LiU-ITN-TEK-A-15/007-SE

Evaluation of Visual Parameters in Volumetric Path Tracing

Robin Skånberg

2015-01-30

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Science and Technology
Linköping University
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

LiU-ITN-TEK-A-15/007-SE

Evaluation of Visual Parameters in Volumetric Path Tracing

Examensarbete utfört i Medieteknik
vid Tekniska högskolan vid
Linköpings universitet

Robin Skånberg

Handledare Daniel Jönsson
Examinator Timo Ropinski

Norrköping 2015-01-30

Abstract

With the development of modern GPUs, the computational power they provide have become more accessible and versatile. This has fueled the development of interactive visualization techniques and among them volumetric visualization. The objective of this project is to implement a volumetric path tracer as a reference renderer for volumetric visualization with OpenCL as the target platform. The implementation is created as a processor in an existing visualization framework called Inviwo. The components of the implementation are validated with the help of an analytic single scattering test scenario. The implemented path tracer is then used to render a series of scenes containing both sparse and dense participating media, where the different terms and parameters are studied. The results show comparisons of extinction coefficients, anisotropic scattering using the Henyey-Greenstein phase function, single vs. multiple scattering and a mixed BSDF that is comprised of both a BRDF and an isotropic phase function. One of the conclusions drawn from the results is that multiple scattering not only result in indirect lighting which light up the shadowed regions of the image, but also results in more vivid colors thanks to the sub-surface scattering that naturally occurs.

1 Introduction

1.1 Motivation

The trend of General Purpose computing on Graphical Processing Units, GPGPU has enabled developers to utilize the computational capabilities of modern Graphics Cards in more general ways. The performance of the GPUs also seem to grow exponentially with each new generation. This development has enabled advancements the fields of interactive rendering and among them the field of Volumetric Rendering, which has always been a computationally expensive area. In the field of Interactive Volumetric Rendering, the research is focused on solving the Volumetric Rendering Equation with the use of clever methods, approximations and simplifications in order to provide an interactive frame rate. In order to evaluate the accuracy of the implemented methods, they need to be compared to some reference solution. One method of solving the equation is to use a method called path tracing, that when implemented correctly, will provide an estimate of the true result. This estimate converges towards the true result with the computational time given, and is in the general case a rather slow process, far from that of any interactive one.

1.2 Objective

The objective is to implement a Volumetric Path Tracer that is true to the Volume Rendering Equation, where the focus lies in creating an accurate solution rather than one optimized for speed. With the implementation in place, the terms and parameters that appear in the Volume Rendering Equation can be studied and a deeper understanding of them can hopefully be achieved. The implementation itself shall fulfill certain set of demands:

GPU support The path tracing shall be done on the GPU to utilize the computational capabilities of modern graphic cards.

Running Estimate A running estimate should be maintained in order for the user to see the resulting image being refined with each iteration.

External Light Sources The implementation shall provide support for different types of light sources placed in the scene by the user.

1.3 Means

The Volumetric Path Tracer will be implemented in Inwivo, a software visualization framework designed for volumetric rendering. The core of the Path Tracer will be written in OpenCL to enable GPU-acceleration and to be able to utilize some pre-written functionality such as a Random Number Generator (RNG) and reflectance functions.

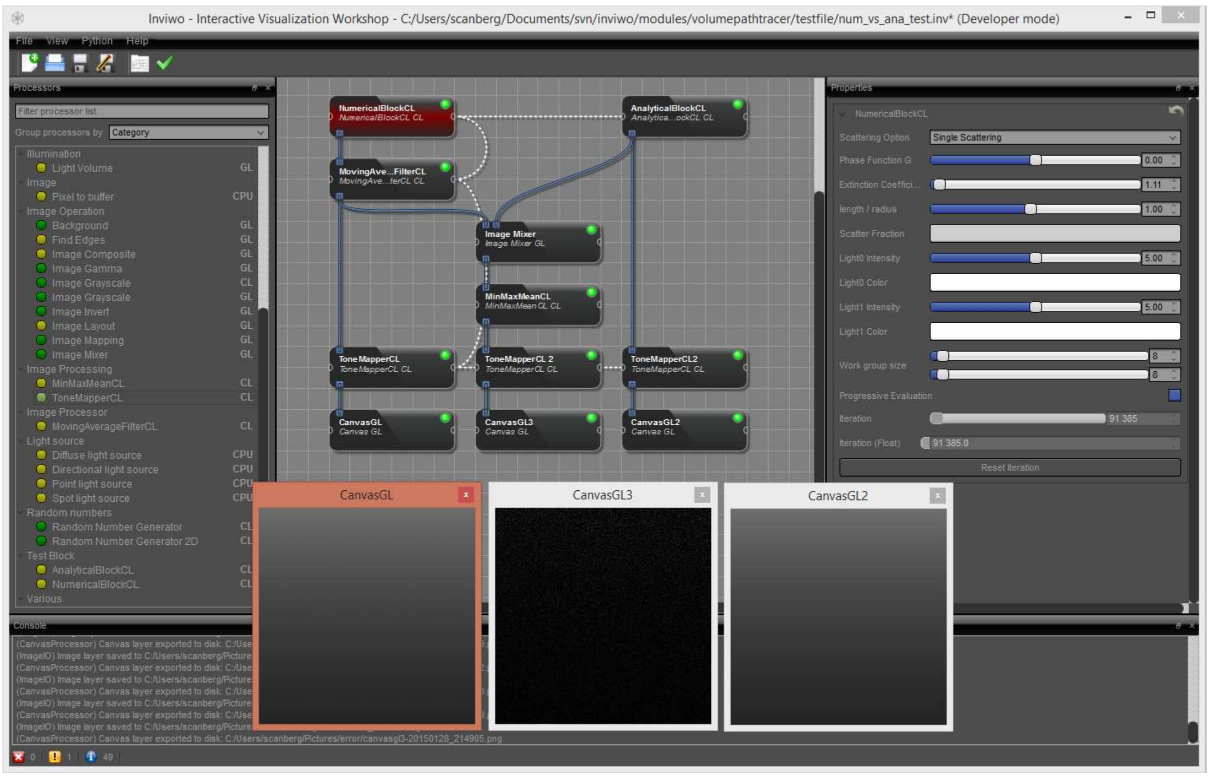
1.4 Outline

The outline of the thesis is to first provide the reader an introduction to the framework of Inviwo. Then there is a larger section of theory, where the aim is to provide the reader with a theoretical basis regarding Physically Based Rendering, which will be referenced in later sections. The theory section is written with the aim of being light and compact, to provide the reader with an intuitive understanding rather than a deep mathematically proven one. The Implementation is then presented where the equations and concepts shown in the theoretical section are put to use and are translated into code. The last section provides the results in form of renders that are studied and conclusions are drawn from these along with ideas for further improvements.

2 Background

2.1 Inviwo

Inviwo, short for Interactive Visualization Workshop, is a software framework that provides rapid visualization prototyping. Processors are connected via a node network to process some input data and output some other data. This framework model provides the user with simple, user friendly and versatile tools to model a flow of data that in the end is rendered into some visible format. Inviwo is written in C++ and uses the popular QT-library for handling and displaying Graphical User Interface, GUI-components. It also provides access to the GPU via the Application Program Interfaces, APIs, OpenGL and OpenCL.



Scattering Option: Single Scattering
Phase Function G: 0.00
Extinction Coeffici...: 1.11
length / radius: 1.00
Scatter Fraction
Light0 Intensity: 5.00
Light0 Color
Light1 Intensity: 5.00
Light1 Color
Work group size: 8, 8
Progressive Evaluation
Iteration: 91 385
Iteration (Float): 91 385.0

```
(CanvasProcessor) Canvas layer exported to disk: C:/Use
(ImageIO) Image layer saved to C:/Users/scanberg/Picture
(CanvasProcessor) Canvas layer exported to disk: C:/Use
(ImageIO) Image layer saved to C:/Users/scanberg/Picture
(CanvasProcessor) Canvas layer exported to disk: C:/Use
(ImageIO) Image layer saved to C:/Users/scanberg/Picture
(CanvasProcessor) Canvas layer exported to disk: C:/Use
(ImageIO) Image layer saved to C:/Users/scanberg/Picture
(CanvasProcessor) Canvas layer exported to disk: C:/Users/scanberg/Pictures/error/canvasgl3-20150128_214905.png
```

Figure 1: An example of a workspace in Inviwo.

2.1.1 Data Source

The flow of data usually begins with some type source. The data sources in Inviwo are modeled as simpler versions of processors and only has an output and some set of properties to control it. The properties depend on the type of source that it represents, for example the property *Color* found in the light source, controls the spectral distribution of the light. Some of the common types of sources can be seen listed bellow:

- Volume Source
- Geometry Source
- Light Source

2.1.2 Processor

A processor represents the key component in the workflow of Inviwo. It has a number of in- and out- ports that are displayed on the top and bottom of the processor and can be connected to other components in the scene. When the processor is active, the input data is collected from the in ports and is processed generating some type of output data that is fed to the out ports. The behavior of the processors can be changed by a set of properties displayed in a property editor. A connection between two ports is displayed as a line from the out port of one processor to the respective in port on the other processor.

Properties The properties in the property editor act as an interface to control some type of underlying data structure that is used by the processor. There are simpler properties like sliders and color pickers that control integers, floats and vectors of floats. More exotic properties such as the Camera-property are built as compositions of simpler properties. Some examples of common properties are listed bellow:

- Slider
- Color picker
- Camera
- Transfer Function
- Drop List

Linking Properties Sometimes the property of a processor has to be equal to some property found in another processor, this is achieved by linking the properties. This essentially means that if one of the properties is changed, this change is propagated to the other one. The link between processors is displayed by a white dotted line between respective processors property linker displayed on the left and right side of the processor.

3 Physically Based Rendering

The parts that are not explicitly referenced in this section are covered in the book [Pharr and Humphreys, 2010], which has served as the main literature for this project.

Physically based rendering signifies that all the mathematical models and units used in the rendering process are based in physics, and if implemented correctly should provide accurate and realistic images. The accuracy of the resulting images should only be limited by the accuracy of the physical models and the computational time given.

List of definitions

x	Vector describing position.
w	Unit vector describing direction.
ϕ	Flux.
I	Intensity.
L	Radiance.
L_e	Self-Emitted Radiance.
ρ	Bidirectional Scattering Distribution Function.
ρ_r	Bidirectional Reflectance Distribution Function.
ρ_p	Phase Function.
σ	A quantity representing probability per unit length.
T_r	The beam transmittance.
n	Index of refraction.
$\int_S$	Integration over the domain of the Hemi-Sphere.
$\int_{S^2}$	Integration over the domain of the Unit-Sphere.
$\|\cos\theta\|$	Cosine-Term, defined as $max(0, \|\cos\theta\|)$.

3.1 Basic Radiometry

There are four basic radiometric quantities central to rendering, each of them describing wavelength dependent electromagnetic energy in some form or context. In the context of computer graphics, wavelength dependent means representing the quantities as spectrums stored as vectors in memory. Often as three floats representing RGB, XYZ or whatever color space is being used.

Flux ϕ Flux is the total amount of energy passing through a region of space or surface per unit time (Joules per Second J/s or Watts W).

Irradiance E **& Radiant Exitance** M Radiant exitance is the area density of flux leaving a surface, and irradiance is the area density of flux arriving at a surface. They both have the units of W/m^2.

Solid Angle sr The solid angle is the extension an angle which on the 2D unit sphere equates to 1D radians, transferred to the 3D unit sphere (Figure 2), where a solid angle equates to 2D *steradians* or sr. To put it simply, it is a measure of the area that some object would subtend if projected on the surface of a unit sphere.

Intensity I Intensity is defined as flux density per solid angle, W/sr. This quantity is however only used to specify the spectral distribution of point lights.

Radiance L Radiance is defined as the flux density per unit area, per unit solid angle, W/m^2sr. The definition is a mouthful, but think of it as a ray of light carrying spectral energy. Radiance is constant throughout empty space, therefore it serves as the natural quantity when ray tracing.

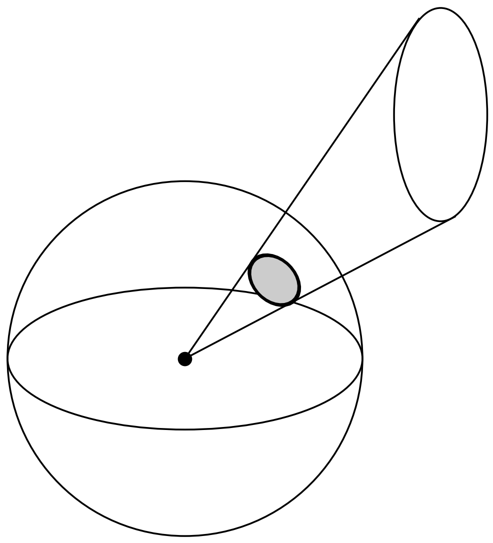


Figure 2: Some object being projected onto the unit sphere, where the solid angle is represented by the gray area.

3.2 Geometric Optics

The model used to describe light and light scattering is that of geometric optics. Light is represented as a spectral quantity traveling in straight lines or rays and is subjected to some basic properties:

Linearity	The combined effect of two inputs is equal to the sum of each inputs effect. Any wave phenomena like interference or diffraction is not handled.
Energy Conservation	When light scatters, the energy after the scatter must be equal or lower than the initial energy. Energy can only be added to the system using emission in various forms.
No Polarization	The polarization of the electromagnetic field is ignored or completely unpolarized.
Wavelength Independency	Each wavelength in the lights distribution is completely independent and may not affect other wavelengths.
Steady State	Light in an environment is assumed to have reached its equilibrium, so its distribution will not change over time.

3.3 The Rendering Equation

In 1986, Kajiya presented the rendering equation [Kajiya, 1986] shown in equation 1, which has formed the basis for modern computer graphics. The paper also presented a method to solve the equation that Kajiya called *Path Tracing*.

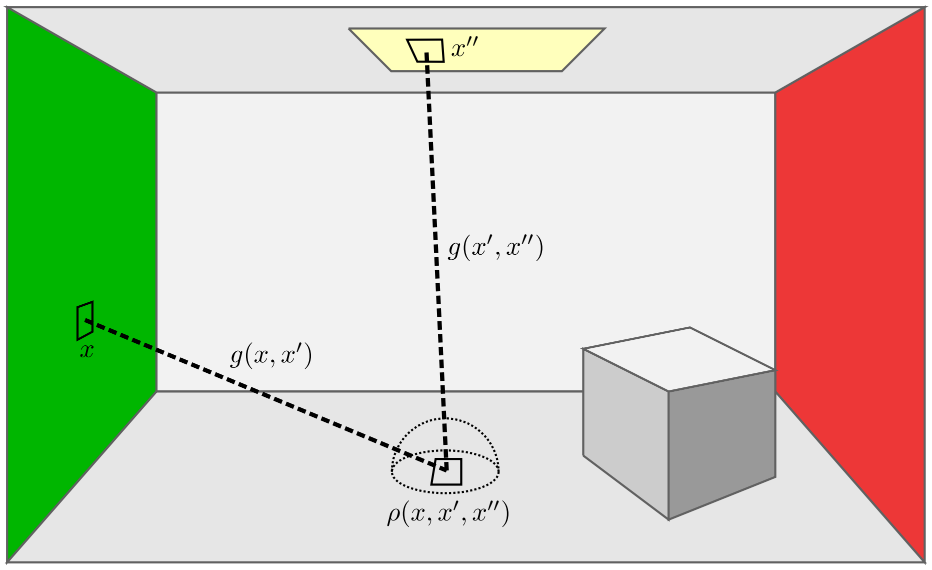


Figure 3: Light being transmitted between two points x and x'' via a third point x' according to the rendering equation.

$$I(x, x') = g(x, x')\left[\epsilon(x, x') + \int_S \rho(x, x', x'')I(x', x'')\mathrm{d}x''\right] \tag{1}$$

The equation is recursive and describes light leaving one point x' and arriving at another point x as a geometrical term g, multiplied by an emitted term ϵ, plus a term that represents incoming light to point x' from all other points in the scene. The last term is formulated as an integral over the entire hemisphere S where each point x'' can contribute using a weighted function ρ that determine the fraction of light coming from point x'' being reflected to point x at the point x'.

The equation is expressed as energy being transferred between points on flat surfaces in a scene using some geometric term to govern their visibility and attenuation with respect to the distance that separate them. However this equation is often reformulated as eq. 2 which uses radiance expressed by points x and unit directions ω. The main reason for using this formulation is that it suits the method used to solve the equation, Path Tracing and this is the formulation that is used in this paper.

$$L(x, \omega_o) = L_e(x, \omega_o) + \int_S \rho(x, \omega_o, \omega_i)L(x, \omega_i)\,|\cos\theta_i|\,\mathrm{d}\omega_i \tag{2}$$

One important limitation of this model is that light is transmitted between flat surfaces that are separated by a vacuum. This means that the rays of light being transmitted between surfaces remain unchanged throughout the extent of the medium being traversed and may only be affected upon impact with another surface. Any volumetric effects, e.g. atmospheric scattering and opaque mediums is not covered by the equation since it requires the possibility of light interacting with the medium it is traversing.

3.4 Path Tracing

Path Tracing is a method developed to solve the rendering equation. It is a ray tracing technique, a technique where rays are traced throughout a scene, hit some surface in it and generates samples from the information of that surface. Path Tracing is usually done *Backwards* in the sense that the it follows the flow of light backwards, which is fine according to the rendering equation and will yield the same result in the end.

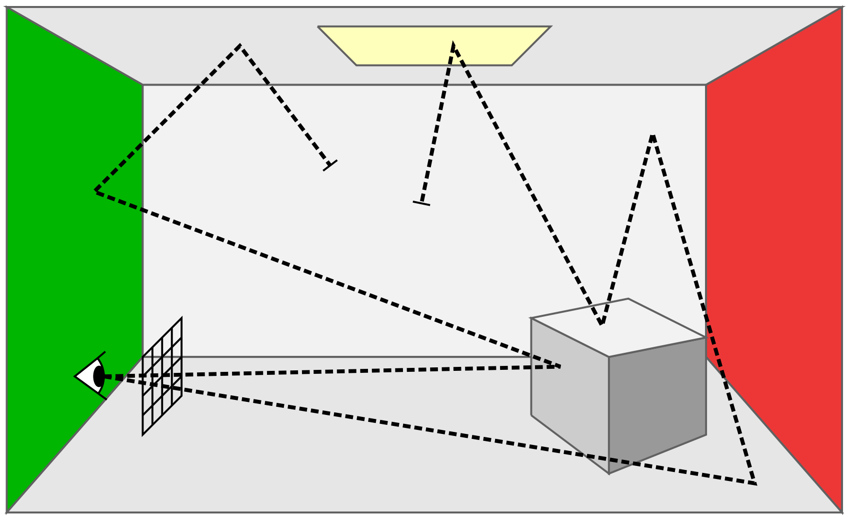


Figure 4: Rays being traced from the eye, through a pixel on the image plane and into the scene. The dotted lines represents the path generated by the algorithm.

The rendering equation can be visualized as a tree structure that at each point of intersection or *scattering point*, represented in the equation by the hemispherical integrand $\int_S$, branch out into multiple branches representing all the possible directions of the hemisphere. This means the tree grows exponentially at each branching point. Instead of trying to sample each possible branch or direction, which is virtually impossible, Path Tracing uses Monte Carlo techniques to sample a single direction as an estimate for all possible directions at this point. But this technique comes at a price: *noise*, which manifests itself as pixels being brighter or darker than they should.

The depth of the tree-structure is in theory unlimited, but for practical reasons the recursion is usually terminated when a certain depth has been reached or when the *throughput* of the path is less than some limit. The section *Terminating The Path* (6.8) covers a method on how this termination be done in an unbiased way, so the resulting contribution of a path is not underestimated.

3.4.1 Direct Light Sampling

The only source of light in the rendering equation is the emission term, which effectively means that the path only gets an increase in radiance when it hits a surface that is emissive, like the surface of a light source. This means that a path may not contribute with any radiance at all, even tough it hits a substantial amount of surfaces, but non of them being emissive. In extension this also means that only area lights can contribute to the light in the scene, opposed to e.g. point lights that are not physically accurate because they do not have a surface area, but are still very popular in rendering. In order to tackle both of these issues, paths not contributing and lights without surface area not being able to contribute, the concept of direct light sampling is introduced.

Direct light sampling is a technique that is common in path tracing and essentially replaces the emissive term with one or more additional *paths* to sample at each branching point. The idea behind it can be expressed as: Instead of waiting for the path to hit an emissive surface, you go straight to the source and sample it. These *paths* do not really have any branches, so it is more correct to call them rays, in fact they are known as *Shadow Rays* as they effectively determine if light reaches the point or not. These Shadow Rays can be seen in figure 5 as dotted lines that are traced from the scattering points in the scene to the light source(s). One important thing here is that in order for this technique to work, each visible light source from the scattering point must have some probability of being sampled by the Direct

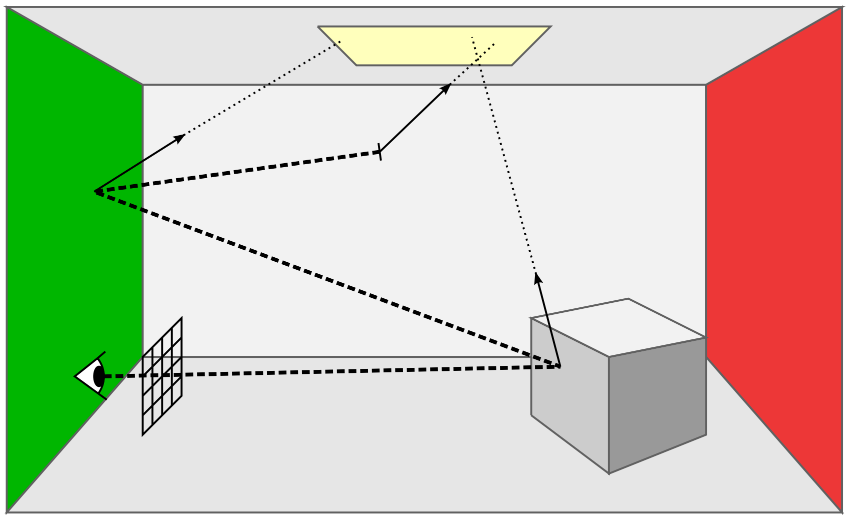


Figure 5: Rays being traced from the eye, through a pixel on the image plane and into the scene. The dashed lines represents the path generated by the algorithm and the dotted lines represents *Shadow Rays*.

Light sampler, otherwise the algorithm would introduce a bias.

$$L(x,\omega_o) = L_e(x,\omega_o)^* + \int_S \rho(x,\omega_o,\omega_i)\,[L_d(x,\omega_i) + L(x,\omega_i)]\,|\cos\theta_i|\,\mathrm{d}\omega_i \tag{3}$$

$$L_e(x,\omega_o)^* = \begin{cases} L_e(x,\omega_o) & \text{if depth} = 0 \text{ or previous sample} = \text{specular reflection} \\ 0 & \text{if } \textit{otherwise} \end{cases}$$

Equation 3 shows the Rendering equation reformulated to account for Direct Light Sampling with an added term L_d that represents the contribution from Direct Light. Notice that the emission term is not completely gone from the equation, but is changed to a conditional statement. This is to handle the situation where an emissive surface is the first surface hit by the path from the eye, since it would be the only opportunity to include emission from directly visible objects. There is a second condition where emission is included, that is when the previous sample in the path was from a *Perfect Specular Reflection*. This is thanks to the delta function in the BRDF that cannot possibly contribute with light from any sampled direction other than the perfect reflection direction.

3.4.2 Bidirectional Path Tracing and other Methods

The Bidirectional Path Tracer [Lafortune and Willems, 1993] is an extension to the concept of Direct Light Sampling. The idea behind is to let the light propagate a path throughout the scene and then connects this light path to the view path. It may not be immediately obvious why this should provide any additional benefits from the Direct Light Sampling technique, but if the light source is obstructed by some geometry, e.g. a lamp shade, the Shadow Ray will probably not reach the actual light source. Figure 6 shows the algorithm where a view path and then a light path is created and then attempts to connect these two paths.

There exists more methods, e.g. Photon Mapping [Jensen, 2001] and Metropolis Light Transport [Veach and Guibas, 1997], that will not be covered nor introduced in this paper, but the general idea behind these algorithms remains the same, connecting rays into paths throughout a scene by using probability.

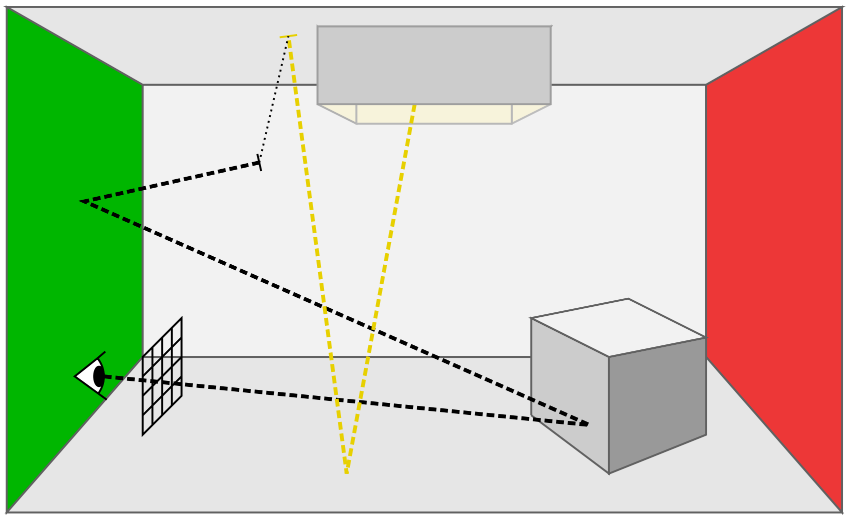


Figure 6: A view-path (dashed black) being traced from the eye into the scene as well as a light-path (dashed yellow) being traced from the light into the scene. The paths are then connected by a Shadow Ray (dotted black).

3.5 Surface Reflection

A reflection occurs when there is an abrupt change in velocity of light traveling through some medium. The Index Of Refraction n of a material specifies the factor of how much the speed of light is slowed down when propagating through that medium compared to vacuum. It is on the interface between the two media with different refractive indices that the reflections occur.

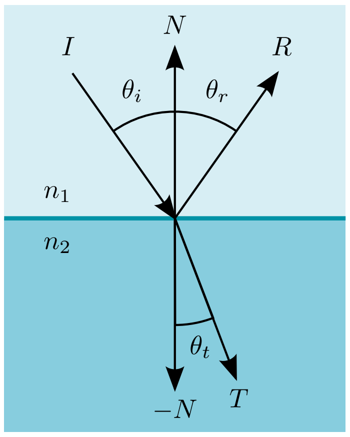


Figure 7: Incoming light I being reflected into R and transmitted into T at an interface between two media with the refractive indices of n_1 and n_2.

3.5.1 Fresnel Equations

The fresnel equations (eq. 4) describe the fraction of energy that gets reflected when light passes through an interface between two different media with the refractive indices of n_1 and n_2. The equations are expressed as two equations, each representing a polarized component of the light with respect to the orientation of the interface, a perpendicular component R_s and a parallel component R_p. Each describe the fraction of energy that gets reflected with respect to the incident angle θ_i.

$$R_s = \left| \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t} \right|^2 \tag{4a}$$

$$R_p = \left| \frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i} \right|^2 \tag{4b}$$

If the light is unpolarized, as in the case of geometric optics, the average of the equations can be used to describe the total fraction of reflective energy.

$$R = \frac{R_s + R_p}{2} \tag{5}$$

The portion of the light that is not reflected but instead gets refracted or transmitted as T into the other medium can be derived by the assumption that the process is energy conserving. The angle of the transmitted ray θ_t can be calculated by using *Snell's Law* (eq. 6).

$$T = 1 - R$$

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{n_2}{n_1} \tag{6}$$

3.5.2 Schlick's Approximation

Schlick present an approximation [Schlick, 1994] to the unpolarized reflectance R (eq. 5) and is shown in equation 7. This approximation is used extensively in the field of rendering because of its low computational cost. The term R_0 is usually precomputed and stored with the definition for the material.

The approximation works quite well, but is limited to non-metals or dielectrics. This is because the refractive indices of metals or conductors are complex numbers, something that this model does not take into account. There exists more sophisticated models [Lazániy and Szirmay-Kalos, 2005] that include better approximations for conductors, but depending on the circumstances, Schlick's model may still be *good enough*.

$$R(\theta_i) = R_0 + (1 - R_0)(1 - \cos\theta_i)^5 \tag{7}$$

$$R_0 = \left(\frac{N_0 - N_1}{N_0 + N_1} \right)^2$$

3.6 Bidirectional Scattering Distribution Function, *BSDF*

$$\rho(x, \omega_o, \omega_i)$$

The BSDF is the concept of a function ρ that describes the total fraction of light coming from a direction ω_i that gets scattered into direction ω_o at a given point x. It is described as concept, because it is a collection of bidirectional distribution functions that each represent one component/layer of the different type of ways that incoming light can be scattered into the outgoing light. In its essence, the BSDF defines the reflective properties of a material or surface. The term bidirectional signifies that the functions input and output directions ω_i, ω_o are interchangeable and the function will yield the same result when the directions are swapped, an invaluable property when it comes to ray tracing.

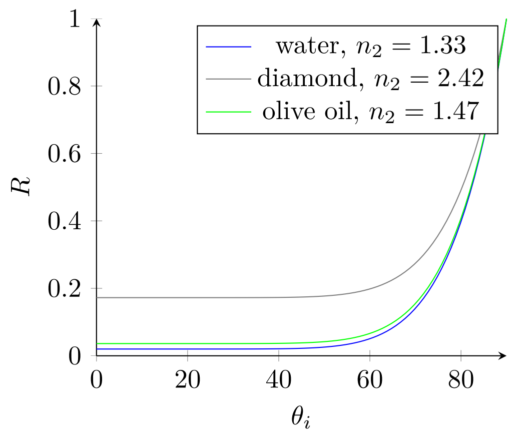


Figure 8: The reflectance R shown for different dielectric media in vacuum ($n_1 = 1$).

The BSDF of a material can be created using some different approaches such as measuring, simulating or using a phenomenological model, this paper will only cover the last option. Phenomenological models are a set of equations or functions that models real-world surfaces using qualitative properties and can produce very convincing results. Most functions used in computer graphics and the ones covered in this paper fall into this category.

There are generally three types of Bidirectional Distribution Functions, the Bidirectional Reflectance Distribution Function (BRDF), the Bidirectional Transmittance Distribution Function (BTDF) and Phase Functions. The BTDF represents the light that gets refracted or transmitted into the surface, but is often implemented as a BRDF with a flipped surface normal and scaled to match the amount of light that gets transmitted. Therefore only BRDFs and Phase functions will be covered in this paper.

3.7 Bidirectional Reflectance Distribution Function, *BRDF*

This section is only intended to give a glimpse of the vast research area of phenomenological BRDF models. The term *Reflectance* in BRDF tells us that it is an representation of reflected light and this paper will use the notation of $\rho_r(x, \omega_o, \omega_i)$ to signify it.

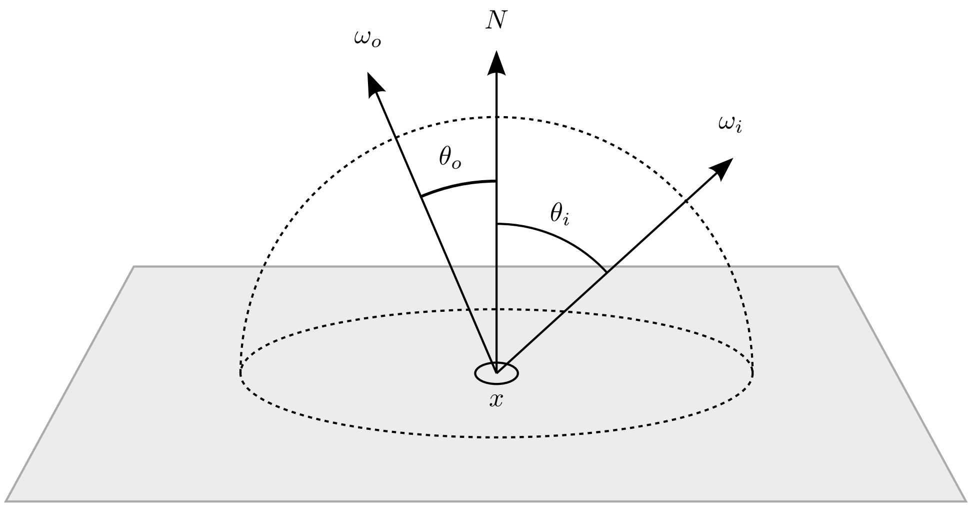


Figure 9: The parameter space for a BRDF at the point x: A hemisphere oriented around the surface normal N with directions ω_i & and ω_o.

By studying real-world materials, it is possible to understand the basis for the BRDF. Materials are often divided into two categories, *conductors* and *dielectrics*. Most of the materials we encounter on a day to day basis are dielectrics, the lemon and the wooden table seen in figure 10 are examples of dielectrics whilst the teaspoon, made of stainless steel is a conductor.

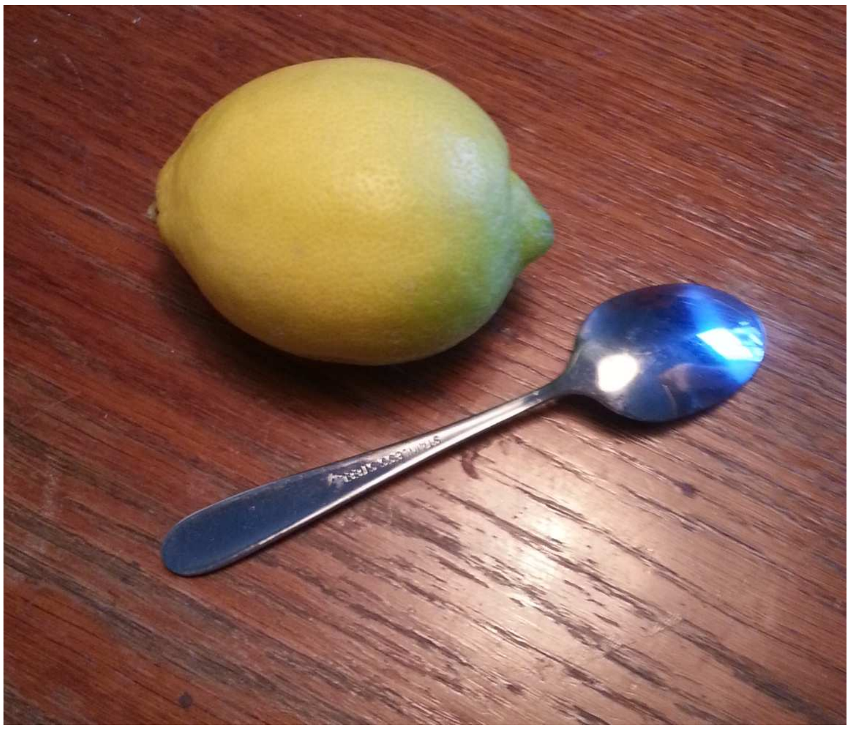


Figure 10: Three very different materials: lemon, stainless steel and wood.

From a macroscopic view there are two general types of surface reflection components, a *diffuse* and a *specular*. The diffuse component is the portion of light that gets transmitted a bit into the surface, bounces a couple of times possibly getting its spectral distribution altered and then exits the surface in a random direction. The specular component however does not enter the surface, instead it gets reflected in a direction somewhere around the perfect reflection with its spectral distribution more or less unaffected. Dielectric materials generally have both a diffuse component and a specular component, with transparent materials like glass as an exception, in those materials the BTDF models the portion of light entering the material. Conductors however do not have any noticeable diffuse component, any energy that gets transmitted into the surface is absorbed and converted into heat. The only significant contribution to the outgoing energy from conductors is by the specular component, and is generally much stronger than the counterpart found in dielectrics.

3.7.1 Diffuse Reflection

The simplest possible diffuse reflection model is the *Lambertian Model* (eq. 8), it assumes the surface to be a ideal diffuse reflector. This means that any light that hits the surface will be scattered equally in all directions and is scaled with a spectral constant R_d to accommodate for any possible change in the spectral distribution. There exists more sophisticated diffuse reflection models [Oren and Nayar, 1994], but the Lambertian model serves as a good enough approximation in most cases and is computationally very inexpensive.

$$\rho_r(x, \omega_o, \omega_i) = \frac{R_d}{\pi} \tag{8}$$

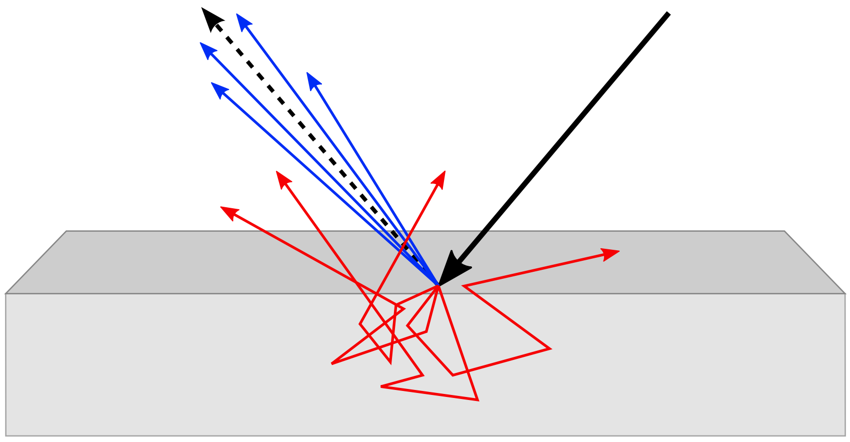


Figure 11: Light (black) being reflected into a diffuse component (red) and a specular component (blue)

3.7.2 Perfect Specular Reflection

The simplest possible specular reflection function is the *Perfect Specular Reflection* (eq. 9) which can be seen in figure 11 as the black arrow with a dashed line. This function simply reflects the incoming light with respect to the surface normal with a possible scale in the spectral distribution according to R_s. Since there is only one perfect reflectance direction per incoming ray, this means that the function is a delta-function that only contributes in the perfect reflection direction.

$$\rho_r(x, \omega_o, \omega_i) = \begin{cases} R_s & \text{if } \omega_o = Reflect(\omega_i) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

3.7.3 Glossy Reflection

Several methods have been developed to accommodate for imperfect reflections or glossy reflections represented as the blue arrows in figure 11. Many of them seem to be based upon the popular Cook-Torrance Microfacet Model [Cook and Torrance, 1981]. The Microfacet Model is shown in eq. 10 and approximates surfaces by using collections of perfectly smooth microscopic mirrors.

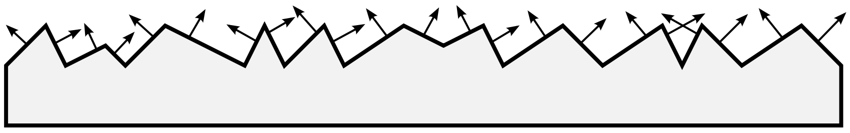


Figure 12: Microfacets of a surface with their respective normals marked as arrows.

The statistical distribution term $D(\omega_h)$ gives the probability of a microfacet having the orientation ω_h and is usually controlled with a parameter called *Roughness* or *Shininess* that controls the sharpness of the distribution. The geometric attenuation term $G(\omega_o, \omega_i)$ models self-shadowing or masking and the last fresnel term $F_r(\omega_h)$ models the microfacets ability to reflect light, where Schlick's Approximation (eq. 7) is often used. What makes this model so popular is the versatility of the model itself because it is comprised of interchangeable terms. The original implementations of the functions D and G is not covered here but is described in the original paper.

$$\omega_h = \frac{\omega_i + \omega_h}{||\omega_i + \omega_h||}$$

$$\rho_r(x, \omega_o, \omega_i) = \frac{D(\omega_h)G(\omega_o, \omega_i)F_r(\omega_o)}{4 \cos\theta_o \cos\theta_i} \tag{10}$$

4 Volume Rendering

The parts that are not explicitly referenced in this section are covered in the book [Pharr and Humphreys, 2010], which has served as the main literature for this project.

There are three different types of processes that may affect the distribution of radiance in an environment with participating media, Absorption, Emission and Scattering. By looking at a slice of differential length along a ray with incident radiance L_i and outgoing radiance L_o the effects of each process can be shown.

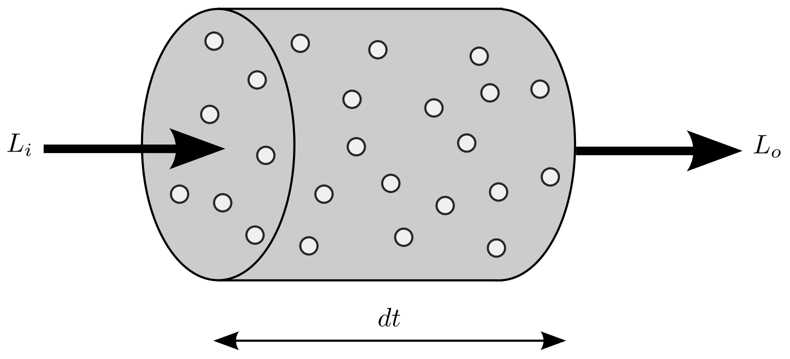


Figure 13: A study in microscopic scale of light interacting with a medium of particles.

4.1 Absorption

The process of radiance being converted to some other form of energy, typically heat, due to collisions with particles that does not scatter the light but instead absorbs it. The parameter σ_a determines the probability of this process.

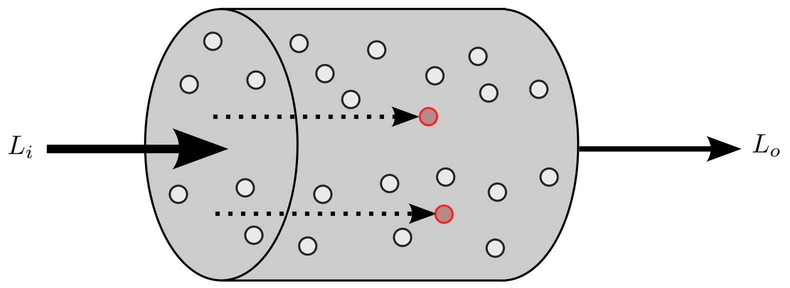


Figure 14: Absorption.

$$\mathrm{d}L_o(x, \omega) = -\sigma_a L_i(x, \omega)\mathrm{d}t \tag{11}$$

4.2 Emission

The process of energy being added to the distribution from luminous particles. Examples of this would be burning flame or the excited gas inside a neon lamp.

$$\mathrm{d}L_o(x, \omega) = L_e(x, \omega)\mathrm{d}t \tag{12}$$

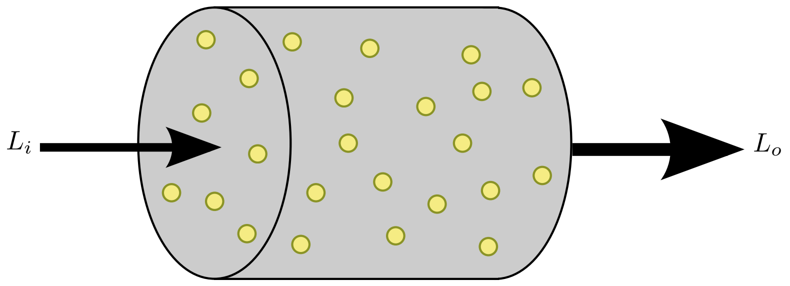


Figure 15: Emission.

4.3 Scattering

The process of light being scattered into new directions due to collisions with particles. This is a two-way process and can both decrease the radiance by *Out-Scattering* and increase the radiance by *In-Scattering*. The parameter σ_s determines the probability of a scatter to occur.

4.4 Out-Scattering & Extinction

Out-Scattering is handled just like absorption but is controlled with the parameter σ_s. *Absorption* and *Out-Scattering* are usually combined into one single parameter *Extinction*, σ_t and defines the total loss in energy.

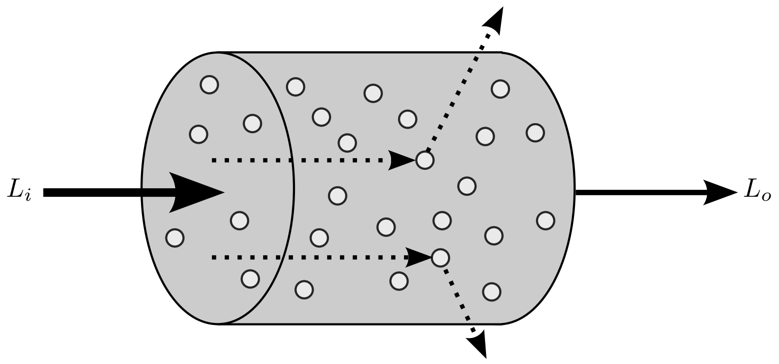


Figure 16: Out-Scattering.

$$\sigma_t = \sigma_a + \sigma_s \tag{13}$$

$$\mathrm{d}L_o(x, \omega) = -\sigma_t(x, \omega) L_i(x, \omega) \mathrm{d}t \tag{14}$$

4.5 Transmittance

The differential equation for extinction, eq. 14, can be solved to find an expression for the *transmittance*, the fraction of radiance that is transmitted between two points.

$$T_r(x_1 \to x_2) = e^{-\int_0^d \sigma_t(x_1 + t\omega, \omega)\mathrm{d}t} \tag{15}$$

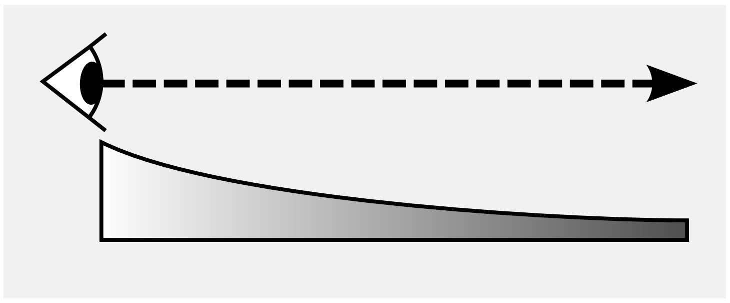


Figure 17: Transmittance in some participating media between the eye and the points along the view ray.

The negated exponent of the transmittance is referred to as the optical density and is denoted τ. If the extinction coefficient is constant, the expression for transmission can simplified into eq. 16, also known as beers law.

$$T_r(x_1 \to x_2) = e^{-\sigma_t d} \tag{16}$$

Because of its nature, the transmittance (eq. 15) is multiplicative along a ray. In practice this means that if the points x_1, x_2 and x_3 lie on the same line and in that order, the transmittance can be written as seen in eq. 17.

$$T_r(x_1 \to x_3) = T_r(x_1 \to x_2)T_r(x_2 \to x_3) \tag{17}$$

4.6 Single Scattering Albedo

Single scattering albedo is a term that shows up every now and then in the field of volume rendering. Even though not used explicitly in the equations of this chapter it does deserve to be mentioned. It is defined as the ratio of scattering efficiency to extinction efficiency and can be seen in the case of light transport as the fraction of light that gets scattered as opposed to absorbed. It is a spectral unitless quantity that in the case of unity would result in none the energy being lost due to absorption as opposed to an albedo of zero that would absorb all the energy.

$$\alpha = \frac{\sigma_s}{\sigma_t} \tag{18}$$

4.7 In-Scattering

This process models all incoming energy (from all directions ω') that is scattered into the direction of the ray (ω), increasing the energy.

$$\mathrm{d}L_o(x,\omega) = \sigma_s(x,\omega)\int_{S_2} \rho_p(x,\omega',\omega)L_i(x,\omega')\mathrm{d}\omega'\mathrm{d}t \tag{19}$$

4.8 Source-Term

The terms that contribute to the net increase of radiance, *Emission* and *In-Scattering*, is typically bundled together in a *Source-Term*, $S(x,\omega)$. This is to generalize the expression of the term that models

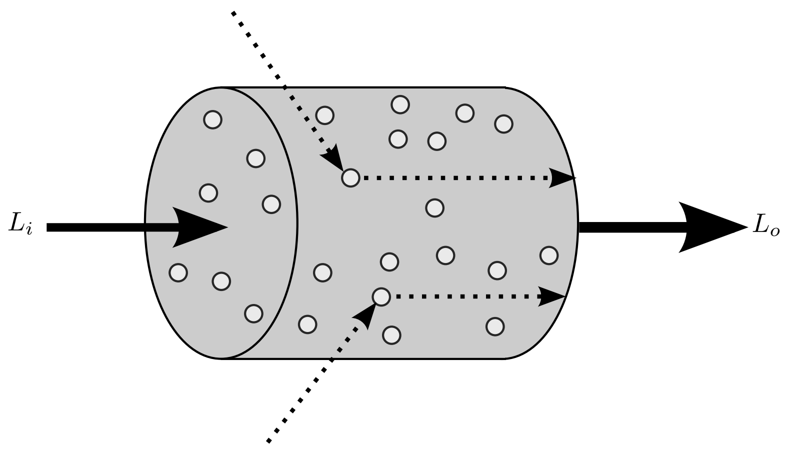


Figure 18: In-Scattering.

the net-increase in light. The fraction of light that is scattered into the ray is determined by $\rho_p(x, \omega', \omega)$, the *Phase Function.*

$$S(x,\omega) = L_e(x,\omega) + \sigma_s(x,\omega) \int\limits_{S_2} \rho_p(x,\omega,\omega')L_i(x,\omega') \tag{20}$$

4.9 Phase Function

The phase function can be seen as the volumetric analogue to the BRDF used for surfaces. However there are two key differences, the first is that phase functions covers the entire unit sphere, compared to the BRDF that only covers the hemisphere and the second is that the phase function have a normalization constraint. This normalization constraint can be seen in (eq. 21) and with this condition fulfilled, the phase function effectively define a probability distribution for scattering in a particular direction. In this paper the general phase function is given the subscript p for phase.

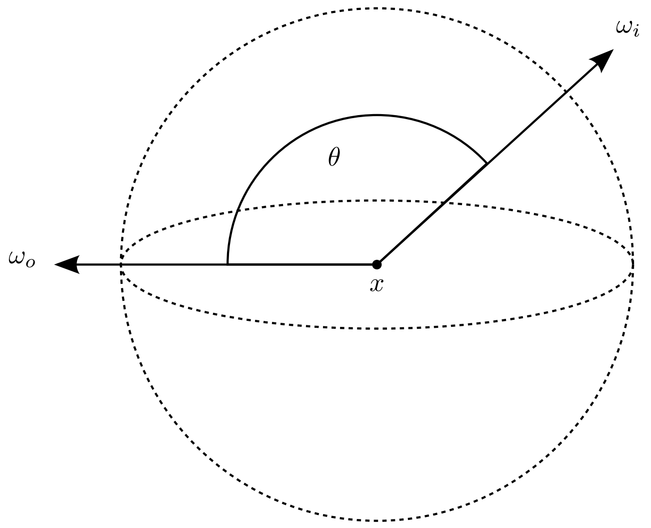


Figure 19: The parameter space for a phase function at the point x: A unit sphere with directions ω_i & and ω_o.

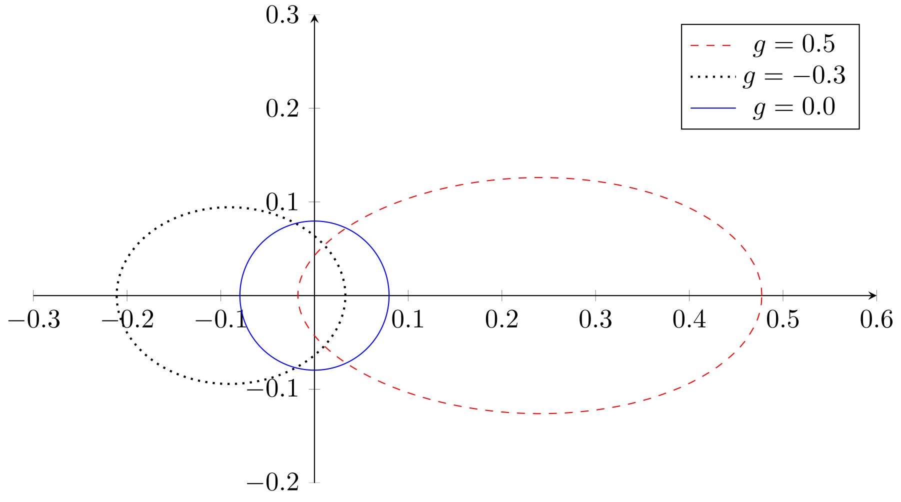


Figure 20: Henyey Greenstein Phase Function plotted with three different values of g. The plots are polar where the angle is θ and the radius is the resulting probability of that angle.

$$\int_{S_2} \rho_p(x, \omega, \omega')\mathrm{d}\omega' = 1 \tag{21}$$

The trivial phase function is the one where each direction of the unit sphere is equally probable, a volumetric analogue of the *Lambertian Model.* It is known as the *Isotropic Phase Function* and can seen in equation 22.

$$\rho_{iso}(x, \omega, \omega') = \frac{1}{4\pi} \tag{22}$$

One particular phase function that is very common is the Henyey-Greenstein phase function Henyey and Greenstein [1941], shown in eq. 23. It uses a single parameter called the asymmetry parameter to control the scattering distribution. The asymmetry parameter g has a range between $[-1, 1]$, where $g = -1$ would be perfectly back scattering, that is all light gets reflected back in the direction it came from and $g = 1$ would be perfectly forward scattering. With a value of $g = 0$ the function collapses into the isotropic phase function with every direction being equally probable.

$$\rho_{hg}(x, \omega', \omega) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{23}$$

Notice that unlike the BRDFs, these Phase Functions do not alter the spectral distribution of the light, they only define the probability of the scattering directions.

4.10 Volume Rendering Equation

By considering each of the processes that may affect the distribution of radiance, it is possible to formulate an equation for the overall differential change in radiance. This equation, often referred to as the *Radiative Transmittance Equation (R.T.E.)* or *The Equation of Transfer* and forms the foundation of volumetric rendering. The first term represents the attenuated contribution of some starting point, perhaps on some surface or background. The second term represents the attenuated contribution from the medium itself by integrating $x' = x_0 + t'\omega$ from point x_0 at $t' = 0$, to x at $t' = t$.

$$L(x, \omega) = T_r(x_0 \to x)L_0(x_0, -\omega) + \int_0^t T_r(x' \to x)S(x', -\omega)\mathrm{d}t' \tag{24}$$

$$S(x,\omega) = L_e(x,\omega) + \sigma_s(x,\omega) \int\limits_{S_2} \rho_p(x,\omega,\omega')L(x,\omega')\mathrm{d}\omega'$$

4.11 Volume Rendering In Scientific Visualization

In the field of Scientific Visualization, volumetric rendering can be seen as a set of techniques to create a two-dimensional projection of a three-dimensional data set. These data sets are often created using medicinal scanners that sample in a regular pattern, often two-dimensional slices. Each slice then serves as a layer in a three-dimensional stack which makes a full data set comprised of discrete voxels. The voxels usually only store a single scalar value that can be seen as the *density* at that particular position (fig. 21). The density is a relative measure and may differ depending on how the data is acquired. The density values are then possibly scaled and quantized to fit a precision of typically 8- to 16-bits, that is 256 to 65536 possible values, to reduce the occupied memory. Since the data is stored as a single scalar density value, this means that some areas may have the same density value even though they are fundamentally different. It may not be obvious why this would be a problem at first glance, but the density value is commonly used as input to a function that maps density into optical properties, a *Transfer Function*. Therefore two different regions that have different visual properties in the real world but have the same density value, will ultimately be mapped into the same optical properties by the transfer function.

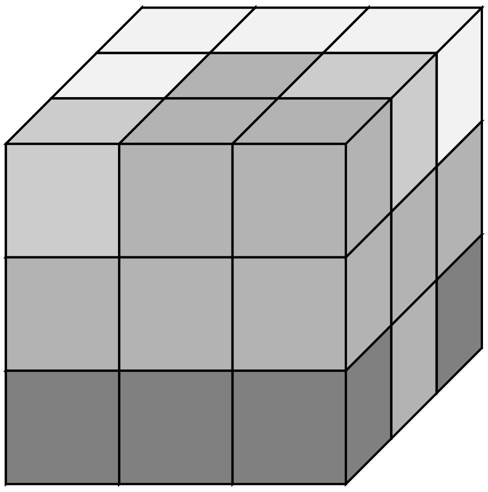


Figure 21: A volumetric data block containing 3^3 voxels with their scalar density values shown in shades of gray.

4.12 Volume Rendering In Computer Graphics

In Computer Graphics, volumetric rendering is usually used as a compliment to surface rendering, to produce scattering effects in the voids between the surfaces. It is common that only optically thin mediums like air, clouds and water are used as opposed to optically dense mediums like the soft tissue of a human, a slab of marble or even milk. Since the volume medium is often optically thin, simplifications to the rendering model can be used without any major visible degradations of end result. Another difference in Computer Graphics compared to Scientific Visualization is how the data is acquired. In this case the data is often modeled using some function, be it constant, exponential falloff with respect to height (e.g. earth's atmosphere) or even simulated (e.g. smoke). Since the data have very similar optical properties, the properties are often simplified to a set of constant parameters plus a varying opacity parameter.

4.13 Emission Absorption Model

The Emission Absorption Model is a rendering model that partly solves the Volume Rendering Equation by simplifying the source term to only cover volumetric emission, $S(x,\omega) = L_e(x,\omega)$. The resulting expression can be seen in eq. 25. In this model the only contribution of light comes from some background light L_0 or the volumetric emission L_e that gets attenuated according to the transmittance T_r along the ray.

$$L(x,\omega) = T_r(x_0 \to x)L_0(x_0,-\omega) + \int_0^t T_r(x' \to x)L_e(x',-\omega)\mathrm{d}t' \tag{25}$$

4.14 Single Scattering Model

A more sophisticated approach that also partly solves the Volume Rendering Equation is the Single Scattering Model. The method replaces the recursion that occurs via $L(x,\omega')$ with direct light sampling, $L_d(x,\omega')$. This enables photons coming from external light sources to contribute to the light distribution in the scene by scattering (once) in the volume. At this scatter the photons release some of its energy that contributes into the direction of the view ray.

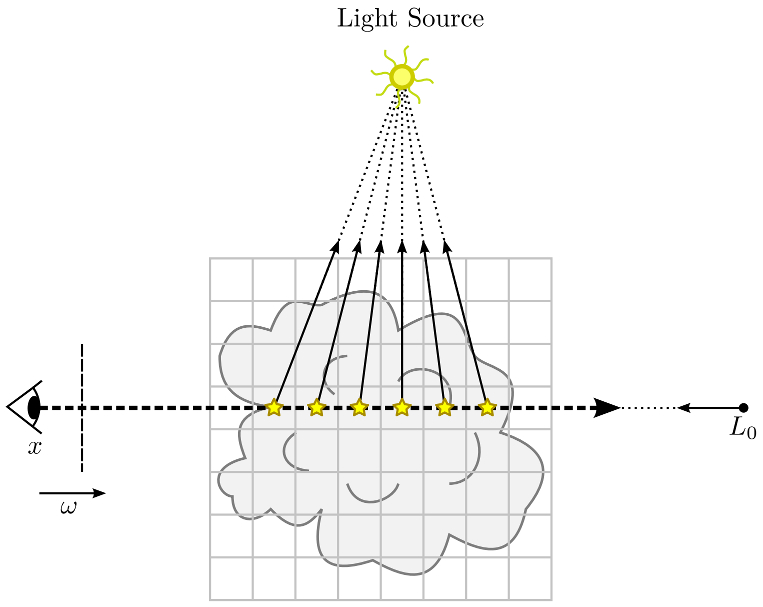


Figure 22: Direct lighting contributes to the view ray via single scattering in a participating media.

$$L(x,\omega) = T_r(x_0 \to x)L_0(x_0,-\omega) + \int_0^t T_r(x' \to x)\left[L_e(x,\omega) + \sigma_s(x,\omega)\int_{S_2} \rho_p(x,\omega,\omega')L_d(x,\omega')\mathrm{d}\omega'\right]\mathrm{d}t' \tag{26}$$

Since the photons still have some energy left after the first scatter event this leads to the overall light distribution being underestimated when compared to the full model.

4.15 Volumetric Path Tracing

Lafortune introduced a method [Lafortune and Willems, 1996] that extends the techniques of Path Tracing to also cover participating media. The method samples a distance from the transmittance along a ray. If this distance is less than the distance of the nearest surface intersection along the ray, a scatter occurred in the media and the path is evaluated from this scatter point rather than the point on the surface. The equation is written with the assumption that the participating medium is homogeneous and does not emit light for simplicity.

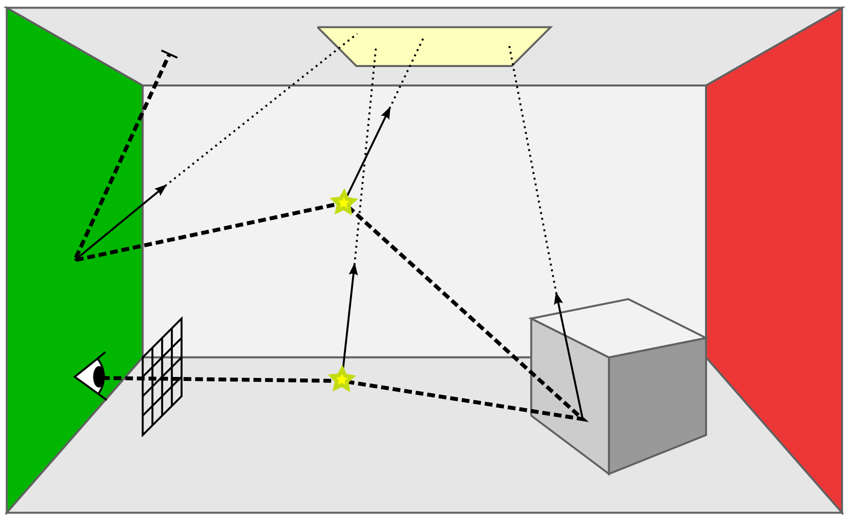


Figure 23: A view-path (dashed) being traced from the eye into the scene where scatter events (stars) occur in the media.

So the path tracing algorithm is basically extended with the probability of light scattering in the media. This means that the fundamental algorithm remains the same, but in the event of scattering, a phase function will be used rather than a surface model. The extended equation for path tracing with the probability of scattering is expressed in eq. 27.

$$L(x,\omega) = \int_0^\infty \left[s < d \ ? \ L_{vol}(x + s\omega, -\omega) : L_{surf}(x + d\omega, -\omega)\right] \sigma_t e^{-\sigma_t s} \tag{27}$$

$$L_{vol}(x,\omega) = \int_{s^2} \frac{\sigma_s}{\sigma_t} \rho_p(x,\omega,\omega') L(x,\omega') \mathrm{d}\omega'$$

$$L_{surf}(x,\omega) = L_e(x,\omega) + \int_s \rho_r(x,\omega,\omega') L(x,\omega') \left|\cos\theta'\right| \mathrm{d}\omega'$$

The extension to non-homogeneous does not change the formulation of the equation, the only affected parameter is the distribution from which the samples are picked that is $\sigma_t e^{-\sigma_t s}$ that becomes $\sigma_t(x + s\omega) T_r(x, x + s\omega)$ when moving to a non-homogeneous medium. When using this formulation, the expression for L_{vol} and L_{surf} are quite similar if the emission term is disregarded. This similarity will be exploited in the implementation of the BSDF.

5 Monte Carlo Integration

Monte Carlo integration is an numerical integration technique that uses random numbers to compute an estimate of a definite integral. The technique was developed by Stanislaw Ulam [Metropolis and Ulam, 1949] when working on nuclear weapon projects at Los Alamos National Laboratory in 1940s. The name *Monte Carlo* came from a casino called *Monte Carlo Casino*, where his uncle used to gamble.

List of definitions

X	A random variable.
$P(x)$	Cumulative Distribution Function, CDF, the probability of a random variable being less than or equal to some value x, $P(x) = Pr(X \leq x)$.
$p(x)$	Probability Density Function, PDF, the relative probability of a random variable taking on a particular value. It is the derivative of the random variable's CDF, $p(x) = \frac{\mathrm{d}P(x)}{\mathrm{d}x}$.
ξ	The canonical uniform random variable, it can take on any value in its domain $[0, 1[$ with equal probability.

$$Exp[f(x)] = \int_D f(x)p(x)\mathrm{d}x \tag{28}$$

The technique is derived from the definition of the expected value $Exp[f(x)]$ (eq. 28) that is defined as the average value of the function $f(x)$ over some distribution $p(x)$ of values over its domain D. In the one-dimensional case a basic Monte Carlo estimator $Est[f(x)]$ with uniform sampling can be generalized as seen in eq. 29, where X_i represents a random sample in the valid interval $[a, b[$. The principle is to randomly sample the integrand and keep track of the mean value of the results. The result is then scaled with the uniform probability of generating a sample within the interval.

$$\int_a^b f(x)\mathrm{d}x$$

$$Est[\int_a^b f(x)\mathrm{d}x] = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i) \tag{29}$$

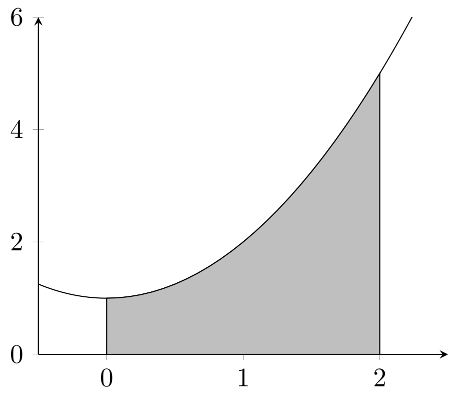


Figure 24: The integral $\int_0^2 (x^2 + 1)\mathrm{d}x$ visualized as the gray area.

An example scenario is shown in figure 24, in this example we would like to estimate the integrand of a simple polynomial $\int_0^2 (x^2+1)\mathrm{d}x$ using Monte Carlo Integration. By using the uniform estimator (eq. 29) we can formulate an estimate for the integrand as seen in eq. 30.

$$Est[\int_0^2 (x^2+1)\mathrm{d}x] = \frac{2}{N}\sum_{i=0}^{N} X_i^2 + 1 \tag{30}$$

By using the estimator seen in eq. 30 and gradually increasing the number of samples N, Table 1 shows how the estimate converges towards the true value with each added sample.

Samples (N)	Estimate (Est)	Difference
1	8.2680	3.6013
10	4.9348	0.2681
100	4.7531	0.0864
1000	4.6675	0.0008

Table 1: The estimate converges towards the true analytical value $\frac{14}{3} = 4.666...$

The error of the Monte Carlo estimator converges at a rate of $O(\sqrt{N})$, which means that in order to cut the error of the estimate in half, you need to quadruple the number of samples. This is the downside of using the Monte Carlo technique, it suffers from diminishing returns. However the convergence rate still holds true regardless of the dimensionality of the integrand, which makes it a powerful tool.

5.1 Improving the Monte Carlo Estimator

The previous section introduced the concept of Monte Carlo integration and a basic estimator, but here the concept is improved by enabling non-uniform sampling. Using eq. 28 as a basis, it is possible to formulate an general estimator for an integral over some one-dimensional domain $[a, b[$ using samples drawn from a *Probability Density Function* $p(X_i)$ as seen in eq. 31.

$$Est[\int_a^b f(x)\mathrm{d}x] = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{31}$$

The choice of a p has to be taken with great care. Its a bit of a dual-edged sword, by using a p that suits the function f the variance can be reduced, however if the p does not suit the function, you will end up with more variance than that of the uniform-estimator. The only restrictions of the function p is that it has to integrate to a value of 1 and needs to be nonzero for all x where $|f(x) > 0|$. If the function p i set to a constant value and apply the restrictions mentioned, it would in the case of the general one dimensional interval of $[a, b[$ result in eq. 32. And when used, the estimator turns into the uniform estimator (eq. 29).

$$p(x) = \frac{1}{b-a} \tag{32}$$

5.2 The Inversion Method

The inversion method uses one or more random variables, and maps these into random variables from a desired distribution. In order to draw samples X_i from an arbitrary PDF $p(x)$, the following steps is taken:

1. Compute the Cumulative Distribution Function (CDF), $P(x) = \int_0^x p(x')\mathrm{d}x'$.
2. Compute the inverse, $P^{-1}(x)$.

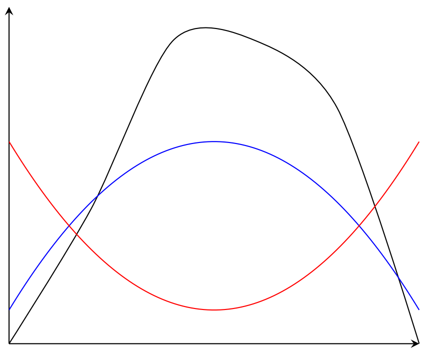


Figure 25: A function f(x) (black) with one PDF that suits f(x) (blue) and one PDF that do not (red).

3. Use a uniformly distributed number ξ and obtain $X = P^{-1}(\xi)$.

As an example and for reference in the latter sections of this paper, the inversion method is applied to an exponential distribution like that of *Beers Law* (eq. 16). First the exponential distribution is normalized to meet the requirement of being a PDF.

$$\int_0^\infty ce^{-ax}\mathrm{d}x = \left[-\frac{c}{a}e^{-ax}\right]_0^\infty = \frac{c}{a} = 1.$$

With the knowledge of $c = a$, the PDF of the exponential distribution can be written as $p(x) = ae^{-ax}$, now the steps of the inversion method can be applied to find $P^{-1}(x)$.

$$P(x) = \int_0^x ae^{-ax'}\mathrm{d}x' = 1 - e^{-ax}$$
$$P^{-1}(x) = -\frac{ln(1-x)}{a}$$
$$X = -\frac{ln(1-\xi)}{a} = -\frac{ln(\xi)}{a} \tag{33}$$

Since ξ represents a uniform random number between [0,1[the term $1-\xi$ can be replaced with ξ.

6 Implementation

The target platform for the implementations is OpenCL, this is to utilize the massive amount of parallel computational power available on modern GPUs. The initial choice of graphics library was that between OpenGL and OpenCL, but the choice fell to OpenCL as some functionality, such as a Random Number Generator, BSDFs and sampling of the BSDFs had already been implemented in *Inviwo* in another project. The code listings shown in the paper are written in a simplified version of the language used when writing kernels for OpenCL, which is very closely related to C.

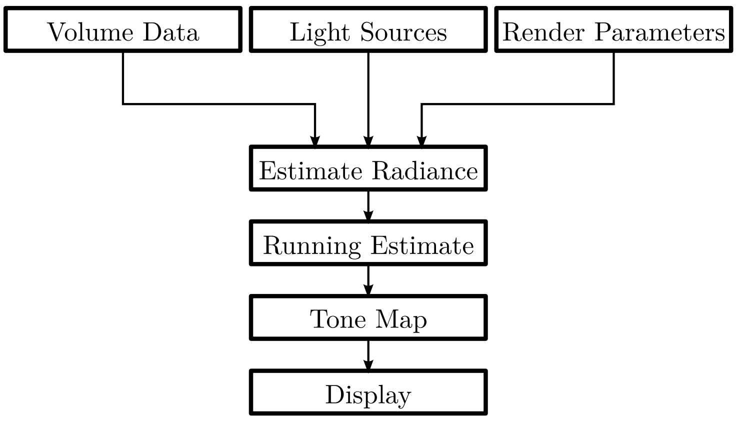


Figure 26: The complete process of rendering to screen.

The process of rendering a pixel onto the screen can be seen in figure 26. A new estimate is created by the Estimate Radiance step, by path tracing the scene defined by the input data. This new estimate is then added to a running estimate which is refined for each new estimate added. The output of the running estimate contains radiometric data that is then tone mapped to fit a signal of [0,1] that can then be drawn onto the screen.

The concept is to run the process continuously for each pixel of the output image and produce many samples. With each new sample the running estimate is refined and converges towards the true value. The process is stopped by the user when the estimate is good enough or has reached a certain amount of samples per pixel.

6.1 Volumetric Path Tracer Processor: Estimating Radiance

Even though the rendering equation is recursive, the actual implementation avoids recursion. The simple explanation is that OpenCL does not support it. The recursion is instead mimicked by a loop where each iteration represents the depth of the recursion.

Features of the Path Tracer

Only volume rendering The scene only consist of the volume itself and the external light sources, no other geometry such as meshes.

No volumetric emission The emission term is left out from the implementation and the only source of light in the scene is external light sources. The explanation for this is that there was no elegant way of implementing the emission without redoing the transfer function, which would mess up compatibility with other processors. One version of the path tracer did have an extra transfer function to control volumetric emission, but it was unintuitive and hard to control and was therefore removed.

External Light Sources As mentioned in the previous point, external light sources are the only sources of lighting in the scene, the type of supported light sources are: Area lights, point lights, directional lights and a single environmental light. The data of all the light sources in the scene are stored in a OpenCL buffer.

Interchangeable BSDF The BSDF, can be changed in the property editor.

Transfer Function The density of the volume is mapped into single scattering albedo α and opacity via a transfer function. Other possibly required parameters for the BSDF are sent as global input parameters.

Variable MAX DEPTH The maximum recursion depth can be set using a slider in the parameters, a depth of 1 would result in a single scatter solution.

6.1.1 Overview

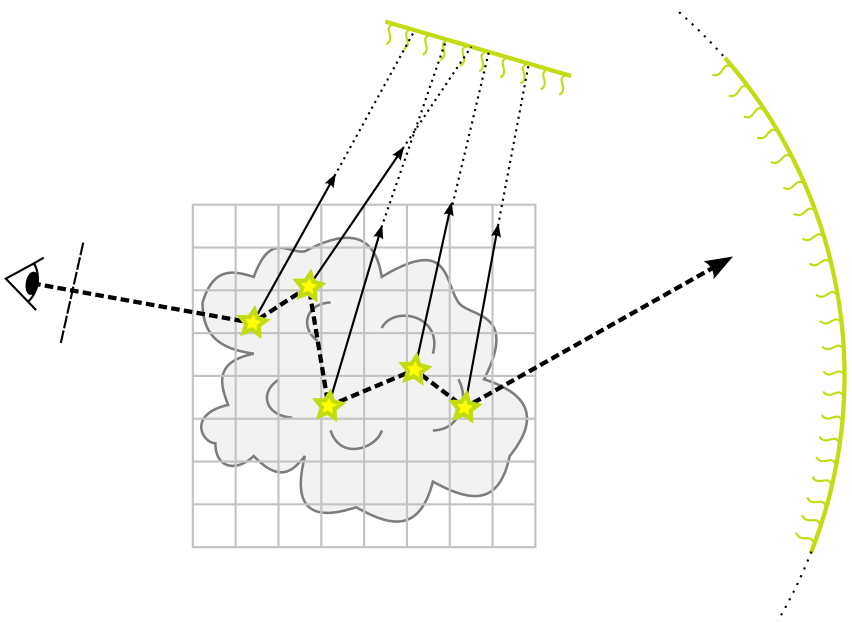


Figure 27: An overview of the implemented algorithm: A view-path is being traced throughout a volumetric object in the scene. Each star represents a scatter event along the view-path, where the explicit light sources are sampled and then a new direction is sampled.

The implementation of the path tracer is based upon the techniques presented in the section Volumetric Path Tracing with an overview of the algorithm is shown in figure 27. First, a ray is fired from the eye, through the pixel and into the scene. A distance is generated along the ray by sampling the transmittance function, this distance represents how far along the ray that a scatter event (marked in the figure as yellow stars) occurs. At the scatter event, the direct light contribution is estimated by sampling the light sources in the scene and then a new direction for the ray is chosen by sampling the BSDF that will serve as an estimate for the indirect light contribution. A new distance is sampled along this new ray direction and the process is repeated until the path is terminated by russian roulette or the path exits the volume. If the ray exits the volume, the contribution from the environment light is added and the path is terminated.

Listing 1: Volumetric Path Tracer

```
float3 EstimateRadiance(float3 x, float3 w) {
  // Resulting radiance L
  float3 L = float3(0.0);          // Resulting Radiance
  float3 tr = float3(1.0);         // Throughput of the path

  for (int i=0; i<MAX_DEPTH; ++i) {
    float d, pdf;                  // Distance and PDF
    d = SampleDistance(x, w);      // Sample a distance along the ray
    if(d < FLT_MAX) {
      x = x + d*w;                 // Update the position

      // Create a scatter event at this position
      ScatterEvent se = CreateScatterEvent(x, w);

      // Add direct light contribution
      L = L + tr * EstimateDirectLight(se, x, -w);

      // Sample a new direction from the BSDF
      float rho = SampleBSDF(se, w, pdf);

      // Check that the there is some reflectance and probability
      if(!IsBlack(rho) && pdf > 0)
        // Scale the throughput to account for the sampled direction
        tr = tr * rho / pdf;
      else
        break;

      if(Terminate(tr)) // Possibly terminate path
        break;
    } else {
      // Add the background radiance
      L = L + Tr * SampleEnvironment(w);
      break;
    }
  }
  return L;
}
```

6.2 Sampling Distance

The term may seem vague, but the concept is to generate a distance along the ray where a scatter event will occur. This is accomplished by using a technique called Woodcock Tracking [Woodcock et al., 1965], a method with origins in the field of nuclear physics. The idea is to simulate the path of a small particle through a medium on its way to a collision or in the case of rendering, a photon on its way to scatter.

The algorithm is based upon generating an underestimated distance using a global mean free path of the medium where a collision may occur, and at this collision determine if it is a true collision or a *virtual* collision. If it is a true collision, the distance has been found and the search is terminated, but if the collision is *virtual*, the mean free path is sampled again and the process is repeated. The probability that determines the if the collision is true or virtual can be seen in eq. 35 and is defined as the ratio between the extinction coefficient σ_t and a global maximum extinction coefficient σ_{max} (eq. 34).

$$\sigma_{max} = max(\sigma_t(x)), x \in \mathbb{R}^3 \tag{34}$$

$$p_{true} = \frac{\sigma_t(x)}{\sigma_{max}} \tag{35}$$

Listing 2: Woodcock Tracking

```
float SampleDistance(float3 x, float3 w) {
  float d, p;           // distance and probability
  do{
    d = -ln(rnd()) / sigma_max(x);
    x = x + d * w;
    if(OutOfBounds(x))
      return FLT_MAX;
    p = sigma_t(x) / sigma_max(x);
  }while(rnd() > p)
  return d;
}
```

The probable collision distance d in listing 2 can be derived by applying the inversion method to the definition of Beers Law, (eq. 33) which is presented in the section *The Inversion Method* (5.2).

The method results an unbiased distance along the ray where a scatter event occurs, which is desirable, but the method in its basic form can be *overly cautious* and severely underestimate the distance between each probability comparison. For example if the extinction coefficient of the volume is large in only a tiny portion of the volume, the σ_{max} will have a large value for the entire region of the volume and will cause the algorithm to take small steps which equates to wasted computions. However more sophisticated methods have been developed [Szirmay-Kalos et al., 2010] where the volume region is subdivided into smaller blocks that each have a local σ_{max}.

6.3 Scatter Event

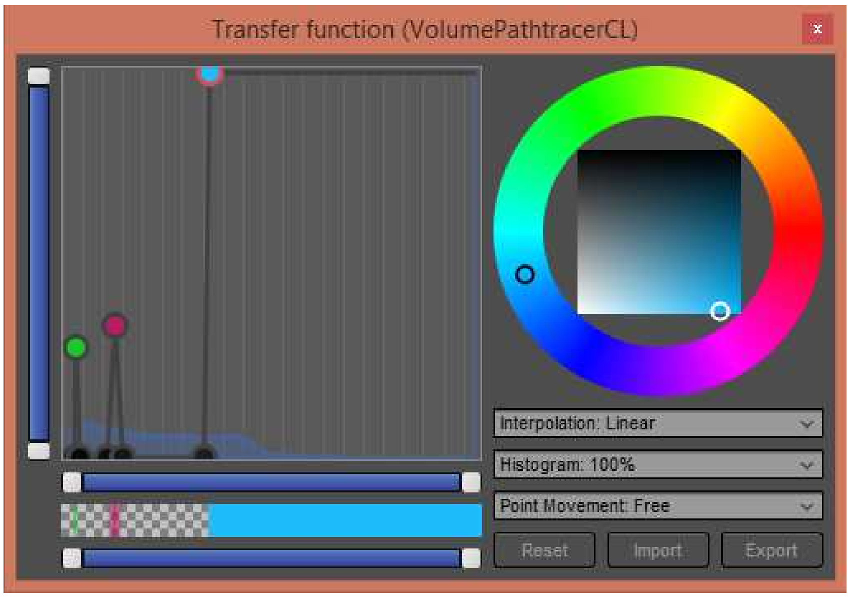
If the sample is not valid, the resulting position of the sampled distance did not end up in the volume. In this case the recursion is terminated and the emission from the Environment Light is added. The contribution from environment light is added at this stage because it is not covered by the Direct Light Sampling.

However if a valid distance has been generated, the position of this sample lies inside the volume and a ScatterEvent object is created using the CreateScatterEvent method. The ScatterEvent object is essentially a collection of optical and geometrical parameters relevant to the point x and direction ω. The main purpose of the object is to sample and derive computationally expensive parameters once, store them in the ScatterEvent and reuse them where they are needed in the calculations. The optical parameters are presented with the transfer function in the next section while some of the geometrical parameters are listed bellow:

- Density
- Gradient & Magnitude
- Volume space coordinates

6.4 Transfer Function and optical parameters

The transfer function is used to map the density value of a volume element into optical properties that is used in the rendering process. It takes a single scalar input and outputs a set of optical properties. The amount of output parameters is determined by the implementation, but the minimum would be 2 parameters, a spectral color and a scalar opacity value, often represented as a vector with 4 values, RGBA with a range of $[0, 1]$. This type of minimalistic transfer function is used in Inviwo, and with the help of a set of global parameters, the required parameters for the volume rendering process can be derived with some limitations.



Interpolation: Linear
Histogram: 100%
Point Movement: Free

Figure 28: The Transfer function used in Inwiwo, x-axis represents density, y-axis represents opacity. The points are placed by the user and represent the color and opacity for the corresponding density value.

Parameters derived from RGBA

σ_t	The extinction coefficient is approximated as a scalar instead of a true spectral quantity and is defined as a function of the value A, $\sigma_t = -ln(1-A)*C$. Where C is the reference sampling interval for the volume, the value 150 is often used.
α	The single scattering albedo (eq. 18) is a straight forward mapping of RGB, $\alpha = RGB$.
σ_s	The scattering coefficient is derived from the albedo α and the extinction coefficient σ_t, by using eq. 18, $\sigma_s = \alpha\sigma_t$.
σ_a	The absorption coefficient, is not explicitly used in the rendering process but can be derived from (eq. 13), $\sigma_a = \sigma_t - \sigma_s$.
R_d	The diffuse spectral scale, is just as single scattering albedo, a straight forward mapping of RGB, $R_d = RGB$.

The Transfer Function is interpolated and stored as a one-dimensional RGBA-texture for fast access and to utilize the hardware interpolation provided by the GPUs when fetching texels. The parameters that are not derived from the Transfer Function are set as global parameters via the property editor in the VolumePathTracer Processor.

Global Parameters

g	The anisotropy parameter.
R_s	The specular spectral scale.
IOR	Index Of Refraction.

6.5 Estimating Direct Lighting

In order to estimate the direct light that falls onto a point in the scene, a uniform sampling strategy is used. This means that each light in the scene has an equal probability of being sampled and that the result has to be scaled with the number of light sources.

Listing 3: Estimation of Direct Light

```
float3 EstimateDirectLight(ScatterEvent se, float3 x, float3 wo) {
  // generate a random light source
  int i = floor(rnd() * NUM_LIGHTSOURCES);
  LightSource ls = GetLightSource(i);

  // Sample the light source
  float3 lx;            // light sample
  float3 wi;            // incoming direction
  float3 Ld;            // light radiance
  float pdf;            // pdf of the light sample

  switch(ls.type) {
    case POINT_LIGHT:
      Ld = SamplePointLight(lx, wi, pdf, ls, x);
      break;
    case DIRECTIONAL_LIGHT:
      Ld = SampleDirectionalLight(lx, wi, pdf, ls, x);
      break;
    case AREA_LIGHT:
      Ld = SampleAreaLight(lx, wi, pdf, ls, x);
      break;
  }

  float3 Tr = EstimateTransmittance(x, lx);
  float3 p = BSDF(se, x, wo, wi);             // calculate the reflectance

  return NUM_LIGHTSOURCES * Tr * p * Ld / pdf;
}
```

6.5.1 Point Lights

Point lights are probably the most commonly used light in computer graphics and serves as an approximation of a small spherical light source that radiates with equal intensity in all directions (isotropic). It is however represented as a point in space without any radius, which is impossible for a view ray to hit, since it has no area. The only way for point lights to contribute to the radiance distribution in a scene is to explicitly sample them. The actual value stored in the variable L for a point light is the intensity I. In order to convert this into actual radiance L, the intensity is divided by the squared distance between the light lx and the evaluated point x.

Listing 4: Point Light Sampling

```
float3 SamplePointLight(float3& lx, float3& wi, float& pdf,
         LightSource ls, float3 x) {
  lx = ls.position;
  wi = Normalize(lx - x);
  pdf = 1.0;
  return ls.L / DistanceSquared(lx, ls.position);
}
```

6.5.2 Directional Lights

Directional lights can be seen as light sources being sampled at an infinite distance, at that distance the rays would be parallel. The sun is often approximated with a directional light in computer graphics,

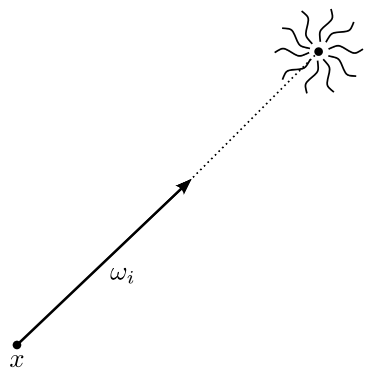


Figure 29: A point light being sampled.

since its so far away from the earth and casts approximately parallel rays of light.

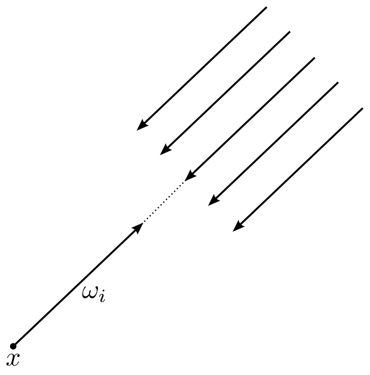


Figure 30: Sampling of a directional light shown as a cluster of parallel rays.

Listing 5: Directional Light Sampling

```
float3 SampleDirectionalLight(float3& lx, float3& wi, float& pdf,
        LightSource ls, float3 x) {
  wi = -ls.direction;
  pdf = 1.0;
  return ls.L;
}
```

6.5.3 Area Lights

Area lights are represented as flat rectangles with a controllable width and height parameter that indirectly controls the area of the light. This is in fact the only implemented light in the framework of Inviwo which requires Monte Carlo techniques to be sampled, since it has an area. It picks a uniformly distributed point on the surface and then samples it using the uniform estimator (eq. 29) with the dimensional space of the integrand being the area of the light.

Listing 6: Area Light Sampling

```
float3 SampleAreaLight(float3& lx, float3& wi, float& pdf,
        LightSource ls, float3 x) {
  float2 lsx = rnd2() - float2(0.5);           // random sample [-0.5,0.5]
  lsx = lsx * float2(ls.width, ls.height);     // random surface sample

```

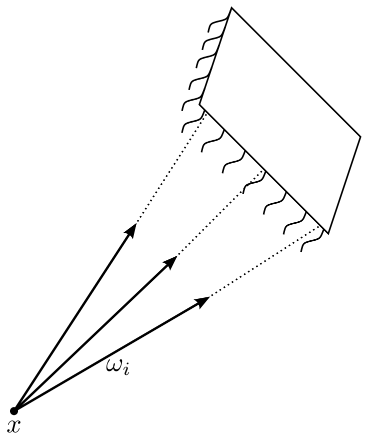


Figure 31: An area light source being sampled with rays, there are multiple rays drawn to signify that the entire surface has a probability of being sampled.

```
    // World space light sample
    lx = ls.lightToWorld * (float3)(lsx.x * ls.width, lsx.y * ls.height);
    wi = Normalize(lx - x);
    pdf = 1.0 / ls.area;

    // Make sure only to contribute from the front of the light
    return Dot(-wi, ls.direction) > 0.0 ? ls.L : 0.0;
}
```

6.5.4 Environment Light or Infinite Area Light

This represents light coming from all directions infinitely far away. Its implemented as a single global light with a constant spectral intensity and is controlled through the parameters for the Path Tracer via a color-picker and an intensity-slider. Note that this light is not sampled by the direct sampling routine, instead its emission is applied when the ray exits the media.

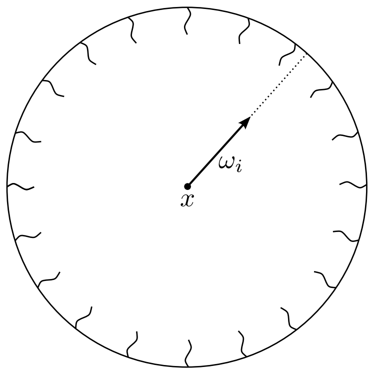


Figure 32: The environment light, emitting an equal amount of radiance from every direction from an infinite distance.

6.5.5 Estimating Transmittance and Visibility

In order to estimate the transmittance (eq. 15) it is conventional to use a uniform sampling strategy. The uniform strategy essentially samples the entire distance in equally spaced steps, with a possible randomized start offset. However, the method used in the implementation performs a *visibility check* by using the SampleDistance method, that in fact produce samples distributed according to the Transmittance.

First a distance d is generated along the ray using the SampleDistance method. If this distance d is less than the max distance t, there is no visibility between the points. However if d is larger than t, then we have full visibility.

Listing 7: Estimating Transmittance

```
float3 EstmateTransmittance(float3 x0, float3 x1) {
  float t = Distance(x0, x1);
  if(t == 0.0)
    return float3(1.0);
  float3 w = (x1 - x0) / t;
  float d = SampleDistance(x0, w);
  return d < t ? float3(0.0) : float3(1.0);
}
```

6.6 BSDF

The function BSDF(se, x, wo, wi) provides the implementation of the reflectance $\rho(x, \omega_o, \omega_i)$. Where the BSDF itself can contain multiple BRDFs or Phase functions that each describe different layers or components of the reflecting light.

The BSDFs are controlled via the material property that is a composition of parameters designed to control whatever parameters that are available to the chosen material. The implemented materials include the isotropic phase function, Henyey-Greenstein phase function and a selection of materials that include specular or glossy BRDFs such as the Blinn-Phong and Cook-Torrance model.

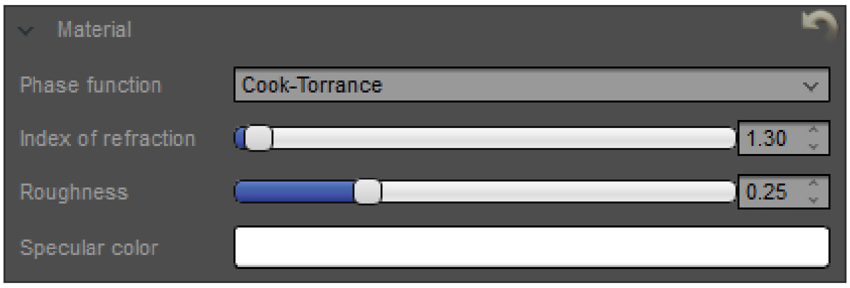

Material
Phase function: Cook-Torrance
Index of refraction: 1.30
Roughness: 0.25
Specular color

Figure 33: The material property where the BSDF is chosen via the Phase function and additional parameters for the selected function are displayed.

6.6.1 Deriving a unified expression

By studying the formulations of L_{vol} and L_{surf} from section Volumetric Path Tracing and disregarding the emission (since it is sampled explicitly), one can clearly see the resemblance between them. The goal is to unify the expressions by baking the terms that differentiate them into their corresponding terms ρ_p and ρ_r. In the case of L_{vol} the term $\frac{\sigma_s}{\sigma_t}$ which equals albedo α is moved into the phase function ρ_p. In the case of L_{surf} the cosine term $|\cos\theta'|$ is moved into the BRDF ρ_r. The formulation can then be unified with a general reflectance term ρ into a more general term L_{in} that covers both surface reflections and scattering in participating media.

$$L_{in}(x, \omega) = \int_{S^2} \rho(x, \omega, \omega')L(x, \omega')\mathrm{d}\omega' \tag{36}$$

This means that phase functions have the ability to change the spectral distribution of incoming light by the albedo α, just like BRDFs. While the surface models, BRDFs, are modified to account for the fact that they represent flat surfaces and will receive less light at grazing angles according to the cosine term $|\cos\theta'|$. As an example this modification is applied to the isotropic phase function, ρ_{iso}.

$$\rho_{iso} = \frac{\alpha}{4\pi}$$

6.6.2 Implementation of the BSDF

When the project began, there were already a handful of BSDFs implemented in another project which have been reused in this implementation. The main reason for reusing the material property and the underlying implementations BSDFs, is to maximize the compatibility of the processor with other implementations in the framework of Inviwo.

The Phase Functions serve as fully defined BSDFs themselves, while the materials that contain specular BRDFs are complimented with the isotropic phase function to serve as the diffuse component. The BRDFs require a surface normal in order to work, and this is provided by the ScatterEvent as the gradient of the volume data. There is also a built in fall back for the BRDFs, since they cannot be evaluated if the gradient is undefined. In that case the implementation is forced to evaluate the isotropic phase function instead.

Since the BSDFs were implemented in another project, they will not be covered here. However there is one BSDF called *Mix* that is used in the results section which serves as the representative of specular or glossy BSDFs in comparisons. The Mix function essentially mixes a single isotropic phase function with an isotropic phase function and a BRDF according to some constant user specified parameter. The specified parameter is also used in the corresponding function SampleBSDF(se, d, pdf) for Mix, where it is compared to the gradient magnitude in order to determine what distribution to sample from.

6.7 Estimating Indirect Lighting

In order to estimate the indirect light contribution, a single direction is sampled from the BSDF by using the function $SampleBSDF(se, w, pdf)$. To sample the BSDF is to sample one of its underlying Distribution Functions, but for simplicity and to focus on the main subject of this paper only sampling of the Henyey-Greenstein phase function (eq. 23) is presented. The expression is derived by applying the inversion method to the definition of the phase function, but that derivation is not presented in this paper.

$$\cos\theta = \begin{cases} \frac{1}{2g}\left(1 + g^2 - \left(\frac{1-g^2}{1-g+2g\xi}\right)^2\right) & \text{if } g \neq 0 \\ 1 - 2\xi & \text{if } g = 0 \end{cases} \tag{37}$$

By using the resulting expression of $\cos\theta$ together with an coordinate system created from an input direction ω, a new direction can be sampled. The probability density for sampling any particular direction on a phase function is simply the value of the phase function itself. This is because the phase functions are in fact PDFs themselves.

6.8 Terminating the Path

In order to terminate the path and to assure that the estimate remains unbiased, a technique called *Russian Roulette* is used. The method uses some termination probability q, that can be chosen freely, and tries to terminate the path with this probability. If the path is terminated, then the recursion stops, but if the path survives the russian roulette, the path continues, but its throughput is scaled with a term $1/(1-q)$ that effectively accounts for all the samples that were skipped.

$$F' = \begin{cases} \frac{F}{1-q} & \xi > q \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

The mathematical derivation or any proof of the method will not be presented in the paper, but can be verified by calculating the expected value of F' to see that the terms $(1-q)$ and $1/(1-q)$ will cancel out and the result will be the expected value of F.

Listing 8: Terminating the path

```
bool Terminate(float3& tr) {
  if(all(tr < 0.01)) {
    float surviveProb = 0.5;
    if(rnd() > surviveProb)
      return true;
    tr = tr / surviveProb;
  }
  return false;
}
```

The russian roulette termination is not applied until the depth of the recursion is greater than 3 and the throughput of the path is less than 0.01. And when it is applied it is with a static probability of 0.5. All of the parameters, the minimum depth, the minimum throughput and the probability itself are hand-picked values that seem to produce a good performance to noise ratio for this implementation. A study would be required to find a set of values that provide an optimal trade off between performance and noise, but this is something for further improvements.

6.9 Running Estimate

The process of generating an estimate for the intensity of a pixel is to create a large number of samples N by tracing rays from the position of the eye, through the pixel and into the scene where it branches into different paths. The resulting intensity of the pixel is then calculated as the average of the all the samples of this pixel, assuming that each sample is equally important, which is generally the case.

$$A_{N+1} = A_N + \frac{x_{N+1} - A_N}{N+1} \tag{39}$$

Instead of calculating the average by first calculating the sum of N samples and then dividing the sum with N, a cumulative running average operator is used. The operator stores the average value of all previous samples, and incrementally updates this value with each added sample. The operator can be seen in eq. 39 where the new average A_{N+1} is calculated using the previous average A_N and the new sample x_{N+1}.

By using the Moving Average Operator (eq. 39) a *running estimate* can be kept, this means that the estimate is always available at hand and can be displayed on screen after each new sample added. The benefits of this is that the user can progressively see the estimate improving and can determine when it is good enough.

6.10 Tone Mapping

The result of the path tracing processor is an image with 3 channels in float32 format, representing the red, green and blue quantity of energy or flux of each pixel. Since flux is a measure of energy, its range of values is in theory unbound $[0, \infty[$, but the max value of float32 is about 3.4×10^{38}. This is not really a problem considering that the sun for example has a radiance measure of about 2×10^{7}. However the screen itself can only represent a brightness far less than that, quantized over 8-bits of precision. This is where the Tone Mapping operator comes into play.

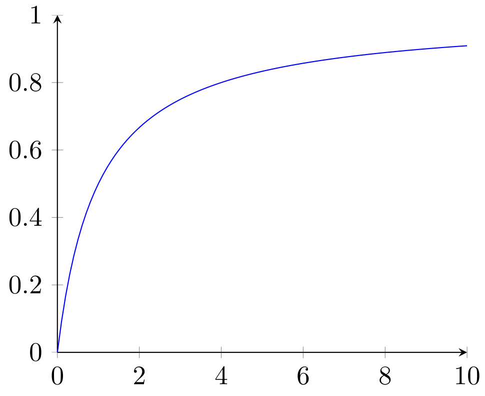
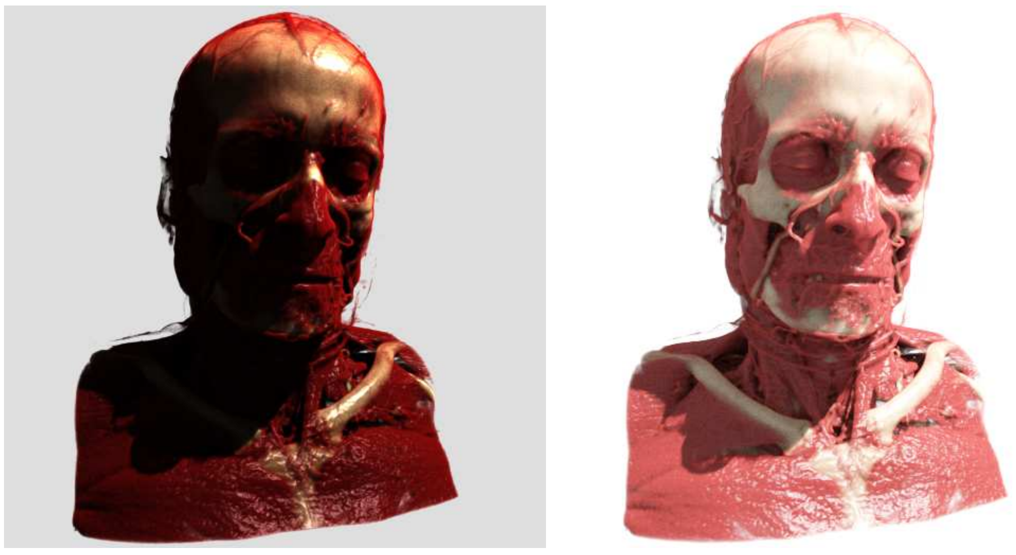


Figure 34: The Basic Reinhard Tone Mapper.

$$f(x) = \frac{x}{x+1} \tag{40}$$

The Tone Mapping operator essentially takes a positive unbound value as input and maps this value via a function to an output value between [0,1] that can be displayed by the screen. The function is designed to mimic the logarithmic response to light as human perceive it. One common Tone Mapper is the basic *Reinhard Tone Mapper* [Reinhard et al., 2002] and can be seen in figure 34 with its corresponding function expressed in equation 40.



(a) Raw radiometric data (b) Tone mapping applied

Figure 35: A demonstration of the visual importance of tone mapping.

7 Validating the Implementation

In order to validate the results of the implementation, a test scene was implemented in Inviwo. The scene contained two processors, one *analytical* and one *numerical* with their outputs connected to a difference processor. The test-scenario had to test the various components and techniques involved and be analytically solvable.

7.1 Analytical Single Scattering Solution

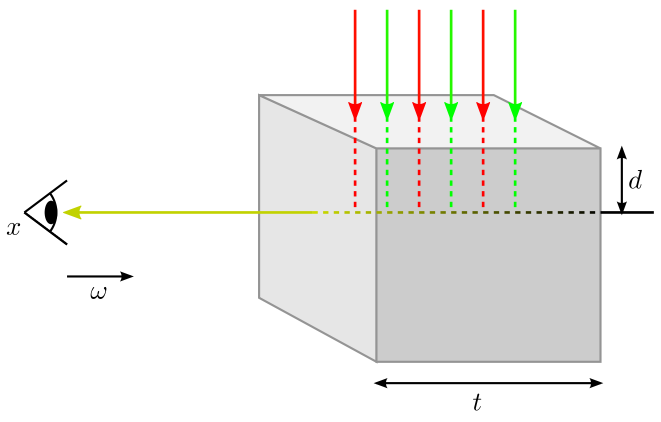


Figure 36: *The single scattering test scenario:* a block of homogeneous media is viewed from the side while being lit from above.

The test-scenario consists of a solid block of homogeneous media with thickness t being lit by directional light from above that has to travel a distance d through the media to reach the view-ray as seen in figure 36. By using this setup, the render equation can be written as seen in eq. 41 with the source term listed in eq. 41.

$$L(x,\omega) = T_r(x_0 \to x)L_0(x_0,-\omega) + \int_0^t T_r(x' \to x)S(x',-\omega)\mathrm{d}t' \tag{41}$$

$$S(x',-\omega) = L_{ve}(x',-\omega) + \sigma_s(x',-\omega)\int_{s^2} \rho(x',-\omega,\omega')L(x',\omega')\mathrm{d}\omega' \tag{42}$$

Now a number of assumptions are applied in order to simplify the scene and to make it analytically solvable.

No Volume Emission	$L_{ve} = 0$
Single Scattering	Recursion is broken and light only contributes through a single scatter via direct lighting from the directional light, $L(x',\omega') = L_d(x',\omega')$.
No Endpoint Contribution	$L_0(x_0,-\omega) = 0$.
Homogeneous media	$T_r(x \to x') = e^{-\sigma_t t'}$, Beers law (eq. 16).
Directional Light Only	$L_d(x',\omega') = T_r(x_{light} \to x')L_e$ where $T_r(x_{light} \to x') = e^{-\sigma_t d}$ and L_e is the light sources radiance.

With these simplifications, equation 41 with corresponding source term (eq. 42) can be rewritten as a single 1D equation (eq. 43) with ω as the axis and $x = 0$. Notice that $\rho(x',-\omega,\omega')$ has been simplified into ρ since the both directions are constant and perpendicular as well as a homogeneous medium is being used. The term L_e represents the combined radiance from the directional lights, $L_e = L_{de1} + L_{e2} \dots L_{en}$, with a variable number of light sources to help verify Monte Carlo Techniques when sampling multiple lights.

$$L = \int_0^t e^{-\sigma_t t'}\sigma_s\rho e^{-\sigma_t d}L_e\mathrm{d}t' = \sigma_s\rho e^{-\sigma_t d}L_e\int_0^t e^{-\sigma_t t'}\mathrm{d}t' \tag{43}$$

The equation contains many constants that can be moved out of the integrand as seen in the final expression of eq. 43. This simplified equation can now be solved by integrating and the final expression can be seen in eq. 44.

$$L = \frac{\sigma_s e^{-\sigma_t d} L_e}{\sigma_t} \left(1 - e^{-\sigma_t t}\right) \tag{44}$$

7.2 Analytical Multiple Scattering Solution

With an analytical single scattering solution in place, the search continued for an analytical solution of multiple scattering. A paper was found [Narasimhan et al., 2004] which provided an analytical solution in a very controlled scenario of an isotropic point light source in the center of a sphere filled with some homogeneous media that scatters according to the Henyey-Greenstein phase function (eq. 23).

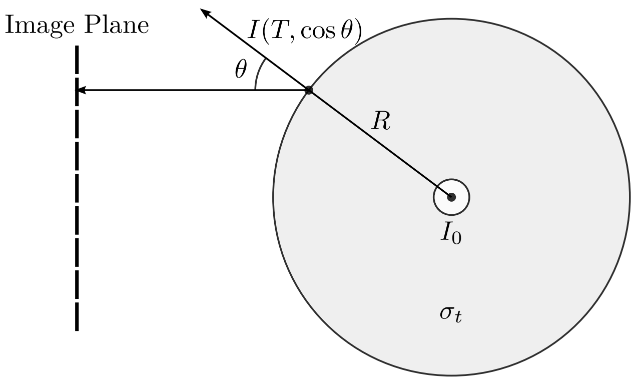


Figure 37: The multiple scattering test scene, an isotropic light source in the center of a sphere filled with some homogeneous medium.

$$T = \sigma_t R$$

$$g_m(T) = I_0 e^{-\beta_m T - \alpha_m \log T}$$

$$\alpha_m = m + 1$$

$$\beta_m = \frac{2m+1}{m} \left(1 - \frac{W_{m-1}}{2m-1}\right)$$

$$W_k = (2k+1)q^k, \text{ k} \geq 1$$

$$I(T, \cos\theta) = \sum_{m=0}^{\infty} (g_m(T) + g_{m+1}(T)) L_m(\cos\theta) \tag{45}$$

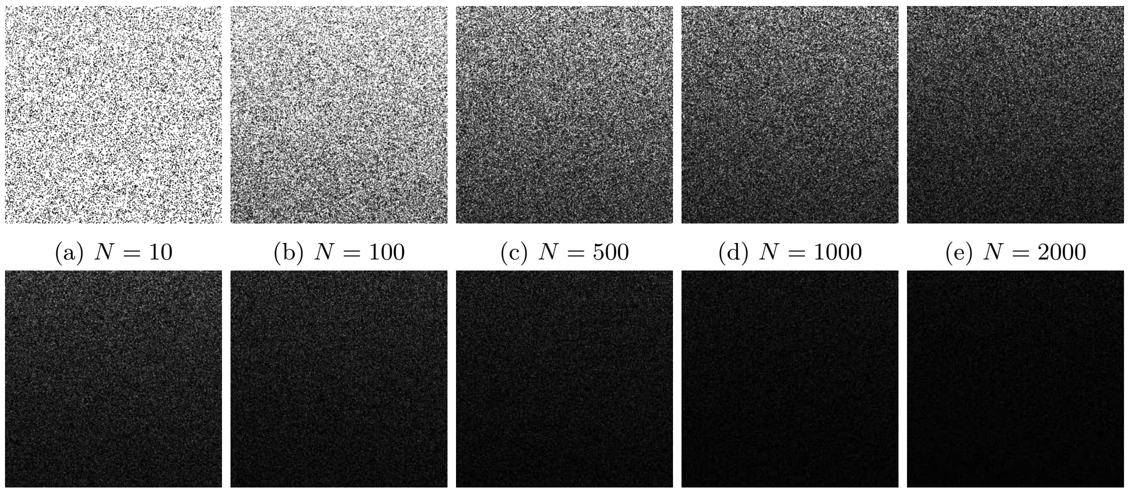
The method was implemented as a processor and as a function in Matlab to enable plotting, but none of my implementations seemed to produce the expected results. Most plots in the paper do not have any parameters specified and are all displayed as normalized point spread functions PSF, which makes it difficult to verify the implementation against. However there is one plot (fig. 2b) in the paper that shows the ringing artifacts that appear when using too few terms when calculating a sharply peaked PSF. When comparing my implementations against this plot under the same circumstances, it clearly shows that there must be something wrong with my implementation. The authors of the paper were contacted in an effort to try and get some pointers or even a working Matlab implementation, but I failed to get a hold of any of them.

The approach of an analytical multiple scattering solution was put on a hold as there were other things that needed to be implemented. And this part of the project had already exceeded its allocated time frame.

8 Results

8.1 Analytical Single Scattering Convergence Test

With the various components of the renderer in place, a numerical solution was implemented and tested against the analytical solution. The difference between the numerical and analytical solutions was sampled at different iterations to serve as data for a graph over the convergence rate. The workspace set-up that was used for this test is shown in figure 1, displayed on page 3.



(a) $N = 10$ (b) $N = 100$ (c) $N = 500$ (d) $N = 1000$ (e) $N = 2000$

(f) $N = 5000$ (g) $N = 10000$ (h) $N = 20000$ (i) $N = 50000$ (j) $N = 100000$

Figure 38: The difference visualized over iterations where the intensity of the images have been scaled with a factor of 64.

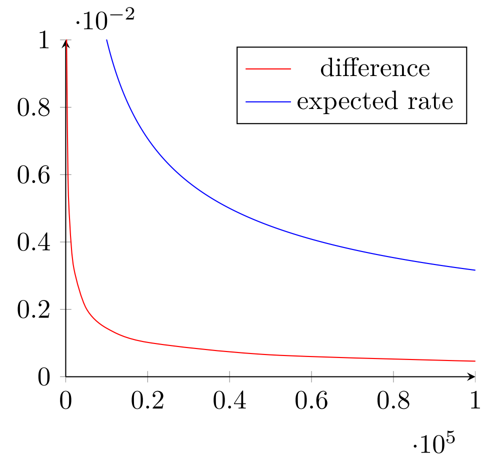


Figure 39: The difference (red) plotted with the expected convergence rate of $\frac{1}{\sqrt{N}}$ (blue).

The difference seem to converge within the expected range as seen in figure 39, thus verifying the validity of the implemented methods in the numerical single scattering solution.

8.2 Extinction Coefficient

The test scene is a single square area light with the surface area 1 m^2 and a radiant exitance of 50 W/m^2 in a scene filled with some homogeneous media. The albedo of the media is 0.9 and multiple scattering is enabled for all renderings unless explicitly stated otherwise. In figure 40 the opacity is varied between a value of 0.01 down to a value of 5.0×10^{-5} in a isotropically scattering media. The corresponding physical parameters related to the tests can be seen in table 2. Each render presented in the figures have been estimated with 25 000 samples per pixel.

Opacity	σ_t (m^{-1})	σ_s (m^{-1})	σ_a (m^{-1})
0.01	0.3377	0.3040	0.0337
0.005	0.1684	0.1516	0.0168
0.003	0.1010	0.0909	0.0101
0.00075	0.0252	0.0227	0.0025
0.0005	0.0168	0.0151	0.0017
0.0003	0.0101	0.0091	0.0010
0.00005	0.0017	0.0015	0.0002

Table 2: The opacity and the corresponding physical parameters.

8.3 Henyey-Greenstein Anisotropy Parameter

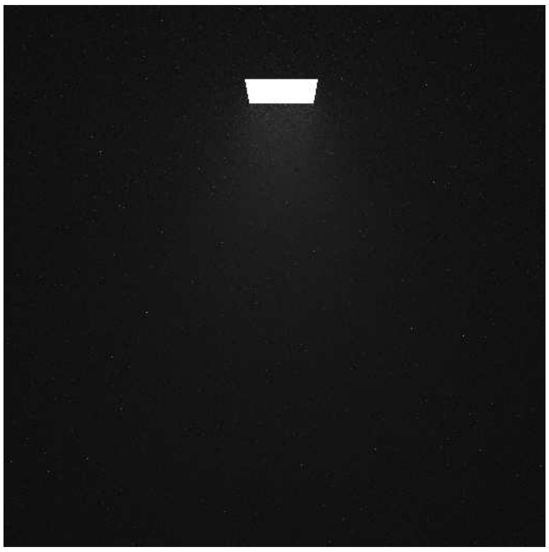
The test scene is the same one used in the extinction tests seen in figure 40, but with an added view where the back-face of the light is shown when studying negative parameters. In figure 41 and 42 the anisotropy parameter is varied in a scene filled with a homogeneous that scatters according to the Henyey-Greenstein Phase Function (eq. 23). The opacity of the medium is set to a constant of 5.0×10^{-4}, with its corresponding physical terms listed in table 2. Each render presented in the figures have been estimated with 25 000 samples per pixel.

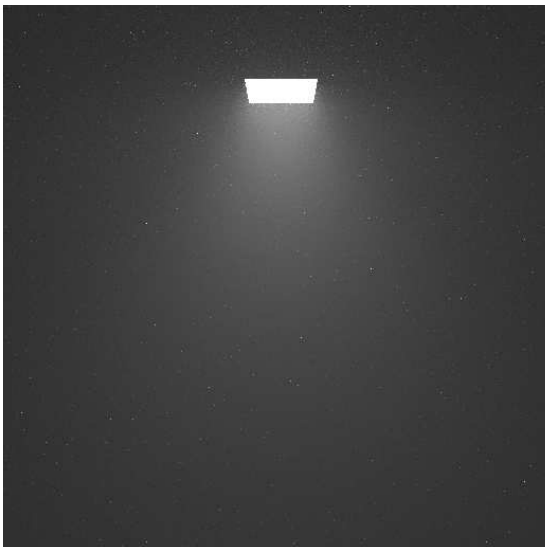
8.4 Sparse Inhomogeneous Participating Media

The test scene seen in figure 43 contains an inhomogeneous media depicting cloud formations from the hurricane Isabel. The scene is lit with a single point light source in the hurricanes eye and rendered using the Henyey-Greenstein Phase Function, with an anisotropic parameter of 0.9. Each render presented in the figure have been estimated with 100 000 samples per pixel.

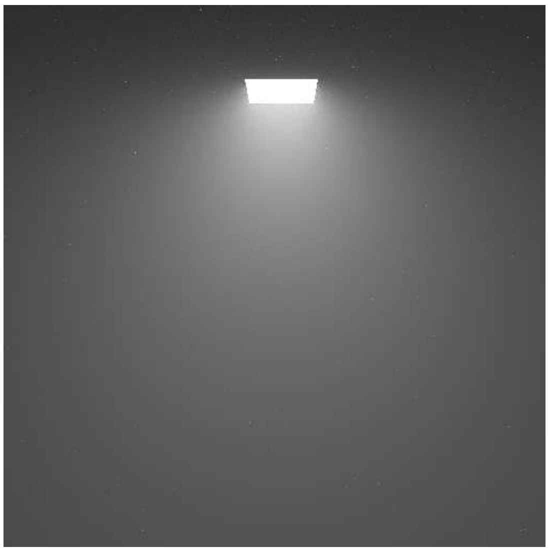
8.5 Dense Participating Media

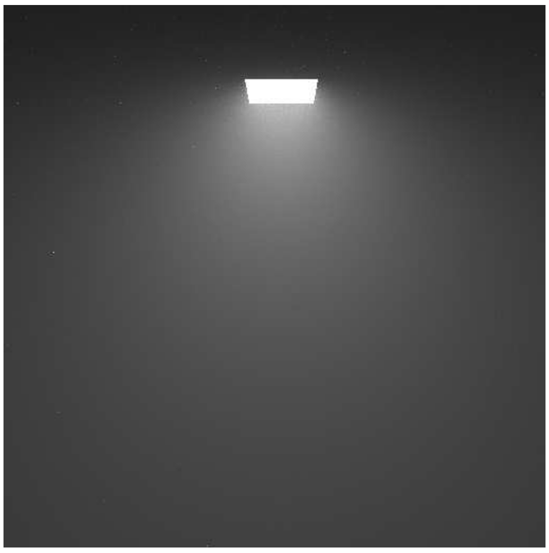
In the scenes seen in figures 44, 45 and 46, dense media is being lit with an area light with from a direction over the viewers right shoulder and is paired constant emission from an environment light. The area light has a surface area of 1 m^2 and a radiant exitance of 50 W/m^2 and is approximately placed at a distance of 3 meters from the center of the object. The emission from the environmental light is changed between the different renderings as some set-ups reflect more light. The scene shown in figure 47 is lit from the same direction and position, but with a point light source with the intensity of 50 W/sr. Each render presented in the figures have been estimated with 1000 samples per pixel.


(a) $o = 0.01$


(b) $o = 0.005$

(c) $o = 0.003$


(d) $o = 0.00075$


(e) $o = 0.0003$

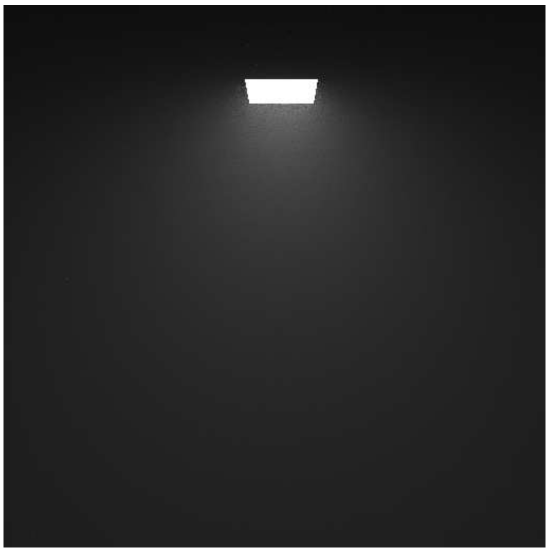
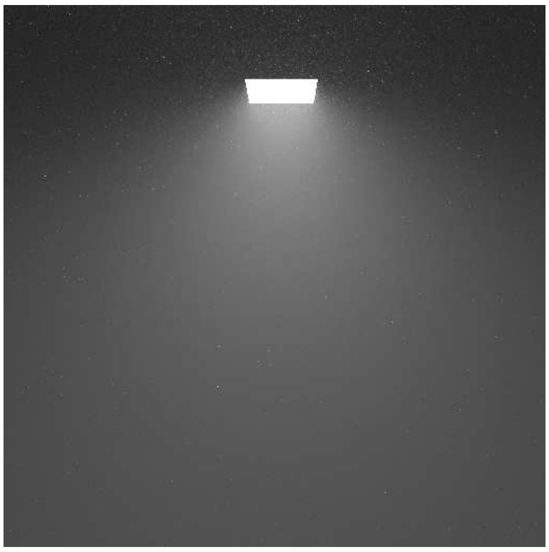
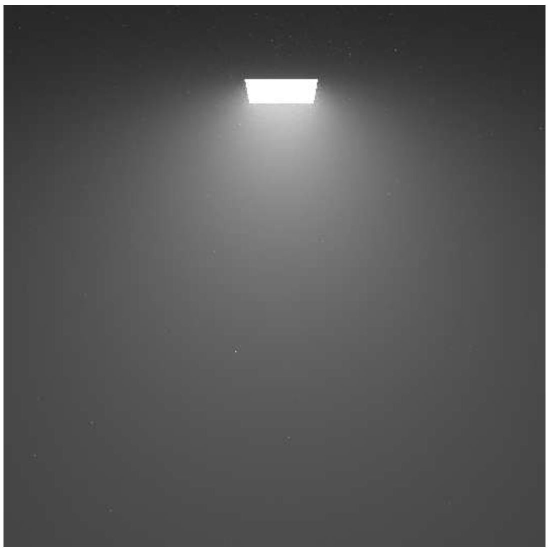

(f) $o = 0.00005$

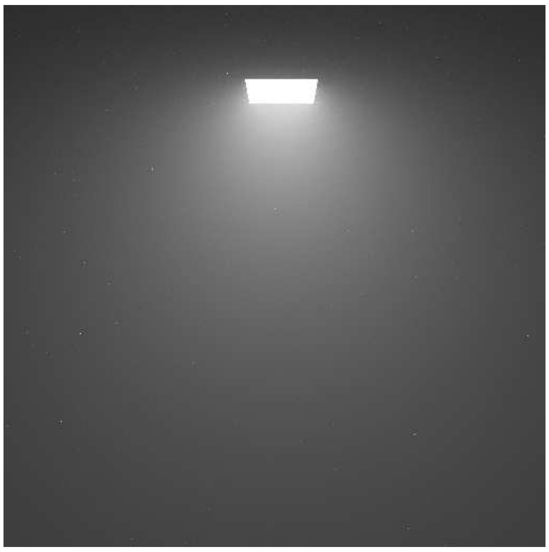
Figure 40: Different opacities used to control the extinction in the scene with sparse participating media.



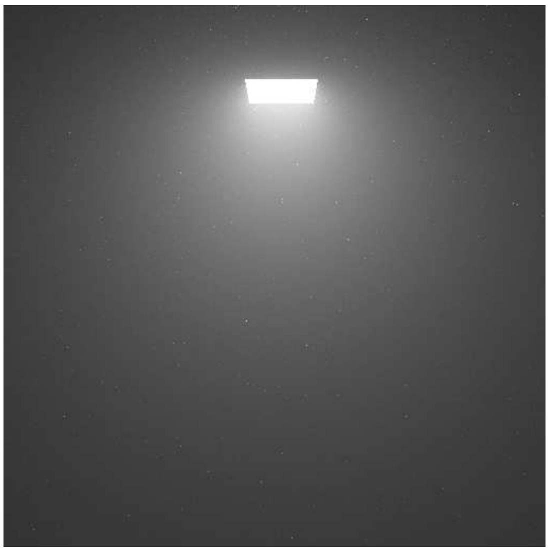
(a) $g = -0.5$



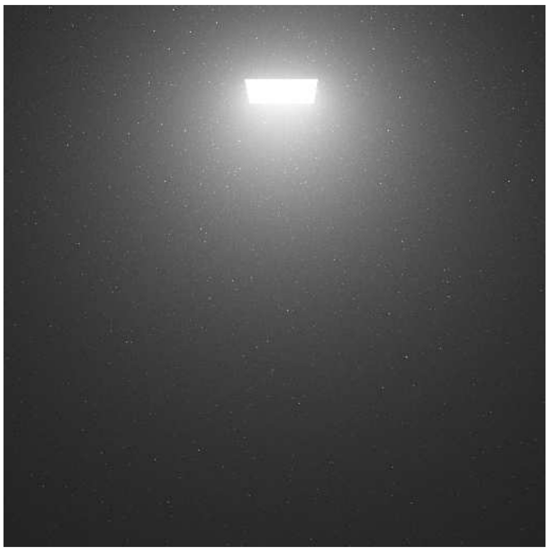
(b) $g = 0.0$



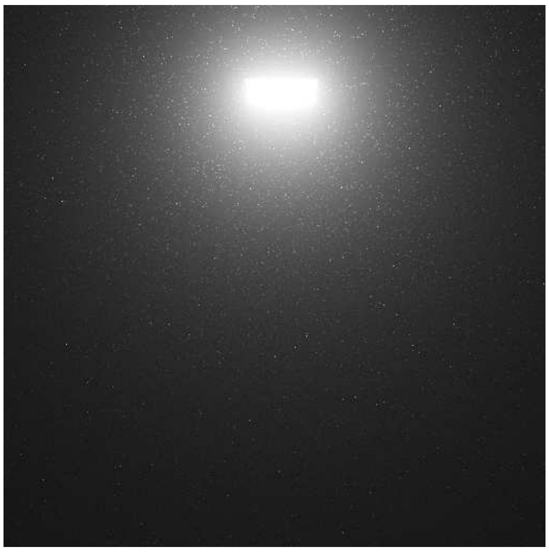
(c) $g = 0.25$



(d) $g = 0.5$

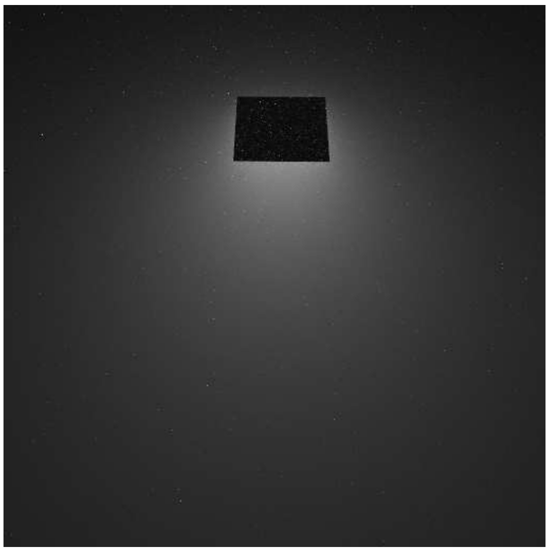


(e) $g = 0.75$

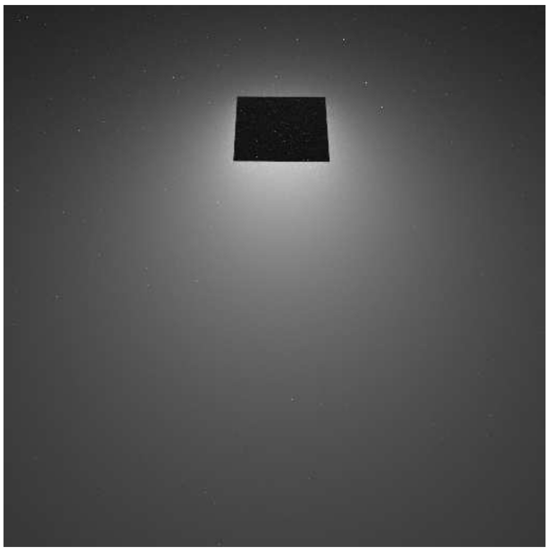


(f) $g = 0.9$

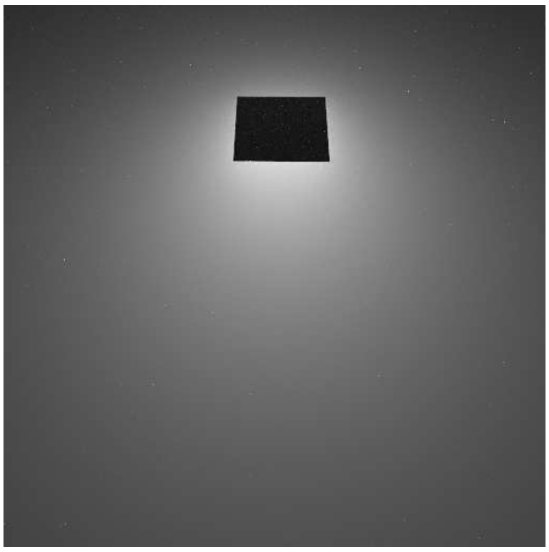
Figure 41: Different anisotropic parameters used for the Henyey-Greenstein function in the scene with sparse participating media.



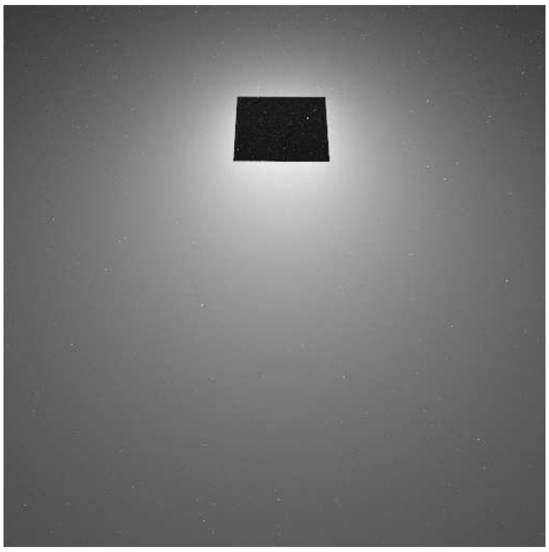
(a) $g = 0.5$



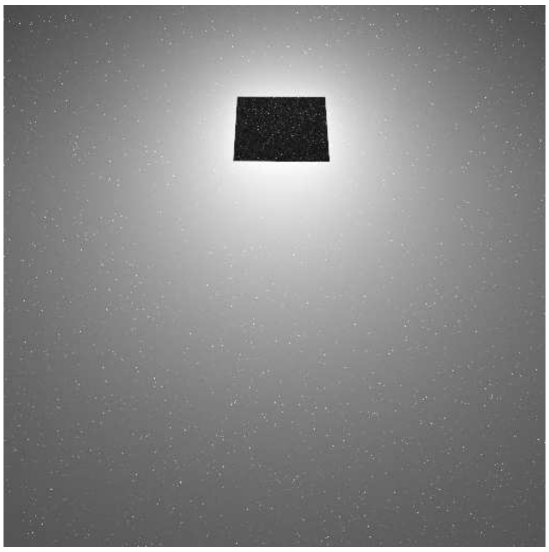
(b) $g = 0.0$



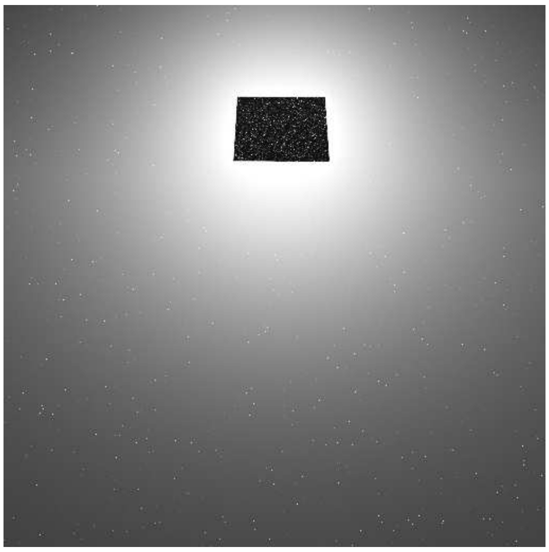
(c) $g = -0.25$



(d) $g = -0.5$



(e) $g = -0.75$



(f) $g = -0.9$

Figure 42: Different anisotropic parameters used for the Henyey-Greenstein function in the scene with sparse participating media.

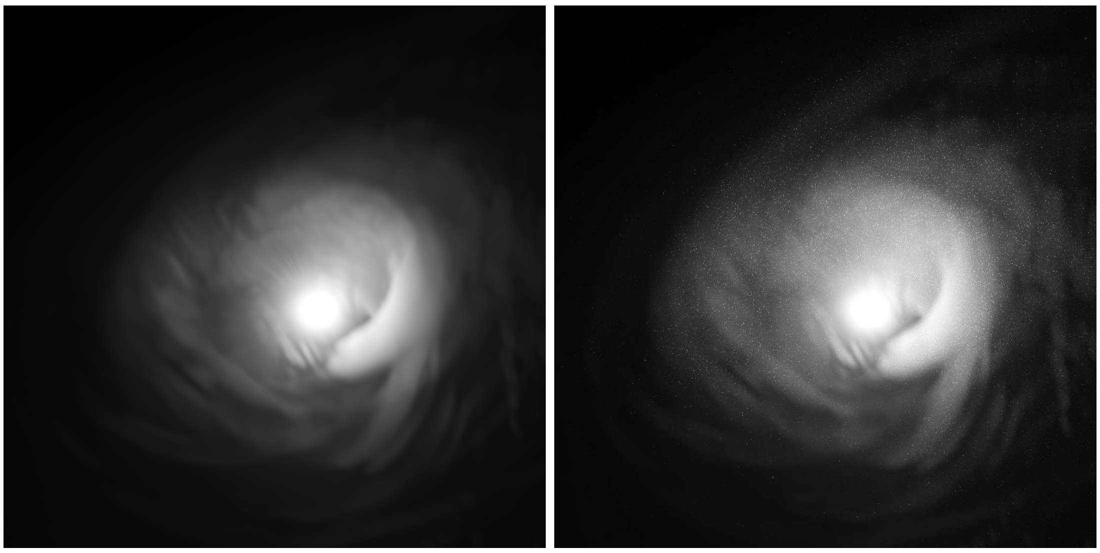

(a) Henyey-Greenstein Phase Function, $g = 0.9$, Single Scattering (b) Henyey-Greenstein Phase Function, $g = 0.9$, Multiple Scattering

Figure 43: A point light source in the eye of the hurricane Isabel.

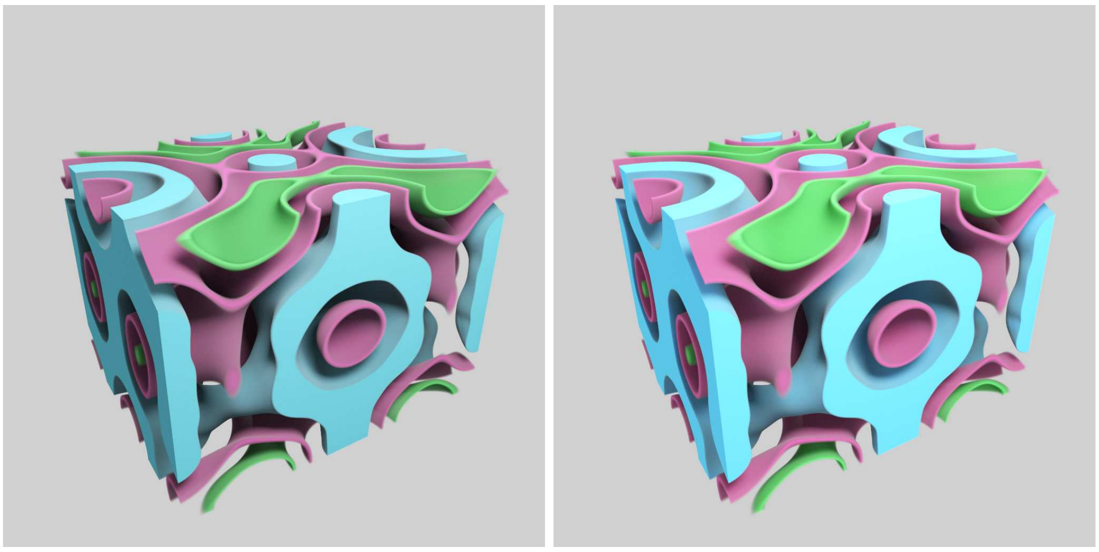
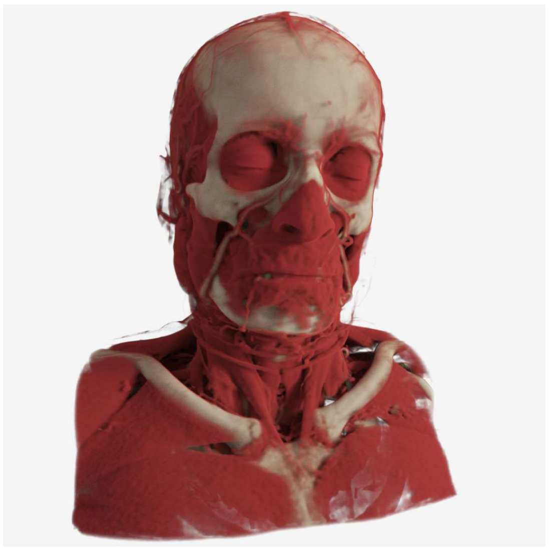
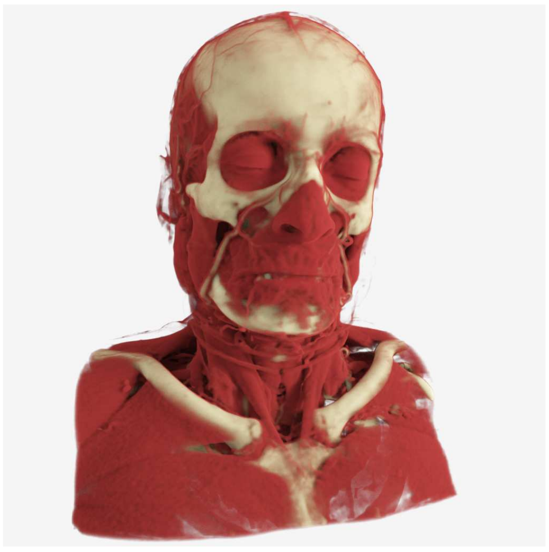
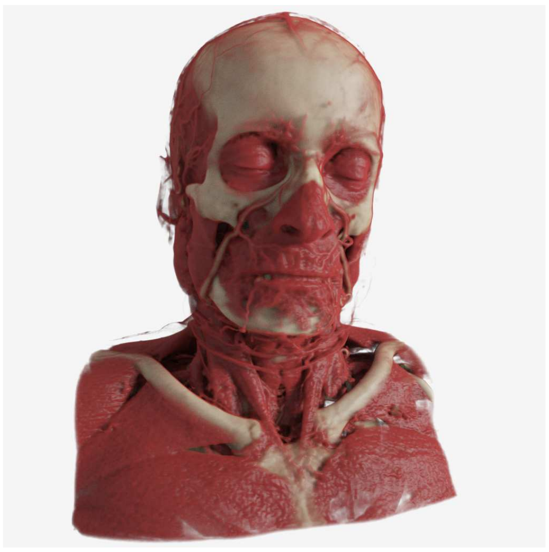

(a) Isotropic Phase Function, Single Scattering (b) Isotropic Phase Function, Multiple Scattering

Figure 44: The volumetric set *Boron* rendered with an isotropic phase function.


(a) Isotropic Phase Function, Single Scattering


(b) Isotropic Phase Function, Multiple Scattering


(c) Mixed BSDF, Single Scattering

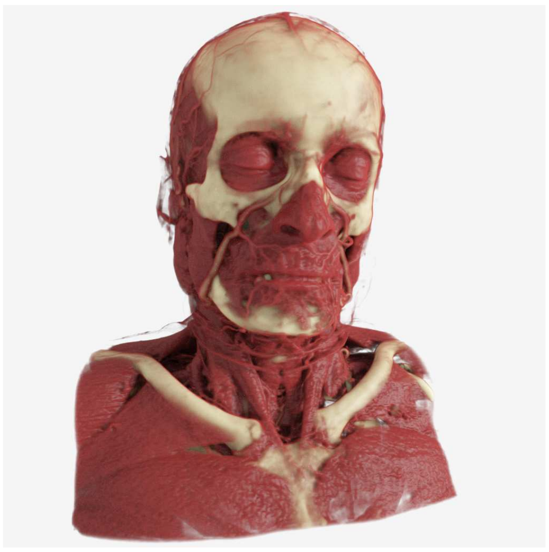
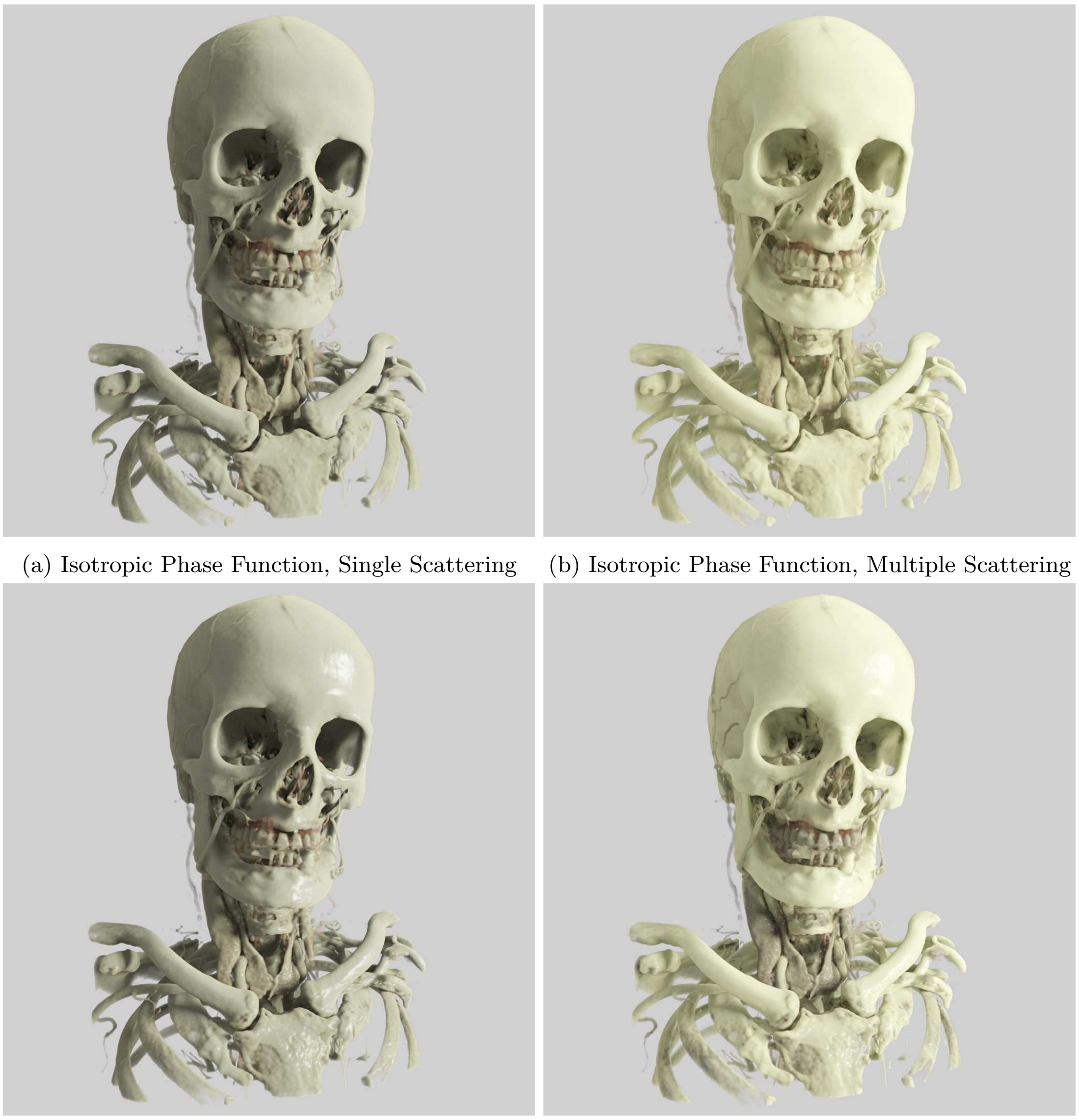

(d) Mixed BSDF, Multiple Scattering

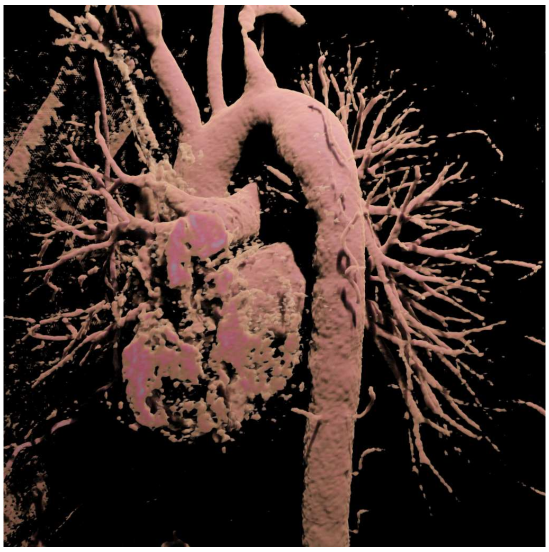
Figure 45: The Manix data set, mapped with the transfer function to show muscle-tissue and bone.



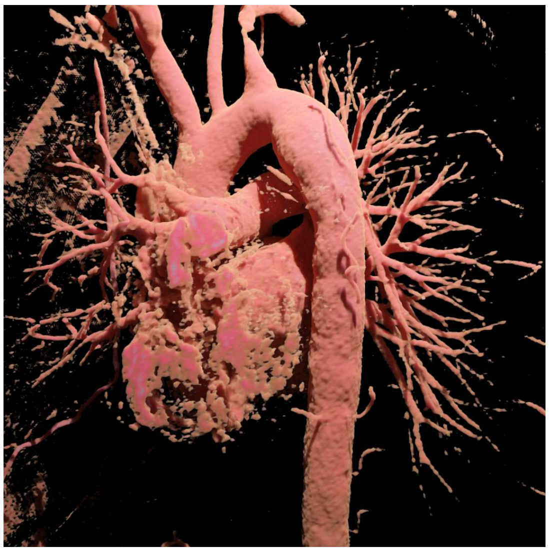
(a) Isotropic Phase Function, Single Scattering (b) Isotropic Phase Function, Multiple Scattering

(c) Mixed BSDF, Single Scattering (d) Mixed BSDF, Multiple Scattering

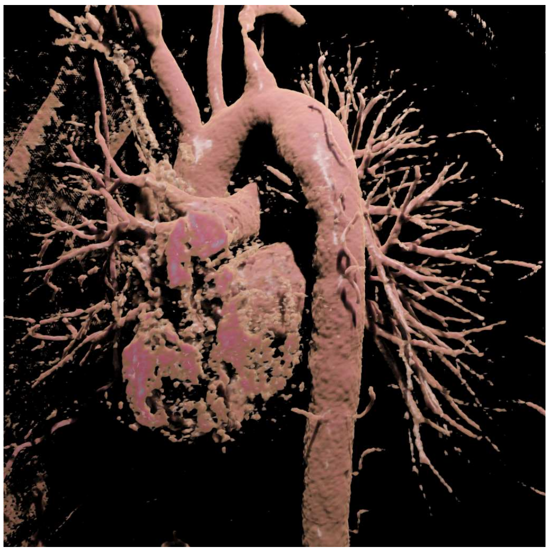
Figure 46: The Manix data set, mapped with the transfer function to show only bone.



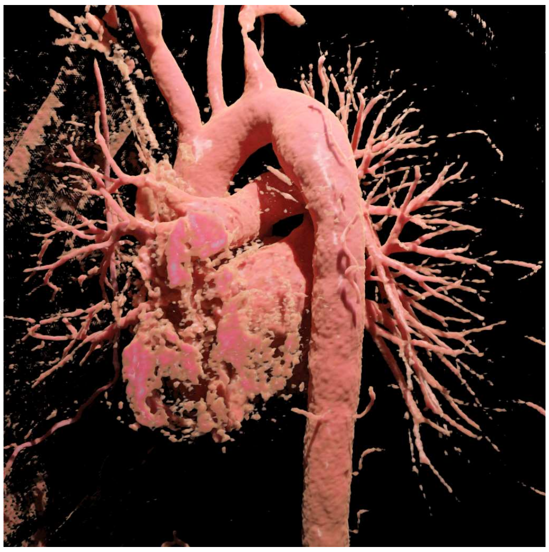
(a) Isotropic Phase Function, Single Scattering



(b) Isotropic Phase Function, Multiple Scattering



(c) Mixed BSDF, Single Scattering



(d) Mixed BSDF, Multiple Scattering

Figure 47: Vascular structures of a subclavia lit with a point light.

9 Conclusion

9.1 Validity of the Implementation

There is no validation in place to prove that the multiple scattering solution is correctly implemented. However the validity of the single scattering solution and the methods used in that scenario is tested over iterations and seem to converge accordingly. Thanks to the methods used, the step from a single scattering solution to a multiple scattering solution only requires a new direction to be sampled from the evaluated Scatter Event and a proper scaling of the paths throughput.

9.2 Single and Multiple Scattering

By looking at the results and comparing the single scattering to the multiple scattering, one can see that the results generated from multiple scattering generally are a bit brighter, which is expected. But the results are also generally more vivid in its colors and not as *flat* when compared to single scattering. This is most certainly thanks to the sub-surface reflections that naturally occurs in volumetric rendering with multiple scattering. Another key feature of multiple scattering is the indirect illumination that occurs. This can clearly be seen in figure 47, where the areas that are not directly lit by the light source have no chance of receiving any incoming light without multiple scattering.

Multiple scattering also introduces more noise when compared to single scattering. This is a side-effect of using a single direction to serve as an estimate of all possible directions when sampling the BSDF. The noise generated by multiple scattering can best be seen in the tests with sparse participating media, especially the one with the point light inside the hurricane. This rendering is especially prone to be noisy because there is a point light source inside the media itself. Point lights are as mentioned before, physically unrealistic, and when sampling them they can be problematic thanks to the weak singularity presented by the $\frac{1}{distance^2}$ term. This term is no problem if the point light resides outside of the participating media, but when it lies within the media, the sampled distance can be so small that the term itself goes to infinity.

9.3 Henyey-Greenstein Anisotropy

When using positive values, which means forward scattering, most of the energy from the scatter continues in the general direction that it traveled in. This scenario is shown in figure 41 where larger values focuses the energy from the scatter giving the light source a visible halo or corona.

When using negative values, which means backward scattering, most of the energy is reflected back towards the direction from where it came. This scenario is shown in figure 42, where only the non emissive back of the light source is shown. All visible light that reaches the eye subsequently comes from the scattering process itself and thus gets and increase with larger negative values of g.

9.4 Extinction in sparse participating media

With higher values of extinction (opacities greater than 0.005), the amount of light that reaches the eye from scattering seems to be reduced. At opacity values around $o = 5.0 \times 10^{-5}$, the media is so optically thin that there is not much matter for the light to interact with.

10 Further improvements

10.1 Extending the Transfer Function

In order to provide more exact control of the optical properties of the medium, every possible parameter that may be spatially varying inside the volume should be controlled with the Transfer Function. This could be achieved by extending the transfer function to include several output textures as a compliment to the single albedo/diffuse + opacity (RGBA) texture provided today. For example, one of the textures could contain *Emission* controlled with RGB + some scale parameter in the alpha channel.

Another crucial change would be to incorporate a non-uniform extinction coefficient, this would enable more realistic material descriptions. One result of this would be that the transmission itself would be spectrally varying. This would enable sampled light to penetrate the medium in a spectrally varying manner, resulting in e.g. colored shadows, which is not possible in the current state.

10.2 Support Surfaces through Meshes

The mathematical models presented in this paper are formulated with surfaces and participating media in mind. However the implementation only truly covers the volumetric part of the equation. The renders that provide the most realistic results that I have seen up to this date, incorporate both explicitly defined surfaces via meshes and volumetric regions for the inside of the meshes. In its essence the two different approaches of volumetric rendering and surface rendering provide solid and proven models for each end of the material-spectrum. And both of the models are needed in order to produce truly realistic images.

The implementation tries to incorporate surfaces via the Mix BSDF, but the surfaces are implicitly defined by the gradient of the volume density. To be physically correct, this should be based upon the change in the Index of Refraction of the medium, but in the implementation this is set as a global parameter for the medium. This leads to the implicit surfaces in volume not being as sharp as they perhaps should be, since the interfaces are approximated from the density of the volume.

10.3 Optimizing the termination parameters

As mentioned in the section *Terminating the Path* (6.8) a study would be required to find the optimal balance between performance and noise when using russian roulette termination. This study could be carried out over the three parameters that govern the termination process, the minimal recursion depth, the minimum throughout and the termination probability itself.

There is a possibility that there is some mathematically proven rule of hand presented in some paper, but this area has not been explored more than the parts mentioned in the literature [Pharr and Humphreys, 2010].

10.4 Successfully Implement Analytical Multiple Scattering Solution

Since the efforts to implement the analytical multiple scattering solution failed, this part of the project was dropped. I have no doubt that the method described in the paper is an accurate solution for the test scene which they describe. It failed, most certainly because of some error on my behalf, either in understanding the solution or the practical implementation of it. This would otherwise have provided a solid test for the implementation of the volumetric path tracer to verify against when using multiple scattering.

References

Robert L Cook and Kenneth E Torrance. A reflectance model for computer graphics. In *ACM Siggraph Computer Graphics*, volume 15, pages 307–316. ACM, 1981.

Louis G Henyey and Jesse L Greenstein. Diffuse radiation in the galaxy. *The Astrophysical Journal*, 93: 70–83, 1941.

Henrik Wann Jensen. *Realistic image synthesis using photon mapping.* AK Peters, Ltd., 2001.

James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

Eric P Lafortune and Yves D Willems. Bi-directional path tracing. In *Proceedings of CompuGraphics*, volume 93, pages 145–153, 1993.

Eric P Lafortune and Yves D Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques' 96*, pages 91–100. Springer, 1996.

István Lazániy and László Szirmay-Kalos. Fresnel term approximations for metals. 2005.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

S Narasimhan, Ravi Ramamoorthi, and S Nayar. Analytic rendering of multiple scattering in participating media. *Submitted to ACM Trans. Graph*, 2004.

Michael Oren and Shree K Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 239–246. ACM, 1994.

Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation.* Morgan Kaufmann, 2010.

Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 267–276. ACM, 2002.

Christophe Schlick. An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.

László Szirmay-Kalos, Balázs Tóth, Milán Magdics, and Balázs Csébfalvi. Efficient free path sampling in inhomogeneous media. *Poster Proc. Eurographics 2010*, 2010.

Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997.

E Woodcock, T Murphy, P Hemmings, and S Longworth. Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, volume 557, 1965.