Photon Beam Methods in Rendering

Simon Kallweit

Bachelor Thesis
August 2013

Supervisor:
Prof. Markus Gross

Advisors:
Dr. Wojciech Jarosz
Dr. Ralf Habel

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney Research

Abstract

This thesis covers rendering methods for participating media, with special focus on recent photon beam methods. We study the theory for rendering with participating media and discuss, among others, four particular rendering methods: Progressive Photon Beams, Virtual Ray Lights, Photon Diffusion and Photon Beam Diffusion. We have reviewed and refactored an existing cross-platform renderer and extended it with consistent implementations of the presented rendering methods. Using our renderer, we are able to compare the rendering methods and discuss their advantages and disadvantages.

Zusammenfassung

Diese Diplomarbeit befasst sich mit der Bildsynthese von voluminösen Materialien mit primärem Fokus auf neuen Methoden basierend auf Photonen Strahlen. Wir beschreiben die grundlegende Theorie zur Bildsynthese von voluminösen Materialen und befassen uns mit vier speziellen Methoden: Progressive Photon Beams, Virtual Ray Lights, Photon Diffusion und Photon Beam Diffusion. Wir haben eine existierende Software überarbeitet und auf dessen Basis einen neuen plattformübergreifenden Renderer implementiert, welcher die vorgestellten Methoden zur Bildsynthese in konsistenter Weise umsetzt. Mit Hilfe der so enstandenen Software vergleichen wir die verschiedenen Methoden und beschreiben ihre Vor- und Nachteile.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Graphics Laboratory ETH Zurich
Prof. Markus Gross

Bachelor's Thesis:

Photon Beam Methods in Rendering

Introduction

The goal of this semester thesis is to implement and compare several methods based on photon mapping and photon beams, namely

- Progressive Photon Beam
- Virtual Ray Lights
- Photon Diffusion
- Photon Beam Diffusion

To achieve this goal, an existing framework needs to be refactored and extended to facilitate the consistent implementation of the mentioned methods.

Tasks

- Orientation, review of related work
- Reviewing code base
- Refactoring code base
- Implementation of Progressive Photon Beams
- Implementation of Photon Diffusion
- Comparison of methods

With these tasks fulfilled, the grade is 4.00

- Implementation of Virtual Ray Lights

With these tasks fulfilled, the grade is 5.00

- Implementation of Photon Beam Diffusion

With these tasks fulfilled, the grade is 6.00

Additional bonus tasks are:

- Implementation of the methods in the research renderer Mitsuba
- Implementation of heterogeneous media

Student

The following student will be working on the project:

Simon Kallweit
Grubenweg 8
3360 Herzogenbuchsee

Dates

Start Date:	18.Feb 2013
End Date:	18.Aug 2013

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The appearance of many natural materials such as liquids, dust, smoke, or even human skin are strongly affected by light interacting with volumetric media, also called *participating media*. Traditional rendering methods, solely based on surface scattering models, cannot capture the effects of participating media, as they only consider the relationship between incoming and outgoing light in relation to a single surface location. In the more general case of objects containing participating media, light is transported inside the medium, separating the location of incoming and outgoing light. Rendering images with participating media, based on the *radiative transport equation* (RTE) [Cha60] to model the transport in participating media, has long been an area of interest in the computer graphics community. The problem is very challenging though, as accurately simulating the interactions between light and participating media is computationally very expensive.

In this thesis, we have thoroughly studied different methods for rendering images with participating media. We mainly focus on recent rendering methods based on *photon beams* [JNSJ11], which can be used to describe the distribution of light in a virtual scene. To generate a set of photon beams, also known as a *photon map*, we can trace beams of light through the scene by scattering in the medium as well as on surfaces. The resulting photon map can then be used to estimate the illumination at arbitrary points in the scene during the rendering process. We have studied three rendering methods based on photon beams:

- *Progressive photon beams* (PPB) [JNT$^+$11], which directly estimates illumination from the photon beams
- *Virtual ray lights* (VRL) [NNDJ12], which considers photon beams as virtual light sources to compute illumination
- *Photon beam diffusion* (PBD) [HCJ13b], which uses diffusion theory to compute illumi-

nation at surfaces

In addition, we have also studied *photon diffusion* (PD) [DJ07], which is a predecessor of the PBD method.

As a part of the thesis, we have implemented the above rendering methods in a small C++ renderer named *PMRender*. We have designed the software such that the implementation of the various rendering methods is cohesive and benefits from their affinity to the photon beams approach.

Using *PMRender*, we are able to compare the rendering methods and discuss their advantages and disadvantages.

The contributions of this thesis are as follows:

- Introduction of the theory for rendering with participating media (Chapter 2)
- Overview of recent rendering methods based on photon beams (Chapter 3)
- Collective implementation of the rendering methods in *PMRender* (Chapter 4)
- Results and comparison of the rendering methods (Chapter 5)

We have tabulated the common notation used throughout the thesis in Table 1.1 .

Symbol	Description
$\Omega_{4\pi}$	Sphere of directions
$\Omega_{2\pi}$	Hemisphere of directions
$\mathcal{A}$	Area
$\mathcal{V}$	Volume
$\mathbf{x}$	Position
$\vec{\mathbf{n}}$	Normalized surface normal
$\vec{\omega}$	Normalized direction (always points away from surface)
$d\vec{\omega}$	Differential solid angle: $d\vec{\omega} = \sin\theta \, d\theta \, d\phi$
$d\vec{\omega}^{\perp}$	Differential projected solid angle
$d\mathcal{A}(\mathbf{x})$	Differential surface area at $\mathbf{x}$
$d\mathcal{A}^{\perp}(\mathbf{x})$	Differential surface area perpendicular to $\vec{\omega}$ at $\mathbf{x}$
$L(\mathbf{x} \leftarrow \vec{\omega})$	Incident radiance at $\mathbf{x}$ from direction $\vec{\omega}$
$L(\mathbf{x} \rightarrow \vec{\omega})$	Outgoing radiance at $\mathbf{x}$ in direction $\vec{\omega}$
L_e	Emitted radiance
L_r	Reflected radiance
L_i	Inscattered radiance
L_m	Medium radiance (accumulated inscattered radiance)
$f_r(\mathbf{x}, \vec{\omega} \leftrightarrow \vec{\omega}')$	BRDF
$S(\mathbf{x}_i, \vec{\omega}_i \leftrightarrow \mathbf{x}_o, \vec{\omega}_o)$	BSSRDF
$p(\mathbf{x}, \vec{\omega} \leftrightarrow \vec{\omega}')$	Phase function
$\sigma_s, \sigma_a, \sigma_t$	Scattering, absorption and extinction coefficients: $\sigma_t = \sigma_s + \sigma_a$
α	Albedo: $\alpha = \sigma_s / \sigma_t$
σ_s'	Reduced scattering coefficient: $\sigma_s' = (1-g)\sigma_s$
σ_t'	Reduced extinction coefficient: $\sigma_t' = \sigma_s' + \sigma_a$
α'	Reduced albedo: $\alpha' = \sigma_s' / \sigma_t'$
$\tau(\mathbf{x} \leftrightarrow \mathbf{x}')$	Optical depth from $\mathbf{x}$ to $\mathbf{x}'$: $\int_{\mathbf{x}}^{\mathbf{x}'} \sigma_t(x) dx$
$T_r(\mathbf{x} \leftrightarrow \mathbf{x}')$	Transmittance from $\mathbf{x}$ to $\mathbf{x}'$: $e^{-\tau(\mathbf{x} \leftrightarrow \mathbf{x}')}$
ξ	Uniformly distributed random number: $\xi \in [0, 1]$

Table 1.1: *Notation used throughout the thesis.*

2

Theory

In this chapter, we will introduce the mathematical background for *rendering*, the process of generating images by means of a computer program. Rendering realistic images requires an accurate simulation of the physical behavior of light. A renderer will typically perform the following three steps:

- Emitting light from various light sources
- Simulating the interactions of light at surfaces and media inside a virtual scene
- Capture light that falls onto a virtual camera sensor, ultimately making up the image *seen* from the camera

In the first section, we will define the physical properties of light, called *radiometry*. Next, we will introduce the *rendering equation*, the fundamental equation we need to solve in order to render realistic images. Then we will introduce the concept of *participating media*, a model to incorporate *volumetric* media such as fog, dust or liquids into the renderings. Next, we derive the *radiative transport equation* (RTE), the fundamental differential equation describing the behavior of light inside participating media, followed by the *volumetric rendering equation*, an extension to the rendering equation incorporating the RTE to account for participating media. In the last section, we will introduce the concept of *subsurface scattering*, a model where we only consider the illumination at the surface, in order to approximate the effects of a participating medium beneath it.

2.1 Radiometry

In this section we will introduce the basic set of mathematical tools to describe the behavior of light, commonly known as *radiometry*. We can then build upon these definitions and basic derivations to derive the theory for the actual rendering algorithms.

Flux

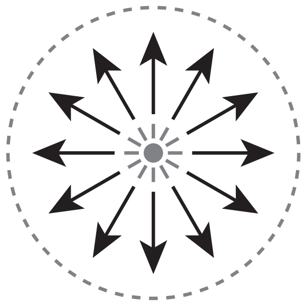


Figure 2.1: *Flux of a light source measured by the surface of an imaginary sphere.*

Radiant flux, or simply *power* is the total amount of energy passing through a surface per unit time. It is usually denoted by the symbol Φ and measured in *watts* $[W = J \cdot s^{-1}]$. We can use *flux* to describe the total amount of light emitted from a light source by placing the source inside an imaginary sphere and measure the light passing through the surface of this sphere (Figure 2.1).

Irradiance and Radiant Exitance

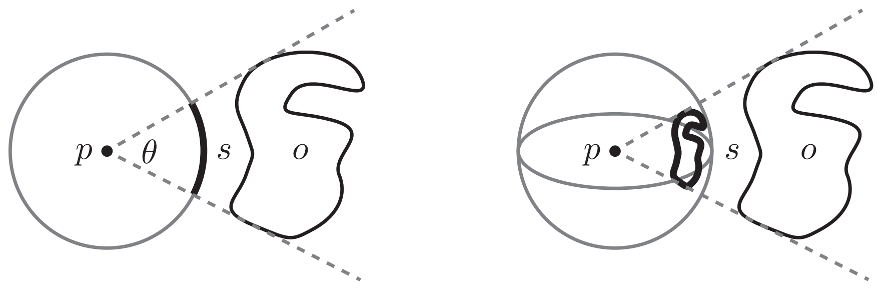
Irradiance describes the area density of flux *arriving* at a surface. It is denoted by the symbol E and is measured in $[W \cdot m^{-2}]$. Irradiance can be expressed in terms of flux as:

$$E(\mathbf{x}) = \frac{d\Phi(\mathbf{x})}{d\mathcal{A}(\mathbf{x})} \tag{2.1}$$

When describing the flux *leaving* a surface we use the term *radiant exitance* denoted by the symbol M.

Solid Angle

Solid angle is the extension of angles in the two-dimensional plane to an angle on a sphere. The *planar angle* is the total angle subtended by an object with respect to some point p. If we project an object o to a unit circle, a part of the circle s will be covered by the projection. This



(a) *Planar angle* ***(b)*** *Solid angle*

Figure 2.2: *Illustration of the concept of planar and solid angle.*

length s corresponds to the subtended angle θ and is measured in *radians* (Figure 2.2a). The entire unit circle subtends an angle of 2π.

The solid angle extends the 2D unit circle into a 3D unit sphere. If we project an object o to the unit sphere, a part of the sphere s will be covered by the projection. This area s corresponds to the subtended solid angle and is measured in *steradians* $[sr]$ (Figure 2.2b). The entire unit sphere subtends a solid angle of 4π, while the hemisphere subtends 2π.

The set of points on the unit sphere centered at a point p can be used to describe all possible directions in 3D space. We will use the symbol $\vec{\omega}$ to denote these directions and assume by convention that $\vec{\omega}$ is of unit length.

Intensity

Intensity describes the directional distribution of light and is denoted by the symbol I and measured in $[W \cdot sr^{-1}]$. Intensity can be expressed in terms of flux as:

$$I(\vec{\omega}) = \frac{d\Phi(\vec{\omega})}{d\vec{\omega}} \tag{2.2}$$

An isotropic point light source with power Φ will have uniform intensity of $I = \frac{\Phi}{4\pi}$.

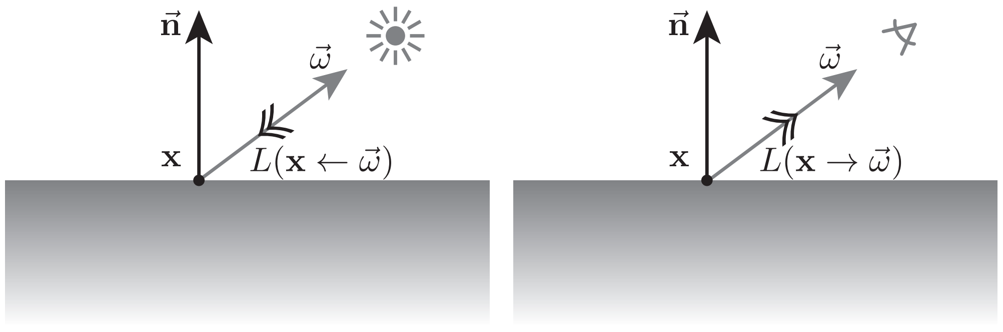
Radiance

The last and most important radiometric quantity is *radiance*, which describes the flux density per unit solid angle, per unit area. Radiance is denoted by the symbol L and measured in $[W \cdot m^{-2} \cdot sr^{-1}]$. Radiance expressed in terms of flux is:

$$L(\mathbf{x}, \vec{\omega}) = \frac{d^2\Phi(\mathbf{x}, \vec{\omega})}{d\vec{\omega}\, d\mathcal{A}(\mathbf{x})^{\perp}} = \frac{d^2\Phi(\mathbf{x}, \vec{\omega})}{d\vec{\omega}\, d\mathcal{A}(\mathbf{x})\, (\vec{\mathbf{n}} \cdot \vec{\omega})} \tag{2.3}$$

where $d\mathcal{A}(\mathbf{x})^{\perp}$ is the projected area of $d\mathcal{A}(\mathbf{x})$ on a hypothetical surface perpendicular to $\vec{\omega}$. In practice, we will usually measure radiance at an actual surface, so we can use $d\mathcal{A}(\mathbf{x})\ (\vec{\mathbf{n}} \cdot \vec{\omega})$ instead, where $\vec{\mathbf{n}}$ is the normalized surface normal. The term $(\vec{\mathbf{n}} \cdot \vec{\omega})$ is also called *foreshortening factor*.

Incident and Exitant Radiance Functions



(a) *Incident radiance* ***(b)*** *Exitant radiance*

Figure 2.3: *Illustration of the difference between incident (arriving) and exitant (leaving) radiance.*

It is useful to differentiate between *incident* and *exitant* radiance, describing light *arriving* at a surface and light *leaving* from a surface. We will furthermore denote incident radiance as $L(\mathbf{x} \leftarrow \vec{\omega})$ (Figure 2.3a) and exitant radiance as $L(\mathbf{x} \rightarrow \vec{\omega})$ (Figure 2.3b), where $\vec{\omega}$ always points *away* from the surface.

Radiometric Relationships

Now that we have defined radiance, it is helpful to define the other radiometric quantities in terms of radiance as well. In order to define flux in terms of radiance, we can take the definition of radiance (2.3) and integrate both sides over the hemisphere $\Omega_{2\pi}$ and the area $\mathcal{A}$:

$$\Phi = \int_{\mathcal{A}} \int_{\Omega_{2\pi}} L(\mathbf{x} \rightarrow \vec{\omega})(\vec{\mathbf{n}} \cdot \vec{\omega})\, d\vec{\omega}\, d\mathcal{A}(\mathbf{x}) \tag{2.4}$$

Using definition (2.1), we can also write irradiance and radiant exitance in terms of radiance:

$$E(\mathbf{x}) = \int_{\Omega_{2\pi}} L(\mathbf{x} \leftarrow \vec{\omega})(\vec{\mathbf{n}} \cdot \vec{\omega})\, d\vec{\omega} \tag{2.5}$$

$$M(\mathbf{x}) = \int_{\Omega_{2\pi}} L(\mathbf{x} \rightarrow \vec{\omega})(\vec{\mathbf{n}} \cdot \vec{\omega})\, d\vec{\omega} \tag{2.6}$$

2.2 Rendering Equation

The rendering equation [Kaj86] relates outgoing radiance at a surface to a sum of emitted and reflected radiance:

$$\underbrace{L(\mathbf{x} \to \vec{\omega})}_{\text{outgoing}} = \underbrace{L_e(\mathbf{x} \to \vec{\omega})}_{\text{emitted}} + \underbrace{L_r(\mathbf{x} \to \vec{\omega})}_{\text{reflected}} \tag{2.7}$$

To obtain the reflected radiance L_r, we first need to introduce the *BRDF*, short for bidirectional reflectance distribution function. The BRDF expresses the relationship between the irradiance and the reflected radiance at a point $\mathbf{x}$ (Figure 2.4):

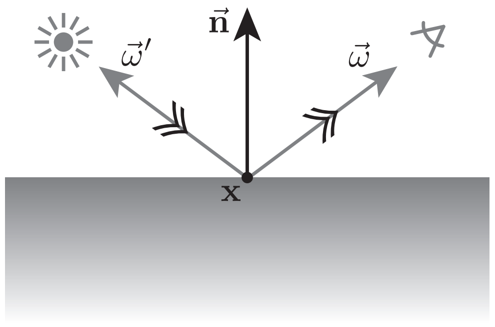


Figure 2.4: *Illustration of the terms involved in the BRDF.*

$$f_r(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{dL(\mathbf{x} \to \vec{\omega})}{dE(\mathbf{x} \leftarrow \vec{\omega}')} = \frac{dL(\mathbf{x} \to \vec{\omega})}{L(\mathbf{x} \leftarrow \vec{\omega}')\,(\vec{\mathbf{n}} \cdot \vec{\omega}')\,d\vec{\omega}'} \tag{2.8}$$

Multiplying both sides of this equation with the denominator and then integrating over the hemisphere we can derive the reflected radiance L_r:

$$L_r(\mathbf{x} \to \vec{\omega}) = \int_{\Omega_{2\pi}} f_r(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega})\, L(\mathbf{x} \leftarrow \vec{\omega}')\,(\vec{\mathbf{n}} \cdot \vec{\omega}')\,d\vec{\omega}' \tag{2.9}$$

We can now rewrite the full rendering equation as:

$$\underbrace{L(\mathbf{x} \to \vec{\omega})}_{\text{outgoing}} = \underbrace{L_e(\mathbf{x} \to \vec{\omega})}_{\text{emitted}} + \underbrace{\int_{\Omega_{2\pi}} f_r(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega})\, L(\mathbf{x} \leftarrow \vec{\omega}')\,(\vec{\mathbf{n}} \cdot \vec{\omega}')\,d\vec{\omega}'}_{\text{reflected}} \tag{2.10}$$

We observe that the radiance L appears on both sides of the equation, meaning that in order to compute the outgoing radiance $L(\mathbf{x} \to \vec{\omega})$ at a point $\mathbf{x}$ in outgoing direction $\vec{\omega}$, we need to compute the incoming radiance $L(\mathbf{x} \leftarrow \vec{\omega}')$ at position $\mathbf{x}$ for all incoming directions $\vec{\omega}'$. The rendering equation is a Fredholm integral equation of the second kind.

Assuming that light travels through a vacuum, radiance will not change at any point $\mathbf{x}$ inside the vacuum, meaning that $L(\mathbf{x} \to \vec{\omega}) = L(\mathbf{x} \leftarrow -\vec{\omega})$ always holds. This means that we can directly relate incoming radiance at position $\mathbf{x}_i$ on surface S_i with outgoing radiance at position $\mathbf{x}_o$ on surface S_o using $L(\mathbf{x}_i \leftarrow \vec{\omega}) = L(\mathbf{x}_o \rightarrow -\vec{\omega})$.

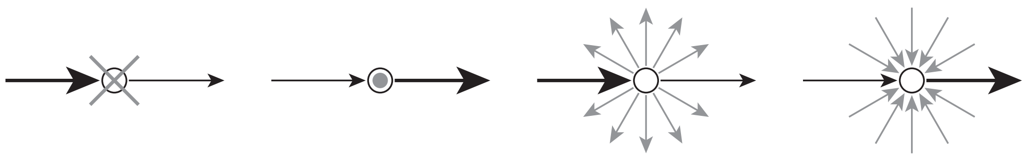
Solving the rendering equation can be done in a number of ways, the most common are *radiosity* [SP94], building upon finite element methods, and *Monte Carlo* based methods such as path tracing. We will not further discuss finite element methods, but introduce volumetric path tracing in Section 3.1.

2.3 Participating Media

While the rendering equation, introduced in the last section assumed a vacuum between surfaces, this is often not the case in reality. The space between surfaces is usually filled with some medium, containing small particles (air, fog, dust, liquids). Light traveling through such a medium is affected by these suspended particles, thus we say that the medium *participates* in the light interactions happening between surfaces.

Because it is infeasible to explicitly compute the interactions between light and individual particles, we assume particles to be very small relative to the viewpoint, and use a probabilistic model to approximate the effects of both sparse and dense participating media.

2.4 Radiative Transport Equation



(a) *Absorption* ***(b)*** *Emission* ***(c)*** *Outscattering* ***(d)*** *Inscattering*

Figure 2.5: *The four interactions modeled by the radiative transfer equation.*

To model the behavior of light in a participating medium, we consider the change in radiance after an infinitesimal step along the light beam. We specifically consider the differential change of radiance $(\vec{\omega} \cdot \nabla)L(\mathbf{x} \rightarrow \vec{\omega})$ at position $\mathbf{x}$ in direction $\vec{\omega}$. The radiance is affected by four possible interactions: absorption, emission, outscattering and inscattering as shown in Figure 2.5.

Absorption

The first interaction models absorption, which occurs when light is converted into other forms of energy (e.g. heat) that are invisible to the human eye and therefore can be omitted from the rendering process. The amount of absorbed light is described by the absorption coefficient σ_a, which allows us to write the change in radiance in differential form as:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = -\sigma_a(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}) \tag{2.11}$$

Emission

In some cases the medium may produce light by itself, for example due to some chemical reaction. The emitted light is described by the *source function* $Q(\mathbf{x} \to \vec{\omega})$ and depends on the position $\mathbf{x}$ and the outgoing direction $\vec{\omega}$. Putting this in differential form we get:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = Q(\mathbf{x} \to \vec{\omega}) \tag{2.12}$$

Outscattering

Some part of the light may be scattered in other directions than $\vec{\omega}$, effectively decreasing the radiance along the ray. The amount of outscattered light is described by the scattering coefficient σ_s, which leads to a similar differential form as seen for absorption:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = -\sigma_s(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}) \tag{2.13}$$

Inscattering

Finally, we must also consider light which is scattered from other directions $\vec{\omega}'$ into direction $\vec{\omega}$, increasing the radiance along the ray. We use the scattering coefficient σ_s to account for the fraction of light that is scattered. To compute the incoming radiance L_i, we integrate over the sphere, multiplying the radiance from direction $\vec{\omega}'$ with the *phase function* $p(\vec{\omega}' \to \vec{\omega})$, which describes the angular distribution of light intensity scattered from incoming direction $\vec{\omega}'$ to outgoing direction $\vec{\omega}$. In differential form we get:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = \sigma_s(\mathbf{x})\, L_i(\mathbf{x} \to \vec{\omega}) = \sigma_s(\mathbf{x}) \int_{\Omega_{4\pi}} p(\vec{\omega}' \to \vec{\omega})\, L(\mathbf{x} \leftarrow \vec{\omega}')\, d\vec{\omega}' \tag{2.14}$$

In an isotropic medium, meaning that light is scattered uniformly in all directions, the phase function is constant. However, most real media are anisotropic, scattering most of the light in one dominant direction. Many applications in computer graphics use the *Henyey-Greenstein* (HG) phase function [HG41] to model both forward and backward scattering media.

To measure the anisotropy of a phase function, we can compute the *mean cosine* g:

$$g = \int_{\Omega_{4\pi}} (\vec{\omega}' \cdot \vec{\omega})\, p(\vec{\omega}' \to \vec{\omega})\, d\vec{\omega}' \tag{2.15}$$

If g is positive, the phase function is mostly forward scattering, if negative, it is mostly backward scattering. If $g = 0$, the phase function is isotropic. By convention, the phase function is assumed to be normalized:

$$\int_{\Omega_{4\pi}} p(\vec{\omega}' \to \vec{\omega})\, d\vec{\omega}' = 1 \tag{2.16}$$

This means that for an isotropic medium, the constant phase function becomes $p = \frac{1}{4\pi}$.

Combining absorption (2.11) and outscattering (2.13) into an extinction term, using the extinction coefficient $\sigma_t = \sigma_a + \sigma_s$, we can now write the *radiative transfer function* (RTE) [Cha60] as:

$$\begin{aligned} (\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) &= -\sigma_t(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}) \\ &+ Q(\mathbf{x} \to \vec{\omega}) \\ &+ \sigma_s(\mathbf{x}) \textstyle\int_{\Omega_{4\pi}} p(\vec{\omega}' \to \vec{\omega})\, L(\mathbf{x} \leftarrow \vec{\omega}')\, d\vec{\omega}' \end{aligned} \tag{2.17}$$

2.5 Volume Rendering Equation

Given the RTE (2.17), introduced in the last section, we can derive an equation describing how light is transported between surfaces. This is essential when rendering images with participating media and has to be evaluated many times to generate an image.

In a simplified setting, we first only consider the effects of extinction (absorption and outscattering) and derive a term to describe what fraction of photons reach a surface from another, called *beam transmittance* or simply *transmittance* T_r. We start with the differential form of extinction:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = -\sigma_t(\mathbf{x}) L(\mathbf{x} \to \vec{\omega}) \tag{2.18}$$

By integrating (2.18) along a ray between $\mathbf{x}$ and $\mathbf{x}'$, we can derive the transmittance T_r:

$$T_r(\mathbf{x} \leftrightarrow \mathbf{x}') = e^{-\tau(\mathbf{x} \leftrightarrow \mathbf{x}')} \tag{2.19}$$

The term τ in the exponent is called *optical thickness* or *optical depth* and is defined as:

$$\tau(\mathbf{x} \leftrightarrow \mathbf{x}') = \int_{\mathbf{x}}^{\mathbf{x}'} \sigma_t(\mathbf{x}) d\mathbf{x} \tag{2.20}$$

In a homogeneous medium where σ_t is constant, the optical depth becomes trivial:

$$\tau(\mathbf{x} \leftrightarrow \mathbf{x}') = \sigma_t \|\mathbf{x}' - \mathbf{x}\| \tag{2.21}$$

Using the definition of transmittance (2.19), we can define the *reduced radiance*, the incoming radiance at position $\mathbf{x}$ due to outgoing radiance at position $\mathbf{x}'$, in a participating medium:

$$L(\mathbf{x} \leftarrow \vec{\omega}) = T_r(\mathbf{x} \leftrightarrow \mathbf{x}')\, L(\mathbf{x}' \rightarrow -\vec{\omega}) \tag{2.22}$$

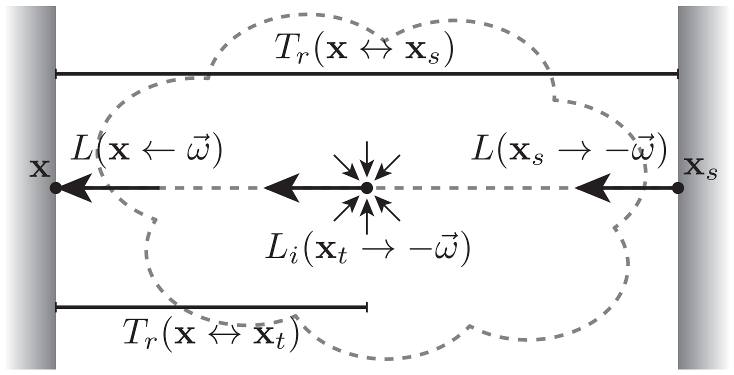

$T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)$
$\mathbf{x}$
$L(\mathbf{x} \leftarrow \vec{\omega})$
$L(\mathbf{x}_s \rightarrow -\vec{\omega})$
$\mathbf{x}_s$
$L_i(\mathbf{x}_t \rightarrow -\vec{\omega})$
$T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)$

Figure 2.6: *Illustration of the terms involved in the volume rendering equation.*

To also consider the effects of emission and inscattering, we can integrate the full RTE (2.17) along a ray. This results in the equation known as the *volume rendering equation*:

$$\begin{aligned} L(\mathbf{x} \leftarrow \vec{\omega}) \;=\; & \underbrace{T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\, L(\mathbf{x}_s \rightarrow -\vec{\omega})}_{\text{reduced surface radiance}} \\ & + \underbrace{\int_{\mathbf{x}}^{\mathbf{x}_s} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\, Q(\mathbf{x}_t \rightarrow -\vec{\omega})\, d\mathbf{x}_t}_{\text{accumulated emitted radiance}} \\ & + \underbrace{\int_{\mathbf{x}}^{\mathbf{x}_s} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\, \sigma_s(\mathbf{x}_t)\, L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\, d\mathbf{x}_t}_{\text{accumulated inscattered radiance}} \end{aligned} \tag{2.23}$$

where the inscattered radiance L_i is defined as:

$$L_i(\mathbf{x} \rightarrow \vec{\omega}) = \int_{\Omega_{4\pi}} p(\vec{\omega}' \rightarrow \vec{\omega})\, L(\mathbf{x} \leftarrow \vec{\omega}')\, d\vec{\omega}' \tag{2.24}$$

In a situation as shown in Figure 2.6, the radiance arriving at $\mathbf{x}$ from direction $\vec{\omega}$ consists of three parts: The first term, *reduced surface radiance*, accounts for the reduced outgoing radiance from the surface at $\mathbf{x}_s$ as described in (2.22). The second term, *accumulated emitted radiance*, accounts for the radiance emitted inside the medium on the ray from $\mathbf{x}_s$ to $\mathbf{x}$. Finally, the last term, *accumulated inscattered radiance*, accounts for all the radiance scattered into direction $-\vec{\omega}$ from within the medium. Computing the accumulated inscattered radiance, furthermore also denoted as L_m (radiance from the medium), is by far the most expensive term to compute, as it needs to be integrated along the ray and over the sphere, and recursively depends on the volume rendering equation.

2.6 Subsurface Scattering

The solution of the RTE (2.17) provides the radiance at arbitrary positions and directions. Solving the RTE however is very difficult and general analytical solutions for the RTE do not exist, allowing it to be only solved by numerical methods such as *finite element methods* or *Monte Carlo integration*.

In a simplified setting, where we only consider the exitant illumination at surfaces, we can apply the concept of subsurface scattering to efficiently render an approximation of the contribution of a participating medium below a surface. Subsurface scattering methods are widely used in production rendering due to their efficiency, which is more important than physically correct rendering.

2.6.1 BSSRDF

To render an image using subsurface scattering, we only consider the exitant radiance at the surface. We can extend the concept of the BRDF into the *bidirectional surface scattering reflection distribution function* (BSSRDF) [NRH+77], which for any two points and directions, relates the incident flux Φ_i at surface position $\mathbf{x}_i$ from direction $\vec{\omega}_i$ to the reflected radiance L_r at surface position $\mathbf{x}_o$ in direction $\vec{\omega}_o$:

$$S(\mathbf{x}_i, \vec{\omega}_i \leftrightarrow \mathbf{x}_o, \vec{\omega}_o) = \frac{dL_r(\mathbf{x}_o \rightarrow \vec{\omega}_o)}{d\Phi_i(\mathbf{x}_i \leftarrow \vec{\omega}_i)} \tag{2.25}$$

S can be decomposed into a summation of three terms, the *reduced radiance* term $S^{(0)}$, accounting for unscattered light, the *single scattering* term $S^{(1)}$, accounting for radiance scattered exactly once and the *multiple scattering* term S_d, accounting for radiance scattered twice and more:

$$S = S^{(0)} + S^{(1)} + S_d \tag{2.26}$$

This separation is useful in order to use specialized algorithms for each term.

To solve for the total reflected radiance L_r, we can integrate (2.25) over both the incident position $\mathbf{x}_i$ and direction $\vec{\omega}_i$ and get:

$$L_r(\mathbf{x}_o \rightarrow \vec{\omega}_o) = \int_{\mathcal{A}} \int_{\Omega_{2\pi}} S(\mathbf{x}_i, \vec{\omega}_i \leftrightarrow \mathbf{x}_o, \vec{\omega}_o)\, L(\mathbf{x}_i \leftarrow \vec{\omega}_i)\, (\vec{\mathbf{n}} \cdot \vec{\omega}_i)\, d\vec{\omega}_i\, d\mathcal{A}(\mathbf{x}_i) \tag{2.27}$$

2.6.2 Searchlight Problem

To evaluate the BSSRDF (2.27) for arbitrary geometry, materials and illumination, we can of course use Monte Carlo particle tracing, which while simple to implement, gives noisy results

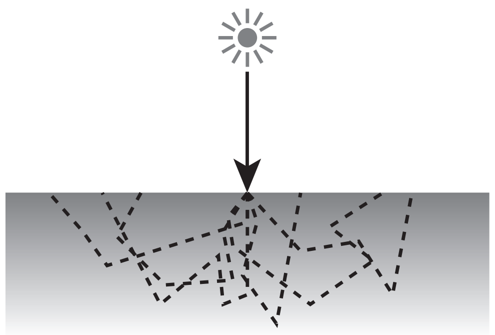


Figure 2.7: *The searchlight problem*

and converges very slowly. For more efficient evaluation, researchers in the fields of medical physics and astrophysics have come up with a simplified setting called the *searchlight problem*.

The searchlight problem, as illustrated in Figure 2.7, consists of a setting where a focused beam of light, called a *pencil beam*, with unit flux is orthogonally incident on a semi-infinite planar slab of a homogeneous medium. Photons enter the medium through a Fresnel boundary and travel downward through the medium until either absorbed or scattered in the medium. Scattered photons may leave the surface directly or get scattered again, ultimately leaving the surface or being absorbed in the medium. The sum of all photons exiting the upper boundary at each point $\mathbf{x}$ forms a spatial reflectance profile. Summing all the photons that have scattered exactly once gives us the diffuse single scattering profile $R^{(1)}(\mathbf{x})$, summing photons that have scattered twice or more gives the diffuse reflectance profile $R_d(\mathbf{x})$, which in case of an orthogonally incident pencil beam will form radially symmetric (1D) profiles, $R^{(1)}(\mathbf{x}) = R^{(1)}(\|\mathbf{x}\|)$ and $R_d(\mathbf{x}) = R_d(\|\mathbf{x}\|)$.

In graphics, most methods approximate S_d, the multiple scattering term of the BSSRDF as [JMLH01, DI11]:

$$S_d(\mathbf{x}_i, \vec{\omega}_i; \mathbf{x}_o, \vec{\omega}_o) = \frac{1}{\pi} F_t(\mathbf{x}_i, \vec{\omega}_i, \eta)\, R_d(\mathbf{x}_o - \mathbf{x}_i)\, \frac{F_t(\mathbf{x}_o, \vec{\omega}_o, \eta)}{4C_\phi(\eta^{-1})} \tag{2.28}$$

where the reflectance profile R_d is centered at the incident position $\mathbf{x}_i$, F_t are transmissive Fresnel terms with η being the relative index of refraction from the outside medium to the inside medium and C_ϕ is a constant needed for normalization, later defined in Section 2.6.4. For efficient evaluation of this expression, most methods rely on diffusion theory to obtain analytic approximations of the reflectance profile R_d.

2.6.3 Classical Diffusion Theory

The classical diffusion approximation only considers the first-order spherical harmonic expansion of the radiance:

$$L(\mathbf{x}, \vec{\omega}) \approx \frac{1}{4\pi}\phi(\mathbf{x}) + \frac{3}{4\pi}\vec{E}(\mathbf{x}) \cdot \vec{\omega} \tag{2.29}$$

where the fluence ϕ and vector flux $\vec{E}$ are the first two angular moments of the radiance distribution:

$$\phi(\mathbf{x}) = \int_{\Omega_{4\pi}} L(\mathbf{x}, \vec{\omega})\, d\vec{\omega} \tag{2.30}$$

$$\vec{E}(\mathbf{x}) = \int_{\Omega_{4\pi}} L(\mathbf{x}, \vec{\omega})\, \vec{\omega}\, d\vec{\omega} \tag{2.31}$$

Assuming an isotropic source function Q, we can substitute the expansion of the radiance (2.29) into the RTE (2.17) and integrate over all directions to obtain:

$$-D\nabla^2\phi(\mathbf{x}) + \sigma_a\phi(\mathbf{x}) = Q(\mathbf{x}) \tag{2.32}$$

where $D = \frac{1}{3\sigma_t'}$ is the diffusion coefficient and σ_t' is the reduced extinction coefficient. We can analytically solve the above equation (2.32), if we choose Q to be an isotropic point source, called a *monopole*, and assuming an infinite homogeneous medium. Assuming a unit power source at the location of the monopole, the solution is the classical diffusion Green's function:

$$\phi^m(\mathbf{x}) = \frac{1}{4\pi D}\frac{e^{-\sigma_{tr}d(\mathbf{x})}}{d(\mathbf{x})} \tag{2.33}$$

where $\sigma_{tr} = \sqrt{\sigma_a/D}$ is the transport coefficient, $d(\mathbf{x})$ is the distance from $\mathbf{x}$ to the monopole and the superscript on ϕ^m indicates the fluence for the monopole.

Source Function

In order to obtain the reflectance profile R_d, we first need to define a suitable source function Q. Theoretically, each beam of light entering the medium through a smooth surface gives rise to a refracted ray within the medium with exponentially decreasing intensity. Assuming unit power at the origin of the refracted beam, we can describe this source function as:

$$Q(t) = \sigma_s' e^{-\sigma_t' t} = \alpha'\sigma_t' e^{-\sigma_t' t} \tag{2.34}$$

where t is the distance from the origin of the beam.

A simpler approach was originally proposed by Farell et al. [FPW92] and later adopted by Jensen et al. [JMLH01], where all the power of the beam is concentrated on a single point at the distance of one *mean free path* $(1/\sigma_t')$ along the refracted beam, turning the source function Q into a delta function. The associated power assuming unit power at the origin of the refracted beam is:

$$\int_0^\infty \sigma_s' e^{-\sigma_t' t} dt = \alpha' \tag{2.35}$$

Boundary Conditions

To apply the diffusion approximation in the semi-infinite slab configuration proposed in the searchlight problem, we need to take into account the boundary condition raised by the surface. This can be accomplished by placing mirrored negative sources above the surface for each source inside the medium, which is generally referred to as the *method of images*. The mirroring plane is placed at an extrapolated boundary above the surface, such that the fluence is zero at a distance $z_b = 2AD$ above the surface. This offset takes into account the mismatch of the refractive index using the reflection parameter A. For classical diffusion, we can use the approximation:

$$A(\eta) \approx \frac{1 + 2C_1(\eta)}{1 - 2C_1(\eta)} \tag{2.36}$$

where η is the relative index of refraction at the surface (from the medium containing the source to the other medium) and C_i is the i^{th} angular moment of the Fresnel function. There exist both analytic solutions to the angular moments [Aro95] as well as approximations [d'E12].

Dipole

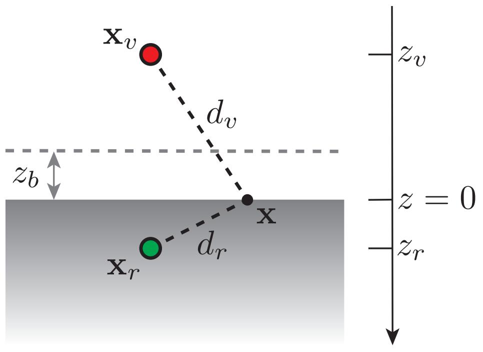


Figure 2.8: *Dipole configuration*

Using the method of images we can now define the fluence imposed by a dipole at distance z_r below the surface with unit power at the origin of the positive source:

$$\phi^d(\mathbf{x}) = \frac{1}{4\pi D}\left(\frac{e^{-\sigma_{tr} d_r}}{d_r} - \frac{e^{-\sigma_{tr} d_v}}{d_v}\right) \tag{2.37}$$

where $d_r = d(\mathbf{x}, \mathbf{x}_r)$ is the distance from $\mathbf{x}$ to the positive pole at $\mathbf{x}_r$ and $d_v = d(\mathbf{x}, \mathbf{x}_v)$ is the distance to the negative pole at $\mathbf{x}_v$. The superscript for ϕ^d indicates the fluence due to the dipole. The described configuration is illustrated in Figure 2.8.

Note that in contrast to most papers on diffusion, we define $z_v = -z_r - 2z_b$ as a position on a Cartesian system with the origin at the surface and positive z direction into the medium, making z_v negative, instead of defining z_v as the distance from the surface to the negative pole. This will lead to more obvious derivations of the reflectance profiles, keeping positive and negative contributions of the poles explicit, as no signs have to be flipped due to the meaning of the z_v value.

Radiant Exitance

In order to define the reflectance profile R_d, we need to derive the radiant exitance on the surface from the fluence of the dipole ϕ^d. We can use Fick's law, which states that the vector flux is the gradient of the fluence:

$$\vec{E}(\mathbf{x}) = -D\vec{\nabla}\phi(\mathbf{x}) \tag{2.38}$$

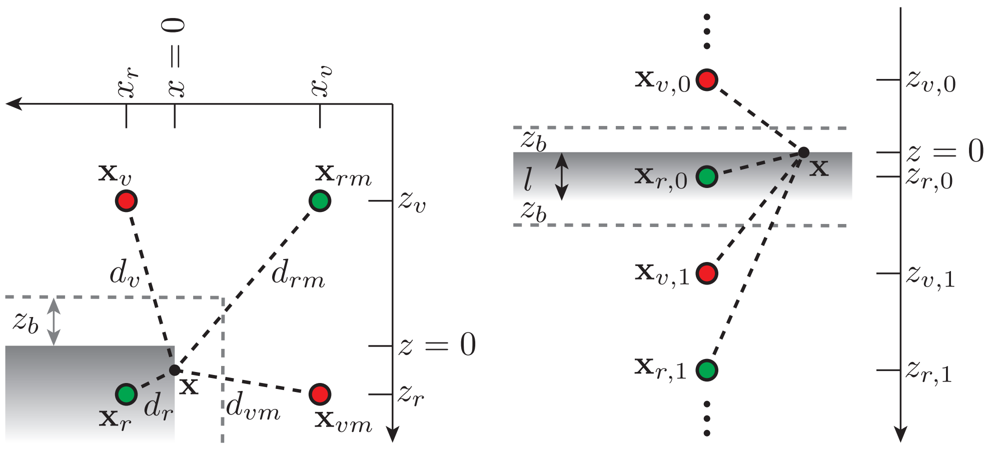
Furthermore, since the radiant exitance on the surface is the dot product of the vector flux with the surface normal, we get:

$$R_d(\mathbf{x}) = \vec{E}(\mathbf{x}) \cdot \vec{\mathbf{n}} = -D(\vec{\nabla} \cdot \vec{\mathbf{n}})\phi(\mathbf{x}) \tag{2.39}$$

So in order to compute the reflectance profile R_d^d due to a dipole, we need to evaluate the directional derivative $(\vec{\nabla} \cdot \vec{\mathbf{n}})$ of the fluence ϕ^d in direction of the surface normal and get:

$$R_d^d(\mathbf{x}) = \frac{1}{4\pi}\left(\frac{z_r(1+\sigma_{tr}d_r)e^{-\sigma_{tr}d_r}}{d_r^3} - \frac{z_v(1+\sigma_{tr}d_v)e^{-\sigma_{tr}d_v}}{d_v^3}\right) \tag{2.40}$$

Quadpole and Multipole



(a) *Quadpole* ***(b)*** *Multipole*

Figure 2.9: *Quadpole and multipole geometries*

Applying the dipole model on arbitrary geometry will lead to considerable errors in the reflectance profiles, as the geometry is essentially *bent* to a semi-infinite slab. We could use an arbitrary number of poles, to satisfy the boundary condition and enforce zero fluence at an offseted geometry with distance $2z_b$ to the actual geometry. This of course is hard to accomplish in practice, but we can extend the dipole model with too additional geometric configurations, the *quadpole* and the *multipole* [DJ07], to derive reflectance profiles which can be applied in a more accurate way to arbitrary geometry.

For the quadpole, we assume a geometry as shown in figure Figure 2.9a, where the main monopole is placed at depth z_r below the top surface, and our shading location $\mathbf{x}$ is placed at the side face at distance x_r. The geometry forms a *corner*, and we take two boundaries into account, one at the top and one at the side face. We can satisfy the second boundary condition by mirroring the dipole around the side boundary at distance $2z_b$. The fluence for the quadpole is defined as:

$$\phi^q(\mathbf{x}) = \frac{1}{4\pi D}\left(\frac{e^{-\sigma_{tr}d_r}}{d_r} - \frac{e^{-\sigma_{tr}d_v}}{d_v} + \frac{e^{-\sigma_{tr}d_{rm}}}{d_{rm}} - \frac{e^{-\sigma_{tr}d_{vm}}}{d_{vm}}\right) \tag{2.41}$$

In the same way as for the dipole we can derive the reflectance profile for the quadpole and get:

$$\begin{aligned} R_d^q(\mathbf{x}) &= \tfrac{1}{4\pi}\tfrac{x_r(1+\sigma_{tr}d_r)e^{-\sigma_{tr}d_r}}{d_r^3} \\ &- \tfrac{1}{4\pi}\tfrac{x_v(1+\sigma_{tr}d_v)e^{-\sigma_{tr}d_v}}{d_v^3} \\ &+ \tfrac{1}{4\pi}\tfrac{x_r(1+\sigma_{tr}d_{rm})e^{-\sigma_{tr}d_{rm}}}{d_{rm}^3} \\ &- \tfrac{1}{4\pi}\tfrac{x_v(1+\sigma_{tr}d_{vm})e^{-\sigma_{tr}d_{vm}}}{d_{vm}^3} \end{aligned} \tag{2.42}$$

For the multipole, we assume a geometry as shown in figure Figure 2.9b, where the main monopole is placed at depth d below the top surface in a slab with thickness l, and our shading location $\mathbf{x}$ is placed at the reflective side at the top. To satisfy the boundary condition, we can mirror the monopole analogously to the dipole. To satisfy the boundary condition at the bottom, we can mirror the two poles of the dipole around a mirror plane at distance z_b below the bottom side. This will inevitably harm the boundary condition at the top, so we can again mirror the two new poles around the mirror plane at the top and continue this process *ad infinitum*. In practice, we can get away by mirroring only a few times, as the error gets neglectable. The depth of the poles are defined as:

$$\begin{aligned} z_{r,i} &= 2i(l+2z_b)+d \\ z_{v,i} &= 2i(l+2z_b)-d-2z_b,\ i=-n,\dots,n \end{aligned} \tag{2.43}$$

and the corresponding fluence induced by the multipole configuration is:

$$\phi^u(\mathbf{x}) = \frac{1}{4\pi D}\sum_{i=-n}^{n}\left(\frac{e^{-\sigma_{tr}d_{r,i}}}{d_{r,i}} - \frac{e^{-\sigma_{tr}d_{v,i}}}{d_{v,i}}\right) \tag{2.44}$$

Applying Fick's law and taking the dot product of the vector flux with the surface normal we get the reflectance profile for the multipole:

$$R_d^u(\mathbf{x}) = \frac{1}{4\pi} \sum_{i=-n}^{n} \left(\frac{z_{r,i}(1+\sigma_{tr} d_{r,i}) e^{-\sigma_{tr} d_{r,i}}}{d_{r,i}^3} - \frac{z_{v,i}(1+\sigma_{tr} d_{v,i}) e^{-\sigma_{tr} d_{v,i}}}{d_{v,i}^3} \right) \tag{2.45}$$

Using the multipole configuration as described above, we evaluate the reflectance profile at the reflective side of the slab. To evaluate at the transmissive side, we can simply replace the depth d of the source with $l - d$, essentially moving the shading location to the transmissive side of the slab.

To apply the dipole, quadpole and multipole configurations to arbitrary geometry, we can use the angle between the surface normal above the main pole and the surface normal at the shading location to blend between the three configurations. This will be explained in more detail in the section on *photon diffusion* (Section 3.6).

2.6.4 Improved Diffusion Theory

Recently, d'Eon et al. introduced several improvements [d'E12] to the classical diffusion introduced in the last section. This includes a modified diffusion equation, improved reflection parameter A as well as more accurate calucation of the exitant radiance.

Improved Diffusion Equation

Grosjean [Gro56, Gro58] introduced a different solution for the approximation of the fluence due to a monopole in an infinite homogeneous medium:

$$\phi^m(\mathbf{x}) = \frac{e^{-\sigma_t' d(\mathbf{x})}}{4\pi d^2(\mathbf{x})} + \frac{\alpha'}{4\pi D} \frac{e^{-\sigma_t' d(\mathbf{x})}}{d(\mathbf{x})} \tag{2.46}$$

where the first term accounts for exact single scattering, and the second accounts for approximate multiple scattering using the improved diffusion coefficient:

$$D = \frac{2\sigma_a + \sigma_s'}{3\sigma_t'^2} = \frac{1}{3\sigma_t'} + \frac{\sigma_a}{3\sigma_t'^2} \tag{2.47}$$

The transport coefficient σ_{tr} is defined using the new diffusion coefficient as $\sigma_{tr} = \sqrt{\sigma_a / D}$.

Removing the first term for single scattering from (2.46), as this can be implemented using various methods we will introduce later on, we can derive the improved diffusion equation, which only differs from the classical diffusion equation by the additional reduced albedo coefficient α', explicitly accounting for multiple scattering:

$$-D\nabla^2\phi(\mathbf{x}) + \sigma_a\phi(\mathbf{x}) = \alpha' Q(\mathbf{x}) \tag{2.48}$$

Boundary Condition

To account for the boundary condition in the semi-infinite slab setting, we can apply the *method of images* in the same way as in the classical diffusion, using the improved reflection parameter:

$$A(\eta) \approx \frac{1 + 3C_2(\eta)}{1 - 2C_1(\eta)} \tag{2.49}$$

Radiant Exitance

In contrast of using Fick's law to derive the radiant exitance as seen in the classical diffusion, d'Eon and Irving use a Robin boundary condition originally introduced by Kienle and Patterson [Aro95, KP97], which uses a linear combination of both fluence and its derivative, the vector flux. This leads to a more accurate radiant exitance, defined as:

$$R_d(\mathbf{x}) = C_\phi \phi(\mathbf{x}) + C_{\vec{E}} \left(-D(\vec{\nabla} \cdot \vec{\mathbf{n}})\phi(\mathbf{x}) \right) \tag{2.50}$$

where $C_\phi = \frac{1}{4}(1 - 2C_1(\eta))$ and $C_{\vec{E}} = \frac{1}{2}(1 - 3C_2(\eta))$. Hence we can write the diffuse reflectance profile of the dipole as a sum of the fluence and the vector flux:

$$R_d^d(\mathbf{x}) = R_d^{d,\phi}(\mathbf{x}) + R_d^{d,\vec{E}}(\mathbf{x}) \tag{2.51}$$

with

$$R_d^{d,\phi}(\mathbf{x}) = C_\phi \frac{\alpha'}{4\pi D} \left(\frac{e^{-\sigma_{tr} d_r}}{d_r} - \frac{e^{-\sigma_{tr} d_v}}{d_v} \right) \tag{2.52}$$

$$R_d^{d,\vec{E}}(\mathbf{x}) = C_{\vec{E}} \frac{\alpha'}{4\pi} \left(\frac{z_r(1 + \sigma_{tr} d_r) e^{-\sigma_{tr} d_r}}{d_r^3} - \frac{z_v(1 + \sigma_{tr} d_v) e^{-\sigma_{tr} d_v}}{d_v^3} \right) \tag{2.53}$$

Note that we can also easily derive the reflectance profiles for the quadpole and multipole configurations, as defined for the classical diffusion theory in (2.42) and (2.45).

Using the improved diffusion theory over classical theory delivers consistently more accurate profiles with neglectable additional computational overhead and should therefore always be preferred in computer graphics applications. A summary on both classical and improved diffusion theory can be found in a recent tech report by Habel et al. [HCJ13a].

3

Rendering Methods

In this chapter, we will introduce different methods for rendering scenes with participating media. As this is a very actively and widely studied research topic, we will primarily focus on recent methods based on so-called *photon beams* [JNSJ11].

For introductory purposes, we will first describe volumetric path tracing, a method using *Monte Carlo integration* to numerically solve the volume rendering equation (2.23). While simple to implement and unbiased by nature, volumetric path tracing is plagued with variance and slow convergence, and may not be feasible in complex scenes and practical applications.

In the next section, we introduce the concept of *photon mapping* [JC98], which improves the efficiency of volumetric path tracing considerably. This is accomplished by tracing a set of photons (discrete packets of light emitted from light sources) through the scene and storing them in a photon map, which is then queried during the rendering stage to get local radiance estimates.

Once we have covered some background, we can then introduce the concept of *photon beams* [JNSJ11], which are an extension to the point based photons used in traditional photon mapping. Photon beams form the framework for the rest of the rendering algorithms presented in this chapter, which we have all implemented in our renderer *PMRender*.

The first two, *progressive photon beams* [JNT+11] and *virtual ray lights* [NNDJ12], use photon beams to efficiently evaluate the volume rendering equation. The former uses progressive radiance estimation on the photon beams and the latter is a so-called many light algorithm, basically treating the photon beams as individual light sources.

The last two methods, *photon diffusion* [DJ07] and *photon beam diffusion* [HCJ13b], are subsurface scattering methods. They are based on diffusion theory to approximate highly scattering homogeneous media.

3.1 Volumetric Path Tracing

Algorithm 3.1 EstimateRadiance($\mathbf{x}, \vec{\omega}$)

```
1:  $t \leftarrow$ Trace($\mathbf{x}, \vec{\omega}$)
2:  $(d, pdf_d, pdf_{ds}) \leftarrow$ SamplePropagation()
3:  if $d < t$ then                                              ▷ Medium Scattering
4:      $\mathbf{x} \leftarrow \mathbf{x} + d\vec{\omega}$
5:      $(\vec{\omega}_i, pdf_i) \leftarrow$ SamplePhaseFunction($\mathbf{x}, \vec{\omega}$)
6:      return $\frac{\sigma_s\, T_r(d)\, p(\vec{\omega}' \rightarrow \vec{\omega})}{pdf_d\, pdf_i} \times$ EstimateRadiance($\mathbf{x}, \vec{\omega}_i$)
7:  else                                                         ▷ Surface Scattering
8:      $\mathbf{x} \leftarrow \mathbf{x} + t\vec{\omega}$
9:      $(\vec{\omega}_i, pdf_i) \leftarrow$ SampleBRDF($\mathbf{x}, \vec{\omega}$)
10:     return $L_e + \frac{T_r(t) f_r(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega})}{pdf_{ds} pdf_i} \times$ EstimateRadiance($\mathbf{x}, \vec{\omega}_i$)
11: end if
```

Volumetric path tracing is a rendering method based on *Monte Carlo integration* to numerically solve the volume rendering equation (2.23). The basic idea is to use random walk sampling, where we recursively trace a ray from the eye through the scene, and randomly sample surface and medium scattering events.

For this simple example, we assume a homogeneous gray medium and only consider the *reduced surface radiance* and *accumulated inscattered radiance* from the volume rendering equation (2.23). Algorithm 3.1 shows the pseudo code for the volumetric path tracer we describe in the rest of this section:

To render an image, we set up a camera ray for each pixel and call the EstimateRadiance function. We typically average over a number of radiance samples to reduce variance.

To compute the radiance, we first determine the distance t to the nearest surface by intersecting a ray with the virtual scene. Next, we need to stochastically choose the distance d to the next scattering event. The probability of light traveling unobstructed in the medium for a distance d is described by the transmittance $T_r(d)$ (2.19). To reduce variance, we can importance sample the distance using a probability density function proportional to the transmittance. For the homogeneous medium we can use the exponential PDF and apply the *inversion method* to sample the propagation distance d:

$$d = -\frac{\log(1-\xi)}{\sigma_t} \quad \text{with} \quad pdf_d = \sigma_t\, e^{-\sigma_t d} \tag{3.1}$$

where $\xi \in [0, 1)$ is a uniform random number, $\sigma_t = \sigma_s + \sigma_a$ is the extinction coefficient and pdf_d is the probability of choosing d. Now we can decide to scatter in the medium ($d < t$) or at the surface ($d \geq t$).

For medium scattering, we first move $\mathbf{x}$ to the medium scattering location. Then we sample the phase function to get an incident direction $\vec{\omega}_i$ and the probability pdf_i. Finally, we compute the returned radiance by recursively calling the EstimateRadiance function and weighting the result with the transmittance and phase function terms divided by the probabilities pdf_d and pdf_i.

Note that for the case of a homogeneous gray medium, the transmittance term will cancel with the probability pdf_d, and for a perfectly importance sampled phase function, the phase function term will cancel with the probability pdf_i:

$$\frac{\sigma_s\, T_r(d)\, p(\vec{\omega}' \to \vec{\omega})}{pdf_d\, pdf_i} = \frac{\sigma_s}{\sigma_t} = \alpha \tag{3.2}$$

This means that we could also apply *Russian roulette* to decide between absorption and scattering, in order to achieve a constant unity weight for the recursive evaluation of ESTIMATERADIANCE.

For surface scattering, we first move $\mathbf{x}$ to the surface scattering location. Then we sample the BRDF to get an incident direction $\vec{\omega}_i$ and the probability pdf_i. Finally, we compute the returned radiance by summing the emitted radiance at the surface L_e with the radiance from recursively calling the ESTIMATERADIANCE function, weighted by the transmittance and BRDF terms divided by the probabilities pdf_{ds} and pdf_i, where pdf_{ds} is the probability of choosing a propagation distance $d \geq t$:

$$pdf_{ds} = \int_{s=t}^{\infty} \sigma_t\, e^{-\sigma_t s} ds = e^{-\sigma_t t} \tag{3.3}$$

Note again that in a homogeneous gray medium the transmittance term cancels with the probability pdf_{ds}. By importance sampling the BRDF, we can achieve cancellation of the BRDF term as well, e.g. by using the *cosine distribution* for a diffuse lambertian BRDF, we can reduce the weight to the diffuse reflectance value (albedo).

The main problem with path tracing algorithms is variance and slow convergence. If for example we assume a scene containing only a small light emitting surface, most random walks generated by the above algorithm will never hit the emitter and therefore will not contribute any radiance to the image. There are many techniques to improve the efficiency of path tracing algorithms, but these are not in the scope of this thesis.

3.2 Photon Mapping

Photon mapping, introduced by Jensen [JC95, Jen96, JC98, Jen01], is an efficient rendering technique to generate images with global illumination. In general, photon mapping generates and caches illumination information in a preprocess step and reuses this information for better efficiency during the actual rendering of the image. Besides being able to capture common global illumination effects such as *color bleeding* and *caustics*, the photon mapping technique also allows to capture the effects of participating media.

Algorithm

Rendering with photon mapping is a two pass process. In a first pass, a number of *photons* are emitted from the light sources and traced through the scene. The resulting set of photons, called a *photon map*, represents a discrete distribution of the flux inside the scene. In the second pass, a Monte Carlo ray tracer is used to generate an image, querying the photon map to compute the local radiance at arbitrary points by means of density estimation.

Photon Map

To generate a photon map (Algorithm 3.2), we start by emitting photons from light sources. The light source is selected using CHOOSELIGHT, giving the light source l to emit from and an associated probability pdf_l. Next, a photon p is emitted by calling EMITPHOTON, which sets the position $\mathbf{x}_p$ and the direction $\vec{\omega}_p$ (importance sampled based on the type of light source) and the power Φ_p. The photon is then traced through the scene and stored at various locations by calling TRACEPHOTON. Once enough photons have been captured, the process terminates and the power of all stored photons is scaled by $1/n_{emitted}$.

Algorithm 3.2 GENERATEPHOTONMAP

```
1: $n_{emitted} \leftarrow 0$
2: **repeat**
3:     $(l, pdf_l) \leftarrow$ CHOOSELIGHT
4:     $(\mathbf{x}_p, \vec{\omega}_p, \Phi_p) \leftarrow$ EMITPHOTON$(l)$
5:     TRACEPHOTON$(\mathbf{x}_p, \vec{\omega}_p, \Phi_p / pdf_l)$
6:     $n_{emitted} \leftarrow n_{emitted} + 1$
7: **until** photon map full
8: scale power of all stored photons by $1/n_{emitted}$
```

Tracing Photons

Tracing a photon through the scene is an iterative process, storing the photon to the photon map at surface and medium scattering positions. The process ends if the photon gets absorbed, or a maximum number of iterations max_{depth} is reached. Simply stopping at the maximum depth, as done in the TRACEPHOTON method, introduces bias. This can be prevented by using *Russian roulette* to decide between continuing and stopping tracing the photon.

At each iteration, we first trace a ray in the direction of the photon, computing the distance t to the next surface, or *infinity*, if there is no surface in the way. Photons travel through the medium for some distance, until they are either absorbed or scattered. Similarly to volumetric path tracing, we can choose the propagation distance to the next absorption or scattering event using the exponential probability density function (3.1):

$$d = -\frac{\log(1-\xi)}{\hat{\sigma_t}} \quad \text{with} \quad pdf_d = \hat{\sigma_t}\, e^{-\hat{\sigma_t} d} \tag{3.4}$$

where $\hat{\sigma_t}$ can be defined arbitrarily, but in order to obtain good sampling, should correspond to the real extinction coefficient σ_t. In a gray medium, we can simply set $\hat{\sigma_t} = \sigma_t$, in a non gray medium we can for example use the average of the components of σ_t.

This type of sampling the propagation distance d assumes that the medium is homogeneous. In a heterogeneous medium, it is more difficult to compute the propagation distance [JC98].

Once the propagation distance d is computed, we can decide if the photon interacts with the medium ($d < t$) or with the surface ($d \geq t$). We then either call SCATTERMEDIUM or SCATTERSURFACE to scatter the photon in the medium or at a surface and finally store the photon in case it did not get absorbed.

Algorithm 3.3 TRACEPHOTON($\mathbf{x}_p, \vec{\omega}_p, \Phi_p$)

```
1: $depth \leftarrow 0$
2: while $depth < max_{depth}$ do
3:     $t \leftarrow$ TRACE($\mathbf{x}_p, \vec{\omega}_p$)
4:     $(d, pdf_d, pdf_{ds}) \leftarrow$ SAMPLEPROPAGATION
5:     if $d < t$ then                                              ▷ Medium Scattering
6:         if not SCATTERMEDIUM($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, d, pdf_d$) then
7:             return
8:         end if
9:         STOREPHOTON($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, depth$)
10:    else                                                          ▷ Surface Scattering
11:        if not SCATTERSURFACE($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, t, pdf_{ds}$) then
12:            return
13:        end if
14:        STOREPHOTON($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, depth$)
15:    end if
16:    $depth \leftarrow depth + 1$
17: end while
```

Surface Scattering

If a photon interacts with a surface (Algorithm 3.4), we use Russian roulette to determine if the photon is absorbed or scattered based on the diffuse reflectance of the BRDF α. In case of scattering, or more precisely transmission or reflection, we can choose the scattered direction by importance sampling the BRDF. We weight the power of the new photon by the transmittance and BRDF terms, divided by the probabilities pdf_{ds} and pdf_i and the albedo α, due to Russian roulette.

Medium Scattering

If a photon interacts with the medium (Algorithm 3.5), we again use Russian roulette to determine if the photon is absorbed or scattered based on the albedo $\alpha = \sigma_s/\sigma_t$. In case of scattering, we choose the scattered direction by importance sampling the phase function. To compute the power of the scattered photon, we take into account the scattering, transmittance and phase function terms, and divide by the probabilities pdf_d and pdf_i and the albedo α, due to using Russian roulette. Note that in case of a gray medium ($\hat{\sigma}_t = \sigma_t$) and perfectly sampled phase

Algorithm 3.4 SCATTERSURFACE($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, t, pdf_{ds}$)

```
1: if $\xi < \alpha$ then                                              ▷ Scattering
2:     $\mathbf{x}_p \leftarrow \mathbf{x}_p + t\vec{\omega}_p$
3:     $(\vec{\omega}_i, pdf_i) \leftarrow$ SAMPEBRDF($\mathbf{x}_p, \vec{\omega}_p$)
4:     $\Phi_p \leftarrow \Phi_p \frac{T_r(t)\, f_r(\mathbf{x}_p, \vec{\omega}_i \leftrightarrow \vec{\omega}_p)}{pdf_{ds}\, pdf_i\, \alpha}$
5:     $\vec{\omega}_p \leftarrow \vec{\omega}_i$
6:     return true
7: else                                                                ▷ Absoprtion
8:     return false
9: end if
```

function, the transmittance and phase function terms cancel with the probabilities, simplifying the scaling factor to $\frac{\sigma_s\, T_r(d)\, p(\mathbf{x}, \vec{\omega}_i \to \vec{\omega}_p)}{pdf_d\, pdf_i\, \alpha} = \frac{\alpha}{\alpha} = 1$, therefore keeping the power of the photons constant.

Algorithm 3.5 SCATTERMEDIUM($\mathbf{x}_p, \vec{\omega}_p, \Phi_p, d, pdf_d$)

```
1: if $\xi < \alpha$ then                                              ▷ Scattering
2:     $\mathbf{x}_p \leftarrow \mathbf{x}_p + d\vec{\omega}_p$
3:     $(\vec{\omega}_i, pdf_i) \leftarrow$ SAMPLEPHASEFUNCTION($\mathbf{x}_p, \vec{\omega}_p$)
4:     $\Phi_p \leftarrow \Phi_p \frac{\sigma_s\, T_r(d)\, p(\mathbf{x}, \vec{\omega}_i \to \vec{\omega}_p)}{pdf_d\, pdf_i\, \alpha}$
5:     $\vec{\omega}_p \leftarrow \vec{\omega}_i$
6:     return true
7: else                                                                ▷ Absoprtion
8:     return false
9: end if
```

Photon Storage

Each time a scattering event occurs, we store a photon $(\mathbf{x}_p, \vec{\omega}_p, \Phi_p)$ in the photon map. Instead of using a list to store the photons, we can also use a spatial acceleration structure to allow for more efficient range lookups during radiance estimation. One popular solution is to use a balanced *kd-tree* [Ben75].

Radiance Estimation

Once the photon map is populated, we can use it to estimate the radiance at surfaces and inside the medium. To compute the reflected radiance at a surface, we can use the definition of the radiance in terms of flux (2.3) and plug it into the equation for reflected radiance (2.9) to get the following term:

$$L_r(\mathbf{x} \to \vec{\omega}) = \int_{\Omega_{2\pi}} f_r(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega}) \frac{d^2\Phi(\mathbf{x} \leftarrow \vec{\omega}')}{d\vec{\omega}'\, d\mathcal{A}(\mathbf{x})} d\vec{\omega}' \tag{3.5}$$

This integral can be approximated using the photon map by computing the *surface radiance estimate* [Jen01]:

$$L_r(\mathbf{x} \to \vec{\omega}) \approx \frac{1}{\mathcal{A}(\mathbf{x})} \sum_{p=1}^{k} f_r(\mathbf{x}, -\vec{\omega}_p \leftrightarrow \vec{\omega}) \Phi_p(\mathbf{x} \leftarrow -\vec{\omega}_p) \tag{3.6}$$

where $\mathcal{A}(\mathbf{x}) = \pi r^2$ and r is the radius of the sphere containing the k nearest photons around $\mathbf{x}$.

Similar to the radiance estimate for surfaces, we can also use the photon map to estimate the radiance of inscattered radiance L_i, which can then be used to compute the radiance from the medium:

$$L_m(\mathbf{x} \leftarrow \vec{\omega}) = \int_{\mathbf{x}}^{\mathbf{x}_s} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\, \sigma_s(\mathbf{x}_t)\, L_i(\mathbf{x}_t \to -\vec{\omega})\, d\mathbf{x}_t \tag{3.7}$$

We can estimate the inscattered radiance using the *volume radiance estimate* [JC98]:

$$L_i(\mathbf{x}_t \leftarrow \vec{\omega}) \approx \frac{1}{\sigma_s(\mathbf{x}_t)\mathcal{V}(\mathbf{x}_t)} \sum_{p=1}^{k} p(\mathbf{x}_t, \vec{\omega}_p \leftrightarrow \vec{\omega}) \Phi_p(\mathbf{x}_t \leftarrow -\vec{\omega}_p) \tag{3.8}$$

where $\mathcal{V}(\mathbf{x}_t) = \frac{4}{3}\pi r^3$ and r is the radius of the sphere containing the k nearest photons around $\mathbf{x}_t$.

Rendering

To render an image using photon mapping, we can use a Monte Carlo ray tracer. A typical implementation will create two seperate photon maps for surface shading, the *global photon map*, containing photons which scattered at least once on a diffuse surface, and the *caustic photon map*, containing photons which scattered at specular surfaces only. This is useful because we need much more accuracy for caustics, as they are visualized directly. When shading surfaces, we can compute the reflected radiance by separating the contributions into different components, each computed individually: direct illumination, specular reflections, caustics and soft indirect illumination. Direct illumination and specular reflections will typically be computed using ray tracing, while caustics and soft indirect illumination can be estimated using the surface radiance estimate introduced in the last section.

To compute the radiance from a participating medium, our main focus, we have to solve the volume rendering equation (2.23). Computing the reduced surface radiance and accumulated emitted radiance terms is trivial, so we will only consider the term for accumulated inscattered radiance (3.7). We can solve this integral by means of a *Riemann sum*, often also referred to as *ray marching* in the context of computer graphics:

$$L_m(\mathbf{x} \leftarrow \vec{\omega}) \approx \sum_{t=0}^{S-1} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\, \sigma_s(\mathbf{x}_t)\, L_i(\mathbf{x}_t \to -\vec{\omega})\, \Delta_t \tag{3.9}$$

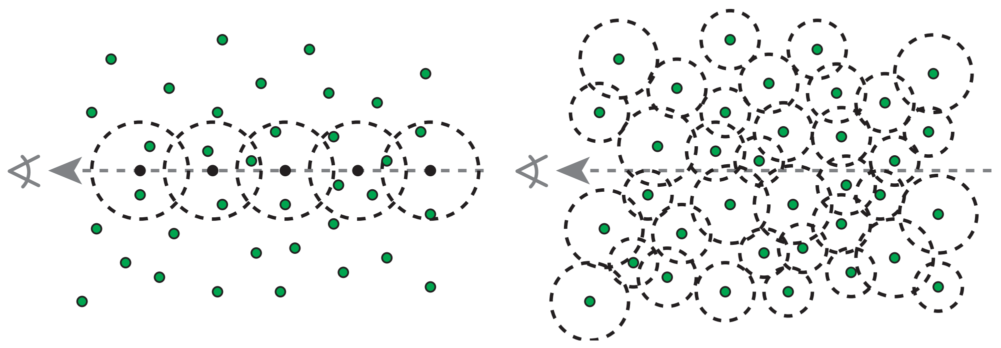

(a) *Ray marching* ***(b)*** *Beam estimate*

Figure 3.1: *Computing the accumulated inscattered radiance using ray marching together with the volume radiance estimate, and the improved beam estimate.*

We split up the ray from $\mathbf{x}$ to $\mathbf{x}_s$ into S segments, each segment with length $\Delta_t = \frac{\|\mathbf{x}_s - \mathbf{x}\|}{S}$, and evaluate the inscattered radiance L_i using the volume radiance estimate (3.8) at points $\mathbf{x}_t = \mathbf{x} + t\Delta_t\vec{\omega}$. A visualization of this is shown in Figure 3.1a.

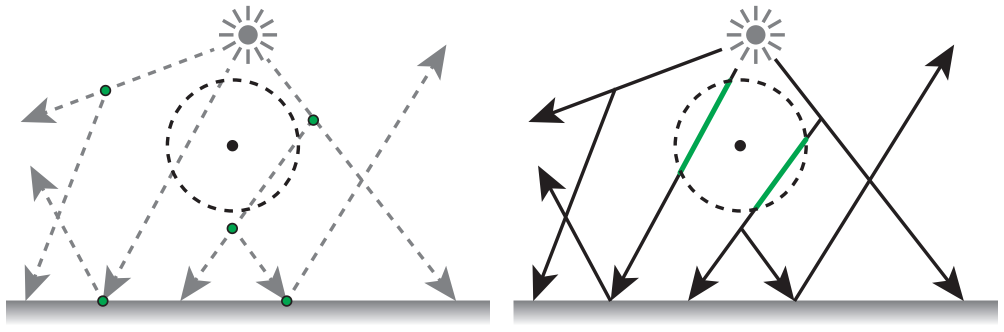
Using ray marching and the volume radiance estimate dramatically improves the efficiency of computing the accumulated inscattered radiance compared to pure path tracing. The improved efficiency does not come for free though, as the photon mapping method introduces bias to the obtained solution in form of blurring, whereas path tracing is unbiased but suffers from variance and slow convergence. Still, this is a tradeoff willingly paid in most situations, as the resulting image shows higher quality at equal render times, despite the errors introduced through bias.

Beam Estimate

The efficency of the ray marching process shown in the last section is primarily bounded by the relativly costly range lookups in the photon map for the radiance estimates. While using too many estimates is bad for performance, and may lead to using the same photons in multiple estimates, using too few estimates may introduce noise and skip many photons from contributing to the inscattered radiance. This observation led to the introduction of the beam estimate [JZJ08], which queries the photon map by means of a *beam*. This is accomplished by assigning variable radii to the individual photons and then intersecting the beam with all the photons in the photon map as shown in Figure 3.1b. The beam estimate method produces significantly higher quality images than conventional photon mapping.

3.3 Photon Beams

Photon mapping, as we have seen in the last section, is a very powerful concept to accelerate rendering methods incorporating effects from global illumination and participating media. By splitting the rendering process into two separate steps, a light emitting step (*photon shooting*) and a light gathering step (*radiance estimation*), we can reuse many computations and greatly improve rendering performance. Representing the *flux field* using only point based photons



(a) *Point based photons* ***(b)*** *Beam based photons*

Figure 3.2: *Radiance estimation using photon points and photon beams. Note that there are no photons inside the query sphere in case of using photon points (left), whereas with photon beams, we have segments of the beams intersecting the query sphere (right).*

at surface and media scattering positions is not optimal though, as we throw away valuable information of the actual photon paths. To render high quality images using point based photon mapping, we either need to compute a high density photon map by shooting a very large number of photons, or we have to use a large search radius for the radiance estimation, which results in high bias. As seen in the last section, we can greatly improve accuracy as well as performance by extending the point based query for the radiance estimate with a beam based query. In the same spirit, we can also extend point based photons into photon beams, to further improve the efficiency and accuracy of radiance estimation.

In their recent paper about volumetric radiance estimation using photon points and photon beams [JNSJ11], Jarosz et al. have introduced a comprehensive theory for radiance estimation, including generalizations of the traditional radiance estimates for point based photons, as well as a set of new radiance estimates using the concept of photon beams.

The main idea of photon beams is to not only store the scattering locations as point based photons in the photon map, but store the full paths of photons in form of photon beams. By storing full photon paths, we get a denser photon map for free, without spending more time creating it. This is visualized in Figure 3.2. Using radiance estimates based on photon beams, we can utilize this denser nature of photon map. The derivation of the photon beam radiance estimates are based on the concept of *photon marching*, where we place photons with decreasing intensity (due to the transmittance term) at regular intervals along the path of the photon. We will not go into the details of the derivation of all the photon beam radiance estimates, but instead only provide the definition of the "Beam $\times$ Beam 1D" estimate, which we use in the progressive photon mapping rendering method Section 3.4.

Photon Beam Shooting

To create a set of photon beams, we still use the same random walk procedure as we use for traditional volumetric photon mapping (Section 3.2), with a few changes:

- We directly store a photon beam when it is emitted from a light source, using the emitted power as the power of the photon beam.

- The length of the photon beam is always defined as the distance to the nearest intersecting surface.
- We use the same approach to sample surface and medium scattering events, but instead of storing a photon, we store a photon beam with its origin at the scattering location and its direction set to the sampled scattering direction and then continue tracing.

We have implemented this algorithm in a module called *photon beam shooter* and use it to create sets of photon beams for the various rendering methods we introduce in the next chapters. The photon beam shooter is configured with the number of beams to create and the requested minimum and maximum media depth required by the respective rendering method. This will be explained in more detail in Section 3.4 and Section 3.6.

Photon Beam Representation

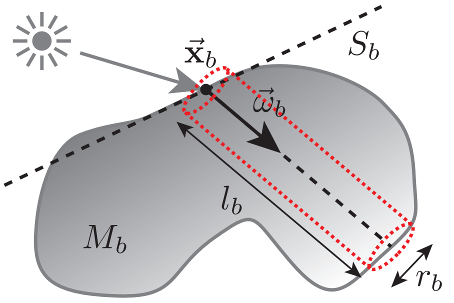


Figure 3.3: *Illustration of the properties associated with a photon beam.*

Each photon beam is described by a set of properties that we use both in our implementation and as a common notation for the rest of the thesis. These properties are visualized in Figure 3.3 and described in Table 3.1.

Beam $\times$ Beam 1D Estimate

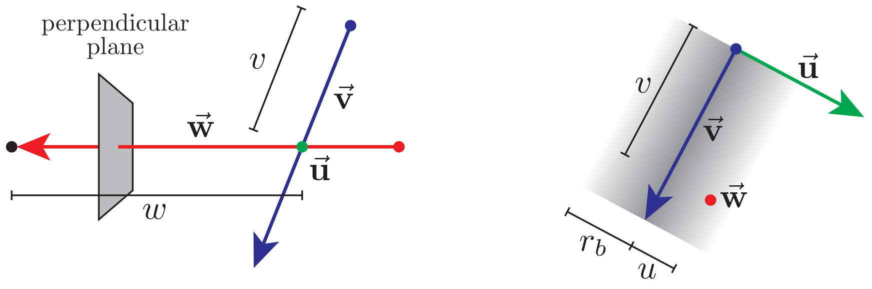


Figure 3.4: *Illustration of the terms involved in "Beam $\times$ Beam 1D" radiance estimate using a single photon beam, viewed from the side, where $\vec{\mathbf{u}}$ extends out of the page (left), and from a plane perpendicular to the query ray, where $\vec{\mathbf{w}}$ extends out of the page (right).*

Symbol	Description
Φ_b	Power of the beam (at origin)
$\mathbf{x}_b$	Position (origin) of the beam
$\vec{\omega}_b$	Direction of the beam
l_b	Length of the beam
r_b	Radius of the beam
S_b	Plane equation of the interface at the beam incident point
M_b	Pointer to the medium the beam is suspended in
m_b	Number of scattering events leading to this beam (0 = not scatter yet)

Table 3.1: *List of properties associated with a photon beam.*

The "Beam $\times$ Beam 1D" radiance estimate directly computes the radiance of a photon beam due to a query ray. In the following, we assume a camera ray at position $\mathbf{x}$ in direction $\vec{\omega}$ and a photon beam with power Φ_b, position $\mathbf{x}_b$, direction $\vec{\omega}_b$ and radius r_b.

To compute the radiance estimate, we create a coordinate system $(\vec{\mathbf{u}}, \vec{\mathbf{v}}, \vec{\mathbf{w}})$, where $\vec{\mathbf{w}} = -\vec{\omega}$, the opposite direction of the camera ray, $\vec{\mathbf{v}} = \vec{\omega}_b$, the direction of the beam, and $\vec{\mathbf{u}} = \vec{\mathbf{v}} \times \vec{\mathbf{w}}$, the direction perpendicular to both of them (Figure 3.4). The radiance estimate is defined as [JNSJ11]:

$$L_m(\mathbf{x} \leftarrow \vec{\omega}) \approx \Phi_b \, k_{r_b}(u) \, \sigma_s(\mathbf{x}_w) \, T_r(w) \, T_r(v) \, \frac{p(\mathbf{x}_w, \vec{\mathbf{v}} \to \vec{\mathbf{w}})}{\sin(\vec{\mathbf{v}} \cdot \vec{\mathbf{w}})} \tag{3.10}$$

where the scalars (u, v, w) are the signed distances to the point $\mathbf{x}_w$ on the photon beam, closest to the query ray, σ_s is the scattering coefficient, $T_r(w)$ and $T_r(v)$ take into account the transmittance along the query ray and along the photon beam and p denotes the phase function. If the query ray is disconnected from the photon beam (both v and w are outside their ray segments), the radiance estimate becomes zero, otherwise the photon beam is blurred using a 1D kernel k_{r_b}, centered on the beam with support width r_b along $\vec{\mathbf{u}}$. In our implementation we use a *Simpson kernel* in the form of:

$$k(x) = (1 - x^2)^2 \quad \text{where} \quad x \in [0, 1] \tag{3.11}$$

3.4 Progressive Photon Beams

One of the main difficulties with photon mapping methods is to deal with its inherent noise and bias. Bias is introduced due to the use of a blurring kernel when computing the radiance estimates and noise results from using too few photons in the photon map. In theory, we can decrease the blurring radius to an infinitesimally small value, and shoot an infinite number of photons to obtain unbiased and noise-free results. In practice, this is of course not feasible, as this would take both an infinite amount of time and storage to compute. Progressive photon mapping (PPM) [HOJ08] alleviates this problem by progressively updating the radiance estimates at measurement points in the scene, gradually converging to an unbiased solution in the limit. While most progressive methods focus on surface illumination, Knaus and Zwicker have also applied the same concept to volumetric photon mapping [KZ11]. But they still used the inferior point based representation of photons, which was later addressed by Jarosz et al. in the work on progressive photon beams (PPB) [JNT+11], combining the theory of PPM with the improved "Beam $\times$ Beam 1D" radiance estimate we introduced in the last section.

Progressive Photon Mapping

The main idea of PPM is to progressively render and accumulate a sequence of passes, generating a final image that converges over time. To analyze the convergence, we can denote the error of the radiance estimation in pass i at some point in the scene by ϵ_i. The average error after N passes can be denoted by $\bar{\epsilon}_N = \frac{1}{N}\sum_{i=1}^{N}\epsilon_i$. As we use a new photon map for each pass, we can treat the errors ϵ_i as samples from independent random variables. We can now define the variance (noise) and expected value (bias) of the average error as:

$$\text{Var}[\bar{\epsilon}_N] = \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}[\epsilon_i] \quad \text{and} \quad \text{E}[\bar{\epsilon}_N] = \frac{1}{N}\sum_{i=1}^{N}\text{E}[\epsilon_i] \tag{3.12}$$

In order to achieve convergence in the limit, we need to make sure that for $N \to \infty$, noise and bias go to zero, i.e. $\text{Var}[\bar{\epsilon}_N] \to 0$ and $\text{E}[\bar{\epsilon}_N] \to 0$.

It is easy to see that if we apply a constant blurring radius to a sequence of passes, we can expect the noise to vanish in the limit (due to the N^2 term in the denominator), whereas the bias will stay constant. The goal of PPM methods is to incrementally reduce the blurring radius such that the overall bias will vanish in the limit. At the same time, we need to make sure that the increased variance in each pass does not inhibit the overall variance from vanishing, thus providing convergence in the limit. We will not go into the details of the convergence analysis, but only provide a brief summary of the results important for our implementation.

Radius Reduction Sequence

Knaus and Zwicker [KZ11] have shown that in order to achieve convergence in PPM, we have to enforce the following ratio of variances between single passes:

$$\frac{\text{Var}[\epsilon_{i+1}]}{\text{Var}[\epsilon_i]} = \frac{i+1}{i+\alpha} \tag{3.13}$$

where $\alpha \in [0, 1]$ is a user specified constant.

Jarosz et al. [JNT+11] have shown that the variance Var$[\epsilon_i]$ from the "Beam $\times$ Beam 1D" estimate is *inversely* proportional to the beam radius scaling factor R_i. Hence we have to enforce the following ratio between scaling factors in order to achieve convergence:

$$\frac{R_{i+1}}{R_i} = \frac{\text{Var}[\epsilon_i]}{\text{Var}[\epsilon_{i+1}]} = \frac{i+\alpha}{i+1} \tag{3.14}$$

Assuming an initial scaling factor at the first pass of $R_1 = 1$, this ratio induces the following sequence of scaling factors:

$$R_i = \left(\prod_{k=1}^{i-1} \frac{k+\alpha}{k}\right)\frac{1}{i} \tag{3.15}$$

In order to provide the user with a single intuitive parameter α to control the convergence, we need to take into account the number of photon beams per pass M, as by increasing M, the radius is scaled more slowly with respect to the total number of photon beams after N passes. A straightforward solution to this is to simply apply M consecutive reductions in each pass, resulting in the following sequence of scaling factors:

$$\frac{R_{i+1}}{R_i} = \sum_{j=1}^{M} \frac{(i-1)M+j+\alpha}{(i-1)M+j+1}, \quad R_i = \left(\prod_{k=1}^{Mi-1} \frac{k+\alpha}{k}\right)\frac{1}{Mi} \tag{3.16}$$

Rendering

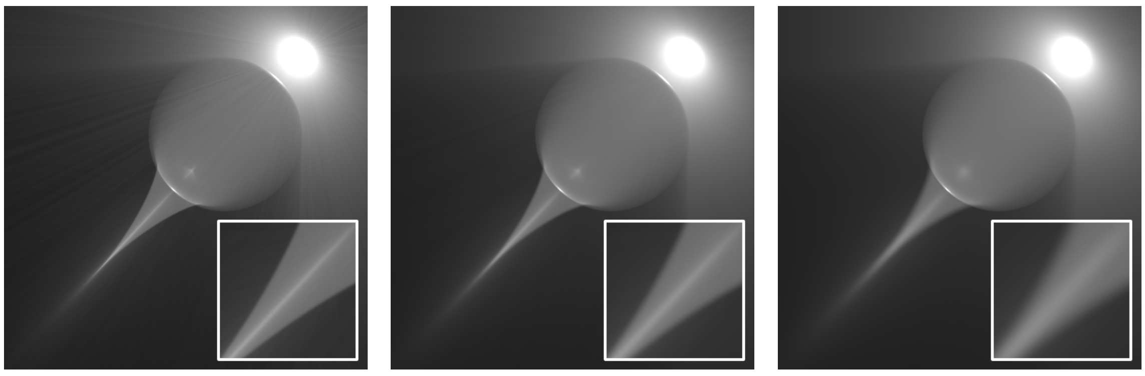

(a) $\alpha = 0.5$ **(b)** $\alpha = 0.7$ **(c)** $\alpha = 0.9$

Figure 3.5: *Volumetric caustics rendered using PPB with 100 passes of 10k beams each. The different settings of the convergence parameter α clearly show the trade-off between convergence and bias.*

Rendering an image with PPB is straightforward. Users have control of the following parameters:

- the number of photon beams to use per pass M

- the initial constant radius of the photon beams R_c
- the convergence parameter α
- the medium depth range $[m_{min}, m_{max}]$

For each pass, we first create a new set of photon beams using the *photon beam shooter*, configured to create photon beams with medium depth $m_{min} \leq m_b \leq m_{max}$. We set the radius of each photon beam to R_cR_i and split them into smaller chunks, so called *sub beams*, and put these into a *bounding volume hierarchy* (BVH) acceleration structure. To evaluate the medium radiance L_m for a given camera ray, we intersect the ray with the BVH and accumulate the radiance for each sub beam $\times$ camera ray intersection using the "Beam $\times$ Beam 1D" estimate with a blurring radius R_cR_i.

The user parameter α represents a trade-off between convergence rate and bias as shown in Figure 3.5. High values of α lead to slow scaling reduction, e.g. faster convergence but more bias, low values of α lead to faster scaling reduction, e.g. less bias but slower convergence.

To render single scattering, we set the medium depth range to $[0, 0]$, creating only photon beams *before* the first scattering event. To render multiple scattering, we set the medium range to $[1, m_{max}]$, creating only photon beams after the first scattering event. To combine single and multiple scattering, we set the medium range to $[0, m_{max}]$, creating photon beams both before and after the first scattering event.

Extensions

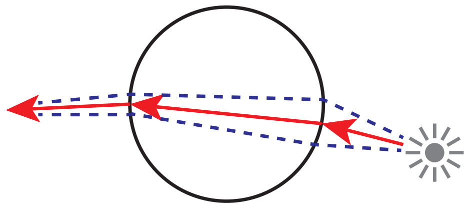


Figure 3.6: *Ray differentials describe spreading (blue) in addition to the origin and direction (red).*

For simplicity, our implementation of PPB resigns from using some of the optimizations proposed in the original paper [JNT+11]. There are two techniques worth mentioning, as they increase the performance of PPB considerably.

First, instead of using a constant radius along the photon beams, we can improve convergence by using a variable radius along the beams, utilizing a method called ray differentials [Ige99]. In contrast to normal rays, ray differentials not only describe the origin and direction of the ray, but also describe the *spreading* (Figure 3.6), allowing photon beams to better fill out the space and reduce overlapping.

Second, the original paper proposes a *splatting* technique, where photons beams are rasterized instead of intersected with query rays. This only works for beams directly visible by the camera, but can improve performance dramatically through leverage of the GPU.

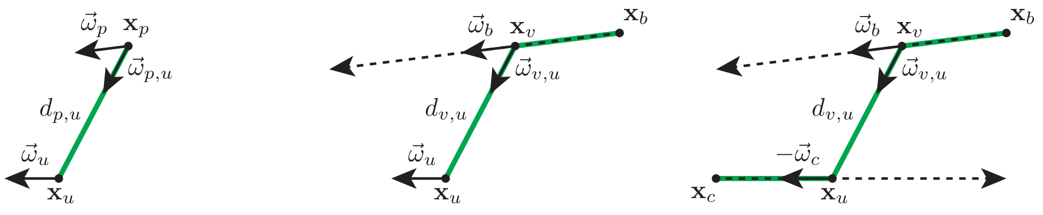
3.5 Virtual Ray Lights

Virtual ray lights (VRL) [NNDJ12] is a so-called many-light algorithm, to simulate indirect illumination from, and within, participating media. The basic idea is to take a set of photon beams, and turn them into *virtual* light sources. These virtual light sources will of course add an additional bounce of light, therefore many-light algorithms are unable to compute single scattering effects. Still, we can use VRLs to compute the more difficult multiple scattering effects on surfaces as well as within the medium.

A comparable approach has been applied to point light sources (VPLs) in various applications: [Kel97] [WFA$^+$05] [WABG06] [RSK08]. The main difficulty with VPLs is their near-source behavior, where they basically collapse into a singularity, introducing high peaks in the computed illumination. A common approach to deal with such singularities is to use clamping or blurring, for example by using virtual spherical lights [HKWB09], which distribute emitted radiance over a sphere instead of emitting all the radiance from a single point. This of course is a practical approach, but at the same time introduces bias, which we typically try to avoid.

Using VRLs over VPLs has many benefits: By using line segments instead of points, we benefit from a denser sampling of the medium radiance and at the same time reduce the amount of singularities, diminishing the need for clamping and blurring. By using an efficient product importance sampling method, we can efficiently compute the total transported radiance from a VRL to a camera ray, even when using anisotropic phase functions. Unlike most other many-light algorithms, VRLs are unbiased and therefor allow for progressive updates trivially by just summing over multiple passes.

Theory



(a) *L_i from a VPL* ***(b)*** *L_i from a VRL* ***(c)*** *L_m from a VRL*

Figure 3.7: *Illustration of the radiance transport using VPLs and VRLs.*

We start by considering the inscattered radiance L_i at position $\mathbf{x}_u$ into direction $\vec{\omega}_u$ due to a single photon (VPL):

$$L_i(\mathbf{x}_u \to \vec{\omega}_u) = \Phi_p \frac{p(\mathbf{x}_p, \vec{\omega}_p \to \vec{\omega}_{p,u})\, p(\mathbf{x}_u, \vec{\omega}_{p,u} \to \vec{\omega}_u)\, T_r(d_{p,u})\, V(\mathbf{x}_p, \mathbf{x}_u)}{d_{p,u}^2} \tag{3.17}$$

where Φ_p, $\mathbf{x}_p$ and $\vec{\omega}_p$ are the power, position and direction of the photon, $\vec{\omega}_{i,j}$ and $d_{i,j}$ are the direction and distance from $\mathbf{x}_i$ to $\mathbf{x}_j$, p is the phase function and $V(\mathbf{x}_i, \mathbf{x}_j)$ is the binary visibility function between points $\mathbf{x}_i$ and $\mathbf{x}_j$. The geometry is illustrated in Figure 3.7a.

To account for the inscattered radiance due to a photon beam, we can integrate the contribution of the VPL (3.17) over the length of the beam while accounting for the reduced intensity along the beam.

$$L_i(\mathbf{x}_u \to \vec{\omega}_u) = \Phi_b \int_0^{l_b} \frac{\sigma_s(\mathbf{x}_v)\, p(\mathbf{x}_p, \vec{\omega}_b \to \vec{\omega}_{v,u})\, p(\mathbf{x}_u, \vec{\omega}_{v,u} \to \vec{\omega}_u)\, T_r(d_{v,u})\, T_r(v)\, V(\mathbf{x}_v, \mathbf{x}_u)}{d_{v,u}^2} dv \tag{3.18}$$

where Φ_b, $\mathbf{x}_b$, $\vec{\omega}_b$ and l_b are the power, origin, direction and length of the photon beam and $\mathbf{x}_v = \mathbf{x}_b + v\,\vec{\omega}_b$ is a position on the beam parametrized by v. The additional factors $\sigma_s(\mathbf{x}_v)$ and $T_r(v)$ take care of the reduced intensity along the beam. The geometry is illustrated in Figure 3.7b.

To compute the reflected radiance L_r at a surface, we can simply replace the phase function at u with the *cosine weighted* BRDF f_r:

$$L_r(\mathbf{x}_u \to \vec{\omega}_u) = \Phi_b \int_0^{l_b} \frac{\sigma_s(\mathbf{x}_v)\, p(\mathbf{x}_p, \vec{\omega}_b \to \vec{\omega}_{v,u})\, f_r(\mathbf{x}_u, \vec{\omega}_{v,u} \leftrightarrow \vec{\omega}_u)\, T_r(d_{v,u})\, T_r(v)\, V(\mathbf{x}_v, \mathbf{x}_u)}{d_{v,u}^2} dv \tag{3.19}$$

We can also derive a function to compute the accumulated inscattered radiance L_m along a camera ray with origin $\mathbf{x}_c$, direction $\vec{\omega}_c$ and length s, by integrating (3.18) along the camera ray, accounting for the additional scattering and transmittance terms, resulting in the following double-integral:

$$\begin{gathered} L_m(\mathbf{x}_c \leftarrow \vec{\omega}_c) = \\ \Phi_b \int_0^s \int_0^{l_b} \frac{\sigma_s(\mathbf{x}_v)\, \sigma_s(\mathbf{x}_u)\, p(\mathbf{x}_p, \vec{\omega}_b \to \vec{\omega}_{v,u})\, p(\mathbf{x}_u, \vec{\omega}_{v,u} \to -\vec{\omega}_c)\, T_r(d_{v,u})\, T_r(v)\, T_r(u)\, V(\mathbf{x}_v, \mathbf{x}_u)}{d_{v,u}^2} dv\, du \end{gathered} \tag{3.20}$$

where $\mathbf{x}_u = \mathbf{x}_c + u\,\vec{\omega}_c$ is now a position on the camera ray parametrized by u. The additional factors $\sigma_s(\mathbf{x}_u)$ and $T_r(u)$ take care of the scattering and transmittance on the camera ray. The geometry is illustrated in Figure 3.7c.

Importance Sampling

To compute radiance from a medium, we have to solve equation (3.20), which defines a 2D integration domain, where one axis is the length u along a camera ray, and the other axis is the length v along a VRL (or photon beam). Unfortunately, there exists no closed-form solution to this double-integral, but we can compute it using a Monte Carlo estimator:

$$L_m(\mathbf{x}_c \leftarrow \vec{\omega}_c) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(u_i, v_i)}{pdf(u_i, v_i)} \tag{3.21}$$

where $g(u_i, v_i)$ is the integrand of (3.20) and $pdf(u_i, v_i)$ is the probability of choosing a point (u_i, v_i) in the domain. In order to reduce variance and compute the estimator efficiently, g should include as many properties of the integrand as possible. Novák et al. propose PDFs for both isotropic and anisotropic scattering functions, but we will concentrate on the former case, as this is what we have implemented in *PMRender*.

Isotropic Scattering

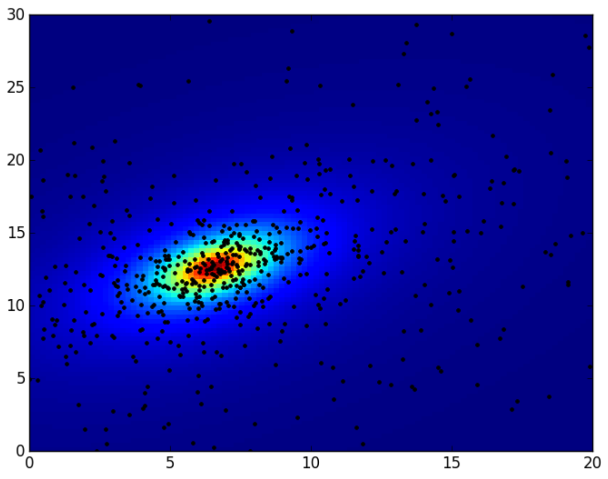


Figure 3.8: *Visualization of the VRL product sampling method. The heat map represents the inverse-squared distance.*

In the isotropic case, we target our PDF to be proportional to the inverse-squared distance term, which makes for the most variation in the integrand. Unfortunately we cannot apply the *inversion method* as it yields no analytical solution.

Instead, we can split up the sampling in a two stage process, first distributing a sample v_i along the VRL using a marginal PDF, and then sample a position u_i along the camera ray using a conditional PDF based on the inverse-squared distance to v_i.

In order to accomplish this, we first apply a change in variables $\hat{u} = u - u_h$ and $\hat{v} = v - v_h$, where u_h and v_h are the parameters of the two closest points along the camera ray and the VRL, with distance h between them. We also define $\hat{u}_0$, $\hat{u}_1$, $\hat{v}_0$, $\hat{v}_1$ as the transformed start and end points on the camera ray and the VRL. Using the law of cosines, we can define the squared distance between two points as $d(\hat{u}, \hat{v}, h, \theta)^2 = h^2 + \hat{u}^2 + \hat{v}^2 - 2\hat{u}\hat{v}\cos\theta$ with $\cos\theta = \vec{\omega}_c \cdot \vec{\omega}_b$

being the dot-product of the camera ray and VRL directions. The marginal PDF can now be expressed as:

$$pdf_v(\hat{v}, \hat{v}_0, \hat{v}_1) = \frac{\int_{\hat{u}_0}^{\hat{u}_1} d(\hat{u}, \hat{v}, h, \theta)^{-2}\, d\hat{u}}{\int_{\hat{v}_0}^{\hat{v}_1} \int_{\hat{u}_0}^{\hat{u}_1} d(\hat{u}, \hat{v}, h, \theta)^{-2}\, d\hat{u}\, d\hat{v}} \tag{3.22}$$

As there is no known closed-form solution to this integral, we can simplify it by assuming the camera ray to be of infinite length, and solve the marginal PDF analytically:

$$pdf_v(\hat{v}, \hat{v}_0, \hat{v}_1) = \frac{\int_{-\infty}^{\infty} d(\hat{u}, \hat{v}, h, \theta)^{-2}\, d\hat{u}}{\int_{\hat{v}_0}^{\hat{v}_1} \int_{-\infty}^{\infty} d(\hat{u}, \hat{v}, h, \theta)^{-2}\, d\hat{u}\, d\hat{v}} = \frac{\frac{\pi}{\sqrt{h^2+\hat{v}^2 \sin^2\theta}}}{\pi \frac{A(\hat{v}_1)-A(\hat{v}_0)}{\sin\theta}} \tag{3.23}$$

where $A(x) = \sinh^{-1}\left(\frac{x}{h}\sin\theta\right)$.

To obtain the marginal CDF, we have to integrate equation (3.23) and get:

$$cdf_v(\hat{v}, \hat{v}_0, \hat{v}_1) = \frac{A(\hat{v}_0) - A(\hat{v})}{A(\hat{v}_0) - A(\hat{v}_1)} \tag{3.24}$$

We can now solve for the inverse CDF and get:

$$cdf_v^{-1}(\xi, \hat{v}_0, \hat{v}_1) = \frac{h \sinh\left(lerp(A(\hat{v}_0), A(\hat{v}_1), \xi)\right)}{\sin\theta} \tag{3.25}$$

Using the inverse CDF, we can now generate a position v_i on the VRL by passing in a random number $\xi_{i,1} \in [0,1)$. In a second step, we use another random number $\xi_{i,2} \in [0,1)$ and equiangular sampling [KF12], to generate a position u_i on the camera ray. The PDF and inverse CDF for equiangular sampling are defined as:

$$pdf_u(\hat{u}, \hat{u}_0, \hat{u}_1) = \frac{h}{(B(\hat{u}_1) - B(\hat{u}_0))(h^2 + \hat{u}^2)} \tag{3.26}$$

$$cdf_u^{-1}(\xi, \hat{u}_0, \hat{u}_1) = h \tan\left(lerp(B(\hat{u}_0), B(\hat{u}_1), \xi)\right) \tag{3.27}$$

where $B(x) = \tan^{-1}\left(\frac{x}{h}\right)$.

The final PDF is then simply defined as the product of the PDFs from the two sampling steps. We have visualized the obtained sampling in Figure 3.8.

Rendering

Rendering with VRLs is straight forward and we can trivially use it in a progressive fashion, as the method is unbiased and multiple passes can simply be accumulated. For each pass, we first shoot a set of photon beams to be used as VRLs. To compute the radiance from the medium L_m, we iterate over all photon beams and evaluate their contribution to the camera ray using the importance sampling framework described in the previous section. To evaluate the binary visibility checks V in the integrand of equation (3.20), we use visibility rays and test them against the scene. As an optimization, visibility rays can be omitted when rendering participating media enclosed by convex or nearly convex objects, as their contribution become negligible. As VRLs are only able to compute multiple scattering, we can use PPB to compute single scattering.

3.6 Photon Diffusion

The classical diffusion theory, as we have introduced in Section 2.6.3, has been applied to computer graphics in various different ways. To mention a few of these applications: Stam used *finite element methods* to compute light transport in multi-layer materials [Sta95]. Jensen et al. used *Monte Carlo* sampling to capture direct illumination and applied diffusion theory to compute subsurface scattering effects [JMLH01]. In later work, Jensen et al. sampled the full irradiance at a fixed set of points on the surface, which allowed them to partly capture global illumination effects and improve rendering efficiency considerably by applying hierarchical evaluation by means of an octree [JB02].

The main contributions of the paper on photon diffusion [DJ07], which we have implemented in our renderer, was the application of diffusion theory to oblique illumination. While most of the previous applications solely relied on the dipole configuration, and assumed incident light to travel into the medium on a beam perpendicular to the surface at the incident location, Donner et al. have applied diffusion theory to beams that travel in an oblique direction to the surface, thus allowing for a more accurate response of light traveling in the medium.

Oblique Illumination

To account for oblique illumination, we have to consider the beam source function as defined in (2.34). We can compute the reflectance profile $R_d^b(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b)$ due to a refracted beam with origin $\mathbf{x}_b$, direction $\vec{\omega}_b$ and unit power after the Fresnel boundary by integrating the reflectance profile $R_d(\mathbf{x}, \mathbf{x}_p)$ due to a source at position $\mathbf{x}_p$ along the beam, while accounting for the reduced intensity due to the source function $Q(t)$:

$$R_d^b(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) = \int_0^\infty Q(t)\; R_d(\mathbf{x}, \mathbf{x}_b + t\vec{\omega}_b) dt \quad \text{with} \quad Q(t) = \sigma_s'\, e^{-\sigma_t' t} \tag{3.28}$$

This integral has no closed form solution, so instead of trying to solve it numerically, Donner et al. proposed to use photon tracing to create a set of photons which is then treated as a set of sources for the dipole, quadpole or multipole profiles.

Symbol	Description
Φ_p	Photon power
t_p	Sampling distance along the beam
$\mathbf{x}_p$	Photon position
$\mathbf{x}_p^p$	Photon position projected to the plane at the beam incident point
S_p	Plane equation at the beam incident point
d_p	Depth of photon in relation to the plane at the beam incident point
l_p	Thickness of the medium approximated by beam length times cosine of refracted angle
M_p	Pointer to the medium the photon is suspended in

Table 3.2: *Properties associated with a diffusion photon.*

Source Distribution

The original paper [DJ07] proposes to use standard Monte Carlo photon tracing for this step, whereas our implementation uses the photon beam shooter and then samples a set of individual sources from the set of beams. Some care needs to be taken when tracing the beams though, as we are only interested in the first scatter locations. This means that we only store the beams when they enter the medium. If a beam scatters inside the medium, or reflects at an interface, we do not store it, as internal reflection as well as multiple scattering are handled by the diffusion approximation. If a beam refracts out of the medium and then re-enters, we treat it the same as beams entered for the first time. Once a set of beams has been generated for each medium to be rendered with photon diffusion, we use exponential sampling to generate a list of sources to be used with photon diffusion. In our implementation we call these sources *diffusion photons* and associate a set of values as described in Table 3.2 with each photon. Algorithm 3.6 shows the details for sampling a photon beam to create a diffusion photon and computing the associated values.

Single Scattering

In the original paper, the single scattering term is proposed to be computed by means of ray marching and density estimation of the photon map. We have not explicitly implemented the single scattering term in our implementation of photon diffusion, as progressive photon beams already produce much better results than the proposed traditional approach.

Multiple Scattering

To evaluate the reflected radiance L_r at the shading location $\mathbf{x}$ in direction $\vec{\omega}$ due to multiple scattering, we can integrate the definition of the BSSRDF (2.25) and use the approximation for

Algorithm 3.6 SAMPLEFROMBEAM($\Phi_b, \mathbf{x}_b, \vec{\omega}_b, l_b, S_b, M_b, \xi$)

```
1: $(d, d_{pdf}) \leftarrow$ SAMPLEPROPAGATION
2: if $d > l_b$ then
3:     return false
4: else
5:     $\Phi_p \leftarrow \Phi_b \frac{\sigma_s' T_r(d)}{d_{pdf}}$
6:     $t_p \leftarrow d$
7:     $\mathbf{x}_p \leftarrow \mathbf{x}_b + d\vec{\omega}_b$
8:     $\mathbf{x}_p^p \leftarrow$ PROJECTTOPLANE($\mathbf{x}_p, S_b$)
9:     $S_p \leftarrow S_b$
10:    $d_p \leftarrow \|\mathbf{x}_p^p - \mathbf{x}_p\|$
11:    $l_p \leftarrow l_b(\vec{\omega}_b \cdot -\vec{\mathbf{n}})$        ▷ where $\vec{\mathbf{n}}$ is the surface normal at $S_b$
12:    $M_p \leftarrow M_b$
13:    return true
14: end if
```

the diffuse multiple scattering part of the BSSRDF (2.28) due to a reflectance profile R_d of a single source. To account for all sources, we simply sum over all diffusion photons placed in the medium:

$$L_r(\mathbf{x} \to \vec{\omega}) \approx \frac{1}{\pi} \frac{F_t(\mathbf{x}, \vec{\omega}, \eta)}{4C_\phi(\eta^{-1})} \sum_p \Phi_p R_d(\mathbf{x}, \mathbf{x}_p) \tag{3.29}$$

where p is a diffusion photon and Φ_p its power. Note that in contrast to (2.28), we have removed the Fresnel terms accounting for the change of incident light, as this is already been taken care of by the photon shooting process.

To account for arbitrary geometry, we linearly blend between the reflectance profiles of the dipole, quadpole and multipole, using the angle γ between the surface normal $\vec{\mathbf{n}}_i$ at the incident location of the photon beam and the surface normal $\vec{\mathbf{n}}$ at the shading location:

$$R_d(\mathbf{x}, \mathbf{x}_p) = \frac{2}{\pi} \begin{cases} \left(\frac{\pi}{2} - \gamma\right) R_d^d(\mathbf{x}, \mathbf{x}_p) + \gamma R_d^q(\mathbf{x}, \mathbf{x}_p), & \text{if } 0 \le \gamma \le \frac{\pi}{2} \\ (\pi - \gamma) R_d^q(\mathbf{x}, \mathbf{x}_p) + \left(\gamma - \frac{\pi}{2}\right) R_d^u(\mathbf{x}, \mathbf{x}_p), & \text{if } \frac{\pi}{2} \le \gamma \le \pi \end{cases} \tag{3.30}$$

where $R_d^d(\mathbf{x}, \mathbf{x}_p)$, $R_d^q(\mathbf{x}, \mathbf{x}_p)$ and $R_d^u(\mathbf{x}, \mathbf{x}_p)$ are the reflectance profiles of the dipole, the quadpole and the multipole due to a source at $\mathbf{x}_p$. We illustrate the profile blending in Figure 3.9.

Dipole

For the dipole configuration (see Figure 2.8), we define $z_r = d_p$, the depth of the photon in relation to the surface plane S_p and $z_v = -z_r - 2z_b$. We compute the distances using $d_r = \sqrt{r^2 + z_r^2}$ and $d_v = \sqrt{r^2 + z_v^2}$, where $r = \|\mathbf{x}_p^p - \mathbf{x}\|$ is the distance between the shading location and the photon position projected to the surface plane S_p.

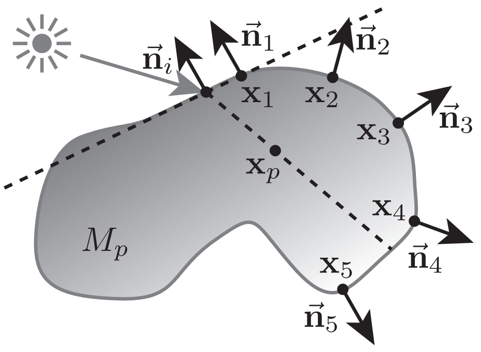


Figure 3.9: *Evaluating the reflectance profile: (1) using only the dipole as the surface normal $\vec{\mathbf{n}}_1$ is equal to $\vec{\mathbf{n}}_i$, (2) blending between the dipole and the quadpole, (3) using only the quadpole as the surface normal $\vec{\mathbf{n}}_3$ is perpendicular to $\vec{\mathbf{n}}_i$, (4) blending between the quadpole and the multipole, (5) using only the multipole as the surface normal $\vec{\mathbf{n}}_5$ is opposite to $\vec{\mathbf{n}}_i$.*

Quadpole

For the quadpole configuration (see Figure 2.9a), we define z_r and z_v equally to the dipole configuration, $x_r = \|\mathbf{x}^p - \mathbf{x}_p^p\|$, where $\mathbf{x}^p$ is the shading location projected to the surface plane S_p and $x_v = -x_r - 2z_b$. The distances are computed as:

$$\begin{aligned} d_r &= \sqrt{x_r^2 + (z_r - s)^2} \\ d_v &= \sqrt{x_r^2 + (-z_v + s)^2} \\ d_{rm} &= \sqrt{x_v^2 + (-z_v + s)^2} \\ d_{vm} &= \sqrt{x_v^2 + (z_r - s)^2} \end{aligned} \tag{3.31}$$

where s is the shortest distance from shading location $\mathbf{x}$ to the surface plane S_p.

Multipole

For the multipole configuration (see Figure 2.9b), we define $d = d_p$, the depth of the photon in relation to the surface plane S_p and $l = l_p$, the approximated thickness of the slab. The definition of $z_{r,i}$ and $z_{v,i}$ follow directly from (2.43), and we compute the distances as $d_{r,i} = \sqrt{r^2 + z_{r,i}^2}$ and $d_{v,i} = \sqrt{r^2 + z_{v,i}^2}$ where again $r = \|\mathbf{x}_p^p - \mathbf{x}\|$ is the distance between the shading location and the photon position projected to the surface plane S_p.

Classical and Improved Diffusion

In the original paper, Donner et al. used the classical diffusion theory to obtain the reflectance profiles. To counter the fact that classical diffusion theory approximates both single and multiple scattering, they introduced a correction term κ, which modulates the power of the diffusion photons based on an estimate of how much of the source's power contributes to non-single-scattered light:

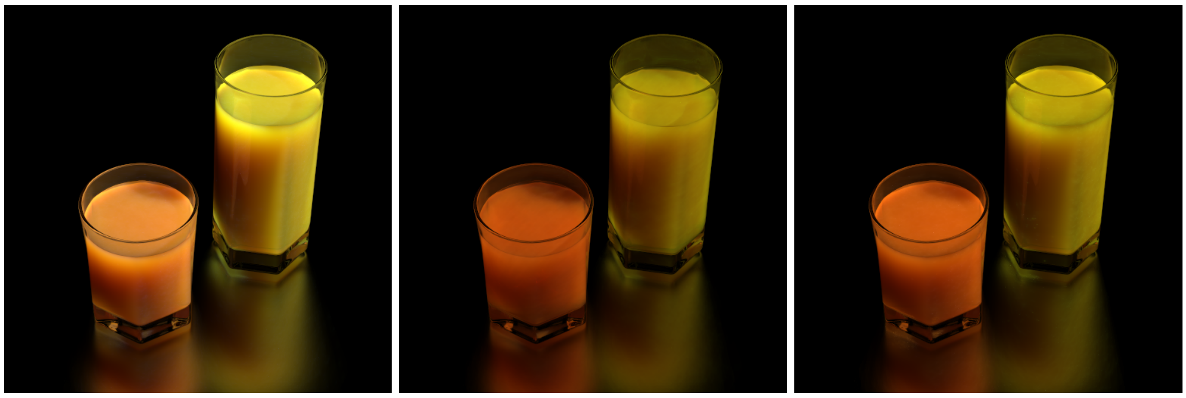
$$\kappa(d) = 1 - e^{-\sigma_t d} \tag{3.32}$$

where d is the distance from the shading location to the photon position. Even with this correction term applied, we observed that rendered objects tend to be too bright.

We have also implemented photon diffusion based on the improved diffusion theory, where the single scattering response is explicitly removed from the diffusion approximation. We have used the empirical correction term proposed by Habel et al. [HCJ13b], which modulates the power for near-surface sources:

$$\kappa(d, t) = 1 - e^{-2\sigma_t (d+t)} \tag{3.33}$$

where d is the distance from the shading location to the photon position and t is the distance from the beam incident location to the photon position. Using the improved diffusion theory results in much more accurate renderings, as shown in Figure 3.10.



(a) *Classical Diffusion* ***(b)*** *Monte Carlo Reference* ***(c)*** *Improved Diffusion*

Figure 3.10: *Comparison between photon diffusion with classical and improved diffusion theory.*

Extensions

For the sake of clarity, we kept our implementation of photon diffusion minimal, omitting some of the ideas proposed in the original paper. We still want to mention two possible extensions, as they are relatively simple to implement.

In order to improve the efficiency of the algorithm, one can implement hierarchical evaluation of the reflectance profiles. The rationale behind this idea is coming from the fact that the contribution from sources falls off exponentially with increased distance to the shading location. This allows distant sources to be clustered together, acting as a single source, similar to what we know from solutions to various N-Body problems.

The second extension is support for volumetric shadowing, which occurs if we have opaque objects embedded in the medium. Shooting photon beams already solves part of the problem by absorbing beams directly hitting the opaque objects, leaving an area of shadow in the medium behind the occluder. When adding the contributions of the different sources during shading, we can trace shadow rays from the shading location to the sources, in order to exclude contributions from sources that are invisible due to an occluding object, therefore allowing for more accurate volumetric shadowing.

3.7 Photon Beam Diffusion

Photon beam diffusion [HCJ13b], recently introduced by Habel et al., is a new rendering method building upon the improved diffusion theory introduced in Section 2.6.4 and a Monte Carlo integration framework to compute subsurface scattering effects. The method is well suited for both traditional subsurface scattering rendering, where irradiance samples in a point cloud are convolved with a reflectance profile to compute reflected radiance, as well as photon beam based rendering, where the contribution of oblique photon beams is directly computed using Monte Carlo integration. In addition to more accurately rendering the multiple scattering response, Habel et al. also introduce a novel approach for rendering exact diffuse single scattering.

Method

The core idea behind the photon beam diffusion method is the numerical integration of the integral describing the reflectance profile due to a beam with unit power after the Fresnel boundary, as previously stated in (3.28):

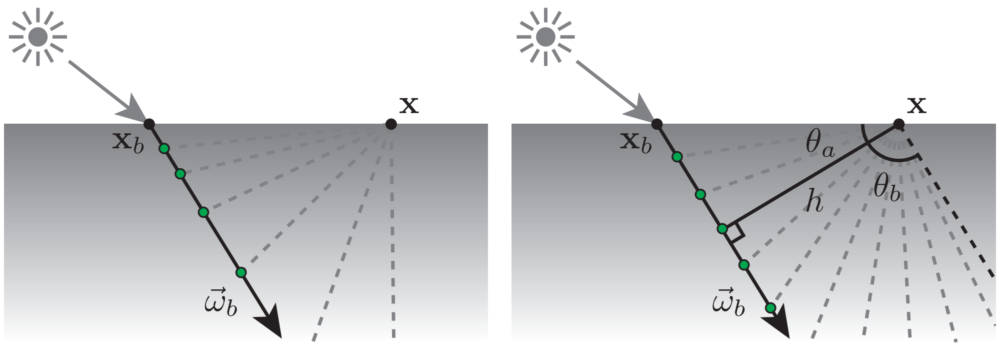
$$R_d^b(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) = \int_0^\infty Q(t)\; R_d(\mathbf{x}, \mathbf{x}_b + t\vec{\omega}_b) dt \quad \text{with} \quad Q(t) = \sigma_s'\, e^{-\sigma_t' t} \tag{3.34}$$

where $\mathbf{x}_b$ is the beam origin, $\vec{\omega}_b$ is the beam direction and $R_d(\mathbf{x}, \mathbf{x}_p)$ is the reflectance profile due to a source at position $\mathbf{x}_p$. We can numerically solve this integral using Monte Carlo integration with importance sampling:

$$R_d^b(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) \approx \sum_{i=1}^{N} \frac{f(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t_i)}{pdf(t_i \mid \mathbf{x}, \mathbf{x}_b, \vec{\omega}_b)} \tag{3.35}$$

where $f(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t_i) = Q(t_i)\; R_d(\mathbf{x}, \mathbf{x}_b + t_i\vec{\omega}_b)$ is the integrand in (3.34) and $pdf(t_i \mid \mathbf{x}, \mathbf{x}_b, \vec{\omega}_b)$ is the PDF of choosing t_i, given the shading location $\mathbf{x}$, beam origin $\mathbf{x}_b$ and beam direction $\vec{\omega}_b$.

To evaluate (3.35), we can use an arbitrary sampling strategy, but preferably choose one that matches the behavior of the integrand f, in order to minimize variance.



(a) *Exponential Sampling* ***(b)*** *Equiangular Sampling*

Exponential Sampling

The traditional way to sample propagation distances inside a homogeneous medium is exponentially-decreasing sampling with the PDF proportional to the source term of $Q(t)$, as shown in Figure 3.11a:

$$t_i = -\frac{\log(1-\xi_i)}{\sigma_t'} \quad \text{with} \quad pdf_{exp}(t_i) = \sigma_t' \, e^{-\sigma_t' t_i} \tag{3.36}$$

where $\xi_i \in [0, 1)$ is uniformly distributed. The PDF is only dependent on t_i and is completely agnostic of the sampling location $\mathbf{x}$, as well as the beam defined by $\mathbf{x}_b$ and $\vec{\omega}_b$. Using photon beam diffusion with exponential sampling is basically the equivalent to the photon diffusion method introduced in Section 3.6.

Equiangular Sampling

Another strategy is to place samples in the angular domain subtended by the refracted beam in respect to the sampling location $\mathbf{x}$ as shown in Figure 3.11b . This corresponds to *equiangular sampling* as proposed by Kulla and Fajardo [KF12] and is defined by:

$$t_i = h \tan\theta_i \quad \text{with} \quad pdf_{ea}(t_i) = \frac{h}{(\theta_b - \theta_a)\,(h^2 + t_i^2)} \tag{3.37}$$

where h is the nearest distance between the shading location $\mathbf{x}$ and the refracted beam, θ_a and θ_b are the start and end angles of the integration in the angular domain, and $\theta_i = lerp(\theta_a, \theta_b, \xi)$, the linear interpolation between θ_a and θ_b.

Deterministic Sampling

While standard Monte Carlo methods usually use random or pseudo random sequences for ξ_i, Habel et al. propose to use a deterministic regular sequence defined by $\xi_i = \frac{i-0.5}{N}$, in order to avoid per-pixel noise from the profile evaluation. This basically turns the Monte Carlo integration into a custom numerical quadrature and produces accurate results even with a low number of samples (3-5).

Multiple Importance Sampling

When comparing the results for profile evaluation using exponential and equiangular sampling, we can find that exponential sampling performs well in the tail of the profile, while underestimating the peak of the profile. In contrast, equiangular sampling performs well in the peak, while underestimating the tail of the profile. Habel et al. propose a *multiple importance sampling* [VG95] (MIS) strategy, to combine the strengths and weaknesses of both sampling strategies. In our own implementation, we solely rely on equiangular sampling for simplicity, as it provides better results in the more important peak region of the profile.

We have found that deterministic equiangular sampling alone introduces bias, which is trivially solved by replacing the regular sequences with $\xi_i = \frac{i-\xi_p}{N}$, where ξ_p is a uniform random number set once per pass.

Multiple Scattering

To evaluate the reflected radiance L_r at the shading location $\mathbf{x}$ in direction $\vec{\omega}$ due to multiple scattering, we can adapt (3.29) and sum over all photon beams instead of the diffusion photons:

$$L_r(\mathbf{x} \to \vec{\omega}) \approx \frac{1}{\pi} \frac{F_t(\mathbf{x}, \vec{\omega}, \eta)}{4C_\phi(\eta^{-1})} \sum_b \Phi_b R_d^b(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) \tag{3.38}$$

where b is a photon beam, Φ_b the power and R_d^b the reflectance profile. To compute the reflectance profile of a single beam, we use the Monte Carlo estimator (3.35) with discrete equiangular sampling. Note that to evaluate R_d in the Monte Carlo estimator, we use the same profile blending as described in Section 3.6.

Diffuse Single Scattering

In addition to improved rendering of multiple scattering, Habel et al. also introduced a novel approach to render diffuse single scattering based on the following Green's function:

$$R^{(1)}(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t) = \frac{p(\vec{\omega}_b \to \vec{\omega}_{x_t x})\ T_r(d_{x_t x})\ F_t(\theta, \eta^{-1})\ \cos\theta}{d_{x_t x}^2} \tag{3.39}$$

where $\mathbf{x}_t = \mathbf{x}_b + t\vec{\omega}_b$ is a point along the beam (source), $d_{x_t x}$ is the distance and $\vec{\omega}_{x_t x}$ is the direction from the source at $\mathbf{x}_t$ to the shading location $\mathbf{x}$ and $\theta = \arccos(\vec{\omega}_{x_t x} \cdot \vec{\mathbf{n}})$ is the incident angle of the light emitted from the source in respect to the shading location on the inside of the surface. Thus, the Fresnel term F_T accounts for the amount of light exiting the medium through the surface.

To compute the single scattering reflectance profile due to a single beam $R^{(1),b}$, we integrate $R^{(1)}$ along the beam and use the same Monte Carlo estimator as used for multiple scattering (3.35), with a redefined integrand $f(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t_i) = Q(t_i)\, R^{(1)}(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t_i)$. Finally, to compute the reflected radiance, we sum over all photon beams:

$$L_r(\mathbf{x} \to \vec{\omega}) \approx \frac{1}{\pi} \sum_b \Phi_b R^{(1),b}(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) \tag{3.40}$$

Note that we do not need the normalization constant as in the multiple scattering case.

Extensions

Our implementation of photon beam diffusion directly uses the photon beams to compute the reflectance at surface points from oblique illumination. As an alternative, the method can also be applied in a more traditional setting, where we convolve a set of irradiance samples with the reflectance profiles. This of course prevents oblique illumination, but due to the symmetry in the reflectance profiles, allows for various optimizations. Habel et al. have proposed a number of techniques, such as caching and smoothing of the reflectance profiles, to enhance both performance and visual fidelity. Unfortunately, due to non-symmetric reflectance profiles, these techniques are difficult to apply in the oblique illumination setting. This is the main reason for omitting the proposed techniques in our implementation of PBD.

4 Implementation

Over the course of this thesis, we have implemented the renderer *PMRender*, used as a testbed for implementation, testing and comparison of the different photon beam based rendering methods presented in Chapter 3. This chapter gives a brief summary on the usage and implementation of *PMRender*.

4.1 Overview

The software is written in portable C++, allowing it to be compiled on various different compilers and run on different systems. We have successfully compiled and run *PMRender* on the following compiler and operating system combinations:

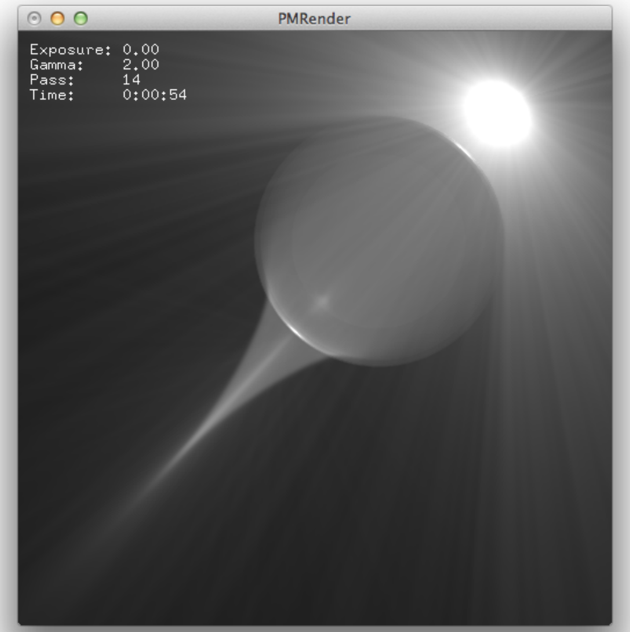
- OSX with LLVM-GCC (32-bit and 64-bit)
- Ubuntu Linux with GCC (32-bit)
- Windows 7 with Visual C++ 2010 (32-bit)

Supporting additional configurations should be relatively easy due to the use of the CMake build system [HMK$^+$00], which allows for automatic generation of project files for various integrated development environments (IDEs) such as XCode, Eclipse or Visual Studio as well as standard Unix Makefiles. In addition to CMake, we also use the following third party libraries:

- **OpenGL/GLUT** for the main application window and fast preview of the scene geometry
- **OpenEXR** [Mag00] for reading/writing EXR image files
- **LodePNG** [Van05] for reading/writing PNG image files

- **tinythread** [Gee10] for cross-platform threading support
- **fmath** [Mit09] for a faster exponential function `exp`
- **rapidjson** [Yip11] for reading JSON files

4.2 User Interface



```
Exposure: 0.00
Gamma:    2.00
Pass:     14
Time:     0:00:54
```

Figure 4.1: *PMRender running on Mac OSX with enabled HUD.*

The application provides a very simplistic GUI (Figure 4.1), showing the scene as it is progressively rendered. In addition, the GUI allows for the following user interactions:

- Display OpenGL preview of the scene (wireframe overlay)
- Display HUD with additional information (number of passes, total render time, etc.)
- Camera navigation (rotate & zoom)
- Exposure and gamma correction of the displayed image
- Saving images

To control the GUI, the following mouse and keyboard commands are available.

`LMB + Drag`	Rotate camera
`RMB + Drag`	Zoom camera
`h`	Toggle help screen
`H`	Toggle HUD
`p`	Toggle OpenGL preview
`e/E`	Decrease/Increase exposure correction
`g/G`	Decrease/Increase gamma correction
`s`	Save current image
`ESC`	Quit

4.3 Software Architecture

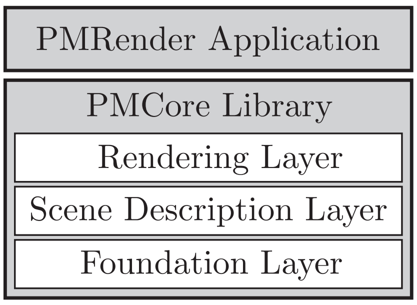


Figure 4.2: *Overview of the software architecture used in PMRender.*

The software architecture for *PMRender* consists of three layers: the foundation layer, the scene description layer and the rendering layer. These are implemented in the *PMCore* library, which is linked against the *PMRender* application, implementing the GUI and the glue code to communicate with the renderer.

We have thoroughly commented the source code in order to automatically generate software documentation using Doxygen [vH97]. The following sections provide a high-level description of the implementation. For details, please consult the source code documentation.

4.3.1 Foundation Layer

The foundation layer consists of an extensive set of classes, providing the basic building blocks for the rest of the application. It also acts as an abstraction layer, allowing the rest of the application to be independent of the underlying operating system. The foundation layer consists, among others, of the following namespaces and classes:

- Math and linear algebra (`Math::` namespace)

- Random number generation and quasi random sequences (`Rand::` namespace)
- OpenGL abstraction layer (`OGL::` namespace)
- Image handling (`Image`, `ImageReader`, `ImageWriter`)
- High resolution timer (`Timer`)
- Threading abstraction layer (`ThreadPool`, `Task`)

4.3.2 Scene Description Layer

The scene description layer consists of a set of classes to describe the virtual scene to be rendered. In addition, many of the classes provide additional methods, used by the rendering layer, to compute various quantities or generate sample data, e.g. the `Shape` classes provide ray intersection methods and the `BSDF` classes provide methods to evaluate as well as sample the BSDFs.

`Scene`

The `Scene` class is used to manage all objects describing a virtual scene, including shapes, BSDFs, textures, media, lights and a camera. Scenes are currently defined in source code, each scene inheriting from the `Scene` class, implementing the `build()` function to create all the assets of the scene.

`Shape`

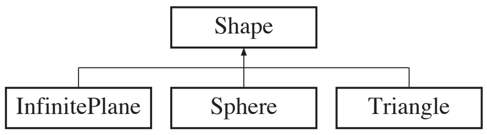


Figure 4.3: *Shape class hierarchy*

The `Shape` classes (Figure 4.3) describe the basic geometric primitives to build a scene. We currently provide only three kinds of shapes, `Triangle` being the most versatile, as it can be used for rendering arbitrary triangular meshes. Each shape has a BSDF assigned to it, describing the appearance of the shape when rendered. For improved ray intersection performance, shapes can be put into a bounding volume hierarchy (BVH).

`BSDF`

The `BSDF` classes (Figure 4.4) provide a set of bidirectional scattering distribution functions used to describe scattering at surfaces. `LambertBSDF` provides a diffuse reflection model, `PhongBSDF` provides a glossy specular reflection model and `FresnelBSDF` provides a smooth dielectric model providing both reflection and transmission. Each BSDF has an *outside* and an

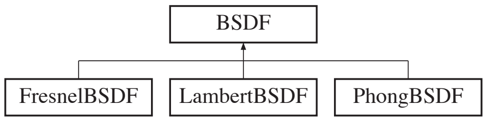


Figure 4.4: *BSDF class hierarchy*

optional *inside* medium assigned to it, the outside medium being in direction of the surface normal. This allows transmissive BSDFs to automatically compute the relative index of refraction and explicitly defines the the two media joining at the interface.

`Texture`

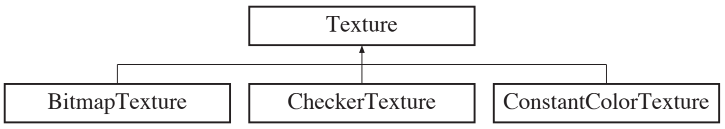


Figure 4.5: *Texture class hierarchy*

The `Texture` classes (Figure 4.5) are used to define spatially varying parameters for the BSDFs, as well as for an optional environment map.

`HomogeneousMedium`

As we currently only support homogeneous media, we have not implemented a class hierarchy representing media. We only provide the `HomogeneousMedium` class, used to describe the properties of a homogeneous participating medium.

`Light`

As we currently only support one type of light source, namely spot lights, we have directly implemented them in the concrete class `Light`.

`Camera`

The `Camera` class is used to describe the virtual camera in the scene.

4.3.3 Rendering Layer

The rendering layer provides a set of base classes for rendering images as well as the implementation of the various rendering methods described in Chapter 3. We tried to keep the design lightweight, yet versatile enough for experimentation, such that new rendering methods can be implemented easily by writing small components and joining them with the rest of the system.

`PhotonBeam`

The `PhotonBeam` class describes a photon beam and consists of the same properties as described in Table 3.1. For efficient ray intersections, e.g. for the progressive photon beam algorithm, photon beams can be split into sub beams, using the `SubBeamAccelObject` wrapper class, and put into a BVH acceleration structure.

`PhotonBeamShooter`

The `PhotonBeamShooter` class implements the photon beam shooting process described in Section 3.3. It essentially creates a list of `PhotonBeam` objects to be used with the various rendering methods.

`DiffusionProfile`

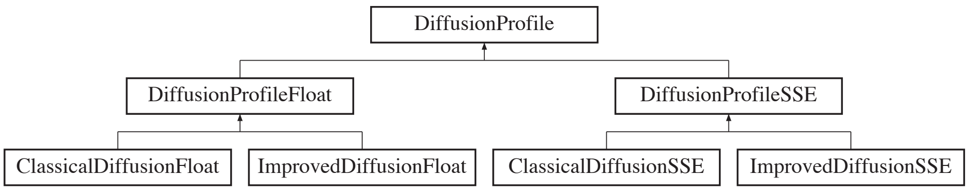


Figure 4.6: *DiffusionProfile class hierarchy*

The abstract base class `DiffusionProfile` defines an interface for computing reflectance profiles using diffusion theory. The `DiffusionProfileFloat` and `DiffusionProfileSSE` base classes implement the profile blending as described in Section 3.6 and define abstract methods to compute the directional derivative of a monopole and an optional correction term. We have implemented floating point and SSE optimized versions. The `ClassicalDiffusionXXX` and `ImprovedDiffusionXXX` classes implement the monopole and correction terms for classical and improved diffusion theory. This abstraction allows the rendering code to easily switch between the two diffusion models and floating point or SSE optimized versions.

`Integrator`

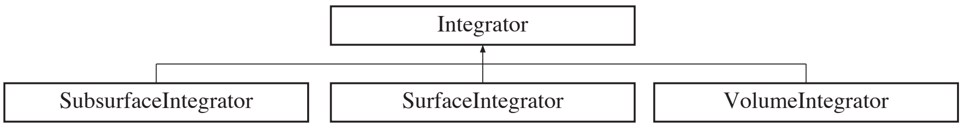


Figure 4.7: *Integrator class hierarchy*

Integrators are used to compute radiance for different types of queries. The abstract base class `Integrator` (Figure 4.7) defines the interface for common tasks such as preprocessing, and lets the descendants define the methods for the actual queries.

`SurfaceIntegrator`

Descendants of the abstract base class `SurfaceIntegrator` are responsible for computing the incident radiance at any point from any direction, e.g. at surfaces or at the camera origin. Surface integrators should only compute scattering at surfaces and rely on the volume and subsurface integrators provided by the renderer, to compute contributions from participating media. We currently only provide a minimalistic path tracer, implemented in `PTSurfaceIntegrator`.

`VolumeIntegrator`

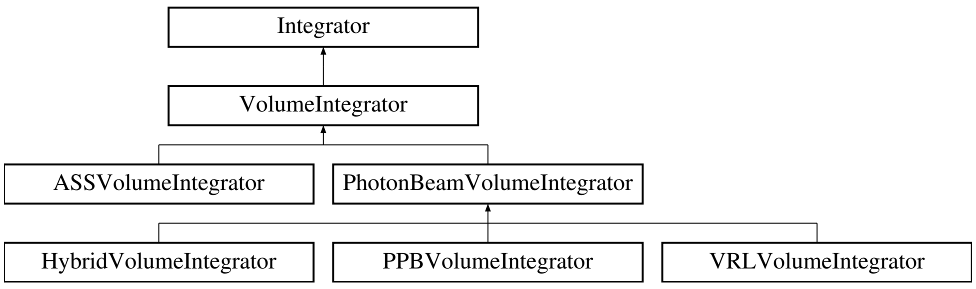


Figure 4.8: *Volume integrator class hierarchy*

Descendants of the abstract base class `VolumeIntegrator` (Figure 4.8) are responsible for computing the accumulated inscattered radiance from a medium along a given ray segment. We have implemented the following volume integrators:

- `ASSVolumeIntegrator`, computing analytic single scattering using Monte Carlo integration
- `PPBVolumeIntegrator`, implementing progressive photon beams (PPB) as described in Section 3.4
- `VRLVolumeIntegrator`, implementing virtual ray lights (VRL) as described in Section 3.5
- `HybridVolumeIntegrator`, using PPB to compute single scattering, VRL to compute multiple scattering

`SubsurfaceIntegrator`

Descendants of the abstract base class `SubsurfaceIntegrator` (Figure 4.9) are responsible for computing exitant radiance at a given surface location. We have implemented the following subsurface integrators:

- `PDSubsurfaceIntegrator`, implementing photon diffusion (PD) as described in Section 3.6
- `PBDSubsurfaceIntegrator`, implementing photon beam diffusion (PBD) as described in Section 3.7

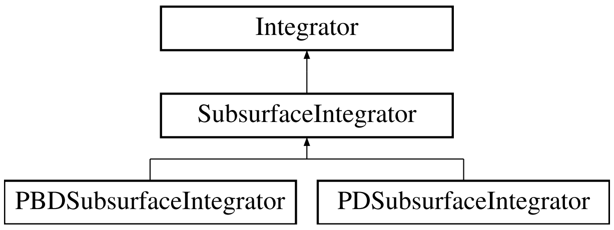


Figure 4.9: *Subsurface integrator class hierarchy*

`ProgressiveRenderer`

The `ProgressiveRenderer` class implements the core of the renderer. It is responsible for progressively rendering passes, which are continuously accumulated to a final image. To allow for rendering with different methods, the renderer is configured with a *surface* and optional *volume* or *subsurface* integrators. To render a pass, the renderer first calls the preprocessing methods of the integrators, then spawns a set of tasks, each rendering an interleaved set of scanlines. For each pixel, we generate a ray and query the surface integrator to compute the incident radiance.

5 Results

5.1 Multiple Scattering

In this section, we compare the multiple scattering solutions of the four rendering methods PPB, VRL, PD and PBD, as VRL and PD do not provide single scattering. To compare the rendering methods, we created a simple test scene (GLASSCOMPARISON) containing a drinking glass filled with a liquid. We chose two materials, the first being *beer*, which is highly absorbing and has very little scattering (low albedo), the second being *milk*, which is highly scattering (high albedo). The material parameters are taken from a list of physical measurements by Narasimhan et al. [NGD+06] and are scaled to match the dimensions of the virtual scene. As both materials are strongly forward scattering, but the PD and PBD rendering methods only handle isotropic media, we used the *principle of similarity* [PH10, Chapter 16.5.3] to derive reduced scattering and extinction coefficients, which were used across all rendering methods, in order to allow for better comparison. In addition, we have lowered the concentration of milk by a factor of 10,

Material	**Absorption coefficient** (σ_a)			**Scattering coefficient** (σ_s)		
	($\times\ 10^{-2}\ mm^{-1}$)			($\times\ 10^{-2}\ mm^{-1}$)		
	R	G	B	R	G	B
Beer (yuengling)	0.8248	2.2215	5.4367	0.0114	0.0120	0.0137
Milk (regular)	0.0154	0.0460	0.1994	51.885	58.090	63.754

Table 5.1: *Scattering and absorption coefficients used for materials in comparison renderings.*

as the material in full concentration was too dense for our test renderings, absorbing almost all light shortly after it enters the medium. Table 5.1 shows the scaled and reduced material parameters as used in our test scene. From the two base materials we derived a series of *mixed* materials by blending the scattering and absorption coefficients [NGD+06]. This allows us to compare the rendering methods at various albedos.

Render Settings

In order to reduce the effects of convergence from the path tracer (anti aliasing, reflections etc.), we balanced the render times per pass across all methods. To achieve this, we adjusted the number of photon beams used per pass in each method, specifically we used 2000 photon beams for PPB, 100 for VRL and 400 for PD and PBD. We used the following render settings for the different methods:

PPB: We set the beam radius to $r_b = 0.03$ and the convergence parameter to $\alpha = 0.8$. We use a maximum photon beam media depth of 250 bounces, which is required especially for the high albedo cases.

VRL: We use one sample per product evaluation and disable visibility rays, as they have negligible effects on the appearance due to the medium being enclosed in a mostly convex volume. We use a maximum of 249 medium bounces, as VRLs perform one additional implicit bounce.

PD: We use the classical diffusion model as proposed in the original paper [DJ07]. A comparison between PD with the improved diffusion model and PBD will yield almost identical results, as they only differ in their sampling strategies. We disabled the contributions from the multipole, as they are computationally the most expensive for evaluation and contribute only a negligible amount of radiance in our test scene.

PBD: We use 1 sample per photon beam for the profile evaluation and also disabled the contributions from the multipole.

All images are rendered at a resolution of 256×512 pixels on an Intel Core i7-2600 CPU @ 3.40 GHz with 16GB RAM, utilizing all 8 cores (4 physical, 4 virtual) in parallel.

Presentation

We present our results in the comparison images starting with Figure 5.2. Each image series shows the final rendered images using PPB, VRL, PD and PBD side by side, all rendered with equal material parameters and total render time. In addition, we visualize the convergence of each method by showing the intermediate results of the same sub area at 5, 10 and 15 minutes. We denote the mix ratio by m, where $m = 0.0$ is beer and $m = 1.0$ is milk. Furthermore, we denote the average albedo by α and the total render time T.

Low Albedo

We start our comparison with the low albedo cases ($\alpha = 0.01$ and $\alpha = 0.23$), shown in Figure 5.2 and Figure 5.3, all rendered at a total render time of $T = 15$ min. The appearance of the liquid is primarily affected by absorption in the medium (transmittance term), such that the contribution from the rendering methods is hardly noticeable. PPB performs the worst in this situation, as most of the rendering time is spent in the photon beam shooter, trying to create photon beams after the first scattering event, which occurs with very low probability only (less than 0.04 % of the emitted beams in the first series). In the second series (Figure 5.3), we start to see a dim gray shimmer in the PD rendering due to the inaccuracies in the classical diffusion model. Despite all methods converging very fast, we could increase rendering performance considerably by ignoring the scattering terms altogether. This will generally lead to good approximations for materials with very low albedos.

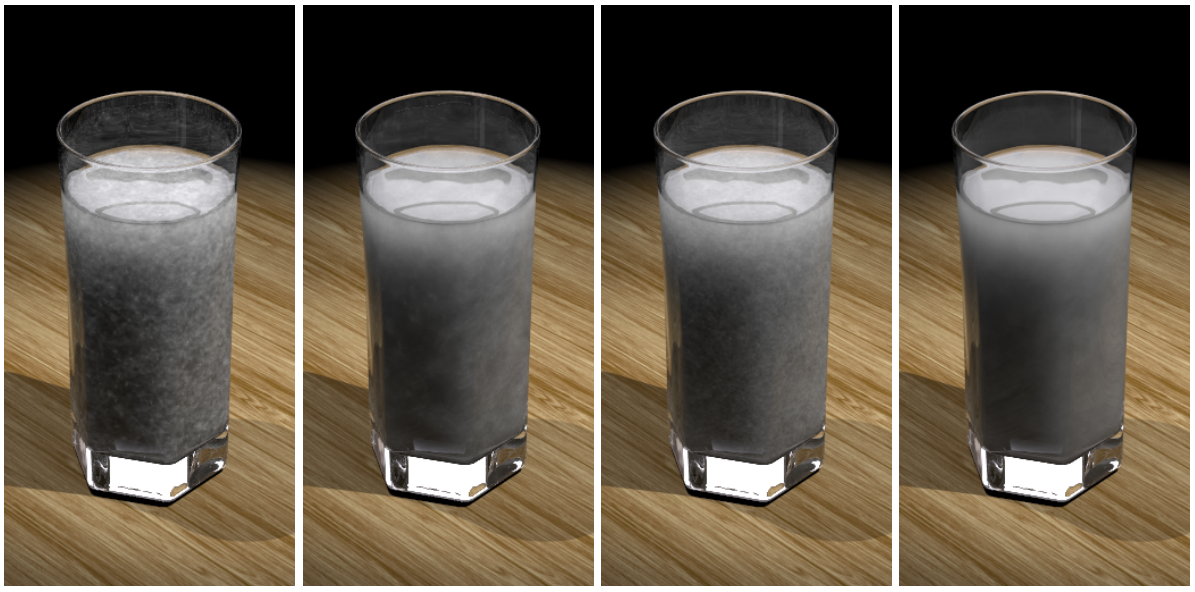
Medium Albedo

Next, we compare the methods for medium albedos ($\alpha = 0.53$ and $\alpha = 0.77$), shown in Figure 5.4 and Figure 5.5. We increase the total render time to $T = 20$ min. The liquid starts to show a *milky* appearance due to more light being scattered within the medium. The inaccuracies of the classical diffusion model used in PD become more visible. The same applies for PBD, which shows some inaccuracies compared to PPB and VRL, primarily in the rim where the liquid meets with glass and air. At this point, especially in the second series (Figure 5.5), we notice the improved convergence rate for the diffusion based methods in return of sacrificed accuracy. While both diffusion methods have converged at $T = 5$ min, PPB and VRL show variance before converging at about $T = 20$ min. The variance in the VRL rendering is more visually pleasing, as it exhibits less high frequency noise than PPB.

High Albedo

In the last three series, we compare the methods for the high albedo cases ($\alpha = 0.91$, $\alpha = 0.96$ and $\alpha = 0.9987$), shown in Figure 5.6, Figure 5.7 and Figure 5.8. We increase the total render time to $T = 30$, $T = 35$ and $T = 40$ min. At this point, PPB and VRL do not fully converge and show increasing variance with higher albedos, even despite the extended render time. At albedos of 0.9 and higher, PPB and VRL start to become infeasible to produce quality results in a reasonable render time. In contrast, the diffusion methods still converge quite fast, but also take increasingly longer to converge with higher albedos. We also note increasing visual differences between the solutions of PPB and VRL compared to the diffusion methods, most obvious at the boundary between liquid and air, which shows higher intensity in the PPB and VRL solutions. This difference presumably follows from the inaccuracies in the diffusion model as well as the empirical correction terms to attenuate near-surface sources.

Comparing the diffusion methods, we observe that PD converges slightly faster than PBD, which can be explained as follows: When rendering with PD, we create a single set of diffusion photons once per pass, as exponential sampling is invariant to the shading location. On the other hand, when rendering with PBD, we need to create the sources on the fly due to using equiangular sampling, which depends on the current shading location. Computing the source intensities involves an exponential (transmittance term), which is expensive to evaluate and needs to be computed many times more for PBD than for PD.



***(a)** PPB (200K beams)* ***(b)** VRL (200K beams)* ***(c)** PPB (1M beams)* ***(d)** VRL (1M beams)*

***Figure 5.1:** Comparison of variance between PPB and VRL using an equal number of photon beams. Note that the render times are vastly different (5 minutes for PPB against 90 minutes for VRL @ 1M photon beams).*

We observe similar convergence between PPB and VRL, with the main difference being the type of variance (higher frequency for PPB, lower frequency for VRL), contradicting the results from the original VRL paper [NNDJ12]. To render the FRUITJUICE scene, which is the most comparable to our test scene, Novak et al. used a CPU implementation of PPB and compared it against a VRL implementation utilizing the GPU to compute the VRL connections. Our implementation of VRL is purely CPU based and is computationally much more expensive than our PPB implementation. A single pass using 2K photon beams takes around 0.35 seconds for PPB and 7.0 seconds for VRL. To demonstrate the superiority of the VRL method, we compare the variance between PPB and VRL using the same number of photon beams (Figure 5.1).

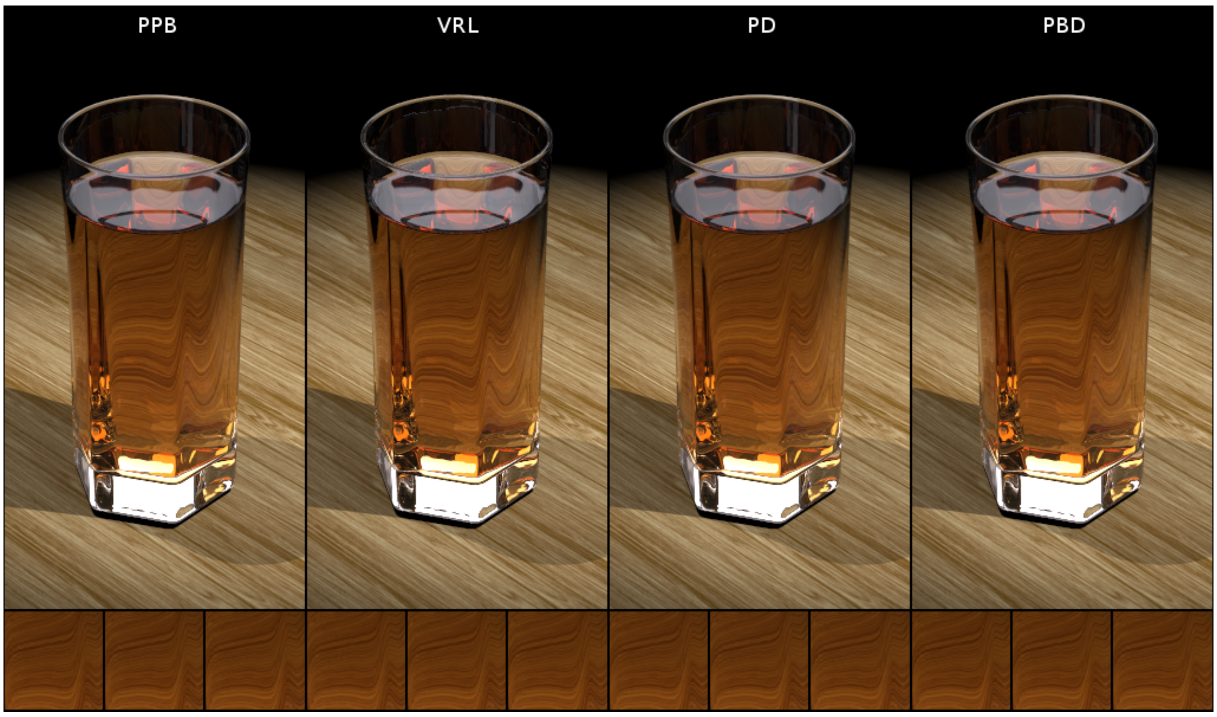


Figure 5.2: *Comparison for* $m = 0.0$ *(beer only),* $\alpha = 0.01$, $T = 15$ *min.*

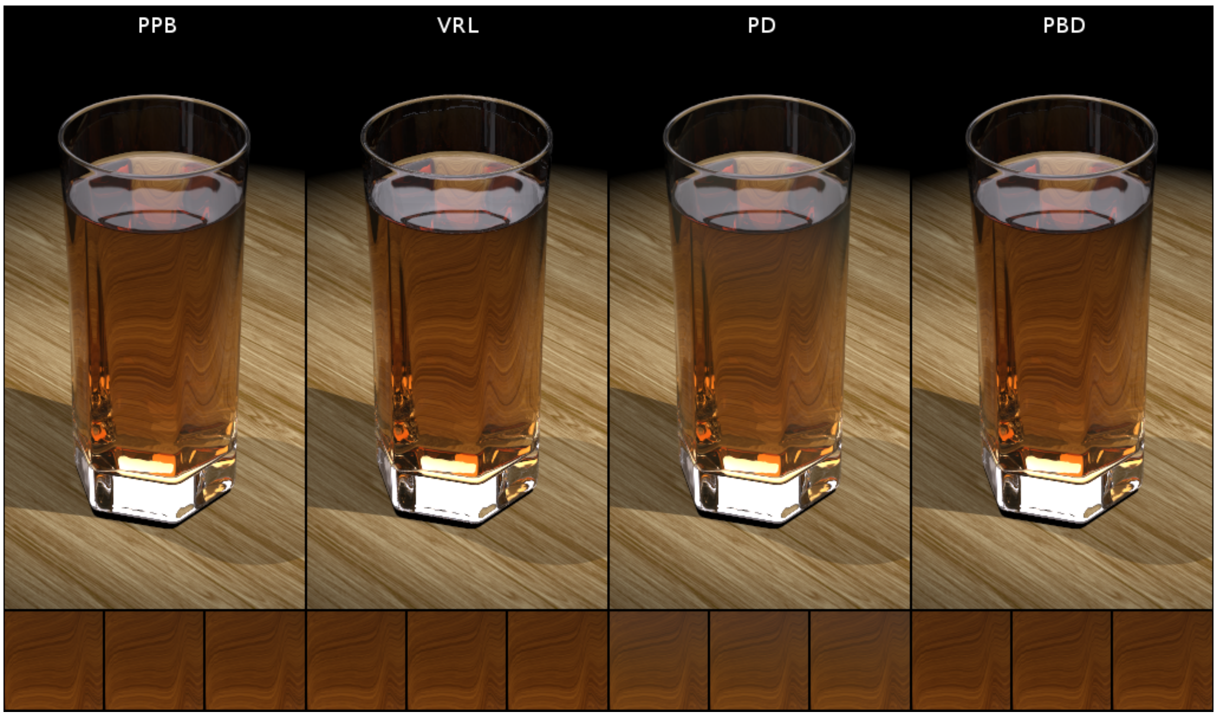


Figure 5.3: *Comparison for* $m = 0.01$, $\alpha = 0.23$, $T = 15$ *min.*

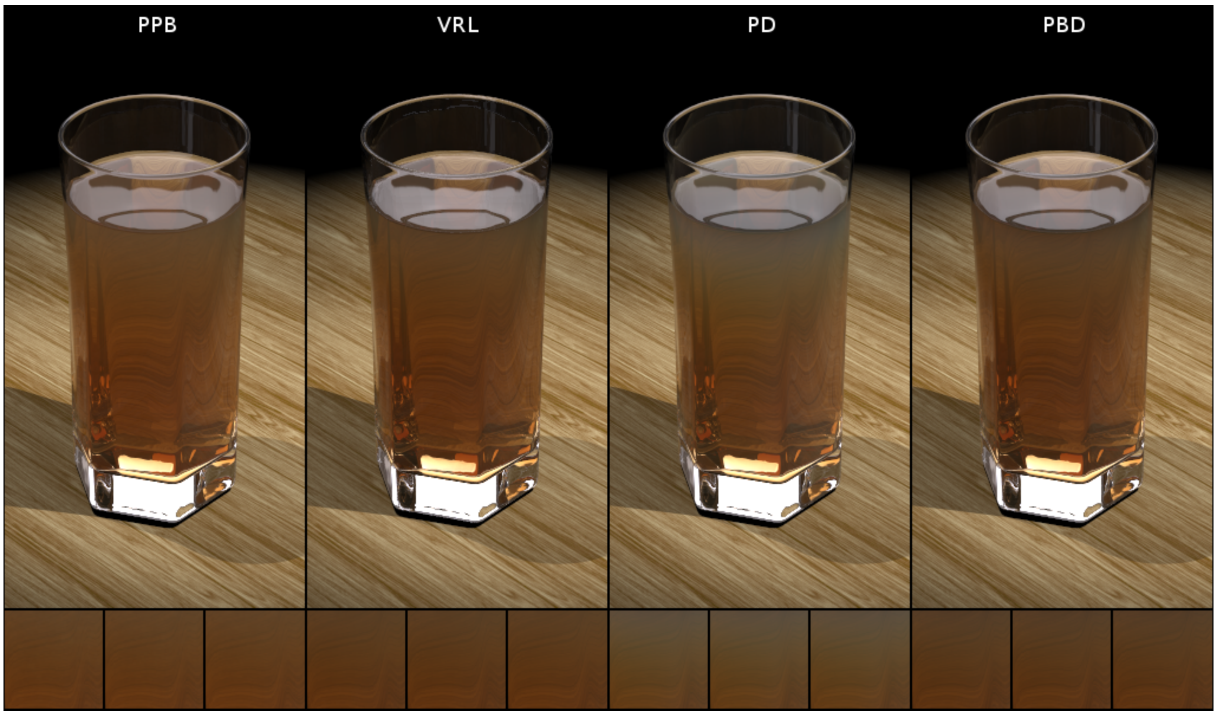


Figure 5.4: *Comparison for* $m = 0.04$, $\alpha = 0.53$, $T = 20$ *min.*

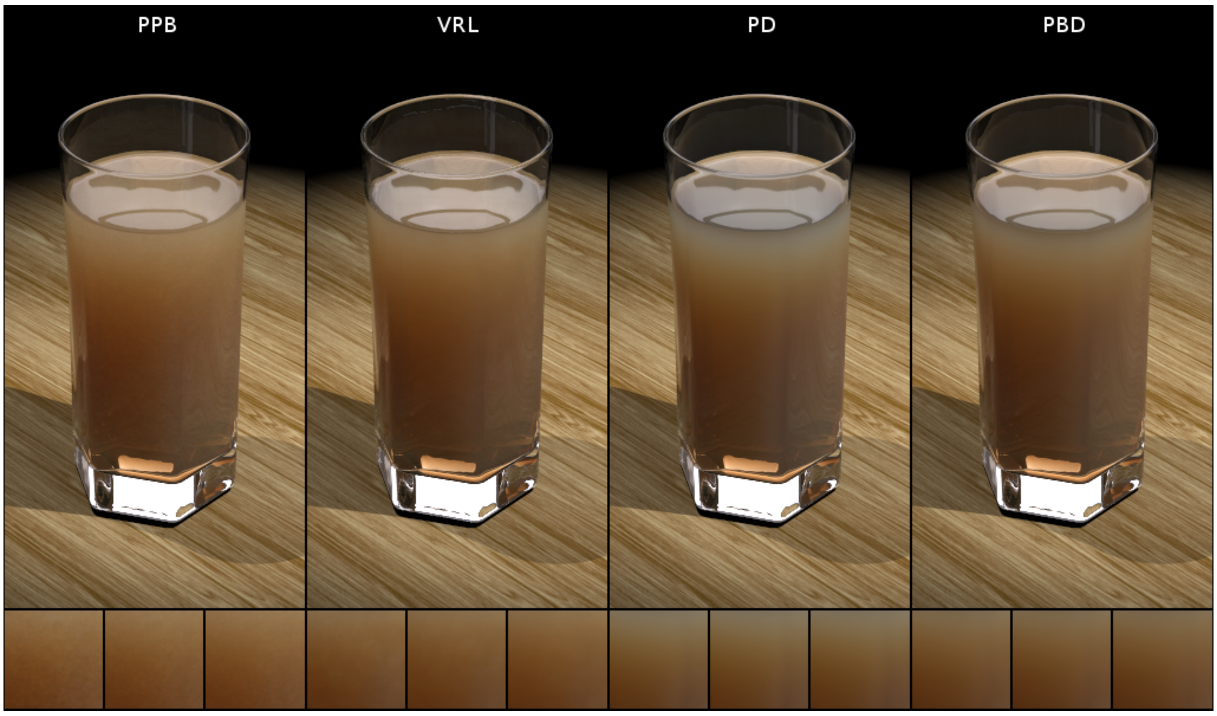


Figure 5.5: *Comparison for* $m = 0.12$, $\alpha = 0.77$, $T = 20$ *min.*

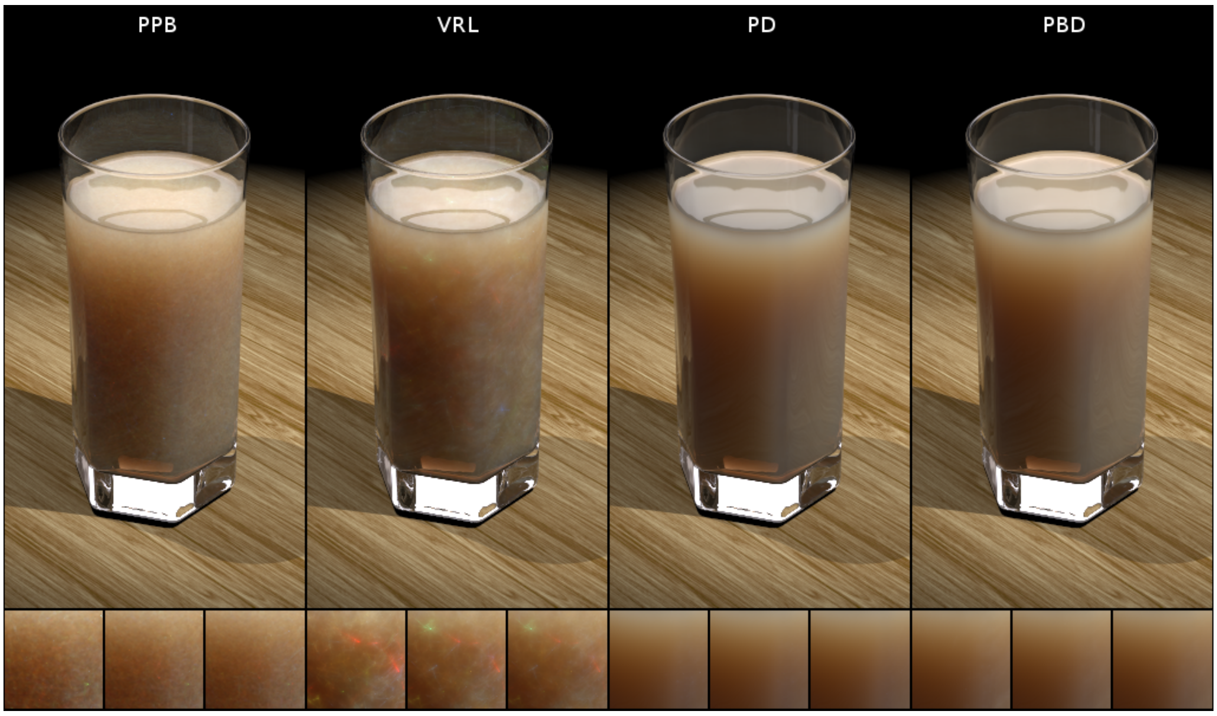


Figure 5.6: *Comparison for* $m = 0.3$, $\alpha = 0.91$, $T = 30$ *min.*

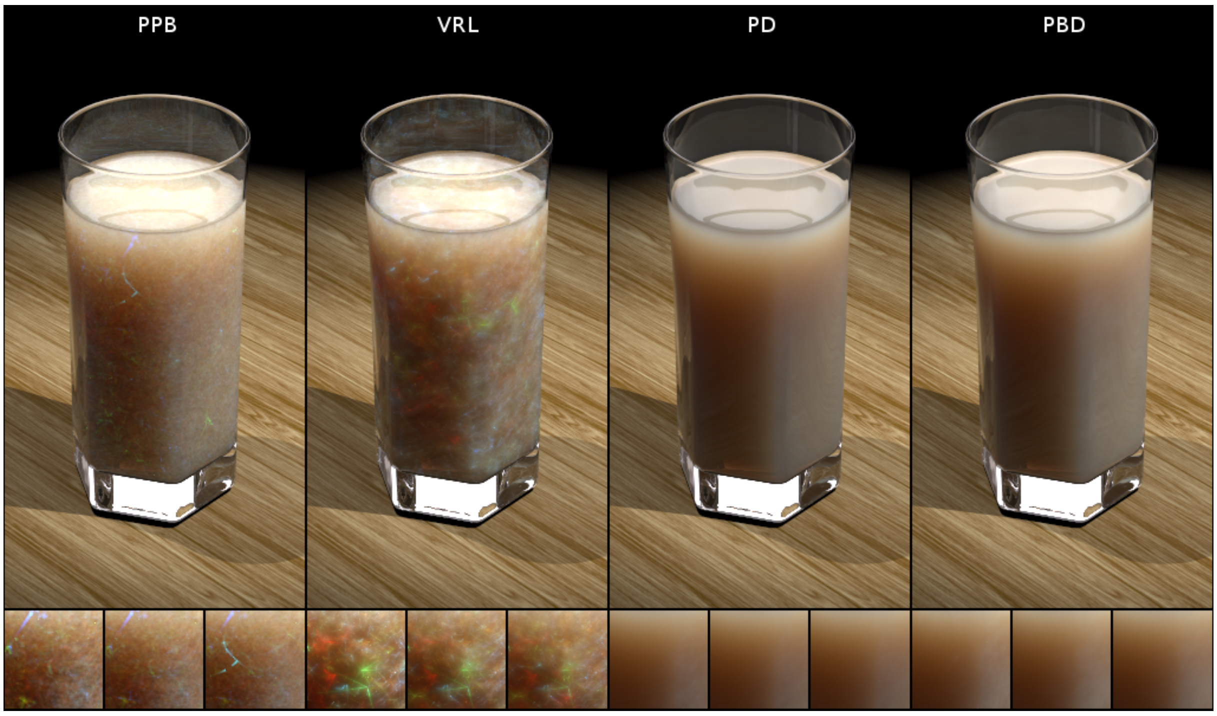


Figure 5.7: *Comparison for* $m = 0.5$, $\alpha = 0.96$, $T = 35$ *min.*

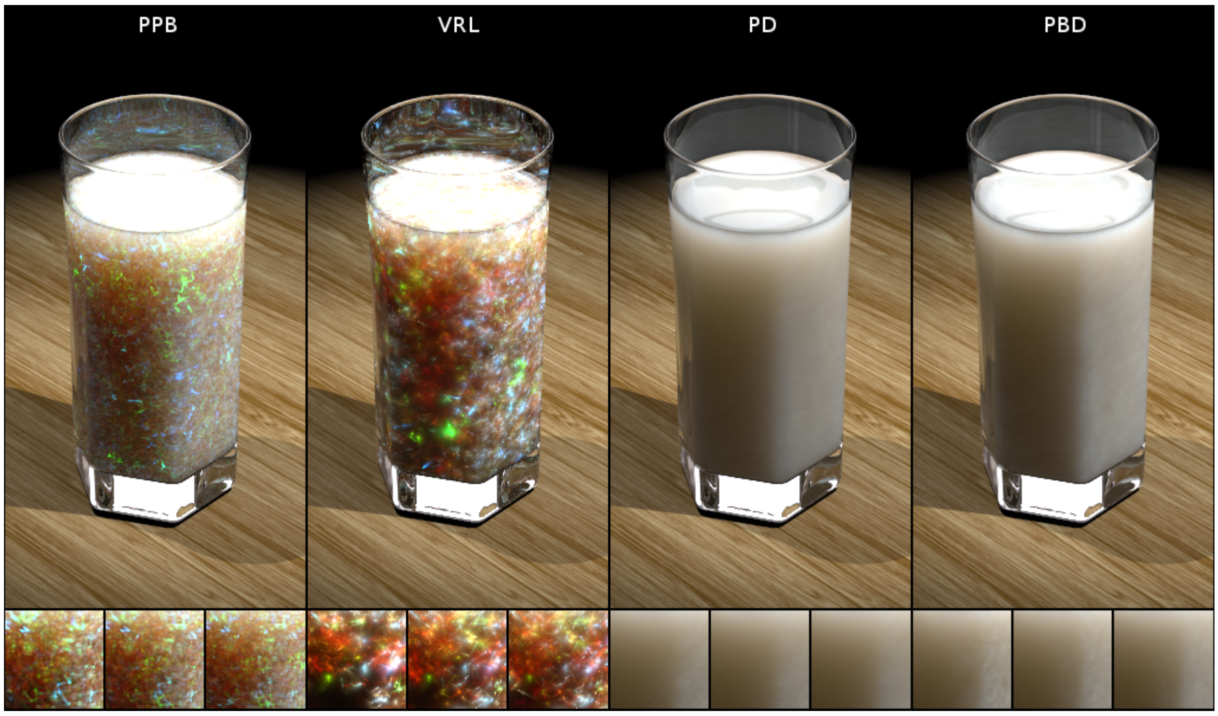
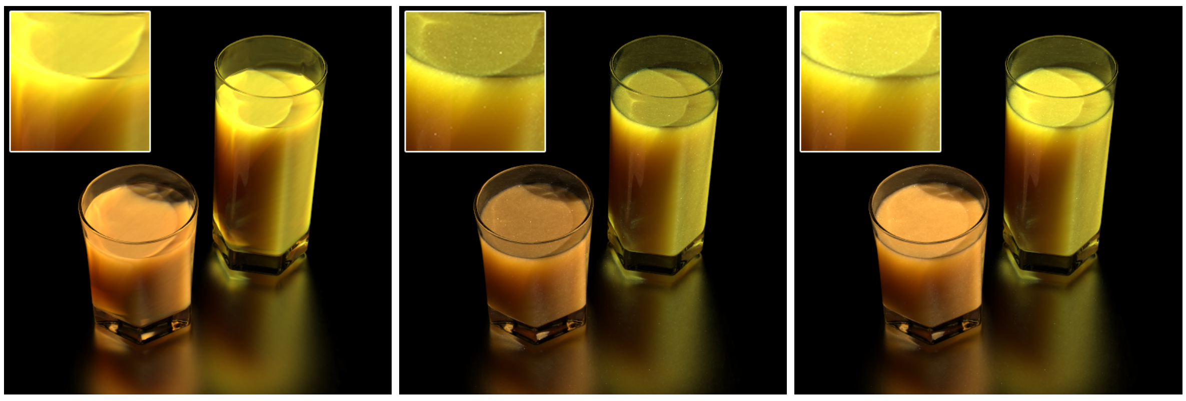


Figure 5.8: *Comparison for $m = 1.0$ (milk only), $\alpha = 0.9987$, $T = 40$ min.*

5.2 Single Scattering



(a) PPB *(b) PBD (original)* *(c) PBD (modified)*

Figure 5.9: *Comparison between single scattering from PPB and two variants of PBD.*

In addition to multiple scattering, we also compare the single scattering from PPB and PBD. We have found that the original diffuse single scattering term (3.39) proposed by Habel et al. [HCJ13b] is incomplete in the presented form, the main culprit being that the reflected radiance is independent of the viewing direction. Also, in the original form, the single scattering response assumes a rough dielectric model (diffuse), but does not derive this in a concise way. Therefore we modified the diffuse single scattering term to be more in line with the multiple scattering term used in PBD, leading to the following equations, replacing (3.39) and (3.40):

$$R^{(1)}(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b, t) = \frac{p(\vec{\omega}_b \to \vec{\omega}_{xtx})\, T_r(d_{xtx})\, \cos\theta}{d^2_{xtx}} \tag{5.1}$$

$$L_r(\mathbf{x} \to \vec{\omega}) \approx \frac{1}{\pi}\, F_t(\mathbf{x}, \vec{\omega}, \eta) \sum_b \Phi_b R^{(1)}(\mathbf{x}, \mathbf{x}_b, \vec{\omega}_b) \tag{5.2}$$

We removed the Fresnel term in (3.39), accounting for the amount of light escaping the medium, and added a Fresnel reshaping term in (3.40), which accounts for the amount of diffuse light *seen* from the viewing direction, analogously to multiple scattering.

We have rendered the FRUITJUICE scene showing two liquids (isotropic media) with single scattering at a resolution of 512 $\times$ 512 pixels for 2 hours each, on the same machine that was used for the multiple scattering comparison. The results are shown in Figure 5.9. We first note differences in the sharpness and intensity of the single scattering solutions between PPB and the two variants of PBD, which is expected due to the PBD methods computing a diffuse response whereas PPB does not. The main difference between the original and the modified PBD method is in the intensity, most visible at the top surface of the liquid. This comes from the high relative index of refraction between liquid and air, leading to reduced contribution from the photon beams due to a narrow critical angle (total internal reflection). In the modified version, the photon beams contribute to the diffuse light along their full length.

In addition, we also note singularities in the PBD solutions due to the inverse squared distance term in the Green's function. To mitigate these singularities, we could clamp the distances, but

Method	Total	Profile	Profile Speedup	Total Speedup
`std::exp` (scalar)	945 ms	910 ms	0.0 %	0.0 %
`fmath::exp` (scalar)	845 ms	810 ms	12.3 %	11.8 %
`fmath::exp` (SSE)	540 ms	505 ms	80.2 %	75.0 %

Table 5.2: *Improved performance gained from `fmath` and SSE optimized versions, with `std::exp` being the baseline. We show the total time per pass, time for profile evaluation per pass, speedup in profile evaluation and total speedup.*

this inevitably leads to lost energy. As an alternative, we discussed replacing the point sources with spherical sources, allowing for better distribution of the energy in the near surface region. Initial tests of this approach have shown promise and further investigation seems worthwhile, especially in combination with a proper derivation of the diffuse single scattering term (3.39).

5.3 Optimized Profile Evaluation

To remain clarity in our implementation of the rendering methods, we have generally left out most of the optimization techniques proposed in the original papers, with one exception worth mentioning. During development of the diffusion methods, we suffered from slow performance in the profile evaluation, which made the development process very cumbersome. The main culprit was the slow execution of the exponential function. Due to the fact that most computations in the profile evaluation are performed on all three components of the spectrum (color), a vectorized implementation using SSE intrinsics seemed worthwhile, as we can then perform 4 single precision floating point operations simultaneously. Both our floating point and SSE optimized versions use the optimized exponential function provided by *fmath* [Mit09]. We have measured the performance enhancements in a modified FRUITJUICE scene (one glass only, no floor), rendering with PD (dipole and quadpole only) at a resolution of 256×256 pixels on a MacBook Pro running an Intel Core i7 @ 2.6 GHz. We obtain a total speedup of 75.0 % from the SSE optimized version compared to the baseline using `std::exp` and scalar floating point code. We summarize the measurements in Table 5.2.

6 Conclusion and Future Work

In this thesis, we studied different state of the art rendering methods to compute images with participating media. We summarized the mathematical background, provided an introduction to traditional rendering methods for participating media as well as a comprehensive overview of recent rendering methods based on the photon beam approach. Starting from an existing code base, we developed a cross-platform renderer implementing the presented methods in a clean and cohesive way. We compared the rendering methods side by side in various settings, based on images computed by our new renderer, and discussed their advantages and disadvantages. We hope that our renderer will be useful for future research, either as a starting point for experimentation with existing rendering methods, or as a framework for implementing new methods.

There is of course much room for improvements and future work, regarding both our renderer as well as the rendering methods in general. Our renderer provides clean reference implementations of the presented methods, which was the primary goal for this thesis, but it can be improved in various areas. This includes both improvements of the APIs exposed to the rendering methods, as well as extensions to the basic feature set of the renderer, allowing the methods to be tested in more diverse situations. The following are a few possible ideas:

- Support for heterogeneous media
- Generic sampling of media and phase functions
- Improved sampling strategies (quasi-random sequences)
- Additional BSDFs for surface shading
- Additional types of light sources (area lights, image based lighting)

Concerning the rendering methods, we mostly discussed possible improvements for the photon beam diffusion method, which primarily suffers from singularities in our current implementation. The techniques proposed by Habel et al. [HCJ13b] to mitigate these singularities only address the case of non-oblique illumination and are hard to adapt for oblique illumination. A promising approach for future work is the extension of the point sources to spherical sources in both single and multiple scattering, which would naturally diminish the singularities without additional smoothing of the reflectance profiles. Our renderer provides a good framework for experimentation with such ideas.

Bibliography

[Aro95] Raphael Aronson. Boundary conditions for diffusion of light. *Journal of the Optical Society of America A*, 12(11):2532–2539, 1995.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Cha60] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover, 1960.

[d'E12] Eugene d'Eon. A better dipole. Technical report, http://www.eugenedeon.com, 2012.

[DI11] Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 30(4):56:1–56:14, 2011.

[DJ07] Craig Donner and Henrik Wann Jensen. Rendering translucent materials using photon diffusion. *Rendering Techniques (Proc. Eurographics Symposium on Rendering)*, pages 243–252, 2007.

[FPW92] Thomas J. Farrell, Micheal S. Patterson, and Brian Wilson. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. *Medical Physics*, 19(4):879–888, 1992.

[Gee10] Marcus Geelnard. tinythread, 2010. http://tinythreadpp.bitsnbites.eu.

[Gro56] C. C. Grosjean. A high accuracy approximation for solving multiple scattering problems in infinite homogeneous media. *Il Nuovo Cimento (1955-1965)*, 3:1262–1275, 1956.

[Gro58] C. C. Grosjean. Multiple isotropic scattering in convex homogeneous media bounded by vacuum. In *Proc. Second International Conference on the Peaceful Uses of Atomic Energy*, volume 413, 1958.

[HCJ13a] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Classical and improved diffusion theory for subsurface scattering. Technical report, Disney Research Zürich, 2013.

[HCJ13b] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 32(4), 2013.

[HG41] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the Galaxy. *Astrophysical Journal*, 93:70–83, 1941.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 28(5), 2009.

[HMK$^+$00] Bill Hoffman, Ken Martin, Brad King, Dave Cole, and Alexander Neundorf. CMake, 2000. http://www.cmake.org.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 27(5), 2008.

[Ige99] Homan Igehy. Tracing ray differentials. In *SIGGRAPH*, SIGGRAPH '99. ACM, 1999.

[JB02] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *Computer Graphics (Proc. SIGGRAPH)*, 21(3):576–581, 2002.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. *Computer Graphics (Proc. SIGGRAPH)*, pages 311–320, 1998.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. *Rendering Techniques (Proc. Eurographics Workshop on Rendering)*, pages 21–30, 1996.

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. *Computer Graphics (Proc. SIGGRAPH)*, 35:511–518, 2001.

[JNSJ11] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics*, 30(1):5:1–5:19, 2011.

[JNT$^+$11] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 30(6):181:1–181:12, 2011.

[JZJ08] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. Eurographics)*, 27(2):557–566, 2008.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)*, 20:143–150, 1986.

[Kel97] Alexander Keller. Instant radiosity. *Computer Graphics (Proc. SIGGRAPH)*, pages 49–56, 1997.

[KF12] Christopher Kulla and Marcos Fajardo. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 31(4):1519–1528, 2012.

[KP97] Alwin Kienle and Michael S. Patterson. Improved solutions of the steady-state and the time-resolved diffusion equations for reflectance from a semi-infinite turbid medium. *Journal of the Optical Society of America A*, 14(1):246–254, 1997.

[KZ11] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics*, 30(3):25:1–25:13, 2011.

[Mag00] Industrial Light & Magic. OpenEXR, 2000. http://www.openexr.com.

[Mit09] Shigeo Mitsunari. fmath, 2009. http://homepage1.nifty.com/herumi/soft/fmath.html.

[NGD$^+$06] Srinivasa G. Narasimhan, Mohit Gupta, Craig Donner, Ravi Ramamoorthi, Shree K. Nayar, and Henrik Wann Jensen. Acquiring scattering properties of participating media by dilution. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 1003–1012, 2006.

[NNDJ12] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 31(4):60:1–60:11, 2012.

[NRH$^+$77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. *National Bureau of Standards*, 1977.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[RSK08] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–606. Springer, 2008.

[SP94] Francois X. Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[Sta95] Jos Stam. Multiple scattering as a diffusion process. *Rendering Techniques (Proc.*

Eurographics Workshop on Rendering), pages 41–50, 1995.

[Van05] Lode Vandevenne. LodePNG, 2005. http://lodev.org/lodepng.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. *Computer Graphics (Proc. SIGGRAPH)*, (29):419–428, 1995.

[vH97] Dimitri van Heesch. Doxygen, 1997. http://www.stack.nl/ dimitri/doxygen.

[WABG06] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), 2006.

[WFA$^+$05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3), 2005.

[Yip11] Milo Yip. rapidjson, 2011. https://code.google.com/p/rapidjson.