A PHOTON MAPPING BASED APPROACH TO COMPUTING CELESTIAL ILLUMINATION

A Thesis

by

JONATHAN PENNEY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2009

Major Subject: Visualization Sciences

A PHOTON MAPPING BASED APPROACH TO COMPUTING CELESTIAL ILLUMINATION

A Thesis

by

JONATHAN PENNEY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Ergun Akleman
Committee Members,	Vinod Srinivasan
	Daniele Mortari
Head of Department,	Tim McLaughlin

May 2009

Major Subject: Visualization Sciences

ABSTRACT

A Photon Mapping Based Approach to

Computing Celestial Illumination. (May 2009)

Jonathan Penney, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Ergun Akleman

For photographers to capture good pictures of their subjects, the lighting conditions must be taken into account and adjusted for accordingly. The same holds true for a satellite attempting to photograph another object in space: it must know the lighting conditions to adjust camera settings and position itself properly to take the best photograph. This thesis presents a photon mapping based algorithm to compute a physically accurate representation of the illumination of objects in orbit around the Earth, taking into account the effects that cause refraction in the atmosphere. I also discuss the assumptions that I have made to utilize the algorithm in an interactive 3D visualization tool, which I implemented to view the illumination on objects at arbitrary positions in space. Finally, I show that the photon mapping method offers improvements over simpler methods of computing illumination.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In recent years, photon mapping has become a popular approach to creating photorealistic computer generated imagery. With faster computers and more efficient algorithms, it has become more and more possible to achieve realism on large scale productions, such as film and television. By simulating the behavior of light, photon mapping is able to create stunningly realistic imagery, and it is this physical realism that is needed to accurately compute the effects of sunlight traveling through the Earth's atmosphere.

Since satellites are unmanned, onboard automation is a key element to a satellite's functionality and sustainability. Scientists are constantly looking to improve a satellite's ability to asses it's situation and act accordingly to achieve some predetermined goal; one such goal is to analyze other objects in the Earth's orbit by taking pictures of them. In order to do so, they must have a notion of where the light is coming from. Therefore, a satellite that can consider the illumination contributions from the Earth would be a huge improvement over one that only factors in direct sunlight.

By developing an algorithm that determines how light interacts with the Earth and atmosphere, major improvements can be made to the way satellites determine the lighting conditions on a specific object. This provides the potential for drastic

The journal model is *IEEE Transactions on Visualization and Computer Graphics*.

improvements to the ability of a satellite to asses its surroundings, determine its position and orientation, and respond accordingly. The method described in this thesis could be used to provide more complete lighting information, so that more informed decisions can be made about the circumstances in which pictures are taken in space. Not only will a satellite be able to use the optimal camera settings, but it will be able to predict what the lighting conditions will be at any point in the future. This allows it to know when a target object will be illuminated, how brightly it will be illuminated, and how to set the camera for an appropriate exposure.

I.1. Terminology

In order to fully understand the processes described in this thesis, it is necessary to understand several terms. I have divided them into sections on illumination and astronomy.

I.1.1. Illumination Terms

Surfaces that reflect light uniformly in all directions are described as *Lambertian surfaces*. These surfaces have only a diffuse component, so lighting is smooth and gradual across the surface of the object.

Albedo describes the Lambertian reflectivity of a particular surface, where a value of 0 means that no light is reflected from the surface, and a value of 1 means that all light is reflected. For perfectly Lambertian surfaces, the albedo is constant across the surface, but for non perfect surfaces, such as the Earth, the albedo varies.

A *photon* is a ray of light that is cast from a light source into a scene. Photons interact with objects by reflection, refraction, or absorption, and are eventually stored in a photon map as a representation of the distribution of light in the scene. The process of casting photons into a scene and storing them is called *photon mapping.*

Global illumination describes any number of techniques that attempt to simulate the realistic behavior of light within a scene. It considers not only direct lighting, but indirect lighting as well, which comes from light bouncing off of other objects. Photon mapping is one example of global illumination.

The *phase angle* is described as the angle that light bounces off of a Lambertian surface in a certain direction. Imagine a line connecting a point on the surface of an object to the light source, and a line in a given direction from that same surface point. The angle between these two lines is the phase angle.

I.1.2. Astronomical Terms

Any arbitrary or unknown object in space is denoted as an *RSO*, which stands for Resident Space Object. More specifically, in this thesis an RSO will be any object in orbit around the Earth that is to be observed.

For any object orbiting the Earth, *altitude* is defined as the shortest distance between the object and the Earth's surface. Altitude will typically be denoted by the symbol h, and will be an important part of defining an RSO's position in space for the sake of the algorithm implemented in this thesis.

I.2. Illumination Contributions

An accurate model of the illumination of objects in orbit around the Earth needs to take into account all the significant light sources in the vicinity. For objects close to the Earth, these sources are the Sun, Earth, and Moon. Each is computed independently and becomes significant in different situations.

I.2.1. Direct Sunlight Contribution

Sunlight is the strongest source of illumination in outer space. In fact, it is so strong that when an object is in direct sunlight, the effect of other celestial light sources cannot be perceived by a typical camera without increasing the shutter speed and blowing out the sunlit portion of the image. The only times when the Sun is not the largest illumination contributor is when the observed object is in the shadow cast by another large body, like the Earth, or when looking strictly at the dark side of the observed object.

I.2.2. Earth's Contribution with Atmospheric Considerations

When the Earth is considered as a source of illumination, the atmosphere's effects must be taken into account. Sunlight bends toward the Earth as it passes through the atmosphere since it has an index of refraction greater than that of the vacuum of space. According to [15], the refractive index of the atmosphere gradually gets closer to zero the farther it gets from the Earth. This is due to the fact that the particles making up the atmosphere are less and less attracted by the Earth's gravitational pull as their distance from the surface increases.

Any sunlight that interacts with the Earth or its atmosphere is considered earthshine. These influences can be broken into three cases:

- Diffuse reflections off of the Earth's surface. This includes all light that bounces off of land masses, bodies of water, clouds, etc.
- Refraction through the atmosphere. The atmosphere can be thought of as a type of lens that bends the light towards the Earth, which actually causes the shadow cast by the Earth to have a smaller footprint than if the atmosphere is ignored.
- Scattering due to particles in the atmosphere. Rayleigh scattering occurs when sunlight reflects off microscopic particles in the atmosphere [15]. The blue wavelengths of light are scattered the most, and the red wavelengths the least. This is the reason that the sky looks blue from Earth and the atmosphere looks blue from space, and the reason that sunsets and the eclipsed Moon are reddish.

The sum of these three phenomena make up the Earth's illumination contribution, however the model implemented in this thesis considers only refraction and diffuse reflection. In order to take scattering into account, it is necessary to record directional information of photons, which also requires changes in the way reflection and refraction are computed. For more details on the changes that are required, see Section VII.1 in Future Work.

I.2.3. Moon's Illumination Contribution

Moonlight is the smallest contributor of the three illumination sources. This is partially due to the large distance from the RSO with respect to the Earth, and partially due to the low albedo of the Moon, which causes it to reflect little light. The only time the Moon is a significant source of illumination is when it is the only source, for instance, when the illuminated object is in the shadow of the Earth.

CHAPTER II

PRIOR WORK

This thesis uses photon mapping and final gathering to compute the illumination that the Earth contributes to orbiting objects. To justify this approach, I will discuss the origins of ray tracing and the major developments that led to its wide acceptance for creating realistic images. Furthermore, since the Earth's atmosphere is a continuous volume, I will describe a brief history of ray tracing through volumes. I will also outline how ray tracing gave rise to methods of global illumination, such as radiosity, photon mapping, and final gathering. Since photon mapping and final gathering are implemented in this thesis, I will go into specific details of their history. It is important to understand the progression of these concepts in order to see how the techniques are applied to this thesis.

Additionally, I will touch on a few of the works that simulate the effects of Earth's atmosphere to create rendered images. The majority of the work in this area of computer graphics has been for the purposes of rendering images of the Earth from space, or from the surface of the Earth, looking at the sky, sunset, or clouds; no major work has been done to model the illumination of objects orbiting the Earth, as this thesis does.

Finally, I will describe the phase integral equation, an analytical equation representing the percentage of light that is reflected off a perfectly Lambertian sphere in a given direction. This equation is used in this thesis to create an approximated analytical model to determine the illumination of RSOs.

II.1. Ray Tracing

Ray casting was introduced by Appel [1] as a non-recursive algorithm that casts a single ray from the eye, through a pixel on the image plane, and into the scene. If the ray encounters an object, the color of the object is determined at the intersection point based on the material properties and lighting of the scene, and then stored as the color of the pixel.

As an extension to ray casting, Whitted [28] was the first to recursively trace rays through a scene to compute reflections, refraction, and shadows. Rays are cast into the scene, where they encounter the first surface and can be reflected or refracted to encounter subsequent surfaces. All surfaces encountered along this tree of rays contribute to the color of the pixel as seen by the viewer.

Both ray casting and ray tracing involve casting a single ray from the eye point through each pixel into the scene. Distributed ray tracing, introduced by Cook et al. [5], is an extension to these methods, that casts multiple rays across an interval and computes their average. For example, depth of field is possible by distributing rays across a lens instead of using a single eye point. A ray is cast from each sample on the lens, through the desired pixel in the image plane, and into the scene. The average color returned by those rays is stored in the pixel, and the process is repeated for each pixel in the image. It is also possible to calculate soft shadows by distributing the rays across an area light source, and blurry reflections can be obtained by randomly varying the direction of the reflected ray within a range.

Further developments in ray tracing involve the rendering of volumes with varying density, such as clouds or smoke. Early approaches used analytical equations to define a ray's path through the volume [13]. Rays are cast through the volume, which is divided into a grid. Each time a ray enters a new grid element, it is reevaluated, iteratively adjusting the intensity, color, and direction based on the properties of the volume in that element. By replacing the grid structure with a series of nested spheres, this method can be used when casting rays through the Earth's atmosphere in order to account for its continuously varying refraction.

II.2. Photon Mapping

The properties of light are extremely complex and difficult to reproduce in computer graphics, but in order to create more realistic images, the physical behavior of light needs to be more accurately modeled. Radiosity is a global illumination rendering approach to compute indirect lighting, based on an analytical model of heat transfer, in scenes with perfectly diffuse surfaces. Developed at Cornell University [7], radiosity only deals with light rays that leave the light source and are diffusely reflected off of objects in the scene before reaching the eye. Although the radiosity algorithm is relatively simple, it is only capable of handling diffusely reflected light and is not structured to handle specular reflections or refractive objects very elegantly.

Photon mapping, developed by Jensen and Christensen [12], is a two-pass approach to rendering a scene that addresses the deficiencies of radiosity. In the first pass, backward ray tracing [2] is used to cast light rays, called photons, into the scene from a light source, where each photon interacts with the scene the way a light ray would in real life. Photons can be absorbed by diffuse surfaces, reflected off of specular surfaces or transmitted through refractive surfaces, such as glass, to form caustics [25]. At each object intersection, a photon's absorbed intensity is stored in a photon map, while any remaining intensity is reflected or transmitted back into the scene based on the surface properties of the object.

In the second stage of photon mapping, the scene is rendered as normal from the eye point, but the photon map is used to supplement lighting information. When a ray encounters an object, it looks at nearby photons in the photon map to approximate the radiance at the point of intersection. This approach enables effects such as color bleeding, caustics, and subsurface scattering, which drastically improve the realism of the generated image.

Further work by Jensen [11] divides the photon map into two parts: a high resolution caustics photon map and a low resolution global photon map. In the rendering step, caustics are visualized directly from the caustics photon map, requiring a high density of photons. The global photon map, however, can have a lower density of photons since it is used as an approximation of the gradual changes of lighting across a surface. A technique called final gathering, previously applied to radiosity by [22, 19, 27] is used to determine the color of a point on a surface. Final gathering rays are cast from the point into the scene, and when one of these rays intersects an object, it uses nearby photons to determine the radiance coming from that direction. Several final gathering rays are averaged together to get the color at the original surface point. By sampling photons in the environment near the surface point, instead of photons directly on the surface around the point, the noise is significantly reduced, providing a much cleaner image.

II.3. Rendering Atmospheric Phenomena

Most of the work that has been done in computer graphics to simulate the effects of the Earth's atmosphere has been for the purposes of rendering images of the sky, sunset, or clouds from the surface of the Earth [3, 14, 20, 23, 18, 8]. A few have been optimized to primarily render images of the Earth from space [21, 9], and some can handle both situations [6, 4]. Some deal only with scaterring [9, 10, 23], while others consider only the effects of refraction [17, 3, 18, 24]. A few take both scattering and refraction into account [8]. A comprehensive survey of many of these methods is provided in [26], but none were designed to model the illumination of objects in orbit around the Earth. Nonetheless, a brief history serves as a solid foundation for the model described in this thesis.

The first method for modeling the effects of the atmosphere was proposed by Klassen [16]. His model assumed that light scattered only once at each of the two atmospheric layers he used: one for the outer layer and one for the more dense layer near the ground. An improved method was developed by Kaneda et al. [14] which employed a layered atmosphere with exponentially varying density based on altitude. Berger et al. [3] applied the same layered technique to the refractive properties of the atmosphere, which vary along with density, as scattering does.

Building upon these methods, Nishita et al. [21] was the first to utilize a set of concentric spherical layers, each with a constant density. Here, scattering happens at the edge of each layer, where the density of the layer is dependent on it's altitude. In a similar approach, Irwin [9] considers only Rayleigh scattering, which is caused by air particles smaller than the wavelength of light. Jackèl and Walter [10] followed suit with a method that only uses Mie scattering, which is caused by particles larger than the wavelength of light, such as rain, clouds, and haze. Both types of scattering are implemented by Riley et al. [23] and Dobashi et al. [6], although several simplifications are made to accelerate render times.

In [20], Nishita et al. introduced a two-pass method that subdivides each atmospheric shell into a number of spherical volume units. In the first pass, these spheres record the distribution and direction of the scattered light, and the second pass gathers the scattered light along the viewing direction.

II.4. Phase Integral Function

Since Lambertian objects reflect light uniformly in all directions, there is an inherent structure to Lambertian spheres that gives them the smooth gradation of light when rotating around the sphere. There is a well known equation that models this structure, based on the direction of incoming light and the viewing direction. It assumes that the light source is a point source, and that the light source, illuminated sphere, and viewing position are considerably far away from each other with respect to the radius of the sphere. Let d_0 be the direction from the sphere to the light source and let d_1 be the direction from the sphere to the viewing position. Then let the angle between d_0 and d_1 be ξ, the phase angle. The phase integral function is given by:

$$p(\xi) = \frac{2}{3\pi}((\pi - \xi)\cos\xi + \sin\xi) \tag{2.1}$$

The phase integral is a function of the phase angle ξ. It gives the percentage of light that is reflected off of the Lambertian sphere in a direction that forms the angle ξ with d_0. This function will be the basis of the analytical models presented in this thesis.

CHAPTER III

METHODOLOGY

There are two basic approaches presented in this thesis to computing the illumination of objects orbiting the Earth, each with it's own advantages and disadvantages. An analytical approach is to assume that the Sun, Earth, and RSO are at infinity with respect to each other, so that the illumination sources can be considered to be point sources and the light rays can be considered parallel. Under these assumptions, an approach called the Lambertian sphere reflectance model can be used to quickly compute the illumination. However, it is a very limited model that is not completely accurate when objects are close relative to their size, like orbiting satellites are to the Earth.

Another approach is to use ray tracing to computationally approximate the behavior of sun rays as they travel from the Sun, bounce off of the Earth, and illuminate an RSO. Ray tracing is extremely slow compared to the Lambertian sphere reflectance model, but it is very versatile, allowing complicated atmospheric effects to be taken into account. This will be the main approach taken in this thesis.

III.1. Analytical Lambertian Sphere Reflectance

Since a Lambertian sphere reflects light uniformly in all directions, it's illumination can be represented analytically. This idea can then be extended to the general case of a set of Lambertian spheres illuminating each other down the chain.

III.1.1. A Single Sphere Illuminated by the Sun

In the simplest case, the Sun illuminates a Lambertian sphere, which reflects some of that light toward the observer, as denoted in Fig. 1.

- Let α, r, and $A = \pi r^2$ be the albedo, radius, and apparent area of the sphere, respectively.
- Let $\overrightarrow{d_0}$ and d_0 denote the vector and distance between the centers of the Sun and the sphere.
- Similarly, let $\overrightarrow{d_1}$ and d_1 denote the vector and distance between the centers of the sphere and the observer.
- Finally, let ξ be the angle between $-\overrightarrow{d_0}$ and $\overrightarrow{d_1}$, where $p(\xi)$ is the phase integral function.

The equation signifying the illumination of the sphere as observed by the observer is then:

$$\frac{1}{4\pi d_0^2} \frac{Ap(\xi)\alpha}{2\pi d_1^2} \tag{3.1}$$

In order to take into account the case where the sphere is in the shadow of the Earth[1], a value, *shadow*, is computed as a number between 0 and 1 that denotes how much total sunlight the sphere receives, due to the Earth's occlusion of the Sun, where 0

[1]Shadows cast by the Moon and other large bodies are ignored due to their infrequent occurrence.

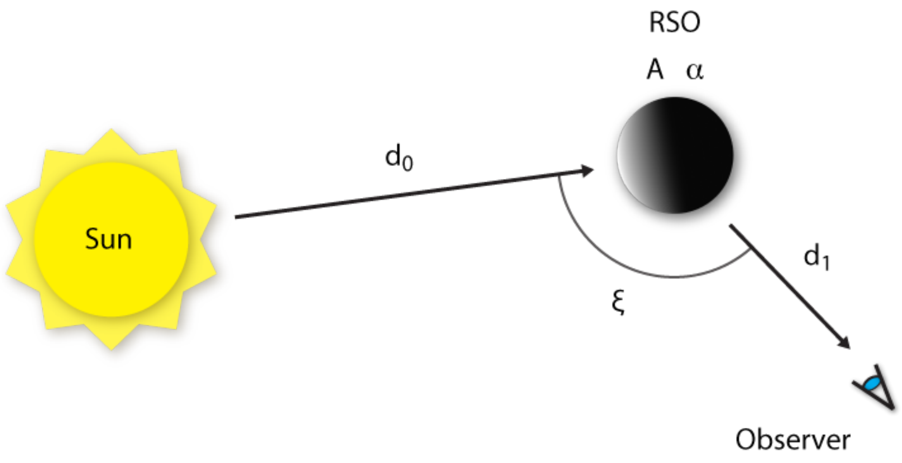


Fig. 1. A single sphere illuminated by the Sun.

means the Sun is fully occluded, and 1 means the Sun is not occluded at all. When factored into equation (3.1), the illumination equation becomes:

$$\frac{1}{4\pi d_0^2} \frac{Ap(\xi)\alpha}{2\pi d_1^2} \, shadow \tag{3.2}$$

This will give the percentage of light emitted from the Sun that bounces off of the sphere and reaches the observer.

III.1.2. A Set of Spheres Illuminated by the Sun

A set of Lambertian spheres illuminating each other is merely a general case of the single sphere case. Here, sphere 1 is illuminated by the Sun, and sphere i is illuminated only by sphere $i-1$. Let there be n number of spheres, where the n^{th} sphere is actually the observer[2] (Fig. 2). In this approach, sphere 1 is the only sphere illuminated by the Sun, so the sunlight is essentially being traced along the chain of spheres, at the end of which it finally reaches the observer.

- Let α_i, r_i, and $A = \pi r_i^2$ be the albedo, radius, and apparent area of sphere i, respectively.
- Let $\overrightarrow{d_0}$ and d_0 denote the vector and distance between the centers of the Sun and sphere i.
- Let $\overrightarrow{d_i}$ and d_i denote the vector and distance between the centers of sphere i and sphere $i+1$ for all $i = 1, 2, \dots, n-1$.
- Let $\overrightarrow{d_n}$ and d_n denote the vector and distance between the centers of sphere n and the observer.
- Finally, let ξ_i be the angle between $-\overrightarrow{d_{i-1}}$ and $\overrightarrow{d_i}$, for all $i = 1, 2, \dots, n$.

If we make the assumption that for all i, $d_i \gg r_i$, the illumination can be computed as follows:

$$\frac{1}{4\pi d_0^2} \prod_{i=1}^{n} \frac{A_i p(\xi_i) \alpha_i}{2\pi d_i^2} \tag{3.3}$$

This will give the percentage of light emitted from the Sun that bounces off of sphere 1, onto sphere 2, ..., onto sphere n, and finally arriving at the observer.

[2]Note that $n = 1$ is the single sphere case.

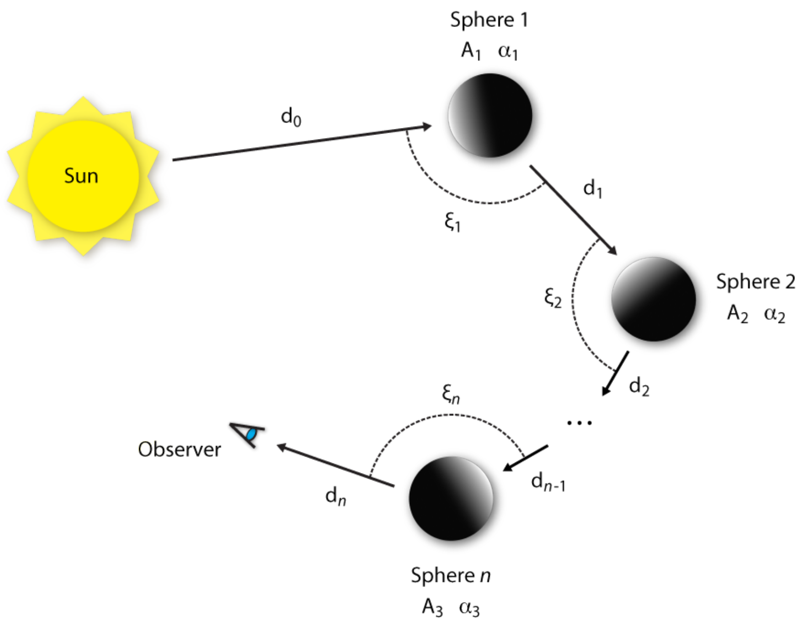


Fig. 2. A set of spheres illuminated by the Sun.

III.2. Analytical Model for Earthshine with No Atmosphere

In a simplified case of the multiple-sphere model with $n = 2$ spheres, earthshine can be computed analytically by taking the Earth as sphere 1 and the RSO as sphere 2 (Fig. 3). Then the illumination of the RSO due to the Earth is given by:

$$I_{Earth} = \frac{1}{4\pi d_0^2} \frac{A_1 p(\theta_1)\alpha_1}{2\pi d_1^2} \frac{A_2 p(\theta_2)\alpha_2}{2\pi d_2^2} \tag{3.4}$$

This function represents the percentage of total sunlight that bounces off of the Earth, then off of the RSO, and finally arrives at a point in space at the observer's location. However, since the Lambertian reflectance model has no capability to handle spheres

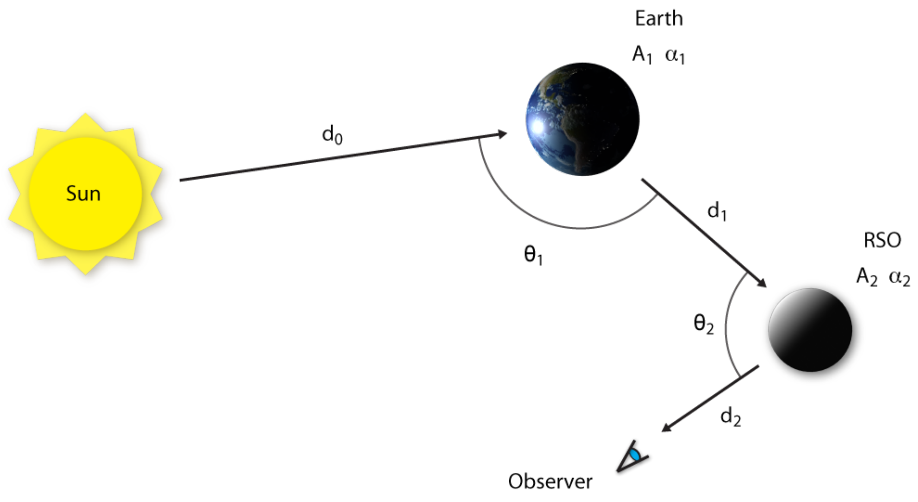


Fig. 3. A Lambertian sphere illuminated by earthshine.

surrounded by an atmosphere, a computational approach must be used to account for the refraction due to the Earth's atmosphere.

III.3. Computational Model for Earthshine with Atmospheric Refraction

Although the analytical model for earthshine is simple and quick to compute, objects orbiting the Earth are too close for it to be accurate, and there is no way to handle the effects of the atmosphere. The computational model, however, can handle both of these problems by employing ray tracing to perform photon mapping and final gathering. Photons will be cast from the Sun towards the Earth, where they will be refracted through the atmosphere and will ultimately be recorded in a photon map. Then, final gathering will be used to determine the amount of light diffusely reflected off of the Earth to specified points in space. Rays that pass directly through the atmosphere without touching the Earth are recorded directly in a photon map, since directional information is required to perform final gathering on these rays.

III.3.1. Assumptions

In order for the computational model to work, we must make a few assumptions about the Earth and its atmosphere. First, the Earth will be represented as a perfectly spherical Lambertian sphere with a uniform albedo. Although the Earth's albedo is constantly varying (e.g. cloud cover and snow fall), and the Earth's radius is not constant, these assumptions allow us to ignore the orientation of the Earth. This will reduce the problem space, which will become important when storing the earthshine radiance values.

Furthermore, the Earth's atmosphere will be represented as a set of nested spheres, each with a constant index of refraction. A discretized, layered approach will be used to trace rays through the atmosphere, so refraction will occur only at the boundaries of the spherical regions. Although the properties of the Earth's atmosphere vary with temperature, altitude, and fluctuating environmental conditions, these assumptions, once again, simplify the algorithm by allowing the orientation of the Earth to be ignored.

III.3.2. An Overview of the Algorithm

Thanks to the above assumptions, for any object at position P in space, the earthshine function will depend on only two variables (see Fig. 4):

1. the angle θ, defined as the angle between a line connecting the Sun to the Earth and a line connecting the Earth and P, and
2. the altitude h, defined as the distance from the Earth's surface to P.

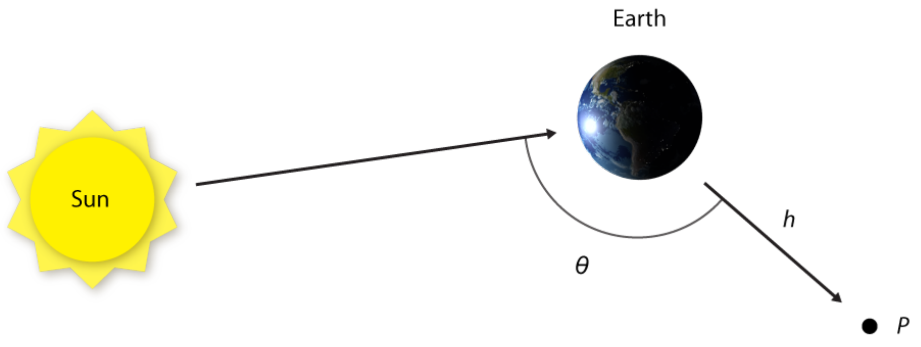


Fig. 4. A point in space illuminated by earthshine.

Then the illumination of point P due to earthshine is given by the function $I(\theta, h)$. This function can be represented as a texture file $T(u, v)$, where $0 \leq u < 1$ and

$0 \leq v < 1$. For a given pair (u, v), the actual angle and altitude can be computed as:

$$\theta = u\pi \quad \text{and} \quad h = (1 - v)h_{min} + vh_{max} \tag{3.5}$$

where h_{min} and h_{max} are the minimum and maximum altitudes that are defined. For instance, if the desired region is between LEO[3] and GEO[4], simply choose h_{min} as the minimum LEO altitude and h_{max} as the maximum GEO altitude.

Note that a texture file is essentially an image, therefore its quality is based on its resolution. Let the integer pair (M, N) denote the horizontal and vertical resolution of the image, and let a single pixel be denoted by the integer pair (i, j), located in the lower left corner of the pixel. Let $i = \lfloor uN \rfloor$ and $j = \lfloor vM \rfloor$. Finally, let $x = u - i$ and $y = v - j$. Using this information, a continuous function can be reconstructed using bilinear reconstruction. The value of $T(u, v)$ is computed from the texture file using bilinear reconstruction as follows:

$$\begin{aligned} T(u,v) \; &= (1-x)(1-y)\, I(i,j) \; + \; (1-x)y\, I(i,j+1) \\ &\quad + \; xy\, I(i+1,j+1) \; + \; x(1-y)\, I(i+1,j) \end{aligned} \tag{3.6}$$

Once we compute $I(\theta, h)$, this provides a way to compute accurate lighting infor-

[3]Low Earth Orbit

[4]Geostationary Orbit

mation for any (θ, h) pair from a discrete texture file. Although $I(\theta, h)$ cannot be written as a simple equation, the process for computing it will be described shortly. But we must first understand how the data structure will be represented.

III.3.3. The Structure of the Earthshine Table

Since the previously mentioned assumptions about the Earth allow its orientation to be ignored, the three-dimensional space around the Earth can be cleanly represented with the two-dimensional table $T(u, v)$, as shown in Fig. 5. To understand how this works, first imagine a single row from the table; across the row, the altitude, h_j, remains constant, and the phase angle varies. Now, in three-dimensions, imagine every point in space at an altitude of h_j, but that vary in their phase angles; all of these points form a spherical shell with radius $r_e + h_j$ that encloses the earth. So a single row in the table with altitude h_j corresponds to a spherical shell at altitude h_j in three-dimensional space.

Likewise, a single column in the table shares a common phase angle, but ranges in altitude from h_{min} to h_{max}. Imagine every point in three-dimensional space that satisfies these two conditions and the result takes on the form of thick disk. So each column of the table corresponds to a disk between h_{min} and h_{max} in three-dimensions.

The intersection of a row and a column in the table produces a single pixel. So in three-dimensions, a pixel corresponds to the intersection of a spherical shell at h_j and a thick disk at θ_i, which is a thin disc with radius $r_e + h_j$ at phase angle θ_i.

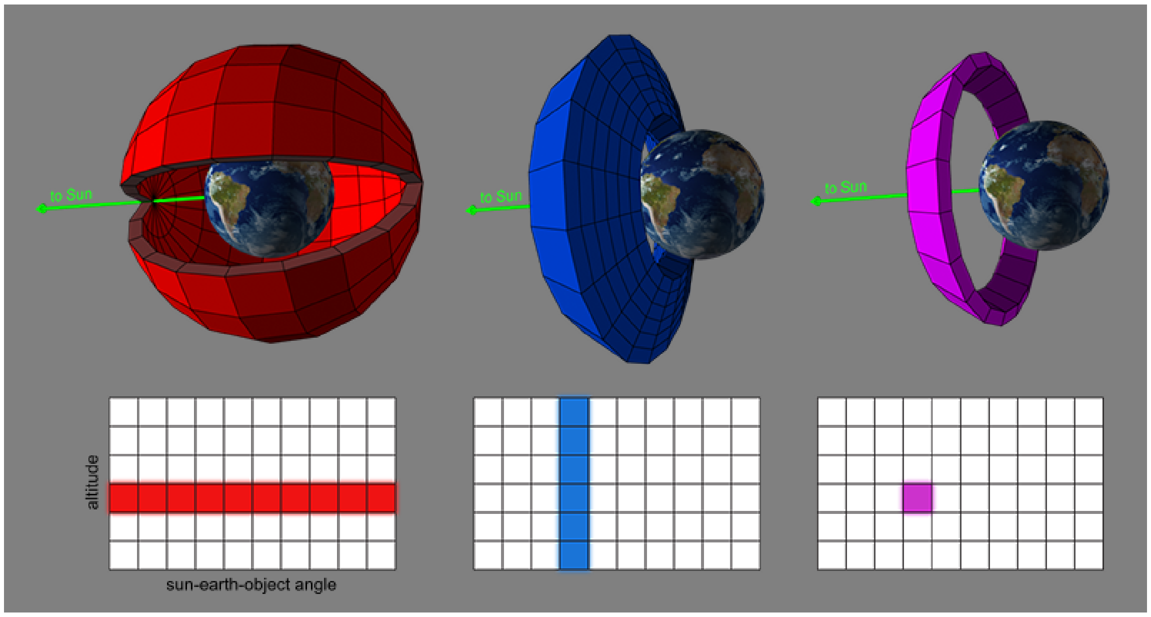


Fig. 5. The structure of the earthshine table.

III.3.4. The Process of the Algorithm

The values of the table are computed using a Monte Carlo algorithm that casts rays from the Sun towards the Earth, where each ray interacts with the Earth and atmosphere before being stored at pixel $I(i, j)$. The process of the algorithm is described below.

1. *Shoot a ray from the Sun towards the Earth.* Let C_e represent a circle sharing its center with the Earth and with a radius slightly larger than the outermost atmospheric shell. Similarly, let C_s represent a circle that shares it's center and radius with the Sun. Choose two random points p_e and p_s inside C_e and

C_s, respectively. The vector between these two points, $v_{se} = p_e - p_s$, gives the direction of a ray emanating from the Sun and striking either the Earth or atmosphere. For now, the ray carries unit radiance for all wavelengths.

2. *Intersect the ray with the Earth and/or any atmospheric shells it encounters.* At each intersection, reflect, refract, or scatter, all or some of the ray's radiance, based on the properties of the intersected object. Continue until the ray exits the outermost atmospheric shell.

3. *Store the ray's radiance in the photon map.* Rays that have been reflected off of the Earth are stored in a separate photon map from those that pass straight through the atmosphere so that the albedo of the Earth can be adjusted without recomputing the earthshine table.

4. *Go to (1) until the specified number of rays, K, have been cast.*

5. *Perform final gathering for each pixel in the earthshine table.* For each pair (θ_i, h_j), cast rays from the corresponding position p_f in space to a random point p_e inside the circle C_e, where C_e is oriented to point in the direction of θ_i. Sample the photon map in this way for several rays, and take their average to determine the amount of diffusely reflected light that reaches p_f. Store this radiance in pixel (i, j) of the earthshine table.

Since each ray carries a unit radiance, the results must be calibrated so that each carries the correct percentage of the Sun's total luminosity. Let I_{sun} denote the luminosity of the Sun, d_{se} denote the distance between the Earth and Sun, h_{atm}

denote the altitude of the outermost atmospheric shell, r_e denote the radius of the Earth, and $I_{unit}(i,j)$ denote the radiance value computed with rays that carry unit radiance. Finally, let $A_{atm} = \pi(r_e + h_{atm})^2$ be the apparent area of the outermost atmospheric shell from the Sun. Then each pixel in the table is calibrated with:

$$I_{total}(i,j) \;=\; \frac{I_{sun}\,A_{atm}}{2\pi d_{se}^2\,K}\,I_{unit}(i,j) \tag{3.7}$$

Note that this assumes that each ray carries the same amount of energy. Even though the rays are not parallel, the distance between the Earth and Sun makes the parallel ray assumption acceptable; thus, it can safely be assumed that each ray carries the same amount of energy.

Based on the representation of the photon map, each pixel represents the radiance that reaches a circular band on the surface of an atmospheric shell that is h_j above the Earth's surface. To compute the radiance per unit area, the total radiance should be divided by the area of this band. The radius of the circular band is given by $r_e + h_j$, where h_j is the altitude of the outermost shell, and the band is bounded by the angles $\theta_i = i\pi/N$ and $\theta_{i+1} = (i+1)\pi/N$. The total area between two angles θ_i and θ_{i+1} on the surface of a sphere with radius $r_e + h_j$ can be computed with the following integral:

$$\begin{aligned} A(i,j) &= \int_{\theta_i}^{\theta_{i+1}} \pi(r_e + h_j)^2 \, \sin\theta \, d\theta \\ &= (\cos\theta_{i+1} - \cos\theta_i) \, \pi(r_e + h_j)^2 \end{aligned} \tag{3.8}$$

Thus, the radiance per unit area is computed as:

$$I(i,j) = \frac{I_{total}(i,j)}{A(i,j)} \tag{3.9}$$

Now, $I(i,j)$ gives the percentage of total sunlight that bounces off of the Earth and reaches an RSO at a position θ_i degrees from the Sun-to-Earth axis and h_j units above the Earth's surface. This value can then be used to determine how bright an object at that position will appear to a camera or observer.

CHAPTER IV

IMPLEMENTATION

In this thesis, I developed two separate C++ applications to (1) compute the earthshine table and (2) display it's effects on objects in orbit around the Earth. The first application computes the earthshine table using photon mapping to cast Sun rays towards the Earth. The second application is a 3D visualizer that utilizes OpenGL, nested inside a Qt[1] window, to display a satellite with accurate lighting information based on its current position relative to the Sun, Earth, and Moon. The earthshine contribution comes from the table computed offline with the first application.

IV.1. Computing the Earthshine Table

To compute the earthshine table, I use a two stage process. First, photons are cast from the Sun towards the Earth, where they interact with the Earth's atmosphere until they exit the outermost shell and are stored in a photon map. Then, I use the diffuse photon map from the outermost atmospheric shell to do final gathering for each pixel of the earthshine table.

Before detailing the semantics of the algorithm, we must understand how light reflects off of a surface patch based on its orientation. In a simple Lambertian surface model, the amount of light reflected off of a flat surface patch toward the viewer is determined by the product of $cos\theta\, cos\phi$, where θ is the angle between the surface-

[1]Qt is a cross-platform API for creating graphical user interfaces that can utilize OpenGL functionality in a special widget.

light line and the surface normal, and ϕ is the angle between the surface-viewer line and the surface normal. Let the unit vector pointing from the surface patch to the light be $\overrightarrow{l}$, let the unit vector pointing from the surface patch to the viewer be $\overrightarrow{v}$, and let the unit surface normal be $\overrightarrow{n}$. Then θ is the angle between $\overrightarrow{l}$ and $\overrightarrow{n}$, and ϕ is the angle between $\overrightarrow{v}$ and $\overrightarrow{n}$. When the surface patch is facing toward the light such that $\overrightarrow{l} = \overrightarrow{n}$, the patch's surface area is fully exposed to the light and $cos\theta = 1$. As the patch rotates away from the light, it's surface area appears smaller from the light's perspective, so the patch receives less light as $\overrightarrow{l}$ and $\overrightarrow{n}$ become closer to perpendicular and $cos\theta$ approaches 0. A similar situation occurs with $cos\phi$, with the viewer in place of the light.

When casting light rays from the Sun towards the Earth, the Earth will receive fewer rays per unit of surface area as θ increases. If many rays are used, this effect replaces the need to multiply by $cos\theta$. Similarly, when casting final gathering rays from the world position corresponding to a pixel in the earthshine table towards the Earth, the area of a surface patch on the Earth will appear smaller to the incoming rays. By making this connection, we verify that the method coincides with the physical behavior of light, and we can describe the algorithm with confidence.

IV.1.1. Computing the Photon Map

To compute the diffuse photon map, a specified number of rays are cast from the Sun towards the Earth. Since the Earth and Sun are spherical, they can be represented by discs instead of full planes; this prevents rays from originating at or being cast to any sample that is outside of the object, reducing the number of intersection tests.

The Sun is represented by a disc shaped area light on a square plane with $N \times N$ samples, centered at the center of the Sun and facing the Earth. The target plane is represented by a disc on a square plane with $N \times N$ samples, centered at the center of the Earth and facing the Sun. Since a disc is used instead of the full square, only about $\pi N^2/4$ of the N^2 samples are used. A ray is cast from each Sun sample to each target sample, producing a total of approximately $\pi^2 N^4/16$ rays. The origin and target of the ray are jittered within each sample.

Since the earth is surrounded by a series of nested atmospheric shells, the first object that the ray will intersect is always the outermost shell. The ray's direction and intensity is adjusted based on the properties of that shell, then it is recursively cast in the new direction. Note that any given ray can only intersect the shell it just intersected or the objects immediately inside it and outside it (either another shell or the Earth). So based on this structure, the ray is only tested for intersections with these three objects, making the intersection calculation $O(1)$, constant with respect to the number of atmospheric shells.

A ray is recursively traced through the atmosphere until it hits the Earth, at which point it is diffusely reflected in a random direction. Randomly reflected diffuse rays are cast in such a way that they are evenly distributed on the surface of a sphere centered at the intersection point. The reflected rays are traced back out through the atmosphere until they encounter the outermost atmospheric shell.

When a ray exits the outermost shell, it needs to be stored in the photon map. Note that when the photon map is computed with no atmosphere, it will be stored on the surface of the Earth; therefore, to generalize, the photon map is always stored on the outermost object. Let P be the intersection position of the ray on the outermost object's surface, let $\overrightarrow{n}$ be the surface normal of the object at that position, and let $\overrightarrow{d_0}$ denote the vector pointing from the center of the Earth to the center of the Sun. Since the Earth and atmosphere are assumed to be perfectly spherical and uniform across the surface, P can be indexed in the photon map simply by the angle, θ, between $\overrightarrow{n}$ and $\overrightarrow{d_0}$. As a texture file, this can be represented by an image that is one pixel high and w pixels wide, where w simply relates to the desired precision of the photon map.

IV.1.2. Final Gathering to Determine Illumination for the Table

Now that we know the distribution of sunlight reaching the Earth, we can use the photon map to do final gathering and determine the illumination at any given point in space. This will be done for each pixel in the earthshine table in order to compute the image. Remember that each pixel is represented by a pair (θ, h), and that the world position of each pixel can be extracted from this information.

Let the earthshine table be U pixels wide and V pixels high. To loop through all of the pixels, let u be the current index along the width and v be the current index along the height, where $0 \leq u < U$ and $0 \leq v < V$. Finally, let h_{min} and h_{max} be the minimum and maximum altitudes defined by the table. Then, the pair (θ, h) can be computed as follows:

$$\theta = u\,\frac{\pi}{U-1}$$
$$h = h_{min} + v\,\frac{h_{max}-h_{min}}{V-1} \tag{4.1}$$

From (θ, h), the world position can now be identified and used as the origin of the final gathering rays.

Let P be the position of the current pixel in world space, corresponding to (θ, h). Also, let $\overrightarrow{d_S}$ be a unit vector pointing from the center of the Earth to the center of the Sun, and let $\overrightarrow{d_{S\perp}}$ be a unit vector perpendicular to $\overrightarrow{d_S}$. Create a vector $\overrightarrow{d_P}$, pointing from the center of the Earth to P, that forms the angle θ with $\overrightarrow{d_S}$, where

$$\overrightarrow{d_P} = \overrightarrow{d_{S\perp}}\sin\theta + \overrightarrow{d_S}\cos\theta \tag{4.2}$$

Now, let C_E and r_E be the center and radius of the Earth, respectively. Then, P can be computed as

$$P = C_E + \overrightarrow{d_P}\,(r_E + h) \tag{4.3}$$

Since P represents the position of the current pixel of the table in world space, this is where all of the final gathering rays for this pixel will originate.

To determine the target position for the final gathering rays, a target plane with the specified number of final gathering samples must be created. Let $\hat{x}$ and $\hat{y}$ be perpendicular unit vectors that form a plane perpendicular to $\overrightarrow{d_P}$, such that $\hat{x}$, $\hat{y}$, and $\overrightarrow{d_P}$ are orthonormal vectors. Then, the plane defined by $\hat{x}$ and $\hat{y}$ will serve as the target plane, through which the final gathering rays will be cast.

The plane is centered at the center of the Earth, and we will use a target disc to sample the photon map on the outermost object. The size of this disc is dependent on the the distance that P is from the object. For positions closer to the object, the field of view must be larger than for positions far from the object. So the radius of the target disc must increase as P becomes closer to the object.

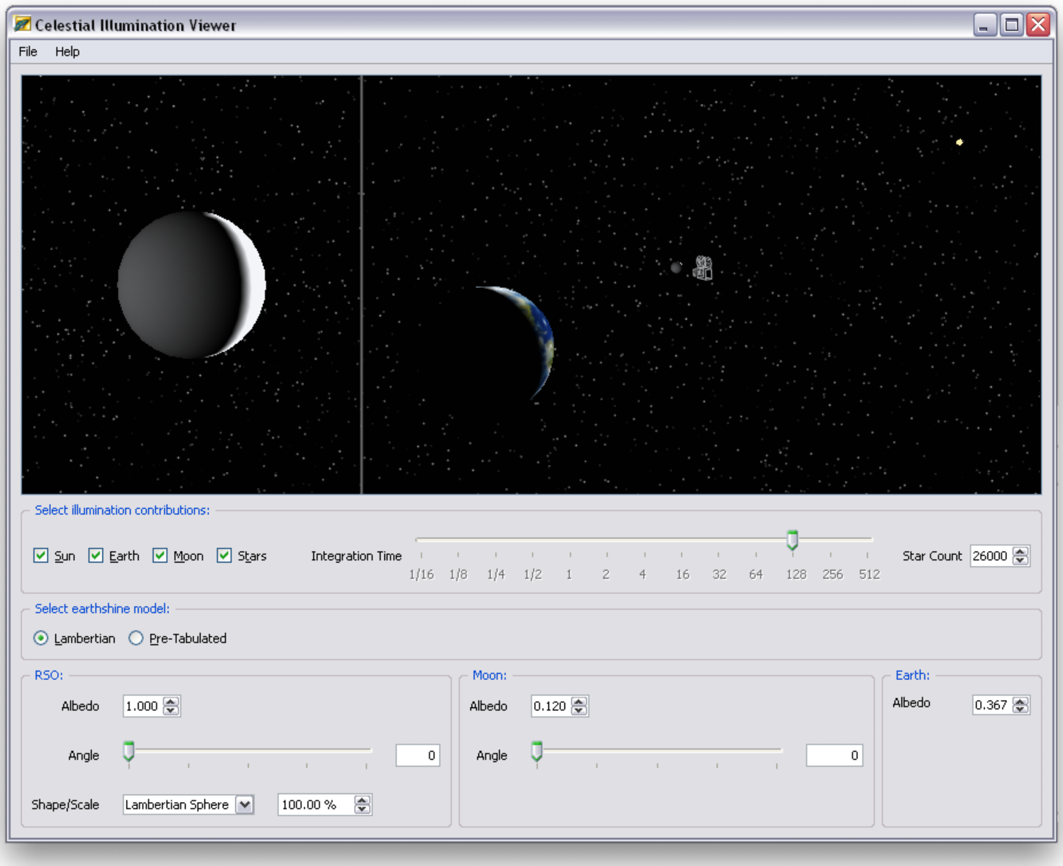
Now, a ray is cast from P to a jittered position within each of the disc's roughly $\pi N^2/4$ samples. When it intersects the outermost object, the coordinating color from the photon map is found[2]. The intensities from all intersecting final gatherings averaged and attenuated for distance falloff by dividing by $4\pi h^2$, where h is the altitude of P. This gives the color in the earthshine table at pixel (u, v).

IV.2. 3D Visualization Tool

In order to visualize the computed earthshine affecting real objects, I implemented a C++ application to display a scene using the earthshine table to supplement the lighting (see Fig. 6 for a screenshot). Qt was used for the windowing, and OpenGL handles the interactive visualization viewport. The Sun, Earth, Moon, and an RSO are all displayed in a 3D scene against a background of stars, and lights are placed at each of the primary sources of illumination: the Sun, Earth and Moon. The user

[2]See Section IV.1.1 on page 29 for an explanation of indexing the photon map.

can interactively move around the scene and change the position of the Moon and RSO, which automatically adjusts the lighting.



Select illumination contributions:

[✓] Sun [✓] Earth [✓] Moon [✓] Stars

Star Count 26000

Select earthshine model:

[✓ Lambertian] Pre-Tabulated

RSO:

Albedo 1.000

Angle 0

Shape/Scale Lambertian Sphere 100.00 %

Moon:

Albedo 0.120

Angle 0

Earth:

Albedo 0.367

Fig. 6. A screenshot from the 3D visualization tool.

IV.2.1. Interactive OpenGL Display

The OpenGL window consists of two viewports: an RSO centric viewport on the left and an independent scene viewport on the right. When the RSO is moved around its orbit, the RSO centric camera stays fixed on the RSO. This camera can only zoom and rotate around the RSO, which always stays in the center of the view. On the other hand, the main viewport camera, can be moved freely around the scene and is not tied to any of the objects. A wireframe representation of the RSO centric camera is visible in the main viewport for reference. Both cameras can be moved around in the same way as the camera in Maya, with the exception that panning is not allowed in the RSO centric viewport.

Four main pieces of geometry are displayed in the two viewports. The Sun, Earth, and Moon are all drawn as spheres made of polygonal triangles, where the polygon count is lower for the Sun and Moon than it is for the Earth. The RSO's shape is read from an external OBJ file; initially a sphere is read in, but several other files can be chosen via a drop down box. The Earth and Moon are textured with external PNG images, while the RSO color is plain white.

A pseudo star field is generated randomly around the scene, of which the density can be adjusted interactively. The stars are stored in a list that contains the position, intensity, and visibility of each star, and they are rendered as points based on this information. Although the star field is completely fabricated, it gives a point of reference when moving the camera and helps add depth to the scene.

In space, all of the natural light originates from the Sun and is reflected off of planets, satellites, etc. To simulate this in the interactive environment, I place a light source not only at the Sun, but at each major reflector as well (in this case, the Earth and Moon). Since objects are at different distances from a specific light source, the intensity of the light that reaches each object is different. Instead of using OpenGL lights with distance attenuation to control the falloff of the intensity, I use a separate light with no falloff at each source to illuminate each object. For example, there are two lights at the Moon: one to illuminate the RSO and one to illuminate the Earth. Each has the appropriate intensity to illuminate it's target object at that distance from the source. This gives me complete control over the amount of falloff and its coefficients, since it must be physically accurate. There is also a small ambient light in the scene representing the light coming in all directions from the many stars in the universe.

The albedo of an object in the scene is a scalar representing the percentage of incoming light that is reflected off of the object's surface. This has two effects on the visual output: (1) if the object is a reflector, the intensity of the lights located at the object are scaled by the albedo, and (2) the amount of light received by the object is scaled by the albedo. So as a reflector's albedo is lowered, less and less light will reach the objects that it illuminates, and it will become dimmer in the viewport.

The light at the Earth that illuminates the RSO (referred to in this thesis as earthshine) can be determined from either the computational earthshine model described in Section III.3 or the analytical model for earthshine from Section III.2. The user is offered this option as a way to compare the two models, which provide a different illumination distribution for objects close to the Earth. Since all other objects are significantly far from each other, the intensities of the other light sources are determined from the Lambertian sphere reflectance model, discussed in Section III.1.

IV.2.2. Qt Graphical User Interface

Three sections are available at the bottom of the window to adjust the albedo, position, and shape of the RSO, Moon, and Earth. Each object has a spinbox to change the corresponding albedo. The RSO and Moon can be rotated around the Earth in a circular orbit using either the Angle slider or the text box next to it. Furthermore, there is a drop down box to change the shape of the RSO by reading in a new OBJ file, and the scale of the RSO can be changed with the available spinbox.

A default number of stars are generated when the program is first executed, but there is a spinbox to adjust the star count interactively. When the star count is increased from the default, new stars are randomly generated and added to the list of existing stars. If the count is decreased, stars from the list are hidden, and when the count is again increased, the previously generated stars are made visible.

In OpenGL, 1.0 is the full intensity of a light, and anything above that has the possibility of blowing out the lighting on an object. So in order to achieve the best visual results, the largest illumination source, the Sun, is initially scaled to have an intensity of 1.0. Using the illumination contribution checkboxes, each source can be toggled on or off, rescaling the intensities such that the maximum is 1.0 and scaling the other lights to maintain the same relative intensities. So, for example, when the Sun is toggled off, the Earth becomes the largest illumination source and is scaled accordingly to 1.0. This allows the user to single out one particular source, if so desired.

Another option to manipulate the observed intensity of light in the scene is to use the Integration Time slider. Moving the slider to the right increases the intensity of the lights by a factor of two for each notch, while moving the slider to the left decreases the intensities by a factor of two. This simulates an exposure setting on a camera. With a longer integration time (or shutter speed), the film is exposed to more light and the pictures become brighter, but a shorter integration time produces dimmer pictures. By moving the slider, it is possible to capture more light on the virtual film and see the effects of lower intensity light sources.

The earthshine contribution can either be based on the computational or analytical model. Selecting the Lambertian radio button uses the analytical model to determine the earthshine intensity, while the Pre-Tabulated radio button uses the computational model. Both models are applied in real-time, since the computational model just requires a lookup in the earthshine table.

CHAPTER V

RESULTS

The objective of the computational model for earthshine is to create a table representing the illumination of a discrete region around the Earth. Remember from Section III.3.3 that the horizontal axis represents the sun-earth-object angle from 0 on the left to π on the right, and the vertical axis represents the altitude from h_{min} at the bottom to h_{max} at the top. The result of the computed earthshine table is shown in Fig. 7.

The table is represented visually and internally to the algorithm as two separate images: one for diffuse reflections off of the Earth's surface, and one for refracted rays that never touch the Earth. The two images are kept separate because the diffuse reflections are influenced by the albedo of the Earth, which can be varied in the 3D visualizer application. Albedo has no effect on refracted rays that never touch the Earth's surface, so they are stored and visualized in a separate block of data.

The top image represents the table for diffuse reflections off of the surface of the Earth. Note that as the altitude increases along the vertical axis, the values decrease due to the quadratic falloff of intensity with distance. Also note that as the angle increases along the horizontal axis, the values decrease because they represent positions moving toward the back side of the Earth in relation to the Sun.

In the bottom image is the table representing refraction of rays through the atmosphere that don't touch the Earth's surface. The edges of the illuminated region are jagged because this file represents the photon map without a final gathering pass.

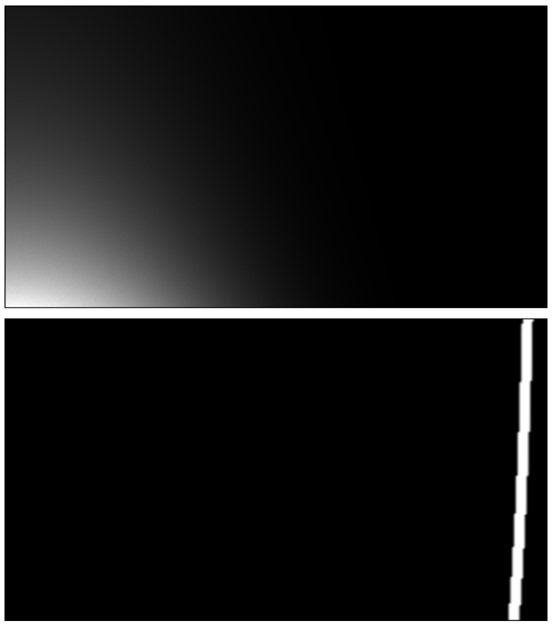


Fig. 7. The earthshine table for diffuse reflections (top) and refraction (bottom).

Since refracted rays illuminate only along the direction of their path (unlike diffuse reflections which radiate uniformly in all directions), directional information must be stored in the photon map in order to perform final gathering. However, this is left as a task for future work.

An ASCII version of each table is stored on disk, along side the images, which is used to initialize the data in the visualizer. These data files are the true representation of the earthshine tables. It is important to store the data in this form to maintain its full precision, since the numbers must be compressed to create a corresponding image that fits nicely in the displayable color range.

To compare the results with the analytical model for earthshine, I extract the

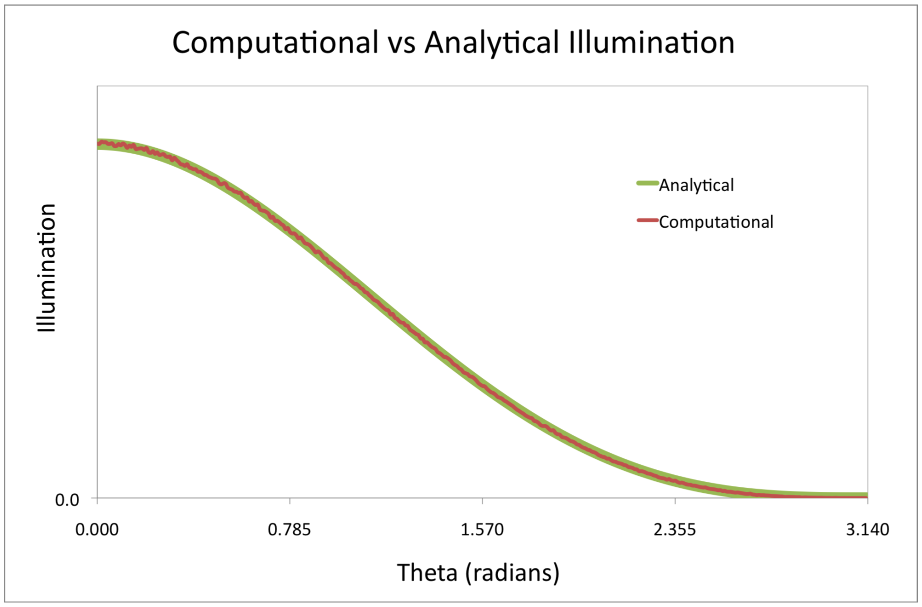


Fig. 8. A comparison of the analytical and computational earthshine model with no refraction.

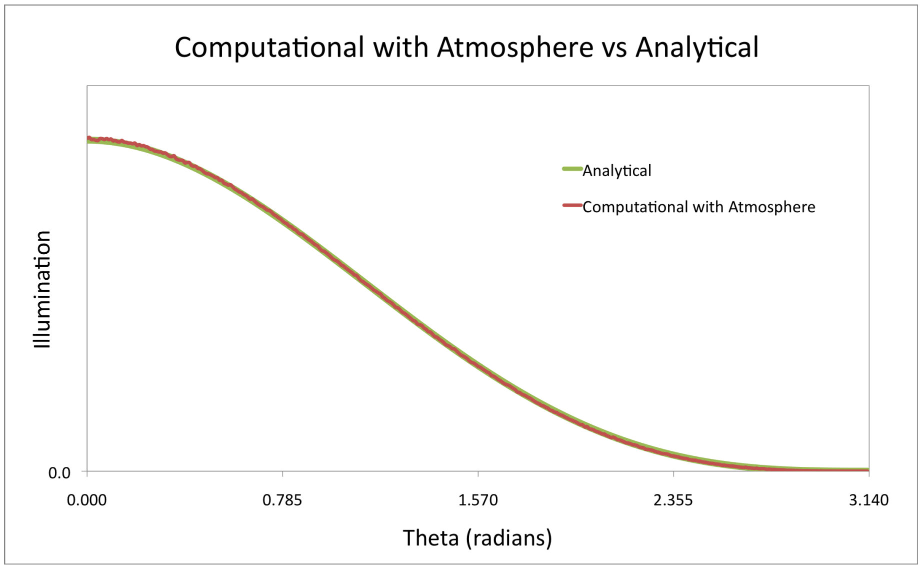


Fig. 9. A comparison of the analytical and computational earthshine model with refraction.

values from a row of the diffuse earthshine table. A row varies only with the phase angle, and if the altitude in the analytical equation is fixed, it too varies only with the phase angle. In this manner, the earthshine values from the computational and analytical models can be charted from 0 to π for equal altitudes. A chart of this form is given in Fig. 8 for the computational model with no atmosphere, and in Fig. 9 for the computational model with atmosphere.

The curves match nicely for the case with no atmosphere, aside from the slight noise in the computational curve. And in the case with atmosphere, the computational curve is only slightly higher before $\pi/2$ and slightly lower from there until π. This means that refraction due to the atmosphere has a very small affect on the diffusely reflected light; however, refracted light that doesn't hit the Earth's surface does have a considerable affect of reducing the size of the shadow cast by the Earth.

Since the index of refraction is higher in the atmosphere than it is in space, rays are refracted towards the Earth as they pass through the atmosphere. Some of these rays illuminate a small region behind the Earth that would be in shadow if the atmosphere was not there. Fig. 10 shows this effect.

The outer edge of the penumbra stays in about the same place, but the inner edge is nearly $3°$ further into the umbra. With the penumbra being about $2.5°$ larger and the umbra being smaller, this increases the region of space that is illuminated by direct sunlight, offering another advantage of the computational model over the analytical model.

One thing to note about Figures 8 and 9 is that the altitude used in these graphs is nearly 400 times the Earth's radius. Since the analytical model is only accurate

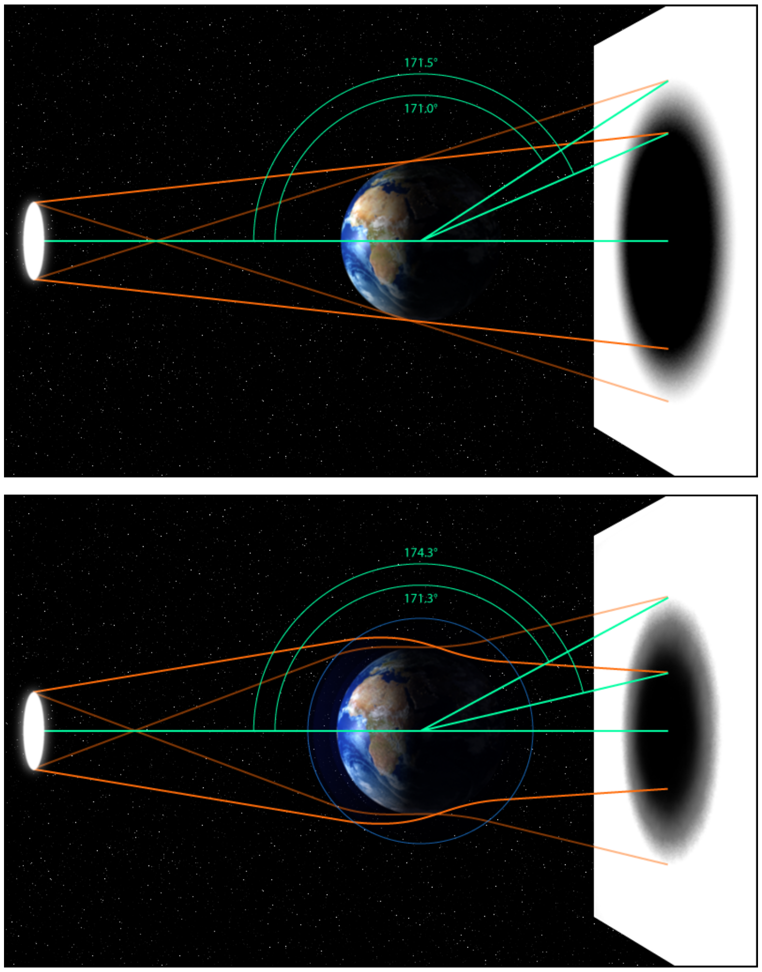


Fig. 10. The shadow of the Earth with (bottom) and without (top) atmosphere.

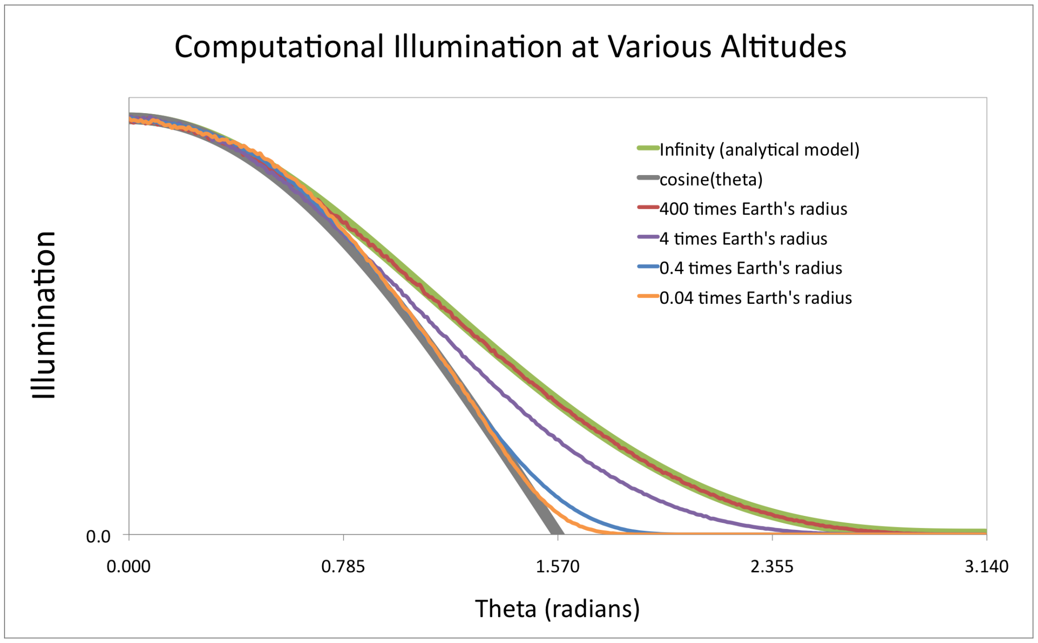


Fig. 11. A comparison of various altitudes from the computational earthshine table.

when the distance between the illuminated object and the Earth is much larger than the Earth's radius, the shape of the computational and analytical models will only match at large altitudes. However, objects orbiting the Earth are much closer, anywhere from less than a tenth to just over six times the radius of the Earth. Fig. 11 shows a row from the earthshine table (with no atmosphere) at four different altitudes, ignoring the distance falloff so we can compare the shape of the curves instead of the relative intensities.

Notice that as the altitude approaches zero, the curve becomes closer to the cosine curve between 0 and $\pi/2$. This is because the cosine curve represents the value of the photon map at a given phase angle (when no atmosphere is present).

The intensity of the light that reaches a portion of the surface is proportional to the cosine of the angle the incoming light forms with the surface normal. So, when the altitude approaches zero, the position is closer and closer to the surface of the Earth, and a more restricted local area is all that contributes to the illumination.

CHAPTER VI

CONCLUSIONS

When a satellite is attempting to photograph an object in orbit, it needs to know the lighting conditions so it can position itself and adjust camera settings to take the best photograph. A satellite that has an accurate method to determine how a subject is lit will be more capable of capturing good pictures than one with a less accurate method. Based on the algorithm presented in this thesis, we have a physically accurate method for computing the illumination of objects in orbit that is more accurate than previous methods.

This model utilizes photon mapping and takes into account diffuse reflection from the Earth's surface and refraction in the atmosphere to compute a table representing the Earth's illumination contribution at a desired point in space; this table can then be used as lighting input when visualizing an illuminated object. A satellite attempting to take pictures with this information at its disposal will be able to make the most informed decision possible.

Knowing how atmospheric refraction reduces the size of the Earth's shadow is also important, as it increases the area that a picture can be taken. This effect is by far the most important one that refraction has on earthshine, since its effect on the diffusely reflected light was not significant. However, until atmospheric scattering is taken into account, no conclusion can be made on the atmosphere's effect as a whole on the distribution of earthshine.

CHAPTER VII

FUTURE WORK

VII.1. Computational Model for Earthshine with Atmospheric Refraction and Scattering

The Earth's atmosphere has two major effects on incoming light: refraction and scattering. However, the computational model described in this thesis only considers refraction. By adding scattering effects, it will be possible to get a more accurate approximation to the actual illumination that would be observed in space. Furthermore, Rayleigh scattering, the main scattering that would be of concern, scatters light differently based on its wavelength. This has the effect of scattering blue light more than red light, so an incoming sun ray would transfer more red light through the atmosphere, while scattering the blue in all directions; this is the reason that the sky is blue and sunsets are orange and red.

In order to compute scattering, a few modifications must be made to the algorithm:

1. Each atmospheric shell will be composed of a number of spheres, called photon spheres. Each photon sphere is composed of segments that represent the direction and intensity of rays that pass through the sphere's center.

2. During the computation of the photon map on the surface of the Earth, rays must be scattered each time they hit an atmospheric shell.

3. Between the photon mapping step and the final gathering step to compute the earthshine table, a step must be added to iteratively recompute the photon map as seen by each atmospheric shell.

First, each atmospheric shell will be made up of K photon spheres. Think of the spheres as being divided into polygonal segments of equal area, where the color of each segment denotes the color of a ray whose direction is equal to that of the surface normal of the segment.

Now, as photons are cast from the sun to compute the photon map, each photon will be recorded in a photon sphere on each atmospheric shell. When the photon intersects a shell, the color of the photon will first be scattered based on a Rayleigh scattering distribution and recorded on the photon sphere. Then, the remaining photon intensity will be recorded on the photon sphere in the direction that the photon is refracted. The photon is then cast in the refracted direction with the intensity of the non-scattered light. This process is repeated at each atmospheric shell until the photon finally hits the Earth and is recorded in the photon map.

Next, an additional final gathering step is performed, before computing the earthshine table, to capture the behavior of photons as they are diffusely reflected off the surface of the Earth. The existing photon spheres will be supplemented to record the direction and color of the reflected photons.

For the first (innermost) atmospheric shell, the colors of the segments on photon sphere k are adjusted by casting rays from the center of the sphere towards the Earth. The position that a ray intersects the surface of the Earth is used to look up the intensity in the photon map at that position. This intensity is attenuated based on the distance of the ray and added to the color of the segment with a surface normal equal to the negative of the ray's direction. After the specified number of rays have been cast from sphere k, the process is repeated with sphere $k+1$, until all K spheres on the innermost atmospheric shell have been touched.

For the remaining shells, the process is slightly different because rays will intersect the shell immediately inside the current shell, instead of the Earth. Let i be the current shell, and let $i-1$ be the shell immediately inside it. Like above, a number of rays are cast from the center of sphere k on shell i towards shell $i-1$. When a ray intersects shell $i-1$, it first determines the closest photon sphere. Then, it determines the color of the segment that has a surface normal equal to the negative of the ray's direction. This color is attenuated based on the length of the ray, then added to sphere k on shell i in the segment with a surface normal equal to the negative of the ray's direction. This process is repeated for each photon sphere on shell i.

Once final gathering has been done for all of the photon spheres on shell i, the process is repeated for shell $i+1$, until the outermost shell is complete. Then, this shell, instead of the Earth's photon map, can be used in the final gathering step to compute the illumination for the earthshine table at any position around the Earth.

VII.2. Nested Atmospheric Shells

A related area that would require more research involves the nested shell approach to representing the atmosphere. Ideally, one would like to represent the atmosphere as a continuous volume, rather than approximate it with a series of shells. Since the assumption is made that the atmosphere is uniform and varies only with altitude, that may offer a structure that can be used to develop a method that more closely approximates a continuous volume. However, without an existing continuous model with which to compare, it's difficult to know how significant the results would ultimately be. It is also difficult to compare with experimental data since the properties of the atmosphere are constantly varying with changing environmental conditions, temperature, etc., contrary to the assumption made in this thesis.

VII.3. Illumination of Extended Objects

One assumption that was made in order to simplify the computational earthshine algorithm was to store one radiance value in the earthshine table for each position in space. This is essentially equivalent to approximating the Earth as a point light source, when in actuality, it is an area light source. For objects that are small with respect to the size of the Earth, this is a reasonable approach; however, when considering larger extended objects, the direction and origin of incoming light also become important. For significantly small objects at a significant distance from Earth, the error is probably small, but for larger objects in orbits closer to the Earth, it may be important to store direction information in addition to intensities in order to get accurate lighting information.

One approach to lighting an extended object would be to use the value in the earthshine table as the intensity of an area light, and then do ray tracing to compute the lighting across the surface of the object. This would give a soft gradation of light, and would allow shadows to be computed on the object. However, the light source itself, Earth, does not radiate light evenly across it's surface, so the lighting would not be completely accurate.

Another idea is to record the distribution of reflected light intensity across the surface of the Earth, instead of just recording the average intensity. This would require significant changes to the structure of the earthshine table to the point that it couldn't actually be represented by a two-dimensional image. For each pixel in the table, there would instead need to be some set of data that described both the intensity and direction of light coming from any desired point on the Earth's surface. However, by not being able to use the average reflected intensity that reaches a given point, the final gathering step would become much more complicated, and the result could be much noisier.

VII.4. Considering Variable Surface Properties of Earth

Another assumption was made that the Earth is a uniform sphere with a uniform albedo, when in actuality, both vary greatly over the surface of the Earth. Clouds and snow have a high albedo, while some land masses have a relatively low albedo. Currently, the best way to handle these varying surface properties is to use the average albedo of the Earth on the side that faces the illuminated object at the desired instant in time. The average albedo can be computed relatively easily from satellite imagery of the Earth, but this may not be accurate enough, especially when considering extended objects close to the Earth.

The most obvious way to account for the Earth's varying surface properties would be to determine the albedo of the Earth each time a photon hits the surface, and attenuate its intensity accordingly. However, this would require that the earthshine table be a three-dimensional representation, since the symmetry around the axis pointing from the Earth to the Sun no longer exists (which is what allowed all of the positions around Earth to be represented simply by the pair (θ, h)). Furthermore, the Earth is not always in the same orientation with respect to the Sun; the Earth rotates and orbits around the Sun, constantly causing a different portion of the surface to be illuminated. So the earthshine table would need to be calculated on a case-by-case basis for each object that the illumination is desired, which is unreasonable with the algorithm's lengthy computation times.

VII.5. Improvements in Visualizer

Several improvements can be made to the 3D visualizer to improve its visual appearance and add desirable functionality. Although the application currently serves the core purpose of displaying the illumination of an object orbiting the Earth, it could become much more versatile and polished with a few additional features.

One such improvement is to have camera parameters, such as aperture, shutter speed, field of view, noise level, etc. be input by the user, filtering the OpenGL viewport through those parameters accordingly. Currently, the integration time can be adjusted, simulating the shutter speed on a real camera. The difference is that the integration time uses a somewhat arbitrary scale, where as most cameras have a specific set of shutter speeds available. Also, the effect of a specific integration time currently changes relative to the most intense light source, where as shutter speeds are absolute. Limiting these type of parameters to real world constraints will not only make the results look more like an actual picture of the scene, but it will also help determine when an object can actually be seen by a physical camera.

Another feature that would be useful in the visualizer would be high dynamic range (HDR) visualization capabilities. Even the most advanced cameras can't capture the range of light that the human eye is able to perceive, since it can adjust to various lighting conditions. Tone mapping algorithms can be used to make HDR images visible on a conventional computer monitor, so a similar method could be used to compress the huge range of light intensities in space to a displayable range. This would make the effects of smaller illumination contributions more noticeable than they would be otherwise, and would more closely resemble what a human might see if he or she were to observe the scene in person.

In order to make the physical layout more accurate, the true orbits of each celestial body could be used, as well as the true positions of stars. Currently, the objects are placed somewhat arbitrarily (although the distances between them are accurate), and the star positions are randomly generated. By entering a date or providing a slider to scrub through time, the visualizer could use the orbital information of each celestial body to place it at the exact position it would be on that particular date. Additionally, the positions and intensities of the visible stars are documented in star catalogs, so by incorporating that data, the star field generated in the viewport would be accurate to real life, and constellations would be discernible. These features could serve important when planning a satellite's course by maximizing the encounter time under good lighting conditions or minimizing the occurrences where the Sun is in the frame, for example.

Finally, the shading of the objects, especially the RSO and Earth could be significantly improved to provide a more visually pleasing result. Material properties could be read from the RSO's OBJ file and applied to the model in the viewport; simply by adding a specular component and texturing, the currently Lambertian shading would look much more realistic. More work could also be done to add a specular component to the Earth, and although it wouldn't be completely realistic, adding city lights to the night side of the Earth would be a nice addition.

REFERENCES

[1] A. Appel. Some techniques for shading machine renderings of solids. In *Proc. of AFIPS Spring Joint Computer Conference*, pages 37–45, May 1968.

[2] J. Arvo. Backward ray tracing. In *Developments in Ray Tracing, Siggraph Course Notes*, pages 259–263, August 1986.

[3] M. Berger, T. Trout, and N. Levit. Ray tracing mirages. *IEEE Computer Graphics and Applications*, 10(3):36–41, 1990.

[4] E. Bruneton and F. Neyret. Precomputed atmospheric scattering. *Computer Graphics Forum*, 27(4):1079–1086, 2008.

[5] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, 1984.

[6] Y. Dobashi, T. Yamamoto, and T. Nishita. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proc. of ACM Siggraph/Eurographics Conference on Graphics Hardware*, pages 99–107, September 2002.

[7] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):213–222, 1984.

[8] D. Gutierrez, F. J. Seron, A. Munoz, and O. Anson. Simulation of atmospheric phenomena. *Computers & Graphics*, 30(6):994–1010, 2006.

[9] J. Irwin. Full-spectral rendering of the earth's atmosphere using a physical model of rayleigh scattering. In *Proc. of Eurographics UK Conference*, pages 103–115, June 1996.

[10] D. Jackèl and B. Walter. Modeling and rendering of the atmosphere using mie-scattering. *Computer Graphics Forum*, 16(4):201–210, 1997.

[11] H. W. Jensen. Global illumination using photon maps. In *Proc. of Eurographics Workshop on Rendering*, pages 21–30, June 1996.

[12] H. W. Jensen and N. J. Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[13] J. T. Kajiya and B. P. V. Herzen. Ray tracing volume densities. *Computer Graphics*, 18(3):165–174, 1984.

[14] K. Kaneda, T. Okamoto, E. Nakamae, and T. Nishita. Photorealistic image synthesis for outdoor scenery under various atmospheric conditions. *Visual Computer*, 7(5):247–258, 1991.

[15] M. Kerker. *The Scattering of Light and Other Electromagnetic Radiation.* Academic Press, 1969.

[16] R. V. Klassen. Modeling the effect of the atmosphere on light. *ACM Transactions on Graphics*, 6(3):215–237, 1987.

[17] W. H. Lehn. A simple parabolic model for optics of the atmospheric surface layer. *Applied Mathematical Modeling*, 9(6):447–453, 1985.

[18] A. Lintu, J. Haber, and M. Magnor. Realistic solar disc rendering. In *Proc. of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 79–86, February 2005.

[19] D. Lischinski, F. Tampieri, and D. P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Proc. of Siggraph '93*, pages 199–208, August 1993.

[20] T. Nishita, Y. Dobashi, K. Kaneda, and H. Yamashita. Display method of the sky color taking into account multiple scattering. In *Proc. of Pacific Graphics*, pages 117–132, August 1996.

[21] T. Nishita, T. Sirai, K. Tadamura, and E. Nakamae. Display of the earth taking into account atmospheric scattering. In *Proc. of Siggraph '93*, pages 175–182, August 1993.

[22] M. C. Reichert. *A two-pass radiosity method driven by lights and viewer position.* Master's thesis, Cornell University, 1992.

[23] K. Riley, D. S. Ebert, M. Kraus, J. Tessendork, and C. Hansen. Efficient rendering of atmospheric phenomena. In *Proc. of Eurographics Symposium on Rendering*, pages 375–386, June 2004.

[24] F. J. Seron, D. Gutierrez, G. Gutierrez, and E. Cerezo. Implementation of a method of curved ray tracing for inhomogeneous atmospheres. *Computers & Graphics*, 29(1):95–108, 2005.

[25] P. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proc. of Graphics Interface*, pages 205–212, June 1990.

[26] J. Sloup. A survey of the modelling and rendering of the earth's atmosphere. In *Proc. of Spring Conference on Computer Graphics*, pages 141–150, April 2002.

[27] B. E. Smits. *Efficient hierarchical radiosity in complex environments.* PhD thesis, Cornell University, 1994.

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

VITA

Jonathan Penney
C108 Langford Center
3137 TAMU
College Station, TX 77843-3137
jpenney@viz.tamu.edu

Education

M.S. in Visualization Sciences	Texas A&M University, May 2009
B.S. in Computer Science	Texas A&M University, May 2006