# Path tracing in Production

## *Part 2: Making movies*


Wenzel Jakob
*EPFL*

Andrea Weidlich
*Weta Digital*

Andrew Beddini
*Blue Sky Studios*

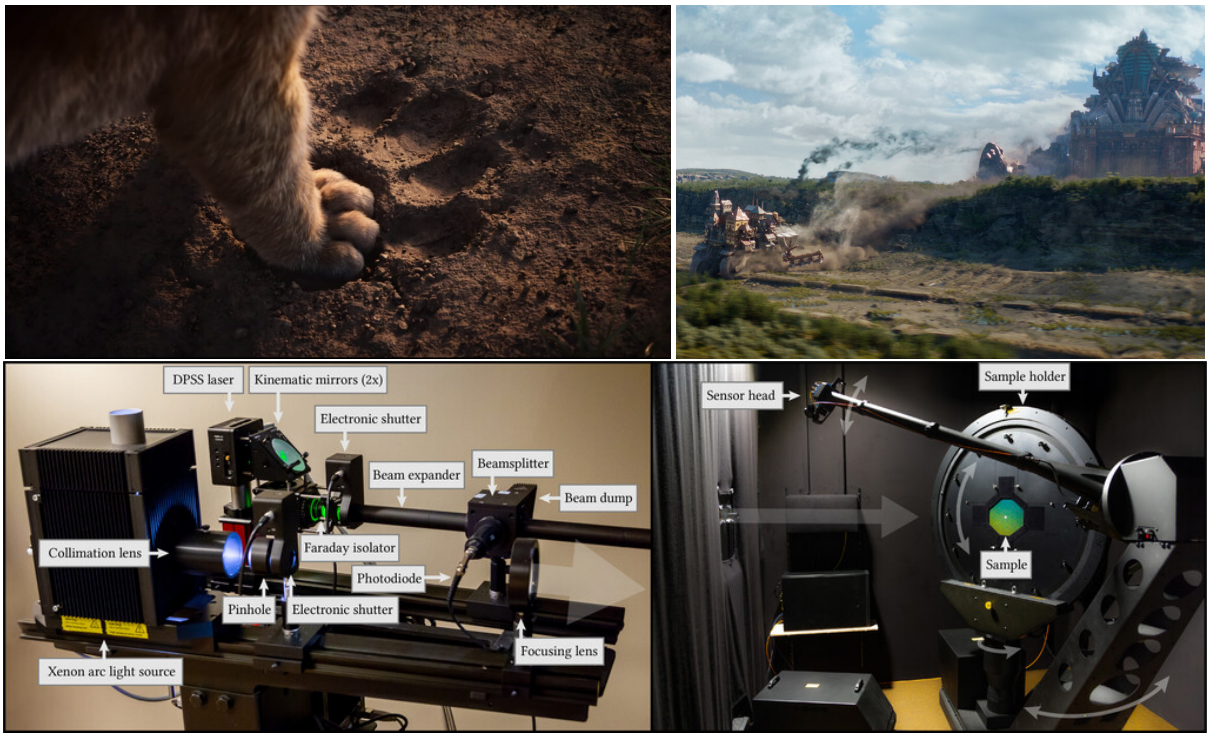
Rob Pieké
*MPC Film*

Hanzhi Tang
*Digital Domain*

Luca Fascione (Organizer)
*Weta Digital*

Johannes Hanika (Organizer)
*Weta Digital*


*SIGGRAPH 2019 Course Notes*




*Top row: two images from recent movie productions, showcasing difficult light transport in fur and dust, as well as high geometric complexity, atmospherics and volumetric scattering. Left:* The Lion King*, image courtesy of MPC Film, ©2019 Disney. All rights reserved. Right:* Mortal Engines *©2018 Universal Studios. All rights reserved. Bottom: Picture of* the machine *from* An Adaptive Parameterization for Efficient Material Acquisition and Rendering *by Jonathan Dupuy and Wenzel Jakob as appeared in* Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018).



SIGGRAPH '19 Courses, July 28 - August 01, 2019, Los Angeles, CA, USA
ACM 978-1-4503-6307-5/19/07.
10.1145/3305366.3328085

## 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 trade-offs 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 background and implementation, the second one focuses on material acquisition and modeling, GPU rendering, and pipeline evolution.*

# Contents

# 1 Objectives

In the past few years the movie industry has switched over from stochastic rasterisation approaches to using physically based light transport simulation: path tracing in production has become ubiquitous across studios. The new approach came with undisputed advantages such as consistent lighting, progressive previews, and fresh code bases. But also abandoning 30 years of experience meant some hard cuts affecting all stages such as lighting, look development, geometric modelling, scene description formats, the way we schedule for multi-threading, just to name a few. This means there is a rich set of people involved and as an expert in one of the aspects it is easy to lose track of the big picture.

This is part II of a full-day course, and it focuses on a number of case studies from recent productions at different facilities, as well as recent development in material modeling and capturing. The presenters will will showcase practical efforts from recent shows spanning the range from photoreal to feature animation work, pointing out unexpected challenges encountered in new shows and unsolved problems as well as room for future work wherever appropriate.

This complements part I of the course, where context was provided for everybody interested in understanding the challenges behind writing renderers intended for movie production work, with a focus on new students and academic researchers. On the other side we will lay a solid mathematical foundation to develop new ideas to solve problems in this context.

# 2 Syllabus

## 2.1 14:00 — Opening statement (Luca Fascione, 15 min)

A short introduction to the topics in this session and to our speakers.

## 2.2 14:15 — Capturing and rendering the world of materials (Wenzel Jakob, 30 min)

One of the key ingredients of any realistic rendering system is a description of the way in which light interacts with objects, typically modeled via the *Bidirectional Reflectance Distribution Function* (BRDF). Unfortunately, real-world BRDF data remains extremely scarce due to the difficulty of acquiring it: a BRDF measurement requires scanning a four-dimensional domain at high resolution–an infeasibly time-consuming process.

In this talk, Wenzel will showcase the ongoing work at *EPFL* on assembling a large library of materials including metals, fabrics and organic substances like wood or plant leaves. The key idea to work around the curse of dimensionality is an adaptive parameterization, which automatically warps the 4D space so that most of the volume maps to "interesting" regions. Starting with a review of BRDF models and microfacet theory, Wenzel will explain the new model, as well as the optical measurement apparatus used to conduct the measurements.

## 2.3 14:45 — Production quality materials (Andrea Weidlich, 30 min)

Recent film productions like *Mortal Engines* or *Alita: Battle Angle* exhibit an unprecedented visual richness that was unthinkable ten years ago. One key component to achieve this is a flexible but expressive material system that is capable of reproducing the complexity of real-world materials but is still simple enough so that it can be used on a large scale. Andrea will talk about material modeling in a production path tracer in general and the constraints that come about when artistically driven decisions meet a physically plausible world. She will demonstrate how a modern layer-based material system as it can be found in Weta Digital's in-house renderer Manuka influences design and look development decisions, and give examples of how it is used in production.

## 2.4 15:15 — Break (15 min)

## 2.5 15:30 — "Everything the Light Touches" – Rendering The Lion King (Rob Pieké, 30 min)

Not long after the success of *Disney*'s *The Jungle Book*, *MPC Film* began work on the retelling of another *Disney* classic: *The Lion King*. The mandate for this project was to bring realistic environments and documentary-style cinematography to the screen, requiring improvements across the board to our rendering-related technology, workflows and pipelines. In this talk, Rob will outline some of the changes to MPC's fur rendering, improvements in outdoor environment rendering efficiency, advancements to deep image workflows and more.

## 2.6 16:00 — Introduction to GPU production path tracing at Digital Domain (Hanzhi Tang, 30 min)

Starting in 2016 *Digital Domain* has been testing *GPU* rendering, trying to see how it would integrate into the production rendering pipeline smoothly. Starting from initial qualitative tests to widespread use on *Avengers: Infinity War* to final production renders on *Captain Marvel*, *Digital Domain* built a robust *GPU* rendering option that sits alongside the main CPU rendering pipeline. Hanzhi Tang will present the development challenges of both hardware and software that were encountered in this implementation of this new renderer.

## 2.7 16:30 — Q&A with all presenters (15 min)

# 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, and he has a movie credit for *Abraham Lincoln: Vampire Hunter*.

# 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 Hanzhi Tang, Digital Domain

Hanzhi Tang is a Digital Effects Supervisor and Head of Lighting at *Digital Domain*. He received an MSc in Physics from Imperial College, University of London and has been with *Digital Domain* for 16 years working with many different production renderers on 23 feature films. He also recently became a member of the Academy of Motion Picture Arts and Sciences and has previously been a Visual Effects Society Awards nominee. He has most recently completed visual effects on *Captain Marvel* from *Marvel Studios*.

## 4.3 Wenzel Jakob, EPFL

Wenzel Jakob is an assistant professor at *EPFL*'s *School of Computer and Communication Sciences*, where he leads the *Realistic Graphics Lab* (https://rgl.epfl.ch/). His research interests revolve around material appearance modeling, rendering algorithms, and the high-dimensional geometry of light paths. Wenzel Jakob is also the lead developer of the *Mitsuba* renderer, a research-oriented rendering system, and one of the authors of the third edition of the book *Physically Based Rendering: From Theory To Implementation*. (http://pbrt.org/)

## 4.4 Andrea Weidlich, Weta Digital

Andrea Weidlich is a Senior Researcher at *Weta Digital* where she is responsible for the material system attached to *Weta*'s proprietary physically-based renderer, *Manuka*. Andrea grew up in Vienna, Austria, where she studied technical computer science, and later focused on computer graphics. After finishing her Ph.D. for predicting the appearance of crystals she switched to the private sector and moved to Munich to work for the automotive industry. Her continuing work on *Manuka* allows *Weta* to produce highly complex images with unprecedented fidelity. Her main research areas are appearance modeling and material prototyping. Andrea holds a Master of Arts in Applied Media from the *University of Applied Arts*, Vienna and a Ph.D. in Computer Science from the *Vienna University of Technology*.

# 5 ``Everything the Light Touches" -- Rendering The Lion King

Rob Pieké, *MPC*

Not long after the success of Disney's *The Jungle Book* (2016), MPC Film began work on the retelling of another Disney classic: *The Lion King* (2019). The mandate for this project was to bring realistic environments and documentary-style cinematography to the screen, requiring improvements across the board to our rendering-related technology, workflows and pipelines.

A large investment was made in revisiting the way we render fur. We drew inspiration from recent publications on fur shading, both from academic research and from other studios, generally involving looking at the components of hair at a microscopic level, and considering the main differences between long human hair and short animal fur. We overhauled our in-house shader to support longitudinal and azimuthal roughness - accentuating the cylindrical shape of hair strands - and introduced new shading lobes to simulate the scattering in a medulla core.

Reflecting on challenges during *The Jungle Book* (2016), we also invested effort into making the fur shading as robust and easy to control as possible. Fur colour is now parameterised based on melanin concentration, appropriately rich and saturated in areas of high fur density, and paler and muted in areas of sparse fur. We also ensured that the fur shader is fully energy-conserving across its now seven lobes, allowing LookDev artists to be confident that their work would appear beautiful and consistent across the wide range of lighting conditions found in the film.

The environments of *The Lion King* (2019) similarly increased in visual complexity and fidelity, with a general shift away from matte painting backgrounds and towards 3D geometry all the way to the horizon. To combat the increase in resources required to process all this data, we extended our shot-based culling and automatic LOD system to more aggressively remove geometry which would not meaningfully contribute to the final image. In addition to considering the camera frustum, we also began considering occlusion (removing, for example, pebbles hidden behind a big boulder).

Increases in complexity found their way into the data footprint for our rendered images, with more and more of our workflows being linked to deep image compositing. Leveraging internal work presented at last year's SIGGRAPH conference, we were able to achieve a 55-75% reduction in the size of our deep images. This reduced not only the impact on our file system but, also, on our global sync queue and on the processing time for our compositing scripts in *Nuke* (with several nodes having execution times linked directly to the number of deep samples). We also adopted internal work presented at last year's *DigiPro* conference on combining colour from 2D images with deep opacity samples.

The entire project was managed less on a shot-by-shot basis than previous films. We introduced new technology to operate more holistically across the show and the sequences. This reduced the complexity in propagating lighting changes across shots, keeping a high-level of consistency even when many artists were working in parallel.

# 6 Introduction of GPU production path tracing at Digital Domain

Hanzhi Tang, *Digital Domain*

This session is about how Digital Domain implemented GPU rendering in a conservative approach designed to make a gradual transition of hardware and software possible in a production friendly way.

## 6.1 Introduction to rendering at Digital Domain

Digital Domain has been rendering feature films for twenty five years. In that time we have, at some point, used every major renderer. Starting as a heavily Houdini-Renderman centric effects house we have over the course of the years also used Mental Ray, V-Ray, Arnold and Mantra. We are no strangers to experimenting. Pushing towards path tracing has been a constant goal since 2007 with the movie *Speed Racer* and since 2010 on *Tron: Legacy* we have settled on V-Ray as our main workhorse renderer in lighting and Mantra in effects.

So it goes without saying that as GPU rendering comes to the forefront we would take a new look at what is available and evaluate what type of investment of time and money it would take to make it viable.

## 6.2 Evolution not Revolution

If one were to start up a new studio today you would probably build it with GPU rendering in mind. However as with most established visual effects studios the train is always in motion and trying to introduce new components to the pipeline can be hazardous. There is never a good time to stop everything and make a change and there is always a fear of the unknown quantities that will be introduced when you have multiple projects each at different stages of production. Some of those reasons for Digital Domain would be:

- Financial cost
- Human cost on support resources (technical director time etc.)
- Adequate testing time
- Unforeseen software limitations
- Training time for artists and TDs to learn the features and limitations
- Catastrophic production stoppage

### 6.2.1 Testing in the Sandbox

Initial testing was very limited in scale. In October of 2015 we first checked what the current state of GPU renderers were on the market and if they were designed with production use in mind. OTOY OctaneRender and Redshift were the most prominent in the market at the time. Keep in mind that RTX features in NVIDIA Turing architecture and the fruition of real-time ray tracing on the GPU was still a few years away.

We started tests on some projects post-delivery and kept it in a sandbox but this allowed us to use real production assets and data that we could use for feature and quality comparison. Redshift quickly became the front runner because its design and workflows were already aligned with visual effects production. It's important to mention that we did have experience with V-Ray RT GPU over the years but its features did not match our intended use in 2015.

Our initial results indicated that with GPUs we could get path traced images with the level of noise that would take 3x to 10x longer on a CPU, depending on content and shading complexity. This meant that if we had a relatively simple use case we could leverage the most acceleration out of the GPU. This is what led us to consider Redshift for animation renders as it's first use case.

### 6.2.2 Animation Renders

The next stage after clearing basic texture and lookdev requirements we moved on to tests that required some integration into our renderfarm. We coincidentally had a concert project that needed to render 10,000 frames

**Table 1:** *Timeline of GPU Rendering Rollout*

| Year | | Project Name | Render Type |
|---|---|---|---|
| 2015 | Oct | LG Commercial | Initial tests |
| | Nov | *Black Sails* | Comparison renders |
| 2016 | Mar | Concert Hologram | Animation and final renders |
| 2017 | Feb | *Avengers: Infinity War* | Initial Thanos animation render tests |
| | Mar | *Spiderman: Homecoming* | Animation renders |
| | May | *Thor: Ragnarok* | Animation renders |
| 2018 | Mar | *Ant-Man and the Wasp* | Animation renders |
| | May | *Call of Duty: Black Ops* | Animation renders |
| | Jun | *Avengers: Endgame* | Animation renders |
| | Jul | *Captain Marvel* | Final renders on two sequences |
| 2019 | Apr | Select projects in production | Final renders |

a night per song just to review performance and at the same time a need arose for animation renders using very heavy assets for *Spiderman: Homecoming*.

These projects combined gave us an opportunity to really test the technology in a contained way. Neither would be mission critical to the final image on the screen but it would still have to perform well to meet strict deadlines. Animation renders are also a safe place to test because they are mostly for work-in-progress reviews and have a different set of expectations for image content. Previously animation renders have ranged from Maya playblasts, Maya software renders or some form of simple colored or lightly textured proxy.

In today's visual effects industry, the level of pre-visualisation (previz) animation quality has risen very fast. The quality of lighting and shading now used for previz means that when directors are reviewing the progress of their movie it's very jarring to jump from textured and lit previz back to overly simple looking animation renders from the VFX vendor. As previz and virtual productions continue to upgrade their rendering technologies and game engine graphics become ever more sophisticated, further pressure will continue to ripple down the line to visual effects houses to raise the bar.

### 6.2.3 Immediate benefits

We found right away that by using GPU rendering we could produce good looking animation renders that provided these desirable features:

- Image based lighting and global illumination (GI)
- Motion blur
- Full resolution production textures
- Render times of the order of several minutes with very low GI noise

Additional advanced features to be introduced later:

- Full hair rendering for animation approval
- Subsurface skin shading for characters

Starting in 2017 GPU animation rendering (animrender) had become very successful for us and provided some degree of confidence in the renderer. We had pushed assets that contained hundreds of texture tiles and very dense heavy models through our animrender pipeline with relative ease, something we had struggled with when using global illumination with a CPU path tracer whilst simultaneously trying to keep render times under a 5 minute per frame cap and aiming for low noise levels at a 2K image resolution.

*Avengers: Infinity War* was a very important CG character show for us and we wanted to provide the highest quality animation render so that the filmmakers could have confidence in what the final performance would look like with lighting and shading that was more representative of the final image.

When *Thor: Ragnarok* went into production we found ourselves having to tackle a new problem. Our animation renders needed a fully instanced city made of hundreds of buildings that we would fly through at

high speed. It was a good test of the impact of instanced geometry on Redshift and we stood back and waited for the renders to drop in performance but it didn't and that became the next confidence building block in a future potential pipeline.

2018 became a big leap for us for GPU rendering, *Avengers: Infinity War* had proven itself on the animation front and it's use in that department continued unabated in *Avengers: Endgame* and we started to dabble with the idea of using it in final production rendering. We were just waiting for a show to try it on that had the right risk-reward profile.

In July of 2018 we began the visual effects for *Captain Marvel*, whilst the character transformation effects would be a complex mix of our traditional CPU renderers in V-Ray and Mantra we considered the use of GPU rendering for other parts of our work. There was a sequence inside a spaceship hangar which we knew would run into the usual noise and sampling issues with multiple bounces of light from over a hundred small light sources. Another sequence set in a desert canyon provided a second environment that would also be a prime candidate for GPU rendering. Both would be a good challenge for a GPU renderer because of the amount of indirect bounced light. The hard surface lookdev was something that was achievable in any physically based shader driven by on roughness and refractive index values and there was no hair and skin to worry about. If results were disappointing we could still back out to our normal pipeline easily. To make the leap from the relative safety of animation renders to final renders we had to resolve some remaining feature requests that would be required before attempting final production use.

## 6.3 Software Challenges

To integrate a GPU renderer into our pipeline we had basic and advanced features we needed to implement. The basic requirements were all we needed for animation renders but when we turned to final renders we needed a good deal of further development.

### 6.3.1 Basic Requirements

To start, we needed an ability to launch and manage renders on our renderfarm. This was a relatively easy addition to our in-house layer manager Atomic, requiring the writing of a new python module to support new renderglobal settings and other render specific calls.

Next we needed to add support to our texture publishing pipeline to produce a Redshift specific format (.rstexbin) that was designed for more efficient GPU access. It's possible to use exr and tif directly, but it still requires on-the-fly translation which would add overhead to the render start-up time. Also part of the texture pipeline development was to make sure we correctly supported linear and gamma encoded images so that our *ACEScg* color pipeline was correctly preserved all the way through to the renders and matched any other products from the publish (e.g `.tx`, `.tif`, `.rat`).

Other publish tools we needed to support for Redshift were material publishing and light rig publishing. These would round out our basic needs for animation renders.

### 6.3.2 Advanced Development

Attempting final renders means meeting the current standards for rendered image outputs going to compositing. This is a fairly high bar that wasn't able to be met until significant software revisions had been completed both on our part and on the part of the Redshift development team throughout 2017 and 2018 (v2.5.x through v2.6.x)

A fully featured production renderer nowadays has to satisfy much more than producing physically realistic images. It also has to output a large number of arbitrary output variables (AOVs) that we have become dependent on for quick fixes as well as complex depth compositing. Deep Opacity and Cryptomatte support are also now commonplace and an expected standard output. Making sure we were able to get a 1:1 feature match with our existing renders was the primary goal, matching everything from naming conventions to value ranges in outputs. Ideally, to a compositor the GPU rendered output should be almost indistinguishable from the output from our other renderers.

Some of Digital Domain's proprietary software also needed to be supported by this new renderer. Our in-house hair and grooming system, Samson, would need a new procedural geometry plugin written. At the same time we needed to write our own alembic procedural plugin for Redshift so that our deferred rendering pipeline was supported.

Our in-house queuing system, *RACE*, was also rewritten with an important upgrade. We could now allocate CPU and GPU jobs concurrently on the same machine. No longer would half the CPU cores sit idle while the GPU did the work. We could extract the maximum performance from each machine, important as machines can now have over twenty cores.

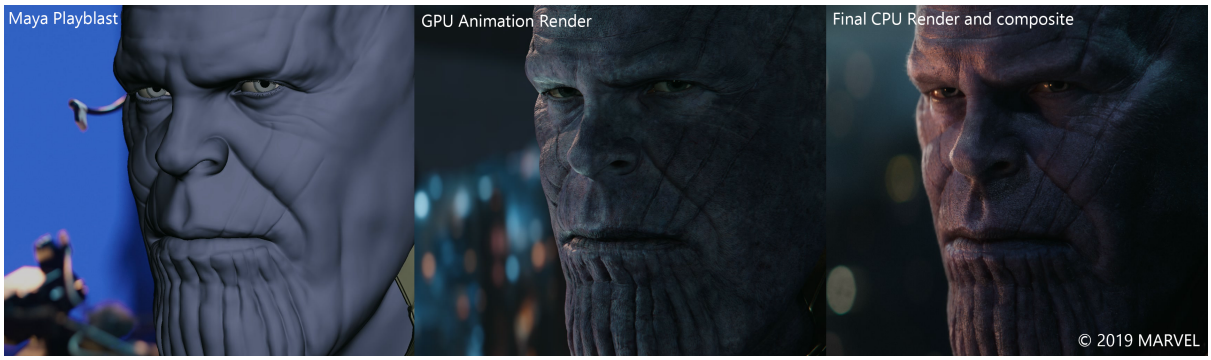



**Figure 1:** *Comparison of Thanos renders from animation to final in Avengers:Infinity War. ©2019 Marvel. All Rights Reserved.*

## 6.4 Hardware

Bringing GPU rendering to the pipeline necessarily means a paradigm shift in the hardware we use. We had to take a hard look at the resources we had at our disposal and see if we had existing infrastructure that could support this new direction in rendering. Infrastructure spending on this scale tends to only happen in 5 or even 10 year cycles.

### 6.4.1 Rendering with artists' desktops

When we began looking at GPU rendering in 2015, the current state-of-the-art NVIDIA hardware was based around the Maxwell architecture. Many of our desktop workstations were still using even older Kepler based Quadros (NVidia Quadro K4200).

Even with these older graphics processors we were able to see the big performance gains possible with GPU accelerated path tracing. We had of course seen the usefulness of GPU acceleration for many years in V-Ray RT GPU. However because it lived as a separate renderer and was restricted to the amount of VRAM on the graphics card, it was most useful for previewing lighting and as a fast feedback look development tool. The arrival of V-Ray Next with GPU support at the end of 2018 addresses many of those concerns and gives us more choices moving forward.

One of the main reasons Redshift caught our attention is because they had recognised those limitations and had implemented out-of-core (memory) rendering from the beginning. Performance may suffer when utilising system memory but the render is able to continue. It was an understated but essential difference that meant it could be used in an already complex visual effects pipeline. We couldn't afford to add yet another memory limitation to our already lengthy troubleshooting laundry list.

### 6.4.2 Building a GPU Render farm

Our render farm until now had been built entirely with headless (no GPU) dual Intel Xeon CPUs that are still ubiquitous in the industry with RAM and CPU cores being the yardstick for performance. This meant that any GPU rendering initially was going to have to run only on the GPUs in the artist workstations. Those workstations had always been part of the overnight renderfarm as extra capacity but now it would become

vital and at the same time a potential bottleneck to the whole process. The workstation GPUs were a scattered mix of three generations of NVIDIA hardware dating back as much as 10 years. Was this where the plan fell apart? We managed to complete two shows with what was available but it did lead to the question of scalability as demand quickly outpaced supply.

It was at this moment in April 2016 that Digital Domain was about to move to a centralised computing center in Portland, Oregon. After existing for six years with twin data centers in Vancouver, British Columbia and Los Angeles, California we were finally going to be on a unified file system and brand new remote workstations doing away with all local desktops. These 450 workstations came with NVIDIA Quadro P5000 graphics cards using the Pascal architecture. The timing couldn't have been better and the specifications were nicely uniform. For many artists we had leapfrogged two generations of NVIDIA chip designs, doubled the number of CUDA cores and doubled the clock speed. This would form the basis of our main GPU rendering pool for the next two years.

During the day we would get a subset of available GPUs and at night we could get the majority of desktops available for rendering on. Balancing this during the daytime and not interfering with artists work became a factor that needed to be thought about.

Looking forward to future purchases we will be looking at dual and multi GPU devices and building more GPU resources into the dedicated renderfarm. It must be noted though that multi-GPU performance on one frame does not always scale linearly after two GPUs and that further testing is required to optimise performance regarding bottlenecks in motherboards and current graphics bus limits (PCI Express x16)


**Figure 2:** *Avengers: Infinity War render comparison. The grayscale image represents how some shots were presented as a reminder that it was for animation discussion. ©2019 Marvel. All Rights Reserved.*

## 6.5 Production Lessons

From a technical standpoint moving to GPU path tracing and reducing render times seems like an obvious choice but it did sprout some interesting and some unforeseen problems along the way for productions.

### 6.5.1 The Learning Curve

First and foremost is having yet another renderer to support and learn. It often takes years to really know the ins and outs of a particular renderer. Fortunately with the advent of physically based rendering (PBR) the skills of an experienced lighter or material developer are more portable than ever. A PBR material is going to react


**Figure 3:** *A final render from Captain Marvel rendered on the GPU using Redshift for CG Canyon and Ships. ©2019 Marvel. All Rights Reserved.*

in roughly the same way when lit by the same illuminant in most modern path tracers. Optimising sampling and controlling noise sources is a universally applicable skill once mastered.

### 6.5.2 Being Prepared Earlier

When we started producing higher quality animation renders using final quality textures it raised a problem of asset state. When rendering during the animation phase of a production the assets are often still in flux, much more so than we are used to in lighting. This means that the rate of renders breaking from updated UVs, mismatched textures and completely new as-yet-unshaded geometry will happen with greater frequency and require specific maintenance just for animation. You end up needing dedicated animation render lighting TDs. This also puts more pressure on having textures ready in time for animation and extra artists to create basic materials for animation use. This changes the staffing priorities at the start of a show.

### 6.5.3 Setting a Look

With much more advanced lighting in animation you are providing a temporary but convincing lighting setup to the filmmakers that could possibly live in the edit of the movie for months before final lighting starts. This does sometimes set up a creative clash between the animation lighting setups and final lighting. Extra diplomacy between departments and clear communication with the client on expectations is required to navigate this. In future it would be more optimal for the lighting department to get involved sooner in the process and maybe take ownership of the animation lighting. It's possible this could be defined as far back as the virtual production or previz stage.

### 6.5.4 Possible confusion

We also found that once the animation lookdev is sufficiently advanced, those renders can become confusingly similar to the final renders. In some cases we were asked to turn the animation renders grayscale to indicate that it was for animation approval and not to diverge into lighting notes. This raises the idea of the two renders possibly blending into one at some point in the future. An alternate possibility is moving animation renders further along into a real time graphics engine, which at present would require a completely new pipeline that doesn't fit into existing tools.

## 6.6 Benefits of GPUs for other VFX processes

Despite whichever renderer you prefer to use currently, investing in GPU resources results in speedups in multiple areas so the hardware investment will never go to waste. We also use GPUs in these other areas of the business:

GPU acceleration is now built into so many tools like Nuke and Houdini that your compositing and effects department can also immediately get a benefit from a GPU enhanced render farm. Volumetric renderers like VortechsFX Eddy already occupy this space.

Beyond rendering we have new toolsets that utilise machine learning and deep learning (convolutional neural networks) that power technologies like our facial animation pipeline and AI driven technologies like denoising which make noise free path tracing a reachable goal. Both of these use the power of GPUs and their high number of tensor cores to process large training datasets

Realtime rendering with game engines also powers our Digital Human Group. Our first real time human was first shown as a live demo at TED 2019 in Vancouver.

Virtual Production has always utilised game engine rendering for real time camera work and performance capture review and presentation (*The Jungle Book* in 2014 and *Ready Player One* in 2015). With the release of V-Ray for Unreal a whole new avenue of possibilities opens up with integrating more advanced production lookdev back into the beginning of the pipeline.

**Figure 4:** *Three stills from our real-time digital human project DigiDoug, rendered in Unreal Engine. (DigiDoug is a virtual representation of our Director of Software R&D and recipient of two Scientific and Technical Academy Awards, Doug Roble.)* 

## 6.7 Conclusion

Over the course of four years we have slowly introduced a new resource for rendering that artists and TDs alike have become comfortable with and generated enough artist interest that it's uptake and acceptance has been a very positive experience, winning over even hardened sceptics. We find that GPUs can help return the long path tracing render times back to hours and minutes instead of days. For tightly scheduled projects like commercials and last minute re-renders a single failed frame on a 20 or 30 hour render could mean missing a delivery. GPU rendering allows us to wind the clock back on render times at least 10 years. Of course the tendency here is to increase the complexity to achieve even more realistic images and light transport and push render times back up. Rendering at 4k and high frame rate would also have this effect.

New GPU powered versions of all the major renderers are now available or in beta form so the choices for renderers continue to abound and further testing continues. The major shift here though will be the types of hardware we commit to in the next five years. Cloud rendering is also catching up at a rapid pace and now GPU cloud resources are becoming readily available this year. However at the scale we render at each night the cost of bursting to the cloud still needs to be evaluated.

## 6.8 Acknowledgements

Our four year voyage of discovery couldn't have been possible without the support of our wonderful pipeline group and pipeline supervisors. Special thanks to Stephen Sloan our lighting pipeline supervisor; Steve Hwan

for Atomic; Keith Gordon for our materials pipeline; Dominique Kwiek for work on our deferred pipeline and alembic procedurals; Sheryn Lu and Elyse Wei for lighting and CFX pipeline support; Elena Driskill and Gene Lin for rewriting our Samson hair plugin; Kenneth Van Aken for animation render pipeline support; Sabrina Nunes for lighting and material support for *Avengers: Endgame*; Deke Kincaid for being our liaison with the Redshift developers and of course Panos Zompolas and his team at Redshift.

## References

- NVIDIA RTX product announcement in August 2018
https://nvidianews.nvidia.com/news/10-years-in-the-making-nvidia-brings-real-time-ray-tracing-to-gamers-with-geforce-rtx
- NVIDIA Turing Architecture Video and White Paper
https://www.nvidia.com/en-us/design-visualization/technologies/turing-architecture/

**Figure 5:** *Spectral rendering of isotropic and anisotropic materials acquired from real-world samples using our method; insets show corresponding reflectance spectra. We measured these* BRDFs *using a motorized gonio-photometer, leveraging our novel adaptive parameterization to simultaneously handle* BRDF *acquisition, storage, and efficient Monte Carlo sample generation during rendering. Our representation requires 16 KiB of storage per spectral sample for isotropic materials and 544 KiB per spectral sample for anisotropic specimens.*

# 7 Capturing and rendering the world of materials

Wenzel Jakob, *EPFL*
Joint work with Jonathan Dupuy

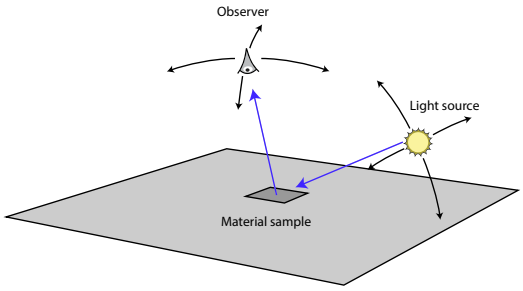
One of the key ingredients of any realistic rendering system is a description of the way in which light interacts with objects, typically modeled via the *Bidirectional Reflectance Distribution Function* (BRDF). Unfortunately, real-world BRDF data remains extremely scarce due to the difficulty of acquiring it it: a BRDF measurement requires scanning a five-dimensional domain at high resolution——an infeasibly time-consuming process.

In this talk, I'll showcase our ongoing work on assembling a large library of materials including metals, fabrics, organic substances like wood or plant leaves, etc. The key idea to work around the curse of dimensionality is an adaptive parameterization, which automatically warps the 4D space so that most of the volume maps to "interesting" regions. Starting with a review of BRDF models and microfacet theory, I'll explain the new model, as well as the optical measurement apparatus that we used to conduct the measurements[1].

## 7.1 Introduction

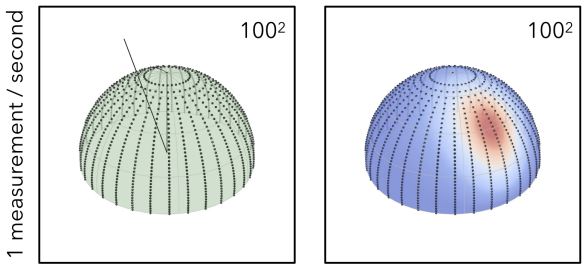
Physically based rendering algorithms simulate real-world appearance by means of an intricate simulation of the interaction of light and matter. Scattering by surfaces denotes the most important type of interaction, and the physics of this process are typically encoded in a quantity known as the *bidirectional reflectance distribution function* (BRDF). High-fidelity BRDF models thus constitute a crucial ingredient to any type of realistic rendering. In this work, we are interested in acquiring real-world BRDF data and propose a practical BRDF representation that simultaneously serves as a mechanism for acquisition, storage and sample generation within rendering software.

[1] This document is an informal writeup of the paper "An Adaptive Parameterization for Efficient Material Acquisition and Rendering" by Jonathan Dupuy and Wenzel Jakob published at SIGGRAPH Asia 2018. For a detailed technical description, please refer to the original paper. Our database and an open source reference implementation is available at https://rgl.epfl.ch/materials




Given illumination arriving from a specific direction, the BRDF specifies the directional profile of light scattered by a surface. The availability of such measurements is important not only because they improve realism, but also because they serve as validation against theoretical models that help advance our understanding of light-material interactions. For instance, the MERL database [Matusik et al., 2003] has served as inspiration and validation of numerous BRDF models over the last fifteen years.

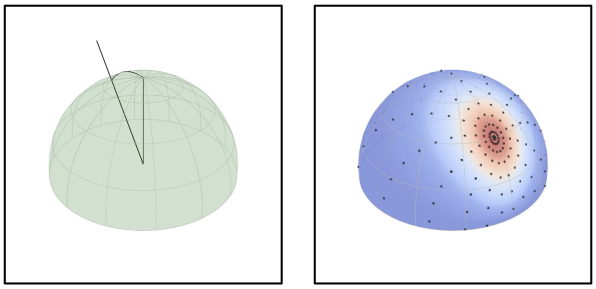
Challenges. Being dependent on two directions and wavelength, the BRDF is a five-dimensional quantity. This makes it extremely costly to discretize or measure at sufficiently high resolution due to the curse of dimensionality. For instance, suppose that a material is to be measured with a (still relatively coarse) $100^2$ regular grid for both incident and outgoing direction, and that the used device is able to perform one measurement per second that captures all wavelengths at once.




These assumptions are in fact fairly optimistic–even so, the measurement would require in excess of three years! Because of these difficulties, existing measurements have been limited to coarse resolutions in either or both the directional and spectral domains.

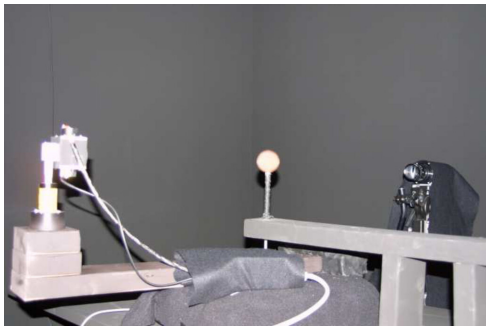
Our objective is to significantly shorten the acquisition time to make this process more practical (on the order of a few hours for isotropic samples, and a few days for anisotropic samples). To our knowledge, we are the fist to provide high resolution BRDF data that includes spectral information and anisotropy.

Prior work. Although we can't completely avoid the curse of dimensionality, there are techniques that can be applied to mitigate its effects by placing samples in a more informed manner. For instance, parameterizations built around the half-angle [Rusinkiewicz, 1998] can be used to naturally align the discretization with the direction of specular reflection, allowing for high-quality acquisition with fewer samples.



It also seems intuitive that different sampling patterns should be used depending on the material properties (e.g. for diffuse, mirror-like, or anisotropic materials). However, current parameterization approaches do not possess this type of adaptivity–the parameterization is always the same.

The work that is most relevant to this project is the technique described in [Matusik et al., 2003], which was used to measure a collection of 100 isotropic BRDFs known as the *MERL database*. Acquisition using this approach involves photographing a spherical material sample from a static camera for a number of lighting directions (the image below is taken from the original paper and shows the measurement setup).



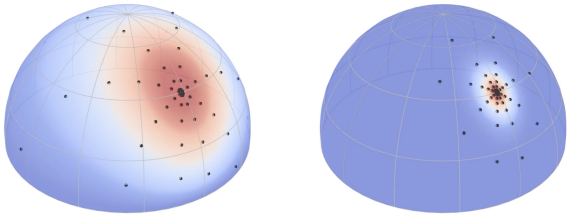
The resulting data is then stored using the Rusienkiewicz [Rusinkiewicz, 1998] parameterization. The use of spherical samples significantly reduces the measurement time, since each photograph provides observation of a large set of normal vectors, corresponding to a two-dimensional slice through the BRDF. On the flipside, this approach suffers from a number of drawbacks that we aim to address in our work

- Photographing shiny spherical targets illuminated with collimated illumination is not unlike taking pictures of the sun–the images have an extremely large dynamic range and will be contaminated with lens flare, which affects the contents of adjacent pixels. We believe that this is the source of a number of artifacts present in the original dataset [Burley, 2012].
- Due to the fixed parameterization, the sampling pattern used is the always identical regardless of the material being scanned. This means that certain materials, for which the sampling density is unsuitable (e.g. very narrowly peaked ones) will suffer from significant errors.
- It is very easy to obtain metallic spheres of various sizes, and some manufacturers of plastic spheres can also be found. Spheres can furthermore be spray-painted to enlarge the space of material somewhat. Beyond this, we have found it extremely challenging to procure suitable spherical samples of materials (e.g. cloth, wood, etc.). Any measurement technique that uses this approach is thus confronted with an inherent bias towards materials (mostly paints) that are available in spherical form. Our goal is to provide a richer set of materials including cloth, organic materials (e.g. plant leaves) and anisotropically brushed metals.
- The database only contains isotropic materials, while many materials are in fact anisotropic. The reason for this is that many surfaces are processed (e.g. polished) or created by tools that operate in a directional manner (turning, milling, lathing, etc.).
- The representation does not provide a natural importance sampling method and must be combined with other techniques that compute auxiliary data structures to be usable in a renderer. Our goal is the design of a single representation that is simultaneously usable for acquisition, storage, and rendering.
- A meta-issue of this dataset is the lack of information on what parts of the data are "real", and what parts are extrapolated or post-processed. In particular, an issue faced by any BRDF measurement device is that certain configurations are unobservable due to hardware limitations and must be approximated using an interpolant. We preserve the raw data of all measurement sessions to this problem.

## 7.2 Adaptive parameterization

Our measurement technique relies on a sampling pattern that adapts to the material being acquired. For instance, rough materials are sampled at a broad set of directions (left), while narrowly peaked materials use

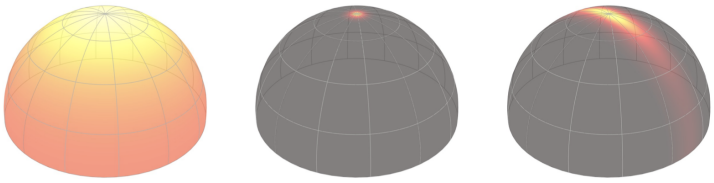
a more compact pattern matching the material's behavior. Anisotropic materials (not shown here) such as brushed metals use a suitably distorted anisotropic pattern.



Determining a good sampling pattern initially appears like a classic chicken-and-egg problem: we must already have acquired the material to make an informed decision on where samples should be placed. Fortunately, it turns out that we can use leverage the rich body of research on scattering from rough surfaces (in particular, microfacet theory) to resolve this conundrum.

### 7.2.1 Retro-reflection

Our approach is based on the observation that retro-reflection configurations (i.e. where $\omega_i = \omega_o$) are comparably easy to measure. Acquisition of such configurations for a single wavelengths unproblematic, since the domain is only two-dimensional (i.e., relatively unaffected by the curse of dimensionality). At the same time, retro-reflection encodes helpful information that characterizes the material behavior. For instance, consider the following retro-reflection plots of three different material classes.




Our objective is to turn this cue into a suitable sampling pattern.

To do so, we shall temporarily introduce an assumption: what if the material being measured perfectly satisfied the predictions of microfacet theory? In this case, the retro-reflective response of the BRDF $f$ is proportional to

$$f(\omega, \omega) \propto \frac{D(\omega)}{\sigma(\omega)\cos\theta} \tag{1}$$

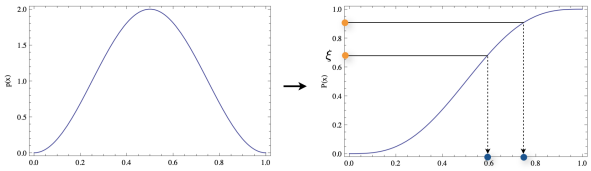
where $D$ is the material's microfacet distribution and $\sigma$ is the projected area of the microfacet surface (which also depends on $D$). There is thus a direct relationship between retro-reflection and microfacet distribution, and we show in the paper that $D$ can easily be reconstructed from such measurements by finding the largest eigenvector of a matrix. In other words: *if* the material indeed satisfies microfacet theory, we know everything only from a single capture of the retro-reflective response. We will now briefly turn to another operation, namely importance sampling.

### 7.2.2 Inverse transform sampling

Recall the canonical mechanism for designing importance sampling strategies, known as *inverse transform sampling*: starting from a density $p(x)$, we first create the cumulative distribution function

$$P(x) = \int_{-\infty}^{x} p(x')\,\mathrm{d}x', \tag{2}$$

and its inverse $P^{-1}$ is then used to map transform uniformly distributed variates on the interval $[0, 1]$. By construction, this yields samples with the right distribution.



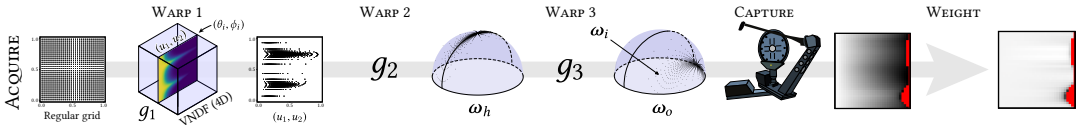
Another view of inverse transform sampling is that it creates a *parameterization* of a domain that expands and contracts space in just the right way so that different parts receive a volume proportional to their integrated density.

Virtually all importance sampling techniques–including those of microfacet BRDF models–are based on inverse transform sampling and thus can also serve as parameterizations if supplied with non-random inputs.

### 7.2.3 Putting all together

These, then, constitute the overall building blocks of our approach:

(i) Starting with a retro-reflection measurement, we compute the properties (i.e. $D, \sigma$) of a hypothetical microfacet material that behaves identically.
(ii) Next, we process $D$ to generate tables for the inverse transform sampling technique.
(iii) Finally, we use the sampling technique as a parameterization to warp the BRDF into a well-behaved function that is easy to sample using a regular grid (which turns into a high-quality sampling pattern after passing through the parameterization).




Starting from a regular grid, the above image visualizes the involved operations, which are the composition of a sequence of invertible operations $g_1$ (inverse transform mapping), $g_2$ (square-to-sphere mapping), and $g_3$ (specular reflection from a microfacet). Also note that this sequence is not just useful for acquisition–it also enables efficient importance sampling at render time.

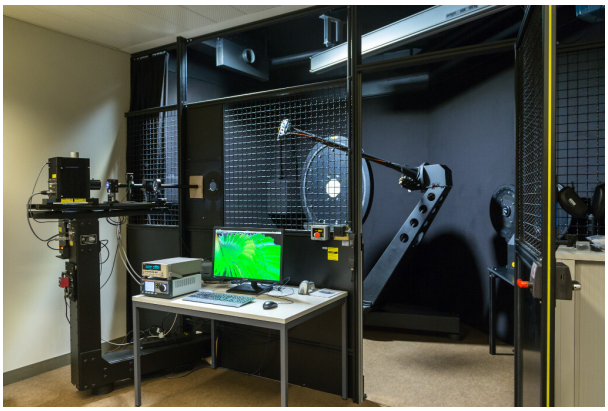
An important aspect of our previous assumption of microfacet theory-like behavior is that it is merely used to generate a parameterization–no other part of our approach depends on it. This means that we are also able to acquire functions that are in clear violation of this assumption, although this will manifest itself in interpolants that are less smooth and hence require a denser discretizations.

We briefly outline two additional important technical details–for further details, please refer to the paper:

1. Different microfacet importance sampling schemes exist, including ones that produce lower variance in traditional Monte Carlo rendering applications. Interestingly, a superior sampling scheme also tends to produce smoother interpolants in our application. We sample distribution of visible normals introduced in [Heitz and d'Eon, 2014] for this reason.
2. When a density function is re-parameterized on a different domain (e.g. from Euclidean to polar coordinates), we must normally multiply by the determinant of the Jacobian to obtain a valid density that accounts for the expansion and contraction of the underlying mapping. In our application, we also observed that it is beneficial to multiply BRDF measurements by the combined Jacobian of the above functions $g_1$ to $g_3$, which yields an even smoother function that can stored at fairly low resolutions (e.g. 32x32).

## 7.3 Hardware

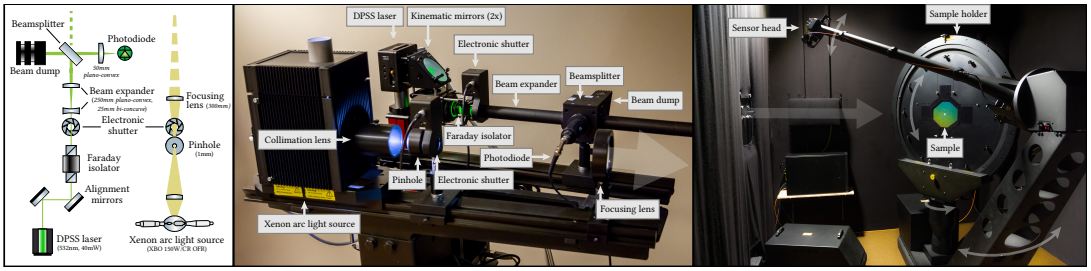
Our acquisition pipeline relies on a modified *PAB pgII* [PAB, 2018] gonio-photometer, which acquires spectral BRDF samples at a rate of approximately 0.5-1 samples per second.



This device illuminates a flat vertically mounted sample from a light source that is mounted one of two stationary rails (we use both during acquisition). To change the incident light direction, the motorized sample mount rotates around the vertical axis and the normal direction of the sample. A sensor head mounted on a motorized two-axis arm measures the illumination scattered by the sample. We use a mounted Zeiss fiber spectrometer to capture dense ($\sim$ 3nm spacing) spectral curves covering the 360-1000nm range.

The machine has two "dead spots" that cannot be measured with the combination of sensor arm and light source: first, the sensor arm cannot view the sample from straight below, since this space is occupied by the pedestal holding the sample mount. Second, shadowing becomes unavoidable when the sensor is within approximately 3 degrees of the beam, making retro-reflection measurements impossible.

The former is unproblematic and can be reconstructed without problems. However, the latter is a significant impediment, since the method presented so far heavily relies on the ability to perform an initial retro-reflection measurement. We work around this limitation by using two separate illumination and measurement paths shown below:




In the first phase, we rotate the sensor arm in a configuration so as to avoid blocking any lighting from the laser onto the sample, and a laser illuminates the sample through a beamsplitter. Retro-reflected illumination from the sample reaches the same beamsplitter and is reflected onto a small photodiode, whose current we measure. The second measurement stage uses a broadband xenon arc light source along with the spectrometer along with the normal spectral sensor on the sensor arm.

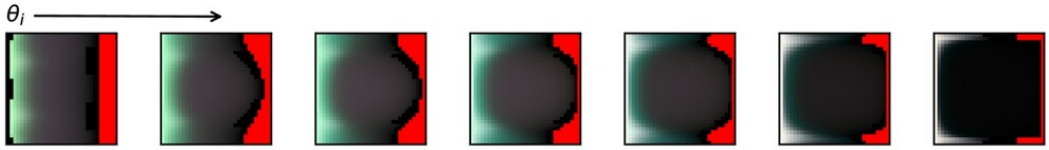
## 7.4 Database

We're currently in the process of acquiring a large database of materials using the presented technique. Our objective is that this database eventually grows to many hundreds of entries that sample the manifold of natural and man-made. Outside help is welcomed during this process: if you have interesting samples, we'd love to measure them!

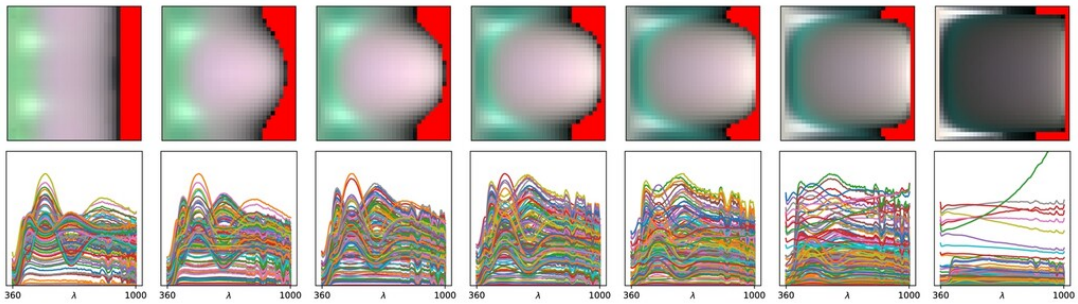
Please see https://rgl.epfl.ch/materials for a web interface of what was acquired thus far. Scans on this page can be interactively viewed and analyzed using the *Tekari* viewer created by Benoît Ruiz, which runs using *WebGL* and *WebAssembly*. For this, simply click on any material to expand the entry, and then push the *Interactive viewer* button. We've also released a reference BRDF implementation, which provides the standard operations needed by most modern rendering systems: BRDF evaluation, sampling, and a routine to query the underlying probability density. The repository also contains a plugin for the *Mitsuba* renderer and *Python*

code to load the data files into *NumPy*, which is useful for visualization and systematic analysis of the database contents. All resources are linked from the material database page.

We briefly discuss two visualizations that are shown on BRDF database page: the below image shows an RGB rendition of what was measured by the spectrometer using our parameterization. Note that the data typically has extremely high-dynamic range, hence some detail in the black may simply be invisible. Red pixels correspond to wasted space of the discretization. The black column on the left is a region that could not be measured.




This following plot contains the same data, but after post-processing. The black regions on the left were filled in, and the data was scaled by the Jacobian of the parameterization (see the paper for details). This has the effect of making the representation significantly easier to interpolate. The bottom row shows spectral curves in the 360-1000nm range corresponding to the row above.




Since our acquisition device captures full color spectra, we've so far scanned many materials with wave-optically interesting behavior (which often have oscillatory reflectance spectra). Figure 6 contains a few selected visualizations.

## 7.5 Conclusion

We have presented the first adaptive BRDF parameterization that simultaneously targets acquisition, storage, and rendering. We used our method to introduce the first set of spectral BRDF measurements that contain both anisotropy and high frequency behavior. Our dataset contains many materials with spectrally interesting behavior (multiple types of iridescent, opalescent, and color-changing paints).

We find that the availability of spectral BRDF datasets is timely: as of 2018, major industrial systems render in the spectral domain (e.g. *Maxwell*, Weta Digital's *Manuka* renderer), and two major open source renderers, *PBRT* and *Mitsuba*, are presently being redesigned for full-spectral rendering [Pharr and Jakob, 2017]. We believe that access to a comprehensive spectral dataset will enable future advances in the area of material modeling.

Going forward, we are also interested in using our technique to model materials with spatial variation–this involves the additional challenge that an ideal sampling pattern is now likely different for each spatial location, and a compromise must thus be reached to turn this into a practical measurement procedure. Furthermore, storage becomes prohibitive and compression strategies are likely needed even during a measurement. The hardware of our gonio-photometer was recently extended, allowing us to mount high-resolution CCD and CMOS cameras onto the sensor arm, and we look forward to facing these new challenges in the future.

# References

Brent Burley. 2012. Physically-based shading at Disney. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*. ACM, New York, NY, USA.

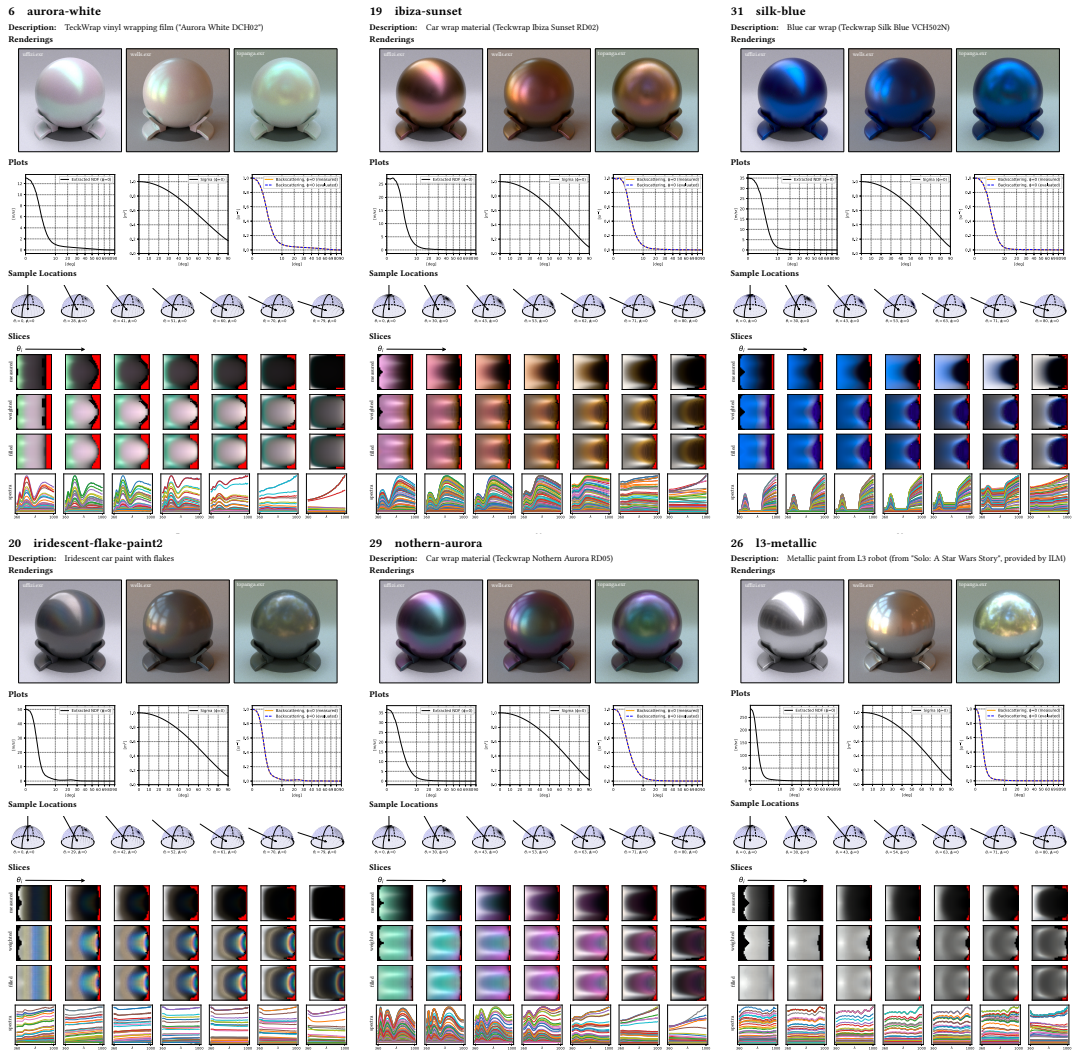



**Figure 6:** *Visualizations of selected materials from our database.*

Eric Heitz and Eugene d'Eon. 2014. Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals. In *Computer Graphics Forum*, Vol. 33. 103–112.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A Data-Driven Reflectance Model. *ACM Trans. Graph.* 22, 3 (July 2003), 759–769.

PAB. 2018. pab advanced technologies Ltd. http://www.pab.eu. Accessed: 2018-01-09.

Matt Pharr and Wenzel Jakob. 2017. personal communication.

Szymon M Rusinkiewicz. 1998. A New Change of Variables for Efficient BRDF Representation. In *Rendering Techniques '98*. Springer, 11–22.

# 8 Material Modeling in a Modern Production Renderer

Andrea Weidlich, *Weta Digital*

## 8.1 Motivation

Modern computer graphics have reached astonishing levels of visual realism and artistic expression. As our technological abilities have progressed, several distinct sub-fields, each with its unique technical challenges and goals, have emerged within the discipline. Examples are non-photorealistic rendering, scientific visualisation or production techniques geared for the entertainment industry.

During the last ten years, physically based rendering managed to gain so much importance that nowadays it can be regarded robust enough for all application domains that it might be considered for. Significant progress has been made towards shading techniques under the constraints of such rendering systems. In part this is because rendering algorithms have advanced so far that we can spend more time generating complex materials. And while for the longest time it was acceptable to limit the creativity of a user to a very small subset of materials which were put together in an uber-shader, more and more applications provide layered models, albeit with varying degree of physical correctness.

The two big advantages of layered models are that they are physically based (if not entirely physically *correct* in the narrow sense of the word because of the approximations that are still introduced into the evaluation) and therefore produce feasible looking results, and are intuitive to use at comparably low computational costs. Getting accustomed to shade with a layered approach instead of using single un-layered models meant a paradigm shift in the workflow and a new way of thinking. And while ten years ago layered shading models were still a niche product and the general agreement was that one can either have physically correctness or artistic freedom, physically based shading models are nowadays a basic component not only in expensive high quality rendering, but also in many realtime applications.

This section of the course will review the development of the material system in *Manuka*, the production path tracer in use at Weta Digital, used on movies like *Mortal Engines* (2018) or *Alita: Battle Angel* (2019). We will discuss the design decisions we had to make and how they affect the way we work with materials in our production environment.

## 8.2 Design Ideas

With hundreds and hundreds of different assets and palettes in a single movie, a modern material systems in a production path tracer needs to be able to fulfill several requirements

- *Flexibility.* Different characters require different visual complexity, and a material system must be versatile enough to adapt to different production requirements. Within one scene we have not only hero characters but also background characters, environments and atmospheric effects. While the majority of our characters are realistic, we also need to be able to model a more cartoony look-and-feel. We do not want to limit ourselves to one particular class of BSDFs (such as having only microfacet models) and want to have the flexibility to adapt the system to evolving requirements.
- *Storage.* Solutions for realistic appearance models often demand heavy pre-computation. Since we rely on visual complexity, we heavily make use of textures and cannot incur long pre-computation times nor excessive per-vertex or per-asset storage costs.
- *Artistic Freedom.* While materials in a path tracer have to fulfill certain physical requirements like energy conservation and Helmholz reciprocity, their usage is not bound to be physically correct. We frequently have to deal with appearances that do not exist in reality and even more often have to tweak materials based on artistic decisions. While we do aim for physical plausibility, we need to be prepared to give users the possibility to break reality.

- *Computation Time.* Since we have only a certain time budget to render a movie, computation time is an important factor. We need to be able to turn certain effects on and off based on budget. Sampling must be efficient.

## 8.3 Manuka's Material System

Weta Digital's proprietary in-house renderer *Manuka* [Fascione et al., 2018] is a spectral path tracer that comes with a powerful material system. It was first used in production on the movie *Dawn of the Planet of the Apes* (2014) and back then had already the same basic components we use today. The key design decision was to aim for the most flexible material system possible. Instead of implementing just one powerful uber-shader, *Manuka* supports a layer-based material system that lets the user combine arbitrary BSDFs and assemble them into a material stack.

Currently we support over a hundred different BSDFs which are all layer-able. These BSDFs include a variety of different appearance descriptions, some have an academic background and were developed inside and outside of the rendering community, others were built in-house. We differ between several BSDF classes: BRDFs reflect only, BTDFs only transmit, BSDFs which scatter in both hemispheres, BCSDFs for curves and EDFs for emission. All BSDFs are flagged according to their type and during light transport we make use of this information to make sampling and computations more efficient.

Regardless of their type, all BSDFs need to implement the same interface. Apart from the functions one would expect in a path tracer, such as Sample(), Eval() and Pdf(), we have functions which inform the renderer about BSDF-specific information like mean directions, IORs and reflectance, average roughness or energy distribution. All BSFDs can be versioned in order to support older and newer versions of the same BSDF int he same process, to facilitate shader and model evolution at different paces for different assets.

### 8.3.1 Shade-before-Hit vs. Shade-on-Hit

Different from the other production renderers we are aware of, *Manuka* has a shading architecture which we called *shade-before-hit* [Fascione et al., 2018], where instead of evaluating shaders upon a ray-hit event, most of the shading computation happens before ray tracing starts. This unique design decision comes with advantages and disadvantages. Since we do not have to evaluate shaders on the fly, we avoid having to execute expensive texture reads upon ray-hit, which can cause difficult to mitigate bottle-necks due to lack of locality of reference inherent in the path tracing algorithm. Therefore, our architecture allows production shading networks to become much larger and more detailed than usual production shader networks, and in fact they commonly have dozens of texture reads per BSDF. Consequently we can achieve much higher visual complexity.

On the other hand, our shading has to be completely view-independent and during light transport we cannot access material information of neighbouring pixels. Instead, everything needs to be done inside the material system. The shaders role is to leave on the micropolygon vertices the BSDF parameters and the weights of BSDFs and layers, as well as the layout of the layer-stack.

Compression of the data is a huge help, since everything is stored on the micropolygon grids. We have observed that 8-bit tends to be enough for most colour data, such as albedo values, however roughness and similar data is stored in 16-bit to avoid banding artifacts.

### 8.3.2 Filtering Techniques

While it is true that *Manuka*'s architecture achieves more complex materials at a lower cost during runtime, this obviously comes with a cost in memory. Our *RenderMan* heritage implies that final shading quality is determined by the shading rate. For this reason, filtering techniques become more important, as we store the data on the vertices and cannot access textures during light transport. For non-linear data such as normals, filtering is not straightforward. Techniques like LEAN mapping [Olano and Baker, 2010] or [Han et al., 2007] become essential to keep the look of an asset consistent between close-up and distant shots. Loss of surface detail has to be recovered with techniques like [Yan et al., 2014] or [Jakob et al., 2014].

However, not all data can be filtered easily. A problematic example is interference data or blackbody temperature. When we pass in numerical data that is converted into colour through strongly non-linear relations, the interpolate-then-filter and filter-then-interpolate approaches have a high chance of producing markedly different results. These filtering techniques are implemented upstream of *Manuka*.

### 8.3.3 Spectral Input Data

*Manuka* is a spectral renderer, and all BSDFs compute their quantities in a spectrally-dependent manner. However, the majority of our input data is in some RGB space, since artist workflows and software at present is only available to us this way. During rendering we spectrally lift the data (see Section 1.4.2 in the first part of this course for further details). This results in the interesting problem that, since there is no unique correspondence between a single spectrum and a specific RGB value, the spectra we generate are somewhat arbitrary and not exactly the one from the photographed reference asset. This far it seems to us that the many of our use cases are somewhat forgiving and that the practical simplicity of an RGB workflow outweighs at present the shortcomings, at least when dealing with props and background assets. However we do offer the possibility to pass in spectral data from the outside for assets where the appearance is of critical importance. Up to this point, in production this has most often been used for spectral light sources, which on one hand can be measured easily and on the other hand have often peculiar spectral distributions than are poorly approximated by our current spectral lifting techniques. Additionally, we have internally stored spectral curves for specific components such as melanin data or real and complex index of refraction for the more commonly used metals. Using spectral data instead of RGB is increasingly important when we deal with absorption or have many indirect bounces, such as in subsurface scattering or hair.

## 8.4 Material Workflows

Every BSDF has a scalar weight, $w_b \in [0, 1]$. BSDFs are combined with layer operators; like BSDFs, layers also have scalar weights $w_l \in [0, 1]$, weights smaller than 1 will result in BSDFs or layers existing only to a certain percentage. Usually weights are textured with masks. In a shader we first define the BSDFs we want to use and combine them later into a layer with code such as this:

```
AddLobe("bsdf1", LambertBsdf());
AddLobe("bsdf2", BeckmannBsdf());
CombineLayers("layer", "mix", "bsdf1","bsdf2");
```

BSDFs can be combined with other BSDFs or layers. Geometry-tied types like BCSDFs can only be combined with the corresponding type. Layering programs (these are dynamically compiled callable entities which embody the layout of the layer stack and BSFDs) are assembled during shading and are constant across a shape, only the input parameters can vary across the surface of a shape. Since the program doesn't store or precompute light transport data, it is lightweight and doesn't cause noticeable memory overhead. Nevertheless, if the same layout is used on multiple shapes, we store it only once. View-dependent components are evaluated on the fly during light transport.

### 8.4.1 Layering Operations

Currently we support four different layering operations to combine BSDFs. Note that we call both *horizontal* operations where we increase the number of BSDFs within one single level and *vertical* operation where we increase the number of BSDFs in a vertical dimension layering. Strictly speaking horizontal operations are not layers, rather they capture phenomena like partial coverage, such as a thin gauze layer or the blending needed to antialias edges, however we use the same layering infrastructure and treat them as a layering operators.

- *Mix.* Mixing two materials is the most basic layering operation. We can mix two or more BSDFs together

within one single operator, mixing weights are normalised.

$$w_1 = \left( \frac{w_{b_1}}{w_{b_1} + w_{b_2} + ...} \right) * w_{l_1} \tag{3}$$

$$w_2 = \left( \frac{w_{b_2}}{w_{b_1} + w_{b_2} + ...} \right) * w_{l_1} \tag{4}$$

A real life example would be the mixing of several different liquids like water and ink (see figure 7).

- *Coat.* Coating one material on top of another is what most people will intuitively associate with layering. Many materials in real life exhibit some form of layering, e.g. car paint or wet materials are the most obvious examples and indeed this makes the coating operation the most frequently used in our production palettes. One material covers a second one, and what energy $a_b$ is not reflected $a_r$ or absorbed $a_{a_1}$ by the first will reach the second material. For this operation we need to know the reflection and transmission albedo $a_t$ for each BSDF. Opaque layers have a transmission albedo of 0.

$$a_{b_1} = a_{r_1} + a_{a_1} \tag{5}$$

$$w_1 = w_{b_1} * w_{l_1} \tag{6}$$

$$w_2 = (1.0 - a_{b_1} * w_{b_1}) * w_{l_1} \tag{7}$$

While weights are direction independent, albedos will vary with directions.

- *Blend.* Blending is again a horizontal layering operation and is similar to mixing except that we mix only two materials and the weights are inverted. Blending is best described as some form of material dithering where one material is substituted with another material.

$$w_1 = w_{b_1} * w_{l_1} \tag{8}$$

$$w_2 = (1.0 - w_{b_1}) * w_{b_2} * w_{l_1} \tag{9}$$

While the same result could be achieved with a standard mix, we tend to use the blend operator more often in production palettes because it includes the layer weight in the inverse weight.

- *Add.* Adding is a vertical operation and will add the contributions of two materials together.

$$w_1 = w_{b_1} * w_{l_1} \tag{10}$$

$$w_2 = w_{b_2} * w_{l_1} \tag{11}$$

This will cause an increase in energy and is under normal circumstances not useful in a path tracer. Hence we restrict this operation to EDFs which are allowed to produce energy and convert this operation into a coat if a user attempts to use it on a BSDF or a part of the stack that does not exist entirely of EDFs. A real-life example which would work in this fashion is luminous paint. However remember all lights in *Manuka* are simply emissive geometry, so all luminaires of any kind flow through often simple forms of this mechanism

All layering operators support not only BSDFs but also layers as input. While in theory we could support many more operations without overhead like e.g. subtraction, view-dependent mixing or similar expressions, we currently do not feel the need to do so. However, our system is easily extensible since a layering operator for us is nothing else than an interchangeable mathematical expression.

### 8.4.2 Single-sided Materials

Our material system has the unique property that we model a material as a whole stack of layers instead of treating front and back as different materials with separate shaders. During shading we generate a second layer program which is built in the inverse order from the original, so that during light transport we evaluate the material back to front instead of front to back depending on the relative orientation of the surface normal versus the incoming ray. This has the advantage that simulating single-sided materials — like dirty windows

**Figure 7:** *Real life examples for various layering operations. From left to right: Mixing, coating, adding, blending.*

that have dirt only on one side of the glass — becomes very natural and an artist does not have to think about whether transmission is the same from both directions because it is a byproduct of the material evaluation. Additionally, we also support symmetric shading, i.e. the material layout is mirrored without storage overhead so that a user does not have to define both sides of a material if they are identical (see figure 8). Note that we do not have to duplicate the geometry to achieve double-sided shading.

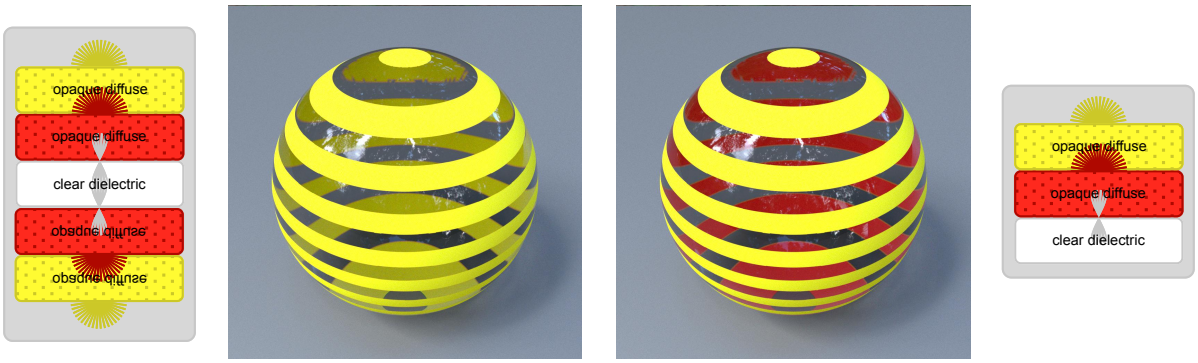



**Figure 8:** *An opaque yellow layer is coated on a red opaque layer which is coated on a clear dielectric. Glass sphere with painted stripes with (left) and without symmetry shading (right). With symmetry shading on we build a symmetric stack, the red is always occluded. Without symmetry shading we can see the back of the stack through the glass.*

### 8.4.3 Accumulation Techniques

Real materials will start changing their appearance once they are combined with other materials. A classic example is wet concrete; dry it has a bright to medium grey diffuse appearance, but once it becomes wet it will become much darker and shinier. We support two different techniques that can be enabled on demand.

- IOR *accumulation.* This technique is useful to simulate e.g. water on top of a material. We inherit the index of refraction of the top material and apply it on the bottom material. Once we reduce the relative index of refraction of the base material (e.g. water on top of skin), less light will be reflected by the bottom layer, reducing or completely removing the specular reflection consequently (see figure 9).
- *Roughness accumulation.* Materials that are coated by a rough layer will inherit the roughness from the top layer (see figure 10). The bottom roughness has to become at least as rough as the top surface. A typical example for this is a layer of frosting on top of a glass window plane. The properties of the glass do not change, but the layer of ice will change the distribution of the transmission rays.

Roughness and IOR accumulation are calculated with a different layer program than the one we use for weights.

Accumulation techniques were first used on *War of the Planet of the Apes* (2016), and by now most of our palettes have both techniques on by default, since both reduce the overhead to produce and align textures.

**Figure 9:** *Comparison between IOR accumulation off and on. Two dielectric materials with a different roughness are coated on top of each other. The IOR of the BRDF on top increases from left to right from* 1.0 *to* 1.5*, the bottom BRDF IOR is fixed at* 1.5*. Without accumulation, the highlights double, with IOR accumulation the bottom highlight becomes weaker and vanishes when they are index matching.*

**Figure 10:** *Metallic pigs with a forward scattering dirt layer on top. The thickness of the dirt increases from left to right, producing more scattering. Top row: Roughness accumulation is off, the highlight below the dust stays artificially sharp. Bottom row: With roughness accumulation on the highlight becomes blurrier.*

Also, on complex layer stacks accumulation techniques can produce effects which would not be possible without significant storage and evaluation overhead. We only have a couple of BSDFs which do not support accumulation techniques. Those are either BSDFs which do not have an understanding of IORs (like for example a Lambert BRDF) or BSDFs that are used for alpha blending where roughness would be counterproductive. In the future we plan to approach BSDFs from a more holistic perspective and aim that all BSDFs, old and new, have an understanding of their surrounding and can interact with it and each other.

## 8.5 Production Examples

We will now discuss two examples on how we setup two different materials in our production pipeline and the thought process that went into it.

### 8.5.1 Leaves

While leaves are not the obvious example when it comes to layered materials, a closer look reveals that their structure is indeed made of layers. Cuticles, upper and lower epidermis protect the internal structure, the mesophyll. The central leaf in turn consists of a palisade layer, tightly packed long cells filled with chloroplasts for photosynthesis and spongy mesophyll with loosely packed parenchyma tissue. When we look at reference measurements in [Bousquet et al., 2005] or [Combes et al., 2007], we can see three basic components: a diffuse reflectance, a diffuse transmittance and a specular reflectance that exhibits Fresnel behaviour. Hence we start with these three components.

We start with the core of the leaf; we know that leaves have a different appearance and different reflectance properties front and back. This is because leaves are built so that the front side gathers energy from the sun and transports it into the core while the back which is not exposed to the sun brings in carbon-dioxide and releases oxygen. The front tends to be glossier than the back and is more colourful, while the back tends to be

duller, has more trichomes and veins can be more visible. We can simulate different colours by coating two opaque, diffuse layers on top. Since they are not transmissive only one colour can be seen at a time, the other one is completely blocked in the stack.

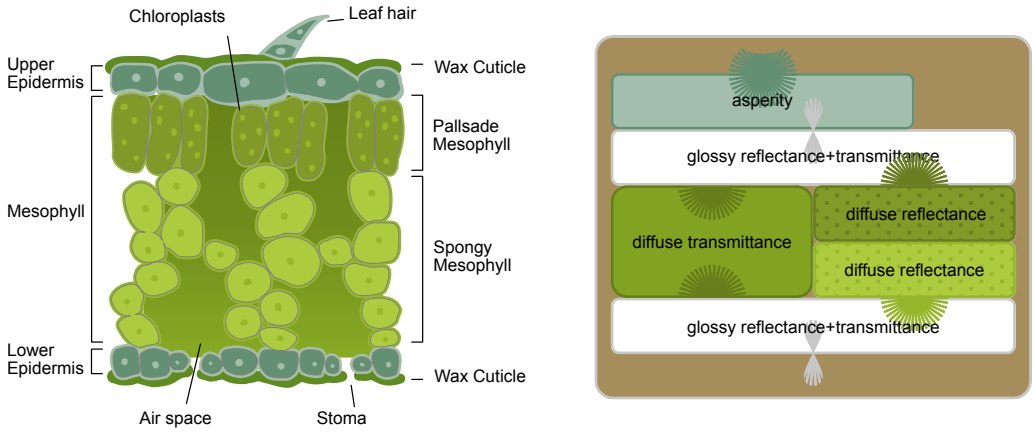



**Figure 11:** *Schematic illustration of a leaf structure and our corresponding layer stack.*

Next we want to generate transmission. Since we want to avoid colour bleeding from front to back and vice versa, we mix a diffuse transmissive BRDF *after* the coating. Light is now either reflected from the diffuse reflector or transmitted, but it can never reach both reflectors at the same time. The transmittance BSDF will be tinted.

The next step is to generate the Fresnel effect. We coat a glossy BSDF on top of the existing stack as well as below it, naturally these two will most likely have different roughness. We can avoid seeing the highlight on the inside by enabling IOR accumulation. Roughness accumulation will take care that the transmission will always be diffuse. Note that we coat the glossy BSDF after we mix the transmission. Otherwise we would loose specular reflectance energy and would need to compensate for it. Figure 11 shows a schematic illustration of the final layer stack, figure 12 shows a comparison between rendering and reference.

Although we are aware that there are dedicated leaf BRDF models (e.g. [Jacquemoud and Frederic, 1990] or [Jacquemoud et al., 2009]), we use a custom setup because it is easily extendable if we have to account for effects like bioluminsecence or asperity.

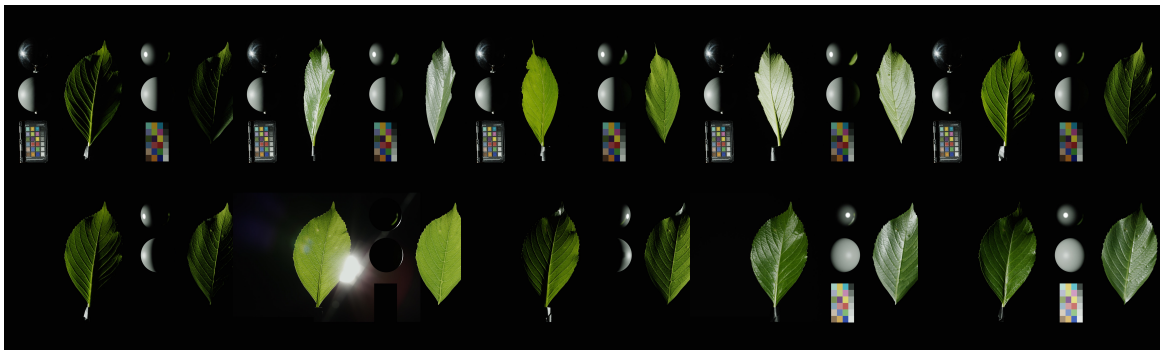


**Figure 12:** *Leaf turntable with comparison of reference and rendered leaf. In the top row the leaf is rotated, in the bottom row the light is rotated.*

### 8.5.2 Skin Variations

While we can learn a lot about materials by looking at measurements and existing research, there is a distinct drawback. Material research tends to deal with "lab grade" materials which are in a perfect and pristine condition. For movie productions on the other hand we are most interested in worn appearance as a tool to tell

stories. A good example for this is skin. Skin is a highly complex material which consists of many different components, varies from person to person, with age and exposure to environmental effects. We need to cater for these different needs since almost all of our hero characters will have skin in one or another place.

We do not have a packaged skin BSDF, but make use of the existing components we already have to combine them in various ways. We normally start with our basic skin setup and then, based on the character, will start changing the scattering and absorption properties. Since *Manuka* was from the beginning capable of fully trace subsurface scattering, we never made use of kernel-based BSSRDFs on hero characters, which allowed us to simplify the look development process. We support albedo data as input for our volumes which we in turn internally convert into volume properties.

Once the basic setup is done we will start fine-tuning the look. This is where our layering approach comes in very handy. Depending on what kind of appearance we want to simulate we will place pigments on a different place in the material stack. By repositioning layers and changing the position of the pigmenting components we can for example convert blood into tattoos. When deciding where to put these components we take care that it corresponds to their real-life counterpart. Figure 13 shows examples of various layer configurations.

This way of working was particularity useful for the humanoid characters on planet Mül in *Valerian and the City of a Thousand Planets* (2017). Depending on the shot, pigments, chromatophores and iridophores could either be generated as geometry and placed within the scattering skin, or they could be converted into layers and used with a layered material approach.

The flexibility we gain with our material system makes it very easy for us to bring selective realism to scenes that are being assembled under artistic control and produce the looks we need to make the audience believe that what they see is real.

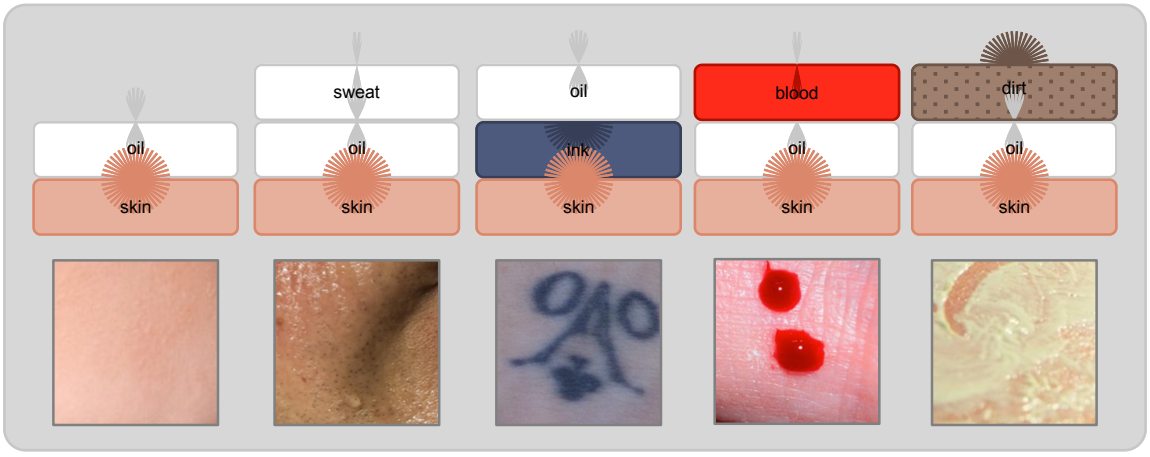



**Figure 13:** *Examples of skin variations. We can change the appearance of a material by repositioning the individual layers.*

## References

Laurent Bousquet, Sophie Lacherade, Stephane Jacquemoud, and Ismal Moya. 2005. Leaf BRDF measurements and model for specular and diffuse components differentiation. *Remote Sensing of Environment* 98 (10 2005), 201–211. https://doi.org/10.1016/j.rse.2005.07.005

Didier Combes, Laurent Bousquet, Staphane Jacquemoud, Hervé Sinoquet, Claude Varlet-Grancher, and Ismal Moya. 2007. A new spectrogoniophotometer to measure leaf spectral and directional optical properties. *Remote Sensing of Environment* 109 (07 2007), 107–117. https://doi.org/10.1016/j.rse.2006.12.007

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomas Davidovic, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Transactions on Graphics* 37 (08 2018), 1–18. https://doi.org/10.1145/3182161

Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. Frequency Domain Normal Map Filtering. *ACM Trans. Graph.* 26, 3, Article 28 (July 2007). https://doi.org/10.1145/1276377.1276412

Stéphane Jacquemoud and Baret Frederic. 1990. PROSPECT: a model of leaf optical properties spectra. *Remote Sensing of Environment* 34 (11 1990), 75–91. https://doi.org/10.1016/0034-4257(90)90100-Z

Stéphane Jacquemoud, Wout Verhoef, Frédéric Baret, Cédric Bacour, Pablo J. Zarco-Tejada, Gregory P. Asner, Christophe Francois, and Susan L. Ustin. 2009. PROSPECT+SAIL models: A review of use for vegetation characterization. *Remote Sensing of Environment* 113 (2009), S56 – S66. https://doi.org/10.1016/j.rse.2008.01.026 Imaging Spectroscopy Special Issue.

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete Stochastic Microfacet Models. *ACM Trans. Graph.* 33, 4, Article 115 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601186

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

Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-resolution Normal-mapped Specular Surfaces. *ACM Trans. Graph.* 33, 4, Article 116 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601155