Path tracing in Production

Part 2: Making movies

SIGGRAPH 2017 Course Notes

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

Abstract

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

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

Contents

1 Objectives 3

2 Syllabus 3

3 Organizers 5
- 3.1 Luca Fascione, Weta Digital 5
- 3.2 Johannes Hanika, Weta Digital 5

4 Presenters 5
- 4.1 Rob Pieké, MPC 5
- 4.2 Christophe Hery, Pixar Animation Studios 5
- 4.3 Ryusuke Villemin, Pixar Animation Studios 6
- 4.4 Thorsten-Walther Schmidt, Lightrig GmbH 6
- 4.5 Christopher Kulla, Sony Pictures Imageworks 6
- 4.6 Daniel Heckenberg, Animal Logic 6
- 4.7 André Mazzone, Industrial Light & Magic 6

5 It all started with a camera - MPC's move to path tracing 7
- 5.1 A brief history 7
  - 5.1.1 Adding rays to REYES 7
  - 5.1.2 Fast & Furious: Supercharged 7
  - 5.1.3 The Jungle Book 7
- 5.2 Our early reflections 7
  - 5.2.1 Representing reality 8
  - 5.2.2 Complexity management 8
  - 5.2.3 Progressive rendering 8
  - 5.2.4 Optimisations and other trickery 8
- 5.3 Denoising 9
- 5.4 Remaining challenges 9

6 Emeryville: where all the fun light transports happen! 10
- 6.1 Photons 10
- 6.2 Metropolis Photon Guiding 10
- 6.3 Manifold Walk 12
- 6.4 Indirect Guiding 12
- 6.5 Per Light Controls 12
- 6.6 Light Editing 14
  - 6.6.1 Interactive Visualization of Light Transport 14
  - 6.6.2 Artistic Manipulation of Light Transport 15
- 6.7 Conclusion 18

7 Arnold at Imageworks: Path Tracing from Monster House to Smurfs: The Lost Village 20
- 7.1 Monster House 20
- 7.2 Bounding Volume Hierarchies 21
- 7.3 Tracing Hair and Fur 21
- 7.4 Open Shading Language 21
- 7.5 Light Path Expressions 22
- 7.6 Advancing the Integrator 22
- 7.7 Subdivision Surfaces 24
- 7.8 Multiple Scattering 24
- 7.9 Simplifying Sampling 24

7.10 Conclusion 25

8 A Change Of Path At Animal Logic **27**
8.1 A Brick Too Far 27
8.2 Putting the Pieces Together 28
8.3 Lost in the Weeds? 28
8.4 Glimpse Today 28
8.4.1 Scene Description 29
8.4.2 Ash Material Composition 29
8.4.3 Global Sampling Controls 30
8.4.4 Energy Redistribution Depth-of-Field 31
8.4.5 Creature Features 31
8.5 Render Anywhere Else? 31
8.6 Reflections 32
8.7 Current and Future Challenges 32
8.7.1 Scale and Scalability 33
8.7.2 Frame Rendering in Production 33

9 ILM's Path to Tracing **35**
9.1 Introduction 35
9.2 History of Techniques 35
9.3 What path-tracing Is 35
9.3.1 Ray-tracing 35
9.3.2 Ray-tracing with a Difference 36
9.3.3 Conceptually Simple 36
9.3.4 Extensible 36
9.4 What Path-tracing is Not 36
9.4.1 Not New 36
9.4.2 Not Always Path-Tracing 36
9.4.3 Not Always Fast 36
9.4.4 Not Always Memory efficient 37
9.4.5 Not Cheat-Free 37
9.4.6 Not Simple to Implement 37
9.4.7 Not Always Simple for the Artist 37
9.4.8 Not Enough 37
9.4.9 Not Mature 37
9.5 Conclusion 38

1 Objectives

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

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

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

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

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

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

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

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

2 Syllabus

14:00 — Opening statement and welcome message

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

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

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

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

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

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

15:40 — Break (15min)

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

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

16:25 — ILM's Path to Tracing (30min)

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

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

3 Organizers

3.1 Luca Fascione, Weta Digital

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

3.2 Johannes Hanika, Weta Digital

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

4 Presenters

4.1 Rob Pieké, MPC

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

4.2 Christophe Hery, Pixar Animation Studios

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

4.3 Ryusuke Villemin, Pixar Animation Studios

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

4.4 Thorsten-Walther Schmidt, Lightrig GmbH

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

4.5 Christopher Kulla, Sony Pictures Imageworks

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

4.6 Daniel Heckenberg, Animal Logic

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

4.7 André Mazzone, Industrial Light & Magic

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

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

Rob Pieké, *MPC*

5.1 A brief history

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

5.1.1 Adding rays to REYES

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

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

5.1.2 Fast & Furious: Supercharged

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

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

5.1.3 The Jungle Book

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

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

5.2 Our early reflections

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

5.2.1 Representing reality

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

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

5.2.2 Complexity management

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

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

5.2.3 Progressive rendering

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

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

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

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

5.2.4 Optimisations and other trickery

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

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

5.3 Denoising

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

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

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

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

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

5.4 Remaining challenges

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

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

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

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

6 Emeryville: where all the fun light transports happen!

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

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

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

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

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

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

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

6.1 Photons

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

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

6.2 Metropolis Photon Guiding

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

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

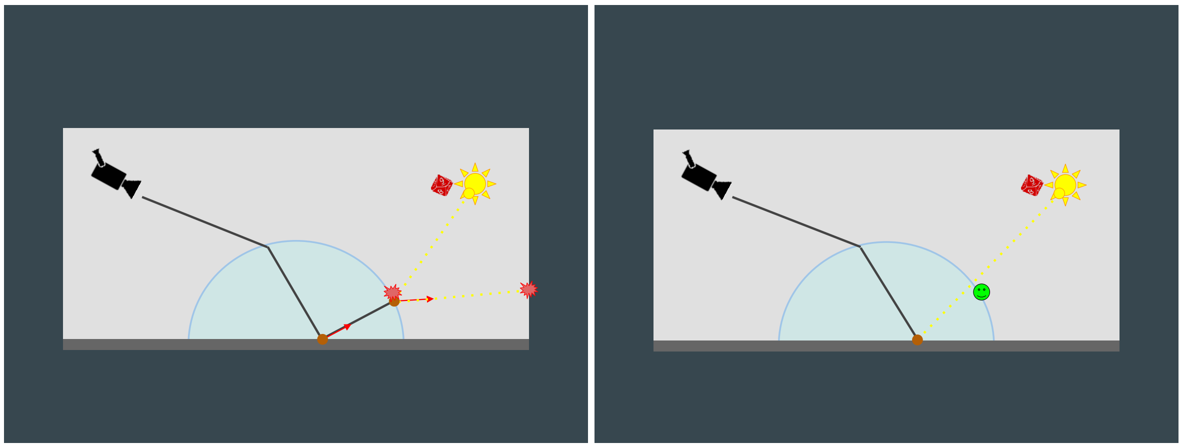

Figure 1: Left: sampling failure trough refraction. Right: thin shadow trick.

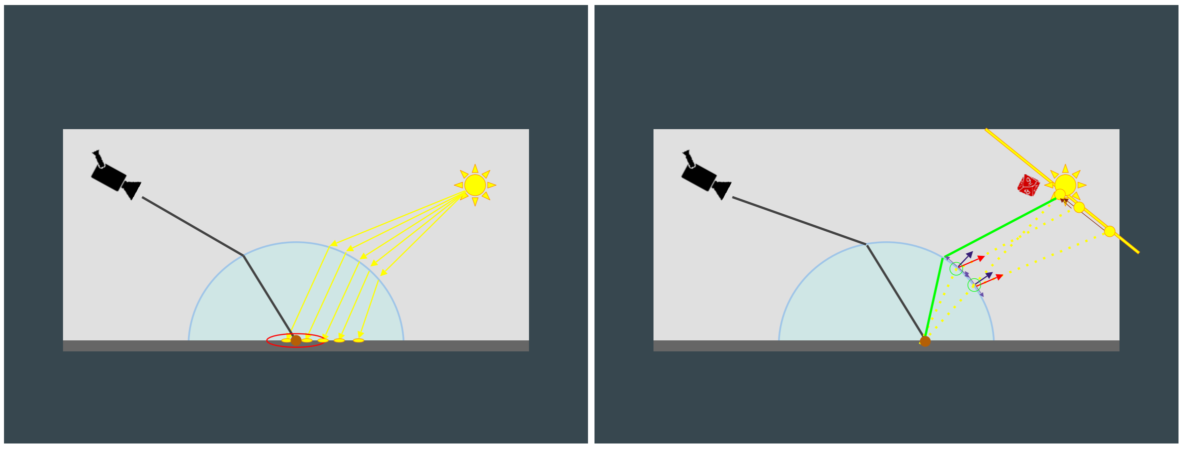

Figure 2: Left: photon mapping approach. Right: manifold walk solution.

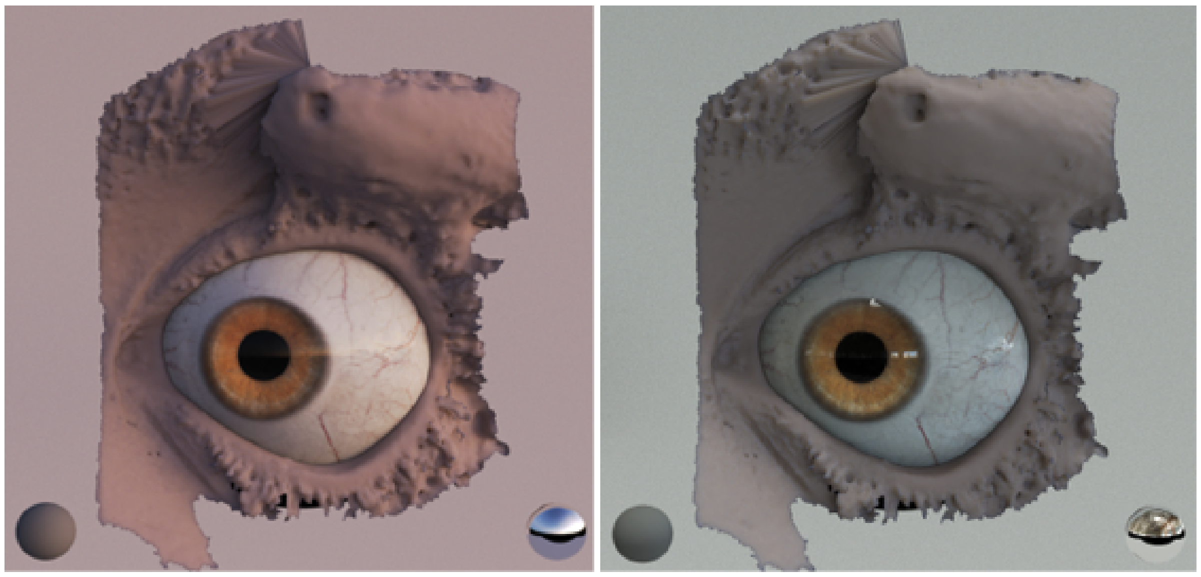

Figure 3: Manifold walk (for gleam/caustics in eye). Left: lighting scenario 1. Right: lighting scenario 2. Eye geometry provided and copyright © by Disney Research Zurich.

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

6.3 Manifold Walk

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

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

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

6.4 Indirect Guiding

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

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

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

6.5 Per Light Controls

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

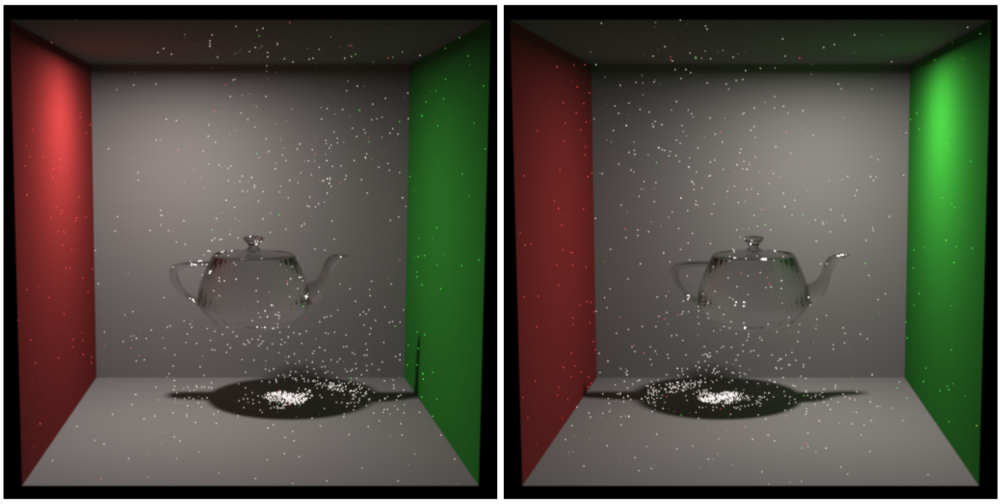

Figure 4: Unidirectional lights.

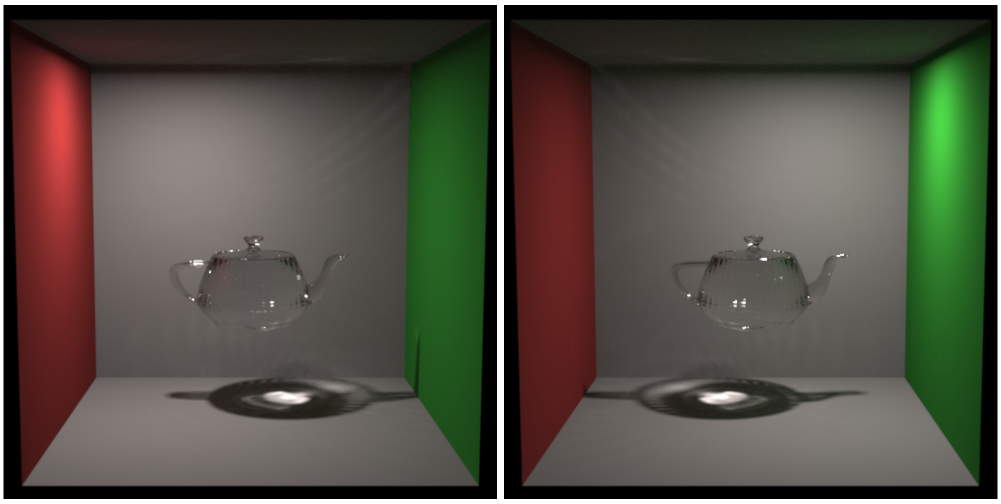

Figure 5: Bidirectional lights.

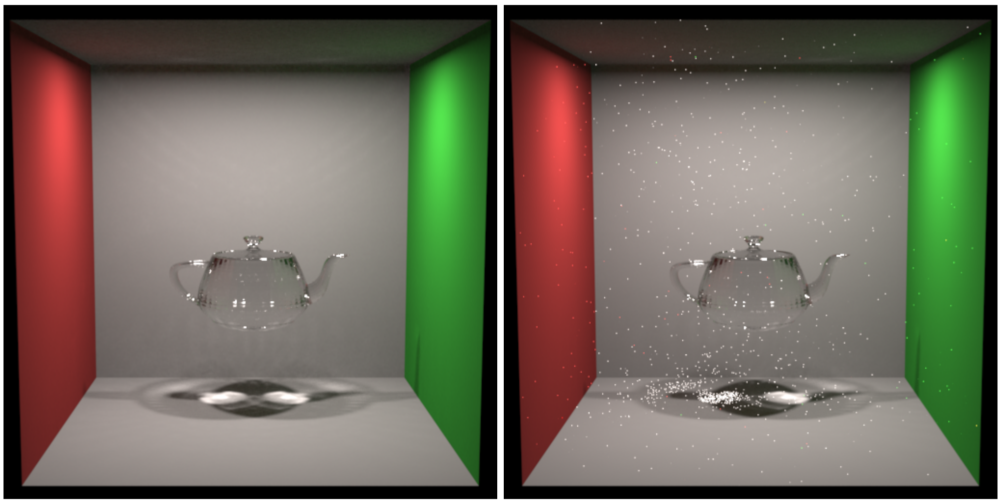

Figure 6: Left: both lights bidirectional. Right: left light bidirectional, right light unidirectional.

6.6 Analyzing and Controlling Light Transport

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

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

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

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

6.6.1 Interactive Visualization of Light Transport

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

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

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

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

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

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

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

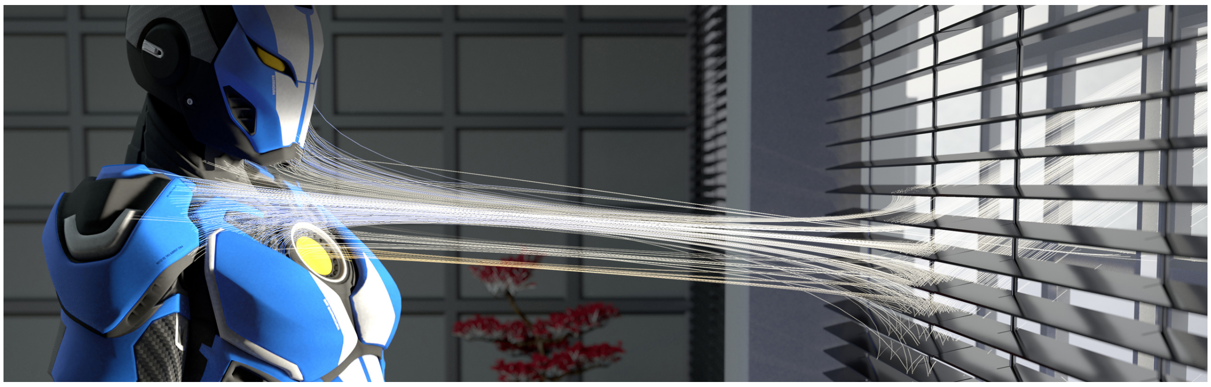

Figure 7: Light transport visualization provides key information about illumination structure within virtual scenes. Light path inspection shows logically grouped bundles of energy that deliver the illumination to the region of interest.

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

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

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

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

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

6.6.2 Artistic Manipulation of Light Transport

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

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

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

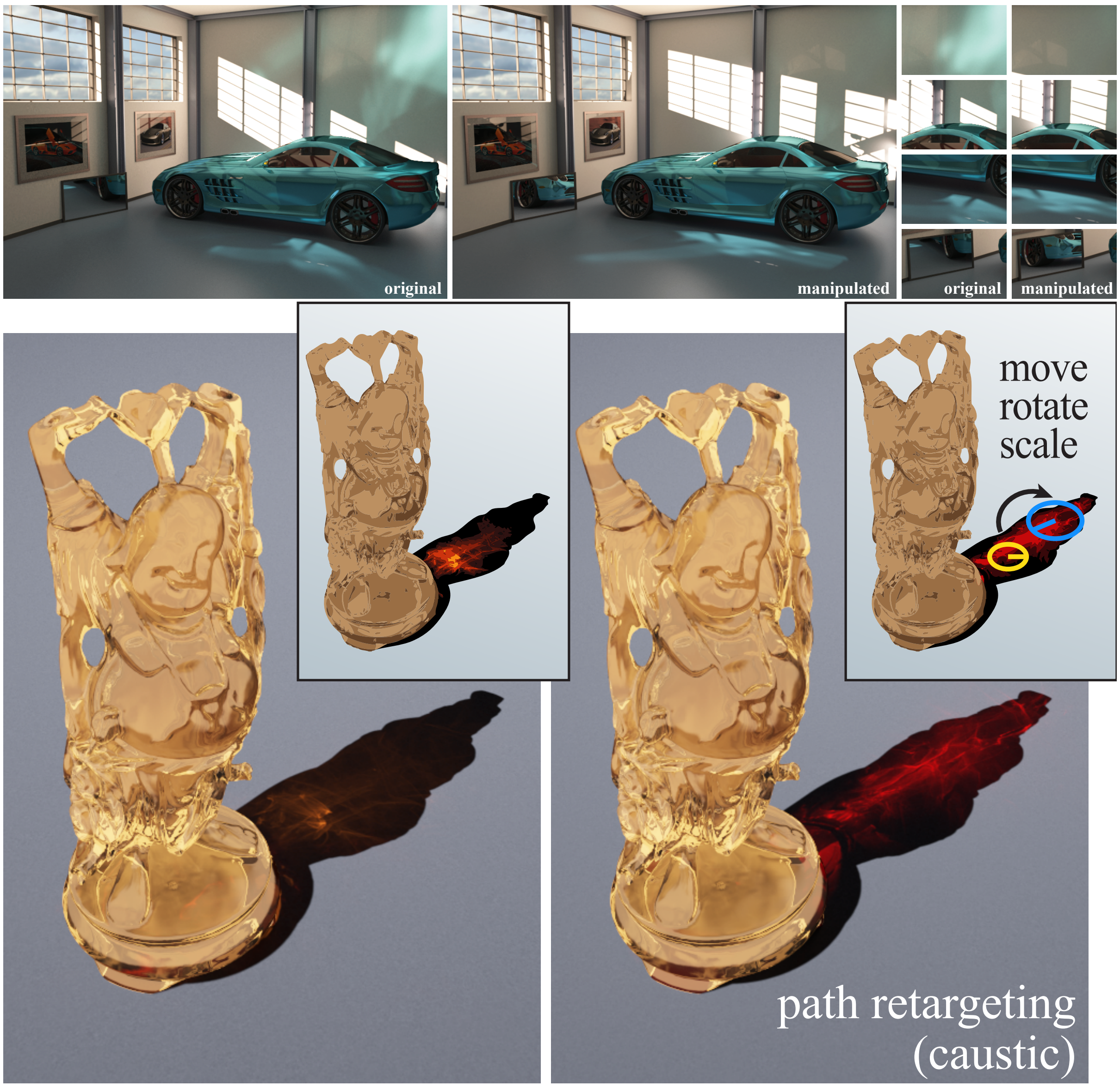

Figure 8: Example edits in the Garage scene. Before/after close-ups (right): removing reflections caused by the car, moving sunlight refracted through the windows, transforming a glossy caustic, and altering the mirror reflection. The Buddha scene shows a path retargeting example. Left: original render.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

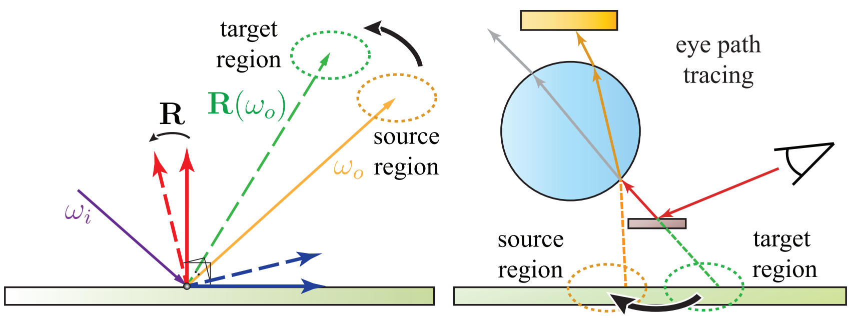

Figure 9: Left: path retargeting can be viewed as a transformation of either the incident or exitant shading tangent frame, depending on the manipulation direction and the direction of the path construction method. Right: when constructing a path in a direction opposite to the manipulation direction,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

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

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

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

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

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

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

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

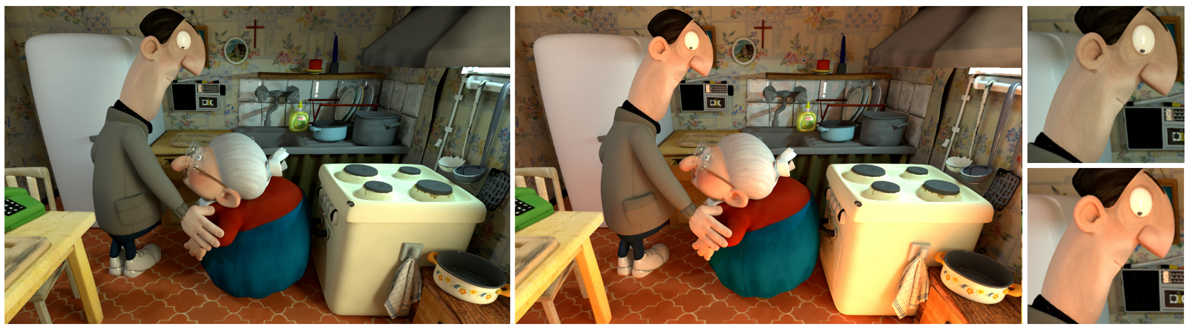

Figure 10: Manipulating diffuse GI in the Visit scene. Left to right: unmodified light transport, indirect diffuse illumination from the floor is manipulated with path retargeting.

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

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

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

6.7 Conclusion

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

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Christopher Kulla, *Sony Pictures Imageworks*

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

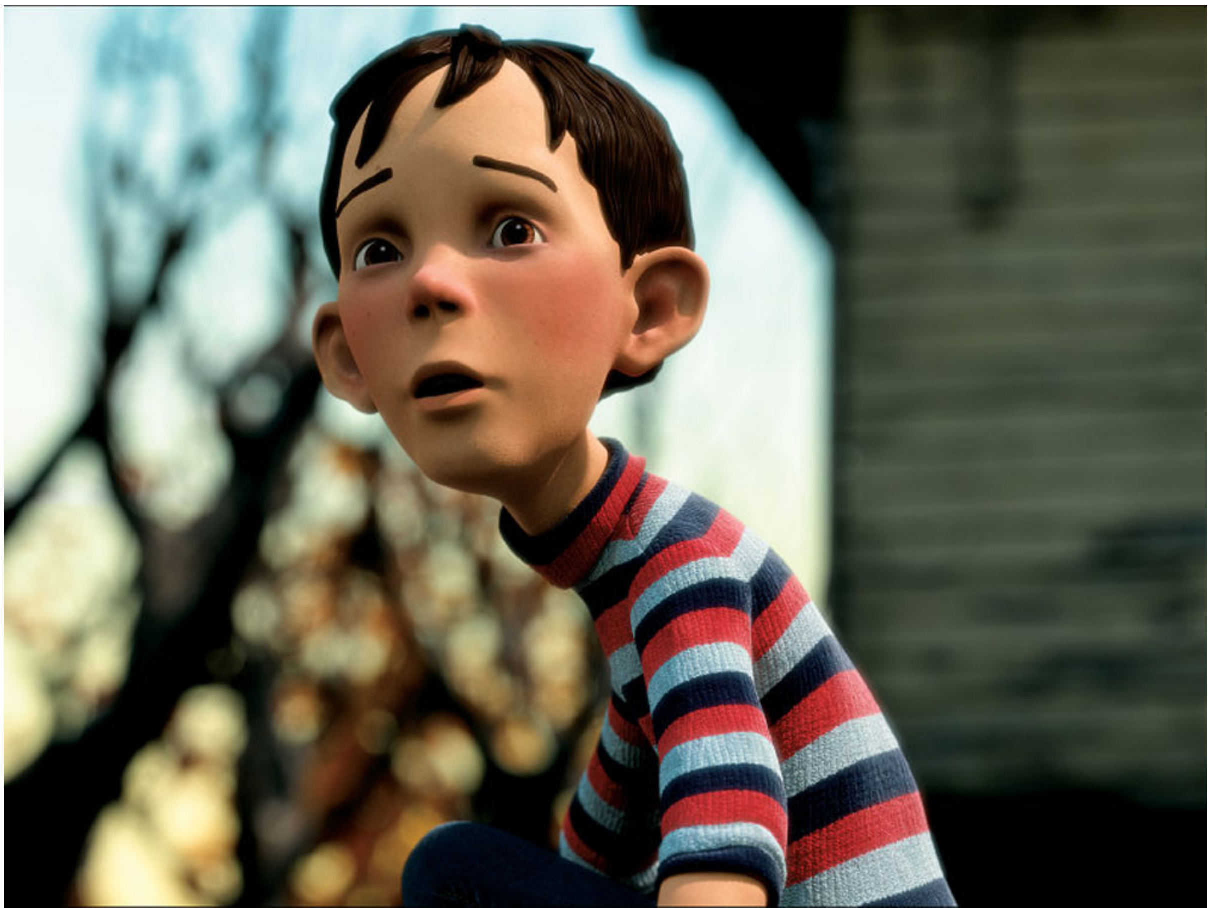

Figure 11: Monster House, the first full length animated feature rendered with *Arnold* and brute force path tracing. ©Columbia Pictures Industries, Inc. and GH One LLC. All Rights Reserved. Courtesy of Columbia Pictures

7.1 Monster House

The production of Monster House marked the first time, to our knowledge, that indirect lighting was rendered by brute force path tracing for a full length animated film. The increased realism gave the film a tangible, stop-motion aesthetic (see Figure 11). A number of compromises were made to achieve this: motion blur was disabled, hair was modeled with geometry, and only a single bounce of diffuse illumination was simulated. At the time *Arnold* was based around uniform grid acceleration structures and its shading system exposed low-level details such as light loops, tracing and sampling calls directly to shader writers. To reduce sampling variance to acceptable levels, lighting artists had to be conscious of any small elements that would be excessively bright to indirect rays which could include manually controlling ray visibility flags, assigning simplified shaders and so on. Despite this amount of hand holding, once an environment or sequence had been configured by senior lighters, many shots would fall into place very quickly. This was far superior to other rendering solutions of the time which required manual pass management in every shot.

7.2 Bounding Volume Hierarchies

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

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

7.3 Tracing Hair and Fur

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

7.4 Open Shading Language

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

From the beginning, the shading language was conceived to expose BSDFs through *closures* rather than exposing any algorithm specific constructs such as light loops or trace calls. Shading derivatives (required for pre-filtering of shading signals and bump mapping) were implemented using dual arithmetic (Piponi [2004]) instead of finite differencing to better support single-point shading architectures common to ray tracers. Our very first runtime for OSL assumed we would be able to amortize the cost of a shading interpreter over bundles of rays. As we connected this brand new interpreter to a brand new packet ray tracing code, we ended up rewriting a large portion of the code base. Around the same time, Marcos Fajardo was establishing SolidAngle S.L. to bring to market the renderer as we had been using it up until then. The rapid churn of unproven code proved to be counter productive to the API stabilization required for integration of the renderer into third party packages, keeping other operating systems (like *Microsoft Windows*) well supported, etc…The teams at Imageworks and Solid Angle therefore mutually agreed to fork development of the renderer, while still keeping in place all joint intellectual property agreements including mutual access to source code. This decision proved fruitful as we were able to complete the push to a production ready renderer built around OSL. All projects entering production at that time embraced this "new" OSL-based renderer, however quickly discovered that performance had taken a major step back. It wasn't until the move to an LLVM based backend that OSL would really become production ready. In moving to LLVM, we also reverted to a single-ray architecture. While packet tracing produced impressive speedups for coherent rays, the speedups on incoherent rays were offset by much greater code complexity and surprising performance pitfalls when bundles became too small. We instead re-dedicated ourselves to the simplicity of our initial rendering architecture.

7.5 Light Path Expressions

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

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

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

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

7.6 Advancing the Integrator

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

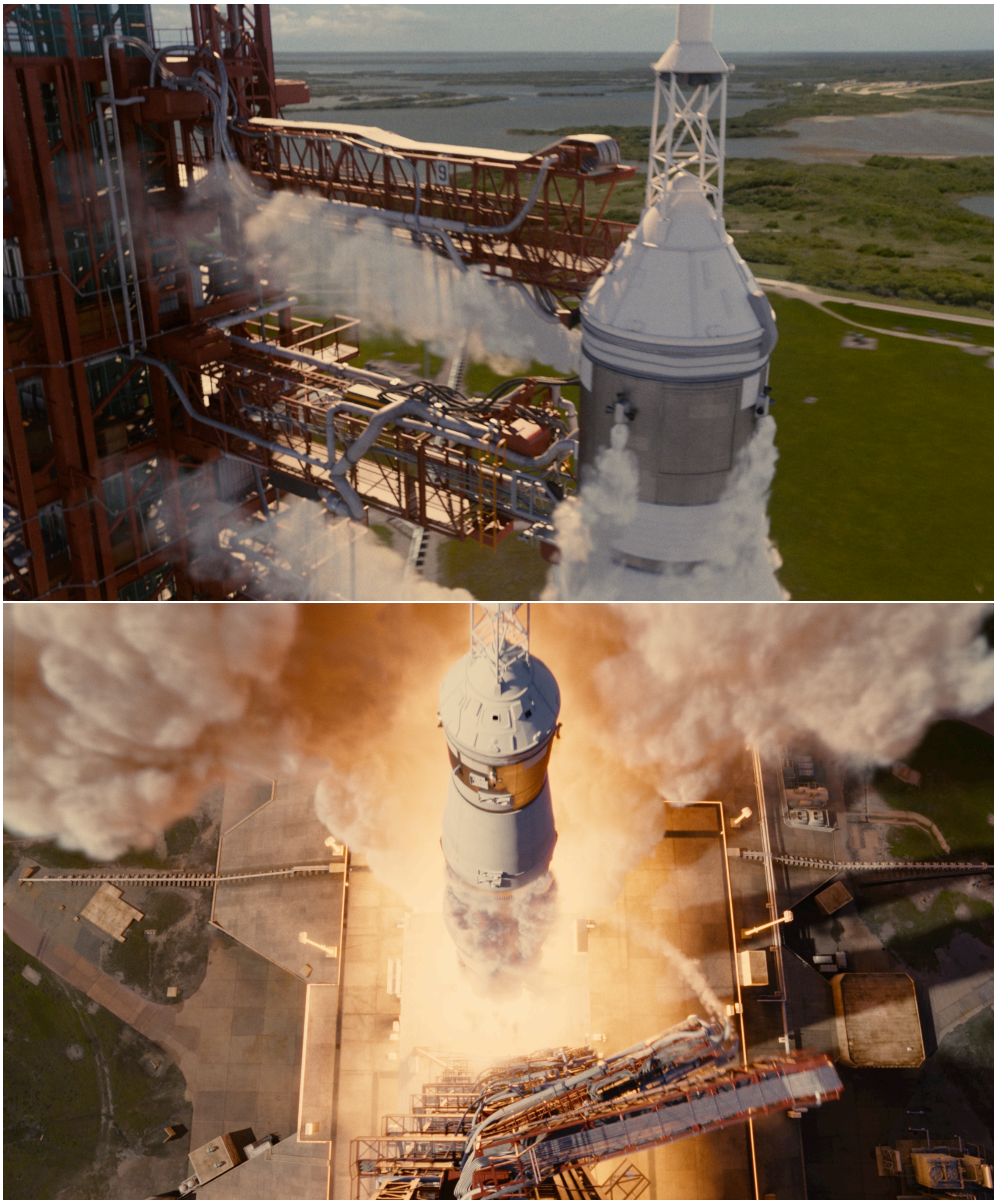
Figure 12: Apollo 11 launch sequence from the film *Men In Black 3*, one of the first uses of fully integrated volumetric effects in the renderer.
©2012 Columbia Pictures Industries, Inc. All Rights Reserved.

a predictable and automatic way. These included simple clamps on ray intensity, roughness clamping to regularize difficult glossy inter-reflections and so on. We were able to introduce volume rendering as a first class citizen to the renderer (Kulla and Fajardo [2012]), which proved invaluable for scenes with smoke interacted with geometry and lighting in complicated ways (see Figure 12). We transitioned mid-production from a subsurface scattering implementation based around point clouds to a fully ray traced solution (King et al. [2013]) with no changes in the shaders or artist scenes. We finally were able to experiment with bi-directional methods and Metropolis samplers (Veach [1998]) which proved very useful for generating ground truth images as well as occasionally rendering caustic passes for real shots.

7.7 Subdivision Surfaces

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

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

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

7.8 Multiple Scattering

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

7.9 Simplifying Sampling

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

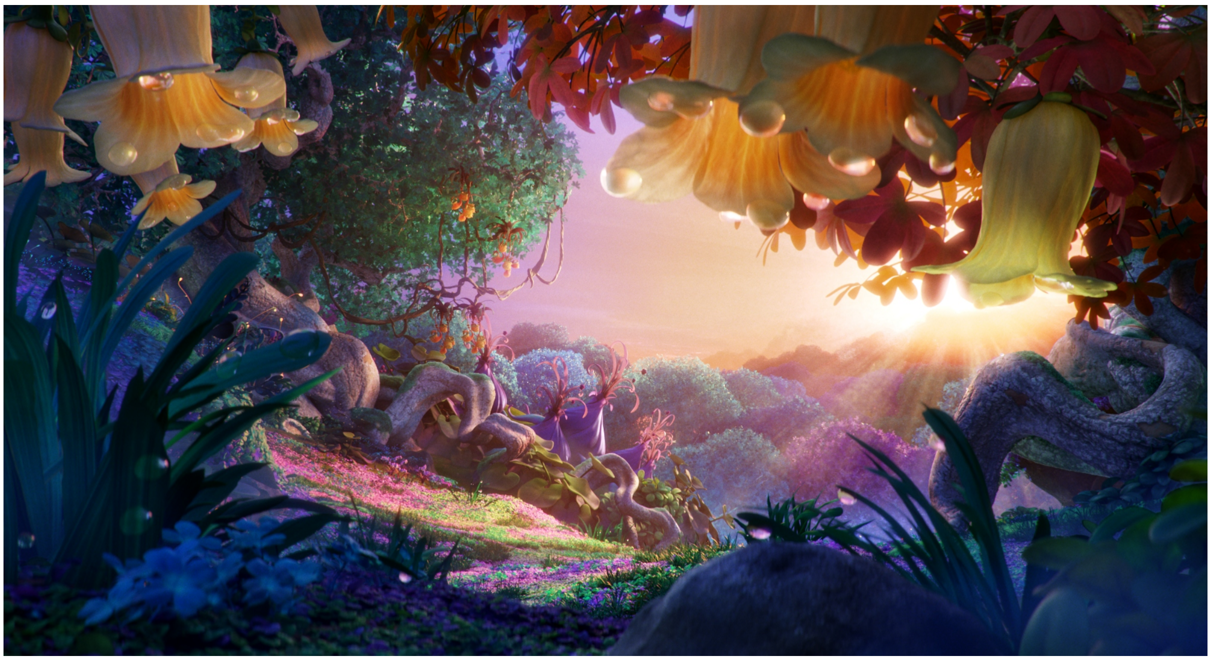

Figure 13: Enchanted forest environment from Smurfs: The Lost Village. Compared to Figure 11, scene and shading complexity have both increased by orders of magnitude. ©2017 Sony Pictures Animation. All Rights Reserved.

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

7.10 Conclusion

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

References

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

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

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

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

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

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

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

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

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

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

8 A Change Of Path At Animal Logic

Daniel Heckenberg, *Animal Logic*

Around 2012, *Animal Logic* was in preproduction for *The LEGO Movie* and contemplating technology and technique choices for the project. We had a very well established toolset based around *Maya* and *Pixar's RenderMan*, a third-generation shading system based on physically-plausible global illumination and…a deadline.

We faced a clear choice of technique: represent *LEGO* models with individual brick geometry or as merged aggregates. The latter approach was seemingly the only option to achieve scale with a rasterising renderer, and matched our usual workflow: carefully authored assets with a minimum of geometry and adaptive detail through subdivision and shading.

We pursued this approach and soon realised the challenges of capturing the shape and appearance of shiny, sharp-edged bricks. Glints and highlights from edges, gaps and studs are lost without sufficient geometry (either explicit or through subdivision creases). Shading approaches don't reliably capture these features, particularly with ray-traced primary visibility.

So we returned to the obvious: preserve individual bricks in our final render geometry as described in Smith et al. [2014]. This approach seems eminently suitable for a renderer that supports instancing, but not so for the rasteriser that we were using, designed around adaptive subdivision and specialisation.

We contemplated a switch to other commercial renderers which offered efficient instancing and that would be well-suited to global illumination. However, the project-wide changes that this would require did not seem achievable in our timeframe. Fortuitously, at just this time *PRMan 17* entered beta with support for instancing and our approach was set.

8.1 A Brick Too Far

As we progressed with the project, we were constantly pushing against performance, memory and quality limits as we built out a giant *LEGO* world. Meanwhile, Max Liani had been developing Glimpse, an interactive preview path tracer, as a personal project with promising results. This led to an experiment, FrankenGlimpse: Luke Emrose took the fast but feature-limited Glimpse and coupled it to *PRMan* as a ray-server.

Exploiting the flexibility of *PRMan*, scenes could be passed into Glimpse via an RIFilter with no outward pipeline or asset changes. Ray-tracing services were provided by Glimpse using RSL ShadeOps to replace built-in functionality. The initial experiment replaced shadow tracing, as the most frequent and least functionally-demanding operation, and yielded immediate performance improvements at modest memory cost for the additional scene representation, largely due to Glimpse's efficient instancing support.

To achieve greater benefits, we incrementally added functionality to Glimpse and FrankenGlimpse. We implemented features in an order that would allow us to progressively replace *PRMan* mechanisms and subsystems:

Feature	Benefit
shadow tracing	avoid shadow maps
subdivision surfaces	improved shadow fidelity
C++ plastic shader	indirect illumination, speed, avoid PRMan RT and caches
textures	improved illumination fidelity
subsurface scatter	avoid PRMan RT and caches

We also started to use Glimpse as a complete stand-alone renderer with basic pipeline support, using its Maya translator and geometry procedurals as input. We produced many review sequences, replacing both OpenGL playblasts and low-quality *PRMan* setups.

8.2 Putting the Pieces Together

By the end of the project we were able to use *PRMan* REYES for primary visibility and shading and Glimpse's ray-tracing for all other shading and light transport. Some final shots of *The LEGO Movie* were even rendered completely with Glimpse. This remarkable outcome was the coincidence of a number of key factors. *PRMan's* extensibility gave us enough plugin and customization opportunities to gradually expand Glimpse's capabilities, with increasing benefits at each step. The broad industry shift to physically-based global illumination and path tracing methods was supported by openly available high quality resources such as PBRT, Embree, OpenShadingLanguage and OpenVDB, along with a collaborative research culture. Finally, *The LEGO Movie* was embarrassingly suitable to exploit instancing at a massive scale with minimal functional requirements (mesh primitives with a single plastic material); our building block was a building block.

As a production, we had also reconfigured to create visual richness through aggregation and interaction rather than specialization. This principle can be seen in a number of domains. We were creating a library of small, simple bricks and their matching textures to be later assembled into a variety of models rather than building complex single-purpose models. In the shading and lighting realm, we were able to use simple physical materials and shading networks with light transport providing emergent complexity.

Many systems beyond rendering were affected by these changes in technique. Tools and workflows for creating, managing and exploiting the brick library and the additional layer of asset interdependencies were necessary. Production schedules and bidding needed to reflect work done against these elements in addition to the final shot assets. Finer grained assets resulted in larger numbers of assets and asset relationships, placing pressure on asset management and tracking systems.

8.3 Lost in the Weeds?

Having bootstrapped Glimpse from interactive preview renderer to , we contemplated the requirements of our next animated feature. At the time, *The LEGO Ninjago Movie* was next in our schedule with design featuring *LEGO* characters and sets amidst natural environments with dense vegetation.

Realising the potential, we committed to Glimpse as our facility renderer. If we were to completely control our renderer, we would "render everywhere". We would exploit the core technology for interactive, review and final rendering. We would integrate the renderer and its algorithms in interesting and unusual ways. But we would strive to maintain the speed and minimal configuration that made Glimpse compelling. This specialisation would be essential to keeping Glimpse competitive in a very active field of commercial renderers.

8.4 Glimpse Today

In its current form, Glimpse is a high performance rendering system with an extensive plugin architecture. In addition to the ray tracing and rendering core, the system includes an expressive, editable scene description which is used for high performance interactive interfaces to DCC applications, procedurals and geometry caches as shown in Figure 14.

The core production-rendering configuration is as a unidirectional path tracer, using adaptive sampling followed by a denoise process. In this setting, we have implemented some novel integration strategies to address particular light transport scenarios.

The production feature checklist:

- Unidirectional + next-event estimation CPU path tracer
- MIS, Russian roulette
- Adaptive subdivision surfaces
- Implicit curve primitives
- Volumes with emissive sampling
- OSL-based shading

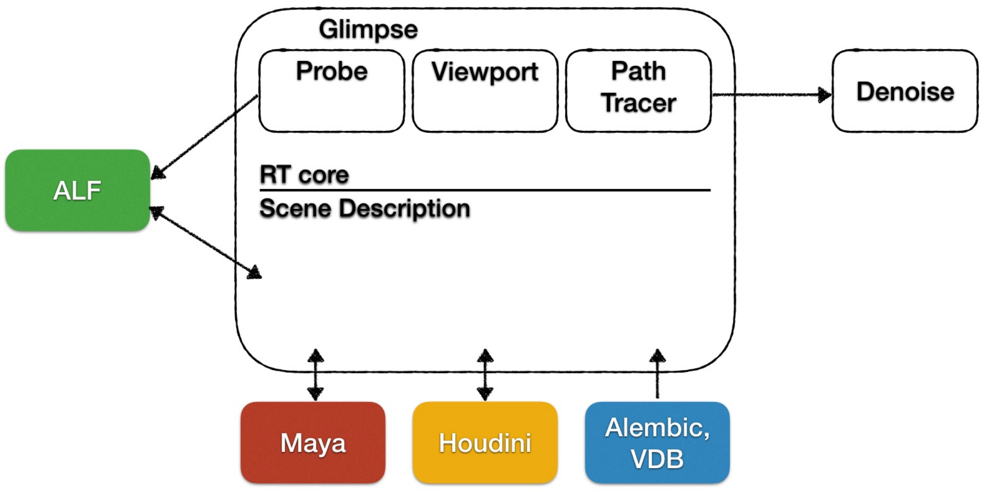

Figure 14: Glimpse architecture

Less typical features:

- Scene description
- Ash material composition
- Adaptive sampling
- Denoise
- ERDoF integrator for defocus
- Prelit integration
- Light culling

8.4.1 Scene Description

Glimpse's origins as an interactive renderer led to a expressive, fine-grained scene description closely bound to the ray tracing core with high performance incremental updating. These capabilities were extended to a complete, stand-alone scene description for efficient and scalable asset representation, with support for rich hierarchical state, procedurals and read/write access for clients such as DCC host applications or pipeline and workflow tools as shown in Figure 14.

One significant client of Glimpse's scene description is ALF, Animal Logic's procedural evaluation engine. As well as providing scene data for rendering fur, vegetation and other procedural geometry, ALF can also sample the full render scene including subdivision and displacement using a special probe rendering plugin. This is used for high accuracy procedural surface coverage across entire sets, such as moss.

The combined use of hierarchical state and unrestricted instancing in a scene description yields expressive and scalable representation of complexity which is very suitable for path tracing. However assembly of large scene from many components, as is common in production, can lead to the difficult situation where a particular object instance can not be specifically edited without a change in the scene topology. For example, Figure 15 shows a case where a change to any scene node will affect more than one final instance. Glimpse allows expressive, hierarchical edits even in these cases by supporting path-based instance overrides.

8.4.2 Ash Material Composition

Glimpse offers a hierarchical material composition mechanism which supports both mask-based material layering and intra-shader component composition to be expressed in the scene hierarchy. A practical

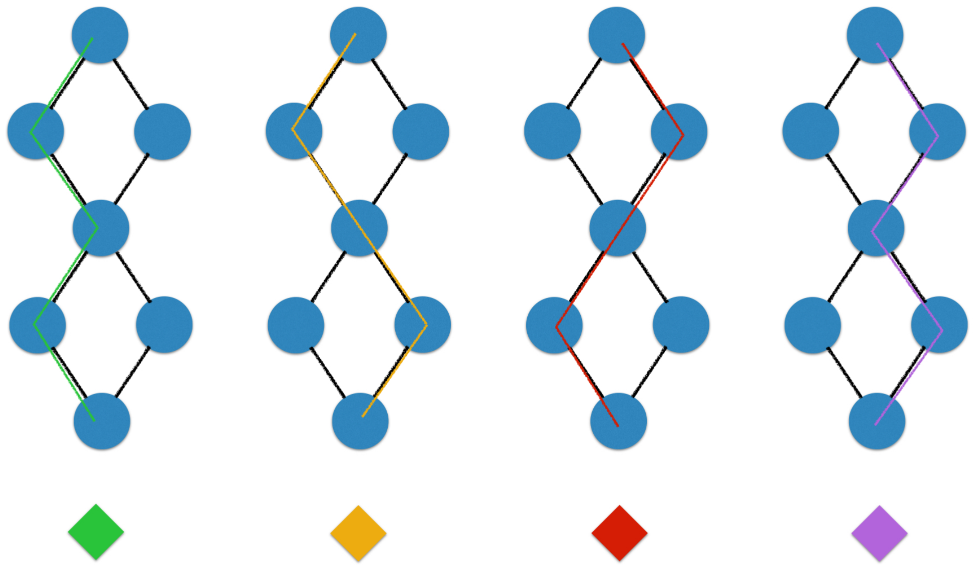

Figure 15: Path-based Instance Overrides

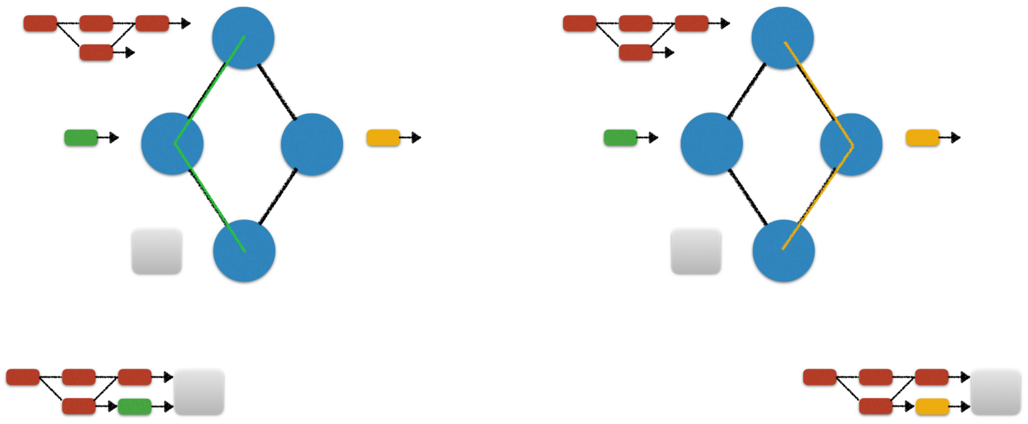

Figure 16: Hierarchical material composition

application of the first mechanism is to add a layer of dust across all objects in the scene by assigning a new material layer to the scene root. The second mechanism allows for flexible and modular material authoring similar to coshaders from *PRMan's* RSL2, without the explicit interface requirements or shading-time performance penalties of that mechanism. Figure 16 illustrates a simple case where edits or additions to a material are collected as the scene hierarchy is traversed to statically produce specialised, composed shaders.

8.4.3 Global Sampling Controls

With the knowledge that variance in Monte-Carlo rendering is scene configuration dependent, Glimpse avoids per-object sampling controls such as super-sampling or splitting rates in favour of global, adaptive controls. Whilst effective in expert hands, fine-grained sampling controls must be tuned spatially and temporally for best results and such specialised setups can not be easily transferred or reused in different scenes. We rely on adaptive primary sampling to refine each part of the image to a target variance, and then deploy a feature-buffer-assisted denoising algorithm.

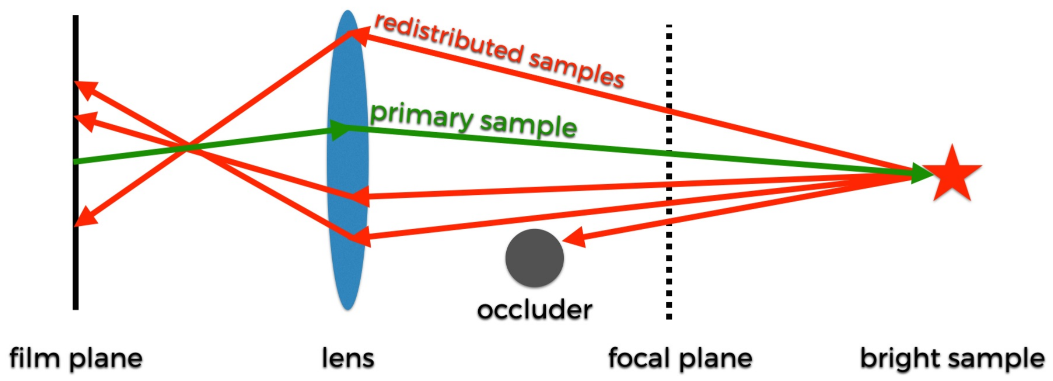

Figure 17: Energy Redistribution Depth of Field

8.4.4 Energy Redistribution Depth-of-Field

Following our "render everywhere" principle, we hoped to simulate lens and depth-of-field effects using an in-render model, rather than 2d approximations. This avoids artefacts from missing data around edges and occluded regions. Avoiding a post-render approach allows depth of field to be immediately and consistently seen in all renders. Two significant problems arise from the choice to capture depth-of-field in-render: firstly it implies that any change in DoF will cause re-rendering, and secondly, pathological sampling cases arise with consequent noise and slow convergence.

Our hope was that by having high quality renders including DoF and other critical lens effects such as distortion available early in the pipeline for layout and lensing work we could lock off these details in most cases. We would also be able to work and review with renders closer to the final frames throughout the pipeline, with obvious benefits of avoiding work that would be blurred or clipped. This proved to be the case for *The LEGO Batman Movie*, where less rework was required for late, creative DoF changes than the mundane work we had to perform for *The LEGO Movie* to re-render and finesse artefacts in compositing-based 2D DoF using deep image inputs.

To address poor sampling from unidirectional path tracing in the common scenario of out-of-focus highlights on specular objects we developed a novel, specialised method, *Energy Redistribution Depth of Field* Heckenberg et al. [2017], inspired by Energy Redistribution Path Tracing in Cline et al. [2005]. In the process of normal path tracing, we record samples with high intensity in out-of-focus regions and then perform a second pass to redistribute these high energy samples by tracing multiple rays (splatting) back through the scene and lens, as depicted in Figure 17. This yields much smoother defocus highlights (with the shape of the lens aperture) and dramatically reduces variance in cases of bright, out-of-focus details as often occur with shiny *LEGO* bricks. The method is unbiased and compatible with our adaptive sampling scheme.

8.4.5 Creature Features

Another unusual feature in our path-tracing integrator is the ability for a set of objects and materials which have already been illuminated ("prelit") to further participate in global illumination with the rest of the scene. This allows high-quality integration of live-action plates with CG elements and sets. We hope to publish details of this method in future.

8.5 Render Anywhere Else?

In addition to final rendering, we also exploit Glimpse in a number of other modes. A Maya viewport renderer plugin allows us to ray trace our render geometry, depth composited with OpenGL geometry and UI elements. Probe rendering interfaces allow our procedural geometry tools to query the render scene for perfect registration in instance placement. Non-camera based ray tracing is also used for our baking and transfer tools.

8.6 Reflections

The change to our rendering approach has had a drastic impact on our productions, in all parts of the facility. Interactivity, speed and simple, robust configuration have led to more efficient workflows. Less technical rendering setups, coupled with more robust physically-based light transport and shading enabled automation of high quality rendering for reviews and much wider sharing of lighting setups for final rendering.

In-house development of Glimpse has brought many benefits:

- more efficient workflow:
  - interactivity
- more robust lighting
  - automated renders
  - consistent rendering requirements (frame to frame)
  - “same-as” workflows across shots
- opportunities
  - customised anything / everything
    - ERDoF integration
    - “prelit” integration
  - integrated and optimised solutions

There have also been significant challenges:

- new lighting techniques required
- less direct/image-related controls for lighting
- rendering developers are hard to find. (ps we’re hiring!)

8.7 Current and Future Challenges

Unidirectional path tracing with next event estimation is the workhorse integration scheme for production rendering for good reasons: it is simple, controllable, scalable and performs well in a wide range of scene configurations. However, there are common and important scenarios where it fails and can not be easily boosted with auxiliary setup: prominent examples being interior lighting, caustics including eyes and sub-surface light transport. A number of “extended” next event estimation schemes have been recently developed which specifically address some of these cases with scene-local multiple-bounce-light connections, e.g. manifold next event estimation for caustics in Hanika et al. [2015] and subdivision next event estimation in Koerner et al. [2016] for sub-subsurface light transport. We expect to continue to use unidirectional path tracing with these and other extensions including ERDoF, possibly through the generalisations suggested in Tessari et al. [2017].

Material models continue to evolve but significant improvements can still be made to robustly produce photorealistic images of the elements central to production rendering: hair, fur and skin. Interestingly, these cases all involve challenges of representation (shape and details of the geometric primitives and their surface and internal structure) as well as practical problems for brute-force path tracing, due to high-order and fine scale light transport effects. Finding artistically-controllable models and efficient implementations for these effects remains a core problem.

Robust and controllable coatings and layered materials are also an elusive goal. Active as a research and practical production problem, this is a key element to making simple material components that can be combined in flexible ways with predictable and realistic results.

8.7.1 Scale and Scalability

Representation of physical objects in current production practice typically involves authoring in a number of different domains and physical scales, with different primitives:

- mesh model
- subdivision (possibly with refinement controls such as creases, hierarchical edits)
- displacement via shading
- bump via shading
- roughness via shading

With most current models and rendering methods, these representations are neither scale invariant, nor easily convertible (e.g. model detail and displacement to shader roughness). Furthermore, different tools, skills and indeed artists are usually involved in the authoring of these layers. Thus objects and materials must be authored for particular screen-scale usage, or manually and extensively re-authored for use at other scales unless appearance change is acceptable.

Some valuable research is being undertaken into alternative surface and material models (e.g. micro flake distributions in Heitz et al. [2015]) and in automated level-of-detail conversion for real-time games and virtual-reality asset generation which we hope may yield progress in these areas.

8.7.2 Frame Rendering in Production

Path tracing over animated sequences offers enormous opportunities to exploit "embarrassing parallelism" at the expense of not sharing information to reduce total work. Almost all production path tracing chooses to treat frames and the pixels and often samples within them as completely independent to maximise parallelism, rather than to couple them to reduce overall computation. This is in contrast to other rendering methods which explicitly factor rendering into reusable, shareable caches such as rasterisation with shadow maps or point-based global illumination.

Some practical techniques have begun to show the potential of sharing information at the expense of parallelism in order to minimize overall work for path tracing. Adaptive rendering is one simple approach to measure samples together, within a pixel or image region, to achieve a variance target. In this case, only sample generation, batching and filtering are affected, in order to locally adjust sample counts to measured variance in a pixel or image region. Denoising also represents a way to couple samples across screen space and time in order to reduce overall rendering work, with limited impact on rendering parallelism as it may be applied in a separate process after rendering.

These techniques only begin to capture the very significant amount of redundancy that is intuitively present in production sequence rendering, without re-introducing the scale limitations, bias, scheduling and configuration complexity associated with explicit caching approaches from production rasterisation methods.

There is promising research to adapt or learn for improved path tracing that we hope may be applicable to production rendering, under the banner of "adaptive" or more-recently "learning" approaches. Adaptive importance sampling Cline et al. [2008] and Vorba et al. [2014] has long been explored within individual frames. Extensions of these techniques across frames of a sequence or repeated renders of similar sequences would be very compelling in a production environment.

At a lower-level, individual frame rendering typically involves a significant amount of start-up time as scene data, models and shaders are loaded and processed for efficient rendering. Allowing for incremental update of this state for renders within or across sequences would offer significant advantages, particularly in distributed rendering scenarios where data transfer has a relatively high cost.

References

David Cline, Daniel Adams, and Parris Egbert. 2008. Table-driven Adaptive Importance Sampling. In *Proceedings of the Nineteenth Eurographics Conference on Rendering (EGSR '08)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1115–1123. https://doi.org/10.1111/j.1467-8659.2008.01249.x

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy Redistribution Path Tracing. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05)*. ACM, New York, NY, USA, 1186–1195. https://doi.org/10.1145/1186822.1073330

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

Daniel Heckenberg, Luke Emrose, Matthew Reid, Michael Balzer, Antoine Roille, and Max Liani. 2017. 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

Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX Microflake Distribution. *ACM Trans. Graph.* 34, 4, Article 48 (July 2015), 11 pages. https://doi.org/10.1145/2766988

David Koerner, Jan Novák, Peter Kutz, Ralf Habel, and Wojciech Jarosz. 2016. Subdivision Next-event Estimation for Path-traced Subsurface Scattering. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations (EGSR '16)*. Eurographics Association, Goslar Germany, Germany, 91–96. https://doi.org/10.2312/sre.20161214

Bryan Smith, Daniel Heckenberg, and Jean-Pascal leBlanc. 2014. The LEGO Movie: Bricks, Bricks and More Bricks. In *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. ACM, New York, NY, USA, Article 15, 1 pages. https://doi.org/10.1145/2614106.2614179

Lorenzo Tessari, Johannes Hanika, and Carsten Dachsbacher. 2017. Local Quasi-Monte Carlo Exploration. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations.*

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

9 ILM's Path to Tracing

André Mazzone, *Industrial Light & Magic*

9.1 Introduction

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

9.2 History of Techniques

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

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

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

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

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

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

9.3 What path-tracing Is

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

9.3.1 Ray-tracing

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

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

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

9.3.2 Ray-tracing with a Difference

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

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

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

9.3.3 Conceptually Simple

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

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

9.3.4 Extensible

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

9.4 What Path-tracing is Not

9.4.1 Not New

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

9.4.2 Not Always Path-Tracing

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

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

9.4.3 Not Always Fast

There are many reasons why path-tracing, especially when performed progressively, might not be the most efficient rendering framework in terms of total render time. Distributing rays across an entire image for every iteration does not allow the efficiencies, available in modern CPUs, that come from coherent memory access. Similarly, spawning a single path at each intersection, as opposed to a distribution, also discards coherence. Similarly, achieving high quality (converged) results often requires lengthy computation. In particular, Monte-Carlo integrators require four times as many samples to halve the variance. Production rendering budgets rarely allow fully converged renders to be computed. Other noise reduction strategies can be employed including post-render filtering (denoising).

9.4.4 Not Always Memory efficient

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

9.4.5 Not Cheat-Free

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

9.4.6 Not Simple to Implement

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

9.4.7 Not Always Simple for the Artist

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

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

9.4.8 Not Enough

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

9.4.9 Not Mature

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

9.5 Conclusion

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