either direct emission or occurs as a result of an accumulated reflection from the surroundings of the object. It is convenient to think of the reflection as being composed of two terms. Firstly there is the specular, or directly mirrored, component, and secondly there is the reflection from all the other objects in the surroundings, in the simple lighting model we call this the ambient and diffuse reflection. The material properties themselves come into play by modulating

the ambient illumination, sometimes in a way that depends on the direction that the ambient light is coming form, this is specified with the use of a BRDF function. In the radiometric model, the BRDF function and light falling on **p** was integrated to determine the resultant reflected component. In an almost endless cycle the reflected light is also distributed around the scene and may indeed contribute again to the illumination of its origin **p**.

So, every point in the scene has the potential to contribute to the illumination of every point in the scene, how can we solve this problem. Setting up an approximation by divided the scene up into a number of tiny patches allows us to formulate the problem as a finite number of summations, the interrelations between patches may be formulated as equations linear in the *radiosity* (how much light is emerging from a point) of each patch, and the whole set of equations can then be solved to determine the true radiosity of each patch. Well that's the theory!

Unfortunately there are many practical difficulties, especially computational, but radiosity has the potential to give the most accurate visual rendition of a scene illuminated diffusely, because, in the limit of infinitely small patches it fullly solves all the physical equations of light transport. There is one advantage of radiosity that has utility for some applications and it is the fact that once the radiosity of every surface has been calculated, it does not matter from where the scene is observed, all the information is there. The radiosity values can be converted into a light map and painted onto the surface in much the same way that an image map is painted onto the surface. By using a light map, a real-time renderer can avoid having to do any lighting calculations at all, and with just an image map lookup, a view of the scene is generated. This is ideal for walk-through and static scene visualization, it is also good for generating very atmospheric environments for virtual reality or computer games in which some additional animated elements are added and rendered as needed, against a background created with a light map generated using radiosity and stored as a texture.

0.2.1 Radiosity in practice

Before beginning rendering, the polygonal faces that describe the scene might need to be broken up into N small patches, each with approximately the same area. The patches most be small enough so that their discretization is not visible in the final image, for example along the edges of a shadow.

The radiosity B_i of patch i is given by :

$$Bi = Ei + Ri \sum_{j}^{N} BjF_{i,j} \tag{13}$$

where B_i is the radiosity, E_i is the emissivity, R_i is the reflectivity and F_i, j is called the form factor. The form factor expresses the contribution to the illumination of patch i due to patch j. This is a geometric relationship depending on the distance between the two patches, their relative orientation and whether

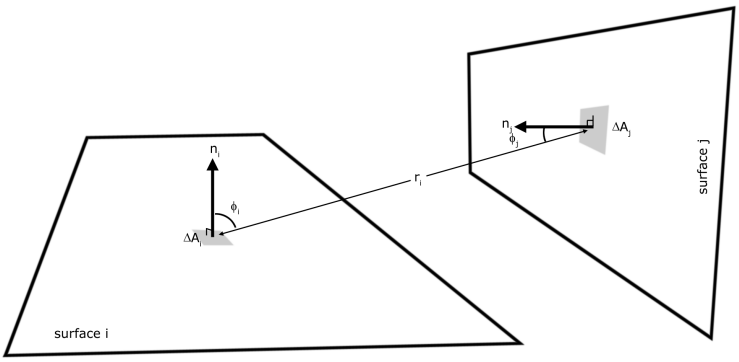

Figure 1: The radiosity form factor between patches i and j depends on their distance apart and their relative orientation.

patch j is directly visible from patch i. The form factor relationship is illustrated in figure 1 and the form factor between two small areas dA_i and dA_j can be determined from:

$$F_{i,j} = \frac{1}{A_i} \sum_{A_i} \sum_{A_j} \frac{\cos\phi_i \cos\phi_j}{\pi r^2} \Delta A_i \Delta A_j \tag{14}$$

The areas and angles in equation 14 are illustrated in figurefig:4radiosity, the ΔAs are the patch areas r is the distance between patches i and j and the ϕs are the angles between the patch normals and the vector linking the patches.

Once the form factors have all been determined then equation 13 is written for each patch, it is then re-arranged as a set of N linear equations that can be solved for the B_i:

$$\begin{bmatrix} 1 - R_1F_{1,1} & -R_1F_{1,2} & .. & -R_1F_{1,n} \\ -R_2F_{2,1} & 1 - R_2F_{2,2} & .. & -R_2F_{2,n} \\ .. & .. & .. & .. \\ -R_{n-1}F_{n-1,1} & -R_{n-1}F_{n-1,2} & .. & -R_{n-1}F_{n-1,n} \\ -R_nF_{n,1} & -R_nF_{n,2} & .. & 1 - R_nF_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ .. \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ .. \\ E_n \end{bmatrix} \tag{15}$$

Despite the elegance of equations 15 and 14 it is not really practical to use them directly. To calculate $F_{i,j}$ it is necessary to determine whether patch j is visible at patch i, this could mean checking if any polygon in the model is putting it in shadow. On its own this calculation is prohibitive. Solving an set of N equations (equation 15) is a well know problem in linear algebra but when N might be several million patches a direct solution is again prohibitively expensive to obtain. In practice the form factor is determined using

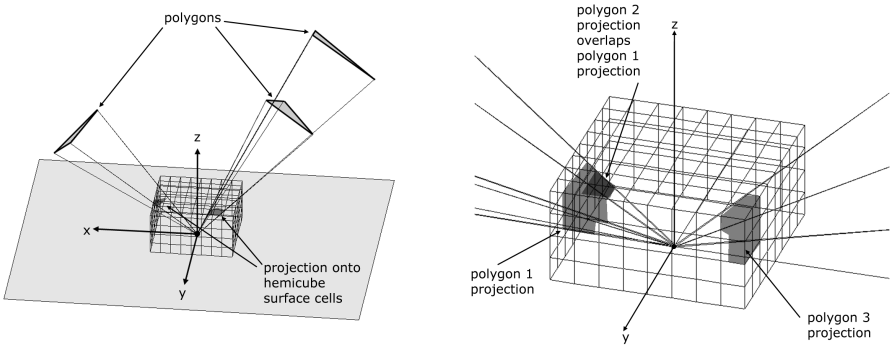

Figure 2: The hemicube surrounding a patch in the radiosity algorithm. Where the projection of one polygon overlaps another the one further away is discarded. Each square cell on the hemicube surface accumulates incident radiation from any polygons that project into it.

the *hemicube* approximation and 14 is solved using a custom iterative approach called progressive radiosity.

The hemicube serves an analogous role to the shadow buffer that is used to render shadows in a real-time rendering engine. The hemicube, depicted in figure2 is a cube that surrounds patch i. The surface of the cube is divided up into small squares and the other patches are projected onto the surface of the cube, any square that surface patch j projects onto records the distance to patch j. Once all the $N-1$ patches have been processed the identity of the patch closest to patch i will be known in each square over the surface of the hemicube. By working through each of the squares on the surface of the hemicube the form factors between any visible patch i can have their contribution to form factors for patch j determined.

The iterative algorithm of progressive radiosity has the advantage that it is a refinement processes and with only a few steps a fair representation of the appearance of a scene can be obtained. The idea is simple enough. Pick any patch i, work through all other patches j, calculate the form factors $F_{i,j}$ and update the radiosity R_i in patch i. Now pick another patch (sayi') and repeat the process, when all the patches have had their radiosities calculated, repeat the process, until there is little change in the radiosity values calculated at each point. This process will eventually converge to the same solution that would be obtained by solving equation 15 but there will always be at least an approximation to the final result available for use at the end of each iterative step.

As we have just described it, this Progressive radiosity algorithm is call the

gathering variant because it determines the radiosity at patch i due to all other patches. If you think of the process in reverse, that is, patch j contributes radiosity B to patch i, i', i'' etc. the the origin of the radiation is being considered first rather than the destination. This reverse process, known as the *shooting* method has many advantages because the most significant emitters of radiation, the lights, can be considered first, and their radiosities distributed across the scene. Since illumination by direct lighting has the most obvious effect on the diffuse reflections that we see: a fair approximation to the final image can be obtained in a single iteration.

In this short section we have only had time to consider the main ideas of radiosity, there are a number of links on the web site to articles and practical codes that are a good place to begin an exploration of radiosity in greater depth.

0.2.2 Photon mapping

In his book Jensen [4] proposed an idea that he called Photon mapping which attempts to offer an alternative to the Monte Carlo feeler rays that need to be used in a ray-tracing program to accurately simulate, for example, caustics, ambient illumination and area lights. As the name implies, the central idea in photon mapping is to perform a pre-rending step in which a large collection of photons are emitted by the light sources and followed as they scatter around the scene. Where the photons come to rest on the surface of the objects is recorded in an appropriate data structure along with their energy and their direction of incidence. A second, rendering pass is then performed to calculate the contribution of caustics and indirect lighting to what we see. This second pass is best done as part of a ray-tracing procedure because trying to use photon mapping alone to simulate specular reflection and direct lighting would require too many photons to be practically useful. Thus, photon mapping is an algorithm primarily used to enhance an existing ray tracer.

Figure 3 illustrates how the photons are traced through a scene during the first pass and where they gather on objects' surfaces. During the lighting calculate of a surface point $\mathbf{p}$, the photon map (the list of photons) is queried to find the nearest photons to $\mathbf{p}$. To conduct a rapid search the photon map is usually structured as a BSP tree or a balanced Kd-Tree where spatial relationships are quickly determined.

Once the nearest photons have been determined they are used to calculate the reflected radiance R emerging from $\mathbf{p}$ in the direction of observation, see figure 4. R may be determined by integrating the surface's BRDF:

$$R = \frac{1}{\pi r^2} \sum_{i=1}^{N} \phi_i BRDF(\mathbf{p}, \mathbf{v}, \mathbf{d}_i) \tag{16}$$

where N in the number of neighboring photons inside the circle of radius r around $\mathbf{p}$, $\mathbf{d}_i$ is the incident direction of photon i and $\mathbf{v}$ is the viewing direction (back along the incident ray of a ray-tracing algorithm) and ϕ is the photon's power. The BRDF function used in equation 16 would take a similar form to

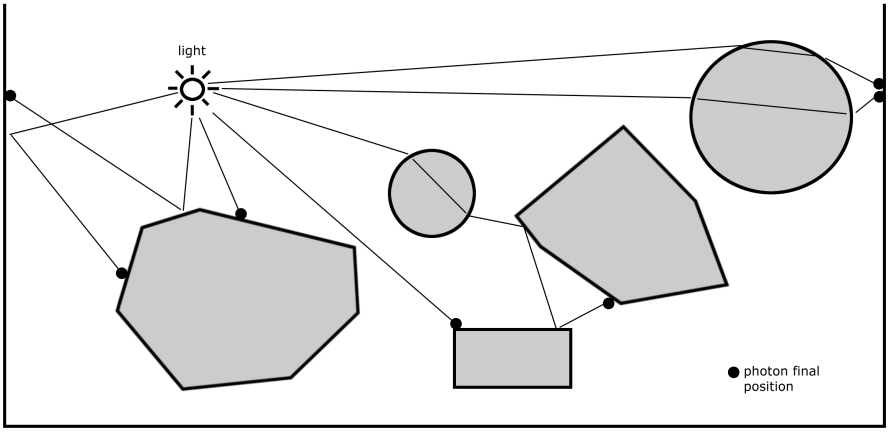

Figure 3: Photons are emitted from the light sources in a scene and scatter around the objects before finally being absorbed. The location where the photons accumulate is recored and used during the rendering phase. Up to 500,000 photons might be used in a practical scene.

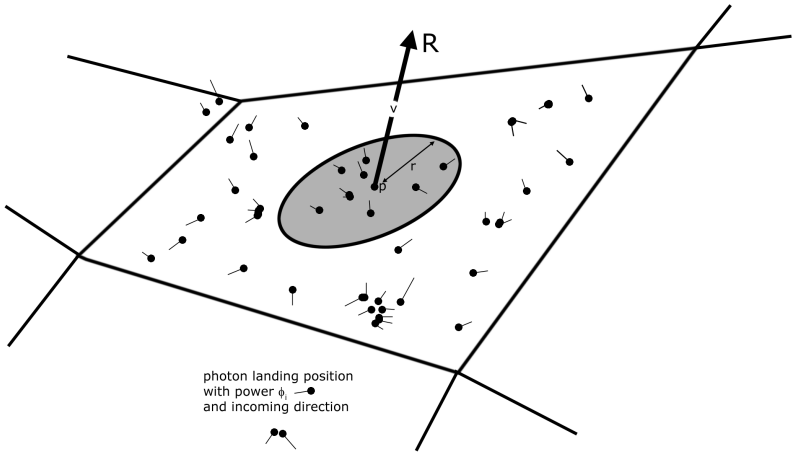

Figure 4: The radiance R emerging from $\mathbf{p}$ is calculated from the power recorded in the N nearest photons that lie inside the shaded area on a locally flat surface. In this example 9 photons that have

those discussed in section 0.1.1 and the assumption is made that the surface is locally flat in the vicinity of p.

Bibliography

[1] T. Akenine-Mōller and E. Haines. *Real-Time Rendering*. Natick MA: A. K. Peters, 1999.

[2] Ian Ashdown. *Radiosity: A Programmer's Perspective*. New York NY: John Wiley and Sons, 1994.

[3] A. S. Glassner (Ed.). *An Introduction to Ray Tracing*. London U.K.: Academic Press, 1989.

[4] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. Wellesley, MA: A.K. Peters, 2001.

[5] C. Lindley. *Practical Ray Tracing in C*. New York NY: John Wiley and Sons, 1993.

[6] K. Perlin and E. M. Hoffert. "Hypertexture." *Computer Graphics* 23.

[7] M. Pharr and G. Humphreys. *Physically Based Rendering From Theory to Implementation*. Burlington, MA: Morgan Kaufmann, 2010.

[8] P. Shirley. *Realistic Ray Tracing*. Natick MA: A. K. Peters, 2000.

[9] A. Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Reading MA: Addison Wesley, 1989.