Real-time Rendering of Translucent Materials

Mircea-Costin Rohat

Kongens Lyngby 2012
IMM-MSc-2012-0060

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-MSc-2012-0060

Summary

The goal of the thesis is to implement and describe the implementation of one solution for capturing the correct appearence of light's complex behavior in translucent materials. The thesis focuses on real-time performance: a quality that physically based solutions do not offer.

Related literature study has helped to shape a solution that combines previous works. Still, the solution detaches itself from the previous ones due to different interpretations and the aim was to develop a more robust algorithm.

The implementation for the chosen solution takes full advantage of the modern graphics processing unit's programmable pipeline and uses programming procedures known as point splatting, texture space sampling, environment map filtering and deferred rendering. The underlying theory comes from radiometry, diffusion approximation and numerical integration.

To reach real-time performance, a trade-off has been made and light-geometry interaction has been limited to account only for single-bounce and multiple scattering events inside the medium. The renderings obtained are visually pleasing, they capture the appearence of the materials they depict and they match (to a degree set by the above mentioned trade-off) their corresponding off-line renderings.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modeling at the Technical University of Denmark in fulfillment of the requirements for acquiring an M.Sc. in Informatics. The author, Mircea-Costin Rohat, has worked on this thesis from February to July 2012 under the supervision of his teacher Jeppe Revall Frisvad.

The thesis deals with real-time rendering of translucent materials using the capabilities of modern computer graphics hardware. This topic is constantly extending in the field of computer graphics and the theory behind it is of great significance in rendering materials such as marble, jade or wax.

The author's interest in this subject arose during the winter semester of 2011. After successfully finishing two important courses dealing with real-time graphics and physically based rendering that took place in the spring semester of 2011, in the winter semester he has chosen to start a course project on real-time teeth rendering. Time was too short to allow a thorough study of the topic and to achieve a full and robust solution to the problem, but nevertheless it was enough for the author to see the great potential that sub-surface light scattering has for rendering real-life images. Without delay, as soon as his final semester has started, he has enrolled in writing his Master's Thesis registered under the name of *Real-time Rendering of Translucent Materials*.

The thesis consists of a practical application and a report. The application was developed solely by the author during the duration of his thesis, and the source code relevant to the topic was written entirely by him. The complete solution, however, includes C/C++ libraries for geometry calculations or the Fourier Transform (FFTW - available under GNU General Public License) or

two source code files (for generating random numbers) available with their authors' permission. The report was written by the author alone, simultaneous with the development of the application. References or source links are provided everywhere he has used the work of other authors. The screenshots present are rendered using the application developed, and photographs (unless the source is specified) were taken by the author or used with permission.

Lyngby, 02-July-2012

Mircea-Costin Rohat

Acknowledgements

First of all, I would like to thank my supervisor and teacher, Jeppe Revall Frisvad, for constant guidance, support and feedback, during the duration of the thesis as well as during the previous two semesters. During the period in which the thesis has been developed, the weekly group meetings, private meetings during his office hours and his e-mail responses to my questions and requests have been of vital aid. I would also like to thank my teacher Jakob Andreas Bærentzen, who, along with Jeppe led the weekly group meetings. Their advice, directions and ideas were a great source of inspiration and motivation.

Many thanks go to my colleagues Meletis Stathis and Martin Skytte Kristensen for contributing with ideas, advice and feedback for my results. As well, Peter Dahl Ejby Jensen from 3Shape has been of great help, showing a lot of interest in my progress and results, offering feedback and directions and pointing out where to emphasize in my report. Nonetheless, all three of them showed great moral support and played a significant role in my progress.

Last but not least, I wouldn't have had the opportunity to achieve the results of my thesis without the university's staff. I sincerely thank you all for providing me a desk inside a project room and a powerful computer and graphics card.

Contents

Chapter 1

Introduction

Translucent materials range from solids and liquids (marble, jade, wax or milk) to organic materials (skin, algae, leaves or flower petals). As opposed to opaque materials which reflect most of the incident light back into the environment and completely transparent materials which allow incident light to pass through them unmodified, these materials allow incident light to penetrate them, scatter inside and to be absorbed or to exit at a different point with another direction. The interaction between light and the material takes place at a very small scale and may be with molecules or atoms (in solids like marble) or cellular structures (in leaves, algae or skin). Besides the particular coloring, this behavior gives front lit materials a diffuse look and is especially seen when the objects are lit from behind (as seen in figure 1.1).

Computer animated movies successfully use off-line rendering methods to render realistic images of scenes that contain translucent materials. Unfortunately, their long rendering time (up to hours) makes these methods not suitable for use in interactive applications. Having a real-time method to simulate these effects adds an important amount of realism to the scene and allows a broader range of models to be rendered accurately, models such as skin, teeth, fruits or gem stones.

This thesis proposes to present a solution to the problem of rendering translucent materials in real-time with accurate depiction of their distinctive appearence. To

Figure 1.1: Examples of Subsurface Scattering. The phenomenon may appear everywhere in the surrounding environment: from solids, plants and fruits to skin and body tissue. Note the distinctive appearence of back lit objects depending on their thickness and composition (the top row and the human hand) and the diffuse appearence of wax (bottom right) and wall tiles (middle left)

complete the goal, related literature and related theory have been studied and a solution has been implemented and tested. A report, containing the essence of both the study phase and the implementation phase, has been written during the duration of the thesis.

The report is structured into 8 chapters. After this short introduction to subsurface scattering is given and the problem is described, the next chapter (chapter 2) gives an overview of the related literature and previous solutions to the problem. The next three chapters are essential to understand the implementation of the solution. First, in chapter 3 the theoretical background is presented followed by chapter 4 which explains in theoretical terms the method chosen. At the same time, chapter 4 will guide the reader from the theoretical principles of chapter 3 to the practical issues in chapter 5. Chapter 5 is dedicated to details of the particular implementation used and it is intended to give enough practical information that a programmer can use to replicate the solution and at the same time to motivate the decisions taken. Chapter 6 presents the results obtained and validates the implementation by comparing the results with other methods. Chapter 7 presents additional work that can be done to improve the solution and gives references to literature useful for the implementation. The final chapter briefly revisits the previous chapters and presents the conclusions.

Chapter 2

Related Work

Probably the most influential work in the field of subsurface scattering has been done by Henrik Wann Jensen et al in *A Practical Model for Subsurface Light Transport* [Jensen et al., 2001]. The authors present a method to evaluate light transport due to subsurface scattering in translucent materials taking into account single scattering and multiple scattering. The multiple scattering term is calculated based on the observation that an incoming infinitesimal beam on light will scatter in an uniform isotropic manner inside a highly scattering medium. Using a method called dipole approximation, the authors calculate the diffuse reflectance from a uniformly lit semi-finite material slab.

Using the theory from [Jensen et al., 2001], Jensen and Buhler presented a new and fast approach to the subsurface light transport problem in [Jensen and Buhler, 2002]. For rendering purposes, they describe a sampling method and hierarchical method of integrating the iradiance from the sample points that are easy to include in a ray tracer implementation. They have successfully achieved a huge speed-up using their BSSRDF approximation method (5 minutes rendering time) against a Monte Carlo simulation method (1250 minutes). Due to the continually increasing computational power and the possibilities offered nowadays by the programmable GPU pipeline their method might be suitable for achieving interactive frame rates.

In his Ph.D. Thesis *Towards Realistic Image Synthesis of Scattering Materi-*

als [Donner, 2006], Craig Donner extended the previous theory on diffusion approximation. Instead of using a dipole source, the new theory uses a number of multipole sources mirrored above and below the slab to calculate the reflectance and transmittance profiles. The condition that the slab should be semi-infinitely thick [Jensen et al., 2001] is now relaxed and the method is dedicated to thick and thin slabs with multilayer components with varying indices of refraction. Combining the profiles of multiple layers is done by applying Kubelka-Munk theory in frequency space. The method can be integrated into ray tracing applications or real-time applications and, as the results from the paper show, it is highly accurate and very close to a Monte Carlo simulation.

A different method, running in real time but based on the theory from [Jensen et al., 2001] was proposed by Carsten Dachsbacher and Marc Stamminger in *Translucent Shadow Maps* [Dachsbacher and Stamminger, 2003]. The algorithm is very similar to shadow mapping: a shadow map is rendered containing the information on irradiance and normals in each pixel, along with 3D position, data which is needed to calculate the translucency. On each pixel seen from the observer's perspective a filter with different precomputed weights is applied to efficiently integrate the contribution from all sample points in the shadow map. The technique relies on mip-map textures and the programmable pipeline of modern GPUs, and unfortunately it is restricted to directional light sources. Another drawback is that the method makes use of 3D textures which are not available on all graphics hardware.

The paper [Chang et al., 2008] presents a novel approach for the problem of accurately rendering translucent materials. Taking full advantage of the GPU, the method uses importance sampling for approximating incident irradiance on the surface of the model and then calculating the subsurface transport. The irradiance is sampled from the focused object's texture space which is created in a pre-processing step by parametrization of the object's triangle mesh. The authors combine local and global translucency calculations to achieve a final accurate visual result at real-time frame rates.

[Shah et al., 2009] present a different solution for rendering subsurface scattering using texture space sampling and point splatting. Using a dual representation of the scene, from the light source's perspective and the observer's perspective, the geometry is rendered twice to obtain the points where light is incident and the points visible to the observer, respectively. Irradiance is sampled from the geometry visible to the light source which is stored in a render texture, and at the center of the sample points screen aligned quads are positioned. The quads are used to shade the points visible to the observer using the diffusion approximation from [Jensen et al., 2001] or [Donner, 2006], thus light transport towards the viewer is approximated from sampled irradiance.

Chapter 3

Background

This section of the report begins with a short overview of radiometry and the basic terminology and continues with basic reflectance distribution functions. Next, volume transport in participating media is described and the diffusion approximation theory needed for rendering translucent materials. At the end of the chapter the numerical approximation methods are presented. Though this chapter contains only a small percentage of the underlying theory behind rendering translucent materials, it is enough to describe the rest of the report.

Note that this chapter presents theoretical principles and formulas that were not derived or developed during this thesis, and are the work of their respective authors. References and citations are provided. Sections 3.1, 3.2 and 3.3 are structured and presented similar to [Pharr and Humphreys, 2004]. Section 3.4 only shows the solutions of [Jensen et al., 2001] and [Donner, 2006] for subsurface light transport. The derivation and a thorough description of the solution can be found in the original papers.

Symbol	Meaning	Description
x	geometric point	
n	geometric normal	normalised vector
ω	direction	normalised vector
$(n \cdot \omega)$	dot product between two vectors	normalised vectors
A	area	m^2
Φ	radiant flux	J/s (W)
E	Irradiance	W/m^2
I	Intensity	W/sr^{-1}
L	Radiance	$\mathrm{W}/(sr^{-1}m^{-2})$
α	albedo	
α'	reduced albedo	
η	index of refraction	
g	asymmetry coefficient	
σ_a	absorption coefficient	
σ_s	scattering coefficient	
σ_t	extinction coefficient	
σ'_s	reduced scattering coefficient	
σ'_t	reduced extinction coefficient	
i	subscript	denotes incoming direction or incidence point
o	subscript	denotes outgoing direction or exit point

Table 3.1: Table of frequently used symbols and notations.

3.1 Basic Radiometry

Radiometry is the study that measures electromagnetic radiation and provides a set of principles and mathematical tools used to describe light propagation and reflectance. The study of energy transfer in the form of electromagnetic radiation is called *radiative transfer* and describes by mathematical equations the propagation of radiation through a medium. Radiative transfer is defined at the geometric optics abstraction level where light propagation is described

in terms of "rays" and successfully theoretizes light interaction with objects much larger than light's wavelength. This means it accounts for scattering, absorption and emission of light but not for phenomena like diffraction and interference [Pharr and Humphreys, 2004].

The basic quantities in radiometry that are important to graphics rendering are: *radiant flux*, *irradiance*, *intensity* and *radiance*.

Radiant Flux

Radiant flux is the total amount of energy passing through a surface area per unit time. It is normally represented as Φ and it is measured in joules per second or watts (J/s or W). In figure 3.1 the total amount of radiant flux (power) arriving on both of the spheres is equal, though locally, per area unit it is different [Pharr and Humphreys, 2004].

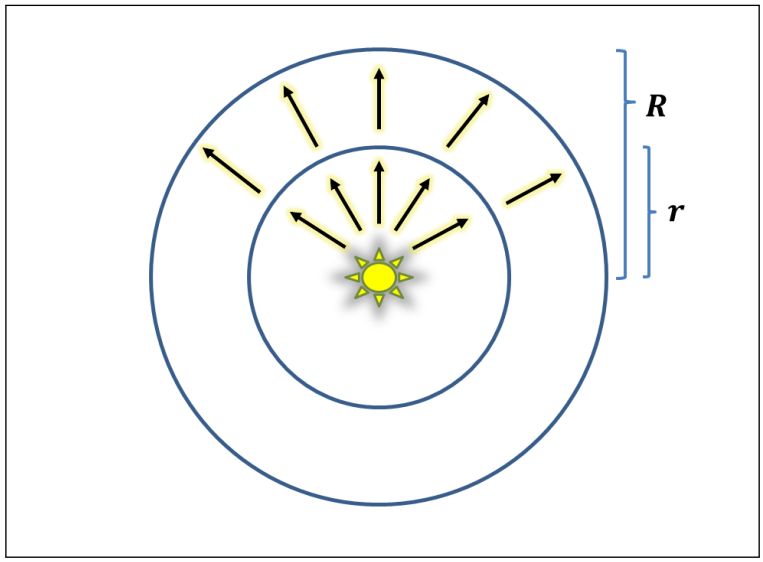

Figure 3.1: Radiant Flux and Irradiance. Two concentric spheres with different radii and a point-light source placed in their center. Radiant flux is equal on both of them, while Irradiance is greater on the smaller, closer sphere.

Irradiance

Irradiance, denoted by E, is the area density of flux arriving at a surface and is measured in watts per square meter: W/m^2. For the sphere with radius r lit by a point light source in figure 3.1 the irradiance is calculated as:

$$E = \frac{\Phi}{4\pi r^2}$$

Note that for the larger sphere in figure 3.1 the irradiance at a local point is less than the irradiance on the smaller sphere. In more general terms irradiance is the differential flux over the differential area:

$$E = \frac{d\Phi}{dA}$$

The converse measure of irradiance is *exitant radiance* which is the density of flux leaving the surface area [Pharr and Humphreys, 2004].

Intensity

Before defining *intensity* the notion of *solid angle* needs to be introduced as the extension of planar angles to an angle on the unit sphere. The solid angle is the total area of an object projected on the unit sphere (see figure 3.2), it is measured in *steradians* and it ranges to 4π (when the object is fully covering the unit sphere).

The set of vectors with origin in a point p and oriented towards the unit sphere centered at p will be denoted by ω and they will be considered normalised.

Having these definitions, the intensity can be calculated as flux density per solid angle [Pharr and Humphreys, 2004]:

$$I = \frac{d\Phi}{d\omega}$$

Radiance

Radiance is the quantity of the flux density per solid angle per projected surface area. The projected surface area is the area of the corresponding perpendicular surface to ω [Pharr and Humphreys, 2004](see figure 3.3).

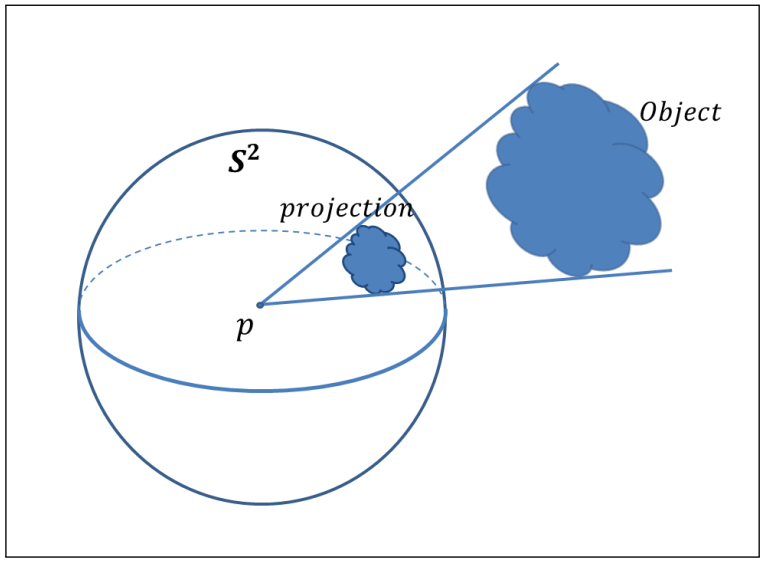

Figure 3.2: Solid angle. The solid angle is the set of directions that surround the object, and is equal to the area on the unit sphere of the projected object.

$$L = \frac{\mathrm{d}\Phi}{\mathrm{d}\omega \mathrm{d}A^{\perp}}$$

3.2 Surface Reflection

When light interacts with an object, at the separation surface between the two media, two phenomena are likely to appear: transmission – when light enters the new medium and reflection – when light scatters on the surface and is reflected back into the first medium. Reflection is characterized by two varying attributes: the spectral distribution and the directional distribution of the reflected light.

Translucent materials exhibit a more complex light–medium interaction: light passes the separation surface, scatters inside and it is absorbed or exits again back into the first medium at another location.

Reflection from a separation surface can be therefore described by two independent mechanisms: bidirectional reflectance distribution function, henceforth de-

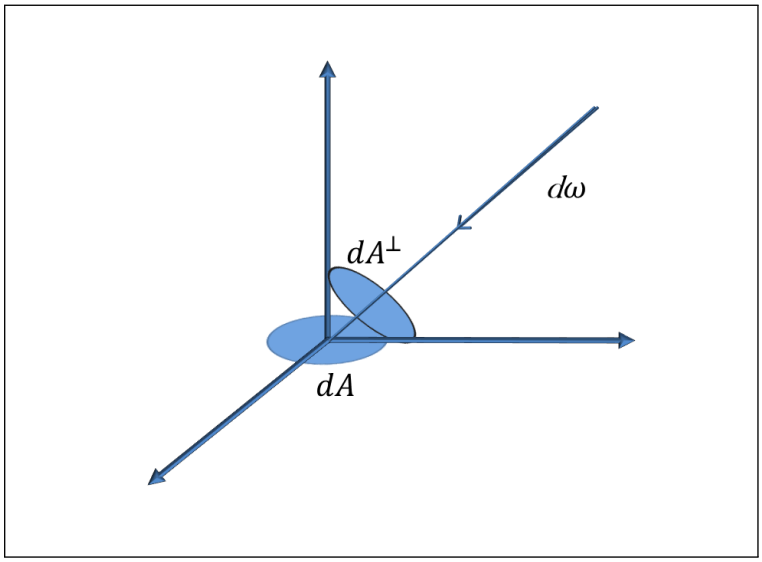

Figure 3.3: Projected Area. To calculate the radiance, the surface is projected on the plane perpendicular to the incoming direction.

noted BRDF and bidirectional surface−scattering reflectance distribution function, denoted BSSRDF in the remaining parts of the report.

3.2.1 Bidirectional Reflection Distribution Function

Surfaces that reflect light without surface scattering can be generically divided into four categories depending on the reflection pattern [Pharr and Humphreys, 2004]: diffuse, glossy specular, perfect specular and retro-reflective surfaces (see figure 3.4). Diffuse surfaces (figure 3.4 a))reflect light equally in all directions in the positive hemisphere (oriented in the normal direction) centered at the incidence point. Glossy specular surfaces (figure 3.4 b)) reflect light in a preferred set of directions which depend on the incidence angle. Such surfaces are plastic, metals or surfaces covered in glossy paint and their distinctive feature is that they show the surrounding environment blurred. Perfect specular surfaces(figure 3.4 c)), on the other hand, reflect light in a single direction, and the environment is shown on them similar to a mirror reflection. Retro-reflective surfaces reflect light in a preferential set of directions that are opposite to the incidence direction (figure 3.4 d)). Examples include velvet and road signs.

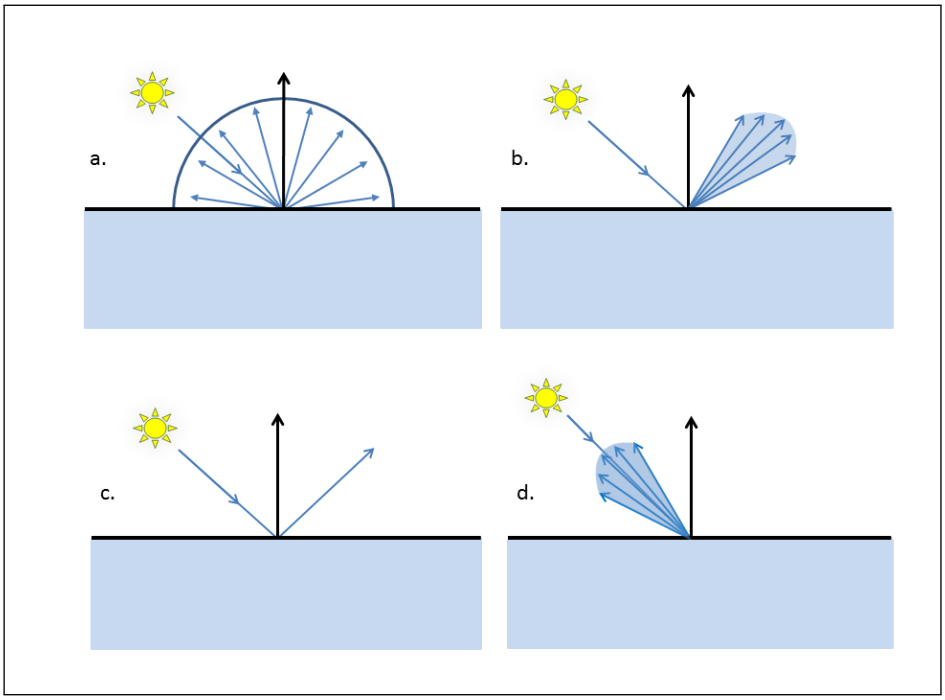

Figure 3.4: Reflection Patterns. The four basic reflection types: a) diffuse; b) glossy specular; c) perfectly specular; d) retro-reflection.

The BRDF is a function describing how light is reflected off an opaque surface and it was first introduced in [Nicodemus et al., 1977]. It is a multi dimensional function depending on material properties of the shaded point and incidence point and the geometrical properties of the scene (see figure 3.5):

x_i	the point where light is incident
n	the surface normal at the incidence point
ω_o	the outgoing direction
ω_i	the incoming direction

Numerically, the BRDF is equal to the ratio between the differential outgoing radiance at point x_i in ω_o direction and the differential irradiance at point x_i from ω_i direction:

$$f_r(x_i, \omega_o, \omega_i) = \frac{\mathrm{d}L_o(x_i, \omega_o)}{\mathrm{d}E(x_i, \omega_i)}.$$

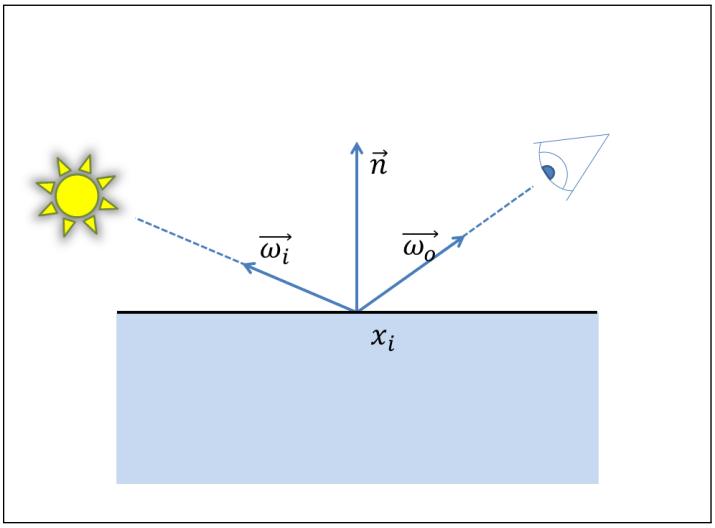

Figure 3.5: Bidirectional Reflectance Distribution Function. The BRDF and its dependencies.

The incoming differential irradiance at x is equal to:

$$\mathrm{d}E(x_i, \omega_i) = L_i(x_i, \omega_i) \cos\,\theta_i \,\mathrm{d}\omega_i,$$

where $\cos\theta_i = n \cdot \omega_i$, therefore the BRDF can be expressed as

$$f_r(x_i, \omega_o, \omega_i) = \frac{\mathrm{d}L_o(x_i, \omega_o)}{L_i(x_i, \omega_i) \cos\,\theta_i \,\mathrm{d}\omega_i}. \tag{3.1}$$

The outgoing radiance at point x_o in direction ω_o due to incoming irradiance at point x_i can be calculated by integrating over all ω_i directions in the surrounding unit sphere :

$$L_o(x_i, \omega_o) = \int_{S^2} f_r(x_i, \omega_o, \omega_i) L_i(x_i, \omega_i) \cos\,\theta_i \mathrm{d}\omega_i \tag{3.2}$$

Lambertian BRDF

One of the simplest BRDFs is the Lambertian distribution:

$$f_r(x_i, \omega_o, \omega_i) = \frac{\alpha}{\pi},$$

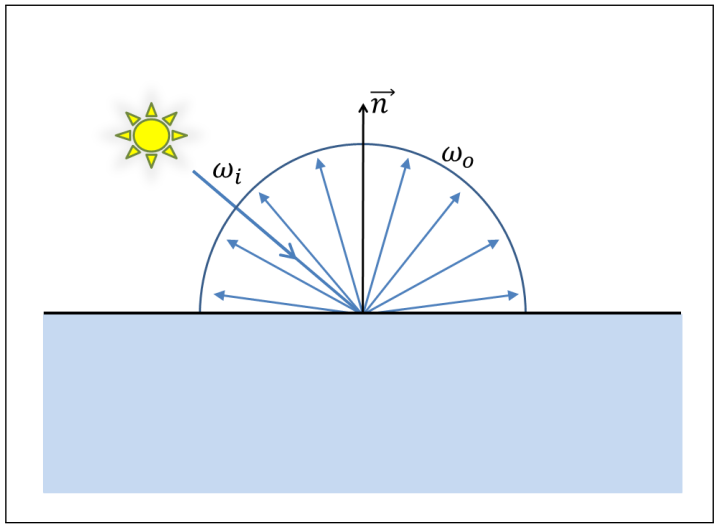

Figure 3.6: Lambertian BRDF. Light is reflected equally in all directions.

where α is the *albedo* of the surface and denotes the reflecting power of the surface calculated as the ration between the outgoing radiance and the incoming radiance. Therefore a lambertian material will reflect incoming light equally in all directions and independent on the viewing angle [Pharr and Humphreys, 2004](see figure 3.6).
There are no perfectly lambertian surfaces found in nature. Matte paper is a material that resembles lambertian reflectance except at small viewing angles, when the material shows a glossy reflection [Pharr and Humphreys, 2004]. *Spectralon* is the material with the highest albedo of any known materials over the visible spectrum (over 99%) [Georgiev and Butler, 2007], fresh snow reflects approximately 80%-90% of the incoming radiation [Markvart and Castaner, 2003] while soil reflects under 20% [Markvart and Castaner, 2003].

Specular BRDFS

Materials usually do not exhibit a single type of surface reflection, but in fact show a mixture of the four reflection types described above. Phong proposed an empirical model in [Phong, 1975], based on the observation that illumination from a surface is the sum of three elements: the ambient, diffuse and specular

reflection. Thus, the intensity reflected at a point is

$$\begin{aligned} I &= I_a + I_d + I_s \\ &= L_a + f_d L_d + f_s L_s \end{aligned}$$

where

L_a	is the ambient illumination term
L_d	is the diffuse illumination term
L_s	is the specular illumination term
f_d	is the diffuse BRDF
f_s	is the specular BRDF

A model for the specular BRDF was proposed by [Blinn and Newell, 1976] as an improvement to the Phong model. Blinn's model is based on the observation that light reflected by a perfectly specular (mirror) surface will be visible to the observer only if the surface normal points in the direction between the viewing and the incidence direction. The median direction is called *half-vector*, denoted by H (see figure 3.7), and it is numerically equal to:

$$H = \frac{L + E}{\|L + E\|}$$

The deviation of the half-vector from the surface normal (calculated as the cosine between the two vectors) will measure the fall-off in the reflection. The degree of sharpness of the highlight is simulated by taking the cosine to a number in connection to the shininess of the material. Thus, the specular BRDF proposed by Blinn is:

$$f_s = (N \cdot H)^s$$

where s is a constant value specific for the shaded material and represents the shininess. Note that the normal N and the half-vector H are normalised vectors, so their dot product is equal to the cosine of the angle between them. When the shininess term tends to infinity, the cosine term will tend to zero, unless it is equal to one:

$$\lim_{s \to \infty} f_t = \begin{cases} 1 & \text{if } (N \cdot H) = 1 \\ 0 & \text{if } (N \cdot H) < 1 \end{cases}$$

The reflection from a surface will tend to become perfectly specular as the shininess term becomes very large.

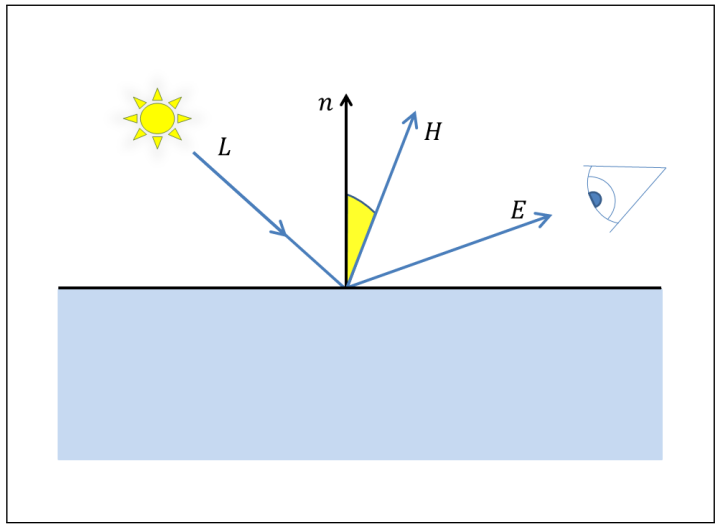

Figure 3.7: The Half Vector. The median direction between light's incidence angle and the viewing direction is called the Half Vector. The deviation of the half vector from the surface normal is measured by the cosine of the angle between them.

3.2.2 Bidirectional Surface-Scattering Reflection Distribution Function

The BSSRDF is a function describing the subsurface light transport in translucent materials as the ration between the differential radiance at incidence point x_i and the differential irradiance at exit point x_o. The phenomenon is shown in figure 3.8.

The analytical equation that describes the BSSRDF is

$$S(\mathrm{x}_o, \omega_o, \mathrm{x}_i, \omega_i) = \frac{\mathrm{d}L_o(\mathrm{x}_o, \omega_o)}{\mathrm{d}\Phi(\mathrm{x}_i, \omega_i)}$$

If the incidence point and the exit point are the same ($\mathrm{x}_i \equiv \mathrm{x}_o$) the BSSRDF is reduced to the BRDF (see equation 3.1).

The scattering equation(3.3) for the BSSRDF is now integrated over the surface

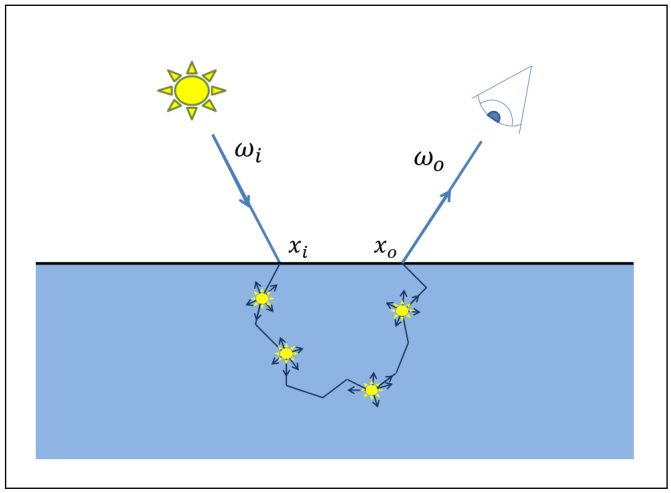

Figure 3.8: Subsurface Scattering. Light enters the medium and scatters inside, to be absorbed inside or exit at another location.

area as well to account for radiance from all incoming incidence points x_i [Pharr and Humphreys, 2004]:

$$L_o(x_o, \omega_o) = \int_A \int_{S^2} S(\mathrm{x}_o, \omega_o, \mathrm{x}_i, \omega_i) L_i(\mathrm{x}, \omega_i) \cos\,\theta_i \mathrm{d}\omega_i \mathrm{d}A \tag{3.3}$$

3.2.3 Fresnel Reflectance

When light hits a separation surface a fraction of light is reflected off the surface while the rest is transmitted through. The amount of reflected light is described by the Fresnel equations [Pharr and Humphreys, 2004]. The Fresnel equations are different depending on the conductivity of the materials: dielectrics (materials that do not conduct electricity - glass) and conductors (metals). Also the Fresnel reflectance is dependent on the polarization of light but a typical approximation in computer graphics is to consider light as being unpolarized. Therefore the result is the average of the squares of the parallel and perpendicular polarization terms [Pharr and Humphreys, 2004].

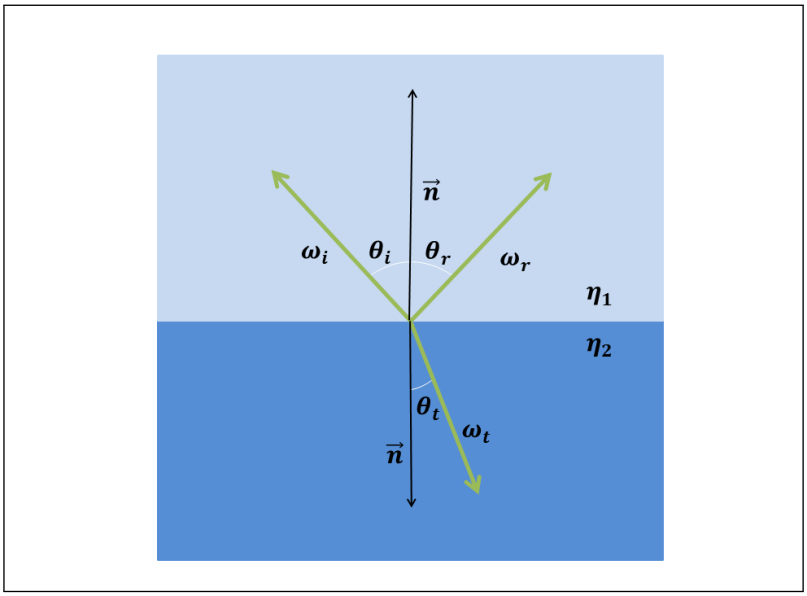

Figure 3.9: Reflection and Transmittance at the separation surface. At the incidence point, depending on the incident direction and the indices of refraction, a fraction of light is reflected while the rest is transmitted in the directions shown in the figure.

Fresnel Equations for Dielectrics

In order to calculate the reflectance at the separation surface between two media the following need to be know: incoming light direction ω_i, transmitted light direction ω_t, the surface normal n and the indices of refraction of the two media. For the following equations it is assumed that both mentioned directions are oriented outwards and the normal is flipped to be on the same side as each of them (see figure 3.9). Therefore the angles θ_i and θ_t are between $\left[0, \frac{\pi}{2}\right]$. θ_t can be found using Snell's law [Rashed, 1990]:

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{\eta_t}{\eta_i}$$

The Fresnel terms for each type of polarization are:

$$r_{\parallel} = \frac{\eta_t \cos(\theta_i) - \eta_i \cos(\theta_t)}{\eta_t \cos(\theta_i) + \eta_i \cos(\theta_t)}$$

$$r_\perp = \frac{\eta_i \cos(\theta_i) - \eta_t \cos(\theta_t)}{\eta_i \cos(\theta_i) + \eta_t \cos(\theta_t)}$$

The average of the squares of each independent term is the Fresnel reflectance for dielectrics for unpolarized light:

$$F_r = \frac{1}{2}\left(r_\parallel^2 + r_\perp^2\right).$$

The Fresnel transmittance term, describing the amount of light transmitted through the separation surface, can be easily calculated taking into account the law of conservation of energy [Pharr and Humphreys, 2004]. This is applied to both conductors and dielectrics:

$$F_t = 1 - F_r$$

3.3 Light Transport in Volumes

The collection of particles that make up a volume and alter light as it passes through it is referred to as *participating media*. Subsurface scattering is possible to render only by considering the influence of the participating media on the incoming radiation. Besides translucent materials, other phenomena that exhibit light interaction with participating media are atmospheric haze, fog, clouds, smoke or light passing through cloudy water.
The medium's properties that influence light's propagation are the *scattering coefficient*, the *absorption coefficient* and the *phase function*. The next subsection will detail these three measures and the processes that take place along a beam of light inside participating media.

3.3.1 Volume Light Transport Processes

Inside participating media a light beam can lose radiance due to absorption and the transfer of light energy to another type of energy, can gain energy from emission of light energy by luminous particles or can scatter in other directions by colliding with the particles. These processes may be constant throughout the volume (*homogeneous* – in solids like wax or marble) or may vary in space and time (*inhomogeneous* – clouds, smoke or cloudy water).

Absorption

Denoted by σ_a, absorption is the rate of radiance attenuation per meter. In homogeneous materials absorption is constant, but otherwise it may vary with position and light direction inside the volume. Absorption is also usually a wavelength dependent measure: for example green jade absorbs most of the blue and red wavelengths and therefore it's green appearence. Another example of absorption in participating media are the shadows cast by clouds: incoming sun light is scattered and absorbed does not reach the surface below.

Scattering

While it travels inside the participating media a light beam interacts (collides) with the particles in the medium and scatters in different directions losing radiance. The process is called out-scattering and the rate of scattering per unit distance is denoted by σ_s. Light that out-scatters from a beam of light can continue in the direction of another beam, thus adding radiance. This is called in-scattering.

It is usefull to define also the extinction (or attenuation) coefficient $\sigma_t = \sigma_a + \sigma_s$ to measure the total reduction in radiance due to absorption and out-scattering.

Emission

Emission increases the radiation along a ray from small scale interactions with luminescent particles. These particles chemically produce radiation (*chemiluminescence*) or absorb specific wavelength radiation and emit back radiation at different wavelengths - in the visible or non visible spectrum. Emission is seen, for example, in living creatures (fireflies, jelly fish) or in fluorescent minerals which absorb ultraviolet light and emit visible light (figure 3.10).

Figure 3.10: Emission. Light is absorbed inside the medium and emitted at different wavelengths. Source: http://en.wikipedia.org/wiki/Fluorescence

3.3.2 Phase Functions

The probability distribution that light will scatter and change its direction from ω to a certain direction ω' is given by the phase function $p(\omega \rightarrow \omega')$. If the medium scatters light equally in all directions with equal probability then its phase function is [Pharr and Humphreys, 2004]:

$$p_{isotropic}(\omega \rightarrow \omega') = \frac{1}{4\pi}$$

This is because the phase function is a probability density function and it is normalised over the domain:

$$\int_{4\pi} p(\omega \rightarrow \omega') d\omega' = 1$$

The phase function defines another characteristic of an isotropic scattering medium: the asymmetry parameter g which is [Pharr and Humphreys, 2004]:

$$\begin{aligned} g &= \int_{4\pi} p(\omega \rightarrow \omega')(\omega \cdot \omega') d\omega' \\ &= 2\pi \int_0^{\pi} p(\cos(\theta)) \cos\theta \sin\theta d\theta \end{aligned}$$

3.4 Diffusion Approximation

The first part of this section presents a model for the BSSRDF as presented by [Jensen et al., 2001] based on a twin light source pair (dipole). Later the multipole approximation is described as presented by [Donner and Jensen, 2005]. These BSSRDFs are dedicated for rendering highly scattering homogeneous translucent materials so, unless specified otherwise, the rest of the report will refer only to this subset of translucent materials.

The full derivation of the formulas in this section can be found in the original articles [Jensen et al., 2001, Donner and Jensen, 2005] or in Craig Donner's PhD thesis [Donner, 2006].

3.4.1 Dipole Approximation

The dipole approximation [Jensen et al., 2001] is used to approximate subsurface transport in optically thick media. The solution is to replace the original light source with a pair of light sources (called dipole sources) placed above and below the separation surface. The two light sources will account for light scattering below the surface. Figure 3.11 shows the setting in which light-surface interaction will be considered. Incoming radiance from direction ω_i is incident at x_i and penetrates through the separation surface. Inside the medium, light scatters (as presented in figure 3.8) and it exits at x_o in direction ω_o. The distance between the incidence and exit points is denoted r.

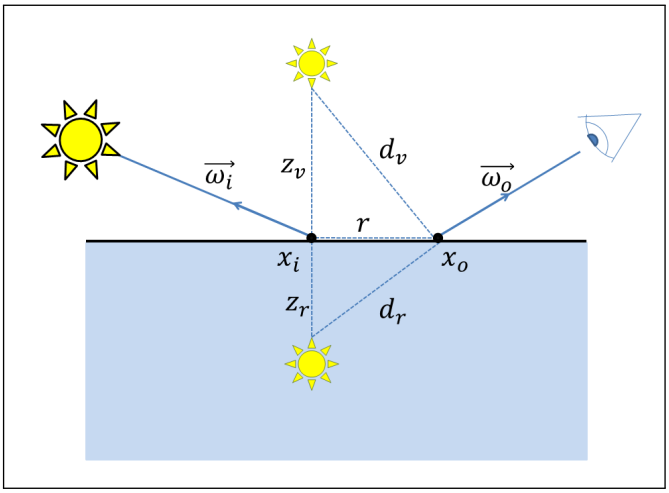

Figure 3.11: Dipole Approximation. The light source is replaced by a pair of light sources placed above and below the separation surface.

The two light sources are placed at distances z_v and z_r from the separation surface [Jensen et al., 2001]:

$$\begin{aligned} z_r &= \frac{1}{\sigma'_t} \\ z_v &= z_r(1 + \frac{4A}{3}) \end{aligned}$$

where σ'_s is the reduced scattering coefficient and A is the Groenhuis parameter [Groenhuis et al., 1983] :

$$A = \frac{1 + F_{dr}}{1 - F_{dr}}$$

In the previous definition, F_{dr} is the average Fresnel reflectance:

$$F_{dr} = \int_{2\pi} F_r(\eta, n \cdot \omega_i)(n \cdot \omega_i) \mathrm{d}\omega_i$$

where n is the normal at the incidence point and η is the relative refraction index of the two media.

The average diffuse Fresnel reflectance can be calculated analytically, but it is commonly approximated with a polynomial expansion [Egan and Hilgeman, 1979]:

$$F_{dr} \backsimeq= \begin{cases} -0.4399 + \frac{0.7099}{\eta} - \frac{0.3319}{\eta^2} + \frac{0.0636}{\eta^3}, & \eta < 1 \\ -\frac{1.4399}{\eta^2} + \frac{0.7099}{\eta} + 0.6681 + 0.0636\eta, & \eta > 1 \end{cases}$$

The distances to the exit points are denoted d_v and d_r (figure 3.11) and are numerically equal to:

$$\begin{aligned} d_r &= \sqrt{r^2 + z_r^2} \\ d_v &= \sqrt{r^2 + z_v^2} \end{aligned}$$

where r is the euclidean distance between x_i and x_o.

The total reflectance at outgoing point x_o is calculated as the sum from the contribution of each dipole source:

$$R_d(r) = \frac{\alpha'}{4\pi}\left[(\sigma_{tr}d_r + 1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3} + (1 + \frac{4A}{3})(\sigma_{tr}d_v + 1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right]$$

where α' is the reduced albedo, σ_{tr} is the effective transport and σ_t' the reduced extinction coefficient [Jensen et al., 2001]:

$$\begin{aligned} \sigma_s' &= (1-g)\sigma_s \\ \sigma_t' &= \sigma_a + \sigma_s' \\ \alpha' &= \frac{\sigma_s'}{\sigma_t'} \\ \sigma_{tr} &= \sqrt{3\sigma_t'\sigma_a} \end{aligned}$$

Note that the diffuse reflectance term (R_d) is a function of the distance between the x_i and x_o points.

The BSSRDF is calculated by multiplying the diffuse reflectance term with the Fresnel reflection for the incoming and outgoing ray directions.

$$S(\mathrm{x}_o, \omega_o, \mathrm{x}_i, \omega_i) = \frac{1}{\pi} F_t(\eta, \omega_i) R_d(\|x_i - x_o\|) F_t(\eta, \omega_o). \tag{3.4}$$

3.4.2 Multipole Approximation

Opposed to the dipole approximation, in the multipole approximation [Donner, 2006] the point light source is replaced by an array of dipoles (see figure 3.12).

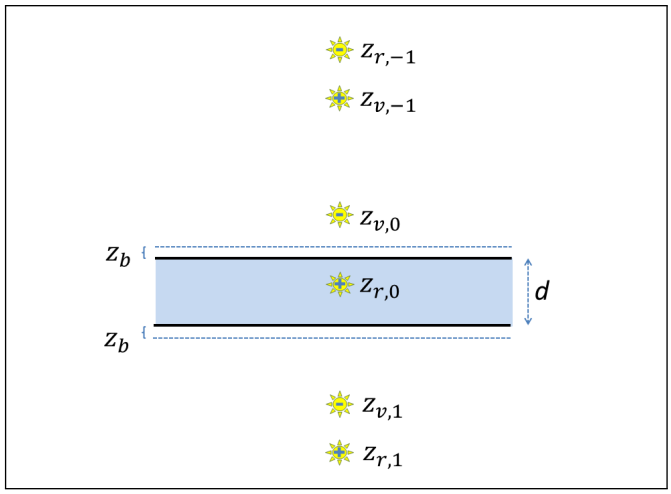

Figure 3.12: Multipole Approximation. The light source is replaced by a number of dipole light sources.

Each pair of dipole sources will have its corresponding position with respect to the separation surface, denoted $z_{v,i}$ and $z_{r,i}$ for the dipole pair i. When the relative index of refraction is equal to 1, the z-coordinates for the each dipole sources are given by [Donner, 2006]:

$$\begin{aligned} z_{r,i} &= 2i(d + 2z_b) + l \\ z_{v,i} &= 2i(d + 2z_b) - l - 2z_b \end{aligned}$$

where z_b is given by:

$$\begin{aligned} z_b &= 2AD \\ A &= \frac{1+F_{dr}}{1-F_{dr}} \\ D &= \frac{1}{3\sigma_t'} \end{aligned}$$

The reflectance and transmittance profiles are calculated by summing up the individual contribution from each of the n dipole pairs:

$$R(r,d) = \sum_{i=-n}^{i=n} \left(\frac{\alpha' z_{r,i}(1+\sigma_{tr}d_{r,i})e^{-\sigma_{tr}d_{r,i}}}{4\pi d_{r,i}^3} - \frac{\alpha' z_{v,i}(1+\sigma_{tr}d_{v,i})e^{-\sigma_{tr}d_{v,i}}}{4\pi d_{v,i}^3} \right) \tag{3.5}$$

$$T(r,d) = \sum_{i=-n}^{i=n} \left(\frac{\alpha'(d-z_{r,i})(1+\sigma_{tr}d_{r,i})e^{-\sigma_{tr}d_{r,i}}}{4\pi d_{r,i}^3} - \frac{\alpha'(d-z_{v,i})(1+\sigma_{tr}d_{v,i})e^{-\sigma_{tr}d_{v,i}}}{4\pi d_{v,i}^3} \right) \tag{3.6}$$

where $d_{r,i} = \sqrt{r^2 + z_{r,i}^2}$ and $d_{v,i} = \sqrt{r^2 + z_{v,i}^2}$.

Note that the diffuse reflection and transmission terms for the multipole approximation are functions of the distance between the x_i and x_o points and of d, the thickness of the slab (see figure 3.12).

Multilayered Materials

Donner [Donner and Jensen, 2005] 3.6 focuses also on calculating the profiles for multi-layered materials. The constraints set at the start of the chapter (highly scattering homogeneous materials) are still considered.

Considering an slab of material covered uniformly with another layer (both being highly scattering and homogeneous) the transmittance profile for both layers can be calculated by convolving each layer's profile [Donner and Jensen, 2005]:

$$T_{12}(r) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} T_1(r')T_2(r'')dx'dy' = T_1(r) * T_2(r); \tag{3.7}$$

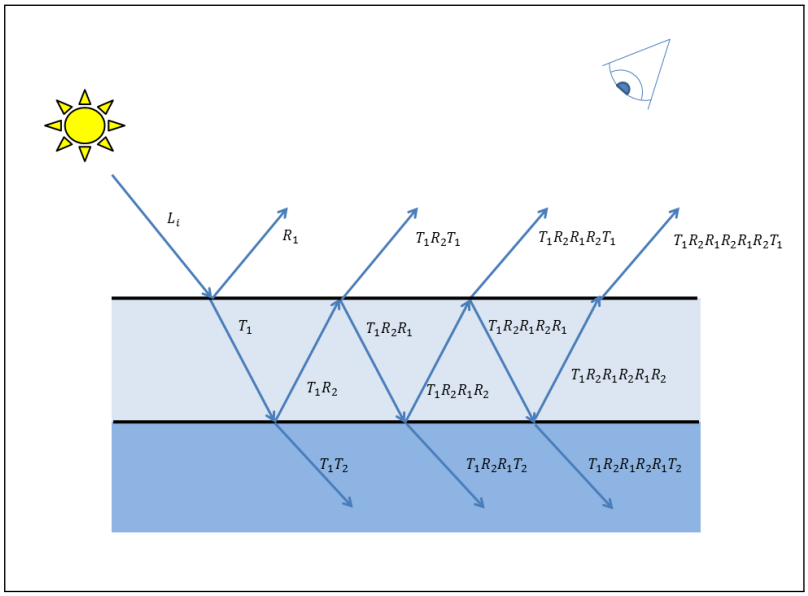

Figure 3.13: Light Reflection and Transmission in multiple layers. Between the two separation surfaces light scatters and suffers multiple reflections, and finally is transmitted in the second underlying layer or back into the first medium.

where T_1 and T_2 are each layer's transmittance profile and $r' = \sqrt{x'^2 + y'^2}$. The previous equation would be true only if no light is reflected off the second layer. The exact phenomenon is shown in figure 3.13 and the correct transmittance profile for both layers is:

$$T_{12} = T_1 * T_2 + T_1 * R_2 * R_1 * T_2 + T_1 * R_2 * R_1 * R_2 * R_1 * T_2 + \ldots \quad (3.8)$$

[Donner and Jensen, 2005] propose to calculate the above mentioned convolutions in frequency domain where they will be simplified to element-wise multiplications. Considering $\mathcal{R}$ and $\mathcal{T}$ to be the Fourier transformed reflectance and transmittance profiles, equation 3.8 becomes:

$$\begin{aligned} \mathcal{T}_{12} = &\quad \mathcal{T}_1\mathcal{T}_2 + \mathcal{T}_1\mathcal{R}_2\mathcal{R}_1\mathcal{T}_2 + \mathcal{T}_1\mathcal{R}_2\mathcal{R}_1\mathcal{R}_2\mathcal{R}_1\mathcal{T}_2 + \ldots \\ = &\quad \mathcal{T}_1\mathcal{T}_2(1 + \mathcal{R}_2\mathcal{R}_1 + (\mathcal{R}_2\mathcal{R}_1)^2 + (\mathcal{R}_2\mathcal{R}_1)^3 + \ldots) \end{aligned} \quad (3.9)$$

The sum of products on the right side can be considered a geometric progression and equation 3.11 can be reduced to a simpler form:

$$\mathcal{T}_{12} = \mathcal{T}_1\mathcal{T}_2\left(\frac{1-(\mathcal{R}_2\mathcal{R}_1)^n}{1-\mathcal{R}_2\mathcal{R}_1}\right)$$

Assuming $\mathcal{R}_2\mathcal{R}_1 < 1$ then $(\mathcal{R}_2\mathcal{R}_1)^n$ will converge to 0, and the formula for the two-layered transmittance profile is:

$$\mathcal{T}_{12} = \frac{\mathcal{T}_1\mathcal{T}_2}{1-\mathcal{R}_2\mathcal{R}_1} \tag{3.10}$$

The same steps can be followed to calculate the total reflectance from two layered materials (consider figure 3.13):

$$\begin{aligned}\mathcal{R}_{12} = &\ \mathcal{R}_1 + \mathcal{T}_1\mathcal{R}_2\mathcal{T}_1 + \mathcal{T}_1\mathcal{R}_2\mathcal{R}_1\mathcal{R}_2\mathcal{T}_1 + \mathcal{T}_1\mathcal{R}_2\mathcal{R}_1\mathcal{R}_2\mathcal{R}_1\mathcal{R}_2\mathcal{T}_1 + \ldots \\ = &\ \mathcal{R}_1 + \mathcal{T}_1\mathcal{R}_2\mathcal{T}_1(1 + \mathcal{R}_2\mathcal{R}_1 + (\mathcal{R}_2\mathcal{R}_1)^2 + (\mathcal{R}_2\mathcal{R}_1)^3 + \ldots) \\ = &\ \mathcal{R}_1 + \mathcal{T}_1\mathcal{R}_2\mathcal{T}_1\left(\tfrac{1-(\mathcal{R}_2\mathcal{R}_1)^n}{1-\mathcal{R}_2\mathcal{R}_1}\right)\end{aligned} \tag{3.11}$$

With the same assumption as in 3.10 the reflectance profile of two layers is:

$$\mathcal{R}_{12} = \mathcal{R}_1 + \frac{\mathcal{T}_1\mathcal{R}_2\mathcal{T}_1}{1-\mathcal{R}_2\mathcal{R}_1} \tag{3.12}$$

Note that the profiles calculated above are still in frequency domain. The real response is obtained by applying the inverse Fourier transform to the profiles. The frequency domain formulas 3.10 and 3.12 are identical to the Kubelka-Munk equations [Kubelka, 1954].

3.5 Numerical Integration

As presented in section 3.1, the rendering equations (for example equations 3.3 and 3.2) are mainly integral equations. These integrals often do not have analytical solutions, therefore a numerical approach is needed to approximate the results.

For single dimensional integrals the trapezoidal method or Simpson's rule can be used to approximate their value. For multidimensional and discontinuous functions a better approach is to use Monte Carlo integration methods, based on random numbers.

3.5.1 Discrete Integration

For one-dimensional functions, the definition for the integral was first introduced with the Riemann equation. Considering the closed interval $[a, b]$ and a *tagged partition* of it defined by the points x_i and t_j:

$$a = x_0 \leq t_1 \leq x_1 \leq t_2 \leq x_2 \leq \cdots \leq x_{n-1} \leq t_n \leq x_n = b$$

the integral of a function f over $[a, b]$ is considered to be the Riemann sum [Anton, 1998]:

$$\int_a^b f(x)dx \approxeq \sum_{i=1}^{N} f(t_i)\triangle_i \tag{3.13}$$

where $\triangle_i$ is defined as $x_i - x_{i-1}$. Infinite partitions (when N tends to infinity) would allow the value of the sum to converge to the value of the integral.

A more practical approximation is the rectangle method. For a function f defined in $[a, b]$, the interval is subdivided into N equal parts of length $h = \frac{b-a}{N}$. Each subinterval has a defined area under the function's graph which can be approximated to the area of a rectangle, and thus much easier to calculate. The area under the whole function (the value of the integral) is therefore approximated with the sum of the areas of the subintervals (see picture 3.14):

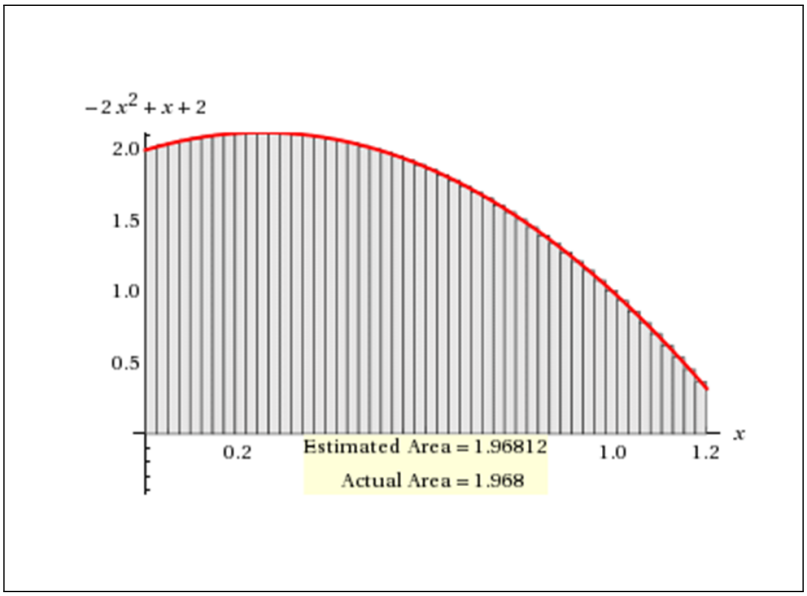

Figure 3.14: The Rectangle Rule. The area under the function is divided into a series of rectangles, for which the area is easily calculated. The integral is approximated to the sum of the areas of the rectangles. Obtained using http://mathworld.wolfram.com/

Another approach is Simpson's Composite rule, based on subdividing the interval into an even number n of subintervals and applying Simpson's Rule to each subinterval. It is more accurate to apply Simpson's rule rather than the rectangle Rule for each subinterval. The equation for Simpson's Composite Rule is:

$$\int_a^b f(x)dx \approxeq \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right] \quad (3.14)$$

The approach to solving a one dimensional integral can be extended to solving a multidimensional integral based in Fubini's theorem [Fubini, 1958, Thomas and Finney, 1995]:

$$\int_{A\times B} f(x,y)d(x,y) = \int_A \left(\int_B f(x,y)dy\right)dx = \int_B \left(\int_A f(x,y)dx\right)dy \quad (3.15)$$

Considering a two-dimensional integration $\int_A f(x)dA$, using Fubini's theorem and a discrete integration method (Riemann's equation) yields:

$$\int_A f(x,y)dA = \int_{x1}^{x2}\int_{y1}^{y2} f(x,y)dxdy \approxeq \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} f(x_i,y_j)\triangle_{x_i}\triangle_{y_j} \tag{3.16}$$

3.5.2 Monte Carlo Integration

Probability Density Function

For a uniform random variable x, the *probability density function* (*pdf*), denoted $p(x)$, is a function describing the probability that x takes a certain value within a defined domain. Given an interval $[a,b]$, the probability that x is inside is:

$$x(a \leq x \leq b) = \int_a^b p(x)dx$$

An important property for any *pdf* is that it is non-negative and it is integrated to 1 over the domain.

Monte Carlo Integration

For a multidimensional function $f(x_1,x_2,\ldots,x_n)$ defined on the domain $\mathcal{V}$, $\{\mathcal{V}|a_i \leq x_i \leq b_i\}$, using Monte Carlo method, the integral can be approximated to:

$$I = \int_{\mathcal{V}} f(X)dX \approxeq \mathcal{V}\frac{1}{N}\sum_{i=1}^{N} f(X_i) \tag{3.17}$$

where X is the tuple $\{x_1,x_2,\ldots,x_n\}$ and V is the dimension of the domain. For a single dimensional function defined on $[a\ldots b]$, equation 3.17 reduces to:

$$\int_a^b f(x)dx \approxeq \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \tag{3.18}$$

In equation 3.17 and 3.18 x_i and X_i are uniformly random sampled variables in the domain of definition.
If not uniformly sampled, the *pdf* can be used for importance sampling. Then, the integral is approximated by:

$$\int_a^b f(x)dx \approxeq \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Chapter 4

Method

This chapter describes the methods chosen for rendering translucent materials under both point-light and environment illumination. The details in this chapter link the theoretical background from Chapter 3 with the GPU oriented implementation presented in Chapter 5.

First, the algorithm is presented along with the motivation behind the method used. The next section describes how to acquire the material properties and how to use them in rendering. The final two sections describe the mathematical approach to solving the equation for outgoing radiance for translucent materials under point light illumination and environment illumination respectively.

4.1 Algorithm Overview

For each screen pixel written during rendering, the outgoing radiance visible to the observer is:

$$L_o(x_o, \omega_o) = \int_A \int_{S^2} S(x_o, \omega_o, x_i, \omega_i) L_i(x_i, \omega_i) \cos\theta_i \mathrm{d}\omega_i \mathrm{d}A \tag{4.1}$$

In a typical off-line ray tracing implementation, the outgoing radiance is cal-

culated by sending a ray through each pixel of the screen and checking for all ray-scene interactions. To account for scattering, the ray is traced inside the scattering medium it hits until it exits at another location or it is absorbed. The number of times a beam of light bounces inside is determined by the optical properties of the medium (see section 3.3.1). Highly scattering materials will change the trajectory of the ray more often and after shorter distances, so simulating this behavior requires a large number of steps. The computational cost of rendering highly scattering translucent materials is therefore very large.

Presented here is a different method for rendering highly scattering translucent materials. The approach is opposed to conventional path tracing: incoming radiance is sampled from the surface of the object at different positions. Centered at the sample positions, "splats" are created that are projected on the screen to shade all points visible to the user. This allows the calculation of the amount of light scattered from the sample point. The splats are screen aligned quads that link the sample position with the points being shaded. Contribution from all sample points is simulated by additively blending all the splats that overlap each pixel and so equation 4.1 is numerically approximated. In a deferred rendering stage the translucent material is rendered on the screen along with the rest of the scene, and specular highlights are added. The presented method was previously implemented and used by [Shah et al., 2009].

It is important to motivate the necessity of screen-aligned rectangles that will shade the model. In highly scattering materials, the distribution of light due to multiple scattering events, tends to become isotropic. This is because each scattering event will blur the previous distribution [Jensen and Buhler, 2002]. Considering a point light source surrounded by an infinite medium of highly scattering homogeneous material as in figure 4.1, the light distribution in it will be radially symmetric, with a high intensity (close to the light source) that dims away as the distance to the light source grows. The spherical light distribution, projected on the screen will be reduced to a circle, independent on the viewing direction. The geometry of the medium may vary, but a screen aligned quad will be enough to cover any light distribution projected on the screen. Of course, other geometric primitives can be used (for example a disc), but a screen aligned quad is much easier to handle during implementation and much more easier to be rendered.

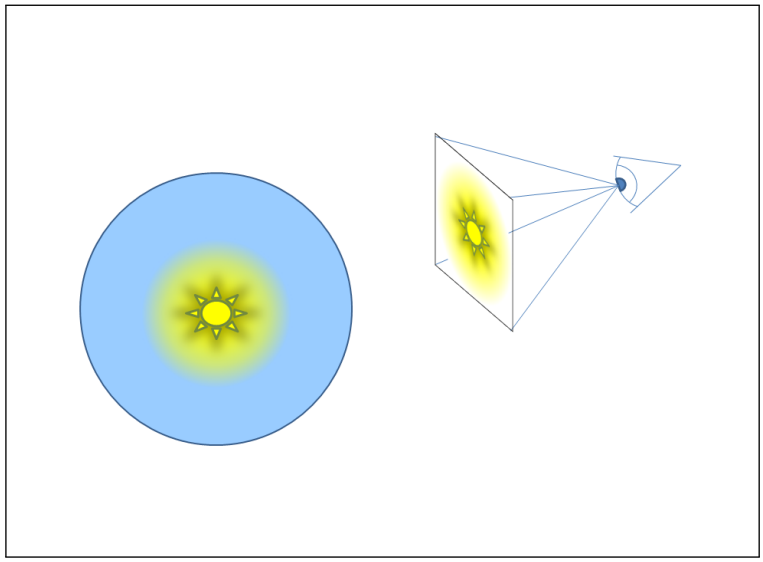

Figure 4.1: The motivation for screen aligned quads. Projected on the screen, any light distribution can be covered by a screen aligned quad.

Figure 4.1 shows that a spherical light distribution is completely covered by a screen aligned quad, but this scenario (a point light source surrounded by an infinite medium) is not often encountered. Usually, the geometry will shape the distribution, but the quads should not be dependent on the geometry being shaded. Figure 4.2 shows how two identical quads with different orientation will yield different results. The left side of the figure shows a quad oriented perpendicular to the surface normal, but its projection on the viewer's screen will not cover the whole distribution, and thus, a significant amount of scattering events can be neglected. The right side of figure 4.2 shows that if the quad is oriented perpendicular to the viewing direction, it will cover the whole distribution. Of course, the first quad (perpendicular to the surface normal) can be scaled up in order to span over a larger area, but it will introduce computational overhead, therefore screen aligned quads will be a better choice for shading. The solution to the minimum size of the quad will be detailed in section 4.3.

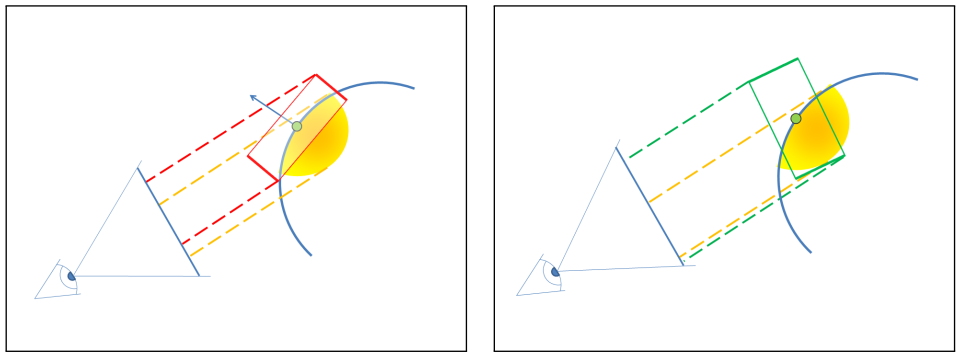

Figure 4.2: Geometry aligned quads. The left image shows a quad oriented perpendicular to the surface normal. It is visible that significant scattering events can be omitted because the screen-projected quad will not fully cover the screen-projected light distribution. The problem is solved by using screen-aligned quads, as depicted in the right image.

Equation 4.1 has two distinct sets of variables: variables that describe outgoing radiance (denoted with o subscript) and variables that describe incoming radiance (denoted with i subscript). The output points x_o are the set of points visible to the viewer, while the x_i point set holds the points where radiance may be incident. To approximate the amount of light scattered from a sample point to a shaded point (or to calculate the BSSRDF in equation 3.3) it is needed to have along with the positions the incident direction and the outgoing direction. This yields to a dual representation of the scene: from light's perspective and from the observer's perspective (see figure 4.3).

In theory, the surface of the model holds an infinity of points where light can be incident. To overcome this, in practice a subset of the points on the surface will be chosen, called in the remainder of the report *sample points*. Each point in this subset will be used to approximate the surface area in its vicinity.

Sample points are generated by sampling a texture space for point light illumination and by directly sampling the 3D model for environment illumination. Each of the methods will be described in detail in chapter 5. The point light illumination is done by splitting the light's perspective texture space into equal discrete intervals and the integration is done as presented in section 3.5.1. The detailed implementation is described in section 5.1 of Chapter 5.

For environment illumination, a Monte Carlo integration as presented in sec-

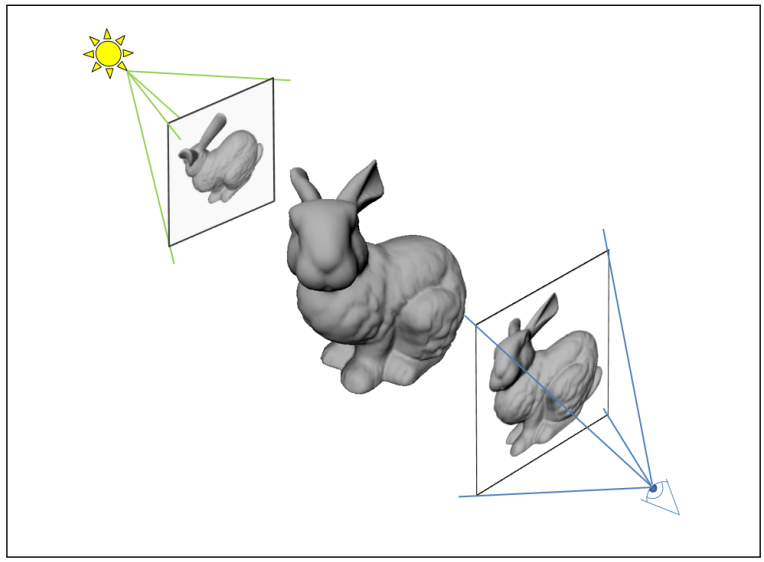

Figure 4.3: Dual representation of the scene. The scene is rendered from the eye's and light source's perspective in separate render passes.

tion 3.5.2 is used to solve equation 4.1. The sample points are randomly placed at first, and uniformly distributed in a pre-rendering step. The environment map is filtered so that a simpler representation is achieved. The detailed implementation is described in section 5.3 of Chapter 5.

4.2 Materials

To simulate scattering inside the translucent medium, the optical properties of the material need to be known. The shape in which these properties are presented varies, and the next section will describe how to handle each case. To calculate the BSSRDF described in section 3.4 (see equations 3.4.1, 3.5 and 3.6) it is needed to know the reduced scattering coefficient, absorption coefficient and the index of refraction.

4.2.1 Acquiring the needed parameters

Direct coefficients

The simplest method is to have the reduced scattering and absorption coefficients directly in RGB color channel format. [Jensen et al., 2001] have developed a method of aquiring these parameters and presented them in their paper *A practical model for light transport* [Jensen et al., 2001]:

Material	$\sigma'_s[mm^{-1}]$			$\sigma_a[mm^{-1}]$			η
	R	G	B	R	G	B	
Apple	2.29	2.39	1.97	0.0030	0.0034	0.46	1.3
Marble	2.19	2.62	3.00	0.0021	0.0041	0.0071	1.5
Potato	0.68	0.70	0.55	0.0024	0.0090	0.12	1.3
Skimmilk	0.70	1.22	1.90	0.0014	0.0025	0.0142	1.3
Spectralon	11.6	20.4	14.9	0.00	0.00	0.00	1.3
Wholemilk	2.55	3.21	3.77	0.0011	0.0024	0.014	1.3

Table 4.1: Reduced scattering and absorption coefficients and index of refraction

Obtaining the reduced scattering coefficient

Sometimes, the medium properties are described in terms of scattering coefficient, absorption coefficient, index of refraction and the mean cosine term of the phase function g(see section 3.3.2). The reduced scattering coefficient can be calculated from the scattering coefficient and the asymetry coefficient [Jensen et al., 2001]:

$$\sigma'_s = \sigma_s(1 - g) \tag{4.2}$$

Imaginary part of the refraction index

The absorption coefficient is sometimes represented as the imaginary part of the refraction index, as seen in table 4.2.

Conversion from complex index of refraction to RGB coefficients is possible with:

	refractive index (η)	
	Re	Im
R	1.5	1.16979e-7i
G	1.5	1.79447e-7i
B	1.5	2.40125e-7i

Table 4.2: Complex index of refraction for marble

$$\sigma_a(\lambda) = \frac{4\pi \cdot Im(\eta)}{\lambda}$$

where λ is the wavelength for which the absorption coefficient is calculated.

The absorption coefficients calculated for marble from complex index of refraction are shown in table 4.3. Note that the absorption coefficient values in table 4.3 and table 4.1 are identical.

	refractive index (η)	λ	absorption coefficient $[m^{-1}]$
R	1.5 + 1.16979e-7i	700.0e-9	2.1
G	1.5 + 1.79447e-7i	550.0e-9	4.1
B	1.5 + 2.40125e-7i	425.0e-9	7.1

Table 4.3: Absorption coefficients calculated from the complex index of refraction for marble

4.2.2 The Diffusion Properties

To calculate the diffuse reflectance and transmittance profiles for a material, equation 3.4.1 or 3.5 and 3.6 have to be evaluated. For this purpose, based on the optical properties of the material, the diffusion properties are calculated. The varying parameters in the above mentioned equations are the distance r and the thickness of the slab, but the diffusion properties remain constant, so it is not needed to recalculate them. A better approach would be to store and reuse them.

The scattering (σ_s) and absorption (σ_a) coefficients can be either received as input directly in RGB format, or transformed to RGB format as shown in section 4.2.1. The extinction coefficient is $\sigma_t = \sigma_a + \sigma_s$ (see section 3.3.1). The

reduced extinction coefficient σ_t' is obtained by substituting σ_s with σ_s' and the effective transport coefficient σ_{tr} and reduced albedo α' can be calculated:

$$\sigma_t' = \sigma_s' + \sigma_a$$
$$\sigma_{tr} = \sqrt{3\sigma_t'\sigma_a}$$
$$\alpha' = \frac{\sigma_s'}{\sigma_t'}$$

Two important parameters in estimating the diffusion are the distances at which to place the dipole sources. The real source is placed at distance z_r below the slab and the virtual source is placed at distance z_v above the surface of the slab (figures 3.11 and 3.12).

The dipole system will have only one pair of values that describe the positioning of the dipole source. The multipole will instead have a number of pairs of dipoles, each having it's own values for z_r and z_v.

Dipole Approximation

The first distance, z_r corresponds to the *reduced mean free path*:

$$z_r = \frac{1}{\sigma_t'}$$

The diffusion coefficient D [Ishimaru, 1978] and the correction coefficient A are used to calculate the offset of the virtual source:

$$D = \frac{1}{3\sigma_t'}$$

$$A = \frac{1-Fdr}{1+Fdr}$$

$$z_v = z_r + 4AD$$

Multipole Approximation

For the multipole diffusion approximation (equations 3.5 and 3.6) the distances z_r and z_v are individual for each dipole pair. The notation will be $z_{r,i}$ for the distance to the real source of dipole pair i, and similar for $z_{v,i}$:

$$z_{r,i} = 2i(d + A(0)D + A(d)D) + z_{r,0}$$

$$z_{v,i} = 2i(d + A(0)D + A(d)D) - z_{v,0}$$

The different refraction indices of the top and bottom layers in multi-layered materials are accounted for by the $A(0)$ term a the top of the slab and the $A(d)$ term at the bottom [Donner, 2006]. For $i = 0$, the $z_{r,0}$ and $z_{v,0}$ pair is the equivalent of the z_r and z_v pair in the dipole approximation:

$$z_{r,0} = \frac{1}{\sigma_t'}$$

$$z_{v,0} = z_{r,0} + 4AD$$

4.3 Reflectance and Transmittance Profiles

The diffusion properties are used in equations 3.4.1 or 3.5 and 3.6. For the dipole diffusion the equation is dependent only on r, the distance from the incidence position of light to the surface position where it exits the object. The distance dependent diffusion profile is shown in figure 4.4

It is important to note two characteristics in the shape of the plot:

- it is monotone and smooth
- its intensity drops quickly with the distance to values that can be neglected

The distance threshold after which the diffuse reflectance is neglectable, can be approximated by calculating where it's contribution is below a certain value [Shah et al., 2009]:

$$\frac{\int_0^\infty R_d(r)\mathrm{d}r - \int_0^{r_{max}} R_d(r)\mathrm{d}r}{\int_0^\infty R_d(r)\mathrm{d}r} < \epsilon \tag{4.3}$$

Since the plot tells that the function is smooth, a numerical integration rule as Simpson's Rule can be applied to solve the integrals. The integrals are considered to start from 0 because the distance should be a positive real number. It is

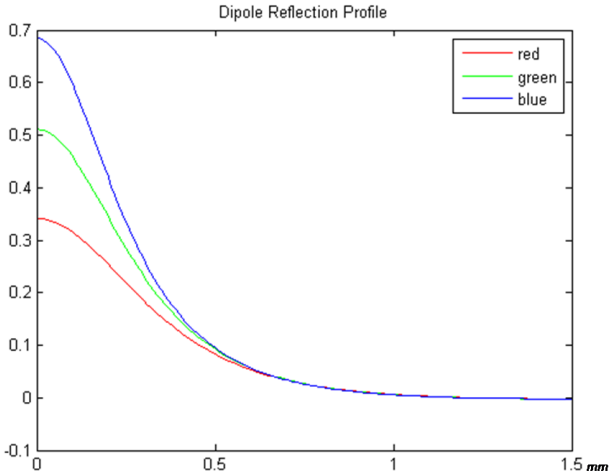

Figure 4.4: Dipole Reflectance Profile. Plot of the reflectance profile for marble calculated using the Dipole Method. The plot was done in Matlab.

not necessary to consider the first integral an infinite one, because the diffusion function $Rd(r)$ will converge to 0 very shortly.

For multipole diffusion approximation, the equation has two varying quantities: the distance between the incidence point and the exit point, and the depth of the slab (as presented in section 3.4.2). The profiles are calculated only for values of the distance in the truncated domain defined above. The depth of the slab influences the transmittance term of the multipole equation. Intuitively, thinner slabs will allow more transmittance, while for thicker slabs the transmittance term will converge to zero. The profiles for marble, at three distinct slab depths, are shown in figure 4.5.

First row shows the reflection and transmittance profiles for very thin slabs, of thickness equal to 2 mean free paths. Notice how the blue color channel is reflected more than the other channels, while the red channel is transmitted the most.

The second shows the profiles at a thickness of 10 mean free paths. Notice that the reflection is almost constant with respect to the thickness of the slab while the transmittance was highly attenuated with 2 orders of magnitude (from 0.8 to 0.0075). At 20 mean free paths (third row) the transmittance is converging to really small values of 10^{-4} order. Note that the plots are similar to the ones

in [Donner and Jensen, 2005].

The multipole diffusion approximation will introduce errors if the thickness of the slab is too small. [Donner and Jensen, 2005] use as a minimum depth a few mean fee paths. Figure 4.6 shows the transmittance profiles for marble at depths of 1.15 and 1.5 mean free paths. Note that the plots are different with an order of magnitude for a very small difference in the depth.

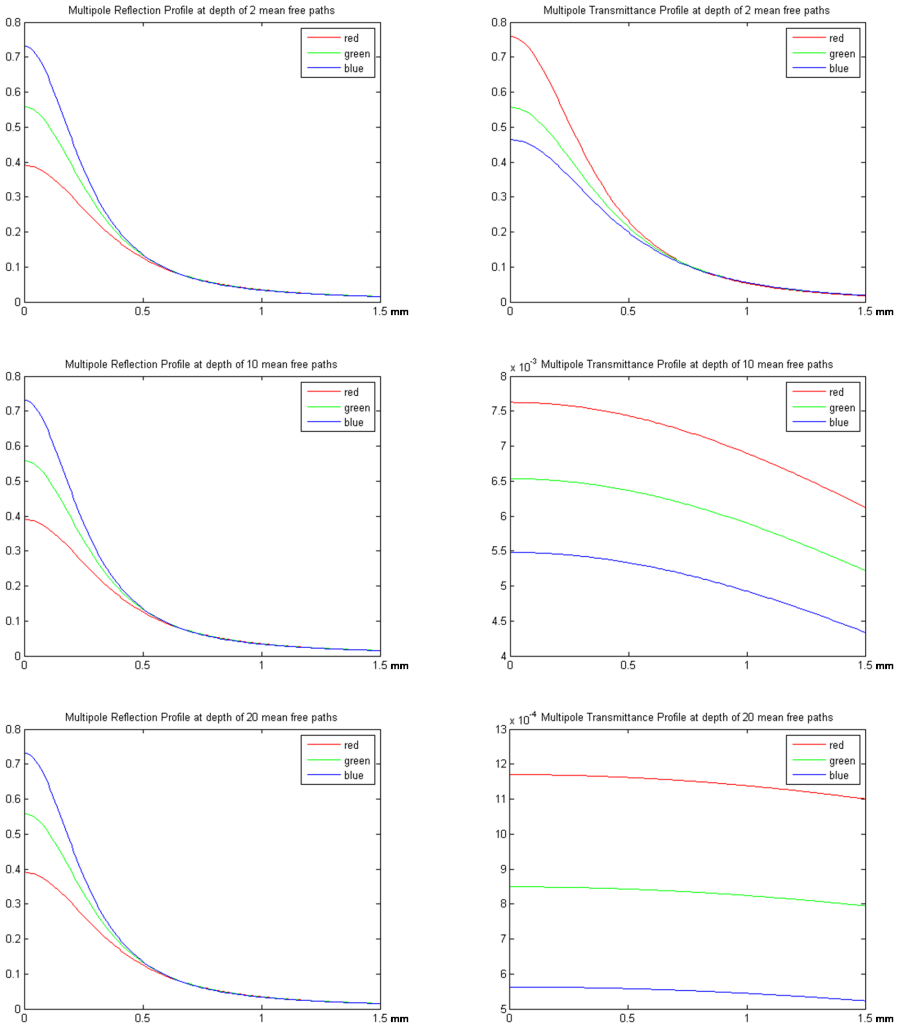

Figure 4.5: Reflectance and Transmittance profiles at different thickness. The plots for the reflectance (left) and transmittance (right) profiles for marble. The vertical axis represents the reflectance and transmittance profiles respectively, and the horizontal axis represents the variation of the distance r. The values for the first row were obtained for a thickness of 2 mean free paths, for the second row 10 mean free paths and for the third row 20 mean free paths

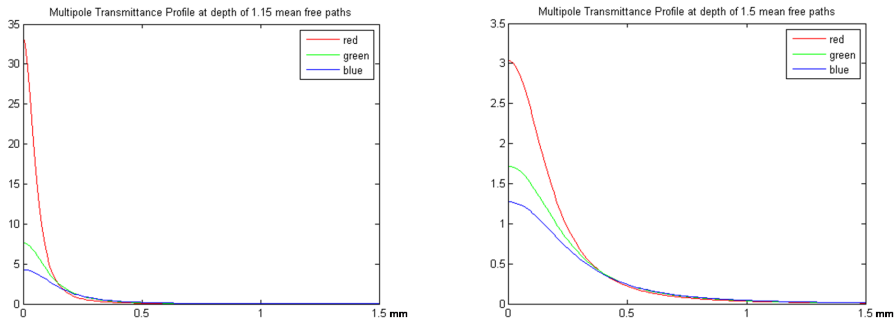

Figure 4.6: The transmittance profile for marble for very thin slabs. The left plot was calculated for a thickness of 1.15 mean free paths and the right plot for 1.5 mean free paths.

4.3.1 Multi-layered materials

In multi-layered materials the contribution from all individual layers influences light's behavior. As presented in section 3.4.2 after successive reflections and refractions light exits the object.

A thorough implementation for multi-layered materials needs depth peeling. In this thesis, the coating layer is considered to have a certain known thickness. In the pre-rendering phase, the profiles of the two materials (the coating layer and the underlying layer) are convoluted and the combined profiles are used in rendering. The theoretical background for the method is presented in section 3.4.2 and the detailed implementation is described in 5.1.2. The rest of the algorithm remains the same.

4.4 Rendering with Point Light Illumination

Recall equation 4.1 calculating the outgoing radiance in ω_o direction:

$$L_o(x_o, \omega_o) = \int_A \int_{2\pi} S(x_o, \omega_o, x_i, \omega_i) L_i(x_i, \omega_i) \cos\theta_i \mathrm{d}\omega_i \mathrm{d}A$$

The integration is done over surface area because more than one point where light is incident can have a contribution on x_o and the total contribution is the sum from all of them. For each of the points, the incoming radiance is calculated by integrating over the positive hemisphere around the point to account for all possible directions. The diffuse BSSRDF (equation 3.4) is the product of three terms:

$F_t(\eta, \omega_o)$	depending on the outgoing direction
$F_t(\eta, \omega_i)$	depending on the incoming direction
$R_d(\|x_i - x_o\|)$	depending on the distance$\|x_i - x_o\|$

Re-arranging the terms inside the integrals, the previous equation can be re-written as:

$$L_o(x_o, \omega_o) = F_t(\eta, \omega_o) \int_A R_d(\|x_i - x_o\|) \int_{S^2} L_i(x_i, \omega_i) F_t(\eta, \omega_i) \cos\theta_i \mathrm{d}\omega_i \mathrm{d}A$$

The second part of the integral, dependent on ω_i is easily calculated for a single

point-light source:

$$\begin{aligned} E(x_i, \omega_i) = & \int_{2\pi} L_i(x_i, \omega_i) F_t(\eta, \omega_i) \cos\theta_i \mathrm{d}\omega_i \\ = & \frac{I(\omega_i)}{\|x_{light} - x_i\|^2} F_t(\eta, \omega_i) \cos\theta_i \end{aligned}$$

For a single point light source, equation 4.1 becomes

$$L_o(x_o, \omega_o) = F_t(\eta, \omega_o) \int_A R_d(\|x_i - x_o\|) E(x_i) \mathrm{d}A$$

The remaining integral does not have an analytical solution and thus, needs to be numerically approximated, but a more suitable shape can be achieved. The integration domain A is the surface of the 3D model which is a triangle mesh, therefore A is the sum of the area of each triangle in the mesh:

$$A = \sum_{i=1}^{N} A_{\triangle i},$$

where N is the number of triangles of the mesh and i denotes the i^{th} triangle. It is known that for definite integrals we can apply the *additivity of integration on intervals* rule:

$$\int_a^b f dx + \int_b^c f \mathrm{d}x = \int_a^c f dx$$

or

$$\int_{A_1} f dx + \int_{A_2} f \mathrm{d}x = \int_{A_1 + A_2} f dx$$

and equation 4.1 is now:

$$L_o(x_o, \omega_o) = F_t(\eta, \omega_o) \sum_{i=1}^{N} \int_{A_{\triangle i}} R_d(\|x_i - x_o\|) E(x_i) \mathrm{d}A_{\triangle i}$$

The term $E(x_i)$ is the incoming radiance from the light source, and if the point x_i is visible in the light's perspective, then the value for $E(x_i)$ will have a positive,

non zero, value. The point may not visible: could be behind the light source or not in it's viewport; or it may be shadowed by other geometry from the scene or by other parts of the mesh (self-shadowing). Therefore, only a restricted area of the mesh is suitable for sampling, and that is the visible surface area from light's point of view. The equation now becomes:

$$\begin{aligned} L_o(x_o, \omega_o) &= F_t(\eta, \omega_o) \int_{visA} R_d(\|x_i - x_o\|) E(x_i) \mathrm{d}A \\ &= F_t(\eta, \omega_o) \sum_{i=1}^{N_{vis}} \int_{A_{\triangle i}} R_d(\|x_i - x_o\|) E(x_i) \mathrm{d}A_{\triangle i} \end{aligned}$$

where the subscript *vis* on A_{vis} and N_{vis} denotes the visible surface area and the number of visible triangles respectively.

The set of visible polygons is in fact the subset of the triangle mesh that reach the rasterizing stage of the GPU pipeline and are written in the light's output render texture. Therefore it is suitable to transform the integration domain from surface area to texture space area (denoted A_{ts} in equation 4.4). This will introduce a scaling factor for the integral [Chang et al., 2008]:

$$L_o(x_o, \omega_o) = \frac{A_{vis}}{A_{ts}} \int_{A_{ts}} R_d(\|x_i - x_o\|) E(x_i) dA_{ts} \tag{4.4}$$

Using discrete numerical integration (briefly discussed in section 3.5.1), the approximate value of $L_o(x_o, \omega_o)$ calculated for N_s samples is:

$$L_o(x_o, \omega_o) \approxeq \frac{A_{vis}}{A_{ts}} \sum_{s=1}^{N_s} R_d(\|x_s - x_o\|) E(x_s) \triangle A_{ts} \tag{4.5}$$

where x_s is the sample point on the texture and $\triangle A_{ts}$ is the differential texture space area associated with x_s:

$$\triangle A_{ts} = \frac{A_{ts}}{N_s}$$

Substituting $\triangle A_{ts}$ in equation 4.5 results in:

$$\begin{aligned} L_o(x_o, \omega_o) &\approx \frac{A_{vis}}{A_{ts}} \frac{A_{ts}}{N_s} \sum_{s=1}^{N_s} R_d(\|x_s - x_o\|) E(x_s) \\ &= \frac{A_{vis}}{N_s} \sum_{s=1}^{N_s} R_d(\|x_s - x_o\|) E(x_s) \end{aligned} \tag{4.6}$$

Unless the model occupies the whole texture space, there will be unwritten pixels in the texture. This does not pose a threat to the method, since the associated irradiance with an empty pixel will be zero and the result of the approximation will remain unmodified.

Increasing the number of samples from the texture will allow the splats to overlap and each pixel of the screen (or x_o point) will have enough samples for the integral to converge.

4.5 Rendering with Environment Illumination

4.5.1 Uniform Sampling

In equation 4.1 to calculate the outgoing radiance, light contribution is integrated from points on the model's surface.

In equation 3.5.2, a *pdf* is used to weigh the contribution of each sample. When the surface of the model is uniformly sampled , the *pdf* of a sample point i is:

$$pdf(x_i) = \frac{1}{A_{surface}}$$

Note that this is a valid *pdf* since the constraint in 3.5.2 is respected:

$$\int_{A_{surface}} pdf(x_i) dA_{surface} = \int_{A_{surface}} \frac{1}{A_{surface}} dA_{surface} = 1$$

Integrating the radiance (equation 4.1) will become:

$$\begin{aligned} L_o(x_o, \omega_o) &= \int_A \int_{2\pi} S(x_o, \omega_o, x_i, \omega_i) L_i(x_i, \omega_i) \cos\theta_i \mathrm{d}\omega_i \mathrm{d}A \\ &= \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{\int_{2\pi} S(x_o, \omega_o, p_j, \omega_i) L_i(x_j, \omega) \cos\theta d\omega}{pdf(x_j)} \\ &= \frac{A_{surface}}{N_s} \sum_{j=1}^{N_s} \int_{2\pi} S(x_o, \omega_o, p_j, \omega) L_i(x_j, \omega) \cos\theta d\omega \end{aligned} \tag{4.7}$$

To reach an uniform distribution, at first, points are generated randomly on the surface of the model and then the distribution is smoothed using a point relaxation procedure. However, a good idea would be to choose the initial random position of the points based on a cumulative distribution function (cdf) according to each triangle's face area. This would allow larger triangle faces to receive more points. The difference between randomly placing the points and placing them according to a *cdf* is shown in figure 4.7, where 100 samples were chosen on a planar triangle mesh.

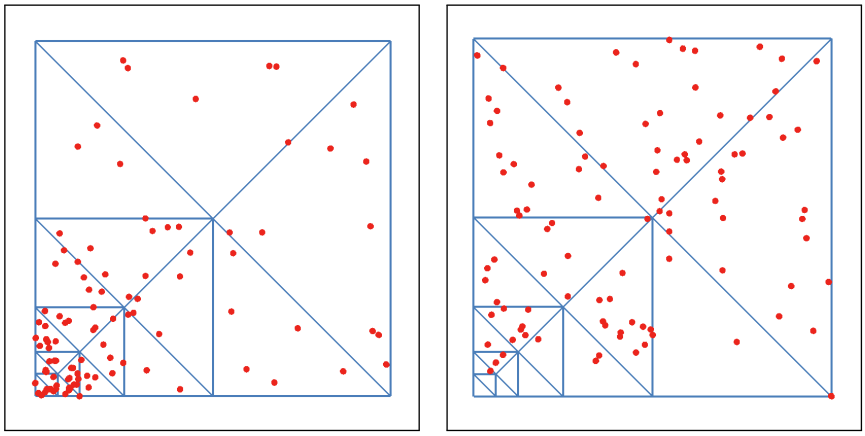

Figure 4.7: Sampling according to a *cdf*. A number of 100 sample points distributed on a planar mesh randomly (left) and according to a cumulative distribution function (right). The distribution is already a step forward towards uniformity in the right image, compared to the left image.

The point relaxation procedure will move each point depending on the influence the points in its vicinity will have on it. The details for creating the *cdf*, sampling the mesh and how to reach a uniform distribution will be presented in section 5.3.1.

4.5.2 Environment Illumination

Complex illumination in scenes is difficult to implement using point light or directional light sources. Usually, to simulate illumination in real scenes it is needed to account also for area lights, very distant and large light sources (like sun light) or the continuous ambient light distribution from the sky.

Blinn and Newel [Blinn and Newell, 1976] introduced reflection maps to simulate environment lighting contribution on mirror-like objects. A generalization that includes a wider class of reflectance models was proposed by Miller and Hoffman [Miller and Hoffman, 1984], based on pre-filtering the environment maps. The pre-filtering is an off-line process done before rendering and it usually stores the map in a more appropriate way to reach interactive frame rates [Greene, 1986].

To render translucent objects under environment illumination, in this thesis, incoming radiance is separated into two distinct components: diffuse contribution (corresponding to low frequency illumination) and specular contribution. The two different types are obtained from different representations of the environment map. The separation into the respective components is motivated by the fact that incoming light will be transmitted or reflected at the surface of the object. Transmitted light will be blurred with each scattering event inside the highly scattering material and will result in diffuse outgoing radiance. Light reflected off the surface is considered to contribute to the specular highlights.

For diffuse illumination, the environment map needs to be filtered and its contribution needs to be reduced to a set of numerical coefficients in a pre-rendering step as in [Ramamoorthi and Hanrahan, 2001]. This is done in order to achieve a suitable representation of the map, for interactive rendering. On the other hand, specular illumination will be obtained directly from the environment map, without pre-filtering.

This section will continue by giving an overview of the method chosen to obtain the diffuse contribution from the environment map. The method is based on using the first 9 spherical harmonics coefficients to filter the environment map as presented in [Ramamoorthi and Hanrahan, 2001]. In section 5.3.1, the implementation of the filtering and rendering of diffuse illumination will be described.

Spherical harmonics are the 3D equivalent on the unit sphere of the 2D Fourier basis functions. For the rest of the report, the spherical harmonics coefficient of rank l and order m will be denoted with Y_l^m. A visual representation is shown in figure 4.8.

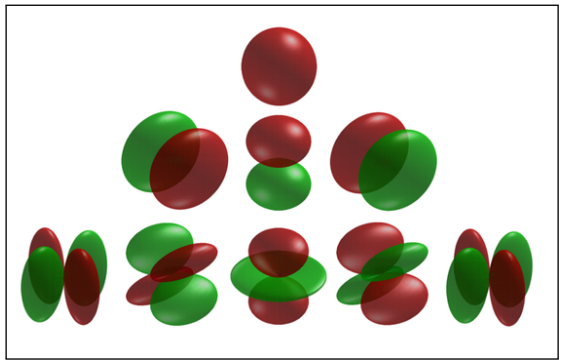

Figure 4.8: The first nine spherical harmonics coefficients. Source: http://en.wikipedia.org/wiki/File:Harmoniki.png

The first 9 spherical harmonics coefficients are [Ramamoorthi and Hanrahan, 2001]:

$$\begin{array}{rcccl}
Y_0^0(\theta,\phi) & = & 0.282095 & = & 0.282095 \\
Y_1^{-1}(\theta,\phi) & = & 0.488603 \cdot \sin\theta\cos\phi & = & 0.488603 \cdot x \\
Y_1^0(\theta,\phi) & = & 0.488603 \cdot \cos\theta & = & 0.488603 \cdot z \\
Y_1^1(\theta,\phi) & = & 0.488603 \cdot \sin\theta\sin\phi & = & 0.488603 \cdot y \\
Y_2^{-2}(\theta,\phi) & = & 1.092548 \cdot \sin\theta\cos\phi\sin\theta\sin\phi & = & 1.092548 \cdot xy \\
Y_2^{-1}(\theta,\phi) & = & 1.092548 \cdot \sin\theta\sin\phi\cos\theta & = & 1.092548 \cdot yz \\
Y_2^0(\theta,\phi) & = & 0.315392 \cdot (3\cos^2\theta - 1) & = & 0.315392 \cdot (3z^2 - 1) \\
Y_2^1(\theta,\phi) & = & 1.092548 \cdot \sin\theta\cos\phi\cos\theta & = & 1.092548 \cdot xz \\
Y_2^2(\theta,\phi) & = & 0.546274 \cdot ((\sin\theta\cos\phi)^2 - (\sin\theta\sin\phi)^2) & = & 0.546274 \cdot (x^2 - y^2)
\end{array} \tag{4.8}$$

The equations above show the coefficients in spherical coordinates and in their corresponding Cartesian coordinates.

The pre-filtering calculates the lighting coefficients L_l^m [Ramamoorthi and Hanrahan, 2001]:

$$L_l^m = \int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi} L(\theta,\phi)Y_l^m(\theta,\phi)\sin\theta \mathrm{d}\theta\mathrm{d}\phi \tag{4.9}$$

The l and m indices refer to the rank and order of the spherical harmonics coefficients. $L(\theta,\phi)$ is the irradiance from the environment map. Each lighting

coefficient L_l^m is calculated by integrating over the environment map the values weighted by the Y_l^m coefficients. The implementation of the numerical approximation for the filtering integral in equation 4.9 will be described in detail in section 5.3.1.

The irradiance is calculated with the following formula [Ramamoorthi and Hanrahan, 2001]:

$$E(\theta,\phi) = \sum_{l,m} \hat{A}_l L_l^m Y_l^m(\theta,\phi) \tag{4.10}$$

In terms of Cartesian coordinates equation 4.11 becomes:

$$\begin{array}{rcl} E(n) & = & \sum_{l,m} \hat{A}_l L_l^m Y_l^m(n) \\ n = (x,y,z) & = & (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \end{array} \tag{4.11}$$

[Ramamoorthi and Hanrahan, 2001] give a formula for the necessary values of $\hat{A}_l$:

$$\begin{array}{rrcl} l=1 & \hat{A}_1 & = & \frac{2\pi}{3} \\ l>1, odd & \hat{A}_l & = & 0 \\ l, even & \hat{A}_l & = & 2\pi \frac{(-1)^{\frac{l}{2}-1}}{(l+2)(l-1)} \left[\frac{l!}{2^l \left(\frac{l}{2}!\right)^2} \right] \end{array} \tag{4.12}$$

Equation 4.11 can be expanded for the first 9 SH terms:

$$\begin{array}{rcl} E(\theta,\phi) & = & \sum_{l,m} \hat{A}_l L_l^m Y_l^m(\theta,\phi) = \\ & = & A_0 L_0^0 Y_0^0(\theta,\phi) + \\ & + & A_1 L_1^{-1} Y_1^{-1}(\theta,\phi) + A_1 L_1^0 Y_1^0(\theta,\phi) + A_1 L_1^1 Y_1^1(\theta,\phi) + \\ & + & A_2 L_2^{-1} Y_2^{-1}(\theta,\phi) + A_2 L_2^{-1} Y_2^{-1}(\theta,\phi) + A_2 L_2^0 Y_2^0(\theta,\phi) + \\ & + & A_2 L_2^1 Y_2^1(\theta,\phi) + A_2 L_2^2 Y_2^2(\theta,\phi) \end{array} \tag{4.13}$$

Using equations 4.8 and 4.12, the authors present the reduced formula by group-

ing the variables with respect to their common multiplier:

$$\begin{aligned} E(n) &= c_1 L_2^{-2}(x^2 - y^2) + c_3 L_2^0 z^2 + c_4 L_0^0 - c_5 L_2^0 + \\ &\quad 2c_1 \left(L_2^{-2} xy + L_2^1 xz + L_2^{-1} yz \right) + \\ &\quad 2c_2 \left(L_1^1 x + L_1^{-1} y + L_1^0 z \right) \end{aligned} \tag{4.14}$$

The set of constants c_i are also given in the paper [Ramamoorthi and Hanrahan, 2001]:

$$\begin{gathered} c_1 = 0.429043 \quad c_2 = 0.511664 \quad c_3 = 0.743125 \\ c_4 = 0.886227 \; c_5 = 0.247708 \end{gathered} \tag{4.15}$$

The constants are obtained from the formulas for Y_l^m and $\hat{A}_l$ (equations 4.8 and 4.12). For example, the constants c_3 and c_5 for $L_2^0 z^2$ and L_2^0 are verified by introducing $l = 2$ and $m = 0$ in equation 4.13:

$$\begin{aligned} E(\theta, \phi)_{l=2, m=0} &= A_2 L_2^0(\theta, \phi) Y_2^0(\theta, \phi) \\ &= 0.785398 L_2^0 \cdot 0.315392 (3 \cos^2 \theta - 1) \\ &= 0.7431247 L_2^0 \cos^2 \theta - 0.2477082 L_2^0 \\ &= c_3 L_2^0 z^2 - c_5 L_2^0 \end{aligned}$$

Equation 4.14 will be used during rendering. Note that it's parameter is a normal vector, meaning that the formula calculates the diffuse illumination from the normal direction.

Chapter 5

Implementation

This chapter will describe the specific details for the algorithm to be implemented. Chapter 4 presented the overview of the algorithm and the method chosen. The next section will describe the implementation for the algorithm in a scene where one translucent object and one point light source are present. Enough information is provided at the end of the section for rendering more objects with more light sources.

5.1 Translucent Materials under Point Light Illumination

In the case of illumination from a single point-light source the algorithm is composed of three main render passes and a final deferred rendering pass. Before the first frame is rendered, the splat size is approximated (r_{max}) and the diffusion profiles are calculated and tabulated into textures. The values are calculated only once and re-used every frame. There is very little work done on the CPU after the profiles have been calculated, the rest being done on the GPU.

To calculate the value of r_{max}, equation 4.3 has to be solved. Because the diffusion profile R_d is a smooth function, it is possible to apply Simpson's Composite

Rule (see equation 3.14) to approximate numerically the integrals in the equation. A function for approximating the value of the integral of Rd between $[a, b]$, using a number of samples is shown in the code example below.

```
Vec3f simpson_s (float a, float b, int samples, DiffusionProperties
    dp)
{
  Vec3f sum (0);
  float stepsize = (b - a) / samples;
  sum = R_d (a, dp) + R_d (b, dp);
  for (int  i = 1; i <= samples - 1; i++)
  {
    if (i%2 == 0)
      sum += 2 * R_d (a + i * stepsize, dp);
    else
      sum += 4 * R_d (a + i * stepsize, dp);
  }
  return sum * stepsize / 3.0;
}
```

Listing 5.1: Simpson's Composite Rule

First, the method is applied for a large enough radius r_{inf} to approximate the infinite integral. The plots of the diffusion profile show that the values come close to zero very shortly, so r_{inf} can be easily substituted with a large value, for example 100. Next, the integral is approximated for an interval of $[0, r_{max}]$, where r_{max} takes discrete values from 0 to r_{inf}, and the first value for which the inequality is satisfied will be the solution. The value used for ϵ was 0.01.

The value of R_d (equation 3.4.1) can be calculated at runtime but since the domain where its parameter r is known (from 0 to the maximum extent where the contribution is noticeable, r_{max} in equation 4.3) it is preferred to have the values pre-calculated. For this reason, a 1D texture with the dipole reflectance profile is sent to the GPU and sampled during shading. Figure 5.1 shows the plot of the diffusion profile with different sampling densities, corresponding to the texture's size. If the size is too small, banding artifacts will appear because the values between two consecutive samples will be linearly interpolated. A texture size of 256 or 512 will be large enough to have the profile well defined and low enough not to introduce overhead in the pre-rendering step.

In the first pass, the scene is rendered from light's point of view and the fragment shader outputs to a double off-screen frame buffer 3D light-space position and light-space normal, similar to the Translucent Shadow Map algorithm [Dachsbacher and Stamminger, 2003] (see section 2). Code samples are shown in the

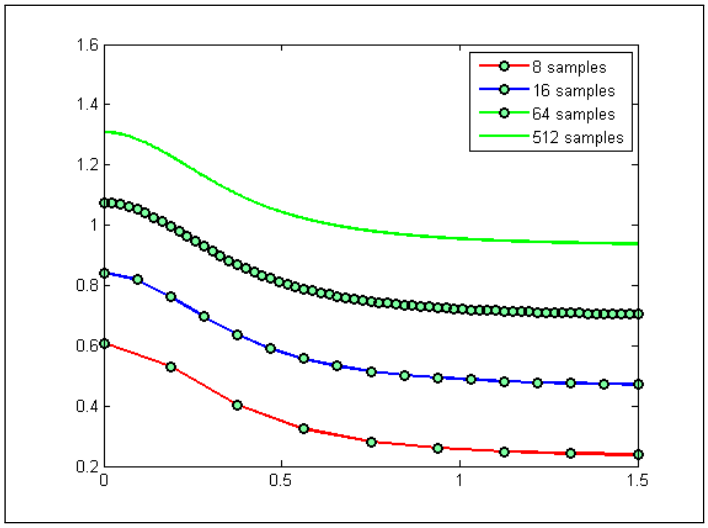

Figure 5.1: Sampling a diffusion profile with different numbers of samples. The green dots represent the sample points on the profile. Values between the sample points are obtained by linear interpolation, therefore small sample counts (bottom lines) will not approximate the shape of the profile sufficiently.

listings below.

```
gl_Position = ftransform();
position_es = (gl_ModelViewMatrix * gl_Vertex).xyz;
normal_es = normalize(gl_NormalMatrix * gl_Normal);
```

Listing 5.2: Vertex Shader

```
gl_FragData[0] = vec4(position_es, 1.0);
gl_FragData[1] = vec4(normal_es, 1.0);
```

Listing 5.3: Fragment Shader

This texture will hold the points where light incident on the model is sampled, equivalent to the x_i point set in equation 4.1. Note that the last channel of the

position buffer is set to 1.0 for each pixel written. Because the buffer is initialized to 0.0, this channel will act as a flag, to see which pixels were written. The same texture can be used as a shadow map for the rest of the scene, so the resolution can be adjusted to fit this purpose. The output from this pass should follow the pattern in figure 5.2.

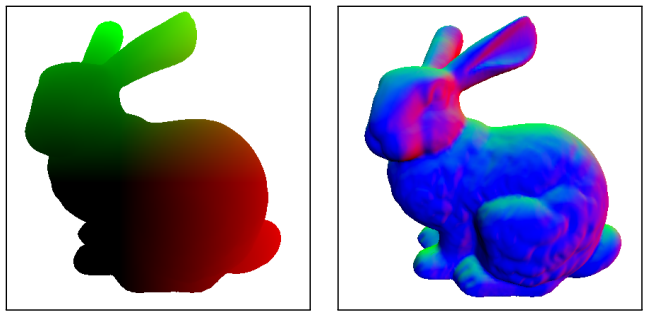

Figure 5.2: The output pattern for the first two render passes. The left picture shows positions and the right picture shows normals

In the second render pass, the set of x_o points and their respective n_o normals from equation 4.1 are generated. Into a different framebuffer, the scene is rendered from the observer's perspective and, in the fragment shader, eye-space positions and normals are written to two different screen-sized textures. The output will follow the same color pattern and aspect as in figure 5.2 but the orientation of the model and its size of course can be different. Again, the last channel will be used as a flag to see which pixels were written during this pass, and it will be needed in the deferred rendering stage.

Before the third pass is rendered on the GPU, the sampling pattern has to be generated on the CPU. A set of texture coordinates is generated so that during the GPU stage, light's texture is sampled at those locations and the sample positions are found. Since the sample map is going to be reused to calculate the shadows in the scene, normally the translucent object will occupy a varying area on the texture, therefore it is important to know where to sample. The sampling region of interest can be calculated by projecting the object's bounding box onto the screen and creating a rectangle from the minimum and maximum coordinates (see figure 5.3). The benefits of sampling just the region of interest will be discussed later in section 5.2.2

The sampling pattern is generated by splitting the region of interest from the sample map into a $N \times M$ grid and choosing the middle of each grid rectangle. The application stage sends to the GPU 2D vertices having as position (u, v)

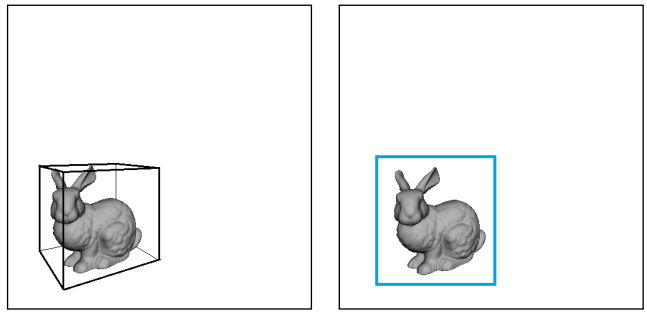

Figure 5.3: The Region of Interest The left picture shows the object and its surrounding bounding box in eye's viewport. On the right, the sampling region is shown, calculated by projecting the bounding box on the screen and choosing the minimum and the maximum values

coordinates and none of these 2D vertices will end on the screen since they are just a fast method to send sampling positions to the GPU.

Note that the number of samples needed for point light illumination is still an open problem. The solution presented by [Shah et al., 2009] did not offer suitable results in this thesis' implementation.

In the vertex shader, 3D positions are extracted from the sample map at the (u, v) coordinates from the sampling pattern received. The values extracted from the sample map will be the new positions of each vertex sent through the pipeline. Each vertex that arrives in the geometry stage of the pipeline is checked to see if it belongs to a translucent object. This is done because light's field of view will not necessarily see only the focused translucent object, but also other objects present in the scene that have a different shading model. As well, sample positions that are outside of the object will be discarded because they will not have irradiance. The flag set in the first render pass states that the pixel stores information from a translucent object.

Since the geometry shader can spawn new geometric primitives [Akenine-Moller et al., 2008] the translucent vertex is transformed into a quad centered at the sample point and oriented towards the screen. A two-dimensional visualization of the process is shown in 5.4. It is important to specify that each quad is rendered in eye space. Its role is to act as a handle between the light sample attributes (3D position and normal) and the points being shaded (which are calculated in the second pass). That is why the splats will need also texture

coordinates which are calculated as well in the geometry stage and are relative to the screen sized textures rendered in the second pass from the eye's perspective. The code sample below shows how to calculate and assign the texture coordinates to the top-right corner of a quad.

```
gl_Position = proj_matrix *
      vec4(position_es.xyz + vec3 (r_max, r_max, 0), 1);
tex_coords = (gl_Position.xy / gl_Position.w + vec2 (1)) * 0.5;
EmitVertex();
```

Listing 5.4: Emitting the top right corner of the screen aligned quad with its respective texture coordinates

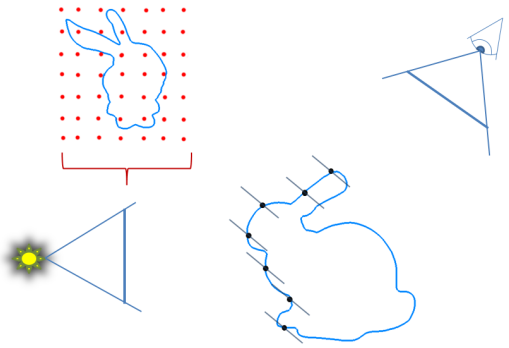

Figure 5.4: The 2D visualization of the splatting procedure. The red dots represent the sample positions in the sample map, which is rendered from light's perspective. The splats are aligned (parallel) with the observer's view plane, and positioned on the geometry

The fragment shader will output the scattering texture to a screen sized render target. The splats will be shaded individually and blended as depicted in figure 5.5. Blending is done by calling glEnable(GL_BLEND) and setting the blending function with glBlendFunc (GL_ONE, GL_ONE). It is important to also disable the depth test (glDisable(GL_DEPTH_TEST)) so the splats will not occlude each other. The generation of the scattering texture is detailed in sections 5.1.1 and 5.1.2.

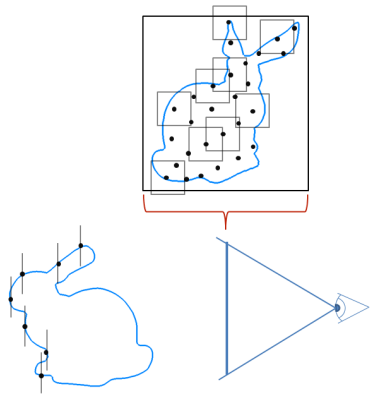

Figure 5.5: The splatting procedure from the eye's perspective. Only a few samples are expanded to quads, to show how some overlap and some span outside of eye's viewport

The last rendering pass renders to the default framebuffer the translucent object with the rest of the scene and with specular reflections. It is a deferred render pass in which the textures obtained from the previous passes are used to shade a quad spanning the whole screen.

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-1.0,1.0,-1.0,1.0,-1.0,1.0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();

glBegin (GL_QUADS);
  glTexCoord2f (0.0,0.0); glVertex3f (-1.0, -1.0, 0.0);
  glTexCoord2f (1.0,0.0); glVertex3f (1.0, -1.0, 0.0);
  glTexCoord2f (1.0,1.0); glVertex3f (1.0, 1.0, 0.0);
  glTexCoord2f (0.0,1.0); glVertex3f (-1.0, 1.0, 0.0);
glEnd ();
```

Listing 5.5: Creating the screen alligned quad for deferred rendering

The render targets from the second and third passes were screen sized textures, so the texture coordinates of each fragment on the deferred shading quad will be used to retrieve information from them. First, note in figure 5.5 that the splats

may span outside of the model. In the fragment shader, the eye space texture is sampled and if the corresponding texel belongs to a translucent material (if the flag set in the second pass is not null) then the scatter texture is sampled at the same position and the value is written in the frame buffer.

Next, specular highlights can be added on the translucent model. For this, the light direction and the viewing direction need to be calculated. The light position in eye space is calculating by transforming the origin of the light space coordinate system to eye space position. The origin of the light space coordinate system is multiplied with the inverse light transformation matrix to obtain the object space light position, and then multiplied with the eye space transformation matrix to obtain the eye space position. The following code sample is used to calculate the specular component of the illumination.

```
vec4 position_es = texture2D (position_tex, tex_coords);
vec4 normal_es = texture2D (normal_tex, tex_coords);
//calculate eye-space light position
vec4 lightPos_es =
  eye_matrix * inverse_light_matrix * vec4 (0,0,0,1);
vec3 view_dir = normalize (- position_es.xyz);
vec3 light_dir = normalize ( lightPos_es.xyz - position_es.xyz);
//calculate half vector
vec3 half_vector = normalize(view_dir + light_dir);
float cosNH = dot (half_vector, normal_es);
if (cosNH > 0)
{
  specPower = pow (cosNH, shininess);
  //add incoming radiance multiplied by specular power
  gl_FragColor += L_i * specPower;
}
```

Listing 5.6: Adding the specular component

5.1.1 Rendering with the Dipole Approximation

Recall the geometrical parameters needed to calculate the outgoing radiance 4.1 and the BSSRDF 3.4: the incoming and outgoing light directions ω_i and ω_o, the sample point x_i and the shaded point x_o. The dipole approximation (equation 3.4.1) is a function of the distance between light's incidence point and the visible point, denoted r, and it is equal to the length of the vector between x_i and x_o:

$$r = \|x_i - x_o\|$$

The texture space position u where the dipole texture is sampled is calculated as:

$$u = \frac{r}{r_{max}} \tag{5.1}$$

Note that the u value calculated above can have only positive values (being, in fact, the ratio of two distances) but its maximum extent is undefined. The functionality of the graphics library or the hardware needs to clamp this value to $[0, 1]$, the range of a texture coordinate[1]. Equation 5.1 assures that the values of r for which light scattering is noticeable will be indexed correctly in the diffusion profile texture. A simple splat, shading a flat surface, thus over a linear radius, is presented in figure 5.6.

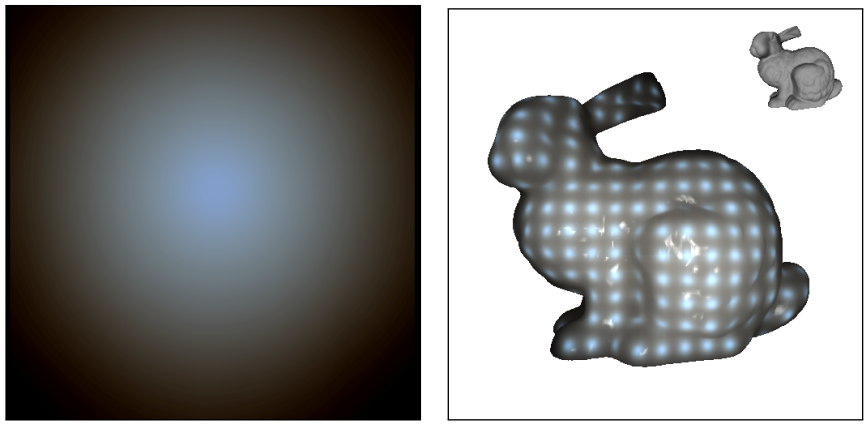

Figure 5.6: The left figure shows a shaded splat. On the right side, 15×15 splats are positioned on the object. The right image's intensity has been increased for the effects to be visible.

Each of the samples from the sample map spawns a rectangle projected on the screen. The right side of figure 5.6 shows how a 15×15 grid is sampled from light's texture and the points are splatted and shaded similar to the left side of the figure 5.6. Light's viewpoint is shown in the small viewport on the top-right side. Note that this is a case of under-sampling and is only shown for visualization and to describe the approach step by step.

Recalling equation 4.6 for numerically solving the integral for outgoing radiation, it is observed that the result of the blending operations during the creation of

[1]The implementation can vary depending on the OpenGL version and hardware. This paper assumes the texture coordinates are normalised, e.g. lie in the $[0, 1]$ range. Available with GL_TEXTURE_RECTANGLE_ARB extension, the possibility to use non-power-of-two sized textures is available, which are indexed with unnormalised coordinates

the scattering texture needs to be scaled down by the ratio between the visible surface area and the number of samples: $\frac{A_{vis}}{N_s}$. The visible surface may be computationally expensive to calculate, thus, a much cheaper and useful method is to approximate it with the visible area of the bounding box (the bounding box previously calculated). In case this scaling factor is omitted, the model may appear brighter when its orientation or the light source's orientation change. This is because changing the domain of integration will change the result of an integral, and in this case, the visible area is the domain of integration.

5.1.2 Rendering with the Multipole Approximation

Rendering with the multipole approximation reduces to changing the shading of each individual splats using the multipole approximation. The algorithm is basically the same as described for the dipole approximation in section 5.1.1, but a few noticeable differences will be presented in this section. Also, the additional information for rendering multi-layered materials is presented at the end of the section.

Single layer Materials

Recall equations 3.5 and 3.6. The values for the reflectance and diffusion profiles are calculated by summing the contribution from a series of dipole pairs. Due to the increased number of calculations needed to solve the mentioned equations, the profiles need to be tabulated in the pre-computational stage and stored in textures. The textures are now two dimensional (as opposed to the dipole approximation): the horizontal dimension of the textures holds the variation of the profiles over the variable radius and the vertical dimension holds the variation over the variable thickness.

Another difference is the number of parameters that it takes to calculate the profiles for a material. The dipole profile was calculated using the distance between the sample point and the visible point, but the multipole profiles also need the local thickness of the model. This thickness is the mean distance that light travels through the material in the incoming direction, until it exits at another position, therefore, it will be relative to light's positioning.

A thorough method to calculate the thickness of the model is depth peeling [Everitt, 2001, Bavoil and Myers, 2008], which uses additional render passes to store each layer of polygons in additional off-screen frame buffers. However, for convex models, an easier and much faster method is presented here next.

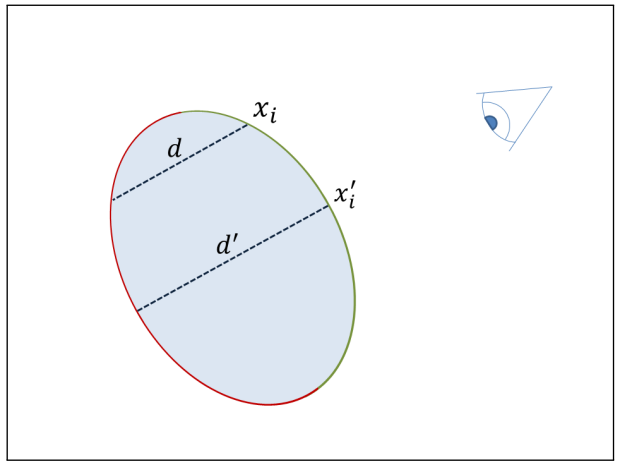

Figure 5.7: Calculating the thickness of convex objects from the observer's perspective.

Recall the first render pass from section 5.1, rendered from light's point of view. In an off-screen framebuffer the position of each point on the visible surface of the model is stored (along with the surface normal). The triangle mesh is filtered through the culling test (passing only front facing polygons) and then through the depth test and, intuitively, the texels from the texture will hold the set of points from the model's surface which are closest to the light source. For a convex model, the thickness is equivalent to the distance from the front facing polygons to the back facing polygons (see figure 5.7) as seen from light's perspective. It is possible to calculate this distance using the GPU's functionality, by rendering the backward facing polygons[2] in a similar render pass as the first one. The fragment shader will output the positions of the point in the furthest away layer. The thickness of the model is now easily calculated by sampling both textures and calculating the length of the vector between the 2 points.

In the case of concave objects is it hard to calculate the exact mean distance that light travels inside the object using only two frame buffers. Consider figure 5.8 of the 2D representation of the behavior: light travels inside the object, exits and then enters again. Using the method presented above, the distance calculated will be $\|x_i - x_o\|$. However, if the observer (the blue observer on the right side of figure 5.8) and the light are pointed in the same direction, a better solution would to approximate the distance as $\|x_i' - x_o'\|$. This is because it is not probable

[2]using the GL calls glEnable(GL_CULL_FACE) and glCullFace (GL_FRONT). The default state is restored calling glCullFace (GL_BACK) and if needed glDisable(GL_CULL_FACE)

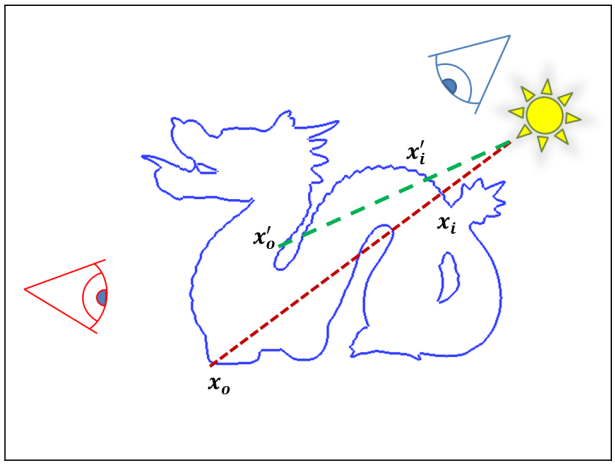

Figure 5.8: Approximating the thickness of the models depending on the orientation of the observer

that light exiting the object will return after successive reflections to the viewer.

On the other hand, if the viewer is located at the opposite side, the more accurate distance would be $\|x_i - x_o\|$. However, in practice, there is no noticeable difference depending on how the thickness is calculated when the viewer is in this position. This is because, as see in the plots in figure 4.5, the transmittance term converges to 0 after just 20 mean free paths.

The distance d used to calculate the multipole profiles is defined in the range of $[d_{min}, d_{max}]$. The minimum distance d_{min} is considered to be a few mean free paths because the errors introduced by the multipole approximation near the source [Donner and Jensen, 2005]. From the tests done, it was observed that a minimum distance, d_{min}, of 4 mean free paths is enough to avoid numerical errors, while a maximum distances d_{max} of 64 mean free paths (see figure 4.5) is large enough for the transmittance to converge to a neglectable value, and at the same time is small enough not to introduce high computational cost.

For the texture lookup, the (u, v) coordinates are calculated as following:

$$\begin{aligned} u &= \frac{r}{r_{max}} \\ v &= \frac{d}{d_{max}} \end{aligned} \tag{5.2}$$

The contribution from each of the profiles is weighted by the cosine of the angle between the normal at the incidence point and the normal at the shaded point. [Donner and Jensen, 2005] proposes to calculate the contribution from both profiles as:

$$P(r,d) = \frac{1}{2}(n_s \cdot n_l + 1)R(r,d) + \frac{1}{2}(1 - n_s \cdot n_l)T(r,d)$$

An intuitive reason for which this shading model is used is that if a point on the surface is back lit only the transmission term will be used, and when it is front lit, only the reflection term will be used. In between these two states both the profiles are used, weighted by the cosine between the normals at the incidence point and at the shaded point. A code sample is provided below:

Listing 5.7: Combining the reflectance and transmittance profiles for shading

```
vec2 tex_coords = vec2(r,d);
R = texture2D (multipoleR, tex_coords);
T = texture2D (multipoleT, tex_coords );

vec4 P = 0.5 * ( dot (Nl, Ns) + 1 ) * R +
  0.5 * ( 1 - dot (Nl, Ns) ) * T;
```

The multipole approximation introduces the transmission term in the BSSRDF and the results for thin objects and in general for back-lit objects are better. Figure 5.9 shows a marble Stanford Dragon front and back lit.

Multi-layered Materials

For multi-layered materials the reflectance and transmittance profiles are calculated in the pre-rendering stage. This section is dedicated to rendering translucent materials covered in a single highly scattering translucent layer. The principle can be extended for rendering any number of layers, and it will be described at the end of the section.

For objects covered in a single layer, first, the individual profiles for each of the materials are generated and stored, similar to the procedure presented in 4.3 and described in section 5.1. In the remainder of this section R and T will denote the reflectance and transmittance profiles, and the subscripts 1 and 2 will

refer to the coating layer and the underlying material respectively. The light reflected or transmitted after all interactions with the two media (see figure 3.13) is approximated into two single profiles (one for reflectance and the other for transmittance) that will characterize the pair of materials as a whole.

To apply the convolution between the two media, first, from the one-dimensional reflectance and transmittance profiles of size N, two dimensional radially symmetric profiles are calculated and stored into two-dimensional arrays of size $2N-1 \times 2N-1$. The first position from the one-dimensional profile will correspond to the center position $[N, N]$ in the two dimensional profile. Next, using the fast Fourier transform, the two dimensional profiles are transformed to frequency domain and equations 3.10 and 3.12 are applied, so the reflectance and transmittance profiles are calculated for the top and bottom materials as a whole. These profiles are then transformed by the inverse Fourier Transform to the time domain and used in rendering as presented in section 5.1. The procedure is shown in figure 5.10.

The transformation from time domain to frequency domain and the converse transformation to time domain were implemented in this thesis using the FFTW library[3]. In his PhD thesis [Donner, 2006], Donner uses the Discrete Hankel Transform[4], but because of the poor on-line documentation for the libraries that implement DHT, in this thesis FFTW was preferred.

To render objects covered in more than two layers (for example skin), the procedure is repeated to convolute the first two layers with the next one, and re-iterated for all remaining layers.

The problem of rendering multi-layered materials is only partially covered in this thesis. The main focus of the thesis is rendering with the dipole and multipole approximation under point light and environment illumination.

[3] http://www.fftw.org/#documentation
[4] http://www.gnu.org/software/gsl/manual/html_node/Discrete-Hankel-Transforms.html

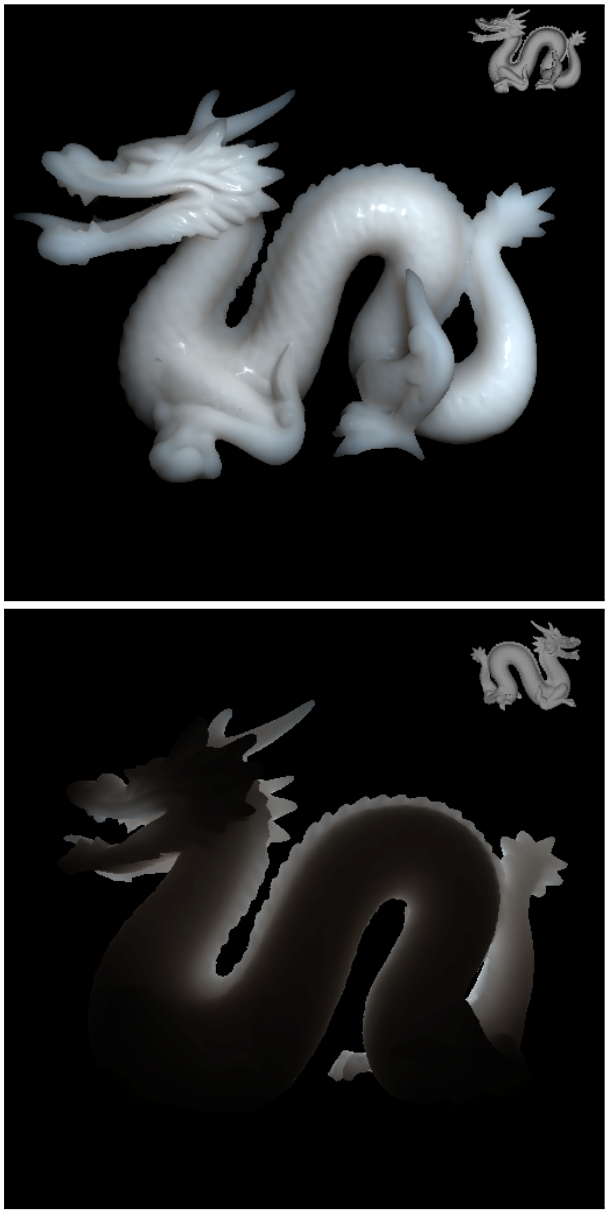

Figure 5.9: The Stanford Dragon rendered with the Multipole Approximation. Note the translucency effect on the thin surfaces like the tip of its back or the tip of its knee

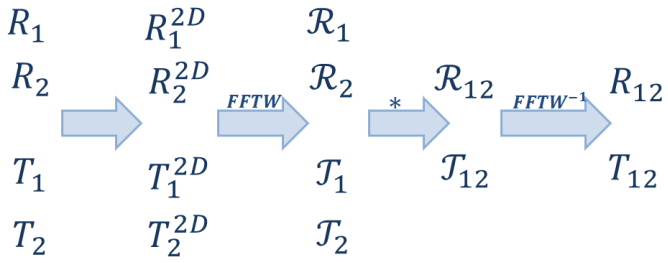

Figure 5.10: Overview of the steps for creating the profiles for multi-layered materials: from the one-dimensional profiles, two-dimensional radially symmetric profiles are created. The new profiles are transformed to frequency domain using FFT, where they are convoluted. Finally, using the inverse FFT, the profiles are transformed to time domain.

5.1.3 Rendering with multiple light sources

The previous sections were dedicated to rendering translucent materials under illumination from a single point-light source. To account for illumination from multiple light sources, the same principle is used repeatedly for each light source.

Considering the scene is lit by N_l light sources, instead of using a single render pass to generate the x_i set of points, the scene is rendered N_l times from all of the light sources' perspectives. The output from each of the N_l passes is written in separate double frame buffers, similar to the technique used for single point-light illumination, thus generating N_l sample maps. The regions of interest where the sample maps will be sampled must be calculated individually for each light source, so the object's bounding box will be projected using the transformation from each perspective. The procedure is depicted in figure 5.11. The render pass from the eye's point of view remains the same.

To create the scattering texture, the sampling pattern is sent to the GPU for each light source. Each sample point will have associated the intensity of the light source it belongs to.

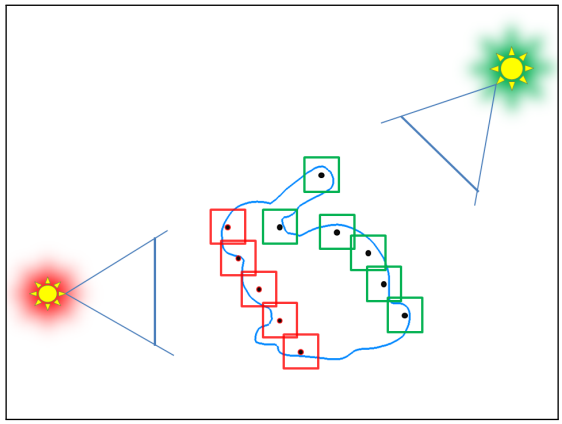

Figure 5.11: The 2D visualization of the splatting process using two light sources. Each light will have its own sample map and the splats will be shaded according to the properties of their light source.

5.1.4 Rendering multiple materials

To render multiple materials, when the scene is rendered from the eye's perspective, instead of using a double render target (positions and normals) a triple render target is used to store positions, normals and material information. Since the render target has four channels, albedo and shininess can be stored as material properties for non-translucent materials. This is usually referred to as the G-buffer [Policarpo and Fonseca, 2005]. Similar to the procedure for rendering single objects, the last channel of the render target containing the positions will now act as a unique shader identifier. If the pixel is written during this render pass, it will be non zero and it will hold the ID of the material used.

In the deferred rendering stage, the shader ID is checked and if the pixel currently shaded covers a translucent material, the scatter texture is sampled and the value is written in the framebuffer.

5.2 Rendering with Point Light Illumination - Discussion

This section will begin by presenting the advantages of the method and a few of the results. The full set of results will be displayed in a later chapter (Chap-

ter 6) and in the appendix (in Appendix A). Next, this section will present some of particularities of the method that have been observed during the implementation, and will end with presenting the drawbacks of the method. A later chapter will propose ideas and references to literature for solutions that can possibly overcome the drawbacks.

5.2.1 Advantages

Recall that the output from the first render pass was two buffers with light space positions and normals. The set of positions seen from light's perspective, or the distances from the visible points to the light source, can be used for a shadow map on the whole scene. This is useful combined with the careful sampling of the texture space, only in the region occupied by the translucent object.

Shadows add a great amount of realism to the rendered scene and can give important clues about the geometry in the scene. The shadow map generated in the first pass needs to be used only on non-translucent objects. The translucent object will have inherent shadows since the sample points will be chosen only from the lit surfaces of the model.

Another important aspect of the method is that soft-shadows, usually computationally expensive in other applications, are obtained at no additional cost. This is due to the blending of overlapping splats, which in fact are radially symmetric diffuse distributions. Figure 5.12 presents a scene composed of two marble objects: the floor and the Killeroo 3D model. Note the diffuse shadow on the floor and the shadowed arms and legs of the character. This demonstrates that the method accounts for occlusion and self-shadowing with no additional cost. Still, the soft-shadows are artificial, since theoretical point light sources generate only robust hard shadows.

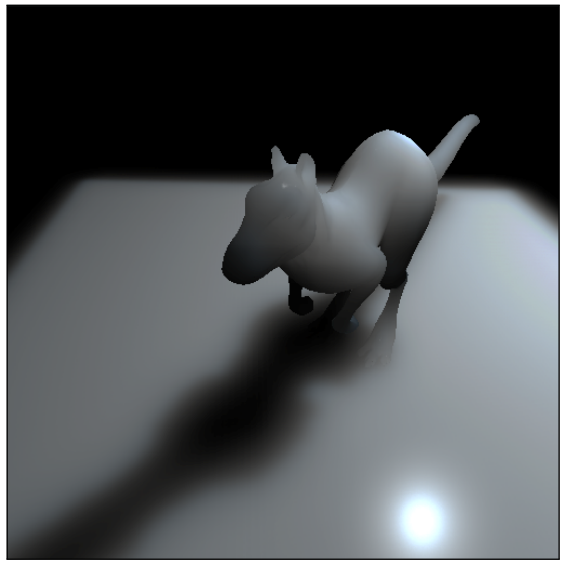

Figure 5.12: Killeroo with soft shadows and specular highlight on the floor.

5.2.2 Caveats

As it was briefly mentioned in the description of the implementation, the sample positions are carefully placed in a region of interest which the translucent object spans in the light source's texture space. For this, the bounding box of the object (easily calculated when the triangle mesh is loaded) is projected on the screen, and the maximum and the minimum values of its eight vertices will shape a rectangle on the screen. This is the region of interest. Instead of sampling the texture space in the $[0 \dots 1] \times [0 \dots 1]$ domain, it is more suitable to sample inside the $[u_{min} \dots u_{max}] \times [v_{min} \dots v_{max}]$ domain, as depicted in figure 5.13.

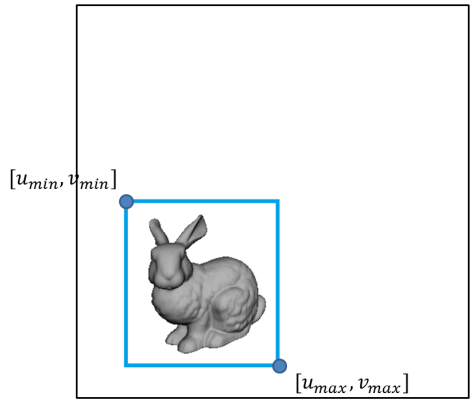

Figure 5.13: The region of interest and its extent

For example, if the light is far away from the scene and the object occupies a small area in light's viewport, not sampling only the region of interest would result in a very large number of sample points to cover the whole texture. The different behaviors are shown in figure 5.14.

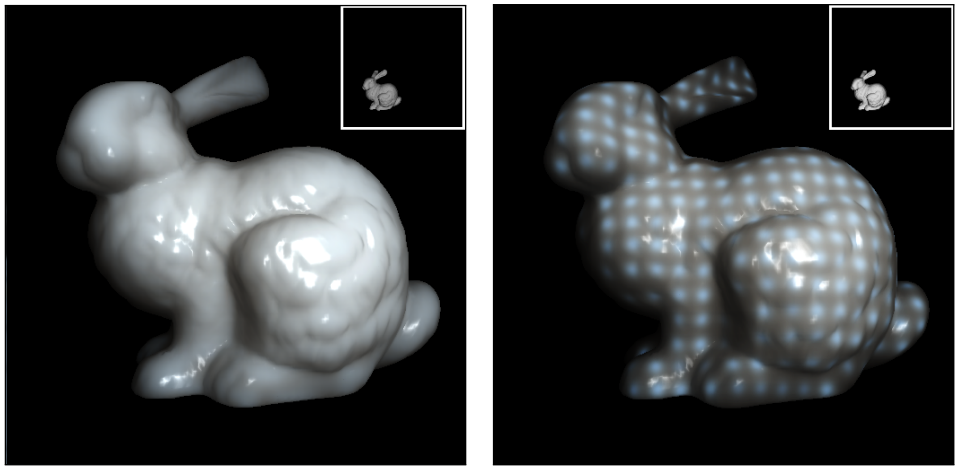

Figure 5.14: Comparison between sampling only the region of interest (left) and the whole sample map (right) with the same sample count. Light's perspective is shown in the top right corner. Also note that the right image's intensity has been increased to show the effect of under-sampling.

In the description of the first render pass, it was mentioned that the last channel of the first render target (the alpha channel of the position texture) will be set to 1 for each pixel written during the pass, acting as a flag. When the texture is sampled, many sample positions may fall outside the object so it is useful to make the distinction between valid samples that belong to the translucent object and invalid samples that fall outside. Not only they will be causing an artifact on the object (as in the left side of figure 5.15), but they will introduce a computational overhead because more blending operation per pixel will be needed to shade them. Invalid samples that are outside of the translucent object will be sent to the geometry shader as vertices situated in $(0, 0, 0)$ so a number of splats will be centered in the origin of the coordinate system. The samples that fall on the border of the translucent object will be interpolated between valid and invalid texels, and their transformed positions will be on the lights view volume (as see in the right picture of figure 5.15).

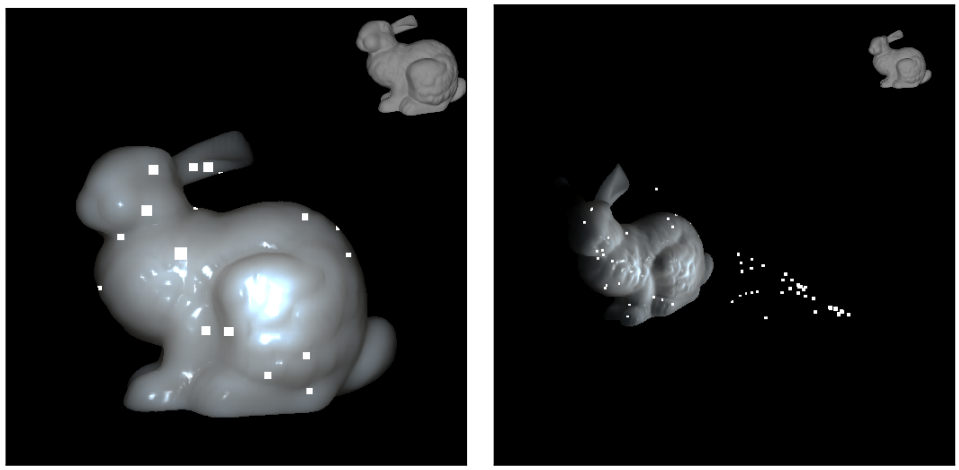

Figure 5.15: The white splats are spawn by invalid samples and they create artifacts on the rendered model. Note how the invalid splats are placed on the border of light's view frustum. The object rendered from light's perspective is shown in the top right corner of each image

This issue can be solved by checking during the geometry shader stage if the sample position is valid or not. Invalid samples will be discarded, with the aid of the geometry shader's functionality [Akenine-Moller et al., 2008].

```
vec4 position_ls = texture2D(light_pos_tex, tex_coords);
validSample = position_ls.w;
```

Listing 5.8: Removing invalid samples - Vertex Shader

```
if (validSample)
{
  //create quad vertices
  //emit vertices
  //end primitive
}
else; //discard sample point
```

Listing 5.9: Removing invalid samples - Geometry Shader

Note that some of the samples may fall very close to the border of the translucent object in the texture space and their value may be the interpolation between neighboring pixels. That is why, only samples with the flag equal to 1 are considered valid.

An important issue with sampling light's texture space is texture filtering. When the number of samples becomes too large, it is very possible that two neighbor samples will be very close to each other that they will share the same texel in the texture. This will not only induce redundancy and computational overhead when their corresponding splats are blended, but will also create a grid-like pattern on the translucent object. Figure 5.16 shows the effect.

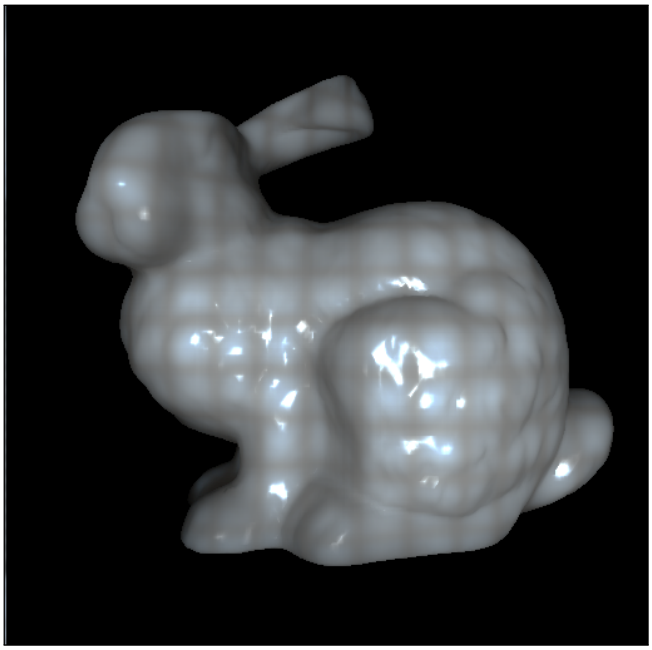

Figure 5.16: The grid-like pattern artifact that appears when the incorrect texture environment is set

To avoid this, the filtering texture parameters of the render target must be set to GL_LINEAR. Thus, OpenGL will do a linear interpolation between the 4 neighboring texels of the sample point, instead of the default behavior of choosing the closest one (GL_NEAREST). This is also recommended for the reflectance and transmittance profiles to avoid banding artifacts.

The reflectance and transmittance textures should also be configured to clamp the texture coordinates to 1, using GL_CLAMP. The behavior of GL_REPEAT is not suitable for the purpose of this thesis. It is also indicated to set the last element in the textures to 0, so that texture coordinates with values larger than 1 will point to null values.

As it was previously mentioned, the sample points in light's texture space are positioned in the middle of the texture space rectangle that they will approximate. This may introduce unwanted patterns on specific models. One way to overcome this is to add a small random deviation on each sample's texture space position, taking care not to exceed its default rectangle.

In [Shah et al., 2009] the authors apply a technique for re-using a subset of

the sample points from one frame to the following one. The motivation is that artifacts may appear when the viewing direction or light's orientation is changed between frames [Shah et al., 2009]. In practice, during the implementation and testing of this thesis, it was noticed that using sufficient sampling points, without losing interactive frame-rates, the artifacts are not visible.

5.2.3 Drawbacks

Because the incoming radiance is sampled on a two dimensional domain (the texture space) which is the screen projection of the visible surface area, under-sampling issues may occur when faces of the model are oriented at a grazing angle with respect to the light's direction. If the surface is relatively small this may remain unnoticed, but as its area increases the samples from the texture space will no longer be enough to shade it. This unwanted behavior is shown in figure 5.17.

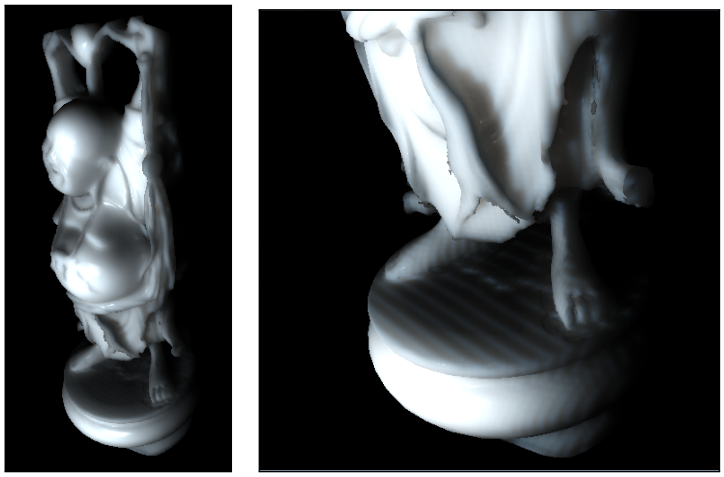

Figure 5.17: Artifacts at small incidence angles. The left side of the figure shows the whole model. The right side, shows the focused area, where artifacts arise due to under-sampling. (The right image's contrast and intensity have been increased for a better visualization in the printed document)

One way to overcome this problem can be to add additional sample points during rendering, in the geometry stage of the third pass. Enough information

to detect the under-sampled areas and to position new samples is obtained from the render textures in the first pass. A more detailed description of a possible solution in presented in Chapter 7 - Future Work.

5.3 Translucent Materials under Environment Illumination

The algorithm consists of a pre-rendering step in which the sample points are positioned on the model and the environment map is filtered, so a more suitable representation is obtained for interactive rendering. Opposed to point light illumination there will be only two main render passes on the GPU: the first in which the object is rendered and the set of x_o points is obtained and the second in which the scattering texture is created. In a last, deferred rendering pass, the translucent object is rendered with added specular highlights and the environment map is placed in the background.

5.3.1 Pre-rendering

Sampling the triangle mesh

To obtain the sample positions, random points are placed on the surface of the model according to a *cdf* based on the surface of each triangle. Once obtained, the sample positions are moved on the surface until they reach an uniform distribution.

The cumulative distribution function is an array with size equal to N_t, the number of triangles in the 3D mesh. Iterating over all triangle faces, each position i in the *cdf* is calculated as the sum of the areas of all triangles visited before it, divided by the total surface area:

$$cdf(i) = \frac{\sum_{j=1}^{N_t} A_{\triangle j}}{A_s}$$

It is intuitive that the values in the *cdf* will be in the $(0 \dots 1]$ interval, will be ordered, and that $cdf(N_t) = 1$. To choose a random point on the surface of the model, first a random triangle face will be chosen, according to the previously calculated *cdf*. For this, an uniform random number is generated between $[0 \dots 1)$ and then the *cdf* is checked to see which interval will the number belong

to. The index of the interval will yield the index of the triangle face. Because the triangle mesh can have a large number of triangles, since *cdf* is an ordered list, it is indicated to perform a binary search.

Now that a triangle has been chosen, the sample point is generated by sampling inside the triangle. This is done by generating two random variables (ξ_1 and ξ_2), again in the $[0 \dots 1)$ interval. From these, the barycentric coordinates of the point are generated:

$$\begin{aligned} \xi_1 &= rand(0,1) \\ \xi_2 &= rand(0,1) \\ u &= 1 - \sqrt{\xi_1} \\ v &= (1-\xi_2)\sqrt{\xi_1} \\ w &= \xi_2\sqrt{\xi_1} \end{aligned}$$

Considering a triangle with vertices A, B and C, the position of the sample point is:

$$P = Au + Bv + Cw$$

The total number of sample points can be approximated to [Jensen and Buhler, 2002]

$$N_s = \frac{A_s}{\pi l_u^2}$$

If the points are distributed uniformly, then the maximum distance between them will not exceed the mean free path, l_u, so if their number is increased, they will blur each other's light distribution . A much lower number of samples will introduce low frequency noise [Jensen and Buhler, 2002].

Uniform point distribution

The point relaxation procedure is applied for each sample point, and will be explained in detail for a single point, called in the remainder of the section *pivot point*. The two dimensional case will be considered first, and then the three dimensional case for a 3D mesh.

First, the neighbors of the pivot point situated in the circle of radius r surrounding it (henceforth called *influence zone*) will be identified. The neighbors will exert a repulsion force on the pivot point, and the sum of all forces will move it to a new location on the triangle mesh (see figure 5.18). It is important to note that the repulsion force may push a point outside the triangle containing it, so it will be moved to the neighboring triangle in the force's direction.

The process is repeated for all the sample points and their position is updated when all of them have been used as pivot points. Iterations of this procedure are needed so that the distribution will converge to uniformity. The algorithm in pseudo-code for the two-dimensional case is described in listing 5.10 below:

```
repeat
  for each point P_i
    identify neighbors
    calculate repulsion force F_i
  end for
  for each point P_i
    update position to F_i + P_i
    update triangle face for P_i
  end for
until uniform
```

Listing 5.10: The pseudocode for the 2D relaxation procedure

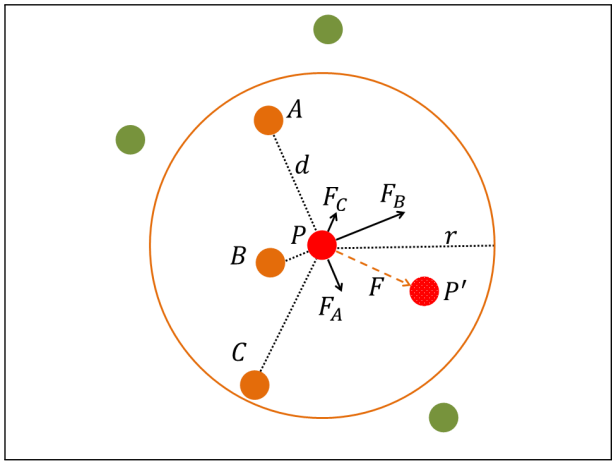

Figure 5.18: The pivot point (red dot), will be pushed according to the repulsion forces of its neighbors (orange dots). The green points outside the influence area (orange circle) are not considered to be influential. The pivot point will be moved according to the sum of the forces of it's neighbors

The lookup radius r is calculated as [Turk, 1991]:

$$r = 2\sqrt{A_s \cdot N_s}$$

where N_s is the number of sample points.

The force that a neighbor point exerts on the pivot point is calculated according to the lookup radius r. The closest the neighbor is, the more powerful the repulsion will be, while neighbors closer to the edge of the influence zone will have weaker repulsion forces. Therefore, the repulsion from a point on the border of the influence zone needs to be null. Every point further away will not be considered as influential. Several heuristics can be used for the equation of the repulsion force, but the simplest may be a linear fall-off according to the distance. As an example, considering the pivot point P, the neighbor point A and the distance between them d (presented in figure 5.18), the repulsion force F_A is:

$$F_A = \frac{P - A}{\|P - A\|} \cdot (r - d) \tag{5.3}$$

The force is therefore null if the neighbor point lies on the border, while if it is exactly near the pivot point, it pushes it to the other side of the influence area.

The new position of P is

$$P' = P + \vec{F}$$

The repulsion force (equation 5.3) is scaled by the distance from the neighbor point to the border of the influence zone. Having many points very close to the pivot point would push it away with a large magnitude force. Scaling down the total force with a small value solves this problem [Turk, 1991], but on the other hand the solution to the exact value is open and depends on the heuristic. Using a really small value would result in a slower convergence of the algorithm, while a large value would push the points back and forward between successive iterations (see table 5.4).

For a planar mesh, the neighbor lookup, calculating the repulsion force and moving the pivot point are straight forward since everything is in two dimensions. Rendering translucent materials, however, will rarely imply planar objects, and will almost always use three dimensional models. In this case, the procedure has a similar workflow but it needs slight modifications so that it will be locally two dimensional, meaning that it will be coplanar with respect to the pivot point's triangle. Below, listing 5.11 shows the algorithm in pseudo-code.

```
repeat
  for each point Pi
    identify neighbors
        rotate non-coplanar neighbors
    calculate repulsion force Fi
  end for
  for each point Pi
    push until Fi is null
    update position
    update triangle face for Pi
    update normal for Pi
  end for
until uniform
```

Listing 5.11: The pseudocode for the 3D relaxation procedure

Considering a pivot point P situated on the triangle face T_p, first the neighboring points are discovered. The influence zone is now the sphere of radius r centered at the pivot point. Some of the neighbors will be inside T_p or coplanar with T_p, while the rest will be outside of the plane. To calculate the repulsion force the points need to be coplanar, therefore the ones that lie outside T_p's plane will be rotated so that they are coplanar.

For a neighbor point situated on an adjacent triangle to T_p, the axis of rotation will be the common edge between them. The angle of rotation is the angle between the two triangle faces as shown in figure 5.19. For a neighbor point situated on a more remote triangle T_q that does not share any edge with T_p, the axis of rotation is the closest edge of T_p to the neighbor point. The angle of rotation is not the dihedral angle between the planes of each triangle face (because there are situations when they are parallel). Instead, a virtual plane is created from the closest edge of T_p and the neighbor point, and the scenario is reduced to the previous situation in which the triangles shared an edge. The angle of rotation is now the dihedral angle between the plane spanned by T_p and the virtual plane.

The rotation procedure is shown in figure 5.19. The same principle as in figure 5.18 can be applied to all neighboring points once they are coplanar, and the force F is calculated.

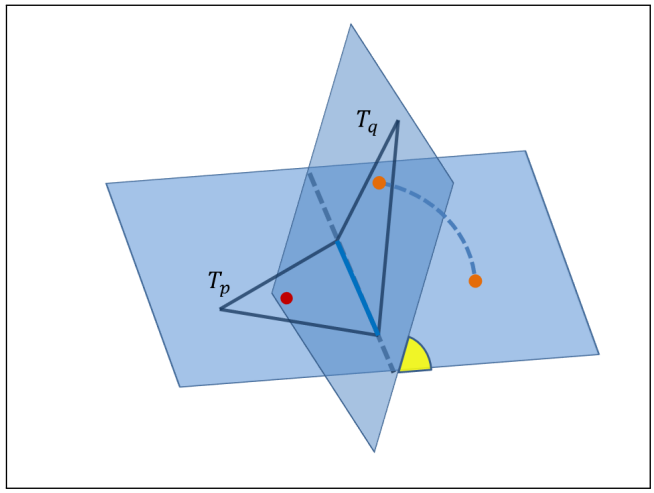

Figure 5.19: The procedure used to allign a neighbor point T_q (orange) with the plane in which the pivot point T_p (red) lies. The angle of rotation (yellow) is the angle between the T_p's plane and the plane defined by T_q and the closest edge (solid light blue) of T_p's triangle

The calculated repulsion force lies in the T_p plane. To move the point, first it is needed to check whether the new position P' is inside the triangle. A quick and useful solution is to check if the force vector $\vec{F}$ is intersecting any of the triangle's edges. If no intersection is detected, then P' is still in T_p. Otherwise, the intersection point P_i is calculated (see figure 5.20). The excess force (the dotted line) is then rotated over the adjacent edge until it is coplanar with the adjacent triangle face. Of course, the new position may not be in the adjacent triangle. This happens if the rotated excess force intersects another edge of the adjacent triangle, in which case the procedure is repeated.

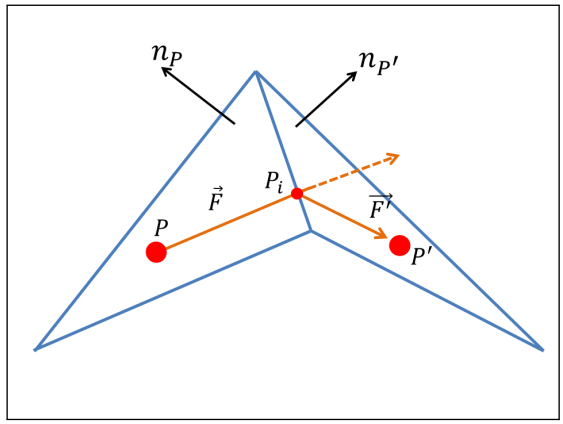

Figure 5.20: Rotating the excess force (dotted orange line) so that it is coplanar with the adjacent triangle

The point is moved across the faces of the model until the repulsion force is null. If the a border edge is crossed then the point is moved until the border and the procedure is stopped. Figure 5.21 shows the difference between a randomly sampled complex model and the same model with a uniform distribution. The uniform distribution was obtained using the described procedure.

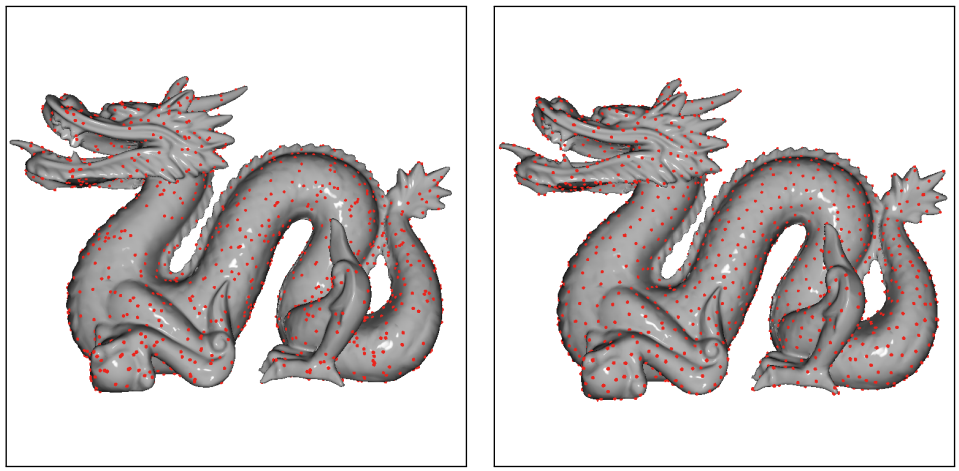

Figure 5.21: Uniform Distribution. The same model with a random distribution of sample points (left) and a uniform distribution of points (right).

Note that the mesh needs to be transformed to triangle mesh in case not all the faces are triangles. This is because the vertices that make up a face in a non-triangle mesh are not always coplanar and the point moving procedure will fail. To handle this, before sampling the set of points, the mesh is transformed to a triangle mesh. The rendering in the next stages can be performed on the initial mesh or the created triangle mesh.

Diffuse Illumination from the Environment

As presented in section 4.5.2 diffuse illumination from the environment will be obtained by filtering the environment map using the first 9 spherical harmonics coefficients [Ramamoorthi and Hanrahan, 2001]:

$$L_l^m = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} L(\theta,\phi) Y_l^m(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{5.4}$$

The above equation will result in 9 lighting coefficients that will be used in rendering. The numerical approximation for the integral (equation 5.4 is calculated using the theory from section 3.5.1. The integration domain $[0..\pi] \times [0..2\pi]$ is split into $height \times width$ equal subintervals of size $d\theta$ and $d\phi$, where $width$ and $height$ represent the map's dimensions respectively. Each subinterval will correspond to a texel in the map and the direction corresponding to it can be calculated by converting from spherical coordinates to Cartesian coordinates:

$$\omega(\theta,\phi) = \vec{d} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$$

The direction is then projected and the corresponding texture coordinates (u, v) of the texel in the environment map are calculated:

$$\begin{aligned} u &= r * \vec{d_x} \\ v &= r * \vec{d_y} \end{aligned}$$

where

$$r = \frac{1}{\pi} * \frac{\cos^{-1}\vec{d_z}}{\sqrt{\vec{d_x^2} + \vec{d_y^2}}}$$

The environment map is sampled at the (u, v) position and the incident radiance from direction $\vec{d}$ is obtained. This is done on the CPU side so normally the

texture would be stored in a two dimensional array of size $width \times height$, where each element has three components corresponding to the RGB channels. Indexing the two-dimensional array with the (u, v) coordinates is done by mapping them to integer coordinates (i, j) inside the $[0..width] \times [0..height]$ domain:

$$i = \lfloor u \cdot width \rfloor$$
$$j = \lfloor v \cdot height \rfloor$$

The result of the integration is done by summing the values for each $d\theta$ and $d\phi$. The final value of each coefficient L_l^m is normalised by multiplying with $d\theta \times d\omega$. The numerical formula for L_l^m therefore is:

$$L_l^m = \sum_{\theta=0}^{\pi} \sum_{\phi=0}^{2\pi} L_l^m(\theta, \phi) Y_l^m(\theta, \phi) \sin(\theta) d\theta d\phi$$

where

$$d\phi = \frac{\pi}{width}$$
$$d\theta = \frac{2\pi}{height}$$

The authors do not offer a solution for the integral in the paper [Ramamoorthi and Hanrahan, 2001]. Instead they provide the source code for the pre-filtering stage in which they use a different method for calculating the lighting coefficients L_l^m. However the results obtained in this thesis and the ones obtained with the author's code are very close (see tables 5.1 and 5.2).

	Eucalyptus Grove						
	Author's results			Thesis results			Maximum Deviation
L_0^0	.38	.43	.45	0.3792	0.4271	0.4517	0.0029
L_1^{-1}	-.29 [2]	-.36[2]	-.41[2]	-0.2888	-0.3587	-0.4146	0.0112
L_1^0	.04	.03	.01	0.0387	0.0303	0.0105	0.0013
L_1^1	-.10	-.10	-.09	-0.1034	-0.1032	-0.0884	0.0034
L_2^{-2}	.06 [2]	.06[2]	.04[2]	0.0619	0.0551	0.0394	0.0049
L_2^{-1}	-.01 [2]	.01 [2]	.05 [2]	-0.0079	0.0147	0.0471	0.0047
L_2^0	-.09	-.13	-.15	-0.0932	-0.1250	-0.1522	0.0050
L_2^1	-.06	-.05	-.04	-0.0583	-0.0514	-0.0376	0.0024
L_2^2	.02	-.00	-.05	0.0215	-0.0031	-0.0438	0.0062

Table 5.1: The nine lighting coefficients obtained by the authors (left column), this thesis (middle column) and the maximum deviation between the two (right column). The coefficients are in RGB format. The environment map used was Eucalyptus Grove, source: http://www.pauldebevec.com/Probes/

	St. Peter's Basilica						
	Author's results			Thesis results			Maximum Deviation
L_0^0	.36	.26	.23	.3637	.2627	.2328	0.0037
L_1^{-1}	-.18[2]	-.14[2]	-.13[2]	-.1784	-.1443	-.1266	0.0043
L_1^0	-.02	-.01	-.00	-.0257	-.0106	-.0014	0.0057
L_1^1	.03[2]	.02[2]	.01[2]	.0348	.0224	.0101	0.0048
L_2^{-2}	-.02	-.01	-.00	-.0198	-.0143	-.0043	0.0043
L_2^{-1}	.05[2]	.03[2]	.01[2]	.0489	.0263	.0124	0.0037
L_2^0	-.09	-.08	-.07	-.0917	-.077	-.074	0.0083
L_2^1	.01	.00	.00	.0043	.0033	-.0004	0.0033
L_2^2	-.08	-.0322 [2]	.00	-.0843	.0362	.0029	0.0043

Table 5.2: The nine lighting coefficients obtained by the authors (left column), this thesis (middle column) and the maximum deviation between the two (right column). The coefficients are in RGB format. The environment map used was St. Peter's Basilica, source: http://www.pauldebevec.com/Probes/

With the presented method it is possible to represent the diffuse illumination

[2]The values obtained with the source code provided by the authors on http://graphics.stanford.edu/papers/envmap/ differ in sign from the values presented in their paper [Ramamoorthi and Hanrahan, 2001]

from any environment map with using just 9 spherical harmonics coefficients. The lighting coefficients have been calculated as well for another environment map:

	Ufizzy Gallery Light Probe		
L_0^0	0.4763	0.4621	0.5257
L_1^{-1}	-0.5572	-0.5530	-0.6445
L_1^0	-0.0055	-0.0051	-0.0063
L_1^1	-0.0122	-0.0127	-0.0171
L_2^{-2}	0.0280	0.0283	0.0370
L_2^{-1}	0.0169	0.0158	0.0183
L_2^0	-0.4197	-0.4140	-0.4788
L_2^1	0.0009	0.0006	0.0004
L_2^2	-0.3626	-0.3615	-0.4262

Table 5.3: The nine lighting coefficients obtained with the method presented in this thesis. The environment map focused was Uffizi Gallery. Source: http://www.pauldebevec.com/Probes/

The values in the tables were used to render three spheres in scenes with three different environment maps. Figure 5.22 shows the diffuse contribution from the environment maps in the three distinct cases.

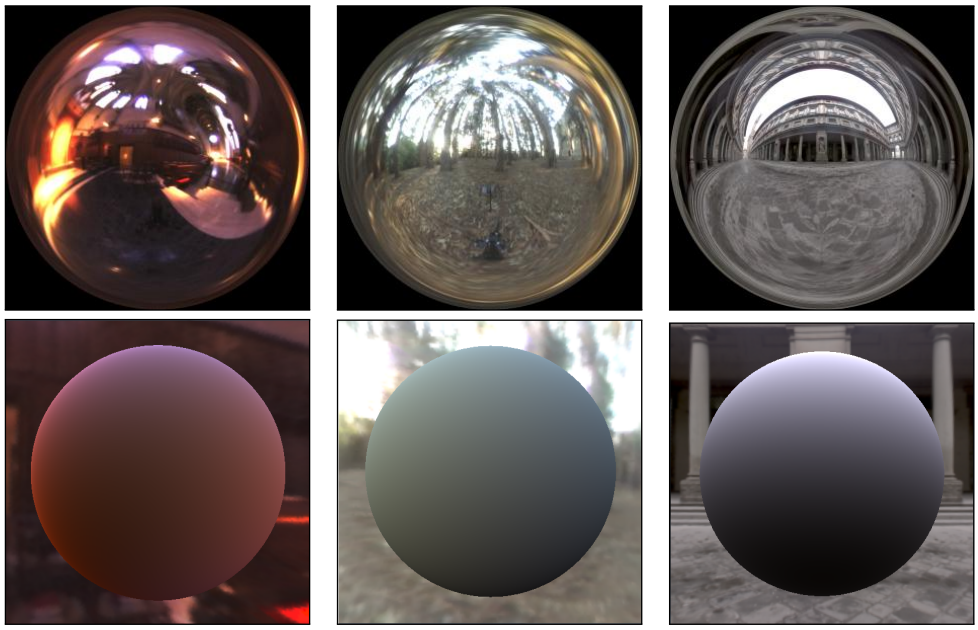

Figure 5.22: Spheres rendered with diffuse illumination from each of the three environment maps, using the described method

5.3.2 Rendering

After the pre-rendering step in which the points are uniformly distributed across the surface of the model and the environment map is filtered, the rendering of the scene can start.

The process is composed of three render passes: the first one renders the translucent object to obtain the x_o points; the second in which the scatter texture is generated and the third in which the scatter texture and specular highlights are applied on the model and the environment map is rendered on the screen. The third pass is done as in deferred rendering.

The total outgoing radiance L_o will be composed of two terms: the diffusion term and the specular term.

$$L_o = L_{diffusion} + L_{specular}$$

The diffusion term, $L_{diffusion}$, corresponds to the outgoing radiance calculated with the dipole approximation (presented in section 3.4.1) and it will be stored in the scatter texture. The filtered environment map is used to calculate the diffuse incoming radiance on the model's surface. $L_{specular}$, the specular term, will be added in the deferred rendering stage, and obtained from the normal, unfiltered, environment map.

In the first render pass, into a double frame buffer, the object is rendered and the fragment shader will output to two different render targets eye space position and eye space normals. This render pass is done from the eye's point of view and the render targets are the size of the screen. Also, besides the model, it is needed to draw the environment. This is done by drawing a large enough sphere that will cover the whole viewport and the model. Again, the last channel in the position texture will act as a flag so a distinction between the model and the environment is made in the final render pass.

In the second render pass, the sample points are sent to the GPU. Object space position and object space normals are sent as vertex attributes (as seen the following code sample 5.12).

```
glBegin (GL_POINTS);
  for (int i = 0 ; i < samples.size(); i ++)
  {
    glNormal3fv (normals[i].get());
    glVertex3dv (samples[i].get());
  }
glEnd();
```

Listing 5.12: Sending the sample positions to the GPU

In the vertex shader, the object space position is transformed to eye-space position by multiplication with the Model-View Matrix. As described in the previous section for point light illuminations, in the geometry shader they will be expanded to screen aligned quads centered at the sample point.

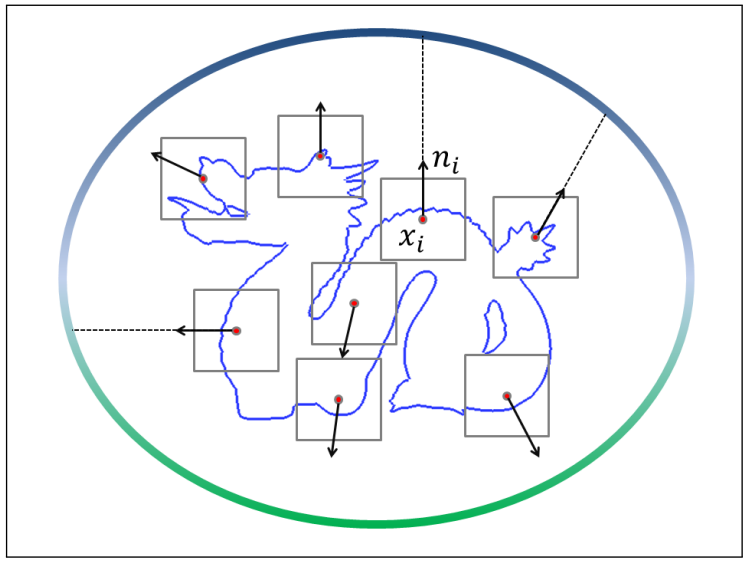

Figure 5.23: The 3D model surrounded by the environment map. The sample points x_i(red dots) are positioned on the model and splats (gray rectangles) are centered on each of them. The contribution from the environment map is obtained by sampling in the normal direction of the sample point, n_i. Again, the splats are oriented towards the viewer.

In the fragment shader, the splats are shaded in a similar manner as in point light illumination. The precomputed lighting coefficients L_l^m are sent as uniforms to the shader because they are dependent on the environment map used. The constants c_i are static variables in the shader and radiance is calculated using equation 4.14 for the eye space normal:

```
vec3 n = normal_es.xyz; // eye-space normal
vec3 E = c1 * L22 * (n.x * n.x - n.y * n.y) +
                c3 * L20 * (n.z * n.z) +
                c4 * L00 -
                c5 * L20 +
                c1 * 2 * (L2_2 * n.x * n.y + L21 * n.x * n.z +
                    L2_1 * n.y * n.z) +
                c2 * 2 * (L11 * n.x + L1_1 * n.y + L10 * n.z);
```

Listing 5.13: Irradiance from Environment Map

Figure 5.23 shows the process from the observer's perspective: sample points (or the set of x_i points in equation 4.1) are positioned on the model (in the pre-rendering stage) and splats are created (during the second rendering pass). The filtered (diffuse) environment map is sampled in the sample point's normal direction to obtain the diffuse illumination on the sample point.

The OpenGL state used in this render pass is the same as the one described in the previous section: the depth test is disabled and additive blending is enabled so that each pixel will receive the contribution from the sample points close to him. After this pass is finished, the depth test is enabled and additive blending is disabled.

Next, the deferred rendering stage starts, similar to the one used in point light illumination. A screen sized quad is drawn and the pixels it spans will be shaded depending on what part of the scene they cover.

On the translucent objects, specular highlights are added, as the projection of the environment map on the surface, weighted by the Fresnel term (described in section 3.2.3).

5.4 Rendering with Environment Illumination - Discussion

The results obtained using the presented integration method used for the pre-filtering step are very close numerically to the ones presented by the authors in their paper. Column three of table 5.1 and table 5.2 show that the error is small and it most probably arises from rounding. The diffuse illumination (or low frequency illumination) from the environment map on a model used in rendering is shown in figure 5.24.

Figure 5.24: Diffuse illumination from the environment map on a complex 3D model

Note that the pre-filtering step was finished in under a few seconds for a 1000×1000 pixels environment map.

The point relaxation procedure needs a variable number of iterations to reach uniformity. If the repulsion forces are scaled down too much, the points will slowly move towards a stable uniform distribution but they will reach it after a large number of iterations. On the other hand, if the force is not scaled down enough, points will rapidly move across the surface, but may never reach a stable uniform distribution. The following table 5.4 shows the approximate number of iterations needed for a number sample points to reach uniformity on the specified model.

Model	Polygon Count	Sample Points	Scaling Factor	Iterations (appx.)	Running Time (appx.)
Stanford Dragon	100 000	10 000	0.001	60	17 sec.
Stanford Dragon	100 000	10 000	0.01	10	4 sec.
Stanford Dragon	100 000	10 000	0.1	did not converge	-
Buddha Statue	100 000	10 000	0.001	100	18 sec.
Buddha Statue	100 000	10 000	0.01	25	6 sec.
Buddha Statue	100 000	10 000	0.1	did not converge	- sec.
Cube	16	4 000	0.001	40	5 sec.
Cube	16	4 000	0.01	10	1 sec.
Cube	16	4 000	0.1	did not converge	- sec.

Table 5.4: The number of iterations and time needed for the point distribution to converge for different 3D models.

It is important to note that the number of polygons in the 3D mesh is not the main parameter in the running time of the relaxation procedure. The level of tessellation is important when a point is pushed outside it's triangle. The number of sample points, on the other hand, dictates the computational cost. The neighbor lookup for each pivot point would require, in a naive implementation, to iterate over the whole set of sample points and check for each of them if they lie inside the influence zone. Using an acceleration structure to store the points (for example a $kd-tree$ [Bentley, 1975]) will reduce the neighbor look up cost. Even if the structure needs to be recreated every iteration, better running times are expected compared to the naive implementation.

Figure 5.25 shows the Stanford Dragon model rendered with illumination from the Eucalyptus Grove environment map. The points have not been distributed uniformly and low frequency noise is visible on the model.

Figure 5.25: Translucent object rendered with diffuse illumination from the environment map. The sample points have not yet been uniformly distributed and low frequency noise is highly visible.

Applying the point relaxation procedure, after a few iteration cumulating approximately 4 seconds, the noise is eliminated. The rendering is shown in figure 5.26.

The specular highlights are added and the final rendering is shown in figure 5.27.

Figure 5.26: Translucent object rendered with diffuse illumination from the environment map. The sample points have been uniformly distributed using the point relaxation procedure described in this section. The low frequency noise is not visible anymore

Figure 5.27: The translucent model rendered as above, but with added specular reflections from the environment map

Chapter 6

Results and Validation

This chapter presents the results obtained using the algorithm described in the previous sections. The first section will focus on performance and the second section will focus on the visual quality of the results. In the third and last section, the method is validated by comparing its results (both performance and visual accuracy) to the results from other methods.

All the tests and renderings have been done on the same hardware system: Intel Core i5 @ 3.1 Ghz with 4GB RAM, running a 64 bit operating system. The graphics hardware used was nVidia Quadro 600. Unless otherwise specified, the resolution of the render targets and final rendering frame buffer was 512×512 pixels.

6.1 Performance

For point light illumination, after the diffusion profiles are calculated and stored into textures, except the OpenGL draw calls, the CPU will be responsible just for the generation of the sampling pattern. Therefore, the main computational effort is done on the GPU, and the performance will be dependent on the amount of calculations executed in the shaders. The render pass in which the scattering texture is created is the main candidate for the bottle neck, since the other are

trivial for any modern GPU. Using gDEBugger as a profiling tool, the supposition was confirmed, therefore the following tests focused on the render pass responsible for creating the scattering texture.

During the creation of the scattering texture, there are three parameters that influence the rendering time: the number of splats, the size of an individual splat, and the number of pixels the object occupies on the screen. This is based on the observation that with increasing number of splats and increasing splat size, an increasing number of splats will overlap on a screen pixel. Thus, more blending operations will be done per pixel. With the increasing number of screen pixels occupied by the object, the overall rendering time will increase.

First, the running time has been analyzed for a 3D model rendered with varying number of splats, a fixed splat size and fully focused in the observer's viewport. The values are presented in the table below:

			Number of splats				
Model	$\triangle\#$	Method	55×38	65×45	75×52	85×60	95×67
Dragon	10^5	Dipole	62.1	48.9	38.9	30.5	25.4
Dragon	10^5	Multipole	40.1	30.7	24.1	19.6	15.2

Table 6.1: Frame rates for rendering a single 3D model with varying number of samples. The middle column from the results corresponds to the minimum number of splats for which no artifacts were visible.

The middle column in table 6.1 corresponds to the number of splats needed for the integral to converge and the model to appear smooth and without sampling artifacts. Note that the algorithm runs in real-time for the dipole approximation method (approximately 39 frames per second) and at an acceptable interactive frame rate for the multipole approximation (24 frames per second). The overhead in the multipole method is because the extra render pass needed to calculate the object's thickness.

Below the threshold in column 3 of table 6.1, the model is under-sampled. Though the frame rates increase, the visual results are not satisfying. Above the threshold the visual results will not change, so the number of samples is not indicated to use.

Figure 6.1 shows the difference between the under-sampled model (left) and the correctly sampled model (right). Increasing the number of splats will not provide additional detail or visual effects, but instead will decrease the performance, as presented in column 5 of table 6.1.

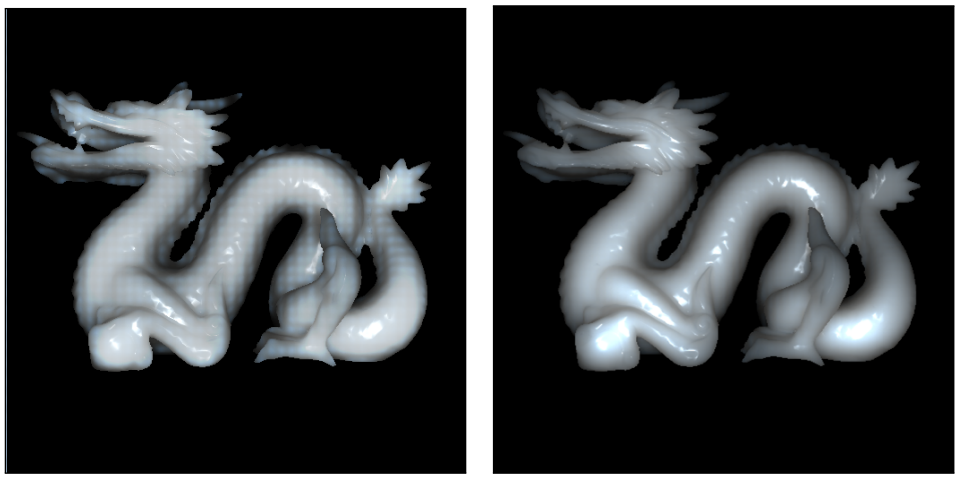

Figure 6.1: The Stanford Dragon 3D model. In the left image, the model is under sampled and artifacts are visible. The right image was rendered with the optimal number of splats at interactive frame rates. Higher number of splats will not offer better visual results, but the frame-rate will drop.

To use the render target from the first render pass as a shadow map, its resolution needs to be increased for a better visual result and for more accurate shadows. Table 6.2 shows the rendering frame rates for a complex 3D model rendered using the dipole and multipole method, and a number of splats of 75×32. Note that even for a texture size of 2048×2048 the algorithm still runs in real-time for the dipole method (26 frames per second) and at interactive frame rates for the multipole method (19 frames per second).

			Texture Size			
Model	$\triangle\#$	Method	512×512	1024×1024	2048×2048	4096×4096
Dragon	10^5	Dipole	38.9	35.2	26.1	13.8
Dragon	10^5	Multipole	24.1	22.8	19.0	11.4

Table 6.2: Frame rates for rendering a complex 3D model with a fixed number of samples and varying render target size.

The following table shows that having a different model, the number of samples needed to reach convergence (shown in column 3) is different. The bunny model, even though it has a lower polygon count will be rendered without artifacts at a similar frame rate as the previous model:

△#	Method	Number of splats 30×29	35×34	40×39	45×44	50×49	55×54
69451	Dipole	62.1	48.8	40.6	31.2	27.0	21.5
69451	Multipole	39.2	30.6	25.2	20.6	16.5	13.1

Table 6.3: Frame rates for rendering a single 3D model with varying number of samples. The minimum number of splats for which no artifacts were visible is 40×39. The frame rate at which the model is rendered without artifacts is close to the one presented in the previous table.

Figure 6.2 shows the difference between under-sampling (left) and correct sampling (right).

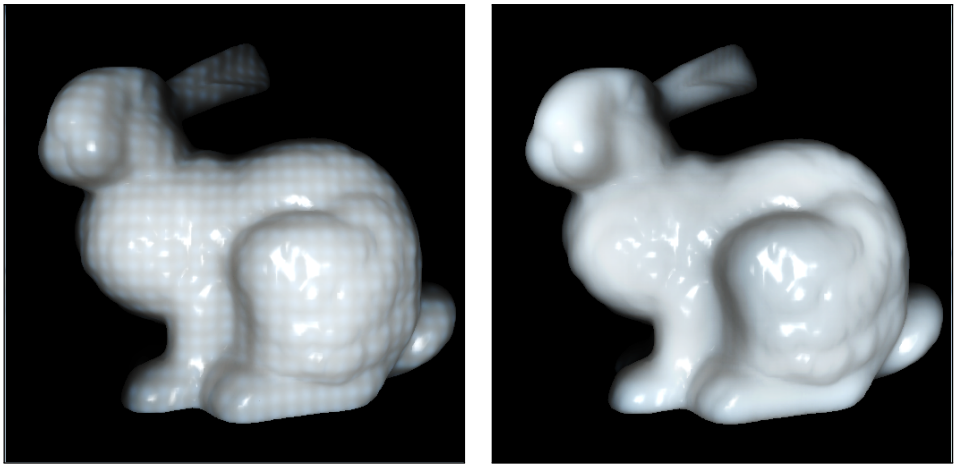

Figure 6.2: The Stanford Bunny 3D model. In the left image, the model is under sampled and artifacts are visible. The right image was rendered with the optimal number of splats at interactive frame rates. Higher number of splats will not offer better visual results, but the frame-rate will drop.

The splat size is calculated according to the material's diffusion properties, Therefore, it is normal that the frame rate is dependent on the material properties. Tables 6.4 and 6.5 present the frame rates obtained for rendering two different materials: marble and an empirically chosen material to resemble green jade. The visual quality was evaluated for each material at each splat count and the values at which algorithm converges were placed in the middle column of each table. It is important to note that the visual results converged at approx-

imately the same frame rate. Figure 6.3 shows the renderings for the Buddha model using green jade and marble properties, both for the under-sampling case (left) and the optimal splat count number (right).

			Number of splats				
Model	$\triangle\#$	Method	40×97	55×137	70×170	85×206	100×243
Jade	10^5	Dipole	98.1	60.9	40.6	28.4	21.3
Jade	10^5	Multipole	58.1	34.8	22.6	15.8	11.6

Table 6.4: Frame rates for rendering the Buddha 3D model with varying number of samples, using scattering properties of green Jade. The middle column from the results corresponds to the minimum number of splats for which no artifacts were visible.

			Number of splats				
Model	$\triangle\#$	Method	20×48	25×60	30×72	35×85	40×97
Marble	10^5	Dipole	94.1	67.5	49.1	38.5	30.1
Marble	10^5	Multipole	59.1	41.8	30.2	23.3	18.2

Table 6.5: Frame rates for rendering the Buddha 3D model with varying number of samples, using scattering properties of marble. The middle column from the results corresponds to the minimum number of splats for which no artifacts were visible.

The next parameter that influences the rendering time is the splat's dimension. In table 6.6, the frame rates were measured for varying splat size. The smallest splat size considered is 40% of the initial splat size calculated in the pre-rendering stage. Scaling down the splat size corresponds to scaling up the 3D model. Each row in the table corresponds to the frame rates for a particular splat scale, while each column represents the number of splats needed for the algorithm to converge at that splat size. Note that the frame rates at which the algorithm converges in the different scenarios are therefore the first diagonal, and they are roughly the same.

△#	Method	Splat Size	Number of splats 75×32	92×64	129×90	150×105
10^5	Multipole	1	24.0	16.6	8.9	6.9
10^5	Multipole	0.8	34.9	24.5	13.3	9.9
10^5	Multipole	0.6	54.9	39.6	22.1	16.8
10^5	Multipole	0.4	96.1	73.5	43.6	24.4

Table 6.6: Frame rates for rendering the Stangford Dragon 3D model with varying number of samples, using scattering properties of marble. The splat size varies vertically and the sample count horizontally. The first diagonal of the results corresponds to the frame rate for which the model was rendered without artifacts. Below the diagonal, artifacts are visible though the frame rates are higher. Above the diagonal, samples are redundant and the visual results do not change, while the frame rate drops.

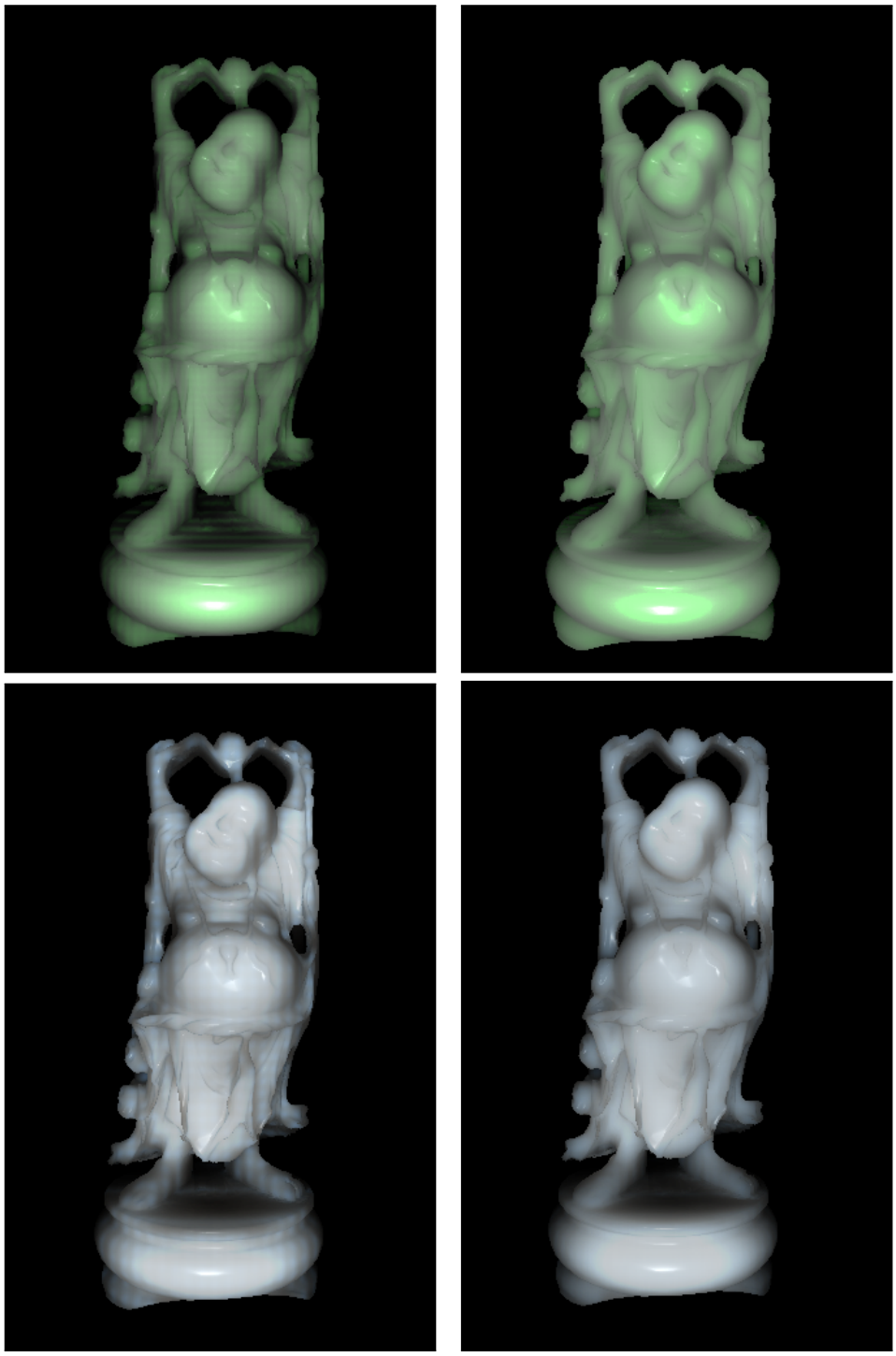

Figure 6.3: The Buddha 3D model, rendered with green jade material properties (top row) and marble (bottom row). The images in the left column show artifacts due to under-sampling. The right column shows the models rendered with the optimal splat count and no

The next test was done to evaluate the frame rate when the object does not fully occupy the observer's viewport. The results (shown in table 6.7) confirm that the rendering time is dependent on the positioning of the object. This is because as the object gets closer and occupies a larger area on the screen, the number of pixels for which blending is performed increases and induces higher overhead.

			Distance to the observer		
△#	Method	# Splats	Very Close	Moderate	Far
69451	Multipole	50×49	16.5	28.8	78.2
69451	Multipole	60×59	11.7	20.4	57.3
69451	Multipole	70×69	8.7	15.2	44.8
69451	Multipole	80×79	6.8	11.9	34.8

Table 6.7: Frame rates for rendering the Stanford Bunny 3D model, with varying number of splats, and with different positioning in the observer's viewport. The first line of the results shows that interactive frame rates are achieved even if the model is very close and occupies most of the viewport.

Figure 6.4 shows the object's position for which the frame rates have been measured.

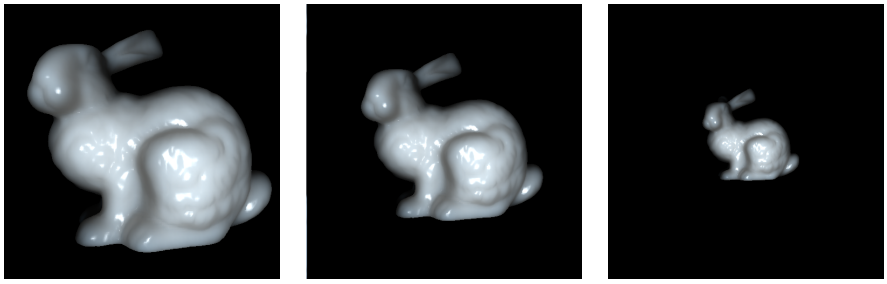

Figure 6.4: The Stanford Bunny 3D model. From left to right, the model is positioned very close, moderate, and far with respect to the observer.

For environment illumination, the frame rate has been measured for varying splat size and varying number of splats. The object's positioning on the screen will influence the rendering time in the same manner as presented in table 6.7. The first column in table 6.8 lists the rendering times for a number of splats calculated as in section 5.3.1. The following columns show the measurements for an increased number of samples, up to double the initial number. A lower

number of samples than the one in the first column was not of interest since low frequency noise appears because of under-sampling. The first line corresponds to the normal size calculated in the pre-rendering stage, while the measurements in the next lines correspond to smaller splats, from 80% to 40% of the normal size. The renderings are presented in figure 6.5 both for the under sampled case (focusing on the artifacts) and the optimal scenario.

		Number of splats			
$\triangle\#$	Splat Size	5789	6947	8105	9263
100000	1	16.7	13.1	11.1	9.4
100000	0.8	22.1	19.3	16.4	14.0
100000	0.6	35.8	31.7	26.9	23.1
100000	0.4	70.1	60.2	53.2	45.6

Table 6.8: Frame rates for rendering the Stanford Dragon model under environment illumination with the multipole method. The splat size varies vertically, while the sample number varies horizontally. The first diagonal of the results in the table corresponds to the frame rates at which the model rendered does not present artifacts.

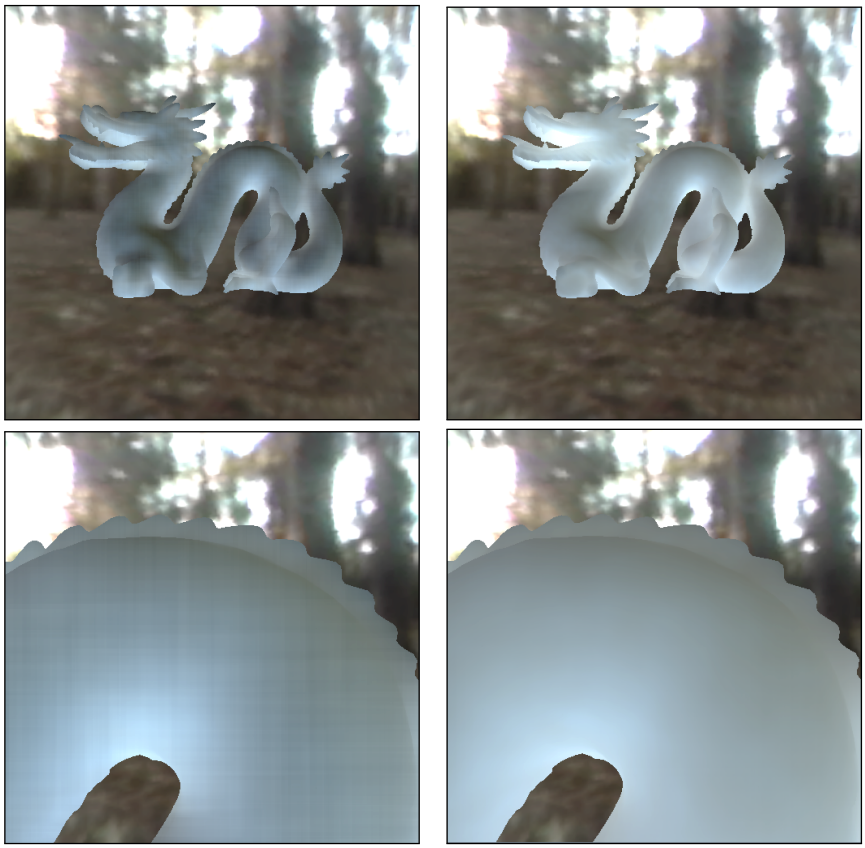

Figure 6.5: The Stanford Dragon 3D model rendered under environment illumination. The left column shows the artifacts that appear when the splat size is lower than the one calculated, while the right column shows the model rendered correctly. The bottom row focuses on the same region of the 3D model to emphasize for comparison between the two scenarios.

6.2 Visual Results

This section shows the visual results of the rendering algorithm presented and it focuses on the different appearence of materials with different scattering properties and the different appearence of a few 3D models under different illumination directions. More images rendered with this algorithm are found in appendix A.

The materials chosen to illustrate the algorithm were marble and green jade.

6.2.1 Translucent Materials under Point Light Illumination

First, the results from the dipole and multipole methods will be compared. In figure 6.6 the Stanford Dragon model is rendered using the dipole method and the scattering coefficients for marble.

Figure 6.6: The Stanford Dragon 3D model rendered with the Dipole method. When the model is illuminated from behind translucency is visible, and the thin surfaces allow more light to exit.

The upper right corner of both images in figure 6.6 shows the model rendered with simple Phong shading from the light's perspective. This demonstrates that the left side of the figure was rendered with front illumination while on the right side the model is back-lit. Note that with the translucent shader the object appears diffuse and shiny and simulates surface-scattering.

The dipole method does not account for transmittance separately, thus, when the model is back-lit (right side of figure 6.6) the transmittance is approximated with the reflectance term.

The multipole method uses separate terms for reflectance and transmittance. Thus, the visual appearence for back-lit objects is more accurate (see figure 6.7). The left side of figure 6.7 shows that front illumination is similar regardless of the method used.

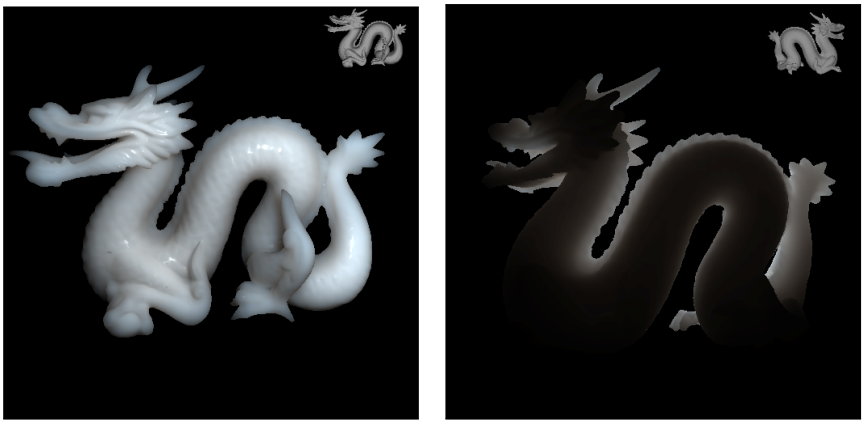

Figure 6.7: The Stanford Dragon 3D model rendered with the Multipole method. When the model is illuminated from behind translucency is more accurately approximated using the Multipole method.

Figure 6.8 shows the Buddha Statue model rendered with material properties that resemble green jade. Note that when the model is illuminated from behind, the translucency is green.

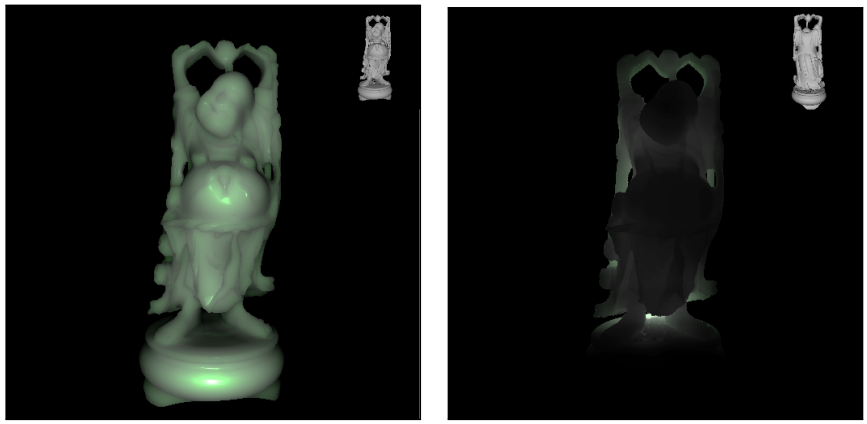

Figure 6.8: The Buddha 3D model rendered with the dipole method and green jade material properties. Note the distinctive shiny and green appearence when the model is rendered from the viewing direction and the green transmittance when the model is back lit.

The multipole method allows to render thin slabs and thus it is possible to render models covered with thin coating layers. Figure 6.9 shows a 3D model rendered with marble properties (left) and marble in a white coating layer (right).

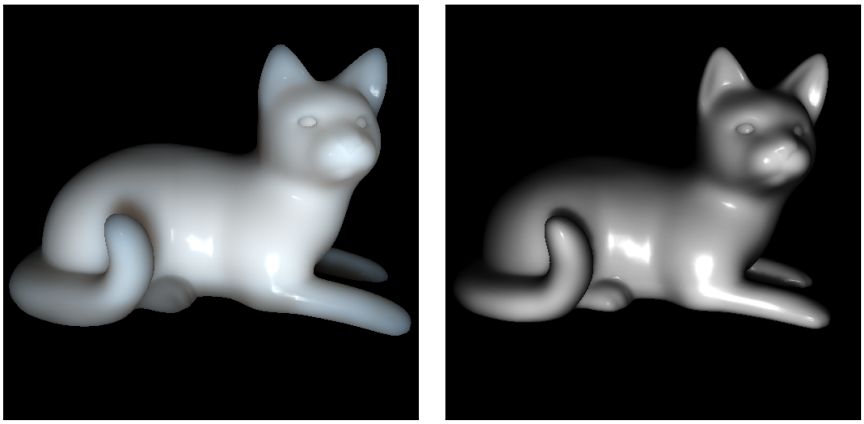

Figure 6.9: Cat 3D model rendered with marble material properties (left) and marble covered in a thin white layer (right). Translucency is not visible on the thin parts when the model is covered in the white layer.

Note that the left image in figure 6.9 presents translucency on thin area of the model (the tip of the tail and the tip of the ears). When rendering with the coating layer, the model is less translucent and almost opaque.

When rendering with the multipole approximation, the thickness of the model can be adjusted by scaling the distance d (the mean distance the light travels inside the medium). Thus, varying scattering properties and thickness can be simulated at runtime, without re-calculating the diffusion profiles. Figure 6.10 shows a 3D model with side illumination and thickness decreasing from the top-left corner down to the bottom-right corner.

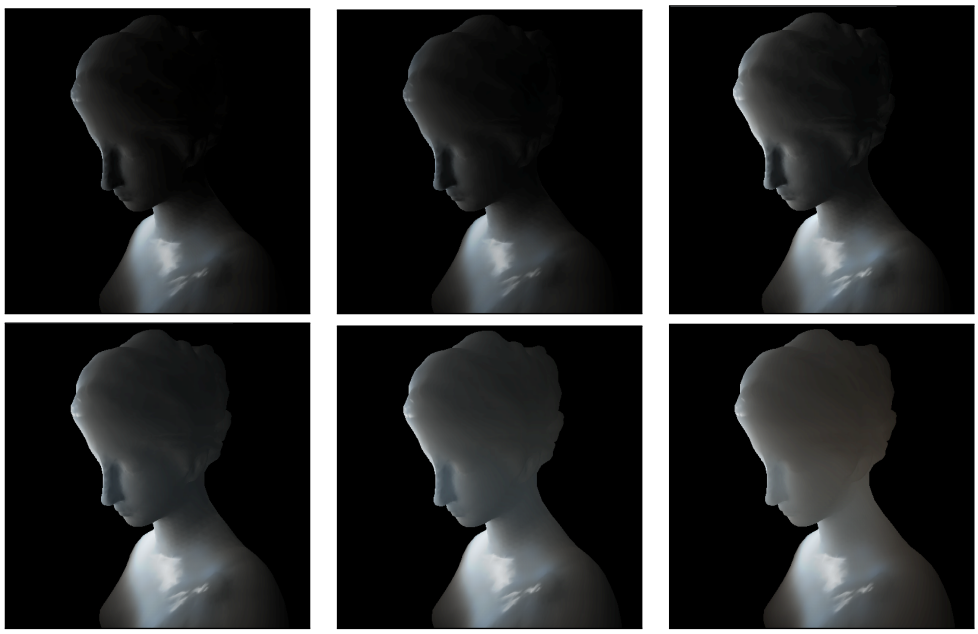

Figure 6.10: The statue model rendered with altered thickness. The thickness is down-scaled from the image in the top-left corner to the image in the bottom-right corner and accordingly the translucency of the object increases. Varying thickness can be adjusted at runtime.

Varying material properties can be simulated during runtime, without recalculating the diffusion profiles, by scaling the look-up radius and the splat size accordingly. This actually corresponds to scaling the 3D model before rendering. Figure 6.11 illustrates the success of the method. The same model was rendered under the same illumination and, but scaling the parameters, its appearence changes from almost opaque to very translucent.

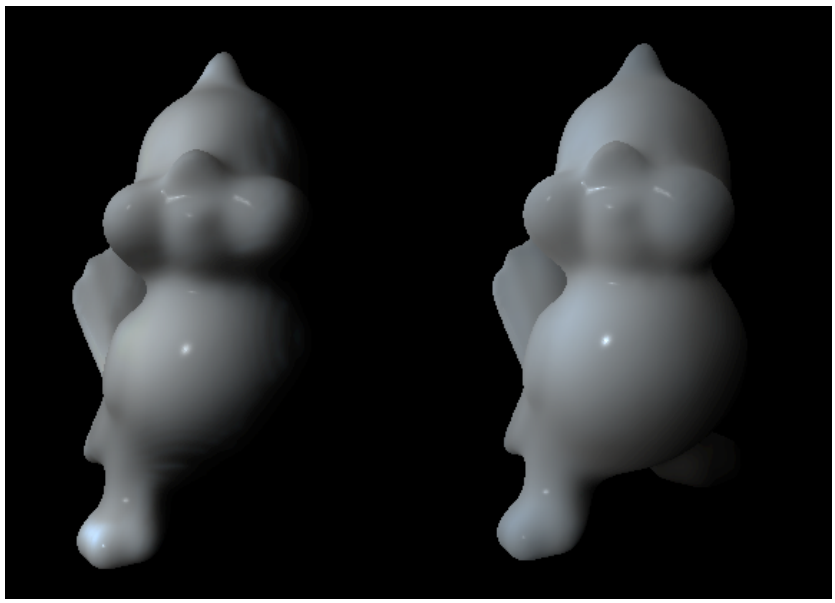

Figure 6.11: The bird model rendered with varying splat size and look-up radius. Scaling both parameters corresponds to altering the object's size and/or the object's material properties.

6.2.2 Translucent Materials under Environment Illumination

With the filtering method and the algorithm presented in section 5.3.1 and the point relaxation procedure in 5.3.1 , models can be rendered under diffuse illumination from the environment map. The following figures will show the renderings from different orientations and environment maps, for the diffuse illumination (top) and the translucent objects under environment illumination (bottom). The specular highlights and tone mapping are applied in the deferred rendering stage.

Figure 6.12: The Stanford Dragon model rendered with simple diffuse illumination from the environment map (top) and with translucent material properties (bottom). Note that the diffuse illumination corresponding to the areas in the environment map: the green-brown shades from the ground and the blue shades from the sky and the forest's canopy.

In the following figure, the model is oriented towards the brighter area of the environment map. The top part of the figure, showing only the diffuse component with a simple shader, shows that illumination on the visible side of the model is mostly dark, from the leaves and the ground. The bottom part image, where the translucent shader was used and specular highlights were added, shows that the thin parts of the model (the tip of its back) are highly lit, even though they are not oriented towards the bright areas. This is due to scattering and corresponds to the single point-light illumination scenario presented in figure 6.7, when the model was illuminated from behind.

Figure 6.13: The Stanford Dragon model rendered with simple diffuse illumination from the environment map (top) and with translucent material properties (bottom). The thin parts of the Dragon's back are lit using diffuse illumination that corresponds to the sky and the forest's canopy and the appearence is similar to the back-lit illumination, since the top image is not bright in those areas.

The images in figure 6.14 were rendered with illumination from an environment map filtered with the procedure presented in section 5.3.1. Note that in the top image of figure 6.14 (the diffuse illumination), the dark areas on the model do not correspond to self shadowing, but to the dark areas in the lower part of the environment map. The bottom image shows the model rendered with marble properties, hence its distinctive diffuse and shiny appearence.

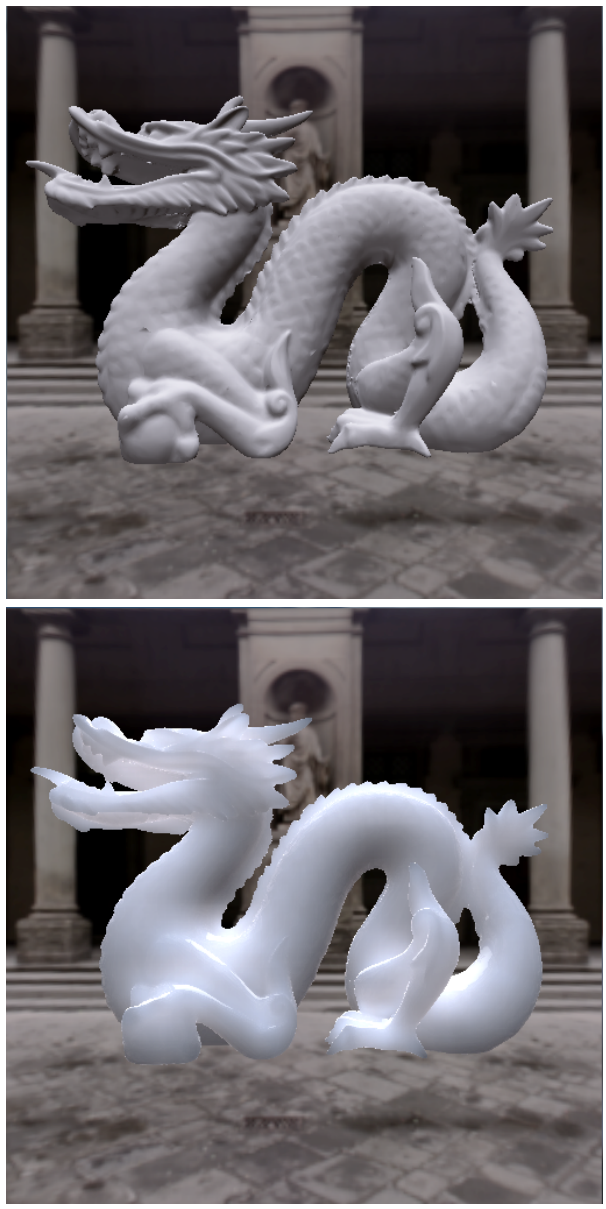

Figure 6.14: The Stanford Dragon model rendered with simple diffuse illumination from the environment map (top) and with translucent material properties (bottom). A new environment map was filtered and its contribution was reduced to a set of lighting coefficients. The bright areas on the renderings correspond to illumination from the sky while the dark areas receive diffuse illumination from the ground.

6.3 Validation

The following section will validate the method and the implementation presented in this thesis against previous work in real-time rendering of translucent materials or against off-line renderings of similar scenes. Both performance and visual results will be considered to assess the validity of this implementation. It can be considered that the validation process has started in the early stages of the implementation: figure 4.5 showing the plots for the reflectance and transmittance profiles match their corresponding plots in [Donner, 2006]. As well, it was shown that the results of the filtering method used in this thesis only have a minor, neglectable deviation from the values given by [Ramamoorthi and Hanrahan, 2001]. Next, the visual results will be compared.

Figure 6.15 compares the Bird model rendered with this implementation (top) and the results from [Shah et al., 2009]. The orientation of the model is different, as it was difficult to obtain the same camera perspective. The images, though, are similar and the distinctive traits of translucent materials are highly visible in both implementations. Figure 6.16 presents the same similarities between the two images, in which the models have been rendered with other material properties. The left image in figure 6.16 uses empirically chosen material properties.

Figure 6.17 compares the rendering of the Bird model using the implementation presented in this thesis (top) and an off-line method (bottom). The results are similar, though the real-time method lacks inter-reflections (visible on the bird's neck). However, the top image was rendered at 22 frames per second while the off-line method presents visible noise after more that 10 hours of rendering.

Figure 6.18 shows the difference between the results from [Dachsbacher and Stamminger, 2003] (first row) and this thesis (second row). Visually, the results on the second row are better, especially for the second column where the material properties have been modified to exhibit more translucency. Note that the diffraction-like pattern visible on the model's tail in the top-left image is not present in this thesis' renderings.

The rendering times obtained in this thesis are presented in the tables throughout section 6.1. The interactive and real-time frame rates are obtained for models rendered with different material properties. Note that even when the size of the render textures was increased up to 2048×2048, the running time was still interactive.

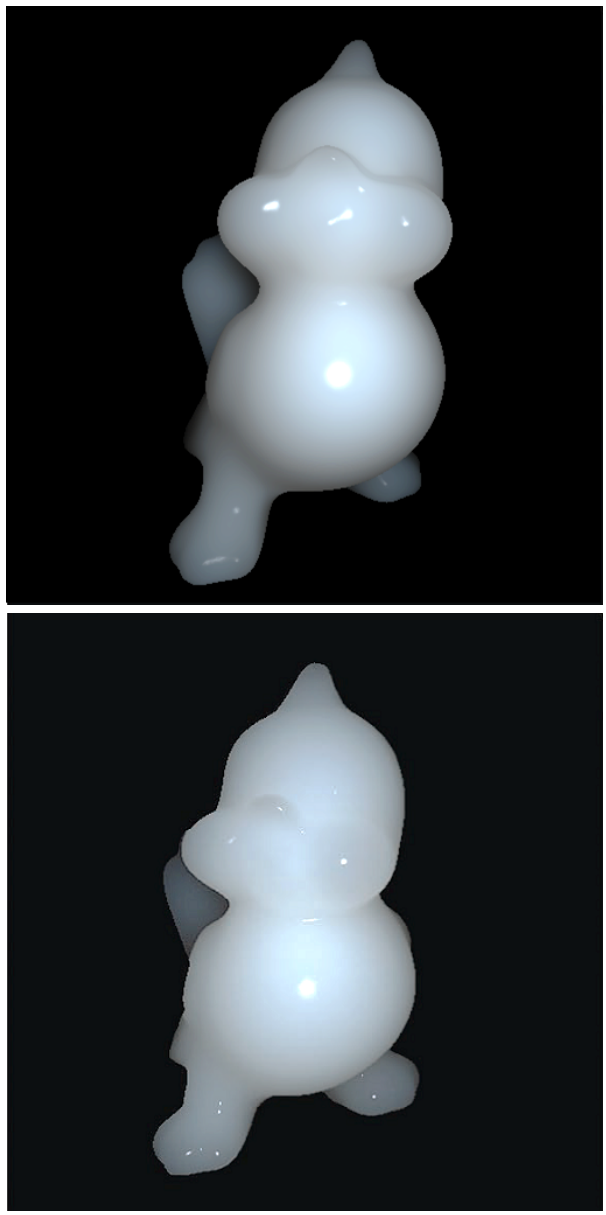

Figure 6.15: The Bird model rendered with the implementation from this thesis (top) and the similar rendering extracted from the original authors [Shah et al., 2009](bottom). Slight differences can be observed, probably caused by different model orientation. The distinctive characteristics of translucent materials are identical in both images.

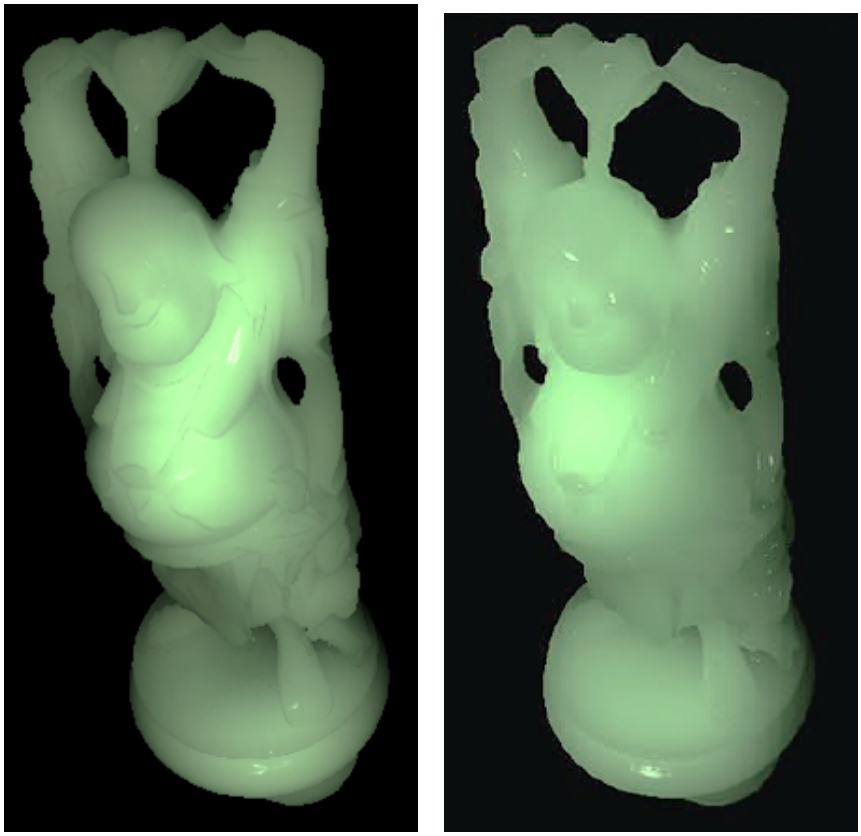

Figure 6.16: The Buddha model rendered with the implementation from this thesis (left) and the similar rendering extracted from the original authors (right). For the left rendering, the material properties have been chosen empirically. The translucency is very similar in both images, as well as the distinctive diffuse look and the distinctive color.

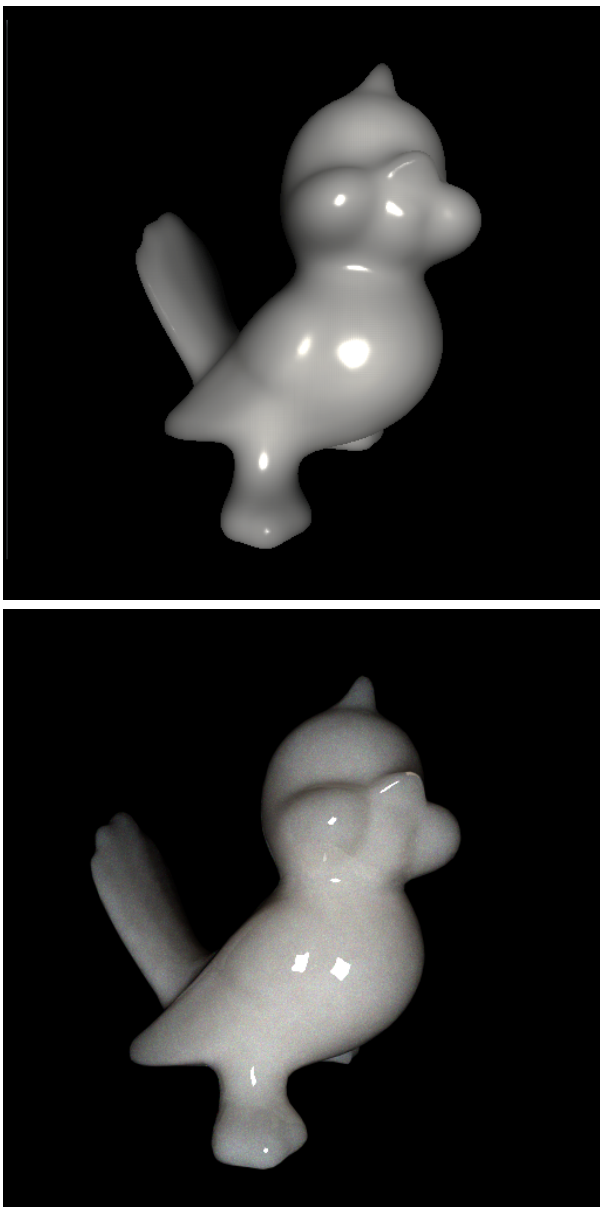

Figure 6.17: The Bird model rendered with the implementation from this thesis (top) and an off-line rendering of the same model. The implementation manages to achieve a similar look and almost identical specular highlights running at interactive frame rates. Note that the bottom image was rendered during approximately 10 hours.

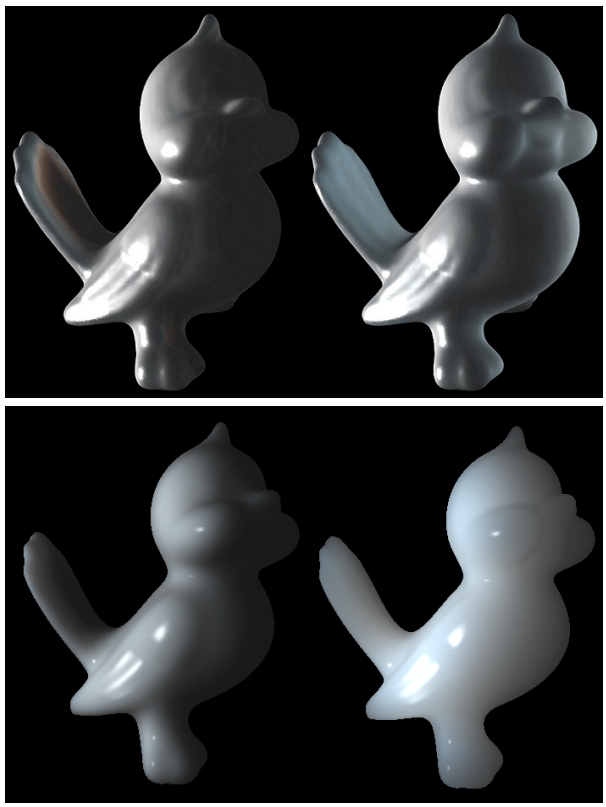

Figure 6.18: The top image is extracted from [Dachsbacher and Stamminger, 2003] and depicts renderings of the same model with varying material properties. The bottom image, rendered with the presented implementation, shows renderings of the same model and focuses as well on varying material properties. A clear improvement in the appearence is noticeable in the results of this thesis.

Chapter 7

Future Work

Chapter 6 presented visually pleasing results of models with different material properties rendered at real-time frame rates. During the implementation, several drawbacks and possible improvements have been encountered. This chapter presents the drawbacks, possible solutions to avoid them and references to related literature that may be used in the implementation of the solutions.

Because the solutions required additional research, study and possibly restructuring the framework built for the results presented so far, the implementation was postponed for future work.

7.1 Improving the solution

Solving Under-sampling Issues

As presented in section 5.1, sampling the texture spaces introduces under-sampling artifacts when light is incident on the surface of the model at grazing angles. The areas where this issue may arise are almost parallel to the light's incidence direction, so if sample points fall inside these areas, they can be identified easily. Figure 7.1 shows a generalized situation in which under-sampling

occurs: a single point light source and two adjacent faces of a cube. In the left image, light is perpendicular on the left face of the cube; the sample points are small white rectangles. In the right image, the light source is slightly deviated from it's original perpendicular direction and illumination is incident on the right face of the cube. The density of the sample points should be the same on the right face of the cube as on the left face.

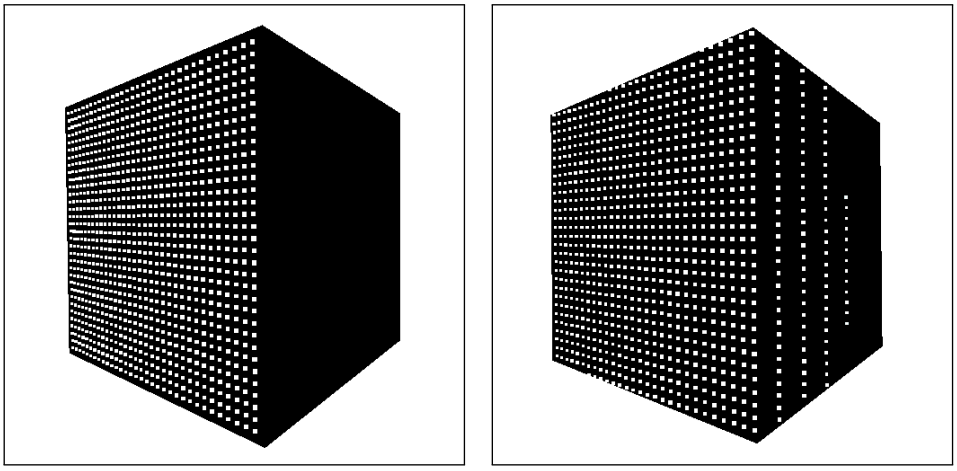

Figure 7.1: Under-sampling at small incidence angles. The left image shows the position of the sample points when light is perfectly perpendicular to the left face of the cube. The right image shows the case when a small deviation is applied to the light direction. Thus, light is incident at a grazing angle, and the surface (the right face) will be under-sampled.

In the third render pass of the point light illumination procedure, in the geometry stage, the application can test if the sample point lies inside an under-sampled area by checking if the cosine of the incidence angle is below a certain threshold value. If so, enough information can be found in the render textures from the first pass to resample the surface in the vicinity of the current sample position. For example, the sample's 3D position and normal can be used to find the equation of the plane in which the 3D sample point lies. The additional samples can be chosen inside this plane, in the vicinity of the initial sample point.

Scattering Artifacts

Because the algorithm runs on the GPU and most of the calculations are considered to be in screen-space (eye-space), there is little information available about the connectivity of the triangle mesh. Figure 7.2 shows a scenario where visual artifacts can appear because a splat centered at sample point x_i will cover a part of the model where it cannot possibly contribute.

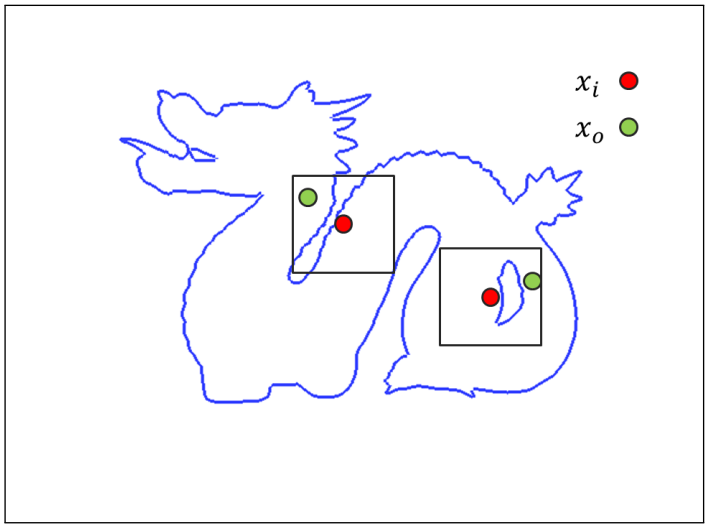

Figure 7.2: Artifacts due to screen-space projection. A splat centered at an incidence point (red dot) can overlap pixels that it can't possibly contribute to.

This artifact can be avoided if the model has associated connectivity information. For example, a mesh segmentation algorithm can be used to split the 3D mesh into patches(see figure 7.3). Then, the x_o and x_i points (see figure 7.2) will belong to different patches and will not be a valid pair. Of course, valid pairs might be rejected because they belong to different patches after the segmentation, though they are very close together and contribution can be possible. To avoid this, the segmentation can be interpolated at the border between two patches (see the right side of figure 7.3), and the validity of a pair of x_i, x_o points can be checked based on how large is the difference between their patch identifiers.

Mesh segmentation techniques include [Shatz et al., 2006] for paper craft models and [Lai et al., 2006, Lai et al., 2008, Yamauchi et al., 2005] for feature sensitive segmentation. The bluring of the segmentation can be performed in a separate

render pass.

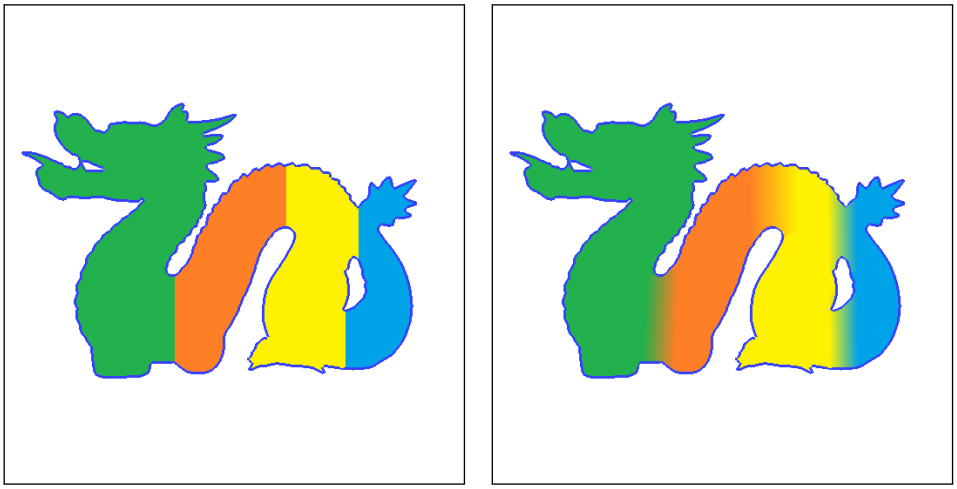

Figure 7.3: Simple connectivity maps. Two simple connectivity maps, listed for visualization. The right image shows blurred borders between patches. This way, valid pairs of incidence and exit points will not be rejected, because the difference between their patch identifiers in the blurred zone will be small.

Rendering Arbitrary Number of Layers

As it was seen in section 5.1.2, the BSSRDF calculated with the multipole approximation is dependent on the thickness of the model. For concave triangle meshes where the correct thickness cannot be calculated as described in section 5.1.2, depth peeling [Bavoil and Myers, 2008, Everitt, 2001] needs to be applied. Using depth peeling for multi-layered objects, the thickness of each layer of coating can be calculated and thus it is possible to render a larger gamma of objects with varying layer thickness.

7.2 Extending the solution

Screen Space Effects

As presented in chapter 5, the first render pass for point light illumination (from light's perspective) is used to obtain the information about the geometry visible to the light source. The render texture used contains enough information to be used with different purposes, for example, as a shadow map.

The output from the second render pass, contains the information for the geometry from the observer's perspective and is useful for a series of real-time screen space techniques as screen space ambient occlusion. For this, after the second pass is finished, for each screen pixel (equivalent to each texel from the render texture), its position (or depth) and normal will be used to roughly estimate the occlusion from the geometry in its vicinity [Bavoil and Sainz, 2008]. [Akenine-Moller et al., 2008] gives a good overview of different methods for simulating ambient occlusion in real-time.

A variety of screen space after effects can be used to alter or enhance the final visual appearence of the rendered scene including edge detection filters, toon shading or blurring. More important in this subset of post effects is global illumination. Several papers can be of reference for screen space global illumination: [Dachsbacher and Stamminger, 2006] uses light texture sampling and splats to simulate first bounce illumination; [Ritschel et al., 2009] propose a general screen space procedure that accounts for ambient occlusion, direct illumination and one-bounce indirect illumination.

Environment Illumination

The environment map filtering method described in this report and used in the implementation in section 5.3.1 of chapter 5 is independent on the material rendered and may de-emphasize certain optical effects. In *Optimizing Environment Maps for Material Depiction* [Bousseau et al., 2011] the authors present a method for optimizing and filtering the environment maps focusing on the optical properties of each individual material in the scene. Using their proposed method, it may be possible to render translucent objects under environment illumination and enhance each object individually. For example, marble objects would be enhanced by a nicer specular appearence while wax objects by a more diffuse look.

Chapter 8

Conclusions

The previous chapters have presented a solution to rendering translucent materials that focuses on performance and generating accurate visual results.

Chapter 2 gave a brief overview of the related literature, showing only a small number of the many attempts that have successfully pushed forward the limits of rendering subsurface light transport.

Chapters 3, 4 and 5, together, offer in detail the theory, the method and the implementation for another solution to the above mentioned topic. The solution combines principles and procedures assimilated by the author during the incipient phase of the thesis (related literature study), then improved and refined during the implementation phase. The desire to reach a complete, global solution has resulted in a more robust approach that differentiates itself from others by using a different sampling strategy, different mathematical approach and by taking more advantage of the programmable pipeline. Nonetheless, different interpretation and framework design certainly bring novelty in this approach.

Chapter 6 validates the implementation by comparing its results with the results from related literature. The performance values obtained fit the requirements for interactive applications (20 frames per second) and real-time applications (30 frames per second) for numerous test scenes. The images rendered capture the correct appearence of the materials they were intended to show and at the same

time match the renderings of previous authors, or are visually more pleasing.

Finally, chapter 7 details a few of the possible improvements that may benefit to this solution. The potential seen so far in this approach makes the list of improvements open and without visible limits in sight.

It is the author's opinion that the goals of this thesis have been reached, based on the results obtained in chapter 6. Together, these chapters may once become the starting point or the partial inspiration for another approach to rendering translucent materials.

Appendix A

Large-Scale Images and Special Renderings

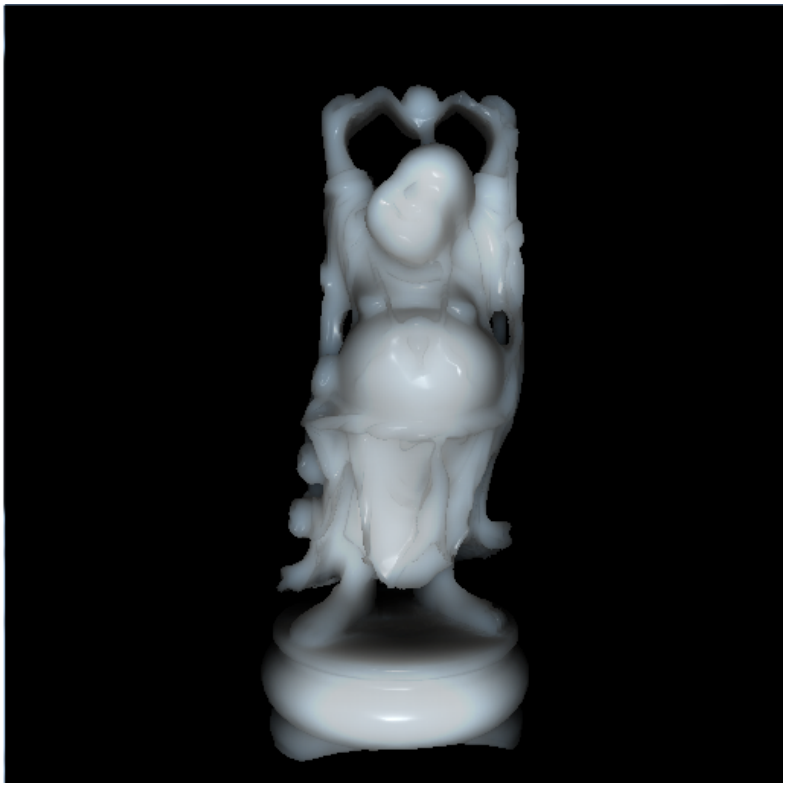

Figure A.1: The Buddha model rendered with marble material properties, using the multipole method. Interactive frame rates have been achieved, shown in the middle column of table 6.5. The image is an enlarged version of the bottom-right image in figure 6.3.

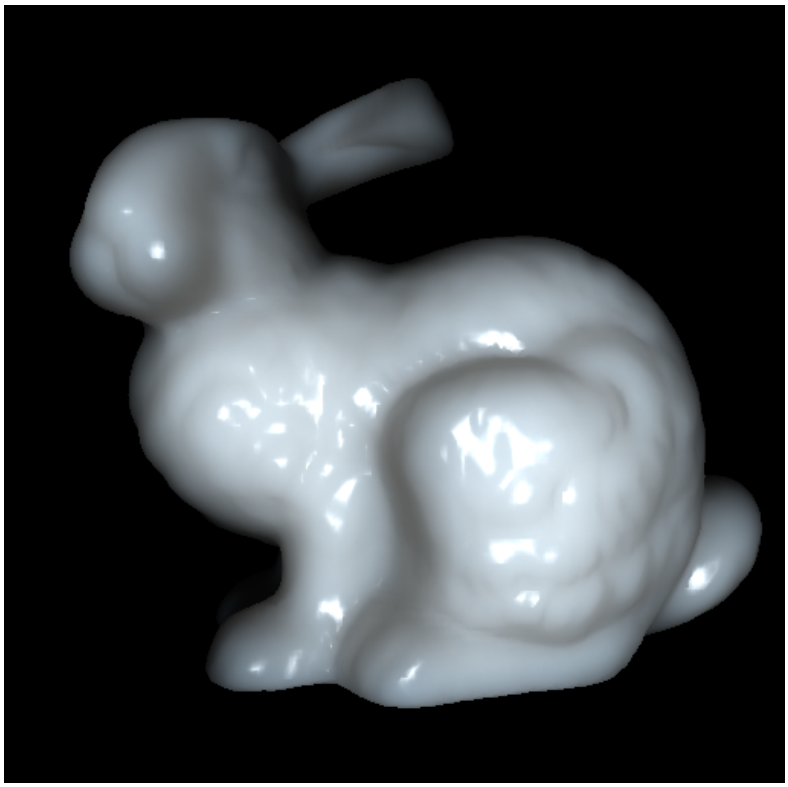

Figure A.2: The Stanford Bunny model rendered with scattering properties corresponding to marble, using the multipole method. The image is an enlarged version of the right image in figure 6.2 and was rendered at interactive frame-rates as presented in the third column of table 6.3.

Figure A.3: The Stanford Dragon model rendered with scattering properties corresponding to marble, using the dipole method. The image is an enlarged version of the right image in figure 6.6, and was rendered in real-time.

Figure A.4: The marble Stanford Dragon model rendered with the multipole method. The image is an enlarged version of the right image in figure 6.7, and was rendered at interactive frame-rates.

Figure A.5: The marble Stanford Dragon Model, rendered with illumination from an environment map at interactive frame rate. The image shows how the bottom of the model is illuminated by the lower part of the environment corresponding to the ground. Light scatters from the brighter nearby areas of the model, hence the diffuse look.

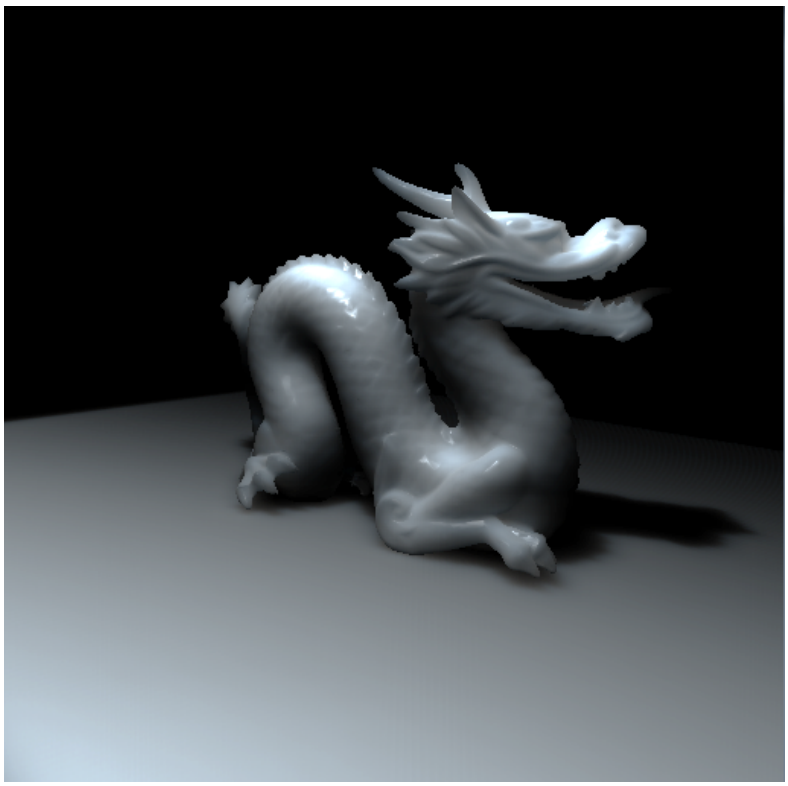

Figure A.6: A scene containing two models: the marble Stanford Dragon and a marble floor. The image was rendered at approximately 15 frames per second and uses a large number of splats to shade the whole floor. This demonstrates that visually pleasing images can be rendered using the method implemented. Note the distinctive look of marble captured in the rendering, the specular highlights and the soft shadow on the ground.

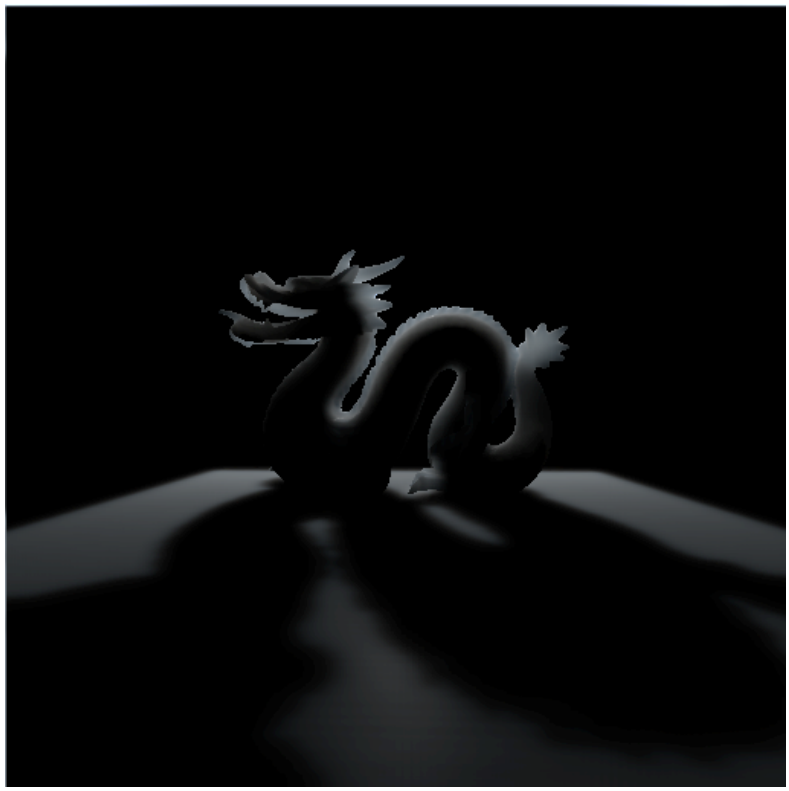

Figure A.7: A scene containing two models: the marble Stanford Dragon and a marble floor. The image was rendered at approximately 21 frames per second. The rendering focuses on the diffuse shadow that follows the shape of the object and on the translucency effect on the Dragon's thin parts.

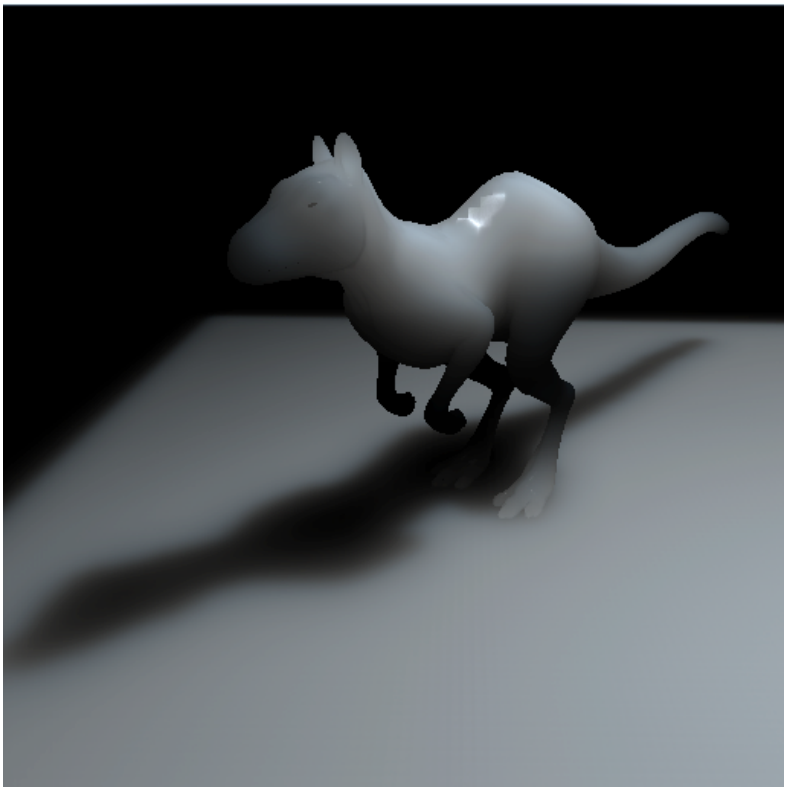

Figure A.8: Rendering of the Killeroo model on a marble floor. Note the soft shadow and the self occlusion on the model.

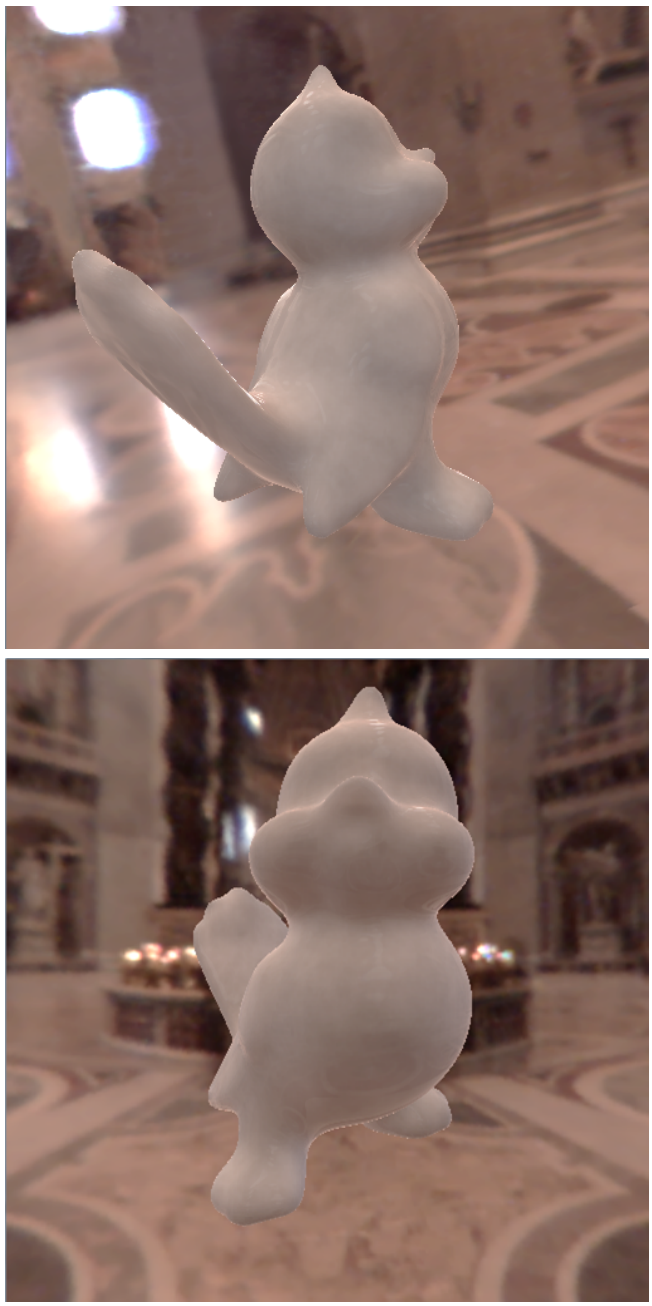

Figure A.9: The Bird model rendered at interactive frame rate under environment illumination from St. Peter's Basilica light probe. Note the diffuse look of the material and slight specular highlights. The color of the model corresponds to the diffuse illumination from the environment.

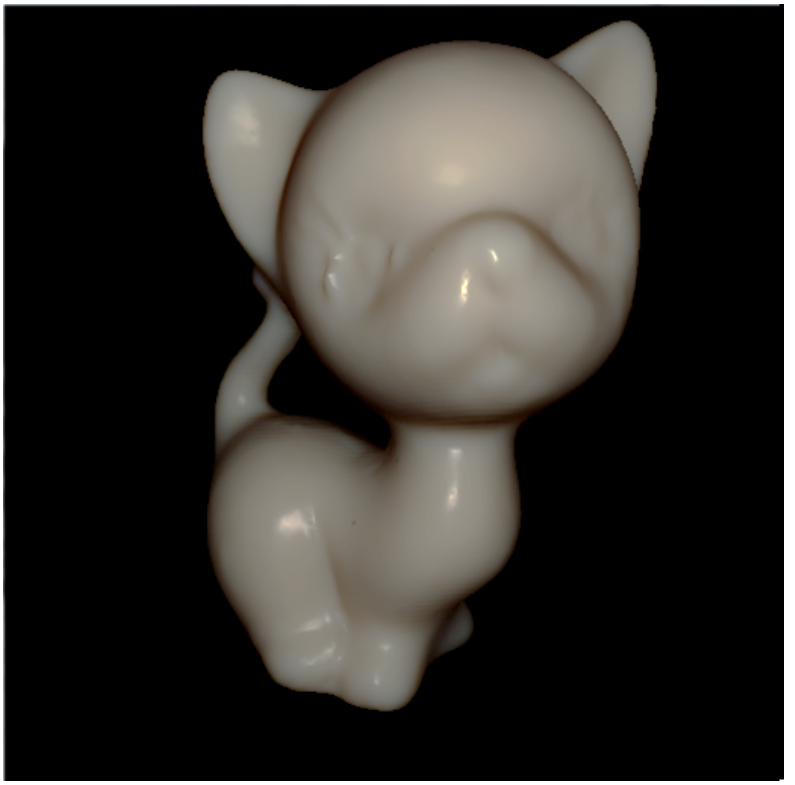

Figure A.10: Cat model rendered with material properties for chocolate milk.

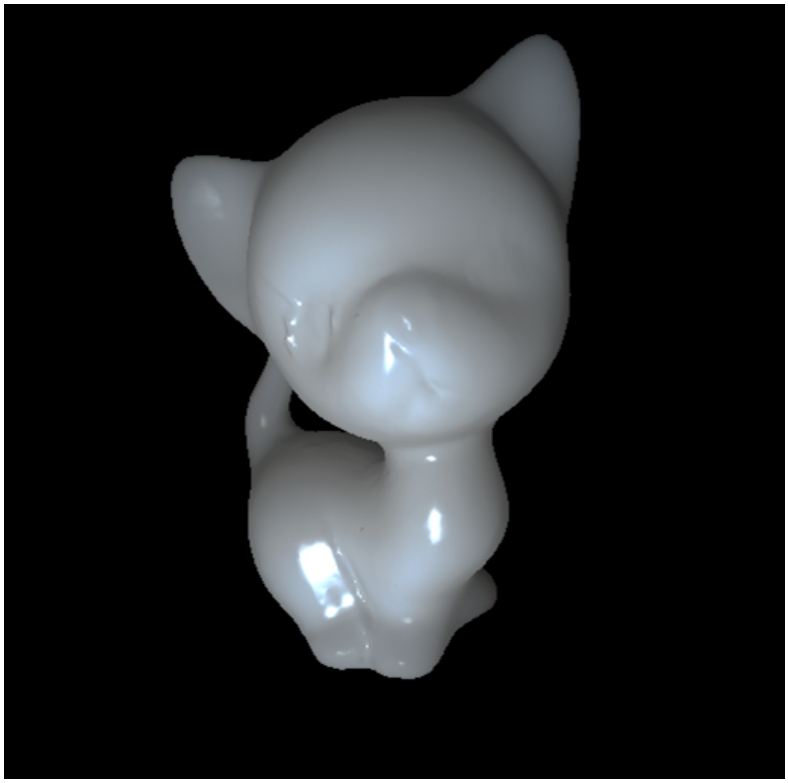

Figure A.11: Cat model rendered with material properties for white jade. The model is illuminated slightly from below, and the scattering effect can be observed on its head and paws.

Bibliography

[Akenine-Moller et al., 2008] Akenine-Moller, T., Haines, E., and Hoffman, N. (2008). *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

[Anton, 1998] Anton, H. (1998). *Calculus: A New Horizon*. Number vb. 1–3. John Wiley & Sons.

[Bavoil and Myers, 2008] Bavoil, L. and Myers, K. (2008). Order independent transparency with dual depth peeling. Technical report, nVidia.

[Bavoil and Sainz, 2008] Bavoil, L. and Sainz, M. (2008). Screen space ambient occlusion. Technical report, nVidia.

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

[Blinn and Newell, 1976] Blinn, J. F. and Newell, M. E. (1976). Texture and reflection in computer generated images. *Communications of the ACM*, 19:542–547.

[Bousseau et al., 2011] Bousseau, A., Chapoulie, E., Ramamoorthi, R., and Agrawala, M. (2011). Optimizing environment maps for material depiction. *Computer Graphics Forum*, Volume 30, Issue 4:1171–1180.

[Chang et al., 2008] Chang, C.-W., Lin, W.-C., Ho, T.-C., Huang, T.-S., and Chuang, J.-H. (2008). Real-time translucent rendering using gpu-based texture space importance sampling. *Eurographics*, 27(2).

[Dachsbacher and Stamminger, 2003] Dachsbacher, C. and Stamminger, M. (2003). Translucent shadow maps. *Eurographics Symposium on Rendering*.

[Dachsbacher and Stamminger, 2006] Dachsbacher, C. and Stamminger, M. (2006). Splatting indirect illumination. *Proceedings of the Symposium on Interactive 3D Graphics*, 2006:93–100.

[Donner and Jensen, 2005] Donner, C. and Jensen, H. W. (2005). Light diffusion in multi-layered translucent materials. *ACM SIGGRAPH Proceedings*, 24 Issue3, July 2005.

[Donner, 2006] Donner, C. S. (2006). *Towards Realistic Image Synthesis of Scattering Materials.* PhD thesis, University of California, San Diego.

[Egan and Hilgeman, 1979] Egan, W. G. and Hilgeman, T. W. (1979). *Optical Properties of Inhomogeneous Materials.* Academic Press.

[Everitt, 2001] Everitt, C. (2001). Interactive order-independent transparency.

[Fubini, 1958] Fubini, G. (1958). Sugli integrali multipli. *Opere scelte*, 2:243–249.

[Georgiev and Butler, 2007] Georgiev, G. T. and Butler, J. J. (2007). Long-term calibration monitoring of spectralon diffusers brdf in the air-ultraviolet. *Appl. Opt.*, 46(32):7892–7899.

[Greene, 1986] Greene, N. (1986). Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29.

[Groenhuis et al., 1983] Groenhuis, R. A. J., Ferwerda, H. A., and Bosch, J. J. T. (1983). Scattering and absorption of turbid materials determined from reflection measurements. 1: Theory. *Appl. Opt.*, 22(16):2456–2462.

[Ishimaru, 1978] Ishimaru, A. (1978). *Wave propagation and scattering in random media.* Academic Press.

[Jensen and Buhler, 2002] Jensen, H. W. and Buhler, J. (2002). A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics*, Volume 21 Issue 3.

[Jensen et al., 2001] Jensen, H. W., Marcschner, S. R., Levoy, M., and Hanrahan, P. (2001). A practical model for subsurface light transport. *SIGGRAPH.*

[Kubelka, 1954] Kubelka, P. (1954). New contributions to the optics of intensely light-scattering materials. part ii: Nonhomogeneous layers. *J. Opt. Soc. Am.*, 44(4):330–334.

[Lai et al., 2008] Lai, Y.-K., min Hu, S., Martin, R. R., and Rosin, P. L. (2008). Fast mesh segmentation using random walks. In *In ACM Symposium on Solid and Physical Modeling.*

[Lai et al., 2006] Lai, Y.-K., Zhou, Q.-Y., Hu, S.-M., and Martin, R. R. (2006). Feature sensitive mesh segmentation. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling*, SPM '06, pages 17–25, New York, NY, USA. ACM.

[Markvart and Castaner, 2003] Markvart, T. and Castaner, L., editors (2003). *Practical handbook of photovoltaics: fundamentals and applications*. Elsevier.

[Miller and Hoffman, 1984] Miller, G. S. and Hoffman, C. R. (1984). Illumination and reflection maps: Simulated objects in simulated and real environments. *SIGGRAPH 84: Advanced Computer Graphics Animation Seminar Notes*.

[Nicodemus et al., 1977] Nicodemus, F. E., Richmond, J., Hsia, J., Ginsberg, I., and Limperis, T. (1977). Geometrical considerations and nomenclature for reflectance. *National Bureau of Standards (U.S.) monograph*.

[Pharr and Humphreys, 2004] Pharr, M. and Humphreys, G. (2004). *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.

[Phong, 1975] Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317.

[Policarpo and Fonseca, 2005] Policarpo, F. and Fonseca, F. (2005). Deferred shading tutorial. Technical report, Pontifical Catholic University of Rio de Janeiro.

[Ramamoorthi and Hanrahan, 2001] Ramamoorthi, R. and Hanrahan, P. (2001). An efficient representation for irradiance environment maps. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

[Rashed, 1990] Rashed, R. (1990). *A Pioneer in Anaclastics: Ibn Sahl on Burning Mirrors and Lenses*.

[Ritschel et al., 2009] Ritschel, T., Grosch, T., and Seidel, H.-P. (2009). Approximating Dynamic Global Illumination in Screen Space. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

[Shah et al., 2009] Shah, M. A., Konttinen, J., and Pattanaik, S. (2009). Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE Computer Graphics and Applications*.

[Shatz et al., 2006] Shatz, I., Tal, A., and Leifman, G. (2006). Paper craft models from meshes. *The Visual Computer*, Volume 22:825–834.

[Thomas and Finney, 1995] Thomas, G. B. and Finney, R. L. (1995). *Calculus and Analytic Geometry*. Addison Wesley.

[Turk, 1991] Turk, G. (1991). Generating textures on arbitrary surfaces using reaction-diffusion. *ACM SIGGRAPH Computer Graphics*, 25 Issue 4, July 1991.

[Yamauchi et al., 2005] Yamauchi, H., Lee, S., Lee, Y., Ohtake, Y., Belyaev, A., and Seidel, H.-P. (2005). Feature sensitive mesh segmentation with mean shift. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, SMI '05, pages 238–245, Washington, DC, USA. IEEE Computer Society.