Diplomová práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

# Realistické zobrazování vodních ploch

Aleš Koblížek

Vedoucí: Ing. Jaroslav Sloup
Obor: Počítačová grafika
Srpen 2020

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

| | | | |
|---|---|---|---|
| Příjmení: | **Koblížek** | Jméno: **Aleš** | Osobní číslo: **484302** |
| Fakulta/ústav: | **Fakulta elektrotechnická** | | |
| Zadávající katedra/ústav: | **Katedra počítačové grafiky a interakce** | | |
| Studijní program: | **Otevřená informatika** | | |
| Specializace: | **Počítačová grafika** | | |

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Realistické zobrazování vodních ploch**

Název diplomové práce anglicky:

**Realistic Rendering of Water Surfaces**

Pokyny pro vypracování:

Prostudujte literaturu pojednávající o zobrazování vodních ploch, zaměřte se zejména na metody založené na vícenásobném rozptylu světla [1]-[4]. Navrhněte a naimplementujte aplikaci, která bude schopna realisticky zobrazit vodní hladinu v libovolnou denní dobu a umožní měnit vlastnosti vody ovlivňující její vizuální vzhled (množství zněčištění, typ vody, hloubka, atd.).
Implementaci založte na metodě popsané v [4] a rozšiřte ji, aby brala v úvahu osvětlení od oblohy (nejen přímé sluneční záření) pro výpočet rozptylu světla pod hladinou (sub-surface scattering). Pro výpočet osvětlení oblohy využijte existující model [5][6].
Vygenerované obrázky porovnejte se skutečnými fotografiemi vodní hladiny. Zhodnoťte rychlost a pamětovou složitost implementované metody.
Implementaci proveďte v C/C++ s využitím OpenGL, CUDA/OpenCL.

Seznam doporučené literatury:

[1] Jinjin Shi, Dengming Zhu, Yingping Zhang, Zhaoqi Wang: Realistically Rendering Polluted Water. The Visual Computer, Vol.28, Num.6-8, p.647-656, Springer-Verlag, 2012, ISSN 0178-2789.
[2] Simon Premože, Michael Ashikhmin: Rendering Natural Waters. Computer Graphics Forum, Vol.20, Num.4, p.189-200, Blackwell Publishers Ltd, 2001, ISSN 1467-8659.
[3] E.Darles, B. Crespin, D. Ghazanfarpour, J.C. Gonzato: A Survey of Ocean Simulation and Rendering Techniques in Computer Graphics. Computer Graphics Forum, Vol.30, Num.1, p.43-60, Blackwell Publishing Ltd, 2011, ISSN 1467-8659.
[4] Chunyong Ma, Shu Xu, Hongsong Wang, Fenglin Tian, Ge Chen: A real-time photo-realistic rendering algorithm of ocean color based on bio-optical model. Journal of Ocean University of China, Vol.15, Num.6, p.996-1006, Springer, 2016.
[5] L.Hošek, A. Wilkie: An analytic model for full spectral sky-dome radiance. ACM Transactions on Graphics (TOG), Vol.31, No.4, p.95, 2012.
[6] L.Hošek, A. Wilkie: Adding a solar-radiance function to the Hošek-Wilkie skylight model. IEEE computer graphics and applications, Vol.33, No.3, pp.44-52, 2013.

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Jaroslav Sloup, Katedra počítačové grafiky a interakce**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **06.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce: **30.09.2021**

| | | |
|---|---|---|
| Ing. Jaroslav Sloup<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

| | |
|---|---|
| . | |
| Datum převzetí zadání | Podpis studenta |

## Poděkování

I would like to thank my supervisor Ing. Jaroslav Sloup for his valuable advice, helpful suggestions, and constructive criticism. I am also grateful to my family and friends for their support during my studies.

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 14. srpna 2020

## Abstrakt

Tato práce se zaměřuje na realistické zobrazování hladiny převážně stojatých vodních ploch, tj. jezer a oceánů bez velkých vln. Možná využití jsou převážně ve filmovém průmyslu, jelikož simulátory a počítačové hry obvykle musejí obětovat realismus aby dosáhly dostatečné snímkovací frekvence. Vzhled vodních ploch se liší poměrně značně, a to nejen kvůli tvaru hladiny, ale i čistotě vody a dopadajícím světlem. Scéna je nasvícena analytickým modelem slunce a světla od oblohy. Tvar vodní hladiny je získán z frekvenčního spektra mořských vln pomocí inverzní Fourierovy transformace. Odrazové vlastnosti hladiny jsou modelovány pomocí mikroploškové BSDF, kde distribuce plošek je založena na statistické distribuci plošek mořské hladiny. Bio-optický model vody je parametrizován koncentrací planktonu, organických a anorganických nečistot. Voda je zobrazena pomocí aproximace difůzním rozptylem, konkrétně metodou isotropního dipólu (v kombinaci s jednonásobným rozptylem) a směrového dipólu. Rychlost konvergense obou metod je porovnána s volumetrickým sledováním cest.

**Klíčová slova:** vodní plochy, renderování, realistické zobrazování, BSDF, mikroplošky, průhledná prostředí, difůzní rozptyl, dipól, Mitsuba, rozptyl pod povrchem

**Vedoucí:** Ing. Jaroslav Sloup

## Abstract

This thesis focuses on rendering realistic images of mostly still water bodies, such as lakes or oceans with only small waves on the surface. Possible uses are especially in the movie industry, as simulators or computer games often need to sacrifice realism to achieve interactive framerates. The appearance of water bodies differs quite substantially based on not only the shape of the surface but also on the water constituents and the color of incident light. An analytical model of sun and sky radiance is used to light the scene. The shape of the surface is obtained from a sea wave spectrum using the inverse Fourier transform. Reflective properties of the surface are modeled by a microfacet BSDF, where the microfacet slope distribution is based on statistical sea slope distribution. The bio-optical model of the water is parametrized by the concentration of phytoplankton, common dissolved organic material (CDOM), and inorganic particulate matter. The scenes are rendered using diffusion approximations, specifically isotropic dipole (in combination with single scattering) and directional dipole. The convergence rate of both methods is compared with volumetric path tracing.

**Keywords:** water surface, realistic rendering, BSDF, microfacet, participating media, diffusion, dipole, Mitsuba, subsurface scattering

**Title translation:** Realistic rendering of water surfaces

# Contents

# Figures

## Tables

# Symbols

| | |
|---|---|
| $\omega$ | direction 7 |
| $E$ | irradiance 7 |
| $L$ | radiance 7 |
| $f_r$ | BRDF 8 |
| $f_t$ | BTDF 8 |
| $\eta$ | index of refraction 9 |
| $f_s$ | BSDF 9 |
| $\xi$ | pseudorandom variable 10 |
| $F$ | Fresnel factor 11 |
| $S$ | BSSRDF 14 |
| $\sigma_a$ | absorbtion coefficient 15 |
| $\sigma_s$ | scattering coefficient 15 |
| $p$ | phase function 15 |
| $\sigma_t$ | attenuation coefficient 16 |
| $\tau$ | transmittance 16 |
| $\lambda$ | wavelength 24 |
| $M$ | radiant exitance 35 |

# Chapter 1

# Introduction

Water appears on Earth in many settings, be it a tiny clear stream in the mountains with occasional waterfalls, a not so deep river slowly flowing through a city, whose bottom can not be seen, or we could abandon moving waters and picture instead a clear mirror-like mountain lake or fjords somewhere in Norway, a muddy fishing pond behind a village, or a puddle in front of your house. But probably the most notable are oceans and seas, which have an entirely different range of looks: a whole scale of blue from light blue to almost black, or during summer transitioning to green due to algae. Also the waves have various shapes and sizes, depending on the wind speed and changes in direction – small waves can have long and smooth or short and sharp crests, and when the direction of wind changes, square waves can form. At higher wind speeds whitecaps may occur (Fig. 1.2). The waves start to roll and break, causing sprays and foam. When the waves get closer to the shore, they are pushed out of the water by the ocean floor, which looks as if the waves were growing, but instead, only their full height is being revealed.

This work focuses on rendering static images of mostly still water bodies (Fig. 1.1), especially calm oceans, but may also be applicable to lakes and ponds or slowly flowing rivers. Only small waves are allowed to avoid breaking waves, sprays, and foam. Another important assumption is that the water is deep, as the rising waves near the shore and the shape of the coastline would add additional complexity. Instead, the work aims to accurately capture the color of the water, which is determined by the water constituents and the characteristics of the light incident on the water surface. The appearance of the water surface is determined by several components. Firstly, it is the light source, specifically the sun, with changing color throughout the day. But not all light comes directly from the sun – some is scattered in the atmosphere

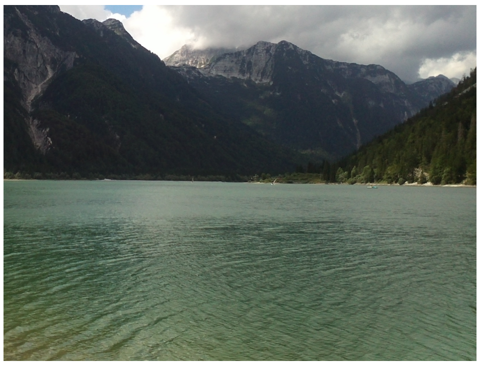


**Figure 1.1:** Clear lake with green tint.

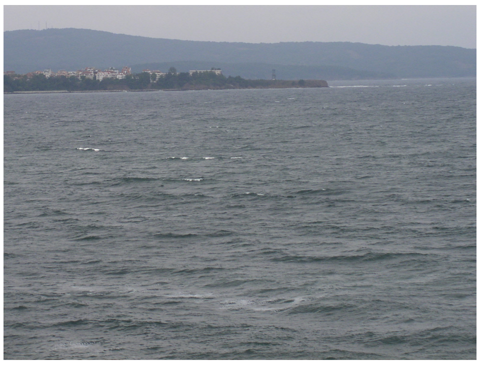


**Figure 1.2:** Sea with starting white-caps, wind around 5 m/s.

first and then hits the surface. The color and intensity of the scattered light depend on the amount of moisture and pollution in the air. When the light hits the surface, some of the incident light is reflected, possibly towards the camera, or it may bounce off of several waves before reaching the camera, while the remaining light is refracted into the water. How the light scatters at the water surface is determined by the shape of the surface represented by the geometric model, and a Bidirectional scattering distribution function (BSDF), which describes the scattering properties of the surface on a finer scale, not visible to the eye. The light that enters below the surface interacts with various particles, as it travels through the water until it leaves at a different point. This is what affects the perceived color of the water, and it is captured by the bio-optical model.

Chapter 2 covers methods of obtaining the geometric model of the water surface, followed by Chapter 3 discussing the finer scattering model of the surface, that is the BSDF – its definition and a particular model to be used in the implementation. Then comes Chapter 4 about modelling the propagation of light in participating media, that is the media in which the light undergoes absorption and scattering, such as fog, smoke, or water. It also describes techniques useful in some of the rendering methods, and a particular model, that will be used in the implementation. Chapter 5 starts by discussing common rendering methods for non-participating media and builds on them, followed by approximate methods based on diffusion. Then comes Chapter 6 explaining the choice of methods and how the content of preceding chapters fits together. Then follows Chapter 7 on the choice of the rendering framework – a basis for the implementation, and how the framework was extended and modified. Chapters 8 and 9 present the results and evaluation of the work, propose possible improvements and extensions and summarize what has been done. The theoretical foundations not presented here, such as radiometric units, Monte Carlo integration and explanation of importance sampling can be found in [PJH16].

# Chapter 2

# Geometry of the water surface

The ocean surface is, unlike that of lakes and ponds, rarely perfectly smooth and mirror-like. Even when no wind is blowing in the area, the ocean waves can travel a long way from a place where atmospheric conditions are entirely different. Also, the wind can change much faster than it takes for already formed waves to dissipate. This mixing of wave trains of various amplitudes and sizes can produce many distinct wave patterns on the water surface. Waves can also be caused by gravity, or foreign objects, which can complicate the matter even further. There are several categories of water surface models. Fluid dynamics models (those based on Navier-Stokes equations) are the most general, but the most expensive to compute. They are typically used to simulate water near the shores – growing and breaking waves and spray, for a list of methods see the survey by Darles et al. [DCGG11]. The methods of modelling a deep-ocean surface are often based on measurements from oceanography. They are usually easy to implement and efficient to compute. Peachey [Pea86] uses superposition of sinusoids to obtain a heightfield, which does not allow for wave curling. He uses particle systems to model sprays. The heightfield can also be generated with Perlin noise [Per85], which does not have a physical basis, but still can produce good looking results. Fournier and Reeves [FR86] use a parametric trochoidal representation of the surface, allowing them to display curling waves. They also use particle systems for sprays. Another way to obtain the heightfield is from the frequency domain, based on sea wave frequency spectrum, with inverse Fourier transform [T$^+$01] – a method introduced to computer graphics by Mastin et al. [MWM87]. This chapter presents two of the mentioned approaches usable for open waters with small waves, without any interaction with other objects: Perlin noise and Fourier synthesis from the sea wave spectrum, along with a wave-height model which will be useful to scale the wave heights.

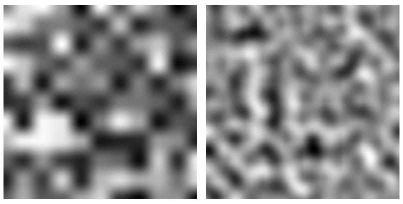


**Figure 2.1:** Value noise (left) and gradient noise. (from [Kob18])

## 2.1 Perlin noise

Perlin noise is a noise function introduced by Perlin [Per85, Per02], most commonly of form $\eta : \mathbb{R}^2 \mapsto [0, 1]$, but it can be generalized for an input vector of higher dimensions. It is computed by interpolation of gradients, therefore its gradients are smooth – see figure 2.1.

First, a uniform grid is created with a pseudo-random gradient vector in each vertex. The vertices are placed at integer coordinates. This is the initialization step. To evaluate the function at a given point:

1. Find the cell which the point falls into.
2. Compute the weight of each vertex of the cell. (see below)
3. Interpolate the vertex weights within the cell according to the fractional part of the point coordinates.

Note that linear interpolation does not produce smooth derivatives at the cell boundaries. The weight of the vertex is computed as the dot product of the gradient vector in that vertex with vector $pointCoord - vertexCoord$.

The number of different frequencies present in the noise ($\eta$) can be adjusted by summing several noise functions ($\eta_i$) with varying frequencies ($f$) and amplitudes ($a$), as demonstrated in Figure 2.2:

$$\eta(x) = \sum_i a_i \eta_i(f_i, x). \tag{2.1}$$

This scheme in combination with the Perlin noise is called fractional Brownian motion. The frequency and amplitude scaling is typically chosen so that higher frequencies have a smaller impact on the overall shape of the function:

$$a_i = \frac{a_{i-1}}{2}, \quad f_i = 2f_{i-1}. \tag{2.2}$$

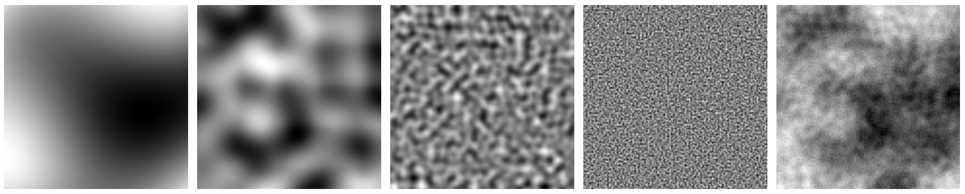


**Figure 2.2:** Four octaves with increasing frequency and decreasing amplitude and their sum. (from [Kob18])

To avoid aliasing, when the frequencies in the above sum reach the Nyquist limit, the higher frequency functions can be ignored, because their average value is zero. See [PJH16] for more detailed explanation and [EMP+03] for much more information and examples of procedural texturing and modelling.

## 2.2 Fourier synthesis

Although the results obtained with Perlin noise may look acceptable, there is no physical basis for this method. Differences are notable on a deep sea with wind waves, assuming constant wind direction. While the frequencies present in the fractal can be controlled, their direction, which is determined by the wind, can not.

Given a wave spectrum of the sea surface, the surface elevation can be obtained easily. A white noise image is transformed into the frequency domain using a discrete Fourier transform. The spectrum amplitudes are then filtered by the sea surface spectrum (see Figure 2.3). The filtered spectrum with the original phases is then transformed back to the time domain (Figure 2.4).

Pierson and Moskowitz [PJM64] proposed a model of the wave spectrum of a fully developed sea surface. A fully developed sea is such that with constant wind and sufficient fetch (the distance on the water surface over which the wind is blowing) the spectrum no longer changes. The downwind power spectrum is

$$F_{PM}(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} e^{-\frac{5}{4}(\frac{f_m}{f})^4}, \tag{2.3}$$

with frequency $f$ in Hz, $f_m = 0.13\frac{g}{u_{10}}$ being the peak frequency, $\alpha = 0.0081$ the Phillips constant, g the gravitational constant, and $u_{10}$ is the wind speed 10 meters above the sea surface. Hasselmann et al. [HDE80] improve the model by accounting for the wind direction, suppressing the peak frequency crests parallel to the wind direction:

$$F(f,\theta) = F_{PM}(f)D(f,\theta), \tag{2.4}$$

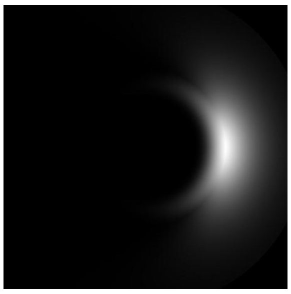


**Figure 2.3:** Water surface spectrum filter. The spectrum is encoded in polar coordinates: distance from the center represents the frequency, the angle translates to the direction of the wave. Here the wind blows to the right. Crests parallel to the wind directions are suppressed.

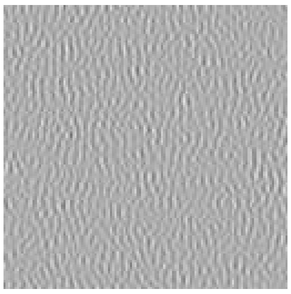


**Figure 2.4:** Water surface heightmap obtained using FFT.

where $\theta$ is the direction of a wave relative to the downwind direction and

$$D(f, \theta) = N_p^{-1} cos^{2p} \left( \frac{\theta}{2} \right), \tag{2.5}$$

$$p = 9.77 \left( \frac{f}{f_m} \right)^{\mu}, \quad \mu = \begin{cases} 4.06, & f < f_m \\ -2.34, & \text{otherwise} \end{cases} \tag{2.6}$$

and the normalization constant

$$N_p = 2^{1-2p} \pi \frac{\Gamma(2p+1)}{\Gamma^2(p+1)}. \tag{2.7}$$

## 2.3 Wave height

Fourier synthesis provides a physically based surface heightfield, as far as frequency of waves and their directions are concerned. The wave amplitudes, however, are not part of the model. According to Svedrup and Munk [SM47], the relation between significant wave height of the highest waves on a fully developed sea and the wind speed is:

$$H_{m,fd} = \frac{0.3}{g} U^2. \tag{2.8}$$

Significant wave height is defined as the mean wave height (trough to crest) of the highest one-third waves that occur in a given period. Many waves in that period will be smaller, and approximately one in ten waves will be twice as high.

# Chapter 3

# Surface scattering model

The light incident on a surface scatters into various directions. The fraction of incident light from one direction and scattered into another direction depends on the material properties. Reflective properties of a material can be modeled by a Bidirectional reflectance distribution function (BRDF), while the transmissive properties are modeled by a Bidirectional transmittance distribution function (BTDF). The overall scattering on the surface is therefore modeled as a sum of BRDFs and BTDFs, giving a Bidirectional scattering distribution function (BSDF).

This chapter starts by defining the BRDF and the BTDF. Then it covers the microfacet theory, which is the basis for many microfacet models, along with the general way of sampling them, which is a practical necessity for Monte Carlo rendering methods. Then follows a description of a microfacet-based water surface BSDF model, as well as the derivation of sampling equations for it. Finally, there is also a definition of BSSRDF, which will be useful for diffusion-based rendering methods.

## 3.1 BRDF

A BRDF is a function describing the reflective properties of a surface [PJH16]. It is defined as a fraction of reflected radiance in direction $\omega_o$ due to differential irradiance $dE(x, \omega_i) = L_i(x, \omega_i)\cos(\theta_i)d\omega_i$, which is the result of incident

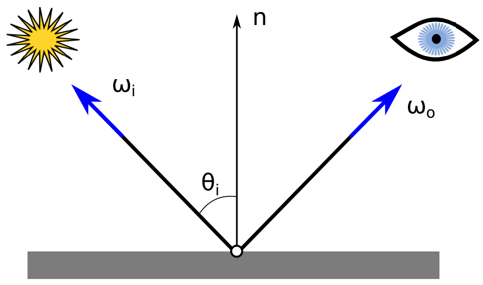


**Figure 3.1:** BRDF: $\omega_i$ is the direction towards the light source, $\omega_o$ is the direction towards the viewer, $n$ is the surface normal.

radiance from $\omega_i$:

$$f_r(x, \omega_i, \omega_o) = \frac{dL_o(x, \omega_o)}{L_i(x, \omega_i) \cos\theta_i \, d\omega_i}, \tag{3.1}$$

where $x$ is a point on the surface, $\omega_i$ is a direction towards a light source and the $\omega_o$ is the direction towards the viewer, and $\theta_i$ is the angle between $\omega_i$ and the surface normal – see figure 3.1. It has the following properties:

- non-negativity: $f_r(x, \omega_i, \omega_o) \geq 0$,
- reciprocity: $f_r(x, \omega_i, \omega_o) = f_r(x, \omega_o, \omega_i)$,
- conservation of energy: $\int_H f_r(x, \omega_i, \omega_o) \; cos\,\theta_o \, d\omega_o \leq 1$.

Conservation of energy ensures that the total energy reflected in all directions $\omega_o$ can not be greater than the energy received from $\omega_i$. The integral is over all directions $\omega_o$ on a hemisphere above the surface.

## 3.2 BTDF

As mentioned above, BTDF is a function that describes the fraction of energy refracted from the incident into the outgoing direction [PJH16]. It is defined similarly to BRDF, but the reciprocity rule differs due to the solid angle compression: when the light refracts into a denser medium (a medium with a higher index of refraction), it is compressed into a smaller solid angle. In the opposite case, that is when refracting from a denser medium, a total reflection occurs at incident angles past the critical angle. Therefore the symmetry rule for the BTDF becomes:

$$f_t(x, \omega_i, \omega_o) = \frac{\eta_o^2}{\eta_i^2} f_t(x, \omega_o, \omega_i), \tag{3.2}$$

with $\eta_i$ and $\eta_o$ being the indices of refraction of the medium containing the incident ray, and the outgoing ray respectively.

## 3.3 Microfacet models

There are many different BRDFs and BTDFs. Some of them are derived based on the so-called microfacet theory [PJH16]. It means that the surface is modeled as a collection of many flat microfacets, which are usually chosen to be ideally diffuse (Oren-Nayar BRDF [ON94]), or specular (Cook-Torrance [CT82]). The roughness of such a surface is described by a statistical distribution of microfacet normals $D(f)$. A closed-form expression for the surface BRDF or BTDF is then derived from the BxDF of the microfacets and the microfacet normal distribution, taking into account the shadowing and hiding of the facets (the $G(\omega_i, \omega_o, f)$ term), as can be seen in Figure 3.2:

$$f_s(\omega_i, \omega_o) = \int \left|\frac{\omega_i \cdot f}{\omega_i \cdot n}\right| f_s^m(\omega_i, \omega_o, f) \left|\frac{\omega_o \cdot f}{\omega_o \cdot f}\right| G(\omega_i, \omega_o, f)\, D(f)\, d\omega_f, \qquad (3.3)$$

where $f$ is the microfacet normal, and $f_m$ is the BSDF of the microfacets. The first of the two fractions converts macrosurface irradiance to the microsurface, while the second transforms the scattered radiance back to the macrosurface. The shadowing and hiding is necessary to maintain energy conservation. In the rendered image, its effect is visible especially for rough surfaces and at grazing angles. Interreflections between the microfacets are usually not taken into account, Heitz et al. try to address this in [HHdD16]. Specular transmission can also be used to model translucent materials [PJH16].

### 3.3.1 Sampling microfacet BSDFs

The equation 3.3, or better yet a closed-form expression derived from it, can be used to evaluate the BSDF. But for some rendering methods that is not enough – Monte Carlo rendering algorithms rely on finding paths between the camera and a light source with high throughput. Therefore they need to, given an incident direction $\omega_i$, randomly select an outgoing direction $\omega_o$ with probability density function $p_{\omega_o} \propto f_s(\omega_i, \omega_o)|\cos\theta_o|$. This is called importance sampling, and it helps to find the high-throughput paths in the scene. It is a popular and practically necessary variance reduction technique for MC rendering algorithms [Wei00]. The sample then needs to be weighted by $\frac{1}{p_{\omega_o}(\omega_o)}$.

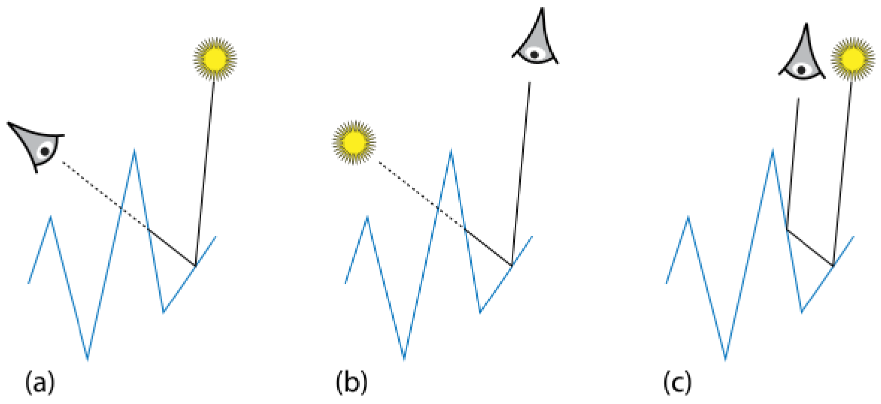


**Figure 3.2:** The geometric effects occurring on a microfacet surface. (a) Microfacet hiding (or masking) – the viewer can not see the microfacet, because it is occluded by another microfacet. (b) Shadowing – The light does not reach the microfacet. (c) Interreflections – the light reflects between the microfacets to reach the viewer. (from [PJH16])

A general approach for sampling 2D functions, as described for instance in [PJH16], is:

1. normalize the function $f$ to obtain a PDF: $p(u,v) = \frac{f(u,v)}{\int\int f(u,v)\,du\,dv}$,
2. compute the marginal density for one of the variables: $p_u(u) = \int p(u,v)\,dv$,
3. compute the conditional density for the other variable: $p_v(v|u) = \frac{p(u,v)}{p_u(u)}$,
4. compute the CDF of both the marginal and the conditional density: $P(x) = \int_{-\infty}^{x} p(q)\,dq$,
5. invert both CDFs, thus obtaining expressions in the form: $u = g_1(\xi_1)$, $v = g_2(u,\xi_2)$, where $\xi$ is a random number in range $[0,1]$.

Most BSDFs can not be sampled this way, because they can not be integrated. In practice, it is often sufficient to sample a significant factor, which for microfacet models is the microfacet normal distribution [WMLT07]. In that case, the probability density function with respect to the microfacet normal has to be converted to a PDF with respect to $\omega_o$. According to Walter [Wal05]:

$$p_{\omega_o}(\omega_o) = p_f(f)\left\|\frac{\partial\omega_f}{\partial\omega_o}\right\| = \frac{p_f(f)}{4(f\cdot\omega_i)}. \tag{3.4}$$

After sampling the microfacet normal, the Fresnel term $F(\omega_i, f)$ can be used to select between reflection and refraction, and it has to be added to the PDF. Then the reflected or transmitted direction can be calculated.

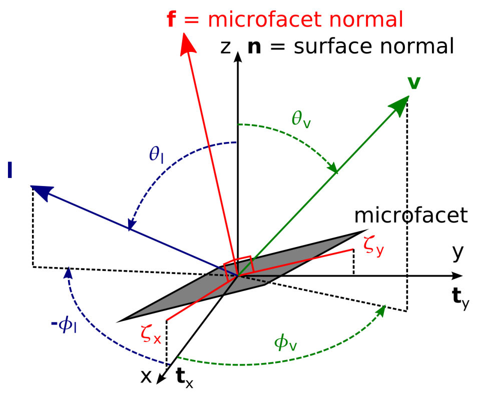


**Figure 3.3:** Microfacet coordinates: $v$ points towards the viewer, $l$ towards the light source, $\zeta_x$ and $\zeta_y$ are the slopes of the facet. The $x$ axis is oriented along with the wind, and the $z$ axis points towards the zenith. (from [RDP05])

This sampling scheme is inefficient for grazing angles, in which case nearly half of the sampled normals have to be rejected because they are facing the opposite direction from the incident ray. Also, the weights are unbounded, which can cause fireflies. Heitz and d'Eon came up with a different scheme that samples only the visible normals [Hd14].

## 3.4 Water surface BSDF

Based on this microfacet theory, Ross et al. [RDP05] derived a BRDF from the distribution of sea microfacet slopes. It takes into account the direction and speed of the wind, and can be evaluated analytically:

$$f_r(v, l) = \frac{q_{vn}(\zeta|v, l)\, F(f \cdot v)}{4 f_z^3 (f \cdot v) \cos \theta_l} \tag{3.5}$$

where $f$, $l$ and $v$ are the microfacet normal, the direction towards the light source and direction towards the viewer, respectively. See figure 3.3. The microfacets are considered to be perfect mirrors, therefore the microfacet normal is the half vector between $l$ and $v$. The Fresnel factor $F(u)$ can be according to Schlick [Sch94] approximated as

$$F(u) \approx R_0 + (1 - R_0)(1 - u)^5, \tag{3.6}$$

with $R_0$ being the reflectivity of water-air interface for normal incidence calculated from indices of refraction of water and air ($n_1$ and $n_2$) as

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2. \tag{3.7}$$

The normalized probability that a facet with slope $\zeta$ will occur, will interact with the incoming ray, and will be visible from the viewer direction, is defined as:

$$q_{vn}(\zeta|v,l) = \frac{p(\zeta)\,W(\zeta,v)\,H(\zeta,v)}{(1+\Lambda(a_v)+\Lambda(a_l))\cos\theta_v}, \tag{3.8}$$

where $W$ gives the projection weight, and $H$ accounts for not seeing the back of the wave:

$$W(\zeta,v) = \frac{f\cdot v}{n_z}, \quad H(\zeta,v) = \gamma(f\cdot v). \tag{3.9}$$

with $\gamma$ being the Heaviside function, which is equal to one for values greater than zero, and zero otherwise.

$\Lambda$ comes from the Smith wave-height-hiding function [Smi67b, Smi67a] and is given by:

$$\Lambda(a) = \frac{\exp(-a^2) - a\sqrt{\pi}\,\mathrm{erfc}(a)}{2a\sqrt{\pi}}, \quad a_{x\in\{v,l\}} = \frac{\cot(\theta_x)}{\sqrt{2}\,\sigma(\phi_x)} \tag{3.10}$$

The microfacet slope PDF is described by a gaussian (multiplied by kurtosis and skewness correction terms, omitted here for brevity):

$$p(\zeta|w) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left(\frac{\zeta_x^2}{\sigma_x^2}+\frac{\zeta_y^2}{\sigma_y^2}\right)\right), \tag{3.11}$$

where $w$ is the wind speed and the slope variances are

$$\sigma_x^2 = 0.00316w, \quad \sigma_y^2 = 0.003 + 0.00192w, \tag{3.12}$$

and

$$\sigma^2(\phi) = \sigma_x^2\cos^2\phi + \sigma_y^2\sin^2\phi. \tag{3.13}$$

As is the case for most of the microfacet models, this model also ignores the reflections between microfacets, which according to the authors should not cause errors higher than 2 to 3 %. The largest errors are for low Sun elevation, high wind speed, and near-horizontal viewer orientation. Heitz et al. derived the missing multiple scattering component for Smith microsurface model [HHdD16], which could possibly be used to solve this.

Based on this BRDF, Ma et al. [MXW$^+$16] derived a BTDF:

$$f_t(v,l) = \frac{q_{vn}^l(\zeta|l)\,F_t(f\cdot l)}{4f_z^3(f\cdot l)\cos\theta_l}, \tag{3.14}$$

where $q_{vn}^l$ is the normalized visibility probability distribution of the light source:

$$q_{vn}^l(\zeta|l) = \frac{p(\zeta)\,H(f\cdot l)}{1+\Lambda(a_l)}. \tag{3.15}$$

### 3.4.1 Sampling the water surface BSDF

As mentioned before, being able to evaluate the BSDF is not enough for MC algorithms, therefore, following the steps in section 3.3.1, the sampling equations for the water surface BSDF were derived. The marginal microfacet slope distribution in the x direction is

$$p_{\zeta_x}(\zeta_x) = \int p(\zeta_x, \zeta_y)\, d\zeta_y = \sqrt{\frac{\pi}{2}} \frac{1}{\pi\sigma_x} e^{-\frac{1}{2}\left(\frac{\zeta_x}{\sigma_x}\right)^2}, \tag{3.16}$$

the CDF is then

$$P_{\zeta_x}(\zeta_x) = \int_{-\infty}^{\zeta_x} p_{\zeta_x}(x)\, dx = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\zeta_x}{\sqrt{2}\,\sigma_x}\right)\right] \tag{3.17}$$

and the CDF inverse for $P_{\zeta_x}(\zeta_x) = \xi_1$:

$$\zeta_x = \sqrt{2}\,\sigma_x \operatorname{erf}^{-1}(2\xi_1 - 1). \tag{3.18}$$

The conditional probability of $\zeta_y$ is

$$p_{\zeta_y}(\zeta_y|\zeta_x) = \frac{p(\zeta_x, \zeta_y)}{p_{\zeta_x}(\zeta_x)} = \sqrt{\frac{\pi}{2}} \frac{1}{\pi\sigma_y} e^{-\frac{1}{2}\left(\frac{\zeta_y}{\sigma_y}\right)^2}, \tag{3.19}$$

which is the same as $p_{\zeta_x}$ (after replacing the $x$ index with $y$). This means that the facet slope in the $y$ direction does not depend on the slope in the $x$ direction, and the sampling equation will also be the same.

Now the slope PDF needs to be transformed to outgoing direction PDF:

$$p_{\omega_o}(\omega_o) = \frac{p_f(f)}{4(f \cdot \omega_i)} \tag{3.20}$$

$$\begin{aligned} p_f(f) &= p_\zeta(\zeta_x(\theta_f, \phi_f), \zeta_y(\theta_f, \phi_f)) \left\| \frac{\partial[\zeta_x, \zeta_y]}{\partial[\theta_f, \phi_f]} \right\| \frac{1}{\sin\theta_f} \\ &= p_\zeta(\zeta_x(\theta_f, \phi_f), \zeta_y(\theta_f, \phi_f)) \frac{1}{\cos^3\theta_f} \end{aligned} \tag{3.21}$$

## 3.5 BSSRDF

Another function that is similar to those defined previously is a Bidirectional Scattering-Surface Reflectance Distribution Function (BSSRDF). It will be

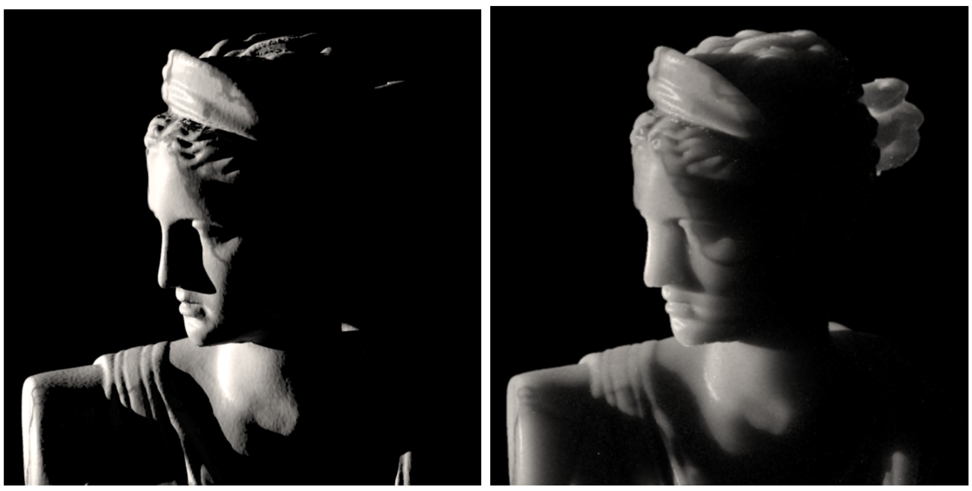


**Figure 3.4:** Subsurface scattering in a marble bust. A comparison of pure reflections (left) and full volumetric simulation (right). (from [JMLH01])

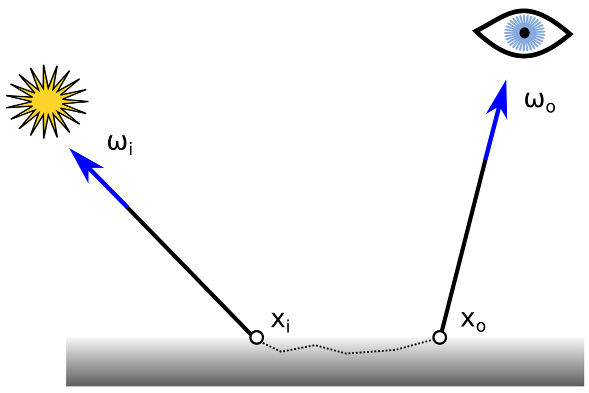


**Figure 3.5:** A BSSRDF captures the light traveling under the surface of the material, exiting at a different point than where it entered.

useful later to lay out the interface of diffusion-based rendering methods. It generalizes the BRDF in the sense that the light incident on a surface can leave the surface at a different point – see Figure 3.5. This is important for materials such as marble or skin, as can be seen from Figure 3.4. The BSSRDF is defined as [NRH$^+$77]:

$$S(x_i, \omega_i, x_o, \omega_o)) = \frac{dL_o(x_o, \omega_o)}{d\Phi_i(x_i, \omega_i)}. \tag{3.22}$$

Therefore to obtain the radiance leaving a surface at a given point, one must integrate not only over the incident directions but also over the area surrounding the point. This function can thus be thought of as a combination of BTDF for the inward and outward direction and another function for the propagation within the material.

Chapter 4

# Participating media

The previous chapter started with an assumption of a transparent medium, such as air, pure water, or glass, in which the light travels in a straight line, so reflection and refraction on surfaces is the only concern. At the end it shifted towards materials that seem opaque or almost opaque, but the light can enter the material and travel under the surface for a short distance. Note that these materials are usually considered to have constant density. Similar to that, although much optically thinner are translucent media, such as smoke, clouds, or polluted water, in which the light also can not travel in a straight line. But in this case, the density is often spatially varying. Both of these scenarios can be modeled as a participating medium, i.e. a medium which affects the transport of light. The chapter starts by formalizing this in the form of the light transport equation. Then the types of scattering are presented, along with the concept of a phase function, which is used to represent the directional distribution of light after it undergoes scattering. The next sections cover the estimation of transmittance along a path segment, and distance sampling, which is used in MC algorithms. Finally, a model of water as a participating medium is described.

The possible interactions of light with the medium are absorption, emission, and scattering [PJH16]. When thinking in terms of light paths (as outside the medium), in case the scattering causes the light to change direction into the light path, this is called in-scattering. The other case, when the light leaves the path, is called out-scattering. Therefore the medium can be characterized by an absorption coefficient $\sigma_a$ $[m^{-1}]$ and a scattering coefficient $\sigma_s$ $[m^{-1}]$, which represent the probability density of a photon being absorbed or scattered per unit of distance traveled. How much the light scatters in which direction is modeled by a phase function $p(\omega_i, \omega_o)$. It gives the ratio of

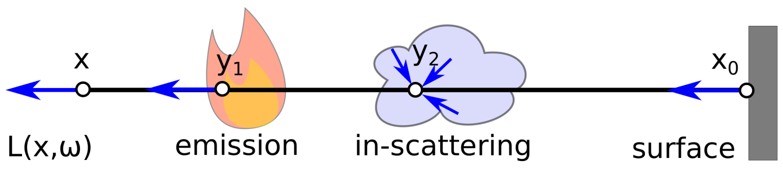


**Figure 4.1:** Terms of the volume rendering equation. The flame depicts an emissive medium, while the cloud represents a scattering medium, both without refractive boundaries. A medium contributes radiance into the ray along the entire ray segment inside the medium. (inspired by [NGHJ18])

incident radiance from $\omega_i$ scattered in $\omega_o$ direction. The change of radiance in direction $\omega$ at point $x$ due to these interactions is given by the radiative transfer equation (RTE) [Cha60]:

$$\begin{aligned}\frac{dL(x,\omega)}{dx} = \underbrace{\sigma_a(x)\, L_e(x,\omega)}_{\text{emission}} + \underbrace{\sigma_s(x) \int_S L(x,\omega_i)\, p(\omega,\omega_i)\, d\omega_i}_{\text{in-scattering}} \\ - \underbrace{\sigma_a(x)\, L(x,\omega)}_{\text{absorption}} - \underbrace{\sigma_s(x)\, L(x,\omega)}_{\text{out-scattering}},\end{aligned} \tag{4.1}$$

where $\int_S$ denotes an integral over directions on a sphere. The coefficients of the inhibiting terms can be integrated separately (with the attenuation coefficient $\sigma_t = \sigma_a + \sigma_s$, sometimes also called extinction coefficient), giving the transmittance:

$$\tau(x_0, x) = e^{-\int_{x0}^{x} \sigma_t(u)\, du}, \tag{4.2}$$

which represents the fraction of light traveling from $x_0$ along a straight line that reaches the point $x$. The RTE is derived for an infinite medium, but in virtual scenes, the medium is typically bounded and also may contain objects inside it. By integrating the RTE and taking into account the boundary conditions, the volume rendering equation is obtained [NGHJ18]:

$$L(x,\omega) = \underbrace{\tau(x_0,x)L(x_0,\omega)}_{\text{surface term}} + \underbrace{\int_0^{t_{max}} \tau(x - t\omega, x)L_s(x - t\omega, \omega)dt}_{\text{medium term}}. \tag{4.3}$$

It gives radiance in direction $\omega$ at point $x$ in space, see Figure 4.1. In the above equation, $x_0 = x - t_{max}\omega$ is the point on the first intersected surface from point $x$ in direction $-\omega$ and $L_o$ is the radiance leaving that surface. The source term is:

$$L_s(y,\omega) = \underbrace{\sigma_a\, L_e(y,\omega)}_{\text{emission}} + \underbrace{\sigma_s(y) \int_S p(\omega',\omega)\, L(y,-\omega')\, d\omega'}_{\text{in-scattering}}. \tag{4.4}$$

| | | |
|---|---|---|
| $\kappa \ll 1$ | Rayleigh scattering | particles much smaller than $\lambda$ |
| $\kappa \approx 1$ | Mie scattering | particles about the same size as $\lambda$ |
| $\kappa \gg 1$ | geometric scattering | particles much larger than $\lambda$ |

**Table 4.1:** Models of elastic light scattering based on particle size. $\kappa = \frac{2\pi r}{\lambda}$, where $r$ is the radius of the particle, $\lambda$ is the wavelength.

## 4.1 Scattering and phase functions

From the three types of interaction, absorption and emission are quite straightforward, but the scattering is slightly more complex. Scattering is a change of direction of light, which happens when the EM wave impacts molecules or particles. If the wavelength changes due to the impact, the scattering is called inelastic. If it stays constant, it is called elastic. The propagation of light is described by Maxwell's equations, but there are also simpler models of scattering. Depending on the size of the particle or molecule, the scattering can be modeled by Rayleigh theory, Mie theory, or geometric optics [HT74, vdH81], see table 4.1.

The directional distribution of scattered light in a particular medium can be described by a phase function $p(\omega_i, \omega_o)$ – a fraction of incident radiance from $\omega_i$ scattered to $\omega_o$ direction [PJH16]. Important properties of a phase function are reciprocity and conservation of energy: $p(\omega_i, \omega_o) = p(\omega_o, \omega_i)$, $\frac{1}{4\pi}\int_S p(\omega_i, \omega_o)\, d\omega_o = 1$. When the light is scattered into all directions equally, then $p(\omega_i, \omega_o) = \frac{1}{4\pi}$. This function is called the isotropic phase function. Some phase functions depend only on the angle $\theta$ between $\omega_i$ and $\omega_o$, this is the case of isotropic media. The scattering in a medium can also be described by a volume scattering function, which is the phase function multiplied by the scattering coefficient.

There are several analytical models of phase functions. One of them is the Henyey-Greenstein [HG41]:

$$p_{HG}(\theta, g) = \frac{1}{4\pi} \cdot \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}}, \tag{4.5}$$

where $g$ is the asymmetry parameter (also called mean cosine). Positive $g$ corresponds to forward scattering (see Figure 4.3, negative to backscattering and for $g = 0$ the function becomes the isotropic phase function.

The Rayleigh scattering is wavelength dependant – shorter wavelengths scatter more than longer ones, which is the reason for the blue color of the sky and the red Sun at sunset [Cha60]. For the purposes of computer graphics,

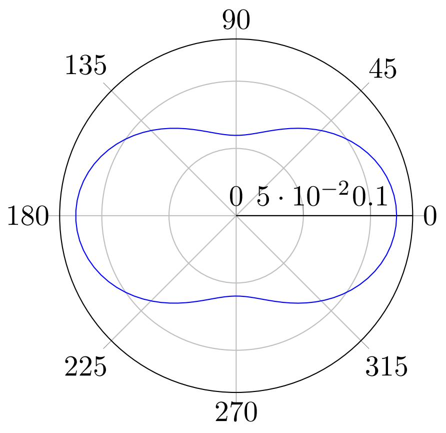


**Figure 4.2:** Rayleigh phase function.

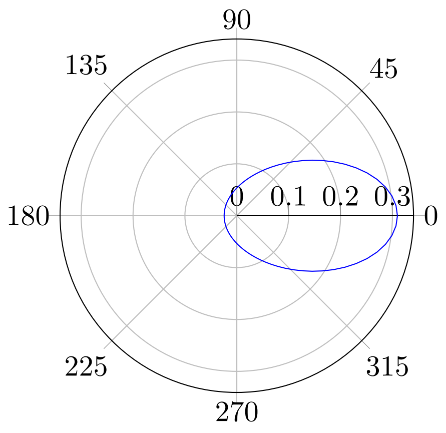


**Figure 4.3:** Henyey-Greenstein phase function with $g = 0.4$.

the wavelength dependence is often removed, and a normalizing factor is added to obtain the phase function [Fri11] (Fig. 4.2):

$$p_R(\theta) = \frac{1}{4\pi}\frac{3}{4} \cdot (1 + \cos^2\theta). \tag{4.6}$$

## 4.2 Transmittance estimation

As illustrated above, the rendering equation (Eq. 4.3) consists of two terms: the surface term and the source term. To evaluate the source term, transmittance between two points needs to be computed. It can be done according to Equation 4.2, where the integral in the exponent is called optical thickness. One group of techniques used to compute the transmittance is by computing the optical thickness integral. For homogeneous media it becomes a mere multiplication. For heterogeneous media, standard integration methods can be used, for example Riemann summation, trapezoid rule, or Simpson's rule, which all lead to an overestimated result. Computing the optical thickness by a Monte Carlo integration is unbiased, but according to [NGHJ18] using an estimate of optical thickness computed by MC integration with random or stratified sampling both lead to a biased estimate of transmittance. There are two more general ways of estimating transmittance: some of the distance sampling techniques can be transformed to estimate transmittance, as well as some null-collision algorithms. For more details, see the survey by Novák et al. [NGHJ18].

## 4.3 Distance sampling

The second part of the rendering equation is the medium term. It contains an integral over a distance with transmittance as part of the integrand, which means that to be able to estimate it efficiently, a method of importance sampling the distance along a ray with respect to the transmittance is needed. There are many methods of doing this. The choice depends on whether the density is constant throughout the medium or not, and also if the density is wavelength-dependant. More methods and their categorization can be found in the recent STAR report by Novák et al. [NGHJ18].

### 4.3.1 Homogeneous media

The simplest case is that of the homogeneous medium (constant density) while considering only a single wavelength (or spectrally uniform attenuation coefficient). The transmittance decreases exponentially with the distance ($\tau(t) = e^{-\sigma_t t}$), so the distance along the ray can be sampled as

$$t = -\frac{ln(1-\xi)}{\sigma_t}, \tag{4.7}$$

with PDF

$$p_t(t) = \sigma_t \, e^{-\sigma_t t}. \tag{4.8}$$

The PDF of sampling the surface term is then complementary

$$p_{surf}(t) = 1 - \int_0^{t_{max}} p_t(t)\, dt. \tag{4.9}$$

In cases when the attenuation coefficient varies between wavelengths, first, the channel is chosen with uniform probability, and then the corresponding scalar value is used to sample the distance. The PDF then becomes the average of the individual PDFs.

### 4.3.2 Heterogeneous media

With density varying throughout the medium, things get more complicated. When the density remains at least locally constant, the medium can be decomposed into voxels with constant density, and the above approach can be applied to the individual voxels. This technique is called regular tracking.

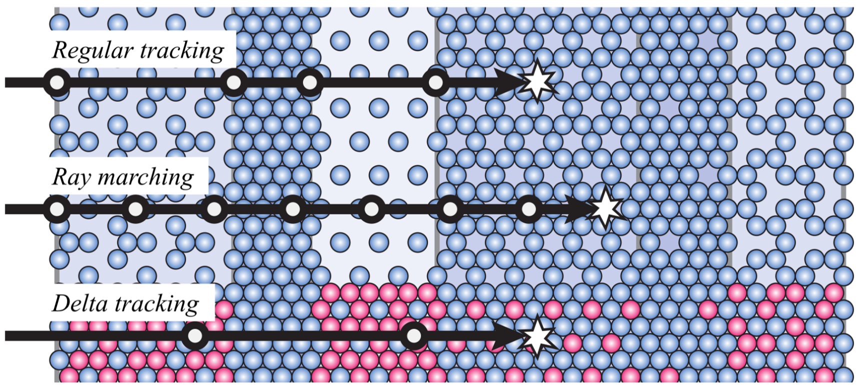


**Figure 4.4:** Distance sampling of a medium composed of homogeneous slabs. Regular tracking finds the boundaries of each slab and then uses the sampling method for homogeneous media. Ray marching steps through the medium in fixed steps, accumulating the optical thickness until a predetermined (sampled) value is achieved. Delta tracking adds virtual particles (here pink) to homogenize the medium and then probabilistically rejects samples based on the local density of the fictitious matter. (from [NGHJ18])

Another possibility is to use ray marching, which discretizes the density along the ray into a piecewise-constant or piecewise-linear function. It works by first randomly choosing a value of the optical thickness, and then marching along the ray while accumulating the thickness along the ray until the predetermined value is reached. This approach introduces bias, which can also form visual artifacts, that is why delta tracking is preferred. All three methods are shown in Figure 4.4.

## Delta tracking

Delta tracking, also known as Woodcock tracking or the null-collision algorithm, introduced by Woodcock et al. [WMHL65] is an unbiased technique based on von Neumann's rejection sampling [VN51]. The approach can be thought of as adding virtual particles into the medium to unify the density, therefore obtaining a homogeneous medium. A similar sampling scheme can be applied, but each sample is accepted randomly based on the fraction of real particles at that point, resulting in the correct distribution of samples. If the sample is not accepted, a new one is calculated as

$$t_i = t_{i-1} - \frac{\ln(1-\xi)}{\sigma_{t,max}}, \tag{4.10}$$

where $t_0 = t_{min}$ and $\sigma_{t,max}$ is the maximum value of the attenuation coefficient throughout the medium. The method can not quantify the PDF of the sample, which could be needed for multiple importance sampling. Instead, it produces the ratio of transmittance to PDF value, which can be used in the integration directly.

The performance of null-collision methods can be increased by decreasing the number of null-collisions, because each null collision requires evaluation or lookup of the medium coefficients. This can be achieved with Decomposition tracking [KHLN17], which partitions the medium density into homogeneous control part that can be sampled analytically (and in which case the coefficients do not have to be looked up again or recomputed) and a residual part. A tighter fit of the control part (or of $\sigma_{t,max}$ in case of delta tracking) can be achieved by spatial partitioning of the medium and fitting locally in each voxel [NGHJ18].

## Spectral tracking

The delta tracking can produce correct results even when the distribution of samples is not proportional to the transmittance, but the samples need to be reweighted. This technique is called Weighted delta tracking [CCT72, Cra78]. Spectral tracking introduced by Kutz et al. [KHLN17] utilizes these weights, extending the weighted delta tracking to handle multiple wavelengths. Algorithm 1 shows how spectral tracking samples the distance $t$ from point $x_s$ along direction $\omega$ and transmittance to PDF ratio on a single path segment in the medium.

**Algorithm 1** SpectralTracking($x_s, \omega$)

```
$w \Leftarrow (1, \dots, 1)_{N_\lambda}$
$t \Leftarrow 0$
while true do
  $t \Leftarrow t - \frac{ln(1-\xi)}{\sigma_{t,max}}$
  $x \Leftarrow x_s + t\omega$
  if $\xi < P_a(x)$ then
    return $w \cdot \frac{\sigma_a(x)}{\sigma_{t,max} P_a(x)}$
  else if $\xi < 1 - P_n(x)$ then
    return $w \cdot \frac{\sigma_s(x)}{\sigma_{t,max} P_s(x)}$
  else
    $w \Leftarrow w \cdot \frac{\sigma_n(x)}{\sigma_{t,max} P_n(x)}$
  end if
end while
```

When there are multiple wavelengths, the scalar-valued maximum attenuation coefficient $\sigma_{t,max}$ is obtained by taking the maximum of the vector. The null-collision coefficient is defined as $\sigma_n = \sigma_{t,max} - \sigma_a - \sigma_s$. The collision probabilities $P_{\sigma \in \{\sigma_a, \sigma_s, \sigma_n\}}$ are then:

$$\begin{aligned} P_\sigma(x) &= \underset{\lambda}{max}(|\sigma(x,\lambda)|)\, c^{-1}, \\ c &= \sum_{\sigma \in \{\sigma_a, \sigma_s, \sigma_n\}} \underset{\lambda}{max}(|\sigma(x,\lambda)|). \end{aligned} \tag{4.11}$$

## 4.4 Optical properties of waters

To be able to model the water as a participating medium, the constituents of the water must be taken into account to obtain accurate absorption and scattering coefficients, and the phase function. These depend on several factors. Jerlov, in his book on ocean optics [Jer76], introduced a water classification into 12 types and assigned each of them a diffuse attenuation coefficient. The types from I to III correspond to open ocean waters from the clearest to the most turbid, and types 1 to 9 represent coastal waters ordered in the same way. Turbidity is caused by suspended particles in the water, which shortens the distance the light penetrates into the water body. Premoze and Ashikhmin [PA01] use a parameter called turbidity to interpolate between tabulated values of attenuation coefficients of Jerlov water types. They then use it to estimate scattering and absorption coefficients. Cerezo and Seron [CS02] focus on waters with a high concentration of phytoplankton. Since chlorophyll contained in phytoplankton significantly influences the optical properties of the water, they based their model on Morel's model of absorbption coefficient [Mor91], which relates the water optical properties to the chlorophyll concentration. Shi et al. [SZZW12] focus on polluted water, therefore their model is based around the concentration of CDOM (colored dissolved organic material) and inorganic suspended particulate matter. Ma et al. [MXW$^+$16] presented a model of the case I waters based on the previous two, description of which now follows.

### 4.4.1 Bio-optical model

The water may contain several kinds of constituents, as hinted above, which have an impact on the propagation of light in the water. Firstly, pure saltwater scatters light more than freshwater due to additional dissolved minerals. Water can also contain colored dissolved organic matter (CDOM), i.e. mostly decaying detritus, which absorbs light and makes the water appear brown to yellow. It also scatters light, but it is inelastic scattering, and the following model ignores it. Another water constituent is particulate

| component | absorption coef. | scattering coef. |
|---|---|---|
| pure water | $\alpha_w$ | $\sigma_w$ |
| phytoplankton | $\alpha_p$ | - |
| particulate matter | - | $\sigma_p$ |
| CDOM | $\alpha_c$ | - |

**Table 4.2:** Model components and their scattering and absorption coefficients. A dash means that the component does not absorb/scatter light.

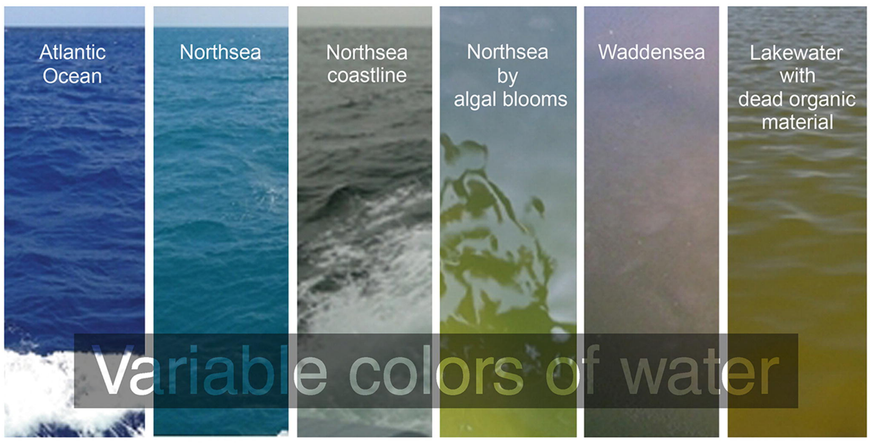


**Figure 4.5:** Influence of water constituents on the water color. Waters with a high concentration of phytoplankton appear green due to the absorption of red and blue wavebands (e.g. Northsea by algal blooms), while SPM produces a red to brown tint due to strong scattering in the red band (e.g. Waddensea). CDOM has similar optical properties to chlorophyll – the water color this ranges from yellow-green to brown (e.g. Lakewater with dead organic material). (from Marcel Wernard, NIOZ)

matter, both organic (viruses, colloids, bacteria, phytoplankton) and inorganic (created primarily by weathering of terrestrial rocks and soils), which causes highly forward scattering. The last component is phytoplankton, which causes significant absorption in the red band, and some absorption in the blue band as well. The impact of the water constituents on the water color is shown in Figure 4.5. The complete model consists of several elemental models of individual components, listed in table[1] 4.2. The two medium coefficients are, in this case, a function of wavelength. Additionally, this medium is isotropic, therefore $\beta(\theta) = \beta(\omega_i \cdot \omega_o)$. The total absorption and scattering coefficients can be obtained simply as a sum:

$$\alpha_s(\lambda, z) = \alpha_w(\lambda) + \alpha_p(\lambda, z) + \alpha_c(\lambda, z), \tag{4.12}$$

$$\sigma_s(\lambda, z) = \sigma_w(\lambda) + \sigma_p(\lambda, z). \tag{4.13}$$

[1]This section uses different notation to remain consistent with ocean optics literature and to improve readability.

The absorption coefficients of phytoplankton and CDOM are functions of wavelength ($\lambda$) and depth ($z$):

$$\alpha_p(\lambda, z) = 0.06\, \alpha_c^*(\lambda)\, C(z)^{0.05}, \tag{4.14}$$

$$\alpha_c(\lambda, z) = 0.012\, C(z)^{0.65} \exp[-0.014(\lambda - 440)]. \tag{4.15}$$

$\alpha_c^*\, [m^2/g]$ is a chlorophyll-specific absorption coefficient of phytoplankton [Mor91] – absorption cross-section per unit of chlorophyll. $C\, [g/m^3]$ is the vertical distribution of chlorophyll concentration:

$$C(z) = C_0 + \frac{h}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{z - z_{max}}{\sigma}\right)^2\right], \tag{4.16}$$

where $C_0$ is the background chlorophyll concentration, $h$ controls the maximal chlorophyll concentration, which is at a depth of $z_{max}$, and $\sigma$ is the standard deviation. The same equation is used to model the distribution of CDOM.

The scattering coefficient of particulate matter is:

$$\sigma_p(\lambda, z) = \frac{550}{\lambda}\, 0.3\, C(z)^{0.62}. \tag{4.17}$$

The absorption coefficient of pure water can be found in [PF97]. The scattering coefficients for pure seawater and freshwater are in [SB81].

The phase function is obtained as a weighted average by the scattering coefficients:

$$\beta(x, \omega_i, \omega_o) = \frac{\sigma_w}{\sigma_s(x_z)}\beta_w(\theta) + \frac{\sigma_p(x_z)}{\sigma_s(x_z)}\beta_p(\theta), \tag{4.18}$$

where $\beta_w$ and $\beta_p$ are the phase functions of pure seawater and particulate matter.

The phase function for pure water is, according to Morel [M+74]:

$$\beta_w(\lambda, \theta) = \widetilde{\beta}(\lambda, 90°)(1 + p\cos^2\theta), \tag{4.19}$$

where $p = 0.835$ is the polarization factor. Values of $\widetilde{\beta}(\lambda, 90°)$ differ for seawater and freshwater and can be found in [M+74].

Ma et al. [MXW+16] approximate the pure seawater phase function with the Rayleigh phase function as:

$$\beta_w(\theta) = \frac{3}{4\pi(3 + p)}(1 + p\, cos^2\theta). \tag{4.20}$$

The particle phase function $\beta_b(\theta)$ can be found in [MGG+93]. It is derived from Petzold's measurements described in [Pet72].

# Chapter 5

# Rendering methods

Now that the means of modeling the scene have been introduced, methods of rendering the scene are needed, or in other words, methods of simulating the light transport in the scene. They aim to solve (at least approximately) the rendering equation, which is introduced at the beginning of this chapter. Then follows an overview of algorithms for scenes without participating media, as they will form the basis for later. Path tracing and photon mapping are probably the most widespread, therefore they are explained in more detail. The next section covers algorithms for rendering participating media, starting with an overview and then providing a more involved description of volumetric path tracing and diffusion-based methods, as they will be used in this work.

The light transport equation (or rendering equation) describes the distribution of radiance in the scene. When disregarding subsurface scattering and taking into account only scattering on surfaces, it has the following form:

$$L(x,\omega) = \underbrace{L_e(x,\omega)}_{\text{emitted radiance}} + \underbrace{\int_S L(x,-\omega_i)\, f_s(x,\omega_i,\omega)\, |\cos\theta_i|\, d\omega_i}_{\text{surface scattered radiance}} . \tag{5.1}$$

It gives the radiance $L(x,\omega)$ leaving a surface point $x$ in direction $\omega = (\theta,\phi)$. For most of the scenes, this equation is practically impossible to be solved analytically. One possible approach is to impose restrictions. The radiosity method introduced by Goral et al. [GTGB84] does this by discretizing the scene geometry into finite elements, which are ideally diffuse. Then a system of equations can be formed, which describes the radiosities of these elements. Although, in theory, it can be solved analytically, numerical methods are usually used. The same applies to the rendering equation – usually, the numerical methods are used, such as photon maps or path tracing.

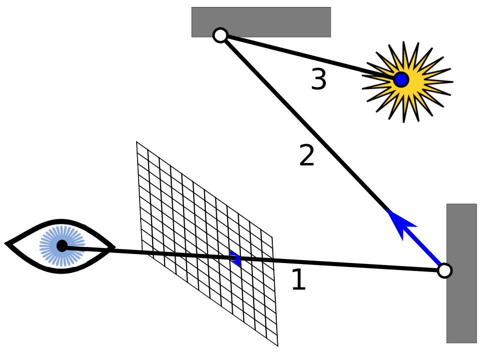


**Figure 5.1:** Path construction. The first segment is obtained by casting a ray from the camera through a pixel in the image plane. At the intersection point, the BRDF is sampled to obtain a new direction, and the process repeats. The final path vertex is found by sampling a light source instead.

## 5.1 Path tracing

Path tracing, introduced by Kajiya [Kaj86], is a Monte Carlo rendering algorithm. It works by constructing random paths from the camera to a light source and evaluating their radiance contribution. The path can be constructed incrementally by casting a ray from the camera through a pixel in the image plane and finding an intersection with the scene, where the BRDF is sampled to obtain the direction of the next path segment, see Figure 5.1. The final vertex of the path is chosen by sampling the surface of a light source. The radiance estimate for a path is given by [PJH16]:

$$\frac{L_e(x_i \to x_{i-1})\, f_s(x_i \to x_{i-1} \to x_{i-2})\, G(x_i \leftrightarrow x_{i-1})}{p_A(x_i)} \cdot \prod_{j=1}^{i-2} \frac{f_s(x_{j+1} \to x_j \to x_{j-1})\, |\cos\theta_j|}{p_\omega(x_{j+1} - x_j)}, \tag{5.2}$$

where $p_A$ is the pdf used for sampling the light source surface, and $p_\omega$ is used for sampling the BRDF. The geometric term $G(x \leftrightarrow x')$ captures the relative attitude between the two points and also accounts for visibility (the binary $V(x \leftrightarrow x')$ term):

$$G(x \leftrightarrow x') = V(x \leftrightarrow x') \frac{|\cos\theta||\cos\theta'|}{\|x - x'\|^2}. \tag{5.3}$$

The contribution of a path typically becomes lower with increasing length (number of vertices), but simply disregarding all paths above certain length would introduce bias – the algorithm would consistently underestimate the radiance. A technique called Russian roulette can be used, which involves

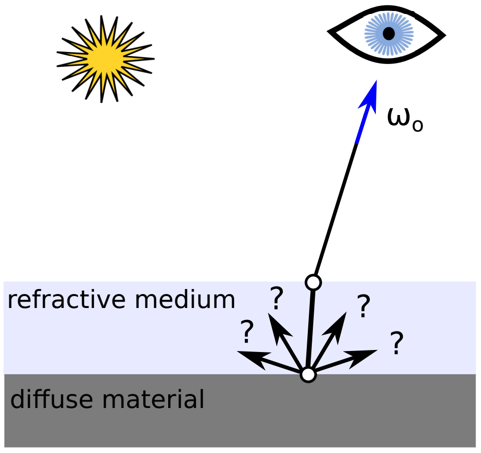


**Figure 5.2:** A diffuse surface behind a refractive medium – importance sampling can not be used here.

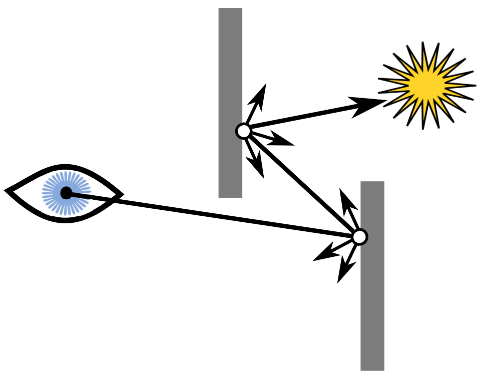


**Figure 5.3:** Indirect lighting on diffuse surfaces. On scenes like this path tracing converges much slower.

terminating the path in each vertex with probability $q$. In that case, the radiance contribution of the path needs to be weighted by $\left(\frac{1}{1-q}\right)^{n-1}$, where $n$ is the number of vertices of the path.

In the case of indirect lighting on diffuse surfaces, as shown in Figure 5.3, it is difficult to find a path from the camera to the light source. A solution is to start building the path both from the camera and light source and then connect the two segments with a shadow ray, which increases the convergence rate significantly in these cases. This method is called bidirectional path tracing [LW93]. Another difficult case for path tracing is a diffuse material behind a refractive medium, as in Figure 5.2. The problem is the refraction between the diffuse surface and the light source, therefore the light source can not be importance sampled. Photon maps can handle this easily. In scenes with fine geometrical details, such as small holes, or with sharp caustics, the most important paths may not be selected frequently enough. Metropolis light transport [VG97] addresses this by mutating the path and selecting the mutations with significant contributions in the final image.

## 5.2 Photon mapping

Photon mapping [JC95, Jen01] is a two-phase rendering method. In the first phase, photons are shot from light sources into the scene. They follow the path of specular reflections and refractions. When the photon hits a mostly diffuse surface, it is stored into a photon map. In the second phase, the photon map is used to estimate the indirect incident illumination at a point by applying a filter over the nearest photons. This makes it a biased algorithm. The advantage of this approach is that it can sample paths that path tracing (including bidirectional PT) can not – for example a photograph behind glass, because neither camera nor the light source can be sampled from the diffuse surface since they are behind a refractive medium.

One issue of this basic version of the algorithm is that to obtain a higher quality image, the number of photons must be increased, which in turn linearly increases the amount of memory needed. Progressive photon mapping [HOJ08] swaps the phases – it first traces paths from the camera, thus finding visible points, which are stored. In the second phase, the photons are shot from the light sources, distributing their energy into the visible points. The stochastic progressive photon mapping [HJ09] lowers the memory requirements even more by performing the two phases repeatedly, but for a very small number of visible points and photons. For a more in-depth description of the algorithm, see [PJH16].

## 5.3 Rendering participating media

The algorithms for rendering participating media are mostly based on those described above. This section starts with categorization and an overview of the methods. Then follows a more detailed description of methods used in this work – volumetric path tracing, as it is probably the simplest exact method, which can be used to obtain a reference image, and the diffusion-based methods. A categorization of volume light transport algorithms can be found in the survey by Cerezo et al. [CPP$^+$05], while a more recent survey by Novak et al. covers Monte Carlo methods [NGHJ18].

**Deterministic methods.** The Zonal method, introduced by Rushmeier [RT87], subdivides the medium into finite elements (voxels), computes their form factors, and creates a system of equations, mimicking the radiosity. Bhate [Bha93] adaptively refines the voxels to locally increase accuracy according to a user-specified error bound, similarly to the hierarchical radiosity. Kajiya and Von Herzen [KVH84] use a system of partial differential equations, express the radiance in truncated spherical harmonics and solve the system by relaxation. Another group of methods is based on the discretization of solid angles around points in a cubic lattice. The solution of the transfer equation is obtained by iterative energy shooting. Patmore [Pat93] shoots the energy along a single ray per each bin, which causes visible ray effects. Max

[Max95] therefore shoots the energy through the entire bin. Another category of methods is based on diffusion approximation. They will be described in more detail below. Premože et al. [PAS03] make the opposite assumption – that the medium is sparse and strongly forward scattering. Their algorithm is based on the path formulation of the RTE. They find the most probable path through the medium and account only for contributions from its surrounding paths.

**Stochastic methods.** Stochastic methods include volumetric path tracing as discussed below, or bidirectional path tracing [LW93], in which case the rays are traced both from light sources and from the camera. Final paths are composed of a light source subpath and a camera subpath connected by a shadow ray. Metropolis light transport, which works by mutating paths between the camera and the light source, has been extended to participating media by Pauly et al. [PKK00]. Another group of algorithms, based on bidirectional path tracing – many-light methods – use light tracing to spawn virtual light sources in the medium and then connect camera paths to them. Keller [Kel97] uses point lights, which cause fireflies in the image due to the singularity in the fallof term $1/d^2$. A way to mitigate this has been proposed by Raab et al. [RSK08]. Novák et al. replace the point lights with ray lights and distribute the energy along the ray [NNDJ12b], which helps to reduce the effect of the singularity. Using spheres [HKWB09] or beams [NNDJ12a] instead of the points or rays also helps, but can lead to overblurring. Density estimation methods were introduced to volumetric light transport by Jensen and Christensen [JC98], who extended photon maps by storing photons in the medium volume at the places of scattering events. The photon map is used only for indirect illumination. Křivánek et al. [KGH$^+$14] analyzed different types of volumetric density estimators (points, beams, planes, volumes) and combined those with complementary benefits in a single algorithm. Adabala and Manohar [AM00] also trace light carrying particles, but instead of storing their energy into a separate data structure, they transfer the energy onto particles of the particle system used to simulate the medium. Only the energy that would scatter into the camera direction is stored, which makes the method view-dependent.

### 5.3.1 Volumetric path tracing

Path tracing, as described above, assumes that the radiance carried by a ray changes only when the ray hits a surface. But when the ray travels through a participating medium, the radiance can also change at any point on the ray. By extending path tracing to account for this, as done by Rushmeier [Rus88], volumetric path tracing is obtained. Therefore when the algorithm reaches

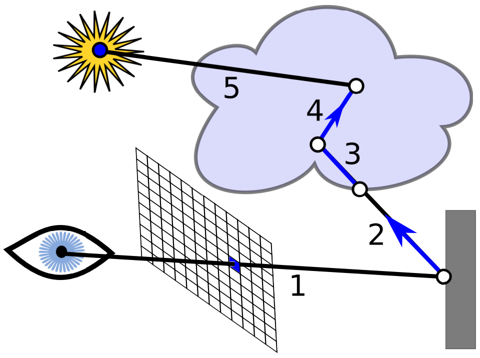


**Figure 5.4:** Path construction in a scene with a medium with a non-refractive boundary. The path segments 1, 2, and 5 are constructed the same way as in the standard path tracing. Segment 3 is constructed by distance sampling. Then a new direction must be chosen by sampling the phase function, and the point of the next scattering event is found again by distance sampling. Note that the transmittance of segments 3, 4, and part of segment 5 needs to be included in the path throughput, as well as the values of the phase function for the two scattering events.

a medium during the path construction, it needs to additionally consider the medium properties, for as long as the ray is inside the medium. This is demonstrated in Figure 5.4. The construction thus takes into account the ability of the medium to transmit light (transmittance), and the directional characteristics of the scattering (phase function). It does so by using the transmittance and phase function sampling routines, which were discussed in Chapter 4.

### 5.3.2 Diffusion approximation

As observed by Stam [Sta95], a ray of light in highly scattering media undergoes many scattering events, therefore the propagation of light in such media can be modeled as a diffusion process of fluence $\phi(x) = \int_S L(x, \omega)\, d\omega$. The diffusion process in a homogeneous infinite medium is modeled by a diffusion equation [Ish78]:

$$D\nabla^2\phi(x) = \sigma_a\phi(x) - Q_0(x) + 3D\nabla \cdot Q_1(x), \tag{5.4}$$

where $D$ is the diffusion constant. The fluence is influenced by the zero- and first-order source terms $Q_0$ and $Q_1$.

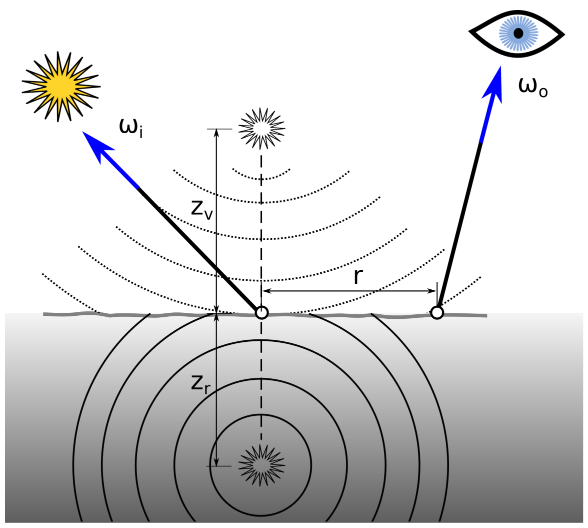


**Figure 5.5:** Dipole uses two point light sources – the positive lies inside the medium and represents the first scattering event, the second light source is negative and is outside the medium positioned in a way to zero out the fluence at the surface.

## Dipole

Jensen et al. [JMLH01] build on this idea – they solve subsurface scattering by combining accurate single scattering with a diffusion dipole:

$$S(x_i, \omega_i, x_o, \omega_o) = F_{t,i} \cdot (S_d(x_i, x_o) + s^{(1)}(x_i, \omega_i, x_o, \omega_o)) \cdot F_{t,o}. \tag{5.5}$$

Farrell et al. [FPW92] proposed to model the incident source distribution as a point source. The point source corresponds to scattering below the surface, therefore it is placed one mean free path $z_r = 1/\sigma_t'$ below the surface. Since the diffusion equation assumes an infinite medium, a second (negative) point source is added $z_v = z_r + 4AD$ above the surface, thus creating a plane boundary, where each point satisfies the boundary condition: the inward diffuse flux at the surface point is zero. The configuration is shown in Figure 5.5. The diffuse BSSRDF term is then $S_d = \frac{Rd}{\pi}$, where $R_d$ is diffuse reflectance due to dipole source, which was obtained from the analytic solution to the diffusion equation for a point source:

$$\begin{aligned} R_d(r) &= -D \frac{n \cdot \vec{\nabla}\phi(x_s)}{d\Phi_i} \\ &= \frac{\alpha'}{4\pi}\left[(\sigma_{tr} d_r + 1)\frac{e^{-\sigma_{tr} d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr} d_v + 1)\frac{e^{-\sigma_{tr} d_v}}{\sigma_t' d_v^3}\right]. \end{aligned} \tag{5.6}$$

In the above equation, the diffusion constant is $D = \frac{1}{3\sigma_t'}$, $\sigma_{tr} = \sqrt{3\sigma_a \sigma_t'}$ is effective transport coefficient, $\alpha'$ is the reduced scattering albedo, and $d_r$ and

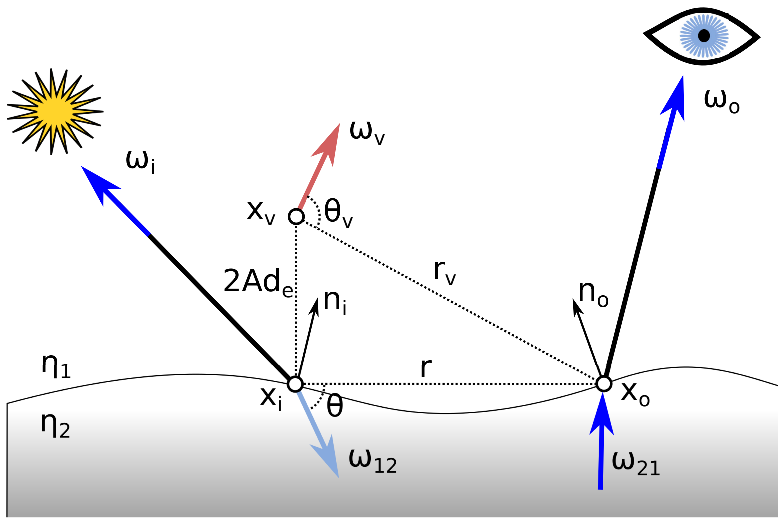


**Figure 5.6:** Directional dipole, like the isotropic dipole, uses two light sources, but they are rays, not points. Additionally, the real light source ($\omega_i$) is positioned at the surface, not below it. Its direction is obtained by refracting $\omega_i$. The virtual light source $\omega_v$ is obtained by mirroring the real light source around the modified tangent plane, which is the plane that contains $r = x_o - x_i$ and is perpendicular to the plane defined by $r$ and $n_i$. Its origin is displaced along the modified tangent plane normal.

$d_v$ are the distances from $x$ to the real and virtual light source, respectively. $\sigma'_t$ is the reduced attenuation coefficient

$$\sigma'_t = \sigma'_s + \sigma_a \quad \text{where} \quad \sigma'_s = \sigma_s(1-g). \tag{5.7}$$

The $g$ is mean cosine of the scattering angle:

$$g = \int_S (\omega \cdot \omega')\, p(\omega, \omega')\, d\omega'. \tag{5.8}$$

It is positive for forward scattering phase functions, negative for mostly backward scattering, and zero for the isotropic phase function.

## Directional dipole

The traditional dipole uses point light sources, which works well for materials with isotropic scattering. In the case of mostly forward scattering materials where the light maintains its direction for longer distances, the resulting image will miss some effects. An improved analytical model has been introduced by Frisvad et al. [FHK14], which uses ray light sources instead, thus accounting for the directionality of light – see Figure 5.6. The BSSRDf is

$$S(x_i, \omega_i, x_o, \omega_o) = F_t(\eta, \omega_i) \cdot (S_d(x_i, \omega_i, x_o) + S_{\delta E}(x_i, \omega_i, x_o, \omega_o)) \cdot F_t(\eta, \omega_o). \tag{5.9}$$

The part of the single scattering that does not change direction is included in the reduced intensity (or direct transmission) term $S_{\delta E}$, and the rest is included in the diffuse term $S_d$. The reduced intensity term is defined implicitly by:

$$L_{r,\delta E}(x_o, \omega_o) = F_t(\eta, \omega_i)\, F_t(\eta, \omega_o)\, e^{-\tilde{\sigma}_t |x_o - x_i|}\, L_i(x_i, \omega_i), \tag{5.10}$$

where the modified coefficients are:

$$\tilde{\sigma}_t = \tilde{\sigma}_s + \sigma_a, \quad \tilde{\sigma}_s = \sigma_s(1 - g^2), \quad \eta = \frac{\eta_2}{\eta_1}. \tag{5.11}$$

The diffuse term is:

$$S_d(x_i, \omega_i, x_o) = S'_d(x_o - x_i, \omega_{12}, d_r) - S'_d(x_o - x_v, \omega_v, d_v). \tag{5.12}$$

Note that it does not depend on the outgoing direction – it is an assumption made based on the large number of scattering events. The $\omega_{12}$ and $\omega_v$ are directions of the real and virtual light sources, respectively, while $d_r$ and $d_v$ are their distances from the exiting point $x_o$. The $S'_d$ term is defined as:

$$\begin{aligned} S'_d(x, \omega_{12}, r) = \frac{1}{4C_\Phi(1/\eta)} \frac{1}{4\pi^2} \frac{e^{-\sigma_{tr} r}}{r^3} \Bigg[ & C_\Phi(\eta) \left( \frac{r^2}{D} + 3(1 + \sigma_{tr} r) x \cdot \omega_{12} \right) \\ & - C_E(\eta) \Bigg( 3D(1 + \sigma_{tr} r) \omega_{12} \cdot n_o - \Bigg( (1 + \sigma_{tr} r) \\ & + 3D \frac{3(1 + \sigma_{tr} r) + (\sigma_{tr} r)^2}{r^2} x \cdot \omega_{12} \Bigg) x \cdot n_o \Bigg) \Bigg], \end{aligned} \tag{5.13}$$

where [1]

$$C_\Phi(\eta) = \frac{1}{4\pi} \int_H F(\eta, \theta_o) \cos\theta_o \, d\omega_o, \tag{5.14}$$

$$C_E(\eta) = \frac{3}{4\pi} \int_H F(\eta, \theta_o) \cos^2\theta_o \, d\omega_o, \tag{5.15}$$

$$\cos\theta_o = n_o \cdot \omega_o. \tag{5.16}$$

The diffusion constant $D$ and the effective transport coefficient $\sigma_{tr}$ were defined previously for the dipole. The direction of the real light source, $\omega_{12}$, is obtained by refracting the incident direction, while that of the virtual source is given by:

$$\omega_v = \omega_{12} - 2(\omega_{12} \cdot n_i^*)\, n_i^*. \tag{5.17}$$

The normal of modified tangent plane is:

$$n_i^* = \begin{cases} n_i, & \text{for } x_o = x_i \\ \frac{x_o - x_i}{|x_o - x_i|} \times \frac{n_i \times (x_o - x_i)}{|n_i \times (x_o - x_i)|}, & \text{otherwise} \end{cases} \tag{5.18}$$

[1] Polynomial approximations can be found in the referenced paper.

The squared distance to the real light source is:

$$d_r^2 = \begin{cases} r^2 + D\mu_0(D\mu_0 - 2d_e \cos\beta), & \text{for } \mu_0 > 0 \text{ (frontlit)} \\ r^2 + 1/(3\sigma_t)^2, & \text{otherwise (backlit)}, \end{cases} \tag{5.19}$$

where the cosine terms are $\mu_0 = -n_o \cdot w_{12}$, and:

$$\cos\beta = -\sqrt{\frac{r^2 - (x \cdot \omega_{12})^2}{r^2 + d_e^2}}, \tag{5.20}$$

and $d_e = 2.131D/\sqrt{(\alpha')}$. The reduced scattering albedo is $\alpha' = \sigma_s'/\sigma_t'$. The distance to the virtual light source $d_v = |x_o - x_v|$ is determined by its position:

$$x_v = x_i + 2A\, d_e\, n_i^*, \tag{5.21}$$

where the reflection parameter $A$ is:

$$A(\eta) = \frac{1 - C_E(\eta)}{2C_\phi(\eta)}. \tag{5.22}$$

Note that the model can (rarely) produce negative values. The authors solve this by clamping them to zero, which, according to them, does not cause visible artifacts. Additionally, the model is not reciprocal – if this is required, the authors propose to use an average of two evaluations of the model with swapped variables. This results in some differences in rendered images of thin media.

## Hierarchical integration

Computing the exiting radiance due to subsurface scattering involves a BSSRDF and therefore computing an area integral over the surface with the irradiance. In practice, this means that the irradiance would have to be computed repeatedly, so ideally, the samples on the surface would be cached. Furthermore, computing the exiting radiance in a single point would mean that BSSRDF would have to be evaluated for each of the samples.

The approach proposed by Jensen and Buhler [JB02] is to distribute points on the surface (for example by using the method in [BWWM10]) and evaluate irradiance at each of them. They then spatially cluster the points using an octree, where the original points are located in the leaves and the inner nodes contain averaged irradiance and position of the sample, both weighted by the areas belonging to the individual samples.

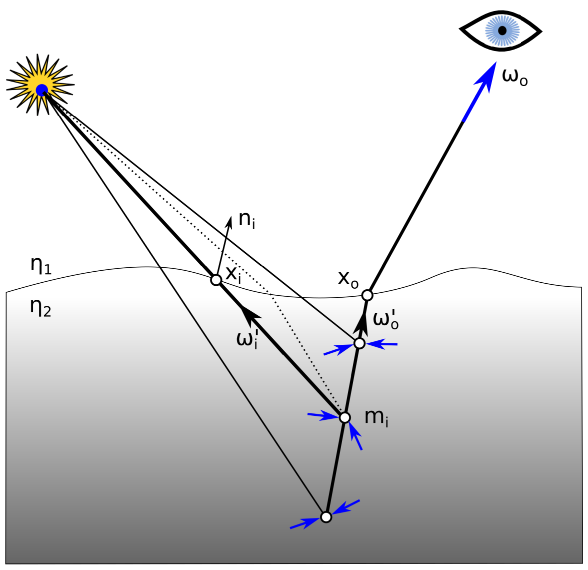


**Figure 5.7:** Single scattering with omitted refraction towards the light source. This enables light source sampling. The correct path is shown by the dotted line.

In case of the standard dipole, the diffuse radiant exitance at point $x$ due to irradiance sample $p$ is calculated as:

$$M_{o,p}(x) = F_{dt}(x)\, E_p\, A_p\, S_d(x,p), \tag{5.23}$$

with $F_{dt}(x) = 1 - F_{dr}(x)$, where $F_{dr}$ is the diffuse Fresnel reflectance, $E_p$ is the irradiance at point $p$, and $A_p$ is the area corresponding to that point. The exiting radiance is:

$$L_o(x,\omega_o) = \frac{F_t(x,\omega_o)}{F_{dr}(x)} \frac{M_o(x)}{\pi}, \tag{5.24}$$

where $M_o$ is the sum of diffuse radiant exitance contributions of all points.

In the case of the directional dipole, the diffuse BSSRDF depends on incident direction, therefore it has to be evaluated for each incident direction separately. The clustering of the irradiance samples works the same way, except that an irradiance sample is replaced by a vector of differential irradiances for a constant set of incident directions.

### 5.3.3 Single scattering

These methods are used to compute the radiance exiting the material after a single scattering event inside the medium. They are useful in combination with diffusion methods, which in turn compute only the multiple scattering part of the subsurface radiance. Alternatively, they can be used as the only BSSRDF for low-scattering materials, where the contribution of dipole would be minimal. The single-scattering BSSRDF term involves integrating along the refracted outgoing ray and all in-scattering directions:

$$S^{(1)}(x_i, \omega_i', x_o, \omega_o') = \int_0^\infty \tau(x_o, m)\, \sigma_s(m) \int_S p(\omega_i', \omega_o')\, \tau(m, x_i)\, d\omega_i' ds. \quad (5.25)$$

In the above equation, primed $\omega$ denote refracted directions, and $m = x_o + s\omega_o'$ is the scattering point in the medium. Computing the radiance this way is not very practical, though, as $\omega_i'$ can not be importance-sampled with respect to a light source outside the medium. Therefore Ma et al. [MXW+16] approximate $\omega_i' = \omega_i$, as shown in Figure 5.7. In that case, $\omega_i'$ can be multiple-importance-sampled with respect to both the phase function and light sources. A simpler, analytical method for flat, uniformly lit, homogeneous media can be found in [HK93]. More analytical (and heavily simplified due to various assumptions) and deterministic methods can be found in [CPP+05]. Most of them were derived for atmospherical effects, therefore the typical assumptions were a single directional light source, homogeneous medium, or a medium with homogeneous layers.

# Chapter 6

# Design

Now that all the models and algorithms have been described, this chapter discusses which algorithms will be used, why they were chosen, and how they cooperate together to achieve the goal outlined in the introduction. Table 6.1 shows an overview of the models involved, their routines, and which rendering algorithms use them. The chapter starts with the models, then follow the rendering algorithms.

## 6.1 Scene model

To be able to synthesize a picture, a model of the scene is needed. An essential part of the scene is the light source – in this case the Sun and the sky, models of which are provided by Hošek-Wilkie [HW12, HW13]. The other part of the scene is the water body. There are three aspects that need to be modeled to mimic the real appearance: the geometric shape of the water surface, the reflective properties of the surface, and the optical properties of the water.

### Surface shape

The shape of the water surface can be obtained in several ways. Since we are aiming for mostly calm waters, we can avoid methods that simulate water

| | volPT | dip | SS | dirDiff | dirTr |
|---|---|---|---|---|---|
| light source | | | | | |
| - sampling | ● | ● | ● | ● | |
| - evaluation | ● | ● | ● | ● | ● |
| surface geometry | | | | | |
| - surface sampling | | ● | | ● | |
| surface scattering (BSDF) | | | | | |
| - sampling | ● | | ● | | ● |
| - evaluation | ● | | ● | | |
| participating medium | | | | | |
| - distance sampling | ● | | ● | | |
| - transmittance evaluation | ● | | ● | | ● |
| - phase f. sampling | ● | | ● | | |
| - phase f. evaluation | ● | | ● | | |

**Table 6.1:** Components of the scene model and their routines, and which method uses them. Note that the sampling routine often internally uses the evaluation routine, but not always. Therefore, this figure shows only direct uses.

dynamics and the effects associated with it, such as white-caps, breaking waves, sprays, and foam. Instead, simpler methods can be used, as presented in Section 2. We chose to use the Fourier synthesis method because it can produce a signal with a specific spectrum. Therefore in combination with a model of the wave spectrum, the surface should look quite realistic. The surface can be represented as a heightmap, which is a greyscale image that stores the relative vertical displacement of the surface and can be loaded into most rendering frameworks. This rough shape is, however, not all that determines the interaction of light with the surface.

## Microsurface

The waves that are too short to be visible by eye (and therefore not captured by the surface shape) but still affect the reflection and transmission of light are represented by a BSDF. Microfacet BSDF models fit this task naturally (as microfacets replace the waves) and the roughness and orientation of the microfacets can be easily adjusted.

There is a strange and unintuitive thing in the article of Ma et al. [MXW+16]. They multiply the radiant intensity leaving the medium by the normalized visibility probability distribution of the viewer and the microfacet

$$q^e_{vn}(\zeta, v) = \frac{p(\zeta)\max(v \cdot f, 0)}{(1 + \Lambda(a_v))\, f_z \cos\theta_v}, \tag{6.1}$$

which is already included in the BTDF. We have tested it on a scene with constant environment emitter with radiance set to one, and a cube with the BSDF applied. The transmitted radiance through the cube hitting the sensor becomes greater than one. Unfortunately, they do not provide a derivation of the equations. As a result, we chose to omit this term.

Importance sampling is described in section 3.4.1; it is based on sampling the distribution function of microfacet normals. Ross et al. [RDP05] use additional terms to correct for kurtosis and skewness, but we leave them out during the sampling to simplify the derivations of sampling equations. We still use them in the evaluation of the BSDF, though.

### Light in the water

Aside from the reflected light, a part of it also refracts below the surface. The water may contain various substances that absorb or scatter light, therefore water can be modeled accurately by a participating medium, as described in Section 4.4.1. Rendering the participating medium, however, can be very costly, depending on its density and scattering characteristics.

## 6.2 Rendering the scene

This process can be thought of as either propagation of light through a participating medium, or subsurface scattering. The former fully models the medium and then simulates the propagation of light through it in detail, accounting for any variations in density, constitutions, or even objects inside the medium. Subsurface scattering, on the other hand, usually assumes a homogeneous, locally flat object. Therefore it is a good choice especially for dense materials, in which the light does not travel very far. Since it often builds on various simplifications, it is much cheaper to compute.

We will be using the subsurface-scattering approach, but to verify that the results are correct, we will need to generate reference images. Therefore we will also use the other approach – participating medium in combination with an unbiased rendering algorithm. We chose to use volumetric path tracing because it is already implemented in many rendering frameworks, and it is easy to understand and tune.

## Rendering participating media

The problem with path tracing, or rather the Monte Carlo integration used in path tracing, is the high variance, which manifests itself in the image as high-frequency noise. Therefore an essential part of path tracing is a variance reduction technique called importance sampling, which means the samples in MC integration are not chosen (pseudo)randomly, but rather based on a probability density function proportionate to the integrand. Rendering scenes with participating media involves several integrations (techniques of importance sampling the integrands follow in the parentheses):

- reflected radiance from the surface (BSDF sampling – 3.4.1),
- in/outscatering along a ray (distance sampling – 4.3.2),
- transmittance in the participating medium (transmittance evaluation – MC),
- inscattered radiance into the ray direction in the medium (phase function sampling – see below).

As for the transmittance computation, the medium density changes quite slowly. Therefore we will use simple MC integration, with sample density per unit distance specified by the user.

The phase function is a linear combination of two phase functions – clear water (Rayleigh phase f.) and particulate matter (tabulated). Sampling a linear combination can be done by first choosing a component with probability respecting the weights. The Rayleigh phase function is quite uniform, so rejection sampling [Fri11] should be efficient enough. The particulate matter phase function, on the other hand, is strongly forward peaked, so importance sampling is vital here. Since it is tabulated, it can be sampled easily by computing a CDF of a step function and inverting it (also covered by the previous article).

## Subsurface scattering

We will be comparing the directional dipole (Section 5.3.2), which takes the direction of incident light into account, with the isotropic dipole (Section

5.3.2). The two models are almost interchangeable, except that the isotropic dipole does not include single scattering. Therefore the light traveling via paths that contain only a single scattering event in the medium must be added. Both setups include direct transmission and reflections on the inner side of the boundary. Reflections on the outer side, however, must be handled by the BRDF.

As can be seen from the definition of BSSRDF, to obtain the radiance leaving the surface, one must integrate the BSSRDF multiplied by the irradiance over the surface of the object. Therefore irradiance should be cached. Hierarchical integration, as described in section 5.3.2, goes one step further and also helps to reduce the number of BSSRDF evaluations. It can be used both for dipole and, with minor modifications for directional dipole as well (explained in the next chapter).

We chose to use accurate single scattering, which is very similar to volumetric path tracing, except that the exiting interface is disregarded so that a path segment from within the medium to the light source can be found by sampling the light source. The distance sampling and phase function sampling used in volumetric path tracing can be used here as well. Compared to the method in [MXW$^+$16], they always sample the light source (in their case, the solar disc), which helps to reduce variance. Since we want to account for illumination from the sky as well, sampling both the light source and the phase function (using MIS) should help to reduce the noise.

# Chapter 7

# Implementation

As the previous chapter suggests, we have decided to use an existing rendering framework to be able to validate our implementation of rendering algorithms more easily against an existing one, and to avoid having to reimplement core structures and basic operations. The main requirements influencing the choice are:

- physically based
- support for path tracing of participating media (to render reference images),
- easily extensible,
- good documentation,
- HDR output,
- opensource, free to use, modify and distribute, without a charge.

According to a comparison made by Glanz [Gla18], the most often used physically based renderers in research are Mitsuba, PBRT-v3, and LuxRender. He also compared the speed and the amount of noise on the classroom scene with settings producing similar results. Mitsuba was approximately five times faster, while also producing less noise in final images.

**Physically based rendering toolkit (PBRT).** PBRT is a rendering system accompanied by the book Physically Based Rendering: From Theory to Implementation [PJH16], which explains the theory behind it and also serves as a very thorough documentation. The first version, together with the first edition of the book, was released in 2004. Currently, the third version from 2016 is available, which supports volume rendering, and aside from standard path tracing includes also bidirectional methods, such as bidirectional path tracing, Metropolis light transport, or stochastic progressive photon mapping. The code is written in C++, and it does not change much so as to maintain consistency with the book.

**Mitsuba.** Mitsuba [Jak10] is based on PBRT and is also written in C++. It is easily extensible – most of the components, such as integrators, materials, and light sources, are implemented as dynamically linked plugins. Many biased and non-biased rendering algorithms are implemented, also supporting the rendering of participating media. A version 2 has been published in 2019 [NDVZJ19], which focuses on retargability – it can be compiled in different vectorization modes (scalar, SIMD, CUDA) or in differentiable rendering mode, which makes it possible to reconstruct a scene based on reference images. It is licensed under a copyleft license (GNU GPL v3).

**LuxCoreRender.** LuxCoreRender started in 2008 as LuxRender and was initially based on PBRT. In 2017 it was completely rewritten. It focuses on the ease of use while maintaining physical correctness. It implements several unbiased rendering algorithms, including bidirectional path tracing and Metropolis sampling. Support for volume rendering is also present. It is licensed under a permissive license (Apache Public License v2.0).

**Others.** YafaRay is another physically based renderer, with the latest release in 2017. It implements path tracing, photon mapping, bidirectional path tracing, and stochastic progressive photon mapping. Volumetric rendering is also supported. It is licensed under a permissive license (LGPL 2.1). Tungsten Renderer from 2014 uses Intel's Embree ray tracing library. Aside from path tracing, bidirectional path tracing and other methods are implemented, as well as support for volumetric rendering. As of now, it lacks documentation. Lightmetrica is a research-oriented renderer, featuring various statistic and performance tests, a portable plugin system and verified reference implementations of rendering algorithms. Appleseed focuses on production. The rendering methods include path tracing and stochastic progressive photon mapping. Volume rendering is supported, including various dipole-based subsurface scattering methods. A part of the rich set of features is also the

C++ plugin system. It is licensed under a permissive (MIT) license. Pixie is a RenderMan compatible renderer with the latest release in 2009.

We chose to use Mitsuba, because it already contains many rendering algorithms, including the volumetric path tracing, which can be used to verify the implementation. It is easily extensible, and it has good documentation. We are using the older version 0.6, because the code base is more stable and because the newer version, being released only several months ago, might contain more bugs. A model of the Sun and sky from Hošek and Wilkie is already integrated as well. Also present is an implementation of the dipole subsurface model with hierarchical integration, that is used to approximately compute the multiple subsurface-scattering. The fast single-scattering algorithm used by Jensen et al. [JMLH01] is also present.

There are many missing components required to create the scene, though. As for the algorithms, the directional dipole remains to be implemented, along with the sampling routines, which are usually tightly bound with the models. Several modifications will have to be made as well. Both the dipole and the single scattering algorithms take scattering and absorption coefficients as constant parameters, which is not suitable for use with the bio-optical model. Furthermore, only one subsurface scattering model per shape is allowed. Alternatively, the medium parameters can be supplied by a medium plugin, which is used only by accurate volume rendering algorithms.

Therefore we had to first remove the restriction of a single subsurface plugin per shape – the radiance computed by each of them is summed together. It would also be convenient to be able to supply the medium properties the same way for a subsurface plugin and for volume rendering. Therefore we modified the subsurface plugins to use the medium interface, which slightly changes the scene file format, namely that a shape can now have both a medium and a subsurface plugin attached. In fact, it is a requirement now for the single scattering and dipole plugins to work. Also, as hinted above, multiple subsurface plugins per shape are now allowed.

The rest of the chapter describes what had to be implemented, how it is done, and why. First comes the 3D shape, which defines the boundary of the water volume. The next section covers the BSDF. Then follows the water medium with the bio-optical model. We also chose to reimplement the single scattering plugin, so the next section explains why and how it differs from the original version. The last section is about the direction dipole. Each section, therefore, corresponds to a newly created plugin.

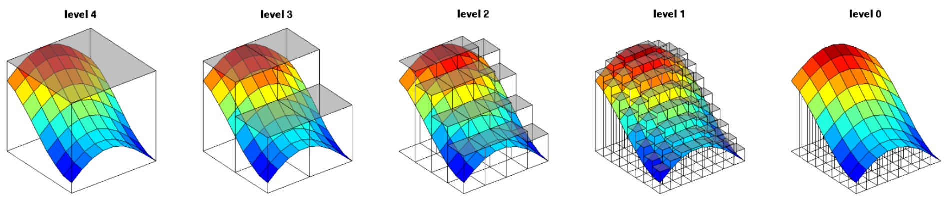


**Figure 7.1:** Heightmap mipmap used for finding a ray-heightfield intersection. It is built bottom-up. The height value stored in the cell is obtained as the maximum of the four underlying cells. (from [TIS08])

## 7.1 Shape

In Mitsuba, the boundary of scene objects is defined by a shape, or rather a plugin implementing that interface. There are several predefined shapes, but none of them is useful for representing a water volume. The cube shape has flat sides, which could be solved by displacement mapping. Unfortunately, that is not implemented in Mitsuba. There is also a heightfield shape – a surface with displacement given by a heightmap. This would be perfect for surface reflections, but there is no volume for simulation of the subsurface scattering. Therefore we created a modified version of the heightfield shape called volumetric heightfield, which is essentially a cube with the surface of one of its sides displaced by the heightfield.

Another requirement is to be able to have the water surface extend all the way to the horizon in the rendered image. Since the heightmap has a finite resolution, it will have to be repeated. The heightfield shape, which we based the shape on, does not support this. It is not as trivial as tiling a simple texture, because Mitsuba uses a special data structure to accelerate the computation of ray intersection, as described in [TIS08].

Basically, it involves building a mipmap of the heightmap, where each pixel in the mipmap stores the maximum value of the pixels of the heightmap portion it covers – see Figure 7.1. When looking for the intersection, the mipmap is used to skip parts of the heightfield, as described in the Algorithm 2. When going down through the levels of the mipmap, on each level the AABBs of the corresponding portions of the heightfield are constructed using the height from the mipmap. Those AABBs which are not intersected by the ray can be skipped. Those that are intersected have to be split using the next level of the mipmap. This process repeats until the lowest level of the mipmap is reached, which represents the actual pixels of the heightmap. It is also useful to traverse the children of a mipmap node in the order of the ray because then the algorithm can be stopped at the first intersection found.

**Algorithm 2** ray_heightfield_intersect($ray$)

```
while ray_above_heightfield do
    height ⇐ mipmap_value(ray.origin, level)
    if level == 0  &&  ray.origin.y <= height then
        ray_intersect_bilinear_patch(ray)
    else
        ray_intersect_bounding_planes(ray, level)
    end if
    if ray.origin.y > height  ||  level == 0 then
        ray.origin = intersection
        update_mipmap_level()
    else
        descend_one_mipmap_level()
    end if
end while
```

The simplest way to add tiling to this algorithm would be to create several instances of the shape. Mitsuba does support instancing, where only one instance is actually stored in the memory, but exactly for this reason, the instances can not have subsurface plugin attached to them. Also, the boundaries between the tiles might cause artifacts when used with diffusion methods. Another possibility is to simply repeat the input heightmap, which would also increase the size of the mipmap. This would be quite simple to implement, but usable only for a limited number of tiles. We chose to modify the shape so that it bounds all the tiles. When intersected, it internally traverses the tiles. For each tile, the point where the ray enters it is transformed into tile-local coordinates, so the algorithm described above can be used without any changes. If an intersection with the heightfield is found in the current tile, it is transformed back to object- (and then world-) space and returned. Thus the mipmap needs to be stored only once.

To generate the heightmap, We used Fourier synthesis (as described in section 2.2), because it can be combined with the spectral and height models for deep open seas. We chose to use Python for this task because all the necessary libraries are easily accessible from the repository, and there is no need to set up build configuration. Additionally, performance is not important here, so using a higher-level language has only positives in this case. The output heightmap is in *exr* format, which supports float values and lossless compression. The values are scaled to the range $[0.0, 1.0]$, where 0.5 corresponds to the sea level. This is to have the sea level at constant $y$ coordinate in the scene, so it is not necessary to shift the object when the wave amplitude changes. The full value range represents a wave height of 30 meters. This convention is used to avoid negative values in the heightmap.

## 7.2 BSDF

The sea-surface BSDF is, again, implemented as a separate plugin. Although Mitsuba has a microsurface dielectric BSDF plugin, which supports several microfacet distributions, we have decided to implement it separately, so we do not have to implement and integrate everything at once, and to simplify the testing. Additionally, we would have to convert between parameters – roughness versus wind speed. The plugin could most likely be integrated into the existing one, even though the BSDF contains some additional terms. The existing plugin also contains a better (and more complex) sampling routine, which accounts for the visibility of the microfacet normals [Hd14]. Another problem might be that the sea-surface BSDF is anisotropic – it assumes the local coordinate frame is oriented in the upwind direction.

## 7.3 Medium

As mentioned above, Mitsuba uses the volume interface to represent a participating medium. There are two plugins that implement it: a homogeneous and heterogeneous medium. A possible way of representing water, which consists of several components (as described in section 4.4.1), might be to use a medium per each component. The question is, how could the joint medium be sampled. Additionally, the heterogeneous medium assumes a spectrally uniform attenuation coefficient. Therefore we have decided to implement the water medium as a separate plugin, which relies on another plugin called data source to provide the concentration of each component of the bio-optical model, making it possible to easily swap, without recompiling, the original gaussian vertical distribution for constant or to specify it using volumetric data.

The phase function is a linear combination of pure water phase function (approximated by the Rayleigh phase function) and the phase function of particulate matter (tabulated), weighted by the scattering coefficients. Mitsuba provides a plugin for mixing phase functions – mixture phase. Unfortunately, it assumes constant weights, but in this case, the scattering coefficient of particulate matter varies spatially with the concentration. Therefore we have decided to implement the phase function as a separate class. Not a plugin, because evaluating the phase function involves querying the scattering coefficients. Therefore a tight coupling with the water medium is necessary – the phase function can not use the generic medium interface.

## 7.4 Single scattering

Mitsuba already has two single-scattering algorithms implemented: an accurate version for participating media with refractive mesh boundaries [Hol15] and an approximate version based on an unknown paper. The shape we are using is not represented by a mesh, so the first algorithm is not an option. The approximate version constructs paths by sampling a light source, refracting a camera ray into the medium, and connecting points on the refracted ray to the light source. When rendering pure water with mostly uniform phase function illuminated by the Sun and the sky, this should work well. But when particulate matter with strongly forward phase function is added, sampling the light source becomes inefficient. Therefore we have decided to use multiple importance sampling. But because we could not find the article the algorithm is based on, we chose to reimplement the single scattering in a way more similar to path tracing, but still disregarding the refraction when exiting the medium so that a path segment from within the medium to the light source can be found only by sampling the light source. Additionally, the original plugin required the medium coefficients and phase function to be supplied directly to the plugin, so in our implementation, we pull the data from the medium of the intersected shape.

## 7.5 Directional dipole

The implementation of the directional dipole is very similar to that of the standard dipole, except for the BSSRDF formula and some differences in the hierarchical integration, so we reused most of the code that was already present. Note that the BSSRDF as described in Section 5.3.2 (and in the paper) uses Fresnel transmittance to compute radiance entering (and leaving) the material, but in the implementation, it is replaced by a generic BTDF.

The hierarchical integration involves sampling the surface of an object. For meshes, Mitsuba uses parallel Poisson disk sampling by Bowers et al. [BWWM10]. But the intersection shape we use is not a mesh, so we generate the samples ourselves. The shape is basically a box with the top side displaced by a heightmap. We select the box side based on their relative surface areas and then sample the rectangle uniformly. We neglect the displacement of the top side – for small values, the error in sample density should not be large. Finally, the samples on the top side have to be projected to the displaced surface.

The main difference in hierarchical integration from the isotropic dipole is that the BSSRDF needs to take the incident direction of radiance into account. Therefore the irradiance samples need to be broken down into a vector of differential irradiance samples for predetermined directions. The directions are identical for all samples to allow for clustering. This prevents any kind of importance sampling, so we choose these directions uniformly over a sphere.

# Chapter 8

# Results and evaluation

Now the implemented models and algorithms need to be tested. The surface geometry and scattering models, and also the bio-optical model can be compared to images obtained with different implementations. Comparison with real photos is also an option, but for the bio-optical model, it becomes tricky, because it is difficult to obtain a photo along with measured properties of the water. The directional dipole can be tested by comparing the results to that of volumetric path tracing. The chapter starts by describing the components of the test scene, summarizes the parameters of both models and algorithms, and presents the values used for the testing, as well as factors impacting the performance. Then the rendered images are presented. Finally, we analyze the images and evaluate the implementation.

## Scene setup

The scenes are composed of a light source, mostly both the Sun and the sky, although some use only the sky (see notes by the figures), a water volume represented by the intersection shape as described in the previous chapter, and a camera. But we found that there were several problems with it. Firstly, the water surface needs to extend all the way to the horizon to avoid a gap between the surface and the sky. We hoped to solve this by tiling the intersection shape, which worked well for path tracing but turned out to be a problem for the diffusion methods because they need to precompute irradiance in sampled points on the surface. Since the sample density is constant, there were either too few samples near the camera or too many samples to fit in the memory. We solved this by rotating the light source so that in front of the camera, the gap disappears, but behind the camera, it opens up. Another issue is that clear water does not scatter light too much – the mean free path of scattering is in the order of hundreds of meters (for red light even thousands), so the water constituents that only absorb light would be barely visible in deep waters. Therefore for these scenes, we use more shallow water, and we have added a diffuse white plane below the water volume.

The parameters of the bio-optical model were adjusted visually to match the photos. We use constant concentrations to reduce the parameter space and therefore simplify the scene setup. This is not realistic – for instance, the vertical distribution of algae is typically shaped as a Gaussian, with the highest concentration at a specific depth and then falls off. Also, note that the wind speed parameter used for BRDF does not always match the wind speed used for wave generation – the motivation was to obtain higher but smooth waves. It should also be pointed out that all the scenes use seawater parameters – freshwater has slightly different scattering and absorption coefficients. Also, the BSDF is based on the distribution of sea slopes, as well as the surface geometry. But the differences for freshwater would likely be negligible. The parameters which have an effect on the appearance are summarized in Table 8.1. In case they differ for a particular scene, it is noted in the figure description. Each scene contains at most one of the pollutants – CDOM, SPM, or phytoplankton; the concentration of the remaining ones is zero.

| Parameter | Value | Unit |
|---|---|---|
| Spectrum | 680, 540, 470 | nm |
| $\alpha_w$ | 0.045, 0.0558, 0.0156 | $m^{-1}$ |
| $\sigma_w$ | 0.0007, 0.0021, 0.0037 | $m^{-1}$ |
| $\alpha_c^*$ | 18.2, 9.7, 33.2 | $m^2/g$ |
| water depth | 30 | m |

**Table 8.1:** Parameter settings of the scenes influencing appearance. Windspeeds and the concentrations of pollutants are listed by the respective figures.

The render times listed by each figure were measured on a dual-core laptop CPU with hyperthreading, specifically Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz, using four render threads. When comparing the performance of standard and directional dipole, note that the standard dipole uses SSE instructions in the BSSRDF evaluation, and therefore can perform vector operations with the entire spectrum. According to the authors, the directional dipole should be as efficient as the standard dipole, but removing the need to use either accurate and slow single scattering, or an approximation. The render times are influenced especially by the number of samples per pixel. Their number is determined by the amount of randomness involved in radiance evaluation. Approximately 100 samples are needed for the noise from the BSDF sampling to disappear. Directional dipole does not involve any other stochastic algorithm, so this number of samples is sufficient. But only in the scenes without a small, high-intensity light source and with optically thick medium, as will be shown later. Depending on the type of medium, single scattering and volumetric PT usually require more samples. In case of the diffusion methods, they also depend on the density of irradiance samples on the surface of the object, but the increase should not proportionately impact rendering time thanks to the hierarchical integration, because when the radiant exitance is being evaluated, the more distant samples are replaced by clustered values, where the size of the clusters depends only on the spatial size, not on the sample density. In the case of volumetric path tracing, the path length limit can also influence the rendering time (and image quality), but only in highly scattering media. As for single scattering, the only relevant parameter here is the number of samples taken along the refracted ray. Table 8.2 lists the parameters which are mostly the same across all scenes. Again, in case the parameter settings are different for the given scene, it is listed next to the figure.

| Parameter | Value | Unit |
|---|---|---|
| vol. PT max depth | 10 | - |
| single scatter. samples | 3 | - |
| transmittance sample density | 1 | $m^{-1}$ |
| diff irradiance directions (dirDip) | 8 | - |

**Table 8.2:** Parameter settings of the scenes influencing performance. Number of samples per pixel and surface samples are listed by the respective figures.

The rendered images are in HDR format, therefore they need to be converted to LDR for display. We used pfstools to achieve that. In order to preserve the images and the intensity ratio between them as much as possible, we chose not to use tone mapping or gamma correction. Therefore these images were obtained by merely multiplying the intensity values by 3.5. The value was chosen empirically as the best fit for all of the images.

## Results

Figure 8.1 shows clear water illuminated only by the sky. The first notable difference between the rendered images and the photo is the shape of waves, which are rounder and have longer crests. This is determined by the surface geometry model, which is not suitable for this particular scene. Also, the sea surface rendered by the combination of the dipole and single scattering is brighter than those produced by the other methods. Figure 8.2 shows water with a high concentration of particulate matter, which shifts the water color towards black. The wide glint is caused by the Sun at high elevation. Also, note the darkened sides of waves in the image rendered with directional dipole compared to the one rendered with isotropic dipole and single scattering. This is due to the BSSRDF being a function of not only the distance between the point of incidence and the point of emergence but also taking into account the direction of the incident and outgoing ray. The water in Figure 8.3 has a high concentration of phytoplankton, and therefore it should have a green tint. This effect is visible well enough only in the dipole image, though. Figure 8.4 shows a Sun glint on calm water at sunset. The color of the sky reflection is more towards white due to high air turbidity. Figure 8.5 compares the effect of reflection and subsurface scattering on the final image in clear water. Figure 8.6, in turn, compares the water color due to different constituents and also shows the images obtained by Ma et al. [MXW$^+$16] for waters of the same constitution.

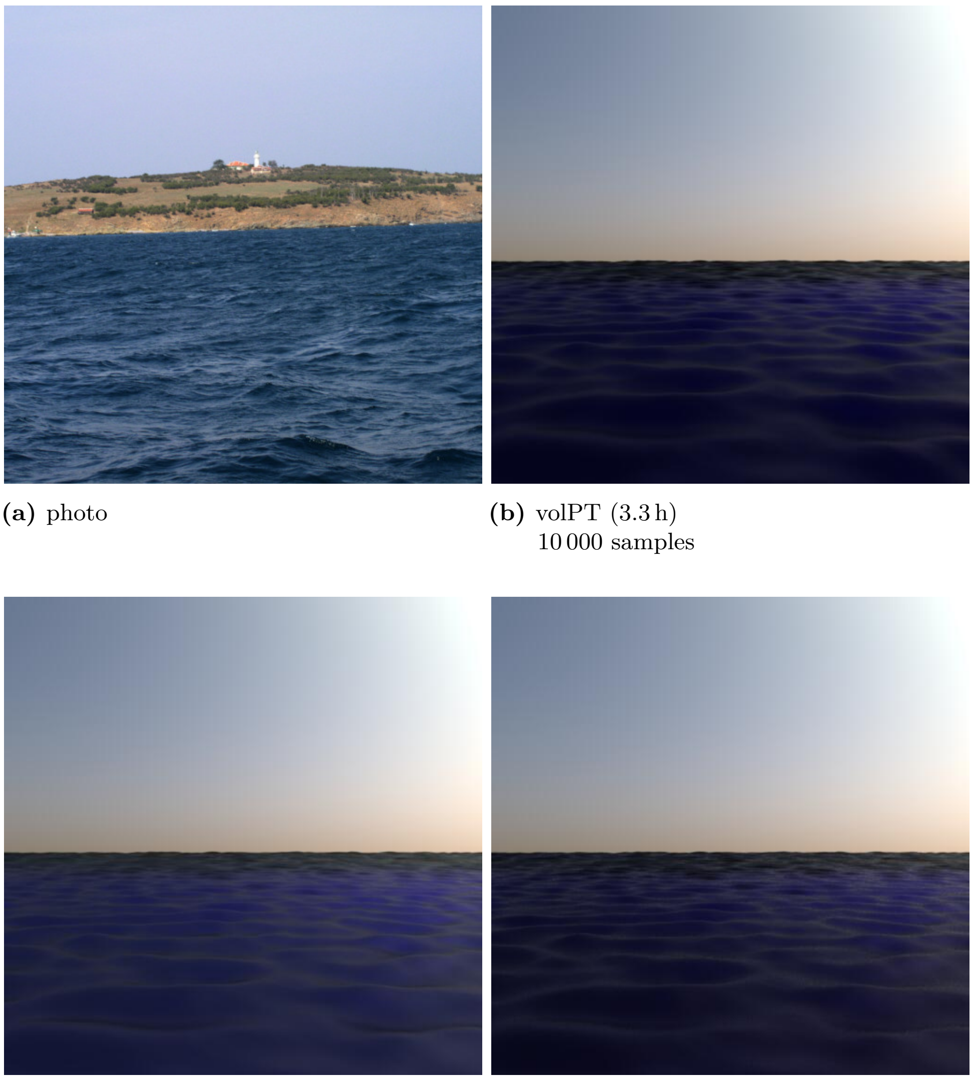
## Evaluation

From Figure 8.5, it is apparent that the skylight has an impact on the water appearance, although it is probably not going to vary much over the surface. From the remaining pictures we conclude that the geometric model of the surface could definitely be improved. When comparing the pictures with those in the original article [MWM87], the results are most likely correct. The problem is that the assumption of constant wind speed from one direction leads to this specific wave pattern, which does not fit the chosen photos. In this case, empirically adjusted Perlin noise would likely be a better choice. On the other hand, Premože and Ashikhmin [PA01] use the same approach and obtain plenty of high-frequency waves, which in our experience were removed by the wave spectrum filter. As for the BSDF, judging by the Sun glint in Figure 8.4 and the comparison with the results of Ma et al. in Figure 8.6, both the Sun and the sky reflections seem correct.

An apparent deviation in the rendered images that is notable from Figure 8.6 is in the water color. Although clear water looks acceptable, the water with phytoplankton looks quite different. Not only is it darker, but it seems rather opaque. This may be because in the reference images the concentration of phytoplankton in vertical direction follows a Gaussian, but we use constant concentration throughout the full volume. In cases of both water with high CDOM and water with high SPM content, the colors differ substantially. Looking at the absorption and scattering coefficients of SPM, the blue light scatters more and is being absorbed less than the red and green light. Furthermore, the incident light is mostly blue, and therefore it seems as the water should appear blue. Therefore the problem lies most likely in incorrect handling of the wavelength dependency of the absorption and scattering coefficients. We chose three wavelengths to represent the red, green, and blue components of the light. Apparently, they should either be chosen differently, or choosing three wavelengths is not sufficient at all.

As for the algorithms, the water surface rendered with the combination of the standard dipole and single scattering is brighter than those rendered with the remaining two methods. We have tried rendering small water cube with dipole and single scattering separately, and the result of single scattering was close to volumetric path tracing, but the result obtained from the dipole alone was brighter. We do not know what exactly is the cause of this, but it is likely not the hierarchical integration, as the directional dipole uses it too. It might be caused by the left-out term, although that is not very likely. The images rendered with the directional dipole compared to the volPT look correct. The noise in Figure 8.5 is a result of sampling only the BSSRDF during the computation of direct transmission. For BSDFs with the glossy or diffuse transmission, such as this case, multiple importance sampling both the BSSRDF and light source when leaving the medium could alleviate the noise. Even though the assumptions under which both diffusion methods are derived are violated, i.e. that the medium is mostly scattering: $r \gg 1/\sigma_s$ and $\sigma_a \ll \sigma_s$, which can hold only with sufficient concentration of SPM.

As can be observed from the design, it focuses on the accuracy of the methods, not performance. We have also not yet reached the point to be concerned with optimizations. Comparing the performance of the methods is, therefore, not very meaningful. The volumetric path tracing is obviously going to be much slower. As for the diffusion methods, the cost of evaluating the directional dipole BSSRDF is approximately the same as the standard dipole [FHK14]. The dipole must be combined with a single scattering algorithm, which can easily make it more expensive, depending on the particular algorithm used. Additionally, the implementation of the dipole BSSRDF uses SSE instructions, which also complicates the comparison. The downside of the directional dipole is that the BSSRDF can not be evaluated only once per irradiance sample, because it needs to take incident direction into account. Also, compared to the standard dipole, it needs more surface samples (unless the water is clear), otherwise the samples will appear in the image as glowing spots. A possible cause might be that the real light source in the directional dipole is placed on the surface, while in the standard dipole it is placed below the surface. Finally, we note that hierarchical integration is not well suited for large objects because the sample density is constant over the object surface. Thus the memory can easily become a limiting factor. This could potentially be solved by pruning the irradiance tree based on node distance from the camera (during construction).

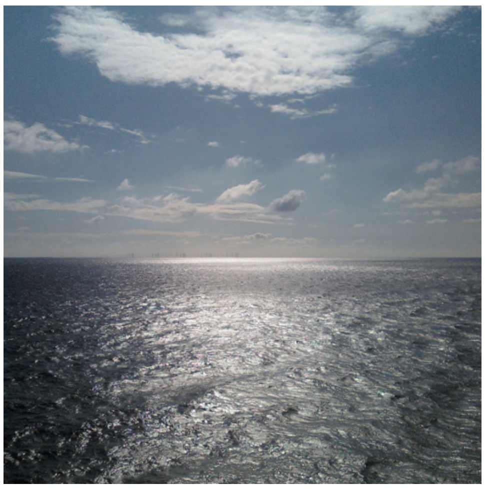

(a) photo

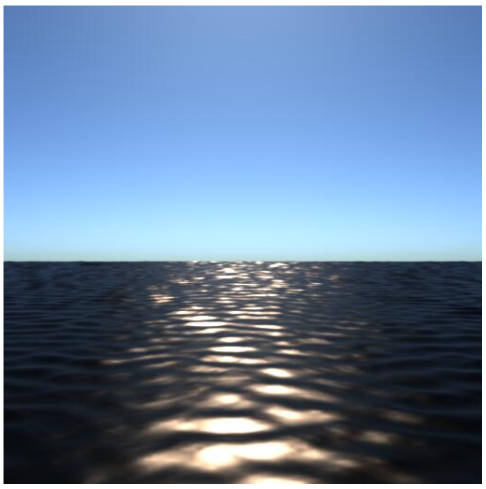
(b) volPT (3.3 h)
10 000 samples

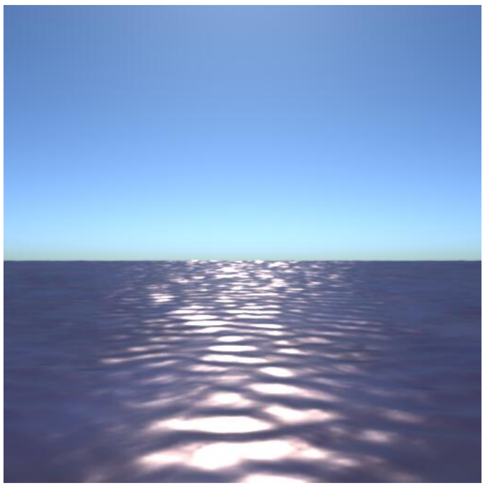
(c) dip + SS (3.1 h)
1 000 samples, 1 111 surf. samples

(d) dirDip (36 min)
100 samples, 1 111 surf. samples

**Figure 8.1:** Clear water illuminated only by the sky, wind 5 m/s.


**(a)** photo


**(b)** volPT (12 h)
10 000 samples


**(c)** dip + SS (6.1 h)
1 000 samples, 87 012 surf. samples

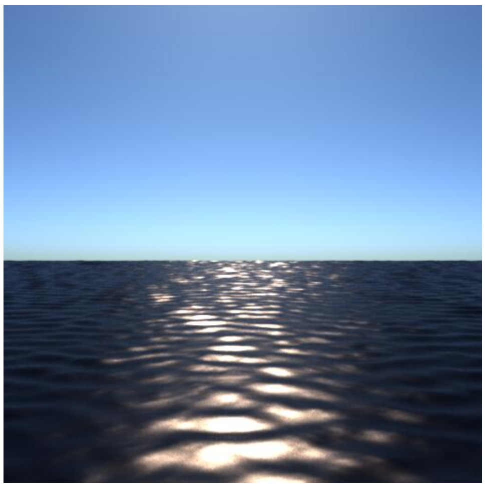
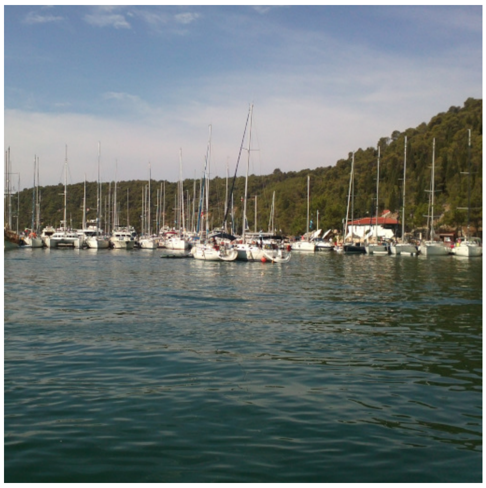
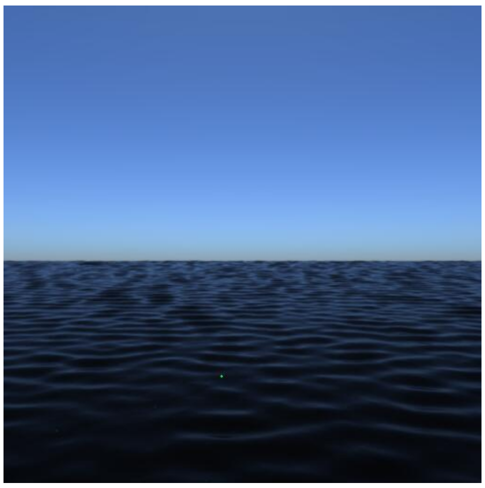
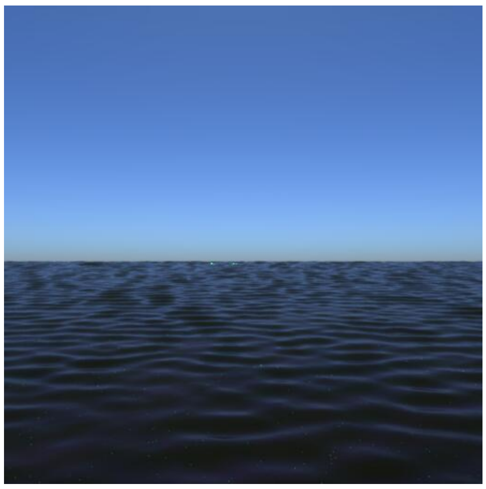

**(d)** dirDip (1.2 h)
100 samples, 871 465 surf. samples

**Figure 8.2:** Water with particulate matter concentration of 85 g/m$^3$, wind 3 m/s.


**(a)** photo


**(b)** volPT (2.3 h)
10 000 samples


**(c)** dip + SS (42 min)
100 samples, 577 125 surf. samples

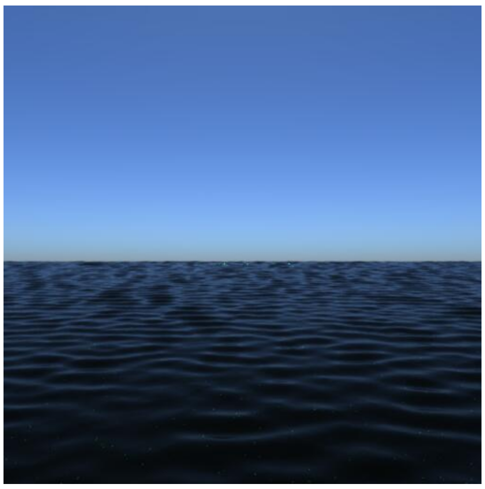
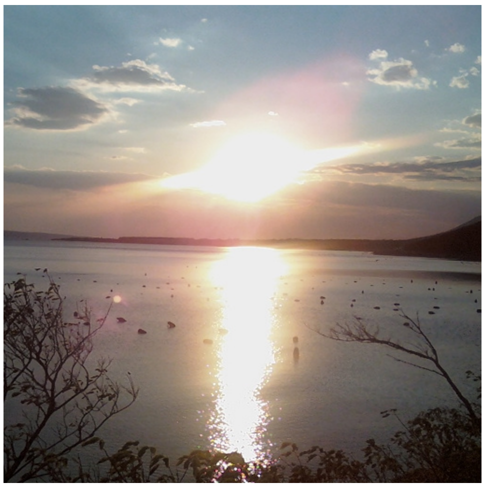
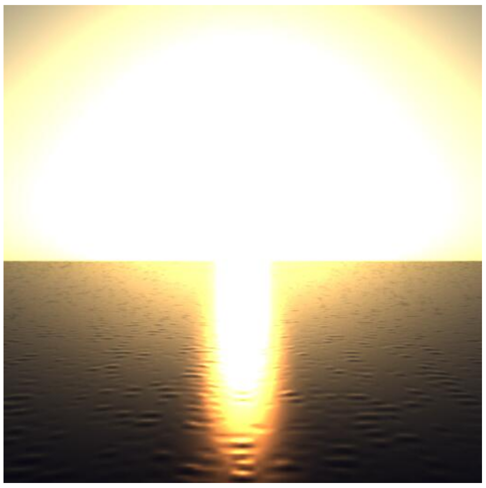

**(d)** dirDip (3.6 h)
100 samples, 2 899 917 surf. samples

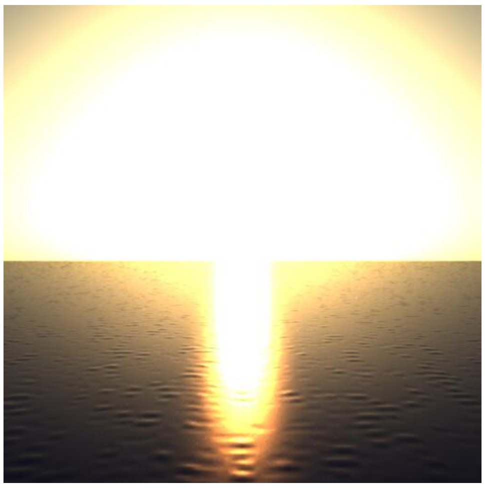
**Figure 8.3:** Water with phytoplankton concentration of 6.5 mg/m$^3$. Wind settings for wave generation was 3 m/s, but for BSDF 0.1 m/s to reproduce the smooth waves. The depth of the water was set to 5 m.



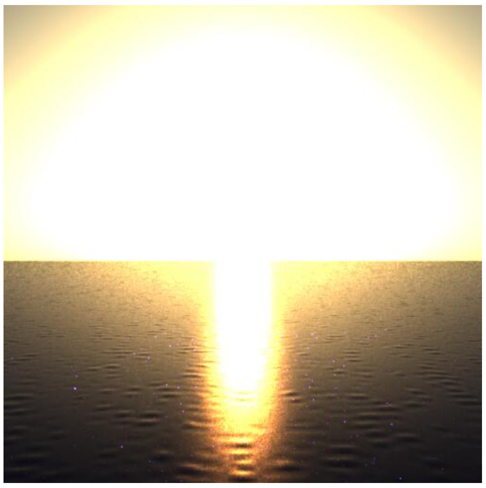
**(a)** photo



**(b)** volPT (2.6 h)
10 000 samples

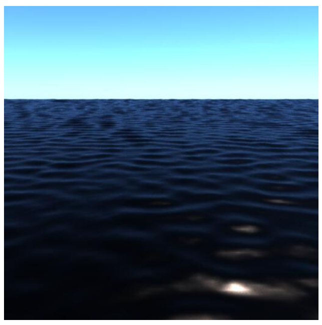


**(c)** dip + SS (3 h)
1 000 samples, 1100 surf. samples

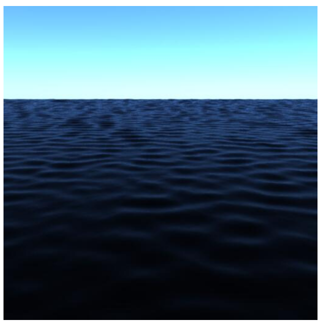


**(d)** dirDip (42 min)
100 samples, 1100 surf. samples

**Figure 8.4:** Clear water with with high air turbidity, wind speed is 1 m/s.

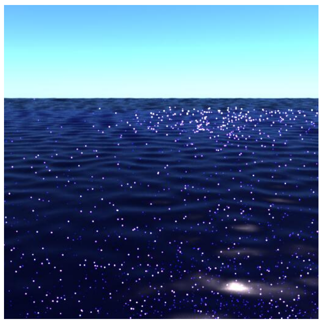


**(a)** reflection (38 s)
Sun + sky

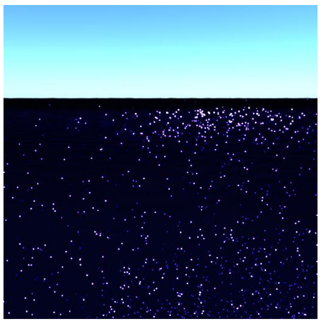


**(b)** reflection (39 s)
sky

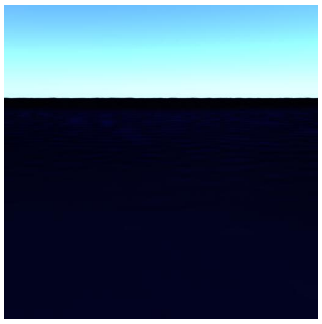
**(c)** reflection (35 s)
Sun



**(d)** reflection + dirDip (1.2 h)
Sun + sky

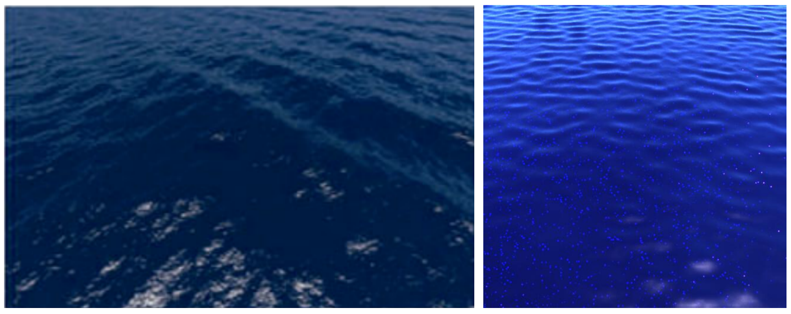


**(e)** dirDip (53 min)
Sun + sky

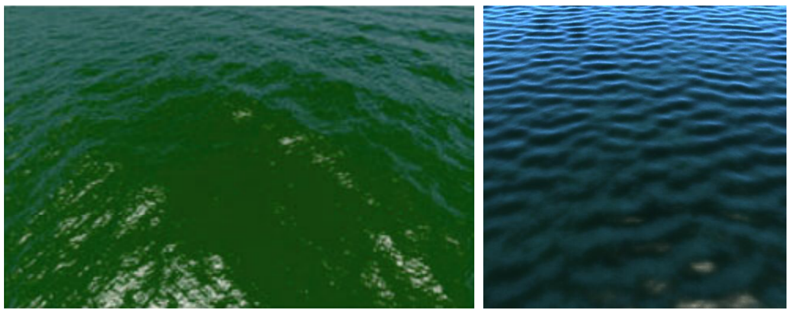


**(f)** dirDip (53 min)
sky

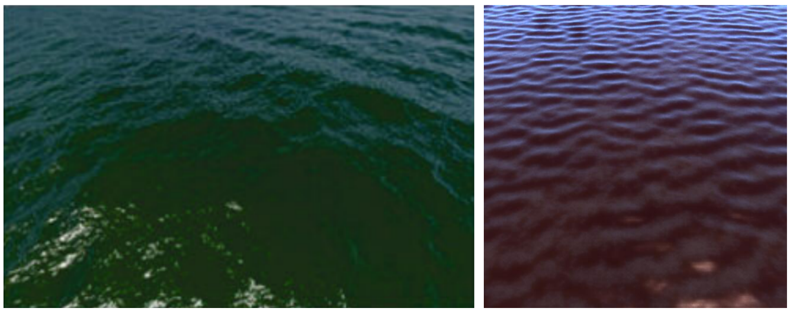
**Figure 8.5:** Reflection and subsurface components of the image, with both separate and combined Sun and sky light sources. All images used 100 samples per pixel and 1114 surface samples for direction dipole. The value of the multiplier for the LDR mapping was 6.



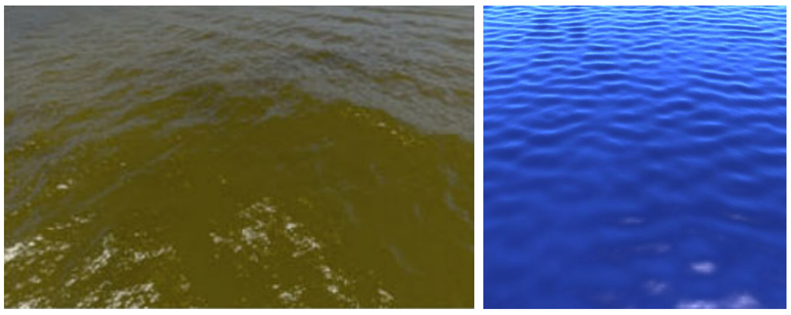
**(a)** clear water (1.6 h), 2 024 surf. samples



**(b)** high chlorophyll water (2.8 h), 1 507 932 surf. samples, phytoplankton conc. 6.5 mg/m$^3$



**(c)** high CDOM water (3.4 h), 1 320 245 surf. samples, CDOM conc. 6200 g/m$^3$



**(d)** high SPM water (2.1 h), 1 629 402 surf. samples, SPM conc. 85.3 g/m$^3$

**Figure 8.6:** Comparison of the effect of water constituents on water appearance. Images on the left are from [MXW$^+$16], on the right side are our images rendered with directional dipole. The wind speed was 3 m/s and the depth of the water was 100 m. In this case the mapping to LDR uses different multiplier value – 35.

# Chapter 9

# Conclusion

A model of a water body has been implemented as part of the Mitsuba framework, consisting of a surface geometry model, which was obtained using a Fourier transform and wave spectrum filter, a microfacet surface scattering model, and a bio-optical model of the water, which can be used to change the water color based on the concentration of various constituents. The Sun and sky model, already implemented in the rendering framework, allows for a variable time of day, air turbidity, etc., which also affects water appearance. The water is rendered with volumetric path tracing to obtain reference images. For this purpose, several sampling methods had to be implemented. A diffusion-based rendering method, directional dipole, has also been implemented and compared with another diffusion-based method – the isotropic dipole combined with accurate single scattering. All three rendering methods account for light transport in the water, and are completely generic – they can work with all light sources available in the framework and should also support objects inside the water. The implementation has been tested on several scenes with different parameters. The rendered images were compared to images obtained with other implementations, and to real photos. This chapter gives a summary of what should be fixed or what could be improved. Additionally, there are several ways the work could be extended to allow for modeling of more real-world phenomena.

## Possible improvements

Even though the model of the surface geometry is technically correct, it is applicable only under the assumption that a wind is blowing in a constant direction. It also lacks capillary waves, which are formed by the surface tension force acting against the wind. The water colors produced by the bio-optical model do not match the reference images, which is probably due to incorrect handling of the spectral dependency of the scattering and absorption coefficients. Perhaps the three chosen wavelengths are merely a bad choice. Or it could help to use average values of the spectral coefficients over a spectrum bin represented by the wavelength. As for the rendering methods, the standard dipole overestimates the radiant exitance of a medium. It might be due to some change in the code we made, or the implementation was already incorrect before, or it might come with the algorithm. The directional dipole currently produces noise when a directly transmitted ray hits a high-intensity light source. Importance sampling could alleviate this for BSDFs with the glossy or diffuse transmission. The BSDF sampling could be improved by sampling the distribution of only visible normals. Finally, most of the code could probably be optimized, taking advantage of SSE instructions in the evaluation of directional dipole BSSRDF would definitely help.

## Extensions

There are several ways the work could be extended to support more distinct water bodies. Aside from being able to render different patterns of small waves, it would be interesting to include larger, curling waves, up to the point when they break and splash. This would also require simulation and rendering of sprays and foam. Other challenges are shallow waters and coastline, where the waves rise up from the water. Including full water dynamics would allow for splashing around rocks and cliffs. Most of this would benefit from being able to render an animated image sequence.

# Appendix A

## Bibliography

[AM00] Neeharika Adabala and Swami Manohar, *Modeling and rendering of gaseous phenomena using particle maps*, The Journal of Visualization and Computer Animation **11** (2000), no. 5, 279–293.

[Bha93] Neeta Bhate, *Application of rapid hierarchical radiosity to participating media*, Proceedings of ATARV-93: Advanced Techniques in Animation, Rendering, and Visualization (1993), 43–53.

[BWWM10] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz, *Parallel poisson disk sampling with spectrum analysis on surfaces*, ACM Transactions on Graphics (TOG) **29** (2010), no. 6, 1–10.

[CCT72] L. L. Carter, E. D. Cashwell, and W. M. Taylor, *Monte carlo sampling with continuously varying cross sections along flight paths*, Nuclear Science and Engineering **48** (1972), no. 4, 403–411.

[Cha60] Subrahmanyan Chandrasekhar, *Radiative transfer*, Courier Corporation, 1960.

[CPP$^+$05] Eva Cerezo, Frederic Pérez, Xavier Pueyo, Francisco J. Seron, and François X. Sillion, *A survey on participating media rendering techniques*, The Visual Computer **21** (2005), no. 5, 303–328.

[Cra78] S. N. Cramer, *Application of the fictitious scattering radiation transport model for deep-penetration monte carlo calculations*, Nuclear Science and Engineering **65** (1978), no. 2, 237–253.

[CS02] Eva Cerezo and Francisco J. Serón, *Rendering natural waters: merging computer graphics with physics and biology*, Advances in

Modelling, Animation and Rendering, Springer, 2002, pp. 481–498.

[CT82] Robert L. Cook and Kenneth E. Torrance, *A reflectance model for computer graphics*, ACM Transactions on Graphics (ToG) **1** (1982), no. 1, 7–24.

[DCGG11] Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour, and Jean-Christophe Gonzato, *A survey of ocean simulation and rendering techniques in computer graphics*, Computer Graphics Forum **30** (2011), no. 1, 43–60.

[EMP$^+$03] David S. Ebert, Kenton F. Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley, *Texturing & modeling: A procedural approach*, third ed., Morgan Kaufmann, 2003.

[FHK14] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen, *Directional dipole model for subsurface scattering*, ACM Transactions on Graphics (TOG) **34** (2014), no. 1, 1–12.

[FPW92] Thomas J. Farrell, Michael S. Patterson, and Brian Wilson, *A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo*, Medical physics **19** (1992), no. 4, 879–888.

[FR86] Alain Fournier and William T. Reeves, *A simple model of ocean waves*, Proceedings of the 13th annual conference on Computer graphics and interactive techniques, 1986, pp. 75–84.

[Fri11] Jeppe Revall Frisvad, *Importance sampling the rayleigh phase function*, JOSA A **28** (2011), no. 12, 2436–2441.

[Gla18] Robert Glanz, *A comparison of physically based rendering systems*, Bachelor's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/E193-02, A-1040 Vienna, Austria, March 2018, [Online; accessed 25-February-2020].

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile, *Modeling the interaction of light between diffuse surfaces*, SIGGRAPH Comput. Graph. **18** (1984), no. 3, 213–222.

[Hd14] Eric Heitz and Eugene d'Eon, *Importance sampling microfacet-based bsdfs using the distribution of visible normals*, Computer Graphics Forum **33** (2014), no. 4, 103–112.

[HDE80] Dieter E. Hasselmann, M. Dunckel, and J. A. Ewing, *Directional wave spectra observed during jonswap 1973*, Journal of physical oceanography **10** (1980), no. 8, 1264–1280.

[HG41] Louis G. Henyey and Jesse L. Greenstein, *Diffuse radiation in the galaxy*, The Astrophysical Journal **93** (1941), 70–83.

[HHdD16] Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher, *Multiple-scattering microfacet bsdfs with the smith model*, ACM Transactions on Graphics (TOG) **35** (2016), no. 4, 1–14.

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen, *Stochastic progressive photon mapping*, ACM Transactions on Graphics (TOG) **28** (2009), no. 5, 1–8.

[HK93] Pat Hanrahan and Wolfgang Krueger, *Reflection from layered surfaces due to subsurface scattering*, Proceedings of the 20th annual conference on Computer graphics and interactive techniques, 1993, pp. 165–174.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala, *Virtual spherical lights for many-light rendering of glossy scenes*, ACM Transactions on Graphics (TOG) **28** (2009), no. 5, 1–6.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen, *Progressive photon mapping*, ACM Transactions on Graphics (TOG) **27** (2008), no. 5, 1–8.

[Hol15] Nicolas Holzschuch, *Accurate computation of single scattering in participating media with refractive boundaries*, Computer graphics forum **34** (2015), no. 6, 48–59.

[HT74] James E. Hansen and Larry D. Travis, *Light scattering in planetary atmospheres*, Space science reviews **16** (1974), no. 4, 527–610.

[HW12] Lukas Hosek and Alexander Wilkie, *An analytic model for full spectral sky-dome radiance*, ACM Transactions on Graphics (TOG) **31** (2012), no. 4, 1–9.

[HW13] Lukáš Hošek and Alexander Wilkie, *Adding a solar-radiance function to the hošek-wilkie skylight model*, IEEE computer graphics and applications **33** (2013), no. 3, 44–52.

[Ish78] Akira Ishimaru, *Wave propagation and scattering in random media*, vol. 2, Academic press New York, 1978.

[Jak10] Wenzel Jakob, *Mitsuba renderer*, 2010, http://www.mitsuba-renderer.org.

[JB02] Henrik Wann Jensen and Juan Buhler, *A rapid hierarchical rendering technique for translucent materials*, Proceedings of the 29th annual conference on Computer graphics and interactive techniques, 2002, pp. 576–581.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen, *Photon maps in bidirectional monte carlo ray tracing of complex objects*, Computers & Graphics **19** (1995), no. 2, 215–224.

[JC98] Henrik Wann Jensen and Per H. Christensen, *Efficient simulation of light transport in scenes with participating media using photon maps*, Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998, pp. 311–320.

[Jen01] Henrik Wann Jensen, *Realistic image synthesis using photon mapping*, AK Peters/CRC Press, 2001.

[Jer76] Nils Gunnar Jerlov, *Marine optics*, Elsevier, 1976.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan, *A practical model for subsurface light transport*, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp. 511–518.

[Kaj86] James T. Kajiya, *The rendering equation*, Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), SIGGRAPH '86, Association for Computing Machinery, 1986, p. 143–150.

[Kel97] Alexander Keller, *Instant radiosity*, Proceedings of the 24th annual conference on Computer graphics and interactive techniques, 1997, pp. 49–56.

[KGH$^+$14] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz, *Unifying points, beams, and paths in volumetric light transport simulation*, ACM Transactions on Graphics (TOG) **33** (2014), no. 4, 1–13.

[KHLN17] Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák, *Spectral and decomposition tracking for rendering heterogeneous volumes*, ACM Transactions on Graphics (TOG) **36** (2017), no. 4, 1–16.

[Kob18] Aleš Koblížek, *Real-time volumetric terrain rendering*, Bachelor's thesis, Brno University of Technology. Faculty of Information Technology. Department of Computer Graphics and Multimedia, 2018, [Online; accessed 5-August-2020].

[KVH84] James T. Kajiya and Brian P. Von Herzen, *Ray tracing volume densities*, ACM SIGGRAPH computer graphics **18** (1984), no. 3, 165–174.

[LW93] Eric P. Lafortune and Yves D. Willems, *Bi-directional path tracing*, Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93) (Alvor, Portugal), December 1993, pp. 145–153.

[M+74] Andre Morel et al., *Optical properties of pure water and pure sea water*, Optical aspects of oceanography **1** (1974), no. 1, 1–24.

[Max95] Nelson Max, *Efficient light propagation for multiple anisotropic volume scattering*, Photorealistic Rendering Techniques, Springer, 1995, pp. 87–104.

[MGG+93] Curtis D. Mobley, Bernard Gentili, Howard R. Gordon, Zhonghai Jin, George W. Kattawar, André Morel, Phillip Reinersman, Knut Stamnes, and Robert H. Stavn, *Comparison of numerical models for computing underwater light fields*, Appl. Opt. **32** (1993), no. 36, 7484–7504.

[Mor91] André Morel, *Light and marine photosynthesis: a spectral model with geochemical and climatological implications*, Progress in oceanography **26** (1991), no. 3, 263–306.

[MWM87] Gary A. Mastin, Peter A. Watterberg, and John F. Mareda, *Fourier synthesis of ocean scenes*, IEEE Computer graphics and Applications **7** (1987), no. 3, 16–23.

[MXW+16] Chunyong Ma, Shu Xu, Hongsong Wang, Fenglin Tian, and Ge Chen, *A real-time photo-realistic rendering algorithm of ocean color based on bio-optical model*, Journal of Ocean University of China **15** (2016), no. 6, 996–1006.

[NDVZJ19] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob, *Mitsuba 2: A retargetable forward and inverse renderer*, ACM Transactions on Graphics (TOG) **38** (2019), no. 6, 1–17.

[NGHJ18] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz, *Monte carlo methods for volumetric light transport simulation*, Computer Graphics Forum **37** (2018), no. 2, 551–576.

[NNDJ12a] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz, *Progressive virtual beam lights*, Computer Graphics Forum **31** (2012), no. 4, 1407–1413.

[NNDJ12b] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz, *Virtual ray lights for rendering scenes with participating media*, ACM Transactions on Graphics (TOG) **31** (2012), no. 4, 1–11.

[NRH+77] Fred Edwin Nicodemus, Joseph C. Richmond, Jack J. Hsia, Irving W. Ginsberg, Thomas Limperis, et al., *Geometrical considerations and nomenclature for reflectance*, vol. 160, Citeseer, 1977.

[ON94] Michael Oren and Shree K. Nayar, *Generalization of lambert's reflectance model*, Proceedings of the 21st annual conference on

Computer graphics and interactive techniques, 1994, pp. 239–246.

[PA01] Simon Premože and Michael Ashikhmin, *Rendering natural waters*, Computer graphics forum **20** (2001), no. 4, 189–200.

[PAS03] Simon Premože, Michael Ashikhmin, and Peter Shirley, *Path integration for light transport in volumes*, Proceedings of the 14th Eurographics workshop on Rendering, Eurographics Association, 2003, pp. 52–63.

[Pat93] Chris Patmore, *Simulated multiple scattering for cloud rendering*, Proceedings of the IFIP TC5/WG5. 2/WG5. 10 CSI International Conference on Computer Graphics: Graphics, Design and Visualization, North-Holland Publishing Co., 1993, pp. 29–40.

[Pea86] Darwyn R. Peachey, *Modeling waves and surf*, ACM Siggraph Computer Graphics **20** (1986), no. 4, 65–74.

[Per85] Ken Perlin, *An image synthesizer*, SIGGRAPH Comput. Graph. **19** (1985), no. 3, 287–296.

[Per02] Ken Perlin, *Improving noise*, ACM Trans. Graph. **21** (2002), no. 3, 681–682.

[Pet72] Theodore J. Petzold, *Volume scattering functions for selected ocean waters*, 1972, Scripps Institution of Oceanography La Jolla Ca Visibility Lab.

[PF97] Robin M. Pope and Edward S. Fry, *Absorption spectrum (380–700 nm) of pure water. ii. integrating cavity measurements*, Appl. Opt. **36** (1997), no. 33, 8710–8723.

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys, *Physically based rendering: From theory to implementation*, Morgan Kaufmann, 2016.

[PJM64] Willard J. Pierson Jr. and Lionel Moskowitz, *A proposed spectral form for fully developed wind seas based on the similarity theory of sa kitaigorodskii*, Journal of geophysical research **69** (1964), no. 24, 5181–5190.

[PKK00] Mark Pauly, Thomas Kollig, and Alexander Keller, *Metropolis light transport for participating media*, Rendering Techniques 2000, Springer, 2000, pp. 11–22.

[RDP05] Vincent Ross, Denis Dion, and Guy Potvin, *Detailed analytical approach to the gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface*, JOSA A **22** (2005), no. 11, 2442–2453.

[RSK08] Matthias Raab, Daniel Seibert, and Alexander Keller, *Unbiased global illumination with participating media*, Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, 2008, pp. 591–605.

[RT87] Holly E. Rushmeier and Kenneth E. Torrance, *The zonal method for calculating light intensities in the presence of a participating medium*, ACM SIGGRAPH Computer Graphics **21** (1987), no. 4, 293–302.

[Rus88] Holly Edith Rushmeier, *Realistic image synthesis for scenes with radiatively participating media*, Phd thesis, Cornell University, 1988.

[SB81] Raymond C. Smith and Karen S. Baker, *Optical properties of the clearest natural waters (200–800 nm)*, Applied optics **20** (1981), no. 2, 177–184.

[Sch94] Christophe Schlick, *An inexpensive brdf model for physically-based rendering*, Computer graphics forum **13** (1994), no. 3, 233–246.

[SM47] Harald Ulrik Sverdrup and Walter Heinrich Munk, *Wind, sea and swell: Theory of relations for forecasting*, no. 303, Hydrographic Office, 1947.

[Smi67a] Bruce G. Smith, *Geometrical shadowing of a random rough surface*, IEEE transactions on antennas and propagation **15** (1967), no. 5, 668–671.

[Smi67b] Bruce G. Smith, *Lunar surface roughness: Shadowing and thermal emission*, Journal of Geophysical Research **72** (1967), no. 16, 4059–4067.

[Sta95] Jos Stam, *Multiple scattering as a diffusion process*, Rendering Techniques' 95, Springer, 1995, pp. 41–50.

[SZZW12] Jinjin Shi, Dengming Zhu, Yingping Zhang, and Zhaoqi Wang, *Realistically rendering polluted water*, The Visual Computer **28** (2012), no. 6-8, 647–656.

[T$^+$01] Jerry Tessendorf et al., *Simulating ocean water*, Simulating nature: realistic and interactive techniques. SIGGRAPH **1** (2001), no. 2, 5.

[TIS08] Art Tevs, Ivo Ihrke, and Hans-Peter Seidel, *Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering*, Proceedings of the 2008 symposium on Interactive 3D graphics and games, 2008, pp. 183–190.

[vdH81] Hendrik Christoffel van de Hulst, *Light scattering by small particles*, Courier Corporation, 1981.

[VG97] Eric Veach and Leonidas J. Guibas, *Metropolis light transport*, Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (USA), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., 1997, p. 65–76.

[VN51] John Von Neumann, *13. various techniques used in connection with random digits*, Appl. Math Ser **12** (1951), no. 36-38, 5.

[Wal05] Bruce Walter, *Notes on the ward brdf*, 2005, Program of Computer Graphics, Cornell University, Technical report PCG-05.

[Wei00] Stefan Weinzierl, *Introduction to monte carlo methods*, 2000, arXiv preprint hep-ph/0006269.

[WMHL65] 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*, Proceedings of the conference on applications of computing methods to reactor problems, 1965, vol. 557, 1965.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance, *Microfacet models for refraction through rough surfaces.*, Rendering techniques **2007** (2007), 18th.

# Appendix B

# Installation and scene setup

This appendix starts with a brief installation guide for Debian based Linux distributions. Other distributions, Windows and Mac are supported as well, see Mitsuba documentation. Then follows the description of a scene file with example and commentary on how to configure the scene for the three rendering methods. Note that there were some changes in the file format, so it is not compatible with the mainline version. The differences are listed in the implementation chapter.

## B.1 Installation

Mitsuba requires GCC version 4.2 or higher. It uses SCons build system on all the supported platforms. First, make sure all the dependencies are installed:

```
sudo apt-get install build-essential scons mercurial
    qt4-dev-tools libpng12-dev libjpeg-dev libilmbase-
    dev libxerces-c-dev libboost-all-dev libopenexr-dev
```

To get Collada support, it needs to be installed as well. Then go to the Mitsuba root directory and choose the build configuration file from the *build* directory:

```
cp build/config-linux-gcc.py config.py
```

Then run the build with:

```
scons
```

In our experience, Scons does not work with Python 3 – version 2 has to be used (it is a Python script). Option *-j core count* can be specified to parallelize the build process.

Then set up the shell environment by:

```
source setpath.sh
```

Now you can run the renderer with:

```
mitsuba scene.xml
```

## B.2 Scene setup

This section shows how to set up a scene and how to render it using the three presented methods. Listing B.1 shows an example of a scene file with the directional dipole subsurface integrator. The description of the parameters is in table B.1.

```
<?xml version="1.0" encoding="utf-8"?>
<scene version="0.5.0">

  <integrator type="path">
    <integer name="maxDepth" value="10"/>
  </integrator>

  <sensor type="perspective">
    <float name="fov" value="50"/>
    <transform name="toWorld">
```

```
      <lookat origin="-4,6,0" target="0,5,0" up="0,1,0"/>
    </transform>
    <sampler type="independent">
      <integer name="sampleCount" value="1"/>
    </sampler>
    <film type="hdrfilm">
      <integer name="width" value="512"/>
      <integer name="height" value="512"/>
      <boolean name="banner" value="false"/>
      <boolean name="attachLog" value="false"/>
    </film>
  </sensor>

  <medium type="waterMedium" id="water">
    <float name="transmittanceSampleDensity" value="1"/>
    <rgb name="sigmaAchlor" value="18.2,9.7,33.2"/>
    <rgb name="sigmaAwater" value="0.045,0.0558,0.0156"/>
    <rgb name="sigmaSwater" value="0.0007,0.0021,0.0037"/>
    <float name="scale" value="1"/>

    <volume type="constvolume" name="Cchlor">
      <float name="value" value="0"/>
    </volume>

    <volume type="constvolume" name="Corganic">
      <float name="value" value="0"/>
    </volume>

    <volume type="constvolume" name="Cpart">
      <float name="value" value="0"/>
    </volume>
  </medium>

  <shape type="volheightfield">
    <transform name="toWorld">
      <rotate x="1" angle="-90"/>
      <scale x="50" z="50"/>
      <translate x="50" y="-15"/>
    </transform>

    <texture type="bitmap">
      <string name="filename" value="heightmap_3_100_512_0.exr"/>
    </texture>

    <float name="bottom" value="-15"/>
    <float name="scale" value="30"/>
    <integer name="repeatY" value="1"/>
    <integer name="repeatX" value="1"/>

    <subsurface type="directionaldipole">
      <integer name="irrSamples" value="1"/>
      <float name="sampleMultiplier" value="1"/>
      <boolean name="irrIndirect" value="false"/>
      <integer name="diffIrrDirections" value="8"/>
      <integer name="rrDepth" value="10"/>
      <boolean name="transmission" value="true"/>
      <bsdf type="seabsdf">
        <float name="windSpeed" value="1.5"/>
      </bsdf>
    </subsurface>

    <bsdf type="seabsdf">
      <float name="windSpeed" value="1.5"/>
      <boolean name="transmission" value="false"/>
    </bsdf>

```

```
77      <ref name="interior" id="water"/>
78    </shape>
79
80    <shape type="cube">
81      <transform name="toWorld">
82        <scale x="50" z="50"/>
83        <translate x="50" y="-31.2"/>
84      </transform>
85
86      <bsdf type="diffuse">
87      </bsdf>
88    </shape>
89
90    <emitter type="sunsky">
91      <transform name="toWorld">
92        <rotate z="1" angle="-3.5"/>
93      </transform>
94      <float name="turbidity" value="1"/>
95      <float name="hour" value="10"/>
96      <integer name="resolution" value="4096"/>
97    </emitter>
98  </scene>
```

**Listing B.1:** Examle of a scene with directional dipole subsurface integrator.

| | |
|---|---|
| 14 | number of samples per pixel |
| 25 | number of medium samples per meter when computing transmittance |
| 26 | chlorophyll-specific abs. coef. [$m^2/g$] |
| 27 | pure water abs. coef. [$m^{-1}$] |
| 28 | pure saltwater scat. coef. [$m^{-1}$] |
| 30 | concentration is in [$g/m^3$] |
| | alternative data source types: gridvolume, vertGaussianVolume |
| 47 | the default size is 2x2, scale to the size covered by the heightmap |
| 48 | keep the sea level at the mean value of the heightfield |
| | the heightfield values are centered around $0.5 * vertical\_scale$ |
| 55 | water depth: $bottom = 15 - water\_depth$ |
| 56 | vertical_scale: up to 30 m wave height (wind speed cca 20 m/s) |
| 57 | tiling – not needed |
| 61 | N of irradiance samples to estimate irradiance at a point on the surface |
| 62 | controls density of the surface samples |
| 64 | number of differential irradiance directions |
| 65 | number of total internal reflections before RR starts to be applied |
| 66 | enables direct transmission component |
| 74 | disable BSDF transmission for the main integrator |
| | because it is accounted for by the subsurface |
| 80 | seafloor – should be positioned slightly below the water volume |
| 83 | the size along y axis is 2, centered at the origin |
| | therefore the y offset is $-(water\_depth + 1 + epsilon)$ |
| 92 | to avoid a gap between the sea surface and the sky |

**Table B.1:** Parameter description and comments of a scene file, which uses directional dipole.

To render the same scene with the standard dipole and single scattering, the subsurface block (lines 60-70 in Listing B.1) should be replaced by Listing B.2.

```
 1 <subsurface type="dipole">
 2   <integer name="irrSamples" value="1"/>
 3   <float name="sampleMultiplier" value="1"/>
 4   <boolean name="irrIndirect" value="false"/>
 5
 6   <bsdf type="dielectric">
 7     <string name="intIOR" value="water"/>
 8     <string name="extIOR" value="air"/>
 9   </bsdf>
10 </subsurface>
11
12 <subsurface type="singlescatter">
13   <boolean name="fastSingleScatter" value="true"/>
14   <integer name="fssSamples" value="3"/>
15   <integer name="singleScatterDepth" value="5"/>
16   <bsdf type="seabsdf">
17     <float name="windSpeed" value="1.5"/>
18   </bsdf>
19 </subsurface>
```

**Listing B.2:** Configuration of dipole and singlescattering integrators.

| | |
|---|---|
| 6 | BSDF is needed here only to specify the indices of refraction |
| 14 | number of samples (scattering events) on the refracted source ray |
| 15 | maximum recursion depth of internal reflections |
| 16 | the same BSDF as used for outside reflections should be used here only with transmissions enabled |

**Table B.2:** Parameters of dipole and single scattering integrator.

To render the same scene in Listing B.1 with volumetric path tracing, the subsurface block needs to be removed, the path tracing integrator, which does not account for volumetric effects needs to be changed to volumetric path tracing (at line 4 change "path" to "volpath"), and transmissions need to be enabled in the BSDF (line 74). Also, the number of pixel samples will need to be increased (line 14).