

Path tracing in Production

Luca FASCIONE (Organizer) Johannes HANIKA (Organizer)
Weta Digital *Weta Digital*

Rob PIEKÉ
MPC

Ryusuke VILLEMIN
Pixar Animation Studios

Christophe HERY
Pixar Animation Studios

Manuel GAMITO
Framestore

Luke EMROSE
Animal Logic

André MAZZONE
Industrial Light & Magic

SIGGRAPH 2018 Course



Fig. 1: *War for the Planet of the Apes*, image courtesy of Weta Digital, ©2017 Twentieth Century Fox Film Corporation. All rights reserved. *Peter Rabbit*, image courtesy of Animal Logic, ©2018 Sony Pictures Animation. All rights reserved. *Blade Runner 2049*, image courtesy of Framestore, ©2017 Warner Bros Inc. All rights reserved. The Millennium Falcon in *SOLO: A STAR WARS STORY* image courtesy of Industrial Light & Magic. ©2018 Lucasfilm Ltd. All Rights Reserved.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGGRAPH '18 Courses, August 12-16, 2018, Vancouver, BC, Canada

©2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5809-5/18/08.

<https://doi.org/10.1145/3214834.3214864>

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.

Contents

1 Objectives	4
2 Syllabus	4
3 Organizers	6
3.1 Luca Fascione, Weta Digital	6
3.2 Johannes Hanika, Weta Digital	6
4 Presenters	6
4.1 André Mazzzone, Industrial Light & Magic	6
4.2 Christophe Hery, Pixar Animation Studios	6
4.3 Ryusuke Villemin, Pixar Animation Studios	7
4.4 Manuel Gamito, Framestore	7
4.5 Luke Emrose, Animal Logic	7
4.6 Rob Pieké, MPC	7
5 Introduction to path tracing and Monte Carlo sampling	8
5.1 Transport equations	8
5.1.1 Assumptions	8
5.1.2 Counting photons	9
5.1.3 Events which change photon count	10
5.1.4 The radiative transfer equation	12
5.2 The path space	14
5.3 The Monte Carlo method	16
5.4 Colour formation in a renderer	16
5.4.1 Colour reproduction	17
5.4.2 Where to get input spectra from	18
5.4.3 Colour noise	19
5.4.4 Importance sampling	19
5.4.5 Radiometry vs. Photometry	20
5.5 Path construction techniques	21
5.5.1 Path tracing	21
5.5.2 Next event estimation	21
5.5.3 Light tracing	22
5.5.4 Photon mapping-based approaches	22
5.5.5 Markov chain-based methods	23
5.6 Conclusion	23
6 Practical Path-Tracing at ILM	27
6.1 Path-Tracing Changes Everything	27
6.1.1 Reduced Artistic Iteration Time	28
6.1.2 Reduced Total Render Time	28
6.1.3 Increased Geometric Complexity	28
6.1.4 Increased Lighting Complexity	29
6.1.5 Simplified Scene Setup	29
6.2 Shortcomings of Path-Tracing	29
6.2.1 Noise	29
6.2.2 Geometric Complexity	29
6.2.3 Surface Complexity	29
6.2.4 Lighting Complexity	29

6.2.5	Camera Complexity	30
6.2.6	Motion Complexity	30
6.2.7	Light Picking	30
6.2.8	Combinatorial Complexity	30
6.3	Remedies	31
6.3.1	Splitting	31
6.3.2	Adaptive Sampling	31
6.3.3	Level of Detail	31
6.3.4	Lighting Simplification	31
6.3.5	Surface Simplification	31
6.3.6	Blurring Motion-blur	32
6.3.7	Post-Render Denoising	32
6.3.8	Denoising in Compositing	32
6.4	Conclusion	32
7	More fun with light transports at Pixar!	33
7.1	Fun with shader simplification	33
7.2	Atomic blonde	36
7.3	Additive Lights in a Pathtracer	39
7.4	Practical Delta-tracking for Path tracing	42
7.5	Invisibility Cloak	44
7.6	An LPE by any other name?	45
7.7	Artisanal Denoising	47
7.8	Acknowledgments	47
8	Path Tracing the Framestorian Way	52
8.1	A Bit of History	52
8.2	Path Tracing those Volumes	52
8.2.1	Transmittance Estimation	53
8.2.2	Free Distance Sampling	54
8.2.3	Emission Integration	56
8.2.4	Direct Volume Illumination	57
8.2.5	Many Lights Sampling	59
8.3	Things to do when we get back from Siggraph	59
9	From Bricks to Bunnies - Adapting a Raytracer to New Requirements	62
9.1	The Road Travelled so Far	62
9.2	Hair Geometry and Level of Detail	62
9.3	Developing the Peter Rabbit Fur Shader	63
9.3.1	Improving Lobe Evaluation	64
9.3.2	Vectorization of Computation	66
9.4	Weave and Clothing	68
9.5	Grass Replacement	69
9.6	Prelit Workflow	70
9.7	Single Pass Render Motion-Blur Integration	71
9.7.1	FBOMBS - Film Back Offset Motion Blur Scale	72
10	Turn it down! - MPC's guide to noise management	74
10.1	A brief history	74
10.1.1	Fast & Furious: Supercharged	74
10.1.2	The Jungle Book	74
10.2	Sampling and noise	74

10.3 Denoising	75
10.3.1 Staying sharp	75
10.3.2 Rise of the machines	76
10.4 Further thoughts and remaining challenges	76

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 different 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.

2 Syllabus

14:00 — Opening statement and introduction to path tracing (15 min)

We will open with a survey of 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 work is connected.

14:15 — Practical Path tracing at ILM (30 min)

Path tracing is now an accepted industry standard for rendering computer generated imagery for visual effects, and its advantages over previous methods are clear. It is a conceptually simple paradigm that can simulate a broad range of useful phenomena. It's also extensible in that new light transport algorithms can be easily incorporated into existing frameworks without modifying underlying surface appearance models. However, it is not free from limitations. Path tracing is notoriously slow to render final quality images. It suffers from numerous sources of sampling noise, each requiring its own strategies for variance reduction. Undesirable image artefacts, with causes that are not always obvious, are quite common as well. Substantial artist involvement is often required to ensure that convergence rates fit within production

deadlines. In this talk André Mazzone will describe some of the ways in which *Industrial Light & Magic*'s workflows have been affected by both the flexibility and constraints that path-tracing brings.

14:45 — More fun with light transports at Pixar! (30 min)

Continuing from where they left off last year, Ryusuke Villemin and Christophe Hery will cover some of their recent tricks for efficient sampling and artistic illumination controls. Among them will be such hot topics as “invisibility cloak”, “atomic blond” and “learning indirectly”.

15:15 — Tracing Paths the Framestore Way (30 min)

Framestore first embraced path tracing technology with the release of *Gravity* in 2013 and has continued to improve on this experience over the years. Not only path tracing but also bidirectional path tracing are now routinely used to render all the shots in its film productions. This successful approach has led to the ongoing development of *Freak*, Framestore's new path tracing renderer. Manuel Gamito will show how volume scattering can be seamlessly integrated with both surface scattering and sub-surface scattering in a unified path tracing framework. The case of sub-surface scattering, in particular, is shown to be a full Monte Carlo solution to a volume scattering problem that is bounded by a surface, making it virtually indistinguishable from the general treatment for volumes. This path tracing framework is renderer agnostic and works with both Arnold and Freak. Finally, an overview of importance sampling techniques for different light models is presented, explaining how next event estimation is treated consistently for all types of scattering events in their path tracer.

15:45 — Break (15min)

16:00 — From Bricks to Bunnies - Adapting a raytracer to new requirements (30 min)

Animal Logic refined the technology of its proprietary renderer *Glimpse* across *The LEGO Movie*, *The LEGO Batman Movie* and *The LEGO NINJAGO Movie* with a specific focus on heavy hierarchical geometric instancing in order to facilitate the rendering of incredibly high polygon-count scenes (tens to hundreds of billions of polygons). Luke Emrose will discuss how, once production on *Peter Rabbit* began, a very different set of challenges were presented: high geometric complexity with no possibility for instancing; high density curve rendering (which was used for fur, clothing, whiskers, grass and edge-fuzz) and fur shading. Additionally the film was no longer fully CG and *Glimpse* was now required to seamlessly integrate with live-action plates containing lens distortion, motion blur; and to simulate bi-directional light transport between live-action and CG elements. Using a range of novel techniques developed to address these issues and refine them within a single show, he will discuss how *Animal Logic* managed to fulfill these requirements with a relatively small team and within a compressed time-frame.

16:30 — Turn down the noise! (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.

17:00 — Q&A with all presenters (15min)

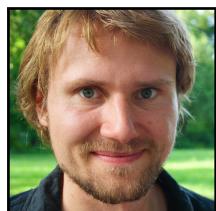
3 Organizers

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 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 physically-based ray tracers. Some of his recent work appears in *Kong: Skull Island*, *Solo: A Star Wars Story*, *Black Panther* and *Ready Player One*.

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*, *Coco* and *Incredibles 2* 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 Manuel Gamito, Framestore



Manuel Gamito is a senior research engineer at *Framestore*, which he joined in 2012, and has worked on all of the company's major films since then. He has developed the in-house volume shading software, that is used with both the *Arnold* and *Freak* renderers, and also the light shader technology. He helps *Framestore* to stay up to date with the latest research results, within the context of a coherent and unified shading framework, while also performing research in new algorithms when circumstances permit. Since 2008, he holds a PhD in Computer Science from the University of Sheffield.

4.5 Luke Emrose, Animal Logic



Luke Emrose is a senior rendering developer at *Animal Logic*. After a previous life of over 9 years as a shader writer, artist, lighter, modeler, on-set previs artist and TD, he has spent the past 5 years dedicated to developing the core rendering technologies of the *Animal Logic* proprietary renderer, *Glimpse*, with a recent focus on hair and curve rendering and shading. Having contributed technology to *The LEGO Movie*, *The LEGO Batman Movie*, *The LEGO NINJAGO Movie* and *Peter Rabbit*, and co-designing the *Glimpse* shading system, he continues to thrive on the difficult and never-ending challenge of photo-realistic rendering.

4.6 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.

5 Introduction to path tracing and Monte Carlo sampling

JOHANNES HANIKA, *Weta Digital*
LUCA FASCIONE, *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 is 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 and the algorithms to solve the equations in exhausting detail. For a broader introduction we refer to Pharr et al. [2017]. A thorough introduction to the radiometric quantities is given by Chandrasekar [1960], and a good entry point for especially the volumetric scattering aspects are the dissertations by Jarosz [2008], Novák [2014] and the recent state of the art report by Novák et al. [2018].

5.1 Transport equations

We need to derive a measurement function which can be used to evaluate throughputs for transport paths as they are constructed during path tracing. Intuitively, this will require us to measure some physical quantity along a ray, i.e. an infinitesimally small direction emergent from an infinitesimally small position. This is very abstract, so we will proceed by simplifying the problem setting somewhat, introducing the notion of photons, and then resort to simply counting photons. From there, we will derive what quantity is transported along one ray (radiance) and the equations that govern the transport. This approach is very much inspired by Arvo [1993].

Readers who are not interested in the physical foundation of the transport equations and the exact assumptions leading there can safely skip to section 5.2 and directly proceed to collecting light along an infinitesimally small ray.

5.1.1 Assumptions

Before we dive into equations, let's make some simplifying assumptions to make our task easier. Most assumptions to be made about light transport have been prominently generalised away in computer graphics literature, so our list here comes with references to works that extended rendering beyond the respective limitation:

- Light consists of particles (*photons*), not waves. In particular we are not interested in interference (as opposed to Werner et al. [2017]) or polarisation (unlike Jarabo and Gutierrez [2016]).
- Photons travel along straight lines, because the index of refraction does not vary continuously but only at interfaces (contrary to Ament et al. [2014]). Also photons are not affected by gravity (as they would on astronomical scales such as Thorne [2014]).
- Light is transported instantly, or, equivalently, we're only interested in some equilibrium state with constant boundary conditions (even though Jarabo et al. [2014] showed that simulating time dependence results in nice visualisations).
- We only model elastic scattering, i.e. there is no energy transfer between wavelengths, no photoluminescence (as described for instance by Glassner [1995], Hullin et al. [2010]).
- Volumes: to derive the transport equations, we will assume a uniform random distribution of light-interacting particles (unlike Bitterli et al. [2018], d'Eon [2018], Jarabo et al. [2018]) which are much larger than the wavelength (as opposed to Rayleigh scattering as introduced by Strutt [1871]), and have isotropic cross-sections (because we avoid anisotropic media described by Jakob et al. [2010]).

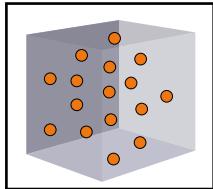
5.1.2 Counting photons

To quantify light, let's start with a straight forward concept: we'll be counting photons. A single photon corresponds to an atomic portion of *energy* E (measured in joule [J]):

$$E = \frac{h \cdot c_m}{\lambda} \quad [J]. \quad (1)$$

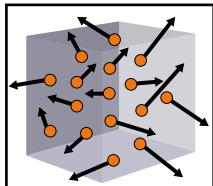
Here, $h \approx 6.62607004 \times 10^{-34}$ [$m^2 kg/s = Js$] is Planck's constant and $c_m = c/\eta_m$ is the speed of light in a material with index of refraction η_m . This is slightly slower than the speed of light in vacuum, which is conveniently fixed by definition at exactly $c = 299,792,458$ [m/s]. In eq. (1), λ [m] is the wavelength of the photon. We'll often measure it in nanometers [nm] instead to help distinguish it from world space lengths.

To determine the overall *radiant energy* in a certain volume, all we need to do is count photons and add up their energies. Assuming they all have the same wavelength, they all have the same energy and we can just multiply the energy by their count $\#P$:



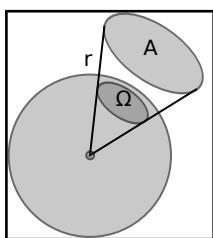
$$Q = \#P \cdot \frac{h \cdot c_m}{\lambda} \quad \left[J = \frac{kg \cdot m^2}{s^2} \right]. \quad (2)$$

Now unfortunately photons can be moving very quickly and thus a more practically useful quantity is how many photons pass through a certain volume *per time*. This is called *radiant power*, or *flux*, and is measured in watts:



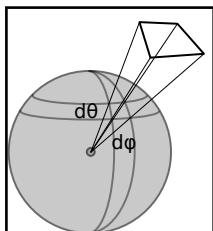
$$\Phi = \frac{dQ}{dt} \quad \left[\frac{J}{s} = W \right]. \quad (3)$$

To derive a quantity that considers the direction of the photon, we need a measure of directions. This two dimensional measure space is called *solid angle* and defined as the area of a piece of surface projected onto the unit sphere, measured in *steradian* [sr]:



$$\Omega_A = A/r^2 \quad [sr]. \quad (4)$$

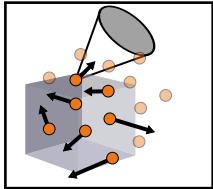
To perform actual integration in this domain, we often times need to re-parametrise the domain into polar coordinates, for instance using the latitude/longitude way. In general, spherical polar coordinates would require also a radius. Since we are only interested in directions ω , the 2D angular domain $\theta \times \varphi$ suffices in this case, but we still need to account for the Jacobian determinant:



$$d\omega = |\sin \theta| d\theta d\varphi, \quad (5)$$

$$\int_{\Omega} d\omega = \int_0^{2\pi} \int_0^{\pi} \sin \theta d\theta d\varphi = 4\pi. \quad (6)$$

Since photons live in *phase space*, i.e. have a 3D position and a 2D direction, we will associate $(\mathbf{x}, \omega) \in V \times \Omega$ with them. With these tools at hand, finally we're ready to count photons per time per volume and per solid angle. Ideally we would count some place holder quantity Φ with some imaginary measurement device that only counts photons inside a certain 3D volume if their direction ω falls within the solid angle subtended by a certain imaginary funnel:



$$\Phi = \int_{\Omega} \int_V ?(\mathbf{x}, \omega) \, d\mathbf{x} \, d\omega \, [W]. \quad (7)$$

We will be assuming a locally uniform distribution of photons such that we can assume a piecewise smooth function $?(\mathbf{x}, \omega)$ and expect to get the same answer by integrating it over a volume $V \times \Omega$ in phase space as we would by counting individual photons.¹ The approach we will take in the next paragraphs to both define such a function and to introduce the transport equations governing its propagation is the following: First we will define the effects that change the distribution of photons in phase space. Collecting all these results in a differential equation, the *radiative transfer equation* (RTE). From there we will derive the more well-known integral equation.

5.1.3 Events which change photon count

The most obvious event is photon *emission*, where particles emit light (for instance modelled by a black body emitter). Other than that, there are two things that obviously change photon count in a certain volume in phase space. The first one is the most natural one: *streaming*. This happens if the photon enters or leaves the volume by just continuing on its way. The other one is *collision*, if the photon does interact with matter inside the volume under consideration. In such an event, a photon can be absorbed or scattered. While absorption is always a loss, scattering may mean both: the new direction of the photon may be inside the currently considered solid angle of the phase space, or have left it. We'll go through them in an order that makes reasoning about the quantities easy.

Absorption Collision with an absorbing particle is easy to model, since all that happens is that the photon disappears. All we need to model is how often such an event happens. We will be using a statistical model which can be used to determine the chance that a photon interacts with such a particle while travelling a unit distance. As mentioned earlier, we assume a locally uniform random distribution of absorbing particles, such that we can compute a locally constant density ρ in $[1/m^3]$. We will also only deal with isotropic cross-sections σ of the particles, i.e. $\sigma [m^2]$ does not depend on direction ω . From these two, we define the *collision coefficient* $\mu = \sigma \cdot \rho \left[\frac{1}{m} \right]$. This is the probability of collision while travelling unit distance in the medium, or, conversely, $1/\mu$ is the *mean free path*. In neutron transport literature, this is sometimes referred to as Σ .

To model heterogeneous media, we will work with $\mu(\mathbf{x})$ to express macroscopic inhomogeneities while still assuming locally uniform random distribution of particles at microscopic scale. This is important for the derivation of the differential equations later on.

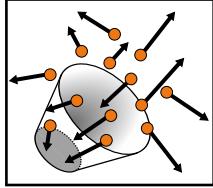
We will need a few such collision coefficients for the different types of events. For absorption, we are dealing with $\mu_a(\mathbf{x})$. If a photon intersects such a particle, it is lost and we can compute the change in flux:

$$- \int_{\Omega} \int_V \mu_a(\mathbf{x}) L(\mathbf{x}, \omega) \, d\mathbf{x} \, d\omega \, [W]. \quad (8)$$

This equation shows that the unit of $\mu_a(\mathbf{x})L(\mathbf{x}, \omega)$ has to be $\left[\frac{1}{m} \cdot \frac{W}{m^2 sr} = \frac{W}{m^3 sr} \right]$ to integrate to watts $[W]$. We take this opportunity to define *radiance* as the core unit which is transported along a light transport path.

¹This seems to be one of the longer lasting assumptions, at least we are not aware of computer graphics literature generalising the equations to count individual photons.

It can be measured by an imaginary differential measurement device which consists of a surface counting photons passing through it and a funnel attached to it. As both the surface and the funnel simultaneously tend to zero size (in area and solid angle, respectively), the measured quantity approaches radiance for the resulting ray (\mathbf{x}, ω) :

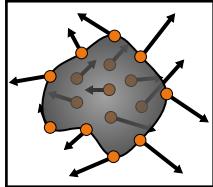


$$L(\mathbf{x}, \omega) = \left[\frac{W}{m^2 sr} \right]. \quad (9)$$

Emission Emission is very similar to absorption, in that a black body emitter model dictates that every photon intersecting an emitting particle is absorbed rather than reflected. This means that emission comes with a loss term very similar to the one by absorption. In fact, some formulations assume that μ_a includes the density of emitting particles, too. For increased clarity, sometimes the emission coefficient is explicitly modelled as $\mu_e(\mathbf{x})$. We introduce another symbol for emitted radiance $L_e(\mathbf{x}, \omega)$ to be able to quantify the amount of photons which are added per time in a certain volume $V \times \Omega$:

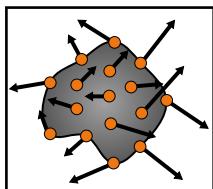
$$\int_{\Omega} \int_V \mu_e(\mathbf{x}) L_e(\mathbf{x}, \omega) d\mathbf{x} d\omega [W]. \quad (10)$$

Streaming To quantify how many photons enter and leave the current piece of phase space $V \times \Omega$, we want to count the number of photons passing through the spatial boundaries of V , which is written as ∂V . This is measured perpendicularly to the surface normal $n(\mathbf{x})$. Since the quantity $L(\mathbf{x}, \omega)$ depends on ω but itself is just a scalar and no vector field, we will measure $\omega \cdot L(\mathbf{x}, \omega)$:



$$- \int_{\Omega} \int_{\partial V} \langle \omega \cdot L(\mathbf{x}, \omega), n(\mathbf{x}) \rangle d\mathbf{x} d\omega. \quad (11)$$

The integration over the boundary ∂V will get in the way later on, so we replace it by an integration over the whole volume V by using Gauss' theorem:



$$= - \int_{\Omega} \int_V \frac{d}{d\omega} L(\mathbf{x}, \omega) d\mathbf{x} d\omega. \quad (12)$$

Scattering Once a photon collides with a particle, it can also be scattered, i.e. experience a change of direction ω . As mentioned before, we want to limit ourselves to *elastic scattering*. This means the energy of the photon remains unchanged during the event. In particular this means it does not change wavelength, so we can limit our analysis to one wavelength at a time, and solve the resulting equations per wavelength. (we'll briefly come back to this assumption of excluding fluorescence in section 5.4).

We model scattering by a kernel $k(\omega_i, \mathbf{x}, \omega)$ which determines the directional change at position \mathbf{x} . Depending on the new direction, this may result in a gain or a loss: the photon may be scattered into the solid angle of our piece of phase space, or out of it. We can write the in-scattering gain as

$$+ \int_{\Omega} \int_V \int_{\Omega} k(\omega_i, \mathbf{x}, \omega) L(\mathbf{x}, \omega_i) d\omega_i d\mathbf{x} d\omega, \quad (13)$$

and similarly the out-scattering loss as

$$-\int_{\Omega} \int_V \int_{\Omega} k(\omega, \mathbf{x}, \omega_i) L(\mathbf{x}, \omega) d\omega_i d\mathbf{x} d\omega. \quad (14)$$

This last term can be simplified considerably by assuming the scattering kernel is separable in space and direction, i.e.

$$k(\omega_i, \mathbf{x}, \omega) = \mu_s(\mathbf{x}) \cdot f_s(\omega \cdot \omega_i), \quad (15)$$

because then we can pull the scattering coefficient μ_s out of the inner integral over solid angle ω_i :

$$-\int_{\Omega} k(\omega_i, \mathbf{x}, \omega) L(\mathbf{x}, \omega) d\omega_i = -\mu_s(\mathbf{x}) \int_{\Omega} f_s(\omega_i \cdot \omega) L(\mathbf{x}, \omega) d\omega_i \quad (16)$$

$$= -\mu_s(\mathbf{x}) L(\mathbf{x}, \omega) \int_{\Omega} f_s(\omega_i \cdot \omega) d\omega_i \quad (17)$$

$$= -\mu_s(\mathbf{x}) L(\mathbf{x}, \omega), \quad (18)$$

where in the last step, we used the property of the *phase function* $f_s()$ that it integrates to one over the sphere Ω . Because it is normalised, this phase function $f_s()$ is a probability density function determining where a particle is reflected to when intersecting a scattering particle, as determined by the collision coefficient μ_s . Often we assume isotropic media, i.e. the phase function only depends on the cosine between incoming and outgoing directions, i.e. $f_s(\omega \cdot \omega_i)$. The most simple incarnation is isotropic scattering, where the outgoing direction does not even depend on the incoming direction at all, i.e. a constant distribution $f_s \equiv \frac{1}{4\pi}$. The more often used variant is the popular phase function by Henyey and Greenstein [1941]:

$$f_s(\cos \theta) = \frac{1}{4\pi} \frac{1-g^2}{(1+g^2-2g \cos \theta)^{3/2}}, \quad (19)$$

which contains a single parameter g , the mean cosine which determines how forward ($g = 1$) or backward scattering ($g = -1$) the phase function is.

To further simplify the equations, we combine the losses due to absorption, shadowing by emissive particles, and out-scattering, and define the *extinction coefficient*:

$$\mu_t(\mathbf{x}) = \mu_a(\mathbf{x}) + \mu_e(\mathbf{x}) + \mu_s(\mathbf{x}). \quad (20)$$

5.1.4 The radiative transfer equation

Now that we discussed all events, we can proceed and combine them in a single equation. As dictated by energy conservation, all terms need to sum up to zero (because we considered all ways photons may be added or lost). Doing so gives us

$$0 = \int_{\Omega} \int_V \underbrace{-\frac{d}{d\omega} L(\mathbf{x}, \omega)}_{\text{streaming}} + \underbrace{\mu_e(\mathbf{x}) L_e(\mathbf{x}, \omega)}_{\text{emission}} - \underbrace{\mu_t(\mathbf{x}) L(\mathbf{x}, \omega)}_{\text{extinction}} \\ + \underbrace{\mu_s(\mathbf{x}) \int_{\Omega} f_s(\omega_i \cdot \omega) L(\mathbf{x}, \omega_i) d\omega_i}_{\text{in-scattering}} d\mathbf{x} d\omega. \quad (21)$$

Since this equation holds for integration over any arbitrary part of phase space, they have to hold for every individual point, too. This means we can leave away the integration over phase space $\Omega \times V$, and arrive at a simpler differential equation, 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):

$$\begin{aligned}\frac{d}{d\omega}L(\mathbf{x}, \omega) &= \mu_e(\mathbf{x})L_e(\mathbf{x}) - \mu_t(\mathbf{x})L(\mathbf{x}, \omega) \\ &\quad + \mu_s(\mathbf{x}) \int_{\Omega} f_s(\omega \cdot \omega_i)L(\mathbf{x}, \omega_i) d\omega_i.\end{aligned}\quad (22)$$

The directional derivative $\frac{d}{d\omega}$ is sometimes written as $\omega \cdot \nabla$. This formulation presents an alternate intuitive way of looking at the terms: $\frac{d}{d\omega}L(\mathbf{x}, \omega)$, the left hand side, is the change of radiance in direction ω . Now to find out how radiance changes along the ray direction ω , we need to add emission at this point, subtract extinction, and add the in-scattered radiance over all incoming directions ω_i .

The mathematical structure of eq. (22) is an integro-differential equation, since it contains differentials and integrals. This makes it hard to solve, especially with path tracing: in rendering we know how to solve difficult high-dimensional integrals with the Monte Carlo method, thus we'll have to work on the differential.

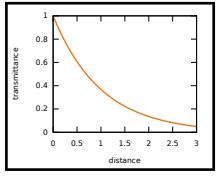
In an effort to integrate both sides of eq. (22), let's look at a simple one dimensional example of a differential equation:

$$\frac{d}{dx}L(x) = -\mu_t L(x). \quad (23)$$

where of course potential similarity in naming of the variables are entirely coincidental. To solve it, we'll need boundary conditions, in our example $x > 0$, $L(0) = 1$. Given this, we know a closed-form solution to this problem, using and integrating factor:

$$L(x) = \exp(-\mu_t \cdot x) \cdot L(0). \quad (24)$$

Transmittance Applying a very similar reasoning as in this last example in eq. (23) to the RTE in eq. (22), we arrive at a very similar exponential factor:



$$T(\mathbf{x}, \mathbf{y}) = e^{-\tau(\mathbf{x}, \mathbf{y})}, \quad (25)$$

$$\tau(\mathbf{x}, \mathbf{y}) = \int_0^d \mu_t(\mathbf{x} + t \cdot \omega) dt, \quad (26)$$

where $d = \|\mathbf{x} - \mathbf{y}\|$ and $\omega = (\mathbf{x} - \mathbf{y})/d$. $T(\mathbf{x}, \mathbf{y})$ is called the *transmittance* and computes the fraction $\in [0, 1]$ of light which will make it from \mathbf{x} to \mathbf{y} . This attenuation is called the Beer-Lambert law, and $\tau(\mathbf{x}, \mathbf{y})$ is called the *optical thickness*. Note how the transmittance for homogeneous materials (i.e. $\mu_t \equiv \text{const.}$) degenerates to a factor very much like the one in eq. (24), illustrated in the graph next to eq. (25). On the other hand, this classic kind of exponential attenuation is challenged by recent works by Bitterli et al. [2018], d'Eon [2018] and Jarabo et al. [2018], who do not assume a uniform random distribution of particles.

Boundary conditions To solve the RTE, we still need to define boundary conditions. These can be found in surface transport, formalised as a recursive integral equation by Kajiya [1986]:

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{\Omega} f_r(\mathbf{x}, \omega, \omega_i)L(\mathbf{x}, \omega_i) d\omega_i^\perp, \quad (27)$$

where the integration domain Ω is the (hemi-)sphere of incoming directions ω_i , and the measure $d\omega_i^\perp = \cos \theta d\omega$ is the projected solid angle measure, which includes a foreshortening factor to account for Lambert's law. The term $f_r(\cdot)$ 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.

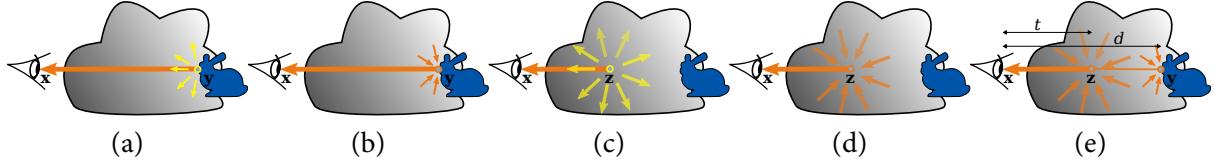


Figure 1: Illustration of all events taking part in the change of radiance along a ray: surface emission (a), surface scattering (b), volume emission (c), volume scattering (d), and the distances as they appear in eq. (28).

The integral form of the RTE With an integrating factor (transmittance) and boundary conditions (surface transport) we are now ready to convert the RTE to integral form. Putting everything together, the result is

$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{y}) \underbrace{\left(L_e(\mathbf{y}, \omega) + \int_{\Omega} f_r(\omega_i, \mathbf{y}, \omega) L(\mathbf{y}, \omega_i) d\omega_i^{\perp} \right)}_{\text{contribution from the point } \mathbf{y} \text{ on surface}} + \int_0^d T(\mathbf{x}, \mathbf{z}) \underbrace{\left(\mu_e(\mathbf{z}) L_e(\mathbf{z}, \omega) + \mu_s(\mathbf{z}) \int_{\Omega} f_s(\omega \cdot \omega_i) L(\mathbf{z}, \omega_i) d\omega_i \right) dt}_{\text{contribution from any point } \mathbf{z} \text{ at distance } t \text{ in volume}}. \quad (28)$$

Figure 1 visualises all terms in the integral form of the RTE. Since this is somewhat bulky, we can simplify the notation by dropping a few parameters and introducing a linear transport operator T , which represents the scattering at either the point on the surface \mathbf{y} or a point \mathbf{z} in the volume, as introduced by Veach [1998]:

$$L = L_e + TL. \quad (29)$$

5.2 The path space

With all the physical foundation and the mathematical tools at hand, we can now determine how much light flows along a single transport path. Explicitly unrolling the recursion in eq. (29) for an example path with five vertices (cf. fig. 2) shows that we need to transport the emitted radiance four times. In general

$$L = T^{k-1} L_e \quad (30)$$

for k path vertices. Substituting the original terms from eq. (28) instead of the transport operator shorthand, this results in the “flat view” of the rendering equation, the measurement contribution function as defined by Veach [1998]:

$$f(\mathbf{X}) = L_e(\mathbf{x}_1) T(\mathbf{x}_1, \mathbf{x}_2) G(\mathbf{x}_1, \mathbf{x}_2) \cdot \left(\prod_{i=2}^{k-1} f_x(\mathbf{x}_i) T(\mathbf{x}_i, \mathbf{x}_{i+1}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) \cdot W(\mathbf{x}_k). \quad (31)$$

This formulation contains two new terms: 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.

The other term, G is called the geometry term and appears here as the Jacobian determinant from projected solid angle $d\omega^{\perp}$ to vertex area measure $d\mathbf{x}$, such that we can integrate $f(\mathbf{X})$ over the path space in product vertex area measure. This measure space is useful because it is agnostic of the transport direction and allows for easy inclusion of *next event estimation* samples which directly sample transport points on surfaces instead of outgoing directions. The geometry term G also contains the mutual visibility $V(\mathbf{x}, \mathbf{y})$

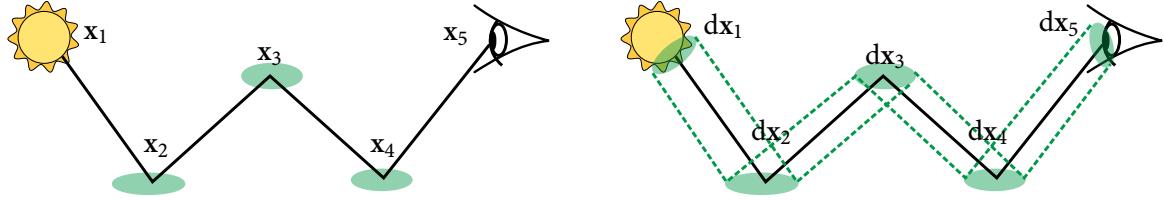


Figure 2: A light transport path is represented as a list of vertices (left). Each vertex comes with an integration domain in vertex area measure, as indicated by dx_i in the right image. The full integration domain consists of the product of all these infinitesimal surface patches, the *product vertex area measure* as indicated by the dashed tube around the path.

between the points. More precisely

$$G(\mathbf{x}, \mathbf{y}) = V(\mathbf{x}, \mathbf{y}) \frac{D(\mathbf{x}, \mathbf{y})D(\mathbf{y}, \mathbf{x})}{\|\mathbf{x} - \mathbf{y}\|^2}, \quad (32)$$

$$D(\mathbf{x}, \mathbf{y}) = \begin{cases} |n(\mathbf{x}) \cdot \omega_{\mathbf{x} \rightarrow \mathbf{y}}| & \text{if } \mathbf{x} \text{ is on a surface,} \\ 1 & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \quad (33)$$

$$L_e(\mathbf{x}, \mathbf{y}) = \begin{cases} L_e(\mathbf{x}, \omega_{\mathbf{x} \rightarrow \mathbf{y}}) & \text{if } \mathbf{x} \text{ is on a surface,} \\ \mu_e(\mathbf{x})L_e(\mathbf{x}, \omega_{\mathbf{x} \rightarrow \mathbf{y}}) & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \quad (34)$$

$$f_x(\mathbf{x}) = \begin{cases} f_r(\mathbf{x}) & \text{if } \mathbf{x} \text{ is on a surface,} \\ \mu_s(\mathbf{x})f_s(\mathbf{x}) & \text{if } \mathbf{x} \text{ is in a medium.} \end{cases} \quad (35)$$

The scattering function f_x is generalised to express the BSDF f_r on surfaces as well as the scattering collision coefficient and phase function $\mu_s(\mathbf{x}) \cdot f_s(\omega, \omega_i)$ inside media. Note that for brevity we dropped the dependency of the scattering functions on the incoming and outgoing directions in eq. (31) and eq. (35).

To model a camera as a measurement apparatus, collecting photons per time incident on the pixel area from a certain solid angle, i.e. to compute radiant power, all we need to do is integrate the measurement contribution function eq. (31) over all vertex areas dx associated with a path. For instance for the path configuration in fig. 2, we get

$$\int \int \int \int \int f(\mathbf{X}) dx_1 dx_2 dx_3 dx_4 dx_5, \quad (36)$$

which means we need to integrate over five individual surface patches in square meters (this would be cubic meters for vertices in volumes). As a shorthand, we define $d\mathbf{X}_5$ as a product measure for paths with five vertices. Now light is not restricted to be transported only by paths of length five, but in general we need to sum up contributions of all path lengths, for instance by explicitly expanding the *Neumann series*

$$L = L_e + TL_e + T^2L_e + \dots \quad (37)$$

To be able to write one single integral for transport paths of any length, we also define $d\mathbf{X}$ to signify the union of all $d\mathbf{X}_k = \prod_{i=1}^k dx$ with $k \geq 2$. This finally enables us to unify lighting computations in one single mathematical framework (be it surfaces, sub-surface scattering, or volume contributions). Intuitively, we 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 are then “counted” on the sensor.

The sampling space we just defined is called the *path space* \mathcal{P} and contains all possible transport paths $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\} \in \mathcal{P}$, which are lists of k path vertices \mathbf{x} (see fig. 2). To compute the color I_p of a pixel p , we simply integrate over this space, weighted by a pixel filter $h_p(\mathbf{X})$, such as for instance the one derived by Harris [1978]. Note that sometimes the pixel filter $h_p(\mathbf{X})$ is folded into the camera responsivity $W(\mathbf{x}_k)$.

In summary, we finally arrive at the path integral for global illumination:

$$I_p = \int_{\mathcal{P}} h_p(\mathbf{X}) \cdot f(\mathbf{X}) d\mathbf{X}. \quad (38)$$

Now the only remaining challenge is to efficiently sample good transport paths \mathbf{X} to numerically evaluate the integral.

5.3 The Monte Carlo method

Equation (38), i.e. integrating the measurement contribution function over path space, presents 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

$$\mathbb{E}(x) = \int x \cdot p(x) dx \quad (39)$$

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

$$\hat{x} = \frac{1}{N} \sum_{i=1}^N x_i \approx \mathbb{E}(x). \quad (40)$$

Applying this same principle to the path space integral, we need to divide out the probability distribution function $p(\cdot)$ from the integrand $f(\mathbf{X})$ to match the definition of the expected value in Eq. (39):

$$I_p = \int_{\mathcal{P}} h_p(\mathbf{X}) \cdot f(\mathbf{X}) d\mathbf{X} \approx \hat{I}_p = \frac{1}{N} \sum_{i=1}^N \frac{h_p(\mathbf{X}_i) \cdot f(\mathbf{X}_i)}{p(\mathbf{X}_i)}. \quad (41)$$

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

$$Var(\hat{I}_p) = \frac{1}{N} \int \left(\frac{h_p(\mathbf{X})f(\mathbf{X})}{p(\mathbf{X})} - I_p \right)^2 p(\mathbf{X}) d\mathbf{X}. \quad (42)$$

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. (42) shows that the primary estimator $h \cdot f/p$ needs to be close to I_p to reduce variance. In practise, 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.

5.4 Colour formation in a renderer

While the first section introduced the path tracing framework and the transport equations for a full path, these were written without dependency on wavelength λ . 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.

Colour can come into a rendering by simulating three discrete wavelengths. This corresponds to RGB transport using the CIE RGB primaries. Note that most other RGB colour spaces correspond to some weighted average of wavelength ranges, potentially employing negative weights. Even when simulating the wavelength domain directly, we usually assume only *elastic* scattering happens. This means that the energy of a photon remains the same after an event and only depends on the wavelength. In particular a photon can only be absorbed or scattered into a different direction, but no inter-wavelength transport is considered (such as it would be required to accurately model fluorescence or phosphorescence).

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:

$$f(\mathbf{X}, \lambda) = L_e(\lambda) \cdot G_1 \cdot f_r(\lambda) \cdot G_2 \cdot W(\lambda). \quad (43)$$

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

$$X = \int_{380..830nm} f(\lambda) \cdot \bar{x}(\lambda) d\lambda, \quad (44)$$

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

$$R = f(\lambda = 700nm), \quad G = f(\lambda = 546.1nm), \quad B = f(\lambda = 435.8nm). \quad (45)$$

In general, using any other RGB space to perform the multiplications in Eq. (43) and replacing the integral in Eq. (44) like in Eq. (45) 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.

5.4.1 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:

$$\int_{\Omega} f_r(\mathbf{x}, \omega, \omega_i, \lambda) d\omega_i^{\perp} \leq 1 \quad \forall \lambda. \quad (46)$$

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

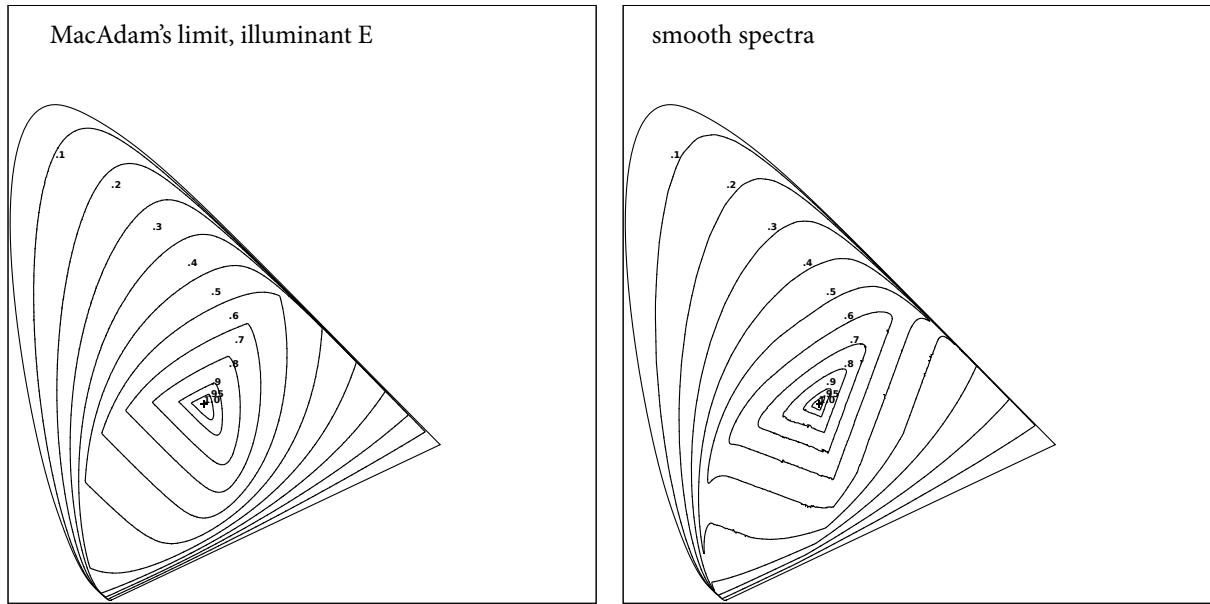


Figure 3: The maximal gamuts of surface reflection, reproduced after Meng et al. [2015]. The graphs show the maximum brightness ($X + Y + Z$) of a surface colour (for instance diffuse albedo), as an iso line graph. The coordinate system as seen from the top is the standard space of chromaticities, i.e. $x = X/(X + Y + Z)$ 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.

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. 3, 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 naive RGB transport.

5.4.2 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.

5.4.3 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 it has found adoption in production (in Weta Digital's Manuka), but may be limiting in some contexts that do not already depend on multiple importance sampling.

5.4.4 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 to incorporate such advanced models into a render.

The most important case, however, is chromatic extinction in participating media. That is, the extinction coefficient $\mu_t(x, \lambda)$ depends on the wavelength. This governs the transmittance term eq. (25) which is simply $\exp(-\mu_t(\lambda) \cdot d)$ 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 sub-surface 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 often the most important one 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).

5.4.5 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 luminosity 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 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 $X_\lambda(\dots, \lambda)$ the corresponding photometric quantity $X_\nu(\dots)$ is simply obtained integrating X_λ against $K_m \cdot \bar{y}(\lambda)$ where K_m is a scaling constant about equal² to 683:

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

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

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

²The 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]

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

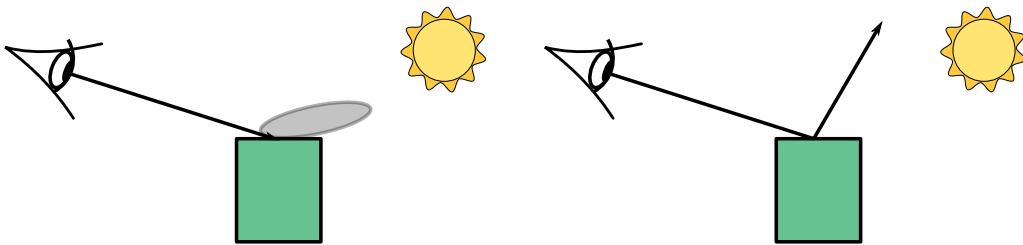
Table 1: Correspondence between radiometric and photometric units. We abbreviate the unit for luminous energy *talbot* as Tb instead of the also common T to avoid confusion with the unit for magnetic flux *tesla*. We also use the convention of subscripting photometric quantities with v (for *visual*),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

5.5 Path construction techniques

This section will give a quick run down on a few basic path construction techniques and the related problems. The examples here are generic and simplified, in practical production scenarios they will be aggravated by difficult geometry, complex materials, and combinations of many lighting effects.

5.5.1 Path tracing

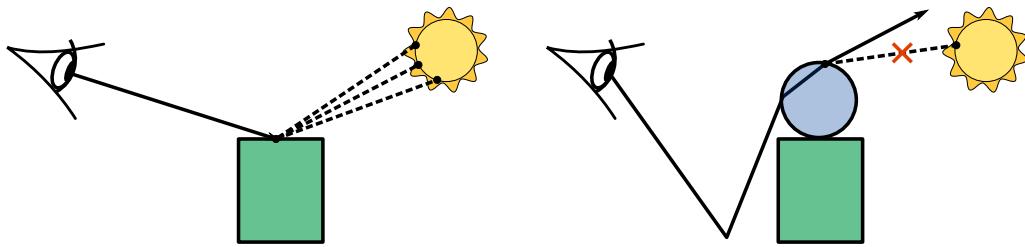
In the most simple incarnation as described by Kajiya [1986], path construction proceeds by sampling path vertices iteratively by successively extending the path starting at the eye or camera sensor. This involves sampling a pixel, a point on the camera aperture, and tracing a ray to the first intersection with the scene. At this point, the BSDF is queried to generate a good outgoing direction given only the incoming ray (left):



This local decision is not necessarily a good idea globally, however. In the example in the right image above, the generated outgoing direction misses the light source. This is especially problematic for small and far away light sources, subtending a small solid angle as seen from the shading point.

5.5.2 Next event estimation

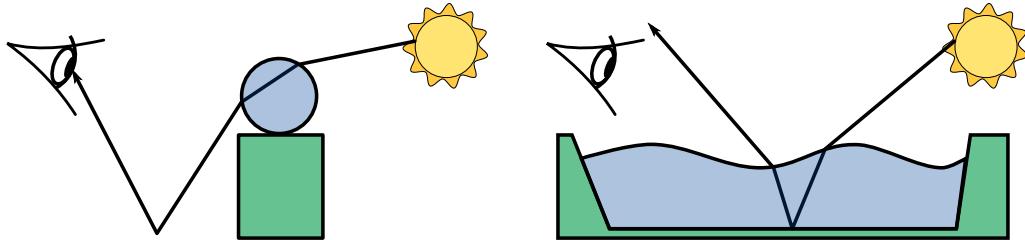
The problem outlined in the previous section is addressed by a technique coined *Next Event Estimation* (NEE). In this context, a path is directly connected to the light sources by explicitly sampling the surface area of the emitter in vertex area measure instead of sampling an outgoing direction in projected solid angle measure. This has been known in neutron transport for a long time and coined *next event estimation* by Coveyou et al. [1967]. Connections can be performed multiple times from the same vertex to increase sampling efficiency for long paths with difficult illumination (left):



This technique, however, disregards the BSDF. In extreme cases this can result in no throughput at all, if the BSDF contains a Dirac delta, such as smooth specular materials in the example on the right. The problem persists with glossy materials, too, and gradually becomes less of an issue for diffuse materials. Unfortunately even those will have sub optimal importance sampling when using straight uniform area sampling as opposed to more advanced mappings such as for instance the one proposed by Ureña et al. [2013].

5.5.3 Light tracing

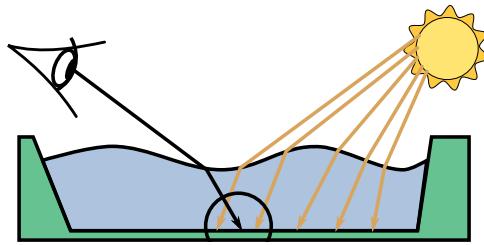
Another technique devised by Arvo [1986] to solve the problem in the last section is called *light tracing* as opposed to path tracing, because the random walk starts at the emitters and performs next event estimation to the eye point (left):



Unfortunately it is very easy to construct symmetric fail cases for this type of next event estimation by putting a smooth specular material between the eye point and the diffuse surface (see right image). Light tracing can be combined via multiple importance sampling with all previously mentioned techniques to generate a more robust composite estimator. The example in the image above is however one that cannot be rendered efficiently by bidirectional path tracing. This is because the diffuse event on the bottom of the pool is encapsulated by specular events on both sides, a so called SDS situation. Again, this is a problem even if the BSDF are not perfectly smooth specular, sampling efficiency will smoothly degrade as the BSDF tends to a Dirac delta.

5.5.4 Photon mapping-based approaches

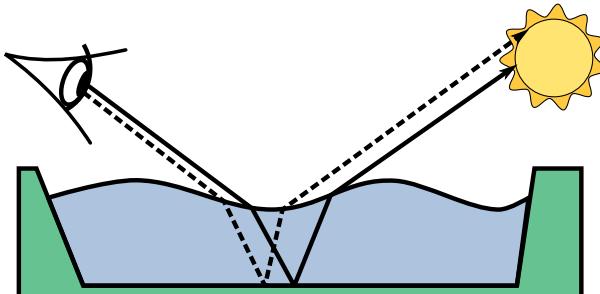
To render such SDS scenarios, one can employ path caches by means of photon mapping (see the works of Georgiev et al. [2012], Hachisuka et al. [2012], Jensen [1996]). Techniques employing this are usually two-pass methods, first tracing fractional light packets (colloquially called photons in computer graphics) or gather points from the light or the eye, respectively. In this example image, "photons" are traced from the light, as indicated by the yellow paths:



It is then possible in a second pass, here performed from the eye (indicated by the black rays), to collect these photons via kernel estimation (indicated by the black circle). Since many "photons" are traced in the first pass, it is likely to intersect them in the second pass, facilitating efficient reuse of computation. Drawbacks of this method include that photons aren't distributed according to importance from the eye and can be wasteful without further refinement of the method, such as the one proposed by Gruson et al. [2016]. The kernel estimation is not always simple and can lead to visible bias near complex geometry, which is unfortunately quite important for production scenes. This is especially apparent on hair and fur.

5.5.5 Markov chain-based methods

An unbiased alternative is the family of Markov chain-based methods. These include Metropolis light transport as introduced by Veach and Guibas [1997], primary sample space Metropolis light transport by Kelemen et al. [2002], and Hamiltonian Monte Carlo as introduced to graphics by Li et al. [2015]. On a high level, these methods are based on a Markov chain that holds a current state (indicated as the dashed path in the figure):



This path is then mutated by iteratively proposing tentative new paths which are drawn from some transition probability distribution. These mutations can be modelled to explicitly handle a certain class of important transport, such as exploring specular manifolds (see Jakob and Marschner [2012]). Markov chains have traditionally suffered from temporal flickering. This is because we need to strike a balance between global discovery of important lighting effects (disconnected islands in path space, potentially with different dimensionality than the current path) and local exploration (small mutations). Plain Monte Carlo is good at the first task, especially when combined with stratified sampling such as when using the randomly scrambled Halton sequence. Markov chain-based methods are usually very good at the second task, but fail to stratify the samples globally, resulting in temporal instability.

5.6 Conclusion

This introductory section summarised the most essential and basic concepts of light transport theory, the related equations, a few thoughts on colour reproduction which is essential to match live footage in production, as well as the most common path space sampling techniques. 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, focusing on the specific set of requirements with respect to speed and quality that govern production rendering.

References

- Steve Agland. 2014. CG Rendering and ACES. [\(2014\).](http://nbviewer.ipython.org/gist/sagland/3c791e79353673fd24fa) <http://nbviewer.ipython.org/gist/sagland/3c791e79353673fd24fa>.
- Marco Ament, Christoph Bergmann, and Daniel Weiskopf. 2014. Refractive Radiative Transfer Equation. *ACM Trans. on Graphics* 33, 2 (April 2014), 17:1–17:22. <https://doi.org/10.1145/2557605>
- James Arvo. 1986. Backward Ray Tracing. In *SIGGRAPH Course Notes*. 259–263.
- James Arvo. 1993. Transfer equations in global illumination. In *SIGGRAPH Course Notes*.
- Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. *A radiative transfer framework for non-exponential media*. Technical Report.
- Subrahmanyam Chandrasekar. 1960. *Radiative Transfer*. Dover Publications Inc. ISBN 0-486-60590-6.
- 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.
- R. R. Coveyou, V. R. Cain, and K. J. Yost. 1967. Adjoint and Importance in Monte Carlo Application. *Nuclear Science and Engineering* 27, 2 (1967), 219–234. <https://doi.org/10.13182/NSE67-A18262>
- Eugene d’Eon. 2018. A reciprocal formulation of non-exponential radiative transfer. 1: Sketch and motivation. *ArXiv e-prints* (March 2018). arXiv:physics.comp-ph/1803.03259
- Sergej Mikhailovich Ermakow. 1975. *Die Monte Carlo Methode und verwandte Fragen*. VEB Deutscher Verlag der Wissenschaften.
- 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.
- Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 31, 6 (2012), 192:1–192:10.
- Andrew S. Glassner. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann.
- Adrien Gruson, Mickaël Ribardière, Martin Šik, Jiří Vorba, Rémi Cozot, Kadi Bouatouch, and Jaroslav Křivánek. 2016. A Spatial Target Function for Metropolis Photon Tracing. *ACM Trans. on Graphics* 36, 1 (Nov. 2016), 4:1–4:13. <https://doi.org/10.1145/2963097>
- Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 31, 6 (2012), 191:1–191:10.
- Frederic J. Harris. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE* 66, 1 (1978), 51–83.
- L. Henyey and J. Greenstein. 1941. Diffuse radiation in the Galaxy. *Astrophysical Journal* 93 (1941), 70–83.
- Matthias Hullin, Johannes Hanika, Boris Ajdin, Jan Kautz, Hans-Peter Seidel, and Hendrik Lensch. 2010. Acquisition and Analysis of Bispectral Bidirectional Reflectance and Reradiation Distribution Functions. *Transactions on Graphics (Proceedings of SIGGRAPH)* 29, 4 (2010), 1–7.

- Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 58:1–58:13.
- Wenzel Jakob, Jonathan T. Moon, Adam Arbree, Kavita Bala, and Steve Marschner. 2010. A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 29, 10 (July 2010), 53:1–53:13. <https://doi.org/10.1145/1778765.1778790>
- Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A Radiative Transfer Framework for Spatially-Correlated Materials. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 37, 4 (2018).
- Adrian Jarabo and Diego Gutierrez. 2016. Bidirectional Rendering of Polarized Light Transport. In *Proceedings of CEIG '16*.
- Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. 2014. A Framework for Transient Rendering. *ACM Trans. on Graphics* 33, 6 (Nov. 2014), 177:1–177:10. <https://doi.org/10.1145/2661229.2661251>
- Wojciech Jarosz. 2008. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. Ph.D. Dissertation. UC San Diego, La Jolla, CA, USA. Advisor(s) Henrik Wann Jensen and Matthias Zwicker.
- Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Proc. Eurographics Workshop on Rendering*. 21–30.
- James T. Kajiya. 1986. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)* (1986), 143–150.
- Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum* 21, 3 (2002), 531–540.
- Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 34, 6 (Nov. 2015), 209:1–209:13.
- David L. MacAdam. 1935. Maximum Visual Efficiency of Colored Materials. *Journal of the Optical Society of America* 25, 11 (1935), 361–367.
- 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.
- Jan Novák. 2014. *Efficient Many-Light Rendering of Scenes with Participating Media*. Ph.D. Dissertation. Karlsruhe Institute of Technology”.
- Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum (Eurographics State of the Art Reports)* 37, 2 (2018), 1–26.
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2017. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc.
- 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.
- Ilya Sobol'. 1994. *A Primer for the Monte Carlo Method*. CRC Press.
- John Strutt. 1871. On the light from the sky, its polarization and colour. *Philos. Mag.* 41, 4 (1871), 107–120,274–279.
- Kip Thorne. 2014. . W. W. Norton & Company.
- Carlos Ureña, Marcos Fajardo, and Alan King. 2013. An Area-preserving Parametrization for Spherical Rectangles. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2013), 59–66. <https://doi.org/10.1111/cgf.12151>
- USAS and ASME. 1967. *USA Standard Letter Symbols for Illuminating Engineering*. United States of America Standards Institute.
- Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.
- Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. *Proc. SIGGRAPH* (1997), 65–76.
- Sebastian Werner, Zdravko Velinov, Wenzel Jakob, and Matthias B. Hullin. 2017. Scratch Iridescence: Wave-optical Rendering of Diffractive Surface Structure. *ACM Trans. on Graphics* 36, 6 (Nov. 2017), 207:1–207:14. <https://doi.org/10.1145/3130800.3130840>
- 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.

6 Practical Path-Tracing at ILM

ANDRÉ MAZZONE, *Industrial Light & Magic*

Path-tracing was introduced to the field of computer graphics Kajiya [1986] as an alternative to distribution ray-tracing, where a complete lighting integration is computed at each ray-object intersection. While distribution tracing was effective at reducing variance due to motion-blur, depth of field and other effects, the technique suffered from exponential increases in execution time with successive ray depths.

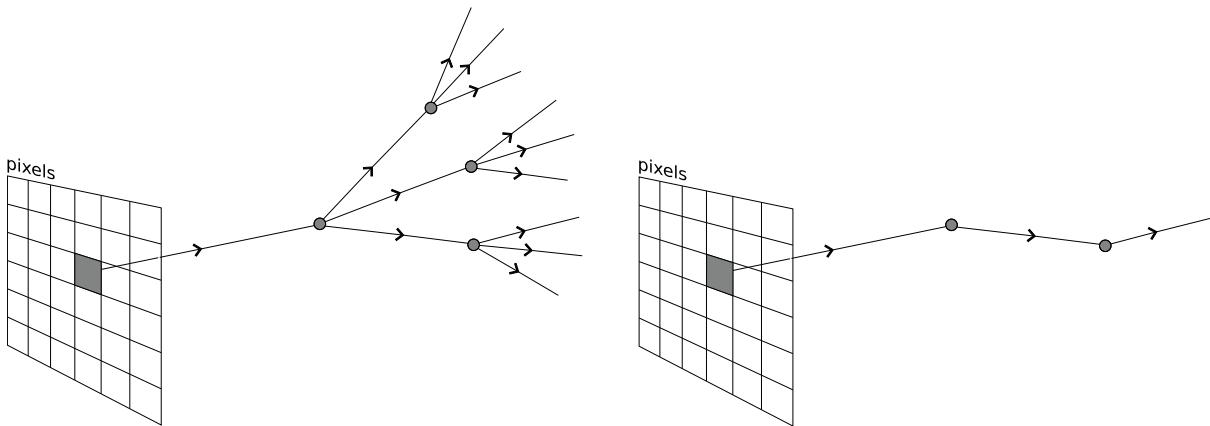


Figure 4: Distribution ray-tracing (left) spawns multiple rays at each object intersection. Path-tracing (right) instead chooses a single path.

6.1 Path-Tracing Changes Everything

ILM has been using path-tracing in production for the majority of its CG production since 2016 with the use of RenderMan RIS in Doctor Strange and Rogue One: A Star Wars Story.



Figure 5: Composited path-traced CG elements. *Rogue One: A Star Wars Story* ©2016 Lucasfilm Ltd.

Previously, to this, RenderMan REYES Cook et al. [1987], a raster-based renderer, was the primary engine delivering CG elements. The adoption of path-tracing caused dramatic changes to ILM's production pipeline.

6.1.1 Reduced Artistic Iteration Time

From an artistic perspective, the immediate advantage of path-tracing over REYES and distribution ray-tracing, is its ability to perform multiple passes over the image and provide progressive updates as the rendering of the scene converges. Instead of waiting for an entire process to complete, immediate feedback allows artistic judgements to be made. This reduces greatly the amount of time between iterations toward the desired artistic result. Interactive re-rendering accelerates this process even further as scene modifications can be made without costly render process restarts. It is important to note that progressive path-tracing is in many cases slower than distribution ray-tracing or raster-based methods when total render times are measured, however, the advantages of progressive updates greatly outweigh the shortcomings.

6.1.2 Reduced Total Render Time

During shot production, daily reviews can now be performed with incomplete renders. Progressive rendering allows checkpoint images to be written at regular intervals and these can often be of sufficient quality to allow project supervisors to provide direction. If changes are required, these in-progress processes can be halted easily. Conversely, if approval is granted, these renders can be allowed to complete. Furthermore, render processes can be halted and restarted, allowing more flexible resource allocation. Thus, even if progressive renders of single frames take longer than non-progressive alternatives, much fewer frames are required to render to completion. Analysis of resource usage at ILM has shown that in some cases, the total computation time over the lifetime of a shot can be as little as half when compared to shots rendered with REYES.

6.1.3 Increased Geometric Complexity

REYES rendering, when combined with ray-traced secondary effects, required multiple in-memory geometric representations of the scene. Furthermore, there was a fixed performance overhead that scaled with the number of vertices in geometric primitives. Switching to path-tracing removed both of these limitations with improved memory utilisation as a result. In some cases, displacement mapping has been replaced with explicit geometric detail without detrimental effects on render performance. Instancing is also more efficient in a ray-racing context making it easier to add geometric complexity with substantially less memory usage.

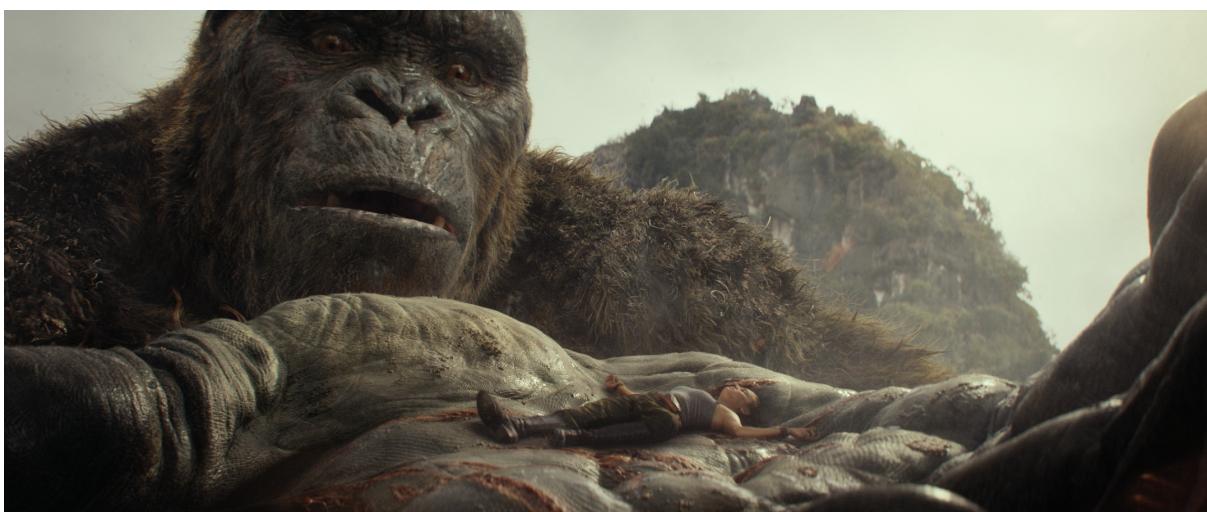


Figure 6: Kong's fingerprints are geometric details not displacement maps. *Kong: Skull Island*. ©2017 Warner Bros.

6.1.4 Increased Lighting Complexity

Switching away from REYES to path-tracing has also allowed renders to increase the number of indirect bounces that are practical to use. With REYES, adding a single bounce of indirect diffuse was a costly decision made per shot and only if necessary. With path-tracing, the typical default maximum ray-depth is now three with more added when appropriate. Volumetric path-tracing is also now routinely used to compute subsurface scattering. Previous diffusion-based approximations were prone to artefacts especially in areas of high curvature. Path-tracing has enabled a higher level of lighting fidelity.

6.1.5 Simplified Scene Setup

Another significant benefit of path-tracing over previous methods is that all computation is performed within the same rendering framework. Legacy raster-based rendering pipelines required a variety of pre-passes to emulate the visually important aspects of light transport. With ray-tracing, many of these pre-passes are made redundant and a wider variety of effects can be emulated within a single render pass. An added benefit of using a single rendering framework is that shader development is also correspondingly simplified. With only one paradigm to support, effort can be focussed and duplication avoided.

6.2 Shortcomings of Path-Tracing

However, despite the numerous advantages of path-tracing over previous methods, it still requires careful tuning if performance is to be achieved.

6.2.1 Noise

When path-tracing is compared with raster-based rendering, one of the most widely reported problems is noise. Integration using Monte Carlo estimators can exhibit many different kinds of noise. Furthermore, convergence rates are such, that in general to halve the variance, four times as many samples must be taken. Commonly observed categories of noise include those caused by the following:

6.2.2 Geometric Complexity

When developing CG assets that are intended to be rendered close to camera, significant geometric complexity is required for photo-realistic results. While this is especially true of hair, detailed meshes can also contribute to geometric noise. This high-frequency detail can require larger sample counts to achieve convergence. Volumetric primitives are also prone to noise even with recent advances in integration techniques.

6.2.3 Surface Complexity

Noise can also be seen when simulating surface properties. Rough specular or glossy lobes, for example, can require large numbers of samples for converged results when simulating simple reflection or refraction. Bump mapping, can also contribute to noisy renders by amplifying challenging integration situations. Subsurface scattering also typically requires large numbers of samples to produce smooth results. When multiple lobes are being sampled on a surface, choosing the appropriate one to be evaluated can also contribute to poor convergence properties

6.2.4 Lighting Complexity

Shadows with large penumbra regions, either due to large lights or complex occluders are often contributors to noisy renders. Small lights illuminating reflective smooth surfaces are also sources of variance as is indirect transport in areas of high contrast lighting conditions. New techniques like path-guiding Müller et al. [2017] show promise in resolving these issues. Caustics are also challenging, though there

are integration techniques that improve this greatly (e.g. manifold next event estimation Hanika et al. [2015]). Rendering scenes with challenging light transport is an area of active research.

6.2.5 Camera Complexity

Using lens models and ray-tracing to resolve depth of field effects can be very costly. ILM typically uses 2D image processing techniques instead, both to avoid this expense, and to provide extra flexibility during compositing.

6.2.6 Motion Complexity

Though admittedly unrealistic in its results, REYES produced practically noise free motion-blur with raster-based methods. Path-tracing is quite effective at resolving the geometric aspect of motion-blur as alpha channels are typically clean after 100 camera sample per pixel in typical cases. However, when high intensity specular reflections are blurred, convergence is not guaranteed even with many thousands of samples per pixel.



Figure 7: Detail of poorly converged motion-blur when sharp specular highlights are present. *Transformers: The Last Knight*. ©2017 Paramount Pictures.

6.2.7 Light Picking

In order to choose a single sample for path-traced lighting, an algorithm must estimate a cumulative probability distribution for the lights in the scene. This is a challenging task to perform efficiently and the quality of the estimation will directly influence convergence rates.

6.2.8 Combinatorial Complexity

Combining any of the issues enumerated above contributes to correspondingly more difficult rendering scenarios.

6.3 Remedies

Given the large list of issues above, the number of tools that can be used to address them is unfortunately limited.

6.3.1 Splitting

Pure path-tracing spawns only a single ray at each scattering event, however, it's commonly observed that geometric complexity accounts for less noise than other contributions from light transport. Thus, increasing the number of camera samples per pixel in such a situation would result in oversampling of the geometry. Adding a branching factor, or splitting, is a useful technique in order to distribute samples more effectively. Sending more samples at the first intersection point and proceeding with regular path-tracing thereafter can often reduce total render times. In the unidirectional path-tracing mode that ILM uses in RenderMan, splits can be added to light, surface, and indirect sampling independently. Choosing the right ratio of each is not always trivial and can also be scene dependent.

6.3.2 Adaptive Sampling

Another advantage of progressive path-tracing is that adaptive sampling fits well into this paradigm. If a suitably accurate estimation of variance can be computed for a pixel, then the adaptive sampler can choose to send more samples into the areas where variance is above a user chosen threshold. Though simple in concept, an effective implementation is still quite challenging to develop. Firstly, calculating exact variance is not possible in an image that has yet to converge, so estimates must be made based on the results of the observed samples. Secondly, finding a heuristic that uses variance to decide which pixels are in need of more samples is also challenging. For example, the case of a motion-blurred highlight, energy is distributed across multiple pixels. A good adaptive sampler, upon discovering a pixel with high intensity samples, should also search in its neighbourhood. Again, this remains an area of active research.

6.3.3 Level of Detail

A common theme in the list of issues is that high-frequency complexity is a cause of noise. A simple approach to avoid some of these issues is to create simpler versions of assets for use in different situations. The details needed near to camera are different from those farther afield. Creating asset variations with different levels of detail is one way to avoid noise.

6.3.4 Lighting Simplification

Light picking performs poorly with large numbers of lights, especially in high contrast environments. Replacing groups of individual lights with simpler aggregate structures can be an effective tool to remove noise. In cases where lights don't contribute significantly to the final image, simply removing them or replacing them with textured geometry can result in substantial savings in render time. Many of the techniques used for optimisation in legacy lighting pipelines still apply to path-tracing.

6.3.5 Surface Simplification

When sharp specular response is a cause of noise, modifying the roughness is often an effective solution. Specular mollification, the process of adaptively roughening surfaces in areas high curvature, is a particular good solution for this since it doesn't require explicit material edits in the scene to achieve results. Roughening surfaces that are in motion is also sometimes a good way of reducing noise as is the blurring of high-frequency bump maps.

6.3.6 Blurring Motion-blur

In many cases, renders of motion-blurred objects do not converge within render budgets. In these cases, smoothing the motion blur with a 2d image convolution can reduce the artefacts sufficiently to allow an element to be used.

6.3.7 Post-Render Denoising

Substantial research has been expended in the field of image filtering in the search for automatic heuristics that can detect noise and remove it without operator intervention. This is commonly known as denoising. Many of the more effective methods use knowledge of the scene, often in the form of feature buffers, to direct an image filter where to blur selectively. It's also common to apply these filters on isolated components of renders separately. For example, the diffuse parts of an image can be divided by the albedo texture in the scene before filtering. Multiplying the albedo back into the filtered result retains texture detail while smoothing the under-sampled lighting. While these denoisers find common use in the CG industry and are sometimes systematically applied as a post-processing step, ILM has historically been very cautious in their use. One limitation of these methods is that a desired rendered effect does not have a corresponding representation in a feature buffer, then it can be filtered away. Good examples of this are small specular highlights and shadows. It's commonly observed at ILM that desirable high-frequency details are over-blurred and this loss of sharpness is considered an unacceptable compromise.

6.3.8 Denoising in Compositing

While ILM hasn't yet found an automatic denoiser to serve its purposes, the need for noise removal has not vanished. The unfortunate consequence of this is that much of the denoising necessary to produce acceptable final images now happens in our compositing pipeline. Various commercially available tools designed to remove film grain and other similar patterns can also be used for denoising with some success. Care must be taken, however, to avoid the over-blurring that makes unsupervised denoising insufficient. We continue to search for new ways to address this issue.

6.4 Conclusion

While Monte Carlo estimators are arguably the best tools currently available to render the complex images that the visual effects industry demands, there are obvious limitations. However, even with these admitted drawbacks, the use of path-tracing at ILM has contributed to a revolution in the way that artists work and produce images. The complexity that recent projects have required would not have been possible without this valuable paradigm.

References

- Robert L Cook, Loren Carpenter, and Edwin Catmull. 1987. The Reyes image rendering architecture. In *ACM SIGGRAPH Computer Graphics*, Vol. 21. ACM, 95–102.
- Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold next event estimation. In *Computer graphics forum*, Vol. 34. Wiley Online Library, 87–97.
- James T. Kajiya. 1986. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)* (1986), 143–150.
- Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 91–100.

7 More fun with light transports at Pixar!

CHRISTOPHE HERY, *Pixar Animation Studios*

RYUSUKE VILLEMIN, *Pixar Animation Studios*

JEREMY NEWLIN, *Pixar Animation Studios*

The film industry has switched to path tracing as its primary image synthesis method, and, at the same time, ad-hoc shading and lighting models were replaced by physically based and unbiased ones (see last year’s course notes on Path Tracing in Production in [Fascione et al. \[2017a\]](#) and [Fascione et al. \[2017b\]](#)).

One of the immediate benefits was opening the door to unbiased integration techniques, which fits well with progressive interactive rendering and can efficiently evaluate high-order multiple scattering events, instead of ad-hoc global illumination techniques requiring heavy preprocessing and producing incomplete results. Additional benefits include high predictability, simplified workflow and controls.

However the transition came with its own set of practical new problems, that are rarely discussed in academic papers.

In this document, we will take you through Pixar’s rendering pipeline, from shading to post-processing noise removal, and present you a collection of tips and tricks that help make physically based path tracing suitable for production rendering. Our goal is to make rendering as user-controllable as it was in the past with ad-hoc techniques, speed up the renders within the budget, but at the same time, take great care to maintain and not lose any of the advantages that came with physically based shading.

7.1 Fun with shader simplification

Even though PxrSurface (which has its major features explained in [Hery et al. \[2017\]](#)) attempts to achieve the best sampling possible, via such approaches as Heitz and D’Eon [2014]’s micro-facet visible normal strategies, Villemin et al. [2016]’s efficient resampling for diffusion models (in particular for Christensen [2015]) and Křivánek and d’Eon [2014]’s Dwivedi variance reduction for Wrenninge et al. [2017]’s brute-force subsurface scattering, and multi-lobe MIS (described in paragraph 3.1 of [Hery et al. \[2017\]](#)), there are still difficult light paths and difficult to converge scenarios that can be encountered. Also arbitrary pattern input shading and complex production materials can lead to a substantial cost for a final render in a path tracer (something that modern architectures like [WetaDigital \[2014\]](#) and [Eisenacher et al. \[2013\]](#) try to overcome in very different ways), especially at the end of non camera rays.

We propose an ad-hoc solution to these issues, leveraging a technique similar to classic Level of detail (see [Wikipedia \[2018\]](#)), and in particular its application for shading, as pioneered in [Hery and Sutton \[2001\]](#), but this time relating the simplification to specific Light Path Expressions (LPEs). First off, we would like to refer the reader to the introduction on LPEs we made in [Hery et al. \[2016\]](#).

It is also important to remember that the cheapest pattern is no pattern. With PxrSurface, this means leaving the various Gains of each lobe at 0, as much as possible (of course, full zeros everywhere lead to a black asset, not very useful). Each specific pattern entry adds up to the global cost. If a pattern input is disconnected, its static value can be employed, which obviously generates no corresponding parsing of the pattern network, but still can enable a feature of the bxdf to be turned on (if the corresponding Gain section is at 0, then early on PxrSurface will short circuit any connection of the subsection parameters and very little time is wasted for that section lobe).

There are also pattern operations that are more costly than others. For instance, bump mapping in a C++ RIS node requires 3 times the evaluation of the subgraph in order to compute proper derivatives and filtering. It turns out that the possible high frequency variation of normal orientations bump accomplishes (by its very definition) can also create a series of incoherent sampling directions, which is both bad for convergence and for subsequent shader evaluations at the end of the rays.

So the first thing we may want to eliminate is bump. In Fig. 8, we demonstrate the visual impact of removing bump from all ray types for a variety of simple materials. If you squint your eyes or imagine the objects seen from far away, this simplification can be acceptable. The computational win is already substantial: 2mn20s vs 1mn40s.

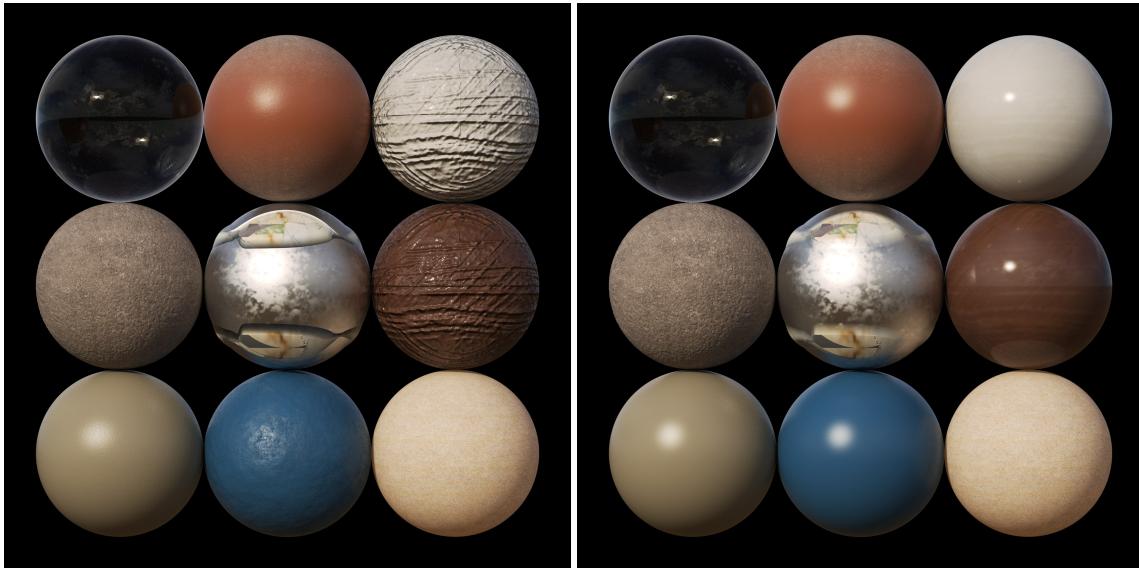


Figure 8: Left: No shader simplification. Right: Bump mapping simplification.

But we can go further. We can try to eliminate the specular computations where appropriate (for instance not on glass objects). We internally transfer the lost energy to the diffuse substrate. This definitely eliminates quite a lot of variance (since we are removing lobes to be sampled). While it is true that Fig. 9 is less convincing, the render time reduction is significant, similar to the removal of bump (1mn36). Moreover the two simplifications can be combined (leading to more savings). To compensate for the loss of details by these approximations, we will make use of LPEs to control the level of detail (LOD) appearance.

We can go all the way and apply all sorts of simplification heuristics to approximate the look. Ultimately the resulting assets only vaguely resemble their original appearance; in particular, the glass sphere, though still transparent, has “lost” its reflectivity. See Fig. 10. The render time in this case is down to 42s.

Now here is the real magic: the power of the LPEs. The simplifications we’ve discussed were applied on all light transports, i.e. through an LPE of $C[SD]^*$. But we can decide to restrict their effects to specific secondary paths. For instance, we can choose to apply them after the 1st bounce (and not through glass). The LPE becomes more complex:

$$\begin{aligned} & C(<.S6'passthru'>|<.S4'passthru'>)^* \\ & (D|S1|S2|S3|S8|<.S4[\wedge 'passthru']>|S5|<.S6[\wedge 'passthru']>|S7)[SD]^* \end{aligned}$$

³

With this rather arcane incantation, we now get results indistinguishable in camera shading, but fast results in secondary contexts (with special attention to glass and clearcoat). In addition we reduced the number of difficult (and variance prone) paths. The render with this expression is shown in Fig. 11. It is rather hard to notice in the image, however, if you look closely, the red sphere at the top gets less detail from the diffuse reflection of the metallic sphere in the middle (there are other differences). If this is not acceptable, one can push the LPE syntax to bounce 2 or bounce 3 or more (by changing the final $[SD]^*$ into $[SD]^+$ or $[SD][SD]^+$). With the bounce 1 syntax, in this simple setup, the render time reduction is also already measurable: from the original 2m20 down to 1mn45. In actual production shots, the savings are likely proportionally higher.

³In PxrSurface, S4 is clearcoat and S6 is glass

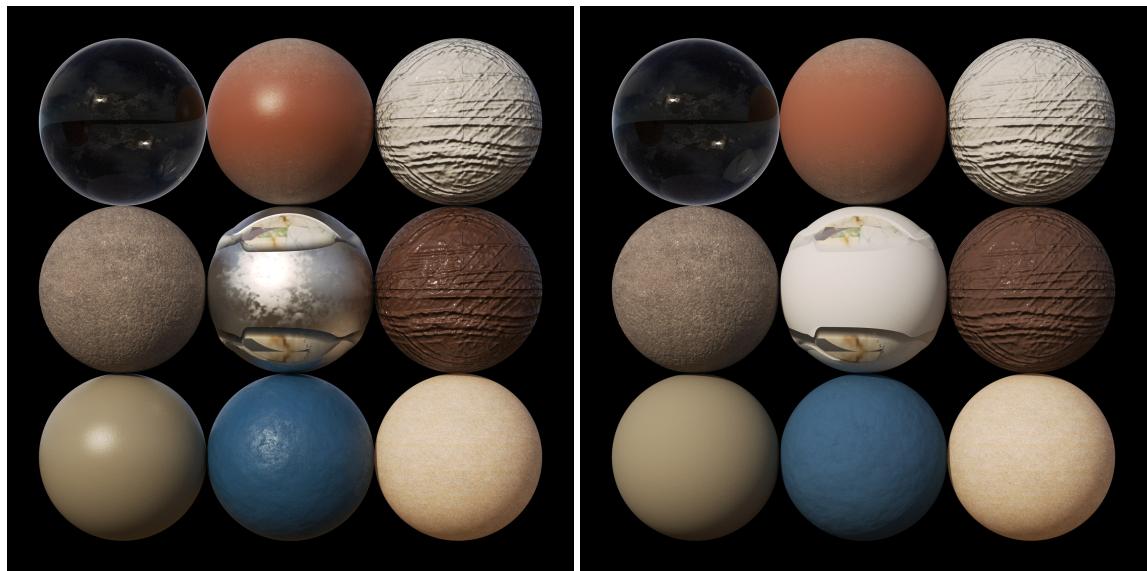


Figure 9: Left: No shader simplification. Right: Specular simplification.

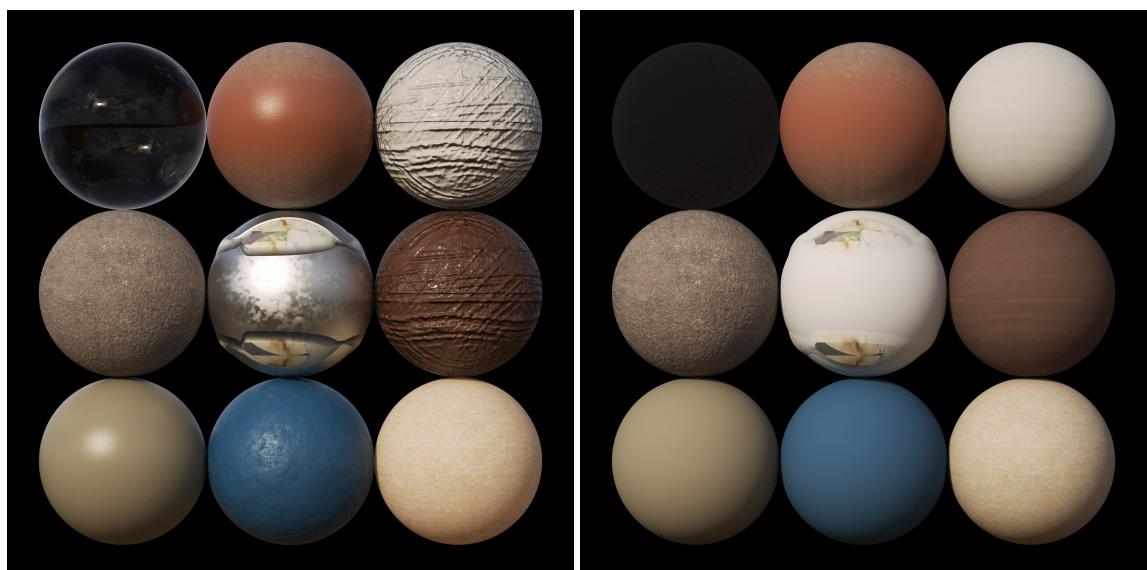


Figure 10: Left: No shader simplification. Right: All simplifications.

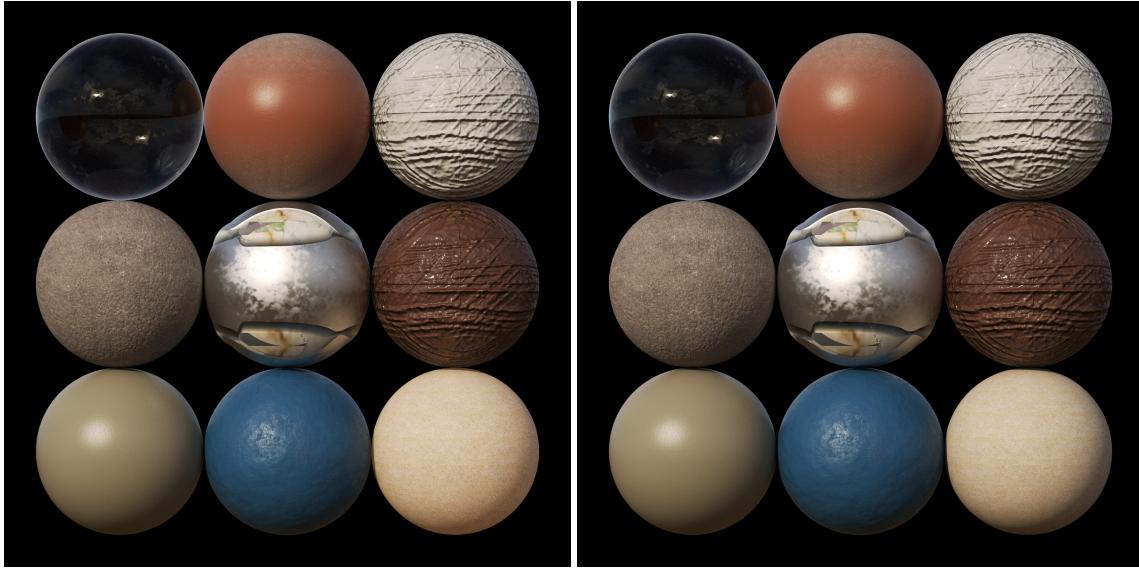


Figure 11: Left: No shader simplification. Right: All simplifications after the first bounce.

7.2 Atomic blonde

PxrMarschnerHair can achieve the look of all sorts of hair, as shown in Hery et al. [2017] and in Pekelis et al. [2015]. However, at the onset of *Incredibles 2*, the render times were deemed too high, in particular for the character *Dash*.

After a bit of investigation, we uncovered that the naturally high albedo of blonde hair generated a lot of energy at deep recursion levels, something that our Russian roulette scheme (similar to what is described in Chapter 10 of Veach [1997]) would not cut soon enough, without introducing significant noise in the image. Convergence was slow.

Though the show had already approved the appearance of *Dash*, they were willing to revisit his hair, if the renders could be done faster. So we came up with the idea of cheating the albedo.

For *Dash*, we went all the way to halving the hair albedo on input (for the TT, TRT and Glints components). To attempt to preserve some of the properties of the look they had achieved, we compensated by boosting the direct lighting on his hair by a value of two⁴.

As intended, this lowered the recursion depth on hair, while maintaining the direct illumination. But it had the side effect that any indirect illumination on hair (and not just from hair) would also be divided by 2. So we resorted yet again to our friend the LPE system to address this. This time, we would assign a filter that would double the energy on the following paths:

`C(<.[DS]'notHairOrSkin'>*<.[DS]'dlmHair'><.[DS]'notHairOrSkin'>+<L.>.`

The production was happy. They measured the savings at about 45% rendering time on *Dash* with this trick. Of course, if you observe the images in Fig. 12, the quality/softness is not fully there anymore⁵.

Analyzing what is going on, the image with the full model involves significant energy all the way past depth 20, as illustrated in Fig. 13. After the hackery, we tend to stop earlier, around depth 10, as per Fig. 14. And adaptive sampling has an easier time converging. The stats in Fig. 15 show that the average samples per pixel has gone down significantly.

⁴this can be done with many different setups; in our case, we created a user attribute assigned to the hair, containing the multiplication factor of 2, that the integrator would query before running the light loop

⁵in truth, the image on the right, with the hack active, is the look they re-tuned after they got the speed-up solution described; when the hack is not active on the left, it is not what they would have gone for, but it showcases the compromise achieved



Figure 12: Left: Without hair multiplying. Right: With hair multiplying.



Figure 13: Left: First 10 bounces without hair multiplying. Right: First 10 bounces with hair multiplying.

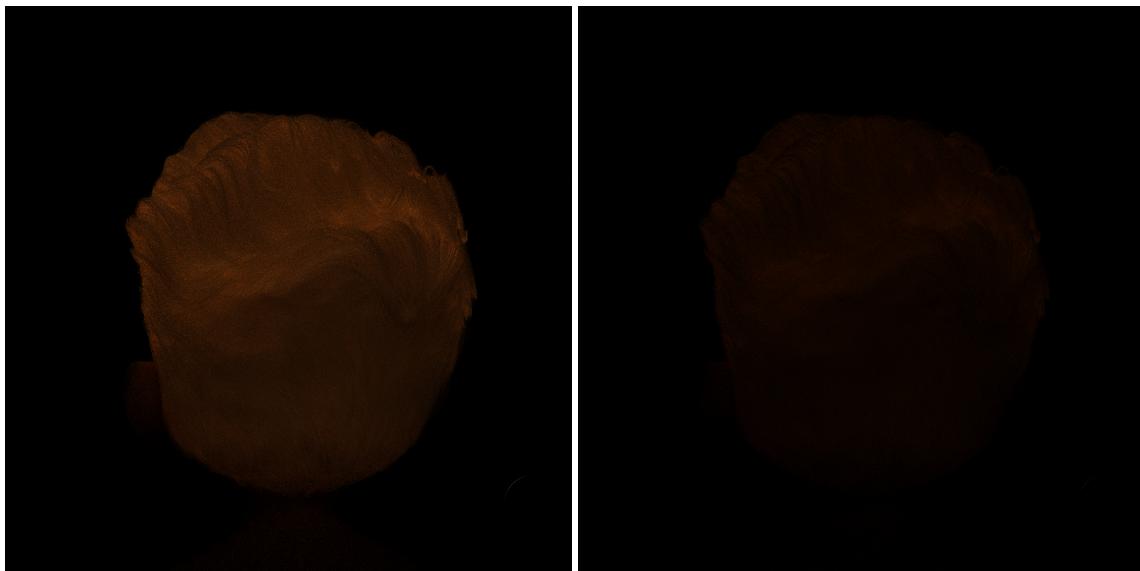


Figure 14: Left: Bounces over 10 without hair multiplying. Right: Bounces over 10 with hair multiplying.

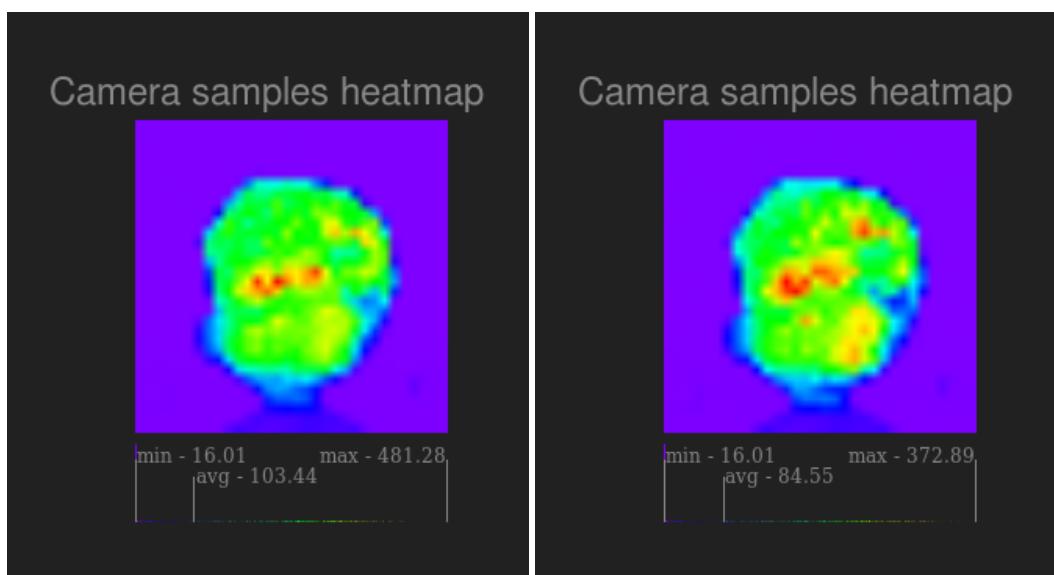


Figure 15: Left: Stats without hair multiplying. Right: Stats with hair multiplying.

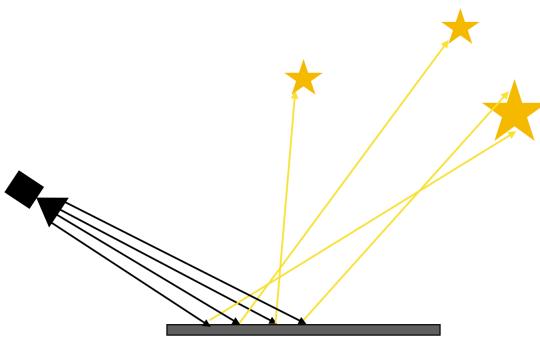


Figure 16: For every hit, we stochastically sample one point on one light.

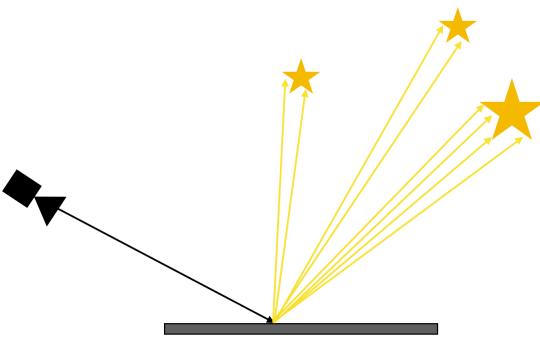


Figure 17: For a single hit, we distribute our lighting budget among all the lights.

7.3 Additive Lights in a Pathtracer

The problem of light sampling in a pathtracer has been extensively covered in the literature. Progressive, interactive rendering being one the great advantages of a pathtracer, the main challenge is to try to preserve one single path from the camera to the light source (Fig. 16), and avoid ray multiplications that could occur when naively sampling all the light sources for every bounce (Fig. 17).

The most basic light picking scheme can be found in Shirley et al. [1996] and consists in randomly picking a light based on an estimated contribution to the currently shaded hit point. This works very well in theory, but unfortunately does not scale as well in practice, since the cost is linear in the number of lights in the scene (in recent production scenes, it is common to find shots with thousands, or even millions of lights in extreme cases). Furthermore, it is not easy to find a good balance between a rough estimate and a computationally intensive precise one.

To remedy this practical scaling problem, every modern production renderer employs some sort of acceleration structure, clustering lights to approximate and accelerate the process of localized light estimation like Estevez and Kulla [2017]. This is still an open problem, as light estimation can be tricky when we incorporate more and more variables, like shadowing, light modifiers (projected textures for example), spectral tinting or even more complex IES profiles. There is no perfect heuristic here, and the techniques are still getting refined both in creating better acceleration structures, but also finding better cluster sampling methods, such as Vévoda et al. [2018].

At Pixar, we have transitioned from a distributed raytracer (where each light had a fixed sample budget), to a pathtracer; and we moved from naive to clustered accelerated stochastic light sampling (see RenderMan [2018] for more details). In practice, this means that instead of the integrator querying the lights directly, we use a proxy class, the lighting services that abstract the number and type of lights in the scene. The integrator can then simply ask "give me N light samples" or "what is the lighting contribu-

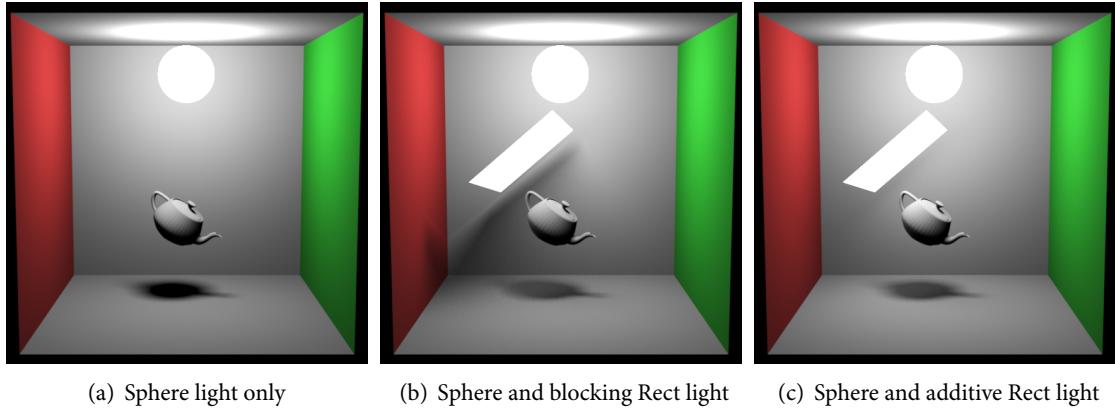


Figure 18: Different modes of additive lights.

tion in this direction”, without worrying about the underlying number or type of lights. In implementing this scheme, we had to overcome one problem. Historically, all the lights used in REYES (Cook et al. [1987]) were some variant of point sources. By their nature they would only produce illumination in the scene and never shadow each other: if you put one point light in front of another, you get the contribution of the two. When we moved to raytracing and arealights, it was not completely clear what the behavior should be. If you put a rect light in front of another one, should it mask it (Fig. 18(b))? Ultimately, we decided that we wanted to keep the additive nature of our lights, to give more control and avoid unpleasant surprises for the users (Fig. 18(a), Fig. 18(c)). If you include a large dim rect light to add some rim, and if it blocks the sun from the environment, this might be unintuitive to the lighter. And if indeed someone wants a light in the front to shadow the light behind it, it is easy enough to do so by adding a separate custom blocker.

From the point of view of light picking, this complicates things though. If the lights are not additive, for a given direction you get one result back: the closest light. If they are additive, for a given direction, you can have multiple light hits (Fig. 21). We could return the sum of all the hits, but that would require tracing multiple shadow rays. In the spirit of keeping one ray per path and not do path splitting, we decide to stochastically pick one of the hits. Depending on the renderer’s API, you might not have a choice anyway (the API might require you to return a single light and not support handing back an array of lights).

So at this point we know how to pick a light, we still have to make a choice on how to support multiple importance sampling (MIS). The easiest way to proceed is for the lighting services to first pick one light, and do MIS as a second step, on that unique light, actually simplifying the problem to one single light (Fig. 19):

1. pick a light among all the lights
 - (a) light sampling: pick a point on that light
 - (b) brdf sampling: pick a brdf direction, test that light

This is correct, but has the disadvantage of heavily assuming that you have a very good and precise light picking heuristic. As we discussed earlier, such heuristics are difficult to achieve in the general case. For a specular object, unless you can precisely pick a light that is in the reflected direction, most of the time, your brdf sample will not return any meaningful contribution effectively increasing variance, leading to the noisy light reflections in Fig. 19.

The solution we adopted was to do MIS over all the lights (Fig. 20):

1. light sampling:
 - (a) pick a light among all the lights
 - (b) pick a point on that light

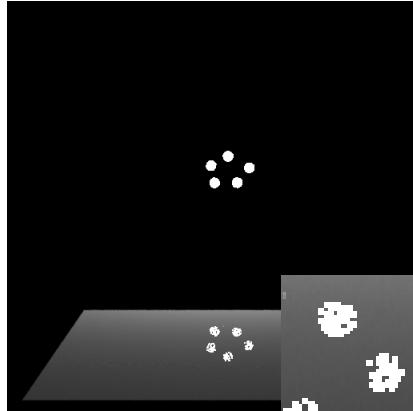


Figure 19: Light reflection due to bsdf sampling is noisy because we first choose which light to test.

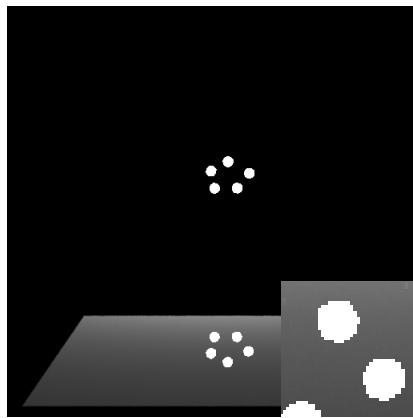


Figure 20: Light reflections are clean because we test all the lights.

2. brdf sampling:

- (a) pick a brdf direction, test all the lights
- (b) pick a hit point among all the hit points

In this approach, we need to be careful when computing the MIS weights. The BSDF sampling now has 2 steps: we need the probability of picking a direction $p(\omega)$; but also the probability of picking one hit between potential overlapping light sources $p(t | \omega)$. We could pick randomly, but since we've already tested the lights, we can directly get the observed contribution at the light hit points. We then build a discrete cumulative density function (CDF), based on the contribution of each light hit point to the currently shaded point (Fig. 21). As a consequence, after picking a point on a light with PDF $p(x)$, in order to estimate the probability density function (PDF) for the BSDF sampling, we need to find all the other hits (if any) for the direction of that light point, on top of the PDF for the BSDF to pick that direction. The BSDF side is straightforward, since we end up with one point on one light, we just need the PDF of picking this point on this particular light.

At the time of this work, the Renderman API did not allow the return of 2 separate PDFs for light evaluation. Direction picking and hit point picking happen in the lighting services, but BSDF sampling happens in the bidirectional scattering distribution function(BSDF). So in theory we need to pass back the hit PDF to the BSDF. In our implementation, we use the same trick we used for the volume light (Villemin and Hery [2013]):

With a balance heuristic (power heuristic would work the same) for the MIS weight, the light sampling estimation is:

$$L_{light} = \frac{p(x)}{p(\omega)p(t | \omega) + p(x)} \frac{f_s() \cdot L}{p(x)} \quad (47)$$

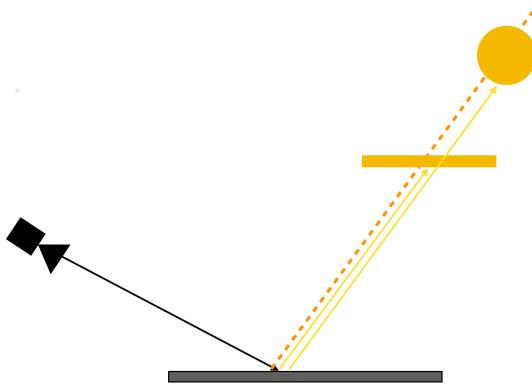


Figure 21: For a single direction, we get 2 hits: one on the rect light, one on the sphere light.

$p(\omega), f_s()$ are the probability of picking a direction and the BSDF value for that direction respectively, and are returned by the BSDF shader.

$p(x), L$ are the probability of picking a point on a light, and the light value at that point respectively, and are returned by the lighting services, along with an additional $p(t | \omega)$ the probability of picking overlapping hit point along the given direction, that needs to be applied on $p(\omega)$.

An optimization we have used for this problem, is to return pre-divided values in our light shader. By making the light shader return $p(x)/p(t | \omega), L/p(t | \omega)$, we can avoid the third extra parameter.

$$\begin{aligned} L_{light} &= \frac{p(x)/p(t | \omega)}{p(\omega) + p(x)/p(t | \omega)} \frac{f_s().(L/p(t | \omega))}{p(x)/p(t | \omega)} \\ &= \frac{p(x)}{p(\omega)p(t | \omega) + p(x)} \frac{f_s().L}{p(x)} \end{aligned} \quad (48)$$

We can apply the same method to the BSDF sampling:

$$\begin{aligned} L_{brdf} &= \frac{p(\omega)}{p(\omega) + p(x)/p(t | \omega)} \frac{f_s().(L/p(t | \omega))}{p(\omega)} \\ &= \frac{p(\omega)p(t | \omega)}{p(\omega)p(t | \omega) + p(x)} \frac{f_s().L}{p(\omega)p(t | \omega)} \end{aligned} \quad (49)$$

7.4 Practical Delta-tracking for Path tracing

As we moved to stochastic arealight sampling for direct lighting, volume sampling also changed (Raab et al. [2008]). For the same reason we described in section 7.3, i.e. the fact that a fixed sample budget per light wouldn't scale well for path tracing, straightforward ray-marching was not fully ideal anymore.

1. choose a marching step size based on some automatic heuristic, or user specification
2. Build a discrete CDF based on that step size
3. sample N samples using that CDF to estimate incident radiance along the ray

There are many problems with this. First, choosing too small a step might result in a very slow CDF construction phase, and could be very memory intensive if tracing multiple rays in parallel. Too large a step, and you might completely miss details in your volume. Unfortunately, being an inherently biased method, you might not even know that you are not computing the right images! With multiple overlapping volumes containing different detail sizes, it might not even be possible to find a unique good step size. All of this led us to have separate integration techniques per volume, (dense volumes, thin volumes, heterogeneous volumes, homogeneous volumes), which made it prohibitively expensive to mix a large number of volumes, or required a costly prepass before rendering to make the data more render friendly.

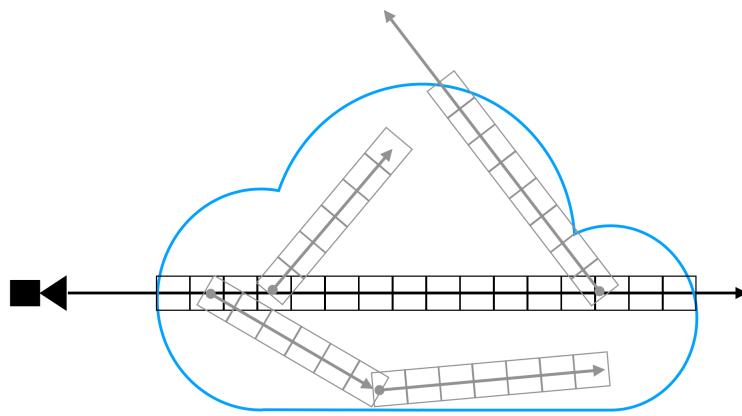


Figure 22: A CDF table is built for every ray. To avoid ray explosion, we only trace a single indirect ray thus using the expensive CDF only once.

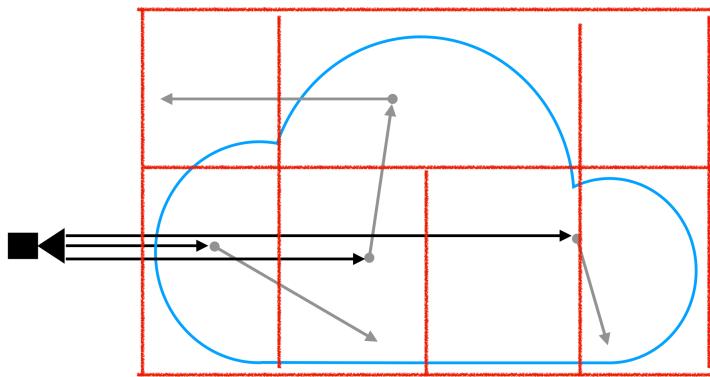


Figure 23: For every ray, we just need to find one scatter point.

Second, even if you do find a good step size, this still remains an expensive operation. We used to be able to amortize that cost by sampling N times (and choosing N large enough to do so), but in a pathtracer, you usually want or need to keep N close to 1. And even if you manage to somehow amortize and optimize the computation for the camera rays, there is no easy way to do so for indirect rays, without leading to an explosion of sampling points (Fig. 22).

So moving to a woodcock tracking sampling method (Woodcock et al. [1965]) was the natural next step to take. It is unbiased, and avoids the ray marching drawbacks. As we don't need to pre-build a CDF table, there is no pre-computation to be done per ray (Fig. 23). However in order to make it practical, we need a good way of estimating density ranges in various regions of space. This is accomplished using aggregate volumes Fong et al. [2017], which also solves the overlapping problem by keeping track of the sum of all the volumes per cells. Additionally by having the min, max and average per cell, we do not need to differentiate between volume types (for example, homogeneous just means $\min = \max$), and we are able to incorporate advanced volumetric integration techniques (Novák et al. [2014]) such as Kutz et al. [2017] decomposition tracking.

With aggregate volumes, we have a fast way to woodcock track in multiple volumes with various types and shapes. One problem remained until recently, woodcock tracking works by putting samples proportional to the density of the volume, which in practice is an important issue for any thin media. At Pixar, we usually render thin atmosphere in separate renders. For those renders, it does make sense to put all the samples in the volume (since all the surfaces are matte out black), but with naive woodcock



Figure 24: Dense areas capture all the samples but thin areas are noisy.

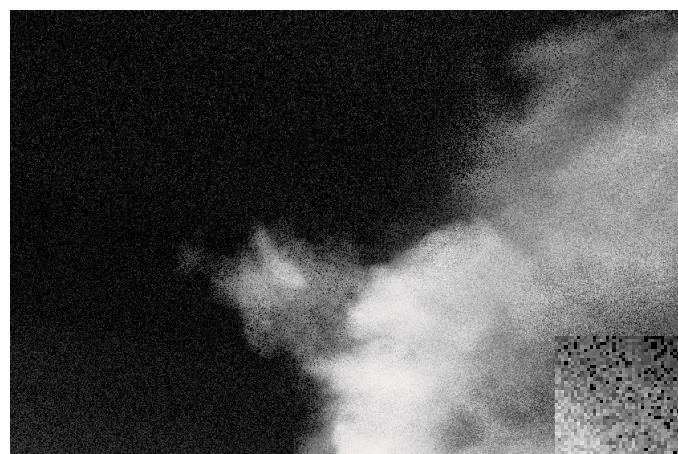


Figure 25: For the same number of samples, thin areas are more converged.

tracking, we would only end up with a percentage of in volume samples proportional to the alpha (for a thin volume with an alpha of 0.1, 90% of your samples were wasted to compute black). Even for render with surfaces and volumes, usually volumes tend to converge slower, so it makes sense to put more samples in them.

The approach we took to solve this consists in 2 changes described in Villemin et al. [2018]:

- change the integration range of woodcock to closely match the optical thickness of the volumes.
This fixes homogeneous volumes
- introduce a gamma correction parameter to homogenize heterogeneous volume

Check results in Fig. 24 and 25.

7.5 Invisibility Cloak

We continue to leverage light path expressions (LPEs) and Lightrig by Schmidt et al. [2013] as a noise control mechanism and to address creative notes. The latest development in that realm is a feature called *invisibility cloak*, which allows the renderer to start ignoring objects after a given LPE is encountered. This capability augments the previous LPE-based operations that simply modulate contributions of those specified paths to the display buffers. For example, consider this test scene with a red cube reflected in a mirror (Fig. 26).

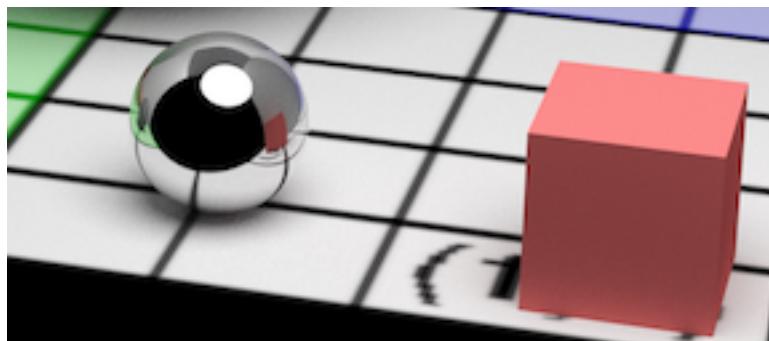


Figure 26: Reference image, notice the red cube reflection in the chrome sphere.



Figure 27: We can kill the contribution of the paths bouncing off the sphere and hitting the cube, but this will still leave black "shadow".

The paths we are interested here all share these events in common:

$C < S' \text{sphere}' > [SD]^* < D' \text{cube}' >$

Before invisibility cloak, in order to remove the reflection of the cube, we would have to clamp the LPE describing that reflection to 0. In many cases that is sufficient, but in this scene it leaves an undesirable black spot in the image (Fig. 27). This is due to the path still terminating at the cube intersection (Fig. 28).

Using invisibility cloak, we can start ignoring the red cube selectively once we've traveled from the camera to the sphere (Fig. 31). This is different and more precise than making the cube invisible to all indirect rays as we are really only affecting its reflection in the designated sphere (Fig. 29, Fig. 30).

7.6 An LPE by any other name?

Pixar relies heavily on LPE-based operations for noise control. One emergent problem with this approach is the inherent complexity of authoring LPEs. Establishing presets can alleviate some of the problem, but we often have to be specific enough that authoring them by hand is often required. Even for simple

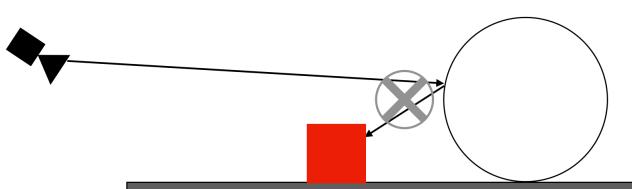


Figure 28: We can kill the contribution of the paths bouncing off the sphere and hitting the cube, but this will still leave black "shadow".

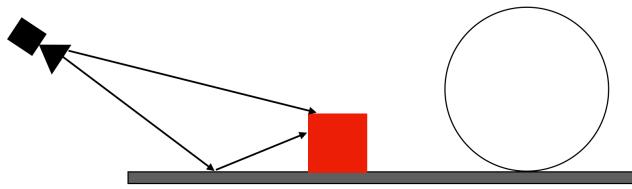


Figure 29: The cloak only affects rays coming from the sphere, so camera rays and other indirect rays can still hit the cube.

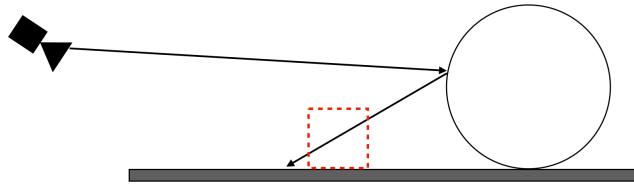


Figure 30: With the cloak, rays bouncing off the sphere won't even see the cube anymore, making it disappear completely from the reflections.

LPEs this can be challenging. Furthermore the only way to error check them is to re-render after you've authored or modified an LPE. Depending on how complex the expression is, users may have to re-render multiple times, which is time consuming and error prone.

The two questions users are trying to answer during this process are:

1. Is this LPE valid?
2. Does this LPE match the correct illumination?

The goal is to be able to answer those two questions without requiring the user to re-render. To check for LPE validity, we can use *Lex* and *Yacc* to discover the structure of an LPE interactively. This allows us to interactively check for superficial errors like errant spaces or missing periods. For example, consider authoring an extremely complicated LPE like this one:

```
C<.[DS]'notHairOrSkin'>*((<.[DS]'allSkin'>+<.[DS]'allHair'>+)|  
(<.[DS]'allHair'>+<.[DS]'all Skin'>+))<L.>
```

An additional space in the lpegroup name 'all Skin' can be time consuming and frustrating to track down. A widget using *Lex* and *Yacc* can provide more immediate feedback (Fig. 32).

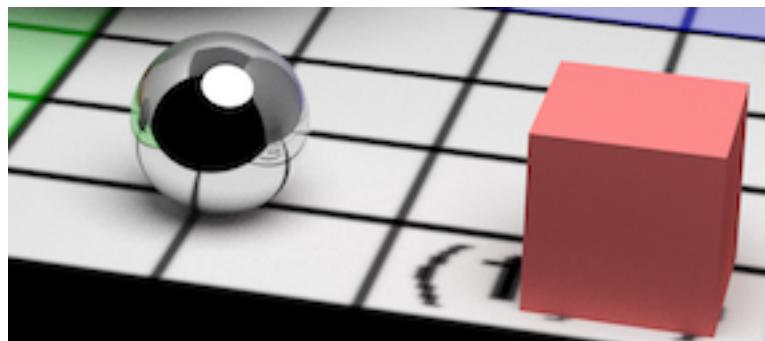


Figure 31: With the cloak, rays bouncing off the sphere won't even see the cube anymore, making it disappear completely from the reflections.

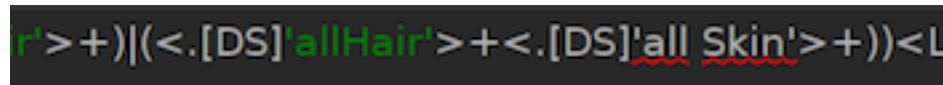


Figure 32: LPE group names cannot contain spaces. Syntax highlighting can flag that error.

Answering "does this LPE match the right illumination?" requires converting the expression to an nondeterministic finite automaton (NFA), which can be used to match against fully formed paths. We've built a standalone library for that purpose, so that users can render a large number of LPE based arbitrary output variables (AOVs) up front, and cycle through paths supplied by Lightrig UI to determine which LPEs match which paths. This reduces the number of re-renders, and can provide further immediate feedback beyond expression compile errors.

7.7 Artisanal Denoising

For *Finding Dory*, *Cars 3*, and *Coco*, Pixar used the Hyperion noise filter to denoise all of the final images (Rousselle et al. [2012], Rousselle et al. [2013], Bitterli et al. [2016]). While it allowed us to render those films with reasonable render times, there was a growing concern about the loss of detail and other artifacts introduced by the noise filter. The worst victims were hair (especially blonde hair), vegetation, fine shading detail, scenes with strong indirect illumination, scenes with strong depth of field, volumes, and many other things that are quite common in our films. And while we got through several films by hand tuning render settings and introducing various pipeline features into denoising to combat those deficiencies, it eventually became clear that we needed a new approach to denoising.

The teams on *Incredibles 2* and *Toy Story 4* began exploring alternative options that would retain more detail. Several off-the-shelf commercial products were vetted, as well as D-Noise, a machine learning based denoiser from Bako et al. [2017]. Early results show that D-Noise retains much more detail, and can produce final quality images from significantly noisier inputs. For those reasons, D-Noise is likely the main denoising approach at Pixar for the foreseeable future. However, the instability introduced by exploring many denoising options has prompted us to move all of the current (and presumably future) methods into Nuke. Standardizing all three approaches as Nuke plugins simplifies our batch rendering pipeline, and has allowed us to transition between approaches easily.

Below is a comparison of the Hyperion (Fig. 34) and D-Noise (Fig. 35) filtered results of the same input, with the original image (Fig. 33) for reference. Note the artifacts in the Hyperion filtering, which are often worse in motion.

7.8 Acknowledgments

We would like to thank the following groups, who contributed one way or another to the work presented in this document:

- the RenderMan team in Seattle for our core rendering framework;
- the Hyperion team at Walt Disney Animation Studios for all the discussions;
- the Shading, Lighting and Renderspeed departments at Pixar on *Incredibles 2* for all the feedback and suggestions.

References

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2017)* 36, 4 (July 2017).



Figure 33: Unfiltered



Figure 34: Hyperion



Figure 35: D-Noise

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Gutián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum (Proceedings of EGSR)* 35, 4 (jun 2016), 107–117. <https://doi.org/10.1111/cgf.12954>

Per H. Christensen. 2015. An Approximate Reflectance Profile for Efficient Subsurface Scattering. In *ACM SIGGRAPH 2015 Talks (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 25, 1 pages. <https://doi.org/10.1145/2775280.2792555>

Robert L. Cook, Loren Carpenter, and Edwin Catmull. 1987. The Reyes Image Rendering Architecture. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. ACM, New York, NY, USA, 95–102. <https://doi.org/10.1145/37401.37414>

Christian Eisenacher, Gregory Nichols, Andrew Selle, and Brent Burley. 2013. Sorted Deferred Shading for Production Path Tracing. In *Eurographics 2013 Papers*.

Alejandro Conty Estevez and Christopher Kulla. 2017. Importance Sampling of Many Lights with Adaptive Tree Splitting. In *ACM SIGGRAPH 2017 Talks (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 33, 2 pages. <https://doi.org/10.1145/3084363.3085028>

Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, and Brian Green. 2017a. Path Tracing in Production - Part 1: Production Renderers. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 13, 39 pages. <https://doi.org/10.1145/3084873.3084904>

Luca Fascione, Johannes Hanika, Rob Pieké, Christophe Hery, Ryusuke Villemin, Thorsten-Walther Schmidt, Christopher Kulla, Daniel Heckenberg, and André Mazzzone. 2017b. Path Tracing in Production - Part 2: Making Movies. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 15, 32 pages. <https://doi.org/10.1145/3084873.3084906>

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering: SIGGRAPH 2017 Course. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 2, 79 pages. <https://doi.org/10.1145/3084873.3084907>

- Eric Heitz and Eugene D'Eon. 2014. Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals. *Computer Graphics Forum* 33, 4 (July 2014), 103–112. <https://doi.org/10.1111/cgf.12417>
- Christophe Hery and Douglas Sutton. 2001. Tutorial On Procedural Primitives. In *Advanced RenderMan, Chapter 4, ACM SIGGRAPH 2001 Courses*. ACM.
- 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*. ACM. <http://graphics.pixar.com/library/BiDir/>
- Christophe Hery, Ryusuke Villemin, and Junyi Ling. 2017. Pixar's Foundation for Materials. In *Physically Based Shading in Theory and Practice, ACM SIGGRAPH 2017 Courses*. <http://graphics.pixar.com/library/PxrMaterialsCourse2017/>
- Jaroslav Křivánek and Eugene d'Eon. 2014. A zero-variance-based sampling scheme for Monte Carlo subsurface scattering. In *ACM SIGGRAPH 2014 Talks*. ACM, 66.
- Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017). <https://doi.org/10.1145/3072959.3073665>
- Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *Proc. SIGGRAPH* 33, 6 (Nov. 2014), 179:1–179:11.
- Leonid Pekelis, Christophe Hery, Ryusuke Villemin, and Junyi Ling. 2015. A Data-Driven Light Scattering Model for Hair. <http://graphics.pixar.com/library/DataDrivenHairScattering/>. (02 2015).
- Matthias Raab, Daniel Seibert, and Alexander Keller. 2008. Unbiased Global Illumination with Participating Media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Alexander Keller, Stefan Heinrich, and Harald Niederreiter (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 591–605.
- RenderMan. 2018. RenderMan's Visuals for Coco. <https://www.fxguide.com/featured/rendermans-visuals-for-coco>. (2018).
- Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-local Means Filtering. *ACM Trans. Graph.* 31, 6, Article 195 (Nov. 2012), 11 pages. <https://doi.org/10.1145/2366145.2366214>
- Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising using Feature and Color Information. 32 (10 2013).
- Thorsten-Walther Schmidt, Jan Novák, Johannes Meng, Anton S. Kaplanyan, Tim Reiner, Derek Nowrouzezahrai, and Carsten Dachsbacher. 2013. Path-space Manipulation of Physically-based Light Transport. *ACM Trans. Graph.* 32, 4, Article 129 (July 2013), 11 pages. <https://doi.org/10.1145/2461912.2461980>
- Peter Shirley, Changyaw Wang, and Kurt Zimmermann. 1996. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics* 15 (1996), 1–36.
- Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford. http://graphics.stanford.edu/papers/veach_thesis/
- Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4 (2018).

Ryusuke Villemin and Christophe Hery. 2013. Practical Illumination from Flames. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (31 December 2013), 142–155. <http://jcgt.org/published/0002/02/10/>

Ryusuke Villemin, Christophe Hery, and Per Christensen. 2016. Importance Resampling for BSSRDF. <https://graphics.pixar.com/library/Resampling/>. (05 2016).

Ryusuke Villemin, Magnus Wrenninge, and Julian Fong. 2018. Efficient Unbiased Rendering of Thin Participating Media. *Journal of Computer Graphics Techniques (JCGT)* (2018).

WetaDigital. 2014. Manuka: Weta Digital's new renderer. <https://www.fxguide.com/featured/manuka-weta-digital-s-new-renderer>. (2014).

Wikipedia. 2018. Level of detail. https://en.wikipedia.org/wiki/Level_of_detail. (2018).

E.R. Woodcock, T. Murphy, P.J. Hemmings, and T.C. Longworth. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems*. Argonne National Laboratory.

Magnus Wrenninge, Ryusuke Villemin, and Christophe Hery. 2017. Path Traced BSSRDF. <https://graphics.pixar.com/library/PathTracedSubsurface/>. (07 2017).

8 Path Tracing the Framestore Way

MANUEL GAMITO, *Framestore*

This section introduces the path tracing pipeline that is used at Framestore with a special emphasis on how volumetric effects are included. For a general introduction to volumetric light transport, the reader is referred to the Introduction in these course notes.

8.1 A Bit of History

Framestore first started using path tracing in 2010 for *GRAVITY*, with release in 2012. The *Arnold* path tracer rendered most of the scenes, using our own surface shaders [Křivánek et al., 2010]. The only shots not rendered with *Arnold* were those that showed the internal ISS fire because no volume rendering was available at the time, and these were rendered instead with *Renderman 16*. This motivated our development of a volume renderer, to be used in future shows, based on Kulla and Fajardo [2012] and that would plug into the *Arnold* atmosphere shader slot. Later, *Arnold* released its own volume rendering capabilities but by then the volume renderer we developed was already well established in our pipeline.

A second development occurred during the production of *JUPITER ASCENDING*, where the *Balem Boardroom*, a cathedral-like environment with light sources hidden in out-of-the-way places, needed to be rendered. Framestore implemented a bidirectional path tracer to that effect with VCM [Georgiev et al., 2012]. In this implementation, *Arnold* was used both as a front-end, sampling the image plane, tracing camera rays and performing pixel filtering, and as a back-end, computing ray-surface intersections. Everything else in the middle was implemented in-house, including a light shader library to perform next event estimation. With the success of this project, Framestore developed more integrators and now maintains the following set:

- A unidirectional path tracer.
- A bidirectional path tracer.
- A utilities integrator.
- A photon mapper.
- A shadow/reflection/refraction catcher.

The latest stage in this sequence of developments is the ongoing work on *Freak*, the new Framestore renderer that relies on *Embree* for handling geometries [Wald et al., 2014]. Our shader technology is renderer-agnostic and all the render specific functionality is hidden behind wrapper classes so that compiling an *Arnold* or a *Freak* version of the shaders becomes a simple task. *Freak* is being tested with lookdev and quality control renders in many of our current shows in production.

8.2 Path Tracing those Volumes

The volume renderer that was first implemented at Framestore as an atmosphere shader had the serious drawback that it would model volume light transport separately from surface light transport. Although it could account for light scattering from surfaces onto volumes, it could not scatter light from volumes onto surfaces. Light scattering in volumes was then reimplemented in our unidirectional path tracer, taking advantage of the latest research results in this area that had since become available.

Any path through the scene can have a mixture of surface vertices, volume vertices and interior vertices (Fig 36). Interior vertices allow for subsurface scattering to be treated as just another case of a volume scattering problem that is rendered with path tracing [Walster, 2017]. Although we still support subsurface shaders based on the traditional dipole model, these are being phased out. In Fig 37, stills from *AVENGERS: INFINITY WAR* show volume scattering (left) and subsurface scattering (right), both rendered with the same algorithms.

We use four types of volume primitives and each one has a different way of interacting with the integration routines. All volumes are bounded and, in the case of homogeneous or procedural volumes that



Figure 36: Path tracing in an environment with surfaces, volumes and surface interiors.

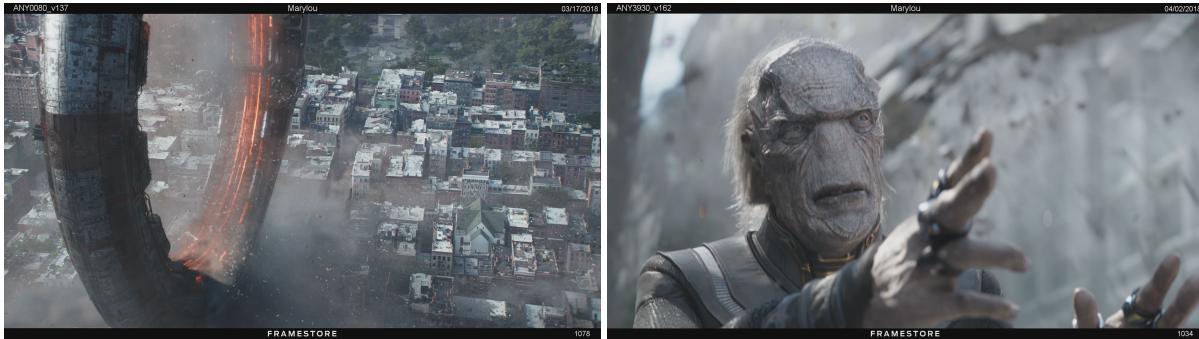


Figure 37: Two stills from AVENGERS: INFINITY WAR, copyright ©2018 Marvel Studios.

could potentially spread over an infinite domain, a boundary is enforced so that lighting from a distant source will not lead to total shadow extinction. The primitives are:

- Voxel datasets.
- Bounded procedural volumes.
- Bounded homogeneous fog.
- Atmospheres.

For voxel datasets, we use our internal *fMote* volume graph toolset [Karefelt and Baas, 2014]. This toolset processes voxels in a two-level hierarchy, where clusters of voxels are grouped together into super-voxels. Each super-voxel stores the local extremal values of all channels defined in the dataset, whether scalar or vector. Atmosphere volumes model a spherical shell with density that decreases exponentially with altitude, to be used in planetary scenes. Arbitrary combinations of these primitives can be set in a scene, allowing for spatial overlap between volumes. In the case of surface interiors, the same volume primitives can be assigned to a surface and will only be activated once a ray crosses into that surface.

8.2.1 Transmittance Estimation

For estimating the volume transmittance between two points in the scene, which is needed for next event estimation, we employ *residual ratio tracking* [Novák et al., 2014], with the loose majorant modifications of Szirmay-Kalos et al. [2017] that allow the sampling extinction within a volume to be an approximation to the true maximum extinction.

Firstly, each volume domain is decomposed into subdomains. For voxel sets, these are naturally the super-voxels. Procedural volumes, use an artificial subdivision of the domain with a supplied number of voxels in each dimension. Atmospheres split the domain into spherical strata that become more densely packed at lower altitudes. The domain subdivisions are then later traversed with a *Digital Differential Analyser* (DDA) algorithm [Amanatides and Woo, 1987]. In the case of voxel spaces, these are first organised into octrees-of-grids so that empty regions can be quickly skipped with a hierarchical DDA algorithm [Revelles et al., 2000].

Secondly, every volume subdomain needs to provide an estimate of the minimum and maximum extinction coefficients within it. For all volume types, these extremal values can be computed accurately, except for procedural volumes where obtaining range information from procedural shaders, although possible, is difficult in practice [Heidrich et al., 1998]. Procedural volumes, therefore, estimate non-exact extremal values by distributing random trial samples within each subdomain. This is done lazily, in a thread-safe manner, so that only those subdomains that are actually traversed by rays need be initialised. Finally, the transmittances for all volumes intersecting a ray are computed separately and their product is returned.

When performing next event estimation for path vertices interior to surfaces, we can optionally employ *manifold next event estimation* [Hanika et al., 2015]. We have a simple extension to the original algorithm, which only applied to finite area lights, to include infinite lights as well (these being distant lights and environment lights). Whereas the original algorithm kept the sample point on the area light fixed during the manifold exploration, the modified algorithm for infinite lights keeps instead the direction toward the light fixed, outside the surface.

8.2.2 Free Distance Sampling

For the sampling of scattering distances within volumes, we have long discarded the traditional ray marching methods with fixed steps that are known to be biased, especially for large step sizes. We use instead unbiased *ray trackers*. These also have the added advantage of being adaptive, in the sense that large regions of low density can be quickly skipped, this provided that accurate estimates of the majorant volume extinction are available in those regions. Ray trackers concentrate samples in areas of higher density, where scattering events are more likely. More details on the (now extensive) literature on ray trackers for path tracing can be found in Novák et al. [2018].

There is a fundamental distinction between *analog* trackers, that model exactly the physics of photon scattering in volumes, and *non-analog* trackers, that do not attempt to simulate exactly this process although, obviously, still remaining unbiased. Non-analog trackers compensate for the deviation from the physical model by maintaining a *tracking weight*, to ensure the algorithm stays correct. The advantages of non-analog trackers for volume path tracing are twofold:

- Approximate estimates of the local extremal values of extinction are allowed but they should be used sparingly to avoid additional variance in the Monte Carlo integral.
- Free distance sampling in chromatic heterogeneous volumes can be handled elegantly, without first needing to choose one particular wavelength of light.

We employ the *spectral decomposition tracker* of Kutz et al. [2017] that decomposes every volume into a piecewise constant control component, that can be sampled analytically, and a residual component. The purpose is to limit the number of volume extinction queries, that are always an expensive part of the algorithm, by relegating them to the residual component only. We have, however, made changes to this algorithm that are worth mentioning here.

The original spectral decomposition algorithm distinguishes between absorption events with probability P_a , where volume self-emission is included, scattering events with probability P_s , where inscattering from some other direction in the scene is included, and null-collision events with probability $P_n = 1 - P_a - P_s$, where the tracker moves forward along the ray. We have found that integrating emission in this way is noisy and that it is preferable to integrate volume emission with a separate algorithm that will be explained in the next section. We therefore set $P_a = 0$ so that the tracker will only be concerned with scattering events.

We have also found that this tracker introduces excessive variance for chromatic homogeneous volumes. We use instead the analytic solution for the free distance $d = \ln(1 - \xi)/\mu_{t,\lambda}$, with the constant extinction coefficient $\mu_{t,\lambda}$ for a chosen wavelength λ and a random number $\xi \in [0, 1]$. Even in the case of heterogeneous volumes, when a particular ray DDA interval of length L crosses a homogeneous subdomain in the volume, the same analytic sampling is applied and the tracking weight is updated with the

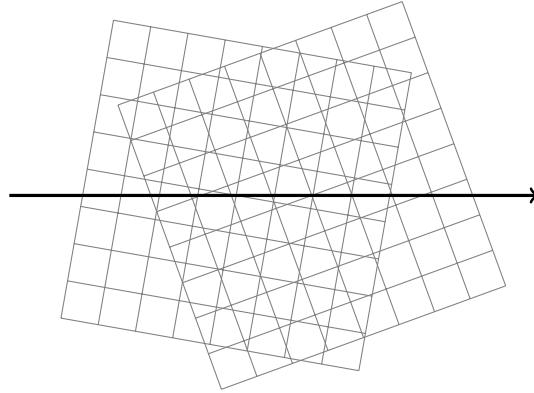


Figure 38: A ray DDA traversal over two overlapping rotated voxel datasets. The resulting DDA is the merge of the DDAs of the two individual volumes.

chromatic value:

$$\begin{cases} \frac{\mu_t e^{-\mu_t d}}{\sum_{\lambda \in \{r,g,b\}} p_\lambda \mu_{t,\lambda} e^{-\mu_{t,\lambda} d}}, & \text{if } d < L \\ \frac{e^{-\mu_t L}}{\sum_{\lambda \in \{r,g,b\}} p_\lambda e^{-\mu_{t,\lambda} L}}, & \text{otherwise} \end{cases} \quad (50)$$

where the wavelength probabilities ($\sum_\lambda p_\lambda = 1$) are the tracking weight at the start of the homogeneous interval times the volume albedo, followed by normalisation. If a scatter event is not found within this homogeneous interval, the tracker continues with the next DDA interval.

The case of multiple overlapping volumes becomes simple with this tracker because we can consider scattering probabilities P_s^i for volumes $i \in \{1, \dots, n\}$ and a corresponding null-collision probability $P_n = 1 - \sum_i P_s^i$. Each probability is further divided into its control and residual components $P_s^i = P_{s,c}^i + P_{s,r}^i$ so that, at every tracking point along the ray, there is a total of $2n + 1$ possible random outcomes for n volumes. For volume i , the control probability is:

$$P_{s,c}^i = \frac{\mu_c^i}{\sum_j \bar{\mu}_j}, \quad (51)$$

where $\mu_c^i = \min_{r,g,b} \{\min_{x \in S} \{\mu_t^i(x)\}\}$ and $\bar{\mu}_i = \max_{r,g,b} \{\max_{x \in S} \{\mu_t^i(x)\}\}$ are the extremal achromatic extinction coefficients of volume i within subdomain S . The residual probability at a ray tracking point x is then:

$$P_{s,r}^i(x) = \left(1 - \sum_j P_{s,c}^j\right) \frac{|\mu_r^i(x)|}{\sum_j |\mu_r^j(x)| + \sum_j |\mu_n(x)|}. \quad (52)$$

The per-volume residual extinction is $\mu_r^i(x) = \mu_t^i(x) - \mu_c^i$ and the null-collision coefficient is $\mu_n(x) = \bar{\mu}_i - \mu_t^i(x)$. Note the presence of the modulus operators that not only convert chromatic values to scalar values but also account for negative coefficients, in cases where the extremal extinctions have not been calculated with precision. Note also that volume extinction evaluations $\mu_t^i(x)$ are only needed for the residual probabilities, while the control probabilities rely on precomputed quantities. If a scattering event is found with probability $P_{s,c}^i$ then all the residual probabilities need not be evaluated⁶.

As the ray DDA progresses across the volume subdomains, the control probabilities are updated, based on the new subdomain into which the ray is entering. As Fig. 38 shows, when multiple volumes overlap, the DDA is the merge of the individual volume DDAs. This merging is performed incrementally

⁶For simplicity, we have not explained here that the residual probabilities are further weighted by the chromatic tracking weight up to that point, to account for the spectral sampling aspect of the tracker. There are several ways this weighting can be done and the reader is referred to the original [Kutz et al., 2017] paper for details.



Figure 39: Still from THOR RAGNAROK, copyright ©2017 Marvel Studios.

- at the current point on the ray, the DDA intervals for all volumes are queried and the one with the shortest length is chosen. The DDA of that volume is then updated to account for the step just taken and the process is repeated. There is a potential shortcoming in this merging in that, for large numbers of volumes, the ray may be broken down into many small intervals but, in practice, we have not found this to be a problem and it is easier than building a global acceleration data structure that contains all volumes [Fong et al., 2017]. Nevertheless, a Bounding Volume Hierarchy is still used to quickly find all volumes that can intersect with a given ray. Once a scattering event is found for some volume j , the scattering phase function $f_s^j(\omega_i \cdot \omega_o)$ of that volume is importance sampled to extend the path along a new direction. Fig. 39 shows a model from THOR RAGNAROK, surrounded by several overlapping voxel datasets.

8.2.3 Emission Integration

For integrating emission, we follow Simon et al. [2017] with some modifications. As the path tracing progresses, the integration of emission up to the next scattering point is computed. Ordinarily, and as equation (28) from Section 5.1 shows, the integration for emission should be done along all the length d of a path segment. This distance d would either be towards the nearest surface or the exit from the volume, whichever is closest. The new emission integration algorithm allows the integration to proceed only up to the next scattering event, while still remaining unbiased, and thereby saving computation time.

Simon et al. [2017] propose that the integration of emission be done by performing a ray DDA over the individual voxels of a dataset and accumulating their emissions along the way. This has the merit that the integration will be computed accurately, if we assume that emission is piecewise constant within each voxel, but it can be slow to march with a DDA over high resolution voxel grids. Furthermore, this approach does not extend to the other volume primitives. We perform instead a Monte Carlo integration of the emission for some number N of samples. Following the notation from Section 5.1:

$$L_e = \int_0^d T(\mathbf{x}, \mathbf{z}) \mu_e(\mathbf{z}) L_e(\mathbf{z}) dt \approx \frac{1}{N} \sum_{i=1}^N \frac{T(\mathbf{x}, \mathbf{z}_i) \mu_e(\mathbf{z}_i) L_e(\mathbf{z}_i)}{pdf_i}, \quad (53)$$

where $\mathbf{z} = \mathbf{x} + t\mathbf{w}$ is a point along the segment and $T(\mathbf{x}, \mathbf{z})$ is the transmittance (25) up to \mathbf{z} . During the free distance sampling of Section 8.2.2, the sequence of tracking steps is saved into a discrete probability distribution. Tracking step i , with length Δt_i , is assigned a probability:

$$p_i \propto |T(\mathbf{x}, \mathbf{z}_i) \langle \frac{\mu_e}{\mu_t} L_e \rangle_j (1 - e^{-\langle \mu_t \rangle_j \Delta t_i})|. \quad (54)$$



Figure 40: Still from DARKEST HOUR, copyright ©2017 Universal Pictures.

The operator $\langle \cdot \rangle_j$ represents the estimated average value of a quantity within the volume subdomain j over which step i is taken. These averages are precomputed similarly to the minimum and maximum extinction coefficients, as mentioned in Section 8.2.1. The transmittance $T(\mathbf{x}, \mathbf{z}_i)$ is taken at the start of each step, with residual ratio tracking. Once a tracking step is selected with probability p_i , a distance t_i is importance sampled within it, according to the achromatic transmittance term $e^{-|\langle \mu_t \rangle_j|t_i}$. Writing the scalar $\mu_j = |\langle \mu_t \rangle_j|$, for short, the probability density of a Monte Carlo emission sample for equation (53) is then:

$$pdf_i = p_i \frac{\mu_j e^{-\mu_j t_i}}{1 - e^{-\mu_j \Delta t_i}}. \quad (55)$$

Emission integration for overlapping volumes is handled straightforwardly by taking the total emission $\sum_i (\mu_e L_e)^i$ and total extinction $\sum_i \mu_t^i$ over all volumes and making the appropriate changes in the above equations. For efficiency, the number N of Monte Carlo emission samples for each new path segment is gradually decreased as the path grows, proportionally to the path transmittance.

Emission integration can also be used for *volume lights*, as explained in the paper by Simon et al. [2017]. A point inside the volume is first sampled by randomly descending an importance octree. This point then determines a sampling direction over which emission is integrated. This scheme allows for MIS where the direction is chosen as before and also sampled from a surface BRDF or a volume phase function, with the two results weighted together. Fig. 40 shows an example of several explosions from DARKEST HOUR rendered in this way. The explosions are sampled as volume lights to account for their illumination on surrounding surfaces and volumes. The explosions themselves are rendered with emission integration.

8.2.4 Direct Volume Illumination

The direct volume illumination accounts for the integration of illumination coming straight from light sources and without any intervening scattering events. For individual path vertices, this is also known as *next event estimation*, but here we take a broader meaning and consider the solution of the integral along a path segment for the direct illumination L_d^i from light i :

$$L_d^i = \int_0^d T(\mathbf{x}, \mathbf{z}) \sum_j \mu_s^j(\mathbf{z}) \int_{\Omega_i} f_s(\omega \cdot \omega_i) L_i(\mathbf{z}, \omega_i) d\omega_i dt. \quad (56)$$

The light subtends a solid angle Ω_i at point \mathbf{z} on the ray and the integration is performed over that solid angle. When Ω_i is not easily parameterisable, a change of variables is usually done so that the integral is

performed over the area of the light instead. See Gamito [2016] for a discussion of these issues in the case of area lights with circular symmetry. The integral (56) already accounts for the possibility of overlapping volumes by summing over all volumes j for which the scattering coefficient $\mu_s^j(\mathbf{z}) > 0$.

We use MIS to combine together two different sampling strategies along the path segment, following ideas initially proposed by Kulla and Fajardo [2012]:

- Equiangular sampling relative to a reference point on the light.
- Transmittance sampling, using transmittance estimates saved during ray tracking.

Equiangular sampling is more efficient for transparent volumes and concentrates ray samples closer to the light. For point or spot lights, the reference point is the light itself. For area lights, each path segment randomly chooses a reference point over the area of the light. Transmittance sampling is best for opaque volumes and concentrates sampling points at the start of the segment, where they are more likely to make a contribution. Similar to how emission was importance sampled in the previous section, a discrete probability distribution is built during ray tracking that stores as probabilities the decreasing transmittance contributions of successive ray steps, times their average scattering coefficient. In the case of infinite light sources, only transmittance sampling is used since equiangular sampling does not provide any advantage and, if used, would have distributed sampling points along the segment with equal probability.

For every point \mathbf{z} on the path segment, sampled with either the equiangular or transmittance strategies, a second level of MIS is performed for sampling the light. If we consider $L_d^i = \int T(\mathbf{x}, \mathbf{z}) L_d^i(\mathbf{z}) dt$, where $L_d^i(\mathbf{z})$ is the illumination contribution at \mathbf{z} , then the latter can be sampled directly from the light:

$$L_d^i(\mathbf{z}) \approx \frac{L_i(\mathbf{z}, \omega_i)}{pdf_i(\omega_i)} \sum_j \mu_s^j(\mathbf{z}) f_s^j(\omega \cdot \omega_i), \quad (57)$$

after a direction ω_i towards the light has been chosen, with solid angle probability density $pdf_i(\omega_i)$ and incoming radiance $L_i(\mathbf{z}, \omega_i)$. Another option is to importance sample a direction from one of the volume phase functions. A volume is chosen with probability $p_j \propto |\mu_s^j(\mathbf{z})|$ and a direction ω_j is sampled from its phase function. The approximation to $L_d^i(\mathbf{z})$ is then:

$$L_d^i(\mathbf{z}) \approx L_i(\mathbf{z}, \omega_j) \frac{\sum_k \mu_s^k(\mathbf{z}) f_s^k(\omega \cdot \omega_j)}{\sum_k p_k pdf_k(\omega_j)}. \quad (58)$$

Finally, estimates (57) and (58) are combined with MIS. We use this two-level MIS approach for direct lighting of path segments in the atmosphere only. Path vertices interior to surfaces use next event estimation (or manifold next event estimation) without any integration along the segment lengths. Since surface interiors are usually very opaque, this allows us to use simpler lighting code for interiors without a noticeable increase in variance. One limitation of ray trackers is that they don't compute the scattering probability density at a path vertex but this is not a problem in this context because the probability density cancels out with the path transmittance in the Monte Carlo integral, leaving only the tracking weight times the known albedo term.

An alternative formulation for the computation of light contributions, that could be considered in the future, involves the *joint importance sampling* of several scattering events, taken together towards a lights source [Georgiev et al., 2013]. The joint sampling of up to 2 scattering events is possible, given a path segment direction and the light. For volumes with isotropic phase functions, this joint sampling can be performed analytically but, for other phase functions, a tabulation of angular probability distributions is required. Some care should be taken because we enforce a maximum path length L in our integrators. The joint sampling of 2 additional scattering events would only be possible up to length $L - 2$ in a path. For a length $L - 1$, the simpler joint sampling that considers only one extra scattering would be required, while for length L only equiangular sampling would be used. Joint importance sampling can be regarded as a generalisation of equiangular sampling with additional phase function directional sampling, leading to lower variance, but it does not consider transmittance attenuation along paths. For dense heterogeneous volumes, some form of MIS with transmittance sampling would be needed.

8.2.5 Many Lights Sampling

So far, illumination from a single light source has been considered and there remains the matter of selecting from a set of possibly many light sources in a scene. We assign a sampling probability to each light that reflects its importance along the whole extent of a path segment. This allows a light to be sampled first, before equiangular sampling is performed along the segment, relative to that light. For a light i , a weighting function $w_i(t)$ is devised, for distances t along the segment. The particular shape of this function depends on the geometry of the light. In the case of a point light at \mathbf{p} , it is simply $w_i(t) = \Phi_i / \|\mathbf{z}(t) - \mathbf{p}\|^2$, where Φ_i is the illuminant power of the light. The sampling probability for light i is then:

$$p_i \propto \int_0^d w_i(t) dt. \quad (59)$$

This integral can be solved analytically for a point light but, generally, a numerical quadrature method is required to compute (59). We use publicly available quadrature software and set it to very low precision (with a relative error of 0.01) since it needs to be fast and we only require a rough approximation to the light sampling weights.

A low light threshold is used that limits the illumination envelope of a light to a finite shape. For example, the envelope of a spot light is a cone with a height that is determined by the light decay down to the supplied threshold. Only those lights whose illumination envelopes intersect the path segment need be considered, at the cost of a small amount of bias being introduced due to the truncation of the light illumination fields.

8.3 Things to do when we get back from Siggraph

There are three aspects that we would like to improve in our volumetric path tracer and these are related to increased efficiency:

- Early cutoffs for the direct illumination integral.
- Volume irradiance caching.
- Many lights sampling with better asymptotic complexity.

At present, the integration of illumination is done along the whole extent of a path segment, until it leaves the volume or hits a surface. The spectral decomposition tracking algorithm is used until a scattering point is found and a switch to residual ratio tracking is done after that, in order to complete the transmittance probability table for transmittance sampling across the whole segment. It would be more efficient to skip this last tracking and integrate illumination only up to the scattering point but some bias compensation needs to be performed to account for the illumination past the scattering point that has been ignored. The emission integration algorithm of Simon et al. [2017] could be repackaged for this purpose since there is nothing in that algorithm that is specific to emission and it could be used to integrate any other quantity $\int T(\mathbf{x}, \mathbf{z})L(\mathbf{z})dt$, including direct illumination, along a ray.

Further speedups could be possible by caching volume irradiance for later reuse. This is an idea that has been explored before [Jarosz et al., 2008, Marco et al., 2018]. These algorithms compute radiance gradients and Hessians, which in turn require volume differentials that are difficult to obtain in general. For that reason, homogeneous volumes are more often considered. The algorithms also suffer from a lack of determinism since the distribution of cache records is dependent on the threading pattern of the render threads. Nevertheless, caching the volume irradiance field is an enticing prospect that should improve render times and reduce noise. The caching of irradiance, rather than radiance, should be acceptable if we restrict the use of the cache beyond two or more scattering events, after which the phase function can be fairly considered to be isotropic.

Finally, we would like to improve the performance of our volume many lights algorithm. It currently has a $O(N)$ complexity with N lights, which can become a bottleneck for scenes with thousands of lights that are not so uncommon in production. Reducing the complexity to $O(\log N)$ would be a desired goal by organising the lights into a hierarchy, similar to Estevez and Kulla [2017]. The heuristics used for

traversing this tree top-down would have to be adapted to account for the influence of light clusters over path segments of any length, not just over path vertices.

References

- John Amanatides and Andrew Woo. 1987. A fast voxel traversal algorithm for ray tracing. *Eurographics* 87, 3 (1987), 3–10.
- Alejandro Conty Estevez and Christopher Kulla. 2017. Importance sampling of many lights with adaptive tree splitting. In *ACM SIGGRAPH 2017 Talks*. ACM, 33.
- Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering. In *ACM SIGGRAPH 2017 Course Notes*. ACM, 96.
- Manuel N Gamito. 2016. Solid angle sampling of disk and cylinder lights. *Computer Graphics Forum* 35, 4 (2016), 25–36.
- Iliyan Georgiev, Jaroslav Křivánek, Tomas Davidovic, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6 (2012), 192–1.
- Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. 2013. Joint importance sampling of low-order volumetric scattering. *ACM Trans. Graph.* 32, 6 (2013), 164–1.
- Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold next event estimation. *Computer Graphics Forum* 34, 4 (2015), 87–97.
- Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. 1998. Sampling procedural shaders using affine arithmetic. *ACM Transactions on Graphics (TOG)* 17, 3 (1998), 158–176.
- Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. 2008. Radiance caching for participating media. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 7.
- Per Karelfelt and Matthias Baas. 2014. Gravity: volumetrics in space. In *ACM SIGGRAPH 2014 Talks*. ACM, 61.
- Jaroslav Křivánek, Marcos Fajardo, Per H Christensen, Eric Tabellion, Michael Bunnell, David Larsson, Anton Kaplanyan, B Levy, and RH Zhang. 2010. Global illumination across industries. *SIGGRAPH Courses* 6 (2010).
- Christopher Kulla and Marcos Fajardo. 2012. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum* 31, 4 (2012), 1519–1528.
- Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 111.
- Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. 2018. Second-Order Occlusion-Aware Volumetric Radiance Caching. *ACM Transactions on Graphics (Presented at SIGGRAPH)* 37, 2 (April 2018). <https://doi.org/10.1145/3185225>
- Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Eurographics State of the Art Reports)* 37, 2 (2018), 1–26. to appear.
- Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.* 33, 6 (2014), 179–1.

Jorge Revelles, Carlos Ureña, and Miguel Lastra. 2000. An efficient parametric algorithm for octree traversal. In *The 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*.

Florian Simon, Johannes Hanika, Tobias Zirr, and Carsten Dachsbacher. 2017. Line Integration for Rendering Heterogeneous Emissive Volumes. *Computer Graphics Forum* 36, 4 (2017), 101–110.

László Szirmay-Kalos, Iliyan Georgiev, Milán Magdics, Balázs Molnár, and Dávid Légrády. 2017. Unbiased Light Transport Estimators for Inhomogeneous Participating Media. *Computer Graphics Forum* 36, 2 (2017), 9–19.

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. 2014. Embree: a kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 143.

Nathan Walster. 2017. Volumetric Skin and Fabric Shading at Framestore. In *Physically Based Shading in Theory and Practice*. ACM, 16. Siggraph 2017 Course Notes.

9 From Bricks to Bunnies - Adapting a Raytracer to New Requirements

LUKE EMROSE, *Animal Logic*



Figure 41: *Peter Rabbit* required *Animal Logic* to move forward development of its proprietary raytracer *Glimpse* to achieve a photorealistic live-action CG hybrid look

9.1 The Road Travelled so Far

Glimpse was developed at Animal Logic to assist interactive lighting placement during *Walking with Dinosaurs 3D* by then Lighting Supervisor Max Liani. As discussed by Daniel Heckenberg last year in his talk *A Change of Path at Animal Logic* (Fascione et al. [2017]), its continued use and development helped us to successfully complete *The LEGO Movie*, *Allegiant*, *The LEGO Batman Movie* (further discussed in Heckenberg et al. [2017b]), *The LEGO Ninjago Movie*, some fractal VFX shots on *Guardians of the Galaxy 2*, and most recently, *Peter Rabbit*. The whole-hearted adoption of our "render everywhere" philosophy has also seen its use throughout the *Animal Logic* pipeline from early modelling and layout renders to final lighting shots.

Until one particular feline in *The LEGO Ninjago Movie*, and the furry-character heavy *Peter Rabbit*, our *Glimpse* development had played directly to the strengths of the *LEGO* style. Judicious use of hard-surface subdivision mesh rendering, multiple-level hierarchical instancing, a small subset of highly specialized shaders with common parameters, and a limited need for live-action integration helped us to develop our core rendering toolkit in a highly specialized fashion. Everything changed with *Peter Rabbit* and we had to add a significant new set of features to *Glimpse* within a compressed time-frame to tackle fur, detailed clothing and extensive live-action integration.

The cat character in *The LEGO Ninjago Movie* (2017) was the first time since *Legend of the Guardians: The Owls of Ga'Hoole* (2010) or *Walking with Dinosaurs 3D* (2013) (both *PRMan* shows using predominantly shadow-map and scan-line or hybrid raytracing rendering techniques) there was a need for us to render curves en-masse. A camera-facing curve primitive and a physically-based fur shader were added to *Glimpse* in a very short time-frame to render the Ninjago cat character and subsequently improved during *Peter Rabbit*.

9.2 Hair Geometry and Level of Detail

The intersectable hair primitive we used for *The LEGO Ninjago Movie* and *Peter Rabbit* (Figure. 41 and Figure. 42) was a simple camera-facing cubic bezier curve segment with varying radius per control-vertex.



Figure 42: Careful storage of curve data in memory and judicious use of level of detail allowed rendering of many detailed characters from a single pass

Each hair was composed of multiple piecewise cubic Bezier curves intersected as linear segments using a similar method to Nakamaru and Ohno [2002] with acceleration for vector processors. Our implementation was initially similar to the method used in an early *Embree* (Wald et al. [2014]) version, but optimized to ensure shading continuity and artefact-free normals at all distances from camera. We quickly found memory usage to be an issue, which was improved by using Catmull-Rom splines in memory, transforming them into cubic Bezier curves and dicing them on-the-fly during ray-intersection. Our Catmull-Rom to cubic Bezier trick reduced memory by a factor of approximately 4x from our first naive implementation. Further memory optimizations were quickly required for shots with a large numbers of characters on screen simultaneously. This was achieved by using stochastic density culling for the hair cache data. Root positions were generated using our in-house node evaluation package *ALF* which distributed the hair across the surface mesh using an efficient blue-noise-like distribution. Due to the nature of the shots on *Peter Rabbit*, there was no need to animate fur density levels across single shots. This meant that artists could select the lowest appropriate level of detail for each character, tuning memory usage against changes in look as appropriate.

9.3 Developing the Peter Rabbit Fur Shader

Chasing a physically-based realistic look for the fur in *Peter Rabbit* meant representing the physical details of real fur in our shader. Real fur consists of cuticle scales covering a coloured partially transparent refractive medium called the cortex, which in turn encloses a second interior light-scattering cylindrical structure called the medulla. Further details of the structure of real fur are covered in Yan et al. [2015]. Shading models for hair and fur are typically modelled on cylinders, broken down into two separable orthogonal representations: one for the longitudinal response ("along" the hair tangent direction), and the azimuthal response ("across" the hair tangent direction). Probability density functions are typically computed separately for these two directions and multiplied together to produce the final shading response for raytracing.

We had implemented the seminal hair scattering technique from Marschner et al. [2003] on previous shows with *PRMan*, but found the specular-only results lacking realism for furry characters where multiple-scattering of the medulla was required to make the hair look "softer" and less glass-like (see Figure. 43). Yan et al. [2015] was of particular interest, as it presented a solution for representing this medulla scattering. The last paper to be published during the research phase of our investigation was Chiang et al. [2015] which was extremely useful, as it provided a thorough description of a project-tested



Figure 43: By implementing a physically based double cylinder fur shading model with targeted optimizations, realistic high-albedo multi-scattering fur could be rendered in a diverse set of lighting conditions

implementation, and rare insight about what worked and what didn't for *Zootopia*. Matt Pharr and Cem Yuksel even published a working implementation of this model (Pharr and Yuksel [2016]) which we used as a starting point. A few issues quickly became evident, however:

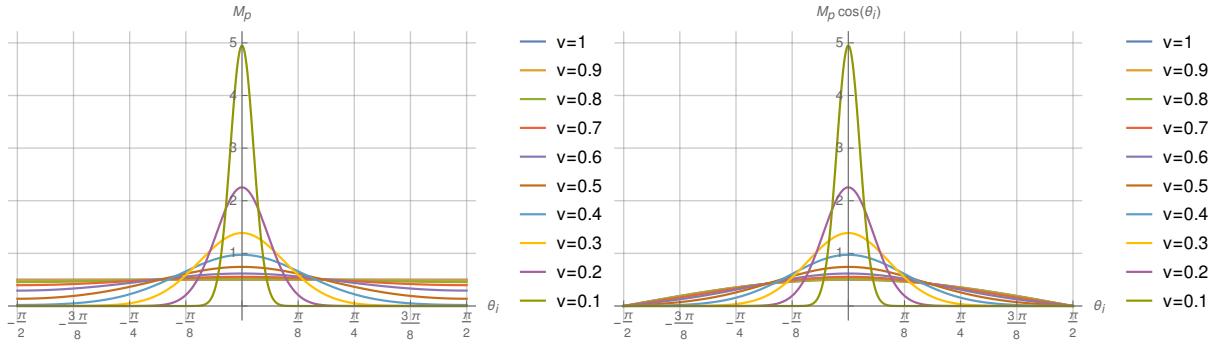
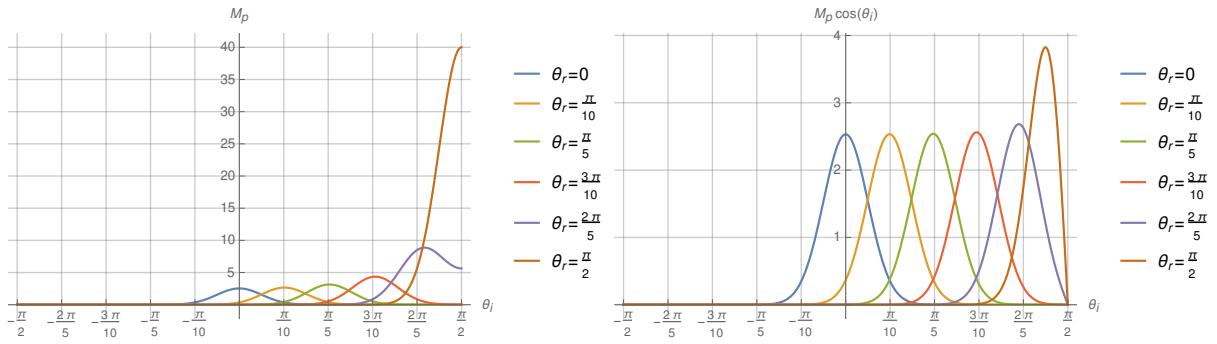
- The fur medulla scattering functions of Yan et al. [2015] were complicated and tabulated, yet could be sufficiently approximated (for our purposes) using a lambertian lobe in the longitudinal direction and a constant value in the azimuthal direction. This also provided a visually-pleasing soft look for our characters and simplified the inverse albedo calculations, which seem like they might otherwise have been intractable.
- The longitudinal lobe scattering function of d'Eon et al. [2011] and d'Eon et al. [2013] (Figure. 44 and Figure. 45) is numerically sensitive and computationally expensive in its original form (though it looks gorgeous!). Extra numerical insights from d'Eon [2013] provided a workable initial solution, but we came up with a few further contributions which will be discussed here.
- The suggestion in Chiang et al. [2015] of the logistic function for azimuthal scattering works well, but the distribution isn't actually circular, and hence for higher roughness values is discontinuous. This is more of an observation than an actual issue in practice, but future implementations might benefit from a circular distribution such as the Von Mises-Fisher distribution (Sir Ronald Fisher [1953]), or even the 2D Henyey-Greenstein (HG) distribution (Heino et al. [2003] and Davis [2006]). However, the 2D HG function doesn't appear to be wide enough around its central lobe to match measured data in a fur-shading context.
- Applying our first fur shader to a character in a scene immediately slowed down renders by 24% compared to a typical microfacet model. This level of shader complexity wasn't something we'd experienced before, and creative solutions were required to improve runtime performance.

9.3.1 Improving Lobe Evaluation

A substantial portion of render time for our prototype fur shader was spent in longitudinal lobe evaluation. The exact form of the longitudinal lobe derived in d'Eon et al. [2013] is:

$$M_p(\nu, \theta_i, \theta_r) = \frac{\cosh\left(\frac{1}{\nu}\right)}{2\nu} e^{\left(\frac{\sin \theta_i \sin \theta_r}{\nu}\right)} I_0\left(\frac{\cos \theta_i \cos \theta_r}{\nu}\right) \quad (60)$$

A quick dissection of why this expression is so difficult to evaluate using floating point arithmetic is useful. Firstly, division by the roughness value ν means small roughness values produce very high $\frac{1}{\nu}$ values. The

Figure 44: Fur longitudinal lobe M_p and $M_p \cos \theta_i$ for various roughness values v Figure 45: Fur longitudinal lobe M_p for $v = 0.025$ with $\theta_r = [0, 0.1\pi, 0.2\pi, 0.3\pi, 0.4\pi, 0.5\pi]$

modified bessel function of the first kind $I_0(x)$ grows in magnitude quickly for moderate values of x (see Figure. 46). For instance, for $x > 92$, $I_0(x) > 3.402823 \cdot 10^{38}$ and hence greater in magnitude than the largest representable floating point number. Since $|\cos(x)\sin(x)| \leq 1$, the smallest roughness value before the bessel expression in Equation. 60 exceeds floating point range is $\frac{1}{v} \approx 92$ which occurs for approximately $v < 0.01$. The suggested technique for numerical evaluation in d'Eon [2013] is to split between two different expressions depending on the roughness value. For roughness $v \geq 0.1$ d'Eon suggests Equation. 60 and for roughness $v < 0.1$ an alternate approximate expression is suggested:

$$M_p(v, \theta_i, \theta_r) \approx \exp \left(\ln \left(I_0 \left(\frac{\cos \theta_i \cos \theta_r}{v} \right) \right) + \left(\frac{\sin \theta_i \sin \theta_r}{v} \right) - \frac{1}{v} + \ln(2) + \ln \left(\frac{1}{2v} \right) \right) \quad (61)$$

Ideally we wouldn't need to special case different roughness values if we came up with an expression that played to the strengths of the floating point representation for both small and high roughness values. The added advantage of not requiring separate code-paths at run time is branchless code, which can be higher performance due to removing the possibility of branch prediction cache misses. Repeated numerical tests resulted in the following form, used for the *Peter Rabbit* fur shader, which handles a large range of roughness values with high precision in floating point. It is also an exact expression, being mathematically identical to Equation. 60, but with better behaviour when computed with floating point range. We observed a wide numerical stability range of roughness values for $0.00001 \leq v \leq 10000$:

$$M_p(v, \theta_i, \theta_r) = \exp \left(\ln \left(I_0 \left(\frac{\cos \theta_i \cos \theta_r}{v} \right) \right) + \left(\frac{\sin \theta_i \sin \theta_r}{v} \right) - \frac{1}{v} + \ln \left(\frac{1}{v} \right) - \ln \left(1 - e^{-\left(\frac{2}{v} \right)} \right) \right) \quad (62)$$

We also found an improved method for evaluating $\ln(I_0(x))$ (Figure. 46) based on the Blair and Edwards

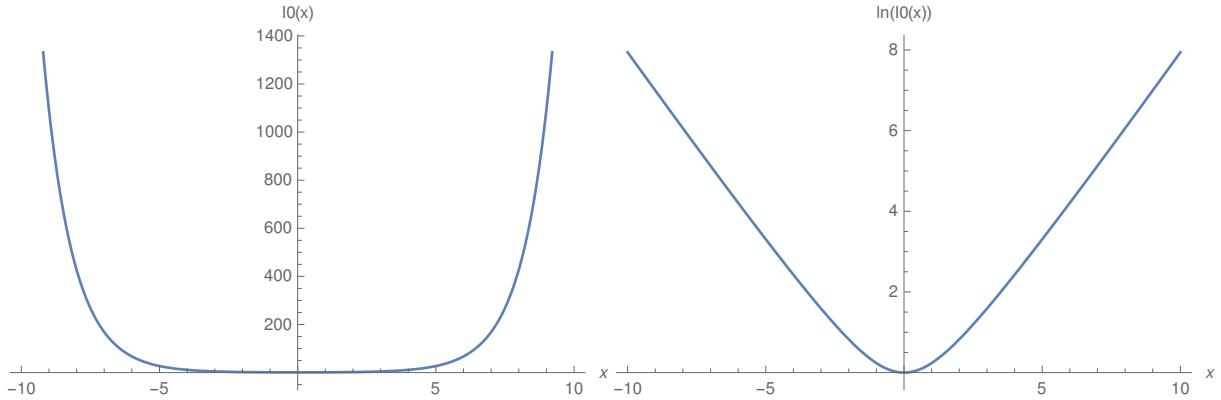


Figure 46: Modified bessel function of the first kind $I_0(x)$ and its natural logarithm $\ln(I_0(x))$

[1974] method for evaluating $I_0(x)$. There are multiple ranges over which their method operates:

$$\begin{aligned} |x| \leq 15 : I_0(x) &\approx \frac{P_n\left(\left[\frac{x}{2}\right]^2\right)}{Q_m\left(\left[\frac{x}{2}\right]^2\right)} \\ |x| > 15 : I_0(x) &\approx \frac{e^{|x|}}{\sqrt{|x|}} \frac{P_r\left(\frac{1}{|x|}\right)}{Q_s\left(\frac{1}{|x|}\right)} \end{aligned} \quad (63)$$

We can simplify these equations when taking their natural logarithm. This also reduces the number of floating point computations required compared to naively taking the natural logarithm of the original expression, by dropping the exponential and square roots using standard identities of logarithms:

$$\begin{aligned} |x| \leq 15 : \ln(I_0(x)) &\approx \ln\left(\frac{P_n\left(\left[\frac{x}{2}\right]^2\right)}{Q_m\left(\left[\frac{x}{2}\right]^2\right)}\right) \\ |x| > 15 : \ln(I_0(x)) &\approx |x| + 0.5 \ln\left(\frac{1}{|x|} \left(\frac{P_r\left(\frac{1}{|x|}\right)}{Q_s\left(\frac{1}{|x|}\right)}\right)^2\right) \end{aligned} \quad (64)$$

It is then straight-forward to write a branch-free implementation of Equation. 64, which evaluates all polynomials separately, selects the appropriate one (squaring if necessary and multiplying by $\frac{1}{|x|}$), then computes the logarithm and, if appropriate, multiplies by 0.5 and adds $|x|$. The result gives us a fast and accurate method of computing $\ln(I_0(x))$ which we can use to evaluate Equation. 62. As we operate in the logarithmic domain, accuracy is extremely good out to almost the full floating point range, since for $x > 15$ our expression is the absolute value of x , plus an expression which decreases extremely slowly. If we were not operating in the logarithmic domain for our modified bessel function calculations, we would quickly run out of floating point range as previously discussed.

To sample M_p we use the high-precision technique given in Jakob [2012]. Techniques for evaluation and sampling of the logistic distribution, used for the azimuthal lobe, are well covered in Pharr and Yuksel [2016]. Further performance optimizations were found using ideas from the *fastapprox* math library by Paul Mineiro to accelerate typically computationally expensive functions like x^y , e^x and $\ln(x)$.

9.3.2 Vectorization of Computation

With our longitudinal and azimuthal scattering lobes performing well and providing plenty of numerical accuracy for low computational cost, the next performance issue was the for-loops required to run over all the differing fur shader light paths. As detailed in Yan et al. [2015], R, TT, TRT, TRRT, TrT, TrRrT, TrRrRrT are for the purely specular cases, and TttT, TtrtT for the cases including medulla multiple-scattering (see Figure. 47). We grouped these cases into 3 separate evaluation modes. Mode 0 occurs

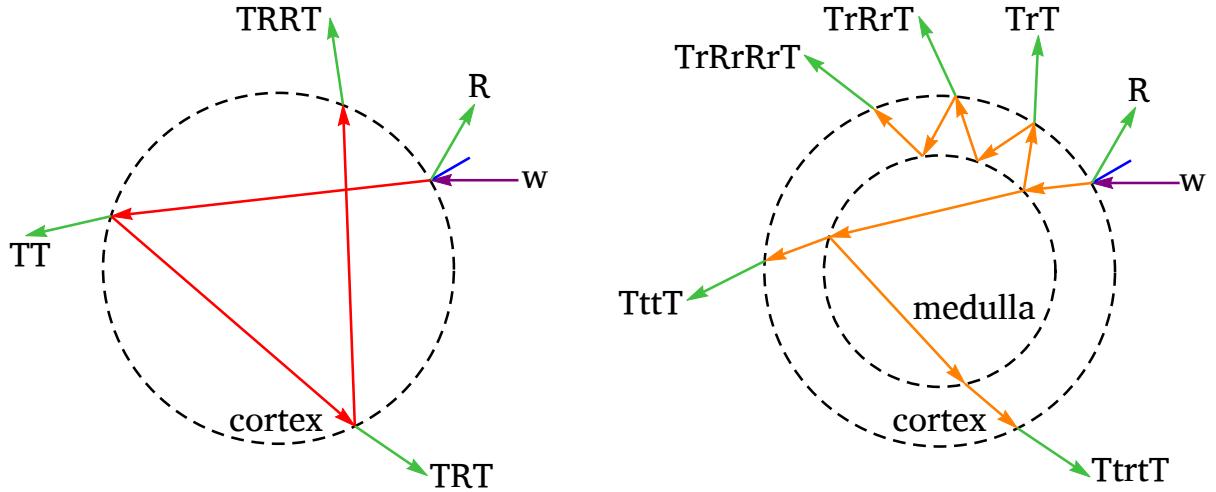


Figure 47: The paths we chose to trace through the double-cylinder scattering fur model for an incident direction w

Mode	Lobes	Description
0	R, TT, TRT, TRRT	no medulla hit
1	R, TrT, TrRrT, TrRrRrT	hit medulla, but only reflected off it
2	R, TrT, TttT, TtrtT	hit medulla, reflected or transmitted off/through it

Table 2: The *Peter Rabbit* fur shader used 3 different path modes with 4 lobes each

when the medulla isn't intersected by the paths, Mode 1 when there is total internal reflection off the medulla (which occurs when the index of refraction of the medulla is lower than the cortex), and Mode 2 when the medulla is hit and total internal reflection does not occur (Table. 2). The advantage of this representation is that medulla scattering only occurs for mode 2, there are 4 lobes computed in each case, and each mode can only occur once per incident ray hit, since each mode is mutually exclusive from a light-transport perspective. At this point a key optimization became obvious. Grouping the lobes into 4 evaluations per mode, and providing we could compute all the mathematics for the entire shader as vector computations, we could evaluate all 4 lobes for each mode simultaneously using SSE2 vector instructions. In order to allow this, we needed to introduce a few new methods for computing the various components of the full shader such as the relative azimuthal direction change originally described in Marschner et al. [2003] and generalized in Zinke and Weber [2007]. We develop our own notation, compatible with vector execution, described by the variables in Table. 3. Each path (of any length and complexity) can be described as a vector of 4 values: (R, T, r, t) (which can be stored in a vector register). We can categorize all previously defined paths: Table. 4 (or even count the infinite set of all possible paths using this notation). To compute the new change in azimuthal direction $\Phi_p(h)$ (to follow the notation of Marschner et al. [2003] and d'Eon et al. [2011]), we can then use the following method (assuming the incident ray points towards the surface, in which case the initial π value is just re-orienting the resulting direction to leave the surface, and that γ_{ic}, γ_{tc} are the azimuthal projected angles of incidence and transmission for the cortex,

Name	Description	Value Sign +	Value Sign -
R	cortex reflection	off cortex	inside cortex
T	cortex transmission	between air and cortex	nil
r	medulla reflection	off medulla	inside medulla
t	medulla transmission	between cortex and medulla	nil

Table 3: A modified signed notation was used to encode paths in a way suitable for vectorized execution

Path	R	T	r	t
R	1	0	0	0
TT	0	2	0	0
TRT	-1	2	0	0
TRRT	-2	2	0	0
TrT	0	2	1	0
TrRrT	-1	2	2	0
TrRrRrT	-2	2	3	0
TttT	0	2	0	2
TtrtT	0	2	-1	2

Table 4: Enumeration of all signed notation paths used by our fur shader

and γ_{im} , γ_{tm} the angles of incidence and transmission for the medulla respectively):

$$\Phi_p(h) = \pi + R(2A + \pi) + r(2B + \pi) + T(\gamma_{ic} - \gamma_{tc}) + t(\gamma_{im} - \gamma_{tm}), \text{ where}$$

$$A = \begin{cases} \gamma_{ic}, & \text{if } R \geq 0 \\ \gamma_{tc}, & \text{otherwise} \end{cases}, \quad B = \begin{cases} \gamma_{im}, & \text{if } r \geq 0 \\ \gamma_{tm}, & \text{otherwise} \end{cases} \quad (65)$$

Replacing the conditional expressions above with branchless vectorized *select* statements allows the azimuthal directions to be computed for multiple paths simultaneously. Due to reasons discussed in Yan et al. [2015] we don't require an equivalent expression for the cuticle-angle dependence of the longitudinal directions, as these can be computed with much simpler expressions well covered in the cited literature. Note also that the longitudinal scattering lobe form of Equation. 62 with the coupled natural log calculation of the modified bessel function of the first kind Equation. 64 can all be trivially vectorized now that we have branch-free versions of them available. As a result it is possible to calculate all 4 paths and lobe distributions for each mode defined in Table. 2 simultaneously. Extensions to wider vector units with 8 or 16 paths are also possible. The notation and Table. 4 are also conducive to automatic computation of the roughness values in the longitudinal and azimuthal directions. Since we know the number of reflection and transmissions for each interaction, along with their type, we can decrease or increase roughness for each of them as required to compute the final combined roughness. The net result of all of these optimizations was a shading cost of 4-6% down from 24%, which was a significant improvement for render times.

9.4 Weave and Clothing

Originally developed to generate curve geometry for the capes and flags on *The LEGO Movie* and *The LEGO Batman Movie*, *Weave* was further developed for *Peter Rabbit* to incorporate woven fabric generation suitable for clothing. Peter's iconic blue denim jacket was the first piece of clothing to be tackled by our system (see Figure. 48). *Weave* outputs Catmull-Rom curves for consumption by *Glimpse*. Curves are generated in the (u, v) space of a base mesh object, which simplifies textural control of the parameters. A rich set of operations are supported for artist control including stitching patterns, fuzz, wear and tear, pillowing, random offsets and displacement (see Figure. 49). The level of detail generated included fraying at the edges of clothing, scraggle (see Figure. 50), wear and tear in the form of holes and loose threads, and even fuzz hairs used for velvet-like rim effects. The purple jacket worn by the Pigling Bland character had a large number of these fuzz hairs covering it, which gave an extremely realistic velvet look when shaded using our fur shader (see Figure. 42).

Rendering of *Weave* curves was performed using a renderer primitive first created for *The LEGO Batman Movie* affectionately labelled "sausages". This primitive consists of conic sections, separated by spheres filleted in a water-tight fashion. The extra visual detail provided by this primitive when used to represent cloth fibres was clearly visible for close-up shots like Figure. 41. These *sausages* curves were shaded using a standard diffuse/specular microfacet model with a Jensen dipole subsurface scattering



Figure 48: Weave was used for generating garments and clothing that were rendered using our affectionately named "sausages" geometric primitive

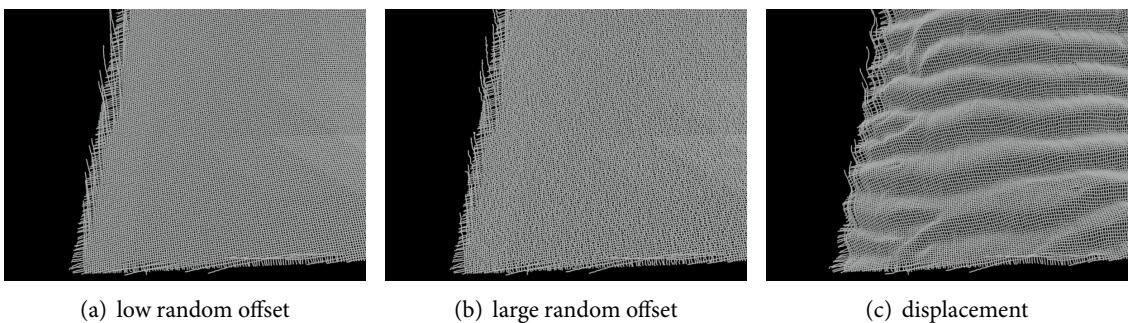


Figure 49: *Weave* system basic operators

component and a raytraced single-scattering component. This allowed realistic backlighting and diffusion effects to be incorporated in an effective fashion without the need to develop any new technologies such as volumetric representations or scattering models. Re-use of existing microfacet models also reduced the time it took for surfacing artists to obtain the desired look since they were already familiar with the controls and how they operated.

9.5 Grass Replacement

During the shoot of *Peter Rabbit* weather conditions were highly variable, switching from sweltering heat to heavy rain. As a result, the condition and palette of the ground in the live-action plates also varied. We required a high degree of interaction to occur between the characters and branches, rocks, props and grass. For elements like branches, rocks and props we planned for them to be fully CG elements, but for the large wide shots of grass it quickly became evident that obtaining the consistency of look and tonal feel we wanted would only be possible if we fully replaced large areas with computer generated grass (Figure. 51(a)).

To render the grass we added a new oriented ribbons primitive to *Glimpse*. Each grass ribbon consisted of multiple bilinear patches following a central Catmull-Rom curve. The ribbons oriented themselves to a reference up vector per control vertex to allow us to accurately define twisting along their length. We found this ribbon primitive produced better results in silhouette than using triangulated or quadrangulated meshes due to the smoother profile of the resulting ruled surface. Memory savings were

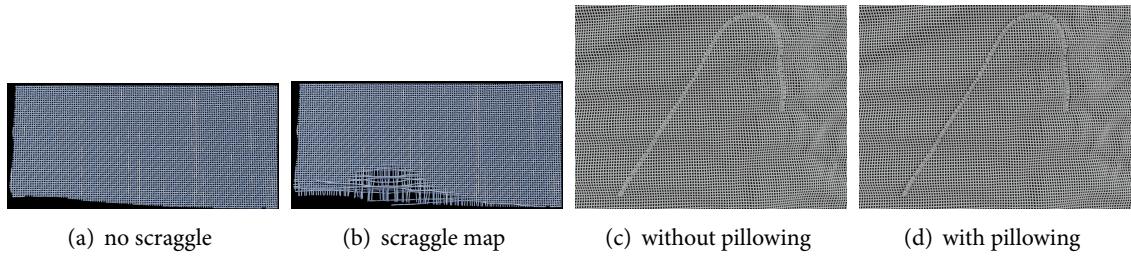
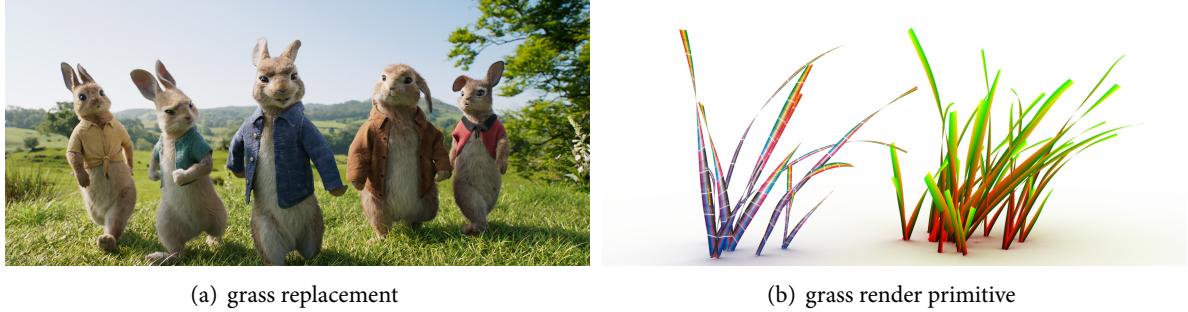
Figure 50: *Weave* scraggle and pillowing operations

Figure 51: (a) Grass replacement was required to allow characters to realistically integrate with their environment and compensate for live-action plate variation, (b) rendering was achieved with a ribbon primitive

also possible due to the use of Catmull-Rom curves coupled with up vectors vs the data required to store fully diced polygonal meshes. A plug-in called *ALFRO* (also used to as our hair/fur grooming tool) for our in-house node evaluation system *ALF* was used to generate the grass ribbon curves and up vector data. An example render of our ribbon render primitive is shown in Figure. 51(b).

9.6 Prelit Workflow

$$L_{Ro}(x, w_o) \approx L_{Po}(x) \int_{\Omega} \frac{L_{Ri}(x, w_i)}{L_{Pi}(x, w_i)} (w_i \cdot n) dw_i \quad (66)$$

To facilitate live-action integration we captured accurate high dynamic range (HDR) lighting data from the set photographically using a process outlined in Heckenberg et al. [2017a]. These HDR light probes were used to help integrate the rendered characters into the photographic plates in a seamless way (Figure. 52). A specialized mechanism called *Prelit Materials* (Steve Agland and Heckenberg [2018]) was developed for *Glimpse* to handle this. We define two subsets of the rendered scene:

- The *prelit scene* (Figure. 53(a)) contains geometry and light sources matching the set.
- The *relit scene* (Figure. 53(b)) contains the scene as we want it to appear in the film, usually the *prelit scene* with additional synthetic elements.

In a single-pass render (Figure. 53(c)) we can use a *combined scene* to sample the irradiance from both the *relit scene* L_{Ri} and the *prelit scene* L_{Pi} . The ratio of these two quantities $\frac{L_{Ri}(x, w_i)}{L_{Pi}(x, w_i)}$ will be low < 1 if a synthetic object is casting a new shadow or high > 1 if a synthetic object is emitting or reflecting new light into the scene. We assume the projected plate is the *prelit* radiance (outgoing light) L_{Po} and multiply it by our irradiance ratio to compute the new *relit* radiance L_{Ro} . Hence our final result is Equation. 66, which currently assumes lambertian diffuse surfaces but generated useful results with little manual work throughout *Peter Rabbit*. Further details about this technique can be found in Steve Agland and Heckenberg [2018].



Figure 52: *Prelit* materials and light transport algorithms were developed to integrate synthetic elements with live-action elements

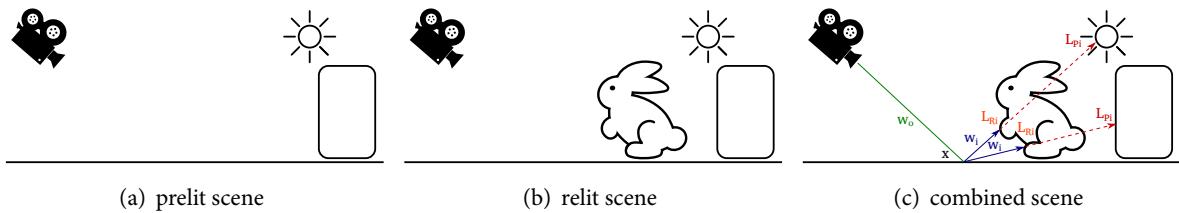


Figure 53: *Prelit* light transport allowed live-action integration to happen per sample directly within the renderer by combining (a) prelit scene with (b) relit scene to produce (c) combined render of the set lighting and synthetic rendered lighting

9.7 Single Pass Render Motion-Blur Integration

To better compose shots once the CG characters were in place, we used cropped framings with animated camera film-back offsets. Since the live-action plates were shot at high resolution, there was a lot of latitude for such manipulation, and it allowed for fine-grained tweaking of character animation timing to emphasize key motions and movements to help tell the story. Camera tracking was performed on the full uncropped plates, then this film-back offset animation was added afterwards during character animation. To match the motion-blur of this added offset we allowed for the full motion-blur of the tracking camera plus the offset animation. In cases where the offset animation was moving in the same direction as the motion of a character, it made the character sharper and helped to keep the character expressions more readable. The net result was a single-pass render allowing for complex multi-layered camera animation offsets without the need to do the extra motion-blur at the compositing stage.

9.7.1 FBOMBS - Film Back Offset Motion Blur Scale

In situations where there were rendered stationary props in the background of a shot, such as rocks or fruit, we needed a way to control the motion-blur of the film-back offset animation. If the motion blur of the combined motions was used on stationary objects, their motion-blur no longer matched that of the plate (which didn't contain the motion-blur of the added offset animation). To handle this case, we added a control that allowed us to blend between motion-blur with the original tracking camera motion, and motion-blur with the combined animation. The control was called "FBOMBS" or Film Back Offset Motion Blur Scale.

References

- J M Blair and C A Edwards. 1974. *Stable rational minimax approximations to the modified Bessel functions $I_0(X)$ and $I_1(X)$* . Technical Report AECL-4928. Chalk River Nuclear Laboratories. <http://cds.cern.ch/record/419431>
- Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2015. A Practical and Controllable Hair and Fur Model for Production Path Tracing. In *ACM SIGGRAPH 2015 Talks (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 23, 1 pages. <https://doi.org/10.1145/2775280.2792559>
- Anthony B. Davis. 2006. Effective Propagation Kernels in Structured Media with Broad Spatial Correlations, Illustration with Large-Scale Transport of Solar Photons Through Cloudy Atmospheres. In *Computational Methods in Transport*, Frank Graziani (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 85–140.
- Eugene d'Eon. 2013. Publons 2867. publons.com/p/2867/. (2013).
- Eugene d'Eon, Guillaume Francois, Martin Hill, Joe Letteri, and Jean-Marie Aubry. 2011. An Energy-conserving Hair Reflectance Model. In *Proceedings of the Twenty-second Eurographics Conference on Rendering (EGSR '11)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1181–1187. <https://doi.org/10.1111/j.1467-8659.2011.01976.x>
- Eugene d'Eon, Steve Marschner, and Johannes Hanika. 2013. Importance Sampling for Physically-based Hair Fiber Models. In *SIGGRAPH Asia 2013 Technical Briefs (SA '13)*. ACM, New York, NY, USA, Article 25, 4 pages. <https://doi.org/10.1145/2542355.2542386>
- Luca Fascione, Johannes Hanika, Rob Pieké, Christophe Hery, Ryusuke Villemin, Thorsten-Walther Schmidt, Christopher Kulla, Daniel Heckenberg, and André Mazzone. 2017. Path Tracing in Production - Part 2: Making Movies. In *ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. Article 15, 32 pages. <https://doi.org/10.1145/3084873.3084906>
- Daniel Heckenberg, Steve Agland, Jean Pascal leBlanc, and Raphael Barth. 2017a. Automated Light Probes from Capture to Render for Peter Rabbit. In *ACM SIGGRAPH 2017 Talks (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 17, 2 pages. <https://doi.org/10.1145/3084363.3085087>
- Daniel Heckenberg, Luke Emrose, Matthew Reid, Michael Balzer, Antoine Roille, and Max Liani. 2017b. Rendering the Darkness: Glimpse on the LEGO Batman Movie. In *ACM SIGGRAPH 2017 Talks (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 8, 2 pages. <https://doi.org/10.1145/3084363.3085090>
- Jenni Heino, Simon Arridge, Jan Sikora, and Erkki Somersalo. 2003. Anisotropic effects in highly scattering media. *Phys. Rev. E* 68 (Sep 2003), 031908. Issue 3. <https://doi.org/10.1103/PhysRevE.68.031908>
- Wenzel Jakob. 2012. Numerically stable sampling of the von Mises Fisher distribution on S^2 (and other tricks). (2012). <https://www.mitsuba-renderer.org/~wenzel/files/vmf.pdf>

Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. 2003. Light Scattering from Human Hair Fibers. *ACM Trans. Graph.* 22, 3 (July 2003), 780–791. <https://doi.org/10.1145/882262.882345>

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

Matt Pharr and Cem Yuksel. 2016. The Implementation of a Hair Scattering Model. (2016). <http://www.pbrt.org/hair.pdf>

F. R. S. Sir Ronald Fisher. 1953. Dispersion on a sphere. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 217, 1130 (1953), 295–305. <https://doi.org/10.1098/rspa.1953.0064> arXiv:<http://rspa.royalsocietypublishing.org/content/217/1130/295.full.pdf>

Matthew Reid Steve Agland and Daniel Heckenberg. 2018. Prelit Materials: Light Transport for Live-Action Elements in Production Rendering. In *ACM SIGGRAPH 2018 Talks (SIGGRAPH ’18)*. ACM, New York, NY, USA, 2. <https://doi.org/10.1145/3214745.3214746>

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.* 33, 4, Article 143 (July 2014), 8 pages. <https://doi.org/10.1145/2601097.2601199>

Ling-Qi Yan, Chi-Wei Tseng, Henrik Wann Jensen, and Ravi Ramamoorthi. 2015. Physically-accurate Fur Reflectance: Modeling, Measurement and Rendering. *ACM Trans. Graph.* 34, 6, Article 185 (Oct. 2015), 13 pages. <https://doi.org/10.1145/2816795.2818080>

Arno Zinke and Andreas Weber. 2007. Light Scattering from Filaments. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (March 2007), 342–356. <https://doi.org/10.1109/TVCG.2007.43>

10 Turn it down! - MPC's guide to noise management

ROB PIEKÉ, MPC

10.1 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. We started experimenting with ray tracing (initially for shadows, followed by reflections) when it first became available in the v11 release in 2003, and continued to use RenderMan's growing capabilities in further releases.

Tracking industry trends, we increased our usage of ray tracing for indirect lighting effects, and restructured 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.

10.1.1 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.

10.1.2 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 newer.

10.2 Sampling and noise

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 tessellation (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.

The effect of some of these individual controls can be seen below in Figure 54.

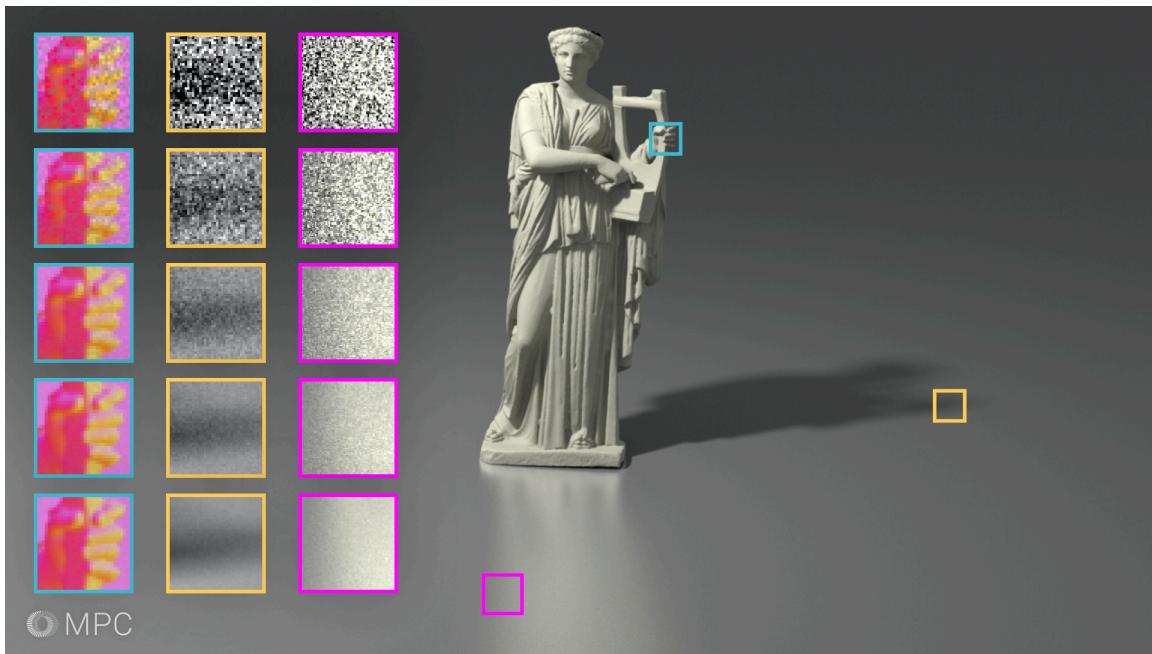


Figure 54: Demonstrating the effect of sampling. The blue square inserts show noise in the geometric normals rendered with 1, 2, 4, 8, 16 camera samples. The yellow square inserts show noise in shadows with 1, 4, 16, 256 light samples. The pink square inserts show noise in glossy reflections with 1, 4, 16, 256 BxDF samples

10.3 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.

The time-quality trade-off is well represented below in Figure 55.

10.3.1 Staying sharp

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":



Figure 55: The pink and blue inserts show, from top-to-bottom, the effect of denoising renders with 1, 4, 16, 64, 256 samples.

First, 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.

Second, we used the tangent vector of the fur to provide 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.

This is illustrated in Figure 56 below.

10.3.2 Rise of the machines

Recently we have started to investigate how machine learning can be applied to image processing tasks in VFX. While we have some potentially interesting ideas on pipelines and workflows well suited to large-scale companies like MPC, the journey is only just beginning and we don't have any results to share at the time of this publication.

10.4 Further thoughts and 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).



Figure 56: Demonstrating some tricks for denoising fur. The top row of blue inserts use standard production settings and fur normals as supplementary information. The second row doubles the render resolution but reduces the sample count to maintain equal render time. The third row replaces the normals with fur tangent information

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.