

國立交通大學

資訊科學與工程研究所

碩士論文

非等向性散射之光影模型及其即時顯像

A Real-Time Analytic Lighting Model for Anisotropic
Scattering

研究生：洪凱生

指導教授：莊榮宏 教授

林正中 教授

中華民國九十五年八月

A Real-Time Analytic Lighting Model for Anisotropic Scattering

by
Roger Hong

A dissertation submitted to the faculty of National Chiao Tung University in partial satisfaction of the requirements for the degree of Master of Science in the Department of Computer Science.



Committee in charge:
Professor Jung-Hong Chuang
Professor Yung-Yu Chuang
Professor Chun-Fa Chang

Spring 2006

A Real-Time Analytic Lighting Model for Anisotropic Scattering

Copyright 2006

by

Roger Hong



A Real-Time Analytic Lighting Model for Anisotropic Scattering

Student: Roger Hong Advisor: Jung-Hong Chuang

Department of Computer Science
National Chiao Tung University

Abstract

Rendering the effects of light scattering for interactive applications is often made difficult by the demands of dynamic viewing, lighting, scene geometry, and environment parameters. Recent advances in rendering of participating media using analytic techniques have enhanced performance significantly to meet this challenge, albeit under assumptions and simplifications made to scenes. In this thesis, we present an analytic lighting model for dynamic, accurate, and single anisotropic scattering that may be used for scenes such as fog, mist, or haze. Our model applies an explicit analytic integration to an angle-formulated single scattering integral for a point light source in a homogenous participating medium. Users may create a dynamic scattering environment, such that scattering conditions may be controlled using simple parameters for density and forward scattering level. Using this model, we demonstrate the effect of forward scattering on glows around light sources and surface radiance. The method can be implemented on programmable graphics hardware in real-time.

To my family,
for abounding encouragement
and levity.



Contents

List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Motivation	2
1.2 Contribution	2
1.3 Limitations	4
1.4 Organization	4
2 Light Scattering for Computer Graphics	5
2.1 Phase Functions and Anisotropic Scattering	6
2.1.1 Rayleigh Scattering	7
2.1.2 Mie Scattering: Henyey-Greenstein Phase Function	7
2.2 Light Scattering Terminology	8
2.2.1 Absorption, Emission, Scattering, and Extinction	8
2.2.2 Transmittance and Optical Thickness	10
2.3 Integral Transport Equation	11
2.3.1 Simplifying Assumptions	12
3 Related Work: Interactive Rendering of Light Scattering Effects	16
3.1 Numerical Methods	17
3.1.1 Shafts of Light and Volumetric Shadows	17
3.1.2 Atmospheric Effects	18
3.1.3 Multiple Scattering	19
3.2 Analytic Methods	21
3.2.1 Effects for Point Light Sources	21
3.2.2 Sky and Terrain Rendering	23
4 Analytic Lighting Model for Accurate and Dynamic Anisotropic Scattering	25
4.1 Introduction	26
4.2 Analytic Lighting Model	27
4.2.1 Angle-Formulated Airlight Model	27

4.2.2	Surface Radiance Model using Angle Formulation	30
4.2.3	Summary of Analytic Lighting Model	35
4.3	Dynamic Anisotropic Scattering	37
4.4	Hardware Implementation	40
5	Experimental Results	44
5.1	Visual Effects of Anisotropic Scattering	44
5.2	Rendering Test Scenes	47
6	Conclusion	51
6.1	Summary	51
6.2	Future Work	52
6.2.1	Analytic Model for Scattering from Anisotropic Light Sources	52
6.2.2	Volumetric Shadows	53
	Bibliography	59



List of Figures

2.1	Henye-Greenstein phase function for $g = 0.5$	8
2.2	Diagram of single scattering integral for point light source	15
2.3	Diagram for single scattering integral of atmospheric model	15
4.1	Diagram illustrating light paths	27
4.2	Diagram for angular reformulation of the airlight integral	28
4.3	3D plot of precomputed function $H(u, v)$	30
4.4	Diagram for surface radiance integral	31
4.5	3D plots of precomputed functions $J^1(T_{sp}, \theta_s)$ and $J^{20}(T_{sp}, \theta_s)$	34
4.6	Illustrative summary of analytic lighting model of Section 4.2	36
4.7	(left)Normalized Henye-Greenstein phase function and its Legendre polynomial expansion. (right)Rms error plot of Legendre polynomial approximation	40
4.8	Fragment shader pseudocode for version without dynamic anisotropic scattering	42
4.9	Fragment shader pseudocode for version with dynamic anisotropic scattering	43
5.1	Glows around light sources in and out of direct view for varying forward scattering conditions	45
5.2	Specular radiance and diffuse radiance for varying forward scattering conditions	46
5.3	Test Scene: varying densities	48
5.4	Test Scene: varying forward scattering	49
5.5	Images of pavilion scene.	50
5.6	Images of atrium scene.	50
6.1	Diagram of scattering with anisotropic light source	53
6.2	Diagram for volumetric shadow algorithm	55
6.3	Image of torus' volumetric shadow	56
6.4	Image of table's volumetric shadow	57
6.5	Image illustrating problem when there are intersecting shadow polygons . .	57
6.6	Image illustrating clamping problems	58
6.7	Image of car's volumetric shadow	58

List of Tables

4.1	Notation and formulae for angle-formulated airlight integral	28
5.1	Table of rendering speeds as a function of number of light sources	47
5.2	Table of computation costs for precomputed function tables	47



Acknowledgments

I would like to thank my advisors, Professor Jung-Hong Chuang and Professor Cheng-Chung Lin, for their guidance and patience. Instilling a focus to my research proved invaluable. An appreciative thanks goes to Tan-Chi Ho for his continual help from the very beginnings to the latter stages . I am fortunate he makes himself readily available. I would also like to thank Yu-Hsian Sun for sparking what became the topic of my thesis. Thanks to Ya-Ching Chiu for painstaking translation work along the way. A big thanks to the entire group: Yi-Chun Lin, Chao-Wei Juan, Yong-Cheng Chen, Chen-Li Hao, Yu-Shuo Lio, Chia-Lin Ko, Ji-Wen Chon, and Yung-Cheng Chen. You guys answer every last question of mine. Thanks to Jyh-Shyang Chang for modeling help under short notice. Thanks to Warren Lin for proofreading and tums. Lastly, I would like to thank my parents for grounding me then lifting me when the time was right.



Chapter 1

Introduction

Rendering effects of light scattering in atmospheric environments has long been a field of interest within the computer graphics community. Light scattering, caused by particles such as water or dust in the air, creates visual effects including visibility loss, glows around light sources, blurred surface radiance, shafts of light, and volumetric shadows. All these visual cues add to the realism of scenes with participating media. The computation of light transport in volumetric environments, however, presents a challenge for developing practical, interactive rendering methods for participating media in computer graphics.

Recent advances in consumer graphics hardware in both speed and flexibility, especially the ability to run code at the vertex and fragment level, have spurred the growth of methods for real-time rendering of scenes with participating media to meet this challenge. Much of the research for interactive rendering of participating media focuses on how to best exploit important hardware capabilities to solve complex scattering equations. Rectangular meshes storing illumination terms at lattice points can be used to render effects such as volumetric shadows and shafts of light [Dobashi et al., 2000a], although scene-dependent banding artifacts caused by undersampling may occur. Glows around point light sources can be rendered in real-time for scenes with fog and volumetric shadows [Biri et al., 2006], but banding artifacts occur when the viewer is very close to a source. Based on a particle system, a complete real-time rendering system for clouds [Harris and Lastra, 2001] allows the viewer to fly in and out for applications such as in

flight simulators. The precomputation of illumination required, however, is scene dependent and can widely vary. The advent of programmable graphics hardware has spawned numerous works exploiting its flexibility. These include methods for real-time atmospheric scattering effects [Hoffman and Preetham, 2003, O’Neil, 2005], but uniform density must be assumed. Also, interactive multiple scattering effects for inhomogeneous media [Hegeman et al., 2005] are demonstrated, although large amounts of scene-dependent precomputation is required. Complex scattering effects on surface radiance [Sun et al., 2005] have been made possible in real-time, but only scenes under the assumption of isotropic scattering is demonstrated.

This work deals specifically with anisotropic scattering within the context of rendering participating media for computer graphics applications. Thus, it is concerned with the variation of atmospheric conditions that result from the interaction of light and different particles in the atmosphere. Different anisotropic scattering conditions are seen in everyday life, including that of fog, mist, haze, or rain. Accurately modeling this type of scattering behavior adds to the visual realism desired in many interactive computer graphics applications.



1.1 Motivation

Rapid advances in consumer graphics hardware has allowed researchers to branch into interactively rendering volumetric environments, no longer restricted to only rendering of surfaces. As such, the overall goal of this work is to develop a practical rendering method for real-time graphics applications that considers increasingly complex scattering effects and dynamic behavior.

1.2 Contribution

We present, over the course of this thesis, a real-time analytic lighting model that is able to handle anisotropic scattering accurately as well as complex visual effects including glows around light sources and effects on surface radiance in scenes with fog, mist, or haze. Previous works either have ignored these complex visual effects or were unable to handle

anisotropic scattering accurately. Furthermore, we extend the lighting model to introduce a *dynamic scattering environment*, such that the scattering conditions of an atmosphere can be manipulated by the user in real-time. The visual appearance of scenes with scattering environments are no longer limited by pre-determined implementation choices. The user may dynamically control the scattering environment using simple parameters of density and forward scattering, generating a limitless variation of scattering conditions as well as receiving instant visual feedback. By using our lighting model, we are able to demonstrate the visual effects of forward scattering on glows and surface radiance. The entire implementation for our model using programmable graphics hardware is detailed and demonstrated in this work.

It focuses on the visual effects present in scenes with fog, mist, and haze, such that the viewer and light sources are both immersed within the medium. It is especially effective for near-field scenes in which the viewer and light sources are in close proximity. Specifically, our model considers effects of scattered radiance emanating directly from light sources, commonly termed *airlight*. The model captures effects for variable levels of forward scattering on glows around light sources, diffusing of specular highlights, and shadowed regions. The scattering behavior of the model may be dynamically controlled by just two simple parameters, the forward scattering parameter g and the extinction coefficient κ_t of the scattering medium. A precomputation step that is entirely independent of the physical parameters of the scene enables the dynamic behavior. Furthermore, the model can be easily integrated into an existing real-time rendering framework. More precisely, we develop an analytic model for single scattered radiance reaching the viewer for a point light source in a homogenous medium. The model may use any phase function of a single parameter θ without significant effect on run-time performance or loss in accuracy, thus making it appropriate as a general model for anisotropic scattering.

This work can largely be viewed as an application of the methodology used in [Sun et al., 2005] towards an alternative angle-formulated version of the single scattering light equations, whereby we obtain new results that enables efficient and accurate anisotropic scattering. In addition to dynamic lighting, geometry, and density of the medium of the previously mentioned work, we introduce dynamic forward scattering, which is pre-

sented as an extension to an initial analytic solution.

1.3 Limitations

To achieve interactive rates, the physics-based lighting model is restricted to *single scattering*, the assumption that light scatters only once in the medium. In environments with low albedo, single scattering is a reasonable approximation [Cerezo et al., 2005]. The demands typical in interactive applications such as dynamic viewing, lighting, scene geometry, and environment parameters require the model to assume isotropic point light sources, homogeneous media, and single scattering.

Although the use of scattering equations in this work is physically-based, visual appeal is the primary goal of the work, as opposed to strict, physical-simulation-like accuracy. This is not to say accuracy is not important, as it significantly affects the visual realism of scenes—the ability of the eye to perceive subtle visual cues in scattering environments is rather impressive. Such is the case with different levels of forward scattering (i.e. the difference between fog, mist, or haze)—though the average observer may have difficulty describing the visual difference among varying scattering conditions, he will invariably perceive a difference.

1.4 Organization

This thesis is organized as follows. Chapter 2 describes background in light scattering necessary for computer graphics. Chapter 3 describes related works regarding the interactive rendering of light scattering effects. Chapter 4 presents the proposed lighting model for accurate and dynamic anisotropic scattering as well as the hardware implementation. Chapter 5 describes the results of our method, including a qualitative description of anisotropic scattering effects. Finally, Chapter 6 concludes this work, describing limitations, possible areas of future work, and a summary.

Chapter 2

Light Scattering for Computer Graphics

When light travels through an environment, it interacts with particles of various sizes within the volume of space, a phenomenon known as light scattering. When the number of interactions is low, such as when there exists low density, small particles, or a short distance to be considered, light scattering can be ignored altogether. But in scenes with participating media, light scattering must be considered if visual realism is a desired result.

In this chapter, some background necessary for rendering light scattering effects is discussed. Firstly, a discussion of scattering distributions described by phase functions is presented. Phase functions will be more formally defined and elaborated on for use in describing various real-world scattering conditions. Secondly, the basic volumetric scattering processes of absorption, emission, scattering, and extinction will be introduced. These processes form the building blocks necessary for considering how to render complex scattering environments. In the same section, other scattering terminology will be defined to be used in the rest of this work. Lastly, a general model for rendering light scattering effects in computer graphics, the integral transport equation, will conclude the chapter.

Before entering the discussion of this chapter, basic knowledge of the radiometric quantity, *radiance*, is assumed. It is defined as the radiant power per unit area, per unit

solid angle, and largely determines the quantity used for computer graphics display.

2.1 Phase Functions and Anisotropic Scattering

The appearance of participating media is affected by both the size and density of particles, among other factors. The size of particles primarily determines light scattering distributions. That is, when there is a light-particle interaction, light scatters in different directions to varying degrees. In computer graphics, light scattering distributions are modeled using *phase functions*. The simplest model assumes *isotropic scattering*, in which light scatters equally in all directions. This, however, is insufficient for real-world conditions. Therefore, modeling *anisotropic scattering*, in which light scatters non-uniformly, is a challenge our work undertakes.

More formally, the phase function $p(\omega_i, \omega_o)$ describes the angular distribution of scattered radiation at a point. A dimensionless quantity, it determines how much light with incoming direction ω_i is scattered into outgoing direction ω_o . The normalization constraint allows the phase function to define the probability distribution for scattering in a particular direction, written as:

$$\frac{1}{4\pi} \int_{\Omega_{24\pi}} p(\omega_i, \omega_o) d\omega_o = 1 \quad (2.1)$$

Another property of phase functions is that they are reciprocal (i.e. $p(\omega_i, \omega_o) = p(\omega_o, \omega_i)$).

Because most phase functions for interactive computer graphics are symmetric about the incident direction, scattering distributions can be parameterized by the single angle θ between ω_i and ω_o , such that a common notation for phase functions is $p(\theta)$. Uniform scattering defined by a constant phase function, $p(\theta) = \frac{1}{4\pi}$, is termed *isotropic scattering*. Scattering defined by a non-constant phase function is termed *anisotropic scattering*. Descriptions of common phase functions used to model different scattering conditions in computer graphics compose the rest of this section. Certainly, more complex phase functions not mentioned in this chapter exist, mostly seen in non-interactive works. One may look to [Pharr and Humphreys, 2004, Premoze et al., 2003] for further reference.

2.1.1 Rayleigh Scattering

The Rayleigh scattering model is appropriate for describing the distribution found in the earth's atmosphere for very small particles with size less than the wavelength of light. In this type of scattering distribution, particles scatter in the forward and backward direction equally. Rayleigh scattering exhibits an important property: dependence on wavelength λ . The amount of scattering is proportional to $\frac{1}{\lambda^4}$. Therefore, light with shorter wavelengths scatters more frequently, causing it to bounce everywhere and yield the color of the blue sky. This is also why one observes colors of longer wavelengths such as orange and red during the sunset because all of the longer wavelengths of light have scattered away before reaching the observer. A simple phase function widely used for Rayleigh scattering is

$$p(\theta) = \frac{3}{16\pi}(1 + \cos^2 \theta) \quad (2.2)$$

One set of wavelength scattering coefficients (constants) used is $6.95 \times 10^{-6} m^{-1}$ for red light, $1.18 \times 10^{-5} m^{-1}$ for green light, and $2.44 \times 10^{-5} m^{-1}$ for blue light [Preetham, 2003]. Further discussion of these coefficients can be found in the referenced work.

2.1.2 Mie Scattering: Henyey-Greenstein Phase Function

Mie scattering describes the scattering distribution for large particles in atmospheres with fog, mist, or haze. It is largely wavelength independent and exhibits strong forward scattering. The *Henyey-Greenstein phase function* is commonly used to describe Mie scattering distributions due to its simple analytic form and easy-to-use parameter appropriate for computer graphics:

$$p(\theta) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g \cos \theta)^{3/2}}, \quad (2.3)$$

where the forward parameter $g \in (-0, 1)$ controls the amount of forward scattering. The exact meaning of g is the average value of the phase function and cosine of the phase angle over a hemisphere of directions:

$$g = \frac{1}{4\pi} \int_{4\pi} p(\omega_o, \omega_i) (\omega_o \cdot \omega_i) d\omega_i. \quad (2.4)$$

Values $g > 0$ correspond to forward scattering, $g = 0$ corresponds to isotropic scattering, and $g < 0$ correspond to backward scattering. Positive values for g can describe most

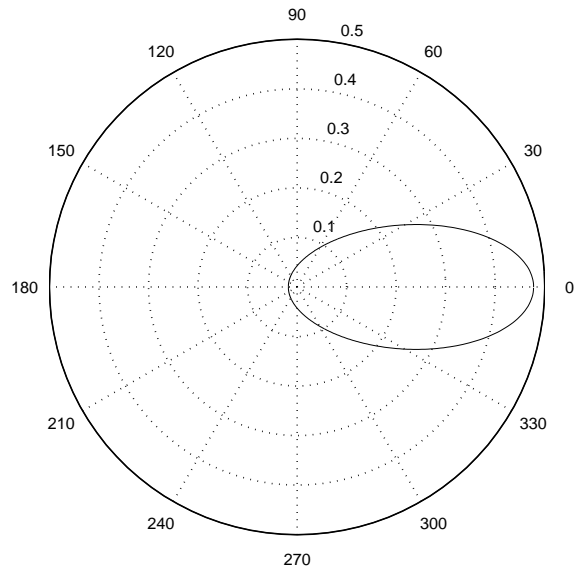


Figure 2.1: *Plot of normalized Henyey-Greenstein phase function for $g = 0.5$.*

weather conditions [Narasimhan and Nayar, 2003]. As an example, a polar plot of the Henyey-Greenstein phase function for $g = 0.5$ is shown in Figure 2.1

2.2 Light Scattering Terminology

This section lays out the groundwork and definitions used throughout this work.

2.2.1 Absorption, Emission, Scattering, and Extinction

The appearance of participating media is governed by the interaction of particles and light. Interactions include absorption, emission, scattering, and extinction.

Absorption

Absorption is the process when a particle absorbs some part of incident radiance. The change of radiance L due to absorption at a point x in direction ω is given by

$$\frac{dL(x, \omega)}{dx} = -\kappa_a L(x, \omega), \quad (2.5)$$

where κ_a is the *absorption coefficient*.

Emission

Emission is the process when a particle converts some form of energy to emitted radiance. The change of radiance L due to emission at a point x in direction ω is given by

$$\frac{dL(x, \omega)}{dx} = L_e(x, \omega). \quad (2.6)$$

Out-Scattering

Scattering is the process when a particle redirects (absorbs and re-emits) incident radiance to a new direction. *Out-scattering* refers to decreased radiance along a solid angle due to scattering outward into new directions. The change of radiance L due to out-scattering at a point x in direction ω is given by

$$\frac{dL(x, \omega)}{dx} = -\kappa_s L(x, \omega), \quad (2.7)$$

where κ_s is the scattering coefficient.

In-scattering

In-scattering refers to increased radiance along a solid angle due to scattering from other directions. Therefore, it must consider all incident radiance over the hemisphere of directions around a point x . The change of radiance L due to in-scattering at x in direction ω is given by

$$\frac{dL(x, \omega)}{dx} = \kappa_s \int_{4\pi} p(\omega_i, \omega) L(x, \omega_i) d\omega_i. \quad (2.8)$$

Scattering coefficient κ_s determines how much of incident radiance actually scatters at point x , whereas the phase function $p(\omega_i, \omega)$ determines how much of the scattered light redirects towards direction ω .

Extinction

Extinction, also known as attenuation, refers to the combined effect of absorption and out-scattering. The *extinction coefficient* is the sum $\kappa_t = \kappa_a + \kappa_s$. So the change of radiance L

due to extinction at a point x in direction ω is given by

$$\frac{dL(x, \omega)}{dx} = -\kappa_t L(x, \omega). \quad (2.9)$$

More Discussion

Alternatively, the coefficients κ_a , κ_s , and κ_t can be viewed as probability densities per unit distance that radiance is absorbed, scattered, or both. This means the coefficients may be any positive value, not necessarily between zero and one. These coefficients are properties of the density and mass of particles in the medium and in many cases, are spatially varying. In all encountered cases, these coefficients are a function of position, though strictly speaking, they could vary with direction. In purposes for computer graphics, it is sometimes convenient to assume uniform coefficients throughout a medium. *Homogenous media* are volumes with constant κ_t . *Inhomogenous media* are volumes with spatially varying κ_t .

2.2.2 Transmittance and Optical Thickness

If we consider a line segment from x_0 to x , a solution to Equation 2.9 can be found, which is

$$L(x, \omega) = L(x_0, \omega) e^{-\int_{x_0}^x \kappa_t(u) du} = L(x_0, \omega) \tau(x_0, x). \quad (2.10)$$

The solution describes the radiance at a new point x given an original point x_0 , with loss of radiance due to out-scattering and absorption.

Transmittance

Transmittance $\tau(x_0, x)$ describes the fraction of radiance that is transmitted from point x_0 to x along some path, where absorption and out-scattering account for the radiance reduction. As seen in the above equation, it is given by

$$\tau(x_0, x) = e^{-\int_{x_0}^x \kappa_t(u) du}. \quad (2.11)$$

An useful case is the assumption of a homogenous medium, where κ_t is constant. Transmittance can easily be evaluated, yielding *Beer's Law*:

$$\tau(x_0, x) = e^{-\kappa_t d}, \quad (2.12)$$

where d is the distance from x_0 to x .

Optical Thickness

The exponent term $-\int_{x_0}^x \kappa_t(u)du$ is known as *optical thickness*. Other names include *optical depth* or *optical length*. On a side note, the symbols for optical thickness and transmittance are sometimes identical in different works, though they represent different quantities. There is great potential for confusion since transmittance and optical thickness are closely related, though distinct, quantities. This is noted so as to avoid confusion on the part of the reader if cross-referencing.

2.3 Integral Transport Equation

The combination of all the processes described in Section 2.2.1 form the differential equation for light transport:

$$\frac{dL(x, \omega)}{dx} = \kappa_a L_e(x, \omega) + \left[\kappa_s \int_{4\pi} p(\omega_i, \omega) L(x, \omega_i) d\omega_i \right] - \kappa_a L(x, \omega) - \kappa_s L(x, \omega). \quad (2.13)$$

Integrating both sides of the equation for a line segment from x_0 to x yields the result:

$$L(x, \omega) = \int_{x_0}^x \tau(u, x) \kappa_a(u) L_e(u, \omega) du + \left[\int_{x_0}^x \tau(u, x) \kappa_s(u) L_i(u, \omega) du \right] + \tau(x_0, x) L(x_0, \omega), \quad (2.14)$$

where $L_i(u, \omega)$ is the in-scattered radiance at a point u . It is given by

$$L_i(u, \omega) = \int_{4\pi} p(\omega_i, \omega) L(u, \omega_i) d\omega_i. \quad (2.15)$$

The terms of Equation 2.14 correspond in order to the terms of the original differential form in Equation 2.13, with the exception that the last two terms of the differential form are combined into one before integration. This integral equation appears in various works in the literature on light transport and is mentioned here since the derivation from the differential form follows logically. In this work, however, we will refer more often to an alternative but equivalent form known as the *integral transport equation*, which is

$$L(x, \omega) = \tau(x_0, x) L(x_0, \omega) + \int_{x_0}^x \tau(u, x) \kappa_t(u) J(u, \omega) du, \quad (2.16)$$

where $J(x, \omega)$ is the *source radiance* that describes the local production of radiance. $J(x, \omega)$, which includes both self-emission and in-scattering, is given by

$$J(x, \omega) = (1 - \Omega(x))L_e(x, \omega) + \Omega(x) \int_{4\pi} p(\omega_i, \omega)L(x, \omega_i)d\omega_i, \quad (2.17)$$

where $\Omega = \frac{\kappa_s}{\kappa_t}$ is the *albedo* of the medium. The integral transport equation can be easier to follow, especially when the source radiance term $J(u, \omega)$ simplifies easily under the common case when albedo is close to one. The relationship between the equivalent integral equations is shown below as a movement from Equation 2.16 to Equation 2.14:

$$\begin{aligned} L(x, \omega) &= \tau(x_0, x)L(x_0, \omega) + \int_{x_0}^x \tau(u, x)\kappa_t(u)J(u, \omega)du \\ &= \tau(x_0, x)L(x_0, \omega) + \\ &\quad \int_{x_0}^x \tau(u, x)\kappa_t(u) \left[(1 - \Omega(u))L_e(u, \omega) + \Omega(u) \int_{4\pi} p(\omega_i, \omega)L(u, \omega_i)d\omega_i \right] du \\ &= \tau(x_0, x)L(x_0, \omega) + \\ &\quad \int_{x_0}^x \tau(u, x)\kappa_t(u) \left[\left(\frac{\kappa_a(u)}{\kappa_t(u)} \right) L_e(u, \omega) + \frac{\kappa_s(u)}{\kappa_t(u)} \int_{4\pi} p(\omega_i, \omega)L(u, \omega_i)d\omega_i \right] du \\ &= \tau(x_0, x)L(x_0, \omega) + \int_{x_0}^x \tau(u, x)\kappa_a(u)L_e(u, \omega)du + \left[\int_{x_0}^x \tau(u, x)\kappa_s(u)L_i(u, \omega)du \right] \end{aligned}$$

The challenge for rendering light scattering effects in computer graphics is developing models to solve the integral transport equation.

2.3.1 Simplifying Assumptions

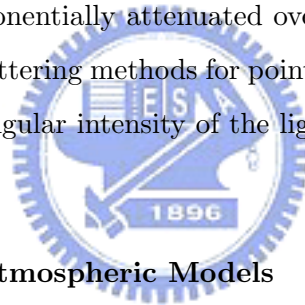
Solving the integral transport equation (Equation 2.16) without making simplifying assumptions is too complex for any practical rendering method. For atmospheric scattering, useful assumptions include homogeneous media, simple geometry, and large albedo close to one. For interactive methods, it is common to assume *single scattering*, which says that light scatters only once in the scene before reaching the viewer. There are widely varying models for multiple scattering, since the complexity for such models grows tremendously. Some models for multiple scattering will be presented in the Related Works of Chapter 3. For now, models for single scattering will be presented that exist throughout many works.

Point Light Source

The model for scattering from a point light source [Nishita et al., 1987] in a homogeneous medium is widely used in many rendering methods due to the simple form for transmittance in Equation 2.12. Under these assumptions, single-scattered radiance is given by

$$L_a = \int_0^{d_{vp}} \kappa_t p(\alpha) \cdot \frac{I_0 \cdot e^{-\kappa_t d}}{d^2} \cdot e^{-\kappa_t x} dx \quad (2.18)$$

The integral ranges from zero to the distance between viewer and nearest surface point, d_{vp} . The intensity of light reaching a scattering point along the view ray is $\frac{I_0 \cdot e^{-\kappa_t d}}{d^2}$, since I_0 is the initial intensity of the point source, intensity is exponentially attenuated over the distance d , and intensity drops off by the inverse square law of distance. The intensity of light that reaches the view ray scatters by amount given by the extinction coefficient κ_t . The scattered light is redirected towards the viewer with frequency $p(\alpha)$. Finally, the remaining intensity is exponentially attenuated over distance x . A diagram is provided in Figure 2.2. Most single scattering methods for point light sources are slight variations of the above equation, such as angular intensity of the light source or visibility terms to consider volumetric shadows.



Single Scattering for Atmospheric Models

Sky and terrain rendering is a large area of research, whose origin largely traces to the single scattering model of [Nishita et al., 1993]. In this work, Rayleigh scattering with wavelength dependence is used, and the amount of scattering has exponential dependence on the height in the atmosphere, as an approximation to measurement analysis. Specifically, the atmospheric model for single-scattered radiance is

$$I_v = I_s(\lambda) K(\lambda) p(\alpha) \int_{P_a}^{P_b} e^{-h/H_0} e^{-t(P P_c, \lambda) - t(P P_c, \lambda)} ds, \quad (2.19)$$

where optical depth t is

$$t(P_a P_b, \lambda) = K(\lambda) \int_{P_a}^{P_b} e^{-h/H_0} ds. \quad (2.20)$$

The equation integrates in-scattered radiance along the view ray from the nearest point in the atmosphere P_a to the farthest point in the atmosphere P_b , which may be the nearest

surface. In-scattered radiance enters the atmosphere at a point P_c and scatters at point P , as shown in Figure 2.3. Rayleigh scattering constant $K(\lambda)$ has dependence on wavelength λ , and the scattering coefficients e^{-h/H_0} are exponentially dependent upon the height h in the atmosphere. Constant H_0 is associated with the height of the top of the atmosphere. Intensity from source $I_s(\lambda)$ may consider radiation other than white light. Phase function $p(\alpha)$ depends on phase angle α , the angle between the directional source direction and the view ray. Many works build upon this model and include more complex effects such as halos, coronas, skylight, and sophisticated, precise measurements for atmospheric data [Nishita et al., 1996, Preetham et al., 1999, Riley et al., 2004].



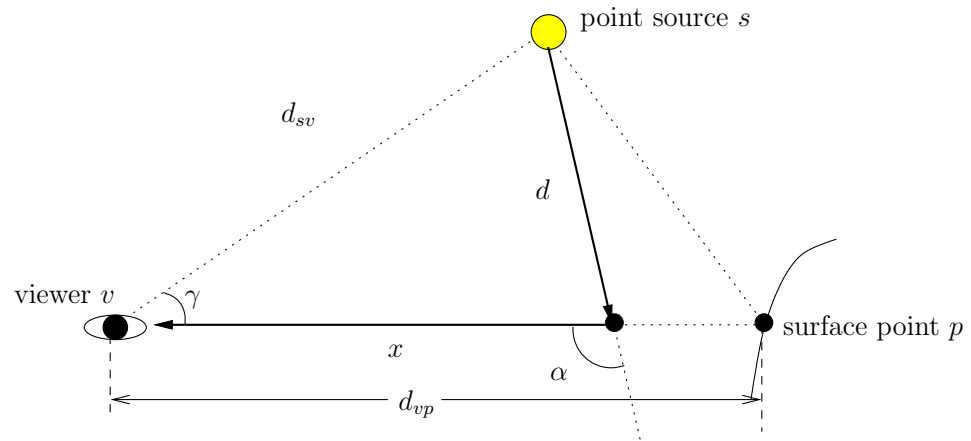


Figure 2.2: *Diagram of single scattering integral for point light source*

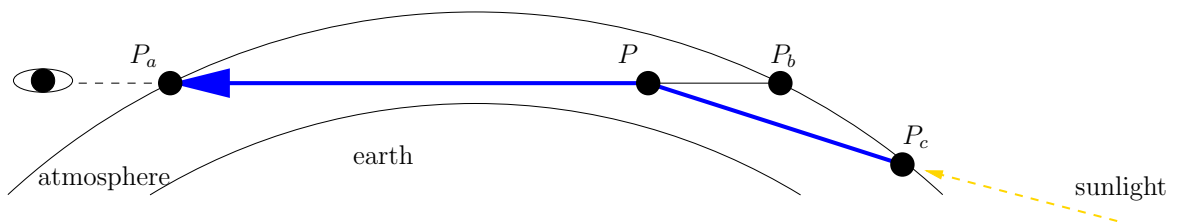
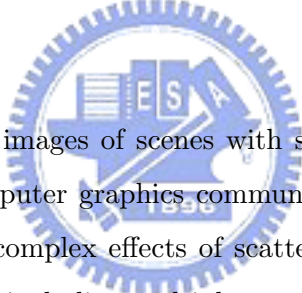


Figure 2.3: *Diagram for single scattering integral of atmospheric model*

Chapter 3

Related Work: Interactive Rendering of Light Scattering Effects



Creating realistic images of scenes with scattering media has long been an area of research within the computer graphics community due to the enhancement it lends to visual realism. The most complex effects of scattering to-date are captured using Monte Carlo ray tracing methods, including multiple scattering effects and non-homogeneous media [Lafortune and Willems, 1996, Jensen and Christensen, 1998, Cerezo et al., 2005]. These methods, however, lack practicality for interactive applications because of the notoriously expensive procedures they require. Guided by the value to interactive applications, we describe in this chapter, therefore, representative methods for rendering of participating media towards interactive rates.

A thorough survey of works for rendering the effects of light scattering can be found in [Cerezo et al., 2005]. There are surveys with a narrower focus in atmospheric rendering [Preetham, 2003, Sloup, 2002]. With a focus on rendering techniques toward interactive rates, the first subset of techniques for displaying scattering effects are characterized by solving the integral transport equation (Equation 2.16) using numerical methods, which we discuss in Section 3.1. Another set of works have explored the use of analytic techniques

for rendering participating media, which we discuss in Section 3.2. Before heading into a full-blown discussion on these sets of techniques, we first mention a small but significant observation that has guided this study: the proliferation of relevant works in recent years is owed not in small part to the acceleration and increased flexibility of commodity graphics hardware. The evolution of research in the field, thus, largely reflects these advances in hardware. This theme, though not explicitly mentioned, should run throughout the following chapter.

3.1 Numerical Methods

Unlike stochastic methods that may take hours to render a single image, many numerical solutions have been developed to accelerate the rendering process, though many still have limitations for real-time rendering applications. A common technique for numerical methods is to use two passes, the first for computing illumination within the medium and the second for rendering the medium.

3.1.1 Shafts of Light and Volumetric Shadows

Shafts of light result from the non-shadow regions of objects in participating media, whereas volumetric shadows result from the shadowed regions of objects in participating media. Illuminated shafts of light were rendered interactively using *virtual planes* [Dobashi et al., 2000b], storing incident illumination on rectangular meshes lined up perpendicular to the view ray at regular intervals. The lighting equation used is Nishita’s single scattering integral for point light sources (Equation 2.18) with a visibility term added, seen here:

$$I_{eye} = I_{obj}\beta(T) + \int_0^T F(\alpha)H(t)I_p(t)\beta(t)dt. \quad (3.1)$$

The notation is as follows: $\beta(t)$ is the transmittance over distance t , T is the distance from the eye to the nearest object, $F(\alpha)$ is the phase function with phase angle α , $H(t)$ is the visibility function (binary) of scattering point P , and $I_p(t)$ is the intensity of light reaching P .

The key hardware acceleration was the storage of the $F(\alpha)I_p(t)\beta(t)$ term of the integrand as an illumination pass onto "virtual planes", lattice meshes that are lined up

perpendicular to the view ray at regular intervals. Storing the $F(\alpha)I_p(t)\beta(t)$ term requires projecting a light map onto each "virtual plane" to calculate part of $I_p(t)$. Ignoring the visibility term $H(t)$ for a moment, the medium can then be rendered by

- sampling the integrand at lattice points of every virtual plane
- rendering the virtual planes with an additive blending function from back to front (volume rendering)

Only lattice points of the "virtual planes" are used to sample the integrand to increase efficiency, and Gouraud interpolation is used to evaluate the term at any arbitrary point. The visibility term of the integrand, $H(t)$, is applied by using the shadow map algorithm when rendering the virtual planes. That is, points on the virtual plane with depth value greater than the corresponding depth in the shadow map will have contribution zero when rendered. Banding problems caused by aliasing due to finite sampling limitations may occur in some situations. Increasing the amount of sampling is the straightforward solution; however, this quantity is highly dependent upon each scene and varies widely.

3.1.2 Atmospheric Effects

The method of [O'Neil, 2005] can render real-time atmospheric scenes either viewing the earth from space or the ground (like a horizon scene). Single-scattering approximations are adequate for these general atmospheric effects. Thus, this work essentially uniformly samples the atmospheric model for single scattering of Equation 2.19 in the vertex or fragment shader using programmable graphics hardware. To evaluate optical depth terms, a polynomial best fit function was found using mathematical analysis software. Precomputed tables in texture memory may be used as well to compute optical depth terms, but polynomial functions were used to avoid texture lookups for hardware without texture lookup capability in the vertex shader. Furthermore, high dynamic range rendering is demonstrated for sky and terrain rendering. Since sampling is performed in the vertex or fragment shader, the cost of using a loop for sampling can dramatically increase if the scene requires greater accuracy. Otherwise, the method is effective for real-time rendering of atmospheric scenes with non-uniform scattering effects.

3.1.3 Multiple Scattering

Inhomogeneous media

By investigating the effects of multiple-scattering, Premoze et. al developed the *point spread function* (PSF), which measures the spatial spreading of incident radiance in analytic form [Premoze et al., 2004]. Based upon an analytic multiple scattering term derived from the point spread function, Hegeman et. al developed a lighting model for an inhomogeneous medium with multiple scattering effects at interactive rates [Hegeman et al., 2005], but illumination and medium parameters must be fixed. Other restrictions include the absence of volumetric shadows for objects within the medium. The lighting model uses both analytic and numerical techniques. Descriptions for the development of the point spread function, analytic multiple scattering term, and lighting model with multiple scattering effects follow.

Using measurement analysis, Premoze et. al developed a point spread function that measures the spatial spreading of incident radiance for a given medium in an analytic form [Premoze et al., 2004]. Spatial spreading is described as the effect of a light beam's broadening as it propagates through a medium. The point spread function has two parameters, the amount of out-scattering l for a path and the distance S from the origin of the beam, and is given by

$$w^2(l, S)d = \frac{\langle \theta^2 \rangle l S^2}{24(1 + \langle \theta^2 \rangle (a/b)l^2/12)}, \quad (3.2)$$

where a and b are the absorption and scattering coefficients, respectively, and $\langle \theta^2 \rangle$ is the mean square scattering angle. The work of [Premoze 2003] introduced the notion of a *Most Probable Path* (MPP), such that only paths close to MPPs contribute to a multiple scattering contribution. Also, an alternative method for modeling light scattering is introduced, known as the *path integral*. The path integral formulation describes an integral over an infinite number of possible paths, for which only out-scattering need be accounted. Using both the path integral formulation and point spread function, an analytic term for multiple scattering contribution in a strongly forward-scattering inhomogeneous medium is developed. It is based upon the assumption that only paths close to the MPP are added to the contribution. Additionally, curved MPPs are approximated by two linear segments.

The result for the multiple-scattering contribution is

$$L_{MS}(x, \omega) = \int_l dl \times e^{-(c/b)l(MPP)_l} e^{-A_c(MPP)_l} \times P_{MS}(l, \omega - \omega') \int_{plane} gauss(x', w(l)) L_0(x', \omega') dx' \quad (3.3)$$

The outer integral is an integral over MPPs, each with scattering parameter l . The inner integration is the gaussian-weighted incident radiance in the plane perpendicular to the light direction. c is the extinction coefficient. A_c is a curvature term. P_{MS} is a multiple scattering phase function.

Based upon the above multiple scattering term, Hegeman et. al developed a lighting model for an inhomogeneous medium with multiple scattering effects at interactive rates [Hegeman et al., 2005]. The scene is restricted to static directional light and a static volume. Strongly forward-scattering with mean square scattering angle $< \theta >^2$ is reasonably assumed. The main contribution of this work was the implementation of the multiple scattering term using graphics hardware and data structures for acceleration. To accomplish this task, a couple of large data structures are created before rendering. A 3D precomputed light volume data structure stored an approximation for the total radiance arriving at each voxel, essentially computing the scattering parameter l of Equation 3.3. A second data structure for accelerated performance, called the *LightImage*, is a visibility map from the perspective of the volume. A mipmap pyramid is constructed where each level blurs over a different width w . The *LightImage* essentially computes the inner integral of Equation 3.3. With these data structures, a technique of volume rendering can be applied to render the medium, effectively sampling the integral for paths close to the view ray. A series of view-aligned planes, whose vertices are assigned texture coordinates corresponding to their intersection with the light volume at regular intervals, are created for each frame and rendered in back-to-front order with an additive blending function.

Cloud Rendering

Based on a two-pass rendering technique employed by [Dobashi et al., 2000a] for clouds under a single scattering assumption, a complete system for real-time cloud rendering [Harris and Lastra, 2001] was developed that used a particle system to model clouds and

precomputed an isotropic multiple forward scattering term in dynamically-generated impostors during an initial illumination pass. Anisotropic single scattering was accounted for during the second visualization pass. To amortize the cost over many frames, impostors are required to achieve real-time rates. Evaluation of the differential form (Equation 2.13) for the integral light transport equation is done as a recursive operation, where each step attenuates the old radiance and adds the source radiance contribution of the present particle from back to front. The particles are rendered using splatting, such that the particles are sorted with respect to the light and rendered back-to-front with a recursive blending equation. The method is specifically tailored to scattering seen in clouds. Another limiting factor is the precomputation required, which is dependent upon the scene illumination and scattering properties.

3.2 Analytic Methods

Analytic methods significantly improve the performance for rendering of scenes with participating media. Their major drawback lies in the increased number of assumptions and simplifications made for the sake of performance.

3.2.1 Effects for Point Light Sources

Assuming point light sources simplifies computation but can be effective for scenes in which the source is inside the participating medium, creating effects of glows around light sources. Complex effects such as multiple scattering for glows and scattering effects on surface radiance can be realized.

Multiple Scattering

The work of [Narasimhan and Nayar, 2003] modeled multiple scattering from an isotropic point light source using an analytic expression termed the *atmospheric point spread function*. Assumptions include homogeneous media, infinite extent, and distant light sources. Therefore, it neglects effects on surface radiance. Using an expansion of the Henyey-Greenstein phase function as a series Legendre polynomials, a series solution to the integral transport

equation (Equation 2.16) is derived, assuming homogeneous media and distant point light sources. The work shows that the shape of the point spread function largely depends on the atmospheric conditions, the source intensity, and the depth of the source. An analysis of glows under different weather conditions is described, including rendered images.

Volumetric Shadows

In [Biri et al., 2006], the authors developed an analytic expression for the single scattering integral for point light sources in a homogeneous medium using a polynomial approximation that can be implemented on graphics hardware. The polynomial approximation is obtained by first using an angular reformation of the single scattering integral for point light sources [Lecocq et al., 2000], which is described in a later chapter (Equation 4.3). Then the integral is approximated as a polynomial of a single angle with degree four. The constants of the polynomial are determined by the type of scattering chosen. Unfortunately, inaccurate glows appear when the viewer is near the source. Also, no effects on surface radiance are considered. An algorithm for incorporating volumetric shadows is also introduced. The basic concept is that the scattering contribution of non-shadowed regions may be rendered by subtracting the contribution of shadowed regions, whose equivalent is rendering the shadow volume using an analytic scattering contribution. There is an exception—shadow polygons which are themselves in shadow should not be rendered. The algorithm essentially constructs a standard shadow volume, sorts shadow polygons according to their estimated depths, and then renders the shadow polygons using a positive scattering contribution for back-facing shadow polygons and a negative scattering contribution for front-facing shadow polygons. Limitations occur when the estimated depths of large polygons have errors.

Scattering Effects on Surface Shading and Environment Map Lighting

The work of [Sun et al., 2005] developed a real-time analytic single scattering model for point light sources that can render visual effects on surface shading and handle complex environment map lighting using programmable graphics hardware. Significant effects such as glows around light sources, spreading of specular highlight on objects, and brightening of shadows were realized in real-time fog scenes. The effects of anisotropic scattering, though

considered in the model, were not demonstrated. The inclusion of anisotropic scattering in the model induces a loss in both performance and accuracy.

Visual results of glows and scattering effects on surface shading were realized in real-time, a difficult task due to the complexity of volumetric scattering for all surface points. Analytic evaluation of the single scattering integral for point light sources [Nishita et al., 1987] was made possible by changing the variables of integration and deriving a form that could be largely precomputed into a table. Complex integrals are expressed as 2D functions that can be precomputed into tables and placed in texture memory. The functions then can be accurately evaluated at run-time using texture lookups with linear interpolation. With this technique, an analytic model for scattering contribution to surface radiance is developed, under the assumption that no objects are in the path of scattered light.

Complex *brdf* and environment map lighting under foggy conditions is also presented, enabling environment maps to include effects of glows around light sources as well as enabling precomputed radiance transfer methods to incorporate scattering effects. By fitting empirical data, the work develops an analytic point spread function of a single angular parameter that can be convolved with an environment map to produce a new environment map which exhibits glows around light sources. This technique can be applied on the environment lighting used in precomputed radiance transfer methods as well.

3.2.2 Sky and Terrain Rendering

The rendering of scenes with both sky and terrain requires models for sky and aerial perspective. Sky models only describe the illumination of skylight, whereas aerial perspective models describe the in-scattering of skylight and out-scattering that occurs before it reaches the viewer. Consequently, aerial perspective models must consider the position of the viewer and usually requires larger amounts of computation. Significantly more efficient and practical than their numerically-simulated counterparts, complex analytic models that fit measured atmospheric data have been developed that exhibit angular dependent and multiple scattering effects [Preetham et al., 1999, Riley et al., 2004]. The utilization of graphics hardware is used in some cases; however, these methods are still far from interac-

tive.

The work of [Hoffman and Preetham, 2003] can render skies using programmable graphics hardware in the vertex or fragment shader in real-time. A simple, analytic form for scattering from a directional light source is made possible when a constant density atmosphere is assumed, thus assuming constant scattering as well. Under this assumption, the scattering coefficient can be expressed as Rayleigh and Mie scattering constants with dependence on wavelength. The result for atmospheric scattering is the lighting equation:


$$L(s, \theta) = L_0 e^{-(\kappa_{s,R} + \kappa_{s,M})s} + \frac{\kappa_{s,R} p_R(\theta) + \kappa_{s,M} p_M(\theta)}{\kappa_{s,R} + \kappa_{s,M}} E_{sun} (1 - e^{-(\kappa_{s,R} + \kappa_{s,M})}), \quad (3.4)$$

where $\kappa_{s,R}$ and $\kappa_{s,M}$ are the Rayleigh and Mie scattering coefficients, respectively. $p_R(\theta)$ and $p_M(\theta)$ are the Rayleigh and Mie phase functions, respectively (Equations 2.2 and 4.1). The scattering coefficient (κ_s) is split into the sum of its parts ($\kappa_s = \kappa_{s,R} + \kappa_{s,M}$). Rendering a sky thus equates to rendering a tessellated sky dome with implementation of Equation 3.4 in the vertex or fragment shader.



Chapter 4

Analytic Lighting Model for Accurate and Dynamic Anisotropic Scattering



Reiterating earlier goals, our objective is to develop a practical rendering method for real-time graphics applications that considers increasingly complex scattering effects and dynamic behavior. Under this premise, we develop a lighting model for atmospheric environments such as fog, mist, or haze that allows users to create and adjust scattering conditions interactively using simple parameters. This type of interactive capability enables what is heretofore referred to as a *dynamic scattering environment*, where the medium's level of forward scattering and density may be changed in real-time.

For most atmospheric conditions, a phase function $p(\theta)$ symmetric about the incident direction is sufficient to describe a scattering distribution [Cerezo et al., 2005]. The Henyey-Greenstein phase function [Ishimaru, 1997] is widely used in computer graphics for its relative simplicity and effectiveness for common atmospheric environments. Its normalized form is repeated here for easy reference:

$$p(\theta) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g \cos \theta)^{3/2}}, \quad (4.1)$$

where the forward scattering parameter $g \in (0, 1)$ controls the amount of forward scattering.

Values $g > 0$ correspond to forward scattering, $g = 0$ corresponds to isotropic scattering, and $g < 0$ correspond to backward scattering. Positive values for g can describe most weather conditions [Narasimhan and Nayar, 2003]. By varying forward scattering parameter g , different scattering conditions can be modeled. The presented lighting model can demonstrate complex visual affects of glows around light sources and effects on surface radiance under varying scattering conditions. These findings are described in a later chapter describing results in Section 5.1.

In this chapter, we present our analytic lighting model with an extension for a dynamic scattering environment that all may be implemented on programmable graphics hardware in real-time. The basic underlying strategy for analytic evaluation of scattering integrals is the use of precomputed two-dimensional tables. By converting the scattering integrals into forms that may be expressed as functions of two parameters, the functions may then be precomputed as two-dimensional tables and placed in the texture memory of graphics hardware. Using programmable graphics hardware, the functions may be evaluated with a simple texture lookup in the vertex or fragment shader.

This chapter is divided into four sections: an introduction section, a detailed description of the proposed analytic lighting model, an extension to the analytic model for dynamic anisotropic scattering, and lastly, a description of the hardware implementation.

4.1 Introduction

Let us consider the paths of light transport for single scattering depicted in Figure 4.1. Light rays may (a) directly transmit through the medium, leading to attenuation; (b) scatter once from source to viewer, leading to glows around sources [Narasimhan and Nayar, 2003]; or (c) scatter once from source to surface point, leading to diffusing of reflected surface radiance and brightening of shadows [Sun et al., 2005]. The aforementioned paths and their visual effects can be captured by the sum of reflected surface radiance L_p with attenuation and airlight from source to viewer L_a [Sun et al., 2005]:

$$L = e^{-\kappa_t d_{vp}} L_p + L_a, \quad (4.2)$$

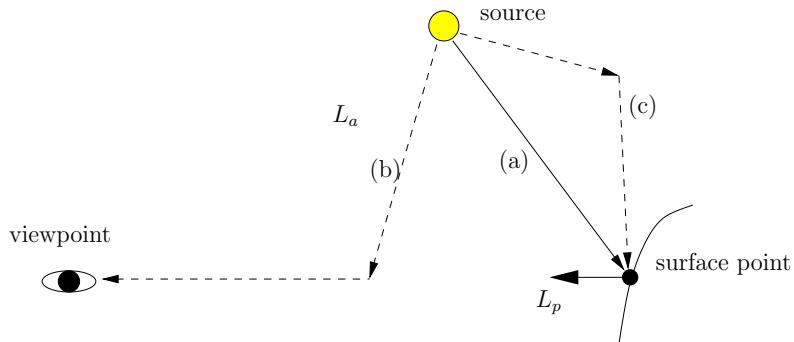


Figure 4.1: Diagram illustrating light paths of (a) direct transmission, (b) airlight from source to viewer, and (c) airlight from source to surface point. Path (b) contributes to airlight from source to viewer L_a . Paths (a) and (c) contribute to reflected surface radiance L_p .

where κ_t is the extinction coefficient of the participating medium and d_{vp} is the distance between the viewpoint and surface point. Subsequent sections develop a solution to this lighting equation.

4.2 Analytic Lighting Model

In this section, we develop analytic models to compute airlight L_a , which is responsible for glows around light sources, in Section 4.2.1 and surface radiance L_p , which is affected by airlight from source to surfaces, in Section 4.2.2. Lastly, a summary of the analytic lighting model is provided in Section 4.2.3

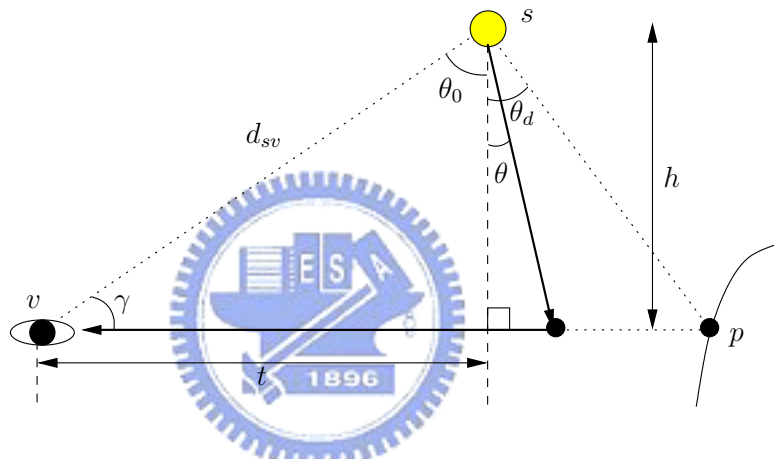
4.2.1 Angle-Formulated Airlight Model

The contribution of airlight depicted as path b in Figure 4.1 is given by the classic integral along the viewing direction for single-scattered airlight from an isotropic point light source [Nishita et al., 1987], which is expressed in terms of the distance along the ray. To reduce the number of physical parameters in the integral, we introduce the angle reformulation of the single scattering integral for airlight from a point light source [Lecocq et al., 2000]:

$$L_a(\gamma, d_{sv}, d_{vp}, \kappa_t) = \frac{\kappa_t I_0 e^{-\kappa_t t}}{h} \int_{\theta_0}^{\theta_d} e^{-\kappa_t h \left(\frac{\sin \theta + 1}{\cos \theta} \right)} p\left(\theta + \frac{\pi}{2}\right) d\theta. \quad (4.3)$$

Table 4.1: *Notation and formulae for angle-formulated airlight integral.*

θ	angle between source emissive direction and perpendicular bisector to the viewing direction
γ	angle between view ray and source
d_{sv}	distance between source and viewpoint
d_{vp}	distance between viewpoint and surface point
d_{sp}	distance between source and surface point
t	$t = d_{sv} \cos \gamma$
h	$h = d_{sv} \sin \gamma$
θ_0	$\theta_0 = \gamma - \frac{\pi}{2}$
θ_d	$\theta_d = \arctan (d_{vp} - t)/h $
I_0	point light source intensity

Figure 4.2: *Diagram for angular reformulation of the airlight integral.*

The integral is expressed in terms of the angle θ between the light source emissive direction and the perpendicular bisector to the viewing direction with respect to s , as depicted in Figure 4.2. θ_d and θ_0 are the bounds of integration. See [Biri et al., 2006] for a derivation of the angle reformulation. Refer to Table 4.1 for the rest of the notation. We refer to Equation 4.3 in the remainder of the paper as the *angle-formulated airlight integral*.

Analytic solution to Airlight L_a

To solve Equation 4.3, it is factorized into an analytic expression dependent on the physical parameters of the scene and a tabulated function independent of the physical parameters.

Let

$$C_0(\gamma, d_{sv}, \kappa_t) = \kappa_t I_0 e^{-\kappa_t t} / h, \quad (4.4)$$

$$C_1(\gamma, d_{sv}, \kappa_t) = \kappa_t h. \quad (4.5)$$

Then

$$L_a = C_0 \int_{\theta_0}^{\theta_d} e^{-C_1 \left(\frac{\sin \theta + 1}{\cos \theta} \right)} p\left(\theta + \frac{\pi}{2}\right) d\theta.$$

The angle-formulated airlight integral now is factorized into an analytic expression C_0 and an integral that may be expressed as a function of two variables. The result for an analytic solution to Equation 4.3 is

$$L_a = C_0 [H(C_1, \theta_d) - H(C_1, \theta_0)]. \quad (4.6)$$

where

$$H(u, v) = \int_{-\frac{\pi}{2}}^v e^{-u \left(\frac{\sin \theta + 1}{\cos \theta} \right)} p\left(\theta + \frac{\pi}{2}\right) d\theta. \quad (4.7)$$

As a function independent of the physical parameters of the scene, $H(u, v)$ can be numerically integrated during the precomputation stage into a table and stored into texture memory. The plot of $H(u, v)$ in the isotropic scattering case $p(\theta) = \frac{1}{4\pi}$ is shown in Figure 4.3, demonstrating that the function is smooth. Thus, the tabulated function $H(u, v)$ in texture memory can be accessed during run-time using linear interpolation for evaluation of the integral. It is worth mentioning that any phase function $p(\theta)$ may be used in our lighting model without significantly affecting run-time performance or accuracy, thus making it general for any type of anisotropic scattering. Airlight from a point source may be computed with just evaluation of analytic functions and two texture lookups in the vertex or fragment shader using programmable graphics hardware. The choice of $p(\theta)$ does not affect run-time performance since $H(u, v)$ is constructed in a precomputation stage. The special case of no objects along the view ray ($d_{vp} = \infty, \theta_d = \frac{\pi}{2}$) is important for the next

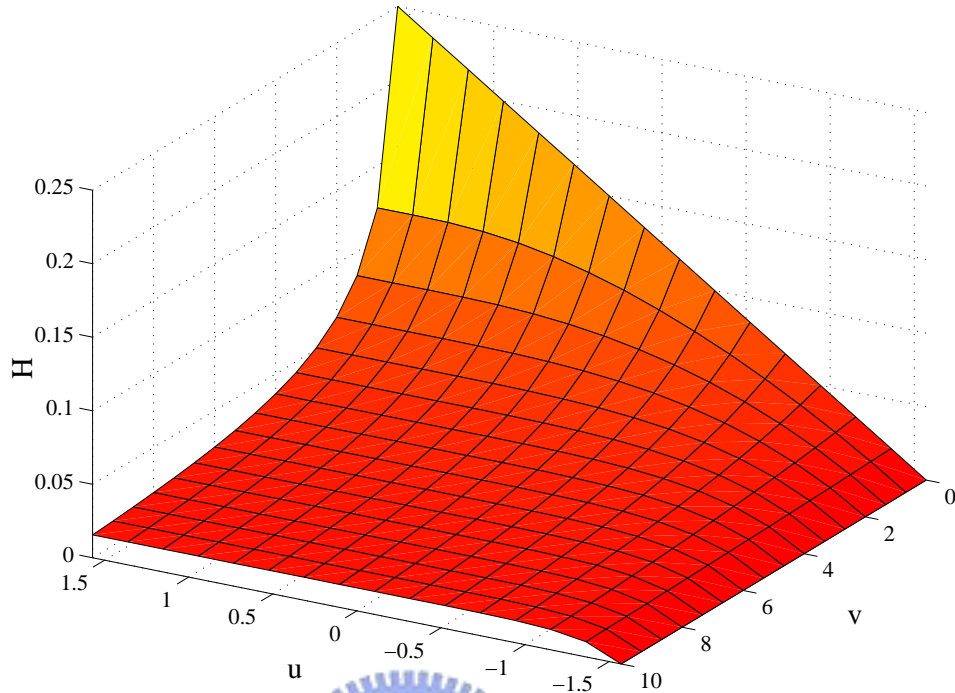


Figure 4.3: 3D plot of function $H(u, v)$ for $u \in [0, 10]$ and $v \in [-\pi/2, \pi/2]$ for $p(\theta) = 1/(4\pi)$. The graph demonstrates that the function is smooth, thus making it appropriate for tabulation during precomputation and look-up during run-time.

subsection. In such a case, the airlight is

$$L_a(\gamma, d_{sp}, \infty, \kappa_t) = C_0[H(C_1, \frac{\pi}{2}) - H(C_1, \theta_0)]. \quad (4.8)$$

4.2.2 Surface Radiance Model using Angle Formulation

Reflected radiance L_p is the sum of contribution from direct transmission $L_{p,d}$ (path a in Figure 4.1) and contribution from airlight $L_{p,a}$ (path c in Figure 4.1), which is given by the *surface radiance single scattering integral* [Sun et al., 2005]. This is written as

$$L_p = L_{p,d} + L_{p,a}, \quad (4.9)$$

where

$$L_{p,d} = \frac{I_0 e^{-\kappa_t d_{sp}}}{d_{sp}^2} BRDF(\theta_s, \phi_s, \theta_v, \phi_v) \cos \theta_s, \quad (4.10)$$

$$L_{p,a} = \int_{\Omega_{2\pi}} L_a(\gamma', d_{sp}, \infty, \kappa_t) BRDF(\theta_i, \phi_i, \theta_v, \phi_v) \cos \theta_i d\omega_i. \quad (4.11)$$

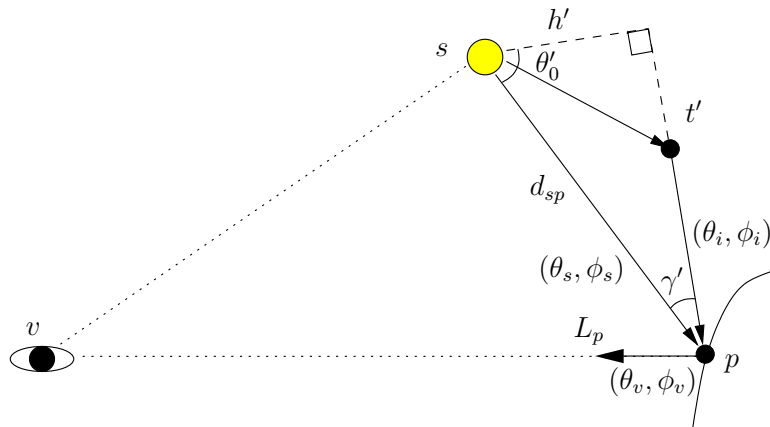


Figure 4.4: Diagram for light paths of direct transmission and airlight to surface point that contribute to surface radiance L_p .

The surface radiance single scattering integral sums airlight contribution to surface radiance over the surface point's hemisphere of incident directions. The angle γ' is the angle between the source direction (θ_s, ϕ_s) and incident direction (θ_i, ϕ_i) , such that $h' = d_{sp} \sin \gamma'$ and $t' = d_{sp} \cos \gamma'$. Refer to Figure 4.4 for a depiction of these variables. To simplify the integral for evaluation, the airlight term L_a in Equation 4.11 assumes no object obstructs each incident direction ($d_{vp} = \infty, \theta_d = \frac{\pi}{2}$).

Analytic solution to Surface Radiance L_p

We solve L_p for both the Lambertian BRDF and Phong BRDF separately. For the Lambertian case, $BRDF_{Lam} = k_d$, where k_d is the diffuse constant of the object. For the Phong case, all angles of Equation 4.9 are reparameterized against the reflection of the view ray about the surface normal [Ramamoorthi and Hanrahan, 2002] to reduce the complexity of the integral. We denote the reparameterized source direction θ'_s . After reparameterization, $BRDF_{Phong} = k_s \cos^n \theta'_s$, where k_s is the specular constant of the object. The resultant solution for $L_{p,d}$ is straightforward:

$$L_{p,d} = \frac{I_0 e^{-\kappa_t d_{sp}}}{d_{sp}^2} (k_d \cos \theta_s + k_s \cos^n \theta'_s). \quad (4.12)$$

To solve $L_{p,a}$, we make the substitution $T_{sp} = \kappa_t d_{sp}$ and use our analytic model for airlight L_a , Equation 4.8. Other substitutions and simplifications are made, allowing us to re-

express the integral portion as a function of two parameters. The details of this derivation for the solution to $L_{p,a}$ are shown below:

Lambertian BRDF case:

$$L_{p,a} = \int_{\Omega_{2\pi}} L_a(\gamma', d_{sp}, \infty, \kappa_t) BRDF(\theta_i, \phi_i, \theta_v, \phi_v) \cos \theta_i d\omega_i$$

1. Substitute $L_a(\gamma', d_{sp}, \infty, \kappa_t)$ with Equation 4.8 and $BRDF(\theta_i, \phi_i, \theta_v, \phi_v)$ with k_d .

$$= \int_{\Omega_{2\pi}} C_0(\gamma', d_{sp}, \kappa_t) [H(C_1(\gamma', d_{sp}, \kappa_t), \frac{\pi}{2}) - H(C_1(\gamma', d_{sp}, \kappa_t), \theta'_0)] k_d \cos \theta_i d\omega_i$$

2. Substitute for $C_0(\gamma', d_{sp}, \kappa_t)$, $C_1(\gamma', d_{sp}, \kappa_t)$, θ'_0 , and $T_{sp} = \kappa_t d_{sp}$.

3. Take constants out of integral.

$$= \frac{I_0 k_d \kappa_t}{d_{sp}} \int_{\Omega_{2\pi}} \frac{e^{-T_{sp} \cos \gamma'}}{\sin \gamma'} [H(T_{sp} \sin \gamma', \frac{\pi}{2}) - H(T_{sp} \sin \gamma', \gamma' - \frac{\pi}{2})] \cos \theta_i d\omega_i$$

4. Recall $\gamma'(\theta_s, \omega_i)$. So

$$= \frac{I_0 k_d \kappa_t}{d_{sp}} J^1(T_{sp}, \theta_s)$$



where

$$J^1(T_{sp}, \theta_s) = \int_{\Omega_{2\pi}} \frac{e^{-T_{sp} \cos \gamma'}}{\sin \gamma'} [H(T_{sp} \sin \gamma', \frac{\pi}{2}) - H(T_{sp} \sin \gamma', \gamma' - \frac{\pi}{2})] \cos \theta_i d\omega_i. \quad (4.13)$$

Phong BRDF case:

$$L_{p,a} = \int_{\Omega_{2\pi}} L_a(\gamma', d_{sp}, \infty, \kappa_t) BRDF(\theta_i, \phi_i, \theta_v, \phi_v) \cos \theta_i d\omega_i$$

1. Let R be the reflection of the view ray about the surface normal.
2. Reparameterize the integral about R .
3. Denote reparameterized source direction θ'_s [s.t. $\gamma'(\theta'_s, \omega_i)$ and $\theta'_0(\theta'_s, \omega_i)$].
4. Substitute $L_a(\gamma', d_{sp}, \infty, \kappa_t) = \text{Equation 4.8}$ and $BRDF(\theta_i, \phi_i, \theta_v, \phi_v) = k_s \cos^n \theta'_i$.

$$= \int_{\Omega_{2\pi}} C_0(\gamma', d_{sp}, \kappa_t) [H(C_1(\gamma', d_{sp}, \kappa_t), \frac{\pi}{2}) - H(C_1(\gamma', d_{sp}, \kappa_t), \theta'_0)] k_s \cos^n \theta'_i d\omega_i$$
5. Substitute for $C_0(\gamma', d_{sp}, \kappa_t)$, $C_1(\gamma', d_{sp}, \kappa_t)$, θ'_0 , and $T_{sp} = \kappa_t d_{sp}$.
6. Take constants out of integral.

$$\begin{aligned} &= \frac{I_0 k_s \kappa_t}{d_{sp}} \int_{\Omega_{2\pi}} \frac{e^{-T_{sp} \cos \gamma'}}{\sin \gamma'} [H(T_{sp} \sin \gamma', \frac{\pi}{2}) - H(T_{sp} \sin \gamma', \gamma' - \frac{\pi}{2})] \cos^n \theta'_i d\omega_i \\ &= \frac{I_0 k_s \kappa_t}{d_{sp}} J^n(T_{sp}, \theta'_s) \end{aligned}$$

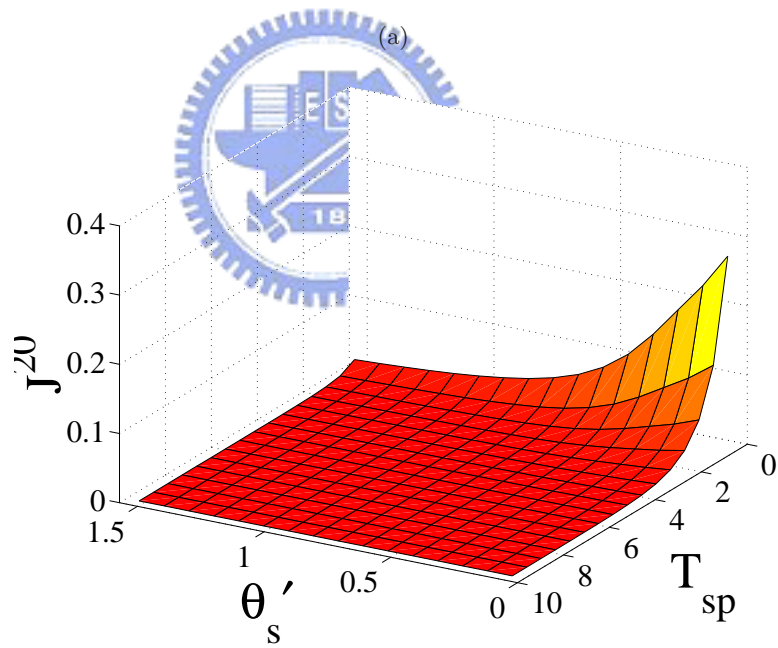
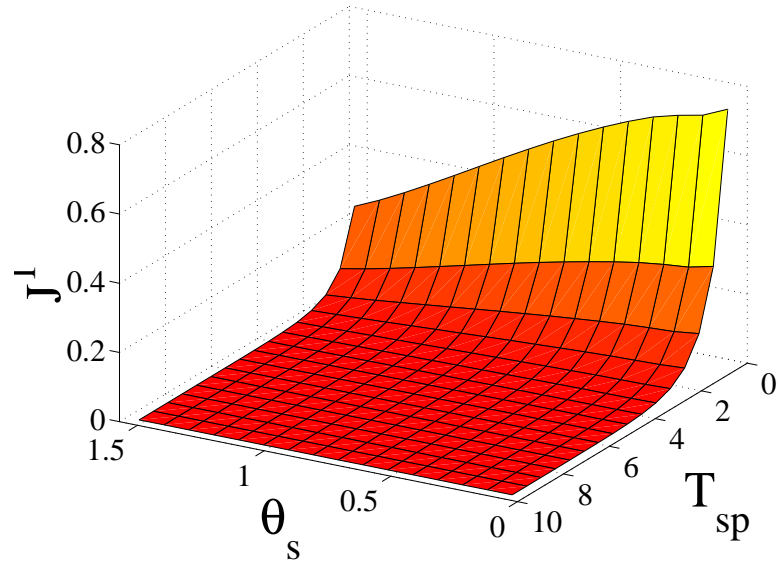
where

$$J^n(T_{sp}, \theta'_s) = \int_{\Omega_{2\pi}} \frac{e^{-T_{sp} \cos \gamma'}}{\sin \gamma'} [H(T_{sp} \sin \gamma', \frac{\pi}{2}) - H(T_{sp} \sin \gamma', \gamma' - \frac{\pi}{2})] \cos^n \theta'_i d\omega_i. \quad (4.14)$$

The resultant solution for $L_{p,a}$ is

$$L_{p,a} = \frac{I_0 \kappa_t}{d_{sp}} [k_d J^1(T_{sp}, \theta_s) + k_s J^n(T_{sp}, \theta'_s)], \quad (4.15)$$

Both $J^1(T_{sp}, \theta_s)$ and $J^n(T_{sp}, \theta'_s)$ can be numerically integrated, tabulated, and stored into texture memory during a precomputation stage, where n denotes the Phong exponent. Consequently, each of these functions can be evaluated with a texture lookup using linear interpolation during run-time. Plots of J^1 and J^n ($n = 20$) are shown in Figure 4.5, demonstrating that they are smooth.



(b)

Figure 4.5: 3D plots of functions (a) $J^1(T_{sp}, \theta_s)$ and (b) $J^{20}(T_{sp}, \theta'_s)$ for $T_{sp} \in [0, 10]$ and $\theta_s \in [0, \pi/2]$. The graphs demonstrate that the functions are smooth, thus making them appropriate for tabulation and look-up during run-time.

4.2.3 Summary of Analytic Lighting Model

In this section, a full analytic solution to lighting equation 4.2 is developed. The model can capture visual effects for glows around light sources and on surface radiance. Additionally, it may consider any anisotropic scattering distribution described by a phase function of one parameter under the assumption of single scattering. The complexity of the phase function has no effect on run-time performance whatsoever, since its computation is isolated to a precomputation stage. An illustrative summary of the analytic lighting model presented thus far is shown in Figure 4.6.

The choice of phase function $p(\theta)$ greatly affects the visual appearance of participating media, since it indicates different scattering conditions. The lighting model discussed up to this point requires the phase function $p(\theta)$ to be predetermined, as is common in many scattering models. Without instant visual feedback, it is often cumbersome to manipulate $p(\theta)$ to achieve desirable visual results. Furthermore, only a pre-determined, limited number of scattering environments are possible for interactive applications with this restriction. In the next section, we develop an extension to our lighting model, allowing the user to change the level of forward scattering in real-time.



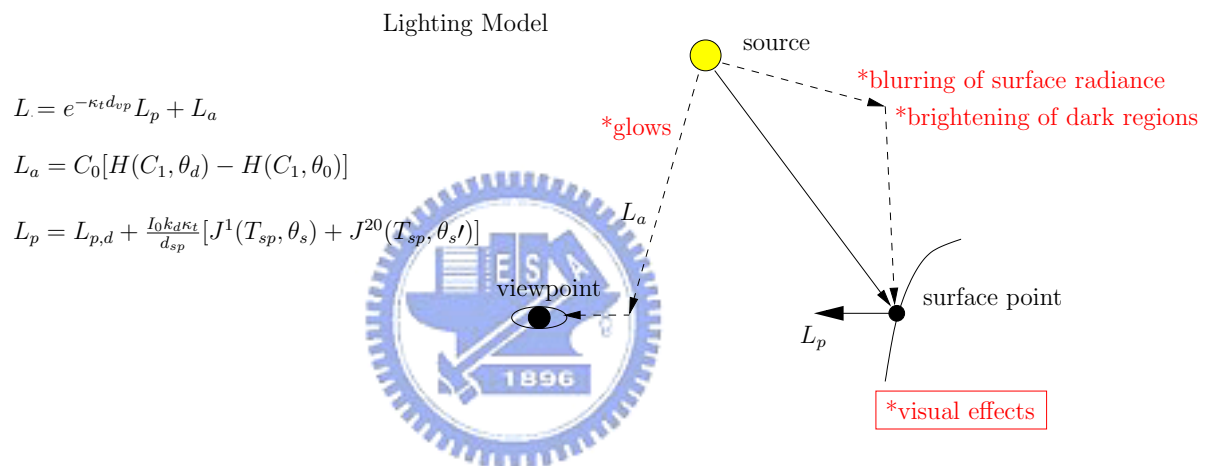


Figure 4.6: Illustrative summary of analytic lighting model in Section 4.2. L is implemented in the vertex or fragment shader using programmable graphics hardware. Each evaluation of function H , J^1 , or J^{20} costs a single texture lookup.

4.3 Dynamic Anisotropic Scattering

In this section, an extension to the analytic model of Section 4.2 for dynamic forward scattering is presented. Many previous works must use a pre-determined phase function before run-time, making it difficult to pre-visualize different scattering conditions and limiting the number of scattering environments possible for interactive applications. With the capability of manipulating the phase function at run-time, instant visual feedback as well as an unlimited variation of scattering environments is possible.

The form of the Henyey-Greenstein phase function in Equation 4.1 hinders interactive modification of scattering parameter g . We use a well known fact in the literature on light transport that phase functions can be written as a series of Legendre polynomials [Dunn, 1997, Narasimhan and Nayar, 2003]:

$$p(\cos \theta) = \frac{1}{4\pi} \sum_{k=0}^N (2k+1)b_k P_k(\cos \theta), \quad (4.16)$$

where $P_k(\cos \theta)$ is the Legendre polynomial of order k . The first four Legendre polynomials are listed below:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_2(x) = \frac{1}{2}(3x^2 - 1)$$

$$P_3(x) = \frac{1}{2}(5x^3 - 3x)$$

In the case of the Henyey-Greenstein phase function, the expansion coefficients are $b_k = g^k$.

Using this alternative expression for the Henyey-Greenstein phase function, we can develop analytic solutions to airlight L_a (Equation 4.3) and surface radiance L_p (Equation 4.9) with dynamic scattering by using the Legendre expansion, Equation 4.16, for $p(\theta)$ and moving forward scattering parameter g outside of any precomputed integrals. These steps are shown as follows:

Solution to Airlight L_a

$$L_a(\gamma, d_{sv}, d_{vp}, \kappa_t) = \frac{\kappa_t I_0 e^{-\kappa_t t}}{h} \int_{\theta_0}^{\theta_d} e^{-\kappa_t h \frac{\sin \theta + 1}{\cos \theta}} p(\theta + \frac{\pi}{2}) d\theta.$$

1. Substitute Equation 4.16 for $p(\theta + \frac{\pi}{2})$.

2. Take constants out of integral.

$$\begin{aligned} &= C_0 \sum_{k=0}^N (2k+1) g^k \int_{\theta_0}^{\theta_d} e^{\kappa_t h \frac{\sin \theta + 1}{\cos \theta}} P_k[\cos(\theta + \frac{\pi}{2})] d\theta. \\ &= C_0 \sum_{k=0}^N (2k+1) g^k [H_k(C_1, \theta_d) - H_k(C_1, \theta_0)] \end{aligned}$$

where

$$H_k(u, v) = \int_{-\frac{\pi}{2}}^v e^{-u(\frac{\sin \theta + 1}{\cos \theta})} P_k[\cos(\theta + \frac{\pi}{2})] d\theta. \quad (4.17)$$

An analytic solution to the angle-formulated airlight integral of Equation 4.3 with dynamic anisotropic scattering then is

$$L_a = C_0 \sum_{k=0}^N (2k+1) g^k [H_k(C_1, \theta_d) - H_k(C_1, \theta_0)], \quad (4.18)$$

The functions H_k for $k = \{0, 1, \dots, N\}$ are precomputed and tabulated in texture memory.

Solution to Scattering Contribution to Surface Radiance $L_{p,a}$

$$L_{p,a} = \int_{\Omega_{2\pi}} L_a(\gamma', d_{sp}, \infty, \kappa_t) BRDF(\theta_i, \phi_i, \theta_v, \phi_v) \cos \theta_i d\omega_i$$

1. Substitute $L_a(\gamma', d_{sp}, \infty, \kappa_t)$ with Equation 4.18.

$$= \int_{\Omega_{2\pi}} C_0 \sum_{k=0}^N (2k+1) g^k [H_k(C_1, \theta_d) - H_k(C_1, \theta_0)] BRDF(\theta_i, \phi_i, \theta_v, \phi_v) \cos \theta_i d\omega_i$$

2. Follow steps for the Lambertian and Phong BRDF cases in the derivation for Equation 4.15, skipping step 1 for the Lambertian BRDF case and step 3 for the Phong BRDF case.

$$= \frac{I_0 \kappa_t}{d_{sp}} \sum_{k=0}^N (2k+1) g^k [k_d J_k^1(T_{sp}, \theta_s) + k_s J_k^n(T_{sp}, \theta'_s)]$$

where

$$J_k^n(T_{sp}, \theta'_s) = \int_{\Omega_{2\pi}} \frac{e^{-T_{sp} \cos \gamma'}}{\sin \gamma'} [H_k(T_{sp} \sin \gamma', \frac{\pi}{2}) - H_k(T_{sp} \sin \gamma', \gamma' - \frac{\pi}{2})] \cos^n \theta_i d\omega_i. \quad (4.19)$$

An analytic solution to $L_{p,a}$ (Equation 4.9) for single scattering contribution to reflected surface radiance with dynamic anisotropic scattering is

$$\frac{I_0 \kappa_t}{d_{sp}} \sum_{k=0}^N (2k+1) g^k [k_d J_k^1(T_{sp}, \theta_s) + k_s J_k^n(T_{sp}, \theta'_s)], \quad (4.20)$$

The functions J_k^n for $k = \{0, 1, \dots, N\}$ are precomputed and tabulated in texture memory. Under these new models with dynamic anisotropic scattering, forward parameter g may be modified interactively.

Implementation of Equations 4.18 and 4.20 requires choosing a finite number of expansion terms N to approximate the Henyey-Greenstein phase function. Larger values for N naturally produce more accurate results, as shown by the plot in Figure 4.7(left). The approximation suffers increasingly for values g closer to 1, as demonstrated by Figure 4.7(right). In practice, we use $N = 3$ for efficiency purposes, and visually, it produces sufficient results.

A key step for efficient implementation is that functions H_k , for $k = 0, 1, 2, 3$, can be numerically precomputed and tabulated into the four channels of a single floating point texture. The same is true for the set of functions J_k^1 and J_k^n . Consequently, only one texture lookup is necessary to compute four function evaluations of H_k , J_k^1 , or J_k^n , since the values indexed are identical across all values of k . For $N = 3$, Equations 4.18 and 4.20 each only require evaluation of analytic functions and two texture lookups. These costs require the same number of texture lookups as the original lighting model (Equations 4.6 and 4.15); thus, adding dynamic anisotropic scattering to the model does not significantly affect performance. A final detail of note is that each term $(2k+1)g^k$ in the series is computed on the CPU and passed as a variable into the vertex or fragment shader to lessen the load on the GPU.

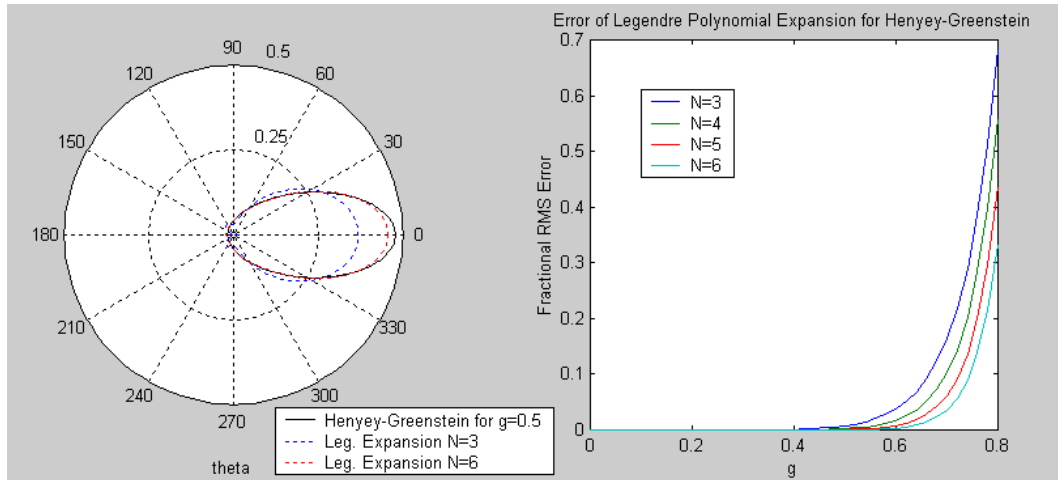


Figure 4.7: (left) Plot of the normalized Henyey-Greenstein phase function in solid line and its Legendre polynomial expansion (Equation 4.16) for expansion terms $N = 3, 6$ in dotted line. Larger N increases the accuracy of the expansion approximation. (right) Plot of the fractional rms error of airlight using the Legendre polynomial expansion for $N = 3, 4, 5, 6$ with varying forward scattering parameter g . Note the loss in accuracy of the expansion approximation for increasing g and smaller N .

4.4 Hardware Implementation

The flexibility created by recent advances in graphics hardware has enabled complex computation on the GPU to be administered by the programmer. As is common for numerous recent interactive rendering techniques, complex lighting calculations can be done by exploiting the programmability of vertex and fragment processing. Our lighting model can be implemented by evaluating Equation 4.2 in either the vertex or fragment shader using programmable graphics hardware. In our implementation, we used the OpenGL Shading Language to implement the fragment shader version for the lighting model with and without dynamic anisotropic scattering.

For static anisotropic scattering, resolution of Equation 4.2 requires implementation of Equation 4.6 for L_a as well as Equations 4.12 and 4.15 for L_p . Non-optimized pseudocode of a fragment shader version without dynamic anisotropic scattering is shown in Figure 4.8. For dynamic anisotropic scattering, resolution of Equation 4.2 requires implementation of Equation 4.18 for L_a as well as Equations 4.12 and 4.20 for L_p . Non-optimized pseudocode of a fragment shader version with dynamic anisotropic scattering is shown in

Figure 4.9 for $N = 3$ Legendre expansion terms. The entire computation for each version only requires evaluation of some analytic expressions and four texture lookups. Extinction coefficient κ_t and asymmetry parameter g , passed as parameters into the fragment shader, can be interactively modified to generate different scattering conditions.

All precomputed functions were computed into tables offline using MATLAB. The time required to compute each table varied from a few seconds to a few minutes. The resolutions for H and H_k were both 64×64 , and the resolutions for J^1 , J^n , and J_k^n were all 512×512 . These resolutions proved adequate to eliminate banding artifacts. Additionally, floating point textures were used, since non-floating point textures exhibited banding artifacts as well.



```

uniform sampler2D H
uniform sampler2D J1
uniform sampler2D J20
uniform float  $\kappa_t$ 

main()

Find  $\gamma, d_{sv}, d_{sp}, \theta_s, \theta'_s$ .
 $h = d_{sv} * \sin \gamma;$ 
 $t = d_{sv} * \cos \gamma;$ 
 $T_{sp} = d_{sp} * \kappa_t;$ 
 $C_0 = \kappa_t * I_0 * \exp(-\kappa_t * t)/h;$ 
 $C_1 = \kappa_t * h;$ 
 $\theta_0 = \gamma - 0.5 * \pi;$ 
 $\theta_d = \arccos(h/d_{sp});$ 

*****airlight Equation 4.6*****
 $h_1 = \text{texture2D}(H, \text{vec2}(C_1, \theta_d));$ 
 $h_2 = \text{texture2D}(H, \text{vec2}(C_1, \theta_0));$ 
airlight =  $C_0 * (h_1.x - h_2.x);$ 

*****diffuse surface radiance, Equation 4.12 and 4.15*****
 $j^1 = \text{texture2D}(J^1, \text{vec2}(T_{sp}, \theta_s));$ 
 $d_1 = I_0 * k_d * \exp(-T_{sp}) * \cos \theta_s / (d_{sp} * d_{sp});$ 
 $d_2 = I_0 * k_d * \kappa_t * j^1.x / d_{sp};$ 
diffuse =  $d_1 + d_2;$ 

*****specular surface radiance, Equation 4.12 and 4.15*****
 $j^{20} = \text{texture2D}(J^{20}, \text{vec2}(T_{sp}, \theta'_s));$ 
 $s_1 = I_0 * k_s * \exp(-T_{sp}) * \cos^n \theta'_s / (d_{sp} * d_{sp});$ 
 $s_2 = I_0 * k_s * \kappa_t * j^{20}.x / d_{sp};$ 

*****Equation 4.2*****
glFragColor = airlight + (diffuse + specular) *  $\exp(-\kappa_t * d_{vp});$ 

```

Figure 4.8: *Fragment shader pseudocode for the lighting model without dynamic anisotropic scattering*

```

uniform sampler2D H(x:H0,y:H1,z:H2,w:H3)
uniform sampler2D J1(x:J01,y:J11,z:J21,w:J31)
uniform sampler2D J20(x:J020,y:J120,z:J220,w:J320)
uniform float  $\kappa_t$ 
uniform  $w_0, w_1, w_2, w_3$  [ $w_k = (2k + 1)g^k$ ]

main()

Find  $\gamma, d_{sv}, d_{sp}, \theta_s, \theta'_s$ .
 $h = d_{sv} * \sin \gamma$ ;
 $t = d_{sv} * \cos \gamma$ ;
 $T_{sp} = d_{sp} * \kappa_t$ ;
 $C_0 = \kappa_t * I_0 * \exp(-\kappa_t * t)/h$ ;
 $C_1 = \kappa_t * h$ ;
 $\theta_0 = \gamma - 0.5 * \pi$ ;
 $\theta_d = \arccos(h/d_{sp})$ ;

*****airlight Equation 4.18*****
 $h_1 = \text{texture2D}(H, \text{vec2}(C_1, \theta_d))$ ;
 $h_2 = \text{texture2D}(H, \text{vec2}(C_1, \theta_0))$ ;
airlight =  $C_0 * [w_0 * (h_1.x - h_2.x) + w_1 * (h_1.y - h_2.y) +$ 
 $w_2 * (h_1.z - h_2.z) + w_3 * (h_1.w - h_2.w)]$ ;

*****diffuse surface radiance, Equation 4.12 and 4.20*****
 $j^1 = \text{texture2D}(J^1, \text{vec2}(T_{sp}, \theta_s))$ ;
 $d_1 = I_0 * k_d * \exp(-T_{sp}) * \cos \theta_s / (d_{sp} * d_{sp})$ ;
 $d_2 = I_0 * k_d * \kappa_t * [w_0 * j^1.x + w_1 * j^1.y + w_2 * j^1.z + w_3 * j^1.w] / d_{sp}$ ;
diffuse =  $d_1 + d_2$ ;

*****specular surface radiance, Equation 4.12 and 4.20*****
 $j^{20} = \text{texture2D}(J^{20}, \text{vec2}(T_{sp}, \theta'_s))$ ;
 $s_1 = I_0 * k_s * \exp(-T_{sp}) * \cos^n \theta'_s / (d_{sp} * d_{sp})$ ;
 $s_2 = I_0 * k_s * \kappa_t * [w_0 * j^{20}.x + w_1 * j^{20}.y + w_2 * j^{20}.z + w_3 * j^{20}.w] / d_{sp}$ ;

*****Equation 4.2*****
glFragColor = airlight + (diffuse + specular)* $\exp(-\kappa_t * d_{vp})$ ;

```

Figure 4.9: Fragment shader pseudocode for the lighting model with dynamic anisotropic scattering using $N = 3$ Legendre expansion terms.

Chapter 5

Experimental Results

The forward scattering level in an environment impacts the visual appearance in both obvious and subtle ways. It is important for someone designing scenes with participating media to understand these subtleties. The lighting model described in chapter 4 is useful for demonstrating these effects due to its dynamic scattering properties. Therefore, this chapter describes the visual effects for various levels of anisotropic scattering in section 5.1, followed by the images and performance obtained by the lighting model for real-time rendering scenes in section 5.2.

5.1 Visual Effects of Anisotropic Scattering

Our experiments have shown that the size of glows around light sources increases for smaller γ and decreases for larger extinction coefficient κ_t , distance from source to viewer d_{sv} , and forward scattering parameter g . For common rendering scenarios, this effectively means glows around light sources in direct view (small γ) increase with larger g . When light sources are outside of direct view, however, visibility increases and glows decrease considerably with larger g , due to the decrease in backscattering. Using Figure 5.1, we illustrate the visual effect of g on glows around light sources in and out of direct view by rendering with varying g ($g = 0, 0.5, 0.9$) while keeping other parameters constant.

As discussed in subsection 4.1, scattering leads to diffusing of specular highlights and brightening of shadows. We illustrate the effect of forward scattering parameter g on

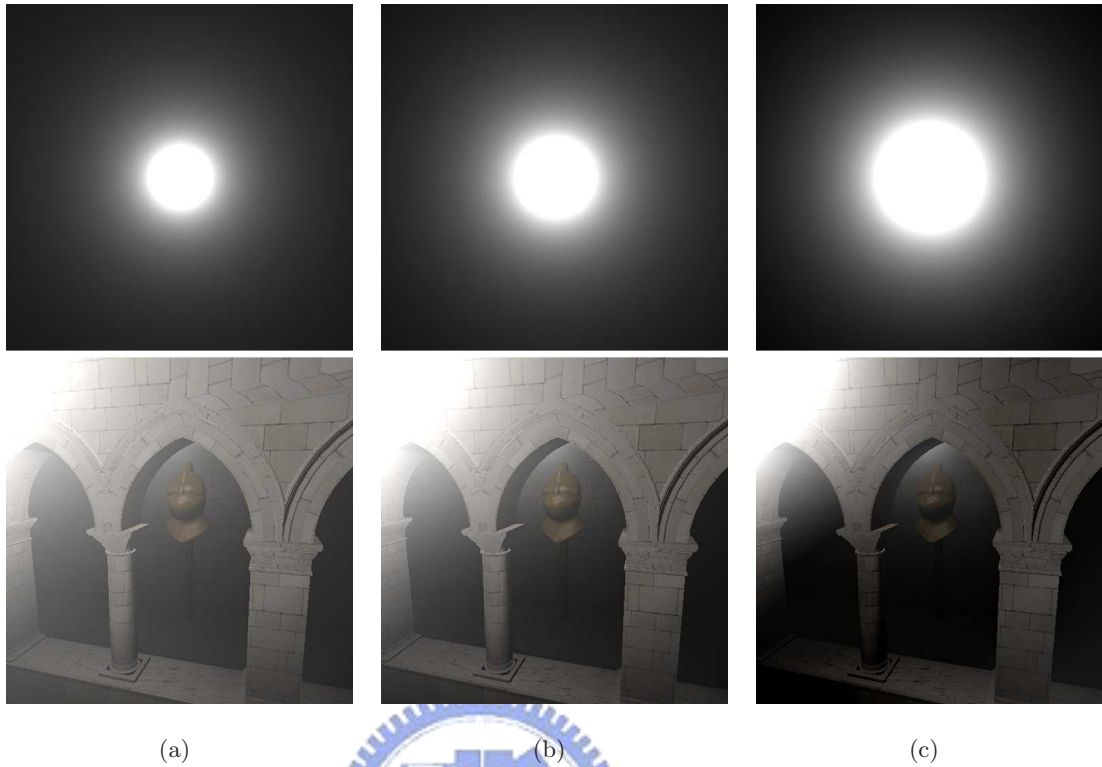


Figure 5.1: Captures images of glows around point light sources in (top row) direct view and (bottom row) out of direct view for forward scattering parameter (a) $g = 0$, (b) $g = 0.5$, and (c) $g = 0.9$. When increasing forward scattering parameter g , glows in direct view increase in size, whereas glows out of direct view decrease in size.

surface shading by rendering a sphere with only specular radiance in the top row and only diffuse radiance in the bottom row of Figure 5.2. When compared to the isotropic scattering case $g = 0$, increasing the amount of forward scattering lessens the diffusing of specular highlights (Figure 5.2c, top row) and lessens the brightening of shadows (Figure 5.2c, bottom row). In our rendering experiments, we have found that the effect of anisotropic scattering on diffuse radiance, namely the dark regions, is rather common. In contrast, the effect of anisotropic scattering on specular radiance can only be seen in rare cases.

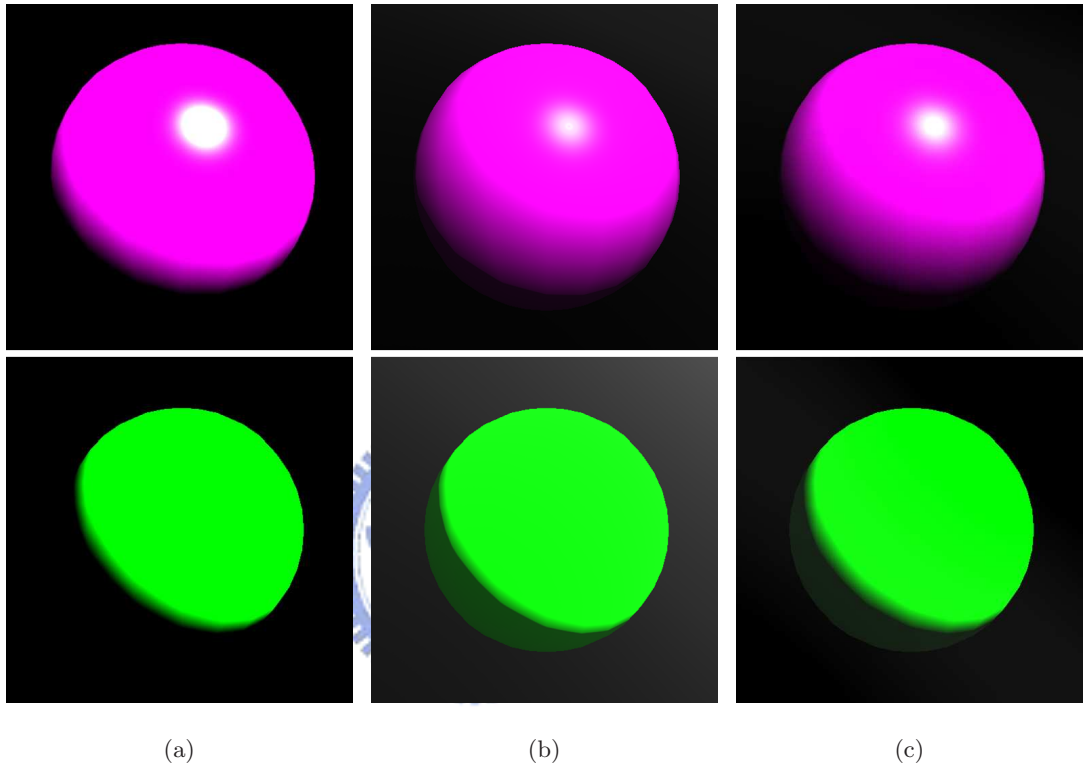


Figure 5.2: Images of rendered sphere using only specular radiance in top row and only diffuse radiance in bottom row where scattering conditions are set to (a) clear day, (b) isotropic scattering $g = 0$, and (c) strong forward scattering $g = 0.9$. Note that the diffusing of highlights exhibited in isotropic scattering (top row b) lessens under strong forward scattering conditions (top row c). Also note that the brightening of shadowed regions exhibited in isotropic scattering conditions (bottom row b) lessens under strong forward scattering conditions (bottom row c).

Table 5.1: *Table of rendering speeds (fps) as a function of number of light sources.*

	number of light sources			
	1	2	3	4
pavilion and pumpkin (32152 triangles)	30+	28-30	19-20	18-20
sponza atrium (83530 triangles)	20+	20-25	16-20	12-13

Table 5.2: *Table of computation costs (seconds) for precomputed function tables.*

	k			
	0	1	2	3
H_k	2.48	133	136	152
J_k^1	171	212	243	432
J_k^{20}	182	212	250	432

5.2 Rendering Test Scenes

Using an NVidia GeForce 6800 graphics card, the lighting model was implemented with and without dynamic anisotropic scattering for $N = 3$ in the fragment shader using the high-level shader language, GLSL. In our implementation, functions H_k are precomputed offline and tabulated into a single 512 x 512 floating point texture. Each set of functions J_k^1 and J_k^n are precomputed offline and tabulated into a single 64 x 64 floating point texture. Using MATLAB, the computation cost for each table varied from a few seconds to a several minutes. These times are shown in Table 5.2. On a side note, functions H , J^1 , and J^n are equivalent to functions H_0 , J_0^1 , and J_0^n ($k = 0$), respectively.

The performance of our lighting model depends on a number of factors and is roughly linear in the number of light sources. Run-time rendering rates as a function of number of light sources are shown in Table 5.1. We demonstrate our lighting model using varying levels of anisotropic scattering and extinction coefficients on various scenes with several light sources. The images in Figure 5.3 and Figure 5.5 were captured at upwards of 18 fps at a resolution of 640 x 480. This pavilion and pumpkin scene consists of 16229 vertices, 32152 triangles, and 4 light sources. Rendered images of an atrium scene are shown in Figure 5.4 and Figure 5.6. This scene consists of 43,479 vertices and 83530 triangles. Images of this scene with 2 light sources were captured at upwards of 20 fps. With 4 light sources, the frame rate became 10-12 fps.



(a)



(b)



(c)

Figure 5.3: Captured images using different densities: (a) clear day, (b) $\kappa_t = 0.05$, and (c) $\kappa_t = 0.20$.



(a)



(b)



(c)

Figure 5.4: Captured images using different levels of forward scattering: (a) clear day, (b) isotropic scattering $g = 0$, and (c) strong forward scattering $g = 0.75$. Notice the glows around light sources and brightening of the shadowed regions on the knight's face in isotropic scattering conditions (b). When increasing the amount of forward scattering in (c), the glows increase in size, and the shadowed regions darken.

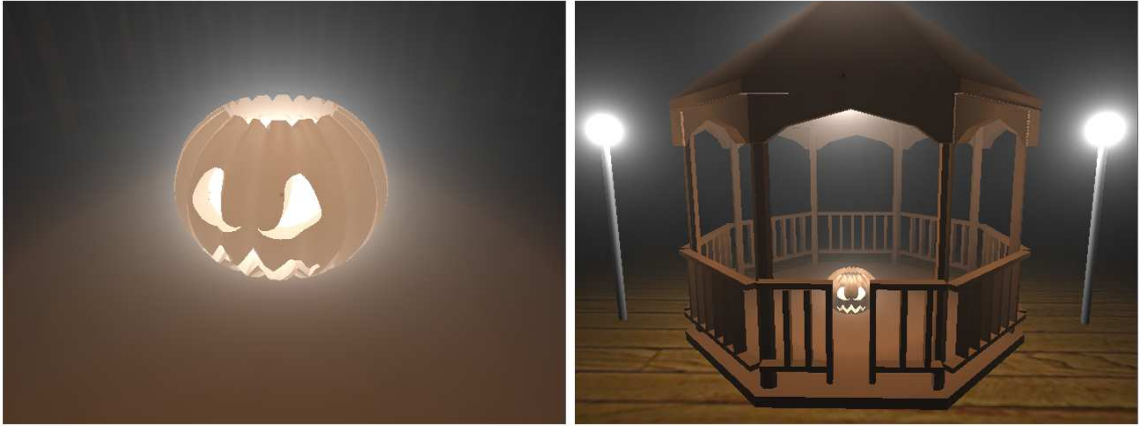


Figure 5.5: *Images of pavilion scene*



Figure 5.6: *Images of atrium scene*

Chapter 6

Conclusion

In this final chapter, a brief summary of this thesis is presented, and future works are discussed.

6.1 Summary

The main contributions described by this thesis include

- a simple, efficient, and accurate analytic lighting model for general anisotropic single scattering that is effective for interactive applications. Straightforward fragment shader pseudocode is provided and implemented in real-time. The model may consider any phase function of a single parameter without effect on run-time performance since its computation is isolated to precomputed two-dimensional tables that are accessed at run-time using texture memory.
- a dynamic scattering environment in which users may control scattering conditions using simple parameters. The user may vary the level of forward scattering and the density of the medium, able to generate an unlimited variation of scattering environments.
- a demonstration of the visual effects for forward scattering on glows around light sources and on surface radiance. The dynamic anisotropic scattering behavior of our



model allows us to render these complex effects in different scattering environments easily.

Our real-time rendering method for accurate anisotropic scattering increases the visual realism that interactive computer graphics applications may consider. Furthermore, simple scattering parameters controlling forward scattering and density may be manipulated in real-time to generate different scattering conditions, enabling the user a high level of interactivity in complex scenes with participating media. This flexibility is a key step towards practical inclusion of complex volumetric scattering effects in more real-time applications.

6.2 Future Work

A possible direction of future work is the development of an analytic angle-formulated scattering model which considers anisotropic light sources, since the intensity is singularly dependent upon only angle. Also, algorithms for volumetric shadows using analytic techniques have potential for real-time applications. A discussion for each of these avenues follows.



6.2.1 Analytic Model for Scattering from Anisotropic Light Sources

Although anisotropic light sources reflect many real-world situations, light scattering effects for scenes with anisotropic light sources are difficult to include in any analytic model due to the non-constant nature of source radiance. A possible direction for future work is the development of a model for scattering from a light source such as the commonly used OpenGL spotlight model. The intensity of the source falls off with angle θ between the source's illumination axis and a light ray, described by

$$I(\theta) = \begin{cases} I_0 \cos^N \theta & \text{if } -\theta_c < \theta < \theta_c \\ 0 & \text{else} \end{cases} \quad (6.1)$$

where N is the spotlight exponent and θ_c is the critical angle when the intensity drops to 0. The angle-formulated analytic model for airlight (Equation 4.6) is a good starting point

for considering scattering effects for a light source with the intensity distribution of Equation 6.1 due to the anisotropic light source intensity's dependence on angle. Substituting Equation 6.1 as the source intensity in our angle-formulated analytic model is shown below:

$$L_a(\gamma, d_{sv}, d_{vp}, \kappa_t) = \frac{\kappa_t e^{-\kappa_t t}}{h} \int_{\theta_0}^{\theta_d} e^{-\kappa_t h \left(\frac{\sin \theta + 1}{\cos \theta} \right)} I(\theta + \alpha) p\left(\theta + \frac{\pi}{2}\right) d\theta.$$

-- > Substitute Equation 6.1 for $I(\theta + \alpha)$

$$L_a = C_0 \int_{\max(\theta_0, \alpha - \theta_c)}^{\min(\theta_d, \alpha + \theta_c)} e^{-C_1 \left(\frac{\sin \theta + 1}{\cos \theta} \right)} I(\theta + \alpha) p\left(\theta + \frac{\pi}{2}\right) \cos^N(\theta + \alpha) d\theta.$$

The angle between the illumination axis and the perpendicular bisector to the view ray with respect the source is denoted α , shown in Figure 6.1. The challenge for developing an analytic model appropriate for real-time rendering is movement of the physical parameter α outside of the integral. A substitution using a series of cosine terms $\cos^N x = \sum_i A_i \cos(ix)$ was considered but proved impractical due to the large number of terms if N is large.

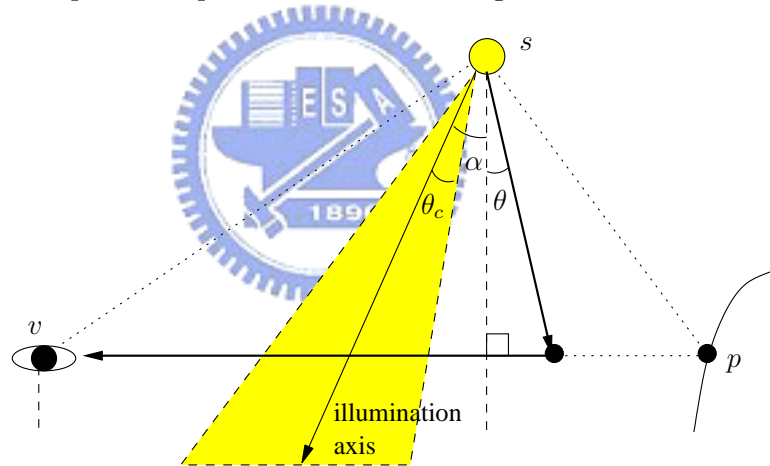


Figure 6.1: *Diagram of scattering with anisotropic light source*

6.2.2 Volumetric Shadows

Rendering volumetric shadows using analytic techniques has potential for real-time applications, as shown by the volumetric shadow algorithm [Biri et al., 2006] of Section 3.2.1. In addition to the high fill rate for rendering the shadow volume, the algorithm also has problems when rendering shadow polygons of relatively large size. A possible improvement

for the latter problem is to avoid having to sort the shadow polygons, which is required to determine which shadow polygons are themselves in shadow. Instead, a shadow map can be used to resolve the visibility of each shadow polygon. This experimental algorithm is summarized in the following paragraphs, along with the unsolved problems encountered.

The algorithm utilizes our analytic lighting model and requires the construction of the classic shadow maps and shadow volumes. With modification to the fragment shader, it can be easily implemented as an extra rendering pass of shadow planes generated by the shadow volume algorithm.

Airlight with Occluders

To include volumetric shadows, we must consider airlight that never reaches the viewer due to occluders along the view ray. For this purpose, shadow polygons are generated by the shadow volume algorithm to indicate where the view ray potentially enters into and exits out of shadowed regions. To determine whether a point along the view ray intersecting a shadow polygon at a distance x in fact enters into or exits out of shadowed regions can be determined by the binary visibility function $V(x)$. We consider a point visible if it lies at the edge of a shadowed region. Thus, $V(x) = 1$ only at those points where the view ray enters or exits shadowed regions, and $V(x) = 0$ otherwise. Refer to Figure 6.2 for an illustration of $V(x)$. Assuming all objects are closed, we may then express the amount of airlight with occluders L'_a as

$$L'_a = L_a(d_{vp}) - \sum_i^n V(d_{i,f})L_a(d_{i,f}) - V(d_{i,b})L_a(d_{i,b}) \quad (6.2)$$

where n is the number of shadow polygons intersecting the view ray, $d_{i,b}$ is the distance to the i^{th} back-facing shadow polygon intersecting the view ray, and $d_{i,f}$ is the distance to the i^{th} front-facing shadow polygon intersecting the view ray.

Implementation Design

Equation 6.2 can be implemented by rendering the shadow polygons of the scene in an extra pass using the fragment shader. The analytic model for airlight, Equation 4.6, is used evaluate the airlight terms L_a in the fragment shader.

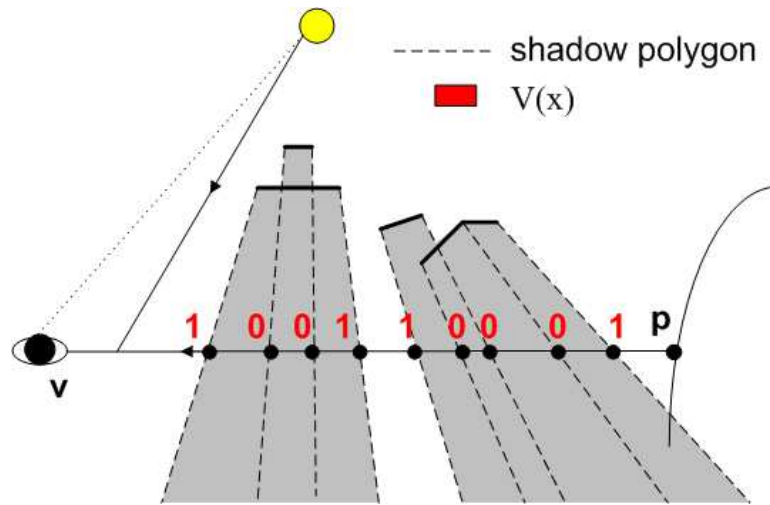


Figure 6.2: *Diagram for volumetric shadow algorithm.*

Shadow maps are used to determine $V(d)$. Lookup of a shadow map in the fragment shader at the transformed coordinates for simply distance d along the view ray will not surprisingly generate inaccurate results since a shadow polygon, conceptually the edge of a shadowed region, may or may not be visible according to a shadow map. Instead, $V(d)$ is determined by a lookup of all adjacent texels to the texel corresponding to the transformed coordinates at d . If any one of the adjacent texels is not under shadow according to the shadow map, then $V(d) = 1$. Otherwise $V(d) = 0$. Our experiments have shown this is a good indicator for $V(d)$, though there definitely is potential for a more efficient indicator.

The steps for rendering volumetric shadows consist of:

SceneGeometry Pass

1. Render scene geometry. If fragment in shadow, assign contribution to 0.

That is, this step computes $L_a(d_{vp})$ of Equation 6.2.

Shadow polygons pass

1. Set depth test write off.
2. Set additive blending.

3. Render front-facing shadow polygons.

That is, this step computes $\sum_i^n V(d_{i,f})L_a(d_{i,f})$ of Equation 6.2.

4. Set reverse subtractive blending.

5. Render back-facing polygons.

That is, this step computes $-\sum_i^n V(d_{i,b})L_a(d_{i,b})$ of Equation 6.2.

Experimental Results

Volumetric shadows for simple objects can be rendered; however, the shadow polygons for complex objects cause unsolved discontinuous shading problems. One source of this problem is the situation when shadow polygons intersect other shadow polygons. In such a case, fragments at the intersection of two shadow polygons are rendered twice. Additionally, clamping problems occur when the sum of contributions exceed one for the pass when rendering front-facing shadow polygons. Lastly, the fill rate for the shadow polygons greatly affects the performance of the algorithm, such that complex scenes slow the frame rate significantly. Images from some experiments are shown in Figures 6.3, 6.4, 6.5, 6.6, and 6.7.

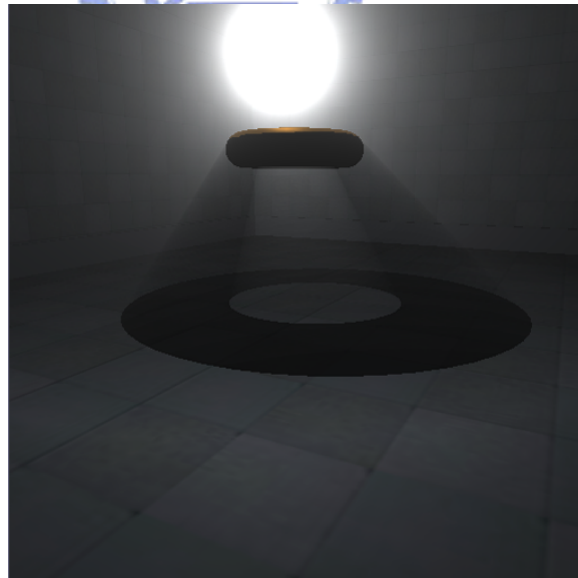


Figure 6.3: *Image of simple torus' volumetric shadow.*



Figure 6.4: *Image of simple table's volumetric shadow.*



Figure 6.5: *When intersecting shadow polygons exist, fragments at the intersection are incorrectly rendered multiple times with non-zero contribution.*

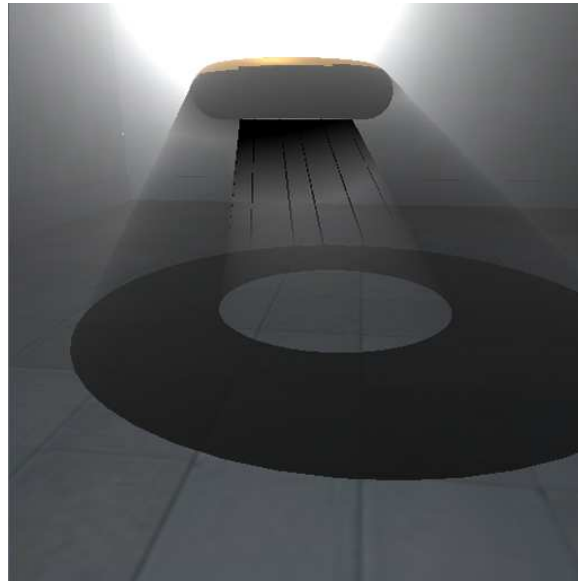


Figure 6.6: Image illustrating clamping problem. RGB values clamp to one incorrectly when the contribution of the scene and front-facing shadow polygons exceeds one. High intensity of light source induces clamping problems, causing general over-darkening of the torus' hole. The really dark strips result where more shadow polygons are rendered, since sharp edges in the silhouette may have concave areas (not a fundamental problem in itself, just a result of clamping problem).



Figure 6.7: Image of car's volumetric shadow. Discontinuous shading problems occur when rendering the shadow of more complex objects. The source of discontinuous shading problems include intersecting shadow polygons, clamping, and perhaps other unforeseen reasons.

Bibliography

- [Ashikhmin et al., 2004] Ashikhmin, M., Premoze, S., Ramamoorthi, R., and Nayar, S. (2004). Blurring of light due to multiple scattering by the medium: a path integral approach. Technical report CUCS-017-04, Columbia University.
- [Biri et al., 2002] Biri, V., Michelin, S., and Arques, D. (2002). Real-time animation of realistic fog. In *Thirteenth Eurographics Workshop on Rendering 2002*.
- [Biri et al., 2006] Biri, V., Michelin, S., and Arques, D. (2006). Real-time rendering of atmospheric scattering and volumetric shadows. In *The 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006*.
- [Cerezo et al., 2005] Cerezo, E., Perez-Cazorla, F., Pueyo, X., Seron, F., and Sillion, F. (2005). A survey on participating media rendering techniques. *The Visual Computer*.
- [Dobashi et al., 2000a] Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., and Nishita, T. (2000a). A simple, efficient method for realistic animation of clouds. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 19–28.
- [Dobashi et al., 2000b] Dobashi, Y., Yamamoto, T., and Nishita, T. (2000b). Interactive rendering method for displaying shafts of light. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 31.
- [Dobashi et al., 2002] Dobashi, Y., Yamamoto, T., and Nishita, T. (2002). Interactive rendering of atmospheric scattering effects using graphics hardware. In *HWWS '02: Pro-*

- ceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 99–107.
- [Dunn, 1997] Dunn, A. (1997). *Light Scattering Properties of Cells*. PhD thesis, University of Texas at Austin.
- [Everitt and Kilgard, 2002] Everitt, C. and Kilgard, M. (2002). Practical and robust stenciled shadow volumes for hardware-accelerated rendering.
- [Harris and Lastra, 2001] Harris, M. J. and Lastra, A. (2001). Real-time cloud rendering. In *EG 2001 Proceedings*, pages 76–84.
- [Hegeman et al., 2005] Hegeman, K., Ashikhmin, M., and Premoze, S. (2005). A lighting model for general participating media. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 117–124.
- [Hoffman and Preetham, 2003] Hoffman, N. and Preetham, A. J. (2003). Real-time light-atmosphere interactions for outdoor scenes. *Graphics programming methods*, pages 337–352.
- [Ishimaru, 1997] Ishimaru, A. (1997). *Wave Propagation and Scattering in Random Media*. IEEE Press.
- [Iwasaki et al., 2003] Iwasaki, K., Dobashi, Y., and Nishita, T. (2003). A volume rendering approach for sea surfaces taking into account second order scattering using scattering maps. In *VG '03: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 129–136.
- [Jensen, 2001] Jensen, H. W. (2001). *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd.
- [Jensen and Christensen, 1998] Jensen, H. W. and Christensen, P. H. (1998). Efficient simulation of light transport in scences with participating media using photon maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320.

- [Kajiya and Herzen, 1984] Kajiya, J. T. and Herzen, B. P. V. (1984). Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174.
- [Lafortune and Willems, 1996] Lafortune, E. P. and Willems, Y. D. (1996). Rendering participating media with bidirectional path tracing. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 91–100, London, UK. Springer-Verlag.
- [Lecocq et al., 2000] Lecocq, P., Kemeny, A., Michelin, S., and Arques, D. (2000). Mathematical approximation for real-time lighting rendering through participating media. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 400.
- [Lloyd et al., 2004] Lloyd, B., Wendt, J., Govindaraju, N., and Manocha, D. (2004). Cc shadow volumes. In *Proceedings of the Eurographics Symposium on Rendering (2004)*.
- [Max, 1995] Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108.
- [Narasimhan and Nayar, 2003] Narasimhan, S. G. and Nayar, S. K. (2003). Shedding light on the weather. In *IEEE CVPR*.
- [Nishita et al., 1996] Nishita, T., Dobashi, Y., and Nakamae, E. (1996). Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 379–386.
- [Nishita et al., 1987] Nishita, T., Miyawaki, Y., and Nakamae, E. (1987). A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 303–310.
- [Nishita et al., 1985] Nishita, T., Okamura, I., and Nakamae, E. (1985). Shading models for point and linear sources. *ACM Trans. Graph.*, 4(2):124–146.

- [Nishita et al., 1993] Nishita, T., Sirai, T., Tadamura, K., and Nakamae, E. (1993). Display of the earth taking into account atmospheric scattering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 175–182.
- [O’Neil, 2005] O’Neil, S. (2005). Accurate atmospheric scattering. *GPU Gems 2*.
- [Pharr and Humphreys, 2004] Pharr, M. and Humphreys, G. (2004). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc.
- [Preetham, 2003] Preetham, A. J. (2003). Modeling skylight and aerial perspective. ACM SIGGRAPH 2003 Course #1 Notes.
- [Preetham et al., 1999] Preetham, A. J., Shirley, P., and Smits, B. (1999). A practical analytic model for daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91–100.
- [Premoze et al., 2004] Premoze, S., Ashikhmin, M., Ramamoorthi, R., and Nayar, S. K. (2004). Practical rendering of multiple scattering effects in participating media. In *Rendering Techniques*, pages 363–373.
- [Premoze et al., 2003] Premoze, S., Ashikhmin, M., and Shirley, P. (2003). Path integration for light transport in volumes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 52–63.
- [Ramamoorthi and Hanrahan, 2002] Ramamoorthi, R. and Hanrahan, P. (2002). Frequency space environment map rendering. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 517–526.
- [Riley et al., 2004] Riley, K., Ebert, D. S., Kraus, M., Tessendorf, J., and Hansen, C. (2004). Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering*, pages 375–386.
- [Sloup, 2002] Sloup, J. (2002). A survey of the modelling and rendering of the earth’s atmosphere. In *Proceedings of the 18th Spring Conference on Computer Graphics*.

- [Sun et al., 2005] Sun, B., Ramamoorthi, R., Narasimhan, S. G., and Nayar, S. K. (2005). A practical analytic single scattering model for real-time rendering. *ACM Transactions on Graphics (SIGGRAPH)*.
- [Wang, 2004] Wang, N. (2004). Realistic and fast cloud rendering. *Journal of Graphics Tools*, 9(3):21–40.
- [Williams, 1978] Williams, L. (1978). Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274.
- [Zhang and Crawfis, 2002] Zhang, C. and Crawfis, R. (2002). Volumetric shadows using splatting. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 85–92, Washington, DC, USA. IEEE Computer Society.
- [Zhang and Crawfis, 2003] Zhang, C. and Crawfis, R. (2003). Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):139–149.

