Path tracing in Production

The Path of Water

Marc Droske (Organizer)
Wētā Digital / Unity

Johannes Hanika (Organizer)
Karlsruhe Institute of Technology

Jiří Vorba
Wētā Digital / Unity

Andrea Weidlich
NVIDIA

Manuele Sabbadin
Wētā Digital / Unity

SIGGRAPH 2023 Course Notes

Image from Avatar: The Way of Water *©2023 Twentieth Century Film Corporation. All rights reserved.*

SIGGRAPH '23 Courses, August 06-10, 2023, Los Angeles, CA, USA
ACM 979-8-4007-0145-0/23/08.
10.1145/3587423.3595520

Abstract

Physically-based light transport simulation has become a widely established standard to generate images in the movie industry. It promises various important practical advantages such as robustness, lighting consistency, progressive rendering and scalability. Through careful scene modelling it allows highly realistic and compelling digital versions of natural phenomena to be rendered very faithfully. The previous *Path Tracing in Production* courses have documented some of the evolution and challenges along the journey of adopting this technology, yet even modern production path-tracers remain prone to costly rendering times in various classes of scenes, of which water shots remain among those most notoriously demanding.

While this series in the past years covered a wide range of different topics within one course, this year we took the unusual step to focus on just one, the water-related challenges that we encountered during our work on *Avatar: The Way of Water*. Despite its seemingly simple nature, water causes a very multifaceted range of issues: specular surfaces cause spiky and sparse radiance distribution at various scales and in different forms, such as underwater caustics, godrays as well as fast-moving highlights and complex indirect on FX elements such as splashes, droplets and aeration bubbles. The purpose of this course is to share knowledge and experiences on the current state of the technology to stimulate active exchange in the academic and industrial research community that will advance the field on some of the challenging industrial benchmark problems.

We will first give an overview of the nature of the singularities and its practical implications and then dive deeper into appearance and material aspects of water and the objects it interacts with. In the remaining sections, the course will focus on some specific aspects in more technical detail, providing both a solid mathematical background as well as practical strategies. Furthermore, we discuss some of the remaining unsolved problems that hopefully will inspire future research.

Contents

1 Syllabus

14:00 — Opening statement and overview (15 min, Marc Droske)

In the opening talk, Marc will introduce the course and provide some motivation giving a broad overview of the challenges that production rendering is faced with when rendering water. This includes some fundamental prototype cases to illustrate some of the fundamental and overarching themes that will be discussed in more detail in the course, as well as stating the general aims and questions to be answered.

14:15 — Reflecting on the Appearance of Water (35 min, Andrea Weidlich)

Water is a remarkable complex material which presents unique challenges in computer graphics. It can question our understanding of appearance and unpredictably change how objects look when submerged in it. In this part, we will discuss the appearance changing properties of water. We will dive deeper into the way water influences colour and alters reflection properties before presenting a novel approach on how to combine artistic workflows and physical properties by introducing water-compliance in traditional BSDFs.

14:50 — Caustics (35 min, Jiří Vorba)

Caustics, the shiny patterns created by focused light when it interacts with curved specular surfaces, are very important phenomena for realistic water appearance in many production shots. Simulating caustic light transport is notoriously hard and multiple algorithms have been developed in the past. We will discuss our motivation for computing caustic light transport over traditional projecting methods while considering pros and cons, challenges caused by specific production scenes for existing transport algorithms, suitability of various techniques in different scenarios, and our improvements in this field. We also show that in practice, some production shots can be rendered with guided forward path tracing when we trade costly precise solution for precision by optimal relaxation of light transport. Last but not least, we will not forget discussing specifics of rendering caustics on materials with subsurface scattering and our approach to rendering so-called god-rays, a type of volumetric caustic.

15:25 — Singularities inside the water volume (35 min, Johannes Hanika)

Participating media come with their own unique set of rendering challenges. Realistic water renditions include this to a large extent. Examples are the cone of a torch light, sun rays, the glow of back-lit spray, and the characteristic blur of light with distance under water. One common mathematical theme here can be summarised as "singularities", or near-singularities in the form of very peaky radiance fields. Water naturally comes with a specular surface material, introducing the first singularity before the light even enters the medium. Under water, the phase functions often encountered on air bubbles or particles are highly forward scattering to an extent that the sampling routines have to treat them as specular. Any kind of deterministic connection between paths will lead to problems: evaluating a scattering response off-peak will yield close to zero contribution. What's more, deterministic connections usually don't control the distance between the connected vertices, introducing one more singularity when dividing by the square distance which can tend to zero. This presentation will detail the challenges and present a recent sampling technique (OMNEE) in this context to approach one particular setting from an unusual angle: sample the angle of a scattering interaction before sampling the position.

16:00 — Taming highlights and FX elements (35 min, Manuele Sabbadin)

Specular highlights are fundamental to being able to read the shape of the water elements and their high-frequency features. This becomes especially true in a VFX scenario, where the CG water has to blend seamlessly with the real footage and ad-hoc light setups are prepared to achieve the required artistic look. FX water elements, such as foam, underwater aeration bubbles, and spray particles occur when fast water elements interact with each other. We will describe our experience in managing such elements, and the different techniques

that we have applied to reach the required visual appearance with an affordable amount of render time and memory.

After a description of the different FX elements arising from a water simulation, we will first discuss how filtering the normal distribution function can help in retrieving highlights in fast-motion scenes. We will show how the same technique can be used to facilitate the sampling in out-of-focus regions. Secondly, we will discuss the design and use of specific geometric primitives, such as bubbles and blobbies, which have been extensively used to blend the main bulk of water with the small isolated particles of spray around it. They have shown to be effective in regions where the structure and topology of the water simulation change quickly. Finally, we will talk about the conversion from a set of particles to a volume, which can be a practical way to reduce the number of particles to consider during light transport, and it is a viable solution whenever the particle size is small enough in comparison to the pixel footprint.

16:35 — Closing remarks and open challenges (10 min, Marc Droske)

16:45 — Q&A with all presenters (15min)

2 Presenters

2.1 Marc Droske, Wētā Digital / Unity

Marc Droske is Head of Rendering Research at Wētā Digital / Unity. He received his PhD in Applied Mathematics in the institute for Numerical Simulation, Duisburg where he focused on differential geometry, inverse problems, variational methods and PDEs. After a PostDoc year at UCLA, he joined Mental Images (later NVIDIA ARC) where he worked on geometry processing algorithms and GPU-accelerated ray-tracers. He joined Wētā Digital in 2014 to work on the light-transport system of their in-house production renderer Manuka.

2.2 Andrea Weidlich, NVIDIA

Andrea Weidlich is a Principal Researcher at Nvidia in the Realtime Rendering Research department. Before joining Nvidia, she worked for Weta Digital where she designed the material system attached to Weta's proprietary physically-based renderer, Manuka. Her main research areas are appearance modelling and material prototyping. Andrea holds a Master of Arts in Applied Media from University of Applied Arts, Vienna and a Ph.D. in Computer Science from Vienna University of Technology.

2.3 Jiří Vorba, Wētā Digital / Unity

Jiří Vorba is a senior rendering researcher and developer at Wētā Digital / Unity. He has received his Ph.D. in Computer Science from *Charles University in Prague* in 2017. In 2014, he implemented path guiding into *Manuka* renderer during his internship with Wētā Digital. Since joining Wētā Digital in 2015, his primary focus has been on light transport and caustic rendering in vast production scenes.

2.4 Johannes Hanika, Karlsruhe Institute of Technology

Johannes was introduced to the secrets of computer graphics in Ulm University, working on low level ray tracing optimisation, quasi-Monte Carlo point sets, light transport simulation, and image denoising. The following ten years he worked as a researcher for Weta Digital, first in Wellington, New Zealand, later remotely from Germany. He is co-architect of Manuka, Weta Digital's physically based spectral renderer. Since 2013, Johannes works as a post-doctoral fellow at the Karlsruhe Institute of Technology. In 2009, Johannes founded the darktable open source project, a workflow tool for RAW photography. Johannes has a movie credit for "Abraham Lincoln: Vampire Hunter".

2.5 Manuele Sabbadin, Wētā Digital / Unity

Manuele Sabbadin is a rendering researcher at Wētā Digital, Wellington. He received his Ph.D. in Computer Science from the University of Pisa, Italy, in 2019, where he also got his M.Sc. degree. For his Ph.D., he completed his research on techniques for the interaction and rendering of harvested 3d data at the Visual Computing Lab, ISTI, CNR, Pisa. Since he joined Wētā in 2019, he worked foremostly on ray tracing of implicit surfaces.

3 Introduction and overview

Marc Droske, *Wētā Digital*

3.1 Problem statement

One of the compelling propositions that contributed to the wide adoption of path-tracing in modern movie production pipelines is the drastic simplification of the rendering setup, compared to traditional approaches in which for example occlusion information was precomputed in a separate pass and then carefully combined in a final beauty. Path-tracers are extraordinarily reliable and robust, and have greatly evolved in terms of handling more and more complex light transport scenarios by adopting more advanced sampling techniques, and supporting different primitives and materials to allow for effective appearance filtering techniques to be employed. In addition, the leap in performance of modern denoising techniques has continously lowered the amount of Monte Carlo samples required to reach a desired target quality.

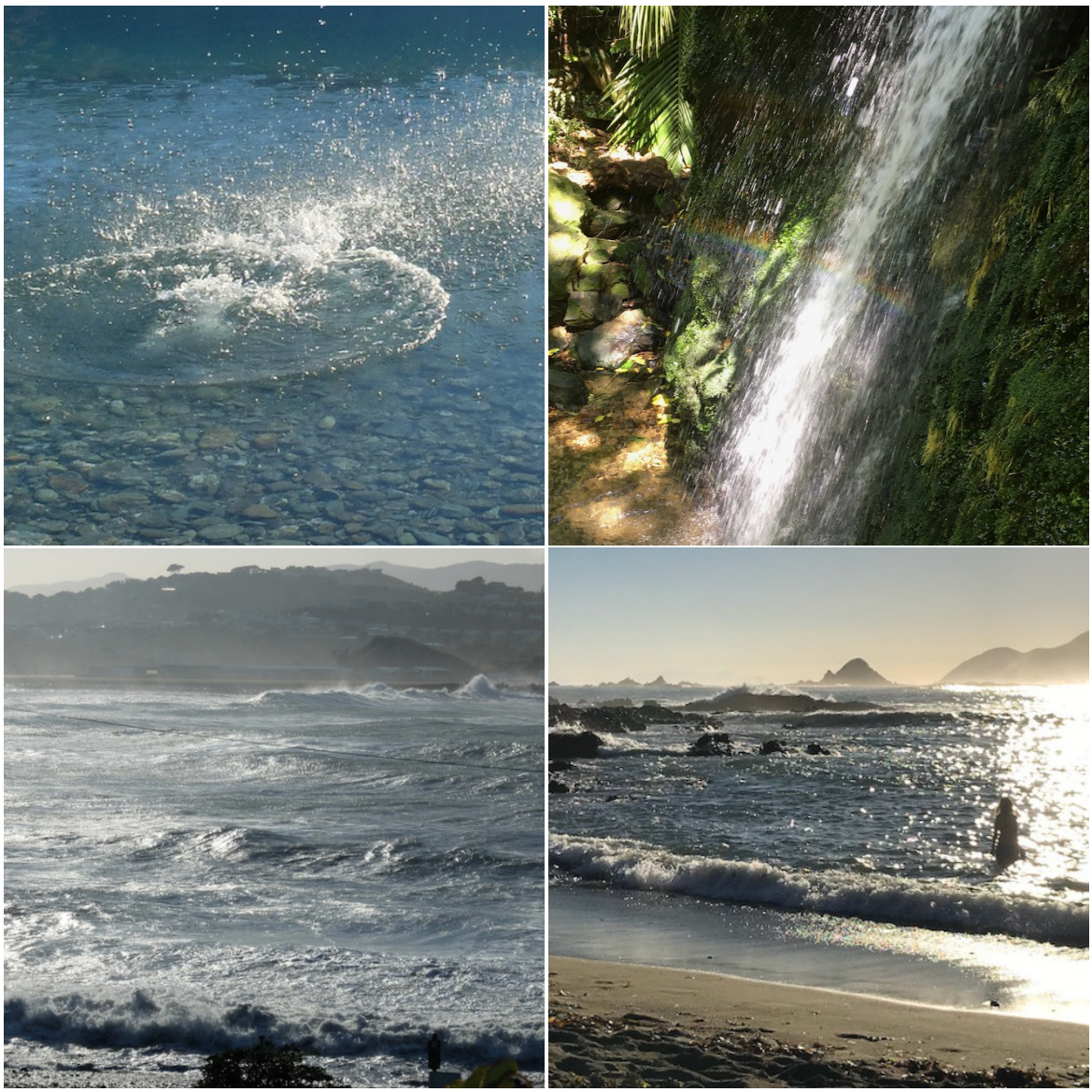


Figure 3.1: *Natural water exhibits a wide variety of specular highlights.*

VFX shots tend to be very demanding in terms of complexity, detail and lighting. Due to the wide range of phenomena that need to be faithfully rendered it still remains common to encounter classes of scenes that path tracers struggle with. Luckily, these are commonly one-offs: a scene with a chandelier perhaps, or a room full of glass jars or a sesquin dress lit by camera flashlights. Such issues are usually due to some spiky, irregular distributions of the radiance field, often caused by the interplay of specular materials, and small, bright light sources or irregular, complex geometry. Especially having the relative cost of CPU versus human time in mind, it may be acceptable to let the machine "converge it out" or applying some tailored optimisations to the scene setup to get through the sequence.

The presence of water typically causes at least some of form of challening light transport scenario. Each of these individually may of course not be specific to water: caustics for example are fairly common on other types of scenes. Just the presence of a whiskey glass, or metallic objects suffices to make these appear. However, water shots challenge a production renderers (and their users) to an extreme degree, since they exhibit a remarkably wide range of challenges combined. Unfortunately, there exists no one-size-fits-all strategy to cope with all the challenges in a unified manner. Brute-force no longer presents a viable option, and this is especially true if a major portion of movie consists of water, like in *Avatar: The Way of Water*.

As we will discuss in more detail in this course, the range of strategies is as wide as the range of problems to be addressed, which causes some design choices to be made, and has implications on the rendering architecture and how it is used effectively in practice. But before we dive in, let's start with a high-level overview of some of the qualitative properties of the water. We won't go too deep into physics here and focus the relevance to optics and light-transport. The next section will cover the appearance of aspects such as the spectral absorption and scattering properties of different ocean water types of in more detail and we refer to Mobley [2022] as an excellent resource on its geometric and optical properties.

3.2 Qualitative properties

Specularity

Water is a liquid with high surface tension. This effectively causes its surface area to be minimized via mean-curvature flow, which is well-known to yield smooth surfaces for any arbitary short time from any input configuration. Therefore, there is locally no roughness in the sense of statistical microfacet distribution, the variation of the normal is due to local differential geometry. More precisely, this corresponds to a question of scale: the area at which a microfacet distribution model becomes a valid representation is more in the range of meters, such as a patch of ocean on the horizon. Smaller features, such as the surfaces of droplets, bubbles, splashes can be considered as locally specular.

Low absorption

At the surface level, photons either reflect or transmit. In clear water the mean free path of in the visible spectrum is roughly in the range of 0.2-80m. Therefore, high throughput paths may be very long, causing complex indirect, and non-local interactions.

Geometry

Gravity waves. Also known as surface gravity waves or ocean waves that refer to the disturbances that propagate along the surface of the ocean due to the gravitational forces acting on the water. These waves are a result of the interaction between wind, the Earth's gravity, and to a lesser degree the water's surface tension.

They are primarily generated by wind blowing across the surface of the water. The frictional drag of the wind transfers energy to the water, causing ripples and waves to form. The size and strength of the waves depend on the wind speed, duration, and fetch (the distance over which the wind blows).

The size distribution is varies ranging from smaller ripples to massive swells, their wavelenght can range from a few centimeters to hundreds of meters. Ocean waves of different wavelengths travel at different speeds. This phenomenon, known as dispersion, causes waves to separate or group together as they

propagate. As waves move away from their generation area, longer waves tend to move faster than shorter waves.

Ocean waves can interact with one another, leading to phenomena such as wave interference, where waves can combine to creating larger waves or canceling each other out. This interaction can result in complex wave patterns, including rogue waves, which are exceptionally large and can occur spontaneously.

A specific mathematical representation derived from the Navier-Stokes equations are so called *Gerstner waves*. They describe the oscillatory motion of water particles caused by the restoring forces of gravity and surface tension.

Capillary waves. Capillary waves are small-scale waves with wavelengths of only a few centimeters. They are often generated by disturbances such as raindrops, other local external forces or light winds. In contrast to gravity waves, where gravity is the primary restoring force, capillaries forces arise due to the cohesive properties of the liquid and tend to restore the disturbed surface to its original state.

Figure 3.2: Left: *Capillary waves.* Middle/Right: *Water splash entrapping bubbles and generating spray.*

Capillary waves have a higher propagation speed compared to gravity waves. Due to the dominance of surface tension forces, capillary waves can travel at speeds exceeding 30 centimeters per second. Gravity waves, however, have slower speeds and are affected by factors such as water depth, wave steepness, and wind conditions.

Bubbles. The breaking of waves under moderate to high wind conditions causes microbubbles to get entrapped beneath the sea surface, which creates a layer of bubbles sustained by the continuous supply of bubbles from frequent wave breaking. to hours. They have diameters of typically less than a millimeter. They gradually merge into the overall population of fine bubbles over a time ranging from minutes to hours. Fine bubbles have diameters ranging from 1 millimeter to several centimeters. They are short-lived, as they rapidly rise to the ocean's surface. These bubbles contribute to the visual appearance of frothy water, particularly in areas with high wave activity.

Larger macro bubbles with diameters typically greater than several centimeters are often associated with specific natural phenomena, sea life or ship propeller action. These bubbles rise more slowly compared to fine bubbles due to their larger size and may persist for longer durations before reaching the surface.

Splashes. These are situations such as when an object impacts or enters the water surface, a wave crashing a against a solid barrier. Naturally, the form and pattern depends on velocity and force of impact, the shape of the objects involved.

The geometric structure of water splashes exhibits a greater degree of irregularity and visual distinctiveness in comparison to smaller-scale agitations, such as bubble formations. It often displays unique sheeting patterns and undergoes various topological changes as a result of the high momentum of the water mass.

The droplets released during the splash disperse into the surrounding air or water, gradually losing momentum and eventually returning to the liquid phase.

Spray. Spray forms as droplets disperse into the surrouding air and can be observed in various scenarios, such as fountains, waterfalls, high-pressure water jets, and splashes. The size varies from tiny droplets of a few micrometers to several millimeters in diameter.

While a droplet is suspended in air, and exposed to external forces, it oscillates around its equilibrium

position, minimizing the surface area.

Foam. Bubbles rising to the sea surface and accumulating leads to the formation of a layer of surface foam. Foam bubbles tend to merge or coalesce with each other over time. This coalescence process contributes to the growth and stability of the foam layer. The merging of bubbles is facilitated by surface tension, creating bubbles from sub millimeter to multiple centimeters in size. Thickness also varies from a very thin layers of bubbles on the ocean surface, to thicker volumetric foam structures.
Depending on the presence of chemical substances, foam bubbles with a longer life-time tend to merge and build inner walls, the minimal surfaces (Plateau's surfaces). This is referred to as *dry foam*, as opposed to *wet foam*, which consists of separated spherical foam bubbles.

From a rendering point of view, this is a quite intimidating set of features. First, we have to deal with wide range of different geometric representations that may undergo topology changes, a combination of particles, droplets and bulk surfaces as well as volumetric representations.

Secondly, indirect light transport becomes extremely complex, high dimensional and chaotic.

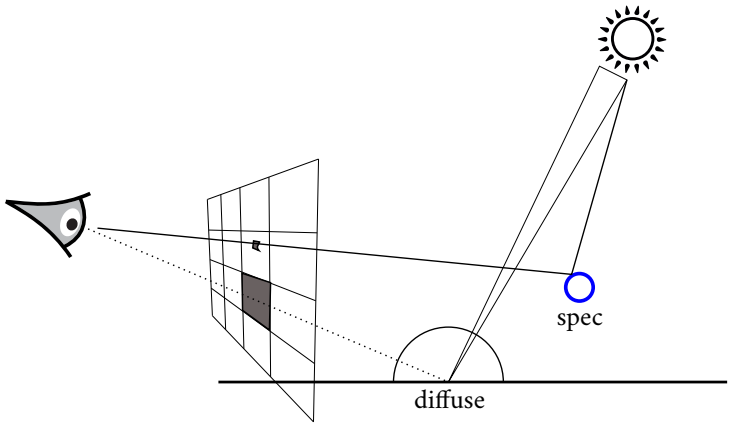


Figure 3.3: *Diffuse object and direct highlight reflecting towards camera sensor under same exposure.*

3.3 High-dynamic range and sun exposure

If it wasn't for the presence of sun light, forward path-tracing would mostly have to cope with long paths in terms of ray-tracing cost corresponding to the number of bounces. The small solid angle and high-intensity of the sun yields a very high dynamic range, that is visually characteristic for its glints, highlights and but challenging to render[1].

Just for purposes for the qualitative illustration of the severity of the challenge, let us look some photometric considerations. The pixel measurement is an integration over radiance L arriving at the sensor area multiplied with the sensor response integrated over area, time and wavelength. Considering, the conversion to luminance

$$L_v(\lambda) = \frac{1}{\int_\Lambda \bar{y}(\lambda)\mathrm{d}\lambda} \int_\Lambda L(\lambda)\bar{y}(\lambda)\,\mathrm{d}\lambda \tag{3.1}$$

the pixel value Y at sensor position p^{img} is given by Luca Fascione [2020]:

$$Y(p^{\text{img}}) \approx \frac{\pi \Delta t\, S}{CN^2} L_v, \tag{3.2}$$

where N is the *f-number*, Δt the shutter opening, S the ISO value and C the camera calibration constant (here 312.5).

[1] Some would say a very "impolite thing to throw at a pathtracer".

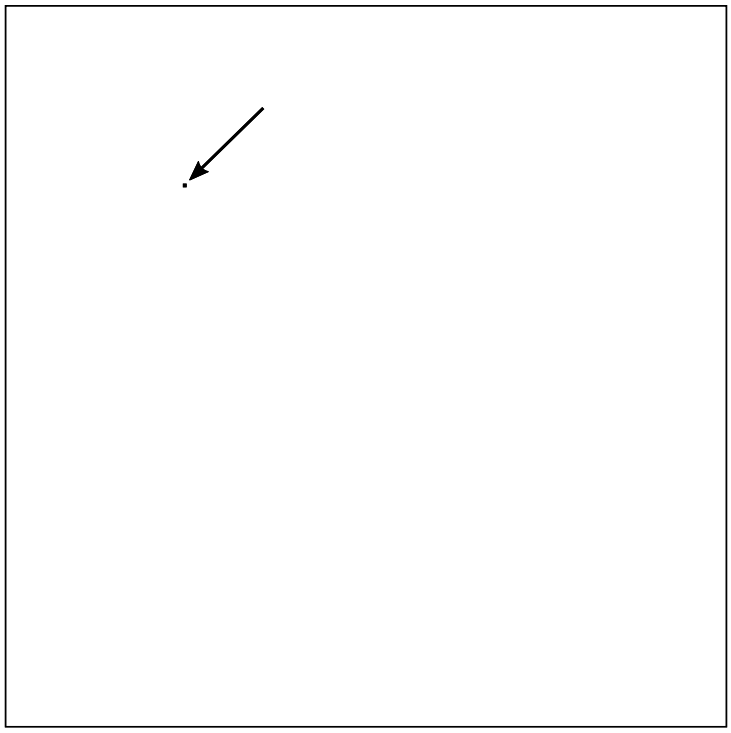


Figure 3.4: *Coverage of κ^{-1} sufficient within pixel square to saturate response.*

This means, that if the scene is exposed to the *camera exposure equation*

$$E_v = C\frac{N^2}{\Delta t\, S}. \tag{3.3}$$

(where E_v corresponds to the illuminance in lx), a diffuse object of albedo k_d yields a pixel value of k_d, since the luminance towards the camera is

$$L_v = \frac{k_d}{\pi}E_v. \tag{3.4}$$

Consider the object being lit by sun at the zenith, the illuminance is

$$E_v = \int_{\Omega^{\text{sun}}} L_v^{\text{sun}}(\omega)\,\mathrm{d}\omega^{\perp} \approx |\Omega^{\text{sun}}|L_v^{\text{sun}}. \tag{3.5}$$

Hence, the reflected luminance L_v from sun luminance L_v^{sun} is:

$$L_v = \frac{k_d}{\pi}|\Omega^{\text{sun}}|L_v^{\text{sun}}. \tag{3.6}$$

The luminance reflected of a diffuse object of albedo $k_d = 0.18$ being exposed under sunlight, the luminance looking at the sun directly is about $\kappa := 256893$ times higher, a whooping 18 stops. The consequences of this are quite evident when looking at bright highlights in a pixel. This makes any random brute-force sampling scheme clearly intractable.

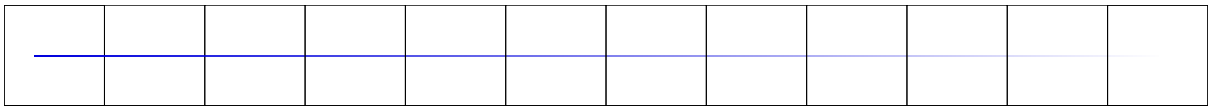


Figure 3.5: *Direct highlight of larger diameter averaging over shutter response to saturate all pixels.*

- Direct highlights only need to cover a tiny area, about $\frac{1}{\kappa F}$ of the pixel to give a significant contribution. (see Fig. 3.4)
- Sun intensity survives extremely short overlap of the shutter interval Δt, i.e., produces extremely long motion streaks. (see Fig. 3.5)
- Indirect, complex specular light transport produces significant intensities even for long chains of low Fresnel throughput. Classic forward path-tracing through a pure dielectric like water typically applies importance sampling of the reflection vs. refraction based on the Fresnel term. As a simple example, consider such probabilities to be 0.1 vs 0.9. A path following the unlikely events of probability 0.1 five times in a run and connecting to the sun, will still saturate the pixel response.

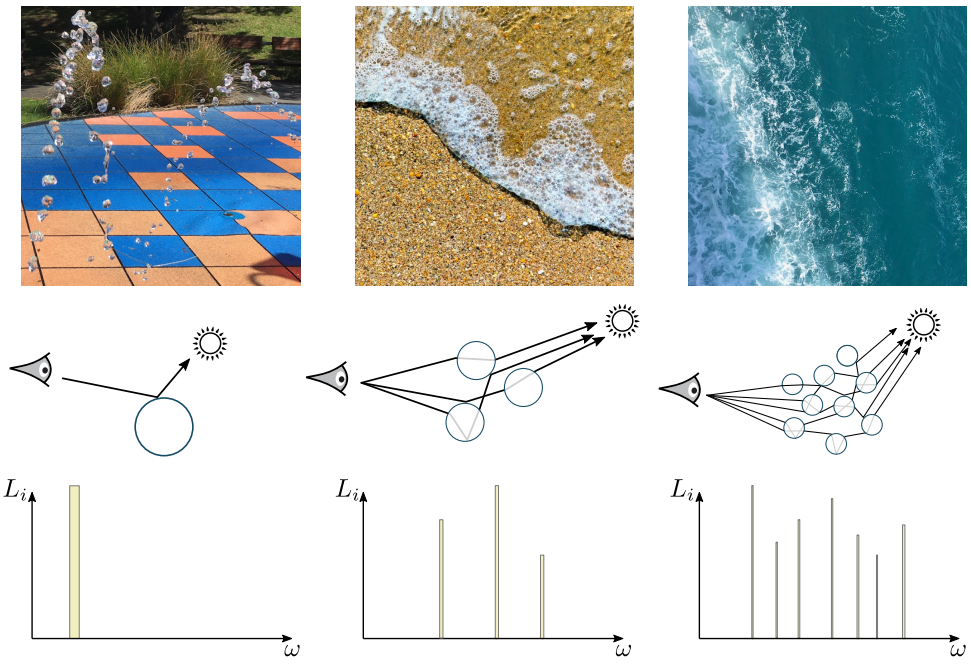


Figure 3.6: *Highlights need to be handled across a continuum of different scales. The bottom illustrates the distribution of radiance from the observer. High-throughput contributions very sparse in high-dimensional pathspace and become increasingly disconnected as feature scale decreases.*

Figure 3.6 show examples at different scales, to illustrate how the characteristics of light-transport changes, just focusing on the highlights case. In the far-field, the high-order multiple scattering becomes amenable to appearance filtering techniques, whereas in the near-field more geometric techniques may be used to render highlights more efficiently. There is a difficult intermediate regime, in which neither of these are applicable or effective.

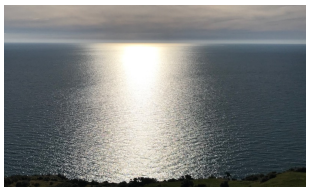
A similar pattern occurs for rendering caustics: for shallow caustics there is distinct prominent incoming radiance direction, whereas objects receive light from more and more surface locations the deeper it is submerged, resulting in a more chaotic patterns. After a certain depth the pattern becomes so chaotic that it becomes forgiving to approximate, however again there is a challenging intermediate regime.

In more general terms, complex indirect specular light-transport and manifests in various different forms. In the following sections the course discusses these in more detail, as well as effective solution strategies, in the form sampling techniques, regularisation, strategic approximations and LOD schemes.

To not suffer from extreme variance, we thus need to accommodate a wide range of different scenarios effectively. Since feature size (and thus local curvature) of the geometry varies continuously across all scales, and there is very little absorption, the types of high-throughput paths that need to handled effectively vary from local, chaotic multiple scattering to non-local, structured highlights or caustics.

3.4 Solution toolbox

3.4.1 Surface appearance filtering



In the farfield regime of the sea surface, where the pixel footprint covers a large enough area of the ocean, geometric detail can be represented by normal distribution functions and employ microfacet theory. LEAN mapping Olano and Baker [2010] builds a mipmap data structure for storing filterable moments that can be efficiently queried for a given footprint to aggregate the statistics into surface roughness. Dupuy et al. [2013] maps to an anisotropic Beckmann distribution including Smith shadowing-masking term. At the finest level, the initial distributions to be aggregated can be extracted from a displacement map or derived from FFT models directly Tessendorf [2001].

This conveniently converts a complex geometric aliasing problem into a closed form that can be evaluated analytically avoiding brute-force sampling and a more compact geometric representation. We refer to Bruneton and Neyret [2012] for a survey of appearance filtering methods (cf. also Bruneton et al. [2010]).

3.4.2 Geometric antialiasing

In the near-field there is a lower frequency variation of the geometry and thus a smaller number of individual highlights need to be resolved, remaining still very hard to find due to their small coverage within the pixel. Instead of deriving roughness via statistics, the differential geometry provides the variation of the normal to be used for mapping onto Beckmann or GGX roughness that can be combined with the base roughness of the surface Kaplanyan et al. [2016], Tokuyoshi and Kaplanyan [2019] and is therefore complementary to LEADR mapping.

Analogously to using the spatial normal variation, the temporal variation can be used to map to an anisotropic temporally regularised NDF to help resolving highlights in motion more efficiently Tessari et al. [2020].

3.4.3 Glints

Glints are characterized by nearly perfect or perfect specular reflections from small surface structures that exhibit high view-dependence. While crucial for achieving realism, these glints often pose challenges for path tracers as they can lead to flickering issues due to their low probability of being sampled. Efficient rendering of glints was, amongst others, addressed in Yan et al. [2016, 2014] Chermain et al. [2019, 2020] and Deng et al. [2022].

3.4.4 Aggregation of multiple scattering

Long paths caused by multiple scattering can be effectively approximated using various methods. One such approach involves approximating the volumetric scattering of granular media, such as sand or spheres Müller et al. [2016] Meng et al. [2015] Moon et al. [2007] or aggregation of volumes Kallweit et al. [2017]. By considering the scattering properties of these media, the interaction of light with the particles can be simulated, capturing the multiple scattering effects and enhancing the realism of the rendered scene. Another technique involves simulating multiple bounces within microfacets by implicitly solving them within the BSDF instead of tracing geometry Heitz et al. [2016] Turquin [2018]

3.4.5 Path space regularization

The perfectly specular nature of water makes next-event estimation almost impossible while caustic paths lead to long render times. Path space regularization introduces roughness in BSDFs to improve the convergence of Monte Carlo integration methods, therefore trading blurriness of reflection on e.g. indirect bounces or motion blurred features for improved convergence Weier et al. [2021] Kaplanyan and Dachsbacher [2013].

3.4.6 Handling fireflies

In path tracing, fireflies refer to bright, noisy pixels that appear in the rendered image. These pixels are typically caused by rare or extreme lighting situations, such as highly reflective or refractive surfaces, strong light sources, or complex indirect lighting interactions. Fireflies can be problematic as they can significantly degrade the visual quality of the rendered image and increase the rendering time. As such, firefly filters are essential in retaining image quality Zirr et al. [2018] Wang et al. [2020].

3.4.7 Specular manifolds

Specular sampling strategies are discussed in Zeltner et al. [2020] who systematically sampling paths within the specular manifold, thus accurately capturing the intricate and visually important high-frequency caustics and glints caused by specular surfaces. Loubet et al. [2020] builds upon an effective parametrization to construct a sampling data structure to handle specular and glossy light transport over one vertex for specular next event estimation.

3.5 Architectural considerations

One key challenge is the requirement for versatility and modularity of the rendering pipeline that results from the above. Manuka, Wētā Digital's in-house production renderer (Fascione et al. [2018]) has advanced very signficantly in many ways to cope with the complexity of rendering *Avatar: The Way of Water*.

Yet, the rendering challenges are of course not only purely addressed in to the rendering architecture, but includes also the hand-off of the scene data from upstream such as the simulation engine Loki (Lesser et al. [2022]) to support using an effective combination of primitive types and materials and providing useful controls to the users to set up their renders efficiently. Ideally, we are striving towards a system that requires as few special-case solutions as possible, to keep the amount of manual intervention small and avoid conflicting situations in which one key component is incompatible with another.

To give a glance on the upcoming course material, let us give a few specific examples of some of the architectural dependencies that we need to keep in mind, as we proceed.

- Light-tracing is a highly effective sampling technique for generating directly visible caustics and godrays. Bidirectional techniques are however more of a special-case in modern production renderers, if supported at all, since they are incompatible with common benefits of path-tracer, such as adaptive sampling, and effective regularisation techniques that rely on the path-prefix, such as ray-differentials

(cf. the sections on caustics and FX elements). Furthermore, gathering of deep samples for the compositing worksflows traditionally relies tracing paths from the camera.

- Bidirectional path-tracing does not handle pure *specular-diffuse-specular* (SDS), such as reflected caustics. Photon mapping is somewhat compatible with the traditional path-tracing advantages, yet it is known to have issues on finely detailed geometry, like corals and hair.
- Path-guiding has become a key ingredient to boost unidirectional path-tracing performance, especially with the interplay of path-regularisation approaches. How far does it bring us towards potentially removing the reliance on bidirectional techniques and photon mapping and side-step their caveats completely?
- The local differential geometry, in particular the curvature tensor, is an essential quanitity used for regularising specular highlights adaptively. Ray tracing against primitives that provide accurate derivatives on detailed geometry is therefore an important feature to be supported by the engine.
- Accurate derivatives are also essential for *Manifold Next-Event estimation* Hanika et al. [2015] (MNEE). This technique is able to connect to a light source through a transmissive specular interface and is well-suited for shallow caustics with moderate frequency content. *Specular Manifold Sampling* Zeltner et al. [2020] generalised MNEE to support multiple overlapping caustics and can even support reflective caustics. They can be very efficient and embed seamlessly into a purely forward path-tracing framework. On the flip-side deeper caustics and high-frequency detail on the water surface are prone to high cost and sensitive to numerical convergence issues.

References

Eric Bruneton and Fabrice Neyret. 2012. A Survey of Nonlinear Prefiltering Methods for Efficient and Accurate Surface Shading. *IEEE Transactions on Visualization and Computer Graphics* 18 (2012), 242–260.

Eric Bruneton, Fabrice Neyret, and Nicolas Holzschuch. 2010. Real-time Realistic Ocean Lighting using Seamless Transitions from Geometry to BRDF. *Computer Graphics Forum* 29, 2 (May 2010), 487–496. https://doi.org/10.1111/j.1467-8659.2009.01618.x EUROGRAPHICS 2010 (full paper) - Session Rendering I.

Xavier Chermain, Frédéric Claux, and Stéphane Mérillou. 2019. Glint Rendering based on a Multiple-Scattering Patch BRDF. *Computer Graphics Forum* 38, 4 (2019), 27–37. https://doi.org/10.1111/cgf.13767 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13767

X. Chermain, B. Sauvage, J.-M. Dischler, and C. Dachsbacher. 2020. Procedural Physically based BRDF for Real-Time Rendering of Glints. *Computer Graphics Forum* 39, 7 (2020), 243–253. https://doi.org/10.1111/cgf.14141 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14141

Hong Deng, Yang Liu, Beibei Wang, Jian Yang, Lei Ma, Nicolas Holzschuch, and Ling-Qi Yan. 2022. Constant-Cost Spatio-Angular Prefiltering of Glinty Appearance Using Tensor Decomposition. *ACM Transactions on Graphics* 41, 2 (2022), 22:1–22:17.

Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. 2013. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), Article No. 211. https://doi.org/10.1145/2508363.2508422

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A batch-shading architecture for spectral path tracing in movie production. *Transactions on Graphics* 37, 3 (2018). https://doi.org/10.1145/3182161

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34, 4 (jun 2015), 87–97. https://doi.org/10.1111/cgf.12681

Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. 2016. Multiple-Scattering Microfacet BSDFs with the Smith Model. *ACM Trans. Graph.* 35, 4, Article 58 (jul 2016), 14 pages. https://doi.org/10.1145/2897824.2925943

Simon Kallweit, Thomas Müller, Brian McWilliams, Markus Gross, and Jan Novák. 2017. Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Trans. Graph. (Proc. of Siggraph Asia)* tbd, tbd, Article tbd (Nov. 2017), 11 pages. https://doi.org/10.1145/3130800.3130880

Anton Kaplanyan and Carsten Dachsbacher. 2013. Path Space Regularization for Holistic and Robust Light Transport. *Comput. Graph. Forum* 32, 2 (2013), 63–72. https://doi.org/10.1111/cgf.12026

Anton S. Kaplanyan, Stephen Hill, Anjul Patney, and Aaron Lefohn. 2016. Filtering Distributions of Normals for Shading Antialiasing. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*, Ulf Assarsson and Warren Hunt (Eds.). The Eurographics Association. https://doi.org/10.2312/hpg.20161201

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: A Unified Multiphysics Simulation Framework for Production. *ACM Trans. Graph.* 41, 4, Article 50 (jul 2022), 20 pages. https://doi.org/10.1145/3528223.3530058

Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob. 2020. Slope-Space Integrals for Specular Next Event Estimation. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39, 6 (Dec. 2020). https://doi.org/0.1145/3414685.3417811

et al. Luca Fascione. 2020. *Physlight*. Wētā Digital. https://github.com/wetadigital/physlight.

Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-Scale Modeling and Rendering of Granular Materials. *ACM Trans. Graph.* 34, 4, Article 49 (jul 2015), 13 pages. https://doi.org/10.1145/2766949

Curtis. D. Mobley (Ed.). 2022. *The Oceanic Optics Book*. International Ocean Colour Coordinating Group (IOCCG).

Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. 2007. Rendering Discrete Random Media Using Precomputed Scattering Solutions. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. Eurographics Association, Goslar, DEU, 231–242.

Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Trans. Graph.* 35, 6, Article 168 (Nov. 2016), 14 pages. https://doi.org/10.1145/2980179.2982429

Marc Olano and Dan Baker. 2010. LEAN Mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '10)*. Association for Computing Machinery, New York, NY, USA, 181–188. https://doi.org/10.1145/1730804.1730834

Lorenzo Tessari, Johannes Hanika, Carsten Dachsbacher, and Marc Droske. 2020. Temporal Normal Distribution Functions. In *Eurographics Symposium on Rendering - DL-only Track*, Carsten Dachsbacher and Matt Pharr (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20201132

Jerry Tessendorf. 2001. Simulating ocean water. In *SIGGRAPH course notes, course 47*.

Yusuke Tokuyoshi and Anton S. Kaplanyan. 2019. Improved Geometric Specular Antialiasing *(I3D '19)*. Association for Computing Machinery, New York, NY, USA, Article 8, 8 pages. https://doi.org/10.1145/3306131.3317026

Emmanuel Turquin. 2018. Practical multiple scattering compensation for microfacet models.

Beibei Wang, Miloš Hašan, and Ling-Qi Yan. 2020. Path Cuts: Efficient Rendering of Pure Specular Light Transport. *ACM Trans. Graph.* 39, 6, Article 238 (nov 2020), 12 pages. https://doi.org/10.1145/3414685.3417792

Philippe Weier, Marc Droske, Johannes Hanika, Andrea Weidlich, and Jirí Vorba. 2021. Optimised Path Space Regularisation. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.14347

Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-Normal Distributions for Efficient Rendering of Specular Microstructure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)* 35, 4 (2016).

Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33, 4 (July 2014), 116:1–116:9. https://doi.org/10.1145/2601097.2601155

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392408

Tobias Zirr, Johannes Hanika, and Carsten Dachsbacher. 2018. Reweighting firefly samples for improved finite-sample Monte Carlo estimates. *Computer Graphics Forum* 37, 6 (2018), 410–421. https://doi.org/10.1111/cgf.13335

4 Reflecting on the appearance of water

Andrea Weidlich, *Nvidia*

Achieving a high level of realism in water simulation can be a time-consuming and computationally expensive process, but the results can be stunning when done correctly. During the production of *Avatar: The Way of Water*, water was a constant feature in almost every shot we rendered; characters and objects would constantly transition from dry to wet and back again, sometimes being fully submerged and sometimes only partially covered with water. From a technical perspective, this presented a range of challenges. Light transport had to be efficient enough to render water and wetness in various forms while the material system needed to be flexible enough to be realistic while being expressive enough to meet the artistic requirements of the project. A critical factor in achieving a consistent and realistic appearance was developing a deep understanding of water and how it interacts with other materials. By leveraging this knowledge and using new technology, we were able to create stunning water renderings that brought the film to life in a truly immersive way.

4.1 Inherent and apparent optical properties of water

4.1.1 Pure water

As shown in Figure 4.1a, pure water exhibits multiple absorption bands outside of the visible spectrum, and its index of refraction (IOR) varies considerably. However, within the visible portion of the light spectrum, it has an IOR of approximately 1.33, with minimal absorption and no scattering. Between 300 and 800nm, the IOR can be approximated with [Quan and Fry, 1995]

$$n(T,\lambda) = 1.31405 - 2.02 \times 10^{-6} T^2 + \frac{15.868 - 0.00423T}{\lambda} - \frac{4382}{\lambda^2} + \frac{1.1455 \times 10^6}{\lambda^3} \tag{4.1}$$

whereas T is the temperature in Celsius and λ is the wavelength in nanometers. Figure 4.1b shows that the difference in the IOR between temperatures is minimal, hence in computer graphics it is common to ignore T and even the dispersive nature of water.

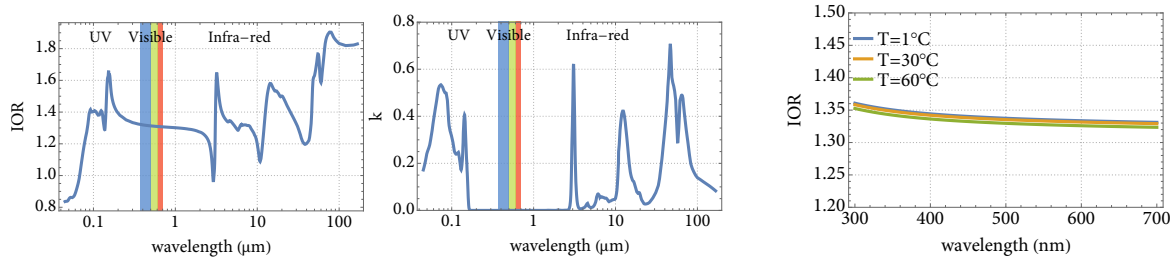


Figure 4.1: *Left: Real and imaginary IOR of ice outside the visible spectrum of light [Warren, 1984]. Note that water is only transparent in the visible part and exhibits strong absorption outside. Right: IOR of pure water for 1°C, 30°C and 60°C. The difference is only minimal.*

As a dielectric medium, the amount of reflection can be described by a classic Fresnel coefficient [Hecht, 1998] and reflections are perfectly specular due to surface tension. For macroscopic features, a Gaussian distribution is suitable to handle surface waves[Ross et al., 2005]. This makes pure water a material that is comparably straightforward to describe for the purpose of rendering.

4.1.2 Ocean water

Things become more complicated once we look at natural water which comes in a variety of colours and states of turbidity. The reason for this is that particulates are suspended in water which will cause absorption and

scattering. The most common particulates are phytoplankton, which contain chlorophyll, and Gelbstoff also known as coloured dissolved organic matter (CDOM), although minerals and detritus will also influence water colour. The absorption spectra of these various components can be found in [Mobley, 2022], highlighting the complexity of accurately rendering natural water.

To simplify classification, the *Jerlov water types* introduced by Jerlov [1976] are often used to classify natural water based on its color and optical properties. Ten different types exist: five for open ocean water (I, IA, IB, II, III) and five for coastal water (1C, 3C, 5C, 7C, 9C). whereas I is the clearest water of the ocean types which can be found in e.g. areas such as the Caribbean and 1C being the clearest among the coastal types. 9C is the most turbid found in coastal areas with high levels of sedimentation and runoff from land with higher concentrations of organic matter and suspended particles in the water. Moser [1992] provides a map of the regional distribution of water types. Solonenko and Mobley [2015] have fitted absorption and scattering parameters for each water type, as shown in Figure 4.2. Water types can be mixed together which allows to build a wider gamut of possible water colours.

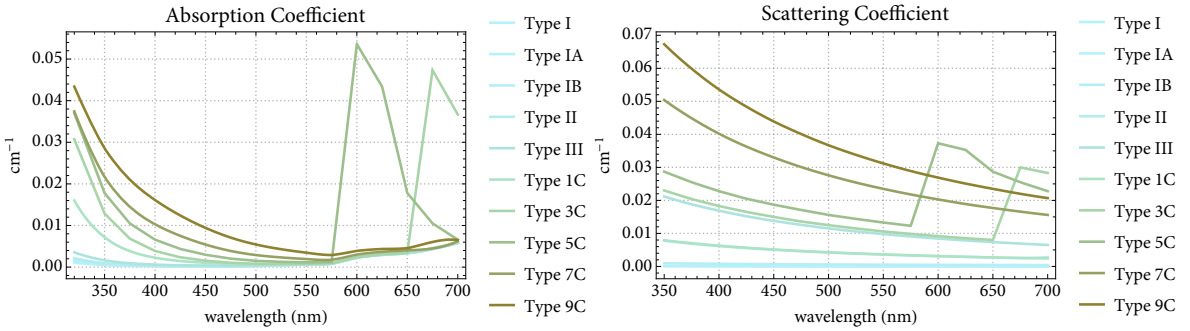


Figure 4.2: *Absorption and scattering coefficients of all Jerlov water types in cm. Data taken from Solonenko and Mobley [2015]. For convenience, we provide the corresponding sRGB values in Table 1.*

It is important to note that absorption and scattering coefficient derived by Solonenko and Mobley [2015] are inherently tied to the scattering distribution, which is characterised by its phase function. The phase function most commonly associated with ocean water is the Fournier-Forand phase function [Fournier and Forand, 1994] which approximates the scattering of particles with a Junge particle size distribution. Another approach is to express the phase function as the weighted sum of a molecular phase function $ph_M(\theta)$ and and the particle phase function $ph_P(\theta)$.

$$ph(\theta) = ph_M(\theta) * w + ph_P(\theta) * (1 - w) \tag{4.2}$$

Table 1 lists the molecular weight w for each water type. Morel and Loisel [1998] established that the asymmetric parameter of the particle phase function g is 0.924[2]. The molecular phase function is symmetric, i.e. g is 0.0.

Interestingly, the clearer the water is, the more isotropic scattering the phase function becomes. This might seem paradoxical, but it can be explained by examining the size and amount of the particles present in the water. Clear water typically contains smaller particles, which leads to the dominance of Rayleigh scattering. In contrast, more turbid water tends to have larger particles, which are more likely to exhibit Mie scattering. However, the overall particle count in clear water is much smaller, hence light is less likely to scatter.

4.1.3 Practical considerations in Avatar: The Way of Water

In order to create a realistic look and feel for *Avatar: The Way of Water*, we heavily relied on the Jerlov water types. Due to the multiple bounces and long paths in water, any deviation in the spectral curve into one of

[2]While using a single Henyey-Greenstein phase function with $g = 0.924$ for the particle phase function may be acceptable, it is not optimal. A better approximation can be achieved by using two Henyey-Greenstein phase functions, the first one strongly forward scattering, while the second one is backscattering. Together they should have a mean cosine of 0.942.

Table 1: *Parameters of Jerlov water types [Solonenko and Mobley, 2015]. Absorption coefficient and scattering coefficient are given in cm and were converted to sRGB assuming CIE1931 standard observer.*

Water type	w	Absorption coefficient (cm)	Scattering coefficient (cm)
I	0.93	(0.00309, 0.00053, 0.00009)	(0.00001, 0.00002, 0.00004)
IA	0.44	(0.00309, 0.00054, 0.00014)	(0.00002, 0.00004, 0.00007)
IB	0.06	(0.00309, 0.00054, 0.00015)	(0.00045, 0.00054, 0.0007)
II	0.007	(0.0031, 0.00054, 0.00016)	(0.0027, 0.00365, 0.00516)
III	0.003	(0.0031, 0.00056, 0.00031)	(0.00737, 0.00998, 0.01413)
1C	0.005	(0.00316, 0.00067, 0.00105)	(0.00274, 0.00372, 0.00526)
3C	0.003	(0.00508, 0.00052, 0.00161)	(0.00904, 0.01071, 0.01532)
5C	0.001	(0.04638, 0.00222, 0.00216)	(0.03589, 0.01382, 0.01857)
7C	< 0.001	(0.00351, 0.00188, 0.00574)	(0.01772, 0.02394, 0.03376)
9C	< 0.001	(0.00398, 0.00349, 0.00995)	(0.02347 0.0318, 0.04496)

its metameric version becomes immediately noticeable. As a result, we decided to use tabulated curves of the Jerlov types instead of the Smits-based uplifting process that we normally use in Manuka [Weidlich et al., 2022]. Since the water was a homogeneous volume, using tabulated curves did not impose any noticeable memory overhead regardless of the table resolution.

Figure 4.3 shows the difference between spectral tables, Smits uplifting and RGB accumulation for the different Jerlov types. While there is little change in the beginning, the error starts to accumulate with successive path length.

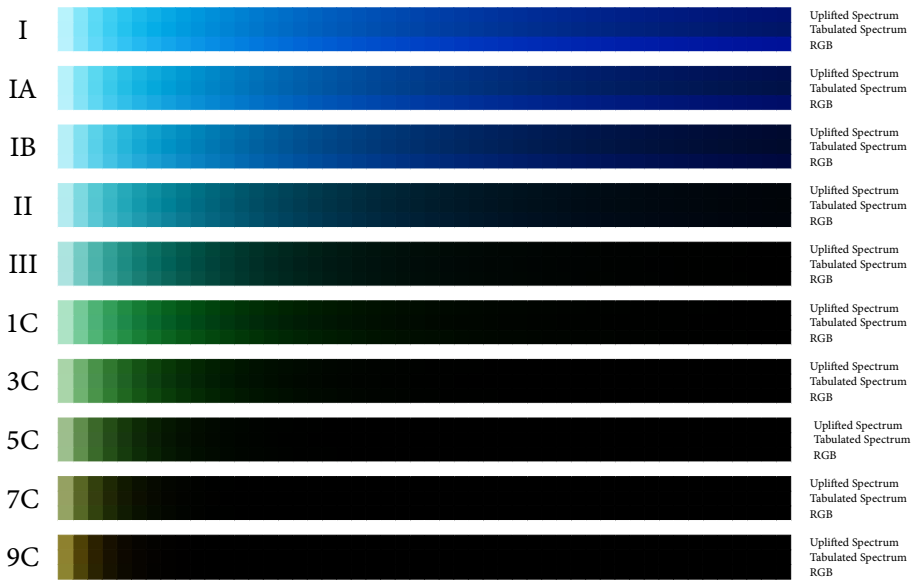


Figure 4.3: *Comparison of Jerlov water types with successive path length and different colour representations. Each box corresponds to 1m. Note that there is a visible difference between 5m and 20m between Smits uplifting and tabulated values whereas the difference in using RGB instead of spectral accumulation becomes more prominent the longer the paths get.*

Different types could be mixed together by mixing several homogeneous volumes. If further colour adjustments were necessary, additional volumes of arbitrary colour could be mixed in. We primarily used water types I and 1C with further tweaks done when necessary.

We further utilised the spectral rendering of Manuka by working with camera filters. As much of the color is absorbed by the water before it reaches the camera, the resulting colours appear desaturated. To account

for this this, we simulated spectral red filters commonly used in underwater photography. Since absorption of water is weakest in the blue and strongest in the red, these filters enhance the absorbed wavelengths and reduce the blue light. We applied the filters before splatting the spectral data to work directly on the spectral information.

For the scattering, we decided against using the Fournier-Forand phase function and approximated it with two Henyey-Greenstein phase functions. This decision was made because the Fournier-Forand phase function lacked perfect importance sampling and did not easily support light transport optimisation algorithms such as path space regularisation [Weier et al., 2021]. Furthermore, parameter editing of the Fournier-Forand phase function is less intuitive than for the Henyey-Greenstein phase functions.

4.2 Objects and water

Rendering water on its own can be a challenging task, but it becomes even more complicated when objects are embedded within it. This is because the appearance of objects can change depending on whether they are surrounded by water or not, This change can sometimes happen in an unpredictable or unintuitive way as illustrated in Figure 4.4.

4.2.1 Volumetric vs thin water

Generally speaking, there are two scenarios we are interested in:

- Objects that are embedded in a large body of water. Their relative IOR will be reduced by the surrounding medium which results in a reduced reflectance and overall duller appearance when rendering from within the water. Additionally, there will be a change in refraction. Although there will be a slight colour shift on subsurface scattering objects, the main colour shift can be attributed to the absorption of the water.
- Objects that are only covered by a thin layer of water, generally referred as wetness. Objects appear to become more specular and will change their colour. This typically results in an increase in saturation and/or darkness compared to their dry counterparts.

From a technical standpoint, both scenarios share a common requirement: a material system that can communicate between materials and materials and surrounding media how their appearance has to change. This involves tracking the volume [Schmidt and Budge, 2002] and ensuring that BSDFs understand the concept of IORs. Throughout the remainder of the course, we will concentrate on the more complicated case of thin water layers, i.e. handling surface wetness.

Figure 4.4: *Photographs of objects reacting to water. Some materials will look duller, others will look more specular or stay the same. Materials might become darker or not.*

4.2.2 Why things get darker when wet - or not

As previously noted, surface wetness is primarily characterised by an increase in specularity, saturation, and a decrease in albedo. The increase in specularity is due to water forming a thin layer on top of the surface, which

is smoother than the base material, resulting in more specular reflections. The change in color can have two causes. First, when light is trapped between the water layer and the base material, it increases the likelihood of light absorption. Second, when water enters a material, it can alter its scattering behaviour by filling small cavities caused by porosity, resulting in more forward scattering.

It is worth noting that less light arriving at the base layer due to reflection on the top layer is not a cause of darkness since the reflection of the water layer does not reduce the amount of reflected energy. It will, however, reflect light into a different direction which will result in a different visual appearance in the presence of small light sources. When considering solid objects and disregarding porosity, the degree of darkening caused by wetness depends on three factors:

- The *albedo* of the base material. Absorption will lead to energy loss after each bounce. Materials with base albedo of close to zero or one will not change their appearance as much as mid-tones since either all the energy is absorbed after the first bounce or no energy is absorbed at all (see Figure 4.5).
- The *IOR* of the top land base ayer. The higher the IOR of the top layer, the more likely it is that light gets trapped between top and base layer, the longer the paths will become. In the presence of absorption this will result in materials becoming darker (see Figure 4.5 and Figure 4.6).
- The *roughness* of top and base material. Rough base materials will scatter light away from the perfect reflection direction will consequently will be affected more often by total internal reflection. Therefore rougher materials will become darker than smooth materials (see Figure 4.6)).

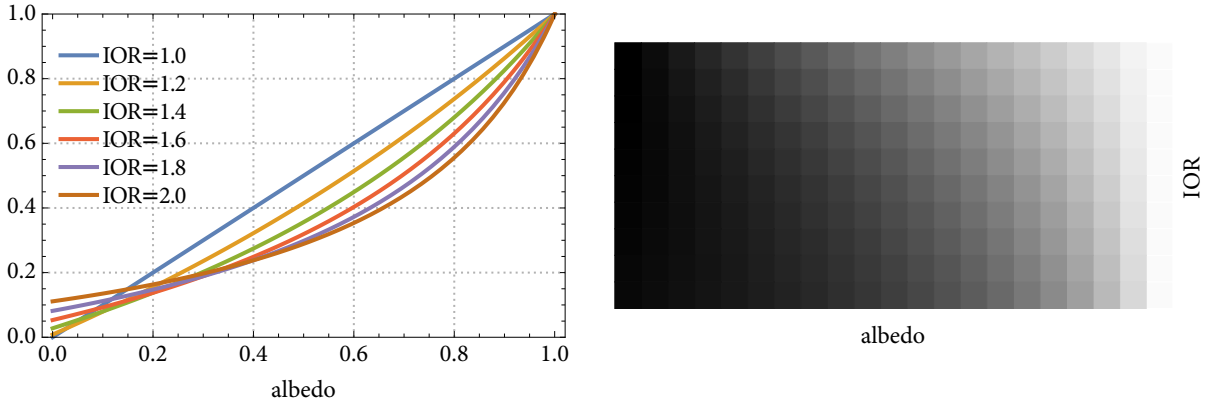


Figure 4.5: *Change in albedo of a diffuse base material with perfectly specular top layer with increasing IOR. Note how non-linear the darkening becomes.*

Note that not all materials will become darker or don't change wet. Subsurface scattering materials like e.g. skin can even become brighter when covered with a layer of water. This is because – similar to the reflecting case – light will be trapped between top and base layer. However, in the case of subsurface scattering, this means that due to the change in relative IOR of the base material, light has a higher chance of leaving the material resulting in shorter paths within the medium and instead bounces between base and top where it will not be affected by absorption.

4.3 Shaded water++

In an ideal scenario, simulating water as geometry rather than using shaders to indicate wetness on surfaces would be preferred. Practical reasons prevent us from doing this; simulating thin water layers would be time and memory consuming, animation would be difficult, the result would be prone to numerical artefacts and light transport time would suffer since we cannot perform next-event estimation. Hence the preferred method of creating wetness is still by shading it.

Our first attempt to tackle the problem of deriving a robust solution of shaded material-water interaction was to implement a specialised diffuse BSDF which would understand the concept of IORs and wetness. It

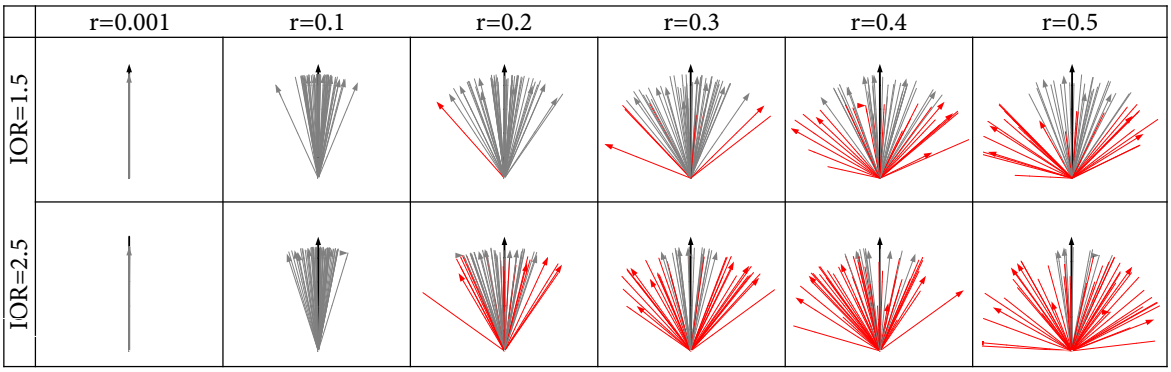


Figure 4.6: *Hemispherical reflection directions of various surface roughnesses r ranging from 0.001 to 0.5 with normal incidence direction. Assuming a smooth top layer, the red rays indicate which rays would be affected by total internal reflection and completely reflected back by the top layer without being able to exit. The higher the IOR of the top layer, the fewer rays can exit. With increasing surface roughness, more and more rays get blocked.*

was specifically designed to efficiently simulate the look of granular media such as sand from dry to wet. For this, we drew on the theories of Hapke [2002] and Lommel-Seeliger [Fairbairn, 2005] and created a material that would change its internal parameters based on the level of wetness. Instead of directly darkening the albedo and leaving the reflection properties untouched, we incorporated changes to the scattering behaviour of the material. Specifically, we used a double Henyey-Greenstein phase function to describe the scattering and made it more forward scattering as the material became wetter. By doing so, we were able to simulate the same amount of darkness that a diffuse material covered by a smooth layer of water would have.

While this approach worked well for sand, we realised that it would be necessary to extend the concept to other BSDFs in order to allow artists to make use of our full set of BSDFs. We found that not all BSDFs are capable of handling IORs. For instance, Lambert BRDF and Oren-Nayar BRDF [Oren and Nayar, 1994], which are commonly used to describe diffuse reflection, have no way to change their color when embedded in a water layer. To enable the use of our standard set of BSDFs, we decided to implement the wetness directly into the layering system.

4.3.1 Layering in a production renderer

Manuka's pre-shading architecture has several advantages and disadvantages compared to other production renderers. One advantage is that it can avoid expensive texture reads by not evaluating shaders on a per-hit basis. This can prevent bottlenecks and allow shading networks to be larger and more detailed, with dozens of texture reads per BSDF, resulting in high visual complexity. However, this unique design decision also has its drawbacks. For instance, the inputs to the shading network must be completely view-independent, and artists cannot access light transport information to drive shaders. As a result, all necessary computations must be done inside the material system.

Manuka's material system is based on BSDFs that can be combined in an arbitrary fashion through the layering system. Several layering operators exist:

- *Mix.* Mixing mixes two or more materials in a view-independent fashion together. This is a horizontal layering operation, i.e. the stack height does not increase.
- *Coat.* Coating is a vertical layering operation where one material is put on top of another and the base material is attenuated with the transmission energy.
- *Blend.* Blending is again a horizontal layering operation and is similar to mixing except that we mix only two materials and the weights are inverted.
- *Add.* Adding is a vertical operation and will add the contributions of two materials together. The operation is restricted to EDFs.

To simulate wetness in our renderer, we extended our layering system by introducing a new layering operator we called *thickcoat*. This operator simulates successive bounces between layers in a closed form without actually tracing rays. However, we made sure not to replace the traditional coat operation, which we now refer to as *thincoat*. The thincoat operation remains more frequently used due to its simplicity and the artistic freedom it brings.

4.3.2 Thickcoat operation

The foundation of the thickcoat operator is a layering system that can handle IOR and roughness accumulation, enabling materials to change their IOR depending on the layer order and giving BSDFs the freedom to change their shape based on previous interactions. For this, we need to be able to traverse layer stacks up and down. We refer to Fascione et al. [2018] for further discussions.

All BSDFs within a thickcoat operation are unified to a single material response, each one building a so-called *thickcoat block*. Similar to roughness and IOR accumulation, BSFDs combined with horizontal operations or with weights < 1.0 are stochastically split so that only a linear stack of vertical operators remains after decimation within each block. It is important to note that BSDFs are not disabled during light transport, but only during parameter initialisation and thickcoat computations to avoid a multiplication of BSDFs, thus ensuring that the number of BSDFs stays constant. Once the stack is reduced to a single stack of blocks layered vertically on top of each other, we calculate the necessary parameters like roughness, IOR and albedo of each block. Computations would only happen between such blocks which reduces the complexity of the problem. An illustration of the process can be seen in Figure 4.7.

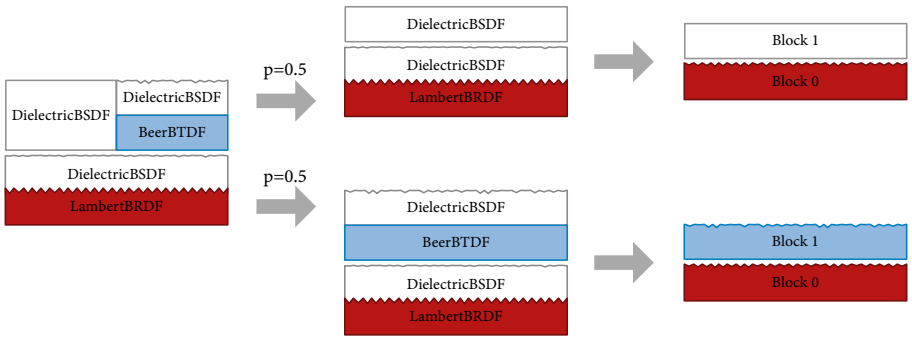


Figure 4.7: *Schematic illustration of the thickcoat operator. We will first build a probability p to flatten the stack based on the weights of the BSDFs and layers and then run the layer program on each subnode of the tree. The result will be stored as a thickcoat block. Computations will only happen between blocks.*

We then calculate the energy that gets absorbed for reflection rays and produced on transmission rays. For reflection rays, we will multiply the amount into the single scattering result. For transmission, we will add the result. An example of how the albedo changes for a reflection block can be seen in Figure Figure 4.8.

Calculating bounces between two layers theoretically requires an infinite series to compute. However, we observed that after a certain number of bounces light reflectance will become uniform enough to assume constant roughness for successive bounces which allows us to split the computation in two parts: a finite series for the first $n - 1$ bounces with varying roughness and a closed form solution for the remaining n bounces assuming constant behaviour. In fact, the first series was later removed, and only the closed-form solution with a fixed roughness was used, as the introduced error was mainly visible at grazing angles and not strong enough to warrant slower performance.

It is important to note that there are always trade-offs when choosing a specific method or approach for a particular problem. While a more advanced approach like the position-free layering from Guo et al. [2018] may have been more accurate, it would have introduced additional noise and a performance overhead that was not feasible for our specific use case. Similarly, the method introduced by Belcour [2018] was too inflexible for the wide range of BSDFs used in Manuka. In the end, we chose a layering system that was flexible enough to

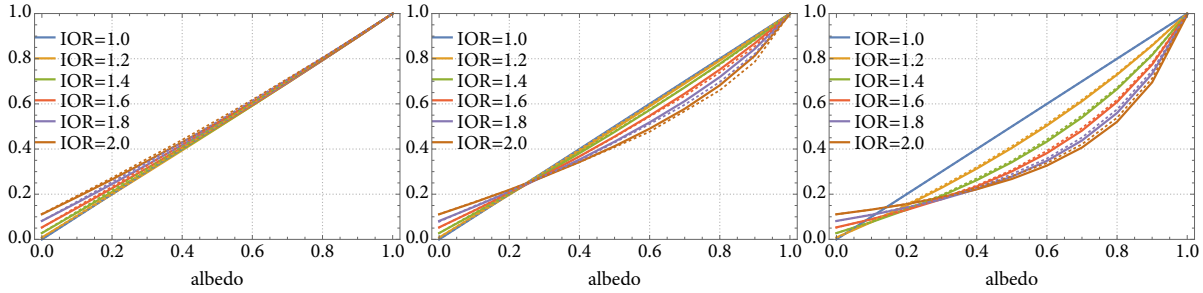


Figure 4.8: *Estimated albedo (solid line) vs. Monte Carlo ground truth (dashed line) for different IORs and albedos at normal incidence. From left to right: Base layer roughness r=0.001, r=0.25, r=0.5. The top layer is smooth.*

support all of our BSDFs while also being efficient and robust enough to handle the demands of production rendering.

4.4 A water-compliant material

Our original objective was to develop a layering node which could simulate multiple bounces within a shaded layer in a closed form to efficiently match the look of geometric and shaded water to seamlessly blend between both. However, we encountered a new problem during production when we noticed that some diffuse materials became darker than anticipated below geometric water due to how diffuse reflection causes total internal reflection within the layer. While this was correct from a light transport perspective, it was nevertheless undesired.

To address this issue, we decided to artificially brighten surfaces under the geometric water. Since this needed to happen in an automated and consistent way, we developed a mechanism called water-compliancy. In air, *thickcoat* would introduce darkening when used to simulate shaded water. But this is not the case below geometric water. Here, the IOR of the surrounding geometric water and the shaded water would index-match, and no multiple bounces would be simulated since all the light could exit immediately. For the water-compliant material, we inverted the process and brightened materials below geometric water and kept the thincoat appearance of the base material below shaded water.

On all materials, we put an additional, final single thickcoated dielectric top layer with an IOR of 1.0. A flag indicated whether water-compliancy was enabled, and if so, we ran the code twice: If enabled, the code was run twice: once with the outside IOR set to the actual IOR of the surrounding medium and once with the outside IOR set to 1.0. For materials not embedded in geometric water, the two amounts cancelled each other out, and the IOR of the top layer being 1.0 resulted in it being ignored during rendering, preserving the original look of the material without darkening. However, for materials below geometric water, there would be a difference between the two amounts. We used this estimate to boost the brightness of the BSDFs to compensate for the darkness that the geometric water would cause.

4.5 Challenges in look development

The question of how two materials influence their appearance is complex, but it can be solved by carefully analysing the physics involved, as long as both materials obey physical principles. However, using empirical BSDFs like Oren-Nayar or measured BSDFs poses significant challenges that require a more elaborate framework, as we did during the production of A. Ideally, all components of a material system should be able to react to their environment without the need for additional complexity. However, as long as we simplify materials and e.g. use diffuse BSDFs to simulate very dense volumetric materials or approximate low reflectance with a dielectric material, we will always face issues once the environment changes. More attention has to be paid towards *material metamerism* – materials that only look the same in one condition but will reveal their differences in another one.

One question remains largely unanswered: how can we efficiently integrate physical processes into a pipeline without burdening artists? As seen with the water-compliant material, workflows may be based on physics, but they might need to diverge for artistic reasons. Therefore, it is crucial that artists have direct control over the output, and inverse rendering might prove to be a valuable tool in this regard. By allowing for the more efficient integration of physics into the pipeline while maintaining artistic control over the output, inverse rendering can help strike a balance between artistic creativity and physical accuracy.

References

Laurent Belcour. 2018. Efficient Rendering of Layered Materials Using an Atomic Decomposition with Statistical Operators. *ACM Trans. Graph.* 37, 4, Article 73 (jul 2018), 15 pages. https://doi.org/10.1145/3197517.3201289

Maxwell B. Fairbairn. 2005. Planetary Photometry: The Lommel-Seeliger Law. *Journal of the Royal Astronomical Society of Canada* 99, 3 (June 2005), 92.

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3, Article 31 (aug 2018), 18 pages. https://doi.org/10.1145/3182161

Georges R. Fournier and J. L. Forand. 1994. Analytic phase function for ocean water. In *Other Conferences*.

Yu Guo, Miloš Hašan, and Shaung Zhao. 2018. Position-Free Monte Carlo Simulation for Arbitrary Layered BSDFs. *ACM Trans. Graph.* 37, 6 (2018).

Bruce Hapke. 2002. Bidirectional Reflectance Spectroscopy: 5. The Coherent Backscatter Opposition Effect and Anisotropic Scattering. *Icarus* 157, 2 (2002), 523–534. https://doi.org/10.1006/icar.2002.6853

Eugen Hecht. 1998. *Optics. 3rd Edition* (3 ed.). Addison Wesley Longman, Reading, Mass.

N.G. Jerlov. 1976. *Marine Optics. 2.ed: Elsevier Oceanography Series 14*. Elsevier. https://books.google.ca/books?id=h_owngEACAAJ

Curtis D. Mobley. 2022. *The Oceanic Optics Book*. International Ocean Colour Coordinating Group (IOCCG), Dartmouth, NS, Canada.

André Morel and Hubert Loisel. 1998. Apparent Optical properties of oceanics waters : dependence on molecular scattering contribution. *Applied Optics* 37 (08 1998), 4765 – 4776. https://doi.org/10.1364/AO.37.004765

Paul M Moser. 1992. Spectral Transmission of Light through Seawater.

Michael Oren and Shree K. Nayar. 1994. Generalization of Lambert's Reflectance Model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. Association for Computing Machinery, New York, NY, USA, 239–246. https://doi.org/10.1145/192161.192213

Xiaohong Quan and Edward S. Fry. 1995. Empirical equation for the index of refraction of seawater. *Appl. Opt.* 34, 18 (Jun 1995), 3477–3480. https://doi.org/10.1364/AO.34.003477

Vincent Ross, Denis Dion, and Guy Potvin. 2005. Detailed analytical approach to the Gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface. *J. Opt. Soc. Am. A* 22, 11 (Nov 2005), 2442–2453. https://doi.org/10.1364/JOSAA.22.002442

Charles Schmidt and Brian Budge. 2002. Simple Nested Dielectrics in Ray Traced Images. *Journal of Graphics Tools* 7 (Jan 2002). https://doi.org/10.1080/10867651.2002.10487555

Michael G. Solonenko and Curtis D. Mobley. 2015. Inherent optical properties of Jerlov water types. *Appl. Opt.* 54, 17 (Jun 2015), 5392–5401. https://doi.org/10.1364/AO.54.005392

Stephen G. Warren. 1984. Optical constants of ice from the ultraviolet to the microwave. *Applied Optics* 23, 8 (1984), 1206–1225. http://ao.osa.org/abstract.cfm?URI=ao-23-8-1206

Andrea Weidlich, Chloe LeGendre, Carlos Aliaga, Christophe Hery, Jean-Marie Aubry, Jiří Vorba, Daniele Siragusano, and Richard Kirk. 2022. Practical Aspects of Spectral Data in Digital Content Production. In *ACM SIGGRAPH 2022 Courses (SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 11. https://doi.org/10.1145/3532720.3535632

Philippe Weier, Marc Droske, Johannes Hanika, Andrea Weidlich, and Jiří Vorba. 2021. Optimised Path Space Regularisation. *Computer Graphics Forum* 40, 4 (2021), 139–151. https://doi.org/10.1111/cgf.14347 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14347

5 Caustics

Jiří Vorba, *Wētā Digital / Unity*

Figure 5.1: *Caustic patterns are important part of water appearance often visible in clear shallow waters under sunny lighting conditions. Image courtesy of Philippe Weier.*

5.1 Introduction

Shallow water looks significantly different when we look at it under an overcast sky or during a sunny day. The main difference lies in the shiny caustic patterns caused by sunlight which gets focused when it travels through a boundary between air and the water volume. The appearance of these patterns is influenced by the shape and frequency of the water waves, as well as the water's depth and volumetric properties. For example, caustics look sharp in shallow clear water whereas they become blurry in murky water or completely invisible in deep or very dirty water. Thus, caustics are a natural and important part of water appearance under sunny lighting conditions and it can contradict our experience if this effect is missing in our renders.

For rendering the many water sequences in *Avatar: The Way of Water*, we decided to leverage physically-based ray-tracing and provide artists with a set of rendering tools to simulate caustic light transport accurately and efficiently. It is worth noting that caustics are not a result of material evaluation under direct lighting but are rather part of the global illumination as they appear on other objects in the scene but not on the water surface directly. This is one of the reasons why rendering plausible caustics has always been considered challenging.

In contrast to precise light transport simulation, another commonly used approach in VFX and architecture visualization is to render so called "fake" caustics as a projection of a caustic texture from a directional light source. The advantage of this method is definitely speed in terms of render time, however, it is not easy to setup as it requires high quality animated textures and needs to be adjusted by artists for each rendered sequence carefully. In addition, we have encountered a number of sequences where we could clearly see animated water surface and the corresponding caustic on a receiver object at the same time. In such cases, mismatch between "fake" caustics and the water simulation can be distracting and thus breaking illusion of realism. Another problem with this method is that it does not respect natural blur of caustics with increased depth. Typical water waves consists of multiple frequencies and thus refracted light travels in various directions. As a result,

caustics due to each water wave frequency are sharpest at different depths and become de-focused at varying rate with increased depth. Mimicking this behavior by "fake" caustics in scenes where terrain spans over multiple depths is hard while this effect is achieved naturally by physically-based light transport simulation.

In this course, we discuss multiple options that we offer for efficiently rendering caustics based on physically-based simulation of light transport. We discuss challenges posed by very complex scenes with respect to materials, scale and geometric details as well as pros and cons of proposed methods and considerations of their integration within existing production rendering system. It is important to note that we take advantage of having very detailed and accurate water simulations. Without such a high quality input, light transport simulation could not provide highly realistic caustics. We argue that the simulation based approach presented in this course can save precious time of artists compared to the "fake" projection based approach as simulated caustics are computed automatically as part of the global illumination without the need of additional setup.

5.2 Movie shots diversity

Numerous rendering methods have been proposed in the light transport literature for rendering caustics. However, it was not straightforward to decide upfront which specific light transport algorithm to chose. Our main criteria was to render caustics efficiently without producing objectionable artefacts and to integrate them easily with our existing light transport simulation code.

The main challenge for choosing the right method with respect to these criteria is the wide variety of the water shots we were facing. We needed to be able to handle scenarios when the camera is above the water, under the water, or even only half-submerged. Caustics could be received on the terrain in various water depths, on tiny geometric details of corals, on skin of creatures with simulated subsurface scattering, in hair, but also in the eyes of characters. Subsurface scattering was also present on plants both above and under the water as well as on corals. In some scenes, characters became wet in which case the thin water film on their skin and little droplets also produce caustics.

With respect to efficiency, for some methods the scale of a typical production scene can be prohibitive. We mainly aimed at outdoor scenes featuring water illuminated by sun, where the water surface can span even over many square kilometers/miles to accommodate all possible camera views. While this is usually not a problem for path tracing, it is significant limitation of naive photon mapping implementation, an algorithm normally considered well suited for rendering caustics. Without additional path guiding extensions, it would not be possible to use it.

We must not forget about so called "god-rays", a specific type of volumetric caustics due to scattering of light within the water volume, which is an important and visually interesting effect which communicates the feeling of being under the water. Specific challenges and methods for efficient light transport in water volumes are discussed in greater detail in Section 6.

To make sure we are able to handle the full variety of possible water scenes with respect to caustics, we decided to explore multiple avenues based on competing light transport methods. In the remainder of the course, we first discuss why caustics are relatively difficult problem for light transport, and then we provide details of each method that we explored. Most importantly, we discuss their pros and cons with respect to efficiency and also to production pipelines and in general, we will advocate for a method that is both efficient, accurate and capable of handling all scenes without forcing separate rendering passes. All of these methods were tested in the production and used to render different shots in the *Avatar: The Way of Water*. In all explored options, path guiding played an essential role.

5.3 Sampling challenges

In this section, we discuss challenges for sampling caustics efficiently with respect to existing Monte Carlo light transport methods. All these algorithms are based on sampling many light paths between the camera and light sources while averaging the amount of light transported along them. Their efficiency depends on their ability to sample the important paths with high probability.

Simulating caustic light transport with *forward path tracing*, the most frequently used method in production rendering systems, is difficult due to the low probability of finding such paths. For example, if we have

a camera under the water and we look at the sand receiving caustics, we need to find the sunlight refracted into the water. That is, if we approximate sand as a diffuse reflector and consider purely specular interface between water and air, we need to sample EDSL (camera-diffuse-specular-light) paths. At the sandy sea terrain, we have high freedom where to sample reflected rays due to its diffuse material, however, the refracted sun occupies only a very small solid angle. Thus, it is very unlikely to sample rays aiming towards the sunlight, and when we arrive at the purely specular water interface, we have only two directions to continue our path given by the law of reflection and Snell's law. The latter gives us the only possible direction of refracted ray and if we already sampled a wrong direction at the sand, we will miss the sun. Note that in classical forward path tracing, we cannot do anything about that at the water interface since any change of the refracted direction violating Snell's law would result into zero contribution due to the purely specular water interface. This is also the reason why next-event estimation, normally a very efficient method for sampling direct light, cannot be applied at the water surface.

If we want to increase the probability of finding the sun, we can relax our physically-based model and increase its solid angle which would result in blurry caustics. This provides desirable artistic control over the look of caustics for some shots, however, for many shots we need to simulate sharp caustics without compromises.

One option to render sharp caustics more efficiently is *light tracing* [Dutré and Willems, 1994, Veach, 1997], where path sampling starts from a light and continues incrementally towards a camera[3]. Because the last vertex on the path is explicitly connected to the camera, light tracing cannot resolve purely specular reflections and transmissions and is inefficient for near-specular materials. As a consequence, light tracing cannot be used for rendering caustics visible from outside (Fig. 5.1), only for caustics shot by a camera fully submerged in the water.

To address this limitation, researchers proposed using *photon mapping* [Hachisuka and Jensen, 2009, Jensen, 1996], which is capable of handling efficiently LS+DS+E[4] (light-specular-diffuse-specular-camera) paths. First, a path is sampled incrementally from the light following all specular reflections/refractions and a so called *photon* is stored at each position where the path interacts with a non-specular (or ideally a diffuse) surface. The photon is in this context a collection of information about the interaction necessary for subsequent Monte Carlo estimation. Next, we trace a path from a camera as we would in path tracing, and the corresponding pixel estimate is updated only once we reach the vicinity of the photon and thus form a full path between the light source and the camera. Because path tracing naturally follows all specular interactions, this method is suitable for sampling SDS paths.

In terms of sampling efficiency, more complex bi-directional path-sampling algorithms such as *bi-directional path tracing* [Lafortune and Willems, 1993, Veach, 1997] and *VCM/UPS* [Georgiev et al., 2012, Hachisuka et al., 2012] have been proposed. However, these methods are not more efficient with respect to sampling caustics and SDS paths in general, they rather provide more robust estimates in the presence of glossy materials in the scene.

A common practical problem for any tracing algorithm that relies on sampling paths from lights, i.e. for light tracing and photon mapping and thus, in turn, for all bi-directional methods, is that we need to select an initial position for starting our light path from distant light sources optimally. For example, if we illuminate a large scene with an ocean by the sun and we select a starting position uniformly to make sure we cover the whole scene, then sampling a caustic visible in the camera can become even less likely than in the case of path tracer. We observe that sampling the starting position is critical to make bi-directional methods practical for realistically illuminated outdoor scenes by sunlight (see Section 5.5.2). If we use a small area light or even point lights with diffuse emission profile for producing caustics, sampling the direction of each emitted ray efficiently is similarly critical as sampling starting positions in the case of distant lights.

[3] We typically have two cameras close to each other for rendering stereo images.

[4] In the Hackbert notation, S+ means one or more specular vertices.

5.4 Choosing a path sampling method

In this section, we go over all methods for rendering caustics which we implemented in Manuka. While doing so, we discuss our experience with each method and our reasoning behind implementing them. Technical details of each method are described in next sections.

Path tracing. As already discussed in Section 5.2, we explored multiple methods for sampling caustic light transport. Because most of the production shots are rendered by path tracing implemented in Manuka, it was very appealing to study its capabilities for rendering caustics. This way, we can handle caustics in hair, volumetric effects, character's eyes, subsurface scattering, ray-facing discs etc., in a single integrator, while we can take also advantage of our existing importance sampling schemes for various problems unrelated to caustics. Furthermore, we can benefit from optimizations based on camera sub-path prefix, like for example optimized path space regularization (OPSR) [Weier et al., 2021], which regularizes indirect illumination to reduce the simulation cost.

As we explained in the previous section, we need to guide paths towards the sun light to enable rendering caustics with a path tracer in a reasonable time. However, we found out that finding initial caustic paths is quite unlikely due to the very small sun and thus path guiding learns slowly. While this provides significant improvement over simple path tracing, resulting render times are still not acceptable for production. To remedy this, we employ OPSR (see Section 5.5.1) which makes sampling of caustic light easier by introducing a small amount of roughness on the water interface. This way, we highly increase the probability of finding a caustic paths also thanks to enabling next-event estimation on the water surface. The disadvantage is that we introduce small amount of blur to caustics which becomes more significant with increased water depth. While this efficiently helps path guiding to learn faster and overall rendering times are significantly reduced, we have found that, with this setup, rendering caustics can be still relatively expensive.

Note that instead of relying on path guiding and OPSR, we also tried *manifold next-event estimation* [Hanika et al., 2015] which allows finding valid specular connections through one transmissive surface. While this method works well for shallow waters with mild frequency content, it turned out to be challenging to make it work reliably with deep water and high-frequency content due to convergence issues.

Light tracing. We experimented also with light tracing which we guided towards the camera. It turns out to be very fast with low-overhead per sampled path and it can be used for rendering caustics shot by a camera fully submerged in the water and it is our best tool for rendering sharp god-rays (see Section 5.5.2 and Section 5.5.4). Unfortunately, it also comes with a long list of disadvantages. While it can be good for some compositing workflows to provide a fast caustic pass when caustics are rendered in isolation, we cannot use it for rendering full shots in one render pass for multiple reasons. Firstly, as we discussed in Section 5.3, it cannot handle specular reflections which are present even in many underwater shots. Secondly, it is not very efficient for rendering full indirect illumination without introducing advanced importance sampling schemes like, for example, importance sampling *once-more scattered* events which we discuss in Section 6.

Another rather important disadvantage of light tracing is that it does not work out-of-the box with our adaptive sampling scheme which we use for the path tracer to distribute error in the image equally and which provides control over the overall image quality which users can specify upfront. While there have been studies to equalize error across the image plane using Markov-chain Monte Carlo (MCMC) [Gruson et al., 2016, Šik and Křivánek, 2019][5], we are not aware of similar work for MC based light tracer. We did not explore the MCMC possibilities further as it is infamous for temporal instability in animation and, more importantly, because we were able to entirely sidestep this limitation thanks to photon mapping.

There are another two issues we had to solve to be able to use light tracing. Firstly, we had to deal with over-blurred details coming from our subsurface scattering (see Section 5.5.2). And secondly, we had to extend our light tracer to produce deep samples to support deep-compositing workflows.

We found out that we can overcome some of the listed limitations, like missing specular reflections, extra implementation effort for deep samples, and inferior efficiency of sampling indirect illumination, when we combine light tracing with path tracing using MIS[6]. However, such a bi-directional combination is still not

[5] These works are designed for photon mapping, however, in principal it should be straightforward to apply the same techniques for light tracing.

[6] Note that we do not implement full bi-directional path tracing with connections between vertices of generated paths.

capable of handling SDS paths, that is scenes with camera above the water and does not solve the missing adaptivity of the light tracing part.

Photon mapping. This is the main reason why we have implemented also photon mapping. We found out that it is a robust technique that can handle the wide variety of our scenes and which do not suffer from the light tracing limitations described above. For example, thanks to the choice of our photon search radius, adaptive sampling is oblivious to the fact, that the contributions come from photon mapping and we can still control the image quality and distribute camera paths optimally. However, this method comes with its own set of limitations, and we had to make careful choices to address or at least mitigate them. One of the most important one is the bias in form of blur or potential light leaks, which are objectionable artefacts. This is closely related to problems with incorrect estimation of the true area of photon incidence for hair and other tiny geometry like little ray-facing discs or spheres often used for modeling special effects like foam. In the case of our hair primitive, photon mapping shares the same issue with light tracing which is the fact that the set of all possible directly visible ray-intersections computed from the camera is slightly different from the set of intersections computed for paths emitted from lights. As a result, the caustic lighting do not match exactly the outcome of our path tracer. While photon mapping can be made consistent[7] [Hachisuka and Jensen, 2009] or even unbiased[8] [Qin et al., 2015], these improvements come at the cost of increased render time. Instead, we fix the search radius so that the search area is comparable to the pixel footprint to ensure we do not blur the details across pixels which provides sufficient precision for our needs. This decision is also important for our adaptive sampler and firefly filter so that we avoid considering noisy but blurred contributions for features.

In the following sections, we describe each of our implemented methods, namely guided path tracing with OPSR, guided light tracing, and photon mapping in greater detail. Next, we devote Section 5.5.4 to our photon mapping combination with light tracing and with selective application of photon look-ups. We argue that this method which enables rendering caustics and other effects efficiently in one rendering pass saves users' time while it still allows for compositing workflows by separating outcomes of light transport into separate AOVs.

5.5 Details of implemented methods

5.5.1 Guided path tracing and optimized regularisation

To make rendering caustics feasible with a path tracer, we employ path guiding to sample paths incrementally towards small light sources or the sun. In our implementation, we use quad trees to represent 2D directional distributions which are cached spatially in the scene akin to the method by Müller et al. [2017] However, this approach suffers from slow initial learning of precise guiding distributions as we need to accidentally hit the sun first while relying on sampling proportionally to BSDFs of the material receiving the caustics (see Section 5.3). Note that this problem is shared between various guiding approaches based on learning regardless of their representation and choice of the learned quantity. The ideal solution for this problem would be exploiting a-priory knowledge about the sunlight direction and water position to limit sampling into a smaller region where light could refract into the volume, thus in turn, increasing guiding learning rate.

Instead, to make rendering caustics more practical with guided path tracing, we propose using optimised regularization (OPSR) of water BSDFs [Weier et al., 2021]. This means in practice that we introduce a small amount of roughness to the dielectric representing the water. The amount of roughness at the water surface for a given path depends on the camera path prefix and roughness of materials at its vertices. Hence, for directly visible caustics, it depends only on the primary path vertex. The function of final roughness is represented by a table which was optimized on a set of scenes outside of Manuka using differentiable rendering techniques.

Not only does this regularisation make light transport less difficult for sampling by filtering high frequencies, it also makes path guiding learn faster also thanks to enabling next-event estimation. Note that without the regularization, it would not be possible to use next-event estimation on purely specular surfaces.

The optimized regularisation provides a control over the trade-off between the speed and the quality. The quality of rendered caustics depends directly on the amount of introduced roughness. More roughness results

[7] The introduced bias vanishes over time.

[8] The error is only in the form of variance which translates into noise similarly as for path tracing.

in blurrier caustics while the light transport becomes more efficient. We offer several levels of roughening from "low" to "aggressive" represented by different tabulated functions coming from the offline optimization.

While guided path tracing with OPSR can be forgiving in many shots in the sense that artists are happy about the result and introduced blurriness, it may not be acceptable in others when the intention is to have sharp accurate caustics everywhere. To that end, we implemented also methods which rely on tracing paths also from lights. However, note that we still rely on path tracing with OPSR in the combined photon mapping method for rendering for example hair or character eye's which are challenging to handle by photons (see Section 5.5.4).

5.5.2 Guided light tracing

We found out that light tracing, despite its bad reputation, can serve as a good and fast strategy for providing directly visible caustics which can already serve for some workflows where caustics are rendered separately and applied in the compositing stage. However, the necessary condition for efficient light tracing in most of the indoor[9] and outdoor scenes is *guided emission* of paths from lights. This method was already covered in the previous *Path Guiding in Production Course* [Vorba et al., 2019], however, we will describe a few more technical details here.

In Section 5.3, we were explaining that in the case of large open water scenes illuminated by sunlight, we need to importance sample the starting position of our rays well. To that end, we model the sampling density of starting positions with a quad tree which is mapped to a disk perpendicular to the direction of the sun. We adapt the distribution proportionally to the contributions of previously sampled paths to the image. Thanks to the intensity of the sunlight, guided emission learns relatively quickly as it usually receives high number of initial samples.

To regularize the quad tree distribution, we apply filtering of the quad tree with progressively decreasing kernel size. While a larger size helps with exploring the space in initial progressions to find the contributing paths, the smaller radius enables to model the distribution more precisely. To determine maximum kernel size, we employ ray-differentials which enables us to relate the kernel size to pixels on the screen.

We found out that along screen edges, we can observe regions of pixels which converge substantially slower than the rest. The reason is that the view importance is discontinuous at the edges of view frustum which the quad tree tends to underestimate. To mitigate this issue, instead of subdividing the quad tree only based on the fraction of energy represented by each tree leaf [Müller et al., 2017], we also encourage splitting of leafs which exhibit high variation of the underlying signal. To that end, we tentatively split the leafs for collecting samples while we still sample ray positions from the original leaf. If the energy collected in tentative leafs is highly non-uniform, we encourage the proper split. In this way, the distribution tends to achieve higher resolution in places corresponding to screen edges.

As we discussed in Section 5.4, light tracing has many limitations like for example not being able to sample reflections. However, it does not only serve as an intermediate step towards photon mapping, but we also rely on it as a method providing single scattering contributions responsible for god-rays [Vorba et al., 2019]. To that end, during tracing light paths and storing photons, we also connect to cameras. To increase the probability of scattering within the water volume, we introduce scattering events on a ray segment intersecting the camera frustum. Furthermore, we importance-sample the geometric factor by using equi-angular sampling [Kulla and Fajardo, 2011].

If we want to apply light tracing for resolving caustics on characters, plants or corals, we have to deal with incorrect blur of texture details. This loss of contrast stems from the fact that we define volume properties with surface textures and for path tracing, the volume for a path is given by the point of its entry into the volume. This needs to be taken into account for bi-directional methods so that we can arrive at the same outcome as with path tracing. Specifically, for light tracing, we have to re-evaluate transmission and scattering coefficients along the whole subsurface chain when we exit the volume because only at exit, we can learn what would have been the volume properties if we have traced the path from the camera. This has potentially negative impact on importance sampling of the subsurface path because phase function and free path distance are sampled based

[9] In case of indoor scenes, light from sun and from environment maps usually enters the room through windows.

on a different volume properties. However, in practice, given our models, this turned out to be a negligible source of variance and caustics can be computed still very efficiently.

5.5.3 Guided photon mapping

We decided to depart from the state-of-the art methods like VCM/UPS which can compute caustic light transport. Their main advantage is robustness with respect to presence of diffuse, glossy and specular materials in one scene which means that render time does not grow substantially. This property stems from the combination of bi-directional path tracing [Lafortune and Willems, 1993, Veach, 1997] and bi-directional photon mapping [**?**] in one framework using MIS [Veach, 1997] instead of heuristics. They comprises both incremental path tracing from lights and from cameras. Subsequently, they form multiple different full paths using vertex connection strategy inherited from bi-directional path tracing, that is they create explicit connections of chosen light and camera sub-path vertices. They also form various full paths based on photon merging at different camera path vertices similarly to bi-directional photon mapping. As a result, the same light transport path can be formed by multiple strategies each with a different probability and to arrive at the correct estimate, each path is weighted by MIS weight. We argue that this creates a significant overhead since many strategies are redundant in the presence of path guiding [Vorba et al., 2014] and maintaining such complex algorithms is not practical in a production renderer [Vorba et al., 2019].

We thus rather return to a heuristic-based approach similar to original method by Jensen [1996]. However, as we have discussed in previous sections, we observed that while guided path tracer can resolve various light transport paths relatively efficiently, it is still not as efficient as strategies which trace paths from lights. Thus, in our photon mapping implementation, we use photons only for resolving caustic light paths which allows us to store only caustic photons. All the other types of light transport are handled by guided path tracing with next-event estimation. This means that while we trace a path from the camera, we search for photons at every non-specular vertex to account for caustic illumination. To avoid computing the same transport twice, we assign zero weight to every camera path which happen to sample caustic illumination, whether it is unidirectional collision with the light source or connection made by next-event estimation[10].

We consider a photon being a caustic photon as long as it was emitted from sunlight and went through specular interactions and landed on a non-specular receiver, that is a surface receiving caustics. Note that we do not emit photons from environment maps (aka IBLs), because they usually do not contain high-frequency details that would require tracing photons. Thus, handling these IBL contributions by the path tracer is usually more efficient.

In practice, it is a problem to rely on component flags telling us which BSDF is purely specular and materials often have at least a small amount of roughness.[11] Thus, we introduce a threshold on roughness to classify vertices as being specular. However, setting up such a threshold robustly for all scenes is difficult and it works out best to ask users for manual tagging of objects which should cast caustics, like for example the water interface. As soon as a photon is deposited and there is no specular component on the material, we terminate tracing of the light path because the photon could no longer be classified as specular.

It may seem that when we decide about storing a photon we face the similar problems with classification of non-specular components at caustic receiver. However, in this case, the threshold on a small amount of roughness seems to work well. While in general, receiving a non-caustic photon on a glossy surface may be the source of a strong variance since it is unlikely that photons will come from optimal directions, we argue that it is not the case for caustic photons as they always represent the only directions responsible for non-zero contribution. In this context, it is important to remind, that all indirect light including glossy reflections except for the caustics is still resolved by the path tracer. This is also the main reason why we do not need all path sampling strategies involved in VCM/UPS which favor strategy with the highest probability for sampling the given transport. We already know a-priory that the best caustic sampling strategy with the highest probability will be vertex merging of the caustic photons.

[10] We can make next-event estimation from almost specular surfaces with small amount of roughness and classify such path as a caustic.

[11] One reason for this is, for example, LEAN mapping which translates geometric details into roughness during filtering.

As opposed to original photon mapping, we use the stochastic approach of SPPM [Hachisuka and Jensen, 2009] and VCM/UPS to lift memory limitations of the original method, that is we interleave tracing photons and paths from camera at every progression. One progression in Manuka is defined by tracing the number of camera paths equal to the image resolution. In summary, one photon mapping progression looks as follows:

1. Trace light paths and store caustic photons.
2. Build a photon map.
3. Trace camera paths and search for caustic photons at every vertex.
4. Compute MIS weights for unidirectional path tracing, next-event estimation, and other possible path sampling techniques like, for example, equi-angular sampling and assign zero weight to path tracing contributions which are resolved by photons.

As we already discussed in Section 5.4, we do not progressively reduce the photon *search radius*, but we rather keep it fixed so that the search area is roughly equal to the pixel footprint. To that end, we employ ray-differentials which we track also through specular interactions to ensure a consistent search radius also for caustics visible in reflections. This strategy also helps with efficient rendering of caustic light at the horizon while not blurring caustics near the camera. Note that this is not possible to achieve by using a uniform radius across all the pixels unless we employed progressive reduction of the search radius. However, such approach could result in significantly non-uniform convergence rates across image plane and would not ensure that adaptive sampler, which distributes camera paths over the image and controls the target quality, does not confuse introduced blur for a feature (see Section 5.4).

Keeping the search radius small and comparable with a pixel footprint also helps to mask visible light leaks. To further suppress these issues, we carefully check the normal at the end of the camera sub-path where we search for the photons and the normal at the incident position of each photon. If these two normals are too different, we do not merge the photon. Additionally, we do not store photons on some special types of geometry like for example hair (see the details in Section 5.5.4)

Manuka is a *spectral* renderer which means that we need to deal with the fact that light paths and camera paths are traced at different wavelengths. More specifically, during merging of the camera sub-path and a photon, we also apply a kernel which merges paths only if wavelengths of both camera and the photon are close enough. From our experience, setting wavelength radius of Epanechnikov kernel to 50 nm worked well in our scenes without introducing noticeable color shift. Because, in our implementation we use hero-wavelength sampling [Wilkie et al., 2014] with four wavelengths per path, we need to consider all possible 16 pairs of contributing wavelengths and weight them appropriately using MIS. To accelerate spatial search for the nearest photons, we maintain one kd-tree for all caustic photons regardless of their wavelength.

As for *subsurface scattering*, we sidestep the bi-directional problems related to the point-of-entry definition of our volumes described in Section 5.5.2. We could re-evaluate the subsurface chain for each photon exiting the volume in the same way as we propose for light tracing. However, we take a more efficient approach where we do not even let photons enter subsurface scattering and store them on the surface instead. Subsequently, when we trace paths from the camera, we look for caustics contributions when we exit the subsurface volume which ensures that the camera paths were traced based on the correct volume properties and we do not need to re-evaluate the path. Note that this also saves some time on tracing subsurface chains for photons. In contrast, it would not be possible to avoid tracing camera paths through subsurface if we stored re-evaluated photons on exit. The reason is that not all transport contributing to subsurface scattering is represented by caustic photons (like for example contributions coming from the environment map which we resolve by path tracing).

In Section 5.5.2, we described *guided emission* for light tracing which is also necessary for efficient photon mapping. We only need to extend the photon structure to remember start position and direction of the initial ray. This in turn allows to convey these data to learning as soon as a full path contribution is formed through photon merging. Furthermore, to enable the regularisation of the guiding distribution by filtering (see 5.5.2), we track both photon differentials with each light path and camera ray-differentials for camera paths to obtain a pixel footprint. Subsequently, to arrive at the filter kernel size when we splat the contribution into the guiding distribution, we scale the pixel footprint using the corresponding photon differentials.

To further increase efficiency of photon mapping in very large scenes, we recompute very early (after only a few progressions) the size of the disk from which we emit the light paths. The reason is that the initial disk size must be as large as the scene to ensure that all visible regions can be hit by a photon. In very large scenes, this can significantly limit the resolution of our quad tree and contributes to slow learning process. Furthermore, note that we still sample a significant number of light paths from uniform distribution over the disk which is our default unguided strategy. Decreasing the disk so that it occupies only area of visible caustics also makes this uniform sampling, which is important for discovering visible caustics, more efficient. To that end, we collect all caustic photons from a first few progressions which made a non-zero contribution to the image. We then use them to compute a size of the new disk where our guiding distribution is defined and from which light paths start. We make sure that the new disk is capable of covering the area of all collected photons while we add a small delta to reduce the risk that we would miss any caustic.

5.5.4 Combined method

In previous sections, we have described multiple techniques for rendering caustics while discussing their strengths and weaknesses. We argue that for users, the ideal method would be the one which could efficiently render caustics in any of our wide variety of scenes together with all other light transport types. Such a method would allow users to avoid splitting renders in more passes than necessary and thus save their precious time.[12]

These requirements are fulfilled with photon mapping described in Section 5.5.3 which is after all a combination of caustic photons and guided path tracing described in Section 5.5.1. Nevertheless, to make the combined method truly applicable on all our water scenes including underwater shots, we must not forget adding explicit connections to the camera made from light tracing when we produce photons as we described in Section 5.5.2. To render once-more scattered events, we can also improve light tracing by the method described in Section 6.

We have mentioned that we do not store photons on hair or eyes and thus caustics on those have to be resolved with guided path tracing and OPSR. Usually hair and eyes have smaller screen coverage than the environment, so slower resolution is acceptable, but it is important not to compromise quality of hero characters. This is the primary reason why we do not use photons for eyes in order to not blur any interesting caustic on the iris with small geometric details. However, we have not experimented with this approach and it would be interesting to see the results. As for hair, it is not only likely that we would obtain light leaks at the pixel level due to having multiple hairs in a single pixel, but we also get different set of intersections compared to path tracer. The reason is that hair are also transparent and modeled as cylinders but light transport is not simulated as subsurface scattering.

At this point, we would like to discuss some limitations. Photon mapping works well with our adaptive image sampler as we discussed in previous sections. However, light tracing contributions are produced independently during the rendering process, and it is not ensured that these contributions converge to a uniform error. We have to simply rely that we will sample enough paths before other any light transport in the scene converges. Thus it would require further research into making light tracing contributions adaptive in the image plane to equalize error.

We considered adapting the amount of traced paths from lights to further speed up the light transport [Grittmann et al., 2018]. This extension could be an interesting avenue for shots with very limited number of pixels featuring caustics, however, it would most likely not help in many underwater shots which are dominated by caustic light in every pixel.

5.6 Removing caustics

In the previous sections, we were looking for efficient physically-based methods to compute highly accurate caustics. However, sometimes artists request the exact opposite which is not to render caustics at all. This is quite straightforward to achieve but more often the request is to preserve the energy while removing the interesting caustic patterns due to light interaction with curved refractive surfaces. As a result, such optimization

[12] It is quite common in production to split passes into environment pass and hero-character pass to save re-rendering time when characters are edited.

typically results in faster light transport as sampling the precise interaction is more difficult for reasons we explained in Section 5.3.

One option is to use aggressive roughening, that is to introduce high roughness to BSDF modeling, for example, water interface. However, this usually results into increased energy in the water volume and thus into much brighter image.

The other option, often used in production renderers and in architecture visualizations, is to make the volume boundary interface transparent so that rays exiting the volume do not bend; we call this *switch to transparency*. We provide users with the ability to tag refractive objects responsible for casting caustics, like for example the water surface, and select one of the predefined light path expressions (LPE) controlling for which paths the object will become transparent. In this way, we can for example allow camera paths to refract through the material and become transparent only after first non-specular interaction. Another example would be to become transparent for all paths which scattered within the volume which effectively suppress volumetric caustics.

Furthermore, we even allow to control the transparency on the level of caustics receiving objects. Normally, the transparency can affect all the paths passing through a refractive material if it satisfies the LPE expression, but our users can select caustic receiving objects and make exceptions for paths interacting with them.

It is worth noting that this switch-to-transparency optimization allows users to make an artistic choice and somewhat depart from physically-based rendering of the caustic light transport. It can be typically used for shots or parts of a shot where it is forgiving not to simulate light bending honestly like for example thin water film on background characters.

5.7 Compositing

In VFX production pipelines, rendering is an intermediate stage, and the final image is prepared and delivered in a compositing department, where artists make adjustments and address director feedback. To support various image manipulations after rendering, which includes also de-noising, Manuka provides a large number of auxiliary buffers, called AOVs (arbitrary output variables). Some of them are breakdowns of the final image into contributions from groups of light sources, contributions based on the type of surface interactions like for example diffuse/glossy/specular or reflect/transmit, or contributions based on the type of light transport. To support compositing workflows for water sequences, we also provide the caustics AOV, which sifts out caustic light paths in a separate buffer. In practice, it is not always easy to classify caustic vs non-caustic light paths due to fuzzy boundary between specular and glossy interactions which we already discussed in Section 5.5.3. To mitigate this issue, we use a combination of automated criteria based on roughness thresholds and user input in the form of tags to classify light bounces as specular. Note that this approach overlaps with our classification of caustic photons.

In addition to 2D image processing, VFX relies on *deep composting*. It operates on a 2.5D representation of the rendered image which includes depth and separates light samples based on various criteria, such as object id. The key advantage of deep images is that they allow to seamlessly merge rendered content and also actual footage captured on stage. To satisfy image quality criteria in production, deep samples need to have accurate *alpha*, i.e., opacity or pixel coverage, which enables artefact free compositing. Sample opacity is easy to compute for path tracing from the camera and thus also for photon mapping when photons are gathered by tracing camera paths. However, it is not straightforward for light tracing. Since we rely on light tracing in volumes to provide *god-rays* in underwater shots, we had to extend our deep display output for this case to fully enable deep composting workflows.

5.8 Conclusion and future works

In this part, we discussed challenges for computing caustic light transport in water scenes of large production scale. We introduced a tool set of methods available in Manuka for rendering caustics, presented their strengths and weaknesses while showing our way towards a practical method which can handle the wide variety of water scenes. More specifically, we described our work towards path tracing capable of rendering

caustics. This is very appealing method for many production rendering systems which rely on its simplicity of implementation and many convenient properties. We identified and discussed limitations of our guided path tracing approach with optimised regularisation and moved towards bi-directional methods while we still rely on path traced solution for hair and character eyes. We argued for simplest possible implementation of photon mapping which use only caustic photons and all other light transport is resolved by guided path tracing. We stressed the importance of guided emission from lights so that the light paths generating caustic photons aim at the visible regions of the scene. To solve the volumetric contributions responsible for god-rays, we proposed using guided light tracing with equi-angular sampling during photon tracing. We have also discussed different means of artistic control with respect to our physically-based approach of resolving caustic light and briefly suggested what we had to considered to fully support compositing workflows.

While our proposed method for solving caustic light transport based on photons works reasonably well, produces sharp caustics and is relatively light-weight for implementation, we would still encourage researchers to find ways of efficient rendering accurate caustic light transport by path tracing. The main motivation, as we state above, would be the simplicity of path tracing implementation and convenient maintenance of such a rendering system. We can imagine that this could be done through further improvements to root-finding methods, or path guiding. It is worth mentioning that it would be also worth exploring capabilities of path cuts [Wang et al., 2020] or slope space integrals [Loubet et al., 2020] in production scenes. Alternatively, interesting avenue would be exploiting machine learning and a-priory knowledge about rendered scenes together with some of the state-of-the art methods named above.

With respect to light tracing, we identified an interesting practical disadvantage. We miss an adaptive sampler which would allow equalizing the target error across the image plane. Ideally, we would like to have a solution that would work reliably within Monte Carlo framework and without temporal issues in animated sequences. At this point, it is worth noting that if light tracer was not necessary for resolving (volumetric) caustics, then research in this direction would become purely academic.

We also touched on the aspect of artistic control of generated caustics. On the light transport level, we can only control the amount of introduced blur or potentially remove the caustics by turning refractive materials into transparent. We highlighted that, generating plausible and interesting caustics through the means of physically-based light transport simulation is heavily dependent on realistic geometry inputs which our studio generates through involved fluid simulations. However, adjusting the caustics quickly to match artistic intention by for example changing the underlying simulation or the frequency of water waves, or even doing so locally at specific receivers is almost impossible. We realize that this challenging problem is stretched across light transport and modeling but it might be interesting to explore capabilities of neural representations in this context.

Acknowledgment

I would like thank to all the people who contributed to developing Manuka. With respect to rendering caustics, I would like to thank namely to Philippe Weier for his work on OPSR, Shijun Haw for his initial photon mapping implementation, Marc Droske and Johannes Hanika for their work of MNEE, Marc Droske, Andreea Bizdideanu for their work on switch-to-transparency, Jorge Schwartzhaupt and Javor Kalojanov for their work on the deep and display subsystems, and Jean-Marie Aubry for his work on the volumetric deep samples for light tracing.

References

Philip Dutré and Yves Willems. 1994. Importance-driven Monte Carlo light tracing. In *Eurographics Workshop on Rendering*.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph. (SIGGRAPH Asia '12)* 31, 6 (2012).

Pascal Grittmann, Arsène Pérard-Gayot, Philipp Slusallek, and Jaroslav Křivánek. 2018. Efficient Caustic Rendering with Lightweight Photon Mapping. *Computer Graphics Forum* 37, 4 (2018). EGSR'18.

Adrien Gruson, Mickaël Ribardière, Martin Šik, Jiří Vorba, Rémi Cozot, Kadi Bouatouch, and Jaroslav Křivánek. 2016. A spatial target function for metropolis photon tracing. *ACM Transactions on Graphics (TOG)* 36 (2016). https://doi.org/10.1145/3072959.2963097

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–8. https://doi.org/10.1145/1618452.1618487

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph. (SIGGRAPH Asia '12)* 31, 6 (2012).

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Comput. Graph. Forum* 34, 4 (jul 2015), 87–97.

Henrik Wann Jensen. 1996. Global Illumination Using Photon Maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*. Springer-Verlag, London, UK, UK, 21–30. http://dl.acm.org/citation.cfm?id=275458.275461

Christopher Kulla and Marcos Fajardo. 2011. Importance Sampling of Area Lights in Participating Media. In *ACM SIGGRAPH 2011 Talks (SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 55, 1 pages. https://doi.org/10.1145/2037826.2037899

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Proc. of Compugraphics '93*.

Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob. 2020. Slope-Space Integrals for Specular Next Event Estimation. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39, 6 (Dec. 2020). https://doi.org/0.1145/3414685.3417811

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Comput. Graph. Forum* 36, 4 (jul 2017), 91–100. https://doi.org/10.1111/cgf.13227

Hao Qin, Xin Sun, Qiming Hou, Baining Guo, and Kun Zhou. 2015. Unbiased Photon Gathering for Light Transport Simulation. *ACM Trans. Graph.* 34, 6, Article 208 (nov 2015), 14 pages. https://doi.org/10.1145/2816795.2818119

Martin Šik and Jaroslav Křivánek. 2019. Implementing One-Click Caustics in Corona Renderer. In *Eurographics Symposium on Rendering - DL-only and Industry Track*, Tamy Boubekeur and Pradeep Sen (Eds.). The Eurographics Association, 61–67. https://doi.org/10.2312/sr.20191221

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Ph.D. Dissertation. Stanford University.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses (SIGGRAPH '19)*. ACM, New York, NY, USA, Article 18, 77 pages. https://doi.org/10.1145/3305366.3328091

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph. (SIGGRAPH '14)* 33, 4, Article 101 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601203

Beibei Wang, Miloš Hašan, and Ling-Qi Yan. 2020. Path Cuts: Efficient Rendering of Pure Specular Light Transport. *ACM Trans. Graph.* 39, 6, Article 238 (nov 2020), 12 pages. https://doi.org/10.1145/3414685.3417792

Philippe Weier, Marc Droske, Johannes Hanika, Andrea Weidlich, and Jiří Vorba. 2021. Optimised Path Space Regularisation. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.14347

A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. 2014. Hero Wavelength Spectral Sampling. In *Proceedings of the 25th Eurographics Symposium on Rendering (EGSR '14)*. Eurographics Association, Goslar, DEU, 123–131. https://doi.org/10.1111/cgf.12419

6 Singularities inside the water volume

Johannes Hanika, *Karlsruhe Institute of Technology*

Immediately visible objects in computer generated imagery are prominently represented by surface geometry and are of course very important for the scenery. Light which scatters between the surfaces, inside the participating medium, also contributes to the look of the image. As the field of rendering matures, the importance of this has been recognised more and more. While clouds and atmospheric scattering obviously depend on volumetric transport, it is now commonly accepted that also many (if not all) objects require treatment of the volume to look real. For instance skin doesn't look like skin without the transport under the surface, and, as we have seen in Andrea's section, interaction of surface reflectance with the surrounding medium is very important to get right. In this chapter we want to focus on underwater ocean rendering and the specific transport effects with their unique set of challenges. In particular we present a recently published method to sample *once-more scattered light paths for next event estimation* (OMNEE) by Hanika et al. [2022], and discuss how it ties into the production rendering context and which effects can be modelled better when using the technique.

6.1 Participating Media

The mathematics and the physical basis of volumetric light transport are well understood and summarised in a report by Novák et al. [2018]. In principle, light is not only collected over the incident hemisphere at surfaces, but also along the line of sight, gathering in-scattered light at every point along the ray. The material properties involved here describe the density and size of the volumetric particles (resulting in collision coefficients μ) and the phase function f_s which is the volumetric equivalent of a BSDF. Light transport in a volume can be described as path space integral

$$\int_{\mathcal{P}} f(\overline{\mathbf{x}})\, \mathrm{d}\overline{\mathbf{x}} \tag{6.1}$$

where $\overline{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_1, \cdots \mathbf{x}_k) \in \mathcal{P}$ are light transport paths consisting of a list of vertices, connecting the sensors and the light sources. The integrand is the *measurement contribution function* and is a product of multiple relevant terms (all summarised by Novák et al. [2018]) and we will detail the ones required here later.

6.2 Singularities

Singularities are a long standing sore topic for light transport simulation. The mathematical description using a Dirac delta is often used ambiguously and the practical implementation is often realised using special cases.

Most prominently, singularities in rendering are connected with SDS scenarios, where a diffuse interaction (D) is surrounded by specular (S) interactions (caustic seen in a mirror). Here, the singularity is caused by the BSDF f_r (the mirror). The second most prominent singularity is caused by scattering vertices coming arbitrary close to each other (in a geometric cavity, where two walls meet). The singularity happens when the distance becomes zero, and the square distance is divided out to compute the geometry term G. The damage done here is often limited: offending geometric cavities are sparse and we don't have to use sampling techniques with deterministic connections. What's more, usually we can alleviate the problem by using multiple importance sampling: the problematic paths will be weighted down and other techniques will take over.

In participating media, the possibilities for problems are amplified: now every path vertex, if on a surface or in the medium, is potentially very close to the next vertex, which might be in the volume. This means that every deterministic connection which employs a geometry term is potentially causing a singularity.

While the equivalent of a mirror BSDF is rare in media (this would essentially remove the medium), Mie scattering does result in very peaky angular distributions. As we have seen in the previous section, ocean water results in phase functions with mean cosines way beyond 0.9. These almost specular peaks degenerate smoothly: the problems with the singularities will gradually appear as firefly pixels in the image. For the other kind of singularity, this means that close by volume vertices have one dimension more to cause problems than surface points in cavities.

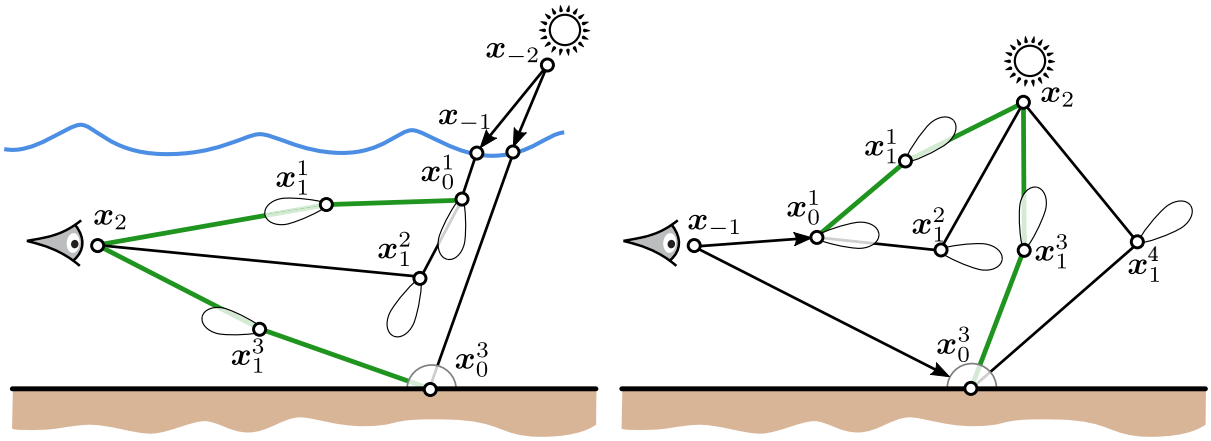


Figure 6.1: *Reproduced from Hanika et al. [2022]. **Left:** Classic light tracing with next event estimation or equiangular sampling samples a direction at a volume vertex $\mathbf{x}_0^1$ or surface vertex $\mathbf{x}_0^3$ towards $\mathbf{x}_1^1$ and $\mathbf{x}_1^3$, respectively, typically by importance sampling the BSDF. The high throughput 2-segment connections (green) that bend at these vertices toward the eye and create an intermediate vertex $\mathbf{x}_1^1$ or $\mathbf{x}_1^3$ are sampled with low probability and thus suffer from variance. Our technique is tailored to sample these connections efficiently. **Right:** The path tracing case analogously samples such 2-segment connections towards the light.*

6.3 The Family of underwater Scattering Effects

Figure 6.1 shows a few configurations of light transport paths with participating media. The left image illustrates light tracing, i.e. starting at the light source and connecting to the eye. The right image shows path tracing, i.e. starting at the eye and connecting to the light source. The former is more suitable for underwater caustics, sunbeams, and point spread simulation (as Jirka has detailed in a previous section).

There are mainly three prominent effects under water: caustics on the ground or on objects, sunbeams in the volume following the movement of the waves, and the more subtle blur of all visible things which increases with distance. This last effect is marked in green in the left side of figure 6.1, and is caused by at least one scattering vertex $\mathbf{x}_1$ in between the eye $\mathbf{x}_2$ and the visible object or sunbeam $\mathbf{x}_0$.

This effect is indeed separate from other features, which can be observed in figure 6.2: the images show three path vertices in a medium, where $\mathbf{x}_0$ is the current end point of a path started at the eye, $\mathbf{x}_2$ is the connection at a light source, and the measurement contribution quantifying the transport in between these is collected over all possible $\mathbf{x}_1$ in between these two vertices, and visualised as a radial plot around $\mathbf{x}_0$.

The transport is governed by the phase function at $\mathbf{x}_0$ (plotted in dashed orange) as well as by the incident radiance, which includes the emission at $\mathbf{x}_2$ as well as the phase function at all possible $\mathbf{x}_1$.

For moderately forward scattering phase functions as shown on the left (with mean cosine $g = 0.5$), the two phase functions form broad lobes and their product becomes another broad lobe (shown in blue). However for more peaky scattering ($g = 0.95$, shown on the right), the effects separate completely. This shows that both phase functions are equally important to simulate, depending on position in path space each one might dominate. It also explains why we can't just simulate one of them and hope that the other one will follow as a by product, as might be the case when two broad lobes merge.

The plot here is conducted with a simple unit test program using a Henyey-Greenstein lobe, but this separating behaviour always occurs as soon as the phase function is peakier than a Gaussian: the product of two Gaussians is another Gaussian, so there would be no separation into two different lobes.

Convolving with a point spread function in post Since mostly the depth-dependent blur of visible objects in subtle, it is tempting to try and reproduce the effect as a post-process and blur an RGBD image (with depth channel) by a point spread function (PSF). While this approach can certainly produce appealing results, doing it really well is at least as expensive as doing the path space sampling: the (approximate) treatment of occlusion certainly requires a deep compositing workflow. Trying to reconstruct visibility accurately may re-

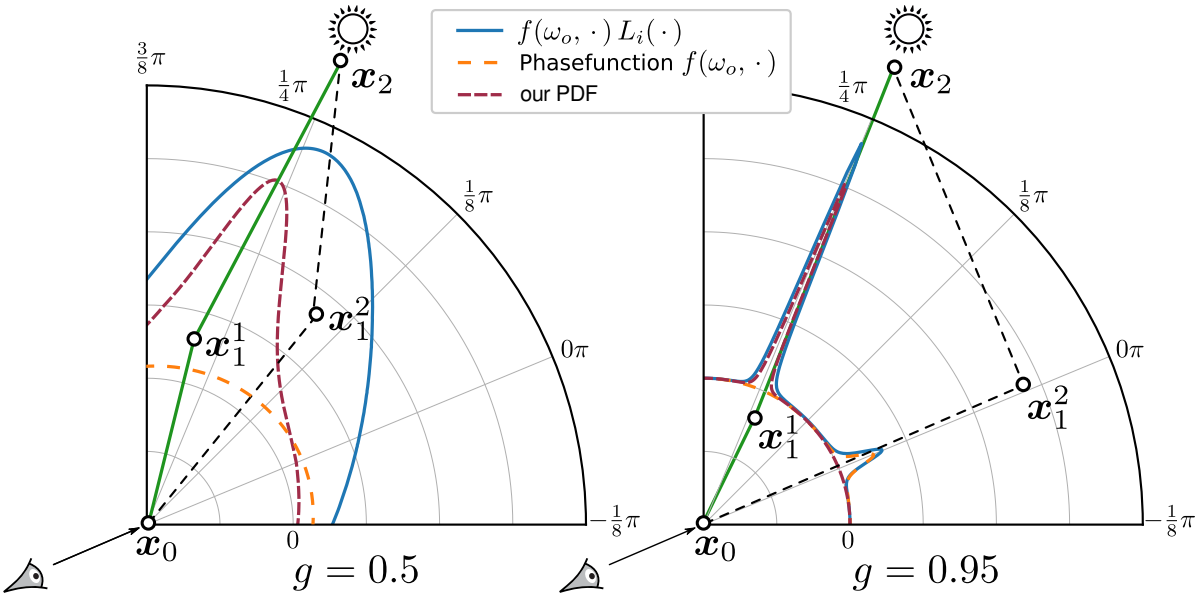


Figure 6.2: *Reproduced from Hanika et al. [2022]. In the volumetric double scattering case, the outgoing radiance $L_o(\mathbf{x}_0)$ is a result of scattering at $\mathbf{x}_0$ and an intermediate vertex $\mathbf{x}_1$. The main product terms, the one-bounce incoming radiance $L_i(\mathbf{x}_0)$ and the phase function f_0, are strongly competing in media with anisotropic phase functions. First sampling a direction according to f_0 and subsequently sampling $\mathbf{x}_1$ handles the forward scattering of the phase function well but misses the distinct peak centred around the direction towards $\mathbf{x}_2$. Our method is tailored to sample paths (green) corresponding to this peak: the dashed bordeaux-coloured line corresponds to the marginalised solid-angle PDF at $\mathbf{x}_0$ of our sampling method.*

quire sophisticated techniques such as the one by Lehtinen et al. [2012]. In addition to that, the support of the PSF is not limited: a very bright light source or glossy reflection may cause an unbounded glare in image space. Simulating this with a 2D convolution filter results in long run times. This can be alleviated by image space importance sampling— a technique that is commonly already present in path space sampling. Nevertheless, in some applications it may be the better workflow to decouple the blur from global illumination: often this enables compositors with greater flexibility, especially if production requests are changing rapidly.

6.4 OMNEE: Once More collided Next Event Estimation

As we have seen in the previous section, there is a particularly interesting (if subtle) effect to sample which can be characterised as next event estimation with one additional collision event on the way from the current path vertex $\mathbf{x}_0$ to the end point of the path $\mathbf{x}_2$. To compute this flux transported from $\mathbf{x}_0$ to $\mathbf{x}_2$ we need to integrate over all possible $\mathbf{x}_1$ in (3D) vertex area measure:

$$I(\mathbf{x}_0 \leftrightarrow \mathbf{x}_2) = \int_{\mathbf{x}_1} f(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, \mathrm{d}\mathbf{x}_1, \tag{6.2}$$

where the integrand is the measurement contribution function

$$f(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = f_r(\mathbf{x}_0) \cdot G(\mathbf{x}_0, \mathbf{x}_1) \cdot G(\mathbf{x}_1, \mathbf{x}_2) \cdot \mu_s \cdot f_s(\mathbf{x}_1) \cdot T(\mathbf{x}_0, \mathbf{x}_1) \cdot T(\mathbf{x}_1, \mathbf{x}_2) \cdot W(\mathbf{x}_2).$$

Here, $W(\mathbf{x}_2)$ indicates that the path is a light tracing path and the next event connection is performed towards the eye. Of course this can also be performed towards the light, as in figure 6.2. The symbol would then be $L_e(\mathbf{x}_2)$. The rest consists of the BSDF f_r at $\mathbf{x}_0$, the phase function f_s as well as the scattering coefficient μ_s at $\mathbf{x}_1$, and the geometry terms G and transmittances T along the two path segments.

In an effort to sample the strong forward peak in figure 6.2 (right), we analyse the integrand, searching for important terms to consider for sampling. Since the peak is so narrow, the total distance travelled from $\mathbf{x}_0$ to $\mathbf{x}_2$ is always similar, leaving the product of the two transmittances T in about the same range. We can

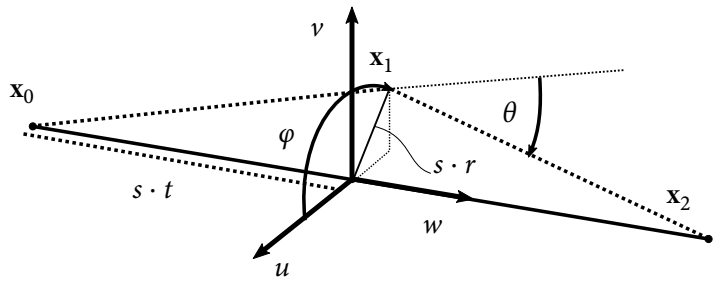


Figure 6.3: *Reproduced from Hanika et al. [2022]. The local vertex area measure coordinate frame (u, v, w) and the rotationally symmetric scattering along φ.*

thus rule these out of the equation for sampling. Certainly the biggest impact stems from the phase function $f_s(\mathbf{x}_1)$, which governs the angular configuration of the path. One more degree of freedom is where to place $\mathbf{x}_1$ with respect to the two end points $\mathbf{x}_0$ and $\mathbf{x}_2$: this has a large impact on the measurement since the geometry terms G are unbounded and can degenerate for small distances between vertices. Assuming furthermore that, as standard NEE, we can ignore $f_r(\mathbf{x}_0)$ and the directional component of W, we arrive at a separation into *constant* (f_c) and *interesting* (f_i) part:

$$f_c(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \cos\theta_0 \cdot \cos\theta_2 \cdot \mu_s \cdot T(\mathbf{x}_0, \mathbf{x}_1) \cdot T(\mathbf{x}_1, \mathbf{x}_2) \cdot W(\mathbf{x}_2), \tag{6.3}$$

$$f_i(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \frac{f_s(\mathbf{x}_1)}{d_2^2 \cdot d_1^2} \tag{6.4}$$

that we wish to importance sample. The geometry of the sub path is illustrated in figure 6.3. The distances between vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ is denoted as d_1, and d_2 signifies the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$.

To arrive at an importance sampling scheme for this, we want to first sample the phase function angle θ, as this is the most important ingredient for this type of paths. Next, we want to sample the fractional distance $t \in (0, 1)$ between $\mathbf{x}_0$ and $\mathbf{x}_2$ where $\mathbf{x}_1$ will be located. Finally, $\mathbf{x}_1$ will be found on a disk perpendicular to the straight connection from $\mathbf{x}_0$ to $\mathbf{x}_2$, and off to the sides in polar coordinates with angle φ (which we will sample uniformly) and a radius that is determined by θ.

Sampling like this requires us to first express the coordinates of the vertex $\mathbf{x}_1$ in a parameter space which includes θ and t and φ as independent coordinates. While this seems like a standard change of variables, the equations in the derivation become a bit lengthy, so we refer to Hanika et al. [2022] here. We only want to repeat the sampling procedure, because surprisingly the end result is quite compact:

- sample an outgoing direction at $\mathbf{x}_1$ via the regular solid angle phase function sampling routine, store θ,
- sample the fractional distance $t \sim p(t|\theta)$ via equation (6.5)
- sample $\varphi \sim \frac{1}{2\pi}$ (or reuse from phase function sampling)
- reconstruct location of $\mathbf{x}_1$ using t and r from equation (6.6).

The inverse CDF to sample the fractional distance t given θ and a uniform random variable $\xi \in [0, 1)$ is:

$$P^{-1}(\xi|\theta) = \cos(\theta - \xi\theta)\sin(\xi\theta)/\sin\theta, \tag{6.5}$$

and as the normalised polar coordinate radius r in terms of t and θ is given as:

$$r(t, \theta) = \sqrt{\frac{1}{4\sin^2\theta} - (1/2 - t)^2} - \sqrt{\frac{1}{4\sin^2\theta} - 1/4}. \tag{6.6}$$

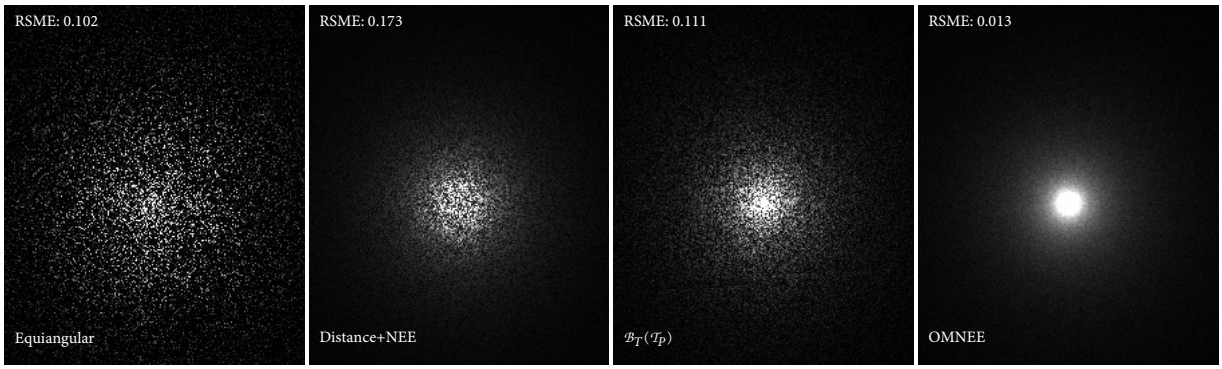


Figure 6.4: *Reproduced from Hanika et al. [2022]. A unit test with an infinite homogeneous medium and a point emitter, mean cosine $g = 0.9$. Here, only forward scattering is enabled and 4 samples per pixel are simulated. The images show only three path vertices, i.e. single scattering in the medium. The reference for the RMSE numbers was distance sampling with next event estimation at 10k samples per pixel. $\mathcal{B}_T(\mathcal{T}_P)$ refers to Bézier warp for transmittance and Taylor expansion for phase function as in the work by Villeneuve et al. [2021].*

To arrive at the world space polar radius, it has to be multiplied by the scale factor $s = \|\mathbf{x}_0 - \mathbf{x}_2\|$ (see figure 6.3). The vertex area measure PDF and the estimator are

$$p(\mathbf{x}_1) = f_s(\mathbf{x}_1) \frac{s}{d_1^2 \cdot d_2^2} \frac{\sin\theta}{\theta} \tag{6.7}$$

$$\langle I \rangle = f_c(\mathbf{x}_0 \leftrightarrow \mathbf{x}_2) \cdot \frac{\theta}{s \sin\theta}. \tag{6.8}$$

Figure 6.4 shows a comparison of the results to other methods for a Henyey-Greenstein phase function with mean cosine $g = 0.9$. This estimator has a few great properties that make it really useful to sample this specific type of scattering event in highly forward scattering media. Most notably it manages to sample both squared distances and the phase function at $\mathbf{x}_1$. There are no assumptions on the phase function and the sampling routine for regular analog Monte Carlo simulation can be reused. Table 2 shows an overview of which terms can or cannot be sampled by different methods.

A few downsides remain. First, the derivation of the formulas seems to be unnecessarily complicated: the simple result suggests that there should be an easier way to derive it. Then, since the phase function sampling is usually performed in outgoing solid angle, there is a $\sin\theta$ term remaining in the estimator which does not cancel out. This comes from the fact that only the angular part (in one dimension) is used in the change of variables here. Since the setting is highly forward scattering, the term $\theta/\sin\theta$ is not really significant though. A larger limitation is due to the parameter t: figure 6.5 shows the geometry of backward scattering events, i.e. $\theta > \pi/2$ cannot be reached by the parameter $t \in [0, 1]$. The next section will summarise a few more possibilities to extend this approach, hopefully overcoming these limitations in the future.

6.5 Open Problems

Backward scattering The choice of the parameter t limits the OMNEE method strictly to the forward scattering hemisphere of the phase function. Picking another parameter independent of θ and φ which is able to characterise the location of $\mathbf{x}_1$ with respect to the two other vertices can solve this issue. While for the sole purpose of rendering PSF-style blurs of some path tracing effects it is sufficient to simulate one additional vertex, in the case of multiple scattering it becomes quite a burden to only consider one hemisphere: The other half needs to be covered by an additional sampling technique. For one highly forward scattering event there is very little energy in this backward half, and it can easily be recovered by any other standard technique (phase function sampling/analog Monte Carlo for instance).

Multiple scattering In multiple scattering cases, there will be a combinatorial explosion of possible paths that include at least one backward scattered hemisphere. It is thus a good idea to re-derive all equations with

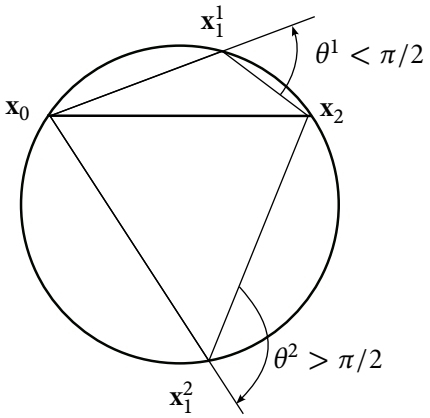


Figure 6.5: *Reproduced from Hanika et al. [2022]. Illustration of the extended geometry required to include backward scattering ($\theta > \pi/2$) in our framework. The part of the drawing above the $\mathbf{x}_0\mathbf{x}_2$ line is the case we consider: the centre of the circle is on the other side of the line as $\mathbf{x}_1^1$. If the centre is on the line, the resulting θ will be exactly $\pi/2$ everywhere on the circle. In the bottom part, $\mathbf{x}_1^2$ results in $\theta^2 > \pi/2$, but the angle θ will again be constant for all $\mathbf{x}_1$ on the lower arc. This arc unfortunately bulges outside the reach of our parameter t on both sides, so another parameterisation has to be chosen.*

Table 2: *Overview of the terms that individual techniques sample. The checkmarks in parentheses indicate that these cannot be done analytically but rely on approximation in the form of tabulation or polynomial fits. In the case of thin homogeneous forward scattering media, the phase function $f_s(\mathbf{x}_1)$ and the reciprocal squared distances d_1^{-2} and d_2^{-2} are of highest importance. Previous methods could not sample this combination analytically. Note that analog Monte Carlo (first row) naturally samples almost all terms encountered on the way, but fails to sample the emitter, which is a bad trade-off more often than not.*

	$f_s(\mathbf{x}_1)$	d_1^2	d_2^2	T_1	T_2	$f_s(\mathbf{x}_0)$	$L_e(\mathbf{x}_2)$
analog Monte Carlo	✓	✓	✓	✓	✓	✓	
OMNEE	✓	✓	✓				✓
std. NEE		✓		✓		✓	✓
Kulla and Fajardo [2012]			✓			✓	✓
Georgiev et al. [2013]	(✓)	✓	✓			(✓)	✓
Villeneuve et al. [2021]	(✓)		✓		(✓)		✓

respect to a distance parameter t^* that covers the full sphere. In addition, finding $\mathbf{x}_1$ from a full set of θ and φ and new distance constraints t^* is a bit more involved than simply evaluating a closed-form equation based on simple geometric considerations.

Constrained input and output directions Imagining for a moment that we had a system to sample full multiple scattering paths given a set of (θ, φ, t^*) coordinates, we would of course immediately be eager to apply it to one of the most complicated cases of participating media: skin rendering. In this setting the (forward scattering) medium is enclosed by a dielectric surface (the oily skin) which constrains the direction of the incident and outgoing rays by Fermat's law. Can we combine geometric and volumetric constraints?

Initial steps All the above considerations are currently being worked on by Urbach [2023] during his MSc. As a distance constraint, $t^* = d_1^2/(d_1^2 + d_2^2)$ is used. To solve for the world space Cartesian vertex coordinates $\mathbf{x}$ given a set of (θ, φ, t^*), Newton's method is used, similar to Jakob [2013]'s manifold walks to satisfy specular constraints. This only requires to compute the derivatives of (θ, φ, t^*) with respect to all $\mathbf{x}$. These are the same terms that make up the Jacobian determinant to be computed to transform sampling densities to path space. In addition, constrained directions on both sides of the sub path can be supported. These have to be considered while solving for the Cartesian coordinates and when evaluating the Jacobian.

Unfortunately, re-deriving equation (6.5), a sampling routine for t^* to analytically cancel the singularity caused by the square distances remains a hard problem: the PDF for t^* should be crafted such that, given all $p(\theta)$ (and $p(\varphi)$) and the Jacobian determinant $|J|$, it would cancel the contribution of the inverse square distances in the measurement. Now since the Jacobian depends on path length and is hard to grasp in analytical form, it seems intractable even getting to the target PDF in closed form, let alone integrating and inverting it. Initial experiments, however, seem to indicate that empirically the distribution of weights is relatively flat even for a simple uniform distribution of t^*.

Embed into other path space methods Since we can evaluate the vertex area measure PDF of our newly constructed path, it is possible to include it in a multiple importance sampling mixture. The once-more collided next event estimation technique worked really well at reducing variance even when applied in path tracing, i.e. connecting to light sources. This is slightly surprising since the blurring effect is even more subtle in this case, but figure 6.2 may deliver an explanation with the two distinct peaks that are necessary to sample. With this in mind maybe it would be time to renew the comprehensive study of volume scattering techniques and their combination, as Křivánek et al. [2014] have done ten years ago.

References

Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. 2013. Joint importance sampling of low-order volumetric scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 32, 6 (Nov. 2013), 164:1–164:14. https://doi.org/10.1145/2508363.2508411

Johannes Hanika, Andrea Weidlich, and Marc Droske. 2022. Once-more scattered next event estimation for volume rendering. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 41, 4 (2022). https://doi.org/10.1111/cgf.14583

Wenzel Jakob. 2013. *Light Transport on Path-Space Manifolds*. Ph.D. Dissertation. Cornell University.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 33, 4 (2014), 1–13.

Christopher Kulla and Marcos Fajardo. 2012. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 31, 4 (June 2012), 1519–1528. https://doi.org/10.1111/j.1467-8659.2012.03148.x

Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. 2012. Reconstructing the Indirect Light Field for Global Illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 31, 4, Article 51 (jul 2012), 10 pages. https://doi.org/10.1145/2185520.2185547

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum (Eurographics State of the Art Reports)* 37, 2 (2018), 1–26. https://doi.org/10.1111/cgf.13383

Yannick Urbach. 2023. Constrained path sampling for forward scattering media. Personal communication about his ongoing master's thesis. (2023).

Keven Villeneuve, Adrien Gruson, Iliyan Georgiev, and Derek Nowrouzezahrai. 2021. Practical Product Sampling for Single Scattering in Media. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20211290

7 Rendering of FX elements

Manuele Sabbadin, *Wētā Digital - Unity*

Figure 7.1: *Water simulations yield a diverse range of visual effects (FX) elements. Even in the most straightforward scenario, such as a wave colliding with a rock, distinct water structures emerge. The main bulk of water splits into blobbies, water droplets and transitions into a gaseous volume. Additionally, on the water surface, the presence of foam and bubbles can be observed, which results from the entrapment of air pockets within the water.*

7.1 Introduction

Water is an element intrinsically hard to manage in a pathtracer since the interaction between light and water particles can lead to long and complex paths. Due to its low roughness, only a few of these paths carry enough energy to create interesting visual effects, such as caustics and bright highlights on the surface. Most of the time, these highlights are fundamental to being able to read the shape of the water elements and their high-frequency features. This becomes especially true in a VFX scenario, where the CG water has to blend seamlessly with the real footage and ad-hoc light setups are prepared to achieve the required artistic look. If highlights are missed or represented incorrectly, artists will not be able to address the director's vision of the final frame. To make it even more challenging, water simulations usually produce millions of particles, each of them with its distinct motion.

While rendering relatively calm water is per se a challenge, fast-paced scenes introduce other factors of complexity. If we consider, for example, a scene with fast-moving waves hitting some rocks, we can observe different events: some of the particles might intersect with others and merge with them, while other particles can split into smaller ones due to the interaction with the solid rock (Fig. 7.2). Such behaviors pose a question of how we should represent each particle, as the use of a fixed topology throughout the entire frame can lead to undesired artifacts. In addition, when fast water elements interact with each other, it is common to find other FX elements hard to render, such as foam and underwater bubbles.

Finally, we should not forget the memory footprint used to represent any FX element, since it can easily outgrow the available system memory.

Figure 7.2: *Most of the FX elements mentioned in Sec. 7.2 are visible here. In particular, towards the left of the image, the splash creates a vertical structure that we identified as blobbies. Surrounding this structure, some disjoint particles are visible. In this case, we are referring to them as water droplets.*

In the following, we will describe our experience in managing such elements, and the different techniques that we have applied to reach the required visual appearance with an affordable amount of render time and memory. After a description of the different FX elements arising from a water simulation, we will first discuss how filtering the normal distribution function can help in retrieving highlights in fast-motion scenes. We will show how the same technique can be used to facilitate the sampling in out-of-focus regions. Secondly, we will discuss the design and use of some ad-hoc primitives, such as bubbles and blobbies, which have been extensively used to blend the main bulk of water with the small isolated particles of foam around it. They have shown to be effective in regions where the structure and topology of the water simulation are more subject to quick changes. Finally, we will talk about the conversion from a set of particles to a volume, which can be a practical way to reduce the number of particles to consider during light transport, and it is a viable solution whenever the particle size is small enough in comparison to the pixel footprint.

7.2 FX water elements

Water simulations yield a diverse range of visual effects (FX) elements. Even in the most straightforward scenario, such as a wave colliding with a rock, distinct water structures emerge. It is crucial to classify and discern the unique attributes of each element to develop tailored techniques that streamline the rendering process. We have found it beneficial to categorize these elements based on various characteristics, including their size, particle density within a defined area, shape, and interaction with air (e.g., presence of air encapsulation). Based on this, we individuated the following FX elements: bulk of water, blobbies, water droplets, volume, bubbles and foam. Most of them are visible in Fig. 7.1 and Fig. 7.2.

In the following, we will give a brief description of each of them. While we will outline the general behaviour of each component, it is up to the FX artist or simulation engine (in our case Loki Lesser et al. [2022]), to split them into different elements to pass to the renderer.

In our analysis, we define the *bulk* of the simulation as the primary body of water, which typically gives rise to all other related elements. This component can manifest in diverse forms and generally encompasses a substantial number of pixels. The density of water particles within the bulk is typically at its maximum, with minimal to no encapsulated air relative to the volume of water. While the simulation engine may adopt various representations of the bulk, our approach involves tessellating

it and depicting it as a polygonal mesh. This categorization encompasses the principal ocean surface as well as its associated waves.

During the occurrence of a water splash, a significant transformation of the bulk of water takes place, transitioning it from a liquid state to a gaseous *volume* consisting of air and minute water particles. In this context, the density of particles reaches its minimum level, as they tend to become disassociated from one another. Each individual particle's size is negligible, whereas the overall volume typically encompasses multiple pixels on the screen. Internally, this element can be represented either as a homogeneous or heterogeneous volume, since the individual water particles are too minuscule to be depicted in any other form. This particular effect becomes more prominent in larger splashes, such as those observed around waterfalls.

In the transitional region between the bulk of water and the gaseous volume, additional FX elements become apparent. As previously mentioned, the distinction between these states is not clearly defined, and often the FX artist or simulation engine must make a decision regarding their categorization. Closest to the primary water surface, we encounter thin water structures characterized by high dynamism and rapidly changing topology. We refer to these elements as *blobbies*. Although blobbies have the potential to encapsulate pockets of air, they generally exhibit a significant density similar to that of the primary bulk of water. The primary differentiation between the bulk and blobbies lies in their nature: the bulk represents a large, cohesive water volume, while blobbies consist of dynamically evolving clusters of water particles. Additionally, blobbies can manifest as thin layers of water surrounding objects emerging from the primary water surface.

In the transitional phase between blobbies and the volume, we encounter clusters of water particles that possess insufficient size to form intricate structures like blobbies, yet are too substantial to be approximated as a gaseous volume (we will delve into this topic later, as there are cases where this approximation makes sense and is feasible). Due to their spherical shape, we will simply refer to them as *water droplets*. Water droplets are characterized by a relatively small size, and we have found that rendering them as spheres preserves the desired level of detail in the final image. Similar to the volume case, these elements become more pronounced in larger splashes, such as those found in waterfalls.

At this point, it is important to acknowledge that while we have presented each FX element as a distinct stage in the transition from the primary water bulk to the gaseous volume made of particles further away from the main splash, it is common for these elements to coexist and overlap within the same spatial region. This is especially true for blobbies, water droplets and volume.

Up until now, we haven't addressed a definitive categorization based on the interplay between air and water. Let's focus on elements characterized by the presence of air, with the most prevalent case being *bubbles*. It's important to note that bubbles can be further differentiated as either underwater or surface bubbles. In the former case, the air pocket is fully submerged within the primary water surface, while in the latter case, the bubble rests atop the surface with a thin layer of water encapsulating the air. Both types can emerge from a water splash, as the previously discussed water elements recombine with the main water surface, trapping air pockets in the process. When the size of a bubble is substantial enough to be visible across multiple pixels, we will refer to it as a bubble element. Otherwise, when multiple surface bubbles of minimal size cluster together, we will refer to them as *foam*.

7.3 NDF filtering

The appearance of vast water surfaces is subject to scale-dependent variations. When observed up close, small and distinct highlights are typically noticeable, while from a greater distance, they tend to appear larger and more diffused. It is rare to encounter perfectly flat water surfaces in nature. The ocean's surface, for instance, can be approximated using Fast Fourier Transform (FFT) techniques that incorporate multiple frequencies Mastin et al. [1987]. When observing the same water plane

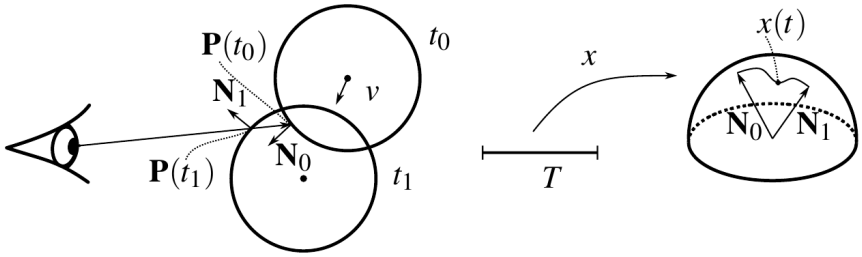
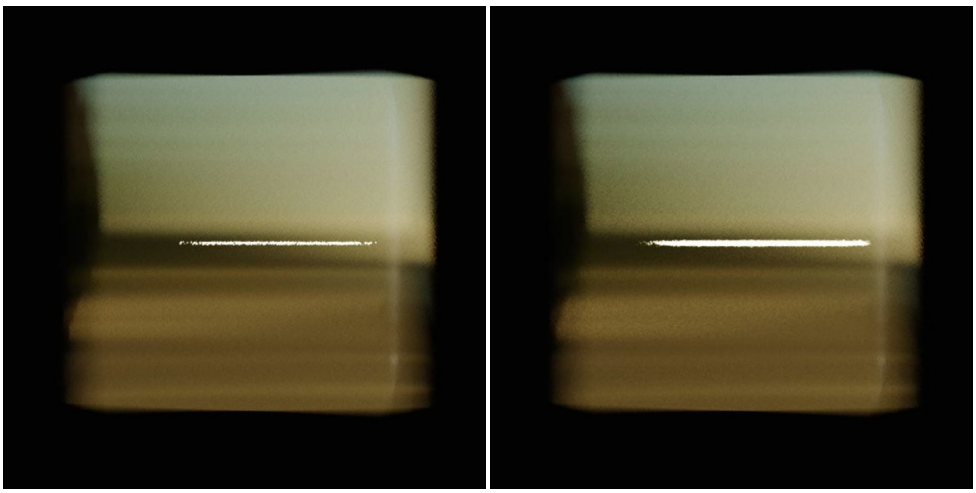


Figure 7.3: *Left: change of normals* **N** *at intersection point* **P** *depending on time* t*. Right: Resulting map* $x(t)$ *from time domain* T *to reference unit sphere.*



(a) *Motion roughness off* **(b)** *Motion roughness on*

Figure 7.4: *Motion roughness helps resolving highlights on specular moving objects. In the above example, a golden cube is rotating along its vertical axis. We used the same low number of progressions (32) to render the two spinning cubes.*

from different distances, an identical solid angle will produce patches of varying sizes on the water surface. As the surface area expands, the variation in normals within it becomes more pronounced. Consequently, highlights appear increasingly blurred in the distance. In order to maintain the fidelity of highlights across all scales and mitigate aliasing artifacts, numerous methodologies have been devised, such as prefiltering of the normal map in LEAN mapping Olano and Baker [2010] or prefiltering of the BSDF in the half vector domain Kaplanyan et al. [2016], Tokuyoshi and Kaplanyan [2019]. These methods operate on the principle of mapping the variation in normals within a specific region onto the roughness attribute, which governs the distribution function of the material's normals.

Less attention has been posed to the distribution of normals due to motion and amount of defocus on the image plane. In the following sections we will discuss how we can take advantage of both these effects to further filter the normal distribution function.

7.3.1 Roughness due to motion

Since water is never static, highlights tend to appear and disappear continuously at different locations on the surface. The water elements previously mentioned, in particular, are frequently associated to highly dynamic setups, where each frame carries a massive amount of particles in motion. In path

tracing, being able to resolve a specular highlight under this conditions is a hard task. Depending on the amount of motion involved, it might take thousands of progressions to be able to converge the final render to a noise-free image. This is due to the fact that samples are spread within the frame duration, and only a few of them will land on the highlight.

As shown in Fig. 7.3, given a time domain T, the normal visible at the intersection point $\mathbf{P}(t)$ varies with time, and can be mapped to a cone of normals through $x(t)$. We used the approach described by Tessari et al. Tessari et al. [2020] to prefilter the NDF of the underlying material in the temporal domain. Let's consider an anisotropic Beckmann distribution with roughness values σ_u and σ_v along the tangent axis, its covariance matrix can be written as:

$$B = \begin{pmatrix} \alpha_u^2 & 0 \\ 0 & \alpha_v^2 \end{pmatrix} \tag{7.1}$$

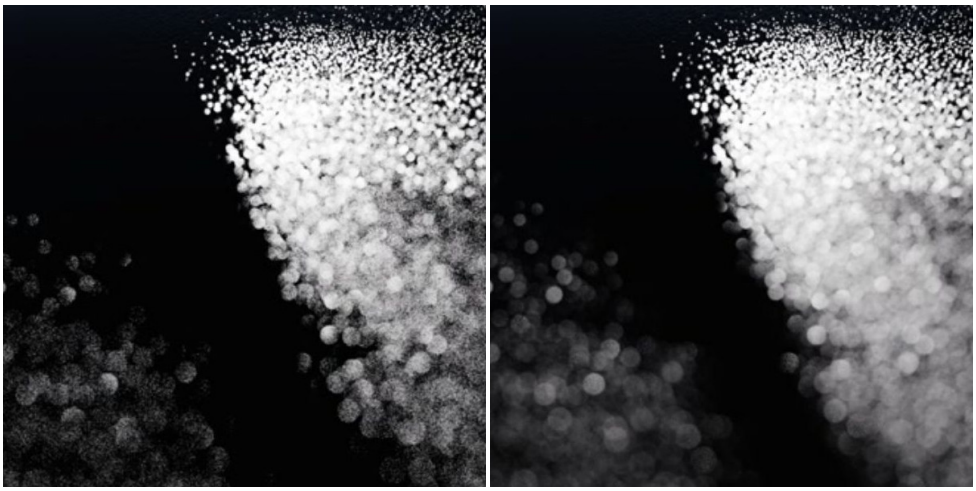
where $\alpha = \sqrt{2}\sigma$ to have a standard Gaussian covariance matrix. As shown in Kaplanyan et al. [2016], we can prefilter it by adding a second Gaussian covariance matrix, namely K, which represents the variation of the normals under the pixel footprint in the slope domain. To extend Kaplanyan et al. [2016] to the temporal domain, Tessari et al. Tessari et al. [2020] considered the variation of the normal in a temporal neighbourhood of the time t, instead of the pixel footprint. To compute the values in K, the main quantity involved is represented by the total derivative of the normal at the intersection point $\mathbf{P}(t)$. The total derivative is then used as a first order approximation to extrapolate the normal away from the center (at $t + \Delta t$). In the paper, the authors show how to compute the above-mentioned quantity:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{N}_\varphi(r(t),t)}{\mathrm{d}t} &= \frac{\mathrm{d}\mathbf{N}_\varphi(\mathbf{o}(t)+s(t)\mathbf{d}(t),t)}{\mathrm{d}t} = \\ &= \frac{\partial \mathbf{N}_\varphi(r(t),t)}{\partial t} + \nabla_x \mathbf{N}_\varphi(r(t),t)\left(\frac{\partial \mathbf{o}(t)}{\partial t} + \frac{\partial s(t)}{\partial t}\mathbf{d}(t) + s(t)\frac{\partial \mathbf{d}(t)}{\partial t}\right) \end{aligned} \tag{7.2}$$

where $r(t) = (\mathbf{o}(t), \mathbf{d}(t))$ is the ray with origin in $\mathbf{o}(t)$ and direction $\mathbf{d}(t)$, with t being the time. We refer to Tessari et al. Tessari et al. [2020] to derive Eq. 7.2 and all the partial derivatives involved. In here, it is worth mentioning that it involves computing both spatial and temporal partial derivatives of the function φ, where φ implicitly defines the local geometry. For a generic polygonal mesh, this is possible by locally approximating it with a quadratic patch, and considering only changes due to rotation and translation (deformation and bending are not considered). Later, in Sec. 7.4, we will detail a general framework to compute such derivatives when the underlying φ is a known implicit function, where we can exploit its analytical form to derive precise values.

7.3.2 Implementation details

In the implementation of the motion roughness algorithm, our aim was to provide artists with extensive control over its application. While the algorithm allows for roughness to be applied to both direct and indirect bounces, we have observed that in many common scenarios involving water elements, it is preferable to restrict the method to only the first bounce. Several factors contribute to this choice. As highlighted in the original paper, the propagation of ray differentials beyond the first bounce can lead to excessive spreading, necessitating clamping. Moreover, while direct highlights can be resolved more rapidly thanks to the additional roughness, it is worth noting that some indirect highlights, which were previously challenging to detect (because generated by long paths), may start to appear in the final render. Addressing this new set of highlights would still require a significant number of progressions, whereas for artistic reasons, the emphasis may be on preserving the sharpness of the direct highlight alone. We have incorporated adjustable settings to regulate the number of vertices that undergo roughening along the path, along with a pair of thresholds to limit the final roughness value. One of these thresholds exclusively applies to the initial vertex along



(a) *Defocus roughness off* **(b)** *Defocus roughness on*

Figure 7.5: *Defocus roughness helps resolving highlights in areas which are out of focus. In the above example, the depth of field is quite shallow and the focus plane far from the camera. We used the same low number of progressions (512) to render the two images. Worth noticing that the defocus roughness algorithm does not add any roughness where the image is in focus.*

the path, and is typically set to a small value to retain maximum sharpness. The second threshold is employed for all subsequent vertices.

Another implementation choice we had to make is about the temporal footprint we want to consider to compute the variation of the normals. In the original paper, this is identified by Δt and used to multiply $\frac{\mathrm{d}\mathbf{N}_\varphi(r(t),t)}{\mathrm{d}t}$, in order to compute the normal at the boundary of the footprint. The variation of the normal is locally approximated by a first-order Taylor polynomial. As such, the approximation is reliable only in a small neighbourhood of t_0. Since water splashes usually exhibit high dynamicity and fast moving particles, we choose to set Δt to a default which is a very small fraction of the shutter range. While this worked properly in most of our scenes, we leave as a possible future work to dynamically scale the temporal footprint, based on the velocity of the local geometry.

At last, since water is typically highly specular, and the roughness added by the motion filter tends to be highly anisotropic, we want to avoid cases where we add roughness only on one component of our local frame, leaving the other unchanged. This could negatively affect the sampling of the underlying BSDF, as it would still exhibit a delta distribution along one of the axis. To mitigate this, we ensure that a minimum level of roughness is present on both axes at the end of the process. This can be achieved by setting an absolute minimum threshold or by employing a minimum ratio with respect to the major axis, as illustrated in Fig. 7.4b. The latter approach preserves the shape of the highlight but may result in thicker highlights if not appropriately adjusted.

7.3.3 Roughness due to defocus

In a similar manner to our approach for motion filtering, we can also apply a filter to the roughness of the Beckmann distribution when the intersection point is out-of-focus. To accomplish this, let M be a parametric surface, and consider a specific point $\mathbf{p} \in M$. At this point, the shape operator $\mathbf{S}$ characterizes the local variation of the normal with respect to the tangent frame. The eigenvectors of $\mathbf{S}$ correspond to the principal directions of curvature. Let $\mathbf{e}_1$ and $\mathbf{e}_2$ denote the eigenvectors of $\mathbf{S}$. We define the radius of the circle of confusion at $\mathbf{p}$ as r and scale $\mathbf{e}_1$ and $\mathbf{e}_2$ accordingly. This has the effect of scaling the tangent frame to include the entire area covered by the circle of confusion. Subsequently, the shape operator is utilized to compute the variation of the normal along the scaled vectors $\mathbf{e}_1$ and $\mathbf{e}_2$:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{N}_\varphi}{\mathrm{d}u} &= \mathbf{S} \cdot \mathbf{e}_1 \\ \frac{\mathrm{d}\mathbf{N}_\varphi}{\mathrm{d}v} &= \mathbf{S} \cdot \mathbf{e}_2 \end{aligned} \tag{7.3}$$

Analogous to the motion case, we multiply the derivative by a delta to reduce the footprint to a local neighbourhood of the intersection point. This information enables us to construct the covariance matrix, representing the normal variation within the region covered by a portion of the circle of confusion, and proceed with the convolution process, akin to the previous case. To simplify the filtering process, we enforce isotropy in the filter. In our implementation, we opt to consider the added roughness on the minor axis and apply it uniformly to both directions. In addition to this, we restrict this filter only to direct highlights.

7.4 FX primitives

Filtering of the normal distribution function, as discussed in the previous section, heavily depends on the quality of the derivatives at each intersection point. Poor temporal derivatives introduce more approximations into the final result. When dealing with polygonal meshes, all we can do is to use the local curvature to approximate the local geometry to a patch of second order. It would be ideal if we could analytically compute the exact derivative at each given point.

During the process of rendering water, we encountered another challenge related to the changing topology in fast-paced scenes. Water particles, due to their liquid form, have a tendency to separate and combine with one another. This phenomenon becomes particularly noticeable in Fig. 7.2, where waves crashing against rocks generate splashes. The particles within a splash often separate from each other, while simultaneously merging with particles from neighboring splashes. When multiple particles interact within a single frame, it becomes necessary to update the topology of the resulting mesh. However, because we typically work with a fixed topology per frame, this can result in potential artifacts.

In order to give an answer to both the above-mentioned issues, we designed some ad-hoc primitives, in the form of implicit surfaces. The mathematical nature of such primitives allows us to compute derivatives in an analytical way. Moreover, there is no topology to take into account, since we directly trace rays against them.

7.4.1 Isosurfaces

Consider a function $\varphi : \mathbb{R}^3 \to \mathbb{R}$ and a constant value C. An isosurface (in the following we will use the terms isosurface and implicit surface as synonyms) can be defined as the collection of points denoted by $M := \{(x, y, z) \,|\, \varphi(x, y, z) = C\}$. Without loosing generality, we will always consider the case where $C = 0$. If $C \neq 0$, we can define φ_2 as $\varphi_2 = \varphi(x, y, z) - C$, achieving the same isosurface with $M := \{(x, y, z) \,|\, \varphi_2(x, y, z) = 0\}$. With this mathematical definition, it becomes feasible to analytically compute the derivatives of the function φ itself or any other function defined on the isosurface.

As discussed in Section 7.3.1, the requirement for motion roughness entails accurate derivatives in both the spatial and temporal domains. To facilitate the discussion, we will redefine φ as a function from $\mathbb{R}^4 \to \mathbb{R}$, by adding the time component: $\varphi(\mathbf{x}, t)$, where $\mathbf{x}$ is a point in three dimensions and t denotes time. While we could embed any variation of the isosurface due to time directly in φ, we found useful to separate linear transforms, such as rotations and traslations, into a different function. This leads to our generic definition of φ as $\varphi(\mathbb{A}(\mathbf{x}, t), t)$, where $\mathbb{A}$ denotes a time-dependent linear transform of the point $\mathbf{x}$, i.e., $\mathbb{A}(\mathbf{x}, t) = \mathbb{A}(t)\mathbf{x}$. The main derivatives required by our motion roughening technique can then be expressed as:

$$\nabla_{\mathbf{x}}\varphi(\mathbb{A}(\mathbf{x},t),t) = \frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\frac{\mathrm{d}\mathbb{A}(\mathbf{x},t)}{\mathrm{d}\mathbf{x}} = \frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\mathbb{A}^T(t) \tag{7.4}$$

$$\frac{\mathrm{d}\varphi(\mathbb{A}(\mathbf{x},t),t)}{\mathrm{d}t} = \frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\frac{\mathrm{d}\mathbb{A}(\mathbf{x},t)}{\mathrm{d}t} + \frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial t} \tag{7.5}$$

$$\frac{\mathrm{d}\nabla_{\mathbf{x}}\varphi(\mathbb{A}(\mathbf{x},t),t)}{\mathrm{d}\mathbf{x}} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\left(\frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\right)\mathbb{A}^T(t) = \mathbb{A}\frac{\partial^2\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)^2}\mathbb{A}^T(t) \tag{7.6}$$

$$\frac{\mathrm{d}\nabla_{\mathbf{x}}\varphi(\mathbb{A}(\mathbf{x},t),t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\right)\mathbb{A}^T(t) + \frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\frac{\mathrm{d}\mathbb{A}^T(t)}{\mathrm{d}t} \tag{7.7}$$

$$\text{where}\quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)}\right) = \frac{\partial^2\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)^2}\frac{\mathrm{d}\mathbb{A}(\mathbf{x},t)}{\mathrm{d}t} + \frac{\partial^2\varphi(\mathbb{A}(\mathbf{x},t),t)}{\partial\mathbb{A}(\mathbf{x},t)\partial t} \tag{7.8}$$

To intersect the isosurface, we define $r = (\mathbf{o}, \mathbf{d})$ as the ray with origin in $\mathbf{o}$ and direction $\mathbf{d}$ with limits $[s_{\min}, s_{\max}]$, parameterized as $r(s) = \mathbf{o} + s\mathbf{d}$. We want to find $s_{\text{int}} \geq s_{\min}$, that is the minimum s such that $\varphi(\mathbb{A}(r(s_{\text{int}}),t),t) = 0$.

The simplest case arises when we can analytically solve the following equation in s_{int}:

$$\mathbf{o} + s_{\text{int}}\mathbf{d} = \varphi(\mathbb{A}(r(s_{\text{int}}),t),t) \tag{7.9}$$

This is a valid option for simple definitions of φ. For example, it is well known how to intersect a ray with a sphere. However, if Eq. 7.9 is not directly approachable, we can use more general methods, such as ray marching Perlin and Hoffert [1989] or sphere tracing Hart [1996]. Isosurfaces can also be converted to polygonal meshes using techniques such as marching cubes Lorensen and Cline [1987], marching tetrahedra Treece et al. [1999] and dual contouring Ju et al. [2002]. Although converting isosurfaces to meshes is a viable solution, we restrict our work to methods to directly intersect them. In the following, we will introduce the two primitives we implemented to facilitate the render of FX water elements: bubbles and blobbies. For each of them, we will briefly discuss how the intersection detection works.

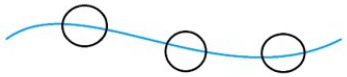
7.4.2 Bubbles

Let's start by examining the *bubble* primitive, which has been designed to resemble the shape presented in the work from Teixeira et al. Teixeira et al. [2015]. The idea behind this primitive is to represent air bubbles of significant size on the water surface (as opposed to smaller bubbles, which are the components of white foam). Previously, the main approach to create bubbles was by scattering points on the surface and placing spheres at those positions. However, this method resulted in unrealistic bubble shapes, as depicted in Fig. 7.6a, because they appeared spherical even at the bottom, where they should be flattened. A second approach involved using the volume of water to "cut" the spheres, so they have a better shape (see Fig. 7.6b). However, this introduces other problems: imagine a shading primvar defined on the water surface, i.e., to guide the appearance of the finer *foam* surrounding the main bubbles. We would like the bubbles of air to remove the foam from the water surface where they are sitting. By using the water volume to shape the bubbles, this cannot happen. To address these challenges, we decided to create an implicit function with a small number of parameters that closely resembled the shape of a bubble (see Fig. 7.6c). By instantiating this function on the water surface, we could use it to "cut" any primvar defined on the surface, achieving the desired visual effect.

To represent our bubble we used the following implicit function:

$$b(x,y,z) = \begin{cases} r\cdot x^2 + h_1\cdot y^2 + r\cdot z^2 - 1 & \text{if } y \geq 0 \\ r\cdot x^2 + h_2\cdot y^4 + r\cdot z^2 - 1 & \text{otherwise} \end{cases} \tag{7.10}$$

where r is the radius of the bubble, h_1 is the height of the upper hemisphere, and h_2 the height of the bottom hemisphere. The bubble is the composition of a quadratic and a quartic surface, one on

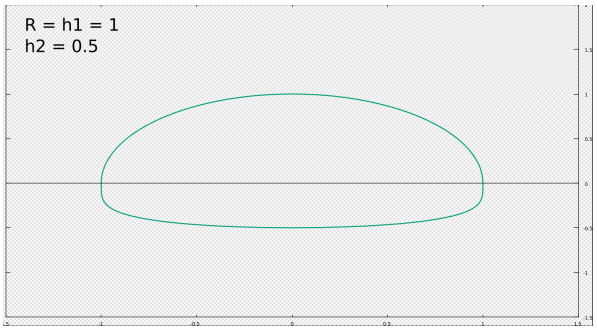


(a) *By using spheres to represent the bubble, we have a poor representation of its lower half, which should be more flat.*

(b) *In yellow, we represent a generic primvar, defined over the water surface. We would like the primvar to be removed in the presence of a bubble.*

(c) *The bubble primitive allowed us to achieve a more realistic look and remove any primvar from the water surface at the same time.*

Figure 7.6: *The three illustrations show some of the problems we were targeting with the design of an ad-hoc bubble primitive.*



(a) *2D slice of the bubble primitive. Here we indicate the radius on the horizontal axis as R. h_1 and h_2 are the heights of the upper and lower hemispheres, respectively.*

(b) *The bubble primitive is rendered in isolation, without water volume. A water material has been applied to the primitive.*

Figure 7.7: *The bubble primitive is represented by an implicit function, where the upper half is a quadratic surface, and the bottom half a quartic surface.*

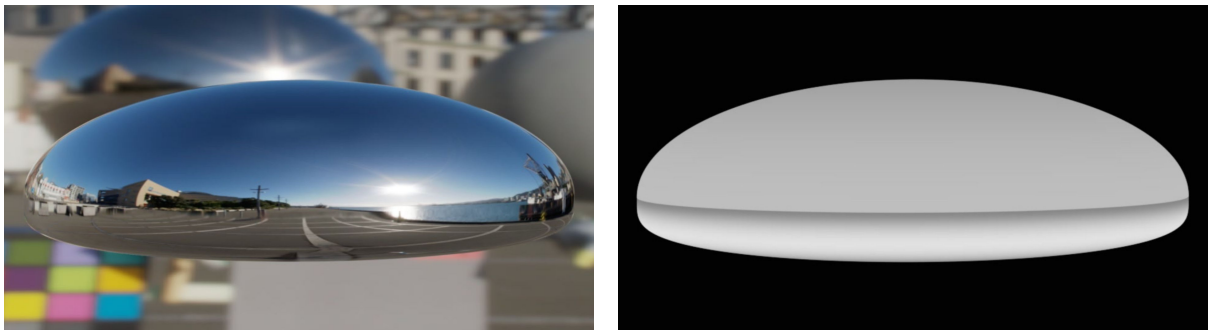
top of each other. A 2-dimensional slice of the bubble is shown in Fig. 7.7a, while Fig. 7.7b is the rendering of a single bubble with a glass material applied.

While, by design, we made sure that both the function $b(x, y, z)$ and its first order derivatives exist and are continuous, we did not provide such guarantee for its second order derivatives. Therefore, the curvature along the surface is not a continuous function and reflections can appear broken at $y = 0$ (see Fig. 7.8). In practice, due to motion and the fact that this part of the bubble is mostly submerged by water, this hasn't been a problem.

Due to the composition of the bubble as two distinct functions, the process of finding the initial intersection along the ray has been divided into two separate components. First, we analytically determine the intersection with the upper hemisphere, denoted as s_{up}. Next, we calculate the intersection with the lower hemisphere, denoted as s_{lo}. Consequently, we set $s_{\text{int}} = \min(s_{\text{up}}, s_{\text{lo}})$. It is assumed that $s_{\text{up/lo}}$ exist and are well-defined, meaning that each intersection point effectively lies on its hemisphere. If this condition is not met, the corresponding value is set to *FLT_MAX*. A successful intersection is identified if $s_{\text{int}} < FLT\_MAX$. We decided to avoid the analytical resolution of a fourth degree polynomial, needed to intersect the lower half, since it involves computationally expensive operations. Instead, we addressed the problem by employing a method similar to the one described by Yuksel Yuksel [2022]. This approach involves isolating an interval with a single root, accomplished by solving the cubic equation resulting from setting the gradient to zero, and subsequently applying a bisection method within the interval.

7.4.3 Blobbies

Blobby surfaces, as first introduced by Blinn Blinn [1982], offer an analytic definition of an isosurface based on the combination of kernel functions around given particles (see Fig. 7.9). The resulting



(a) *By applying a mirror material to the bubble primitive, discontinuities on the surface curvature are easier to spot. They appear where lower and upper hemispheres meet, for* $y = 0$.

(b) *This image represents the surface curvature of the bubble primitive. Since the function* $b(x, y, z)$ *is not* C^2*, the curvature is not continuous on the plane with* $y = 0$.

Figure 7.8: *The bubble primitive is not* C^2 *by design. In practical use of the primitive, this has not been a problem, since the discontinuous part is usually submerged in water.*

Figure 7.9: *Water splashes produce a variety of FX elements. Among them, blobbies structures are characterized by dynamically evolving clusters of water particles. Moreover, they can form thin layers or stripes of water.*

surfaces are smooth and are nicely compactly represented by points with some parameters. However, their wobbly appearance makes it difficult to represent thin structures which are typically generated by water splashes. The definition of blobbies extends easily to anisotropic surfacing Yu and Turk [2013], which overcomes these issues and makes them a compelling modeling representation for various forms of splashes and fluid droplets. We extensively covered our implementation of blobbies in Sabbadin and Droske [2021]. Here we will give a summary of it.

A blobby particle B_i is represented by an implicit field $\psi_i : [t_0, t_1] \times \mathbb{R}^d \to \mathbb{R}$ defined over the time interval $[t_0, t_1]$ and space. A set of blobbies defines an implicit field $\varphi(\mathbf{x}, t)$ in the following way:

$$\varphi(\mathbf{x}, t) := \sum_i \psi_i(\mathbf{x}, t) - T \tag{7.11}$$

where T is a threshold parameter that influences the blending of the individual blobbies. We focus on the classic blobby variant

$$\psi_i(\mathbf{x}, t) = \begin{cases} (1 - R_i^{-2}\|\mathbf{x} - \mathbf{x}_i(t)\|^2)^3 & \|\mathbf{x} - \mathbf{x}_i\| < R_i \\ 0 & \text{otherwise} \end{cases} \tag{7.12}$$

where $\mathbf{x}_i(t)$ defines the center of the particle at time t and R_i the radius of the influence region. We call R_i the *bounding radius* of the blobby B_i.

This can easily be generalised to anisotropic particles, by writing it in the form

$$\psi_i(\mathbf{x}, t) = k(g(\mathbf{x} - \mathbf{x}_i(t), \mathbf{x} - \mathbf{x}_i(t))) \quad \text{where} \quad k(y) := (1 - y)^3 \tag{7.13}$$

and g is a scalar-product $g_A(\mathbf{u}, \mathbf{v}) := \langle A\mathbf{u}, A\mathbf{v} \rangle$ that encodes the anisotropy and size. In particular, for a unit basis $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ and radii R_0, R_1, R_2 we can set $A_i := \text{diag}(\frac{1}{R_0}, \frac{1}{R_1}, \frac{1}{R_2}) \cdot [\mathbf{b}_0\, \mathbf{b}_1\, \mathbf{b}_2]^T$. The values R_i can be easily chosen depending on T such that the isosurface of an isolated blobby describes and ellipsoids with the prescribed lengths r_i of the axes. In the following, we will refer to r_i as the *inner radius* of the blobby B_i.

To efficiently handle blobbies and identify particles that intersect with a ray segment, we employ a classic Bounding Volume Hierarchy (BVH) data structure. Each particle is associated with its bounding radius R_i, inner radius r_i, velocity, and acceleration, enabling representation of higher-order motion blur. It is crucial to ensure the creation of tight bounding boxes for both leaf nodes and inner nodes of the BVH. Rather than using a single large bounding box that encompasses the entire motion of an individual particle, we leverage Bézier curves to represent the particle's motion. Since the value of a Bézier curve is given by linearly interpolating its control points twice, the result is a second-order polynomial of the form:

$$\mathbf{B}_i(t) = (1 - t)^2\mathbf{P}_{i,0} + 2(1 - t)t\mathbf{P}_{i,1} + t^2\mathbf{P}_{i,2} \tag{7.14}$$

where $\mathbf{P}_{i,0}, \mathbf{P}_{i,1}, \mathbf{P}_{i,2}$ are the three control points for the curve B_i. We can equate Eq. 7.14 to the parabola describing the motion of the i-th particle to the determine the value of the three control points. This approach enables us to establish tight bounding boxes for each time step t within the range of motion.

In the case of blobbies, the absence of a simple analytical form hinders the direct computation of intersections with a given ray. Consequently, we have opted to employ traditional root-finding methods as an alternative approach. The key requirement is to identify an interval, denoted as I, along the ray where the function $\varphi(\mathbf{x}, t)$ assumes both positive and negative values and exhibits monotonic behavior. Achieving this necessitates evaluating the function $\varphi(\mathbf{x}, t)$ multiple times, which can be computationally expensive due to the considerable number of particles typically involved, ranging in the order of thousands or even millions. To mitigate this computational burden, we exploit the finite support of the kernel function. Consequently, not all particles need to be considered for evaluating the function at a given position $\mathbf{x}$. By selectively limiting the number of particles utilized in our iterations to the minimum required, we reduce computational overhead. Subsequently, we determine an initial estimate for the interval I, using only this subset of *active* particles. Following this, we refine the interval in a front-to-back order, and ultimately identify the root within the refined interval.

We refer the reader to Sabbadin and Droske [2021] for the detailed description of how to compute the set of *active* particles. At a high level, the procedure works by discerning cases where the ray is hitting the isosurface from outside or inside. Moreover, we exploit the intersections with *bounding* and *inner* spheres as a criteria to stop the BVH traversal.

To guarantee an interval I_i to contain a single root it is sufficient for φ to be monotone and the two extremes of the range B_I to have different sign. Therefore, to isolate the roots, we successively refine the interval I until an interval is reached in which $s \mapsto \varphi(r(s), t)$ is not bounded away from zero and is monotone (can't contain multiple roots), i.e., the derivative with respect to s is guaranteed not to attain zero. The interval refinement can be done by bisection or an *Interval Newton* method Caprani et al. [2000] which uses the bounds of the derivatives in the refinement process.

Therefore, for a given ray $r(s) := \mathbf{o} + s\mathbf{d}$ the intersection distance interval relies on computing the bounds of $f(s) = \varphi(r(s), t)$ and its derivative $f'(s)$ in an arbitrary subrange $[s_{\min}, s_{\max}]$. Of course, the efficiency of the refinement process depends on how tight the bounds are.

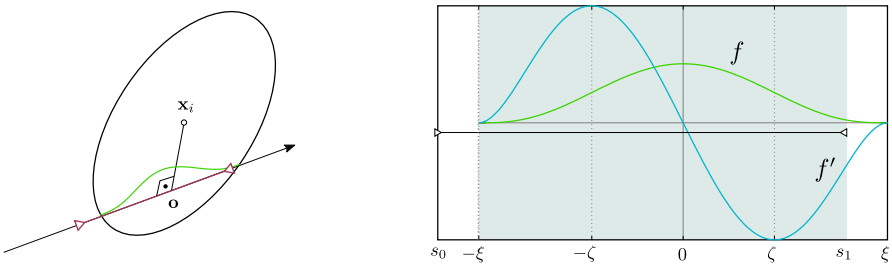


Figure 7.10: *Calculating bounds of ψ along the ray by identifying monotone intervals.*

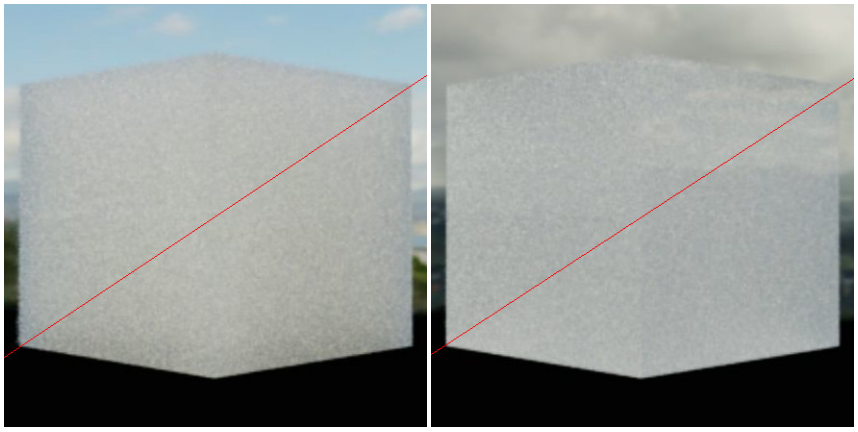


Figure 7.11: *The images depict the representation of water droplets as volumes under different lighting setups. In both cases, the top-left half of the image is rendered using multiple distinct water droplets, while the bottom-right half is rendered as a volume. Due to the small size of the original water droplets, the visual distinction between the two techniques is minimal.*

The interval bounds of $s \mapsto \varphi(r(s), t)$ is a simple additive composition of the intervals of the individual blobs (via the relation $[\underline{a}, \overline{a}] \oplus [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$). This composition (as are similar compositions for other operators) may loosen the bounds. However, there are two things we can pay specific attention to. Firstly, we want to compute tight bounds for the individual ψ_i. Secondly, to exploit that $\psi_i \geq 0$ for early termination of the check whether the bounds may contain 0.

To compute tight bounds for ψ_i, one can simply exploit the well-known fact that the bounds on a monotone function are given by the values at the end-points. We assume for simplicity that $\mathbf{o}$ is at the projection (with respect to g_A) of the center $x_i(t)$ to the ray:

$$g_A(\mathbf{x}_i(t) - \mathbf{o}, \mathbf{d}) = 0, \tag{7.15}$$

by computing the projection and shifting $[s_{\min}, s_{\max}]$ accordingly (see Fig. 7.10 left). Defining $\alpha := g_A(\mathbf{o} - \mathbf{x}_i, \mathbf{o} - \mathbf{x}_i)$ and $\beta := g_A(\mathbf{d}, \mathbf{d})$, we would then like to find the bounds of $f(s) = k(\alpha - s^2\beta) = (1 - \alpha - s^2\beta)^3$.

It can be easily seen that due to (7.15) the support of f is $[-\xi, \xi]$ with $\xi := \sqrt{(1-\alpha)/\beta}$. It is monotonely increasing in $[-\xi, 0]$ and decreasing in $[0, \xi]$ and furthermore has inflection points at $-\zeta, 0$ and ζ with $\zeta := \sqrt{1/5}\,\xi$ (see Fig. 7.10 right).

Therefore, computing the bounds on f in $[s_{\min}, s_{\max}] =: I$ amounts to evaluating at the end-points of the subintervals $[-\xi, 0] \cap I$ and $[0, \xi] \cap I$. Similarly, the bounds of f' are obtained by evaluating f' at the values $-\xi, -\zeta, 0, \zeta, \xi$ clipped to I.

7.5 Approximation to volume

In Section 7.2, we introduced the term *water droplets* as a general classification for FX elements that encompass a blend between the *blobbies* regime and the concept of *volume*. We also mentioned that water droplets are commonly represented as spheres. While this approximation deviates from the true shape, it has minimal impact on the final frame since droplet sizes are typically smaller than the pixel size. This approximation offers the advantages of accelerating the rendering process and reducing the memory footprint, as storing spheres is more efficient than polygonal meshes. However, tracing rays against millions of these tiny particles can be computationally demanding. If we consider the case of a splash far from the camera, although individual droplets are too small to contribute to visible features to the final image, their accumulation as an aggregation does. Therefore, our objective is to replace such droplet aggregations with a volume representation, further reducing storage requirements and intersection costs. By doing so, we are pushing the boundary that separates our definition of *volume* vs *water droplet* fx elements.

7.5.1 Volume representation

In physically-based rendering, a volume is defined by its phase function and some coefficients (absorption, scattering, emission). While the phase function describes the distribution of direction the light scatters towards at any interaction with the volume, the scattering and absorption coefficients describe the likelihood of interacting with the volume when tracing through it. Meng et al. Meng et al. [2015] introduced a method for computing such coefficients and a phase function, given an aggregation of particles. A volume using those coefficients would simulate the interaction of the light with the particles, by stochastically sampling the position of the next particle along the ray. The phase function would then redirect the ray, the same way an actual particle would if we were tracing through it for real.

In the case of water droplets, we have assumed that those are perfectly specular, therefore all the light interacting with such particles is scattered, and none is absorbed. So the absorption coefficient is 0.0 in our case.

7.5.2 Scattering coefficient

The computation of the scattering coefficient described by Meng et al. [2015] or Müller et al. [2016] relies on 3 statistics on the particle set:

- R^2: average particle radii squared $\rightarrow R^2 = \frac{1}{N}\sum_i^N r_i^2$
- R^3: average particle radii cubic $\rightarrow R^3 = \frac{1}{N}\sum_i^N r_i^3$
- f : fraction of particles in the volume $\rightarrow f = \frac{\text{volumeParticles}}{\text{volumeTotal}}$

Those statistics are easy to compute given a set of particles. Moreover, the simulation engine can directly provide those statistics per voxel, thus avoiding to have a constant scattering coefficient in large areas, which would not reflect the real distribution of the water droplets. Müller et al. Müller et al. [2016] describe the following equation for computing the scattering coefficient from the statistics:

$$\begin{aligned} \sigma_s &= \frac{1}{\lambda_s+\lambda_v} \\ \text{where} \quad \lambda_s &= \frac{4R^3}{3R^2}\frac{1-f}{f} \\ \text{and} \quad \lambda_v &= 1.44 \cdot \frac{R^3}{R^2} \end{aligned} \tag{7.16}$$

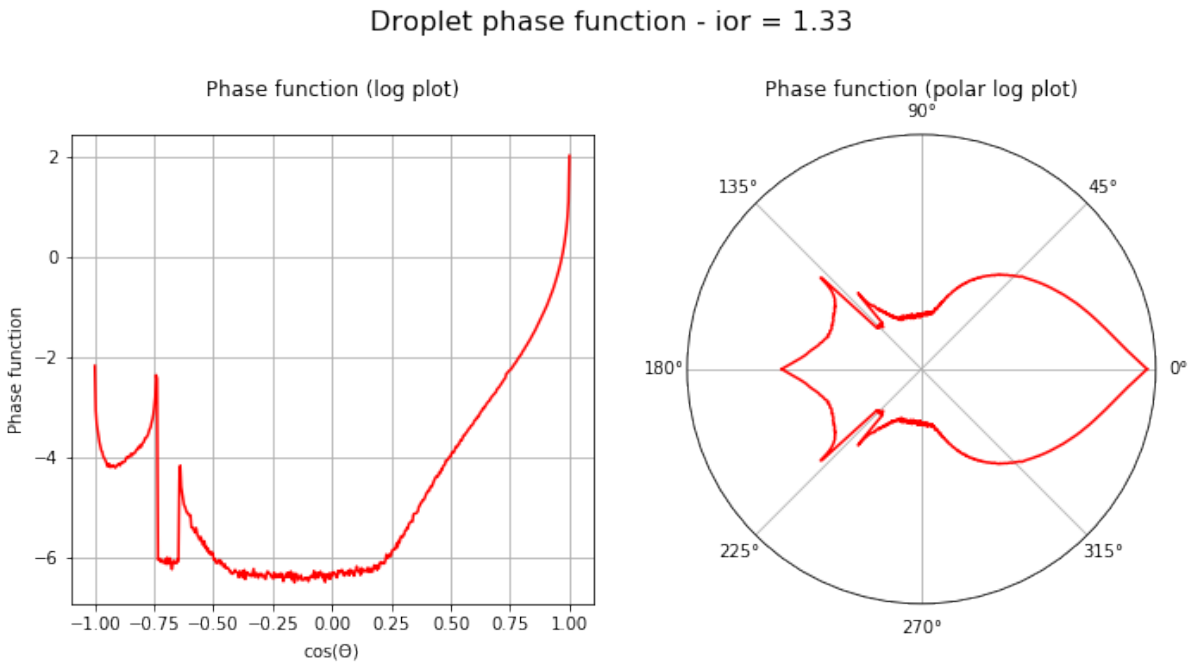


Figure 7.12: *The phase function for a water droplet has been found by simulating millions of rays hitting the particle, and recording the outgoing directions.*

7.5.3 Phase function

The phase function can be computed by running a simulation, tracing million of rays coming from all directions towards the water droplet and recording the outgoing direction. This histogram of directions can be normalized to produce a distribution function that can be used as a phase function in the renderer. In Fig. 7.12 we show the plot of the phase function for a water droplet.

7.6 Results

Fig. 7.11 compares rendering distinct water droplets vs converting them into a volume. Due to the size of the single water particle, the difference between the two images is hardly visible. Differently, in Fig. 7.13 (left), we applied the conversion technique to particles with a bigger size. In this case,

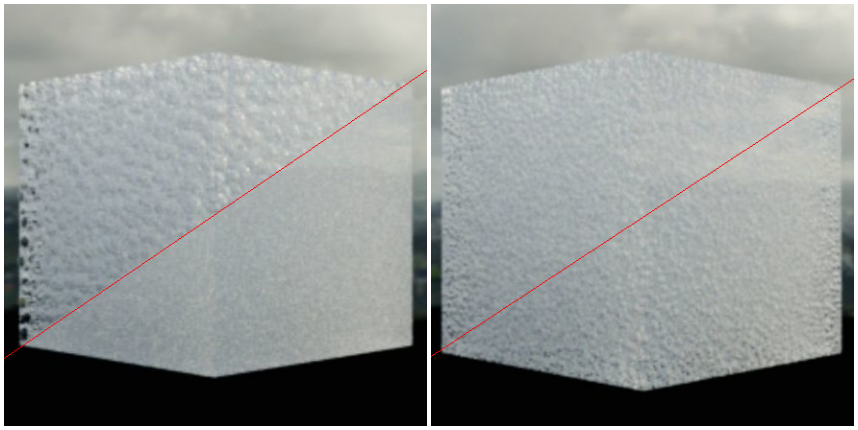


Figure 7.13: *If the size of the single water droplet is not small enough, the difference with the conversion to volume becomes apparent (left). In this case, we restrict the conversion to the inner part of the cube, and keep the water droplets as separate particles on the outside (right).*

the difference is noticeable, especially at the edges of the cube. This is due to the silhouette of the single water droplets, which is now covering more pixels and hence adding visual details to the final render. To overcome this specific case, one possibility is to render an outer layer of water droplets and convert them to volume only inside the cube, as shown in Fig. 7.13 (right).

7.7 Acknowledgments

The work presented in this section of the course is the culmination of collaborative efforts by numerous researchers and artists. Over the years, various members of the rendering team have contributed to the techniques discussed. We extend our gratitude to Sébastien Speierer, whose dedicated work during his time with our team resulted in the material presented in Sec. 7.5. Special thanks are also due to Lorenzo Tessari for his research on motion roughening, which led to the content presented in Sec. 7.3.1. We acknowledge the continuous collaboration with the simulation team, enabling the integration of some of our techniques into Loki Lesser et al. [2022], Wētā Digital's simulation engine. Finally, we express our appreciation to all the artists at Wētā FX who generously provided valuable feedback and tested our tools.

References

James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.* 1, 3 (jul 1982), 235–256. https://doi.org/10.1145/357306.357310

Ole Caprani, Lars Hvidegaard, Mikkel Mortensen, and Thomas Schneider. 2000. Robust and Efficient Ray Intersection of Implicit Surfaces. *Reliable Computing* 6 (02 2000), 9–21. https://doi.org/10.1023/A:1009921806032

John C. Hart. 1996. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545. https://doi.org/10.1007/s003710050084

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 339–346.

Anton S. Kaplanyan, Stephen Hill, Anjul Patney, and Aaron Lefohn. 2016. Filtering Distributions of Normals for Shading Antialiasing. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*, Ulf Assarsson and Warren Hunt (Eds.). The Eurographics Association. https://doi.org/10.2312/hpg.20161201

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: A Unified Multiphysics Simulation Framework for Production. *ACM Trans. Graph.* 41, 4, Article 50 (jul 2022), 20 pages. https://doi.org/10.1145/3528223.3530058

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH Comput. Graph.* 21, 4 (aug 1987), 163–169. https://doi.org/10.1145/37402.37422

Gary A. Mastin, Peter A. Watterberg, and John F. Mareda. 1987. Fourier Synthesis of Ocean Scenes. *IEEE Computer Graphics and Applications* 7, 3 (1987), 16–23. https://doi.org/10.1109/MCG.1987.276961

Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. 2015. Multi-Scale Modeling and Rendering of Granular Materials. *ACM*

Transactions on Graphics (Proceedings of SIGGRAPH) 34, 4 (July 2015). https://doi.org/10/gfzndr

Thomas Müller, Marios Papas, Markus Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient Rendering of Heterogeneous Polydisperse Granular Media. *ACM Trans. Graph.* 35, 6, Article 168 (dec 2016), 14 pages. https://doi.org/10.1145/2980179.2982429

Marc Olano and Dan Baker. 2010. LEAN Mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '10)*. Association for Computing Machinery, New York, NY, USA, 181–188. https://doi.org/10.1145/1730804.1730834

K. Perlin and E. M. Hoffert. 1989. Hypertexture. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '89)*. Association for Computing Machinery, New York, NY, USA, 253–262. https://doi.org/10.1145/74333.74359

Manuele Sabbadin and Marc Droske. 2021. *Ray Tracing of Blobbies*. Apress, Berkeley, CA, 551–568. https://doi.org/10.1007/978-1-4842-7185-8_35

Miguel A. C. Teixeira, Steve Arscott, Simon J. Cox, and Paulo I. C. Teixeira. 2015. What is the Shape of an Air Bubble on a Liquid Surface? *Langmuir* 31, 51 (2015), 13708–13717. https://doi.org/10.1021/acs.langmuir.5b03970 arXiv:https://doi.org/10.1021/acs.langmuir.5b03970 PMID: 26605984.

Lorenzo Tessari, Johannes Hanika, Carsten Dachsbacher, and Marc Droske. 2020. Temporal Normal Distribution Functions. In *Eurographics Symposium on Rendering - DL-only Track*, Carsten Dachsbacher and Matt Pharr (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20201132

Yusuke Tokuyoshi and Anton S. Kaplanyan. 2019. Improved Geometric Specular Antialiasing *(I3D '19)*. Association for Computing Machinery, New York, NY, USA, Article 8, 8 pages. https://doi.org/10.1145/3306131.3317026

G.M. Treece, R.W. Prager, and A.H. Gee. 1999. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics* 23, 4 (1999), 583–598. https://doi.org/10.1016/S0097-8493(99)00076-X

Jihun Yu and Greg Turk. 2013. Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels. *ACM Trans. Graph.* 32, 1, Article 5 (Feb. 2013), 12 pages. https://doi.org/10.1145/2421636.2421641

Cem Yuksel. 2022. High-Performance Polynomial Root Finding for Graphics. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 27 (jul 2022), 15 pages. https://doi.org/10.1145/3543865