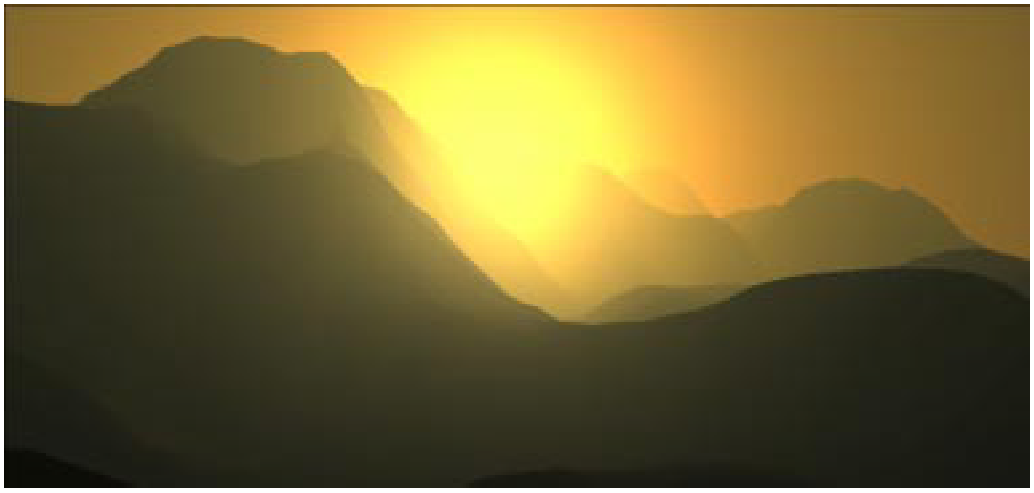
Light and Color in the Outdoors

A SIGGRAPH 2003 Course



Course Organizer

Simon Premože
University of Utah

Lecturers

Mark J. Harris
University of North Carolina at Chapel Hill

Nathaniel Hoffman
Naughty Dog

AJ Preetham
ATI Research

Simon Premože
University of Utah

Light and Color in the Outdoors

A SIGGRAPH 2003 Course

Time and Day

Sunday, July 27, 2003 from 8:30 AM to 12:15 PM

Course Organizer

Simon Premože
Computer Science Department
University of Utah
50 S Central Campus Dr. Rm 3190
Salt Lake City, UT 84112
email: `premoze@cs.utah.edu`

Presenters Contact Information

Mark J Harris
University of North Carolina at Chapel Hill
`harrism@cs.unc.edu`

Nathaniel Hoffman
Naughty Dog
`naty@io.com`

AJ Preetham
ATI Research
`preetham@ati.com`

Simon Premože
University of Utah
`premoze@cs.utah.edu`

Course Level

Advanced

Abstract

Simple and practical methods for daytime and night-time skylight illumination and the appearance of clouds, including approximate and practical methods for outdoor global illumination. Special focus: practical methods appropriate for real-time applications.

Prerequisites

A good understanding of basic physics. Familiarity with illumination models and light-surface interactions. Familiarity with modern graphics hardware.

[0]Image on the front page is courtesy of AJ Preetham and Natty Hoffman.

Topics List

Fundamentals of scattering, scattering in the atmosphere, appearance and modeling of daytime and night-time sky, aerial perspective, practical cloud illumination models, interactive methods for rendering sky and clouds, approximate methods for outdoor global illumination.

Throughout the course, attention will be paid to most recent developments and practical problems. In addition, the course will begin with a talk covering relevant background and end with a panel discussion.

Course Presenters' Biographies

Mark J. Harris has been a PhD student in Computer Science at The University of North Carolina at Chapel Hill since 1998. He received a BS in Computer Science from The University of Notre Dame in 1998. His dissertation research, supervised by Dr. Anselmo Lastra, is on real-time cloud rendering and simulation. His other research interests include global illumination, graphics hardware, and physically based simulation. Mark taught an undergraduate programming course at UNC in 2002. He presented a short course on real-time cloud rendering for games at the 2002 Game Developers Conference, and co-presented a short course at GDC 2003 on simulation and procedural animation using graphics hardware.

Nathaniel Hoffman received the BSc degree in computer engineering from the Technion - Israel Institute of Technology in 1993. Until 1997 he was a microprocessor architect at Intel where he contributed to the Pentium Processor with MMX Technology and the SSE and SSE 2 instruction set extensions. From 1997 to early 2003 he worked at Westwood Studios as a graphics and optimization specialist, where he also did research into outdoor rendering. Recently he has joined Naughty Dog (a wholly owned subsidiary of Sony Computer Entertainment America Inc.). Naty has presented various real-time techniques at the Game Developers Conference over the past three years: outdoor illumination in 2001, skylight and aerial perspective in 2002, and advanced illumination in 2003.

AJ Preetham is a software engineer working on various rendering techniques for next generation graphics hardware at ATI Research. Prior to this, he developed 3D modeling and reverse engineering software at Paraform Inc., and worked on rendering atmospheric effects for flight simulators at Evans & Sutherland. He graduated with an MS degree in Computer Science from University of Utah under supervision of Peter Shirley and with Bachelors of Technology from Indian Institute of Technology, Madras, India. While a graduate student at the University of Utah he developed a practical skylight model. He presented real time techniques for skylight and aerial perspective at the 2002 Game Developers Conference.

Simon Premože is a Ph.D. student in Computer Science at the University of Utah working with professor Peter Shirley. He obtained a BS degree in Computer Science from the University of Colorado at Boulder in 1996. Part of his degree was completed at the University of Ljubljana, Slovenia while pursuing a degree in Applied Mathematics. His current research interests include global illumination and rendering algorithms, modeling natural phenomena and reflectance models. He has done research on modeling and rendering natural phenomena (snow, water, night sky). Previously he worked on computer simulation and visualization of liquid crystal phase transitions and dynamics of liquid crystals. He previously participated in Siggraph 2000 course titled *Image-Based Surface Details*.

Course Schedule

8:30 AM **Introduction** **Premože**

8:35 AM **Background and Overview** **Premože**

1. Complexity of Light Transport in Outdoors
2. Overview of Scattering
   a. Definitions
   b. Basic Physics Overview
3. Overview of Existing Methods of Solving Light Transport in Outdoors
4. Practical Issues for Modeling and Rendering

8:50 AM **Global Illumination in Outdoors** **Premože**

1. Overview of Light Transport in the Outdoors
   a. Modes of Transport
   b. Interaction with Geometry
   c. Existing Methods
   d. Shortcomings
2. Approximations to Global Illumination
   a. Environment Mapping
   b. Spherical Harmonics
   c. Ambient Occlusion
   d. Semi-Local Illumination Models

9:15 AM **Skylight Modeling and Aerial Perspective** **Preetham**

1. Theory of Scattering in Atmosphere
   a. Atmosphere
   b. Scattering
2. Skylight Models
   a. Simulation Based Methods
   b. Analytic Models and Approximations
3. Aerial Perspective Model
4. Accelerated Techniques for Modeling Sky and Aerial Perspective

10:05 AM	**Night Sky Illumination**	**Premože**

1. Sources of Illumination at Night
2. Methods and Algorithms for Night Sky
   a. Sky Modeling
   b. Appearance of Moon
   c. Star Modeling
3. Tone Mapping Problems

10:15 AM	**Coffee Break**	
10:30 AM	**Practical Global Illumination**	**Hoffman**

1. Terrain
   a. Horizon mapping for Sun's contribution
   b. Hemispherical lighting factor
   c. Polynomial Texture Mapping
   d. Spherical Harmonics Transfer Functions
2. Objects
   a. Environment mapping
   b. Local Illumination with Spherical Harmonics
3. Terrain-Object interactions
   a. SH 'contact texture' for ground objects
   b. Combining illumination and shadowing
4. Demonstration

11:20 AM	**Real-time Rendering of Clouds**	**Harris**

1. Cloud Illumination Model
   a. Overview of Scattering in Clouds
   b. Illumination Approximations
   c. Simple Cloud Illumination Model
2. Accelerated Cloud Rendering
   a. Impostors
   b. Extensions to Traditional Impostors
3. Writing Cloud Rendering Engine
   a. Implementation Issues
   b. Demonstration

12:10 AM	**Summary, Questions and Answers**	**Harris, Hoffman, Preetham, Premože**

Light Transport in Participating Media

Simon Premože

Department of Computer Science
University of Utah
Email: premoze@cs.utah.edu

April 25, 2003

1 Introduction

Image synthesis of natural scenes is an important and challenging problem. The appearance of natural phenomena has always been intriguing. The fascination is evident from the vast amount of depictions of natural phenomena created, ranging from early cave paintings to impressionistic masterworks and photography.

The visual simulation of natural scenes has many practical applications. Many industries, from entertainment to architectural design, are using computer generated imagery of outdoor terrain scenes for their purposes. Therefore, it is important to design convincing visual simulation for natural scenes. For example, in flight simulators, it is critical that clouds and terrain be carefully depicted because they serve as vital visual cues to a pilot. On the other hand, the entertainment industry requires control in order to create desired effects. Therefore, the simulation of natural phenomena is subjected to two constraints: visual consistency and user control.

While scientists need to predict and simulate the behavior of certain phenomena, artists want to choreograph it in order to create a desired mood or effect. Movies require special effects not commonly found in nature or occurring very rarely. Even if the real phenomena can be found, controlling it can be difficult and cumbersome task. This makes modeling and rendering even more challenging problem. While purely physically-based simulation would yield a predictable result, the number of parameters controlling the simulation is unwieldy and unintuitive for non-expert users.

Significant progress has been made in the last decade in understanding how to generate realistic renderings of indoor scenes. The general approach is to analyze the physics of light transport in such environments and then to embody approximations to the physics in computational algorithms. Correct modeling of illumination and material properties is vital. It is now known that a sense of realism depends critically on accounting for shadows, secondary illumination, and non-uniform reflectance functions. Accurately approximating the effect of these properties involves great computational expense. As a result, methods for rendering realistic imagery almost always exploit assumptions about the nature of the geometric structure and illumination and materials properties likely to be encountered. Most of these assumptions derive from a presumption of indoor environments.

Natural scenes, especially outdoor scenes, present very different computational characteristics than indoor scenes. While the physics is the same, geometry, illumination, and reflectance properties are all distinctly different. Many of the techniques developed to support realistic rendering of indoor scenes require substantial modifications for natural, outdoor environments. The most difficult computational problem to overcome is the need to be able to aggregate the effects of micro-structures into large enough units that they can be rendered effectively, while at the same time preserving key aspects of visual appearance. This problem exists across a wide range of scales, ranging from foliage, in which a collection of individual leaves generates a collective appearance that is quite different than that of the constituent members, to distant landmarks, where detail must be suppressed without removing those properties that make landmarks distinctive and thus useful.

1.1 Why the Outdoors?

There are many reasons why study appearance, illumination and light transport in natural environments:

- **Great beauty**

- **Geometric, material and illumination richness and complexity**

- **Many modes of light transport** Light transport in natural environments is extremely complex as it varies from simple modes that can be described by traditional shading paradigms to very complicated modes that require correct physical description and full simulation.

- **Challenging to many classic algorithms** Due to complex interactions between geometry, illumination and material properties, many "traditional" algorithms cannot handle natural environments adequately. Many existing models merely apply methods devised for "indoor" environments to natural scenes. Most often visual appearance of rendered natural scenes is not adequate. Furthermore, computational demands of natural scenes are overwhelming and "smarter" methods are needed to cope with complexity and computational demands.

- **There has been lots of research focusing on man-made materials and indoor environments, but not that much on natural and outdoor environments.**

Figure 1 shows the richness of illumination, geometric complexity and reflectances.

2 Motivation

Renderings of natural outdoor scenes have had a cartoon-like quality that significantly distracts from a sense of realism. Partially, this is due to computational and source data constraints that limit the geometric complexity of scenes to be rendered. On the other hand, illumination variety, complex reflectance functions and complex and not fully understood interactions between geometric complexity, illumination and material properties has also severely limited realism of natural scenes.
Why is the quality of computer generated images of the natural outdoor scenes inadequate?

- Source data and current computational constraints limit the geometric complexity of the environment.

- Illumination plays an equally important role in creating a sense of realism.

- We do not yet fully understand interactions between geometry, illumination and material properties.

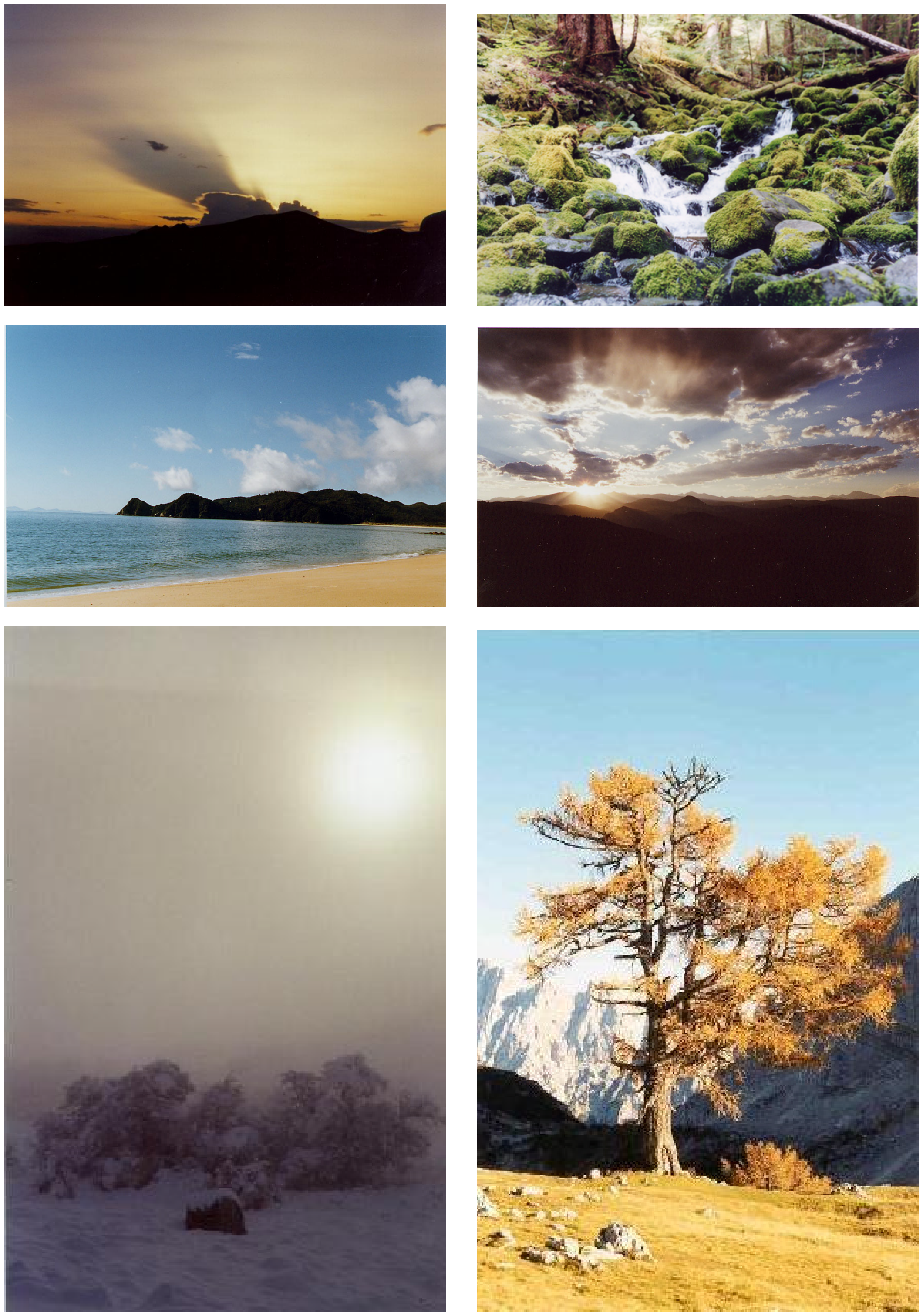


Figure 1: *Richness and variety of illumination, geometric complexity and reflectances found in natural scenes.*

Deussen *et. al.* demonstrated that improving geometric complexity of the scene also improves the perceived realism of the scene without any fancy illumination or material properties [15]. There has been a lot of research and progress made in understanding light transport in indoor environments and then approximating the physics in computational algorithms.

3 Problems

3.1 Illumination and Appearance

The human observer routinely has to deal with objects that are far from Lambertian. Many objects ubiquitous in the daily environment strongly deviate from Lambertian or Phong. Natural materials such as biological objects (leaves, skin), food (milk, fruits), or inorganic objects (sky, water, snow, clouds, weathered materials, rocks) exhibit significant subsurface or volumetric light transport. Light transport in arbitrary scattering media is very important for realistic depiction of materials [37] and scenes [57]. Many applications ranging from special effects to flight simulators and architectural design rely on subtle lighting effects and cues that often cannot be described by simplifying the light transport equation and without including effects of multiple scattering and global illumination within a scattering medium [10, 11].

For example, translucent material frequently occur in nature. The effects of translucency occur at quite different scale. While in some cases (e.g. cheese) the conventional shading paradigm may work reasonably well on the scale of the object, translucency becomes apparent in the appearance of small cracks and sharp edges. In other cases (clouds, smoke) the shading paradigm is fully inadequate.

Illumination and material appearance are at the heart of computer graphics. At the lowest level they are controlled by complex scattering events that are computationally expensive to model, hard to understand and cumbersome to control in practical applications. This is especially true for illumination and appearance in the natural outdoor environments. [11] shows that “many common observations cannot be explained by single-scattering arguments: the variation of brightness and color of the sky; the brightness of clouds; the whiteness of a glass of milk; the appearance of distant objects; the blueness of light transmitted in snow and other natural ice bodies; the darkening of sand upon wetting.” While scattering events determine the illumination and appearance, it is very cumbersome to illumination and appearance using pure physical quantities such as particle density distributions, scattering coefficents, phase functions, etc. Most often these quantities are not known and are very non-intuitive for non-expert users. While numerical methods such as Monte Carlo methods ultimately produce the correct light distribution in an environment (or material), computational cost involved in accounting for all scattering events is prohibitive.

Accurate computation of light transport is therefore very complex, computationally expensive and sometimes hard to control and understand for an inexperienced user. For image synthesis purposes, approximations with intuitive parameters may often be enough to capture the appearance of almost any material.

3.2 Global Illumination

For many years, the goal of realistic image synthesis has been to simulate reality. This goal has driven the field to create a large number of algorithms to solve the light transport equation. Smits [63] argues that "none of which [the algorithms] are or will be practical for most application. In general, this is caused by ignoring the applications when designing algorithms. The opposite effect also takes place. An algorithm that focuses on a particular application tends to fail dramatically in other application areas, and tends to be criticized for this. A better understanding of the problem space should allow us to create better algorithms and give us better standards by which to evaluate algorithms."

While Deussen *et. al.* demonstrated that the realism of natural scenes can be greatly improved by increasing geometric complexity [15] using only local illumination, the importance of global illumination for realistic image synthesis has been demonstrated in recent experiments [24, 41, 67]. The presence of shadows, specular reflections, caustics, and diffuse interreflection provide important cues to the human visual system and help determine relationships between objects. Ward [70] posed a question of "how correct do these details need to be in order to be convincing?"

While Smits [63] pointed out that most of the current global illumination algorithms are not practical for real scenes, this is even more true for natural scenes that tend to be much more complex in terms of geometric complexity, materials, illumination and modes of light transport. Most existing algorithms are therefore impractical for solving global illumination in natural scenes. Furthermore, there have been recent trends in replacing complex geometry with point primitives [14, 66]. This addresses the geometric complexity issue and makes it managable. However, this introduces another problem of global or semi-global light interactions between these point primitives. None of the existing algorithms is appropriate for this new representation. While some attempts have been made to make global illumination computation more manageable from the systems point of view [54], it is clear that more effective algorithms are needed if the realism of natural scenes is to be improved.

4 Scattering and Light Transport

Interaction of light with particles is a fundamental physical phenomena that helps explaining the appearance of surfaces and arbitrary volumetric materials and participating media. *Scattering* is a process by which a particle or surface interacts with light. Scattering has a number of variations depending on the size of interacting particles. If interacting particles are much smaller than the wavelength λ of the incident light, the process is called *Rayleigh scattering*. Molecules found in the atmosphere fall into this category and blue sky is consequence of Rayleigh scattering. On the other hand, *Mie scattering* models scattering by particles that are roughly the same size as the wavelength λ. Water vapor, fumes, dust are the main scatters in the Earth's atmosphere. This type of scattering is responsible for spectacular red/orange appearances of the sky in the evenings, especially if there has been a forest fire, or a volcanic eruption. *Non-selective scattering* occurs when the particles are much larger than the incident radiation. This type of scattering is not wavelength dependent and is the primary cause of haze. Scattering process in which light undergoes scattering only once is called *single scattering*. Many common phenomena such as the appearance of clouds, brightness and color variations of the sky, aerial perspective cannot be explained by single scattering [11]. *Multiple scattering* is a scattering process in which light undergoes more than one interaction with particles. Figure 2 illustrates scattering in participating media. A basic understanding of scattering is required for understanding of appearance of natural phenomena and illumination. In the following sections we describe scattering process in more detail.

Optical Properties

In an arbitrary medium, the underlying optical properties at location $\mathbf{x}$ in space depend on bulk material properties such as *density* $\rho(\mathbf{x})$, *temperature* $T(\mathbf{x})$, and the particle absorption and scattering cross-sections, σ_a and σ_s. Optical properties of the medium are then described in terms of the *scattering coefficient*

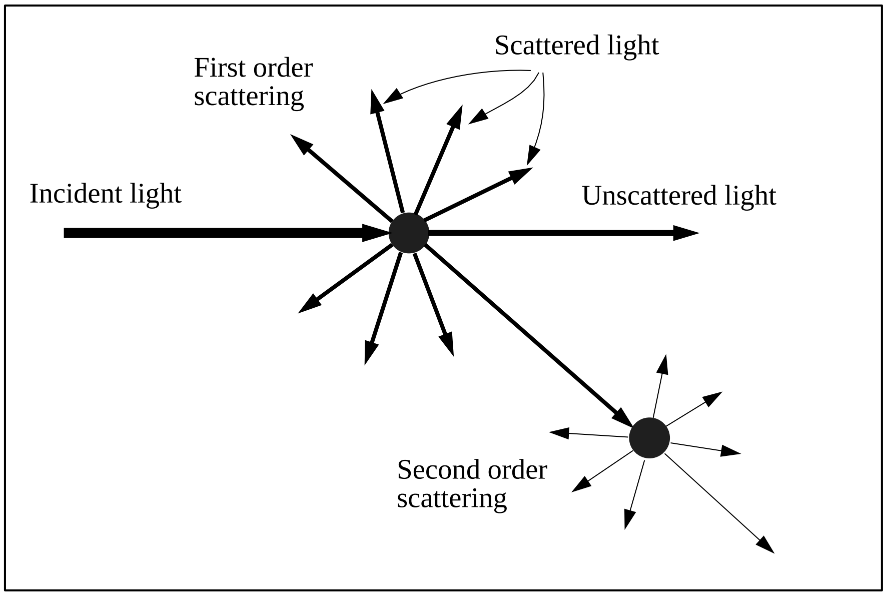


Figure 2: Scattering. (Figure courtesy of AJ Preetham)

$b(\mathbf{x}) = \sigma_s \rho(\mathbf{x})$, the *absorption coefficient* $a(\mathbf{x}) = \sigma_a \rho(\mathbf{x})$, the *extinction coefficient* $c(\mathbf{x}) = a(\mathbf{x}) + b(\mathbf{x})$, and the *phase function* $P(\mathbf{x}, \vec{\omega}, \vec{\omega}')$. Absorption and scattering coefficients are typically measured in inverse units of length (mm^{-1} or m^{-1}). Reciprocal of these coefficients is the average distance that light will travel before being absorbed or scattered. *Single scattering albedo* $W = \frac{b}{a+b}$ is the ratio of scattering to the sum of scattering and absorption. It is the percentage of all scattering events that are not absorption events. If $W = 1$, there is no absorption in the medium. Conversely, if $W = 0$, there is no scattering and the light is only attenuated due to absorption. Another important optical property is *diffuse attenuation* $K(\mathbf{x})$ which can be written in terms of the extinction coefficient (*beam extinction*): $K(\mathbf{x}) = c(\mathbf{x}) f(b/c)$. The diffuse attenuation coefficient is an important quantity because it is an *apparent* optical property of the medium and therefore depends on the structure of the incoming light field. It is defined as a ratio so it is easily measurable quantity that does not require absolute measurements. Other apparent and inherent optical properties can be expressed in terms of diffuse extinction. The *optical depth* τ will be defined later in this section.

Phase Function

The phase function $P(\mathbf{x}, \vec{\omega}, \vec{\omega}')$ is the probability that light coming from an incident direction $\vec{\omega}$ will scatter into an exitant direction $\vec{\omega}'$ upon a scattering event at point $\mathbf{x}$. The phase function can be seen as a true probability distribution and is therefore normalized:

$$\int_{4\pi} P(\vec{\omega}, \vec{\omega}') d\omega' = 1.$$

The phase function P only depends on the phase angle $\cos\theta = \vec{\omega} \cdot \vec{\omega}'$ and is reciprocal: $P(\mathbf{x}, \vec{\omega}, \vec{\omega}') = P(\mathbf{x}, \vec{\omega}', \vec{\omega})$.The mean cosine g of the scattering angle is defined as:

$$g = \int_{4\pi} P(\vec{\omega}, \vec{\omega}')(\vec{\omega} \cdot \vec{\omega}') d\omega'.$$

If a mean cosine is 0, the scattering is isotropic. On the other hand, if g is negative, backward scattering dominates; and if g is positive, the scattering is predominantly in the forward direction. The phase function only describes what happens when light is scattered by the particle and does not tell you anything when light gets absorbed upon the scattering event. The shape of the phase function strongly depends on size and orientation of particles in the medium. In general, the phase function will differ from particle to particle. For simplicity and practical reasons, an average phase function that describes the most important features of the scattering process is used. For clarity we will drop positional dependence of optical parameters through the rest of the notes.

ISOTROPIC PHASE FUNCTION. The simplest phase function is the isotropic phase function:

$$P(\vec{\omega}, \vec{\omega}') = \frac{1}{4\pi}. \tag{1}$$

The light will scatter in random direction over the entire sphere with equal probability.

HENYEY-GREENSTEIN PHASE FUNCTION. The Henyey-Greenstein (HG) phase function was first introduced by Henyey and Greenstein [23] to describe scattering of radiation in a galaxy. The Henyey-Greenstein phase function has proven to be useful in approximating the angular scattering dependence of single scattering events in biological tissues, water, clouds and many other natural materials. It is very popular, because it is a fast and simple approximation to true Mie scattering phase function that is very expensive to evaluate. The HG phase function is:

$$P_{HG}(\vec{\omega},\vec{\omega}') = \frac{1-g^2}{(1+g^2-2g\cos\theta)^{\frac{3}{2}}} \tag{2}$$

where the asymmetry parameter g controls the shape of light redistribution upon scattering event. Note that the phase function P_{HG} still needs to be normalized by multiplying it by $1/4\pi$.

Double Henyey-Greenstein Phase Function. The Henyey-Greenstein phase function cannot capture scattering events that have two lobes, one in forward direction and the other in backward direction. A simple extension has been proposed by Kattawar [34] that combines a forward and backward elongated HG phase functions:

$$P(\vec{\omega},\vec{\omega}') = (1-f)P_{HG}(\vec{\omega},\vec{\omega}',g_1) + fP_{HG}(\vec{\omega},\vec{\omega}',g_2) \tag{3}$$

where $g_1 > 0$ (forward scattering) and $g_2 < 0$ (backward scattering). Further extensions are possible by combining more than two lobes.

Schlick Phase Function. While the Henyey-Greenstein phase function is a good approximation to Mie scattering, it is still relatively expensive to evaluate. Schlick observed that the accurate shape is often not crucial for rendering applications and he replaced a relatively expensive exponentiation with even simpler expression [7]:

$$P(\vec{\omega},\vec{\omega}') = \frac{1-k^2}{(1+k\cos\theta)^2} \tag{4}$$

where k is a parameter similar to the asymmetry parameter g: $-1 \le k \le 1$. The phase function still needs to be normalized by multiplying it by $1/4\pi$, so that it will integrate to 1.

Rayleigh Phase Function. In order for Rayleigh scattering to be valid, the size of the particle must be much smaller than the wavelength λ of the incident light. Small particles (about 0.1λ) scatter light equally in forward and backward directions:

$$P(\vec{\omega},\vec{\omega}') = \frac{3}{4}\frac{(1+\cos^2\theta)}{\lambda^4}. \tag{5}$$

Unlike phase function that approximate Mie scattering, Rayleigh scattering phase function is inversely proportional to the fourth power of wavelength of light. This

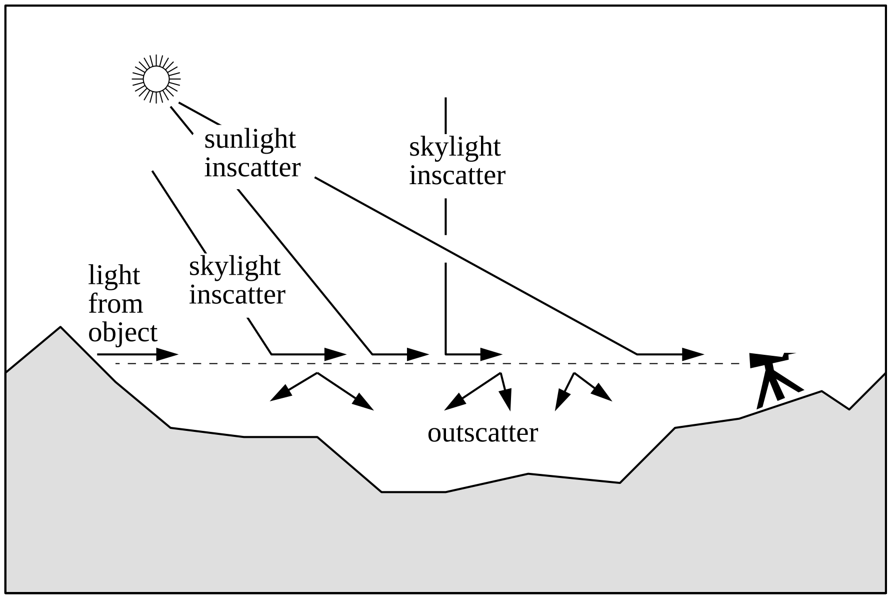


Figure 3: Scattering in participating media. The intensity of light in viewing direction is reduced due to absorption and outscattering. Scattering of light into viewing direction increases the intensity and modifies the color of the light seen by the observer. (Figure courtesy of AJ Preetham)

means that red light (700*nm*) is scattered about ten times less than blue light (400*nm*).

CORNETTE-SHANKS PHASE FUNCTION. Cornette and Shanks [13] modified the Henyey-Greenstein phase function and gave it a more reasonable physical expression:

$$P(\vec{\omega},\vec{\omega}') = \frac{3}{2}\frac{(1-g^2)}{(2+g^2)}\frac{1+\cos^2\theta}{(1+g^2-2g\cos\theta)^{\frac{3}{2}}}. \qquad (6)$$

This phase function is especially useful for clouds. Note that if $g = 0$, this function is equivalent to Rayleigh scattering. As before, the phase function needs to be normalized by multiplying it by $1/4\pi$.

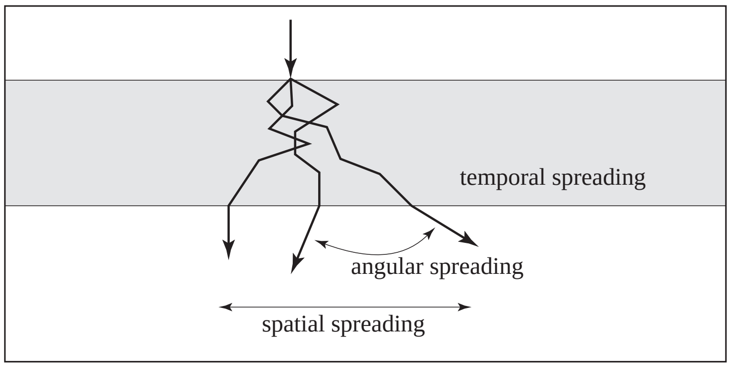


Figure 4: Scattering in a highly scattering medium. Original radiance undergoes a series of scattering events that result in angular, spatial and temporal spreading of the original radiance distribution.

4.1 Light Transport

Upon entering the medium, incoming light undergoes a series of scattering and absorption events that modify both the directional structure of the incoming light field and its intensity. Light intensity in viewing direction $\vec{\omega}$ is reduced (attenuated) due to *absorption* and *outscattering*. On the other hand, as a result of scattering, light can also scatter into a viewing direction (*inscattering* from arbitrary direction and change the light intensity and color in viewing direction. Figure 3 illustrates scattering events and their contributions to the final color and intensity of the light. As a result of multiple scattering events, the original radiance distribution undergoes angular, spatial and temporal spreading which result in different radiance distribution. Figure 4 shows spreading effects in an arbitrary highly scattering medium. Table 1 summarizes terms and quantities used in these notes. We now examine the amount of attenuation and inscattering due to absorption and scattering in arbitrary participating media. We first write the change in radiance when light is moving through a segment of size ds.

ABSORPTION. Absorption coefficient a describes the probability of a photon being absorbed per unit length. The change in radiance dL in direction $\vec{\omega}$ due to absorption is:

$$dL(\mathbf{x},\vec{\omega}) = -a(\mathbf{x})L((\mathbf{x},\vec{\omega})ds. \tag{7}$$

OUTSCATTERING. Scattering coefficient b describes the probability of a photon

$\mathbf{x}$	Generic location in $\mathbb{R}^3$
$\vec{\omega}$	Generic direction
$a(\mathbf{x})$	Absorption coefficient at a point
$b(\mathbf{x})$	Scattering coefficient at a point
$c(\mathbf{x})$	Extinction coefficient at a point
g	Mean cosine of the scattering angle
Q	Volume source distribution
$P(\vec{\omega}, \vec{\omega}')$	Phase function
T	Transmittance
τ	Optical depth

Table 1: Notation and quantities used in these notes.

being scattered per unit length. The change in radiance dL in direction $\vec{\omega}$ due to scattering is:

$$dL(\mathbf{x}, \vec{\omega}) = -b(\mathbf{x})L((\mathbf{x}, \vec{\omega})ds. \tag{8}$$

EXTINCTION. The total change in radiance due to absorption and outscattering in direction $\vec{\omega}$ along the segment length ds is:

$$dL(\mathbf{x}, \vec{\omega}) = -c(\mathbf{x})L((\mathbf{x}, \vec{\omega})ds \tag{9}$$

where $c = a + b$ is the attenuation (extinction) coefficient that describes the probability that the photon will be either scattered or absorbed.

INSCATTERING. As mentioned before, the light can scatter into the viewing direction $\vec{\omega}$ from all directions. The change in radiance over segment ds in direction $\vec{\omega}$ is:

$$dL(\mathbf{x}, \vec{\omega}) = b(\mathbf{x}) \int_{4\pi} P(\mathbf{x}, \vec{\omega}, \vec{\omega}')L(\mathbf{x}, \vec{\omega'})d\omega' ds \tag{10}$$

where $P(\mathbf{x}, \vec{\omega}, \vec{\omega}')$ is the phase function. Since it is possible that light can scatter from any direction, the incident radiance must be integrated over entire sphere of directions. In practice, this results in computationally very expensive computation.

EMISSION. It is possible the volumetric medium is also emitting light. The change in radiance dL due to emission withing the medium is:

$$dL(\mathbf{x}, \vec{\omega}) = -a(\mathbf{x})L_e(\mathbf{x}, \vec{\omega})ds \tag{11}$$

where $L_e(\mathbf{x}, \vec{\omega})$ is the emitted radiance at point $\mathbf{x}$ in direction $\vec{\omega}$.

Optical Depth and Transmittance. Optical depth τ over a uniform segment of length ds is a product of extinction coefficient c and segment length ds. Optical depth $\tau(s)$ over a segment of length s in arbitrary inhomogeneous is then:

$$\tau(s) = \int_0^s c(\mathbf{x} + s'\vec{\omega})ds'. \tag{12}$$

More generally, the optical depth $\tau(\mathbf{x}, \mathbf{x}')$ defined over an arbitrary line segment starting at parameter s and ending at parameter s' is:

$$\tau(s, s') = \int_s^{s'} c(\mathbf{x} + t\vec{\omega})dt. \tag{13}$$

Optical depth related to transmittance $T(s, s')$ over a line segment from s to s' as follows:

$$T(s, s') = \exp(-\tau(s, s')). \tag{14}$$

The transmittance can be interpreted as the percentage of light that reached point $\mathbf{x}'$ at parameter s' starting at point $\mathbf{x}$ and parameter s. Opacity is just an inverse of the transmittance $T(s, s')$:

$$\alpha(s, s') = 1 - T(s, s'). \tag{15}$$

Light Transport Equation. We have so far described the change in radiance dL over distance ds due to absorption, outscattering, emission and inscattering. By combining all these components, the total change in radiance $L(\mathbf{x}, \vec{\omega})$ at point $\mathbf{x}$ and in direction $\vec{\omega}$ is written in terms of the *light transport equation* [2, 26]:

$$\begin{aligned}(\vec{\omega} \cdot \nabla)L(\mathbf{x} + s\vec{\omega}) &= -c(\mathbf{x})L(\mathbf{x}, \vec{\omega}) + b(\mathbf{x}) \int_{4\pi} P(\mathbf{x}, \vec{\omega}, \vec{\omega}')L(\mathbf{x}, \vec{\omega}')d\omega' \\ &\quad + a(\mathbf{x})L_e(\mathbf{x}, \vec{\omega})\end{aligned} \tag{16}$$

It is often convenient to split the total radiance within the medium into components and write it as the sum of *unscattered* (direct) radiance L_{un}, the *emission* L_e and the *scattered* radiance L_{sc}:

$$L(\mathbf{x}, \vec{\omega}) = L_{un}(\mathbf{x}, \vec{\omega}) + L_{sc}(\mathbf{x}, \vec{\omega}) + L_e(\mathbf{x}, \vec{\omega}). \tag{17}$$

Here L_{un} is the radiance which intensity has been reduced due to absorption and outscattering along the length S. L_{sc} is the radiance that has undergone a series of scattering events and finally scattered into a small cone around the observation direction $\vec{\omega}$.

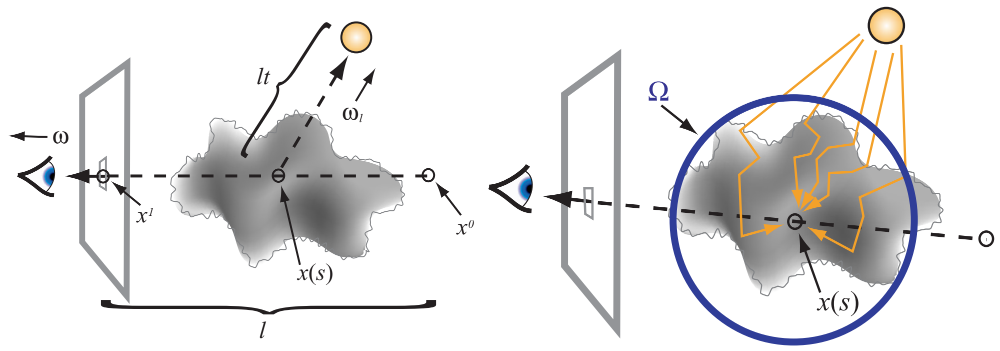


Figure 5: Monte Carlo Ray Tracing solution of the light transport equation in an arbitrary inhomogeneous medium. Left: direct lighting computation. For every sample point ong the viewing ray, a ray toward each light source is sent and light intensity is computed via raymarching. Right: Inscattering computation. At every sample point on the viewing ray, indirect light contribution is integrated over entire sphere of directions. (Figures from Kniss *et. al.* [36])

4.2 Solving Light Transport Equation Monte Carlo Ray Tracing

Analytic solutions for light transport equation for inhomogeneous media expressed in equation 17 are not possible. Numerical solutions are necessary to solve this integral equation. There are many different methods of solving this equation ranging from very robust Monte Carlo methods to very specialized solutions that make additional assumptions about optical properties of the media and boundary conditions.

Monte Carlo ray tracing is an accurate algorithm for solving the radiative transfer equation in arbitrary media. We march through the medium in direction $\vec{\omega}$ sampling points along the ray. At every sampling point along the ray, we send a a ray toward each light source. The contribution of each light source is computed by marching along a ray toward light direction $\vec{\omega}$ and computing attenuation (see Figure 5, left). Then the light from previous step is attenuated and the light that is inscattered into the viewing direction $\vec{\omega}$ is gathered (see Figure 5, right).

Direct Lighting. The light that has been scattered in the participating media *exactly* once on the way from a light source to the viewer is the *direct lighting*. Using standard ray marching, the direct lighting portion of equation 17 is:

$$\begin{aligned} L_{n+1}(\mathbf{x}+\Delta s\vec{\omega},\vec{\omega}) &= \sum_{l}^{allNlights} L_l(\mathbf{x},\vec{\omega}_l')P(\mathbf{x},\vec{\omega},\vec{\omega}_l')b(\mathbf{x})\Delta s + \\ &+ e^{-c(\mathbf{x})\Delta s}L_n(\mathbf{x},\vec{\omega}) \end{aligned} \tag{18}$$

where $L_l(\mathbf{x},\vec{\omega}_l')$ is the contribution from light source l, and Δs is the step say in ray march. The light intensity $L_l(\mathbf{x},\vec{\omega}_l')$ is computed by shooting a *shadow ray* towards each light source, ray marching through the volume until the light source is hit and computing transmittance T (equation 14) along the shadow ray:

$$L_l(\mathbf{x},\vec{\omega}_l') = TI_l(\vec{\omega}_l') \tag{19}$$

where $I_l(\vec{\omega}_l')$ is the intensity of the light source in direction $\vec{\omega}_l'$.

INDIRECT LIGHTING. The light that has scattered multiple times (inscattered light) is collected recursively for each ray:

$$L_{n+1}(\mathbf{x}+\Delta s\vec{\omega},\vec{\omega}) = \left(\frac{4\pi}{M}\sum_{i=1}^{M} L_{sc}(\mathbf{x},\vec{\omega}_i)P(\mathbf{x},\vec{\omega},\vec{\omega}_i)\right) b(\mathbf{x})\Delta s \tag{20}$$

where M is the number of directional samples taken and Δs is the ray marching step size. Computing indirect contribution involves integrating over M directions. The light contribution from each direction involves recursive computation that grows exponentially.

COMPLETE LIGHTING COMPUTATION. By combining the direct and indirect contributions, we compute the total radiance L in participating media:

$$\begin{aligned} L_{n+1}(\mathbf{x}+\Delta s\vec{\omega},\vec{\omega}) = \quad & \sum_{l}^{allNlights} L_l(\mathbf{x},\vec{\omega}_l')P(\mathbf{x},\vec{\omega},\vec{\omega}_l')b(\mathbf{x})\Delta s + \\ & \left(\frac{4\pi}{M}\sum_{i=1}^{M} L_{sc}(\mathbf{x},\vec{\omega}_i)P(\mathbf{x},\vec{\omega},\vec{\omega}_i)\right) b(\mathbf{x})\Delta s + \\ & e^{(-c(\mathbf{x})\Delta s)}L_n(\mathbf{x},\vec{\omega}). \end{aligned}$$

Equation 21 computes direct contribution, indirect contribution and adds contribution from the previous segment L_n.

While Monte Carlo ray tracing is robust and powerful, it is also very slow because of the large number of rays needed. At evry sampling point along the ray, exponential number of rays is spawned in order to compute inscattering. There are many improvements over the basic brute fore Monte Carlo ray tracing algorihtm just described that improve both convergence rate (adaptive raymarching based on material's density) and quality (Russian roulette, importance sampling, etc.).

Monte Carlo methods are also often used to compute radiative transport within a medium. Although simple and powerful, these methods suffer from slow convergence. Finite element methods are also used, but they require large amounts of storage to capture discontinuities and strong directional light distributions. A brief overview of many existing methods is presented in Section 5. Solutions specially tailored for computer graphics applications for efficient implementation on modern graphics hardware will be presented in later chapters of these course notes.

5 Background

A vast amount of literature exists on scattering and light transport. A non-linear integral scattering equation that describes the scattering events inside a volume has been studied extensively by Ambarzumian [1], Chandrasekhar [12], Bellman *et. al.* [4] and van de Hulst [68]. Their work ranges in complexity from semi-infinite homogeneous isotropic atmospheres to finite inhomogeneous anisotropic atmospheres. Mobley [44] applied these one-dimensional scattering equations to a variety of problems, mainly in hydrologic optics. Pharr and Hanrahan [53] described a mathematical framework for solving this scattering equation in the context of computer graphics and a variety of rendering problems and also described a numerical solution in terms of a Monte Carlo sampling method. Pharr and Hanrahan exploited interaction principle which encapsulates all transfer properties within a layer. Adding and doubling method extends interaction between more than two homogeneous layers [20].

Siegel and Howell [60] provide a fundamental description of light transport as a classic equation of transfer. In a seminal work, Blinn [9] presented a model for the reflection and transmission of light through thin clouds of particles based on probabilistic arguments and single scattering approximations in which Fresnel effects were considered. He recognized the importance of light transport for computer graphics applications. The first methods for solving light transport in participating media for computer graphics only accounted for direct illumination

(Max [43], Klassen [35]). Analytical approximations to the light transport equation exist, but they are severely restricted by underlying assumptions such as homogeneous optical properties and density, simple lighting, or unrealistic boundary conditions. Numerical methods and algorithmic approaches are needed to address the global illumination in environments including participating media and volumetric materials. We briefly review several different methods. Perez *et. al.* [52] survey and classify global illumination algorithms in participating media in more detail. An alternative description of light propagation was done by Pharr and Hanrahan [53] who described a mathematical framework for solving the scattering equation in the context of a variety of rendering problems and also described a numerical solution in terms of a Monte Carlo sampling method.

Monte Carlo Methods

Monte Carlo methods are robust and widely used techniques for solving light transport equation. Hanrahan and Krueger modeled scattering in layered surfaces with linear transport theory and derived explicit formulas for backscattering and transmission [21]. Their model is powerful and robust, but it relies on Monte Carlo methods and therefore suffers from noise artifacts and slow convergence. Blasi *et. al.* [8, 7] described an algorithm based on a particle light tracing simulation. The interaction points in the media are spaced at a constant distance. Similarly, Pattanaik and Mudur [50] also presented a Monte Carlo light tracing algorithm. Their method generates random walks starting from the light source, and interaction points in the medium are sampled according to transmittance of the volume. Lafortune and Willems [38] improved upon the method by tracing paths both from light sources and the eye. Baranoski and Rokne [3] simulated light transport in leaves using the Monte Carlo method. Jensen and Christensen [30] presented a two pass volume photon density estimation method using a photon map. Their method is simple, robust and efficient but suffers from additional memory requirements to store photons if the extent of the scene is large or the lighting configuration is very difficult. Dorsey *et. al.* [18] described a method for full volumetric light transport inside stone structures using a volumetric photon map representation. Photon map was also used for depicting scattering in wet materials [31], smoke [19] and fire [45]. Veach and Guibas [69] proposed a novel global illumination algorithm that found an important path that contributed the most to the final pixel intensity by Markov Chain Monte Carlo method. Once the important path was found, the path space was explored locally because it was likely that other important paths would be nearby. Pauly *et. al.* [51] extended the method for participating media and proposed suitable mutation strategies for paths. Although extremely general and robust, as it could handle any lighting condition and configuration, it still suffered from classical Monte Carlo artifacts.

Finite Element Methods

Finite element methods provide an alternative approach to solving integral equations. Rushmeier [59, 58] presented zonal finite element methods for isotropic scattering. Bhate [6] described an improvement over the zonal method that included a progressive refinement of elements. Sillion [61] extended the classical hierarchical radiosity algorithm to include isotropically scattering participating media. Spherical harmonics were also used by Kajiya and von Herzen [33] to compute anisotropic scattering in volumetric media while Languenou *et. al.* [39] used discrete ordinate methods. Bhate [5] extended the zonal method to include the interactions between surface and volume elements that were not accounted for by Kajiya and von Herzen [33]. Patmore [49] formulated a local solution for non-emitting volumes and the global solution was found by iterative expansion of local solutions on a cubic lattice. Max *et. al.* [42] used a one-dimensional scattering equation to compute the light transport in tree canopies by solving a system of differential equations by application of the Fourier transform. Their method became expensive for forward peaked phase functions as the hemisphere needed to be more finely discretized. All of these finite element methods require discretization of volumetric media in space and angles, and therefore require a large amount of memory to effectively compute interactions between elements, especially if discontinuities or glossy reflections are to be captured.

Other methods

Alternative methods that do not rely on Monte Carlo or finite element methods have also been proposed. Stam [64] presented a solution to multiple scattering by solving the diffusion equation using a multigrid method. Jensen *et. al.* [32] introduced an analytical diffusion approximation to multiple scattering, which is especially applicable for materials that exhibit considerable subsurface light transport. Their method relies on the assumption that the multiply-scattered light is nearly isotropic and cannot be easily extended to inhomogeneous materials. Lensch *et. al.* [40] implemented this method in graphics hardware and Jensen and Buhler [29] extended this diffusion approximation to be computationally more efficient by storing illumination in a hierarchical grid.

There have also been some specialized approximations that are not applicable to arbitrary participating or volumetric media. Nishita *et. al.* [48] presented an approximation to light transport inside clouds. Similarly, Irwin uses adaptive Simpson quadrature method to compute sky radiance while only accounting for Rayleigh scattering [25]. Jackel and Walter presented a method of renndering sky based on Mie scattering using extinction correction method to deal with multiple scattering [28]. Harris *et. al.* [22] described a fast hardware accelerated method for realistic depiction of clouds. Several other hardware algorithms for approximating light transport in volumetric media has been described by Nishita *et. al.* [47], Dobashi *et. al.* [16] and Iwasaki *et. al.* [27]. Premože and Ashikhmin [56] presented a model for light transport in water. Their approximation was specialized in that it could only be applied to natural water bodies. Nishita [46] presented an overview of light transport and scattering methods for natural environments. Stam described an efficient but highly specialized illumination model for a skin layer [65]. Preetham *et. al.* employed Monte Carlo simulations for sky simulations [55]. The results of simulations were then fit to a parametric model to obtain a practical model of a daylight sky. Dobashi et al. [17] proposed a fast method for rendering the atmospheric scattering effects based look-up tables that store the intensities of the scattered light, and these are then used as textures. Sloan *et. al.* precomputed radiance transfer in low frequency illumination environment and stored the transferred radiance using spherical harmonics basis functions [62]. Kniss *et. al.* proposed an empirical volume shading model accounting for scattering in translucent materials by blurring illumination within a cone [36].

References

[1] AMBARZUMIAN, V. A. A new method for computing light scattering in turbid media. *Izv. Akad. Nauk SSSR 3* (1942).

[2] ARVO, J. Transfer equations in global illumination. Global Illumination, SIGGRAPH ‘93 Course Notes, August 1993.

[3] BARANOSKI, G. V. G., AND ROKNE, J. G. An algorithmic reflectance and transmittance model for plant tissue. *Computer Graphics Forum 16*, 3 (Aug. 1997), 141–150. Proceedings of Eurographics ’97. ISSN 1067-7055.

[4] BELLMAN, R. E., KALABA, R. E., AND PRESTRUD, M. C. *Invariant imbedding and radiative transfer in slabs of finite thickness.* American Elsevier Publishing Company, New York, 1963.

[5] BHATE, N. Application of rapid hierarchical radiosity to participating media. In *Proceedings of ATARV-93: Advanced Techniques in Animation, Rendering, and Visualization* (Ankara, Turkey, July 1993), Bilkent University, pp. 43–53.

[6] BHATE, N., AND TOKUTA, A. Photorealistic volume rendering of media with directional scattering. *Third Eurographics Workshop on Rendering* (May 1992), 227–245.

[7] BLASI, P., LE SAEC, B., AND SCHLICK, C. A rendering algorithm for discrete volume density objects. In *Eurographics '93* (Oxford, UK, 1993), R. J. Hubbold and R. Juan, Eds., Eurographics, Blackwell Publishers, pp. 201–210.

[8] BLASI, P., SAEC, B. L., AND SCHLICK, C. An importance driven monte-carlo solution to the global illumination problem. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 173–183.

[9] BLINN, J. F. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics (SIGGRAPH '82 Proceedings) 16*, 3 (July 1982), 21–29.

[10] BOHREN, C. F. *Clouds in a glass of beer*. John Wiley and Sons, 1987.

[11] BOHREN, C. F. Multiple scattering of light and some of its observable consequences. *American Journal of Physics 55*, 6 (June 1987), 524–533.

[12] CHANDRASEKAR, S. *Radiative Transfer*. Dover, New York, 1960.

[13] CORNETTE, W. M., AND SHANKS, J. G. Physically reasonable analytic expression for the single-scattering phase function. *Applied Optics 31*, 16 (1992), 3152–3160.

[14] DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. Interactive visualization of complex plant ecosystems. In *IEEE Visualization 2002* (Boston, MA, October 2002).

[15] DEUSSEN, O., HANRAHAN, P. M., LINTERMANN, B., MECH, R., PHARR, M., AND PRUSINKIEWICZ, P. Realistic modeling and rendering of plant ecosystems. In *Proceedings of SIGGRAPH 98* (Orlando, Florida, July 1998), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH / Addison Wesley, pp. 275–286. ISBN 0-89791-999-8.

[16] DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles. In *Proc. of the 9th Pacific Conference* (2001), pp. 390–399.

[17] DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proceedings of Graphics Hardware* (2002), pp. 1–10.

[18] DORSEY, J., EDELMAN, A., JENSEN, H. W., LEGAKIS, J., AND PEDERSEN, H. Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH 1999* (August 1999), pp. 225–234.

[19] FEDKIW, R., STAM, J., AND JENSEN, H. W. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH 2001* (August 2001), Computer Graphics Proceedings, Annual Conference Series, ACM Press / ACM SIGGRAPH, pp. 15–22. ISBN 1-58113-292-1.

[20] GOODY, R., AND YUNG, Y. *Atmospheric Radiation: Theoretical Basis*, second ed. Oxford University Press, 1989.

[21] HANRAHAN, P., AND KRUEGER, W. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH 93* (Anaheim, California, August 1993), Computer Graphics Proceedings, Annual Conference Series, pp. 165–174.

[22] HARRIS, M. J., AND LASTRA, A. Real-time cloud rendering. In *Eurographics 2001 Proceedings* (September 2001), pp. 76–84.

[23] HENYEY, L. C., AND GREENSTEIN, J. L. Diffuse radiation in the galaxy. *Astrophys. J 93* (1941), 70–83.

[24] HU, H. H., GOOCH, A. A., THOMPSON, W. B., SMITS, B. E., RISER, J. J., AND SHIRLEY, P. Visual cues for imminent object contact in realistic virtual environments. In *IEEE Visualization 2000* (Salt Lake City, Utah, Sept. 2000).

[25] IRWIN, J. Full-spectral rendering of the earth's atmosphere using a physical model of rayleigh scattering. In *Proceedings of the 14th Annual Eurographics UK Conference* (March 1996).

[26] ISHIMARU, A. *Wave Propagation and Scattering in Random Media, Volume I: Single scattering and transport theor y; Volume II: Multiple scattering, turbulence, rough surfaces and remote sensing*. Academic Press, New York, 1978.

[27] IWASAKI, K., DOBASHI, Y., AND NISHITA, T. Efficient rendering of optical effects within water using graphics hardware. In *Proc. of the 9th Pacific Conference* (2001).

[28] JACKÈL, D., AND WALTER, B. Modeling and rendering of the atmosphere using mie-scattering. *Computer Graphics Forum 16*, 4 (1997), 201–210. ISSN 1067-7055.

[29] JENSEN, H. W., AND BUHLER, J. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics 21*, 3 (July 2002), 576–581. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[30] JENSEN, H. W., AND CHRISTENSEN, P. H. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH 98* (Orlando, Florida, July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 311–320.

[31] JENSEN, H. W., LEGAKIS, J., AND DORSEY, J. Rendering of wet materials. In *Eurographics Rendering Workshop 1999* (Granada, Spain, June 1999), Springer Wein / Eurographics.

[32] JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001* (August 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 511–518.

[33] KAJIYA, J. T., AND VON HERZEN, B. P. Ray tracing volume densities. In *Computer Graphics (SIGGRAPH '84 Proceedings)* (July 1984), H. Christiansen, Ed., vol. 18, pp. 165–174.

[34] KATTAWAR, G. W. A three-parameter analytic phase function for multiple scattering calculations. *J. Quant. Spectr. Radiat. Transfer 15* (1975), 839–849.

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

[36] KNISS, J., PREMOŽE, S., HANSEN, C., AND EBERT, D. Interactive translucent volume rendering and procedural modeling. In *Proceedings of IEEE Visualization* (October 2002).

[37] KOENDERINK, J. J., AND VAN DOORN, A. J. Shading in the case of translucent objects. In *Human Vision and Electronic Imaging VI* (2001), pp. 312–320.

[38] LAFORTUNE, E. P., AND WILLEMS, Y. D. Rendering participating media with bidirectional path tracing. In *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), X. Pueyo and P. Schröder, Eds., Eurographics, Springer Wien, pp. 91–100. ISBN 3-211-82883-4.

[39] LANGUENOU, E., BOUATOUCH, K., AND CHELLE, M. Global illumination in presence of participating media with general properties. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 69–85.

[40] LENSCH, H. P. A., GOESELE, M., BEKAERT, P., KAUTZ, J., MAGNOR, M. A., LANG, J., AND SEIDEL, H. P. Interactive rendering of translucent objects. In *Proceedings of Pacific Graphics* (2002).

[41] MADISON, C., THOMPSON, W., KERSTEN, D., SHIRLEY, P., AND SMITS, B. Use of interreflection and shadow for surface contact. *Perception and Psychophysics* (2000).

[42] MAX, N., MOBLEY, C., KEATING, B., AND WU, E. Plane-parallel radiance transport for global illumination in vegetation. In *Eurographics Rendering Workshop 1997* (New York City, NY, June 1997), J. Dorsey and P. Slusallek, Eds., Eurographics, Springer Wien, pp. 239–250. ISBN 3-211-83001-4.

[43] MAX, N. L. Light diffusion through clouds and haze. *Computer Vision, Graphics and Image Processing 33*, 3 (Mar. 1986), 280–292.

[44] MOBLEY, C. D. *Radiative transfer in natural waters*. Academic Press, 1994.

[45] NGUYEN, D. Q., FEDKIW, R. P., AND JENSEN, H. W. Physically based modeling and animation of fire. *ACM Transactions on Graphics 21*, 3 (July 2002), 721–728. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[46] NISHITA, T. Light scattering models for the realistic rendering. In *Proceedings of the Eurographics* (1998), pp. 1–10.

[47] NISHITA, T., AND NAKAMAE, E. Method of displaying optical effects within water using accumulation buffer. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 373–381. ISBN 0-89791-667-0.

[48] NISHITA, T., NAKAMAE, E., AND DOBASHI, Y. Display of clouds and snow taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 379–386. held in New Orleans, Louisiana, 04-09 August 1996.

[49] PATMORE, C. Simulated multiple scattering for cloud rendering. In *International Conference on Computer Graphics '93 Proceedings* (Feb. 1993), pp. 59–70. held in Bombay, India.

[50] PATTANAIK, S. N., AND MUDUR, S. P. Computation of global illumination in a participating medium by monte carlo simulation. *The Journal of Visualization and Computer Animation 4*, 3 (July–Sept. 1993), 133–152.

[51] PAULY, M., KOLLIG, T., AND KELLER, A. Metropolis light transport for participating media. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering* (June 2000), Eurographics, pp. 11–22. ISBN 3-211-83535-0.

[52] PÉREZ, F., PUEYO, X., AND SILLION, F. X. Global illumination techniques for the simulation of participating media. In *Eurographics Rendering Workshop 1997* (New York City, NY, June 1997), J. Dorsey and P. Slusallek, Eds., Eurographics, Springer Wien, pp. 309–320. ISBN 3-211-83001-4.

[53] PHARR, M., AND HANRAHAN, P. M. Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proceedings of SIGGRAPH 2000* (July 2000), Computer Graphics Proceedings, Annual Conference Series, pp. 75–84.

[54] PHARR, M., KOLB, C., GERSHBEIN, R., AND HANRAHAN, P. Rendering complex scenes with memory-coherent ray tracing. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 101–108. ISBN 0-89791-896-7.

[55] PREETHAM, A. J., SHIRLEY, P. S., AND SMITS, B. E. A practical analytic model for daylight. In *Proceedings of SIGGRAPH 99* (Los Angeles, California, August 1999), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH / Addison Wesley Longman, pp. 91–100. ISBN 0-20148-560-5.

[56] PREMOŽE, S., AND ASHIKHMIN, M. Rendering natural waters. In *Proceedings of Pacific Graphics* (October 2000).

[57] RUSHMEIER, H. Rendering participating media: Problems and solutions from application areas. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 35–56.

[58] RUSHMEIER, H. E. *Realistic Image Synthesis for Scenes with Radiatively Participating Media*. Ph.d. thesis, Cornell University, 1988.

[59] RUSHMEIER, H. E., AND TORRANCE, K. E. The zonal method for calculating light intensities in the presence of a participating medium. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., vol. 21, pp. 293–302.

[60] SIEGEL, R., AND HOWELL, J. R. *Thermal Radiation Heat Transfer*, second ed. Hemisphere Publishing Corp., New York, 1981.

[61] SILLION, F. X. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics 1*, 3 (Sept. 1995), 240–254. ISSN 1077-2626.

[62] SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3 (July 2002), 527–536. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[63] SMITS, B. Global illumination: Do we have the right goal? Unpublished Manuscript, 1999.

[64] STAM, J. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop 1995* (June 1995), Eurographics.

[65] STAM, J. An illumination model for a skin layer bounded by rough surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering* (London, England, June 2001).

[66] STAMMINGER, M., AND DRETTAKIS, G. Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering* (June 2001), Eurographics, pp. 151–162. ISBN 3-211-83709-4.

[67] THOMPSON, W. B., SHIRLEY, P., SMITS, B., KERSTEN, D. J., AND MADISON, C. Visual glue. Tech. Rep. UUCS-98-007, Computer Science Department, University of Utah, March 1998.

[68] VAN DE HULST, H. *Multiple Light Scattering*. Academic Press, New York, NY, 1980.

[69] VEACH, E., AND GUIBAS, L. J. Metropolis light transport. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 65–76. ISBN 0-89791-896-7.

[70] WARD, G. J. Local illumination is all that matters for graphics realism. Perceptually Adaptive Graphics. ACM Siggraph and Eurographics Campfire, 2001.

Modeling Skylight and Aerial Perspective

Arcot J. Preetham
ATI Research
preetham@ati.com

Theory of Scattering in Atmosphere

Atmosphere

Earth is enveloped by a vast amount of air called the atmosphere. It is relative thin compared to the size of the earth and fades away with increasing distance from the earth's surface. The earth's atmosphere is categorized into 4 layers - troposphere, stratosphere, mesosphere and thermosphere. Troposphere where all the weather takes place and stratosphere, the two closest layers to earth's surface constitute more than 99% of the atmosphere. The troposphere and stratosphere extend up to 12 km and 53 km respectively from the earth's surface. The earth's atmosphere is composed of many gases. Nitrogen from decay of biological products constitutes 98% and Oxygen from photosynthesis constitutes 21% of the atmosphere. Gravity holds the atmosphere close to the earth's surface and explains why the density of the atmosphere decreases with altitude.

The density and pressure of the atmosphere vary with altitude and depends on solar heating and geomagnetic activity. The simplest is an exponential fall off model where pressure and density decrease exponentially with altitude. In 1976, US Standard Atmosphere Model was adopted by COESA (Committee on Extension to the Standard Atmosphere) which describes the earth's atmosphere as composed of 7 layers up to 86 km and pressure, temperature and density are specified for each layer [McCartney1976]. The density is calculated using a perfect gas relationship.

In addition to the various gases, atmosphere also contains water vapor, dust particles, etc. The molecules and particles absorb energy at discrete wavelengths, which are determined by their internal properties. For example, molecular oxygen and ozone absorbs light in the ultra violet spectrum. Water vapor, methane, nitrous oxide, ozone, and CO_2 absorb light in the infrared range.

In addition to absorption, molecules and particles also scatter energy out from its original direction. Sun's white light is scattered once (primary scattering) or multiple times (secondary scattering) into the viewing ray as shown in Figure 1. The scattered light is received at the earth's surface from all directions as diffuse skylight or daylight.

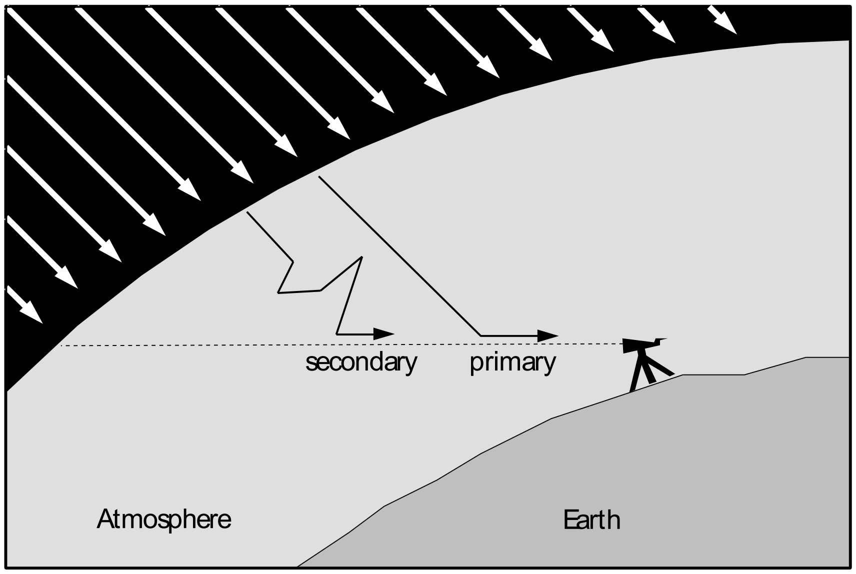


Figure 1: Scattering of sunlight in atmosphere.

Earth's surface is not flat and this plays a very important role in atmospheric optics. *Optical mass* of a path is defined as the mass of the medium in that path of unit cross-section and is given by

$$m = \int_0^s \rho(x)dx,$$

where *$\rho(x)$* is the density of medium. *Optical length* for a path is defined as the optical mass divided by the density at the earth's surface ρ_0 and is given by

$$l = \frac{1}{\rho_0}\int_0^s \rho(x)dx$$

Optical length has dimensions of length. The optical length in the zenith direction for molecules is 8.4 km and for aerosols is 1.25 km. Figure 2 shows the optical lengths for different directions in the atmosphere.

The relative optical length is defined as the ratio of the optical length of any path to optical length at zenith direction and is given by following approximation [Iqbal1983]

$$\frac{l(\theta_s)}{l_{zenith}} = \frac{1}{\cos\theta_s + 0.15(93.885 - \theta_s)^{-1.253}},$$

where θ_s is the angle from zenith in degrees.

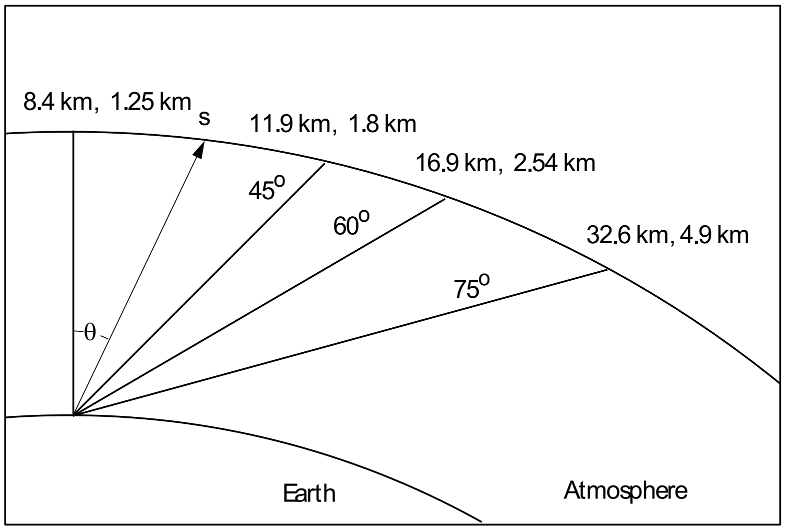


Figure 2: Optical lengths for molecules and aerosols for different paths in the atmosphere.

The sun's light travels through a much larger atmosphere when it is close to the horizon. Therefore, a larger amount of blue light is scattered away causing the sun to appear orange-red. Increased atmosphere presence in the horizon direction makes the stars appear brighter in the zenith direction compared to the horizon direction.

Scattering

Scattering is a process by which a particle redistributes a fraction of the incident energy into a total solid angle. The scattering properties depend on the refractive index and size of the particles.

It is common for smaller particles to scatter uniformly in the forward and backward directions and for larger particles to scatter strongly in the forward directions. Scattering by one particle is independent of the other as long as the distance between the particles are greater than the particle size. This is known as *independent scattering*. A scattering event for the first time is known as first order scattering. Scattered light may be scattered again by another particle and is said to undergo second order scattering. In reality, sun's light is scattered multiple times in the atmosphere.

The amount of scattering is linearly proportional to the density of the atmosphere, which varies with altitude. A choice of an analytic exponential model or a lookup table based on US Standard Atmospheres for density is available for use.

Rayleigh scattering

Particles smaller than the wavelength and usually less than 0.1 times the wavelength of light exhibit Rayleigh scattering [Rayleigh1871]. Discovered by the Nobel Prize winner Lord Rayleigh, Rayleigh scattering is observed by molecules in the earth's atmosphere. The amount of scattering for such particles is inversely proportional to the 4th power of the wavelength. These particles scatter equally in the forward and backward directions. The total scattering coefficient β and the angular scattering coefficient $\beta(\theta)$ are given by

$$\beta = \frac{8\pi^3(n^2-1)^2}{3N\lambda^4}(\frac{6+3p_n}{6-7p_n}),$$

$$\beta(\theta) = \frac{\pi^2(n^2-1)^2}{2N\lambda^4}(\frac{6+3p_n}{6-7p_n})(1+\cos^2\theta),$$

where n is the refractive index of air and is 1.0003 in the visible spectrum, N is the number of molecules per unit volume and is 2.545x10^{25} for air at standard temperature and pressure, p_n is the depolarization factor with a value of 0.0035 standard for air. The total scattering coefficient for blue light (400 nm) is 2.44x10^{-5}m^{-1}, for green light (530 nm) is 1.18x10^{-5}m^{-1} and for red light (700 nm) is 6.95x10^{-6}m^{-1}. This means that the blue light is scattered more than the red light, which explains the blue color of the sky and the red color of the sun at low altitudes.

Angular scattering coefficient is also equivalent to the total scattering coefficient times the phase function. The phase function for Rayleigh scattering $f_{air}(\theta)$ is given by

$$f_{air}(\theta) = \frac{3}{16\pi}(1+\cos^2\theta)$$

Mie Scattering

Larger particles scatter strongly in the forward direction and this scattering phenomenon is called Mie scattering named after Gustav Mie. The scattering is inversely proportional to the second order of the size of the particles and is independent of wavelength. The phase function for angular scattering was approximated by Henyey-Greenstein and is given by the following equation [Henyey1941].

$$f_{HG}(\theta) = \frac{1}{4\pi}\frac{1-g^2}{\left(1-2g\cos\theta+g^2\right)^{3/2}}.$$

Positive values of g represent forward scattering and negative values of g represent backward scattering. The total scattering coefficient is given by

$$\beta = 0.434c\pi(\frac{2\pi}{\lambda})^{v-2}K,$$

where c is the concentration factor and varies around 6x10^{-17} to 25x10^{-17} as the turbidity increases from clear to overcast, v is the Junge's exponent and a value of 4 is standard for sky model, and K varies from 0.656 for 400 nm to 0.69 for 770 nm. For more details, see [Bullrich1964].

Skylight Models

The atmosphere scatters sun's light multiple times and scattered light is received at the earth's surface in all directions collectively known as skylight.

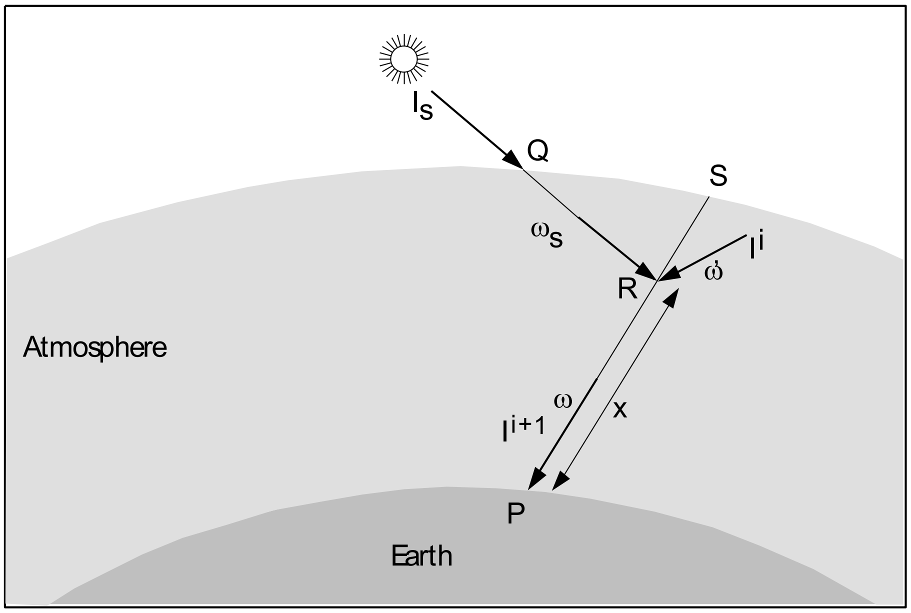


Figure 3: The variables used in skylight computation.

In Figure 3, we would like to compute the skylight in the viewing direction (ω) PS. R is a variable point on PS and RQ is the direction (ω_s) of sunlight. Points Q and S are at the top of the atmosphere. Light from the sun is attenuated as it travels the distance QR, is scattered by particles at R in the direction RP, and then attenuated as it travels a distance RP. Let l_{AB} denote the optical length from A to B. The differential amount of light reaching point P through the path QRP, scattered by a differential volume R is given by

$$dI = E_s e^{-\beta l_{QR}} \beta(\omega, \omega_s) e^{-\beta l_{RP}} dx\,,$$

where E_s is the irradiance of the sun outside the earth's atmosphere, β is the total scattering coefficient, $\beta(\omega,\omega_s)$ is the angular scattering coefficient between directions ω and ω_s and dx is the differential optical length. The total light received at P from all points R on PS is given by

$$I^1(\omega) = \int_P^S E_s e^{-\beta l_{QR}} \beta(\omega, \omega_s) e^{-\beta l_{RP}} dx\,.$$

Superscript "1" on $I(\omega)$ is used to indicate first order scattered light. Light from the sun is scattered more than once before reaching the earth's surface. Let $I^i(\omega')$ be the light reaching point R from direction ω' after being scattered i times. Light scattered into viewing direction ω from all directions ω' at point R is denoted by $S(\omega,x)$ and is given by the integral

$$S(\omega, x) = \int_0^{4\pi} I^i(\omega')\beta(\omega, \omega')d\omega'$$

The light scattered in direction ω at R is attenuated as it travels the distance RP through the atmosphere. Light received at earth's atmosphere after i+1 scattering events is given by

$$I^{i+1}(\omega) = \int_P^S e^{-\beta l_{RP}} S(\omega, x) dx$$

$$I^{i+1}(\omega) = \int_P^S e^{-\beta l_{RP}} \int_0^{4\pi} I^i(\omega')\beta(\omega,\omega')d\omega' dx$$

Starting from the equations for $I^1(\omega)$ and $I^{i+1}(\omega)$ in terms of $I^i(\omega)$, one can compute $I^2(\omega)$, $I^3(\omega)$ and so on, where $I^2(\omega)$ and $I^3(\omega)$ is the second and third order scattered light into viewing direction ω. Total light scattered into the viewing ray $I(\omega)$ is the sum of first order, second order and higher orders of scattered light and is given by

$$I(\omega) = I^1(\omega) + I^2(\omega) + I^3(\omega) + \cdots$$

The above equations assume only one kind of particles in the atmosphere. Real atmosphere consists a number of different kinds of particles, of most importance being molecules and aerosols. The above equations can be extended to multiple particles and is left as an exercise for the reader.

Simulation Based Methods

All simulation methods are based on the general atmosphere equations presented above. The variations arise from use of different models for atmosphere density, different atmosphere composition, scattering coefficients etc.

One of the first models for atmospheric scattering was presented by Klassen [Klassen1987] which is a must read for anyone interested in modeling atmospheric scattering. He used a simple constant density atmosphere model on a flat earth surface. Flat earth model performs poorly for skylight computations on viewing rays close to the horizon. For aerial perspective discussed later, where distances viewed are usually of the order of a few tens of kilometers, the flat earth model is a good approximation.

Kaneda et al employed similar concepts to Klassen and used a more realistic atmospheric model for his simulations [Kaneda1991]. He modeled a spherical earth with an exponential decay density distribution.

Nishita et al [Nishita1996] take a step closer to reality and model higher orders of scattering, which is responsible for the whitening effect of the sky. He proposes a fast method for single scattering computations.

Analytic Models and Approximations

Simulation based methods are computationally very expensive making it unusable for all practical purposes. Analytic or approximate models for skies that represent simulated data or real world atmospheric data collected over the years is preferable for ease of use in computer graphics. Of all real skies, the two extremes are clear and overcast skies.

CIE Overcast sky luminance model

An overcast sky is equivalent to a dark sky with plenty of clouds. Moon and Spencer proposed a luminance model for overcast skies. This was later simplified and adopted by the International Commission on Illumination - abbreviated as CIE from its French title Commission Internationale de l'Eclairage [CIE1994]. The overcast sky luminance Y_{OC} is given by

$$Y_{OC} = Y_z \frac{1 + 2\cos\theta}{3},$$

where θ is the angle from the zenith and Y_z is the zenith for overcast skies and can be obtained from analytic formulas adopted by CIE [CIE1994].

CIE Clear sky luminance model

Pokrowski proposed a sky luminance model from theory and sky measurements for clear sky. Kittler improved this clear sky model, which was later adopted by the CIE for the clear sky model in 1973 [CIE1994]. The clear sky luminance Y_C is given by

$$Y_C = Y_z \frac{(0.91 + 10e^{-3\gamma} + 0.45\cos^2\gamma)(1 - e^{-0.32/\cos\theta})}{(0.91 + 10e^{-3\theta_s} + 0.45\cos^2\theta_s)(1 - e^{-0.32})},$$

where Y_z is the zenith luminance and the angles are defined in Figure 4.
The CIE luminance models provide us with relative luminance and not absolute luminance.

ASRC-CIE model luminance model

ASRC-CIE model is a linear combination of four luminance models - the standard CIE cloudless sky, a high turbidity formulation of the latter, a realistic formulation for intermediate skies proposed by Nakamura and the standard CIE overcast sky. The sky clearness and sky brightness factors are used to determine the weights for these four skies [Littlefair1994].

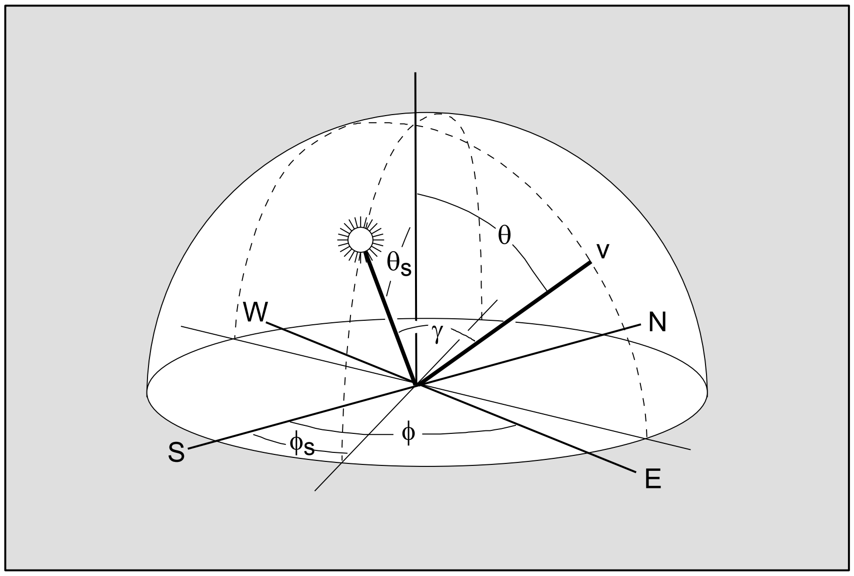


Figure 4: Angles and directions on the sky dome [Preetham1999].

Perez all weather luminance model

Perez et al proposed an all weather sky luminance model [Perez1993]. This model was based on 5 different parameters, which related to darkening or brightening of the horizon, luminance gradient near the horizon, relative intensity of the circumsolar region, width of the circumsolar region and relative backscattered light. The Perez model is given by

$$F(\theta,\gamma) = (1 + Ae^{B/\cos\theta})(1 + Ce^{D\gamma} + E\cos^2\gamma),$$

where A, B, C, D and E are the distribution coefficients and γ and θ are the angles shown in Figure 4. The Perez model sky luminance Y_P is given by

$$Y_P = Y_z \frac{F(\theta,\gamma)}{F(0,\theta_s)}.$$

Perez model is similar to CIE model and has been found to be more accurate and has also been validated by Ineichen [Ineichen1994].

Preetham et al spectral radiance model

Similar to the Perez's all weather luminance model, Preetham et al proposed an analytical model for spectral radiance of sky [Preetham1999]. This model shows variation from clear to overcast sky through a parameter called *turbidity*. Analytic model was arrived from atmospheric simulations using US Standard Atmospheres, and modeling skylight up to second order scattering. Luminance Y and chromaticity values x and y are given by

$$Y = Y_z \frac{F(\theta,\gamma)}{F(0,\theta_s)},$$

$$x = x_z \frac{F(\theta,\gamma)}{F(0,\theta_s)}, \text{ and}$$

$$y = y_z \frac{F(\theta,\gamma)}{F(0,\theta_s)},$$

where $F(\theta, \gamma)$ is from Perez's model with different values of A, B, C, D and E for Y, x and y. The distribution coefficients for luminance and chromaticity values x and y, absolute values for zenith luminance Y_z, zenith chromaticity x_z and y_z are all given in [Preetham1999]. Conversion of chromaticity values into spectral radiance is given by the following standard method for D-illuminants [Wyszecki1982]. The relative spectral radiant power $S_D(\lambda)$ is given by

$$S_D(\lambda) = S_0(\lambda) + M_1 S_1(\lambda) + M_2 S_2(\lambda),$$

where $S_0(\lambda)$ is the mean spectral radiant power and $S_1(\lambda)$ and $S_2(\lambda)$ are the first two eigen vector functions used in daylight illuminants. M_1 and M_2 are functions of chromaticity values x and y and are given by

$$M_1 = \frac{-1.3515 - 1.7703x + 5.9114y}{0.0241 + 0.2562x - 0.7341y},$$

$$M_2 = \frac{0.0300 - 31.4424x + 30.0717y}{0.0241 + 0.2562x - 0.7341y}.$$

A scene rendered using this model is shown in Figure 5.

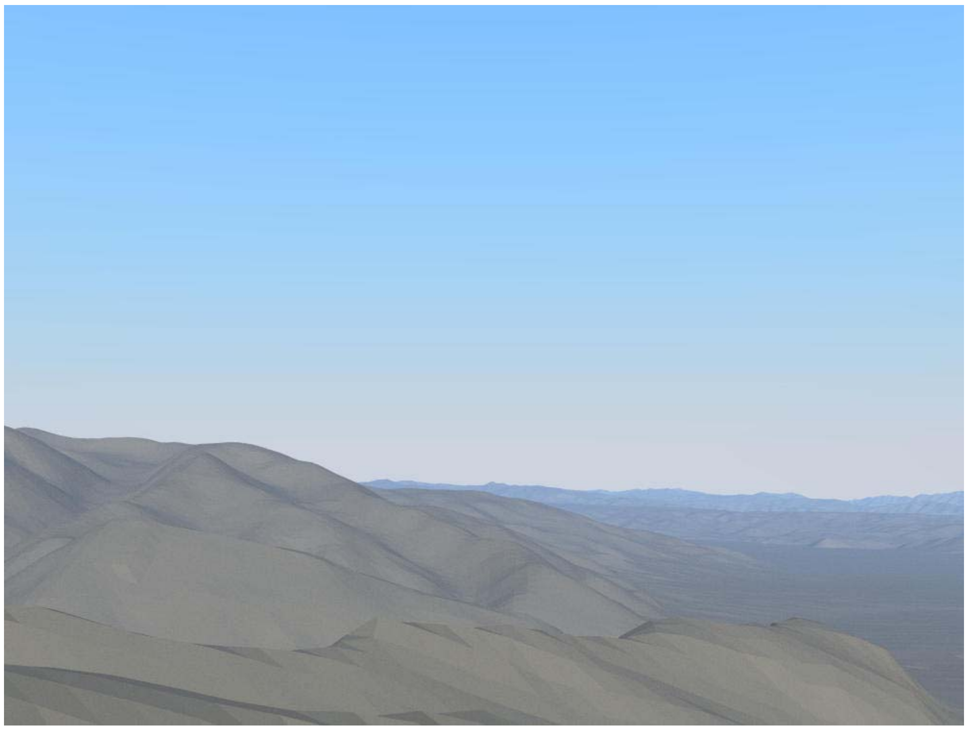


Figure 5: Spectral radiance model for sky [Preetham1999].

© Copyright 1999 by ACM, Inc.

Aerial Perspective Model

Many artists have recorded the effect of distant mountains fading away in history. In the presence of a medium such as atmosphere distant objects appear blue. This shift in color helps us in perceiving depth and is known as *aerial perspective*.

Distant objects appear hazier and this is attributed to scattering and absorption along the viewing ray as light travels from the source to the viewer. The light from the source loses intensity and undergoes a spectral shift as scattering and absorption depend on wavelength. In addition to this loss of light, light from other sources like sun, sky and ground are scattered into the viewing ray.

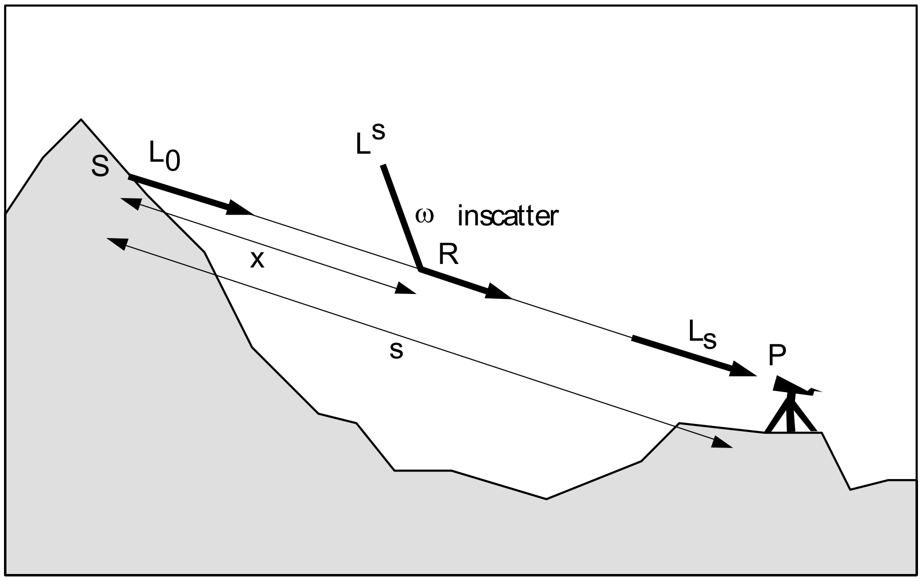


Figure 6: The variables in aerial perspective computation.

In Figure 6, L_0 is the radiance of the distant hill and L_s is the radiance of the ray at the viewer. If f is the extinction factor and L_{in} is the in-scattered light as L_0 travels a distance s to the eye, then

$$L_s = fL_0 + L_{in}$$

The extinction factor f for light traveling the path PS with optical length l_{PS} is given by

$$f = e^{-\beta l_{PS}}$$

Let $L^s(\omega)$ denote the spectral radiance of the sun and sky in the direction ω. Let $S(\omega, x)$ be the term that denotes the light scattered from direction ω into the viewing direction at point R. Therefore,

$$S(\omega, x) = \int_0^{4\pi} L^s(\omega')\beta(\omega,\omega')d\omega'$$

The total light scattered into viewing direction at point R is

$$L_{in} = \int_P^S e^{-\beta l_{RP}} S(\omega, x)dx\,,$$

$$L_{in} = \int_P^S e^{-\beta l_{RP}} \int_0^{4\pi} L^s(\omega')\beta(\omega,\omega')d\omega' dx\,.$$

For landscape scenes that focus on aerial perspective, the viewing rays are close to the earth's surface and it can safely be assumed that the density of the medium is a constant and is equal to that at the earth's surface. For such rays, the optical length l_{AB} is equal to the distance AB. Therefore, l_{PS} is equal to s, l_{RP} is equal to (s-x) and the above equations simplify to

$$f = e^{-\beta s} \text{ and}$$

$$L_{in} = \int_0^s e^{-\beta(s-x)} \int_0^{4\pi} L^s(\omega')\beta(\omega,\omega')d\omega' dx\,.$$

The primary contribution to in-scattered light is the direct light from the sun. Therefore, we can safely ignore second order scattering i.e. the light from the sky without loss of quality. Therefore, if E^s is the irradiance from the sun at the earth's surface and ω^s is the sun's direction, then

$$\int_0^{4\pi} L^s(\omega')\beta(\omega,\omega')d\omega' = E^s\beta(\omega,\omega^s)\,.$$

And the total in-scattered light simplifies to

$$L_{in} = \int_0^s e^{-\beta(s-x)} E^s\beta(\omega,\omega^s)dx\,,$$

$$L_{in} = E^s\frac{\beta(\omega,\omega^s)}{\beta}(1-e^{-\beta s})\,.$$

While the above equations for aerial perspective are valid for scattering by one kind of particles, they can easily be extended to many kinds of particles, and is left as an exercise for the reader.

Accelerated Techniques for Modeling Sky and Aerial Perspective

Accurate calculation of the sky color is expensive because of numerical integration of scattered light along the viewing ray. Modeling higher orders of scattering results in double and triple integration and the computational cost increases exponentially. Dobashi et al discuss various ways to represent and evaluate the sky intensity across the hemispherical domain for various altitudes of the sun [Dobashi1995]. A simple and naive approach would be to represent the sky dome by a mesh grid of sample points. Skylight intensity at any point on the hemisphere can be obtained by linearly interpolating between these sampled points. Any function can be represented by the summation of a set of basis functions and another approach to representing the sky function over the hemisphere is to use spherical harmonics. The weights to these basis functions can be calculated in a preprocess step.

Dobashi et al proposed another set of basis functions called the cosine functions, which use less memory than the spherical basis functions and is faster to evaluate. With recent advances in the area of programmable graphics processors, these cosine functions can be evaluated in real time in shaders.

Researchers have constantly looked into using graphics hardware to accelerate modeling of sky and aerial perspective [Dobashi2000][Dobashi2002][Hoffman2002].

Rendering shafts of light through clouds has been done in CPU by numerical integration of in-scattered light along the viewing ray taking into account the visibility of the sample points. The numerical integration can be approximated by a summation of terms along the viewing ray. Dobashi et al achieved this summation by rendering many virtual planes along the viewing ray and accumulating the various terms using blending [Dobashi2000]. The visibility information at any virtual plane is calculated using the standard shadow map technique [Williams1978]. The light information at the virtual plane is evaluated using projective light textures technique. By careful choice of the number of virtual planes and the resolution of the mesh for the virtual plane, photo realistic scenes with atmospheric scattering can be generated at interactive frame rates.

Dobashi et al accelerated their previous techniques of rendering a sequence of virtual or sampling planes and accumulating the terms using blending [Dobashi2002]. The novelty in aerial perspective is that the summation is written as a product of a high frequency term and a low frequency term. The low frequency term can be evaluated accurately in a preprocess step and is written as a product of two terms; one term is stored in the vertices of the mesh grid of the sampling plane and the other is stored as a texture map. A sequence of planes is drawn with blending to get the final image. A rendering from their publication is shown in Figure 7.

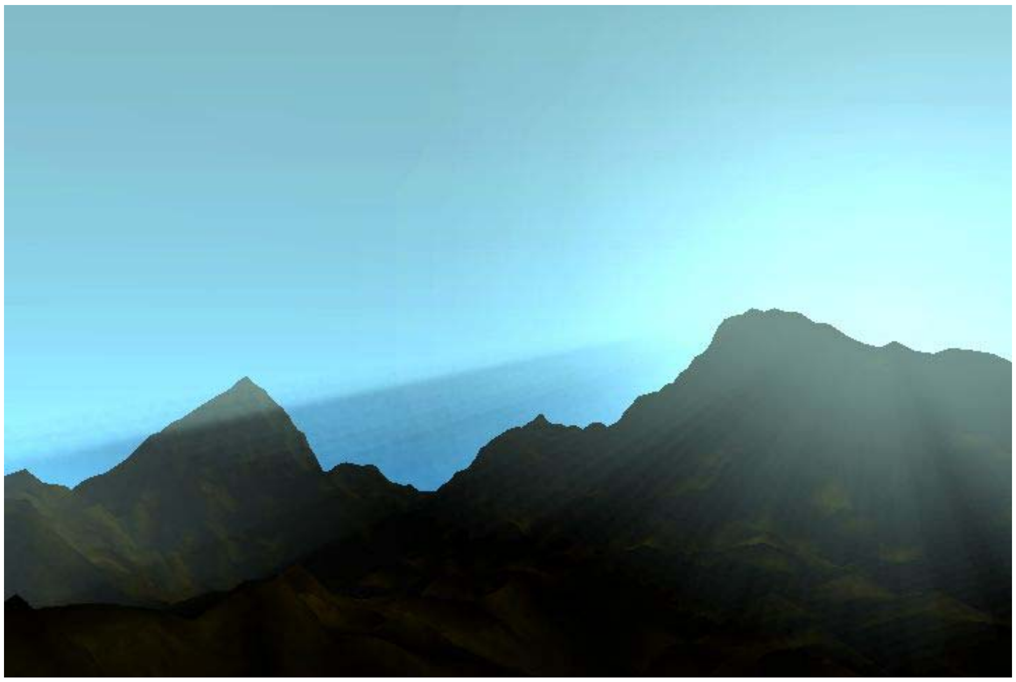


Figure 7: Interactive atmospheric scattering [Dobashi2002].

© Copyright 2002 by Eurographics. Included by permission.

Modern graphics hardware (e.g. Radeon 8500, Radeon 9700, GeForce 4, GeForce FX) allows the user to specify a vertex shader for transformation and lighting. Hoffman et al presented a new technique to render sky color and aerial perspective in real time using programmable graphics hardware [Hoffman2002]. The two terms for aerial perspective are extinction factor f and in-scattered light L_{in}. These are extended to two kinds of particles - molecules and aerosols. f and L_{in} are calculated in the vertex shader on a per vertex basis and is passed on to the fragment program through the color or texture registers. The final color is computed in the fragment program using the equation

$$L_s = fL_0 + L_{in}\,.$$

With the latest generation of graphics hardware boasting of a full floating point pipeline (e.g. Radeon 9700, Geforce FX), the entire computation of aerial perspective terms (f and L_{in}) can be done at a pixel level rather than a vertex level as was presented. The advantage of a per pixel computation is that it does not require very fine tessellation of the geometry to avoid artifacts due to linear interpolation.

The image in Figure 8 was rendered on 600MHz Pentium III with a Radeon 8500 at about 60 frames per second.

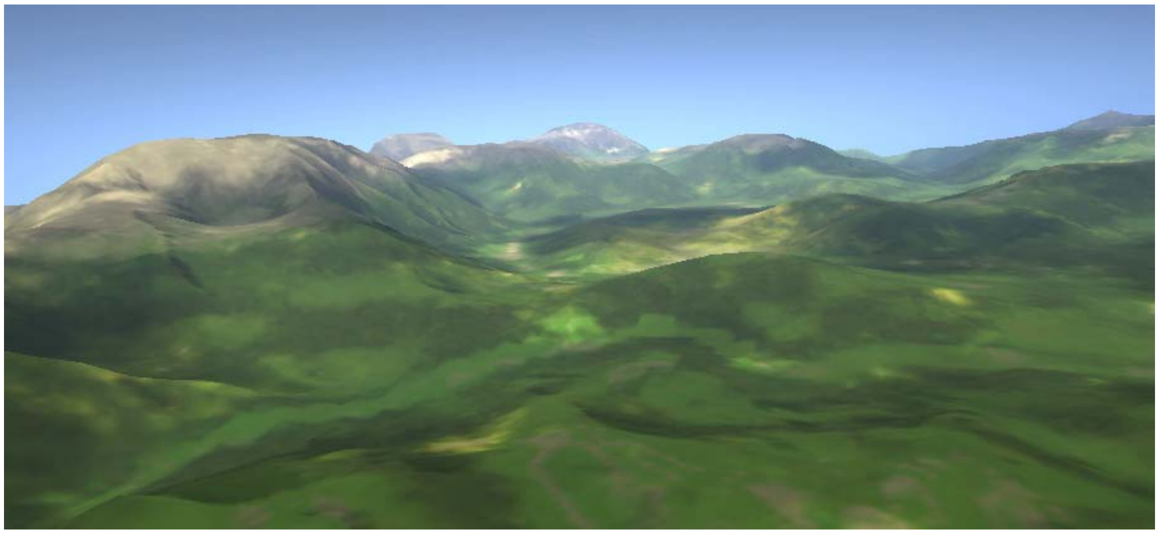


Figure 8: Real time atmospheric scattering [Hoffman2002].

References

[Bullrich1964] Bullrich, K. Scattered radiation in the atmosphere. In *Advances in Geophysics,* 10, Academic Press, 1964.

[CIE1994] CIE-110-1994. Spatial distribution of daylight – luminance distributions of various reference skies. Tech. Rep., International Commission on Illumination, 1994.

[Coulson1975] Coulson, K.L. *Solar and terrestrial radiation.* Academic Press, 1975.

[Dobashi1995] Dobashi, Y., Nishita, T., Kaneda, K., Yamashita, H. Fast display method of sky color using basis functions. In *Pacific Graphics,*1995.

[Dobashi2000] Dobashi, Y., Yamamoto, T., Nishita, T. Interactive rendering method for displaying shafts of light, In *Pacific Graphics,* 2000.

[Dobashi2002] Dobashi, Y., Yamamoto, T., Nishita, T. Interactive rendering of atmospheric scattering effects using graphics hardware. *Proceedings of the conference on Graphics hardware,* 2002.

[Hoffman2002] Hoffman, N., Preetham, A.J. Rendering outdoor light scattering in real time. *Game Developers Conference*, 2002.

[Henyey1941] Henyey, L.G., and Greenstein, J.L. Diffuse reflection in the Galaxy. *Astrophysical Journal 93:70*, 1941.

[Ineichen1994] Ineichen, P., Molineaux, B., and Perez, R. Sky luminance data validation: comparison of seven models with four data banks. *Solar Energy 52*, 4, 337-346, 1994.

[Iqbal1983] Iqbal, M. *An introduction to Solar Radiation*. Academic Press, 1983.

[Kaneda1991] Kaneda, K., Okamoto, T., Nakame, E., and Nishita, T. Photorealistic image synthesis for outdoor scenery under various atmospheric conditions. *The Visual Computer* 7, 5 and 6, 247-258, 1991.

[Klassen1987] Klassen, R.V. Modeling the effect of the atmosphere on light. *ACM Transactions on Graphics* 6, 3, 215-237, 1987.

[Littlefair1994] Littlefair, P.J. A comparison of sky luminance models with measure data from Garston, United Kingdom. *Solar Energy*, 53, 4, 315-322, 1994.

[Lynch1995] Lynch, D.K., and Livingston, W. *Color and light in nature,* Cambridge University Press, 1995.

[McCartney1976] McCartney, E.J. *Optics of the atmosphere*. Wiley publication, 1976.

[Minnaert1954] Minnaert, M. *Light and color in the open air*. Dover, 1954.

[Nishita1996] Nishita, T., Dobashi, Y., Kaneda, K., and Yamashita, H. Display method of the sky color taking into account multiple scattering. In *Pacific Graphics*, pp. 117-132, 1996.

[Preetham1999] Preetham, A.J., Shirley, P., Smits, B.E. A practical analytical model for daylight. In *Proceedings of Siggraph*, 91-100, 1999.

[Robinson1966] Robinson, N. *Solar radiation*. Elsevier Publishing Company, 1966.

[Rayleigh1871] Rayleigh, L. On the scattering of light by small particles. *Philosophical Magazine 41*, 447-451, 1871.

[VanDeHulst1981] van De Hulst, H.C. Light scattering by small particles. *Dover Publications*, 1981.

[Williams1978] Williams, L. Casting curved shadows on curved surfaces. In *Proceedings of Siggraph*, 270-274, 1978.

[Wyszecki1982] Wyszecki, G., and Stiles, W.S. *Color Science*. Wiley-Interscience publication, 1982.

A Physically-Based Night Sky Model

Henrik Wann Jensen[1] Frédo Durand[2] Michael M. Stark[3] Simon Premože[3]
Julie Dorsey[2] Peter Shirley[3]

[1]Stanford University [2]Massachusetts Institute of Technology [3]University of Utah

Abstract

This paper presents a physically-based model of the night sky for realistic image synthesis. We model both the direct appearance of the night sky and the illumination coming from the Moon, the stars, the zodiacal light, and the atmosphere. To accurately predict the appearance of night scenes we use physically-based astronomical data, both for position and radiometry. The Moon is simulated as a geometric model illuminated by the Sun, using recently measured elevation and albedo maps, as well as a specialized BRDF. For visible stars, we include the position, magnitude, and temperature of the star, while for the Milky Way and other nebulae we use a processed photograph. Zodiacal light due to scattering in the dust covering the solar system, galactic light, and airglow due to light emission of the atmosphere are simulated from measured data. We couple these components with an accurate simulation of the atmosphere. To demonstrate our model, we show a variety of night scenes rendered with a Monte Carlo ray tracer.

Keywords: Natural Phenomena, Atmospheric Effects, Illumination, Rendering, Ray Tracing

1 Introduction

In this paper, we present a physically-based model of the night sky for image synthesis, and demonstrate it in the context of a Monte Carlo ray tracer. Our model includes the appearance and illumination of all significant sources of natural light in the night sky, except for rare or unpredictable phenomena such as aurora, comets, and novas.

The ability to render accurately the appearance of and illumination from the night sky has a wide range of existing and potential applications, including film, planetarium shows, drive and flight simulators, and games. In addition, the night sky as a natural phenomenon of substantial visual interest is worthy of study simply for its intrinsic beauty.

While the rendering of scenes illuminated by daylight has been an active area of research in computer graphics for many years, the simulation of scenes illuminated by nightlight has received relatively little attention. Given the remarkable character and ambiance of naturally illuminated night scenes and their prominent role in the history of image making — including painting, photography, and cinematography — this represents a significant gap in the area of realistic rendering.

1.1 Related Work

Several researchers have examined similar issues of appearance and illumination for the daylight sky [6, 19, 30, 31, 34, 42]. To our knowledge, this is the first computer graphics paper that describes a general simulation of the nighttime sky. Although daytime and nighttime simulations share many common features, particularly

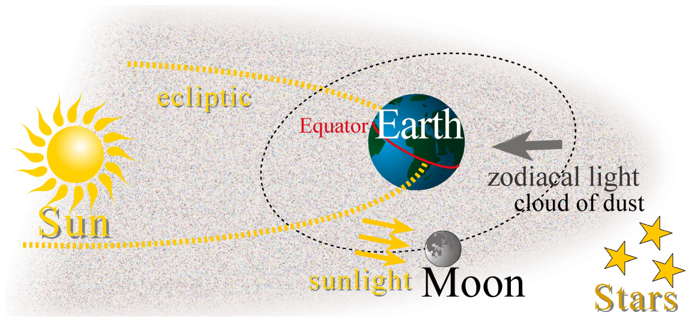


Figure 1: Elements of the night sky. Not to scale.

the scattering of light from the Sun and Moon, the dimmer nighttime sky reveals many astronomical features that are invisible in daytime and thus ignored in previous work. Another issue unique to nighttime is that the main source of illumination, the Moon, has its own complex appearance and thus raises issues the Sun does not. Accurately computing absolute radiances is even more important for night scenes than day scenes. If all intensities in a day scene are doubled, a tone-mapped image will change little because contrasts do not change. In a night scene, however, many image features may be near the human visibility threshold, and doubling them would move them from invisible to visible.

Some aspects of the night sky have been examined in isolation. In a recent paper, Oberschelp and Hornug provided diagrammatic visualizations of eclipses and planetary conjunction events [32]. Their focus is the illustration of these events, not their realistic rendering. Researchers have created detailed models of the appearance of Saturn [3, 4] and Jupiter [47]. These models were intended for simulation of space scenes and would be overkill for renderings of Earth scenes. Baranoski et al. did a careful simulation of the aurora borealis [1]. Although we do not simulate aurora phenomena in our model, the techniques of Baranoski et al. could be added seamlessly to our simulations because aurora are an emission phenomenon with little correlation to the Earth's position relative to the Sun, and thus can be added independently to other nighttime effects.

1.2 Overview

The next section introduces the components of our model. Section 3 describes astronomical models to compute the accurate positions of the elements of the night sky in a framework appropriate for use in computer graphics. Sections 4–6 introduce our approach for modeling and rendering the key sources of illumination in the night sky. We discuss our implementation and results in Section 7. Finally we conclude in Section 8 with some discussion and directions for future work.

2 Sources of Night Illumination

To create realistic images of night scenes, we must model the characteristics of nighttime illumination sources, both in terms of their contribution to the scene, and their direct appearance in the sky. These sources are illustrated in Figures 1 and 2 and summarized below.

- **The Moon.** Most of the visible moonlight is actually sunlight, incident on the Moon and scattered from its surface in all directions. Light received directly from the Moon and moonlight scattered by the atmosphere account for most of the available light at night. The appearance of the Moon itself is important in the night sky due to its proximity to and visibility from the Earth.

Component	Irradiance $[W/m^2]$
Sunlight	$1.3 \cdot 10^{3}$
Full Moon	$2.1 \cdot 10^{-3}$
Bright planets	$2.0 \cdot 10^{-6}$
Zodiacal light	$1.2 \cdot 10^{-7}$
Integrated starlight	$3.0 \cdot 10^{-8}$
Airglow	$5.1 \cdot 10^{-8}$
Diffuse galactic light	$9.1 \cdot 10^{-9}$
Cosmic light	$9.1 \cdot 10^{-10}$

Figure 2: Typical values for sources of natural illumination at night.

- **The Sun.** The sunlight scattered around the edge of the Earth makes a visible contribution at night. During "astronomical twilight" the sky is still noticeably bright. This is especially important at latitudes greater than 48° N or S, where astronomical twilight lasts all night in midsummer.

- **The planets and stars.** Although the light received from the stars is important as an illumination source only on moonless nights, the appearance of stars in the sky is crucial for night scenes. The illumination and appearance of the other planets are comparable to that of bright stars.

- **Zodiacal light.** The solar system contains dust particles that scatter sunlight toward the Earth. This light changes the appearance and the illumination of the night sky.

- **Airglow.** The atmosphere has an intrinsic emission of visible light due to photochemical luminescence from atoms and molecules in the ionosphere. This accounts for one sixth of the light in the moonless night sky.

- **Diffuse galactic and cosmic light.** Light from galaxies other than the Milky Way.

The atmosphere also plays an important role in the appearance of the night sky. It scatters and absorbs light and is responsible for a significant amount of indirect illumination.

In general, the above sources cannot be observed simultaneously in the night sky. In particular, the dimmest phenomena can only be seen on moonless nights. In fact, the various components of night light are only indirectly related to one another; hence, we treat them separately in our model.

2.1 Model Overview

The main components of our model are illustrated Figure 3 and outlined below. Subsequent sections will discuss each of these components in greater detail.

Our general approach is to model the direct appearance of the celestial elements and the illumination they produce differently, as the latter requires less accuracy and a simpler model is easier to integrate and introduces less variance. We demonstrate our model in a spectral rendering context; however the data can be converted to the CIE XYZV color space (including a scotopic component V for rod vision).

- **Astronomical positions.** We summarize classical astronomical models and provide a simplified framework to compute accurate positions of the Sun, Moon, and stars.

- **Moon.** The Moon is simulated as explicit geometry illuminated by two directional light sources, the Sun and the Earth. We include a model based on elevation and albedo data of the surface of the Moon, and on a specialized BRDF description. A simpler model is presented for the illumination from the Moon.

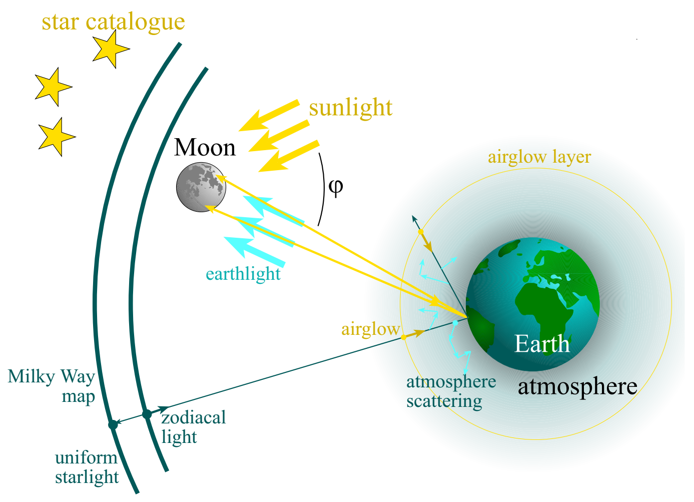


Figure 3: Components of the night sky model.

- **Stars.** The appearance of the brightest stars is simulated using data that takes into account their individual position, magnitude and temperature. Planets are displayed similarly, albeit by first computing their positions. Star clouds — elements that are too dim to observe as individual stars but collectively produce visible light — including the Milky Way, are simulated using a high resolution photograph of the night sky processed to remove the bright stars. Illumination from stars is treated differently, using a simple constant model.

- **Other astronomical elements.** Zodiacal light, atmospheric airglow, diffuse galactic light, and cosmic lights are simulated using measured data.

- **Atmospheric scattering.** We simulate multiple scattering in the atmosphere due to both molecules and aerosols.

3 Astronomical Positions

We introduce classical astronomical formulas to compute the position of various celestial elements. For this, we need to review astronomical coordinate systems. This section makes simplifications to make this material more accessible. For a more detailed description, we refer the reader to classic textbooks [7, 10, 23, 24] or to the year's *Astronomical Almanac* [44].

All of the formulas in the Appendix have been adapted from high accuracy astronomical formulas [23]. They have been simplified to facilitate subsequent implementation, as computer graphics applications usually do not require the same accuracy as astronomical applications. We have also made conversions to units that are more familiar to the computer graphics community. Our implementation usually uses higher precision formulas, but the error introduced by the simplifications is lower than 8 minutes of arc over five centuries.

3.1 Coordinate Systems

The basic idea of positional astronomy is to project everything onto *celestial spheres*. Celestial coordinates are then spherical coordinates analogous to the terrestrial coordinates of longitude and latitude. We will use three coordinate systems (Figure 4) — two centered on the Earth (equatorial and ecliptic) and one centered on the observer (horizon). Conversion formulas are given in the Appendix.

The final coordinate system is the local spherical frame of the observer (horizon system). The vertical axis is the local zenith, the

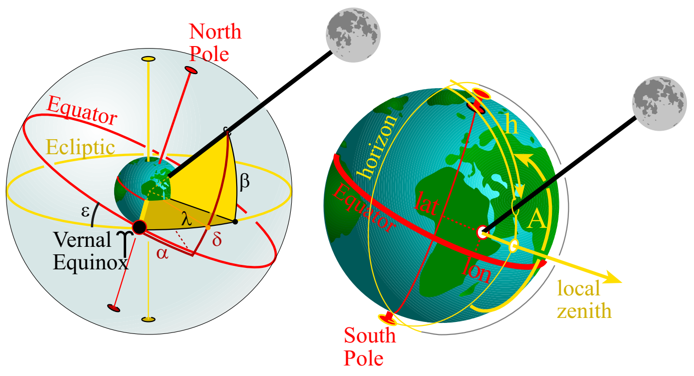


Figure 4: Coordinate systems. Left: equatorial (α, δ) and ecliptic (λ, β). Right: Local coordinates (A, h) for an observer at longitude lon and latitude lat.

latitude is the altitude angle h above the horizon, and the longitude, or azimuth A, is measured eastward from the south direction (Figure 4).

The two other systems are centered on the Earth but do not depend on its rotation. They differ by their vertical axis, either the North pole (and thus the axis of rotation of the Earth) for the equatorial system, or the normal to the ecliptic, the plane of the orbit of the Earth about the Sun, for the ecliptic system (Figures 1 and 4).

The Earth's rotational axis remains roughly parallel as the Earth orbits around the Sun (the angle between the ecliptic and the Equator is about $23.44°$). This means that the relationship between the angles of the two systems does not vary. In both cases, the reference for the longitude is the Vernal Equinox, denoted ♈, which corresponds to the intersection of the great circles of the Equator and the ecliptic (Figure 4).

However, the direction of the axis of the Earth is not quite constant. Long-term variations known as *precession* and short-term oscillations known as *nutations* have to be included for high accuracy, as described in the Appendix.

3.2 Position of the Sun

The positions of the Sun and Moon are computed in ecliptic coordinates (λ, β) and must be converted using the formulas in Appendix. We give a brief overview of the calculations involved. Formulas exhibit a mean value with corrective trigonometric terms (similar to Fourier series).

The ecliptic latitude of the Sun should by definition be $\beta_{\text{Sun}} = 0$. However, small corrective terms may be added for very high precision, but we omit them since they are below $10''$.

3.3 The Moon

The formula for the Moon is much more involved because of the perturbations caused by the Sun. It thus requires many corrective terms. An error of $1°$ in the Moon position corresponds to as much as 2 diameters. We also need to compute the distance d_{Moon}, which is on average 384 000 km. The orbital plane of the Moon is inclined by about $5°$ with respect to the ecliptic. This is why solar and lunar eclipses do not occur for each revolution and are therefore rare.

The Moon orbits around the Earth with the same rotational speed as it rotates about itself. This is why we always see nearly the same side. However, the orbit of the Moon is not a perfect circle but an ellipse (eccentricity about 1/18), and its axis of revolution is slightly tilted with respect to the plane of its orbit (Figure 5). For this reason, about 59% of the lunar surface can be seen from the Earth. These apparent oscillations are called *librations* and are

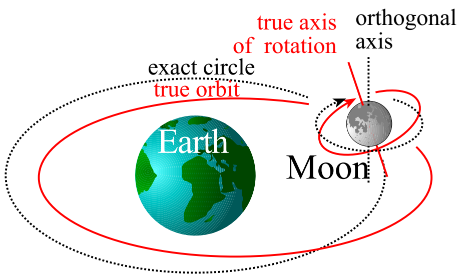


Figure 5: (a) Librations of the Moon are due to the eccentricity of its orbit and to the tilt of its axis of rotation. (b) Angle for the BRDF of the Moon.

included in our model as a result of our direct modeling approach. (See the Appendix for formulas.)

3.4 Position of Stars

For the stars, we used the Yale Bright Star Catalog [14]. It contains about 9000 stars, including the roughly 6000 visible to the naked eye. The stellar positions are given in equatorial coordinates. The catalogue contains the position for 2000 A.D. and the *proper motion* of stars, caused by their rectilinear motion through space. The position of a star for a given date is then computed using a linear approximation, usually in rectangular coordinates (the apparent motion can be neglected if the date is less than 5 centuries from 2000 A.D.). The positions of the planets are computed using formulas similar to those used for the Moon and the Sun [23].

4 Moonlight

The large-scale topography of the Moon is visible from the Earth's surface, so we render its direct appearance using a geometric model containing elevation and albedo illuminated by two directional light sources, the Earth and the Sun.

4.1 Modeling the Moon

To model the Moon accurately, we use the positions computed in the previous section and measured data of the lunar topography and albedo [27]. The Moon is considered a poor reflector: on average only 7.2% of the light is reflected [20]. The albedo is used to modulate a BRDF model, which we present in the next section. The elevation is measured with a precision of a quarter of a degree in longitude and latitude (1440×720), and the albedo map has size 800×400.

The Moon is illuminated by the Sun, which can be treated as a directional light source. We use the positions of the Moon and Sun to determine the direction of illumination. The Sun is modeled as a black body at temperature $5900K$ (see Appendix for conversion), and power $1905\frac{W}{m^2}$. The Sun appears about 1.44 times brighter from the Moon than from the surface of the Earth because of the absence of atmosphere. We do not include the Earth as an occluder, which means we cannot simulate lunar eclipses. This could easily be done by modeling the Sun as a spherical light source to simulate penumbra.

The faint light visible on the dark side of the Moon when it is a thin crescent is known as *earthshine*. Earthshine depends strongly on the phase of the Earth. When the Earth is full (at new Moon), it casts the greatest amount of light on the Moon, and the earthshine is relatively bright and easily observed by the naked eye. We model earthshine explicitly by including the Earth as a second light source for the Moon surface.

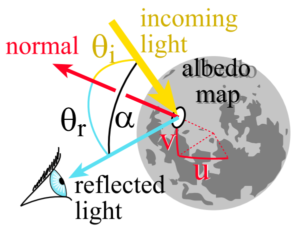


Figure 6: Angle for the BRDF of the Moon.

Accuracy is not crucial for earthshine except for the new Moon, so we simply use the percentage of lit Earth visible from the Moon and multiply it by the intensity of the full earthshine, which is $0.19\frac{W}{m^2}$. Given the Earth phase, that is, the angle $\pi - \varphi$ between the Moon and the Sun as seen from the Earth (the opposite of the Moon phase φ), we obtain [46]:

$$E_{em} = 0.19{*}0.5\left[1 - \sin(\frac{\pi-\varphi}{2})\tan(\frac{\pi-\varphi}{2})\ln(\cot(\frac{\pi-\varphi}{4}))\right] \quad (1)$$

4.2 BRDF of the Moon

The Moon has a low albedo and a reddish color, and it exhibits backscattering reflection (it is much brighter at full Moon) [11]. Furthermore, the apparent disc of the full Moon has a remarkable photometric property: its average brightness at the center is the same as at the edge [11]. The Moon is therefore said to exhibit no *limb darkening*. This can be explained by the pulverized nature of its surface. We use the complete Hapke-Lommel-Seeliger model of the reflectance function of the Moon, which provides a good approximation to the real appearance and a good fit to measured data [12]. The BRDF f consists of a retrodirective function B and a scattering function S. As the BRDF of the Moon is rather uniform, but the albedo is variable, we model them independently. We multiply the BRDF by the albedo and by the spectrum of the Moon.

The geometry for the BRDF is summarized in Figure 6. φ is the lunar phase angle (angle Sun-Earth as seen from the Moon, or equivalently for our purpose, the angle between incident and reflected light). θ_r is the angle between the reflected light and the surface normal; θ_i is the angle between the incident light and the surface normal.

Note that to compute the contribution of the earthshine, the Sun has to be replaced by the Earth using Equation 1 for the intensity. φ is then null by definition.

The BRDF, f, of the Moon can be computed with:

$$f(\theta_i, \theta_r, \varphi) = \frac{2}{3\pi} B(\varphi, g) S(\varphi) \frac{1}{1 + \cos\theta_r / \cos\theta_i}. \quad (2)$$

The retrodirective function $B(\varphi, g)$ is given by

$$B(\varphi, g) = \begin{cases} 2 - \frac{\tan\varphi}{2g}\left(1 - e^{-g/\tan\varphi}\right)\left(3 - e^{-g/\tan\varphi}\right), & \varphi < \pi/2 \\ 1, & \varphi \geq \pi/2, \end{cases} \quad (3)$$

where g is a surface density parameter which determines the sharpness of the peak at the full Moon. We use $g = 0.6$, although values between 0.4 (for rays) and 0.8 (for craters) could be used.

The scattering law S for individual objects is given by [12]:

$$S(\varphi) = \frac{\sin|\varphi| + (\pi - |\varphi|)\cos|\varphi|}{\pi} + t\left(1 - \frac{1}{2}\cos|\varphi|\right)^2, \quad (4)$$

where t introduces a small amount of forward scattering that arises from large particles that cause diffraction [37]. $t = 0.1$ is a good fit to Rougier's measurements [16] of the light from the Moon.

The spectrum of the Moon is distinctly redder than the Sun's spectrum. Indeed, the lunar surface consists of a layer of a porous pulverized material composed of particles larger than the wavelengths of visible light. As a consequence and in accordance with the Mie theory [45], the albedo is approximately twice as large for red (longer wavelengths) light than for blue (shorter wavelengths) light. In practice, we use a spectral reference that is a normalized linear ramp (from 70% at 340nm to 135% at 740nm). In addition, light scattered from the lunar surface is polarized, but we do not include polarization in our model. Figure 7 demonstrates our Moon model for various conditions.

4.3 Illumination from the Moon

For illumination coming from the Moon, it is sufficient to use a simple directional model since the Moon is very distant. This moreover introduces less variance in the illumination integration.

Using the Lommel-Seeliger law [46], the irradiance E_m from the Moon at phase angle φ and a distance d can be expressed as:

$$E_m(\varphi, d) = \frac{2\ C\ r_m^2}{3\ d^2} \left\{ E_{em} + E_{sm} \left(1 - \sin\frac{\varphi}{2} \tan\frac{\varphi}{2} \log\left(\cot\frac{\varphi}{4}\right)\right) \right\}, \tag{5}$$

where r_m is the radius of the Moon, E_{sm} is the irradiance from the Sun at the surface of the Moon, and E_{em} is the earthshine contribution as computed from Equation 1. Recall that the phase angle of the Moon with respect to earthshine is always null. The normalizing constant C is the average albedo of the Moon ($C = 0.072$).

5 Starlight

Stars are important visual features in the sky. We use actual star positions (as described in Section 3.4), and brightnesses and colors from the same star catalogue [14]. In this section, we describe how stars are included in our model, present the calculation of their brightness and chromaticity, and finally, discuss the illumination coming from stars.

5.1 Rendering Stars

Stars are very small, and it is therefore not practical to use explicit ray tracing to render them as rays would easily miss them. Instead, we use an image-based approach in which a separate star image is generated and composited using an alpha image that models attenuation in the atmosphere. The use of the alpha image ensures that the intensity of the stars is correctly reduced due to scattering and absorption in the atmosphere. The alpha map records for every pixel the visibility of objects beyond the atmosphere. It is generated by the ray tracer as a secondary image. Each time a ray from the camera leaves the atmosphere, the transmitivity is stored in the alpha image. The star image is multiplied by the alpha image and added to the rendered image to produce the final image.

For star clusters, such as the Milky Way, where individual stars are not visible, we use a high resolution (14400x7200) photographic mosaic of the Night Sky [25]. The brightest stars were removed using pattern-matching and median filtering.

5.2 Color and Brightness of Stars

A stellar magnitude describes the apparent star brightness. Given the visual magnitude, the irradiance at the Earth is [21]:

$$E_s = 10^{0.4(-m_v - 19)}\ \frac{W}{m^2}. \tag{6}$$

For the Sun, $m_v \approx -26.7$; for the full Moon, $m_v \approx -12.2$; and for Sirius, the brightest star, $m_v \approx -1.6$. The naked eye can see stars with a stellar magnitude up to approximately 6. However, this is the visible magnitude, which takes into account the atmospheric scattering. Since we simulate atmospheric scattering, we must discount this absorption, which accounts for 0.4 magnitude:

$$E'_s = 10^{0.4(-m_v - 19 + 0.4)}\ \frac{W}{m^2}. \tag{7}$$

The color of the star is not directly available as a measured spectrum. Instead, astronomers have established a standard series of measurements in particular wave-bands. A widely used *UBV system* introduced by Johnson [18] isolates bands of the spectrum in the blue intensity B, yellow-green intensity V, and ultra-violet intensity U. The difference $B - V$ is called the *color index* of a star, which is a numerical measurement of the color. A negative value of $B - V$ indicates a more bluish color, while a positive value indicates a redder hue. UBV is not directly useful for rendering purposes. However, we can use the color index to estimate a star's temperature [33, 40]:

$$T_{eff} = \frac{7000K}{B - V + 0.56}. \tag{8}$$

To compute spectral irradiance from a star given T_{eff}, we first compute a non-spectral irradiance value from the stellar magnitude using Equation 7. We then use the computed value to scale a normalized spectrum based on Planck's radiation law for black body radiators [39]. The result is spectral irradiance.

In the color pages (Figure 9) we have rendered a close-up of stars. We use the physically-based glare filter by Spencer et al. [41] as a flare model for the stars. This model fits nicely with the observations by Navarro and Losada [28] regarding the shape of stars as seen by the human eye.

Figure 10 is a time-lapse rendering of stars. Here we have simulated a camera and omitted the loss of color by a human observer. The colors of the stars can be seen clearly in the trails. Notice, also the circular motion of the stars due to the rotation of the Earth.

5.3 Illumination from Stars

Even though many stars are not visible to the naked eye, there is a collective contribution from all stars when added together. We use a constant irradiance of $3 \cdot 10^{-8} \frac{W}{m^2}$ [38] to account for integrated starlight.

6 Other Elements

A variety of phenomena affect the appearance of the night sky in subtle ways. While one might assume the sky itself is colored only by scattered light in the atmosphere, that is in fact only one of four specific sources of diffuse visible color in the night sky. The other three are zodiacal light, airglow, and galactic/cosmic light. We include all of these in our model. These effects are especially important on moonless nights, when a small change in illumination can determine whether an object in the scene is visible or invisible.

6.1 Zodiacal Light

The Earth co-orbits with a cloud of dust around the Sun. Sunlight scatters from this dust and can be seen from the Earth as *zodiacal light* [5, 36]. This light first manifests itself in early evening as a diffuse wedge of light in the southwestern horizon and gradually broadens with time. During the course of the night the zodiacal light becomes wider and more upright, although its position relative to the stars shifts only slightly [2].

The structure of the interplanetary dust is not well-understood. To to simulate zodiacal light we use a table with measured values [38]. Whenever a ray exits the atmosphere, we convert the direction of the ray to ecliptic polar coordinates and perform a bilinear lookup in the table. This works well since zodiacal light changes slowly with direction and has very little seasonal variation. A result of our zodiacal light model is illustrated Figure 8.

6.2 Airglow

Airglow is faint light that is continuously emitted by the entire upper atmosphere with a main concentration at an elevation of approximately 110 km. The upper atmosphere of the Earth is continually being bombarded by high energy particles, mainly from the Sun. These particles ionize atoms and molecules or dissociate molecules and in turn cause them to emit light in particular spectral lines (at discrete wavelengths). As the emissions come primarily from Na and O atoms as well as molecular nitrogen and oxygen the emission lines are easily recognizable. The majority of the airglow emissions occur at 557.7nm (O-I), 630nm (O-I) and a 589.0nm - 589.6nm doublet (Na-I). Airglow is the principal source of light in the night sky on moonless nights.

Airglow is integrated into the simulation by adding an active layer to the atmosphere (altitude 80km) that contributes a spectral in-scattered radiance ($5.1 \cdot 10^{-8} \frac{W}{m^2}$, 3 peaks, at 557.7nm, 583.0nm and 630.0nm).

6.3 Diffuse Galactic Light and Cosmic Light

Diffuse galactic light and cosmic light are the last components of the night sky that we include in our model. These are very faint (see Figure 2) and modeled as a constant term ($1 \cdot 10^{-8} \frac{W}{m^2}$) that is added when a ray exits the atmosphere.

6.4 Atmosphere Modeling

Molecules and aerosols (dust, water drops and other similar-sized particles) are the two main constituents of the atmosphere that affect light. As light travels through the atmosphere it can be scattered by molecules (Rayleigh scattering) or by aerosols (Mie scattering). The probability that a scattering event occurs is proportional to the local density of molecules and aerosols and the optical path length of the light. The two types of scattering are very different: Rayleigh scattering is strongly dependent on the wavelength of the light and it scatters almost diffusely; aerosol scattering is mostly independent of the wavelength, but with a strong peak in the forward direction of the scattered light.

We model the atmosphere using a spherical model similar to that of Nishita et al. [29, 31] and use the same phase functions to approximate the scattering of light. To simulate light transport with multiple scattering, we use distribution ray tracing combined with ray marching. A ray traversing the atmosphere uses ray marching to integrate the optical depth, and it samples the in-scattered indirect radiance at random positions in addition to the direct illumination. Each ray also keeps track of the visibility of the background, and all rays emanating from the camera save this information in the alpha image (as discussed in Section 5.1). This method is fairly efficient because the atmosphere is optically thin. Also, the method is very flexible and allows us to integrate other components in the atmosphere such as clouds and airglow.

We model clouds procedurally using an approach similar to the one described in [9]. Clouds are similar to the atmosphere, but as they have a higher density, the number of scattering events will be larger. For clouds we use the Henyey-Greenstein phase-function [13] with strong forward scattering.

7 Implementation and Results

We implemented our night sky model in a Monte Carlo ray tracer with support for spectral sampling. We used an accurate spectral sampling with 40 evenly-spaced samples from 340nm to 740nm to allow for precise conversion to XYZV color space for tone mapping, as well as for accurately handling the wavelength dependent Rayleigh scattering of the sky. Simpler models could be used as well, at the expense of accuracy.

Because our scenes are at scotopic viewing levels (rod vision), special care must be taken with tone mapping. We found the histogram-adjustment method proposed by Ward et al. [22] to work best for the very high dynamic range night images. This method "discounts" the empty portions of the histogram of luminance, and simulates both cone (daylight) and rod (night) vision as well as loss of acuity. Another very important component of our tone-mapping model is the blue-shift (the subjective impression that night scenes exhibit a bluish tint). This phenomena is supported by psychophysical data [15] and a number of techniques can be used to simulate it [8, 17, 43]. We use the technique for XYZV images as described in [17].

Many features of our model have been demonstrated through the paper. Not all of the elements can be seen simultaneously in one image. In particular, the dimmest phenomena such as zodiacal light are visible only for moonless nights. Phenomena such as airglow and intergalactic light are present only as faint background illumination. Nonetheless, all these components are important to give a realistic impression of the night sky. The sky is never completely black.

Our experimental results are shown in the two color pages at the end of the paper. The captions explain the individual images. All the images were rendered on a dual PIII-800 PC, and the rendering time for most of the individual images ranged from 30 seconds to 2 minutes. It may seem surprising that multiple scattering in the atmosphere can be computed this quickly. The main reason for this is that multiple scattering often does not contribute much and as such can be computed with low accuracy — as also demonstrated in [29]. The only images that were more costly to render are the images with clouds; we use path tracing of the cloud media and this is quite expensive. As an example the image of Little Matterhorn [35] with clouds (Figure 13) took 2 hours to render.

A very important aspect of our images is the sense of night. This is quite difficult to achieve, and it requires carefully ensuring correct physical values and using a perceptually based tone-mapping algorithm. This is why we have stressed the use of accurate radiometric values in this paper. For a daylight simulation it is less noticeable if the sky intensity is wrong by a factor two, but in the night sky all of the components have to work together to form an impression of night.

8 Conclusions and Future Work

This paper has presented a physically-based model of the night sky. The model uses astronomical data and it includes all the significant sources of natural light in the night sky. We simulate the direct appearance of the Moon, stars, the Milky Way, the zodiacal light, and other elements. In addition we model the illumination from these sources of light including scattering in the atmosphere. Finally, we use accurate spectral sampling and tone-mapping in order to render convincing images of night scenes.

Our model suggests several interesting areas for future work. Additional optical effects that we would like to include involve the modeling of small discrepancies, such as the hiding effect that causes the Full Moon to be brighter than expected, the luminescence of the Moon due to ionization caused by solar particles, and the seasonal and diurnal variations of airglow. More work remains to be done to accurately incorporate artificial light sources, city glow, and light pollution [26].

Tone mapping for night and twilight scenes presents many challenges. The complex interaction between rods and cones results in non-linear phenomena, in particular for brightness and color perception. Reproducing the alteration of motion perception at night is yet another topic for future work.

Acknowledgments

Thanks to Pat Hanrahan for several insightful comments, to Marc Levoy for last-minute creative suggestions, to Axel Mellinger for his full-resolution Milky Way mosaic, to Stephen Duck for modeling the night city skyline and to Frank Stark and Maryann Simmons for a conscientious read back. Additional thanks to Max Chen, Claus Wann Jensen, Steve Marschner, Ministry, Les Norford, and Byong Mok-Oh. The work of the first author was supported in part by NSF/ITR (IIS-0085864) and DARPA (DABT63-95-C-0085). In addition this work was supported by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219), an NSF CISE Research Infrastructure award (EIA09802220), an NSF grant (97-31859), and a gift from Pixar Animation Studios.

Appendix

In this Appendix we present formulas for coordinate conversion and low-precision formulas for the positions of the Sun and Moon. The formulas are adapted from [10, 23, 24, 44]. All the angles are in radians, unless otherwise noted. R_x, R_y and R_z are the standard counter-clockwise rotation matrices about the principal axes.

Time Conversion

Astronomers often use *Julian dates*. For the date $M/D/Y$ ($Y > 1582$) at time $h{:}m{:}s$ ($0 \leq h < 24$), the Julian date is given by

$$\begin{aligned} JD &= 1720996.5 - \lfloor Y'/100 \rfloor + \lfloor Y'/400 \rfloor + \lfloor 365.25 Y' \rfloor \\ &\quad + \lfloor 30.6001(M'+1) \rfloor + D + (h + (m + s/60)/60)/24 \end{aligned}$$

where Y' and M' are the adjusted year and month: if M is 1 or 2, then $Y' = Y - 1$ and $M' = M + 12$ otherwise $Y' = Y$ and $M' = M$. $\lfloor\ \rfloor$ denotes the floor integer truncation function.

Local time is GMT with a zone correction. *Terrestrial Time* (TT) is essentially the time kept by atomic clocks. As it is not corrected for the slowing of the Earth's rotation, it gains on GMT by about a second per year. The current difference ΔT is about 65 sec. It should be added to s in Equation (9) above for precise computation.

The variable T in this Appendix is the time in Julian centuries since January 1, 2000: $T = (JD - 2451545.0)/36525$.

Coordinate Conversion

Coordinate conversion is most easily done in rectangular coordinates using rotation matrices. The generic rectangular conversion is:

$$\begin{aligned} x &= r \cos(\text{longitude}) \cos(\text{latitude}) \\ y &= r \sin(\text{longitude}) \cos(\text{latitude}) \\ z &= r \sin(\text{latitude}). \end{aligned} \qquad (10)$$

A point in ecliptic coordinates may be converted to equatorial coordinates using the matrix $R_x(\epsilon)$, where ϵ is the *obliquity of the ecliptic*. The mean value in radians is $\epsilon = 0.409093 - 0.000227T$, with T as above.

Converting to local horizon coordinates is harder. The rotation of the Earth is abstracted by the *local sidereal time* of an observer, which when measured in angular hours (1 hour = 15 degrees) gains about 4 arc-minutes a day on local solar time. If lon is the observer's longitude in radians (East is positive), the local mean sidereal time, in radians, is given by $\text{LMST} = 4.894961 + 230121.675315T + \text{lon}$. Here T is as above, but in GMT *without* the correction ΔT. All other formulas in this Appendix assume the ΔT correction has been included in the computation of T.

The matrix for converting mean equatorial coordinates to horizon coordinates is $R_y(lat - \pi/2) R_z(-\text{LMST}) P$, where lat is the observer's latitude in radians, positive to the North. P is a rotation matrix that corrects for precession and nutation. The effect of precession is about one degree per century, so P can be omitted near 2000. For precession only, P is approximately $P = R_z(0.01118T) R_y(-0.00972T) R_z(0.01118T)$. Nutation never amounts to more than about 20 arc-seconds.

Position Computations

We conclude by giving low precision formulas for the position of the Sun and the Moon. The formulas are for the ecliptic longitude and latitude (λ, β) corrected for precession. The Sun position formula is accurate to about one arc-minute within five centuries of 2000; and the Moon position formula is accurate to better than eight arc-minutes within five centuries of 2000.

Sun

For the coordinates of the sun, compute $M = 6.24 + 628.302T$,

$$\begin{aligned} \lambda &= 4.895048 + 628.331951T + (0.033417 - 0.000084T) \sin M \\ &\quad + 0.000351 \sin 2M, \\ r &= 1.000140 - (0.016708 - 0.000042T) \cos M - 0.000141 \cos 2M, \end{aligned}$$

and $\beta = 0$. The geocentric distance r is in astronomical units (1au = 1.496×10^{11}m = 23455 Earth radii.) For the position in local horizon coordinates, convert to rectangular coordinates, then rotate using the matrix $R_y(lat - \pi/2) R_z(-\text{LMST}) R_x(\epsilon)$.

Moon

The ecliptic geocentric coordinates of the Moon are computed from

$$\begin{aligned} l' &= 3.8104 + 8399.7091T & m' &= 2.3554 + 8328.6911T \\ m &= 6.2300 + 628.3019T & d &= 5.1985 + 7771.3772T \\ f &= 1.6280 + 8433.4663T \end{aligned}$$

$$\begin{aligned} \lambda = {}& l' \\ & +0.1098 \sin(m') \\ & +0.0222 \sin(2d - m') \\ & +0.0115 \sin(2d) \\ & +0.0037 \sin(2m') \\ & -0.0032 \sin(m) \\ & -0.0020 \sin(2f) \\ & +0.0010 \sin(2d - 2m') \\ & +0.0010 \sin(2d - m - m') \\ & +0.0009 \sin(2d + m') \\ & +0.0008 \sin(2d - m) \\ & +0.0007 \sin(m' - m) \\ & -0.0006 \sin(d) \\ & -0.0005 \sin(m + m') \end{aligned}$$

$$\begin{aligned} \beta = {}& +0.0895 \sin(f) \\ & +0.0049 \sin(m' + f) \\ & +0.0048 \sin(m' - f) \\ & +0.0030 \sin(2d - f) \\ & +0.0010 \sin(2d + f - m') \\ & +0.0008 \sin(2d - f - m') \\ & +0.0006 \sin(2d + f) \end{aligned}$$

$$\begin{aligned} \pi' = {}& +0.016593 \\ & +0.000904 \cos(m') \\ & +0.000166 \cos(2d - m') \\ & +0.000137 \cos(2d) \\ & +0.000049 \cos(2m') \\ & +0.000015 \cos(2d + m') \\ & +0.000009 \cos(2d - m) \end{aligned}$$

The distance is $d_{\text{Moon}} = 1/\pi'$ in units of Earth radii. To correct for the observer's position on the Earth, convert to rectangular coordinates and to local horizon coordinates as for the sun. Then subtract the vector $(0, 0, 1)$, the approximate position of the observer in horizon coordinates.

Rotation and Phase of the Moon

The Moon's geometry is modeled in a fixed lunar coordinate system. To orient the Moon in ecliptic coordinates, rotate by the matrix $R_z(f+\pi)R_x(0.026920)R_z(l'-f)$, with f as above. Then rotate by the equatorial to horizon conversion matrix for the orientation in horizon coordinates. Translating by the Moon's position in rectangular horizon coordinates (scaled by 6378137m) completes the position and orientation of the Moon in local horizon coordinates.

Using rectangular coordinates makes it easy to compute the position of the shadow terminator. If $\vec{m}$ is the topocentric position of the Moon, and $\vec{s}$ is that of the sun (in Earth radii: multiply d_{Moon} above by 23455; there is no need to correct for parallax for the sun's position) then the vector $(\vec{s}-\vec{m})\times(\vec{s}\times\vec{m})$ points from the center of the Moon to the leading (light to dark) terminator.

Temperature to Luv Conversion

Temp. (K)	u	v	Temp. (K)	u	v
100 000	0.18065	0.26589	4000	0.22507	0.33436
50000	0.18132	0.26845	3636	0.23243	0.33901
33333	0.18208	0.27118	3333	0.24005	0.34305
25000	0.18293	0.27407	3077	0.24787	0.34653
20000	0.18388	0.27708	2857	0.25585	0.34948
16667	0.18494	0.28020	2667	0.26394	0.35198
14286	0.18611	0.28340	2500	0.27210	0.35405
12500	0.18739	0.28666	2353	0.28032	0.35575
11111	0.18879	0.28995	2222	0.28854	0.35713
10000	0.19031	0.29325	2105	0.29676	0;35822
8000	0.19461	0.30139	2000	0.30496	0.35906
6667	0.19960	0.30918	1905	0.31310	0.35968
5714	0.20523	0.31645	1818	0.32119	0.36011
5000	0.21140	0.32309	1739	0.32920	0.36038
4444	0.21804	0.32906	1667	0.33713	0.36051

References

[1] Baranoski, G., Rokne, J., Shirley, P., Trondsen, T., and Bastos, R. Simulating the aurora borealis. In *Proc. of Pacific Graphics* (2000).

[2] Blackwell, D. E. The zodiacal light. *Scientific American 54* (July 1960).

[3] Blinn, J. The jupiter and saturn fly-by animations, 1980.

[4] Blinn, J. F. Light reflection functions for simulation of clouds and dusty surfaces. *Proc. of SIGGRAPH* (1982).

[5] Dermott, S. F., and Liou, J. C. Detection of asteroidal dust particles from known families in near-earth orbits. In *AIP Conference Proceedings* (July 1994), vol. 301(1), pp. 13–21.

[6] Dobashi, Y., Nishita, T., Kaneda, K., and Yamashita, H. A fast display method of sky colour using basis functions. *J. of Visualization and Computer Animation 8*, 3 (Apr. – June 1997), 115–127.

[7] Duffett-Smith, P. *Astronomy with your personal computer*, 2nd ed. Cambridge University Press, 1990.

[8] Durand, F., and Dorsey, J. Interactive tone mapping. *Eurographics Workshop on Rendering* (2000).

[9] Ebert, D., Musgrave, K., Peachey, D., Perlin, K., and Worley, S. *Texturing and Modeling: A procedural Approach*. Academic Press, 1994.

[10] Green, R., Ed. *Spherical Astronomy*. Cambridge Univ. Pr., 1985.

[11] Hapke, B. Optical properties of the lunar surface. In *Physics and astronomy of the Moon*, Kopal, Ed. Academic Press, 1971.

[12] Hapke, B. W. A theoretical photometric function of the lunar surface. *Journal of Geophysical Research 68*, 15 (1963), 4571–4586.

[13] Henyey, L. G., and Greenstein, J. L. Diffuse radiation in the galaxy. *Astrophysics Journal 93* (1941), 70–83.

[14] Hoffleit, D., and Warren, W. *The Bright Star Catalogue*, 5th ed. Yale University Observatory, 1991.

[15] Hunt. Light and dark adaptation and the perception of color. *Journal of the Optical Society of Am. A 42*, 3 (1952), 190.

[16] J. van Diggelen. Photometric properties of lunar carter floors. *Rech. Obs. Utrecht 14* (1959), 1–114.

[17] Jensen, H. W., Premoze, S., Shirley, P., Thompson, W., Ferwerda, J., and Stark, M. Night rendering. Tech. Rep. UUCS-00-016, Computer Science Dept., University of Utah, Aug. 2000.

[18] Johnson, H. L., and Morgan, W. W. Fundamental stellar photometry for standards of spectral type on the revised system of the yerkes spectral atlas. *Astrophysics Journal 117*, 313 (1953).

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

[20] Kopal, Z. *The Moon*. D. Reidel Publishing Company, Dordrecht, Holland, 1969.

[21] Lang, K. Astrophysical formulae. *Astronomy and astrophysics library* (1999).

[22] Larson, G. W., Rushmeier, H., and Piatko, C. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Trans. on Visualization and Computer Graphics 3*, 4 (Oc. - Dec. 1997), 291–306.

[23] Meeus, J. *Astronomical Formulae for Calculators*, 4th ed. Willman-Bell, Inc., 1988.

[24] Meeus, J. *Astronomical Algorithms*, 2nd ed. Willmann-Bell, Inc., Richmond, VA, 1999.

[25] Mellinger, A. A 360°x180° all-sky panorama. http://canopus.physik.uni-potsdam.de/˜axm/images.html, 2000.

[26] Minnaert, M. *Light and Color in the Outdoors*. Springer-Verlag, 1974.

[27] Naval Research Laboratory. Clementine deep space program science experiment. http://www.nrl.navy.mil/clementine/.

[28] Navarro, R., and Losada, M. A. Shape of stars and optical quality of the human eye. *Journal of the Optical Society of America (A) 14*, 2 (1997), 353–359.

[29] Nishita, T., Dobashi, Y., Kaneda, K., and Yamashita, H. Display method of the sky color taking into account multiple scattering. In *Proc. of Pacific Graphics* (1996).

[30] Nishita, T., and Nakamae, E. Continuous tone representation of three-dimensional objects illuminated by sky light. In *Computer Graphics (SIGGRAPH '86 Proceedings)* (1986), vol. 20(4).

[31] Nishita, T., Sirai, T., Tadamura, K., and Nakamae, E. Display of the earth taking into account atmospheric scattering. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (1993), vol. 27.

[32] Oberschelp, W., and Hornug, A. Visualization of eclipses and planetary conjunction events. the interplay between model coherence, scaling and animation. In *Proc. of Computer Graphics International* (2000).

[33] Olson, T. The colors of the stars. In *IST/SID 6th Color Imaging Conf.* (1998).

[34] Preetham, A. J., Shirley, P., and Smits, B. A practical analytic model for daylight. In *Proc. of SIGGRAPH* (1999).

[35] Premoze, S., Thompson, W., and Shirley, P. Geospecific rendering of alpine terrain. In *Eurographics Workshop on Rendering* (1999).

[36] Reach, W. T., Franz, B. A., Kelsall, T., and Weiland, J. L. Dirbe observations of the zodiacal light. In *AIP Conference Proceedings* (January 1996), vol. 348, pp. 37–46.

[37] Richter, N. B. The photometric properties of interplanetary matter. *Quarterly Journal of the Royal Astronomical Society 3* (1962), 179–186.

[38] Roach, F., and Gordon, J. *The Light of the Night Sky*. Geophysics and Astrophysics Monographs, V. 4. D Reidel Pub Co, 1973.

[39] Siegel, R., and Howell, J. R. *Thermal Radiation Heat Transfer*, 3rd ed. Hemisphere Publishing Corporation, 1992.

[40] Smith, R. C. *Observational Astrophysics*. Cambridge University Press, 1995.

[41] Spencer, S., Shirley, P., Zimmerman, K., and Greenberg, D. Physically-based glare effects for digital images. In *Computer Graphics (Proc. Siggraph)* (1995).

[42] Tadamura, K., Nakamae, E., Kaneda, K., Baba, M., Yamashita, H., and Nishita, T. Modeling of skylight and rendering of outdoor scenes. In *Eurographics '93* (1993), Blackwell Publishers.

[43] Upstill, S. *The Realistic Presentation of Synthetic Images: Image Processing in Computer Graphics*. PhD thesis, Berkeley, 1985.

[44] U.S. Naval Observatory, R. G. O. *The Astronomical Almanac for the Year 2001*. U.S. Government Printing Office, 2001.

[45] van de Hulst, H. *Light Scattering by Small Particles*. Wiley & Sons, 1957.

[46] van de Hulst, H. *Multiple Light Scattering*. Academic Press, 1980.

[47] Yaeger, L., Upson, C., and Myers, R. Combining physical and visual simulation – creation of the planet jupiter for the film ”2010”. *Proc. of SIGGRAPH)* (1986).

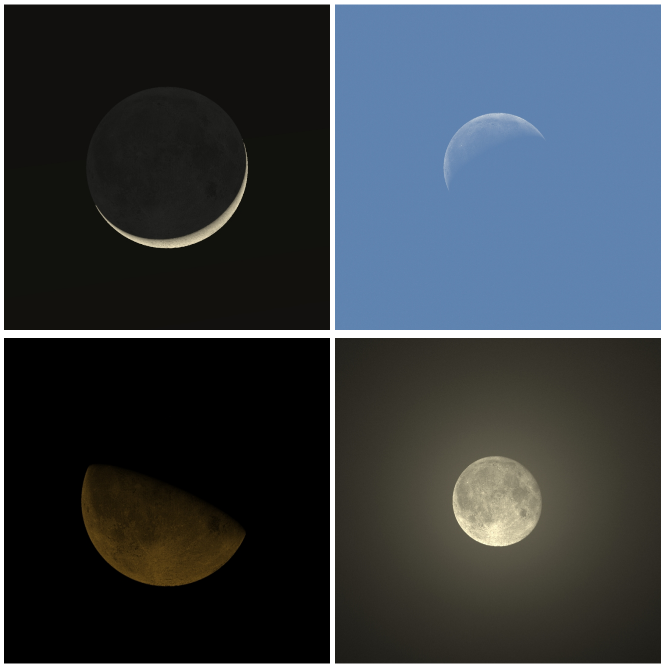


Figure 7: The Moon rendered at different times of the month and day and under different weather conditions. Clockwise from top left. The sliver moon is a simulation of the moon on Dec. 5; Earth-shine makes all of the moon visible. The Quarter Moon (Dec. 12) is a day simulation and the moon is seen behind the blue sky. The Gibbous Moon (Dec. 17) is a rendering of the moon low in the sky; it is colored red/orange due to attenuation in the atmosphere of the blue part of the light. The full moon (Dec. 22) is a simulation of the moon seen through a thin layer of clouds with strong forward scattering; the scattering in the cloud causes the bright region around the moon.

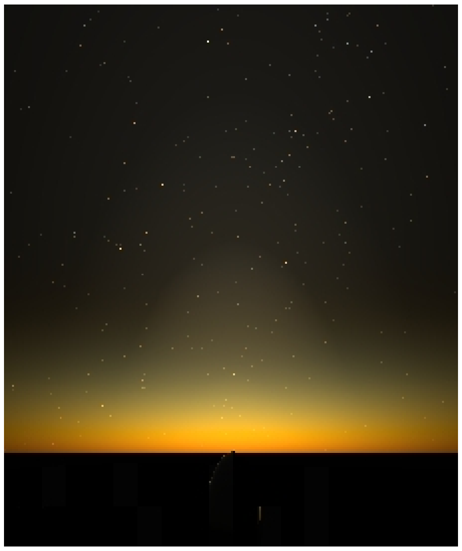


Figure 8: Zodiacal light seen as a wedge of light rising from the horizon in an early autumn morning. Zodiacal light is easiest to see on spring evenings and autumn mornings from the Northern hemisphere.

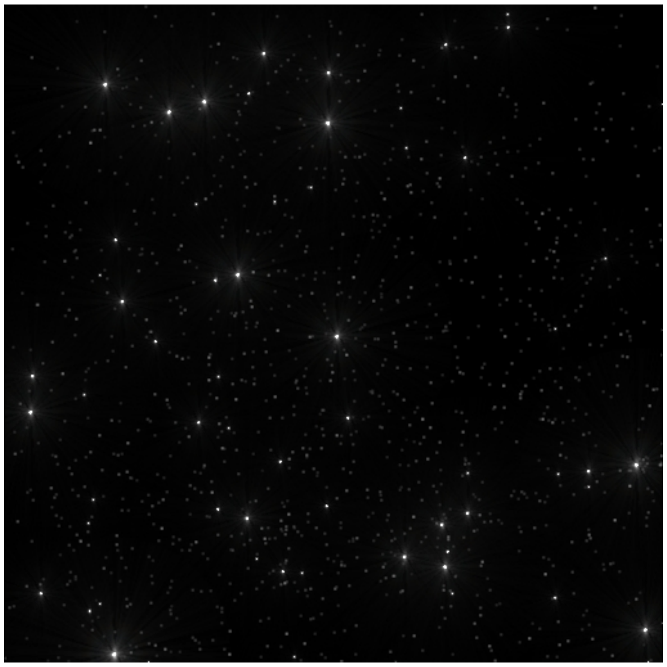


Figure 9: Close-up of rendered stars. Note the glare simulation around the brighter stars. The Big Dipper (in the constellation of Ursa Major) is clearly recognizable at the top of the image.

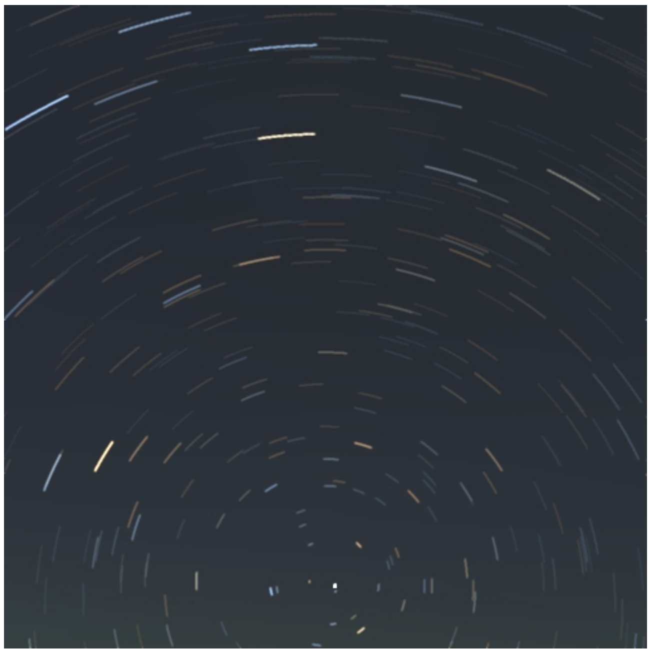


Figure 10: Time-lapse rendering of stars. A simulation of a 30-minute camera exposure of the sky. Note how the stars move in circular curves due to the rotation of the earth. Also, note the color of the stars (we did not apply tone-mapping for a human observer, and the true colors of the stars are seen in the trails).

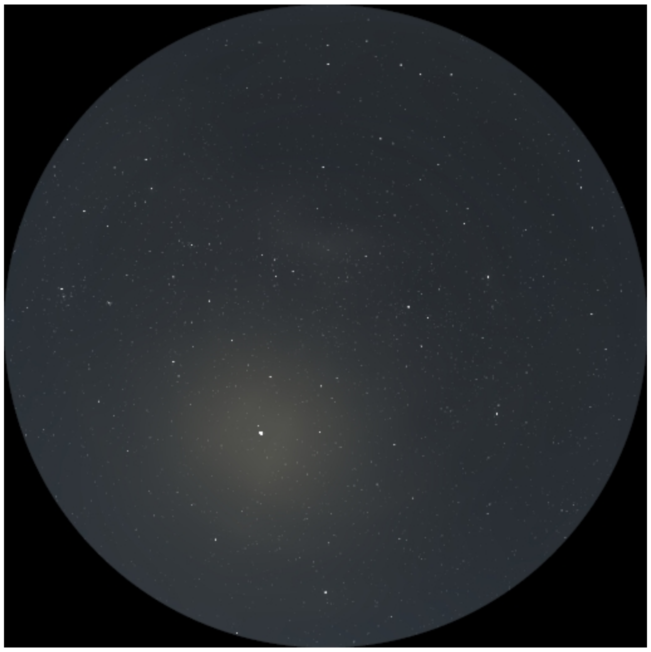


Figure 11: Fisheye lens projection of a hazy night sky illuminated by the full moon. Note the scattering in the atmosphere around the moon. Note also the Big Dipper near the top of the image.

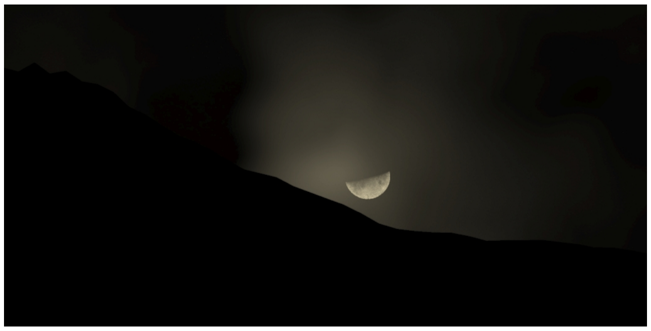


Figure 12: Moon rising above a mountain ridge. The only visible feature in this dark night scene is the scattering of light in the thin cloud layer.

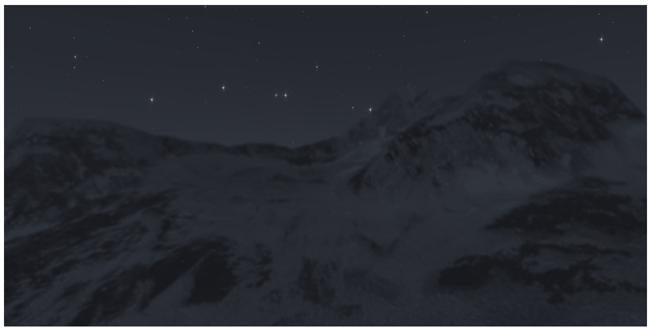


Figure 13: A simulation of Little Matterhorn illuminated by the full moon on a clear night. Notice how the tone-mapping combined with the blue shift give a sense of night.

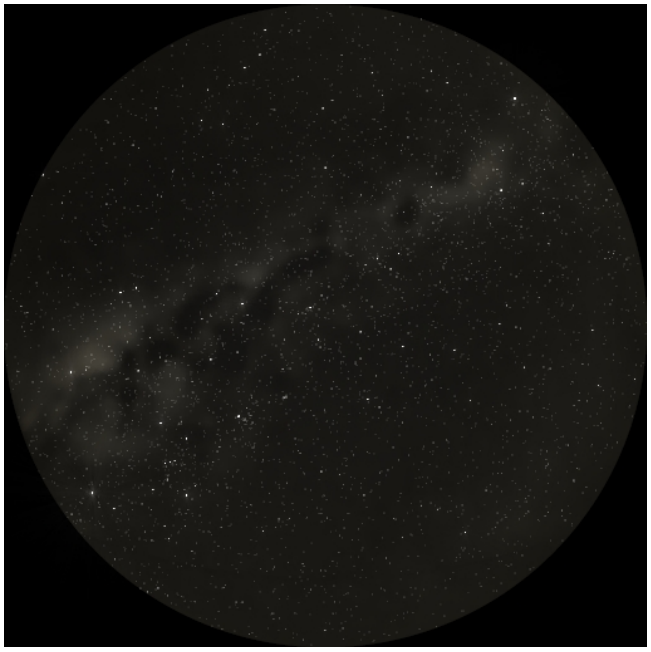


Figure 14: Fisheye lens projection of a clear moonless night. The Milky Way band is visible across the sky as are the dimmer stars. The Orion constellation can be seen at the lower left of the image.

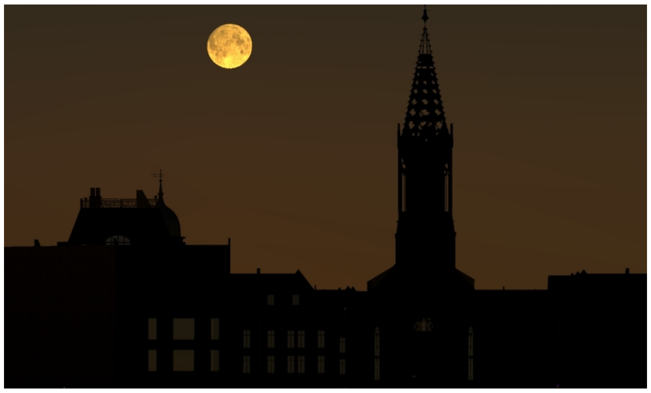


Figure 15: The low moon setting over a city skyline in the early morning. The red sky is illuminated via multiple scattering from the low rising sun.

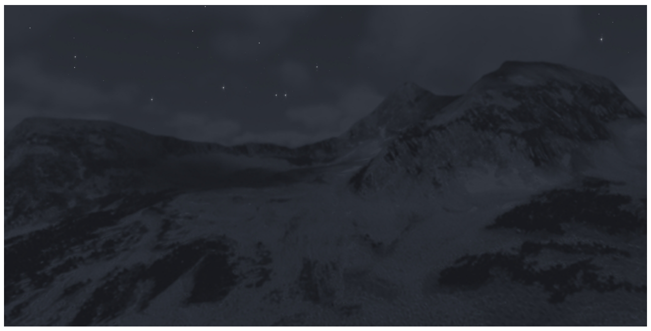


Figure 16: A simulation of Little Matterhorn illuminated by a full moon on a cloudy night sky. Note the reduced visibility of the stars as well as the shadows of the clouds on the mountain.

An Efficient Representation for Irradiance Environment Maps

Ravi Ramamoorthi Pat Hanrahan

Stanford University [*]

Abstract

We consider the rendering of diffuse objects under distant illumination, as specified by an environment map. Using an analytic expression for the irradiance in terms of spherical harmonic coefficients of the lighting, we show that one needs to compute and use only 9 coefficients, corresponding to the lowest-frequency modes of the illumination, in order to achieve average errors of only 1%. In other words, the irradiance is insensitive to high frequencies in the lighting, and is well approximated using only 9 parameters. In fact, we show that the irradiance can be procedurally represented simply as a quadratic polynomial in the cartesian components of the surface normal, and give explicit formulae. These observations lead to a simple and efficient procedural rendering algorithm amenable to hardware implementation, a prefiltering method up to three orders of magnitude faster than previous techniques, and new representations for lighting design and image-based rendering.

CR Categories: I.3.7 [*Computer Graphics*]: Three-Dimensional Graphics and Realism—Environment Maps

Keywords: Environment Maps, Rendering Hardware, Signal Processing, Irradiance, Radiance, Illumination, Lambertian Reflectance, Prefiltering, Spherical Harmonics

1 Introduction

Lighting in most real scenes is complex, coming from a variety of sources including area lights and large continuous lighting distributions like skylight. But current graphics hardware only supports point or directional light sources. One reason is the lack of simple procedural formulas for general lighting distributions. Instead, an integration over the upper hemisphere must be done for each pixel.

One approach to using general lighting distributions is the method of environment maps. Environment maps are representations of the incident illumination at a point. Blinn and Newell [3] used them to efficiently find reflections of distant objects. Miller and Hoffman [14], and Greene [8] *prefiltered* environment maps, precomputing separate reflection maps for the diffuse and specular components of the BRDF. Cabral et al. [5] handled general BRDFs by using a 2D set of prerendered images. Prefiltering is generally an offline, computationally expensive process. After prefiltering, rendering can usually be performed at interactive rates with graphics hardware using texture-mapping.

This paper focuses on the Lambertian component of the BRDF. We use the term *irradiance environment map* for a diffuse reflection map indexed by the surface normal, since each pixel simply stores the irradiance for a particular orientation of the surface. For applications like games, irradiance maps are often stored directly on the surface, instead of as a function of the normal vector, and

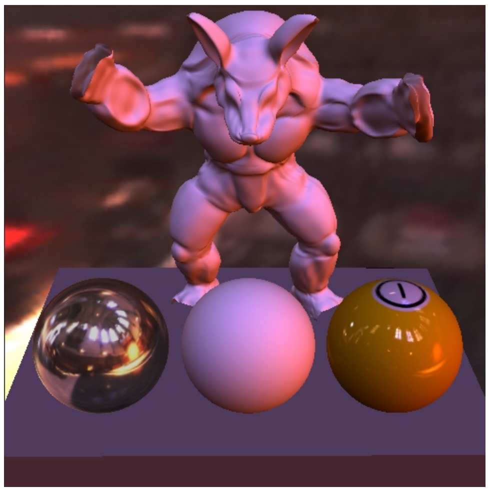


Figure 1: *The diffuse shading on all the objects is computed procedurally in real-time using our method. The middle sphere, armadillo, and table are white diffuse reflectors. The colors come from the environment—owing to a variety of colored sources, including blue stained-glass windows. Our method can also be combined with standard texture mapping—used to modulate the albedo of the pool-ball on the right—and reflection mapping—used for specular highlights on the pool-ball, and for the mirror sphere on the left. The environment is a light probe of the Grace Cathedral. Tone mapping is used to convey high dynamic range for the background and the mirror sphere; the remaining objects are shaded using a linear scale.*

are called *light maps*. Irradiance environment maps can also be extended to spatially varying illumination by computing an *irradiance volume*, as done by Greger et al. [9]. Many of the same ideas can be applied to speeding up global illumination algorithms. The slowly varying nature of irradiance has led to Ward and Heckbert [18] proposing interpolation using irradiance gradients, while the idea of storing irradiance as a function of surface orientation in *orientation lightmaps* has been proposed by Wilkie et al. [19].

The key to our approach is the rapid computation of an analytic approximation to the irradiance environment map. For rendering, we demonstrate a simple procedural algorithm that runs at interactive frame rates, and is amenable to hardware implementation. The procedural approach is preferable to texture-mapping in some applications. Since irradiance varies slowly with orientation, it need only be computed per-vertex and interpolated across triangles. Further, we require only a single texturing pass to render textured objects with irradiance environment maps, since the irradiance is computed procedurally. On the other hand, the standard approach requires a separate texture for the irradiance, and needs *multitexturing* support or multiple texturing passes. In other applications, where per-fragment texture-mapping is relatively inexpensive, our method can be used to very efficiently compute the irradiance environment map texture. Our novel representation also suggests new approaches to lighting design and image-based rendering.

2 Background

Empirically, it is well known that the reflected intensity from a diffuse surface varies slowly as a function of surface orientation. This qualitative observation has been used to justify representing irradiance environment maps at low resolutions [14], and in efficiently computing the shading hierarchically [11, 12]. Our goal is to use an analytic quantitative formula for the irradiance which formalizes these observations, and allows for principled approximations.

Let L denote the distant lighting distribution. As is common

[*] (ravir,hanrahan)@graphics.stanford.edu

with environment map algorithms, we neglect the effects of cast shadows and near-field illumination. The irradiance E is then a function of the surface normal $\mathbf{n}$ only and is given by an integral over the upper hemisphere $\Omega(\mathbf{n})$.

$$E(\mathbf{n}) = \int_{\Omega(\mathbf{n})} L(\omega)(\mathbf{n}\cdot\omega)\, d\,\omega \tag{1}$$

Note that $\mathbf{n}$ and ω are unit direction vectors, so E and L can be parameterized by a direction (θ, ϕ) on the unit sphere.

We must scale E by the surface albedo ρ, which may be dependent on position $\mathbf{p}$ and be described by a texture, to find the radiosity B, which corresponds directly to the image intensity.

$$B(\mathbf{p}, \mathbf{n}) = \rho(\mathbf{p})E(\mathbf{n}) \tag{2}$$

Our main concern will be approximating E. We [16] have been able to derive an analytic formula for the irradiance. Similar results have been obtained independently by Basri and Jacobs [2] in simultaneous work on face recognition. Our original motivation was the study of an inverse rendering problem—estimating the lighting from observations of a Lambertian surface, i.e. from the irradiance. In this paper, we will apply the formulae to a forward rendering problem—rendering diffuse objects with environment maps.

Our formulae are in terms of spherical harmonic [4, 13, 17] coefficients. Spherical harmonics Y_{lm}, with $l \geq 0$ and $-l \leq m \leq l$, are the analogue on the sphere to the Fourier basis on the line or circle. The first 9 spherical harmonics (with $l \leq 2$) are simply constant ($l = 0$), linear ($l = 1$), and quadratic ($l = 2$) polynomials of the cartesian components (x, y, z). and are given numerically by

$$\begin{aligned}
(x, y, z) &= (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \\
Y_{00}(\theta,\phi) &= 0.282095 \\
(Y_{11}; Y_{10}; Y_{1-1})\,(\theta,\phi) &= 0.488603\,(x; z; y) \\
(Y_{21}; Y_{2-1}; Y_{2-2})\,(\theta,\phi) &= 1.092548\,(xz; yz; xy) \\
Y_{20}(\theta,\phi) &= 0.315392\,\left(3z^2 - 1\right) \\
Y_{22}(\theta,\phi) &= 0.546274\,\left(x^2 - y^2\right)
\end{aligned} \tag{3}$$

Note that these basis functions are closely related to the spherical polynomials used by Arvo [1] in his irradiance tensor formulation.

$E(\theta, \phi)$ and $L(\theta, \phi)$ can be represented by the coefficients—E_{lm} and L_{lm}—in their spherical harmonic expansion.

$$\begin{aligned}
L(\theta,\phi) &= \sum_{l,m} L_{lm}Y_{lm}(\theta,\phi) \\
E(\theta,\phi) &= \sum_{l,m} E_{lm}Y_{lm}(\theta,\phi)
\end{aligned} \tag{4}$$

We also define $A = (\mathbf{n}\cdot\omega)$ with coefficients A_l. Since A has no azimuthal dependence, $m = 0$ and we use only the l index.

$$A(\theta) = \max\left[\cos\theta, 0\right] = \sum_l A_l Y_{l0}(\theta)$$

With these definitions one can show [16] that

$$E_{lm} = \sqrt{\frac{4\pi}{2l+1}} A_l L_{lm} \tag{5}$$

It will be convenient to define a new variable $\hat{A}_l$ by

$$\hat{A}_l = \sqrt{\frac{4\pi}{2l+1}} A_l \tag{6}$$

For rendering, it will be convenient to expand out the irradiance.

$$E(\theta,\phi) = \sum_{l,m} \hat{A}_l L_{lm} Y_{lm}(\theta,\phi) \tag{7}$$

An analytic formula for A_l can be derived [16]. It can be shown that $\hat{A}_l$ vanishes for odd values of $l > 1$, and even terms fall off very rapidly as $l^{-5/2}$. The analytic formulae are given by

$$\begin{aligned}
l = 1 \quad & \hat{A}_1 = \frac{2\pi}{3} \\
l > 1, odd \quad & \hat{A}_l = 0 \\
l \;\; even \quad & \hat{A}_l = 2\pi\frac{(-1)^{\frac{l}{2}-1}}{(l+2)(l-1)}\left[\frac{l!}{2^l(\frac{l}{2}!)^2}\right]
\end{aligned} \tag{8}$$

Numerically, the first few terms are

$$\begin{gathered}
\hat{A}_0 = 3.141593 \quad \hat{A}_1 = 2.094395 \quad \hat{A}_2 = 0.785398 \\
\hat{A}_3 = 0 \quad \hat{A}_4 = -0.130900 \quad \hat{A}_5 = 0 \quad \hat{A}_6 = 0.049087
\end{gathered} \tag{9}$$

Approximation: For rendering, the key observation is that $\hat{A}_l$ decays so fast that we need consider only low-frequency lighting coefficients, of order $l \leq 2$. Equivalently, **the irradiance is well approximated by only 9 parameters**—1 for $l = 0, m = 0$, 3 for $l = 1, -1 \leq m \leq 1$, and 5 for $l = 2, -2 \leq m \leq 2$. By working in frequency-space, we exploit the low-frequency character of $A = (\mathbf{n}\cdot\omega)$, using a few coefficients instead of a full hemispherical integral. The simple form of the first 9 spherical harmonics, given in equation 3, makes implementation straightforward.

3 Algorithms and Results

In this section, we discuss three applications of this result. First, we show how to rapidly prefilter the lighting distribution, computing the coefficients L_{lm}. Next, we develop a simple real-time procedural shader for rendering that takes these coefficients as inputs. Finally, we discuss other applications of our representation.

3.1 Prefiltering

For a given environment map, we first find the 9 lighting coefficients, L_{lm} for $l \leq 2$, by integrating against the spherical harmonic basis functions. Each color channel is treated separately, so the coefficients can be thought of as RGB values.

$$L_{lm} = \int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi} L(\theta,\phi)Y_{lm}(\theta,\phi)\sin\theta\, d\theta d\phi \tag{10}$$

The expressions for the Y_{lm} are found in equation 3. The integrals are simply sums of the pixels in the environment map L, weighted by the functions Y_{lm}. The integrals can also be viewed as moments of the lighting, or as inner-products of the functions L and Y_{lm}.

Since we compute 9 numbers, the prefiltering step takes O(9S) time, where S is the size (total number of pixels) of the environment map. By comparison, the standard method of computing an irradiance environment map texture takes $O(T \cdot S)$ time, where T is the number of texels in the irradiance environment map. Our method will therefore be approximately $T/9$ times faster[1]. Even if a conventional irradiance environment map is computed at a very low resolution of 64×64, corresponding to $T = 4096$, our method will be nearly 500 times faster.

We have implemented prefiltering as a preprocessing step for a given environment map. Values of L_{lm} for a few light probes are tabulated in figure 2. The computation time for a 300x300 environment map was less than a second. This indicates that our approach might be able to handle scenes with dynamic lighting in the future. By contrast, the standard method of performing a hemispherical integral for each pixel to compute the irradiance environment map took approximately two hours. In fact, if an explicit representation of the irradiance environment map texture is required, we believe

[1]It may be possible to use a hierarchical integration scheme, as demonstrated by Kautz et al. [11] for Phong BRDFs, to speed up both our method and the conventional approach. Hardware acceleration may also be possible.

	Grace Cathedral			Eucalyptus Grove			St. Peters Basilica		
L_{00}	.79	.44	.54	.38	.43	.45	.36	.26	.23
L_{1-1}	.39	.35	.60	.29	.36	.41	.18	.14	.13
L_{10}	-.34	-.18	-.27	.04	.03	.01	-.02	-.01	-.00
L_{11}	-.29	-.06	.01	-.10	-.10	-.09	.03	.02	.01
L_{2-2}	-.11	-.05	-.12	-.06	-.06	-.04	.02	.01	.00
L_{2-1}	-.26	-.22	-.47	.01	-.01	-.05	-.05	-.03	-.01
L_{20}	-.16	-.09	-.15	-.09	-.13	-.15	-.09	-.08	-.07
L_{21}	.56	.21	.14	-.06	-.05	-.04	.01	.00	.00
L_{22}	.21	-.05	-.30	.02	-.00	-.05	-.08	-.06	.00

Figure 2: *RGB values of lighting coefficients for a few environments. These may be used directly for rendering, and for checking the correctness of implementations. Note that L_{1-1}, corresponding to the linear moment along the y-axis or vertical direction, is relatively large and positive for all environments. This is because the upper hemisphere is significantly brighter than the lower hemisphere, owing to skylight or ceiling lamps. For the Eucalyptus grove, where the lighting is almost symmetric about the y-axis, the other moments are relatively small. Therefore, for that environment, the most significant lighting coefficients are the constant (ambient) term L_{00}, and L_{1-1}.*

the best way of computing it is to first compute the 9 coefficients L_{lm} using our method, and then use these to very rapidly generate the irradiance environment map using the rendering method described below.

It is important to know what errors result from our 9 parameter approximation. The maximum error for any pixel, as a fraction of the total intensity of the illumination, is 9% and corresponds to the maximum error in the order 2 approximation of $A(\theta)$. Furthermore, the average error over all surface orientations can be shown to be under 3% for any physical input lighting distribution [2]. For the environment maps used in our examples, corresponding to complex natural illumination, the results are somewhat better than the worst-case bounds—the average error is under 1%, and the maximum pixel error is under 5%. Finally, figure 3 provides a visual comparison of the quality of our results with standard prefiltering, showing that our method produces a perceptually accurate answer.

3.2 Rendering

For rendering, we can find the irradiance using equation 7. Since we are only considering $l \leq 2$, the irradiance is simply a quadratic polynomial of the coordinates of the (normalized) surface normal. Hence, with $\mathbf{n}^t = (x\ y\ z\ 1)$, we can write

$$E(\mathbf{n}) = \mathbf{n}^t M \mathbf{n} \tag{11}$$

M is a symmetric 4x4 matrix. Each color has an independent matrix M. Equation 11 is particularly useful for rendering, since we require only a matrix-vector multiplication and a dot-product to compute E. The matrix M is obtained by expanding equation 7:

$$M = \begin{pmatrix} c_1 L_{22} & c_1 L_{2-2} & c_1 L_{21} & c_2 L_{11} \\ c_1 L_{2-2} & -c_1 L_{22} & c_1 L_{2-1} & c_2 L_{1-1} \\ c_1 L_{21} & c_1 L_{2-1} & c_3 L_{20} & c_2 L_{10} \\ c_2 L_{11} & c_2 L_{1-1} & c_2 L_{10} & c_4 L_{00} - c_5 L_{20} \end{pmatrix}$$

$$c_1 = 0.429043 \quad c_2 = 0.511664$$
$$c_3 = 0.743125 \quad c_4 = 0.886227 \quad c_5 = 0.247708 \tag{12}$$

The entries of M depend[2] on the 9 lighting coefficients L_{lm} and the expressions for the spherical harmonics. The constants come from the numerical values of $\hat{A}_l$ given in equation 9, and the spherical harmonic normalizations given in equation 3.

On systems not optimized for matrix and vector operations, it may be more efficient to explicitly write out equation 7 for the irradiance as a sum of terms, i.e. expand equation 12:

$$\begin{aligned} E(\mathbf{n}) &= c_1 L_{22}\left(x^2 - y^2\right) + c_3 L_{20} z^2 + c_4 L_{00} - c_5 L_{20} \\ &+ 2c_1\left(L_{2-2}xy + L_{21}xz + L_{2-1}yz\right) \\ &+ 2c_2\left(L_{11}x + L_{1-1}y + L_{10}z\right) \end{aligned} \tag{13}$$

We implemented equations 11 and 13 as procedural shaders in the

[2]A symmetric 4x4 matrix has 10 degrees of freedom. One additional degree is removed since $\mathbf{n}$ lies on the unit sphere.

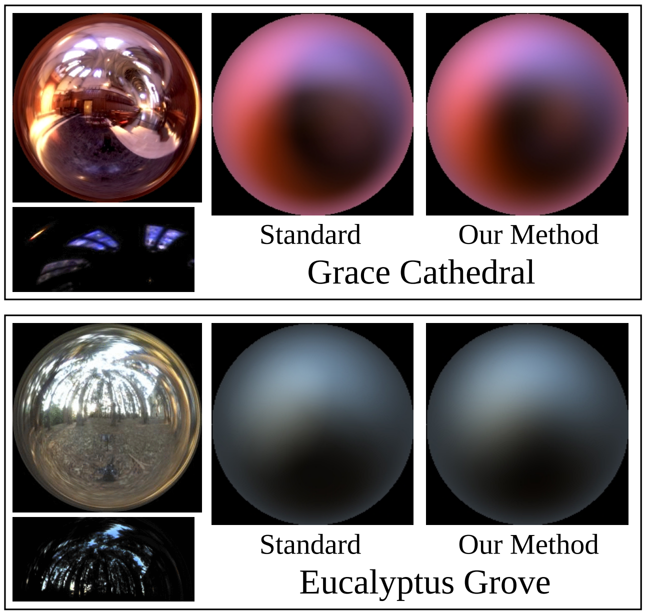


Figure 3: *A comparison of irradiance maps from our method to those from standard prefiltering. The irradiance map resolutions are 256x256. For each light probe, the left image is a tone-mapped version of the environment. Below that, we show the brightest parts of the environment on a linear scale. Both environments have bright bluish lights—from stained-glass windows, and the sky respectively—which are not apparent in the tone-mapped images. This accounts for the bluish portions of the irradiance maps. It can be seen that our method produces a result very close to the correct answer. Note that our rendering algorithm does not actually use irradiance maps; we computed them here solely for the purposes of the quality comparison. The coordinate mapping in the images is such that the center of the image is straight forward ($\theta = 0$, the north pole or +Z), the circumference of the image is straight backwards ($\theta = \pi$, the south pole or -Z), and θ varies uniformly in the radial direction from 0 to π. The azimuthal angle ϕ corresponds to the image polar angle.*

Stanford real-time programmable shading system [15]. We used the ability of that system to perform computations per-vertex. Since E varies slowly, this is adequate and the shading is insensitive to how finely the surfaces are tessellated. The irradiance computations may be performed in software or compiled to vertex programming hardware, if available. The simple forms of equations 11 and 13 indicate that a per-fragment method could also be implemented in programmable hardware.

We were able to achieve real-time frame rates on PCs and SGIs. As shown in the accompanying video—available on the SIGGRAPH 2001 conference proceedings videotape—we can interactively rotate objects and move our viewpoint, with the irradiance being procedurally recomputed at every frame. We can also rotate the lighting by applying the inverse rotation to the normal $\mathbf{n}$. Images rendered using our method look identical to those obtained by texture-mapping after precomputing irradiance environment maps.

3.3 Representation

Conceptually, the final image is composed of a sum of spherical harmonic basis functions, scaled by the lighting coefficients L_{lm}. These 3D irradiance basis functions depend on the surface normal and are defined over the entire object, making it possible to generate an image from any viewpoint. We may also manually adjust the 9 lighting coefficients L_{lm} to directly control appearance, as shown in figure 4. The lighting coefficients can often be assigned intuitive meanings. For instance, L_{1-1} is the moment about the vertical or y-axis, and measures the extent to which the upper hemisphere is brighter than the lower hemisphere. As can be seen from figure 2, L_{1-1} is usually large and positive, since most scenes are lit from above. By making this value negative, we could give the appearance of the object being lit from below.

Our representation may also be useful in the future for *image-based rendering with varying illumination*. Hallinan [10] and Epstein et al. [7] have observed empirically that, for a given view, images of a matte object under variable lighting lie in a low-

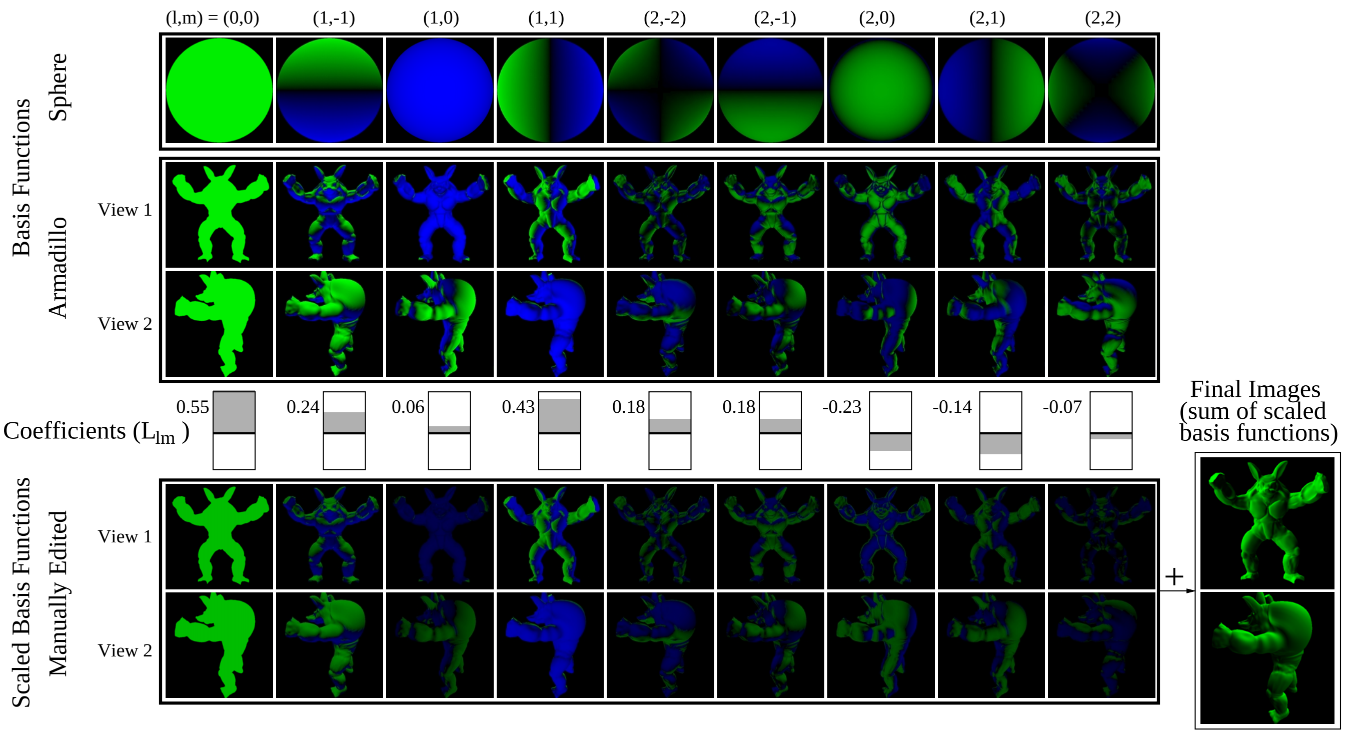


Figure 4: *Illustration of our new representation, and applications to controlling appearance. The basis functions have both positive values, shown in green, and negative values, shown in blue. Topmost, we show the spherical harmonic basis functions on a sphere—note that these are actual images, not the coordinate mappings of figure 3—and the armadillo. The basis functions are defined over the entire object surface; we show only two views. The rightmost 5 functions are dimmer since they have the highest frequency* $(l = 2)$ *and contribute the least. Conceptually, the basis functions are then scaled by the lighting coefficients* L_{lm} *and added to produce renderings.* L_{lm} *are actually RGB values; for simplicity, we show the coefficients for only one color (green). The coefficients* L_{lm} *may be adjusted manually to manipulate appearance. This editing can be fairly intuitive—for instance, we make* L_{11} *large and positive to darken the right side (with respect to us) and left arm of the armadillo image, since the basis function* $(1,1)$ *is negative in that region.*

dimensional subspace. Our theory explains this observation, and indicates that a 9D subspace suffices. Basri and Jacobs [2] have obtained similar theoretical results. To synthesize images of a diffuse object under arbitrary illumination, we therefore need only the 9 basis functions, which could be computed from a small number of photographs. Such an approach would significantly speed up both acquisition and rendering in a method such as Debevec et al. [6].

4 Conclusions and Future Work

We have described a novel analytic representation for environment maps used to render diffuse objects, and have given explicit formulae for implementation. Our approach allows us to use an arbitrary illumination distribution for the diffuse component of the BRDF, instead of the limitation of current graphics hardware to point or directional sources. We simply specify or compute the first 9 moments of the lighting. Even where more conventional texture-mapping methods are desired, our approach allows us to very efficiently compute irradiance environment map textures. In the future, we wish to develop similar frequency-space methods for the specular BRDF component, and more general non-Lambertian BRDFs. We would also like to further explore the applications to lighting design and image-based rendering discussed above.

Acknowledgements: We thank Kekoa Proudfoot and Bill Mark for explaining the details of the Stanford Real-Time Programmable Shading system, and helping us implement the algorithms within its framework. Thanks also to all those who read early drafts of the paper—Bill Mark, Kekoa Proudfoot, Sean Anderson, David Koller, Ron Dror, and the anonymous reviewers. Finally, we thank Venkat Krishnamurthy for providing the armadillo model and Paul Debevec for allowing us to use his light probes (available from http://www.debevec.org). This work was supported in part by a Hodgson-Reed Stanford graduate fellowship and NSF ITR grant #0085864 “Interacting with the Visual World.”

References

[1] J. Arvo. Applications of irradiance tensors to the simulation of non-lambertian phenomena. In *SIGGRAPH 95*, pages 335–342, 1995.

[2] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *International Conference on Computer Vision*, 2001.

[3] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19:542–546, 1976.

[4] B. Cabral, N. Max, and R. Springmeyer. Bidirectional reflection functions from surface bump maps. In *SIGGRAPH 87*, pages 273–281, 1987.

[5] B. Cabral, M. Olano, and P. Nemec. Reflection space image based rendering. In *SIGGRAPH 99*, pages 165–170, 1999.

[6] P. Debevec, T. Hawkins, C. Tchou, H.P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH 00*, pages 145–156.

[7] R. Epstein, P.W. Hallinan, and A. Yuille. 5 plus or minus 2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *IEEE Workshop on Physics-Based Modeling in Computer Vision*, pages 108–116, 1995.

[8] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics & Applications*, 6(11):21–29, 1986.

[9] G. Greger, P. Shirley, P. Hubbard, and D. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, 1998.

[10] P.W. Hallinan. A low-dimensional representation of human faces for arbitrary lighting conditions. In *CVPR 94*, pages 995–999, 1994.

[11] J. Kautz, P. Vázquez, W. Heidrich, and H.P. Seidel. A unified approach to prefiltered environment maps. In *EuroGraphics Rendering Workshop 00*, pages 185–196, 2000.

[12] P. Lalonde and A. Fournier. Filtered local shading in the wavelet domain. In *EGRW 97*, pages 163–174, 1997.

[13] T.M. MacRobert. *Spherical harmonics; an elementary treatise on harmonic functions, with applications*. Dover Publications, 1948.

[14] G. Miller and C. Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. *SIGGRAPH 84 Advanced Computer Graphics Animation seminar notes*, 1984.

[15] K. Proudfoot, W. Mark, S. Tzvetkov, and P. Hanrahan. A real-time procedural shading system for programmable graphics hardware. In *SIGGRAPH 01*, 2001.

[16] R. Ramamoorthi and P. Hanrahan. On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. *To appear, Journal of the Optical Society of America A*, 2001.

[17] F. X. Sillion, J. Arvo, S. H. Westin, and D. Greenberg. A global illumination solution for general reflectance distributions. In *SIGGRAPH 91*, pages 187–196.

[18] G. Ward and P. Heckbert. Irradiance gradients. In *EGRW 92*, pages 85–98, 1992.

[19] A. Wilkie, R. Tobler, and W. Purgathofer. Orientation lightmaps for photon radiosity in complex environments. In *CGI 00*, pages 279–286, 2000.

Polynomial Texture Maps

Tom Malzbender, Dan Gelb, Hans Wolters

Hewlett-Packard Laboratories[1]
http://www.hpl.hp.com/ptm

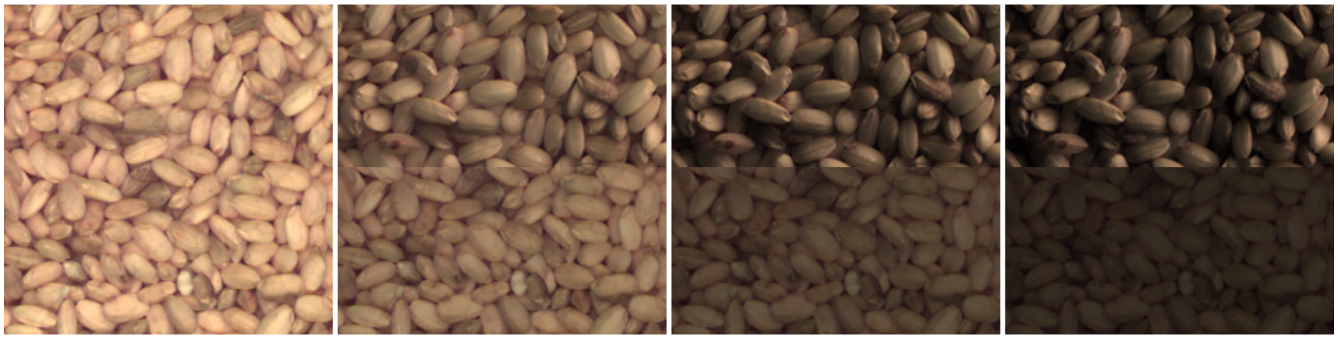


Figure 1: Top: Lighting changes across a polynomial texture map, bottom: across a conventional texture map**.**

Abstract

In this paper we present a new form of texture mapping that produces increased photorealism. Coefficients of a biquadratic polynomial are stored per texel, and used to reconstruct the surface color under varying lighting conditions. Like bump mapping, this allows the perception of surface deformations. However, our method is image based, and photographs of a surface under varying lighting conditions can be used to construct these maps. Unlike bump maps, these Polynomial Texture Maps (PTMs) also capture variations due to surface self-shadowing and interreflections, which enhance realism. Surface colors can be efficiently reconstructed from polynomial coefficients and light directions with minimal fixed-point hardware. We have also found PTMs useful for producing a number of other effects such as anisotropic and Fresnel shading models and variable depth of focus. Lastly, we present several reflectance function transformations that act as contrast enhancement operators. We have found these particularly useful in the study of ancient archeological clay and stone writings.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism - Color, shading, shadowing and texture; I.4.1 [Image Processing and Computer Vision] Digitization and Image Capture - Reflectance

Keywords: Graphics Hardware, Illumination, Image Processing, Image-Based Rendering, Reflectance & Shading Models, Texture Mapping

[1] {malzbend,dgelb,wolters}@hpl.hp.com

1 Introduction

Polynomial Texture Mapping enables greatly improved realism over conventional methods. Traditional texture mapping is used to give the impression of geometric detail in a model using an image. For example, a photograph of a brick wall may be used as a texture map on a planar surface to avoid modeling the complex surface detail of the brick. However, if the lighting in the synthetic environment where the texture map is used is different from the lighting the texture map was captured under, the resulting rendering will appear incorrect and unrealistic. Worse yet, when the texture is blended with the calculated lighting of a geometric surface, the resulting rendering will look very flat and smooth to the viewer.

Bump mapping [Blinn 78] is one proposed solution to this problem where the surface normals of underlying geometry are allowed to vary per texture element (texel). Introducing variations in the surface normals causes the lighting method to render the surface as though it had local surface variations instead of just a smooth surface. As a result, when the light is moved around the object, highlights appear due to the bump map and the surface appears to be rough, grooved or similarly modified as desired. Bump maps can be either hand modeled or, more typically, calculated procedurally. Creating a bump map to be used with real world textures from photographs is generally difficult. Methods have been developed that attempt to automatically generate bump maps from a set of input images under known light directions [Rushmeier 97]. These methods have difficulty with generating bump maps for objects with large surface variations that cause self-shadowing and intra-object interreflections. In addition, current bump map rendering techniques do not render shadows due to surface variation or brightened regions due to interreflections.

In contrast, our method is an image-based technique that requires no modeling of complex geometry or bump maps. The input data required is a set of images of the desired object to be used as a texture, each one under illumination from a different known direction, all captured from the same view point. The input light directions need not be uniformly spaced in the hemispherical set of possible light directions. We choose to represent the variation in surface color for each pixel independently with a simple biquadratic polynomial. Although approximate, this representation is compact and allows fast color reconstruction during rendering. One implementation of our method uses a polynomial to approximate the luminance of each texel, keeping the chromaticity constant. The result of our method is a texture map that properly reproduces the effects of variations in the illuminant direction relative to the object, whether due to the surface orientation of the texture-mapped object, or to changing of the location of the source. Intensity and color variations that are also due to self shadowing, sub-surface scattering and interreflections are also captured and modeled by PTMs. Figure 1 compares our method to conventional texture mapping as lighting varies. Renderings using our method are very realistic, and require little or no user input once the input images are acquired.

2 Background and Previous Work

The characterization of surface reflectance properties is essential to achieving photorealistic renderings. The Bidirectional Reflectance Distribution Function [Nicodemus 77] characterizes the color of a surface as a function of incident light (Θ_i, Φ_i) and exitant view (Θ_e, Φ_e) directions.

$$BRDF(\Theta_i, \Phi_i, \Theta_e, \Phi_e, \lambda) \tag{1}$$

The BRDF is the ratio of the reflected intensity in the exitant direction to the incident energy per unit area along the incident direction. It does contain a dependence on wavelength, λ, but in practice this is often approximated by independent BRDFs per color channel. [Marschner 99] presents an efficient method of collecting BRDFs from samples of curved surfaces with uniform reflectance properties. There have been a large number of techniques developed to accurately and compactly represent the 4D (per discrete wavelength) BRDF. These include linear basis functions such as spherical harmonics [Cabral 87], [Sillion 91], [Wong 97], physically based analytic models [He 91], [Stam 99], and empirical models [Phong 75], [LaFortune 97].

[Dana 99] defines the Bidirectional Texture Function (BTF) by allowing the BRDF to vary spatially across a surface parameterized by *u,v*.

$$BTF_{r,g,b}(\Theta_i, \Phi_i, \Theta_e, \Phi_e, u, v) \tag{2}$$

The BTF as defined by [Dana 99] does not actually store the ratio of exitant to incident energy like the BRDF. Instead, the BTF captures the pre-integrated lighting condition for a particular light source. They provide samples of the BTF from photographs, but face two difficulties due to the high dimensionality of the model. First, each photograph of a texture patch can be seen spanning *u,v*, but only point sampling the remaining four dimensions. Numerous photographs will be required to adequately sample this space. Second, camera pose must be accurately calibrated to allow measurements across the viewing dimensions. We avoid these difficulties by holding two of these dimensions constant, namely the exitant direction. Each photograph now becomes a sample of a 2-dimensional space, and the need for camera calibration is avoided entirely since the viewpoint does not change.

This is similar to the approach taken by both [Debevec 00] and [Georghiades 99] where a per pixel reflectance model is acquired for a static scene, in particular human faces for those references. In our work we advocate polynomial models for these reflectance functions and their application in real-time rendering as texture maps. The per pixel reflectance maps that we collect have the following dependence,

$$I_{r,g,b}(\Theta_i, \Phi_i, u, v) \tag{3}$$

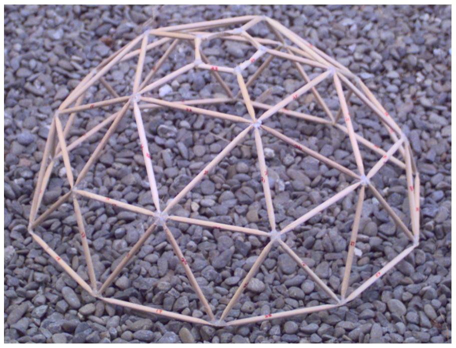
namely two spatial coordinates *u,v* and two parameters, Θ_i, Φ_i encoding the direction of the incident illumination. By not including the exitant direction dependence, we sacrifice the ability to capture view dependent effects such as specularity, but retain the ability to represent arbitrary geometric shadowing and diffuse shading effects across a surface. For the remainder of the paper, we assume that the surfaces being photographed are either diffuse, or their specular contributions have been separated out through the use of polarizers on both the light sources and cameras [Debevec 00]. Although the acqusition methodology we will present is limited to diffuse objects, PTMs can be used to render specular as well as diffuse effects. This is described in Section 3.5.

Other image-based relighting methods have been developed that also allow a scene to be rendered under novel lighting conditions, based on a set of input images acquired under known illuminants. These methods take advantage of the linearity of light to generate output renderings. [Nimeroff 94] and [Teo 97] showed how basis images can be linearly combined to generate rerendered synthetic scenes. [Nishino 99] describes a method where input range and color images are compressed in eigenspace. Images under novel lighting conditions are generated by interpolating in eigenspace and subsequent mapping from eigenspace into image space. [Wood 00] applies surface light fields inferred from photographs and laser range scans to generate view dependent effects. [Georghiades 99] applies image-based relighting to real human faces, adding the constraint that the surface reflectance is Lambertian to eliminate the requirement that the input lighting conditions are known. [Debevec 00] is able to render faces with more general reflectance functions after acquiring a large number of images under known lighting. Employing high dynamic range imaging, this method is extended to render arbitrary objects under novel illumination conditions. [Wong 97] creates a 6D image-based representation, an extension to [Levoy 96][Gortler 96], to render objects with novel lighting from new viewpoints. [Epstein 95] points out that low dimensional lighting models are adequate to model many objects, a result confirmed in this work as well as [Ramamoorthi 01].

3 Polynomial Texture Maps

3.1 Photographic Acquisition of PTMs

As done by [Debevec 00], [Georghiades 99], [Epstein 96] and in the field of photometric stereo in general, we collect multiple images of a static object with a static camera under varying lighting conditions. Figure 2 shows two devices we have constructed to assist with this process. The first is a simple once-subdivided icosahedral template that assists in manually positioning a light source in 40 positions relative to a sample.



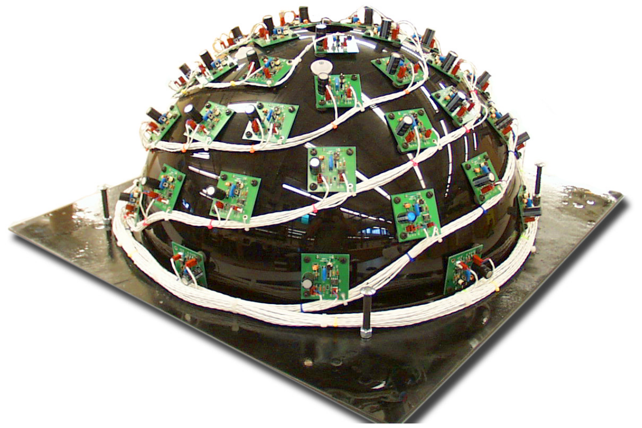


Figure 2: Two devices for collecting PTMs.

Although very simple, this method is capable of achieving good results, such as the archeological samples shown later in the paper. The second device allows fully automated acquisition of 50 source images, each illuminated with an individual strobe light source. In both cases the camera (not shown) is mounted in the apex of the dome and samples are placed on the floor. In this manner multiple registered images are acquired with varying light source direction. Note that since the camera is fixed we avoid the need for any camera calibration or image registration.

3.2 Polynomial Color Dependence

Interpolating these input images to create textures from arbitrary light directions would be very costly both in memory and bandwidth. For each texel in our texture map we would be required to store a color sample for each input light position. One source of redundancy among these images is that the chromaticity of a particular pixel is fairly constant under varying light source direction; it is largely the luminance that varies. We take advantage of this redundancy by computing an unscaled color per texel $(R_n(u,v),G_n(u,v),B_n(u,v))$ that is modulated by a luminance model, L(u,v) again dependent on the texel:

$$\begin{aligned} R(u,v) &= L(u,v)R_n(u,v); \\ G(u,v) &= L(u,v)G_n(u,v); \\ B(u,v) &= L(u,v)B_n(u,v); \end{aligned} \qquad (4)$$

We prefer this simple but redundant representation over color spaces such as LUV and YC_bC_r due to the low cost of evaluation, although we have implemented the method in these color spaces as well. For diffuse objects, we have also found the dependence of luminance on light direction to be very smooth, even for very textured objects with high *spatial* frequencies. We choose to model this dependence with the following biquadratic per texel:

$$\begin{aligned} L(u,v;l_u,l_v) = a_0(u,v)l_u^2 + a_1(u,v)l_v^2 + \\ a_2(u,v)l_ul_v + a_3(u,v)l_u + a_4(u,v)l_v + a_5(u,v) \end{aligned} \qquad (5)$$

where (l_u,l_v) are projections of the normalized light vector into the local texture coordinate system *(u,v)* and *L* is the resultant surface luminance at that coordinate. The local coordinate system is defined per vertex, based on the normal and on the tangent and binormal derived from the local texture coordinates. Coefficients $(a_0$-$a_5)$ are fit to the photographic data per texel and stored as a spatial map referred to as a Polynomial Texture Map. Given N+1 images, for each pixel we compute the best fit in the L_2 norm using singular value decomposition (SVD) [Golub 89] to solve the following system of equations for a_0-a_5.

$$\begin{bmatrix} l_{u0}^2 & l_{v0}^2 & l_{u0}l_{v0} & l_{u0} & l_{v0} & 1 \\ l_{u1}^2 & l_{v1}^2 & l_{u1}l_{v1} & l_{u1} & l_{v1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{uN}^2 & l_{vN}^2 & l_{uN}l_{vN} & l_{uN} & l_{vN} & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_5 \end{bmatrix} = \begin{bmatrix} L_0 \\ L_1 \\ \vdots \\ L_N \end{bmatrix} \qquad (6)$$

Note that the SVD needs to be computed only once given an arrangement of light sources and then can be applied per pixel. The quadratic model proposed in Eq. 5 provides only an approximate fit to the observed color values. Note however that the resultant smoothing that occurs due to this approximation manifests itself only across the space of light directions and does not introduce any spatial blurring. This light space smoothing can have the effect of muting sharp specularities, softening hard shadows and essentially changing point light sources to area lights. However, arbitrarily high spatial frequencies in the original source photographs are preserved. Furthermore, we have verified that the general shape of the function described by the input data is well preserved. We have computed the root mean square error over all the pixels, and obtained a maximum error of roughly 10 per 8-bit color channel for typical examples such as the seeds shown in Figure 1 using the RGB PTM described below. Figures 5a-b and 6a-b show examples of a source photograph and a PTM reconstruction for the same light direction. Note the minimal loss in quality.

The representation just described is called an LRGB PTM since it explicitly separates and models luminance per texel. We have found that representing each color channel directly with a biquadratic polynomial is useful as well, especially for applications where we are modeling variations of pixel color due to other parameters besides incident lighting direction. We call this format an RGB PTM and a specific application is the depth of focus PTMs described in Section 4.2.

3.3 Scale and Bias

The results of the fitting operation described above are six floating-point coefficients per texel. We would like to store these as 8 bit integers for evaluation speed and so that all polynomial coefficients can be stored compactly. This is a non-trivial problem since there are typically several orders of magnitude difference between high and low order coefficients. To eliminate this problem we store 6 scale (λ) and bias (Ω) values with each PTM, one for each coefficient. During reconstruction these values are applied to the stored 8 bit coefficients, a_i', to recover their final values a_i:

$$a_i = \lambda(a_i' - \Omega) \qquad (7)$$

3.4 Hardware Implementations

PTMs were specifically designed to be implemented in VLSI. It is well known that fixed point operations generally require a smaller gate count than their floating point equivalents when the operands need limited precision. Similarly, multiplication and addition are straightforward and yield compact designs compared to the operations of division, square root, exponentiation, etc. It follows that evaluating low order polynomials consisting of fixed point coefficients and arguments can be done at low hardware cost. In particular, the per-pixel costs involved in evaluating the PTM for a given light direction consist of 11 multiplies and 5 adds[2], each of which can be done at low precision fixed point in parallel. This low complexity has associated speed advantages that allow PTM evaluations to be performed at high pixel fill rates.

We have developed an interactive software viewer with Intel MMX™ optimizations for interacting with PTMs directly. SIMD MMX™ multiply-add instructions allow four PTM coefficient terms to be evaluated simultaneously. We have also created an implementation that makes use of OpenGL hardware with extensions. Current consumer graphics cards have sufficient functionality in programmable texture operations and programmable vertex processing to perform our algorithm in hardware. The parameters (l_u,l_v) are calculated in the vertex-processing pipeline, scan converted, and passed to the texture stage via the color channels. Two texture maps are used to store the coefficients of the polynomial in Eq. 5. The polynomial is evaluated per texel in the programmable texture stage.

3.5 Converting Bump Maps to PTMs

An alternative to photographically producing PTMs with the methods described above is to use existing bump maps and convert them into PTMs. This provides a method for rendering bump maps with any graphics hardware that renders PTMs directly. The approach we have developed uses lookup tables and is capable of converting bump maps as they are read from disk. We compute a single PTM from the bump map, which will be evaluated twice, first using the light direction, L, to yield the diffuse contribution and second using the halfway vector H (between the light source and the viewer) and exponentiation to yield the specular contribution.

Starting from the Phong Lighting equation:

$$I = I_a k_a + I_d k_d (N \cdot L) + I_s k_s (N \cdot H)^n \qquad (8)$$

we note that the diffuse and specular contributions at a texel are dependent exclusively on dot products which involve the normal ***N*** at that particular location. Let us define a general function *F( )* as the dot product between a surface normal vector ***N*** and general normalized vector described by (l_u,l_v):

$$F(l_u,l_v) = l_u N_u + l_v N_v + \sqrt{1 - l_u^2 - l_v^2}\, N_w \qquad (9)$$

The goal is to approximate this function with the biquadratic polynomial $L(l_u,l_v)$ introduced in Eq. 5. This means that we need to minimize the quantity

$$\int_{-1}^{1}\int_{-1}^{1} (L(l_u,l_v) - F(l_u,l_v))^2 \, dl_u dl_v \qquad (10)$$

which is the square of the continuous L_2 norm. Note that for simplicity, we are minimizing over the domain $[-1,1]^2$ even though we are only interested in the smaller domain $C=\{l_u,l_v : l_u^2+l_v^2 <= 1\}$ of the unit circle. The reason is that the natural basis functions for domain *C* are spherical harmonics [Born 80] which involve the computation of trigonometric functions. In order to minimize the expression above, we first transform the polynomial basis $B=\{1, l_u, l_v, l_u l_v, l_u^2, l_v^2\}$ into a basis *B'* which is constructed to be orthonormal on $[-1,1]^2$.

$$B' = \{\frac{1}{2}, \frac{\sqrt{3}}{2} l_u, \frac{\sqrt{3}}{2} l_v, \frac{3}{2} l_u l_v, \frac{\sqrt{45}}{4} l_u^2 - \frac{\sqrt{45}}{12}, \frac{\sqrt{45}}{4} l_v^2 - \frac{\sqrt{45}}{12}\} \qquad (11)$$

By using fundamentals from Approximation Theory [Watson 80], we know that the coefficients a'_i of the best approximant *L'* can be computed directly as values of integrals:

$$a'_i = \int_{-1}^{1}\int_{-1}^{1} L'_i(l_u,l_v) F(l_u,l_v) \, dl_u dl_v \qquad (12)$$

where L'_i is the *i*-th basis polynomial of *B'*. The final step then is to transform the a'_i back into a_i, the PTM coefficients associated with a particular projected surface normal.

$$\begin{aligned}
a_0 &= \frac{1}{2} a_0' - \frac{\sqrt{45}}{12}(a_5' + a_4') \\
a_1 &= \frac{\sqrt{3}}{2} a_1' \\
a_2 &= \frac{\sqrt{3}}{2} a_2' \\
a_3 &= \frac{3}{2} a_3' \\
a_4 &= \frac{\sqrt{45}}{4} a_4' \\
a_5 &= \frac{\sqrt{45}}{4} a_5'
\end{aligned} \qquad (13)$$

This computation only needs to be done once for all bump maps, and the results put into a lookup table which will convert bump map normals into PTMs. In summary the steps involved are:

1) Precompute lookup table. We sample the space of normals on the hemisphere by subdividing the range of latitude and longitude uniformly. Here *i* denotes the index in the latitudinal direction and *j* denotes the index in the longitudinal direction. For each normal $N_{i,j}$ we compute the polynomial coefficients as described above and store them in a look-up table. Note that this step is independent of the particular bump map and thus needs to be performed only once as a preprocess.

2) Convert height field: Given a height field bump map and user-defined scale factor, we compute surface normals per texel [Blinn 78].

3) Convert normals to PTM: Using the longitude and latitude of the surface normal as an index into the precomputed lookup table, we recover the PTM coefficients per texel. Here we make use of the fact that the polynomial is linear in its coefficients and hence we can perform six bilinear interpolations to compute the final polynomial.

[2] 8 multiplies and 5 adds for the luminance polynomial evaluation and 3 more multiplies to get color per pixel.

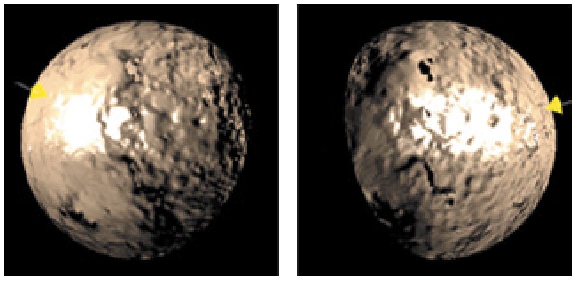


Figure 3: Bump Mapping with PTMs, two light positions.

4) Render the PTM: For each pixel we have computed a polynomial that approximates the dot product of the bump map normal with any other given normalized vector. That means that we can render the bump map by evaluating this polynomial for light vector (diffuse) as well as halfway vector (specular). This is illustrated in Figure 3. It shows two images of a bump map textured onto a sphere. The original data is a height field capturing data simulating the surface of Mars. The sphere is lighted with specular and diffuse terms under varying light positions.

3.6 PTM Filtering

One of the difficulties with using traditional bump maps to display surface detail is filtering them to match a particular sampling rate. Approximate solutions are discussed in [Kilgard 00]. The naive approach using mip-maps winds up smoothing the effective surface, removing the bumps themselves [Schilling 97]. The function that we would like to integrate over is the light radiated from a bumpy patch, instead of surface perturbations themselves. Since PTMs are image-based representations, we can achieve this effect by mip-mapping the polynomial coefficients (a_0-a_5) directly. At first this may seem odd, but a glance at Eq. 5 reveals that the light dependence is linear in the polynomial coefficients. Integrating over a patch Ω with n samples, we have:

$$\frac{1}{n}\sum_{i,j\in\Omega} L_{r,g,b}(a_{0-5}(u_i,v_j)) = L_{r,g,b}\left(\frac{1}{n}\sum_{i,j\in\Omega} a_{0-5}(u_i,v_j)\right) \tag{14}$$

This states that accumulating polynomial coefficients over a patch and then evaluating the sum is equivalent to accumulating the colors each texel contributes when independently evaluated. Not only does this allow us to mip-map the coefficients directly, but nonisotropic filtering techniques that yield improved image quality such as footprint assembly [Schilling 96] can be supported.

3.7 Anisotropic / Fresnel / Off-Specular

The light vector dependent PTMs that we have described can be combined with existing per-vertex or per-pixel lighting hardware to generate a variety of useful effects. One method is to modulate the specular component of standard lighting computations by the result of evaluating the PTM. The existing Phong lighting equation becomes:

$$I = I_a k_a + I_d k_d (N \cdot L) + I_s k_s (N \cdot H)^n PTM(u,v,l_u,l_v) \tag{15}$$

In the trivial case where the PTM varies spatially but is constant for all incident light directions this is equivalent to specular maps or gloss maps [Blythe 99]. The dependence on light direction that is not available with standard specular maps could be used to reproduce effects such as shadowing where texels do not exhibit specular highlights due to shadowing by small scale surface variations.

Many surfaces exhibit increased specularity for low angles of incidence due to decreases in the apparent roughness [Lafortune 97], a case of off-specular reflections. This can be reproduced using our technique by modulating a Phong specular lobe with a PTM whose magnitude increases as the incident angle approaches grazing.

Incident illumination dependence can also be used to approximate Fresnel effects. Fresnel effects are important for many surfaces, such as metals and glass where reflectance increases greatly for grazing incident angles. The reflectance of glass and other dielectrics are low when the incident illumination direction is near the surface normal. Metals also exhibit Fresnel reflectance, including wavelength dependence [Hall 89]. The color of reflected light changes as the incident angle changes due to reflectance variation for different wavelengths. These dependencies can be approximated by the polynomial function stored at each texel. If color changes over varying incident angles need to be captured, then RGB PTMs may be necessary. The combination of the illumination dependent effects that we described can all be stored in a single PTM. In addition, since texel polynomials in the PTM are independent, these properties can vary across the surface. A PTM can represent different materials in a single texture map.

Anisotropic materials can also be modeled using PTMs. Techniques to render anisotropy such as [Banks 94], [Stalling 97] and [Heidrich 99] define normal planes for surface points aligned with the direction of anisotropy. The projection of the incident light direction into the normal plane is then used in the lighting calculations. This then drops off as the incident light direction moves away from the preferred direction. For our technique each texel in the PTM stores the magnitude of the light vector projected onto the normal plane as a function of incident light direction. When the incident light direction is aligned near the preferred direction of anisotropy the evaluated PTM polynomial has a large magnitude. The result of evaluating the PTM is then modulated with calculated specular lighting so that specular highlights occur only in regions where the incident light direction aligns with the direction of anisotropy. Our technique allows us to render anisotropic surfaces under perspective views with local light sources, and spatially variant BRDFs in hardware. Figure 4 shows an example of a disc rendered with a synthetic anisotropic PTM.

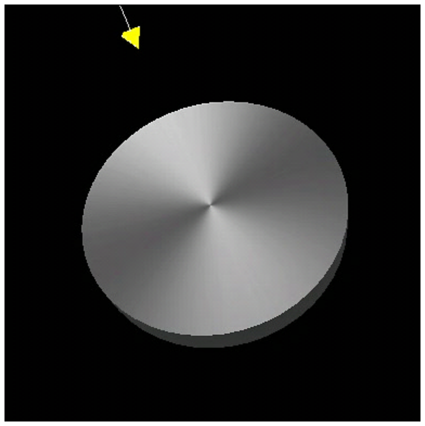


Figure 4: Anisotropic Disc

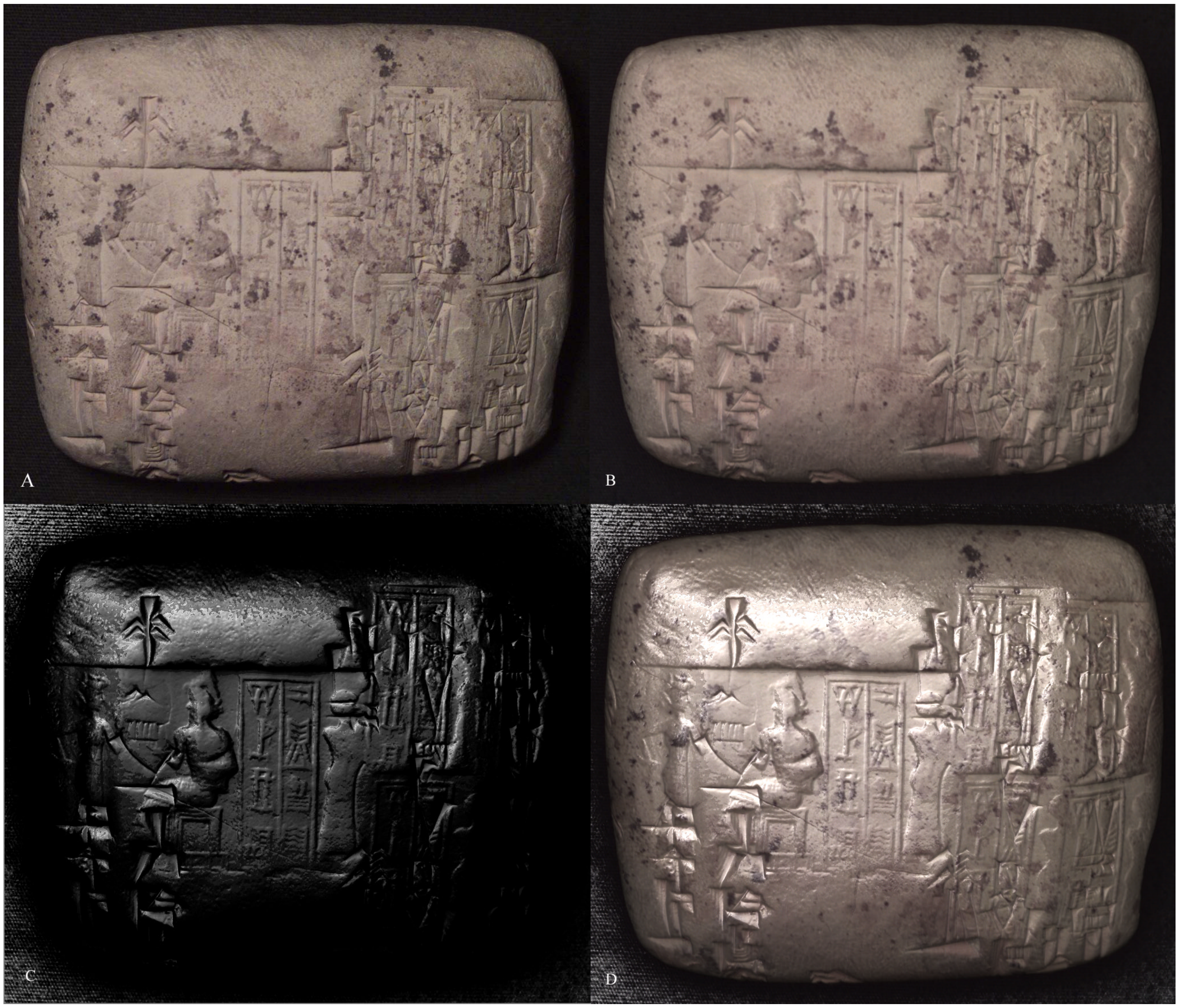


Figure 5: Specular enhancement: (A) Original Photograph (B) Reconstruction from PTM. (C) An image computed by extracting surface normals for each pixel and applying a specular lighting model per pixel. (D) Highlights computed in (C) added to (B). Light direction is the same for all four images. This artifact is a 4000 year old neo-Sumerian tablet.

4 2D Applications of PTMs

Although PTMs were developed as an extension to texture mapping, we have found them to have a number of uses in 2-dimensional applications where images are commonly employed. The next section discusses new contrast enhancement mechanisms and their application in the study of ancient writings. Additionally we demonstrate photographic depth of focus effects using PTMs and the mapping of short image sequences to PTMs.

4.1 PTMs for Contrast Enhancement

We have found that PTMs provide a valuable representation for the study and archiving of ancient artifacts, in particular early clay writings [Malzbender 00]. PTMs allow interactive control of lighting conditions that enable greatly increased perception of surface structure compared to photographs of these artifacts. In addition, we have developed three novel contrast operators, two of which rely on surface normals extracted from the coefficients themselves. Assuming a diffuse surface being photographed under the variable lighting apparatus shown in Figure 2, we can extract an estimate of the surface normal per pixel by solving for the

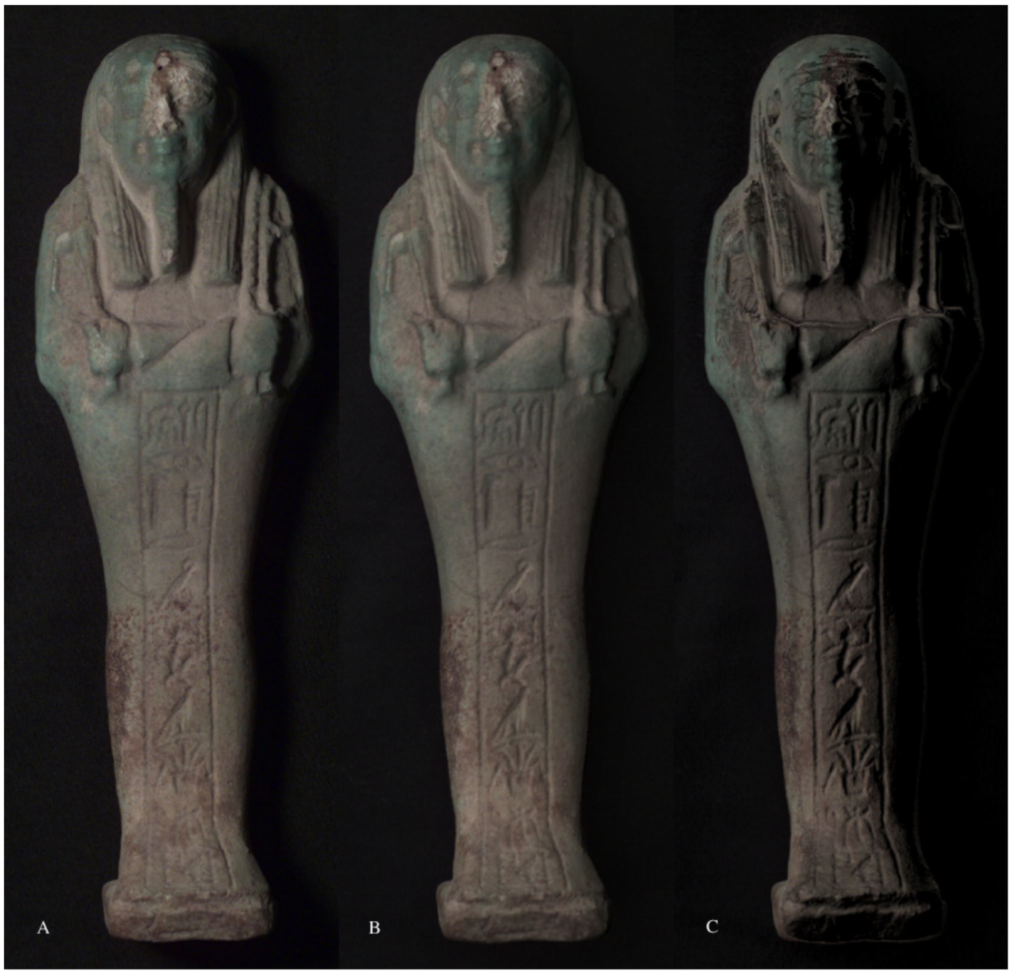


Figure 6: Diffuse gain shown on a 3000 year old Egyptian funerary statuette: (A) Original photograph. (B) Reconstruction from PTM. (C) Diffuse gain—a transformation that exaggerates the diffuse reflectance properties by a gain factor (g = 1.9 here), keeping the surface normal estimate constant. These images have identical light directions and intensities.

maximum luminance of Equation 5. Setting $\frac{\partial L}{\partial u} = \frac{\partial L}{\partial v} = 0$, we arrive at:

$$l_{u0} = \frac{a_2 a_4 - 2a_1 a_3}{4a_0 a_1 - a_2{}^2} \tag{16}$$

$$l_{v0} = \frac{a_2 a_3 - 2a_0 a_4}{4a_0 a_1 - a_2{}^2} \tag{17}$$

for the values of the projected surface normal (l_{u0}, l_{v0}) that maximizes the biquadratic of Eq. 5. The full estimated surface normal is then simply:

$$\vec{N} = (l_{u0}, l_{v0}, \sqrt{1 - l_{u0}{}^2 - l_{v0}{}^2}) \tag{18}$$

Photometric stereo methods from computer vision would provide an alternate approach to this stage of surface normal recovery [Rushmeier 97] and would also be affected by the presence of self-shadowing, as is our method.

Method 1: Specular enhancement. For every texel (*u,v*), the normal vector can be used in a lighting equation (such as Eq. 8) to add either diffuse or specular shading effects to the image. Simulation of specularity is particularly effective at enhancing the perception of surface shape. We are free to change the properties of the surface (specularity k_s, specular exponent *n*) and simulated light source (position, color) under interactive control. Our method is demonstrated in Figure 5 on a 4000 year old neo-Sumerian tablet. We have effectively modified the reflectance properties of the tablet to enhance perception of surface shape, making the inscription more readable.

Method 2: Diffuse gain. For diffuse objects, the original photographed surfaces typically exhibit a gently varying change in surface intensity across light direction that we have fit with a 2-dimensional convex parabola. The Gaussian curvature of this parabola (its second spatial derivative) can be increased arbitrarily by a gain factor *g* by computing new luminance coefficients using the following transformation:

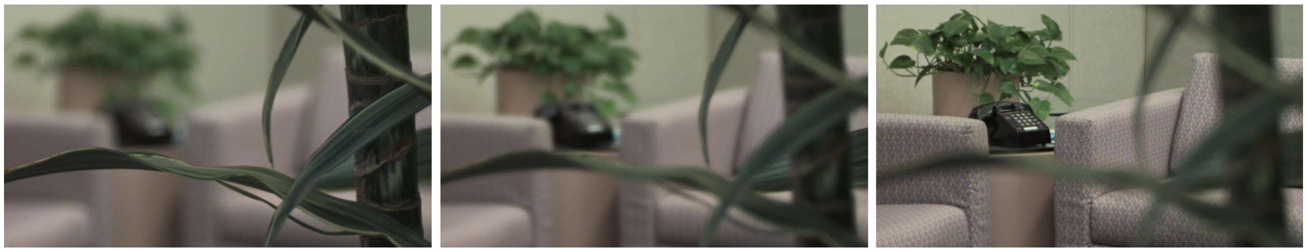


Figure 8: A single PTM can be used to provide views at continuously variable focus depths.

$$\begin{aligned}
a_0' &= g a_0 \\
a_1' &= g a_1 \\
a_2' &= g a_2 \\
a_3' &= (1-g)(2a_0 l_{u0} + a_2 l_{v0}) + a_3 \\
a_4' &= (1-g)(2a_1 l_{u0} + a_2 l_{v0}) + a_4 \\
a_5' &= (1-g)(a_0 l_{u0}^2 + a_1 l_{v0}^2 + a_2 l_{u0} l_{v0}) + \\
&\quad (a_3 - a_3')l_{u0} + (a_4 - a_4')l_{v0} + a_5
\end{aligned} \tag{19}$$

This keeps the luminance maximum at the same location, namely (l_{u0}, l_{v0}), thereby retaining the surface normal direction. It also maintains the luminance at (l_{u0}, l_{v0}) retaining the surface albedo and color in this direction. However, the directional sensitivity of the surface to light is increased by this transformation, enhancing the reflectance properties of the sample. Figure 6 shows the method on a 3000 year old funerary statuette called an *ushabti* from ancient Egypt. Made out of a glazed frit material (faience), its purpose was to carry out work on behalf of the deceased in the netherworld. Note the enhanced definition of the inscription.

Method 3: Light direction extrapolation. Light source direction is a normalized quantity encoded by the projected light direction l_u, l_v. Physically realizable light sources are limited to

$$\sqrt{l_u^2 + l_v^2} \le 1 \tag{20}$$

since the light direction is described by a normalized vector with three real components, as shown in Eq. 18. However, this constraint need not be applied when evaluating PTMs, allowing one to extrapolate the reflectance model beyond what is geometrically achievable. This extrapolation does not have an exact physical analog, but it provides lighting that is more oblique than any used in the original source images, providing enhanced contrast. An example is shown in Figure 7 on a trilobite fossil.

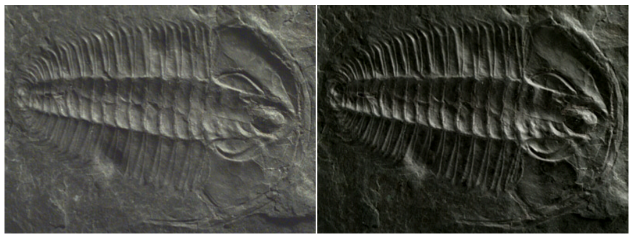


Figure 7: Light Direction Extrapolation. Left: Source photo. Right: Reconstruction with extrapolated light source direction.

4.2 Depth of Focus / Time Varying PTMs

The free parameters l_u, l_v in Equation 5 can be used to interpolate between a number of related images. One useful application is representing a static scene under varying focus conditions. Figure 8 demonstrates this capability. First we photographed the scene 6 times across a range of focus depths. Next we assigned each image a value spanning $(-1.0 < l_u < 1.0)$ holding l_v constant, and fit Eq. 5 for each color channel in each pixel. The resultant RGB PTM now offers a continuously variable depth of field. Note that in this case we have only used one of the two available free variables, l_u, hence fitting only a univariate quadratic PTM. We have found it useful to tie the other variable, l_v, to the camera aperture, allowing control of the spatial extent that is in focus as well as the specific depth that is maximally focused. This capability allows the graphics artist to specify the focus parameters of an image during final composition, not when it is originally photographed. We will see shortly that the additional storage required for this capability is not substantial compared with an individual conventional image.

This capability is not restricted to images that vary by focus parameters. Shown in Figure 9 are 3 reconstructions from a PTM whose input was a sequence of 40 images taken throughout the course of the day as the tide was going out. This RGB PTM is also computed as a univariate quadratic.

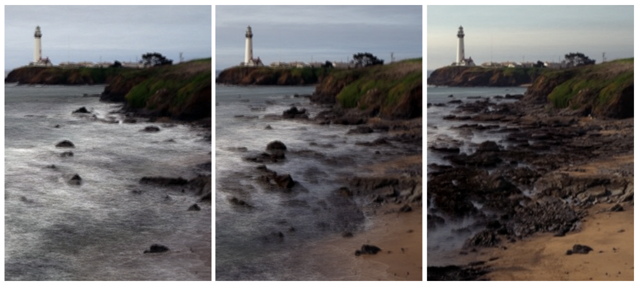


Figure 9: Three reconstructions from a single PTM. Input images from Sundial California published by Aladdin Systems Inc. Copyright © 1996 John M. Neil. Used with permission.

5 Compression / Palletization

The PTMs discussed in this paper are either LRGB or RGB PTMs. For LRGB PTMs we store 9 bytes per pixel (3 colors plus 6 polynomial coefficients) and for RGB PTMs we store 18 bytes per pixel (6 polynomial coefficients per color channel). We have found that both of these representations are highly compressible, due to both spatial correlation and correlations between byte-planes. Compression methods developed for multispectral images can be directly applied.

Our first compression method is to simply palletize the PTMs much as one palletizes images, by computing a color lookup table. Using K-means clustering we compute a lookup table for an LRGB PTM that includes the 6 polynomial coefficients per entry. We have found that a 256 entry table is sufficient to be indistinguishable from the original. This representation allows a one byte value to index the lookup table, with the 3 remaining bytes of color all packed into a 32 bit field. Since 24 bit images are often packed into a 32 bit format anyway, the size of this representation compared to a conventional image grows only by a small amount, the size of the lookup table (1.5 K). The analogous representation for RGB PTMs requires 18 fields (bytes) in the lookup table and we have found that 12 bit indices yield indistinguishable results compared to the original PTMs. Note that this method allows random access and individual PTM textels can be evaluated independently.

Although decoding is simple and fast with this lookup table approach, we have gotten better results by decorrelating the polynomial coefficients into planes and applying JPEG or JPEG-LS to the results [Motta 00]. Visual inspection of the polynomial coefficient planes and RGB planes makes it apparent that strong dependencies and correlations exist between these planes, and that further room for compression exists. Inter-plane prediction is employed to reduce this dependency. Given a reference plane, pixel values in planes being encoded are estimated, resulting in residual images. These are further compressed by intra-plane methods that estimate pixels values based on their causal neighbors. Table 1 below overviews the results for the lossless and perceptually lossless JPEG-LS cases for the intra-pane prediction. Note that this includes both color and polynomial coefficients.

Dataset	lossless	loss = 1	loss = 2	loss = 4
seeds	30.3 bits	19.2 bits	15.7 bits	12.6 bits
ushabti	31.3 bits	19.8 bits	15.9 bits	12.3 bits
tablet	33.8 bits	21.3 bits	16.9 bits	13.0 bits
lighthouse	14.6 bits	8.61 bits	6.33 bits	4.27 bits
trilobite	34.0 bits	21.6 bits	17.1 bits	13.0 bits
focus	20.5 bits	12.3 bits	9.30 bits	5.67 bits
average	**27.4 bits**	**17.1 bits**	**13.5 bits**	**10.1 bits**

Table 1: PTM Compression: Number of bits per texel for varying amounts of imprecision tolerated in each color channel during reconstruction (out of 256 levels). These levels of loss are all imperceptible. Empirically we have found that visible artifacts start appearing at approximately 4 bits per pixel.

6 Conclusions

We have presented a method that requires only images to generate high quality photorealistic renderings of a textured surface. Our method captures light dependent effects, whether they are due to changes in the illuminant direction, or surface orientation of the texture mapped object. It can render changes in brightness due to shadows and indirect lighting that cannot be reproduced with existing bump mapping hardware, or Phong illuminated geometric objects. Our technique could be integrated into non-programmable graphics hardware with minor additional hardware requirements. It can be efficiently implemented using modern programmable shading hardware. Sample PTMs and an interactive browser program for viewing them are available for downloading at http://www.hpl.hp.com/ptm .

PTMs were originally developed to model the dependence of surface color on a parameterization of the light direction for interactive applications. We have shown extensions that allow Fresnel, anisotropic, off-specular, depth of focus and contrast enhancement effects to be realized. Several unexplored uses remain. One possibility is to allow rudimentary modeling of occlusion effects due to camera or object motion by indexing a PTM by the view vector instead of the light direction vector. Another is the application of PTMs to model surface opacity in addition to color. Both of these methods would allow modeling geometry in image-based manner. Lastly, the method presented here could be generalized to higher order polynomials or non-polynomial basis functions.

Acknowledgements

Many thanks to Religious Studies scholar Bruce Zuckerman who provided access to the ancient texts imaged at the Yale Babylonian Collection, as well as the USC archaeology lab. Thanks to Kevin Wu for Mesa implementations and developing the light vector projection approach. Irwin Sobel suggested the approach of palletizing PTMs to reduce storage and Giovanni Motta produced the results on compression using JPEG and JPEG-LS. Paul Debevec provided thoughtful comments on the paper. We also greatly appreciate Fred Kitson's constant support.

References

[Banks 94] Banks, D.C., "Illumination in Diverse Codimensions", *Computer Graphics (SIGGRAPH 94 Proceedings)*, July 1994, pp. 327-334.

[Blinn 78] Blinn, J.F., "Computer Display of Curved Surfaces", Ph.D. Thesis, University of Utah, 1978.

[Blythe 99] Blythe, D., McReynolds, T., "Lighting and Shading Techniques for Interactive Applications", *Course Notes (Siggraph 99 Course 12)*, August, 1999, p. 101.

[Born 80] Born, Max, Wolf, Emil, "Principles of Optics", 6th edition, Appendix VII, Cambridge University Press, Cambridge, 1980.

[Cabral 87] Cabral, B., Max, N., Springmeyer, R., "Bidirectional Reflection Functions from Surface Bump Maps", *Computer Graphics (SIGGRAPH 87 Proceedings),* July 1987, pp. 273-281.

[Dana 99] Dana, K., Van Ginneken, B., Nayar, S., Koenderink, J., "Reflectance and Texture of Real-World Surfaces", *ACM Transactions on Graphics*, Vol. 18, No. 1, January 1999, pp. 1-34.

[Debevec 00] Debevec, P., Hawkins,T., Tchou, C., Duiker, H., Sarokin, W., Sagar, M., "Acquiring the Reflectance Field of a Human Face*", Computer Graphics (SIGGRAPH 2000 Proceedings)*, July 2000, pp. 145-156.

[Epstein 95] Epstein, R., Hallinan, P., Yuille, A., "5 +/- 2 Eigenimages Suffice: An Empircal Investigation of Low-Dimensional Lighting Models, *IEEE Workshop on Physics-Based Vision*: 108-116, 1995.

[Epstein 96] Epstein, R., Yuille, A.L., Belhumeur, P.N., "Learning Object Representations from Lighting Variations", Object Representation in Computer Vision II Workshop, ECCV96, April 1996, pp.179-199.

[Georghiades 99] Georghiades, A., Belhumeur, P., Kriegman, "Illumination-Based Image Synthesis: Creating Novel Images of Human Faces Under Differing Pose and Lighting", *IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes*, 1999, pp. 47-54.

[Golub 89] Golub, G., van Loan, C., "Matrix Computations", Johns Hopkins University Press, Baltimore, 1989.

[Gortler 96] Gortler, S., Grzeszczuk, R., Szeliski, R., Cohen, M., "The Lumigraph", *Computer Graphics (SIGGRAPH 96 Proceedings),* August 1996, pp. 43-54.

[Hall 89] Hall, R. *Illumination and Color in Computer Generated Imagery*, Springer-Verlag New York Inc., New York, 1989, pp. 193-197.

[He 91] He, X., Torrance, K., Sillion, F., Greenberg, D., "A Comprehensive Physical Model for Light Reflection", *Computer Graphics (SIGGRAPH 91 Proceedings),* July 1991, pp.175-186.

[Heidrich 99] Heidrich, W., Seidel, H., "Realistic, Hardware-accelerated Shading and Lighting", *Computer Graphics (SIGGRAPH 99 Proceedings),* August 1999, pp.171-178.

[Kilgard 00] Kilgard. M.,"A Practical and Robust Bump-mapping Technique for Today's GPUs", Game Developers Conference (GDC) 2000: Advanced OpenGL, also available at nvidia.com.

[Lafortune 97] Lafortune, E., Foo, S.-C., Torrance, K., Greenberg, D., "Non-Linear Approximation of Reflectance Functions", *Computer Graphics (SIGGRAPH 97 Proceedings),* August 1997, pp. 117-126.

[Levoy 96] Levoy, M., Hanrahan, P., "Light Field Rendering", *Computer Graphics (SIGGRAPH 96 Proceedings),* August 1996, pp. 31-42.

[Malzbender 00] Malzbender, T., Gelb, D., Wolters, H., Zuckerman, B., "Enhancement of Shape Perception by Surface Reflectance Transformation", Hewlett-Packard Technical Report HPL-2000-38, March 2000.

[Marschner 99] Marschner, S., Westin, S., Lafortune, E., Torrance, K., Greenberg, D., "Image-Based BRDF Measurement Including Human Skin", *Rendering Techniques 99: Proceedings of the 10th Eurographics Workshop on Rendering*, June 1999, ISBN 3-211-83382-X, pp. 131-144.

[Motta 00] Motta, G., "Compression of Polynomial Texture Maps", Hewlett-Packard Laboratories Technical Report, HPL-2000-143, October 30, 2000.

[Nicodemus 77] Nicodemus, F.E., Richmond, J.C., Hsai, J.J., "Geometrical Considerations and Nomenclature for Reflectance", *U.S. Dept. of Commerce, National Bureau of Standards*, October 1977.

[Nimeroff 94] Nimeroff, J., Simoncelli, E., Dorsey, J., "Efficient Re-rendering of Naturally Illuminated Environments", *Eurographics Rendering Workshop Proceedings* 1994, pp. 359-374.

[Nishino 99] Nishino, K., Sato, Y., Katsushi, I., "Eigen-texture Method – Appearance Compression based on 3D Model", *IEEE Computer Vision and Pattern Recognition,* June 23-25 1999, Vol. 1, pp.618-624.

[Phong 75] Phong, B.-T., "Illumination for Computer Generated Images", Communications of the ACM 18, 6, June 1975, pp. 311-317.

[Ramamoorthi 01], Ramamoorthi, R. and Hanrahan, P., "An Efficient Representation for Environment Irradiance Maps", *Computer Graphics (SIGGRAPH 01 Proceedings),* August 2001.

[Rushmeier 97] Rushmeier, H., Taubin, G., Gueziec, A., "Applying Shape from Lighting Variation to Bump Map Capture", *Eurographics Rendering Workshop Proceedings* 1997, pp. 35-44, 1997.

[Schilling 96] Schilling, A., Knittel, G., Strasser, W., "Texram: A Smart Memory for Texturing", *IEEE Computer Graphics and Applications*, Vol. 16, No. 3, May 1996, pp. 32-41.

[Schilling 97] Schilling, A., "Towards Real-Time Photorealistic Rendering: Challenges and Solutions", *Proceedings of the 1997 Siggraph/Eurographics Workshop on Graphics Hardware*, Aug. 3-4, 1997, pp.7-15.

[Sillion 91] Sillion, F., Arvo, J., Westin, S., Greenberg, D., "A Global Illumination Solution for General Reflectance Distributions", *Computer Graphics (SIGGRAPH 91 Proceedings)*, July 1991, pp.187-196.

[Stalling 97] Stalling, D., Zöckler, M., Hege, H.-C., "Fast Display of Illuminated Field Lines", *IEEE Transactions on Visualization and Computer Graphics*, 3(2):118-128, 1997.

[Stam 99] Stam, J., "Diffraction Shaders", *Computer Graphics (SIGGRAPH 99 Proceedings),* August 1999, pp.101-110.

[Teo 97] Teo, P., Simoncelli, E., Heeger, D., "Efficient Linear Re-rendering for Interactive Lighting Design", Stanford Computer Science Department Technical Report STAN-CS-TN-97-60. October 1997.

[Watson 80] Watson, G.A., "Approximation Theory and Numerical Methods", A.J.Wiley & Sons, Chichester, 1980.

[Wood 00] Wood, D., Azuma, D., Aldlinger, K., Curless, B., Duchamp, T., Salesin, D., Stuetzle, W., "Surface Light Fields for 3D Photography", *Computer Graphics (Siggraph 2000 Proceedings),* July 2000, pp. 287-296

[Wong 97] Wong, T., Heng, P, Or, S, Ng, W., "Image-based Rendering with Controllable Illumination", Rendering Techniques 97: *Proceedings of the 8th Eurographics Workshop on Rendering*, June 16-18, 1997, ISBN 3-211-83001-4, pp. 13-22.

Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments

Peter-Pike Sloan
Microsoft Research
ppsloan@microsoft.com

Jan Kautz
Max-Planck-Institut für Informatik
jnkautz@mpi-sb.mpg.de

John Snyder
Microsoft Research
johnsny@microsoft.com

Abstract

We present a new, real-time method for rendering diffuse and glossy objects in low-frequency lighting environments that captures soft shadows, interreflections, and caustics. As a preprocess, a novel global transport simulator creates functions over the object's surface representing transfer of arbitrary, low-frequency incident lighting into *transferred radiance* which includes global effects like shadows and interreflections from the object onto itself. At run-time, these transfer functions are applied to actual incident lighting. Dynamic, local lighting is handled by sampling it close to the object every frame; the object can also be rigidly rotated with respect to the lighting and vice versa. Lighting and transfer functions are represented using low-order spherical harmonics. This avoids aliasing and evaluates efficiently on graphics hardware by reducing the shading integral to a dot product of 9 to 25 element vectors for diffuse receivers. Glossy objects are handled using matrices rather than vectors. We further introduce functions for radiance transfer from a dynamic lighting environment through a preprocessed object to neighboring points in space. These allow soft shadows and caustics from rigidly moving objects to be cast onto arbitrary, dynamic receivers. We demonstrate real-time global lighting effects with this approach.

Keywords: Graphics Hardware, Illumination, Monte Carlo Techniques, Rendering, Shadow Algorithms.

1. Introduction

Lighting from area sources, soft shadows, and interreflections are important effects in realistic image synthesis. Unfortunately, general methods for integrating over large-scale lighting environments [8], including Monte Carlo ray tracing [7][21][25], radiosity [6], or multi-pass rendering that sums over multiple point light sources [17][27][36], are impractical for real-time rendering.

Real-time, realistic global illumination encounters three difficulties – it must model the complex, spatially-varying BRDFs of real materials (*BRDF complexity*), it requires integration over the hemisphere of lighting directions at each point (*light integration*), and it must account for bouncing/occlusion effects, like shadows, due to intervening matter along light paths from sources to receivers (*light transport complexity*). Much research has focused on extending BRDF complexity (e.g., glossy and anisotropic reflections), solving the light integration problem by representing incident lighting as a sum of directions or points. Light integration thus tractably reduces to sampling an analytic or tabulated BRDF at a few points, but becomes intractable for large light sources. A second line of research samples radiance and preconvolves it with kernels of various sizes [5][14][19][24][34]. This solves the light integration problem but ignores light transport complexities like shadows since the convolution assumes the incident radiance is unoccluded and unscattered. Finally, clever techniques exist to simulate more complex light transport, especially shadows. Light integration becomes the problem; these techniques are impractical for very large light sources.

Our goal is to better account for light integration and light transport complexity in real-time. Our compromise is to focus on low-

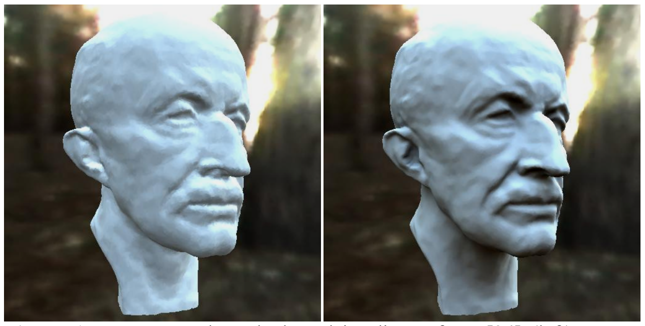


Figure 1: Precomputed, unshadowed irradiance from [34] (left) vs. our precomputed transfer (right). The right model can be rendered at 129Hz with self-shadowing and self-interreflection in **any** lighting environment.

frequency lighting environments, using a low-order spherical harmonic (SH) basis to represent such environments efficiently without aliasing. The main idea is to represent how an object scatters this light onto itself or its neighboring space.

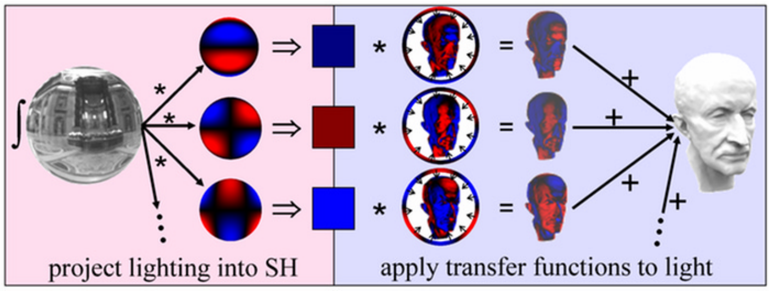
To describe our technique, assume initially we have a convex, diffuse object lit by an infinitely distant environment map. The object's shaded "response" to its environment can be viewed as a *transfer function*, mapping incoming to outgoing radiance, which in this case simply performs a cosine-weighted integral. A more complex integral captures how a concave object shadows itself, where the integrand is multiplied by an additional transport factor representing visibility along each direction.

Our approach is to precompute for a given object the expensive transport simulation required by complex transfer functions like shadowing. The resulting transfer functions are represented as a dense set of vectors or matrices over its surface. Meanwhile, incident radiance need not be precomputed. The graphics hardware can dynamically sample incident radiance at a number of points. Analytic models, such as skylight models [33] or simple geometries like circles, can also be used.

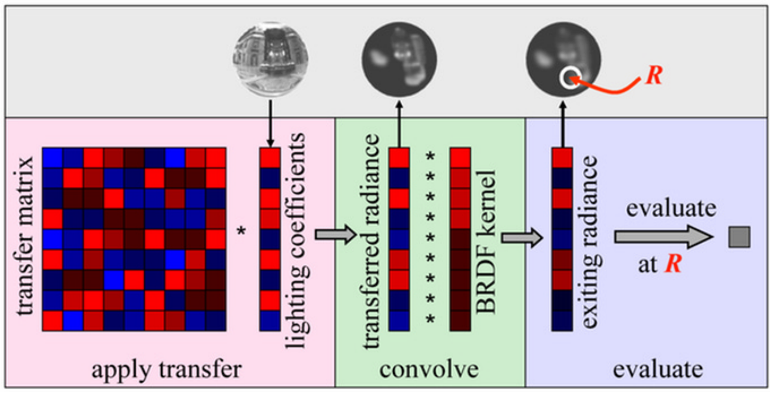
By representing both incident radiance and transfer functions in a linear basis (in our case, SH), we exploit the linearity of light transport to reduce the light integral to a simple dot product between their coefficient vectors (diffuse receivers) or a simple linear transform of the lighting coefficient vector through a small transfer matrix (glossy receivers). Low-frequency lighting environments require few coefficients (9-25), enabling graphics hardware to compute the result in a single pass (Figure 1, right). Unlike Monte-Carlo and multi-pass light integration methods, our run-time computation stays constant no matter how many or how big the light sources, and in fact relies on large-scale, smooth lighting to limit the number of SH coefficients necessary.

We represent complex transport effects like interreflections and caustics in the transfer function. Since these are simulated as a preprocess, only the transfer function's basis coefficients are affected, not the run-time computation. Our approach handles both surface and volume-based geometry. With more SH coefficients, we can even handle glossy (but not highly specular) receivers as well as diffuse, including interreflection. 25 coefficients suffice for useful glossy effects. In addition to transfer from a rigid object to itself, called *self-transfer*, we generalize the technique to *neighborhood-transfer* from a rigid object to its neighboring space, allowing cast soft shadows, glossy reflections, and caustics on dynamic receivers, see Figure 7.

Overview As a preprocess, a global illumination simulator is run over the model that captures how it shadows and scatters light onto itself. The result is recorded as a dense set of vectors (diffuse case) or matrices (glossy case) over the model. At run-time (Figure 2), incident radiance is first projected to the SH basis. The model's field of transfer vectors or matrices is then applied to the lighting's coefficient vector. If the object is **diffuse**, a **transfer vector** at each point on the object is dotted with the lighting's coefficients to produce correctly self-scattered shading. If the object is **glossy**, a **transfer matrix** is applied to the lighting coefficients to produce the coefficients of a spherical function representing self-scattered incident radiance at each point. This function is convolved with the object's BRDF and then evaluated at the view-dependent reflection direction to produce the final shading.



diffuse surface self-transfer



glossy surface self-transfer

Figure 2: Self-Transfer Run-Time Overview. Red signifies positive SH coefficients and blue, negative. For a diffuse surface (top row), the SH lighting coefficients (on the left) modulate a field of transfer vectors over the surface (middle) to produce the final result (right). A transfer vector at a particular point on the surface represents how the surface responds to incident light at that point, including global transport effects like self-shadowing and self-interreflection.. For a glossy surface (bottom row), there is a matrix at each point on the model instead of a vector. This matrix transforms the lighting coefficients into the coefficients of a spherical function representing transferred radiance. The result is convolved with the model's BRDF kernel and evaluated at the view-dependent reflection direction R to yield the result at one point on the model.

2. Related Work

Scene relighting precomputes a separate global illumination solution per light source as we do; linear combinations of the results then provide limited dynamic effects. Early work [2][11] adjusts intensities of a fixed set of sources and is not intended to fit general lighting environments. Nimeroff, et al. [33] precompute a "steerable function" basis for general skylight illumination on a fixed view. Their basis, essentially the spherical monomials, is related to the SH by a linear transformation and thus shares some of its properties (e.g., rotational invariance) but not others (e.g., orthonormality). Teo, et al. [40] generalize to non-infinite sources, using principal component analysis to reduce the basis set. Our work differs by computing a transfer field over the object's surface in 3D rather than over a fixed 2D view to allow viewpoint changes. Dobashi, et al. [10] use the SH basis and transfer vector fields over surfaces to allow viewpoint change but restrict lighting changes to the directional intensity distribution of an existing set of non-area light sources in diffuse scenes. Debevec, et al. [9] relight faces using a directional light basis. Real-time rendering requires a fixed view.

Shadow maps, containing depths from the light source's point of view, were first used by Williams [43] to simulate point light source shadows. Many extensions of the basic technique, some suitable for real-time rendering, have since been described: *percentage-closer filtering* [35], which softens shadow edges, *layered depth maps* [26] and *layered attenuation maps* [1], which more accurately simulate penumbra shape and falloff, and *deep shadow maps* [29], which generalize the technique to partially transparent and volume geometry. All these techniques assume point or at least localized light sources; shadowing from larger light sources has been handled by multi-pass rendering that sums over a light source decomposition into points or small sources [17][27][36]. Large light sources become very expensive.

Another technique [39] uses FFT convolution of occluder projections for soft shadowing with cost independent of light source size. Only shadows between pre-segmented clusters of objects are handled, making self-shadows on complex meshes difficult.

Finally, *accessibility shading* [32] is also based on precomputed global visibility, but is a scalar quantity that ignores changes in lighting direction.

Methods for nonlocal lighting on micro-geometry include the *horizon map* [4][31], which efficiently renders self-shadowing from point lights. In [20], this technique is tailored to graphics hardware and generalized to diffuse interreflections, though interreflection change due to dynamic lighting is still not real-time. By precomputing a higher-dimensional texture, *polynomial texture maps* [30] allow real-time interreflection effects as well as shadowing. A similar approach using a steerable basis for directional lighting is used in [3]. Like our approach, these methods precompute a simple representation of a transfer function, but one based on directional light sources and thus requiring costly multi-pass integration to simulate area lights. We compute self-transfer directly on each preprocessed 3D object rather than mapping it with 2D micro-geometry textures, allowing more global effects. Finally, our neighborhood transfer extends these ideas to cast shadows, caustics, and reflections.

Caching onto diffuse receivers is useful for accelerating global illumination. Ward et. al. [41] perform caching to simulate diffuse interreflection in a ray tracer. Photon maps [21] also cache but perform forward ray tracing from light sources rather than backwards from the eye, and handle specular bounces in the transport (as does our approach). We apply this caching idea to real-time rendering, but cache a transfer function parameterized by a SH lighting basis rather than scalar irradiance.

Precomputed transfer using light-field remapping [18] and dynamic ray tracing [16] has been used to achieve highly specular reflections and refractions. We apply a similar precomputed, per-object decomposition but designed instead for soft shadows and caustics on nearly diffuse objects in low-frequency lighting. Irradiance volumes [15] allow movement of diffuse receivers in precomputed lighting. Unlike our approach, lighting is static and the receiver's effect on itself and its environment is ignored.

Spherical harmonics have been used to represent incident radiance and BRDFs for offline rendering and BRDF inference [4][38][42]. Westin, et al. [42] use a matrix representation for 4D BRDF functions in terms of the SH basis identical to our transfer matrix. But rather than the BRDF, we use it to represent global and spatially varying transport effects like shadows. The SH basis has also been used to solve ambiguity problems in computer vision [12] and to represent irradiance for rendering [34].

3. Review of Spherical Harmonics

Definition Spherical harmonics define an orthonormal basis over the sphere, $\boldsymbol{S}$, analogous to the Fourier transform over the 1D circle. Using the parameterization

$$s = (x, y, z) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta),$$

the basis functions are defined as

$$Y_l^m(\theta,\varphi) = K_l^m\, e^{im\varphi} P_l^{|m|}(\cos\theta),\ l \in \boldsymbol{N},\ -l \le m \le l$$

where P_l^m are the associated Legendre polynomials and K_l^m are the normalization constants

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} .$$

The above definition forms a complex basis; a real-valued basis is given by the simple transformation

$$y_l^m = \begin{cases} \sqrt{2}\,\mathrm{Re}(Y_l^m), & m > 0 \\ \sqrt{2}\,\mathrm{Im}(Y_l^m), & m < 0 \\ Y_l^0, & m = 0 \end{cases} = \begin{cases} \sqrt{2}\,K_l^m \cos(m\varphi) P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}\,K_l^m \sin(-m\varphi) P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

Low values of l (called the *band* index) represent low-frequency basis functions over the sphere. The basis functions for band l reduce to polynomials of order l in x, y, and z. Evaluation can be done with simple recurrence formulas [13][44].

Projection and Reconstruction Because the SH basis is orthonormal, a scalar function f defined over $\boldsymbol{S}$ can be *projected* into its coefficients via the integral

$$f_l^m = \int f(s)\, y_l^m(s)\, ds \tag{1}$$

These coefficients provide the n-th order reconstruction function

$$\tilde{f}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m\, y_l^m(s) \tag{2}$$

which approximates f increasingly well as the number of bands n increases. Low-frequency signals can be accurately represented with only a few SH bands. Higher frequency signals are bandlimited (i.e., smoothed without aliasing) with a low-order projection.

Projection to n-th order involves n^2 coefficients. It is often convenient to rewrite (2) in terms of a singly-indexed vector of projection coefficients and basis functions, via

$$\tilde{f}(s) = \sum_{i=1}^{n^2} f_i\, y_i(s) \tag{3}$$

where $i=l(l+1)+m+1$. This formulation makes it obvious that evaluation at s of the reconstruction function represents a simple dot product of the n^2-component coefficient vector f_i with the vector of evaluated basis functions $y_i(s)$.

Basic Properties A critical property of SH projection is its rotational invariance; that is, given $g(s) = f(Q(s))$ where Q is an arbitrary rotation over $\boldsymbol{S}$ then

$$\tilde{g}(s) = \tilde{f}(Q(s)) \tag{4}$$

This is analogous to the shift-invariance property of the 1D Fourier transform. Practically, this property means that SH projection causes no aliasing artifacts when samples from f are collected at a rotated set of sample points.

Orthonormality of the SH basis provides the useful property that given any two functions a and b over $\boldsymbol{S}$, their projections satisfy

$$\int \tilde{a}(s)\, \tilde{b}(s)\, ds = \sum_{i=1}^{n^2} a_i\, b_i . \tag{5}$$

In other words, integration of the product of bandlimited functions reduces to a dot product of their projection coefficients.

Convolution We denote convolution of a circularly symmetric kernel function $h(z)$ with a function f as $h * f$. Note that h must be circularly symmetric (and hence can be defined as a simple function of z rather than s) in order for the result to be defined on $\boldsymbol{S}$ rather than the higher-dimensional rotation group $\boldsymbol{SO}(3)$. Projection of the convolution satisfies

$$(h * f)_l^m = \sqrt{\frac{4\pi}{2l+1}}\, h_l^0\, f_l^m = \alpha_l^0\, h_l^0\, f_l^m . \tag{6}$$

In other words, the coefficients of the projected convolution are simply scaled products of the separately projected functions. Note that because h is circularly symmetric about z, its projection coefficients are nonzero only for m=0. The convolution property provides a fast way to convolve an environment map with a hemispherical cosine kernel, defined as $h(z) = \max(z, 0)$, to get an irradiance map [34], for which the h_l^0 are given by an analytic formula. The convolution property can also be used to produce prefiltered environment maps with narrower kernels.

Product Projection Projection of the product of a pair of spherical functions $c(s) = a(s)b(s)$ where a is known and b unknown can be viewed as a linear transformation of the projection coefficients b_j via a matrix $\hat{a}$:

$$\begin{aligned} c_i &= \int a(s)\left(b_j\, y_j(s)\right) y_i(s)\, ds \\ &= \left(\int a(s)\, y_i(s) y_j(s)\, ds\right) b_j = \left(a_k \int y_i(s)\, y_j(s)\, y_k(s)\, ds\right) b_j = \hat{a}_{ij}\, b_j \end{aligned} \tag{7}$$

where summation is implied over the duplicated j and k indices. Note that $\hat{a}$ is a symmetric matrix. The components of $\hat{a}$ can be computed by integrating the triple product of basis functions using recurrences derived from the well-known Clebsch-Gordan series [13][44]. It can also be computed using numerical integration without SH-projecting the function a beforehand. Note that the product's order n projection involves coefficients of the two factor functions up to order $2n$-1.

Rotation A reconstruction function rotated by Q, $\tilde{f}\left(Q(s)\right)$, can be projected into SH using a linear transformation of f's projection coefficients, f_i. Because of the rotation invariance property, this linear transformation treats the coefficients in each band independently. The most efficient implementation is achieved via a *zyz* Euler angle decomposition of the rotation Q, using a fairly complicated recurrence formula [13][44]. Because we deal only with low-order functions, we have implemented their explicit rotation formulas using symbolic integration.

4. Radiance Self-Transfer

Radiance self-transfer encapsulates how an object $\boldsymbol{O}$ shadows and scatters light onto itself. To represent it, we first parameterize incident lighting at points $p \in \boldsymbol{O}$, denoted $L_p(s)$, using the SH basis. Incident lighting is therefore represented as a vector of n^2 coefficients $(L_p)_i$. We sample the lighting **dynamically** and **sparsely** near the surface, perhaps at only a single point. The assumption is that lighting variation over $\boldsymbol{O}$ not due to its own presence is small (see Section 6.1). We also **precompute** and store **densely** over $\boldsymbol{O}$ transfer vectors or matrices.

A *transfer vector* $(M_p)_i$ is useful for diffuse surfaces and represents a linear transformation on the lighting vector producing scalar exit radiance, denoted L'_p, via the inner product

$$L'_p = \sum_{i=1}^{n^2} (M_p)_i\, (L_p)_i . \tag{8}$$

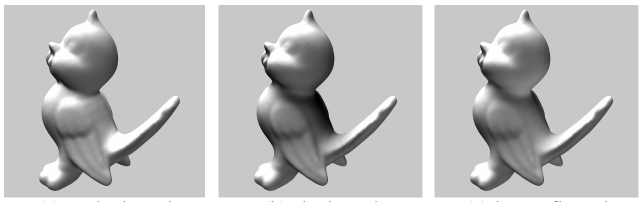

(a) unshadowed (b) shadowed (c) interreflected

Figure 3: Diffuse Surface Self-transfer.

In other words, each component of $(M_p)_i$ represents the linear influence that a lighting basis function $(L_p)_i$ has on shading at *p*.

A *transfer matrix* $(\mathcal{M}_p)_{ij}$ is useful for glossy surfaces and represents a linear transformation on the lighting vector which produces projection coefficients for an entire spherical function of transferred radiance $L'_p(s)$ rather than a scalar; i.e.,

$$(L'_p)_i = \sum_{j=1}^{n^2} (\mathcal{M}_p)_{ij} (L_p)_j \quad . \tag{9}$$

The difference between incident and transferred radiance is that $L'_p(s)$ includes shadowing/scattering effects due to the presence of $\boldsymbol{O}$ while $L_p(s)$ represents incident lighting assuming $\boldsymbol{O}$ was removed from the scene. Components of $(\mathcal{M}_p)_{ij}$ represent the linear influence of the *j*-th lighting coefficient of incident radiance $(L_p)_j$ to the *i*-th coefficient of transferred radiance $(L'_p)_i$. The next sections derive transfer vectors for diffuse surfaces and transfer matrixes for glossy surfaces due to self-scattering on $\boldsymbol{O}$.

4.1 Diffuse Transfer [transfer vector for known normal]

First assume $\boldsymbol{O}$ is diffuse. The simplest transfer function at a point $p \in \boldsymbol{O}$ represents *unshadowed diffuse transfer*, defined as the scalar function

$$T_{DU}(L_p) = \left(\rho_p/\pi\right) \int L_p(s) H_{Np}(s)\, ds$$

producing exit radiance which is invariant with view angle for diffuse surfaces. Here, ρ_p is the object's albedo at *p*, L_p is the incident radiance at *p* assuming $\boldsymbol{O}$ was removed from the scene, N_p is the object's normal at *p*, and $H_{Np}(s) = \max(N_p \bullet s, 0)$ is the cosine-weighted, hemispherical kernel about N_p. By SH-projecting L_p and H_{Np} separately, equation (5) reduces T_{DU} to an inner product of their coefficient vectors. We call the resulting factors the *light function*, L_p, and *transfer function,* M_p. In this first simple case, $M_p^{DU}(s) = H_{Np}(s)$.

Because N_p is known, the SH-projection of the transfer function $(M_p^{DU})_i$ can be precomputed, resulting in a *transfer vector*. In fact, storing is unnecessary because a simple analytic formula yields it given N_p. Because M_p^{DU} is inherently a low-pass filter, second-order projection (9 coefficients) provides good accuracy in an arbitrary (even non-smooth) lighting environment [34].

To include shadows, we define *shadowed diffuse transfer* as

$$T_{DS}(L_p) = \left(\rho_p/\pi\right) \int L_p(s) H_{Np}(s) V_p(s)\, ds$$

where the additional visibility function, $V_p(s) \rightarrow \{0,1\}$, equals 1 when a ray from *p* in the direction *s* fails to intersect $\boldsymbol{O}$ again (i.e., is unshadowed). As with unshadowed transfer, we decompose this integral into two functions, using an SH-projection of L_p and the transfer function

$$M_p^{DS}(s) = H_{Np}(s) V_p(s) \quad . \tag{10}$$

Separately SH-projecting L_p and M_p again reduces the integral in T_{DS} to an inner product of coefficient vectors.

Transfer is now nontrivial; we precompute it using a transport simulator (Section 5), storing the resulting transfer vector $(M_p)_i$ at many points *p* over $\boldsymbol{O}$. Unlike the previous case, second-order projection of M_p^{DS} may be inaccurate even for smooth lighting environments since V_p can create higher-frequency lighting locally, e.g., by self-shadowing "pinholes". 4-th or 5-th order projection provides good results on typical meshes in smooth lighting environments.

Finally, to capture diffuse interreflections as well as shadows, we define *interreflected diffuse transfer* as

$$T_{DI}(L_p) = T_{DS}(L_p) + \left(\rho_p/\pi\right) \int \overline{L}_p(s)\, H_{Np}(s) \left(1 - V_p(s)\right) ds$$

where $\overline{L}_p(s)$ is the radiance from $\boldsymbol{O}$ itself in the direction *s*. The difficulty is that unless the incident radiance emanates from an infinitely-distant source, we don't actually know $\overline{L}_p(s)$ given the incident radiance only at *p* because $\overline{L}_p$ depends on the exit radiance of points arbitrarily far from *p* and local lighting varies over $\boldsymbol{O}$. If lighting variation is small over $\boldsymbol{O}$ then $\overline{L}_p$ is well-approximated as if $\boldsymbol{O}$ were everywhere illuminated by L_p. T_{DI} thus depends linearly on L_p and can be factored as in the previous two cases into a product of two projected functions: one light-dependent and the other geometry-dependent.

Though precomputed interreflections must make the assumption of spatially invariant incident lighting over $\boldsymbol{O}$, simpler shadowed transfer need not. The difference is that shadowed transfer depends only on incident lighting at *p*, while interreflected transfer depends on many points $q \neq p$ over $\boldsymbol{O}$ at which $L_q \neq L_p$. Thus, as long as the incident radiance field is sampled finely enough (Section 6.1), local lighting variation can be captured and shadowed transfer will be correct.

The presence of $\overline{L}$ makes it hard to explicitly denote the transfer function for interreflections, $M_p^{DI}(s)$. We will see how to compute its projection coefficients numerically in Section 5.

4.2 Glossy Transfer [transfer matrix for unknown direction]

Self-transfer for glossy objects can be defined similarly, but generalizes the kernel function to depend on a (view-dependent) reflection direction *R* rather than a (fixed) normal *N*. Analogous to the *H* kernel from before, we model glossy reflection as the kernel *G*(*s*,*R*,*r*) where a scalar *r* defines the "glossiness" or broadness of the specular response. We believe it is possible to handle arbitrary BRDFs as well using their SH projection coefficients [38] but this remains for future work.

We can then define the analogous three glossy transfer functions for the unshadowed, shadowed, and interreflected cases as

$$T_{GU}(L_p, R, r) = \int L_p(s) G(s, R, r)\, ds$$

$$T_{GS}(L_p, R, r) = \int L_p(s) G(s, R, r) V_p(s)\, ds$$

$$T_{GI}(L_p, R, r) = T_{GS}(L_p) + \int \overline{L}_p(s)\, G(s, R, r) \left(1 - V_p(s)\right) ds$$

which output scalar radiance in direction *R* as a function of L_p and *R*, quantities both unknown at precomputation time. Since transfer is no longer solely a function of *s*, it can't be reduced to a simple vector of SH coefficients

Instead of parameterizing scalar transfer by *R* and *r*, a more useful decomposition is to transfer the incident radiance $L_p(s)$ into a whole sphere of transferred radiance, denoted $L'_p(s)$. Assuming the glossy kernel *G* is circularly symmetric about *R* (i.e., a simple Phong-like model) $L'_p(s)$ can then be convolved with $G_r^*(z) = G\left(s, (0,0,1), r\right)$ and evaluated at *R* to produce the final

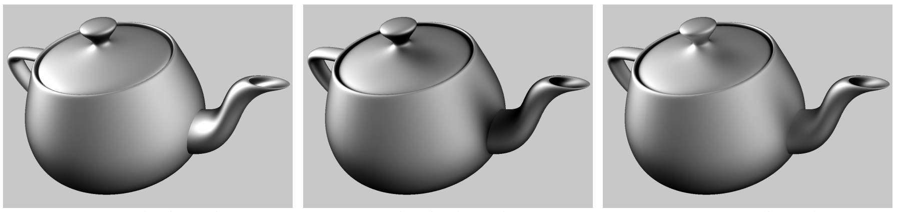

(a) unshadowed (b) shadowed (c) interreflected

Figure 4: Glossy Surface Self-transfer.

result (see bottom of Figure 2, and further details in Section 6).

Transfer to L'_p can now be represented as a matrix rather than a vector. For example, glossy shadowed transfer is

$$\mathcal{M}_p^{GS}(L_p,s) = L_p(s)V_p(s) \qquad (11)$$

a linear operator on L_p whose SH-projection can be represented as the symmetric matrix $\hat{V}_p$ via equation (7). Even with smooth lighting, more SH bands must be used for L'_p as $\boldsymbol{O}$'s glossiness increases; non-square matrices (e.g., 25×9) mapping low-frequency lighting to higher-frequency transferred radiance are useful under these conditions. For shadowed glossy transfer (but not interreflected), an alternative still uses a vector rather than a matrix to represent $\mathcal{M}_p^{GS}$ by computing the product of V_p with L_p on-the-fly using the tabulated triple product of basis functions in equation (7). We have not yet implemented this alternative.

4.3 Limitations and Discussion

An important limitation of precomputed transfer is that material properties of $\boldsymbol{O}$ influencing interreflections in T_{DI} and T_{GI} (like albedo or glossiness) must be "baked in" to the preprocessed transfer and can't be changed at run-time. On the other hand, the simpler shadowed transfer without interreflection does allow run-time change and/or spatial variation over $\boldsymbol{O}$ of the material properties. Error arises if blockers or light sources intrude into $\boldsymbol{O}$'s convex hull. $\boldsymbol{O}$ can only move rigidly, not deform or move one component relative to the whole. Recall also the assumption of low lighting variation over $\boldsymbol{O}$ required for correct interreflections.

Finally, note that diffuse transfer as defined produces radiance *after leaving the surface*, since it has already been convolved with the cosine-weighted normal hemisphere, while glossy transfer produces radiance *incident on the surface* and must be convolved with the local BRDF to produce the final exit radiance. It's also possible to bake in a fixed BRDF for glossy $\boldsymbol{O}$, making the convolution with G unnecessary at run-time but limiting flexibility.

5. Precomputing Radiance Self-Transfer

As a preprocess, we perform a global illumination simulation over an object $\boldsymbol{O}$ using the SH basis over the infinite sphere as emitters. Our light gathering solution technique is a straightforward adaptation of existing approaches [7][25] and could be accelerated in many ways; its novelty lies in how it parameterizes the lighting and collects the resulting integrated transfers. Note that all integrated transfer coefficients are *signed* quantities.

The simulation is parameterized by an n-th order SH projection of the unknown sphere of incident light L; i.e., by n^2 unknown coefficients L_i. Though the simulation results can be computed independently for each L_i using the SH basis function $y_i(s)$ as an emitter, it is more efficient to compute them all at once. The infinitely-distant sphere L will be replaced at run-time by the actual incident radiance field around $\boldsymbol{O}$, L_p.

An initial pass simulates direct shadows from paths leaving L and reaching sample points $p \in \boldsymbol{O}$. In subsequent passes, interreflections are added, representing paths from L that bounce a number of times off $\boldsymbol{O}$ before arriving at p (Lp, LDp, $LDDp$, etc.). In each pass, energy is gathered to every sample point p. Large emitters (i.e., low-frequency SH basis) make a gather more efficient then a shooting-style update [6]. Note that this idea of caching onto diffuse (or nearly diffuse) receivers is not new [21][41].

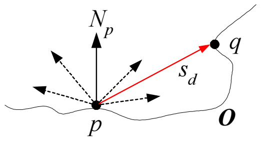
To capture the sphere of directions at sample points $p \in \boldsymbol{O}$, we generate a large (10k-30k), quasi-random set of directions $\{s_d\}$, $s_d \in \boldsymbol{S}$. We also precompute evaluations for all the SH basis functions at each s_d. The s_d are organized in hierarchical bins formed by refining an initial icosahedron with 1→2 bisection into equal-area spherical triangles (1→4 subdivision does not lead to equal area triangles on the sphere as it does in the plane). We use 6 to 8 subdivision levels, creating 512 to 2048 bins. Every bin at each level of the hierarchy contains a list of the s_d within it.

In the first pass, for each $p \in \boldsymbol{O}$, we cast shadow rays in the hemisphere about p's normal N_p, using the hierarchy to cull directions outside the hemisphere. We tag each direction s_d with an occlusion bit, $1 - V_p(s_d)$, indicating whether s_d is in the hemisphere and intersects $\boldsymbol{O}$ again (i.e., is self-shadowed by $\boldsymbol{O}$). An occlusion bit is also associated with the hierarchical bins, indicating whether any s_d within it is occluded. Self-occluded directions and bins are tagged so that we can perform further interreflection passes on them; completely unoccluded bins/samples receive only direct light from the environment.

For diffuse surfaces, at each point $p \in \boldsymbol{O}$ we further compute the transfer vector by SH-projecting M_p from (10). For glossy surfaces, we compute the transfer matrix by SH-projecting $\mathcal{M}_p$ from (11). In either case, the result represents the radiance collected at p, parameterized by L. SH-projection to compute the transfers is performed by numerical integration over the direction samples s_d, summing into an accumulated transfer using the following rules:

diffuse:	$(M_p)_i^0 \mathrel{+}= \left(\rho_p/\pi\right) V_p(s_d)\, H_N(s_d)\, y_i(s_d)$
glossy:	$(\mathcal{M}_p)_{ij}^0 \mathrel{+}= V_p(s_d)\, y_j(s_d)\, y_i(s_d)$
Transfer integration over s_d [<u>shadow</u> pass, iteration 0]	

The superscript 0 refers to the iteration number. The vector M_p or matrix $\mathcal{M}_p$ at each point p is initialized to 0 before the shadow pass, which then sums over all s_d at every p. The rules are derived using equation (1) for diffuse transfer integration, and equation (7) for glossy transfer integration.



Later interreflection passes traverse the bins having the occlusion bit set during the shadow pass. Instead of shadow rays, they shoot rays that return transfer from exiting illumination on $\boldsymbol{O}$. If the ray (p,s_d) intersects another point $q \in \boldsymbol{O}$ (where q is closest to p), we sample the radiance exiting from q in the direction $-s_d$. The following update rules are used, where the superscript b is the bounce pass iteration:

diffuse:	$(M_p)_i^b \mathrel{+}= \left(\rho_p/\pi\right)\left(1 - V_p(s_d)\right)(M_q)_i^{b-1}\, H_N(s_d)$
glossy:	$(\mathcal{M}_p)_{ij}^b \mathrel{+}= \left(1 - V_p(s_d)\right) \left(\sum_k \alpha_k\, (G^*_{r_q})_k\, (\mathcal{M}_q)_{kj}^{b-1}\, y_k\left(\text{reflect}(-s_d, N_q)\right)\right) y_i(s_d)$
Transfer integration over s_d [<u>interreflection</u> passes, iteration b]	

As in the shadow pass, we begin by initializing transfer vectors or matrices to 0 before accumulating transfer over directions s_d. The diffuse rules are derived from the definition of T_{DI} and equation (1); glossy rules from the definition of T_{GI} and equations (6) and (7). The middle factor in the glossy transfer definition represents radiance emanating from q back to p from the previous bounce pass, b-1. Since $\mathcal{M}_q$ stores incident radiance, it must be convolved with $\boldsymbol{O}$'s BRDF at q to obtain exiting radiance in the $-s_d$ direction, yielding a summation over k. Recall that α_k is the k-th convolution coefficient, expressed in singly-indexed notation. The "reflect" operator simply reflects its first vector argument with respect to its second. We observe that equation (7) implies $(\mathcal{M}_p)_{ij}$ is a symmetric matrix for shadowed glossy transfer since it is formed by the product of two spherical functions; this is untrue for interreflected glossy transfer.

Interreflection passes are repeated until the total energy of a given pass falls below a threshold. For typical materials, it diminishes quite rapidly. The sum of transfers from all bounce passes then accounts for interreflections. Our implementation simulates diffuse and glossy transfer at the same time.

A simple enhancement to this simulation allows mirror-like surfaces within $\boldsymbol{O}$. We do not record transfers on such surfaces. Instead, a ray striking a mirrored surface is always reflected and then propagated until a non-mirrored surface is reached. Thus our paths at successive iterations can be represented as $(L[S]^*p$, $L[S]^*D[S]^*p$, $L[S]^*D[S]^*D[S]^*p$, etc.), where D is a diffuse or glossy bounce and S is a specular one. This captures caustics onto diffuse or glossy receivers that respond dynamically to lighting change (Figure 9).

6. Run-time Rendering of Radiance Transfer

We now have a model $\boldsymbol{O}$ capturing radiance transfer at many points p over its surface, represented as vectors or matrices. Rendering $\boldsymbol{O}$ requires the following steps at run-time:

1. compute incident lighting $\{L_{Pi}\}$ at one or more sample points P_i near $\boldsymbol{O}$ in terms of the SH basis,
2. rotate these L_{Pi} to $\boldsymbol{O}$'s coordinate frame and blend them (see below) to produce a field of incident lighting L_p over $\boldsymbol{O}$, and
3. perform a linear transformation on $(L_p)_i$ at each point p on $\boldsymbol{O}$ to obtain exit radiance. This requires a dot product with $(M_p)_i$ for diffuse surfaces (equation (8)), or a matrix-vector multiplication with $(\mathcal{M}_p)_{ij}$ for glossy surfaces (equation (9)).
4. Glossy surfaces need a final step in which the radiance vector resulting from step 3 is convolved with $\boldsymbol{O}$'s BRDF at p, and then evaluated at the view-dependent reflection direction R.

Step 1 can load a precomputed environment map, evaluate analytic lighting models in software, or sample radiance using graphics hardware. Rotation for Step 2 is outlined in Section 3, and is done once per object, not for each p. It is necessary because transfer is stored using a common coordinate system for $\boldsymbol{O}$. If $\boldsymbol{O}$ is rigidly moving, it is more efficient to rotate the few radiance samples in L_{Pi} to align with $\boldsymbol{O}$ than it is to rotate $\boldsymbol{O}$'s many transfer functions. We currently perform this rotation in software.

For diffuse surfaces, a simple implementation of step 3 is to store the transfer vector per vertex and perform the dot product in a vertex shader. The transfer vectors can also be stored in texture maps rather than per-vertex and evaluated using a pixel shader. Since the coefficients are signed quantities not always in the [-1,1] range, DirectX 8.1 pixel shaders (V1.4) or their OpenGL counterpart (extension by ATI) must be used, since they provide a larger range of [-8,8]. Our pixel shader needs 8 instructions to perform the dot-product and stores L_P's coefficients in constant registers.

For colored environments or simulation of color bleeding on $\boldsymbol{O}$, three passes are required, each performing a separate dot-product for the r, g, and b channels. Otherwise a single pass suffices.

For glossy self-transfer, we perform the matrix transform from equation (9) in software because the transfer matrix is too big to be manipulated in either current vertex or pixel shaders. The result is $(L'_p)_i$ the SH coefficients of transferred radiance at points p over $\boldsymbol{O}$. Then in a pixel shader, we perform a convolution with a simple cosine-power (Phong lobe) kernel for G^* and evaluate the result in the reflection direction R. The result can be written

$$\sum_{i=1}^{n^2} \alpha_i \, G_i^* \left(\sum_{j=1}^{n^2} \left(\mathcal{M}_p\right)_{ij} \left(L_p\right)_j \right) y_i(R) \qquad (12)$$

We evaluate SH-projections up to n=5 on graphics hardware.

6.1 Spatial Sampling of the Incident Radiance Field

A simple and useful approach for dynamically sampling incident radiance is to sample it at $\boldsymbol{O}$'s center point. To handle local lighting variation over $\boldsymbol{O}$, a more accurate technique samples incident lighting at multiple points (Figure 5). A good set of sample points can be obtained using the ICP (iterated closest

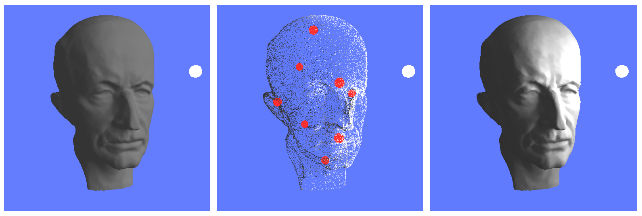

(a) single sample (b) ICP points (c) multiple samples

Figure 5: ICP can be used to precompute good locations for sampling the incident radiance field over an object. Note the improved locality of lighting in (c) compared to (a) when the lighting is sampled at the 8 points in (b) rather than at the object center.

point) algorithm [28] as a preprocess, given a desired number of points as input. This produces a representative set of points P_i near $\boldsymbol{O}$ and distributed uniformly over it where incident lighting can be sampled at run-time. We can also precompute coefficients at each p over $\boldsymbol{O}$ that blend contribution from each of the resulting sampled radiance spheres L_{Pi} to produce an incident radiance field over $\boldsymbol{O}$, denoted previously by L_p.

6.2 Sampling SH Radiance on Graphics Hardware

Graphics hardware is useful to capture the radiance samples $\{L_{Pi}\}$ in a dynamic scene. To do this, 6 images are rendered from each P_i corresponding to the 6 faces of the cube map spherical parameterization. $\boldsymbol{O}$ itself should be removed from these renderings. Cube map images can then be projected to their SH coefficients using the integral in equation (1), as was done in [4].

For efficiency, we precompute textures for the basis functions weighted by differential solid angle, $B_l^m(s) = y_l^m(s)\, ds(s)$, each evaluated over the cube map parameterization for s. The resulting integral then becomes a simple dot product of the captured samples of $L_P(s)$ with the textures $B_l^m(s)$.

Ideally, this computation would be performed on the graphics hardware. Precision issues and inability to do inner products in hardware force us to read back the sampled radiance images and project them in software. In this case, it is important to reduce the resolution of read-back images as much as possible.

Low-order SH projection can be computed with very low-resolution cube maps, assuming they have been properly bandlimited. For example, spherical signals already bandlimited to 6-th order can be projected using six 4×4 images with about 0.3% average-case squared error and about 1% worst-case squared error, where error is normalized by assuming unit-power signals (i.e., signals whose integrated square over the sphere is 1).[1] For 6×8×8 maps, this error reduces to 0.003% mean and 0.02% worst-case. Unfortunately, typical signals aren't spherically bandlimited. Another analysis shows that, assuming continuous bilinear reconstruction over the sampled 2D images, projection to 6-th order using 6×8×8 images yields 0.7% and 2% average and worst-case squared error, while 6×16×16 yields 0.2% and 0.5% squared error, and 6×32×32 yields 0.05% and 0.1% squared error.

We extract 6×16×16 images from the hardware. As is always true in point-sampled rendering, aliasing of the 2D images is still a problem because the above analysis uses bilinear reconstruction from point samples as the reference. To reduce aliasing, we supersample the cube map images by a factor of 2 in each dimension, and do a box-filtered decimation in hardware before reading back and projecting. The basis function textures are also supersampled and decimated in the same way as a preprocess. A radiance sample, including read-back and SH projection, takes about 1.16ms on a PIII-933 PC with an ATI Radeon 8500.

[1] More precisely, average-case error is the integrated squared difference between the reference and reconstruction signals, averaged over all unit-power signals. Worst-case error is the same integrated error, but for the worst-case unit-power signal.

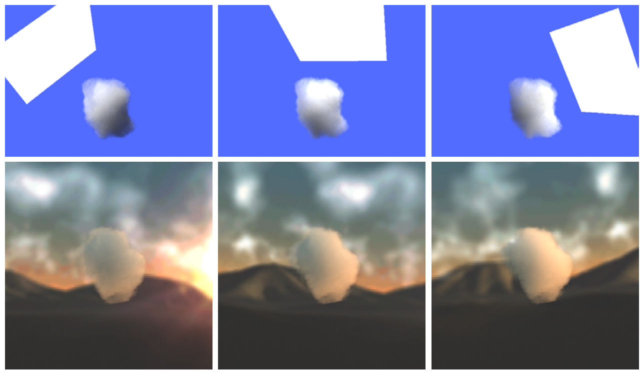


Figure 6: Volumetric self-transfer captures how this cloud model shadows itself. Lighting can be changed in real-time (first row). The same model can also be placed in other environments (second row).

7. Self-Transfer for Volumetric Models

Self-transfer on volumetric data uses the same framework as surfaces. The resulting precomputed model allows run-time changes to the lighting, with correct shadowing and interreflections in any low-frequency lighting environment (Figure 6).

Our simple simulator currently works only for diffuse volumes. As with surface transfer, a preprocessing step simulates lighting on the volume using the SH basis functions as emitters. For shadowed transfer without interreflection (i.e., direct shadowing), we gather energy from the emitter to every voxel p of the volume, attenuated by its path through the volume. The required numerical integration over directions s_d can be expressed as

$$(M_p)_i^0 \mathrel{+}= A(p \to p{+}Ds_d)\; y_i(s_d)$$

where $A(p \to q)$ is the volume's integrated attenuation along the path from p to q, and D is the distance until the ray (p,s_d) exits the volume. To include interreflections, we traverse every voxel p and forward-scatter its transfer along random directions s_d. The transfer is deposited to all voxels q along s_d until exiting the volume, using the rule

$$(M_q)_i^b \mathrel{+}= A(p \to q)\,(M_p)_i^{b-1}$$

More passes over the volume produce further indirect bounces.

Rendering is performed in the traditional way: by drawing slices through the 3D volume in back to front order using alpha blending to account for transparency. Each slice is a 2D image containing samples of the transfer vector. A pixel shader computes the dot-product between the lighting's coefficients and the transfer vector's required to shade each slice.

8. Radiance Neighborhood-Transfer

Neighborhood-transfer precomputes an object ***O***'s influence on its neighboring environment with respect to parameterized, low-frequency lighting. Transport simulation is identical to that for self-transfer in Section 5, but takes place with respect to points in a 3D space surrounding ***O***, not on it. At run-time, an arbitrary receiver ***R*** can be placed in this volume to capture shadows, reflections, and caustics cast by ***O*** onto ***R*** without knowing ***R*** in advance. For example, a moving vehicle ***O*** can cast shadows over a terrain ***R*** (Figure 7). Cast shadows and lighting also respond to lighting change; for example, moving the lights move soft shadows on ***R***. This generalizes irradiance volumes [15] by accounting for glossy transfer and allowing dynamic lighting.

Because ***R*** is unknown during the precomputation step, ***O***'s neighborhood volume must store a transfer matrix rather than a vector. This is true even for diffuse receivers, because we do not know in advance what ***R***'s normal will be. Our current implementation precomputes the transfer matrix $\mathcal{M}_p$ at each point within a simple 3D grid surrounding ***O***. At run-time, we perform the matrix transform from equation (9) in software at each point

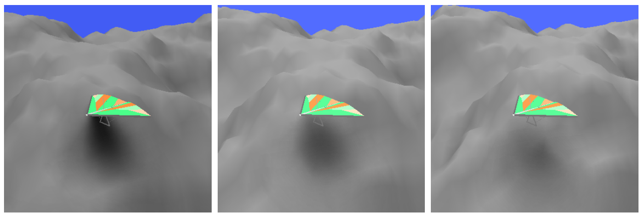


Figure 7: Neighborhood transfer captures how this hang glider blocks light to a volume of points below it. This allows cast soft shadow onto a bumpy terrain as the glider moves.

in the volume and upload the result to the graphics hardware. The result is a volume texture containing coefficients of transferred radiance $(L'_p)_i$ which is applied to ***R***.

Then in a pixel shader this transferred radiance is used to light the receiver. A diffuse receiver convolves the radiance with the cosine weighted hemisphere H^* using equation (6) and then evaluates the resulting SH projection at ***R***'s normal vector. Glossy receivers perform equation (12).

Receivers having precomputed self-transfer raise the difficulty that ***O*** and ***R*** do not share a common coordinate system. Thus, one of the two object's dense set of transfer samples must be dynamically rotated to align with the other's. The SH-rotation operation required is currently impractical for hardware evaluation. Improving hardware should soon ease this difficulty.

Compared to self-transfer, neighborhood-transfer incurs some additional approximation errors. Cast shadow or light from multiple neighborhood-transfer objects onto the same receiver is hard to combine. Local lighting variation not due to ***O*** or ***R***'s presence is also a problem; lighting must be fairly constant across ***O***'s entire neighborhood to provide accurate results. In particular, errors such as missing shadows will result when objects besides ***O*** and ***R*** intrude into ***O***'s neighborhood. ***O***'s neighborhood must also be large enough to encompass any cast shadow or light it may cast on ***R***. Nevertheless, neighborhood transfer captures effects impossible to obtain in real-time with previous methods.

9. Results

Full statistics are summarized in Table 1. We achieve real-time performance for all models except the transfer matrix ones (teapot, buddha, glider). For these models, multiplication with 25x25 or 9x25 transfer matrices over the surface in software forms the bottleneck. Real-time results can be achieved even for these models after first fixing the light (allowing the view to be moved) or the view (allowing the light to be changed) and represent the second and third "render rate" entries in the table after slashes. The reason is that fixing either the view (which then fixes the

model	transfer type	transfer shape	transfer sampling	preproc. time	render rate
head (fig 1)	DS	25-M	50k ver. mesh	1.1h	129
bird (fig 4)	DS	25-M	50k ver. mesh	1.2h	125
ring (fig 9)	DS	25-M	256x256 grid	8m	94
buddha_d (fig 11)	DS	25-M	50k ver. mesh	2.5h	125
buddha_g (fig 11)	GS	25x25-$\mathcal{M}$	50k ver. mesh	2.5h	3.6/16/125
tyra_d (fig 11)	DS	25-M	100k ver. mesh	2.4h	83
tyra_g (fig 11)	GS	25x25-$\mathcal{M}$	100k ver. mesh	2.4h	2.2/9.4/83
teapot (fig 5)	GS	25x25-$\mathcal{M}$	150k ver. mesh	4.4h	1.7/7.7/49
cloud (fig 6)	DV	25-M	32x32x32 vol.	15m	40
glider (fig 7)	N	9x25-$\mathcal{M}$	64x64x8 vol.	3h	4/120/4

Table 1: Results. Transfer types are DS=diffuse surface self-transfer, GS=glossy surface self-transfer, DV=diffuse volume self-transfer, and N=neighborhood transfer. Timings are on a 2.2GHz Pentium 4 with ATI Radeon 8500 graphics card. Render rates are in Hz.

reflection vector R) or the light L_p reduces the computation in (12) to a simple dot product, which can then be done in hardware. Fixed light rendering is slower than fixed view because the fixed light mode requires evaluation of the SH basis at a changing view vector, followed by a dot product, while fixed view requires only the dot product and is identical in performance to diffuse transfer.

Rendering quality can be judged from images in this paper (Figures 1 and 3-12) all of which were computed with the PC graphics hardware. Self-shadowing and interreflection effects are convincing and robust. No depth tolerances are required to prevent self-shadowing artifacts as they are with the standard shadow buffer technique [43]. Even when the lighting contains very high frequencies (Figure 9, top row of Figure 10, and Figure 12), pleasing images are produced without temporal artifacts but with some blurring of self-shadows; the result looks, and indeed is, identical to blurring the incident lighting.

Figure 10 compares shadowing results across different SH orders. Small light sources (top row) require more bands; larger ones are approximated very well with fewer bands. Using up to the quartic band (fifth order with 25 coefficients) provides good results and is a good match to today's graphics hardware. Note that quality is not dictated solely by how much energy is ignored in SH-projecting the lighting – diffuse and glossy objects are effective low-pass filters of incident radiance. With self-transfer effects though, the extent of this low-pass filtering depends on the object's geometry, varies spatially, and typically requires more than third order (quadratics), unlike unshadowed diffuse transfer [34].

Because of its rotational invariance (equation (4)), we consider the SH basis especially useful for our low-frequency lighting application compared to alternatives like spherical wavelets [37]. When dynamically sampling incident radiance, this property eliminates aliasing which would otherwise produce temporal artifacts, like shading "wobble", if projected to other bases with the same number of coefficients. Ringing or Gibbs phenomenon (oscillatory undershoot and overshoot in the reconstruction function) can be a problem when the lighting environment has significant energy near its highest represented band [10][42]. We notice ringing artifacts only on very simple models such as the ones in Figures 9 and 10; artifacts are masked on complex meshes. Of course, reducing lighting frequency by attenuating higher frequency bands, called "windowing", also reduces ringing (see Figure 10, columns f and g).

10. Conclusions and Future Work

Much important shading variation over a complex object such as a human face is due to itself. Precomputed radiance self-transfer is a general method for capturing the occlusion and scattering effects an object has on itself with respect to any low-frequency lighting environment. When the actual incident lighting is substituted at run-time, the resulting model provides global illumination effects like soft shadows, interreflections, and caustics in real-time. Using graphics hardware, incident lighting can be sampled every frame and at multiple points near the object allowing dynamic, local lighting. Neighborhood-transfer generalizes the concept by recording transfer over 3D space, allowing cast soft shadows and caustics onto arbitrary receivers.

In future work, we want to apply precomputed transfer to more sophisticated transport models, especially subsurface scattering [22]. We believe the smoothness of exiting radiance produced by this model makes it particularly suitable for SH-parameterized transfer. It would also be valuable to combine existing shadowing techniques with ours, by decomposing the scene's lighting into high and low frequency terms. Compression of transfer fields is an important but unaddressed problem. Extension to deformable objects like human characters could be achieved by parameterizing the precomputed self-transfer in the same way as the deformation, assuming the number of independent degrees of freedom remains manageable. Finally, we are interested in tracking fast-improving PC graphics hardware so that all computation, including transfer matrix transforms and SH-rotation, may eventually be performed on the GPU.

Acknowledgements: Thanks to Paul Debevec for his light probes (http://www.debevec.org), to the Stanford University Computer Graphics Laboratory for the happy Buddha model (http://www-graphics.stanford.edu/data/3Dscanrep), and Cyberware for the tyrannosaur model. Jason Mitchell and Michael Doggett of ATI and Matthew Papakipos of NVidia kindly provided graphics hardware. We also wish to acknowledge Michael Cohen for early discussions and Charles Boyd and Hans-Peter Seidel for support.

References

[1] Agrawala, M, Ramamoorthi, R, Heirich, A, and Moll, L, Efficient Image-Based Methods for Rendering Soft Shadows, SIGGRAPH '00, 375-384.

[2] Airey, J, Rohlf, J, and Brooks, F, Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments,1990 Symposium on Interactive 3D Graphics, 24(2), 41-50.

[3] Ashikhmin, M, and Shirley, P, Steerable Illumination Textures, ACM Transactions on Graphics, 2(3), to appear.

[4] Cabral, B, Max, N, and Springmeyer, R, Bidirectional Reflection Functions from Surface Bump Maps, SIIGRAPH '87, 273-281.

[5] Cabral, B, Olano, M, and Nemec, P, Reflection Space Image Based Rendering, SIGGRAPH '99, 165-170..

[6] Cohen, M, and Wallace, J, Radiosity and Realistic Image Synthesis, Academic Press Professional, Cambridge, 1993.

[7] Cook, R, Porter, T, and Carpenter, L, Distributed Ray Tracing, SIGGRAPH '84, 137-146.

[8] Debevec, P, Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photogaphy, SIGGRAPH '98, 189-198.

[9] Debevec, P, Hawkins, T, Tchou, C, Duiker, H, Sarokin, W, and Sagar, M, Acquiring the Reflectance Field of a Human Face, SIGGRAPH 2000, 145-156.

[10] Dobashi, Y, Kaneda, K, Nakatani, H, and Yamashita, H, A Quick Rendering Method Using Basis Functions for Interactive Lighting Design, Eurographics '95, 229-240.

[11] Dorsey, J, Sillion, F, and Greenberg, D, Design and Simulation of Opera Lighting and Projection Effects, SIGGRAPH '91, 41-50.

[12] D'Zmura, M, Shading Ambiguity: Reflection and Illumination. In Computational Models of Visual Processing (1991), Landy and Movshon, eds., MIT Press, Cambridge, 187-207.

[13] Edmonds, A, Angular Momentum in Quantum Mechanics, Princeton University, Princeton, 1960.

[14] Greene, N, Environment Mapping and Other applications of World Projections, IEEE CG&A, 6(11):21-29, 1986.

[15] Greger, G., Shirley, P, Hubbard, P, and Greenberg, D, The Irradiance Volume, IEEE Computer Graphics And Applications, 6(11):21-29, 1986.

[16] Hakura, Z, and Snyder, J, Realistic Reflections and Refractions on Graphics Hardware with Hybrid Rendering and Layered Environment Maps, Eurographics Workshop on Rendering, 2001, 289-300.

[17] Haeberli, P, and Akeley, K, The Accumulation Buffer: Hardware Support for High-Quality Rendering, SIGGRAPH '90, 309-318.

[18] Heidrich, W, Lensch, H, Cohen, M, and Seidel, H, Light Field Techniques for Reflections and Refractions, Eurographics Rendering Workshop 99,195-375.

[19] Heidrich, W, Seidel H, Realistic, Hardware-Accelerated Shading and Lighting, SIGGRAPH '99, 171-178.

[20] Heidrich, W, Daubert, K, Kautz, J, and Seidel, H, Illuminating Micro Geometry based on Precomputed Visibility, SIGGRAPH '00, 455-464.

[21] Jensen, H, Global Illumination using Photon Maps, Eurographics Workshop on Rendering 1996, 21-30.

[22] Jensen, H, Marschner, S, Levoy, M, and Hanrahan, P, A Practical Model for Subsurface Light Transport, SIGGRAPH '01, '511-518.

[23] Kautz, J, and McCool, M, Interactive Rendering with Arbitrary BRDFs using Separable Approximations, Eurographics Workshop on Rendering 99,.247-260.

[24] Kautz, J, Vazquez, P, Heidrich, W, and Seidel, H, A Unified Approach to Prefiltered Environment Maps, Eurographics Workshop on Rendering 2000, 185-196.

[25] Kajiya, J, The Rendering Equation, SIGGRAPH '86, 143-150.

[26] Keating, B, and Max, N, Shadow Penumbras for Complex Objects by Depth-Dependent Filtering of Multi-Layer Depth Images, Eurographics Rendering Workshop, 1996, pp.205-220.

[27] KELLER, A, Instant Radiosity, SIGGRAPH '97, 49-56.
[28] LINDE, Y, BUZO, A, AND GRAY, R, An algorithm for Vector Quantizer Design, IEEE Transactions on Communication COM-28, 1980,84-95.
[29] LOKOVIC, T, AND VEACH, E, Deep Shadow Maps, SIGGRAPH '00, pp.385-392.
[30] MALZBENDER, T, GELB, D, AND WOLTERS, H, Polynomial Texture Maps, SIGGRAPH '01, 519-528.
[31] MAX, N, Horizon Mapping: Shadows for Bump-Mapped Surfaces, The Visual Computer, July 1998, 109-117.
[32] MILLER, G, Efficient Algorithms for Local and Global Accessibility Shading, SIGGRAPH '94, 319-326.
[33] NIMEROFF, J, SIMONCELLI, E, AND DORSEY, J, Efficient Re-rendering of Natural Environments, Eurographics Workshop on Rendering 1994, 359-373.
[34] RAMAMOORTHI, R, AND HANRAHAN, P, An Efficient Representation for Irradiance Environment Maps, SIGGRAPH '01, 497-500.
[35] REEVES, W, SALESIN, D, AND COOK, R, Rendering Antialiased Shadows with Depth Maps, SIGGRAPH '87, '283-291.
[36] SEGAL, M, KOROBKIN, C, VAN WIDENFELT, R, FORAN, J, AND HAEBERLI, P, Fast Shadows and Lighting Effects Using Texture Mapping, SIGGRAPH '92, '249-252.
[37] SCHRÖDER, P, AND SWELDENS, W, Spherical Wavelets: Efficiently Representing the Sphere, SIGGRAPH '95, '161-172.
[38] SILLION, F, ARVO, J, WESTIN, S, AND GREENBERG, D, A Global Illumination Solution for General Reflectance Distributions, SIGGRAPH '91, 187-196.
[39] SOLER, C, AND SILLION, F, Fast Calculation of Soft Shadow Textures Using Convolution, SIGGRAPH '98, '321-332.
[40] TEO, P, SIMONCELLI, E, AND HEEGER, D, Efficient Linear Re-rendering for Interactive Lighting Design, October 1997 Report No. STAN-CS-TN-97-60, Stanford University, 1997.
[41] WARD, G, RUBINSTEIN, F, AND CLEAR, R, A Ray Tracing Solution for Diffuse Interreflection, SIGGRAPH '88, '85-92.
[42] WESTIN, S, ARVO, J, TORRANCE, K, Predicting Reflectance Functions from Complex Surfaces, SIGGRAPH '92, 255-264.
[43] WILLIAMS, L, Casting Curved Shadows on Curved Surfaces, SIGGRAPH '78, 270-274.
[44] ZARE, R, Angular Momentum: Understanding Spatial Aspects in Chemistry and Physics, Wiley, New York, 1987.

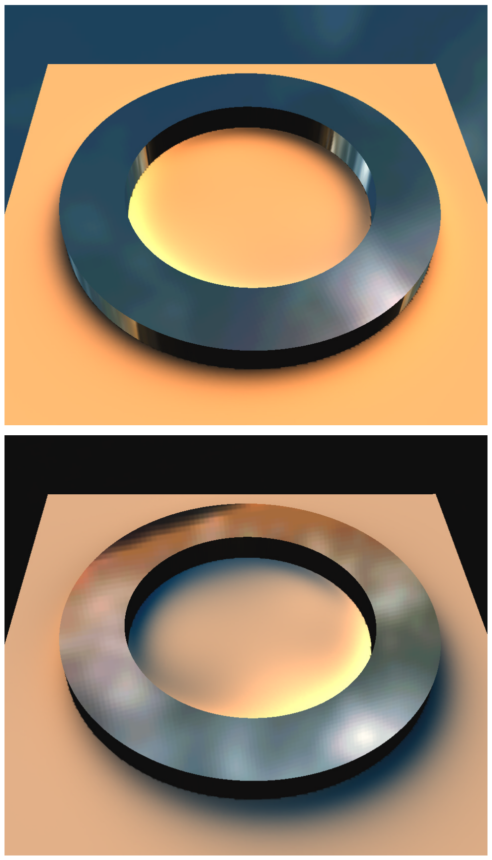


Figure 9: Real-time, dynamic caustics and shadows using diffuse self-transfer. The ring model is rendered with a traditional environment map; the ground plane was rendered using pre-computed self-transfer results from a caustic simulation combining both the specular ring and a diffuse ground (Section 5). A 25 component transfer vector was recorded over a 256x256 grid on the ground plane. These two images were generated by rotating an acquired environment around the model. A frame rate of 130Hz is obtained in our implementation .

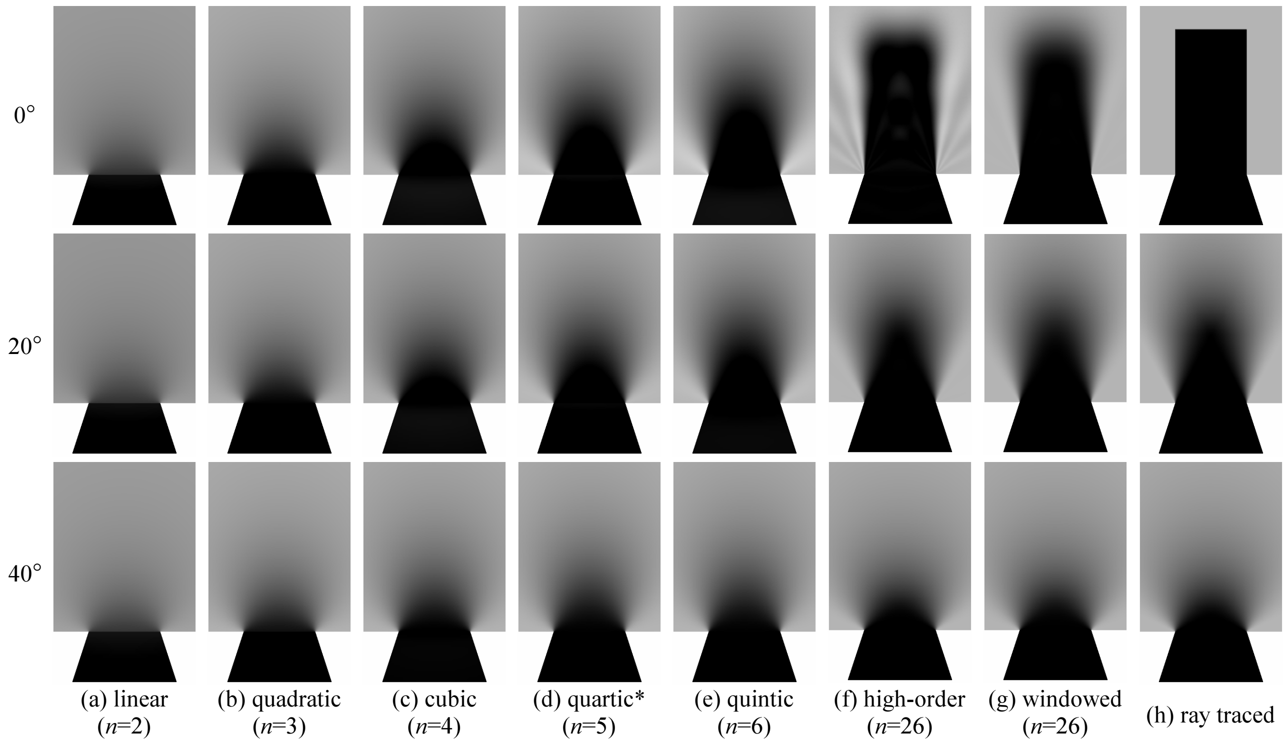


Figure 10: Comparison of SH orders for representing diffuse self-transfer. Shadows are cast onto a ground plane from a single polygon seen obliquely at bottom. Angular radius of a constant circular light used for illumination is shown at left. Higher-orders provide greater accuracy for small lights, but give rise to ringing (which is reduced by windowing in g). Note that n^2 coefficients are required for projection order n. We use the quartic projection (starred) for other result images in this paper.

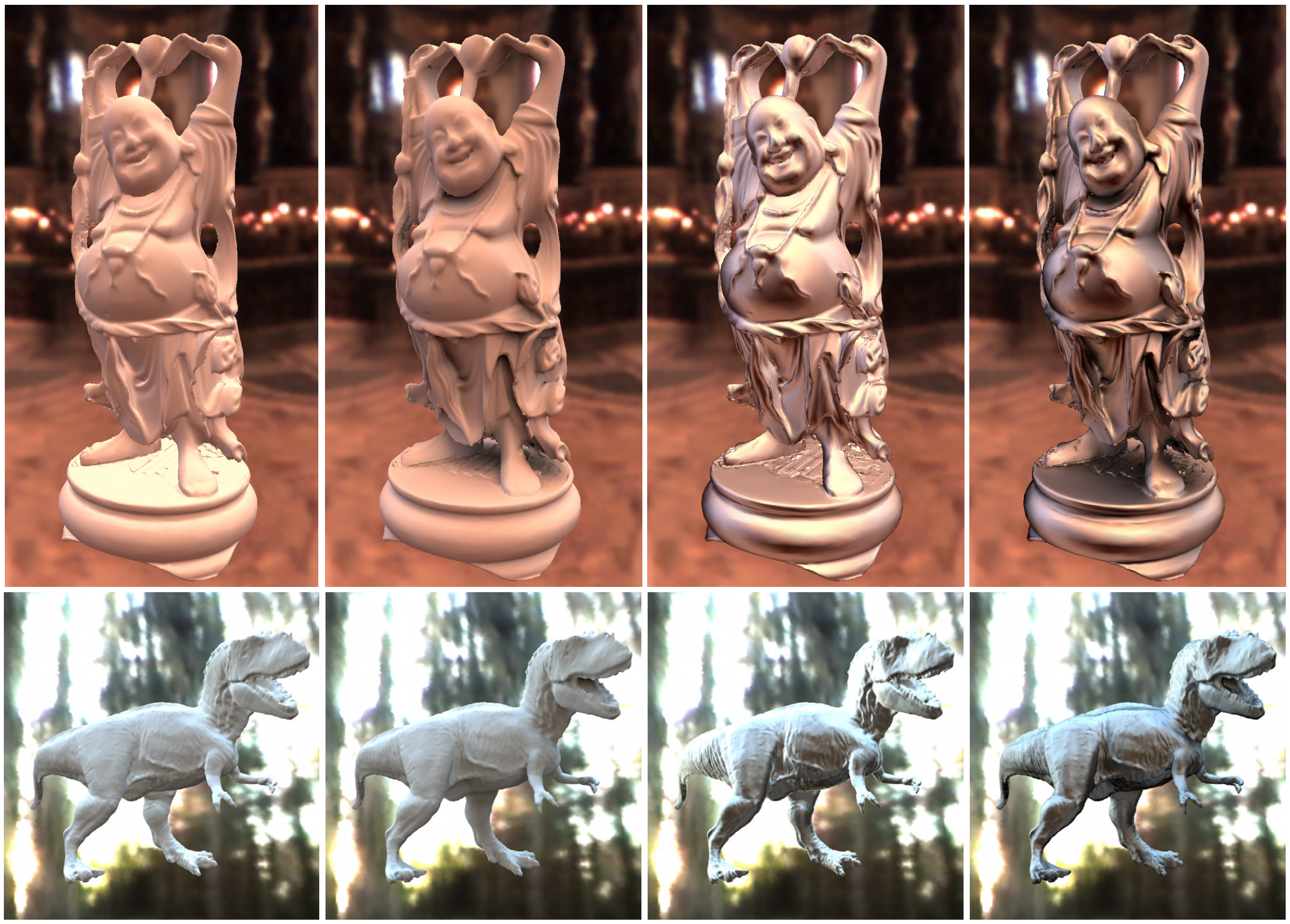


Figure 11: Diffuse and Glossy Self-transfer. Unshadowed transfer (a,c) includes no global transport effects. Interreflected transfer (b,d) includes both shadows and interreflections.

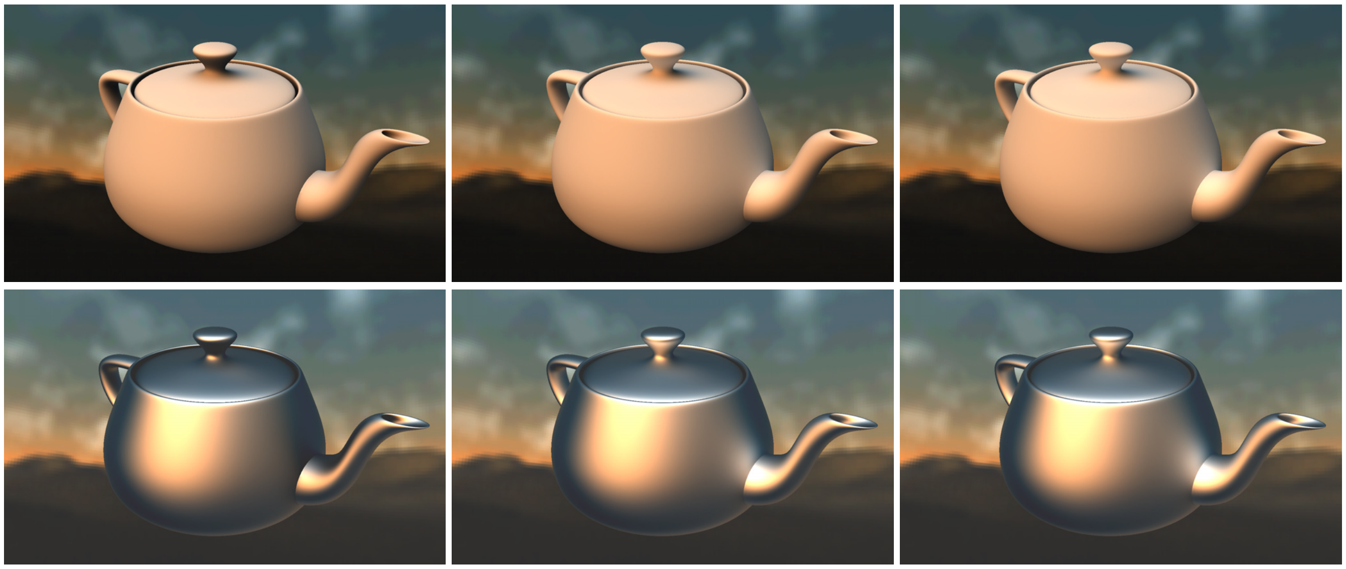

(a) no interreflections (shadowed transfer)
(b) 1-bounce interreflections
(c) 2-bounce interreflections

Figure 12: **Interreflections in Self-Transfer.** Top row shows diffuse transfer; bottom row shows glossy transfer. Note the reflections under the knob on the lid and from the spout onto the body. Run-time performance is insensitive to interreflections; only the preprocessed simulation must include additional bounces. Further bounces after the first or second typically provide only subtle change.

Cloud Rendering for Interactive Applications

Mark J. Harris
Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA
harrism@cs.unc.edu

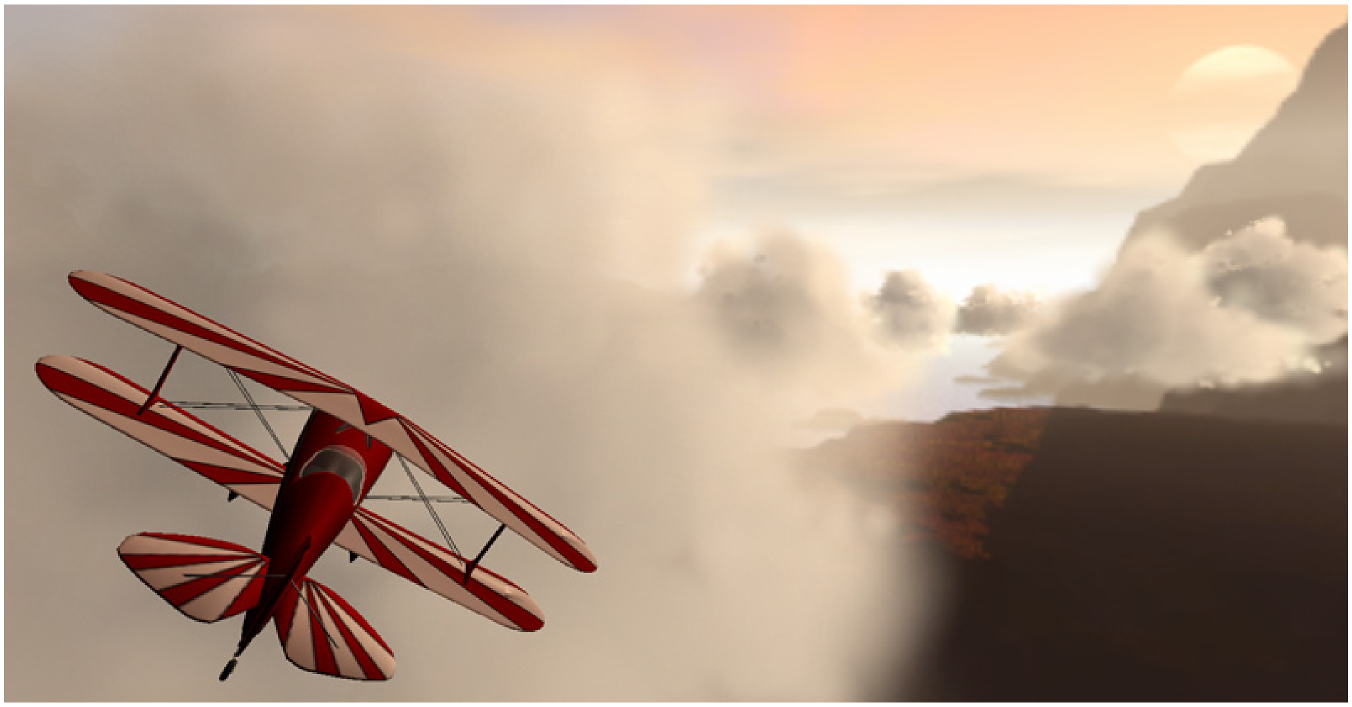


Figure 1: *Clouds add realism to interactive flight.*

1 INTRODUCTION

Clouds are an important feature of the sky; without them, synthetic outdoor scenes seem unrealistic. Game and flight simulator designers know this, so their applications nearly always have some form of clouds present. Applications in which the user's viewpoint stays near the ground can rely on techniques similar to those used by renaissance painters in ceiling frescos: distant and high-flying clouds are represented by paintings on an always distant sky dome. Flight simulators and other flying games don't have it so easy – their users are free to roam the sky.

Many techniques have been used for clouds in games and flight simulators. They have been represented with planar textures – both static and animated – or rendered as semi-transparent textured objects and fogging effects. These techniques leave a lot of effects to be desired. In a flight simulator, for example, we would like to fly in and around realistic, volumetric clouds, and to see other flying objects pass within and behind them. The goal of these course notes is to introduce the reader to existing techniques for modeling and rendering clouds. The emphasis is on techniques that are amenable to real-time implementation.

In particular, we will focus on high-speed, high-quality rendering of constant-shape clouds as described in [Harris and Lastra 2001]. We will concentrate on the rendering of realistically shaded static clouds, and will not address issues of dynamic cloud simulation. This choice enables us to generate clouds ahead of time, and to assume that cloud particles are static relative to each other. This assumption speeds cloud rendering because we need only shade them once per scene at application load time.

Before describing these techniques in detail, we provide background on other existing techniques, and on the basic physics underlying the interaction of light with clouds.

1.1 Previous Work

We will address two important areas of previous work: cloud modeling and cloud rendering. Cloud modeling deals with the data used to represent clouds in the computer, and how the data are generated and organized.

1.1.1 Cloud Modeling

As with the modeling of any object or phenomenon, there are multiple ways to represent clouds. The five techniques we will describe are particle systems, metaballs, voxel volumes, procedural noise, and textured solids. Note that these techniques are not mutually exclusive; elements of multiple techniques can be combined with good results.

Particle Systems

Particle systems model objects as a collection of *particles* – simple primitives that can be represented by a single 3D position and a small number of attributes such as radius, color, and texture. Reeves introduced particle systems as an approach to modeling clouds and other such "fuzzy" phenomena in [Reeves 1983], and described approximate methods of shading models composed of particles in [Reeves and Blau 1985]. Particles can be placed by hand using a modeling tool, procedurally generated, or with some combination of the two. Particles can be rendered in a variety of ways. The method we will describe in detail later in these notes builds clouds with particles, and renders each particle as a small, textured sprite (or "splat").

Particles have the advantage that they usually require only very simple and inexpensive code to maintain and render them. Because a particle implicitly represents a spherical volume, a cloud built with particles usually requires much less storage than a similarly detailed clouds represented with other methods. This advantage may diminish as detail increases, because many tiny particles are needed to achieve high detail. In such a situation other techniques, such as those described below, may be more desirable.

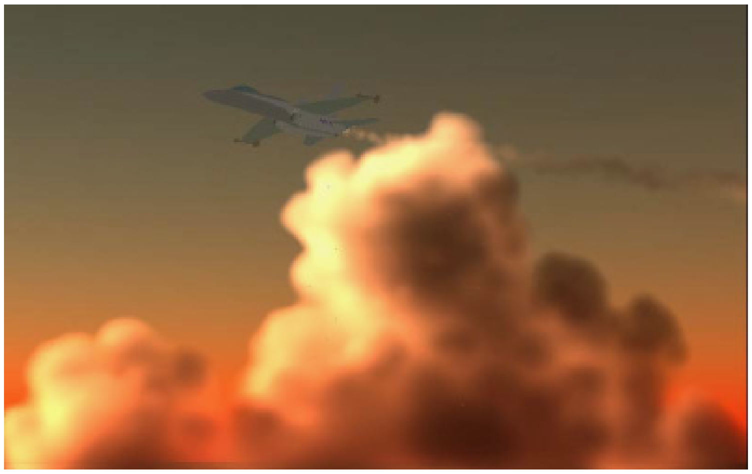


Figure 2: *These clouds, modeled with metaballs, exhibit multiple anisotropic scattering and are illuminated by sunlight and skylight. [Nishita, et al. 1996](Image courtesy of Tomoyuki Nishita)*

Metaballs

Metaballs (or "blobs") represent volumes as the superposition of potential fields represented as a set of sources defined by their center, radius, and strength [Blinn 1982a]. Such volumes can be rendered in a number of ways, including ray tracing and splatting. Alternatively, isosurfaces may be extracted and rendered (however this may not be appropriate for clouds). Metaballs were used for modeling clouds by [Nishita,

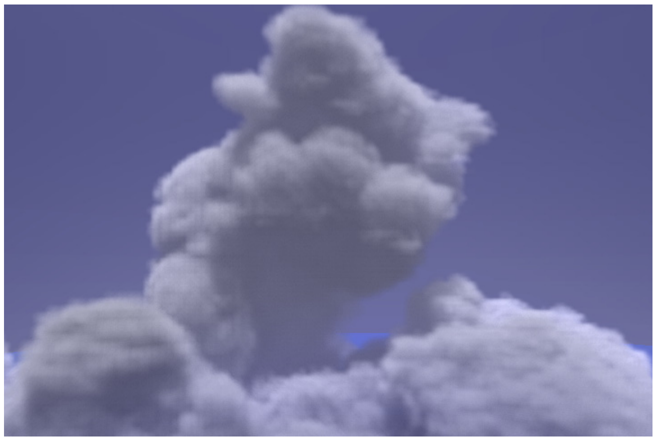


Figure 4: *This cloud was simulated using a Coupled Map Lattice model. [Miyazaki, et al. 2001](Image courtesy of Tomoyuki Nishita).*

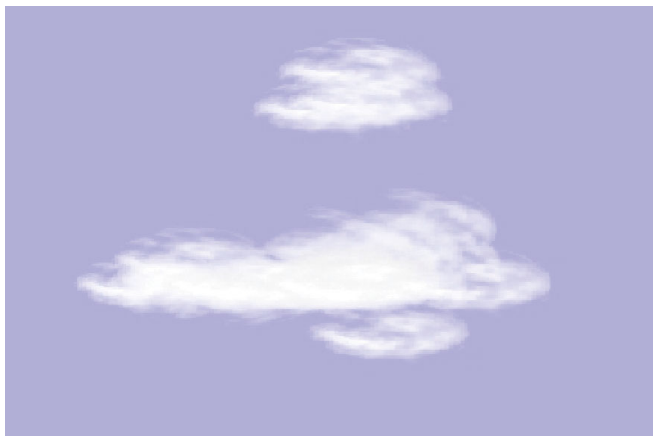


Figure 4: *Multiple textured ellipsoids used to create clouds [Elinas and Stürzlinger 2001]. (Image courtesy of Wolfgang Stürzlinger.)*

et al. 1996], who first created a basic cloud shape by hand-placing a few metaballs, and then added detail via a fractal method of generating new metaballs on the surface of existing ones (Figure 2). Metaballs were used in [Dobashi, et al. 1999] to model clouds extracted from satellite images. In [Dobashi, et al. 2000], clouds in a voxel grid were converted into metaballs for rendering with splatting (Figure 6).

Voxel Volumes

Voxels are another common representation for clouds. Voxel models provide a uniform sampling of the volume, and can be rendered with both forward and backward methods. There is a large body of existing work on volume rendering which can be drawn upon when rendering clouds represented as voxel volumes. [Kajiya and Von Herzen 1984] performed a simple physical simulation of clouds and stored the results in voxel volumes which they rendered using ray tracing.

As mentioned above, voxel grids are typically used when physically-based simulation is involved. [Dobashi, et al. 2000] simulated clouds on a voxel grid using a cellular automata model similar to [Nagel and Raschke 1992], and rendered them using metaballs, as mentioned above. [Miyazaki, et al. 2001] also performed cloud simulation on a grid using a method known as a Coupled Map Lattice (CML), and then rendered the resulting clouds in the same way. [Overby, et al. 2002] solved a set of partial differential equations to generate clouds on a voxel grid.

Procedural Noise

Procedural solid noise techniques are another important technique for generating models of clouds. These methods use noise as a basis, and perform various operations on the noise to generate random but continuous density data to fill cloud volumes [Lewis 1989;Perlin 1985]. David Ebert has done much work in modeling "solid spaces" using procedural solid noise, including offline computation of realistic images of smoke, steam, and clouds [Ebert 1997;Ebert and Parent 1990]. Figure 5 shows a cloud generated from a union of implicit functions. The

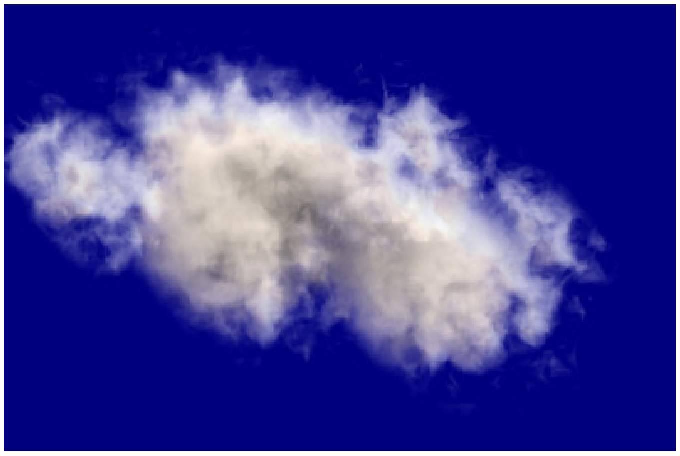


Figure 5: *A cloud generated using implicit functions and procedural noise. (Image courtesy of David Ebert.)*

solid space defined by the implicit functions is perturbed by procedural solid noise, and then rendered using a scan line renderer.

Textured Solids

Others have chosen to use surfaces to represent clouds rather than the volumetric methods described above. [Gardner 1985] used fractal texturing on the surface of ellipsoids to simulate the appearance of clouds. By combining multiple textured and shaded ellipsoids, he was able to create convincing cloudy scenes. [Lewis 1989] also demonstrated the use of ellipsoids for clouds, this time with procedural solid noise. More recently, [Elinas and Stürzlinger 2001] used a variation of Gardner's method to interactively render clouds composed of multiple ellipsoids (Figure 4).

1.1.2 Cloud Rendering

Rendering clouds is difficult because realistic shading requires the integration of the effects of optical properties along paths through the cloud volume, while incorporating the complex light scattering within the medium. Much effort has been made to approximate the physical characteristics of clouds at various levels of accuracy and complexity, and to use these approximate models to render images of clouds. Blinn introduced the use of density models for image synthesis in [Blinn 1982b], where he presented a low albedo, single scattering approximation for illumination in a uniform medium.

Kajiya and Von Herzen extended Blinn's work with methods to ray trace volume data exhibiting both single and multiple scattering [Kajiya and Von Herzen 1984]. Their method used two passes. In the first pass, scattering and absorption were integrated along paths from the light source through the cloud to each voxel where the resulting intensities were stored. In the second pass, eye rays were traced through the volume of intensities and scattering of light to the eye was computed, resulting in a cloud image. For multiple scattering, the authors derive a discrete spherical harmonic approximation to the multiple scattering equation, and solve the resulting matrix of partial differential equations using relaxation (This matrix solution replaces the first pass of the above algorithm). Following Kajiya and Von Herzen's lead, two pass techniques for computing light scattering in volumetric media – including the one we will present later – are now common.

Nishita et al. introduced additional approximations and rendering techniques for global illumination of clouds accounting for multiple anisotropic scattering and skylight [Nishita, et al. 1996]. In their method, illumination is computed on a voxel grid. The complexity of computing multiple scattering is high because it requires integrating illumination over the sphere of incoming directions. Nishita et al. showed how this complexity can be reduced by sampling only the most important directions on the sphere. Because scattering from cloud water droplets is anisotropic, with a strong peak in the forward direction, the number of sample directions is greatly reduced, saving a large amount of computation.

The rendering approach described in detail later in these notes draws most directly

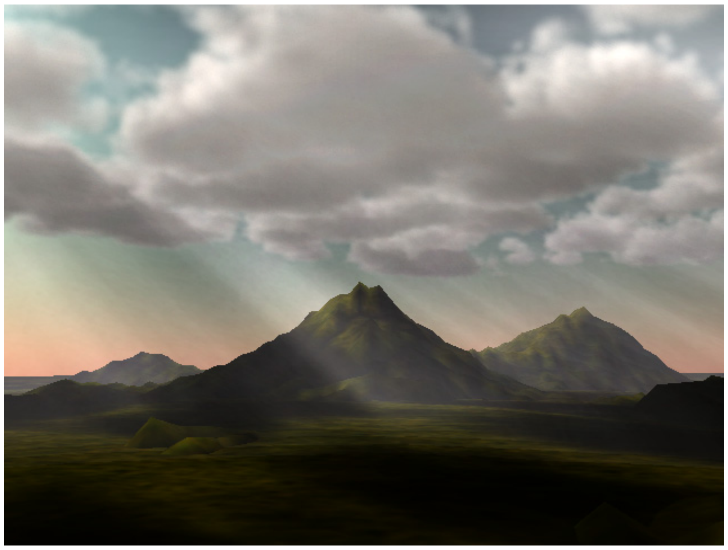


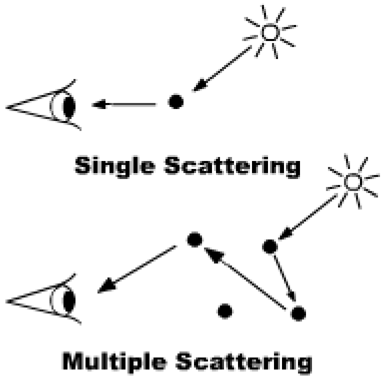
Figure 6: *These clouds were simulated using cellular automata and rendered using splatting. (Image courtesy of Tomoyuki Nishita.)*

from the rendering technique presented by [Dobashi, et al. 2000]. Their method renders clouds using a two-pass splatting algorithm in which the clouds are represented with particles. The first pass traverses the particles in sorted order moving away from the light source, using splatting and frame buffer read back to compute the amount of light that reaches each particle. In the second pass the particles are sorted with respect to the camera and then splatted from back to front into the frame buffer, using the illumination computed in the first pass as the particle color. The end result is a realistic, self-shadowing image of the cloud (Figure 6). The method we will describe later extends this method with an approximation to multiple anisotropic forward scattering. By computing the first pass only once at application load time, we are able to render static clouds at high frame rates.

2 Radiometry

This section provides a brief review of radiometric terminology. An excellent reference to the spectrum of optical models used in volume rendering, including derivations of the integral equations, is [Max 1995].

2.1 Essential Definitions



To improve clarity in the next few sections, we will review some basic radiometry terms. *Absorption* is the phenomenon by which light energy is converted into another form upon interacting with particles in a medium. For example, your skin warms in sunlight because some of the light is absorbed and transformed into heat energy. *Scattering* can be thought of as an elastic collision between matter and a photon in which the direction of the photon is changed. *Extinction*, *K*, describes the attenuation of light energy by absorption and scattering:

$$K = K_s + K_a, \qquad (1)$$

where K_s is the coefficient of scattering and K_a is the coefficient of absorption. Any light that interacts with a medium undergoes either scattering or absorption. If it does not interact, then it is *transmitted*. Extinction (and therefore scattering and absorption) is proportional to density.

Single Scattering is scattering of light by a single particle. In optically thin media (media that are either physically very thin, or very transparent), scattering of light can be approximated using single scattering models. Clear air and steam from a cup of coffee can be approximated this way, but clouds cannot. *Multiple Scattering* is scattering of light from multiple particles in succession. Models that account for only single scattering cannot accurately represent optically thick media such as clouds. Multiple scattering is the reason that clouds appear much brighter (and whiter) than the sky, since most of the light that emerges from a cloud has been scattered many times.

The *Single Scattering Albedo* is the percentage of attenuation by extinction that is due to scattering, rather than absorption:

$$\varpi = \frac{\alpha}{\alpha + \beta}. \qquad (2)$$

Single scattering albedo is the probability that a photon will "survive" an interaction with a medium. *Optical Depth* is a dimensionless measure of how opaque a medium is to light passing through it. It is the product of the physical material thickness, *d*, and the extinction coefficient *K* (assuming the material is homogeneous). An optical depth of 1 indicates that there is $e^{-1} \approx 37\%$ chance that the light will travel at least the distance *d* without scattering or absorbing. An optical depth of infinity means that the medium is opaque. The exponential given above is the *transparency* of the medium. Thus a medium with optical depth of 1 is 37% transparent.

A *Phase Function* is a function that determines, for any angle between incident and outgoing directions (the *phase angle*), how much of the incident light intensity will be scattered in the outgoing direction. For example, scattering by very small particles such as those found in clear air can be approximated using *Rayleigh scattering* [Strutt 1871]. The phase function for Rayleigh scattering is

$$p(\theta) = \frac{3}{4}\left(1 + \cos^2\theta\right), \qquad (3)$$

where θ is the phase angle. Gustav Mie developed a theory of scattering by larger particles [Mie 1908]. *Mie scattering* theory is much more complicated than Rayleigh scattering, but some simplifying assumptions can be made. A commonly used approximation for Mie scattering is the Henyey-Greenstein phase function [Henyey and Greenstein 1941]:

$$p_{HG}(\theta) = \frac{1}{4\pi}\frac{1-g^2}{(1-2g\cos\theta+g^2)^{3/2}}. \qquad (4)$$

This is the polar form for an ellipse centered at one of its foci. Anisotropy of the scattering is controlled by g, the eccentricity of the ellipse. Positive values of g will cause most of the light to be scattered in the forward direction, negative values result in backward scattering, and $g = 0$ results in isotropic scattering.

2.2 Light Scattering Illumination

Scattering illumination models simulate the emission and absorption of light by a medium as well as scattering through the medium. *Single scattering* models simulate scattering through the medium in a single direction. This direction is usually the direction leading to the point of view. *Multiple scattering* models are more physically accurate, but must account for scattering in all directions (or a sampling of all directions), and therefore are much more complicated and expensive to evaluate. The rendering algorithm presented in [Dobashi, et al. 2000] computes an approximation of cloud illumination with single scattering. This approximation has been used previously to render clouds and other participating media [Blinn 1982b;Kajiya and Von Herzen 1984].

In a multiple scattering simulation that samples N directions on the sphere, each additional order of scattering that is simulated multiplies the number of simulated paths by N. Fortunately, as demonstrated by [Nishita, et al. 1996], the contribution of most of these paths is insignificant to cloud rendering. Nishita *et al.* found that scattering illumination is dominated by the first and second orders, and therefore they only simulated up to the 4th order. They reduce the directions sampled in their evaluation of scattering to sub-spaces of high contribution, which are composed mostly of directions near the direction of forward scattering and

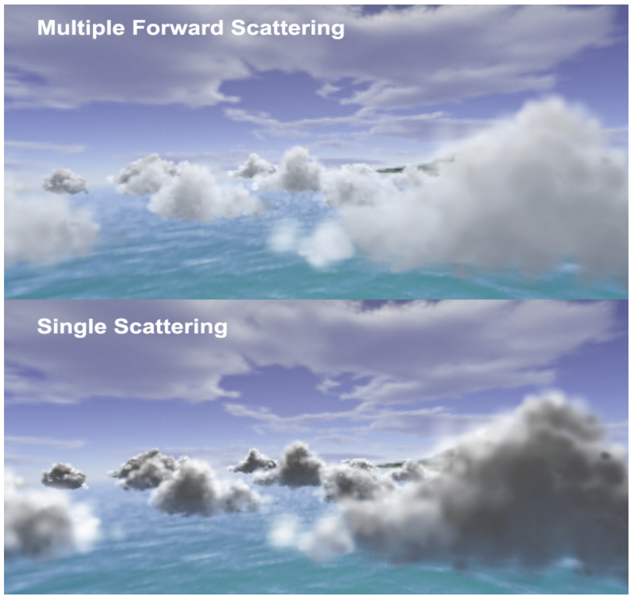


Figure 7: *A comparison of multiple forward scattering and single scattering approximations. Clouds shaded with only single scattering appear unrealistically dark.*

those directed at the viewer. Because of the dominance of the forward scattering direction, the technique we use simplifies even further, approximating multiple scattering only in the light direction – or *multiple forward scattering* – and anisotropic single scattering in the eye direction.

Our cloud rendering method is a two-pass algorithm similar to the one presented in [Dobashi, et al. 2000]: we precompute cloud shading in the first pass, and use this shading to render the clouds in the second pass. The algorithm of Dobashi *et al.*, however, uses only an isotropic single scattering approximation. If realistic values are used for the optical depth and albedo of clouds shaded with only a single scattering approximation, the clouds appear very dark [Max 1995]. This is because much of the illumination in a cloud is a result of light scattered forward along the light direction. Figure 7 shows the difference in appearance between clouds shaded with and without the multiple forward scattering approximation.

2.2.1 Multiple Forward Scattering

The first pass of our shading algorithm computes the amount of light *incident* on each particle P in the light direction, l. This light, $I(P, l)$, is composed of all direct light from direction l that is not absorbed by intervening particles, plus light scattered to P from other particles. The multiple scattering model is written

$$I(P,\omega) = I_0 \cdot e^{-\int_0^{D_P} \tau(t)dt} + \int_0^{D_P} g(s,\omega) e^{-\int_s^{D_P} \tau(t)dt} ds\,, \tag{5}$$

where I_0 is the sunlight intensity incident on the cloud, D_P is the depth of particle P in the cloud along the light direction, and

$$g(x,\omega) = \int_{4\pi} r(x,\omega,\omega')I(x,\omega')d\omega' \tag{6}$$

represents the light from all directions ω' scattered into direction ω at the point x. Here $r(x,\omega,\omega')$ is the bi-directional scattering distribution function (BSDF). It determines the percentage of light incident on x from direction ω' that is scattered in direction ω. It expands to $r(x,\omega,\omega') = a(x)\cdot\tau(x)\cdot p(\omega,\omega')$, where $\tau(x)$ and $a(x)$ are the optical depth and scattering albedo at position x, and $p(\omega,\omega')$ is the phase function (explained later).

A full multiple scattering algorithm must compute this quantity for a sampling of all light flow directions. We simplify our approximation by only sampling a small solid angle around the forward light direction, and thus compute only multiple forward scattering. So, $\omega \approx l$, and $\omega' \approx -l$, and (6) reduces to $g(x,l) = r(x,l,-l)I(x,-l)/4\pi$.

We divide the light path from 0 to D_P into discrete segments s_j, for j from 1 to N, where N is the number of cloud particles along the light direction from 0 to D_P. By approximating the integrals with Riemann Sums, we have

$$I_P = I_0 \cdot \prod_{j=1}^{N} e^{-\tau_j} + \sum_{j=1}^{N} g_j \prod_{k=j+1}^{N} e^{-\tau_k}\,. \tag{7}$$

I_0 is the intensity of light incident on the edge of the cloud. In discrete form g(x,l) becomes $g_k = a_k \tau_k p(l,-l) I_k / 4\pi$, where intensity I_k, albedo a_k, and optical thickness τ_k are represented at discrete samples (the particles) along the path of light. In order to easily transform (7) into an algorithm that can be implemented in graphics hardware, we rewrite it as an equivalent recurrence relation:

$$I_k = \begin{cases} g_{k-1} + T_{k-1} \cdot I_{k-1}, & 2 \le k \le N \\ I_0, & k = 1 \end{cases}. \qquad (8)$$

If we let $T_k = e^{-\tau_k}$ be the transparency of particle p_k, then (8) expands to (7). This representation can be intuitively understood. Starting outside the cloud, the intensity reaching particles at the cloud edge is I_0. As we trace into the cloud along the light direction, the light incident on particle k is equal to the intensity of light scattered to k from k-1 (the g_{k-1} term) plus the intensity transmitted through p_{k-1} (the $T_{k-1} \cdot I_{k-1}$ term). Notice that if g_{k-1} is expanded in (8) then I_{k-1} is a factor in both terms. Section 2.3 explains how we combine frame buffer read back with hardware blending to efficiently evaluate this recurrence.

2.2.2 Eye Scattering

In addition to multiple forward scattering and absorption, which we precompute, we also implement single scattering toward the viewer as in [Dobashi, et al. 2000]. The recurrence for this is subtly different:

$$E_k = S_k + T_k \cdot E_{k-1}, \qquad 1 \le k \le N. \qquad (9)$$

This says that the light, E_k, exiting any particle p_k is equal to the light incident on it that it does not absorb, $T_k \cdot E_{k-1}$, plus the light that it scatters, S_k. In the first pass described in the previous section, we computed the light I_k *incident* on each particle from the light source. In the second pass we are interested in the *exitant* portion of this light that is scattered toward the viewer. When S_k is replaced by $S_k = a_k \tau_k p(\omega, -l) I_k / 4\pi$, where ω is the view direction, and T_k is the same transparency factor used above, this recurrence approximates single scattering toward the viewer.

It is important to mention that (9) computes light emitted from particles using results (I_k) computed in (8). Since illumination is multiplied by the phase function in both recurrences, one might think that the phase function is multiplied twice for the same light. This is not the case, since in (8), I_{k-1} is multiplied by the phase function to determine the amount of light P_{k-1} scatters to P_k in the light direction, and in (9) I_k is multiplied by the phase function to determine the amount of light that P_k scatters in the view direction. Even if the viewpoint is directly opposite the light source, since the light *incident* on P_k is stored and used in the scattering computation, the phase function is never taken into account twice at the same particle when computing the *exitant* intensity.

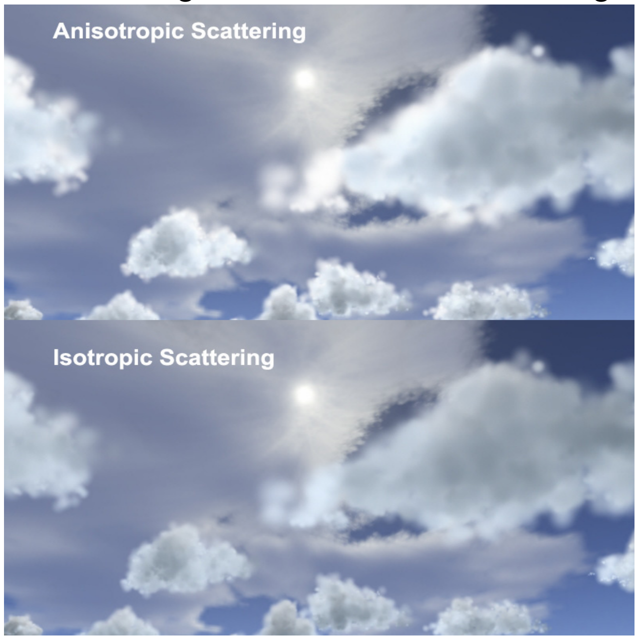


Figure 8: *A comparison of clouds rendered with isotropic and anisotropic scattering.*

2.2.3 Phase Function

The phase function $p(\omega, \omega')$ mentioned above is very important to cloud shading. Clouds exhibit anisotropic scattering of light (including the strong forward scattering that we assume in our multiple forward scattering approximation). The phase function determines the distribution of scattering for a given incident light direction. The use of phase functions in cloud rendering is discussed in detail in [Blinn 1982b;Max 1995;Nishita, et al. 1996]. The clouds in Figures 1, 7, 8, 10, 12, 13, and 14 were generated using a simple Rayleigh scattering phase function given in section 2.1. Rayleigh scattering favors scattering in the forward and backward directions. While a Mie scattering function would be more realistic, we have achieved good results with the simpler Rayleigh scattering model. Figure 8 demonstrates the differences between clouds shaded with and without anisotropic scattering. Anisotropic scattering gives the clouds a characteristic "silver lining" when viewed looking into the sun.

2.3 Rendering Algorithm

Armed with recurrences (8) and (9) and a standard graphics API such as OpenGL or Direct3D, computation of cloud illumination is straightforward. The algorithm described here is similar to the one presented by [Dobashi, et al. 2000] and has two phases: a shading phase that runs once per scene and a rendering phase that runs in real time. The key to the implementation is the use of hardware blending and pixel read back.

Blending operates by computing a weighted average of the frame buffer contents (the *destination*) and an incoming fragment (the *source*), and storing the result back in the frame buffer. This weighted average can be written

$$c_{result} = f_{src} \cdot c_{src} + f_{dest} \cdot c_{dest} \,. \tag{10}$$

If we let $c_{result} = I_k$, $f_{src} = 1$, $c_{src} = g_{k\text{-}1}$, $f_{dest} = T_{k\text{-}1}$, and $c_{dest} = I_{k-1}$, then we see that (8) and (10) are equivalent if the contents of the frame buffer before blending represent I_0. This is not quite enough, though, since as we saw before, $I_{k\text{-}1}$ is a factor of both terms in (8). To solve the recurrence for a particle p_k, we must know how much light is incident on particle $p_{k\text{-}1}$ beforehand. To do this, we employ pixel read back.

To compute (8) and (9), we use the procedure described by the pseudocode in Figure 9. This pseudocode shows that we use a nearly identical algorithm for preprocess and runtime. The differences are as follows. In the illumination pass, the frame buffer is cleared to white and particles are sorted with respect to the light. As a particle is blended into the frame buffer, blending attenuates the intensity of each fragment by the opacity of the particle, and increases the intensity by the amount the particle scatters in the forward direction. The percentage of light that reaches p_k, is found by reading back the color of pixels in the frame buffer onto which the particle projects immediately before rendering it. I_k is computed by multiplying this percentage by

```
Source_blend_factor = 1;
destination_blend_factor = 1 – source_alpha;
texture mode = modulate;
$l$ = direction from light;
if (preprocess) then {
  $\omega = -l$;
  view cloud from light source;
  clear frame buffer to white;
  particles.Sort(ascending order by distance to light);
}
else {
  view cloud from eye position;
  particles.Sort(descending order by distance to eye);
}
foreach particle $p_k$ {
[$p_k$ has extinction $\tau_k$, albedo $a_k$, radius $r_k$, color, and alpha] {
  if (preprocess) then {
    $x$ = pixel at projected center of $p_k$;
    $i_k$ = color($x$) * light_color;
    $p_k$.color = $a_k * \tau_k * i_k / 4\pi$;
    $p_k$.alpha = 1 - exp($-\tau_k$);
  }
  else {
    $\omega = p_k$.position – view_position;
  }
  $c = p_k$.color * phase($\omega$, $l$);
  render $p_k$ with color c, side 2*$r_k$;
}
```

Figure 9: *Pseudocode for illuminating and rendering clouds.*

the light intensity. I_k is used to compute multiple forward scattering in (8) and eye scattering in (9).

The runtime phase uses the same algorithm, but with particles sorted with respect to the viewpoint, and without reading pixels. The precomputed illumination of each particle I_k is used in this phase to compute scattering toward the eye.

In both passes, we render particles in sorted order as polygons textured with a Gaussian "splat" texture. The polygon color is set to the scattering factor $S_k = a_k \tau_k p(\omega, -l) I_k / 4\pi$ and the texture is modulated by this color. In the first pass, ω is the light direction, and in the second pass it is the direction of the viewer. The source and destination blending factors are set to one and one minus source alpha, respectively. All cloud images in these notes were computed with a constant τ of 80.0 (units are *length*$^{-1}$), and an albedo of 0.9.

2.3.1 Skylight

The most awe-inspiring images of clouds are created by the multi-colored spectacle of a beautiful sunrise or sunset. These clouds are often not illuminated directly by the sun at all, but by skylight – sunlight that is scattered by the atmosphere. The fact that light accumulates in an additive manner provides us with a simple extension to our shading method that allows the creation of such beautiful clouds. We simply shade clouds from multiple light sources and store the resulting particle colors (i_k in the algorithm above) from all shading iterations. At render time, we evaluate the phase function at each particle once per light. By doing so, we can approximate global illumination of the clouds.

While this technique is not completely physically-based, it is better than an ambient light approximation, since it is directional and results in shadowing in the clouds as well as anisotropic scattering from multiple light directions and intensities. We obtained best results by using the images that make up the sky dome we place in the distance over our environments to guide the placement and color of lights. Figure 14 shows a scene at sunset in which we use two light sources, one orange and one pink, to create sunset lighting. In addition to illumination from multiple light sources, we optionally use a small ambient term to provide some compensation for scattered light lost due to our scattering approximation.

3 Dynamically Generated Impostors

While the cloud rendering method described above provides beautiful results and is fast for relatively simple scenes, it suffers under the weight of many complex clouds. The games for which we developed this system dictate that we must render complicated cloud scenes at fast interactive rates. Clouds are only one component of a complex game environment, and therefore can only use a small percentage of a frame time.

The integration (section 2.2) required to accurately render volumetric media results in high rates of pixel overdraw. Clouds have inherently high depth complexity, and require blending, making rendering them a difficult job even for current hardware with the highest fill rates.

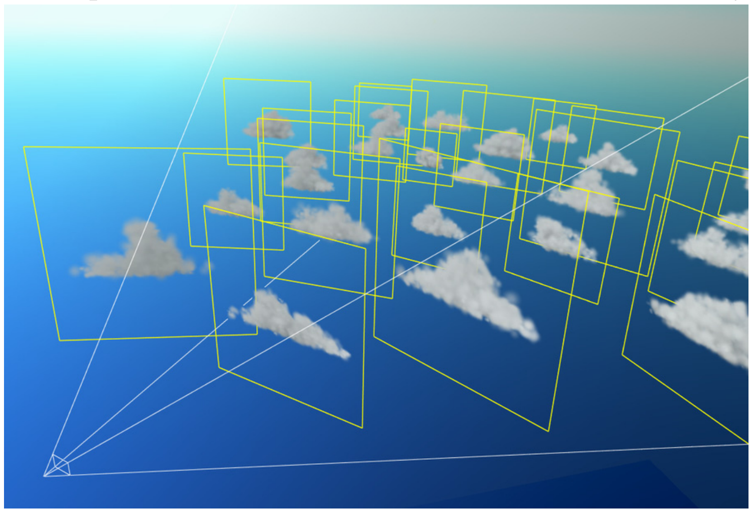


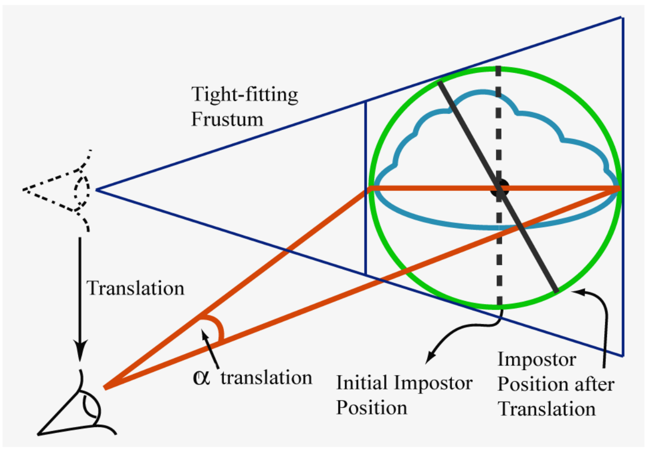
Figure 10: *Impostors, shown outlined in this image, are textured polygons oriented toward the viewer.*

In addition, as the viewpoint approaches a cloud, the projected area of that cloud's particles increases, becoming greatest when the viewpoint is within the cloud. Thus, pixel overdraw is increased and rendering slows as the viewpoint nears and enters clouds.

In order to render many clouds made up of many particles at high frame rates, we need a way to surmount fill rate limitations, either by reducing the amount of pixel overdraw performed, or by amortizing the rendering of cloud particles over multiple frames. *Dynamically generated impostors* allow us to do both.

Impostors are a common technique for accelerating interactive rendering [Maciel and Shirley 1995;Schaufler 1995;Shade, et al. 1996]. An impostor replaces an object in the scene with a semi-transparent polygon texture-mapped with an image of the object it replaces (Figure 10). The image is a rendering of the object from a viewpoint *V* that is valid (within some error tolerance) for viewpoints near *V*. Impostors used for appropriate points of view give a very close approximation to rendering the object itself. An impostor is valid (with no error) for the viewpoint from which its image was generated, regardless of changes in the viewing direction. Impostors may be precomputed for an object from multiple viewpoints, requiring much storage, or they may be generated only when needed. We choose the latter technique, called *dynamically generated impostors* by [Schaufler 1995].

We generate impostors using the following procedure. A view frustum is positioned so that its viewpoint is at the position from which the impostor will be viewed, and it is tightly fit to the bounding volume of the object (Figure 11). We then render the object into an image used to texture the impostor polygon.


Tight-fitting Frustum
Translation
α translation
Initial Impostor Position
Impostor Position after Translation
Figure 11**:** *Impostor translation error metric.*

As mentioned above, we can use impostors to amortize the cost of rendering clouds over multiple frames. We do this by exploiting the frame-to-frame coherence inherent in three-dimensional scenes: the relative motion of objects in a scene decreases with distance from the viewpoint, and objects close to the viewpoint present a similar image for some time. This lack of sudden changes in the image of an object allows us to re-use impostor images over multiple frames. We can compute an estimate of the error in an impostor representation that we use to determine when the impostor needs to be updated. Certain types of motion introduce error in impostors more quickly than others [Schaufler 1995] presents two worst-case error metrics for this purpose. The first, which we will call the *translation error*, computes error caused by translation away from the viewpoint at which the current impostor was generated. The second computes error introduced by moving straight toward the object, which we call the *zoom error*.

We use the same translation error metric, and replace zoom error by a texture resolution error metric. For the translation error metric, we simply compute the angle α_{trans}, shown in Figure 11, and compare it to a specified tolerance. The zoom error metric compares the current impostor texture resolution to the required resolution for the texture, computed using the following equation [Schaufler 1995]

$$resolution_{texture} = resolution_{screen} \cdot \frac{object\ size}{object\ dist}. \qquad (11)$$

If either the translation error is greater than an error tolerance angle or the current resolution of the impostor is less than the required resolution, we regenerate the impostor from the current viewpoint. We find that a tolerance of about 0.15 degree reduces impostor "popping" to an imperceptible level while maintaining good performance. For added performance, tolerances up to one degree can be used with more noticeable (but not excessive) popping.

In the past, impostors were used mostly to replace geometric models. Since these models have high frequencies in the form of sharp edges, impostors have usually been used only for distant objects. Nearby objects must have impostor textures of a resolution at or near that of the screen, and their impostors require frequent updates. We use impostors for clouds no matter where they are in relation to the viewer. The couds we model have very few high frequency details like those of geometric models, so artifacts caused by low texture resolution are less noticeable. Clouds have very high fill rate requirements, so cloud impostors are beneficial even when they must be updated every few frames.

3.1 Head in the Clouds

Impostors can provide a large reduction in overdraw even for viewpoints inside the cloud, where the impostor must be updated every frame. The "foggy" nature of clouds makes it difficult for the viewer to discern detail when inside them. In addition, in games and flight simulators, the viewpoint is often moving. These factors allow us to reduce the resolution at which we render impostor textures for clouds containing the viewpoint by about a factor of 4 in each dimension.

However, impostors cannot be generated in the same manner for these clouds as for distant clouds, since the view frustum cannot be tightly fit to the bounding volume as described above. Instead, we use the same frustum used to display the whole scene to generate the texture for the impostor, but create the texture at a lower resolution, as described above. We display these impostors as screen-space rectangles sized to fill the screen.

3.2 Objects in the Clouds

In order to create effective interactive cloudy scenes, we must allow objects to pass in and through the clouds, and we must render this realistically. Impostors pose a problem because they are two-dimensional. Objects that pass through impostors appear as if they are passing through images floating in space, rather than through fluffy, volume-filling clouds.

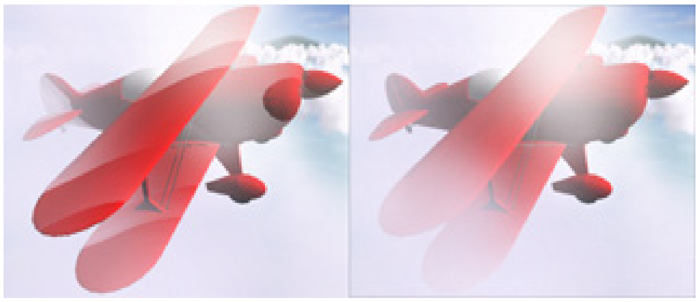


Figure 12 *An airplane in the clouds. On the left, particles are directly rendered into the scene. Artifacts of their intersection with the plane are visible. On the right, the airplane is rendered between impostor layers, and no artifacts are visible.*

One way to solve this problem would be to detect clouds that contain objects and render their particles directly to the frame buffer. But by doing so we would sacrifice the benefits that impostors provide us. Instead, we detect when objects pass within the bounding volume of a cloud, and split the impostor representing that cloud into multiple layers. When an object resides inside a cloud, the cloud is rendered as two layers: one for the portion of cloud particles that lies approximately behind the object with respect to the viewpoint, and one for the portion that lies approximately in front of the object. If two objects lie within a cloud, then we need three layers, and so on. Since cloud particles must be sorted for rendering anyway, splitting the cloud into layers adds little expense. This "impostor splitting" results in a set of alternating impostor layers and objects. This set is rendered from back to front, with depth testing enabled for objects, and disabled for impostors. The result is an image of a cloud that realistically contains objects, as shown on the right side of Figure 12.

Impostor splitting provides an additional advantage over direct particle rendering for clouds that contain objects. When rendering cloud particles directly, the billboards used to render particles may intersect the geometry of nearby objects. These intersections cause artifacts that break the illusion of particles representing elements of volume. Impostor splitting avoids these artifacts (Figure 12).

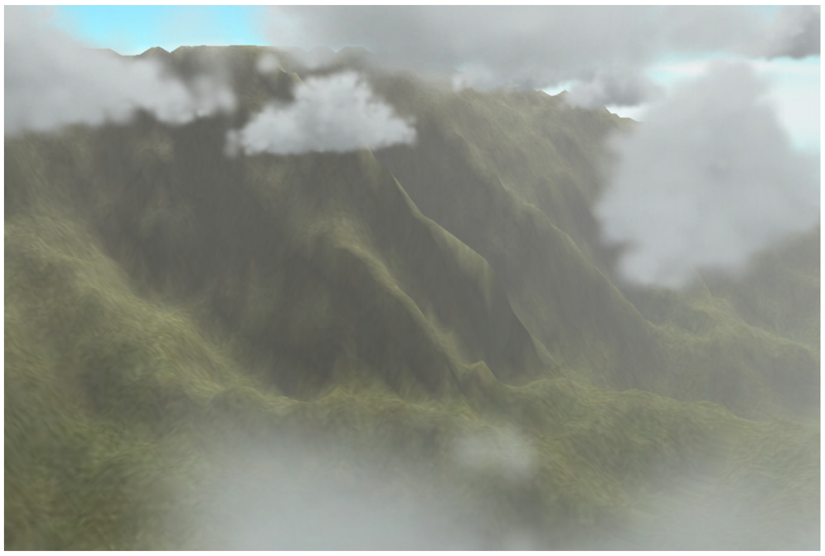


Figure 13: *Clouds rendered in real time in* SkyWorks.

4 Results

We have implemented the cloud rendering system described here using the OpenGL API. The code was originally developed to run on a PC with an NVIDIA GeForce 256 graphics processor (circa 1999). Even on older graphics cards like this, we can achieve very high frame rates by using impostors and view-frustum culling to accelerate rendering. Scenes containing hundreds of thousands of particles render at greater than 50 frames per second. If the viewpoint moves slowly enough to keep impostor update rates low, we can render a scene of more than 1.2 million particles at about 10 to 12 frames per second. Slow movement is a reasonable assumption for flight simulators and games because the user's aircraft is typically much smaller than the clouds through which it is flying, so the frequency of impostor updates remains low. On the most recent hardware, performance is much higher. Much more complex scenes can be rendered at over 100 frames per second.

As mentioned before, cloud shading computations are performed in a preprocess. For scenes with only a few thousand particles shading takes less than a second and scenes of a few hundred thousand particles can be shaded in a few seconds per light source.

SkyWorks, an efficient open source implementation of this cloud rendering system, can be downloaded at http://www.cs.unc.edu/~harrism/SkyWorks (Figure 13).

5 Conclusion

These notes presented methods for shading and rendering realistic clouds at high frame rates. The shading and rendering algorithm simulates multiple scattering in the light direction, and anisotropic single scattering in the view direction. Clouds can be illuminated by multiple directional light sources, with anisotropic scattering from each.

This method uses impostors to accelerate cloud rendering by exploiting frame-to-frame coherence and greatly reducing pixel overdraw. Impostors are an advantageous representation for clouds even in situations where they would not be successfully used to represent other objects, such as when the viewpoint is in or near a cloud. Impostor splitting is an effective way to render clouds that contain other objects, reducing artifacts caused by direct particle rendering.

6 Acknowledgements

This work was supported by iROCK Interactive, NVIDIA Corporation, NIH National Center for Research Resources, Grant No. P41 RR 02170, and Department of Energy ASCI program, National Science Foundation grant ACR-9876914.

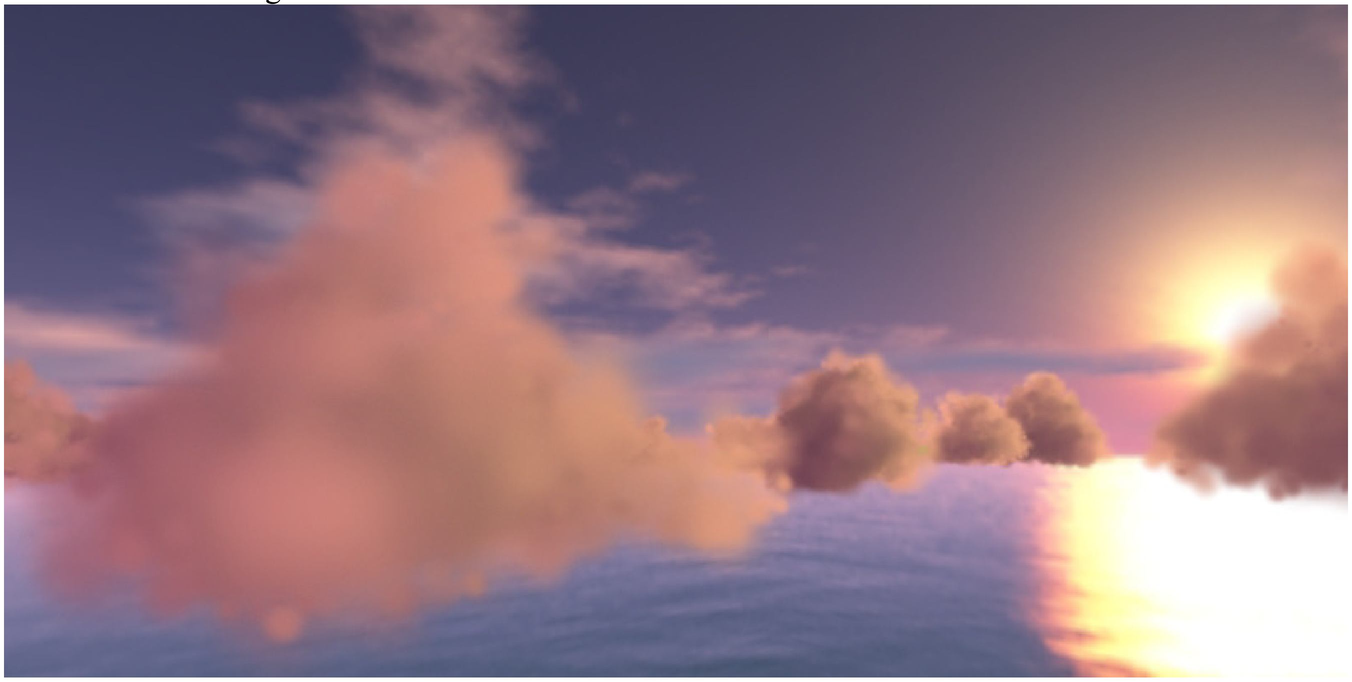


Figure 14: *These clouds are shaded with multiple light sources to approximate skylight.*

6.1 References

For more information, updates, lecture notes, and demos, see http://www.cs.unc.edu/~harrism/clouds, and http://www.cs.unc.edu/~harrism/SkyWorks. Another good reference site for clouds is http://www.vterrain.org/Atmosphere/Clouds/index.html.

[Blinn 1982a] Blinn, J.F. A generalization of algebraic surface drawing. (*Proceedings of SIGGRAPH 1982a)*, ACM Press, 273-274. 1982a.
[Blinn 1982b] Blinn, J.F. Light reflection functions for simulation of clouds and dusty surfaces. Computer Graphics (*Proceedings of SIGGRAPH 1982b)*, ACM Press, 21-29. 1982b.
[Dobashi, et al. 2000] Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T. and Nishita, T. A Simple, Efficient Method for Realistic Animation of Clouds. Computer Graphics (*Proceedings of SIGGRAPH 2000)*, ACM Press / ACM SIGGRAPH, 19-28. 2000.
[Dobashi, et al. 1999] Dobashi, Y., Nishita, T., Yamashita, H. and Okita, T. Using Metaballs to Modeling and Animate Clouds from Satellite Images. *The Visual Computer*, *15*. 471-482.1999.
[Ebert 1997] Ebert, D.S. Volumetric modeling with implicit functions: a cloud is born. *ACM SIGGRAPH 97 Visual Proceedings*. 147.1997.
[Ebert and Parent 1990] Ebert, D.S. and Parent, R.E. Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques. Computer Graphics (*Proceedings of SIGGRAPH 1990)*, ACM Press, 357-366. 1990.
[Elinas and Stürzlinger 2001] Elinas, P. and Stürzlinger, W. Real-time Rendering of 3D Clouds. *Journal of Graphics Tools*.2001.

[Gardner 1985] Gardner, G.Y. Visual Simulation of Clouds. Computer Graphics (*Proceedings of SIGGRAPH 1985)*, 297-303. 1985.
[Harris and Lastra 2001] Harris, M.J. and Lastra, A. Real-Time Cloud Rendering. Computer Graphics Forum (*Proceedings of Eurographics 2001)*, Blackwell Publishers, 76-84. 2001.
[Henyey and Greenstein 1941] Henyey, L.G. and Greenstein, J.L. Diffuse Radiation in the Galaxy. *The Astrophysical Journal*, *90*. 70-83.1941.
[Kajiya and Von Herzen 1984] Kajiya, J.T. and Von Herzen, B.P. Ray tracing volume densities. Computer Graphics (*Proceedings of SIGGRAPH 1984)*, ACM Press, 165-174. 1984.
[Lewis 1989] Lewis, J.P. Algorithms for solid noise synthesis. Computer Graphics (*Proceedings of SIGGRAPH 1989)*, ACM Press, 263-270. 1989.
[Maciel and Shirley 1995] Maciel, P.W.C. and Shirley, P. Visual navigation of large environments using textured clusters. (*Proceedings of Symposium on Interactive 3D Graphics 1995)*, ACM Press, 95 - ff. 1995.
[Max 1995] Max, N. Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, *1* 2. 99-108.1995.
[Mie 1908] Mie, G. Bietage zur Optik truver medien Speziell Kolloidaler Metallosungen. *Annallen der Physik*, *25* 3. 377.1908.
[Miyazaki, et al. 2001] Miyazaki, R., Yoshida, S., Dobashi, Y. and Nishita, T. A Method for Modeling Clouds Based on Atmospheric Fluid Dynamics. (*Proceedings of The Ninth Pacific Conference on Computer Graphics and Applications 2001)*, IEEE Computer Society Press, 363-372. 2001.
[Nagel and Raschke 1992] Nagel, K. and Raschke, E. Self-organizing criticality in cloud formation? *Physica A*, *182*. 519-531.1992.
[Nishita, et al. 1996] Nishita, T., Dobashi, Y. and Nakamae, E. Display of clouds taking into account multiple anisotropic scattering and sky light. Computer Graphics (*Proceedings of SIGGRAPH 1996)*, ACM Press, 379-386. 1996.
[Overby, et al. 2002] Overby, D., Melek, Z. and Keyser, J. Interactive Physically-Based Cloud Simulatin. (*Proceedings of Pacific Graphics 2002)*, 469-470. 2002.
[Perlin 1985] Perlin, K. An Image Synthesizer. Computer Graphics (*Proceedings of SIGGRAPH 1985)*, ACM Press, 287-296. 1985.
[Reeves 1983] Reeves, W. Particle Systems - A Technique for Modeling a Class of Fuzzy Objects. Computer Graphics (*Proceedings of SIGGRAPH 1983)*, ACM Press, 359-375. 1983.
[Reeves and Blau 1985] Reeves, W. and Blau, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems. (*Proceedings of SIGGRAPH 1985)*, ACM Press, 313-322. 1985.
[Schaufler 1995] Schaufler, G. Dynamically Generated Impostors. (*Proceedings of GI Workshop "Modeling - Virtual Worlds - Distributed Graphics" 1995)*, infix Verlag, 129-135. 1995.
[Shade, et al. 1996] Shade, J., Lischinski, D., Salesin, D.H., DeRose, T. and Snyder, J. Hierarchical image caching for accelerated walkthroughs of complex environments. (*Proceedings of SIGGRAPH 1996)*, ACM Press, 75-82. 1996.
[Strutt 1871] Strutt, J.W. (Lord Rayleigh). On the light from the sky, its polarization and colour. *Philos. Mag.*, *41*. 107-120, 274-279.1871.