Bournemouth University

FACULTY OF MEDIA & COMMUNICATION

MSc Computer

Animation & Visual

Effects

September 2023

Volume Rendering: Enhancing Visual Realism through Path Tracing

by

Gabriela Cintra de Freitas Almeida

Abstract

This research critically examines the application of path tracing in rendering, with an emphasis on volume rendering. The study presents the development of a path tracer renderer, specifically designed to visualise volumes. A notable aspect of this renderer is its ability to factor in volume transmittance when casting rays trough the scene, ensuring a consistent representation of reality. This implementation is particularly relevant for the film and animation industries, offering a technical approach to enhance visual outputs. By drawing a clear connection between theoretical knowledge and its practical implementation, this work contributes to the broader understanding of current rendering techniques.

Keywords: Path Tracing, Volume Rendering, Transmittance, Computer Graphics.

Acknowledgements

This work greatly benefitted from the insights and expertise of Professors Jon Macey, Ian Stephenson, Jian Chang, and Jian Jun Zhang. My heartfelt gratitude also goes to my family and friends, with special appreciation for my father and mother, who have been my pillars of support throughout this journey.

Contents

List of Figures

1 Introduction

Within the film and animation industry, visual storytelling operates at the intersection of narrative coherence and technological sophistication. Historically, filmmakers and animators have employed a diverse range of rendering techniques. Each method, while offering distinct advantages, also presented its unique challenges. The field of visual rendering, however, is constantly evolving and changing. It continually evolves, responding to the demands of audiences who seek experiences that resonate with realism and immersion.

The digital revolution marked an important moment in the history of the film and animation industry. While narrative themes and character development retained their central roles, there was a noticeable shift in the technological methodologies employed. Among the emerging techniques, path tracing began to gain significant academic and industry traction.

From a computational perspective, path tracing models the interactions between light and objects, producing images that align closely with real-world visual experiences. Unlike many of its predecessors, which often relied on computational shortcuts or subjective artistic interpretations, path tracing emphasises the accurate simulation of light's physical behaviour. The rise of this technique can be attributed to two primary factors: the evolution of computational algorithms tailored for this purpose and the exponential growth in hardware capabilities.

Nevertheless, the industry's push into path tracing represents merely the initial phase of a broader exploration. A particular extension of this method is its application in volume rendering. Unlike traditional rendering techniques that primarily focus on external surfaces, volume rendering is concerned with visualising data that exists in a three-dimensional continuum. This approach is especially pertinent in specialised sectors such as medical imaging, wherein the detailed representation of internal structures, for instance, within the human anatomy, is of paramount importance.

The integration of path tracing with volume rendering is not merely an incremental advancement; it signifies a paradigmatic shift, offering levels of depth and a comprehensive three-dimensional representation. However, the fusion of these methodologies is not without its complexities. Achieving seamless integration necessitates an understanding of both domains and demands innovative solutions to address inherent challenges.

This thesis is structured to provide a comprehensive analysis of the development and refinement of a path tracer designed explicitly for volume rendering applications. In doing so, it aims to bridge theoretical foundations with their pragmatic implementations.

In the following chapters, readers will explore the historical evolution, technical foundations, and wider implications of these techniques. The aim is to provide the reader with a comprehensive understanding of the topic, including its relevance and potential applications.

2 Background

The evolution of computer graphics techniques, specifically volume rendering and path tracing, has impacted visual representation. This chapter will cover these domains' historical and contemporary advancements, emphasising their convergence.

The mid-1980s brought innovative contributions to computer graphics. Kajiya (1986) was important in this era, unveiling the rendering equation, an integral equation that encapsulated many known rendering algorithms. This foundation paved the way for the Monte Carlo solution and introduced Hierarchical sampling, a variance reduction technique.

Parallel to Kajiya's work, Cook (1984) championed ray tracing as an elegant technique in computer graphics. His research underscored ray tracing's capabilities in simulating shadows, reflections, and refracted light. By innovatively distributing ray directions based on their sampling analytic function, Cook enabled ray tracing to embrace fuzzy phenomena, resolving challenges related to motion blur, depth of field, and fuzzy reflections.

Complementing these developments was Cohen's (1986) exploration of the radiosity method. This approach modelled light interactions between diffusely reflecting surfaces, offering predictions of global illumination effects with precision. Cohen's emphasis on environment sampling underscored the importance of detailed meshes, especially in high gradient intensity zones. His work was a cornerstone in predicting global illumination without exorbitant computational overhead.

Global illumination techniques gained even more attention in the 1990s. Veach (1995) shifted the narrative towards Monte Carlo methods, positioning them as viable alternatives to finite-element techniques. These methods, adaptable to varied scene descriptions, eliminated the complicated meshing process. Veach's (1998) subsequent contributions delved deeper into light transport algorithms, emphasising their capability to simulate light interactions in diverse settings, crafting realistic images.

In parallel to those advances, Levoy (1987) propelled volume rendering techniques to attention, emphasising the potential of representing surfaces from three-dimensional scalar functions. This technique highlighted direct shading and projection of each sample, eliminating the need for geometric primitive fittings. This approach was further enhanced by Drebin (1988), who focused on rendering images of mixed material volumes, amplifying the intricacies of both material interiors and their boundaries.

Heidrich *et al.* (1995) introduced an innovative perspective on maximum projection, a volume rendering technique. By implementing simple affine transformations, this approach facilitated interactive manipulations of volume data, allowing for an enhanced balance between interactivity and image quality.

The latter part of the decade saw Engel (2006) presenting a comprehensive tutorial on real-time volume rendering techniques optimised for consumer graphics hardware. This tutorial encapsulated techniques to harness the power of modern graphics hardware and high-level shading languages, ensuring real-time rendering of volumetric data and effects.

The integration of volume rendering with path tracing gained momentum with Fong *et al.* (2017) acknowledgement of path tracing's supremacy in movie production. Emphasising the role of evolving computational power and refined techniques, Fong highlighted how volume rendering could leverage path tracing's evolution, crafting photoreal images.

In conclusion, the synthesis of foundational works from pioneers like Kajiya and Cook, coupled with contemporary innovations, underlines the progress in volume rendering and path tracing. Their integration has redefined the boundaries of realism in visualisations and set a promising path for the future of computer graphics.

3 Methodology

Overview

In this chapter, it is explored the techniques used to integrate volume rendering with path tracers. Through empirical methods, the aim is to provide a comprehensive understanding of how to achieve photorealistic images using rendering methods. The methodology used is based on research findings, ensuring a thorough exploration of the topic. Readers can expect clear explanations of the techniques used in this research.

3.1 Global Illumination

According to Jensen (2001), global illumination is the term used to describe the physics-based simulation of light scattering in a synthetic model. This technique aims to replicate all light reflections in a model and accurately estimate light intensity at any point in the model. To simulate global illumination, it is required to describe the geometry, materials, and light sources presented in the scene. The global illumination algorithm then calculates the interaction between the light leaving the light sources and the described ambient.

3.1.1 Nature of Lights

Global illumination algorithms imitate how light behaves. Therefore, it is necessary to understand its nature thoroughly. Throughout history, knowledge of light has evolved through discoveries and theories. One enduring model is the ***ray optics model***, which portrays light as a series of independent rays following geometric rules for reflection, refraction, and image formation. In computer graphics, ray optics is widely used due to its effectiveness in representing and manipulating light in a computational environment.

It is essential to know that the ray optics model oversimplifies how photons disperse and does not consider complex phenomena like diffraction and interference. It assumes that light travels infinitely fast, which may seem problematic, but it can still simulate most visible light phenomena. Despite its limitations, the emphasis on ray optics in computer graphics shows how well it can capture how light interacts with its surroundings, often enough to make things look realistic and immersive.

3.1.2 Radiometry

The branch of optical physics that deals with measuring light and energy is called Radiometry. The basic unit of light is called a *photon,* and the energy of a photon $(e\lambda)$ depends on its *wavelength* (λ). *Spectral radiant energy* $(Q\lambda)$ is the total energy of a group of photons $(n\lambda)$ with a specific wavelength (λ). To find the total energy of a collection of photons, also called *radiant energy* (Q), you can integrate $(Q\lambda)$ over all possible wavelengths. Radiant flux (Φ) is the rate at which radiant energy flows over time in a direction. This is also known as "flux".

The *radiant flux area density* is defined as $d\Phi/dA$ (differential flux per differential area (at a surface)). This measurement is divided: *radiosity* (B), which measures the flux living from a surface, and irradiance (E), which measures the flux arriving at a particular point on a surface (x). This relationship is expressed mathematically as:

$$\mathrm{E}(x) = \frac{\mathrm{d}\Phi}{dA} \tag{3.1}$$

Radiance (L) is an essential measurement in global illumination and is often represented as $L(x,\omega)$. The x denotes position, and the ω indicates the direction. Radiance is the amount of light energy per unit of solid angle per projected area and captures the inherent colour of an object. It is widely used in ray tracing algorithms, which highlights its importance in computer graphics.

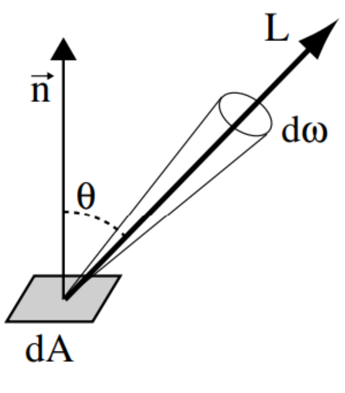

Figure 3.1 - Radiance, L, is defined as the radiant flux per unit solid angle, and $d\vec{\omega}$, per unit projected area, dA. (Jensen 2001)

3.1.3 Local Ilumination

The simulation of light scattering across different surface materials or mediums is a crucial aspect of computer graphics. This complex process is commonly referred to as local illumination.

- *Bidirectional Reflectance Distribution Function (BRDF)*

Nicodemus et al. (1977*)* introduced the Bidirectional Reflectance Distribution Function (BRDF) as a way to understand how light reflects off surfaces. Essentially, BRDF assumes that the light reflects at the exact point it hits a surface. This model, pivotal in explaining local illumination, provides the means to calculate the reflected light in all directions when the incoming radiance at a surface location is known.

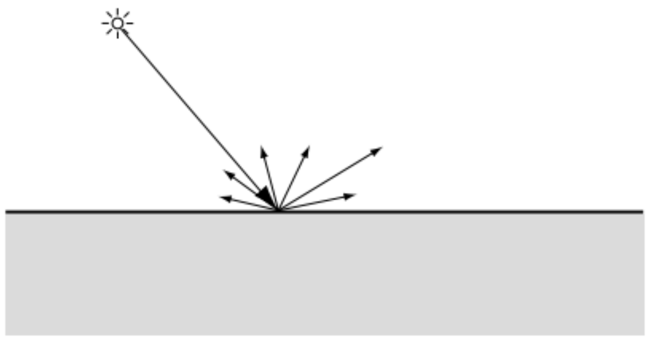

Figure 3.2 - The BRDF is a model that explains surface reflection of a light. It assumes that all light is reflected from the location where it hits a surface. (Jensen 2001)

The BRDF, symbolised by f_r, establishes the relationship between reflected radiance and irradiance as expressed in equation (3.2). It is defined by the following formula:

$$\mathrm{f_r}(\mathrm{x}, \vec{\omega}', \vec{\omega}) = \frac{\mathrm{dL_r}(\mathrm{x}, \vec{\omega})}{\mathrm{dE_i}(\mathrm{x}, \vec{\omega}')} = \frac{\mathrm{dL_r}(\mathrm{x}, \vec{\omega})}{\mathrm{L}_i(\mathrm{x}, \vec{\omega}')(\vec{\omega}' \cdot \vec{\mathrm{n}})\overrightarrow{\mathrm{d}\omega'}} \tag{3.2}$$

where $\vec{n}$ is the normal at x.

What BRDF illustrates is the idea of local illumination. Once the incoming radiance at a surface point is recognised, the amount of light reflected in every direction can be determined by integrating the incident radiance, L_i, as demonstrated in equation (3.3):

$$L_r(x,\vec{\omega}) = \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega})dE(x,\vec{\omega}') = \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega})Li(x,\vec{\omega}')(\vec{\omega}' \cdot \vec{n})d\overrightarrow{\omega'} \qquad (3.3)$$

Here Ω represents the hemisphere of incoming directions at point x, and θ is the angle between the surface normal and the light source direction, $(\vec{\omega}' \cdot \vec{n}) = cos\,\theta'$.

Jensen (2001) stated that the BRDF has an essential property called Helmholtz's law of reciprocity, which means that the BRDF does not depend on the direction. This is a crucial feature that most global illumination algorithms use to track light paths in both directions. It also allows for easy validation of the BRDF by checking if it is reciprocal. Another essential consideration in evaluating BRDF is the principle of energy conservation, meaning that a surface cannot reflect more light than it absorbs.

The measure of the light that a surface reflects in comparison to the light that strikes it is known as the reflectance (ρ) of the surface. It is defined in equation (3.4) as:

$$\rho(x) = \frac{d\Phi_r(x)}{d\Phi_i(x)} \qquad (3.4)$$

In this context, $\rho(x)$ represents the fraction of incident light that is reflected, while the remainder is absorbed or transmitted. In physically-based rendering, this value is expected to be within the range of zero to one.

- *Diffuse Reflection*

When light hits a surface with diffuse reflection, it reflects in all directions. This type of reflection usually happens on rough surfaces or materials with subsurface scattering. Figure 3.3 (a) shows an example of light reflecting in random directions.

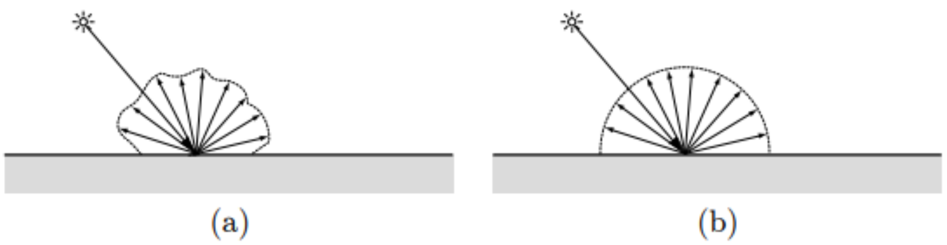

Figure 3.3 - Light reflects in all directions when it hits diffuse materials. The two types of diffuse

reflection are general diffuse reflection (a) and Lambertian reflection (b) (Jensen 2001).

Lambertian or ideal diffuse reflection is a specific type of diffuse reflection where the reflected direction is completely random (Figure 3.3 (b)). Jensen (2001) states that this results in a constant reflected radiance in all directions, regardless of the irradiance, which leads to a constant BRDF ($f_{r,d}$):

$$L_r(x,\vec{\omega}) = f_{r,d}(x)\int_{\Omega} dE_i(x,\vec{\omega}') = f_{r,d}(x)E_i(x) \tag{3.5}$$

Considering the above equation and that $\int_{\Omega}(\vec{n}\cdot\vec{\omega})d\vec{\omega} = \pi$, it is then possible to find the diffuse reflectance ρ_d for Lambertian surface:

$$\rho_d(x) = \frac{d\Phi_r(x)}{d\Phi_i(x)} = \frac{L_r(x)dA\int_{\Omega}(\vec{n}\cdot\vec{\omega})d\vec{\omega}}{E_i(x)dA} = \pi f_{r,d}(x) \tag{3.6}$$

For a Lambertian surface, the reflected light's direction is completely random, as previously stated. To determine the cosine-weighted reflected direction ($\overrightarrow{\omega_d}$), two randomly distributed numbers can be used ($\xi 1 \in [0,1]$ and $\xi 2 \in [0,1]$) and the following equation:

$$\overrightarrow{\omega_d} = (\theta,\phi) = \left(\cos^{-1}\left(\sqrt{\xi_1}\right), 2\pi\xi_2\right) \tag{3.7}$$

This equation uses spherical coordinates (θ,φ) to describe the direction, with θ representing the angle with the surface normal and φ representing the rotation around the normal.

- *Specular Reflection*

Light hitting a smooth surface, like a shiny metal or glass, can cause a specular reflection. However, due to imperfections, most surfaces are not perfectly smooth and will reflect light in a cone shape around the mirror direction (Figure 3.4(a)). The level of imperfection, such as roughness and gloss, is often considered in reflection models for these glossy surfaces. In contrast, when the surface is perfectly smooth, it creates a perfect specular reflection where light is only reflected in the mirror direction (Figure 3.4 (b)).

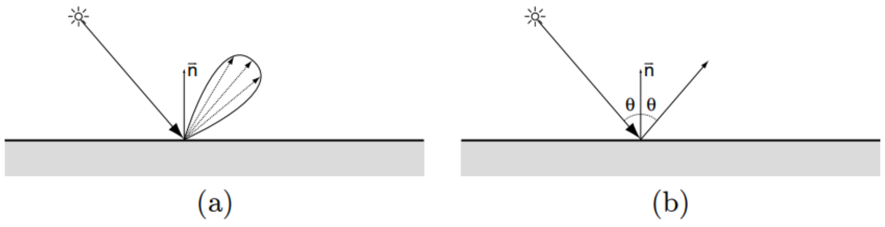

Figure 3.4 - When light hits a specular surface, it reflects in the same direction as a mirror. (a) displays a glossy reflection, while (b) features a perfect specular reflection. (Jensen 2001)

The radiance reflected due to a specular reflection is represented as:

$$L_r(x, \overrightarrow{\omega_s}) = \rho_s(x) L_i(x, \vec{\omega}') \quad (2.25) \tag{3.8}$$

For an ideal specular reflection, the direction of the mirror denoted as $\overrightarrow{\omega_s}$, is defined by:

$$\overrightarrow{\omega_s} = 2(\vec{\omega}' \cdot \vec{n})\vec{n} - \vec{\omega}' \quad (2.26) \tag{3.9}$$

It is important to note that both $\overrightarrow{\omega_s}$ and $\vec{\omega}'$ point away from the surface.

The BRDF for perfect mirror reflection, when using spherical coordinates for direction, is described as:

$$f_{r,s}(x, \vec{\omega}', \vec{\omega}) = 2\rho_s \delta(\sin^2\theta' - \sin^2\theta)\delta(\phi' - \phi \pm \pi) \tag{3.10}$$

- *Bidirectional Scattering Distribution Function (BSDF)*

With a broader perspective, the Bidirectional Scattering Distribution Function, or BSDF, emerges as a holistic model enclosing both reflection and transmission of light at a surface. While BRDF delves into the reflection of light on surfaces, BSDF offers a more comprehensive view, accounting for both the scattering and transmission of light when it encounters a material. This concept is essential when it comes to volumes. The BRDF can be seen as a specialised version of BSDF, narrowing its focus exclusively to reflection.

3.1.4 The Rendering Equation

Jensen (2001) states that the rendering equation, introduced by Kajiya (1986), serves as the mathematical basis for all global illumination methodologies. This equation states the necessary conditions for light transport to be balanced in models. To determine the outgoing radiance at any surface location in a model, the rendering equation can be utilised. This involves calculating the sum of the emitted radiance, L_e, and the reflected radiance, L_r, which together form the outgoing radiance, L_o.

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + L_r(x,\vec{\omega}) \tag{3.11}$$

Using Equation 3.3, it is possible to calculate the reflected radiance:

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega})L_i(x,\vec{\omega}')(\vec{\omega}' \cdot \vec{n})d\vec{\omega}' \tag{3.12}$$

This equation integrates over all possible incoming light directions ($\vec{\omega}$) and calculates the outgoing radiance based on the surface's properties and the incoming light.

For finite element algorithms, the rendering equation is expressed as an integral over surface locations. By utilising the differential solid angle, it is possible to express de equations in terms of surface locations and normals.

$$d\vec{\omega}'(x) = \frac{(\vec{\omega}' \cdot \vec{n}')dA'}{||x' - x||^2} \tag{3.13}$$

It is possible to modify the rendering equation to incorporate the visibility between surfaces and light transport by introducing G. This term accounts for the geometric relationship between two surface points (x and x') and their normal ($\vec{n}$ and $\vec{n}'$),.

$$G(x,x') = \frac{(\vec{\omega}' \cdot \vec{n}')(\vec{\omega}' \cdot \vec{n}')}{||x' - x||^2} \tag{3.14}$$

$$\begin{aligned} L_o(x,\vec{\omega}) &= L_e(x,\vec{\omega}) \\ &\quad + \int_S f_r(x,x' \to x,\vec{\omega})L_i(x' \to x)V(x,x')G(x,x')dA' \end{aligned} \tag{3.15}$$

In this equation:

- $L_i(x' \to x)$ is the radiance leaving x' in the direction towards x;

- S is the set of all surface points;
- $V(x, x')$ is the following visibility function:

$$V(x, x') = \begin{cases} 1, & x \text{ and } x' \text{ are mutually visible} \\ 0, & \text{otherwise} \end{cases} \tag{3.16}$$

It is then possible to formulate the rendering equation entirely in terms of surface locations x, x', and x'':

$$\begin{aligned} L_o(x' \to x) = {} & L_e(x' \to x) \\ & + \int_S f_r(x'' \to x' \to x) L_i(x'' \to x') V(x', x'') G(x', x'') dA'' \end{aligned} \tag{3.17}$$

The equation considers how light is emitted or reflected from point x'' reaches point x' and contributes to the radiance at point x.

<u>3.1.5 Monte Carlo Integration</u>

Equation 3.17 is similar to the original rendering equation that Kajiya (1986) presented in his seminal paper, where he showed how Monte Carlo methods solve the same equation. Monte Carlo integration involves randomly sampling a function to evaluate its properties. To integrate a function $f(x)$ over a one-dimensional domain from a to b:

$$I = \int_a^b f(x) dx \tag{3.18}$$

The mean value of $f(x)$ is calculated over the interval a to b and multiplied by the length of the interval $(b - a)$. This mean is obtained by averaging the values of $f(x)$ at N locations $\xi_1, \xi_2, \ldots, \xi_N$, where $\xi_1, \ldots, \xi_N$ are random numbers uniformly distributed between a and b.

$$I_m = (b - a) \frac{1}{N} \sum_{i=1}^{N} f(\xi_i) \tag{3.19}$$

I_m is the Monte Carlo estimate of the integral. As the number of samples (N) increases, this estimate becomes more accurate. In the limit, it is found that:

$$\lim_{N\to\infty} I_m = I \tag{3.20}$$

To understand how fast the estimator I_m converge towards the correct result I it is computed the variance σ^2 of the estimate I_m:

$$\sigma^2 = \frac{1}{N}\left(\int_a^b f^2(x)dx - I^2\right) \tag{3.21}$$

According to Jensen (2001), the Monte Carlo integration is easily applied to most problems, but convergence is slow. However, for high-dimensional integrals (such as those in rendering), better convergence is often achieved than with any other method.

The variance σ_s^2 of the sampling distribution can also be estimated:

$$\sigma_s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(f(\xi_i) - I_m)^2 \tag{3.22}$$

Since this calculation contains the factor $1/(N - 1)$, the variance (the noise) of samples converges as slowly as the variance of the estimate (Jensen (2001)). Fortunately, several variance-reduction techniques are available.

Variance-Reduction Technique – Multiple Importance Sampling

The Monte Carlo integration benefits from techniques that reduce errors and improve results. One such technique is Multiple Importance Sampling (MIS).

MIS combines several sampling methods to evaluate the same rendering equation. Not all methods work equally well, so MIS uses multiple methods and gives them weights based on their effectiveness. This approach, introduced by Veach (1998), ensures that the sampling is balanced and optimised.

The MIS is an extension of the Importance Sampling technique. At its core, Importance Sampling aims to reduce variance in Monte Carlo estimations by picking samples not uniformly but according to a given probability distribution that reflects the importance or relevance to the problem at hand. In simple terms, it tries to sample more frequently where the function has higher values, thus producing a better average with fewer samples.

With this foundation, Multiple Importance Sampling (MIS) emerges as a natural evolution. Instead of relying on a single importance sampling strategy, MIS recommends the simultaneous use of multiple strategies. By doing so, it reduces the overall variance in the estimation.

According to *Pharr et al.* (2016), The nature of MIS lies in the computation of a weighted average of the estimations from each sampling method. The weighting mechanism is critical to the success of MIS. A common approach employed is the *balance heuristic*, which calculates the weight based on the number of samples and the probability density function (PDF) of each method.

Mathematically, the weight for a sample from a given method is computed as follows:

$$\omega_i(x) = \frac{n_i p_i(x)}{\sum_{j=1}^{k} n_j p_j(x)} \tag{3.23}$$

- $\omega_i(x)$: This represents the weight of the i^{th} method's sample for the function value at point x. It is this weight that determines the contribution of the sample from the i^{th} method to the final estimate.

- n_i: Represents the number of samples taken from the i^{th} method. In the context of MIS, it is crucial to keep track of how many samples are drawn from each method, as this directly influences the weight calculation.

- $p_i(x)$: This is the probability density function (PDF) of the i^{th} method evaluated at point x. The PDF gives us a measure of how likely a sample from the i^{th} method is to land at point x. A well-crafted PDF will have higher values where the function is significant, guiding the sampling process more effectively.

- $\sum_{j=1}^{k} n_j p_j(x)$: This is the summation of all the products of the number of samples and their corresponding PDFs from all k methods at point x. It acts as a normalisation factor, ensuring that the weights are distributed appropriately among all methods.

The adaptability of MIS ensures accurate and efficient integral estimation. In practical applications like rendering, MIS balances different contributions for accurate results with reduced costs.

3.1.6 Monte Carlo Ray Tracing

Ray tracing is a method used to estimate the interaction of light with a model by tracing a path of light through it and determining the resulting radiance. This technique is widely employed in computer graphics, particularly when rendering shadows and specular surfaces. Introduced by *Whitted (1980)*, the recursive ray-tracing algorithm is both straightforward and effective, simulating light from the viewer's perspective all the way back to its source.

For effective ray tracing, certain data are essential: the observer's position, an image plane (which includes its viewing direction and field of view), a comprehensive description of the scene's geometry, and details about the light sources and materials. The primary objective is to determine the colour of each pixel on the image plane. This process involves sending one or more rays through each pixel and averaging the radiance they capture. The direct rays sent from the observer and through the pixels are termed primary rays.

To ascertain the radiance of a primary ray, one must pinpoint the closest object it intersects. At this intersection point, denoted as x, the goal becomes deducing the emitted radiance in the ray's direction. This requires knowledge of the surface normal, $\vec{n}$, at x, as well as the Bidirectional Reflectance Distribution Function (BRDF), f_r. With these details in hand, the illumination from each light source can be computed by estimating the irradiance at x.

Ray tracing is adept at rendering reflections on shiny surfaces by tracing rays in the mirror direction, $\overrightarrow{\omega_s}$. However, it faces challenges in calculating indirect illumination on diffuse surfaces. Consequently, according to Jensen (2001), ray tracing does not serve as a comprehensive global illumination algorithm.

3.6.1 - Path Tracing

Path tracing builds on the idea of ray tracing, making it possible to compute a complete global illumination solution according to Jensen (2001). The path-tracing technique was also introduced by Kajiya *(1986)* as a solution to the rendering equation. This technique was inspired by Cook *et al. (1984)* paper on the distribution ray tracing algorithm, which employed random sampling to capture effects like soft shadows and motion blur. Path tracing takes this a step further, sampling across all possible light paths.

Path tracing is an enhancement to ray tracing that calculates complex lighting effects. It does this by sending out a “random” ray to estimate specific light interactions. For instance, when a ray hits a surface that scatters light in all directions, the method calculates the scattered light by sending out another random ray.

In path tracing, the lighting pattern is determined by sending rays in random directions along all potential light paths. By considering many sample rays for a single point in the image, an average value of all light interactions at that point is obtained.

A key feature of path tracing is its efficiency. While it only uses one reflected ray to estimate light bouncing off surfaces, it ensures that equal effort goes into surfaces directly visible to the viewer. However, to get a precise result for a point in the image, it is often necessary to consider the average of multiple primary rays, sometimes thousands.

A challenge with path tracing is the inconsistency in its results, which can appear as noise in the final image. This noise arises when too few samples are considered for a point, leading to an inaccurate light interaction estimate. However, there are methods capable of reducing this inconsistency, as the multiple importance sampling explained in section 3.1.5, leading to more accurate results.

Jensen (2001) defines the path-tracing algorithm as the one seen in Figure 3.5. The main point here is that all rays (not just specular reflections) are traced in the shade() function and that additional elements for each ray (such as pixel position) can be randomly sampled.

```
render image using path tracing
  for each pixel
    color = 0
    for each sample
      pick ray from observer through random position in pixel
      pick a random time and lens position for the ray
      color = color + trace( ray )
    pixel-color = color/#samples

trace( ray )
  find nearest intersection with scene
  compute intersection point and normal
  color = shade( point, normal )
  return color

shade( point, normal )
  color = 0
  for each light source
    test visibility of random position on light source
    if visible
      color = color + direct illumination
  color = color + trace( a randomly reflected ray )
  return color
```

Figure 3.5 - The path-tracing algorithm (Jensen (2001))

3.2 Volume

Fong *et al.* (2017) describe a volume as a collection of particles. These particles can be as tiny as atoms and molecules or as vast as stars that contribute to the transport of galactic radiance. For a volume to function as intended, the average density of these particles should be on the lower side. This ensures that the individual size of these particles becomes minor compared to the mean distance separating them. Such a setup is conducive for the particles to collide in a statistically independent way, a phenomenon frequently seen in gaseous environments. However, this is not true with denser substances like sand or snow. When photons traverse through such a volume, they might collide with other particles. These interactions shape the radiance within the volume. Given the impracticality of individually mapping out every particle, they are instead represented using fields that predict the likelihood of collisions. The odds of such photon collisions are indicated by a coefficient, $\sigma(x)$, representing the likelihood of a collision for every unit of distance the photon covers inside the volume.

3.2.1 Properties

Volumes have specific properties, mainly determined by absorption, scattering coefficients, phase function, and emission (Fong *et al.* (2017)):

- *Absorption* ($\sigma_a(x)$): is when a photon gets absorbed by the volume, effectively disappearing from the area of interest. In simple terms, the energy from the photon gets transferred, often turning into heat.

- *Scattering* ($\sigma_s(x)$): This is when a photon, upon collision, changes direction but keeps its radiance value intact. Any alteration in the radiance value is managed through absorption and emission.

- *Phase Function* $f_p(x, \omega, \omega')$: It defines the e angular distribution in which radiance is scattered. Phase functions ensure that the scattered radiance remains consistent and balanced. Fong *et al.* (2017) state that phase functions operate similarly to the BSDF in surface rendering since its directions are interchangeable (they also follow Helmholtz's law).

  Isotropic volumes scatter light uniformly in every direction. Their phase function is defined in equation 3.24:

$$f_p(x, \theta) = \frac{1}{4\pi} \tag{3.24}$$

  In contrast, anisotropic volumes can have more complex phase functions. The most commonly used phase function in practical settings is the Henyey-Greenstein (1941) phase function (equation 3.24), which can model various scattering behaviours based on the parameter δ. This parameter controls the asymmetry of the phase function modelling backwards scattering when $\delta < 0$, isotropic scattering when $\delta = 0$, and forward scattering when $\delta > 0$.

$$f_p(x, \theta) = \frac{1}{4\pi} \frac{1 - \delta^2}{(1 + \delta^2 - 2\delta \cos \theta)^{\frac{3}{2}}} \tag{3.25}$$

- *Emission* $L_e(x, \omega)$: Volumes can also emit radiance, similar to other light sources. The way this radiance is projected can vary, but in general, volumes do not emit in any particular direction, making their radiance uniform across all directions. If a volume does not emit, its emission value is zero.

- *Extinction coefficient* ($\sigma_t(x)$):: This refers to the sum of the absorption and scattering coefficients, symbolised as $\sigma_t = \sigma_a + \sigma_s$. Informally, it is often termed as the density or the attenuation coefficient. Essentially, the extinction coefficient captures the total decrease in radiance due to both absorption and scattering within a volume. When an 'extinction collision' occurs, it means both absorption and scattering play a role, and it is necessary to ensure that the scattered light is accurately adjusted in calculations using the single scattering albedo.

- *Single scattering albedo (α)*: Denoted by $\alpha = \frac{\sigma_s}{\sigma_t}$, measures a volume's overall reflectivity, similar to the surface albedo. It determines how much radiance gets scattered. If α equals 0, all the radiance is absorbed, as seen in substances like black coal dust. Conversely, an α of 1 indicates that there is no absorption, leading to lossless scattering, much like in clouds.

3.2.2 Light propagation

The *radiative transfer equation* (RTE) is pivotal in delineating the distribution of radiance within volumes. Introduced by Chandrasekhar (1950), the RTE explicates the equilibrium radiance field denoted as $L(x, \omega)$ when mapped against the position x and direction ω. This is contextualised within the parameters set by a specific volume, and it duly incorporates any boundaries dictated by geometrical constraints and light sources.

The terms that constitute the RTE are the following, according to Fong *et al.* (2017)

- Absorption: When discussing a radiance beam symbolised as $L(x, \omega)$, which originates at point x and is directed towards ω, it is discerned that the derivative of this radiance in the direction of ω is a function of the radiance present at that specific location. This relationship is modulated by the absorption coefficient, σ_a, leading to the expression:$(\omega \cdot \nabla)L = -\sigma_a(x)L(x, \omega)$. This equation is essentially a spatial rendition of the Beer-Lambert Law that depicts the decrease in radiance resulting from absorption.

- Out-Scattering: The said radiance beam, $L(x, \omega)$, also witnesses a decline in its radiance due to out-scattering. This is not a loss in the cumulative radiance field but specifically to its original direction ω. The decrement is proportionate to the radiance $L(x, \omega)$, and this relationship is steered by the scattering coefficient, σ_s, as illustrated in the equation: $(\omega \cdot \nabla)L(x,\omega) = -\sigma_s(x)L(x,\omega)$

- Emission, depicted as a distinct radiance field $L_e(x,\omega)$, is indicative of the radiance added to $L(x,\omega)$. Its mathematical representation can be expressed as$(\omega \cdot \nabla)L(x,\omega) = \sigma_a(x)L_e(x,\omega)$. It is imperative to note that certain sources, including PBRT, approach emission by establishing a radiance or source term sans the absorption coefficient, $\sigma_a(x)$. However, for a thorough and congruent explication of radiative transfer, it is crucial to incorporate this coefficient to ascertain the correct set-up of the emissive radiance vis-à-vis absorption.

- In-Scattering: In-scattering emerges from the out-scattering spanning all other directions ω' at point x, thus amplifying the net radiance of the initial radiance beam. Mathematically, this can be represented as: $(\omega \cdot \nabla)L(x,\omega) = \sigma_s(x)\int_{S^2} f_p(x,\omega,\omega')L(x,\omega')\mathrm{d}\,\omega'$. The term S^2 specifically denotes the spherical domain surrounding the position x. In this context S^2 represents a unit sphere that encapsulates all possible directions from which light can scatter into the volume at that position.

In synthesising the definitive form of the RTE, all the components are combined. Given their inherent kinship, both absorption and out-scattering are consolidated under the umbrella term of the extinction coefficient, $\sigma_t = \sigma_a + \sigma_s$. The resultant equation representing this amalgamation is

$$(\omega \cdot \nabla)L(x,\omega) = -\sigma_t(x)L(x,\omega) + \sigma_a(x)L_e(x,\omega) \\ + \sigma_s(x)\int_{S^2} f_p(x,\omega,\omega')L(x,\omega')d\omega' \tag{3.26}$$

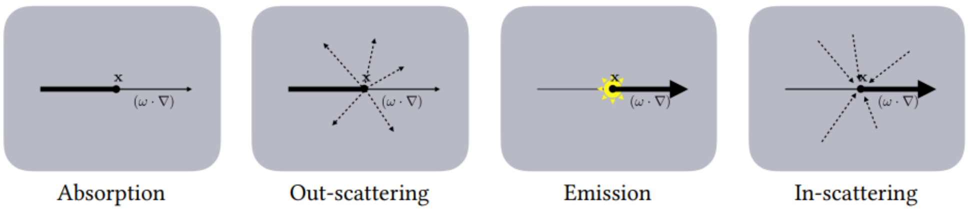

Figure 3.6 – RTE terms (Fong et al. (2017))

3.2.3 Rendering Equation

The Radiative Transfer Equation (RTE) elucidates how radiance disperses, taking a forward-transport approach. It uses gradients to describe the progression of a radiance beam. While suitable for finite element methods, its application to path tracing requires adaptation. This necessity has led to the formulation of the *Volume Rendering Equation* (VRE).

To streamline this mathematical representation, the in-scattering process is condensed, and by integrating both sides of the RTE, the gradients are transmuted into an integral, thus providing an explicit formula for $L(x, \omega)$. This transformation produces the VRE, denoted as:

$$L(x, \omega) = \int_{t=0}^{d} \exp\left(-\int_{S=0}^{t} \sigma_t(x_s)\, ds\right) [\sigma_a(x_t) L_e(x_t, \omega) + \sigma_s(x_t) L_s(x_t, \omega) + L_d(x_d, \omega)]\, dt \tag{3.27}$$

This equation's formulation capitalises on the negative part of the radiance beam. By parameterising in this manner, the VRE focuses on the origins of radiance rather than its destinations. This backwards accumulation of contributions ensures that the direction ω consistently indicates the flow of radiance, eliminating any complexities stemming from directional ambiguities.

Enhancing clarity, the term that captures the exponential integral is further abbreviated as:

$$T(t) = \exp\left(-\int_{S=0}^{t} \sigma_t(x_s) \mathrm{d}\, s\right) \tag{3.28}$$

This term, referred to as ***transmittance***, quantifies the cumulative effect of absorption and out-scattering from the ray's origin to any given point. Employing this simplification, the VRE is rendered in a more succinct form, conducive for path tracing applications. This version of the VRE can be represented as:

$$L(x, \omega) = \int_{t=0}^{d} T(t) [\sigma_a(x) L_e(x_t, \omega) + \sigma_s(x) L_s(x_t, \omega)]\, dt + T(d) L_d(x_d, \omega) \tag{3.29}$$

It is pivotal to note that radiance at any point on the ray needs to account for the transmittance up to that location to determine its contribution to $L(x, \omega)$. Upon visualisation, one discerns that the VRE, encapsulated in this equation, is an extension of the rendering equation (Kajiya (1986)) to incorporate volume structures, considering surfaces as highly dense volumes with intricate phase functions.

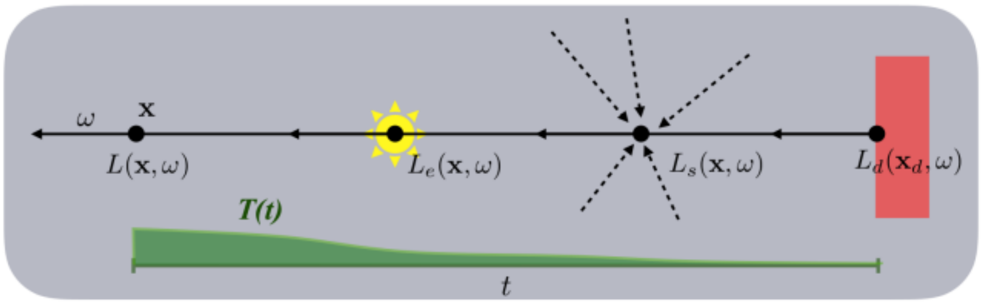
ω
$\mathbf{x}$
$L(\mathbf{x}, \omega)$
$L_e(\mathbf{x}, \omega)$
$L_s(\mathbf{x}, \omega)$
$L_d(\mathbf{x}_d, \omega)$
$T(t)$
t

Figure 3.7 – visualisation of the VRE (Fong et al. (2017))

3.2.4 Tracking Approach

Tracking approaches are one of the principal stochastics Monte Carlo approaches that can be used to solve the VRE. The tracking methodology utilises strategies such as Russian roulette and rejection sampling to determine a single type of photon collision to be modelled. By simulating photon interactions within a volume, this method avoids fragmenting the radiance beam's contributions. However, a challenge arises when determining the distances a photon travels between different collision types.

Closed-form Tracking

In simpler volumes, such as those with consistent or polynomial extinction rates, free paths can be determined using inverse transform sampling. Especially for the homogeneous volume, transmittance is expressed through the **Beer-Lambert law** as:

$$T(t) = \exp(-\sigma_t t) \tag{3.29}$$

This function describes the exponential decrease of radiance. For tracking within such a volume, the goal is to pinpoint the exact distance a photon covers given a consistent transmittance, and normalising the transmittance function, a probability density function is formed, denoted as $p(t) = \sigma_t \exp(-\sigma_t t)$. According to Fong *et al.* (2017), this can then be perfectly importance sampled, resulting in $t_0 = -\frac{\ln(1-\xi)}{\sigma_t}$, which provides the photon's travel distance. When this is integrated into the Volume Rendering Equation (VRE) for homogeneous volumes, it becomes necessary to factor in normalisation and decide which photon collision to model, whether absorption/emission or in-scattering.This decision is made using a Russian roulette approach.

The closed-form tracking algorithm efficiently manages this procedure, updating the ray's path based on the collision encountered. This method holds paramount significance in production rendering, being exceptionally effective in homogeneous volumes like fog or atmosphere. Moreover, for translucent objects requiring subsurface scattering rendering, such as skin, this method proves invaluable.

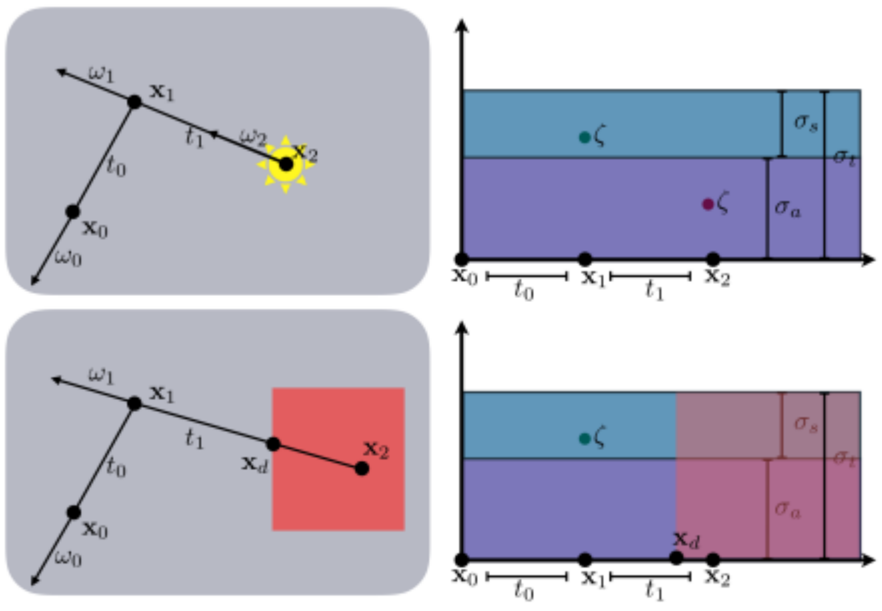

Figure 3.8 - A path scatters once and absorbs some emission (top) A path scatters scattering and then hitting a boundary (bottom). (Fong et al. (2017))

Delta Tracking

Delta tracking is a technique founded on von Neumann's rejection sampling, with origins in both neutron transport and plasma physics (Fong et al. (2017). Also known by other names such as Woodcock tracking, it introduces a fictitious collision type referred to as a "null-collision" to make the sampling process in heterogeneous volumes behave as though the medium were homogeneous.

The concept of null-collision results in scattering in the same incoming direction without any impact on the actual light transport. This is articulated through a null-collision coefficient $\sigma_n(x)$, which functions analogously to the physical coefficients. The overall volume is homogenised by choosing $\sigma_n(x)$ such that the sum of all coefficients, termed the free-path coefficient $\overline{\sigma}$, remains constant, expressed as $\overline{\sigma} = \sigma_a(x) + \sigma_s(x) + \sigma_n(x) = \sigma_t(x) + \sigma_n(x)$. Furthermore, it is defined that $\overline{\sigma} \geq \sigma_t(x)$, leading to $\sigma_n(x) = \overline{\sigma} - \sigma_t(x)$.

Since $\bar{\sigma}$ remains constant, it can replace the constant extinction in the closed-form tracking equation, resulting in a new expression for distance sampling, $p_n(t) = \overline{\sigma}\exp(-\overline{\sigma}t)$. With three collision types instead of two, probabilities for the collisions are defined by the equations, $P_a(x) = \frac{\overline{\sigma}}{\sigma_a(x)}$, $P_s(x) = \frac{\overline{\sigma}}{\sigma_s(x)}$, $P_n(x) = \frac{\overline{\sigma}}{\sigma_n(x)}$, with the sum equalling one.

The application of the null-collision probability gives rise to a recursive form that closely resembles the closed-form version of tracking. In the recursion, Russian Roulette selects one of the three collision types. If the null collision is chosen, the process continues unaltered. The method requires $\bar{\sigma}$ to be close to the maximum of σ_t, and to achieve efficiency, volumes may be subdivided into domains, allowing for a tightly defined $\bar{\sigma}$ in Delta tracking.

This approach, through the introduction of null-collisions, aligns heterogeneous media with closed-form tracking while maintaining the efficiency of the calculation and introducing the flexibility to handle variations within the medium. It represents an elegant solution to a complex problem, balancing computational rigour with the physical realities of light transport.

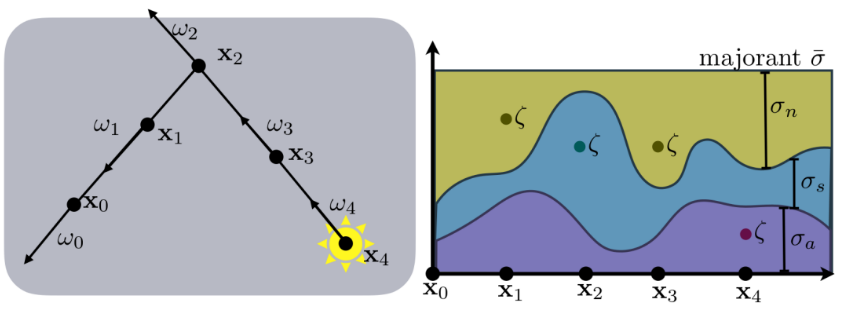

Figure 3.9 - A path scatters once and absorbs some emission (top) A path scatters scattering and then hits a boundary (bottom). (Fong et al. (2017))

4 Design and Implementation

Overview

This chapter will delve into the complexity of constructing a stand-alone renderer fortified by path-tracing techniques that are adept at rendering specific volumes. Drawing from books such as Shirley *et al.* (2023) and seminal works such as *Fong et al.* (2017), the foundation is meticulously crafted to foster the integration of diverse volumetric elements into path tracers. The chapter delineates the objectives of crafting scenes with precision and correct material properties, employing Physically-Based Rendering (PBR) techniques. Building upon a prior project in ray tracing, this chapter recounts the assimilation of essential libraries, culminating in an industry-standard outcome. An exposition of the software framework is provided, emphasising its modular architecture, promoting ease of understanding and further enhancement. The chapter further elucidates the nuanced dynamics of volume integration, underpinning the various challenges and solutions in this domain. Through exploration, the reader gains a comprehensive understanding of the renderer's design, functionalities, and the techniques employed in achieving the envisaged objectives.

4.1 Objectives

This thesis is dedicated to the creation of a stand-alone renderer, supported by path tracing techniques, with the capacity to render some volumes. The foundational structure for this work is heavily influenced by the methodologies elucidated in Shirley *et al.* (2023) and Fong *et al.* (2017). These pivotal papers offer techniques that facilitate the creation of a renderer and the integration of a variety of volumetric elements into path tracers.

The objective of this project is to render scenes where objects are presented with precision and correct material properties. In achieving this, PBR (Physically-Based Rendering) techniques were harnessed. Furthermore, the capability to render geometries with the nuances of homogeneous and heterogeneous volumes was established.

This project is a progressive continuation of an earlier work on ray tracing. This antecedent project was part of the Animation Software Engineering (ASE) module, which I undertook in the first semester of the MSc Computer Animation and Visual Effects course. The structure of that project was based on Buck (2018).

4.2 Libraries

In this project, certain dependencies were selected to align with industry norms. These decisions were necessary to take advantage of existing solutions while focusing on the primary objectives.

The integration of the Intel Threading Building Blocks (TBB) library is important. TBB is a C++ template library that offers a set of components for parallel programming. This library enhances the program's performance by optimising the execution of tasks.

The NGL library also plays an important role, offering a suite of tools tailored for graphics and rendering in C++. While complex in its capabilities, it streamlines the rendering process, ensuring that graphic representations are correct.
For the construction and generation process, the vcpkg toolchain was utilised. This ensures that dependencies are managed and integrated, simplifying the build process and ensuring compatibility.

Furthermore, Qt, a widget toolkit, was utilised to create a graphical user interface to visualise the rendered image.

In terms of versioning, the libraries were chosen as follows:

- TBB (as per the version offered by vcpkg)
- NGL (as per the local configuration)
- Qt (prioritising version 6, but falling back to version 5 if not available)

All libraries were incorporated to aid in the development and ensure that the RayTracer maintains performance and robustness. It is pertinent to mention that while building the project, using CMake with a minimum version of 3.12 is advocated to ensure seamless integration of all the components.

4.3 Design

The software framework organises functions and features in a clear way, making it easy to understand and expand upon, enabling the creation of realistic scenes.

- **Modular Design:** At the core of the framework is a modular design ethos. Each component or functionality is encapsulated in its own class, ensuring clear separation and minimal overlap. This approach promotes ease of maintenance and potential future expansions.
- **Rendering Components:** The rendering pipeline relies on foundational classes like Ray and Intersection, with the NGLScene class serving as the primary rendering context.
- **Lighting:** The framework introduces a class for AreaLight, as well as Bidirectional Scattering Distribution Function (BSDF) subclasses like LambertianBSDF (a surface BSDF) and IsotropicBSDF (a volume phase function), to ensure realistic and physically based lighting in rendering.
- **Materials:** Materials determine how surfaces interact with light, reflecting realism. The architecture includes Material and subclasses like Lambertian and BeersLawHeterogeneousMaterial for nuanced properties and shading.
- **Geometric Primitives:** The framework can handle different geometric shapes. The Sphere and Triangle subclasses represent specific shapes, while the Group class enables complex scene compositions with a hierarchical structure.
- **Volume Rendering:** Incorporating volumetric effects adds layers of depth to the scenes. The framework includes classes such as Volume and its subclasses like BeersLawVolume. These subclasses are the ones responsible for defining a volume transmittance and its tracking approach.
- **Utilities:** To streamline operations and foster efficiency, utility classes like Utility and Transformations have been incorporated. These serve as essential tools, assisting in common tasks and ensuring a seamless workflow.
- **Scene Management:** The Scene class orchestrates the assembly of objects, lights, and other elements, encapsulating the essence of a structured approach to 3D environment composition.
- **Viewing Mechanisms:** Perspective and projection play a crucial role in rendering. The Camera and Canvas classes in the architecture ensure dynamic viewing angles and diverse perspectives.
- **File Handling:** Interactivity and compatibility are achieved through the ObjFile

class, which allows for the integration of external 3D models with the OBJ format.

In resume, the software framework offers a meticulously crafted design, prioritising realism and flexibility. Its modular and comprehensive nature ensures that it stands as a robust platform for both current and future rendering implementations. A class diagram that elucidates the overall relation of the classes is shown in Figure 4.1. The system implementation can be found online in the following GitHub repository: https://github.com/gabicfa/volume-rendering-through-path-tracing

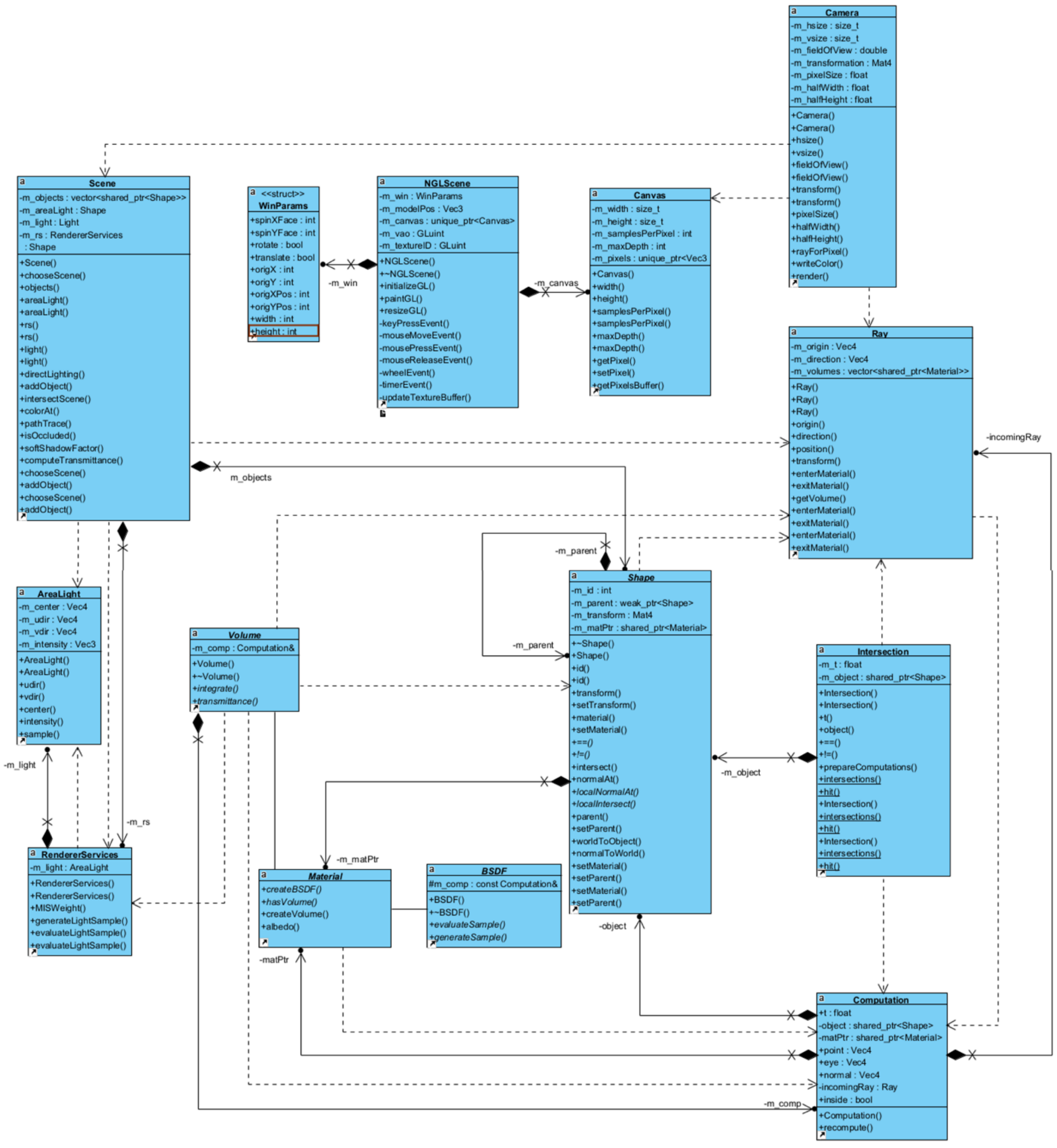

Figure 4.1 – Class Diagram

4.4 Volume integration

According to Fong *et al.* (2017), no single preferred integration technique for volumes works well across every type of participating media. Simpler cases can be efficiently addressed through a direct beam transmittance computation incorporated into a path tracing algorithm's throughput calculation. Conversely, intricate situations, for instance, light dispersion through media with low albedo properties like smoke, predominantly demand algorithms tailored for single scattering. The utmost complex media categories, exemplified by phenomena like clouds, necessitate intensive multiple scattering coupled with sophisticated anisotropic phase functions.

To optimise the complexity of various volume integration models within a lighting integrator, there is a concerted effort to separate volume integration from the overarching light integration problem. This approach sees the integration domain partitioned into smaller volumetric subdomains, especially when participating media comes into play. Control over the integration within these domains is then designated to more compact volume modules.

To do so, for each geometric primitive present in the scene, a corresponding instance of the Material class is attached to that specific primitive. Upon the incidence of a ray striking a geometric primitive, the geometric attributes at the point of impact are consolidated within a Computation, encompassing essential parameters such as the position of the point of contact, the surface is normal, and the reverse direction of the incoming ray, denoted as eye.

Every Material inherently possesses a createBSDF() method. This method yields a BSDF object when provided with a Computation. Furthermore, this BSDF object is structured to accommodate both an evaluateSample() method and a generateSample method(). These methods, potentially using the data from the Computation, determine their operational procedure. The evaluateSample() is utilised to ascertain the BSDF's response to a sample of light, especially when given an incoming ray direction labelled as sampleDirection and the outgoing ray's direction derived from Computation.eye. Meanwhile, the generateSample() method's primary function is to sample the BSDF. It achieves this by producing an outgoing ray direction, the sampleDirection, which stems from an evaluation of a BRDF, and the subsequent PDF associated with that ray direction.

Central to this process is the light integrator module within the Scene class, which takes on the role of executing a path-tracing algorithm. This operation is facilitated by a RendererServices object, which furnishes services essential for sampling the scene's lights. The RendererServices object incorporates a generateLightSample() method that is instrumental in sampling the light database using a shading context derived from a particular point. The outcomes from this method are a sampled direction – sampleDirection, the radiance L reaching the point from this direction, the related probability density pdf, and crucially, the beamTransmittance between the point and the source of light. A parallel method, the evaluateLightSample(), designed for BSDF sampling, delivers a similar set of results. However, for both methods to be executed, the calculation of beamTransmittance necessitates the initiation of a transmission ray.

Incorporating volumes into the path-tracer required the Material class to be adapted to produce a Volume object and led to the creation of the Volume interface. A Volume, when instantiated, utilises a Computation, which fundamentally represents the geometric context of the hit point that initiates the volume interval. The volume's integration starts with the integrate() method, which requires an input ray featuring direction wi and origin Computation.point, specifying the direction of the volume interval. This choice is typically influenced by the BSDF::generateSample() executed at the hit point Computation.point, and a RendererServices object that aids the Volume during integration. The outcomes encompass the radiance L along the ray, the radiance estimate's weight, the beam transmittance across the volume integration interval, and the hit point. The volume integrator determines the distance d and returns x_d in the output parameter P, and concurrently offers $T(d)$ in the output parameter transmittance.

It is pivotal to recognise that within this configuration, a volume integrator does not operate by itself. Instead, it is an component of a Material, governing solely the volumetric integration within a geometric entity. The surface attributes of this geometry are exemplified by the BSDF, which is a product of the createBSDF() method within that Material

5 Results

The results delineated within this chapter derive from exploration the path tracer tailored for volume rendering. The main objective of this empirical analysis is to discern the effectiveness and potential areas of improvement in the developed rendering methodology.

Several scenes were rendered to elucidate the capabilities of the lights and BSDF. A salient aspect evaluated was the influence of ‘samplesPerPixel’ on the rendered image’s quality. Additionally, the impact of MIS Sample was examined. A further exploration was conducted with spheres employing both Homogeneous Beer’s Law and Heterogeneous Beer’s Law volumes to it.

The first image analysed was a scenario where two specular balls were positioned within a Cornell Box. The image was rendered with 1600 samples per pixel, and the results are illustrated in Figure 5.1

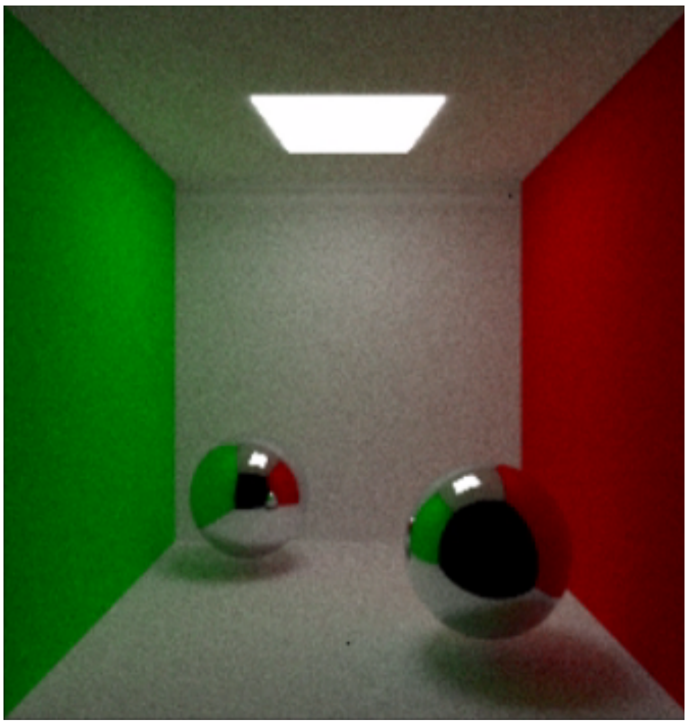
Figure 5.1 – Cornell box with two specular balls

Upon analysis, it is evident that the reflections captured are accurate, with the wall’s colour influencing the hues of the shadows. Although the image quality is good with few noise, some noise can still be discerned.

Following this, the Cornell Box was rendered with two boxes and a reduced resolution of 600 samples per pixel (spp), showcased in Figure 5.2

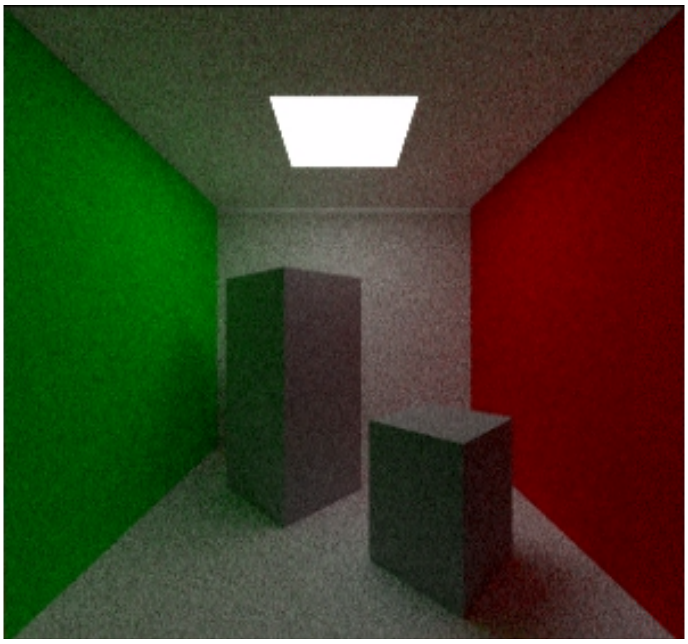

Figure 5.2 – Cornell box with 600 spp

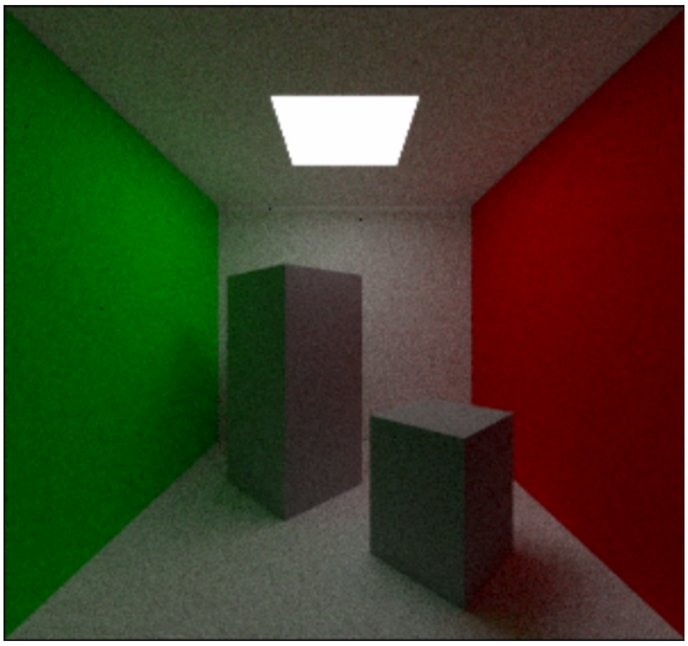

Figure 5.3 – Cornell box with 1200 spp

While the reflections remain accurate, the image displays a pronounced noise level, suggesting that an increase in samples per pixel might enhance the quality.

To test this hypothesis, the Cornell Box was rendered again, this time with an elevated 1200 samples per pixel. As seen in Figure 5.3, the increased number of samples results in a significant noise reduction, thus underscoring the pivotal role of sample count.

Shifting the focus to volume rendering, spheres were rendered using the Homogeneous Beer's Law. This method interprets light absorption as it traverses a shape, with the specific volume designated to its BSDF, much akin to fog without internal light scattering. The absorption coefficient of the medium determines how much light is absorbed as it travels through the volume. In Figure 5.5, a sphere with 0.4 absorption coefficient is presented, revealing a pronounced transparency.

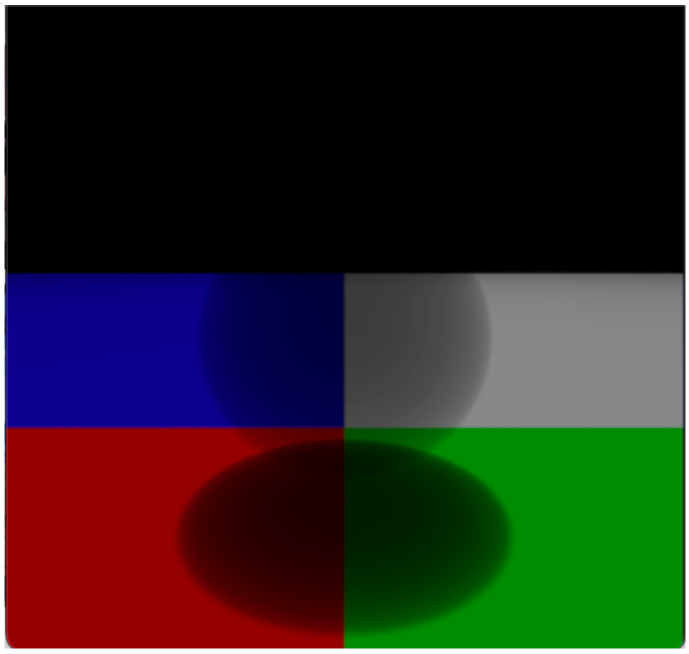

Figure 5.4 – Beer's Law Homogeneous volume (absorption 0.4)

With absorption levels ratcheted up to 0.8, as depicted in Figure 5.6 there is a discernible escalation in density, making the volume less see-through. Also, the shadows in these renderings are rightly defined, as they consider volume transmittance when being rendered.

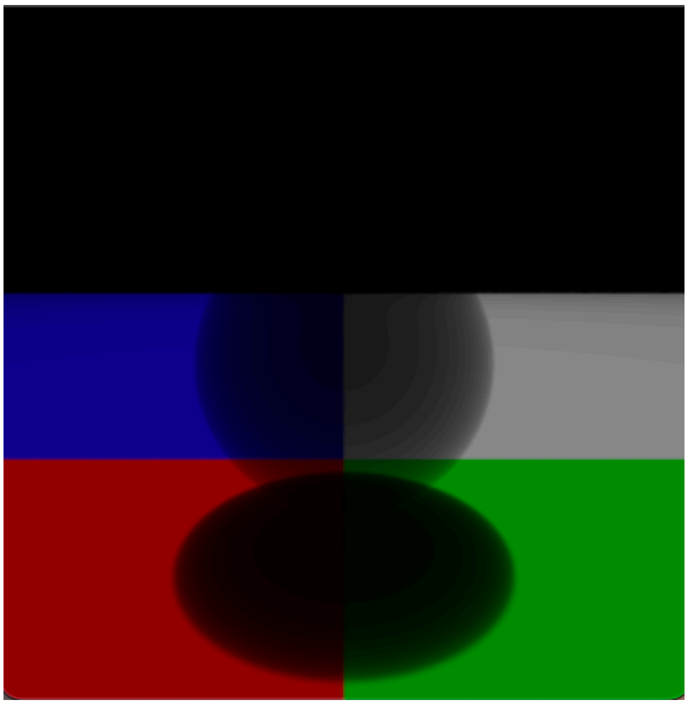
Figure 5.5 – Beer’s Law Homogeneous volume (absorption 0.8)

Further delving into volume rendering, the Heterogeneous Beer’s Law Volume was harnessed. This modality combines Beer’s Law within a heterogeneous medium, permitting variations in absorption properties. The absorption disparity was computed employing Perlin noise, predicated on ray intersection points with the shape. Figure 5.7 and Figure 5.8 register the renderings using this method, elucidating the effects of differential maximum absorption intensities. Here too, the shadows rendered are particularly noteworthy, as they take into account volume transmittance, further accentuating the accuracy of the technique.try defined, as they consider volume transmittance when being rendered.

Concluding the image analyses, a rendering was assessed where a specular object was juxtaposed next to a volume. As delineated in Figure 5.8. It is possible to see that the specular object delivers a reflection through the volume, amplifying the visibility of entities positioned behind the volume.

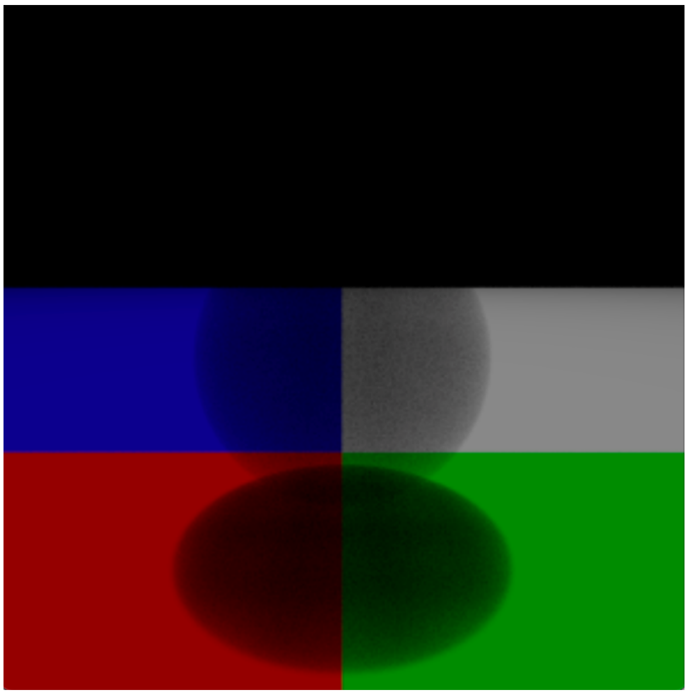

Figure 5.6 – Beer's Law Homogeneous volume (max absorption 0.5)

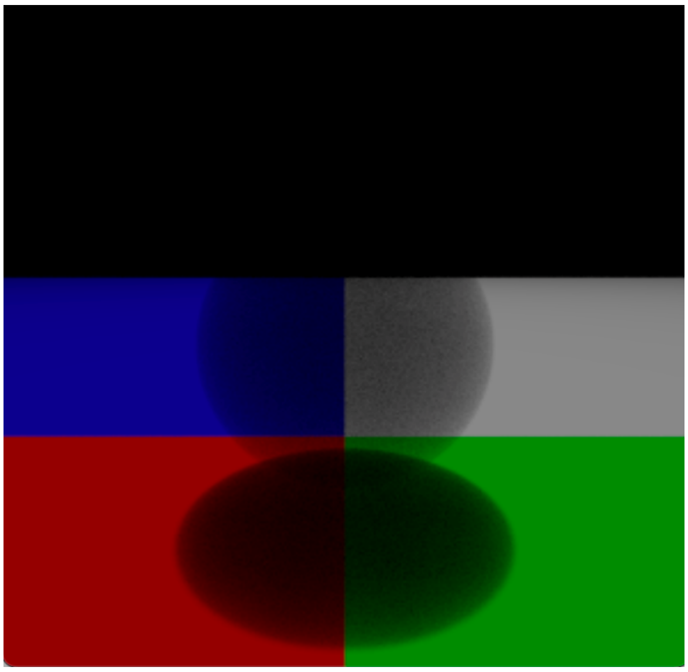

Figure 5.7 – Beer's Law Homogeneous volume (max absorption 0.9)

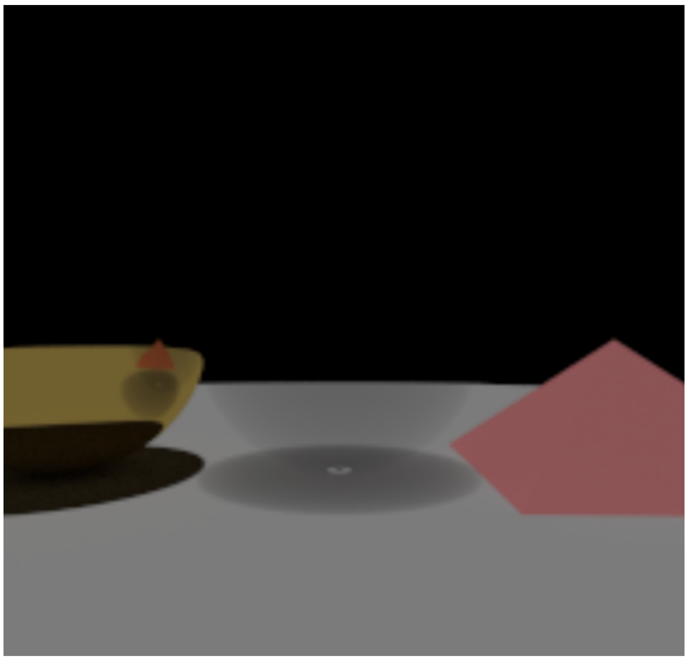

Figure 5.8 – Scene with volumetric, specular and diffuse shapes

To conclude, the empirical results delineated within this chapter offer a comprehensive insight into the capabilities and nuances of the developed path tracer tailored for volume rendering. Through analysis of various renderings encompassing diverse environments and absorption levels, the research underscores the role of precise sample counts and the application of both Homogeneous and Heterogeneous Beer's Law. Especially noteworthy is the renderer's adeptness at accounting for volume transmittance, ensuring that media and shadows are rendered accurately.

6 Conclusion

The completion of this research project is encapsulated in this Conclusion chapter, providing an overview and synthesis of the main findings and implications.

At the beginning of this project, the primary objective was to develop a stand-alone path tracer renderer with a specific capability to render volumes akin to atmospheric phenomena such as fog. This endeavour was successful, with the renderer visualising such volumes. Moreover, another important result relates to the Bidirectional Scattering Distribution Function (BSDF) of diffuse and specular objects, which were accurately rendered in the developed system. Furthermore, the role of the 'samples per pixel' parameter was highlighted, with the research elucidating its direct impact on image clarity and accuracy.

Throughout the research process, it was important to maintain a sense of critical examination, ensuring the methods employed and results obtained were both reliable and valid. The stand-alone path tracer renderer's development, which is proficient in volume rendering, represents a significant milestone. However, some aspects need to be examined more closely. The image quality, although satisfactory, still exhibits a degree of noise, even with higher sample counts. Moreover, the renderer is proficient at rendering volumes but can be quite slow.

As the research concludes, it highlights potential avenues for exploration in subsequent stages of development. The challenge of performance optimisation is central among these. While promising, the current renderer version could benefit from enhancements that enable faster render times. As the demand for high-quality graphics grows, ensuring that the renderer is accurate and efficient and delivers optimal results without extended wait times becomes crucial.

Simultaneously, there is an expanding interest in rendering more complex volumes. The realm of computer graphics is increasingly moving towards replicating intricate real-world phenomena. One such example is the rendering of single scatter volumes, akin to the visual properties of smoke. Another intriguing prospect is the rendering of multi-scatter volumes, which could simulate the appearance of clouds. The renderer could be improved to render more realistic and immersive visual experiences by accommodating these complex volumes.

In summation, the research has achieved its objectives. However, there are areas that could be improved.

References

Buck, J. (2018). The Ray Tracer Challenge: A Test-Driven Guide to Your First 3D Renderer. The Pragmatic Programmers.

Chandrasekhar, S. (1950). Radiative Transfer. Oxford University Press, London.

Cohen, M.F., Greenberg, D.P., Immel, D.S. and Brock, P.J., 1986. An efficient radiosity approach for realistic image synthesis [online]. IEEE Computer Graphics and Applications, 6(3). Available from: https://doi.org/10.1109/MCG.1986.276629 [Accessed 12 August 2023].

Cook, R.L., Porter, T. and Carpenter, L., 1984. Distributed Ray Tracing [online]. ACM SIGGRAPH Computer Graphics, 18(3), pp.137-145. Available from: https://doi.org/10.1145/964965.808590 [Accessed 12 August 2023].

Drebin, R.A., Carpenter, L. and Hanrahan, P., 1988. Volume rendering [online]. ACM SIGGRAPH Computer Graphics, 22(4), pp.65-74. Available from: https://doi.org/10.1145/378456.378484 [Accessed 12 August 2023]

Engel, K., Hadwiger, M., Kniss, J.M., and Rezk-Salama, C., 2006. Real-Time Volume Graphics [online]. EUROGRAPHICS 2006 Tutorial. Vienna: The Eurographics Association. Available from: https://webdocs.cs.ualberta.ca/~pierreb/Visualization2006/Real-Time-Volume-Rendering.pdf [Accessed 12 August 2023].

Fong, J., Wrenninge, M., Kulla, C. and Habel, R., 2017. Production Volume Rendering [online]. In: SIGGRAPH 2017 Course, 44th International Conference and Exhibition on Computer Graphics and Interactive Techniques, Los Angeles 30 July-3 August 2017. Available from: https://graphics.pixar.com/library/ProductionVolumeRendering/paper.pdf [Accessed 8 June 2023].

Heidrich, W., McCool, M. and Stevens, J., 1995. Interactive Maximum Projection Volume Rendering [online]. In: Proceedings Visualisation ’95, Atlanta, GA, USA, 29 October - 03 November 1995. Piscataway, NJ: IEEE. Available from: https://ieeexplore.ieee.org/abstract/document/480790/citations#citations [Accessed 12 August 2023].

Henyey, L.G. and Greenstein, J.L., 1941. Diffuse radiation in the Galaxy. Astrophysical Journal, 93(January), pp.70–83.

Jensen, H. W., 2001, Realistic image synthesis using photon mapping. Natick, MA: A K Peters.

Kajiya, J.T., 1986. The rendering equation. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’86, New York, NY, USA. ACM, pp.143-150.

Levoy, M., 1988. Display of surfaces from volume data [online]. IEEE Computer Graphics and Applications, 8(3). Available from: https://doi.org/10.1109/38.511 [Accessed 12 August 2023].

Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., and Limperis, T., 1977. Geometric considerations and nomenclature for reflectance. Monograph 161. National Bureau of Standards (US).

Pharr, M. and Humphreys, G., 2010. Physically Based Rendering: From Theory to Implementation (2nd ed.) [online]. San Francisco, CA: Morgan Kaufmann Publishers Inc. Available from: https://www.pbr-book.org/ [Accessed 12 August 2023].

Shirley, P., Black, T. D., & Hollasch, S. (2023). Ray Tracing: The Next Week (Version 4.0.0-alpha.1). Ray Tracing in One Weekend Book Series. Available from https://raytracing.github.io/books/RayTracingTheNextWeek.html [Accessed 12 August 2023].

Veach, E., 1995. Bidirectional Estimators for Light Transport [online]. Available from: https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Veach94.pdf [Accessed 12 August 2023].

Veach E. Robust Monte Carlo Methods for Light Transport Simulation [online]. PhD thesis, Stanford, CA, USA, 1998. Available from: https://graphics.stanford.edu/papers/veach_thesis/thesis-bw.pdf [Accessed 12 August 2023].

Whitted, T., 1979. An improved illumination model for shaded display [online]. ACM

SIGGRAPH Computer Graphics, 13(2), pp.14-14. Available from:
https://doi.org/10.1145/965103.807419 [Accessed 12 August 2023].