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.

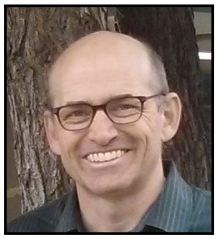
3.4 Per Christensen, Pixar

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

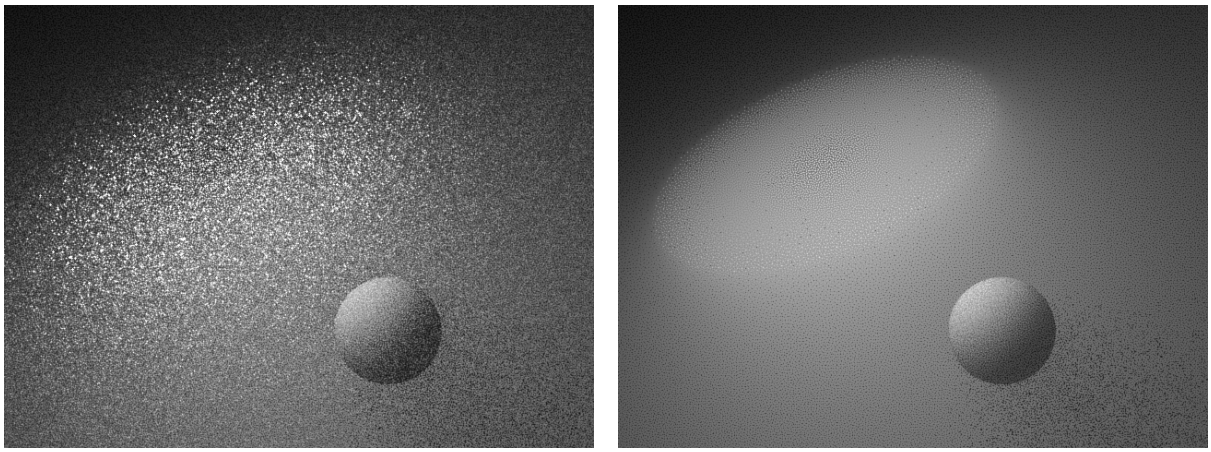


Figure 2: These images show next event estimation in a participating medium, lit by a disk-shaped area light source. Left: naive area sampling compared to right: importance sampling of the solid angle. This technique is described in detail by Guillén et al. [2017].

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.

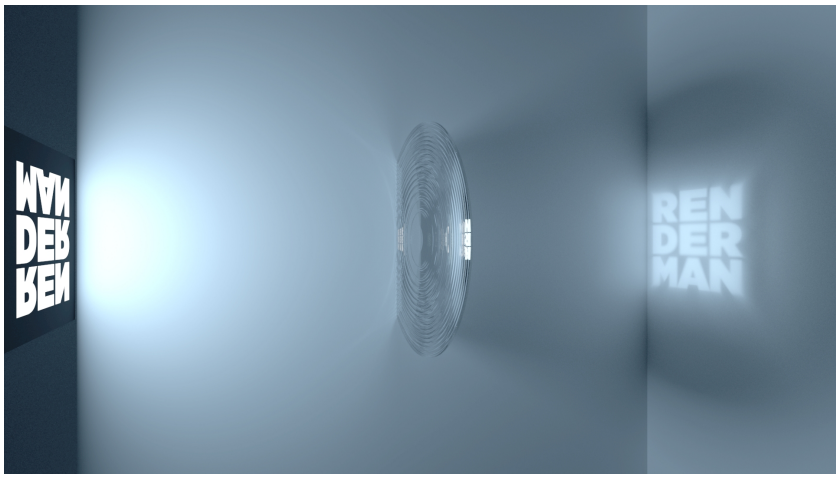


Figure 4: VCM rendering of light from a textured light source being refracted through a Fresnel lens and focused on a diffuse wall. (Image courtesy of Andrew Kensler.)

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.

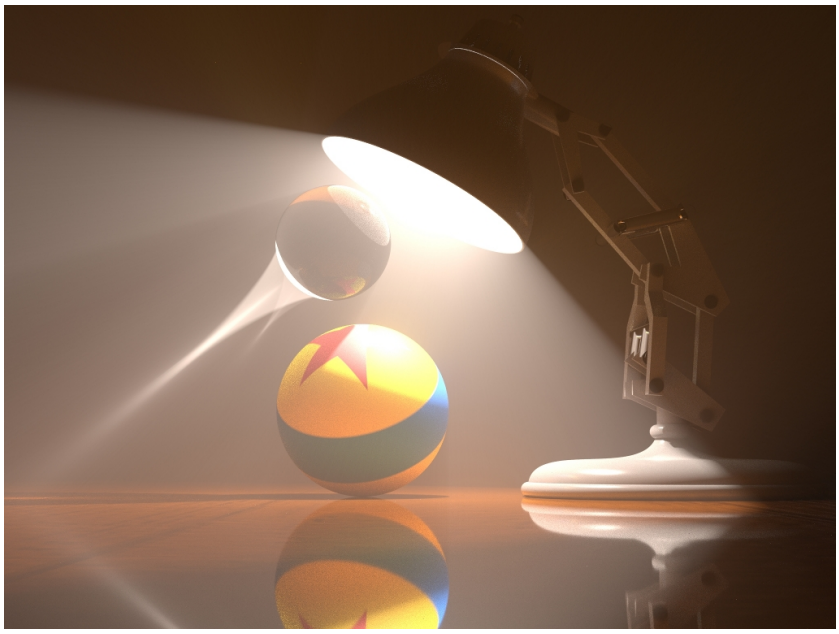


Figure 5: UPBP variation on the classic Luxo Jr. scene. Note the volume caustic under the glass sphere and the reflection in the table of the volume caustic. (Image credit: Brian Savery, Martin Sik, and Per Christensen.)

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]).

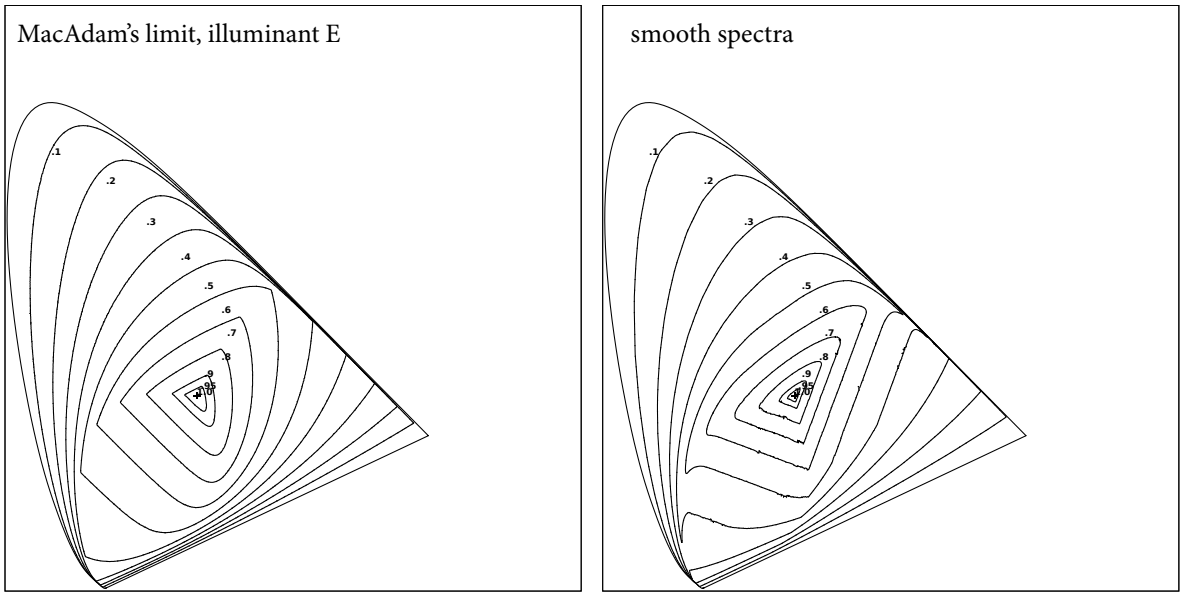


Figure 6: The maximal gamuts of surface reflection, reproduced after Meng et al. [2015]. The graphs show the maximum brightness ($X+Y+Z$) of a surface colour (for instance diffuse albedo), as an iso line graph. The coordinate system as seen from the top is the standard space of chromaticities, i.e. $x = X/(X+Y+Z)$ 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-

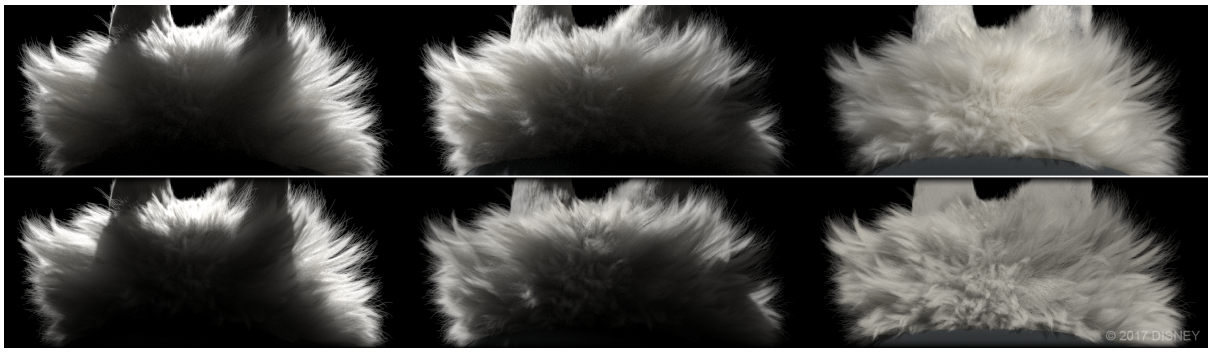


Figure 9: Fur rendered with path-traced multiple fiber scattering (top) vs. its Dual Scattering approximation, both using the same lighting setup and same absorption coefficients in the fur strands. For more details, we refer the reader to Chiang et al. [2016a]. (Copyright © Disney Animation 2017.)

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.

Path tracing in Production

Part 2: Making movies

SIGGRAPH 2017 Course Notes

Fig. 1: Examples for the rich worlds that challenge physically-based rendering engines in movie production. Top row: Stills from The Jungle Book, left: MPC and right: Weta Digital (copyright ©2016, Walt Disney Pictures). Bottom Row: Rogue One: A Star Wars Story (ILM, copyright ©2016 Lucasfilm Ltd. All Rights Reserved). Bottom right: Gotham city, completely built from individual bricks and rendered with Animal Logic's Glimpse for The LEGO Batman Movie. Image ©Warner Bros Inc., The LEGO Corporation. All rights reserved.

http://dx.doi.org/10.1145/3084873.3084906

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. While path tracing reached most VFX houses and animation studios at a time when a physically-based approach to rendering and especially material modelling was already firmly established, the new tools brought with them a whole new balance, and many new workflows have evolved to find a new equilibrium. Letting go of instincts based on hard-learned lessons from a previous time has been challenging for some, and many different takes on a practical deployment of the new technologies have emerged. While the language and toolkit available to the technical directors keep closing the gap between lighting in the real world and the light transport simulations ran in software, an understanding of the limitations of the simulation models and a good intuition of the tradeoffs and approximations at play are of fundamental importance to make efficient use of the available resources. In this course, the novel workflows emerged during the transitions at a number of large facilities are presented to a wide audience including technical directors, artists, and researchers.

This is the second 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

As the movie making industry moves towards rendering using path tracing, the objective of this course is to provide the audience with insight into the challenges posed by the new paradigm, in a comparison to the familiar world of rasterization-based pipelines.

The speakers span a wide range of stories, both from VFX houses and animation studios, covering a variety of different adoption stories, in terms of length, depth and composition.

While the path tracing approach comes with a promise of sweeping simplifications at the global workflow scale of a facility, the large bag of tools of the trade that the Technical Directors had accumulated over the years has lost its efficacy, and the many techniques that were once available are now in need to be deconstructed, carefully analyzed, and rebuilt in the new context.

The fundamental trait of path tracing-based light transport, in which light "just behaves like in the real world" has at first occasionally turned out to be a double edged sword. The basic tools of the old days, such as matte objects, shadow-casting proxy geometry, shadow-less fill lighting, have analogs in the new world with very different trade-offs in terms of performance optimisation.

New phenomena are part of the path tracing world, possibly the most prominent being various forms of Monte Carlo noise, which in turn requires all users to be versed in denoising techniques. Notwithstanding the fact that in many contexts the aggregate render time and iteration count for a given target image quality has actually been reduced, the much longer per-iteration render times of the new process pose a new challenge in terms of making efficient use of one's day at work.

While separating large renders into passes has been an established technique for more than a decade, allowing both for better use of hardware, as well as a certain amount of flexibility in minimising invalidation of frames when small changes in a scene occur, due for example to revision to secondary animation, the concept of passes in the highly realistic world of path tracing poses new challenges, which the various houses have met with different strategies.

The course will review how these challenges have been met in five VFX and animation houses, describing the novel workflows that have emerged, the facility-wide approaches to the rendering of large scenes and ever-improving appearance models, as well as new approaches to virtual cinematic lighting. Examples from recent productions will be used extensively to illustrate these points.

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 will 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

14:00 — Opening statement and welcome message

14:10 — It all started with a camera – MPC's move to path tracing (30 min)

MPC began its move to path tracing in early 2014, initially for access to the programmable camera support in an early beta of *PRMan* 19. The gains in visual quality and complexity management led them to transition more of their productions to path tracing, including *The Jungle Book* which had only just started at the time, and it quickly became the standard approach to rendering. Optimisation in this new paradigm has meant discarding old tricks and discovering new ones (such as denoising). This ongoing effort to make path tracing practical at *MPC* will be the focus of the talk.

14:40 — Path tracing in production at Pixar (30min)

In this part of the course, Christophe Hery and Ryusuke Villemin will continue from where Per Christensen left it in the morning, and outline some of the various attempts on light transport since the advent of Physically Based Shading for Monsters University. On the topic of exotic integration techniques and specific artist-friendly controls in production, Thorsten-Walther Schmidt will highlight the experience with integrating Lightrig's third-party light transport visualization and artistic editing tools.

15:10 — Path Tracing at Imageworks: From *Monster House* to *Smurfs: The Lost Village* (30 min)

Sony Imageworks has been using an in house version of the *Arnold* path tracer in production since 2004 on a wide variety of animated and live action projects. Christopher Kulla will discuss the evolution of the renderer through the lens of the various films it was used on. From the early days when a single bounce of diffuse lighting was a luxury to today when global illumination is commonplace, Christopher will review the changes in hardware and software that have raised the bar in capabilities available to artists. He will discuss some of the major milestones of the *Sony Imageworks* branch of the renderer as well as the motivation for diverging from the commercial product, and conclude with a look ahead to current and future challenges in the rapidly evolving competitive landscape of production rendering.

15:40 — Break (15min)

15:55 — A change of path at Animal Logic (30min)

As the industry as a whole moved to path tracing for production rendering, *Animal Logic* has migrated from a mature REYES rendering pipeline to an entirely new proprietary ray tracer, *Glimpse*. Daniel Heckenberg will discuss techniques used to achieve production continuity and enable incremental development to arrive at an entirely new rendering system. A dramatic shift to interactive workflows, changes to how *Animal Logic* crafts scenes and represents complexity, as well as practical methods to manage noise in Monte Carlo rendering will be addressed.

16:25 — ILM's Path to Tracing (30min)

Industrial Light & Magic has used, and experimented with, a large variety of rendering pipelines, from a pure REYES-based solution, with all of its pre-passes, to a single pass path-tracing approach, having tried almost every commercially available production ray tracer in the process. In this talk André Mazzone will tell tales from this history and share some of the potentially less-than-obvious lessons learnt along the way.

16:55 — Q&A with all presenters (20min)

3 Organizers

3.1 Luca Fascione, Weta Digital

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

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.

4 Presenters

4.1 Rob Pieké, MPC

Rob Pieké is the Head of New Technology at *MPC* in the heart of London. Having recently celebrated his tenth year with the company, Rob has been involved in the development of custom artist tools for dozens of films, from *Harry Potter* to *Guardians of the Galaxy* and, most recently, *The Jungle Book*. Rob started programming in BASIC as a kid, and went on to get a degree in Computer Engineering from the *University of Waterloo* in Canada. With his passion for computer graphics — rendering and physical simulation in particular — the visual effects industry caught Rob's eye quickly, and he's never looked back since.

4.2 Christophe Hery, Pixar Animation Studios

Christophe Hery joined *Pixar* in June 2010, where he holds the position of Senior Scientist. He wrote new lighting models and rendering methods for *Monsters University* and *The Blue Umbrella*, and more recently for *Finding Dory*, *Piper*, *Cars3* and *Coco*, and continues to spearhead research in the rendering arena. An alumnus of *Industrial Light & Magic*, Christophe previously served as a research and development lead, supporting the facility's shaders and providing rendering guidance. He was first hired by *ILM* in 1993 as a Senior Technical Director. During his career at *ILM*, he received two Scientific and Technical Awards from the *Academy of Motion Pictures Arts and Sciences*.

4.3 Ryusuke Villemin, Pixar Animation Studios

Ryusuke Villemin began his career at *BUF Compagnie* in 2001, where he co-developed *BUF*'s in-house ray tracing renderer. He later moved to Japan at *Square-Enix* as a rendering lead to develop a full package of physically-based shaders and lights for *mental ray*. After working freelance for a couple of Japanese studios (*OLM Digital* and *Polygon Pictures*), he joined *Pixar* in 2011 as a TD. He currently works in the Research Rendering department, on light transport and physically-based rendering.

4.4 Thorsten-Walther Schmidt, Lightrig GmbH

Thorsten Schmidt is CEO and co-founder of the small German start-up Lightrig, which focuses on delivering novel light transport visualization and artistic illumination editing approaches to existing physically based production renderers, since 2014. Meanwhile, he's also busy wrapping up his PhD research on said topic, which formed the origin of Lightrig in 2012, at the Karlsruhe Institute of Technology.

4.5 Christopher Kulla, Sony Pictures Imageworks

Christopher Kulla is a principal software engineer at *Sony Pictures Imageworks* where he has worked on the in-house branch of the *Arnold* renderer since 2007. He focuses on ray tracing kernels, sampling techniques and volume rendering. In 2017 he was recognized with a Scientific and Engineering Award from the Academy of Motion Picture Arts and Sciences for his work on *Arnold*.

4.6 Daniel Heckenberg, Animal Logic

Daniel Heckenberg leads graphics R&D at *Animal Logic*. After wrangling birds, dragons, dinosaurs and billions of LEGO bricks to behave more or less according to the laws of physics and compelling narrative he has recently focused on development of *Animal Logic*'s proprietary path tracer, *Glimpse*.

4.7 André Mazzone, Industrial Light & Magic

André Mazzone has been with *Industrial Light & Magic* since 2002 and works in the rendering team. During this time, he has been intimately involved with the use and deployment of *RenderMan* in its many incarnations, *mental ray* and *Arnold*. Before *ILM* he spent four years at *Blue Sky Studios* working with *cgiStudio*, one of the seminal production-focused ray tracers. Some of his recent work appears in *The Revenant*, *Warcraft*, *Doctor Strange*, *Kong: Skull Island* and *Rogue One: A Star Wars Story*.

5 It all started with a camera - MPC's move to path tracing

Rob Pieké, *MPC*

5.1 A brief history

MPC has been using Pixar's RenderMan to produce virtually all of its final renders for film VFX for the last 20 years. In the early days this meant our rendering was entirely based around a multi-pass pipeline: creating shadow maps, etc.

5.1.1 Adding rays to REYES

We have typically been quite aggressive in our adoption of new RenderMan releases and new features, and started experimenting with ray tracing (initially for shadows, followed by reflections) when it first became available in the v11 release in 2003.

We continued to track RenderMan's growing capabilities and industry trends, increasing our usage of ray tracing for indirect lighting effects, and restructuring our in-house shading library to better express complex light transport. By 2013, a decade after we'd first started tracing rays, our custom shaders were heavily grounded in physics and ray tracing was prevalent in the majority of our renders.

5.1.2 Fast & Furious: Supercharged

In early 2014 we began work on *Fast & Furious: Supercharged*, an immersive theme-park experience where the audience was surrounded by a horseshoe-shaped screen. This presented us with the challenge of rendering omnidirectional stereo imagery, which was not practical using existing technology. Further complicated by the desire to artistically-control the camera's focal length, it became clear that we needed to develop a unique camera model which traced primary rays into the scene.

Fortuitously, Pixar was simultaneously working on a new path tracing framework for RenderMan called RIS, and we collaborated on the design of a plugin API for custom camera projections. We switched to the RIS framework for the project, using an early beta of v19, upgrading as new releases became available. For the first time in MPC's history, every camera ray, reflection, shadow, and other bounce of light went through a single-pass path-traced render.

5.1.3 The Jungle Book

Running largely in parallel, *The Jungle Book* provided MPC with the largest rendering challenge it had ever faced: more than 1,250 shots of incredibly detailed jungle environments and photorealistic hero animals. Inspired by the performance and quality of imagery we were achieving on *Fast & Furious: Supercharged*, we made an early decision to use RIS for this project as well. We leveraged the various shading APIs and sample code provided by Pixar to extend the out-of-the-box patterns, BxDFs and integrators, targeting the desired controls and technical expertise of our Lighting, Look Development, and Compositing artists.

Since *The Jungle Book*, every show at MPC has been rendered using the RIS path-tracer in RenderMan v20 or v21.

5.2 Our early reflections

Now three years into our path tracing endeavours, we can contrast the process and results against our previous ways of working. Some of these observations will be generically true of path tracing, which others may be more specific to our RenderMan context.

5.2.1 Representing reality

Ultimately we feel path tracing has made it easier for us to work, as we can speak in a more natural and physically-inspired way about the composition of our scenes, the lighting, and the properties of our materials. If, for example, we know a set is being lit by a softbox of a certain size, it is easy for us to model the same geometry and set up the same lighting properties (wattage, etc).

By being more expressive and using real-world terms as we prepare our scenes, we find that the results we get are more visually predictable as well, reducing the number of up-front experimental iterations required to hone in on the desired look.

5.2.2 Complexity management

Much like other studios, we've benefited greatly from the single-pass nature of path traced rendering. While we notice the incurred cost of re-rendering lighting effects that previously could have been baked-and-reused (shadow maps, global illumination pointclouds, etc), the simplicity in pipeline setup and file wrangling vastly outweighs this - particularly given our multi-site structure, where data is often shared between multiple globally-distributed teams.

We have also observed vast improvements in the memory-efficiency of our rendering. In contrast to *The Jungle Book* where we could typically render the entire environment in a single layer, our pre-RIS projects would often see a simpler environment requiring partitioning into 5-10 layers. Wrangling a full environment with many complex creatures still requires some amount of layering, but its now rare for us to nervously ask ourselves "how are we going to get this through the renderer?"

5.2.3 Progressive rendering

To produce a final image, we find ourselves commonly tracing between 16 and 512 camera rays for each pixel. Sacrificing some amount of performance and ray coherency, we iteratively send a single ray for each pixel, very quickly giving us a full-resolution image which progressively refines as we repeat the process.

We find that even a very small number of rays can generate tremendously-informative images. It is immediately obvious if objects are missing, if shader parameters are wildly-incorrect, etc. Artists can very quickly address issues revealed anywhere in the image, without needing to wait for the final pixel to be rendered in final quality.

This naturally extends to the renderer becoming an integrated part of our Lighting and Look Development process. With RenderMan's support for interactive scene/material/lighting edits, our artists are able to effectively work in the context of the final image, getting coarse but useful feedback in seconds that they can continually iterate upon.

A secondary benefit of progressive rendering is that it facilitates time limits. In the past, if we killed a render process at the 50% mark, we would have half the image at final quality and the other half completely empty. Now we can guarantee that every pixel will have some amount of data, even if it's not final quality at the point the process is terminated. This is hugely useful for the forecasting of renderfarm resources and scheduling of internal reviews, where we can take snapshots of render processes at a fixed time.

5.2.4 Optimisations and other trickery

The switch in rendering framework from REYES to RIS has meant a switch in render settings, and thankfully the new ones are proving to be both fewer in number and more intuitive to control. Historically we've always invested a fair amount of time tuning RenderMan's shading rate to control the quality of geometric tesselation (and corresponding memory footprint) and fidelity of shading calculations.

Now our main tweaking is in the distribution of samples or rays. Very fine geometric detail, possibly out of focus or moving, requires a high number of camera samples to generate a crisp image. In contrast, a frosted glass requires many secondary samples to capture the broad range of directions that both reflected and refracted light could contribute. As there is a multiplicative result to this management, we start by setting our secondary sampling as low as possible and only increase the number of camera samples until

we achieve the desired clarity in our geometry. Then we start increasing the sampling of the lights and BxDFs to combat noise.

5.3 Denoising

One of the seemingly-inherent side-effects of path-traced rendering is the presence of noise in the image. Given enough time and iterations, visually-acceptable convergence can be achieved, but the cost may be prohibitive. Further, the rate of convergence tends to decay as the render progresses, making it hard to predict how long it will take to achieve a fully noise-free image.

Inspired by Disney and Pixar, we began using a post-process denoiser as part of our pipeline on *The Jungle Book*. After a fairly exhaustive experimentation period where we explored the time-quality trade-offs of when to invoke the denoiser (i.e., what render settings should be used and how long should the render be allowed to run), we ultimately found that a full-quality 10-hour rendered image could be perceptually matched by a denoised 5-hour rendered image, effectively saving us almost 50% of the render cost.

One of the concerns with such a post-process is the risk of softening, where meaningful details get blurred-away with the rest of the noise in the image. The Pixar-provided denoiser that we use operates as a post-process on the rendered image, aided by a variety of supplementary data channels (geometric normals, depth, etc). Ultimately if the desired detail to maintain is not in one of these channels, it's unrealistic to expect the denoiser to maintain it. We found that very fine geometry, such as fur, proved especially challenging to denoise. To ensure there was sufficient detail to inform the denoising process, we used a combination of two "tricks":

First, we used the tangent vector of the fur to proved the denoiser with "normals" as it proved to have higher contrast than either the true normals of the fur, or the normals of the skin that the fur was spawned from.

Secondly, we rendered the images at double the resolution, lowering the render settings appropriately so as not to increase the overall render time. This is somewhat akin to letting the denoiser work on a subpixel level, where each fur curve has a larger footprint and a more cleanly-defined edge between it and the other curves overlapping the same final pixel.

5.4 Remaining challenges

While we benefit from more beautiful images with vastly increased visual complexity, we find there are scenarios where path tracing takes significantly longer to produce an image than the old REYES days. In particular, very fine geometry (such as fur or particulate matter) suffers from being simply statistically unlikely to be hit by a ray, requiring a very large ray count to counter the odds.

The faithful reproduction of reality is also sometimes at odds with story-telling. In the past, with shadow maps, we found it easier to art-direct lighting effects, where we might generate a specular highlight at a specific point on a surface, but then have the geometry cast a shadow in a slightly different direction for aesthetic reasons. There are interesting approaches to solve this, but we have not yet adopted one.

Unexpectedly, we are noticing the complexity of our file management is growing again. The move to path tracing initially allowed us to discard all of our shadow maps, global illumination caches, etc. but, as we leverage the new workflows described above, we are beginning to see multiple image outputs (in particular: snapshots of renders-in-progress, and pre-/post-denoised images).

Lastly, we note that managing settings for the renderer and the denoiser still needs to be done contextually for optimal results. We found that blindly reapplying the same setup used on *The Jungle Book* to *Passengers* and *Pirates of the Caribbean: Dead Men Tell No Tales* did not give us the visual quality we expected, and we needed to go through a new round of experimentation to find the ideal settings for denoising shiny spaceship interiors and moonlit ocean surfaces.

6 Emeryville: where all the fun light transports happen!

Christophe Hery, *Pixar Animation Studios*
Ryusuke Villemin, *Pixar Animation Studios*
Anton Kaplanyan, *Lightrig GmbH*

The trend is clear: animation and visual effects productions are embracing Physically Based illumination models, and more recently modern Monte Carlo (MC) techniques embedded in Path Tracers.

At Pixar, this evolution started with *Monsters University*, through a studio specific series of bsdfs, lights and dedicated integrators, taking full advantage of the ray-tracing extensions of the very well respected Reyes rendering engine, and of the structured Shading Language 2.0. For reference, this system was described in Hery and Villemin [2013]. During *Finding Dory* (and for the short film *Piper*), we ported these plugins to the new RIS architecture, and ended up providing them as part of the RenderMan 21 product. Their properties and design philosophies are laid out in the companion course Hery et al. [2017].

As examplified in this morning's talk by Christensen [2017], graphics practitioners can extend RIS with state of the art integration techniques. We developed our own set of unique features and practical tricks, leveraging our attempts during Finding Dory, as shown in Hery et al. [2016]. Additionally, we have incorporated brand new structures to handle complex volumes: see Wrenninge and Fong [2017].

The transition is complete and like many other production companies, Pixar is using a generic Path Tracer. Through careful collaboration between shading and lighting, we can achieve what would have been considered unreachable a few years ago, such as infinite bounce indirect lighting, multi-scatter in hair and volumes, or path-traced subsurface.

Does this mean we have reached our goal of rendering perfect images? Unfortunately (or fortunately): no! Artists keep pushing for even more complex light transports (like caustics), while maintaining complete artistic control over their image, which might sound contrary to the new Physically Based models we are using.

This particular part of the course will focus on the more exotic approaches we employ to solve the new problems exposed by these more complex light scenarios, as well as how we ended up integrating Lightrig's third-party visualization and editing tools to still retain the artists level of expressiveness they desire.

6.1 Photons

Photon tracing, Jensen and Christensen [1994], is still one the most efficient techniques to render caustics. It has been refined over the years. One of the biggest improvement has been Stochastic Progressive Photon Mapping (SPPM) described in Hachisuka and Jensen [2009], which solved the problem of memory space. By iteratively adding photons to the results, we can in practice have an infinite number of photons in a finite memory budget.

At Pixar, we also have the option to combine this with UPS from Hachisuka et al. [2012] / VCM from Georgiev et al. [2012], although in practice, we often render a separate pass for photons, and combine back to the main render. This lets the artist play and modify the photon map, like changing the surface of the water during the photon pass, blur or filter the photon map image.

6.2 Metropolis Photon Guiding

SPPM has trouble when rendering outdoor environments, where photons can scatter over the entire scene, most of them not ending in front of the camera and thus not contributing to the image. To remedy this issue, we implemented the Robust Adaptive Photon Tracing Technique of Hachisuka and Jensen [2011], in order to concentrate the photons in front of the camera.

We supplemented it with a manually placed ellipsoid which acts as a photon attractor. The target function is null outside of the frustrum as described in the paper, but also outside of the ellipse. In order to alleviate the usual impredictibility of Markov Chain Monte Carlo (MCMC) renders and also benefit from

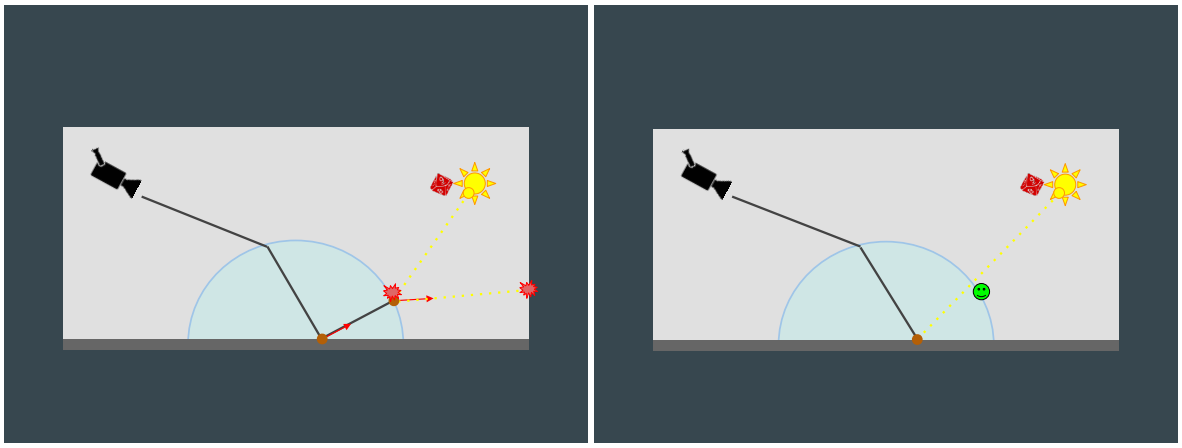


Figure 1: Left: sampling failure trough refraction. Right: thin shadow trick.

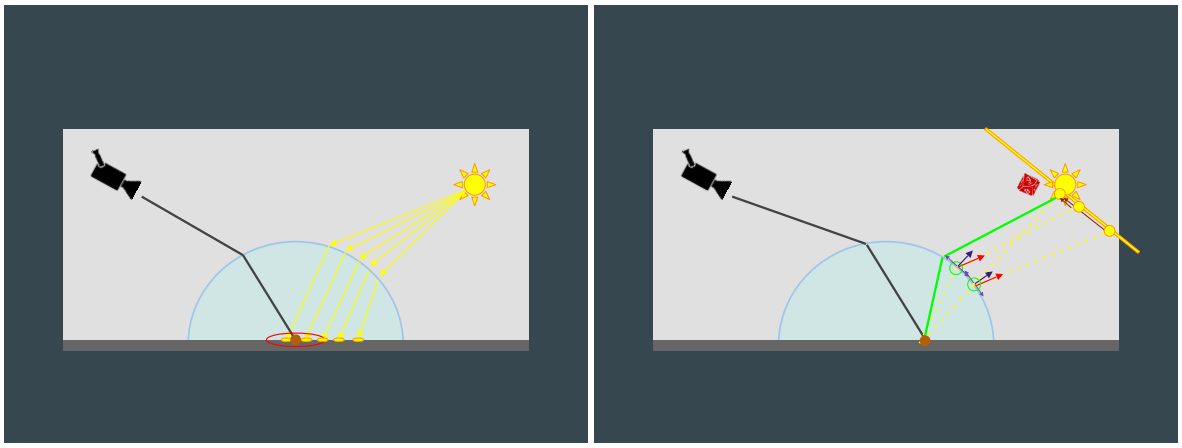


Figure 2: Left: photon mapping approach. Right: manifold walk solution.

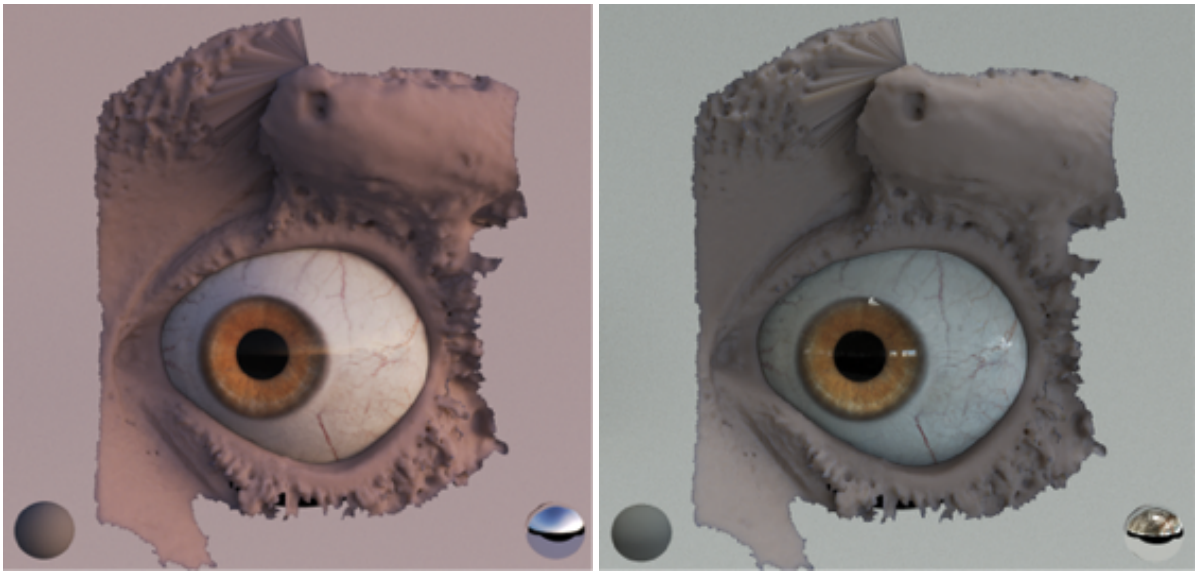


Figure 3: Manifold walk (for gleam/caustics in eye). Left: lighting scenario 1. Right: lighting scenario 2. Eye geometry provided and copyright © by Disney Research Zurich.

multithreading and vectorization, we trace about 100 to 10000 independent Markov chains in parallel. This reduces the probability of a new path popping late in the render, and keeps the noise closer to a MC render.

6.3 Manifold Walk

Photon mapping was until recently the only way to compute Specular-Diffuse-Specular paths (minus MCMC which has its own problems). But even with Hachisuka and Jensen [2011], it is still difficult and hard to setup when rendering a small part of the screen using photons as is the case for eyeballs.

Also in production we are still not completely bidirectionally symmetrical in every aspect (like diffusion Subsurface Scattering), so tracing light can give slightly different results. Finally some old tricks are not easily compatible with light tracing (among them shadow falloff or trace subsets), so whenever possible we prefer to stay unidirectional. This means that we tend to use the normal "trick", which is to not bend the shadow rays, as per Hery et al. [2016]: see Fig. 1.

With manifold walk from Hanika et al. [2015], we can compute and solve the right caustics paths by only doing forward ray-tracing. This has been implemented in our studio integrator, and used whenever a photon map render is not suitable, but we still want to see the light compression and caustics. Figs. 2 and 3 illustrate this.

6.4 Indirect Guiding

Manifold walk solves the problem of refractive caustics, but bidirectional path tracing is much more general and solves even more complex problems. Even within a scene with only diffuse surfaces, we can end up with fireflies and strong noise as soon as we have a concentrated source of high indirect illumination, like a practical light shining on the ceiling, or a strong bounce from the sunlight through a window. In order to improve those problematic cases without having to resort to bidirectional raytracing, we use machine learning to guide the indirect samples, a new technique from Disney Research Zurich (DRZ) Müller et al. [2017].

Since the move to pathtracing, our renders are progressive, meaning that we go over the entire image over and over, till the variance vanishes and we get a converged image. That also means we can use those recursive iterations to learn a little more about the scene every time, and use that knowledge to improve subsequent iterations.

The advantage of DRZ's method is that it can be used in combination with any integration method (unidirectional, bidirectional), and we found that in practice using with a standard unidirectional raytracer leads to very good resuls.

6.5 Per Light Controls

All of the above techniques, while useful, are not free. In production even a 10% increase is a big deal, so in pratice we enable those only where important. That means that the controls are on a per light basis. We can have a full shot rendering with Unidirectional Path Tracing, except for one light which will do Photon Mapping, or Manifold Walk or some other transport algorithm. This provides a way to tightly control the render cost and spend time only when and where it matters. For example, Figs. 4, 5 and 6 illustrate the photon tracing switch we have per light, allowing us to spend the entire photon budget on the lights that need the caustics.

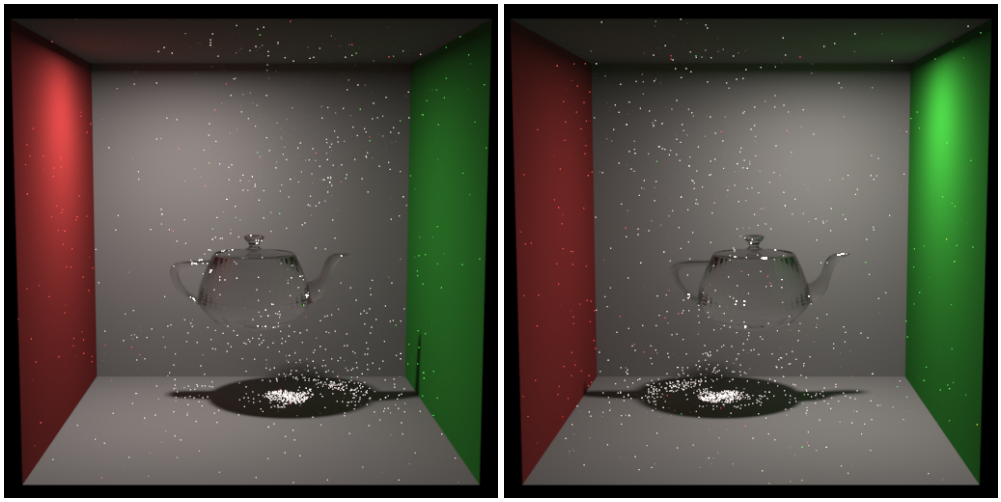


Figure 4: Unidirectional lights.

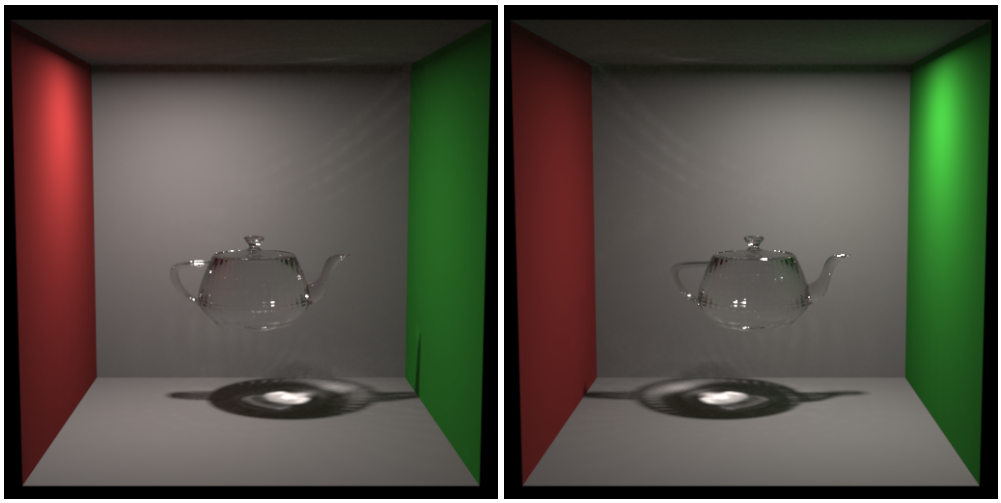


Figure 5: Bidirectional lights.

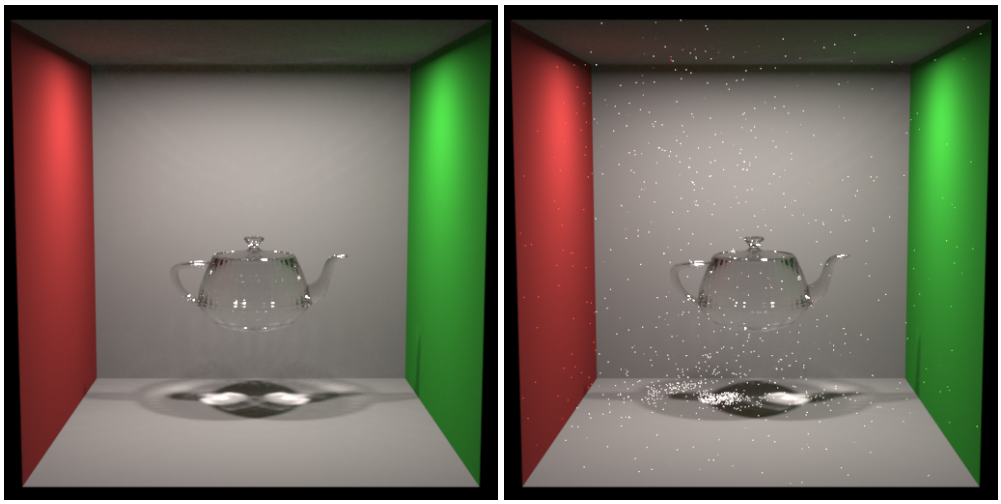


Figure 6: Left: both lights bidirectional. Right: left light bidirectional, right light unidirectional.

6.6 Analyzing and Controlling Light Transport

While a physically accurate global illumination is a correct and exact solution, it can be hard to comprehend, analyze, and control for a human due to the complex interplay of light and materials.

A final rendered image contains a lot of information: we can potentially infer the scene shape, and its materials and light sources. Many illumination phenomena are recognizable for a trained viewer, however, even lighting professionals often lack "the big picture". For example, how the illumination is being formed? An accessible visual analysis of the light transport solution can be a useful component of a workflow in many industries.

Industry-leading studios intensively adopt physically-based global illumination across their art generation pipelines. As a consequence, many existing artistic light editing tools become unsuitable as they address only local effects without considering the underlying physically based concepts and consequences. Artistic control over physically based global illumination can also be desired when creating artwork, such as visual effects and computer-animated films, with generated imagery.

In this section, we present interactive light transport visualization tools, as well as a light transport manipulation technique for artists that operates directly in the path space.

6.6.1 Interactive Visualization of Light Transport

Light transport visualization is a difficult task: even in a static scene, the data at hand is a five-dimensional light field caused by light propagating through space in every possible direction at the same time and reflecting at scene surfaces. On the other hand, conveying this information in a visually meaningful way can be of great help for the end users of the light transport simulation, e.g., digital artists, architects, engineers, and lighting designers.

Selection and Classification of Light Transport. A key observation is that, in order to understand the light transport, it is necessary to analyze it locally. To this end, all tools support what we refer to as *selective* visualization: they can visualize and manipulate only a particular, user-selected subset of light paths. This selection helps the user focusing on lighting phenomena of particular interest.

Extended Path Classification and Enriched Path Vertices. In order to provide an efficient selection and filtering of light transport, we extend the notation by Heckbert [1990] for light paths, by enriching them with additional information, such as interaction types and object IDs. This provides a detailed differentiation of individual path interactions. We distinguish between diffuse (D), glossy (G), and specular (S) interactions. We additionally classify interactions as reflections (superscript $\square^R$) or transmissions ($\square^T$). We also define a token X_P corresponding to *any* surface interaction (D, G, or S) that (optionally) belongs to a user-selected volumetric region P within the scene.

We assign all objects and light sources in the scene a unique ID and store the ID of the object at each interaction. The scene object IDs carry semantic information for manually modeled scenes, and thus typically support intuitive visualization.

We store *enriched path vertices* with the following additional attributes:

- all material interaction types along the path (e.g., diffuse, glossy, specular reflections, refractions, directly visible light, etc.).
- the IDs of the hit objects (or light sources) for each interaction
- additional attributes for each interaction along the path (including hit position, and path throughput)

Clustering of Light Paths. When dealing with visualization of a collection of light paths, it is important to avoid clutter. Therefore, it is useful to cluster the gathered paths and group them into bundles for more intuitive visualization.

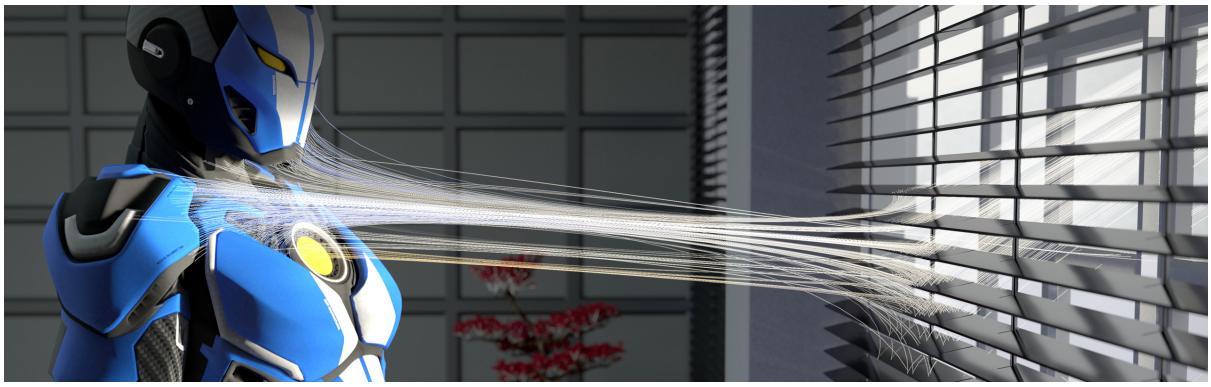


Figure 7: Light transport visualization provides key information about illumination structure within virtual scenes. Light path inspection shows logically grouped bundles of energy that deliver the illumination to the region of interest.

Clustering Based on Light Path Expressions. The first heuristic clusters by the last interactions of a light or an eye subpath. For example, if the last interactions are *LDDD*, *LDSS*, and *LDSD*, we obtain two clusters: (1) diffuse interactions *LDDD* and *LDSD*, and (2) specular interactions *LDSS*.

To cluster further interactions along the path, we apply the clustering algorithm recursively. In our example, we would partition cluster (1) according to the second last interactions, which yields two subclusters *LDDD* and *LDSD*.

Clustering Based on Object IDs. The second clustering heuristic is based on the object IDs obtained from the scene authoring process. The recursive clustering works the same way as in the previous method, except that the object ID is used instead of the interaction type.

Light Transport Visualization Tools. In order to efficiently examine the whys and wherefores of illumination phenomena, we introduce a set of *light visualization tools*.

One of them, the *light path inspection* tool visualizes the key light paths that contribute to the lighting of a user-specified region. It displays the gathered paths using several bundle-like arrows. In order to reduce clutter, we cluster the collected paths based on their type and create bundles for each clustered bundle (Fig. 7). We use two path clustering strategies described in Section 6.6.1.

6.6.2 Artistic Manipulation of Light Transport

The availability of efficient *physically based rendering* (PBR) systems with interactive preview has promoted its rapid adoption in the feature-film and gaming industries, as observed by Krivánek et al. [2010], McAuley et al. [2012]. Artists' productivity depends on the flexibility of the light editing tools. Many existing tools either target non-PBR systems or consider only specific PBR effects. Our motivation for light editing is to keep the manipulated illumination as physically plausible as possible, without restricting artistic freedom.

We present *path retargeting*, an artistic light editing tool built on top of physically based light transport, that enables intuitive manipulation of illumination, including effects that result from complex light paths (see Fig. 8, top). Path retargeting operates *directly* on light paths and allows the user to select an illumination feature using visualization and selection techniques from Section 6.6.1 and then manipulate the selected paths related to this feature (Fig. 8, bottom). We also discuss how to render manipulated paths using bidirectional light transport algorithms.

Manipulation with Path Retargeting. *Path retargeting* provides control over lighting features by manipulating on an optional subset of the path space as well as an optional user-defined region(s) of the scene.

Figure 8: Example edits in the Garage scene. Before/after close-ups (right): removing reflections caused by the car, moving sunlight refracted through the windows, transforming a glossy caustic, and altering the mirror reflection. The Buddha scene shows a path retargeting example. Left: original render. Right: using path retargeting to displace light paths forming the caustic. Adapted from Schmidt et al. [2013].

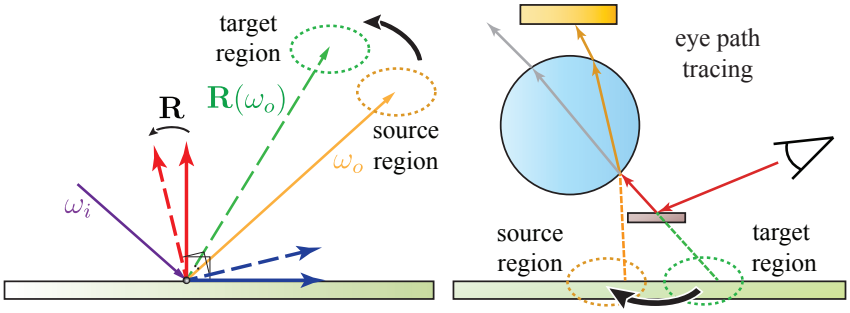


Figure 9: Left: path retargeting can be viewed as a transformation of either the incident or exitant shading tangent frame, depending on the manipulation direction and the direction of the path construction method. Right: when constructing a path in a direction opposite to the manipulation direction, at each path edge, we apply the inverse transformation to check for potential manipulations from the opposite direction. Adapted from Schmidt et al. [2013].

First, the transport phenomenon is selected, by filtering the subpaths that will be manipulated using the extended path notation (Section 6.6.1). For example, the user can select a light (e.g., a diffuse indirect illumination LD^RX_P) or an eye subpath (e.g., specular reflection X_PS^RE). If the path selection is empty, all paths are manipulated.

Then, the user can also specify a volumetric region of interest, a *source region*, serving as the origin of the retargeting transformation. The transformation is then an affine mapping defined by placing a *target region* (Fig. 8, bottom) in the scene. The tool redirects path segments, therefore, implicitly propagating the manipulation to the secondary effects, such as inter-reflections and indirect shadows. Both source and target regions are not linked to any scene object, however, their transformations can be keyframed for animation, and optionally linked to the transformation of an object. In addition, we introduce basic appearance modifiers for path throughput, such as intensity scaling and hue editing, in the spirit of Obert et al. [2008].

Robust Bidirectional Manipulation. Retargeting works by redirecting either a light or an importance flow in the light transport simulation based on the selected manipulation. Therefore, it is easy to apply the manipulation during rendering in case if the path construction direction coincides with the direction of the manipulation. However, a path can usually be constructed using multiple techniques, by connecting subpaths of different lengths, therefore, it is not always the case that the direction of path construction would coincide with the manipulation direction. In this case, we have to ensure that all bidirectional path construction techniques can consistently apply the manipulations.

Therefore, we formulate path retargeting as a modifier that acts on the scene surfaces. This allows to seamlessly use path retargeting in various light transport methods such as path tracing, bidirectional path tracing, and Metropolis Light Transport after Veach and Guibas [1997]. Path retargeting effectively transforms *a surface's shading tangent frame* such that an outgoing segment points towards the user-specified target (Fig. 9, left). This introduces a non-symmetric BRDF (see Veach [1998]) for bidirectional transport, similarly to modified or bump-mapped shading normals. Given a surface point with normal $\mathbf{n}$, BSDF f, and a path with incident and outgoing directions $\mathbf{i}$ and $\mathbf{o}$, path retargeting defines a transformation operator $\mathbf{R}$ of the tangent frame that depends on the source and the target regions, and modifies the BSDF as

$$f'(\mathbf{i} \to \mathbf{o}) = f(\mathbf{i} \to \mathbf{R}(\mathbf{o})),$$

and its adjoint is derived similarly to [Veach, 1998, Section 5.3.2] as

$$f'^*(\mathbf{i} \to \mathbf{o}) = f(\mathbf{R}^T(\mathbf{o}) \to \mathbf{i}) \frac{|\mathbf{R}^T(\mathbf{o}) \cdot \mathbf{n}|}{|\mathbf{o} \cdot \mathbf{n}|}.$$

Figure 10: Manipulating diffuse GI in the Visit scene. Left to right: unmodified light transport, indirect diffuse illumination from the floor is manipulated with path retargeting.

The adjoint BSDF f^* is used when the path construction direction does not coincide with the manipulation direction. If the user specifies a source region, the manipulation must be handled using rejection sampling for paths constructed from the direction opposite the manipulation direction. For example, if an eye subpath is constructed for a path that was retargeted from the light direction, we apply the inverse source-target transformation to ensure that the unmodified light subpath would hit the source region (Fig. 9, right); otherwise, the retargeting modification is rejected.

Example Edits. Fig. 8 (top) demonstrates multiple sequential edits, here modifications are applied to $LS^TG^RX_P$ and LS^TX_P subsets of the path space (glossy reflection off the car and sunlight through the window), followed by a rotation and scaling of a glossy reflection on the floor, as well as the modification of the reflection in the mirror (X_PS^RE paths).

Fig. 10 shows light editing using path retargeting in an architectural scene. We applied two local transformations to indirect diffuse illumination (LD^RX_P paths), first "stretching" the color bleeding on the wall (top green helper gizmo) and scaling the indirect lighting down the stairs (bottom green helper gizmo).

6.7 Conclusion

We presented an interactive set of tools for intuitive visualization, selection, and manipulation of light transport. Light visualization tools use interactive visualization techniques. They support users in understanding the light transport in virtual scenes. Individual tools are useful and appropriate for different tasks. Operating directly on path-space solutions of the rendering equation enables the manipulation of complex transport phenomena, including secondary shading effects.

We also demonstrated some of the more exotic light integration solutions we have in the studio, and how we enable them only where needed, thereby allowing a good compromise between rendering speed and image quality.

References

Per Christensen. 2017. Advanced path tracing in Pixar's RenderMan. In *Path tracing in Production Part 1, ACM SIGGRAPH 2017 Courses.*

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 141, 8 pages. https://doi.org/10.1145/1661412.1618487

Toshiya Hachisuka and Henrik Wann Jensen. 2011. Robust Adaptive Photon Tracing Using Photon Path Visibility. *ACM Trans. Graph.* 30, 5, Article 114 (Oct. 2011), 11 pages. https://doi.org/10.1145/2019627.2019633

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. Graph.* 31, 6, Article 191 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366210

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Comput. Graph. Forum* 34, 4 (July 2015), 87–97. https://doi.org/10.1111/cgf.12681

Paul S. Heckbert. 1990. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (Proc. SIGGRAPH)* 24, 4 (1990), 145–154.

Christophe Hery and Ryusuke Villemin. 2013. Physically Based Lighting at Pixar. In *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2013 Courses*. http://graphics.pixar.com/library/PhysicallyBasedLighting/index.html

Christophe Hery, Ryusuke Villemin, and Florian Hecht. 2016. Towards Bidirectional Path Tracing at Pixar. In *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2016 Courses*. http://graphics.pixar.com/library/BiDir/index.html

Christophe Hery, Ryusuke Villemin, and Junyi Ling. 2017. Pixar's Foundation for Materials: PxrSurface and PxrMarschnerHair. In *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2017 Courses*.

Henrik Wann Jensen and Niels Jørgen Christensen. 1994. Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects. (1994).

Jaroslav Krivánek, Marcos Fajardo, Per H. Christensen, Eric Tabellion, Michael Bunnell, David Larsson, and Anton S. Kaplanyan. 2010. Global Illumination Across Industries. In *ACM SIGGRAPH Courses*.

Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. 2012. Practical physically-based shading in film and game production. In *ACM SIGGRAPH Courses*.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proceedings of EGSR) - to appear* (June 2017).

Juraj Obert, Jaroslav Krivánek, Fabio Pellacini, Daniel Sýkora, and Sumanta N. Pattanaik. 2008. iCheat: A Representation for Artistic Control of Indirect Cinematic Lighting. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 27, 4 (2008), 1217–1223.

Thorsten-Walther Schmidt, Jan Novak, Johannes Meng, Anton S. Kaplanyan, Tim Reiner, Derek Nowrouzezahrai, and Carsten Dachsbacher. 2013. Path-Space Manipulation of Physically-Based Light Transport. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 32, 4 (2013), 129:1–129:11.

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Ph.D. Dissertation. Stanford University. AAI9837162.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. *Proc. SIGGRAPH* (1997), 65–76.

Magnus Wrenninge and Julian Fong. 2017. Motion Blur and Aggregate Volumes. In *Production Volume Rendering, ACM SIGGRAPH 2017 Courses*.

7 ILM's Path to Tracing

André Mazzone, *Industrial Light & Magic*

7.1 Introduction

The use of path-tracing is widespread in film and television production and for very good reasons. It's a flexible technique that can render a wide range of phenomena in a consistent paradigm. Compared with rasterising pipelines and even distribution ray-tracing, path-tracing has clear advantages, especially in the simplification of artist work-flow. What follows is an attempt to chronicle the history of its adoption at Industrial Light & Magic.

7.2 History of Techniques

RenderMan and the REYES algorithm served as ILM's primary rendering pipeline since its first forays into computer graphics with Jurassic Park in 1993. With its emphasis on efficiency and memory compactness it proved to be a very successful tool for the production of CG elements for live-action integration. REYES was particularly adept at rendering fast, high quality, though approximate, motion-blur, a feature that would ensure continued use many years past the introduction of practical ray-tracing. All shadowing was done with depth-maps rendered from light positions, which made for complicated dependencies and extensive re-rendering when scenes changed.

In 2001, with Pearl Harbor, ILM introduced the use of ambient occlusion. Utility passes rendered in screen space were generated in Mental Ray and queried at render time to provide more accurate integration with captured environments. Naturally this came at the cost of more pre-passes. Ambient occlusion and the related technique of reflection occlusion was ILM's first use of ray-tracing in production.

In 2002, RenderMan added ray-traced shadows, which eliminated the light depth map pre-passes necessary to render shadows, simplifying the render pipeline for most geometry.

With point clouds (in 2004) came the ability to render a large number of new effects, including, sub-surface scattering, fast approximations for diffuse inter-reflection and ambient occlusion, again at the cost of increasing the number of pre-passes and slower iteration times. The pattern of adding artistic expression at a large cost continued.

Distribution ray-tracing simplified scene configuration drastically, as sophisticated light transport could now be done in a single pass. Start-up times were reduced as well, though rendering costs increased substantially. Ray-traced motion-blur, in particular, often would not fit inside render budgets, with post motion-blur used extensively instead.

It was only with progressive path-tracing that iteration times started to fall dramatically. Fast, though noisy, renders during lighting and look development for the first time allowed artists to be more efficient creatively. Total render times were still longer than with a REYES pipeline, but artists were able to make creative decisions at a much higher frequency since frames would not need to render to convergence unless a final image was actually required.

7.3 What path-tracing Is

What is it about path-tracing that makes it so useful? There are various properties that make it desirable in a visual effects production pipeline.

7.3.1 Ray-tracing

Path-tracing has all of the advantages of ray-tracing:

- There is a separation of material definition from rendering algorithm,
- It's conceptually similar to physical processes (at micro-scale phenomena and above),
- It supports a large set of interesting visual phenomena.

- It supports a large set of geometric and volumetric primitives within the same framework.
- It's extremely amenable to parallel processing.

7.3.2 Ray-tracing with a Difference

However, path-tracing adds advantages over distribution tracing. Because of the reduced cost of single ray paths, fast progressive rendering is possible even with high ray depths. Low quality, but representative, results are delivered quickly.

Additionally, a path-tracing framework makes it easy to implement useful features including:

- Renders can be terminated at any time allowing time-constrained or iteration-limited computation.
- Renders that were previously terminated can be restarted if further convergence is desired.
- Adaptive sampling can be used to render regions with high variance preferentially.
- Light Path Expressions (LPEs) can allow a greater level of light transport introspection, yielding increased flexibility with arbitrary outputs (AOVs) or even light path culling.

7.3.3 Conceptually Simple

Since rendering computation is done in a single pass, scene configuration is greatly simplified which reduces pipeline fragility and increases predictability. Fast preview with less accuracy allows approximate estimation of how a converged image might appear when resolved.

When extra quality is required, there are a few global settings that, at the cost of render times, will reduce variance globally when desired. Accuracy is a parameter of the shading process.

7.3.4 Extensible

Path-tracing provides an easy framework in which to install new lighting integrators without changing the underlying material sampling and evaluation. While unidirectional path-tracing is commonly used, more exotic algorithms including Metropolis Light Transport, Manifold Next-Event Estimation, Specular Manifold Exploration, and Path Guiding can all be implemented using the same underlying infrastructure.

7.4 What Path-tracing is Not

7.4.1 Not New

Path tracing has been around for a very long time (The Rendering Equation, Kajiya 1986), though it's only relatively recently that it's found popular use.

7.4.2 Not Always Path-Tracing

Properly speaking, unlike distribution tracing path tracing is designed to spawn only one ray at each intersection. However, geometric complexity is often resolved with fewer rays than shading complexity and the concept of splitting by a specified factor can allow renders to converge quicker.

Additionally, as more sophisticated integrators are applied, auxiliary data structures can be employed, for example in the case of Vertex Connection and Merging (VCM) where photon mapping can be used to connect light paths in new ways.

7.4.3 Not Always Fast

There are many reasons why path-tracing, especially when performed progressively, might not be the most efficient rendering framework in terms of total render time. Distributing rays across an entire image for every iteration does not allow the efficiencies, available in modern CPUs, that come from coherent memory access. Similarly, spawning a single path at each intersection, as opposed to a distribution, also

discards coherence. Similarly, achieving high quality (converged) results often requires lengthy computation. In particular, Monte-Carlo integrators require four times as many samples to halve the variance. Production rendering budgets rarely allow fully converged renders to be computed. Other noise reduction strategies can be employed including post-render filtering (denoising).

7.4.4 Not Always Memory efficient

Unlike some scan-line algorithms, scene geometry is resident for the entire render which can lead to high memory footprints. Conversely there are cases where path-tracing is more efficient, for example in the case where visibility point lists are applied to hair primitives in a REYES renderer.

7.4.5 Not Cheat-Free

In the effort to reduce variance, certain approximations can be entertained that add complexity to rendering. One such technique, so-called thin shadows, allows simplified shadowing paths for diffuse material response. Especially in unidirectional tracing, it's expensive to compute caustic paths, so approximate shadows, using opacity can yield acceptable results at the cost of accuracy. Combining this with accurate sampling of direct lighting during specular transmission events requires that certain light paths be culled in order to eliminate double illumination. This greatly reduces noise but at a cost of increased rendering complexity and also divergent rendering strategies. More generally, light path extinction using LPEs can be used to cull specific classes of paths that would otherwise require large sample counts to converge.

7.4.6 Not Simple to Implement

Writing a fully-featured path-tracer is a significant technological investment, especially when more sophisticated effects are modelled for example, subsurface scattering and volumetric integration. That being said, a path-tracer is significantly simpler to implement than a REYES renderer, for example.

7.4.7 Not Always Simple for the Artist

Even with advances in integration algorithms and processor speed, given ILM's rendering resources, it's still not practical to enable all of the available rendering features and expect production scenes to have reasonable completion times. There is an inescapable balancing act between geometric and shading complexity, available resources and artistic direction. Artists at ILM still spend significant effort tuning renders to produce timely, high quality results.

Additionally, removing objectionable artefacts is a time-consuming process, often requiring a knowledge of the underlying rendering algorithms. Noise can be caused by a variety of different sources including light sampling, surface sampling, indirect transport, subsurface scattering, motion-blur to name but a few.

7.4.8 Not Enough

Despite the versatility of path-tracing, sophisticated and intricate image compositing is still required for successful integration of computer graphics elements into live-action footage. The list of AOVs that compositors use to coax and massage final renders is ever-growing.

7.4.9 Not Mature

The field of rendering still contains many unsolved problems, especially in the cases of complicated light transport. Adaptive sampling is still a challenging process and largely remains an unsolved problem. Regardless of the arguments around optimal settings, it's clear that metrics currently used to measure variance do not correspond to perceptual quality. Significant improvement is required before adaptive sampling can yield predictable results. There are new ways of rendering various phenomena emerging

constantly. Finding unified frameworks that can render a diverse set of phenomena is an area of active research.

7.5 Conclusion

The desire to use ray-traced techniques for full-scale production existed decades before it became practical to entertain them. Path-tracing, and the new techniques that it enables, combined with the large number of high quality commercial rendering solutions has led to a Renaissance in the computer graphics community. Large-scale productions are now able to use these once-impractical techniques to produce images with levels of photo-realism not previously achievable. If the activity in academia can be considered any indication, there are no signs that innovation in this field is on the decline.

8 Arnold at Imageworks: Path Tracing from Monster House to Smurfs: The Lost Village

Christopher Kulla, *Sony Pictures Imageworks*

When establishing the look of picture for the film Monster House back in 2004, the creative team at Sony Pictures Imageworks made a bet on a new workflow for rendering motion picture imagery. A few years after that film was completed the studio decided to adopt this software for all of its rendering needs. This document gives a brief overview of the technical evolution of the software in the years since this adoption and explores the benefits that come from tailoring a renderer to one's own production pipeline. We also give a historical account of the divergence point between our renderer and the commercial *Arnold* product.

Figure 11: Monster House, the first full length animated feature rendered with *Arnold* and brute force path tracing. ©Columbia Pictures Industries, Inc. and GH One LLC. All Rights Reserved. Courtesy of Columbia Pictures

8.1 Monster House

The production of Monster House marked the first time, to our knowledge, that indirect lighting was rendered by brute force path tracing for a full length animated film. The increased realism gave the film a tangible, stop-motion aesthetic (see Figure 11). A number of compromises were made to achieve this: motion blur was disabled, hair was modeled with geometry, and only a single bounce of diffuse illumination was simulated. At the time *Arnold* was based around uniform grid acceleration structures and its shading system exposed low-level details such as light loops, tracing and sampling calls directly to shader writers. To reduce sampling variance to acceptable levels, lighting artists had to be conscious of

any small elements that would be excessively bright to indirect rays which could include manually controlling ray visibility flags, assigning simplified shaders and so on. Despite this amount of hand holding, once an environment or sequence had been configured by senior lighters, many shots would fall into place very quickly. This was far superior to other rendering solutions of the time which required manual pass management in every shot.

8.2 Bounding Volume Hierarchies

The adoption of the renderer at the facility would be cemented during the next two major projects to use it: Cloudy with a Chance of Meatballs and Alice in Wonderland. Both of these projects featured both motion blur and hair/fur. These required a shift away from uniform grids and towards the now industry standard bounding volume hierarchy (BVH). The predictable memory usage of BVHs combined with the ease of extending them to support motion blur made the choice clear compared to other popular alternatives like kd-trees. Unlike other BVH implementations, we did not implement spatial splits as primitive sizes are usually very small, and knowing that each primitive can be hit at most once reduces the book-keeping required for transparent shadows.

Code simplicity has been a guiding principle throughout the project: we are careful to avoid code duplication whenever possible. A single BVH implementation is used for all primitive types, with judicious use of C++ templates whenever per-primitive specialization is required. Likewise we have a single BVH builder for both static and motion blurred cases as the differences between the two are rather minimal. This choice means that improvements like parallelizing the BVH construction immediately benefited all cases at once.

8.3 Tracing Hair and Fur

Hair and fur ray tracing were added via a dedicated curve primitive (Nakamaru and Ohno [2002]). A key design decision was to adopt cubic control points without tessellation to minimize memory usage. In particular, curves are directly stored with a B-spline basis so that only $3+n$ control points are required for n curve segments (compared to $3n+1$ for a Bezier basis for example). To minimize the expense of non-axis aligned hairs, we build a shallow BVH and store a tight oriented bounding shape in the leaves that allows many curve segments to be intersection tested at once. This approach has proved to have excellent performance and is simpler to implement than mixing oriented bounding volumes in the hierarchy itself (Woop et al. [2014]). Performance was further improved by matching the BVH width to the natural SIMD width of the hardware (4 for Intel SSE instructions) and taking advantage of these instructions when intersecting primitives as well.

8.4 Open Shading Language

Around 2009, the shading API that had slowly evolved since Monster House was beginning to complicate improvements to sampling techniques inside the renderer. Careful coordination between renderer and shaders was required to implement techniques like multiple importance sampling, or even just to properly track shading AOVs. As we were contemplating the need to move to even more advanced integration schemes and new ray bundling techniques, we knew we had to decouple the task of the shader writer from the technical burden of keeping up with renderer internals. This lead us to create the *Open Shading Language* (OSL) project (Gritz et al. [2010]). Some early discussions with other studios suggested we were not the only ones thinking along these lines so the project was made open source.

From the beginning, the shading language was conceived to expose BSDFs through *closures* rather than exposing any algorithm specific constructs such as light loops or trace calls. Shading derivatives (required for pre-filtering of shading signals and bump mapping) were implemented using dual arithmetic (Piponi [2004]) instead of finite differencing to better support single-point shading architectures common to ray tracers. Our very first runtime for OSL assumed we would be able to amortize the cost of a shading interpreter over bundles of rays. As we connected this brand new interpreter to a brand new packet ray

tracing code, we ended up rewriting a large portion of the code base. Around the same time, Marcos Fajardo was establishing SolidAngle S.L. to bring to market the renderer as we had been using it up until then. The rapid churn of unproven code proved to be counter productive to the API stabilization required for integration of the renderer into third party packages, keeping other operating systems (like *Microsoft Windows*) well supported, etc…The teams at Imageworks and Solid Angle therefore mutually agreed to fork development of the renderer, while still keeping in place all joint intellectual property agreements including mutual access to source code. This decision proved fruitful as we were able to complete the push to a production ready renderer built around OSL. All projects entering production at that time embraced this "new" OSL-based renderer, however quickly discovered that performance had taken a major step back. It wasn't until the move to an LLVM based backend that OSL would really become production ready. In moving to LLVM, we also reverted to a single-ray architecture. While packet tracing produced impressive speedups for coherent rays, the speedups on incoherent rays were offset by much greater code complexity and surprising performance pitfalls when bundles became too small. We instead re-dedicated ourselves to the simplicity of our initial rendering architecture.

8.5 Light Path Expressions

An interesting fallout of our push to OSL was a rethinking of how shading AOVs should be handled by a modern path tracer. Previous shading languages (and our own C API) placed the burden on shader writers to maintain a coherent array of output colors that split the image into meaningful components such as direct and indirect diffuse, specular, subsurface scattering, etc…As the importance of indirect lighting grew, tracking this efficiently throughout all shaders became a major source of complexity. We did not wish to burden the OSL language with said complexity. Instead, drawing inspiration from Paul Heckbert's early work on classifying light paths (Heckbert [1990]) we designed a regular expression engine that could compactly encode the desired set of AOVs. This engine was initially outside of the OSL project and part of the renderer, but we published a description of its capabilities alongside the OSL documentation to suggest a possible way that AOVs could be supported orthogonally to the language. Because the engine constructs a state machine from the user supplied expressions, only a single 32 bit word of state per ray is required, making the runtime cost relatively low. On the other hand, there were some unexpected consequences that made us ultimately reconsider the use of light path expressions.

The biggest downside was that they constrained some sampling decisions like the ability to merge diffuse and specular lobes at deeper bounces. We also discovered that very few artists (and in some cases, even ourselves) could understand how to properly write a coherent set of LPEs that would be guaranteed to add back up to the beauty image. For this reason, we wound up moving back to a hand-written set of rules, maintained by the renderer.

The core idea of LPEs on the other hand, proved to be popular, and by community request we ended up including the matching engine as part of the OSL project around the same time we removed support for it from our renderer. To our knowledge, several of the implementations of LPEs in commercial products have directly made use of this code.

In retrospect, this small feature ends up being a great case study in the difference between in-house vs. commercial renderers. A commercial renderer cannot envision all possible use cases a customer might face, and must provide very powerful mechanisms to override built-in behavior. On the other hand, in-house renderers cater to a much narrower set of users facing known problems. When looking at the problems that LPEs solve, it turns out that very few really required the full generality they offer, and we therefore benefited from keeping our code simpler.

8.6 Advancing the Integrator

The abstraction layer of OSL cemented the distinction between materials and integrator for our renderer. This marked the beginning of a rapid move towards improving the out-of-the-box experience for artists. Both because shader writers were able to focus on better pattern creation tools, and by moving what had been per-shader variance reduction tricks into core renderer features that could be applied in

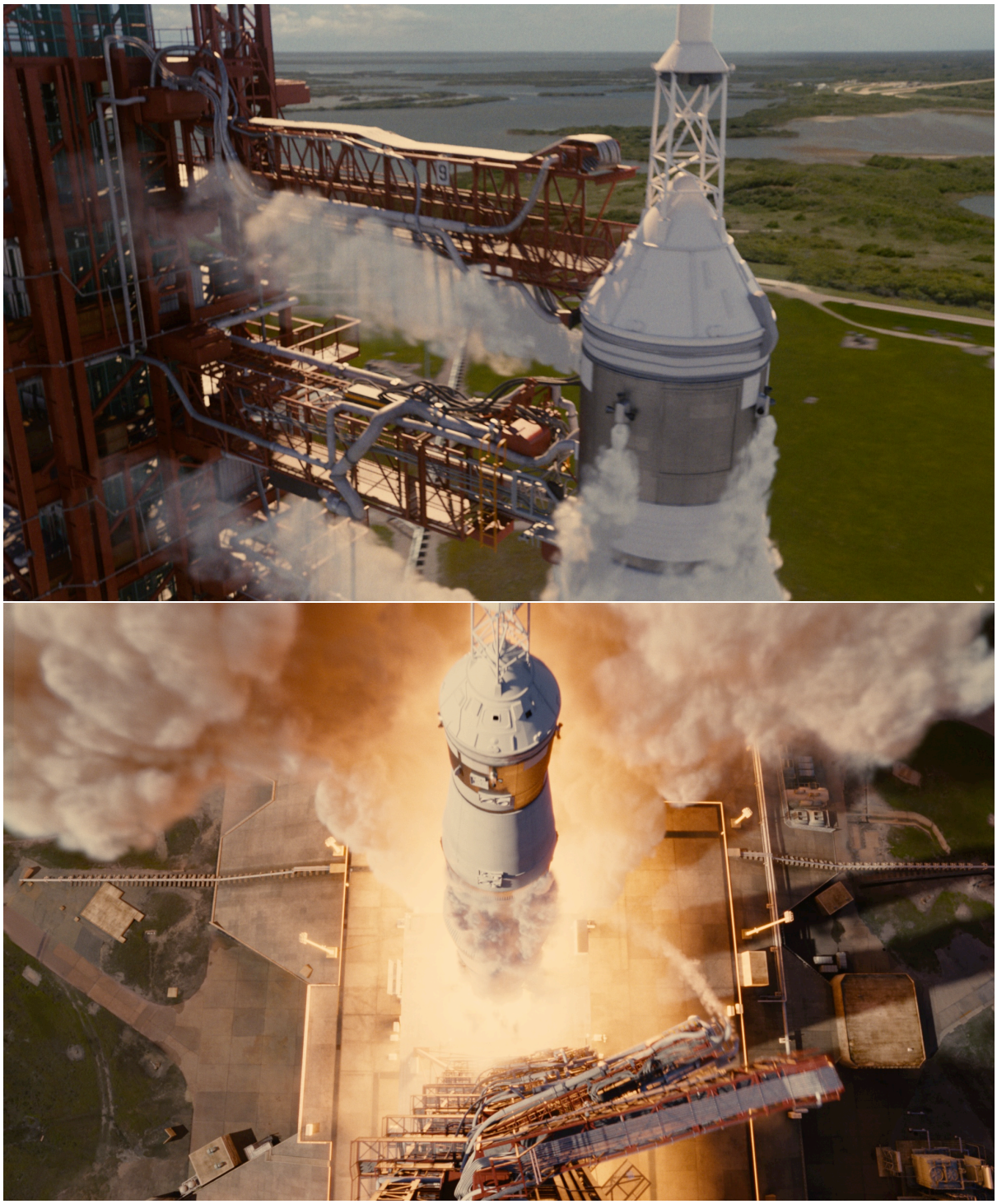


Figure 12: Apollo 11 launch sequence from the film *Men In Black 3*, one of the first uses of fully integrated volumetric effects in the renderer.

a predictable and automatic way. These included simple clamps on ray intensity, roughness clamping to regularize difficult glossy inter-reflections and so on. We were able to introduce volume rendering as a first class citizen to the renderer (Kulla and Fajardo [2012]), which proved invaluable for scenes with smoke interacted with geometry and lighting in complicated ways (see Figure 12). We transitioned mid-production from a subsurface scattering implementation based around point clouds to a fully ray traced solution (King et al. [2013]) with no changes in the shaders or artist scenes. We finally were able to ex-

periment with bi-directional methods and Metropolis samplers (Veach [1998]) which proved very useful for generating ground truth images as well as occasionally rendering caustic passes for real shots.

8.7 Subdivision Surfaces

In parallel to the integrator improvements brought by OSL, we continued to push forward in supporting greater and greater geometric complexity. A key aspect of this has been improving support for (displaced) subdivision surfaces. While these had always been supported by our renderer, the first implementations were rather naive implementations of the Catmull-Clark subdivision rules, resulting in uniform tessellation to a fixed depth. This made displacement particularly expensive, and required artists to manually tune tessellation rates in shots. Moreover, ever increasing model complexity meant that frequently, the *base mesh* itself was sufficiently dense for all but the most extreme closeups. One complication to tessellating the scene adaptively is that our pipeline heavily relies on *instancing*.

Inspired by the work leading up to the OpenSubdiv library, we designed an adaptive tessellation engine that decomposes meshes into patches. Because we control the definition of subdivision in the entire pipeline, and know that our models contain very few extraordinary vertices, we were able to use approximate constructions that do not exactly conform to the true Catmull-Clark limit surface. This removes much of the complexity needed to support these cases. With a patch-oriented architecture, adaptive tessellation and multi-threading become significantly easier. Moreover, the natural patch based structure also offers numerous opportunities for efficient storage. We were careful to design the storage such that no vertices ever need to be duplicated across edges. At low tessellation rates (say 2×2), duplicating patch edges leads to unacceptable memory bloat.

Assigning per-patch edge rates can be done with simple screen space heuristics that target either distance to the limit surface or edge length depending on the presence of displacement. This even works in the case of instancing: each patch can be evaluated for all instances and tessellated to fit the worst case. As long as early exits are provided for models entirely on-screen or off-screen, the bottleneck of testing every patch of every instance can be avoided. Since this adaptive tessellation system has been put in place, we have seen most of our scenes render using 10Gb or less for an average of 100M to 200M unique triangles per frame without artists having to think about any manual management of dicing rates.

8.8 Multiple Scattering

Most recently, we have focused on scaling the performance of the renderer to increasing numbers of bounces. While it was initially considered sufficient to only compute a single bounce of indirect diffuse transport, we are now expected to compute many bounces of all frequency illumination. This is particularly important to the appearance of volumes and hair where multiple scattering effects dominate. To ameliorate the performance in these cases, we have decoupled the calculation of direct lighting from the seed path. This means we are free to trace shadow rays for only a subset of path vertices, using local shading properties at each vertex to guide the selection. Techniques like Russian roulette help in equalizing the path weight over many bounces and keeps paths as short as they need to be. Randomly shadowing only a subset of vertices improves performance on longer paths. We note that this kind of technique is only practical when tracing a single ray at a time. For bundles of rays the storage requirements for long paths could become quite large. The exact dependency between path length and number of vertices to be shadowed is also highly sensitive to implementation details, an aspect which we carefully re-evaluate over time.

8.9 Simplifying Sampling

The trend on sampling controls has also been towards greater and greater simplicity. In the beginning, we frequently tried to minimize the use of camera rays and relied heavily on path splitting to reach final quality. As path lengths continue to rise, camera rays become an overall smaller fraction of the total number of rays and their relative cost is reduced. Therefore we are getting closer to eliminating the need

Figure 13: Enchanted forest environment from Smurfs: The Lost Village. Compared to Figure 11, scene and shading complexity have both increased by orders of magnitude. ©2017 Sony Pictures Animation. All Rights Reserved.

for path splitting. In addition, we have made our sampling patterns progressive and added a layer of adaptive sampling to every pixel. This allows computation to be focused only to the more noisy regions automatically, reducing the need for artists to manually fine tune sampling knobs. Adaptive sampling has the other benefit of making the noise perceptually uniform, allowing image space denoising techniques (Sen et al. [2015]) to effectively remove the last bit of noise from difficult areas.

8.10 Conclusion

We have come a long way since the first renders we did for Monster House. The industry's shift to path tracing (Christensen and Jarosz [2016]) has confirmed that the decision we made over 13 years ago was a good one. Looking ahead we will continue to see ever greater fidelity to the underlying physics of light transport, while we continue to be challenged by the rapidly evolving hardware landscape of CPUs and GPUs. Current trends point us to revisit the packet based tracing ideas we had explored when first developing OSL. The tension between wanting to keep the code simple and faithful to the beauty of the underlying transport equations, while properly exploiting the underlying hardware is likely to keep us busy for the next decade at least.

References

Per H. Christensen and Wojciech Jarosz. 2016. The Path to Path-Traced Movies. *Foundations and Trends® in Computer Graphics and Vision* 10, 2 (Oct. 2016), 103–175. https://doi.org/10.1561/0600000073

Larry Gritz, Clifford Stein, Chris Kulla, and Alejandro Conty. 2010. Open Shading Language. In *ACM SIGGRAPH 2010 Talks (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 33, 1 pages.

Paul S. Heckbert. 1990. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '90)*. ACM, New York, NY, USA, 145–154. https://doi.org/10.1145/97879.97895

Alan King, Christopher Kulla, Alejandro Conty, and Marcos Fajardo. 2013. BSSRDF Importance Sampling. In *ACM SIGGRAPH 2013 Talks (SIGGRAPH '13)*. ACM, New York, NY, USA, Article 48, 1 pages. https://doi.org/10.1145/2504459.2504520

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. *Comput. Graph. Forum* 31, 4 (June 2012), 1519–1528. https://doi.org/10.1111/j.1467-8659.2012.03148.x

Koji Nakamaru and Yoshio Ohno. 2002. Ray Tracing for Curves Primitive. In *WSCG*.

Dan Piponi. 2004. Automatic Differentiation, C++ Templates, and Photogrammetry. *Journal of Graphics Tools* 9, 4 (2004), 41–55. https://doi.org/10.1080/10867651.2004.10504901 arXiv:http://dx.doi.org/10.1080/10867651.2004.10504901

Pradeep Sen, Matthias Zwicker, Fabrice Rousselle, Sung-Eui Yoon, and Nima Khademi Kalantari. 2015. Denoising Your Monte Carlo Renders: Recent Advances in Image-space Adaptive Sampling and Reconstruction. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 11, 255 pages. https://doi.org/10.1145/2776880.2792740

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

Sven Woop, Carsten Benthin, Ingo Wald, Gregory S. Johnson, and Eric Tabellion. 2014. Exploiting Local Orientation Similarity for Efficient Ray Traversal of Hair and Fur. In *High-Performance Graphics 2014, Lyon, France, 2014. Proceedings*. 41–49.