Path tracing in Production

Part 1: Production renderers

SIGGRAPH 2017 Course Notes

Fig. 1: Imagery illustrating technical challenges for physically-based light transport throughout the range of movie productions, be it visual effects or animation. Top row: Stills from the movie Cars 3 (copyright ©2017 Pixar/Disney), rendered in *RenderMan*, and Moana (copyright ©2016 Disney), rendered in *Hyperion*. Bottom row: Left: Image rendered in Solid Angle's Arnold at Trixter from the movie Captain America: Civil War (copyright ©2016 Marvel Studios. All Rights Reserved). Right: Image rendered in Weta Digital's Manuka, from the movie The BFG (copyright ©2016 Storyteller Distribution Co., LLC. All Rights Reserved).

http://dx.doi.org/10.1145/3084873.3084904

Abstract

The last few years have seen a decisive move of the movie making industry towards rendering using physically-based methods, mostly implemented in terms of path tracing. Increasing demands on the realism of lighting, rendering and material modeling, paired with a working paradigm that very naturally models the behaviour of light like in the real world mean that more and more movies each year are created the physically-based way. This shift has also been recently recognised by the Academy of Motion Picture Arts and Sciences, which in this year's SciTech ceremony has awarded three ray tracing renderers for their crucial contribution to this move. While the language and toolkit available to the technical directors get closer and closer to natural language, an understanding of the techniques and algorithms behind the workings of the renderer of choice are still of fundamental importance to make efficient use of the available resources, especially when the hard-learned lessons and tricks from the previous world of rasterization-based rendering can introduce confusion and cause costly mistakes. In this course, the architectures and novel possibilities of the next generation of production renderers are introduced to a wide audience including technical directors, artists, and researchers.

This is the first part of a two part course. While the first part focuses on architecture and implementation, the second one focuses on usage patterns and workflows.

Contents

1 Objectives

The objective of this course is to provide the audience with insight into the inner workings of a modern, production oriented, physically-based path tracer, as well as the ecosystem of tools and workflows surrounding it. As more and more movies are rendered with this paradigm, powerful means to support the intuition of technical directors at all levels of seniority are of primary importance in the solution of rendering problems. As the early adopters found out and rapidly shared with the community, the new paradigm has obvious advantages in terms of of simpler and faster realistic lighting and material modeling, while enabling novel workflows and reclaiming a few patterns that were typical of a rasterisation-based world. It is immediately apparent that material and lighting modeling with physically-based entities is more intuitive for artists, as the virtual world behaves more and more closely like the real world. An angle that maybe is somewhat less obvious is how this also allows a separation of rendering algorithms from material descriptions, resulting in more portable assets which require far less tweaking as they flow through the pipeline from look development to lighting.

The world of architectural and product visualisation have illustrated how measured materials and light sources can help attain a whole new level of realism, while the production environment is still in the process of understanding how to harness the possibilities of these tools while allowing the necessary control for artists.

To facilitate this kind of control, physically-based path tracing can be enriched by sidecar data such as light path expressions or arbitrary output variables. To aid compositing, production tricks can be applied inside the path tracing loop, involving matte objects or holdouts, as well as including non-physical effects for instance via illuminance loops. Even though most of these concepts have been known for a long time, new ways to best incorporate this into a physically-based renderer had to be found.

The path tracing paradigm has been around for a long time, but it was only through the advances of the last decade, both in terms of algorithmic research and hardware capability evolution, that is has become a viable proposition for large-scale movie-making. Indeed, the recent evolution in the domain of noise reduction algorithms, as well as the now ubiquitous use of progressive refinement have been instrumental in this phase. To compute noise free and temporally stable images, many optimisation and tuning points have to be built into the algorithms inside the renderer but also into the tools surrounding it. This enables finer grained intervention such as for instance more precise compositing and an ever expanding family of adaptive sampling strategies.

The course will review the coherent state of the art in path tracing for movie production, its novel workflows, and software architectures that can face the challenges of the gigantic amounts of geometry, textures, and light sources that make up a movie frame. Examples from recent productions (see Fig. 1) provide evidence of the benefits of using path tracing in movie production.

Although advanced in nature, we welcome the opportunity to frame the course to engage a wide audience including technical directors, artists, their producers and managers, as well as researchers. With all the spearheading companies sharing and explaining the lessons they learned on the field, we hope we'll be able to further improve the adoption of path tracing and help the audience gain the confidence to explore, create and invent in the new world.

2 Syllabus

9:00 — Introduction to path tracing and Monte Carlo sampling

Luca Fascione will survey the principles of path tracing and modeling with physically-based entities, which will serve as the foundation for all subsequent presentations. Monte Carlo integration is introduced, and then applied to the light transport simulation context. The themes for the upcoming sections will be then briefly introduced, illustrating how they relate to each other and together make the fundamental components of a modern path tracing renderer. At the same time, the course presenters will be introduced and it will be pointed out how their revolutionary work is connected.

9:20 — Arnold and How Path Tracing Took Over (30 min)

Monte Carlo path tracing is now the standard rendering approach in film VFX, animated films, commercials and pre-rendered video game intros. The Arnold renderer from Solid Angle played a significant role in the transition from rasterization-based technology. In this talk Marcos Fajardo will provide some historical context on how studios made this transition and describe the key benefits that motivated it. Marcos will also discuss some of the latest developments in the Arnold renderer as well as the challenges that still lie ahead in the never-ending quest for increased detail and visual realism.

9:50 — Advanced path tracing in RenderMan: bidirectional, progressive photon mapping, VCM, UPBP (30 min)

RenderMan is a modern extensible and programmable path tracer with many features essential to handling the fiercely complex scenes in movie production. In this talk Per Christensen will describe the theory and practice of advanced path tracing techniques: bidirectional path tracing, progressive photon mapping, vertex connection and merging (VCM), and unified points, beams, and paths (UPBP). These techniques can overcome some of the challenging lighting situations where regular path tracing will fail (i.e. converge extremely slowly). Like regular path tracing, these techniques have gone from being pure research techniques to now being implemented in some production renderers. But unlike regular path tracing, they are not yet in mainstream use in movie production. Per will discuss some of the technical and practical reasons why these advanced path tracing techniques have not yet caught on.

10:20 — Break (10min)

10:30 — Manuka, Weta's Physically-Based Spectral Renderer (30 min)

In terms of color reproduction, *Weta Digital*'s renderer *Manuka* is taking physically-based seriously and computes all transport on spectral power distributions instead of using the traditional RGB-based approach. Johannes Hanika will motivate this choice in his talk by showing theoretical and practical benefits. This includes advantages of using real radiometric quantities during transport simulation and the effect on importance sampling, as well as using actual photometric quantities on the UI side for the lighters. The main differences to the RGB pipeline are explained along with the newly required tools, and the impact on rendering gamut and color noise is discussed.

11:00 — Disney's Hyperion Renderer (30 min)

Brent Burley will present changes to *Hyperion* made in support of *Zootopia*, *Moana*, and upcoming *Disney* animated productions. Significant developments include a transition away from multiple scattering approximations to brute-force path tracing for hair and fur, skin, snow, and high-albedo volumes such as clouds. Brent will continue describing how the Hyperion team addressed challenges with artistic control and efficiency, as well as future directions such as path tracing of cloth fibers. Additionally, Brent will share some details about how *Hyperion*'s batch-based ray tracing architecture is evolving to accommodate unbounded path lengths and leverage increasing core counts.

11:30 — Moonray, DreamWorks's new Path Tracing Renderer (30 min)

MoonRay has been in development for the past four years and its growth has recently crossed the threshold that enables production use. Brian Green will describe the inception and development of a brand new physically-based path tracing production rendering system, leveraging, leading and improving various open source components. He will highlight some of the unique aspects of MoonRay's vectorized path tracing, shading and texturing architecture, and discuss the challenges and benefits of growing a production rendering feature-set on top of a core rendering system designed with scalability and high performance in mind.

12:00 — Q&A with all presenters (15min)

3 Presenters

3.1 Luca Fascione, Weta digital (Organizer)

Luca Fascione is Head of Technology and Research at *Weta Digital* where he oversees Weta's core R&D efforts including Simulation and Rendering Research, Software Engineering and Production Engineering. Luca architected Weta Digital's next-generation proprietary renderer, Manuka with Johannes Hanika. Luca joined *Weta Digital* in 2004 and has also worked for *Pixar Animation Studios*. The rendering group's software, including *PantaRay* and *Manuka*, has been supporting the realization of large scale productions such as *Avatar*, *The Adventures of Tintin*, the *Planet of the Apes* films and the *Hobbit* trilogy. He has recently received an Academy Award for his contributions to the development of the facial motion capture system in use at the studio since *Avatar*.

3.2 Johannes Hanika, Weta digital (Organizer)

Johannes Hanika received his PhD in media informatics from *Ulm University* in 2011. After that he worked as a researcher for *Weta Digital* in Wellington, New Zealand. There he was co-architect of *Manuka*, *Weta Digital*'s physically-based spectral renderer. Since 2013 he is located in Germany and works as a post-doctoral fellow at the *Karlsruhe Institute of Technology* with emphasis on light transport simulation, continuing research for *Weta Digital* part-time. In 2009, Johannes founded the *darktable* open source project, a workflow tool for RAW photography.

3.3 Marcos Fajardo, Solid Angle

Marcos is the founder and CEO of Madrid and London-based Solid Angle, where he leads the research and development team working on the Arnold path tracing renderer. Previously he was a visiting software architect at Sony Pictures Imageworks, a visiting researcher at USC Institute for Creative Technologies under the supervision of Dr. Paul Debevec, and a software consultant at various CG studios around the world. He studied Computer Science at University of Malaga, Spain. Marcos is a frequent speaker at SIGGRAPH, FMX and EGSR. He has recently received an Academy Award for the design and implementation of the Arnold renderer. His favorite sushi is engawa.

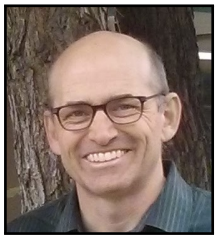
3.4 Per Christensen, Pixar

Per Christensen is a principal software developer in *Pixar*'s *RenderMan* group in Seattle. His main research interests are efficient ray tracing and global illumination in very complex scenes. He received an M.Sc. degree in electrical engineering from the *Technical University of Denmark* and a Ph.D. in computer science from the *University of Washington*. Prior to joining *Pixar*, he worked at *ILM* in San Rafael, *Mental Images* in Berlin, and *Square USA* in Honolulu. He has received an Academy Award for his contributions to point-based global illumination and ambient occlusion.

3.5 Brent Burley, Walt Disney Animation Studios

Brent Burley is a Principal Software Engineer at *Walt Disney Animation Studios* leading the *Hyperion* development team. Previously he led the development of the physically-based shading model used in all WADS productions since *Wreck-It Ralph*, and created *Ptex*, an open-source texture mapping system for subdivision surfaces used on all WADS productions since *Bolt*. Prior to joining *Disney* in 1996, he worked at *Philips Media* developing a cross-platform game engine, and also worked on aircraft training simulators at *Hughes Training Inc*.

3.6 Brian Green, DreamWorks



Brian Green is a principal engineer in the R&D group at DreamWorks Animation, where he has been working on film production rendering since 2005. Prior to that, he held a similar position at Rhythm and Hues studios since 1997. Brian's work has primarily focused on sytem-level aspects of production rendering, including distributed rendering, multi-threading, and vectorization. Brian also worked on various aspects of computer graphics, such as shading frameworks, AOVs, animation systems, curve editors, and media tools. Brian was awarded a Technical Achievement Academy Award in 2016 for his work on *Eve*, part of the Rhythm & Hues digital daily review system. Brian has a long list of film credits including every DreamWorks animated film sinch *Over The Hedge*.

4 Introduction to path tracing and Monte Carlo sampling

Johannes Hanika, *Weta Digital*

This section summarises a few basic concepts of light transport and introduces the Monte Carlo integration scheme. This forms the foundation of the path tracing family of algorithms, which can be used to synthesise pictures in a unified way. While we want to introduce all commonly used equations and explain the terms therein, this will mainly form a basis for common notation rather than explain the underlying principles in exhausting detail. For a more in-depth introduction we refer to Pharr et al. [2017].

4.1 The path space

To be able to unify lighting computations in one shared renderer (be it surfaces, subsurface scattering, or volume contributions), we need to define a common mathematical frame work. Intuitively, we want to track all possible paths that photons could take from all light sources via multiple interactions with objects and their materials, into the camera lens. These photons will then be "counted" on the sensor.

The mathematical way of expressing all these individual transport paths is called the *path space* $\mathcal{P}$ and contains all possible transport paths $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\} \in \mathcal{P}$, which are lists of k path vertices $\mathbf{x}$.

To compute the color I_p of a pixel p, we then simply integrate over this space, weighted by a pixel filter $h(\mathbf{X})$, such as for instance the one derived by Harris [1978]:

$$I_p = \int_{\mathcal{P}} h(\mathbf{X}) \cdot f(\mathbf{X}) \, \mathrm{d}\mathbf{X}, \tag{1}$$

where $f(\mathbf{X})$ is the *measurement contribution function*, which evaluates the differential flux through an infinitesimally small hose around the vertices of the path. To illustrate this, the unit is $W/m^{2 \cdot k}$, i.e. watts per one square meter for each of the k vertices along the path. Accordingly, the measure $\mathrm{d}\mathbf{X}$ is the product vertex area measure, i.e. a product of square meters: $\mathrm{d}\mathbf{X} = \prod_{i=1}^{k} \mathrm{d}\mathbf{x}$. As an additional mathematical convenience, the measure $\mathrm{d}\mathbf{X}$ is defined in such a way that it contains paths of all lengths k.

Veach [1998] introduced all this very carefully in his excellent dissertation. There is likely little need to point this out here, however, many readers will have a printed copy of it on their bedside table (if not you probably should have).

4.2 Transport equations

So far we introduced the mathematical framework, but did not talk about how light is actually transported. This is expressed by the measurement contribution function $f(\mathbf{X})$. The exact form can be derived from the radiative transfer equation as discussed by Chandrasekar [1960], which defines the change of radiance L at a point $\mathbf{x}$ in direction ω is due to emission (μ_e), extinction (μ_t), and scattering (μ_s):

$$\begin{aligned}(\omega \cdot \nabla) L(x, \omega) = \mu_e(x) L_e(x) - \mu_t(x) L(x, \omega) \\ + \mu_s(x) \int_{\Omega} \varphi(x, \omega, \omega') L(x, \omega') d\omega'.\end{aligned} \tag{2}$$

This has been reformulated into a concise recursive integral equation by Kajiya [1986]. For brevity, here is the version for vacuum transport, i.e. without participating media:

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{\Omega} f_r(\mathbf{x}, \omega, \omega_i) L(\mathbf{x}, \omega_i) \, \mathrm{d}\omega_i^{\perp}. \tag{3}$$

The integration domain Ω is the (hemi-)sphere of incoming directions ω_i, and the measure $\mathrm{d}\omega^{\perp} = \cos\theta \, \mathrm{d}\omega$ is the projected solid angle measure, which includes a foreshortening factor to account for Lambert's law. The term $f_r(.)$ is the *bidirectional scattering distribution function* (BSDF) and characterises how incoming irradiance is converted to outgoing radiance. This is responsible for the look of surface materials, such as texture or glossiness.

Expanding this equation by unrolling the recursion to yield a path of length k results in the "flat view" of the rendering equation, the measurement contribution function

$$f(\mathbf{X}) = L_e G_{k-1} \left(\prod_{i=2}^{k-2} f_{r,i} G_i \right) W. \tag{4}$$

Note that this contains geometry terms G, which need to be applied for every edge of the path. These convert the projected solid angle measure $\mathrm{d}\omega^{\perp}$ to vertex area measure, such that we can integrate $f(\mathbf{X})$ over the path space in product vertex area measure.

The geometry terms G also deal with occlusion, by means of a visibility operator. This can also be extended to include transmittance in participating media, for instance attenuation of light passing through smoke. If the BSDF f_r is also generalised to express the scattering collision coefficient and phase function $\mu_s \cdot \varphi(\omega, \omega_i)$, this equation is suitable to render participating media, too.

The last term in Eq. (4), W, is the sensor responsivity function. It models how the sensor reacts to light. While this is mostly used to un-do the vignetting caused by most camera models, it may also model a spectral response corresponding to the colour filter array of the sensor.

4.3 The Monte Carlo method

Inserting Eq. (4) into Eq. (1), i.e. integrating the measurement contribution function over path space, yields a high dimensional integration problem. Depending on path length, the integral can easily contain hundreds of dimensions. This makes many popular integration schemes perform poorly (for instance quadrature rules would yield exponential complexity in the number of dimensions).

The method of choice due to its behaviour for high dimensionality is the Monte Carlo method (see for instance Ermakow [1975] or Sobol' [1994] for an introduction).

The main idea is to make use of the definition of the expected value of a continuous random variable x distributed with a *probability distribution function* (PDF) $p(x)$ to solve the integral

$$\mathbb{E}(x) = \int x \cdot p(x)\, \mathrm{d}x \tag{5}$$

by drawing a few random trials from x instead of solving the integral analytically. This results in a noisy Monte Carlo estimator

$$\hat{x} = \frac{1}{N} \sum_{i=1}^{N} x \approx \mathbb{E}(x). \tag{6}$$

Applying this same principle to the path space integral, we need to divide out the probability distribution function $p(.)$ from the integrand $f(\mathbf{X})$ to match the definition of the expected value in Eq. (5):

$$I_p = \int_{\mathcal{P}} h(\mathbf{X}) \cdot f(\mathbf{X})\, \mathrm{d}\mathbf{X} \approx \hat{I}_p = \frac{1}{N} \sum_{i=1}^{N} \frac{h(\mathbf{X}) \cdot f(\mathbf{X})}{p(\mathbf{X})}. \tag{7}$$

The crucial difficulty in designing good estimators is now to find an appropriate PDF $p(\mathbf{X})$ which minimises the integration error of this approximation. Such error manifests itself mostly due to variance

$$Var(\hat{I}_p) = \frac{1}{N} \int \left(\frac{h(\mathbf{X}) f(\mathbf{X})}{p(\mathbf{X})} - I_p \right)^2 p(\mathbf{X})\, \mathrm{d}\mathbf{X}. \tag{8}$$

Mostly here means that we assume all employed algorithms will be unbiased, such that the error is spread around the correct mean and the deviation will be only random noise, decreasing with higher sample count N. Eq. (8) shows that the primary estimator $h \cdot f/p$ needs to be close to I_p to reduce variance. In practice, this can be achieved by variance reduction techniques, such as importance sampling. This tries to choose $p(\mathbf{X})$ to follow $f(\mathbf{X})$ as closely as possible (of course the PDF will be normalised while f is not). This goal is all but trivial to achieve in general for the high dimensional path space.

The following sections will give a run down through how these concepts are applied to image formation and embedded into the different pipelines at different studios.

References

Subrahmanyan Chandrasekar. 1960. *Radiative Transfer*. Dover Publications Inc. ISBN 0-486-60590-6.

Sergej Mikhailovich Ermakow. 1975. *Die Monte Carlo Methode und verwandte Fragen*. VEB Deutscher Verlag der Wissenschaften.

Frederic J. Harris. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE* 66, 1 (1978), 51–83.

James T. Kajiya. 1986. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)* (1986), 143–150.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2017. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc.

Ilya Sobol'. 1994. *A Primer for the Monte Carlo Method*. CRC Press.

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.

5 Arnold and How Path Tracing Took Over

Marcos Fajardo, *Solid Angle*

5.1 Historical context

After playing around as a student with pre-opensource ray-tracers like Vivid and POV-Ray, and educating himself in the pre-PBR literature at the time, such as the Ray Tracing News ASCII-based journal by Eric Haines (Haines [2010]), and Andrew Glassner's books (Glassner [1989], Glassner [1994]), the skeleton of what later became the Arnold renderer was started in 1997 by this author. The goal was in writing a tool that would allow artists to take advantage of the increased realism offered by physically-based, brute-force ray-tracing, a rendering technique that was starting to become popular, but that was prohibitively costly at the time. Rather than developing a polished product with a user interface, the renderer was designed as a C-based application programming interface, or API, so that it could be easily integrated into arbitrary 3D applications in both CAD and entertainment industries.

A visit to Blue Sky Studios in 1998, where the pioneering ray-tracer CGI-Studio was already being used in the production of films and commercials, sparked this author's interest into Monte Carlo ray-tracing techniques, and based on published academic research by Jim Kajiya (Kajiya [1986]), Peter Shirley (Shirley [1991], Shirley [1992], Shirley and Chiu [1994]) and others, the basic sampling components and other bits started to crystalize into what was now clearly a path tracer.

Figure 1: "Pepe" image courtesy of Daniel Martinez Lara. © 1999 Daniel Martinez Lara. All rights reserved.

Around 1999, a 3dsMax plugin for the renderer was developed. By now, the renderer was tentatively called "Arnold", as a nod to fellow VFX wizard Andy Lesniak and his particularly hilarious impersonations of then Governor of California, actor and bodybuilder Arnold Schwarzenegger. Schwarzenegger's Austrian accent was in stark contrast to the beautiful and flawless Spanish accent of dubbing actor Constantino Romero, which was a massive culture shock when this author first moved to the United States and watched films like *End of Days*. The 3dsMax plugin allowed the animated short film *Pepe* (see Fig. 1) to be rendered with unbiased global illumination on just a handful of machines by Spanish animator Daniel Martinez Lara. This short film popularized the term "global illumination" amongst CG artists around the world and, as they say, the rest is history.

In 2004, Arnold was adopted at Sony Pictures Imageworks for the rendering of the animated movie *Monster House*, which required a particularly realistic and tactile marionette look that was very difficult to achieve with commercial rendering products at the time. The success of this project led to Imageworks' licensing of the Arnold source code, the expansion of its in-house rendering development team, and a full migration to Arnold for all their future films. The production demands of their next animated film, *Cloudy with a Chance of Meatballs*, led to further development of important features such as ray traced curves, sub-surface scattering, and deformation motion blur, as well as speed and memory optimizations. With the production experience gained while at Imageworks, the Madrid-based company Solid Angle was formed in 2009 to continue to develop and market the renderer to a wider audience. A series of conference talks and papers on the benefits of path tracing, and on specific algorithmic advances that were developed in-house at both Solid Angle and Imageworks, contributed to the industry's realization

that path tracing was here to stay, and that not only was it a viable alternative to rasterization-based renderers, but it was clearly the most promising direction of research. Both VFX and animation studios jumped at the opportunity to simplify their pipelines, while at the same time increasing the realism of the images that they could produce on ever tightening budgets.

5.2 Motivation

At a purely technical level, realism in CG images boils down to two things: scene detail (number of polygons and hairs, number and size of textures, variety of materials) and quality of lighting simulation (soft shadows, reflections, refractions, diffuse bounce light, etc). The prevailing systems at the time, in the early 2000's, could either render very complex geometric data sets with poor lighting simulation (REYES and rasterization-based renderers), or render simpler scenes with much higher quality, ray-traced shading effects. You couldn't get the best of both worlds, and often had to make concessions, such as in shadow quality. The goal of Arnold was to be the first physically-based production ray tracer that scaled to film complexity. The basic architecture was therefore motivated by the need for an efficient, light-weight system capable of generating artifact-free images in a single pass, avoiding expensive pre-processing, and minimizing both disk and memory usage. The system should also be easy to use, with a minimal set of controls, enough to provide a basic level of art direction. Path tracing ticked most of these boxes from the very beginning. In particular, shadows were always perfect. The only artifact was a fine-grain noise in the rendered images, which, as annoying as it can be at low sample settings, is guaranteed to converge to the right result. This consistency and reliability proved to be one of the main cost savings for studios, which no longer had to spend hours fighting with the renderer just to identify where image artifacts were coming from. At the same time, the fact that path tracing allows for a lower time-to-first-pixel and progressive-sampling of the image plane, meant that artists could quickly try variations in shading and lighting without the need for a multi-hour render to finish. In essence, it proved to be more cost effective to let the machines render final frames for longer, than to have artists sitting around fighting with a more complicated renderer that could do faster final frames. An hour of an artist's time can cost hundreds of times more than an hour of CPU time.

5.3 Architecture

Arnold is an unbiased, uni-directional (backwards) CPU path tracer, based on a classical ray tracing kernel. It is built on top of a programmable, node-based architecture, with different types of nodes such as geometric primitives, shaders, cameras, or lights. Nodes can be interconnected in a node network, for example in a shader network to form complex materials. Geometric primitives include polygon meshes, hair curves, volumes, procedurally-created geometry, and simple quadrics. A two-level hierarchy of BVH ray acceleration data structures is used to hold the scene's geometry. This BVH is able to intersect different types of primitives at the leaf level, whether it's polygons, hairs, or particles. A great deal of effort was spent in optimizing these ray accels to minimize memory use, build time, and traversal time. Geometric primitives can be instanced any number of times, which allows the easy creation of massive scenes containing vegetation, debris, crowds, etc. Even without instancing, the system is efficient enough that around a billion polygons can be stored in 24 GB of memory. Primitives are stored in memory in a compact representation using both lossless and lossy compression techniques where appropriate.

5.4 Sampling

As first described by Cook et al. [1984], at the core of the renderer lie several numerical integration subproblems that are solved via Monte Carlo sampling, such as soft shadows, depth of field, indirect lighting, or motion blur. Stratification and importance sampling are the two most important techniques for reducing the variance of the estimators involved (see e.g. Glassner [1994]), and this is where we spend most of our research efforts. In particular, the sub-problem of direct lighting by next-event estimation is an area where, even today, after decades of publications on the subject, improvements are still found. Arnold

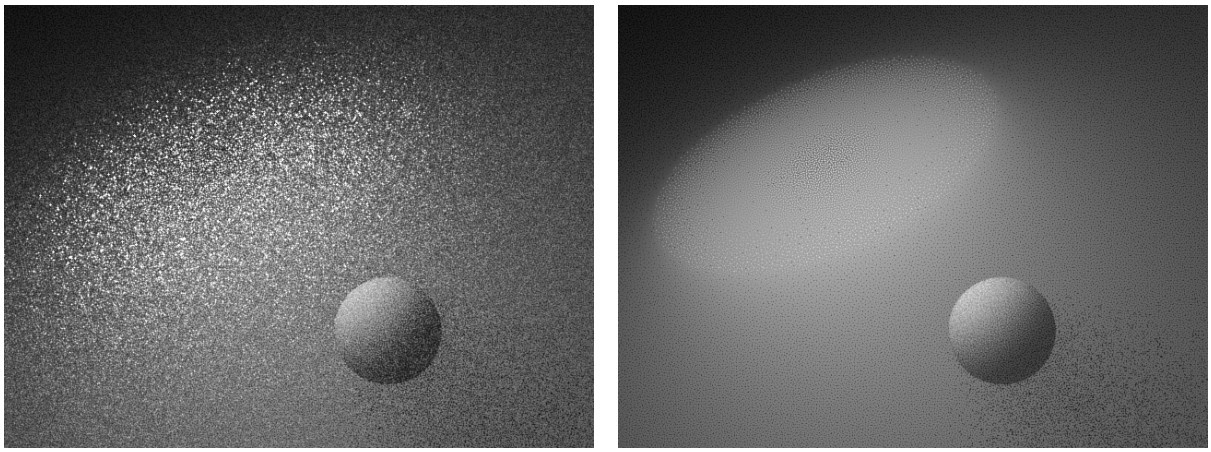


Figure 2: These images show next event estimation in a participating medium, lit by a disk-shaped area light source. Left: naive area sampling compared to right: importance sampling of the solid angle. This technique is described in detail by Guillén et al. [2017].

makes extensive use of the solid angle domain when sampling direct lighting (see Fig. 2). Ideally, all area lights would be sampled according to the cosine-weighted, projected solid angle of the light with respect to the shading point. This, however, is extremely difficult to do with closed-form, analytical expressions for the sample directions.

Some of the recent sampling improvements in Arnold include equi-angular and decoupled ray marching (Kulla and Fajardo [2012]), BSSRDF importance sampling (King et al. [2013]), solid angle (Ureña et al. [2013]) and cosine-weighted solid angle sampling for quad area lights (Arvo [2001]), solid angle sampling for disk lights (Guillén et al. [2017]), and blue-noise dithering patterns that perceptually improve the distribution of the sampling error across adjacent pixels (Georgiev and Fajardo [2016]).

As for indirect lighting, as much as we have tried to experiment with bi-directional techniques (such as Lafortune and Willems [1993], Veach and Guibas [1994]), Arnold is still a uni-directional path tracer. It turns out that uni-directional path tracing can efficiently solve a very wide subset of the interesting scenes that we must render in production, with little to no technical work required from the artist. While bi-directional techniques can work very well in some difficult lighting scenarios, they can also introduce artifacts and inefficiencies in the larger number of "easier" scenes where uni-directional already works very well. It is not clear how to automatically select the algorithm that works best for each scene, other than let the artist select the algorithm, and add a number of obscure heuristics and controls that the artist then needs to master. It is our hope that we will eventually find an algorithm that is more robust to all of these different scenes without the need for human intervention, yet is as efficient as the simpler, uni-directional path tracing algorithm.

References

James Arvo. 2001. Stratified Sampling of 2-Manifolds.

Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 137–145.

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-noise Dithered Sampling. In *ACM SIGGRAPH 2016 Talks*.

Andrew S. Glassner (Ed.). 1989. *An Introduction to Ray Tracing*. Academic Press Ltd., London, UK, UK.

Andrew S. Glassner. 1994. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ibón Guillén, Carlos Ureña, Alan King, Marcos Fajardo, Iliyan Georgiev, Jorge López-Moreno, and Adrian Jarabo. 2017. Area-Preserving Parameterizations for Spherical Ellipses. *Computer Graphics Forum (Proceedings of EGSR)* 36, 4 (2017).

Eric Haines. 1987-2010. http://raytracingnews.org. (1987-2010).

Alan King, Christopher Kulla, Alejandro Conty, and Marcos Fajardo. 2013. BSSRDF Importance Sampling. In *ACM SIGGRAPH 2013 Talks (SIGGRAPH '13)*.

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. *Comput. Graph. Forum* 31, 4 (June 2012).

Eric Lafortune and Yves Willems. 1993. Bi-Directional Path Tracing. In *Proc. of COMPUGRAPHICS*. 145–153.

Peter Shirley. 1992. Time Complexity of Monte Carlo Radiosity. (1992).

Peter Shirley and Kenneth Chiu. 1994. Notes on Adaptive Quadrature on the Hemisphere. (1994).

Peter S. Shirley. 1991. *Physically Based Lighting Calculations for Computer Graphics*. Ph.D. Dissertation. Champaign, IL, USA. UMI Order NO. GAX91-24487.

Carlos Ureña, Marcos Fajardo, and Alan King. 2013. An Area-preserving Parametrization for Spherical Rectangles. In *Proceedings of the Eurographics Symposium on Rendering (EGSR '13)*.

Eric Veach and Leonidas Guibas. 1994. Bidirectional Estimators for Light Transport. 147–162.

6 Advanced path tracing in Pixar's RenderMan

Per Christensen, *Pixar Animation Studios*

Pixar's RenderMan renderer is a modern, extensible, and programmable path tracer with many features that are essential to handling the fiercely complex scenes encountered in movie production.

6.1 Historic background

RenderMan was originally a scanline renderer based on the Reyes algorithm by Cook et al. [1987] which offered ground-breaking efficiency in terms of geometric complexity, texture caching, antialiasing, high-quality motion blur and depth-of-field effects, SIMD shader execution, and more – see Upstill [1990] and Apodaca and Gritz [2000]. Pixar's early short films, the first feature-length CG animated movie, *Toy Story*, and many many other movies have been rendered with this algorithm.

However, shadows had to be computed with shadow maps, reflections with reflection maps, and indirect diffuse illumination had to be "faked" by manually placing additional light sources (a very labor intensive and high-expertise task).

Over the years, many features were added to RenderMan on top of the Reyes algorithm: ray tracing for shadows, specular reflections, and ambient occlusion (see Christensen et al. [2003]), point-based algorithms for noise-free subsurface scattering, ambient occlusion, and indirect diffuse illumination (see Christensen [2008]), and efficient distribution ray tracing for indirect diffuse illumination (see Christensen et al. [2012], Cook et al. [1984]). At last count, nearly 400 CG and VFX movies have been rendered with the help of RenderMan.

6.2 Modern architecture

Over the last few years, we have rewritten RenderMan as a path tracer Kajiya [1986]. The modern version of RenderMan has been used to render the Pixar movies *Finding Dory* and *Cars 3*, as well as recent movies by other studios such as *Terminator Genisys*, *Ant Man*, *The Jungle Book*, and *Star Wars Rogue One*.

Figure 3: A pivotal moment in the *Cars 3* movie. (Copyright © Pixar/Disney 2017.)

The reasons for the switch to path tracing were that it is a unified and flexible algorithm, it is well suited for progressive rendering, geometric complexity can often be handled through object instancing, and it is a single-pass algorithm (unlike the point-based approaches that require a pre-pass to generate point clouds). Also, the Achilles' heel of path tracing – noisy images and slow convergence – has been adressed by new and effective denoisers (as described by e.g. Zimmer et al. [2015], Zwicker et al. [2015]) and improved

sampling. Furthermore, it was getting increasingly hard to scale the Reyes-based architecture to run efficiently beyond 16 threads.

It was important to still provide flexibility and programmability similar to the original RenderMan architecture. This is to enable our customers to create their own materials and experiment with different rendering algorithms based on path tracing. We use a plug-in architecture, where users can write their own rendering algorithms and materials ("integrators" and "bxdfs"). The interface is largely inspired by the PBRT book by Pharr et al. [2017].

We keep groups of ray hits on the same material together in "shade groups" so that their bxdf can be evaluated together in a single function call. This enables compiler optimizations such as SIMD execution, vectorization, loop unrolling, etc. in the bxdf execution, as also gives improved data locality.

Other properties that have been maintained in the "new world" of path tracing is the use of multi-resolution textures and multiresolution tessellated geometry, with the appropriate resolution determined by ray (path) differentials as described by Christensen et al. [2003], Igehy [1999], Suykens and Willems [2001]. This is necessary to be able to handle heavily textured surfaces (more than 20 textures per surface is common) and hugely complex scenes when the geometry is not suitable for instancing.

6.3 Surface (and volume) materials

Surface materials are specified by bxdfs and by texture patterns to control the parameters of the bxdfs. "Bxdf" is an abbreviation of "bidirectional reflection/transmission/surface-scattering distribution function" for surfaces, and also encompasses phase functions for volumes. The textures can be computed procedurally (for example using OSL) or by looking up in texture maps.

Sadly, gone are the days where a novice TD could quickly learn to write an RSL (RenderMan Shading Language) shader computing surface reflection and transparency. With physically-based rendering it is much more complex: bxdfs are written in C++ and require knowledge about optics, sampling and probability theory, probability density functions (pdfs), and much more.

The two main functions specifying a bxdf are Evaluate() and Generate(). Evaluate() takes as input an array of incident directions and an array of exitant directions, and returns an array of rgb bxdf values (each value being the ratio of reflected radiance in the exitant direction over differential irradiance from the incident direction) and two arrays of pdf values. Generate() takes as input an array of incident directions and generates an array of "random" exitant directions along with two arrays of pdf values for those directions.

Users can write their own bxdfs or simply use one of the bxdfs provided with RenderMan. The most general of the provided bxdfs is PxrSurface, a general-purpose "uber-bxdf" developed by Pixar's studio tools illumination group and used on *Finding Dory*, *Cars 3* and future Pixar movies. Depending on the input parameters this bxdf can look like plastic, metal, paint, glass, skin, and many other materials. (We also provide a simpler bxdf based on Disney's bxdf model developed by Burley [2015].)

Subsurface scattering in skin and other translucent materials is rendered with efficient local path tracing. The amount of scattering is determined using various diffusion approximations: dipole diffusion by Jensen and Buhler [2002], Jensen et al. [2001], quantized diffusion by d'Eon and Irving [2011], photon beam diffusion by Habel et al. [2013], or normalized diffusion by Burley [2015], Christensen and Burley [2015]. Lately we are also experimenting with brute-force subsurface scattering, i.e. subsurface scattering computed with Monte Carlo simulation of a homogeneous volume (an approach pioneered by Weta and Disney). For the brute-force approach, intuitive surface scattering parameters are converted to equivalent volume scattering parameters (volume albedo and extinction coefficients) used in the simulation.

For hair, fur, and feathers we provide a bxdf that is based on the model by Marschner et al. [2003] with later improvements by Hery and Ramamoorthi [2012], Pekelis et al. [2015].

Volumes can have zero scattering (just attenuation with Beer's law), single scattering or multiple scattering, can have homogeneous or heterogeneous density, and can have isotropic or anisotropic scattering. Volumes can be nested within other volumes, or overlap each other. To complicate matters further, the volumes can be emissive (as a flame, fire, neon tubes, etc.) Villemin and Hery [2013] and have motion blur Wrenninge [2016]. RenderMan keeps track of the volumes that a ray enters and exits, and integrates

over all volumes covering a region; it also samples emitting volumes as light sources. RenderMan's volumes are covered in much more detail in the course notes by Fong et al. [2017].

6.4 Rendering algorithms

RenderMan has an interface that allows the implementation of various rendering algorithms ("integrators") based on path tracing. Unidirectional path tracing is the simplest production-quality integrator, but even that is far from trivial due to stochastic light selection in scenes with many light sources, Russian roulette, arbitrary output variables (AOVs) specified by light path expressions (LPEs), deep output images, volume tracking, mattes, and many other features that allows tweaking the rendering to obtain the results a movie director may demand. Even though the path tracing algorithm is based on a Monte Carlo simulation of physics, there is still a need to "cheat" from time to time, for example by altering shadows, or adjusting the position and intensity of highlights and indirect diffuse illumination.

Bidirectional path tracing was developed independently by Lafortune and Willems [1993] and by Veach and Guibas [1994]. It traces paths from the light sources in addition to paths from the camera, and connects the paths with shadow rays. Bidirectional path tracing is advantageous in scenes where the illumination is mainly indirect, for example in interiors with the light sources "hidden" behind lighting fixtures. Despite this advantage, bidirectional path tracing has not caught on as much as unidirectional path tracing in actual movie production. One of the reasons is that the texture cache accesses are much less well-behaved: for unidirectional path tracing the rays are either coherent or can use coarse levels in texture maps, but for bidirectional path tracing the light paths are sometimes both incoherent and require rather fine levels in the texture maps – a combination that is deadly for texture cache performance. Also, many of the non-physical tricks that have been developed for unidirectional path tracing do not work so well for the light paths in bidirectional path tracing.

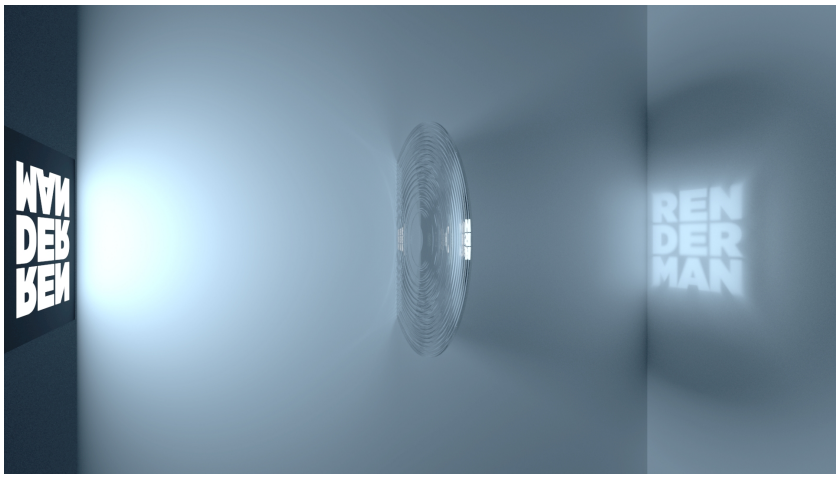


Figure 4: VCM rendering of light from a textured light source being refracted through a Fresnel lens and focused on a diffuse wall. (Image courtesy of Andrew Kensler.)

Vertex connection and merging (VCM, also known as unified path sampling or UPS) by Georgiev et al. [2012] and Hachisuka et al. [2012] is a combination of bidirectional path tracing with progressive photon mapping (as developed by Hachisuka et al. [2008]). VCM excels at rendering caustics and reflections of caustics (which simpler algorithms have a harder time rendering to convergence).

The UPBP (unified points, beams, and paths) algorithm by Křivánek et al. [2014] is a generalization of VCM to volumes. It is particularly suitable for rendering volume caustics and reflections of volume caustics.

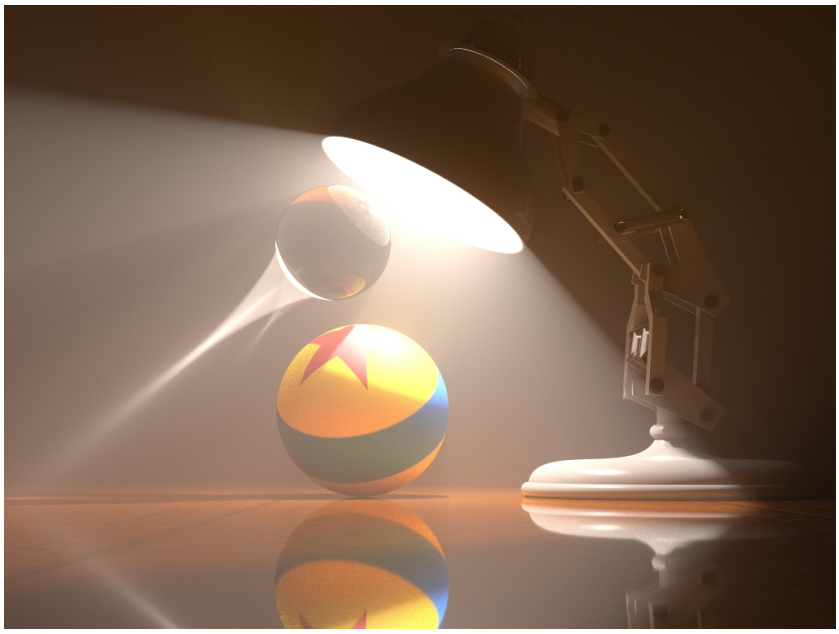


Figure 5: UPBP variation on the classic Luxo Jr. scene. Note the volume caustic under the glass sphere and the reflection in the table of the volume caustic. (Image credit: Brian Savery, Martin Sik, and Per Christensen.)

6.5 Interactive rendering

RenderMan has traditionally been mainly used for final-quality movie rendering. But we are also increasingly focusing on interactive rendering, including optimizing the time to first pixel, time to first complete iteration (one sample per pixel), and "time to first decision" (typically just a few samples per pixel).

With progressive path tracing the first images are of course very noisy, but users are able to make creative decisions (change of scene geometry, texturing, illumination, etc.) very quickly despite the noise.

References

Anthony Apodaca and Larry Gritz. 2000. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann.

Brent Burley. 2015. Extending the Disney BRDF to a BSDF with integrated subsurface scattering. In *'Physically Based Shading in Theory and Practice' SIGGRAPH Course*.

Per Christensen. 2008. *Point-based approximate color bleeding*. Technical Report 08-01. Pixar Animation Studios.

Per Christensen and Brent Burley. 2015. *Approximate reflectance profiles for efficient subsurface scattering*. Technical Report 15-04. Pixar Animation Studios.

Per Christensen, George Harker, Jonathan Shade, Brenden Schubert, and Dana Batali. 2012. Multiresolution radiosity caching for global illumination in movies. In *SIGGRAPH Tech Talks*.

Per Christensen, David Laur, Julian Fong, Wayne Wooten, and Dana Batali. 2003. Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes. *Computer Graphics Forum (Proceedings of Eurographics)* 22, 3 (2003), 543–552.

Robert Cook, Loren Carpenter, and Edwin Catmull. 1987. The Reyes image rendering architecture. *Computer Graphics (Proceedings of SIGGRAPH)* 21, 4 (1987), 95–102.

Robert Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed ray tracing. *Computer Graphics (Proceedings of SIGGRAPH)* 18, 3 (1984), 137–145.

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

Julian Fong, Ralf Habel, Magnus Wrenninge, and Christopher Kulla. 2017. Production Volume Rendering. In *SIGGRAPH Courses*.

Iliyan Georgiev, Jaroslav Křivánek, Tomas Davidovic, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (2012).

Ralf Habel, Per Christensen, and Wojciech Jarosz. 2013. Photon beam diffusion: a hybrid Monte Carlo method for subsurface scattering. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 32, 4 (2013), 27–37.

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

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (2012).

Christophe Hery and Ravi Ramamoorthi. 2012. *Importance sampling of reflections from hair fibers*. Technical Report 12-11. Pixar Animation Studios.

Homan Igehy. 1999. Tracing ray differentials. *Proceedings of SIGGRAPH* 33 (1999), 179–186.

Henrik Wann Jensen and Juan Buhler. 2002. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 21, 3 (2002), 576–581.

Henrik Wann Jensen, Steve Marschner, Marc Levoy, and Pat Hanrahan. 2001. A practical model for subsurface light transport. *Proceedings of SIGGRAPH* 35 (2001), 511–518.

Jim Kajiya. 1986. The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (1986), 143–150.

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 (Proceedings of SIGGRAPH)* 33, 4 (2014).

Eric Lafortune and Yves Willems. 1993. Bi-directional path tracing. In *Proceedings of Compugraphics*. 145–153.

Stephen Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. 2003. Light scattering from human hair fibers. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 22, 3 (2003), 780–791.

Leonid Pekelis, Christophe Hery, Ryusuke Villemin, and Junyi Ling. 2015. *A data-driven light scattering model for hair*. Technical Report 15-02. Pixar Animation Studios.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2017. *Physically Based Rendering: From Theory to Implementation* (3nd ed.). Morgan Kaufmann.

Frank Suykens and Yves Willems. 2001. Path differentials and applications. *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)* (2001), 257–268.

Steve Upstill. 1990. *The RenderMan Companion*. Addison Wesley.

Eric Veach and Leonidas Guibas. 1994. Bidirectional estimators for light transport. In *Proceedings of the Eurographics Workshop on Rendering*. 147–162.

Ryusuke Villemin and Christophe Hery. 2013. Practical illumination from flames. *Journal of Computer Graphics Techniques* 2, 2 (2013), 142–155.

Magnus Wrenninge. 2016. Efficient rendering of volumetric motion blur using temporally unstructured volumes. *Journal of Computer Graphics Techniques* 5, 1 (2016).

Henning Zimmer, Fabrice Rouselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 34, 4 (2015), 131–142.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rouselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. In *Eurographics STAR Reports*.

7 Manuka, Weta's Physically-Based Spectral Renderer

Johannes Hanika, *Weta Digital*
Luca Fascione, *Weta Digital*

While the first session introduced the path tracing framework and the transport equations for a full path, these were written without wavelength dependency. Actually most of the terms such as BSDF, transmittance, emission, or sensor responsivity have spectral equivalents and depend on wavelength. Modelling this wavelength dependency as closely as possible to physical reality results in much improved fidelity, as well as better importance sampling.

7.1 Colour formation in a renderer

Mostly, the rendering equation is written without explicit dependency on wavelength λ. This is because even when doing colour or spectral transport, the equation is usually interpreted as grey transport, i.e. every wavelength can be treated independently. In the most simple case, colour is treated completely detached from the path sampling. This means the path is constructed and colour is added in by multiplying colour dependent BSDFs, transmittances, etc. Ignoring cases where path creation has a strong dependency on wavelength (we'll get to that in a bit), this boils down to multiplying chromatic factors, for instance for a simple path:

$$f(\mathbf{X}) = L_e(\lambda) \cdot G_1 \cdot f_r(\lambda) \cdot G_2 \cdot W(\lambda). \tag{9}$$

This will be integrated in the frame buffer, to yield tristimulus colour:

$$X = \int_{380..830nm} f(\lambda) \cdot \bar{x}(\lambda)\, \mathrm{d}\lambda, \tag{10}$$

where X is the first channel of the CIE XYZ tristimulus colour space and $\bar{x}(\lambda)$ is the normalised colour matching function for this channel. The Y and Z channels are computed analogously. See for instance Fairman et al. [1998] for more information on the colour matching functions.

What happens in RGB transport, when using the CIE RGB primaries, this equation will only be evaluated for three distinct wavelengths of 700 nm (red), 546.1 nm (green) and 435.8 nm (blue). It is clear that a lot of information between these wavelengths is lost because it is never evaluated. On the other hand, the transport for these wavelengths is evaluated physically correctly. The tristimulus values which end up in the frame buffer are then directly

$$R = f(\lambda = 700nm), \quad G = f(\lambda = 546.1nm), \quad B = f(\lambda = 435.8nm). \tag{11}$$

In general, using any other RGB space to perform the multiplications in Eq. (9) and replacing the integral in Eq. (10) like in Eq. (11) is wrong and will yield non-physical transport. This is especially apparent for indirect illumination. Agland [2014] performed extensive comparisons on the impact of the rendering colour space.

This is why Manuka performs all light transport computations in spectral, and only converts to a colour in the frame buffer. Many options to transport and represent spectra have been devised in literature. The simplest method is to just transport one wavelength bin for every five nanometers of spectral resolution. Since this is also the resolution of the CIE colour matching functions, the results are expected to be good.

Discretising the domain, however, theoretically introduces some bias. While this would likely not matter in this case at this resolution, it also has implications on importance sampling. Thus, Manuka transports a continuously sampled wavelength in the spirit of the Monte Carlo method. This wavelength is then used to drive the importance sampling of the path. Naturally, introducing a random variate introduces noise. The wavelength has to be chosen carefully, and we further employ path reuse and stratification to reduce colour noise (the hero wavelength scheme, as detailed by Wilkie et al. [2014]).

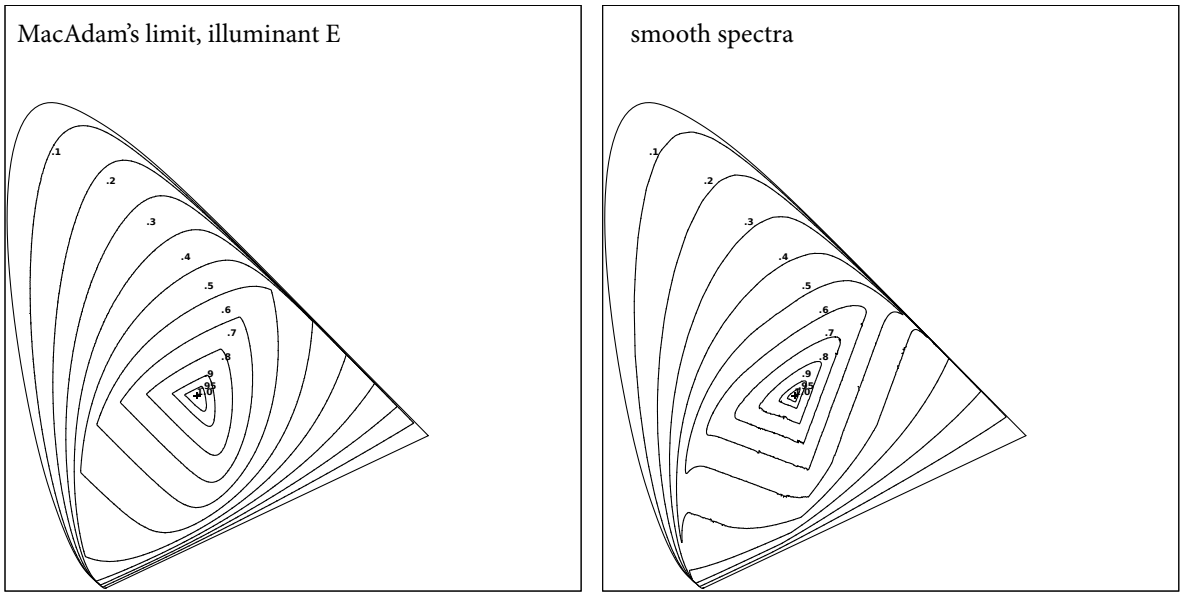


Figure 6: The maximal gamuts of surface reflection, reproduced after Meng et al. [2015]. The graphs show the maximum brightness ($X+Y+Z$) of a surface colour (for instance diffuse albedo), as an iso line graph. The coordinate system as seen from the top is the standard space of chromaticities, i.e. $x = X/(X+Y+Z)$ and $y = Y/(X+Y+Z)$. Left: the theoretical maximum which can possibly be achieved using step functions as spectra. Right: a slightly smaller gamut that can be achieved using more natural, smooth spectra.

7.2 Colour reproduction

With spectral rendering, precise colour reproduction is simple. All relevant formulas are collected in the fundamental book by Wyszecki and Stiles [2000]. Just model the light source emission, the surface reflection, and the camera responsivity with measured spectral data and the result will be correct. Modeling the spectral camera response will also give a render directly in camera RGB space rather than XYZ. This means that the render will even show the same metamerism as the live footage. There are, however, a few subtleties to keep in mind.

As often, physical plausibility has advantages and downsides. A possible downside, especially when rendering cartoons, may be that energy conservation poses a limit on colour saturation and brightness of a surface. This has been recognised early on by Schrödinger [1919] (and who are we to argue with that).

The issue is that energy conservation dictates that, in the absence of fluorescence, no wavelength λ may result in more reflected than incoming energy:

$$\int_\Omega f_r(\mathbf{x}, \omega, \omega_i, \lambda)\, \mathrm{d}\omega_i^\perp \leq 1 \quad \forall\lambda. \tag{12}$$

For a diffuse BSDF, $f_r = \rho(\lambda)/\pi$, where $\rho(\lambda)$ is the albedo, this means that $\rho(\lambda) \leq 1$ for all wavelengths λ. Now the total brightness of the surface as seen in an RGB image has something to do with the XYZ brightness, i.e. $X + Y + Z$, which is essentially the integral of $\rho(\lambda)$. Naturally, a more saturated colour means a more peaky spectral shape, which forces the integral to diminish since the maximum cannot be increased.

Fig. 6, reproduced after Meng et al. [2015], shows the limits on brightness of a surface colour ($X + Y + Z$), depending on colour saturation. As the chromaticity of the colour moves towards the edge of the spectral locus, the maximum achievable brightness becomes dimmer. The gamut shown on the left is the one derived by MacAdam [1935], who gave a constructive proof which spectra will yield the highest possible brightness for a given chromaticity. The one on the right is derived by Meng et al. [2015] and uses more natural smooth spectra. These lead to more believable indirect lighting, since the shape is usually closer to most reflectances encountered in the wild.

In the future, to add even more realism, renderers may require fluorescence to exceed the MacAdam gamut, similar to Couzin [2007]. Note that this is only an issue when such bright and saturated colours are required. For the more regular case, that realistic surface reflection needs to be reproduced, this limit of energy conservation poses a natural restriction on the look of the materials. This automatically avoids unrealistically bright and glowing surfaces. Together with smooth spectra, this results in much more life-like indirect lighting than using RGB transport.

7.3 Where to get input spectra from

We can get spectral definitions for some light sources or cameras from the manufacturers. Also some special materials, such as for instance spectral absorption of melanin (for hair) and hemoglobin (for skin) can be readily found in text books.

Even for these it is sometimes useful to be able to overrule or modulate them by artist-drawn textures. Since these are usually working in RGB, there is a need to convert tristimulus data to full continuous spectra.

Early work by MacAdam [1935] facilitates this, but with the limitation that the resulting colours will always be as bright as possible and thus box functions in shape. Since natural reflection spectra are usually smooth, this results in unnatural looking indirect lighting.

Smits [1999] devised a method to upsample RGB values to spectra, taking into account smoothness and optimising the process to try and achieve energy conservation too. Depending on the input tristimulus coordinate, it may not be possible to meet both goals: chromaticity and energy conservation. Also, this method only works for within a certain RGB working space, not for the whole gamut of visible colours. Meng et al. [2015] recognise this and separate the process into two steps: first, the colour from tristimulus values is upsampled, disregarding energy conservation. Secondly, a gamut mapping step is performed that enforces energy conservation in case the input colour was too saturated and bright for a physically plausible reflectance value. This is not needed in case a light source emission is upsampled from RGB values.

This approach ensures a surface lit by illuminant E will look the same when using the RGB reflectances and the upsampled spectrum, when observed with the CIE XYZ colour matching functions.

7.4 Colour noise

As mentioned above, introducing a randomly sampled wavelength λ into the path tracing process introduces noise. Fortunately, natural reflectance spectra are smooth, and also forced to be this during a potential upsampling step from tristimulus texture input. It is thus an effective strategy to use stratified samples in the wavelength domain to resolve colour.

Wilkie et al. [2014] do this in combination with efficient path reuse: the path construction is still performed with one main wavelength, and a set of 3 stratified wavelengths are evaluated alongside with it. The final contribution is weighted using multiple importance sampling (MIS), resulting in a much lower variance picture.

The evaluation of the PDF and wavelength-dependent BSDF can be performed in SSE, evaluating four wavelengths in four lanes in one instruction.

Note that this method requires precise computation of PDFs (that is, a stochastically evaluated or approximate PDF may lead to problems). Due to its usefulness for noise reduction we adopted this scheme in Manuka, throughout all sampling techniques. More advanced MIS techniques share the requirement on consistent PDF evaluation, so enforcing this on all our sampling techniques actually resolved a few headaches when experimenting with new path construction algorithms.

7.5 Importance sampling

While at first sight it may seem path construction can be performed independently of wavelength, there are a few important special cases.

The first is obviously chromatic dispersion in dielectrics, causing the prominent rainbow like colours in caustics, for instance under a glass of water on a table in the sun. This is one obvious effect that is hard to model in an RGB-based rendering system. On the other hand, shots with such effects are relatively rare. This is even more so because usually the visually rich materials in VFX have fine details such as scratches, grease stains on glass, or dirt particles scattering the light under water. All this blurs or masks away such subtle dispersion effects most of the time.

There are some scattering models which include a spectral shape of the lobe. This includes diffraction at surface points as well as Rayleigh scattering in the atmosphere. Using spectral sampling, it is easy for us to incorporate such advanced models into our render.

The most important case, however, is chromatic extinction in participating media. That is, the extinction coefficient $\mu_t(\mathbf{x}, \lambda)$ depends on the wavelength. This governs the transmittance term

$$\tau(t) = \exp\left(-\int_0^t \mu_t(\mathbf{x}(s), \lambda)\,\mathrm{d}s\right), \tag{13}$$

which is simply $\exp(-\mu_t(\lambda)\cdot t)$ for homogeneous media. The mean free path in the medium $1/\mu_t$ depends on the wavelength in chromatic media, resulting in very different importance sampling strategies for red vs. blue photons.

This is important for instance when using fully ray traced subsurface scattering in skin: skin has a particular look that scatters red light farther than blue light. This is the reason why black and white portrait photography looks smoother with a red filter.

The domain of distance sampling is fairly extreme: $[0, \infty)$. This means that scattering vertices will be sampled far apart when importance sampling the transmittance for different wavelengths. In some cases, when one wavelength does not interact with the medium at all, this leads to infinite variance, as recognised by [Raab et al., 2008, Sec. 3.2].

This application of spectral importance sampling is the important one for us, since it is very hard to perform principled importance sampling which can be combined in a flexible way with generic sampling strategies in RGB transport (such as combination with equi-angular sampling or sampling distances by scattering coefficient instead of extinction).

7.6 Radiometry vs. Photometry

Radiometric quantities (such as watts for flux or watts/square meter/steradian for radiance) are great to work with during light transport, since they allow a 1:1 mapping to the equations we find in physics books.

For a lighter, however, it may be more intuitive to work with photometric quantities. These account for the fact that different colours appear to be of different brightness for a human observer. To be precise, a spectral power distribution can be converted from radiometric quantities to photometric ones by weighting by a luminousity function. Usually the photopic, daytime brightness function of the CIE is used. This allows us to express radiant power not in watts but as *lumen*, which is then called luminous power, for instance. For all radiometric quantities, there are equivalent photometric ones (cf. Tab. 1). Designing user interfaces for lighters around this notion allows them to change the colour of a light source while maintaining the perceived brightness in a principled way.

As said earlier, when dealing with spectral light sources, the photopic luminosity function $\bar{y}(\lambda)$ is used: this is the result of a series of experiments and tabulations first published by the International Commission on Illumination (CIE) in 1924 (the function was called $V(\lambda)$ at the time) and then included in the color matching functions for the standard 2 degree colorimetric observer, published in 1931.

At this point we have enough information to write equations correlating radiometric quantities to their corresponding photometric ones: given a radiometric spectral quantity $X_\lambda(\ldots, \lambda)$ the corresponding photometric quantity $X_v(\ldots)$ is simply obtained integrating X_λ against $K_m \cdot \bar{y}(\lambda)$ where K_m is a scaling

Radiometric spectral				Photometric			
name		unit	symbol	name		unit	symbol
Radiance		$W/(m^2 \cdot sr \cdot m)$	L_λ	Luminance	*nit*	$nt = lm/(m^2 \cdot sr)$	L_v
Irradiance		$W/(m^2 \cdot m)$	E_λ	Illuminance	*lux*	$lx = lm/m^2$	E_v
Radiosity		$W/(m^2 \cdot m)$	J_λ	Luminosity	*lux*	$lx = lm/m^2$	J_v
Radiant emittance		$W/(m^2 \cdot m)$	M_λ	Luminous emittance	*lux*	$lx = lm/m^2$	M_v
Radiant intensity		$W/(sr \cdot m)$	I_λ	Luminous intensity	*candela*	$cd = lm/sr$	I_v
Radiant power	*watt*	W/m	Φ_λ	Luminous power	*lumen*	lm	Φ_v
Radiant energy	*joule*	$J/m = W \cdot s/m$	Q_λ	Luminous energy	*talbot*	$Tb = lm \cdot s$	Q_v

Table 1: Correspondence between radiometric and photometric units. We abbreviate the unit for luminous energy *talbot* as *Tb* instead of the also common *T* to avoid confusion with the unit for magnetic flux *tesla*. We also use the convention of subscripting photometric quantities with v (for *visual*), radiometric quantities with e (for *energetic*) and spectral radiometric quantities with λ. this follows the recommendations in documents such as USAS and ASME [1967].

constant about equal [1] to 683:

$$X_v(\ldots) = K_m \int X_\lambda(\ldots, \lambda)\bar{y}(\lambda)d\lambda.$$

For example, given spectral radiant power $\Phi_\lambda(\lambda)$, the corresponding luminous power Φ_v is

$$\Phi_v = K_m \int \Phi_\lambda(\lambda)\bar{y}(\lambda)d\lambda.$$

7.7 Conclusion

Spectral rendering is an integral part of the Manuka renderer, and one we wouldn't want to roll back. The hero wavelength scheme ensures that it almost doesn't cost us anything in terms of performance when compared to RGB transport. We have seen that employing RGB transport is in general not computing physically based light transport, and can lead to visibly wrong indirect lighting and high variance. As a VFX studio, we care a lot about a precise match of render and plate, and spectral rendering helps us to achieve this: using measured light source spectra, camera responsitivities, and advanced material models.

References

Steve Agland. 2014. CG Rendering and ACES. http://nbviewer.ipython.org/gist/sagland/3c791e79353673fd24fa. (2014).

CIE. 1996. *The Basis of Physical Photometry*. Commission Internationale de l'Éclairage, CIE Central Bureau.

Dennis Couzin. 2007. Optimal fluorescent colors. *Color Research & Application* 32, 2 (2007), 85–91.

Hugh Fairman, Michael Brill, and Henry Hemmendinger. 1998. How the CIE 1931 color-matching functions were derived from Wright-Guild data. *Color Research and Application* 22, 1 (1998), 11–23.

David L. MacAdam. 1935. Maximum Visual Efficiency of Colored Materials. *Journal of the Optical Society of America* 25, 11 (1935), 361–367.

[1] The scaling constant K_m is actually closer to 683.002, because the value of $\bar{y}$ is about 0.999 998 at 555.016 *nm*, but the value of 683 can safely be used for all practical applications CIE [1996], Wyszecki and Stiles [2000]

Johannes Meng, Florian Simon, Johannes Hanika, and Carsten Dachsbacher. 2015. Physically Meaningful Rendering using Tristimulus Colours. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34, 4 (June 2015), 31–40.

Matthias Raab, Daniel Seibert, and Alexander Keller. 2008. Unbiased Global Illumination with Participating Media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. 591–606.

Erwin Schrödinger. 1919. Theorie der Pigmente größter Leuchtkraft. *Annalen der Physik* 367, 15 (1919), 603–622.

Brian Smits. 1999. An RGB-to-spectrum conversion for reflectances. *Journal of Graphics Tools* 4, 4 (1999), 11–22.

USAS and ASME. 1967. *USA Standard Letter Symbols for Illuminating Engineering*. United States of America Standards Institute.

Alexander Wilkie, Sehera Nawaz, Marc Droske, Andrea Weidlich, and Johannes Hanika. 2014. Hero Wavelength Spectral Sampling. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 33, 4 (July 2014), 123–131.

G. Wyszecki and W. S. Stiles. 2000. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons.

8 Recent Advancements in Disney's Hyperion Renderer

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, Daniel Teece[2]

Figure 7: A production frame from *Moana*, rendered using Disney's Hyperion Renderer. (Copyright © Disney Animation 2017.)

8.1 Introduction

Path tracing at Walt Disney Animation Studios began with the Hyperion renderer, first used in production on *Big Hero 6*. Hyperion is a custom, modern path tracer using a unique architecture designed to efficiently handle complexity, while also providing artistic controllability and efficiency.

The concept of physically based shading at Disney Animation predates the Hyperion renderer. Our history with physically based shading significantly influenced the development of Hyperion, and since then, the development of Hyperion has in turn influenced our philosophy towards physically based shading.

8.1.1 History of Physically Based Rendering at Disney Animation

A major theme in the past decade of rendering at Disney Animation has been the advantages of physically based solutions over biased approximations, for both visual richness and artistic controllability. Early successes with physically inspired hair shading on *Tangled* led to the development of the Disney BRDF by Burley [2012] during *Wreck-It Ralph*, and was subsequently extended into the modern Disney BSDF with subsurface scattering and refraction during *Big Hero 6* (as described by Burley [2015]). Adopting physically meaningful parameters made shader response more predictable and intuitive for artists.

At the same time, moving from REYES-style rasterization rendering to physically based path tracing has removed the considerable data management overhead imposed on artists to manage the shadow maps, point clouds, and more that rasterization rendering necessitated. We continue to strive for ease of use by simplicity and consistency—"it just works".

8.1.2 Inception of Disney's Hyperion Renderer

The Hyperion renderer was developed at Walt Disney Animation Studios with the aim of providing global illumination within a physically based framework while retaining the benefits of highly coherent shading

[2] Author names after Brent Burley are presented in alphabetical order by last name.

Figure 8: A production frame from *Zootopia*, rendered using our fully path-traced fur model. (Copyright © Disney Animation 2017.)

shown in previous production-proven rasterization-based strategies. Beginning in 2011 and continuing through 2012, an initial period of research and exploration was followed by a prototype and proof-of-concept stage where successful ideas were tested at a production scale. The results were compelling enough to drive development into a full production renderer over a short time period from 2013 to 2014, coinciding with the production of *Big Hero 6*. Hyperion has subsequently been used to render the feature films *Zootopia* and *Moana*, and is being used to render all projects currently in production at the studio.

The core of Hyperion's architecture is *sorted deferred shading*. Starting with primary rays we perform ray sorting, binning rays by direction and grouping them into large, sorted ray batches of fixed size. Next, we perform scene traversal, one sorted ray batch at a time. We use a two-level quad BVH with streaming packet cone traversal in the top level, and single-ray traversal in the bottom. We exploit the fact that our ray batches are directionally coherent to perform approximate front-to-back traversal at each node. The result of traversal is a list of hit points, one per ray. Next, hit point sorting organizes ray hits with the aim of maximizing coherent access from the texture cache. If a shading task has many hit points, it is partitioned into sub-tasks, further increasing parallelism. The shader also feeds secondary rays back into ray sorting to continue ray paths. Increasing the batch size provides improved coherence and better performance for traversal and shading. A more detailed description of this architecture is presented by Eisenacher et al. [2013]

The introduction of Hyperion has produced numerous benefits across the studio. Due to the ease of use, more departments can render full frames, and we now render shots continuously in all stages of production with full global illumination. Higher quality rendering in all stages of the production process provides a much earlier view into the look of each show. Because of the predictability of the results, artists are able to final frames in fewer iterations than before path tracing. As the complexity of our films continues to increase, the Hyperion renderer grows and evolves to meet the unique challenges of each show.

8.2 Transitioning from Multiple Scattering Approximations to Brute-force Solutions

Historically, phenomena requiring large amounts of multiple scattering were prohibitively computationally expensive to evaluate using physically correct brute-force solutions, so approximations were typically used instead. Many of our projects since *Big Hero 6* have required complex multiple-scattering effects, such as the fur in *Zootopia*, snow in *Frozen Fever*, various cases in *Moana*, and volumes in upcoming shows. Moving to path tracing has allowed us to discard previous approximate solutions and move to-

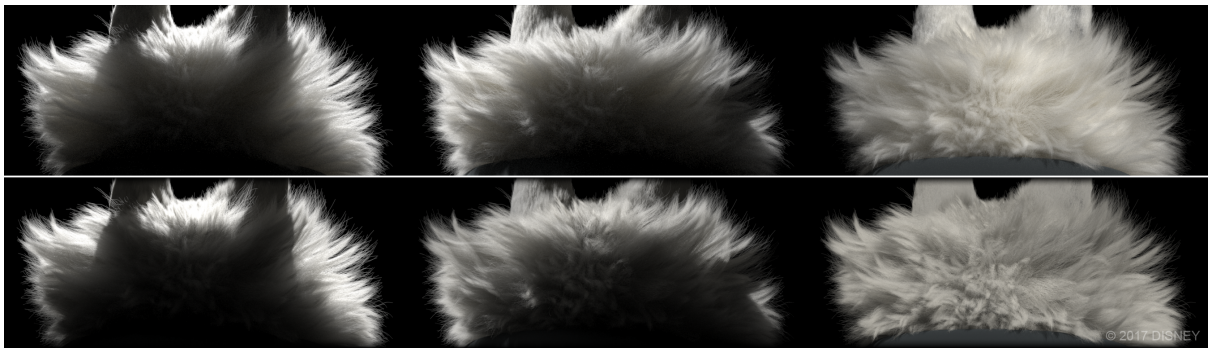


Figure 9: Fur rendered with path-traced multiple fiber scattering (top) vs. its Dual Scattering approximation, both using the same lighting setup and same absorption coefficients in the fur strands. For more details, we refer the reader to Chiang et al. [2016a]. (Copyright © Disney Animation 2017.)

wards brute-force solutions for these effects. In this section, we describe some of these phenomena, and also discuss some of the careful parameterization work that is often necessary to make these models more artist-friendly.

8.2.1 Path-traced Hair and Fur

Prior to *Zootopia*, we used an artistically controlled Dual Scattering hair model originally developed for *Tangled* by Sadeghi et al. [2010]. While this model was more physically inspired than previous ad hoc models, we found that the Dual Scattering model lacked the richness that multiple scattering already provided in other subsystems of the Hyperion renderer. The lack of multiple scattering in fur and hair often contributed to a coarse and stiff look that became amplified with the presence of high albedo fibers (Figure 9). In order to address this problem, we came up with a physically based single fiber scattering model that allows for efficient Monte Carlo rendering of path-traced multiple fiber scattering in production (Chiang et al. [2016a]).

One common sampling strategy for previous physically based fiber scattering models (such as the one proposed by d'Eon et al. [2011]) is to focus on eliminating the shading variation across the width of a fiber. However, often in production path-traced global illumination, a more prominent source of sampling variance per fiber is the illumination coming from the complex surrounding scene, as well as illumination coming from all nearby fibers. This outside illumination requires a large number of shader evaluations to fully converge. We realized that by relying on the general Monte Carlo framework to integrate over fiber width, we can greatly reduce the complexity of the per-sample evaluation. We also introduced a fourth lobe that re-injects the energy lost from only representing R, TT and TRT interactions to achieve perfect energy conservation, even for non-absorbing fibers. These improvements made brute force path tracing of fur and hair possible in production, and significantly contributed to the look of *Zootopia* (Figure 8).

One issue with a physically based model is that its parameters, such as the absorption coefficient, are often not intuitive for the artists. Also, physically based parameters can have very little visual connection with the final material appearance, which comes from the result of multiple scattering. To address artistic controllability, we re-parameterized the fiber roughness to be perceptually uniform. We also allow the artists to specify multiple scattering albedo directly, which is used internally to derive the absorption coefficient for rendering. These enable efficient artist workflows while remaining physically consistent, empowering the artists to achieve wider ranges of appearances of hair and fur with great efficiency.

We continue to use the techniques described in this section for all productions beyond *Zootopia*, to great success. For example, in *Moana*, human characters are covered in fine, groomed peach fuzz, which provides effects such as rim lighting through brute-force multiple scattering of back lighting instead of requiring a dedicated light type (Figure 10).

Figure 10: Rim lighting effects on Moana and Grandma Tala, simulated through brute-force multiple scattering of back-lighting through fine hairs. (Copyright © Disney Animation 2017.)

8.2.2 Path-traced Subsurface Scattering for Snow and Skin

Subsurface scattering is the phenomenon of light scattering inside an object and exiting at a different place than it entered. This phenomenon produces effects like softness, light bleeding, and shadow saturation. Preventing translucent materials like skin and snow from looking too opaque or hard is essential.

For years, the diffusion approximation was the method of choice. During *Big Hero 6*, we used *normalized diffusion*, introduced by Burley [2015], as our primary subsurface scattering solution. To simplify the problem, most diffusion approximations assume that the medium is a semi-infinite slab of homogeneous material. Diffusion works well even when the geometric assumption is violated, but only if the distance that the light scatters is small compared to the size of the geometric details on the surface. In geometrically small and thin regions, diffusion causes energy loss, while on convoluted surfaces, diffusion causes energy gain. Furthermore, interesting optical effects and important visual cues caused by light scattering through objects of different sizes and shapes are lost when using diffusion.

The physical process of subsurface scattering through arbitrary geometry can be simulated much more accurately using volumetric path tracing. However, since light can potentially scatter hundreds or thousands of times inside an object before exiting, this approach has the potential be extremely computationally expensive. Furthermore, highly directional scattering and small bright light sources increase noise.

We experimented with several approaches to improve the appearance of snow in *Frozen Fever*, and ended up implementing and using a limited brute-force volumetric-path-tracing solution (Figure 11). A number of design decisions were made to circumvent performance problems:

- We performed simulated scattering inside only a user-defined subset of the scene surfaces.
- We assumed that the volumes were completely homogeneous.
- We performed free-flight sampling according to a monochromatic scattering coefficient and calculated the amount of chromatic absorption based on the full path length.

Figure 11: A production frame from *Frozen Fever* with characters made from path-traced subsurface-scattering snow. (Copyright © Disney Animation 2017.)

- We only supported isotropic phase functions and only index-matched diffusely-transmitting interfaces.
- We introduced a hack to increase scatter distances after several hundred bounces.

Although this system worked well, it was very difficult to achieve a desired overall color and desired scatter distances for each color channel. After a good deal of research (which resulted in Koerner et al. [2016]), we came up with an artist-friendly parameterization for setting chromatic scattering and absorption coefficients and a sampling strategy that efficiently handled these chromatic coefficients (Chiang et al. [2016b]). The parameterization is based on fitting curves to sets of simulated results. The sampling strategy uses the path throughput and single-scattering albedo to bias free-flight distributions. We also introduced internal reflection to make results more accurate and reduce unrealistic brightening of edges.

We first used our current path-traced subsurface scattering solution on selected prop elements in *Moana*, although skin for characters continued to use normalized diffusion. When comparing normalized diffusion and path-traced subsurface scattering on *Moana* characters, we discovered that details such as creases and wrinkles had been modeled deeper than expected to compensate for detail loss that occurs from diffusion, which produced different looks with path-traced subsurface scattering. All of our current productions have fully switched over to path-traced subsurface scattering for everything from snow to skin, which has simplified modeling and shading workflows because compensations for diffusion artifacts no longer need to be modeled into geometry. We show an example of a test character rendered with path-traced subsurface scattering skin in Figure 12.

8.2.3 Volume Rendering

Hyperion's sorted deferred architecture provides a significant challenge for implementing volumetric rendering. During *Big Hero 6*, a volume-rendering system was developed that was heavily designed around the sorted deferred concept. Up until recently, one fundamental requirement of the sorted deferred architecture was that a ray must hit a surface before a new ray could be fired. In our current volume-rendering system, Hyperion traces a ray completely through a heterogeneous volume, calculating a transmittance estimate using residual-ratio tracking (Novák et al. [2014]) which is followed by constructing PDFs to sample in-scattering and emission. In-scattering rays are generated to be treated like any other ray and are added to the list of rays for processing in the sorted deferred shading queue. Multiple scattering is achieved by recursing on the procedure.

Figure 12: Skin rendered using path-traced subsurface scattering. For similar render times as our old normalized diffusion technique, our new path-traced subsurface scattering provides richer visual quality and better predictability. (Copyright © Disney Animation 2017.)

Figure 13: A production frame from *Moana* demonstrating large, complex, dense smoke plumes dominated by low-order scattering. Our current residual-ratio tracking based volume rendering system is not as efficient with high-order multiple scattering, but handles low-order scattering very efficiently. (Copyright © Disney Animation 2017.)

Figure 14: A cloudscape with high-albedo clouds and thousands of multiple-scattering bounces, rendered using our new brute-force volume-rendering system. Our new volume-rendering system makes use of our spectral and decomposition tracking techniques, introduced by Kutz et al. [2017]. (Copyright © Disney Animation 2017.)

This solution produces high quality estimates at high compute costs per sample and is optimized for low-order scattering, such as in smoke plumes and dust clouds with low albedos (Figure 13). This approach avoided major modifications to the core sorted deferred architecture. However, building a high quality PDF per ray makes high-order multiple scattering with potentially tens of thousands of bounces unfeasibly expensive. Initially, efforts were made to rely on only Hyperion's existing volume-rendering solution, and fill in missing energy from high-order scattering using various approximations and cheats. These efforts failed; artists found these approximate techniques difficult to control and were not able to achieve the highly realistic look they were targeting.

Starting in 2016, a project arose within the studio that required rendering enormous quantities of high-albedo clouds with very high-order multiple scattering. We are in the process of transitioning into a new volume system that allows us to render thousands of scattering events per path. To make such long paths more practical to render, we derived new, advanced versions of tracking (Kutz et al. [2017]). Architecturally, this new volume renderer is made possible by changes within the sorted deferred system that remove the requirement for a ray to always end at a surface, meaning that the volume renderer can immediately re-scatter a ray upon finding a scattering event. The brute-force and therefore predictable nature of our new volume-rendering system has allowed artists to hit their target looks with significantly greater ease than before, while also providing significantly faster iteration and feedback loops. In Figure 14, we demonstrate an example of a scene with many clouds that Hyperion's new volume rendering system handles with ease.

8.3 Future Directions

Over the course of three feature films and several more short films, we have gained significant experience with both general path tracing, and the practicalities of our sorted deferred architecture. Using lessons learned from production, we continue to evolve Hyperion's architecture going forward.

8.3.1 Unbounded Path Lengths

Hyperion has allowed us to universally adopt multi-bounce global illumination studio-wide with significant efficiency, up to a certain number of bounces. However, Hyperion's architecture imposes some interesting restrictions that shaped our light sampling strategy, along with our ability to support truly unbounded path lengths efficiently.

Hyperion originally only supported a single ray type, which makes direct light sampling via next-event-estimation difficult to implement. Instead of using next event estimation, we relied on explicit light sampling, splitting rays at each scattering event so that we could shoot separate samples towards lights and along BSDFs. This splitting approach alone results in a geometric increase in the number of rays at each bounce, which places considerable memory pressure on a system that already keeps tens of millions of rays in flight at once. To keep the total number of rays in flight manageable, we rely on an aggressive Russian Roulette strategy to cull rays.

While this approach has generally worked well for us, it has some consequences at higher bounces. As we reach higher and higher bounces, our aggressive Russian Roulette begins to dominate over splitting, resulting in large drops in ray counts. We eventually reach a point where there are too few rays in flight simultaneously to justify the overhead of our sorted deferred ray batches, meaning that paths with extremely high lengths can become inefficient to compute.

One current active area of development is loosening our requirement for a single ray type, allowing us to replace our existing sampling strategy with a more conventional next-event-estimation approach. Changing the sampling strategy changes the relationship between samples-per-pixel and variance, which presents an interesting user-education topic. To make unbounded paths more efficient, we are also examining ways to loosen the definition of a ray batch, along with different methods for scheduling ray batches. We plan to have more to report on this topic in the near future.

8.3.2 Leveraging Increasing Core Counts

As the complexity of our films continues to grow, we anticipate our studio's rendering needs to outstrip the increases in computational power predicted by Moore's Law in the near future. As a result, we are concerned not just about scaling Hyperion to increasingly more cores on a single render node, but also about distributing and scaling both memory and compute for a single render job beyond the bounds of a single render node. This topic continues to be an active area of research that we hope to be able to report more on in the near future.

8.4 Conclusion

We presented a series of recent advancements made in Disney's Hyperion Renderer, with a particular focus towards replacing multiple scattering approximations with true, brute-force path-traced solutions. Adopting a path-tracing renderer studio-wide has allowed us to pursue effects and features that were previously considered completely infeasible to solve using brute-force methods in production, all while also providing simpler, more intuitive controls for artists. We continue to investigate ways to evolve our architecture and increase the scalability and efficiency of the Hyperion renderer even further, in support of even more future advancements to production quality and artist controllability.

References

Brent Burley. 2012. Physically Based Shading at Disney. *Practical Physically-Based Shading in Film and Game Production, SIGGRAPH 2012 Course Notes* (July 2012).

Brent Burley. 2015. Extending Disney's Physically Based BRDF with Integrated Subsurface Scattering. *Physically Based Shading in Theory and Practice, SIGGRAPH 2015 Course Notes* (July 2015).

Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2016a. A Practical and Controllable Hair and Fur Model for Production Path Tracing. *Computer Graphics Forum (Proc. of Eurographics)* 35, 2 (May 2016), 275–283.

Matt Jen-Yuan Chiang, Peter Kutz, and Brent Burley. 2016b. Practical and Controllable Subsurface Scattering for Production Path Tracing. In *SIGGRAPH 2016 Talks*. 49:1–49:2.

Eugene d'Eon, Guillaume Francois, Martin Hill, Joe Letteri, and Jean-Marie Aubry. 2011. An Energy-Conserving Hair Reflectance Model. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 30, 4 (June 2011), 1181–1187.

Christian Eisenacher, Gregory Nichols, Andrew Selle, and Brent Burley. 2013. Sorted Deferred Shading for Production Path Tracing. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 32, 4 (July 2013), 125–132.

David Koerner, Jan Novák, Peter Kutz, Ralf Habel, and Wojciech Jarosz. 2016. Subdivision Next-Event Estimation for Path-Traced Subsurface Scattering. In *Eurographics Symposium on Rendering 2016: Experimental Ideas and Implementations*. 91–96.

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 36, 4 (July 2017), 111:1–111:16.

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 33, 6 (Nov. 2014), 179:1–179:11.

Iman Sadeghi, Heather Pritchett, Henrik Wann Jensen, and Rasmus Tamstorf. 2010. An Artist Friendly Hair Shading System. *ACM Transactions on Graphics* 29, 4 (July 2010), 56:1–56:10.

9 Hello Moonray! – A new production path tracer

Brian Green, *DreamWorks*

Moonray is DreamWorks Animation's new production path tracer. It was written from scratch over the last four years. It it built on top of some of the most well known open-source components available for path tracing. These include Embree Wald et al. [2014], OpenImageIO Gritz [2008], ISPC Corpoation [2012], LLVM Lattner [2002], and OpenShadingLanguage Gritz [2009]. Its design is heavily influenced by the ideas described in Pharr et al. [2017].

Moonray, although now a true production renderer, was not initially developed with production rendering as its primary goal. Its initial goal was to provide a near real-time rendering system that could be deployed in a cloud using a SAS (software-as-a-service) model. The system needed to meet the following criteria:

1. **Easy to use** The system needed to produce good looking photo-realistic images with a minimum of user setup. The users were not likely to be production artists with expertise in multi-pass rendering or complex shading models.
2. **Easy to integrate** The renderer needed to be easy to integrate with many different 3rd party authoring and lighting tools.
3. **Highly scalable**. Broadly speaking we interpreted this to mean that the more hardware you gave us, the faster the renderer would go.

DreamWork's existing micro-polygon based rendering system, Moonlight, while powerful and production proven, failed to meet these most basic criteria. To achieve good results with decent performance, many complex pre-passes needed to be setup. For integration, fairly heavy, not easily portable libraries needed to be directly linked into any 3rd party application. But worst of all, the system just did not scale that well as core and machine counts rose.

Path tracing checked all the above boxes. Using simple, physically based shaders, users could quickly achieve photorealistic results with no parameter tweaking at all. Path tracing is embarassingly parallel, and Moonray easily scales to thirty or more machines with 32+ cores. A cloud deployment strategy requires only a thin "SDK" wrapper library be used in the client.

After four years of development, it became clear at DreamWorks that Moonray was ready to take the leap and become our next production renderer. What we initially thought was only important to non-artists, actually turned out to be pretty important to artists as well. What follows is a sampling of what we think make Moonray a unique (and not at all unique!) addition to the production path tracing universe.

9.1 Our basic path tracing algorithm

Our basic algorithm is most compactly described as "backwards path tracing with multi-importance sampling." But a brief overview of what we precisely mean by this will help in understanding the next sections and remove possible ambiguities. See Veach [1998], for a detailed description of multi-importance sampling.

Broadly speaking we divided the rendering process up into two distinct pieces:

1. **Render Preparation** This is where geometry procedurals are run and a fully tessellated and displaced BVH is built. More generally, any work that can be done once per-frame and used in the next phase takes place here.
2. **Mcrt Rendering** During this phase we cast rays into the scene accumulating returned radiance values (and other outputs) into the various frame buffers.

After render prep, rays are generated from the camera and cast into the scene. When they hit a surface, a material shader is run which produces a Bsdf (bi-directional scattering distribution function) which gives us two important methods given a viewing (wo) direction

1. **Next ray direction, wi**. We can randomly determine a direction for the next ray along the path, as well as the relative importance of this direction (i.e the value of the probability density function) of this path.
2. **Reflected radiance**. Given wo, and wi, we can compute the reflected radiance back along the wo direction.

The artist controls the number of surface samples at the hit point. But only for the first non-mirror bounce. After the first non-mirror bounce, we use only a single surface sample.

In addition to surface samples, which are chosen based on the surface Bsdf, we also evaluate samples based on the lights. Given a light, a surface position, and an outgoing wo direction, the light provides two functions similar to the Bsdf which randomly generate a wi direction with importance probability and an outgoing radiance value from the light along the wi direction.

The artist controls the number of light samples taken at a hit point. But as with surface samples, after the first non-mirror bounce only (and exactly) one sample per active light is used.

A sample wi direction may do one of the following 3 things:

1. **Hit nothing**. If the wi direction hits nothing, then there is no energy and the path is terminated
2. **Hit a light**. If the wi direction has an unoccluded view of a light, then the sample is a "direct lighting" result. The Bsdf value is multiplied by the light value along with the MIS heuristic to produce a final value for the sample.
3. **Hit a geometry**. If the wi direction hits another geometry, then the sample is lit indirectly based on the radiance value computed recursively at the wi/geometry intersection point.

The artist has control on the amount of path depth recursion. In fact, we further break these controls up based on the type of surface we are evaluating (diffuse, glossy, or mirror).

9.2 Vectorization

Moonray has, what we believe to be, a novel approach to vectorized path-tracing. "Keeping all the lanes of all the cores busy all the time with meaningful work" has been our mantra.

At the time of this writing, a paper describing Moonray's vectorization approach has been submitted to HPG '17. Pending acceptance and its presentation, we are unable to provide details of the algorithm in these notes. But at the time of the course, we will present an overview of the algorithm and the results.

9.3 Arbitrary output variables

Moonray has a full featured system for *arbitrary output variables* (AOVs). In our experience, AOVs serve two primary roles in production: diagnostics, and compositing work flows.

For diagnostic purposes, Moonray has introduced a "Material AOV" syntax that is used to extract material properties at a primary ray intersection point:

[('<GL>')+\.][('<ML>')+\.][('<LL>')+\.][(SS|R|T|D|G|M)+\.][fresnel\.]<property>

<GL>, <ML>, and <LL>: user specified labels
SS|R|T|D|G|M: types of bsdf lobes
fresnel qualifies property
<property> are diagnostic parameters such as roughness, normal, or color.

For example 'spec'.MG.roughness specifies an AOV that is the average roughness of all mirror and glossy Bsdf lobes that also have the 'spec' label assigned.

For compositing work flows, we make use of light path expressions as defined by the OpenShadingLanguage distribution. The important distinction we make between material AOVs and lighting AOVs is that former has no interaction with lighting while the later necessarily involves some light integration and always returns a radiance value.

9.4 Automatic differentiation

One key difference between production rendering systems and real-time systems is the near ubiquitous use of derivatives (or more generically areas) during shading. Evaluating shaders over ray-differential areas is needed to properly select MIP levels for texture mapping, avoid temporal bump mapping noise, and many other reasons.

For the most part, our shaders operate using dual numbers of three variables. This allows the shader code to compactly compute not just its value, but also the partial derivative of its value with respect to the ray differential (dx, dy, and dz). See Piponi [2004] for details.

Of course, these additional computation does come at a cost, which segues nicely into our last section.

9.5 Just in time (JIT) compilation

At compile time, our shaders are compiled from ISPC into LLVM bitcode. At run-time we use the LLVM API to analyze and perform the following operations:

1. **Shader attribute values**. We replace all attribute value references with the actual values set by the user (saves function calls, allows constant folding).
2. **Shader connections**. Shaders are built in isolation, but connected together in a network by the artist. At the connection points, which are basically just function pointer dereferences, we replace these calls with actual inline code. This allows the optimizer to see the entire network.
3. **Autodiff**. If the code calling a shader does not use the derivatives computed by the shader, the computations associated with the derivatives can be optimized out.

After we have finished the analysis and code substitutions, we run the LLVM code optimizer and backend to produce a callable shade function. This occurs only once per frame during the render preparation phase. Intuitively, the more information you provide to an optimizing compiler, the better job it can do optimizing your code. By delaying the final compilation until run-time, we give the compiler the maximal possible information to work with.

References

Intel Corpoation. 2012. Intel SPMD Program Compiler. https://ispc.github.io/. (2012).

Larry Gritz. 2008. Open Image I/O. https://github.com/OpenImageIO/oiio/. (2008).

Larry Gritz. 2009. Open Shading Lanaguage. https://github.com/imageworks/OpenShadingLanguage/wiki/OSL-Light-Path-Expressions. (2009).

Chris Lattner. 2002. *LLVM: An Infrastructure for Multi-Stage Optimization*. Master's thesis. Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL. *See* http://llvm.cs.uiuc.edu.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2017. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc.

Dan Piponi. 2004. Automatic Differentiation, C++ Templates, and Photogrammetry. *Journal of graphics, GPU, and game tools* (2004), 41–55.

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Trans. Graph.* (2014), 143:1–143:8.