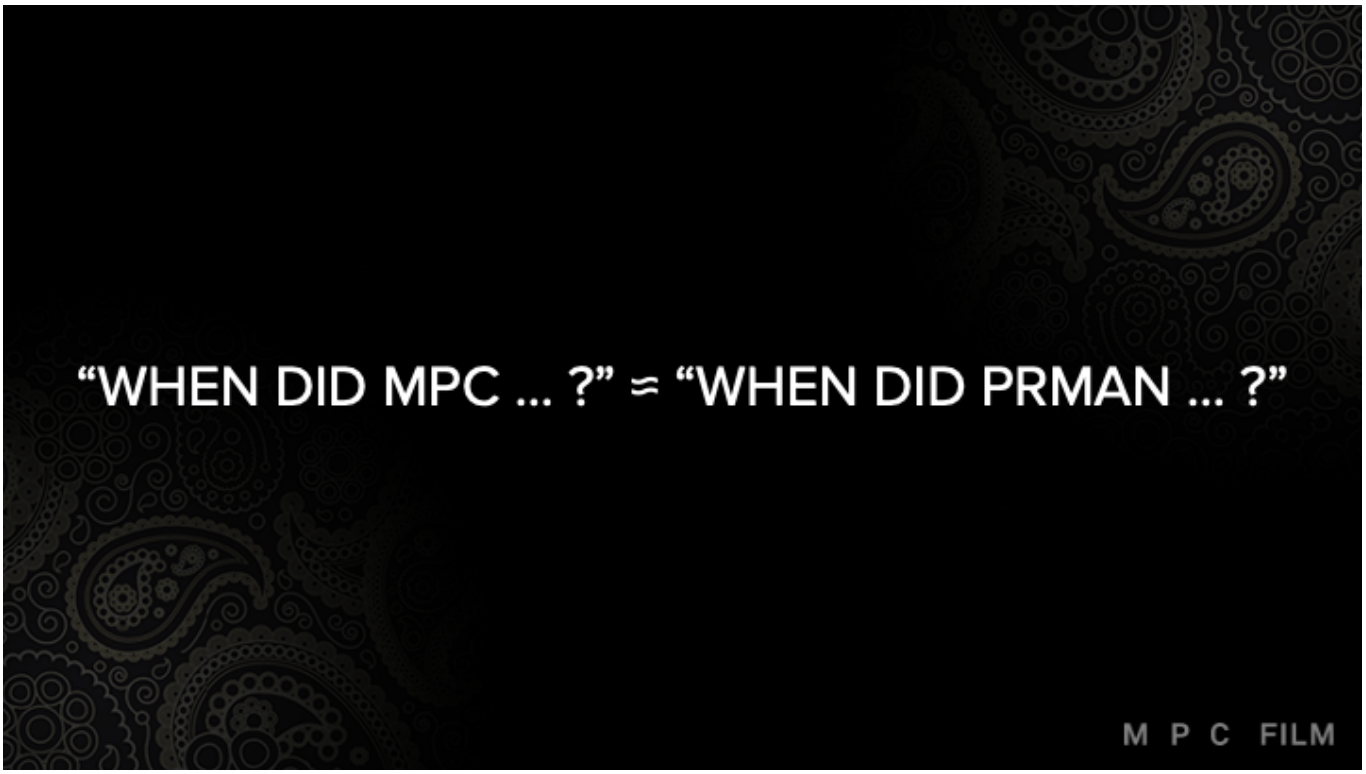
Hello and good morning, afternoon, evening or night, depending on where you're joining us from. My name is Rob Pieké and, on behalf of my co-contributors Igor and Will, I'd like to welcome you to a journey through MPC's adoption of physically based shading over the last twenty years. I'll provide some background and context as I quickly cover the years leading up to 2015, Igor will talk in depth about the work we've done in recent years on hair shading, and Will will close with a commentary on where things stand today and where we see further challenges slash exciting opportunities for the future.

MPC has been pursuing increasingly physically motivated and grounded approaches to shading and rendering for the last fifteen years. This journey has involved a constant reassessment of pragmatic decision making based on: the computational power available, the way our artists speak about and interact with shading, and advances by both academia and elsewhere in the industry. So, without any further ado...

Where did it all begin for us? I wasn't able to track down, with any confidence, what software MPC used to render its very first image, but I was able to get a 20+ year old email from one of our systems engineers where he thanked Pixar for coming to visit us and showing us a demo of the latest RenderMan release on CD (remember, this is back in 1998).

{ VIDEO DIALOGUE }

Where did it all begin for us?



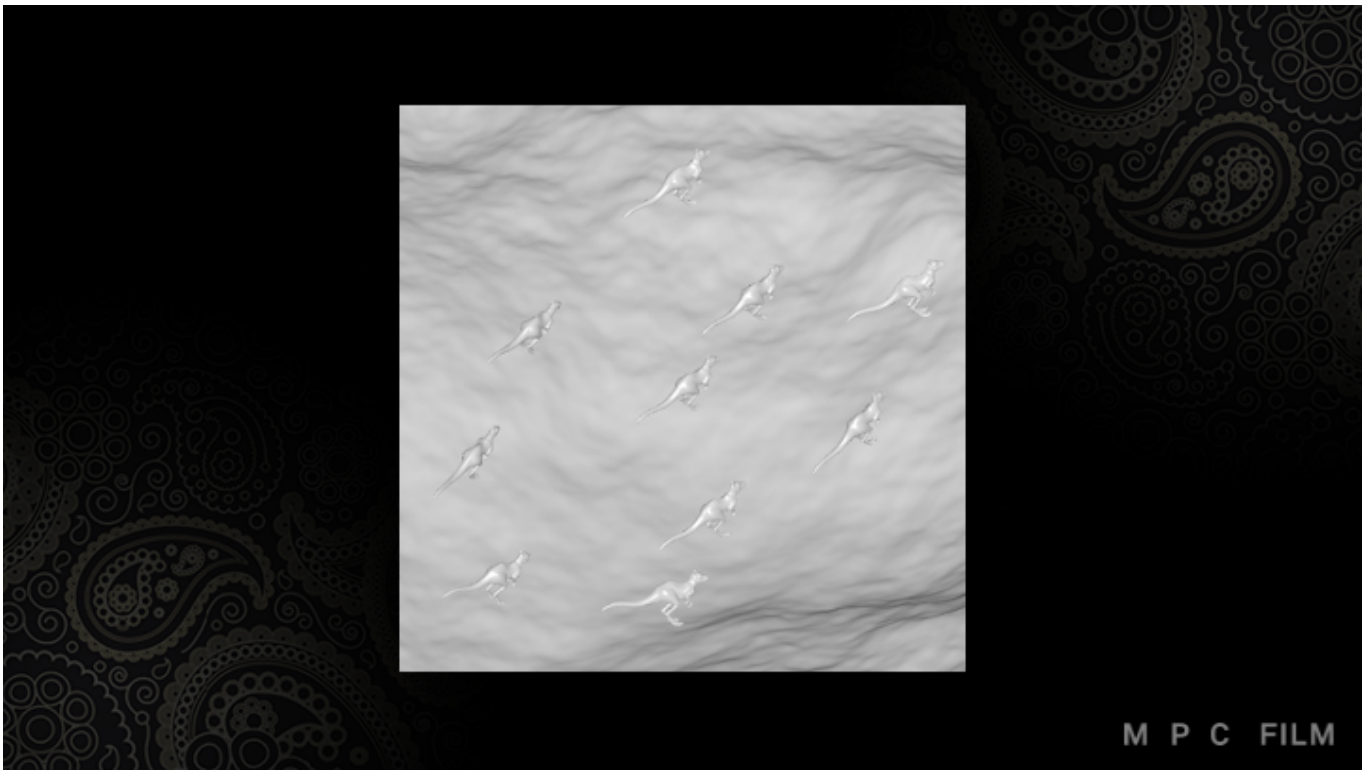
If you want to know when we adopted a certain advance in rendering, look at the RenderMan release notes until that feature is mentioned, and you’ll be within a few months. We have a track record of being very aggressive in our adoption of new releases of RenderMan, generally more so than with many other third-party software packages we use. It’s not uncommon for us to throw beta versions into production.

I also really want to highlight this slide to be upfront about the bias (no rendering pun intended) in this presentation. Some of what we say today may be universal, some things might be more RenderMan specific, and many may just be the wonky and wonderful way we work at MPC.

My memory of the past is probably more romantic than the reality of it, but it was for the most part a simpler time. Shaders were generally pretty short and not very complicated to write. We cheated everything like crazy. There was little to no physical motivation behind our shading models at the time; it was driven very heavily by artistic intent. I don’t mean to say that’s necessarily a bad thing unto itself, but it did have consequences.

Irrespective of the shading models, this traditional multi-pass approach to rendering, meant we ended up generating a lot of byproducts.

Let’s pretend this is one of the iconic battle shots from MPC’s work on Troy. Shadows are pretty critical in selling that the Trojan soldiers are actually standing on the ground, and it’s all done using depth maps ...



... rendering the world from the viewpoint of the light and recording the distance between the light and the geometry of the scene.

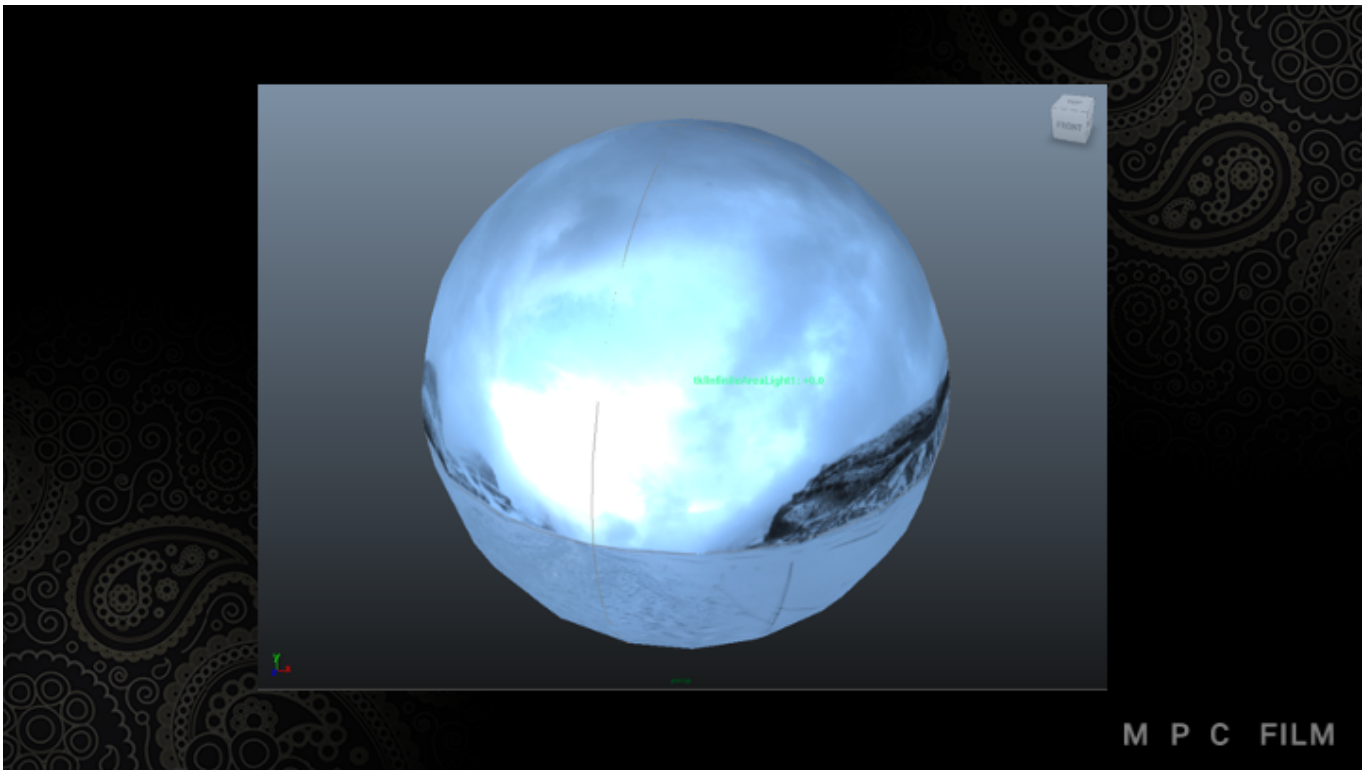
If we focus on just the soldiers, we can see there's pretty sparse coverage of the frame, and this led to the custom development of tools ...

... which would look at the soldiers one-by-one ...

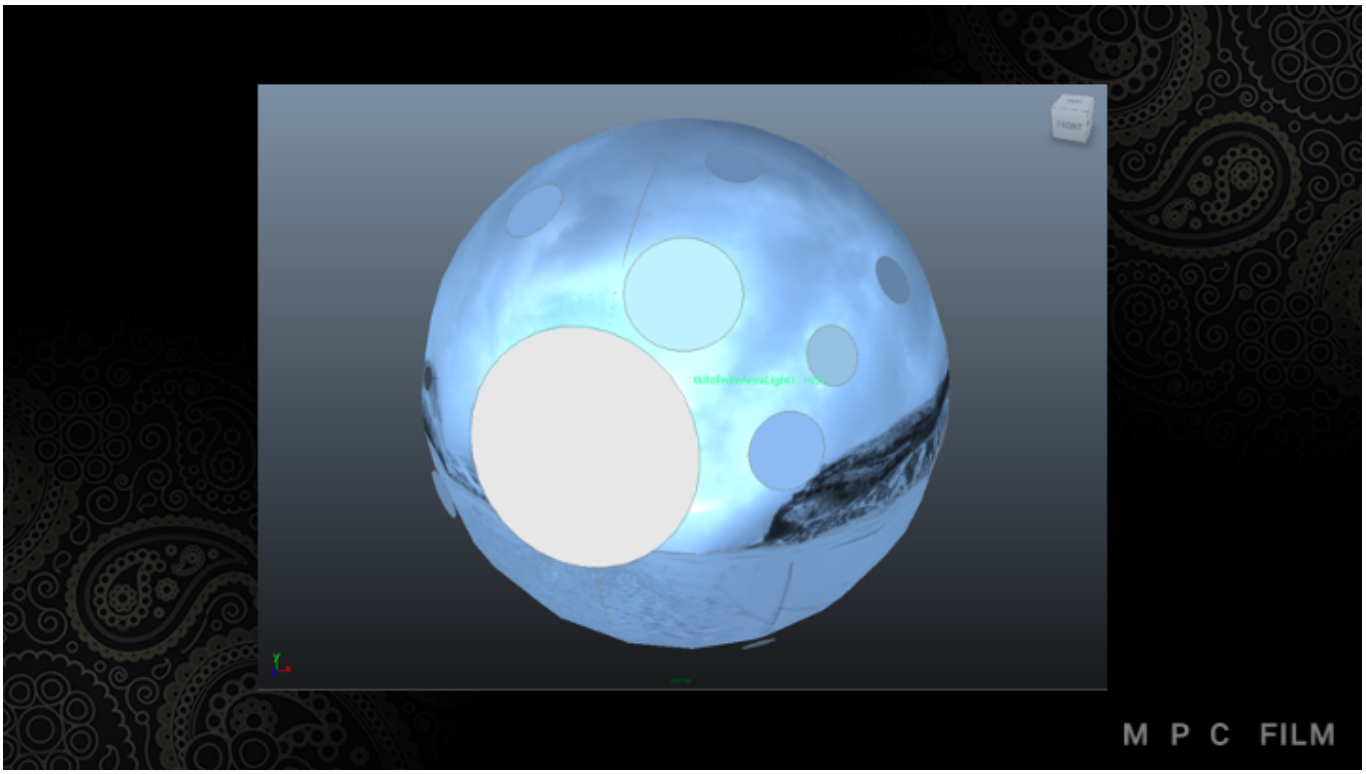
... and figure out their screen-space coverage so we can capture them individually.

This is awesome because each map is now quite tight and efficient, and we can theoretically reuse a lot of data if a later iteration of the render only requires an animation update to a single soldier.

This is not-so-awesome because one shadow map has suddenly turned into 10 or 100 or 1000, and tracking the correspondence between shadow maps and animation changes (or lack thereof) is a significant challenge.

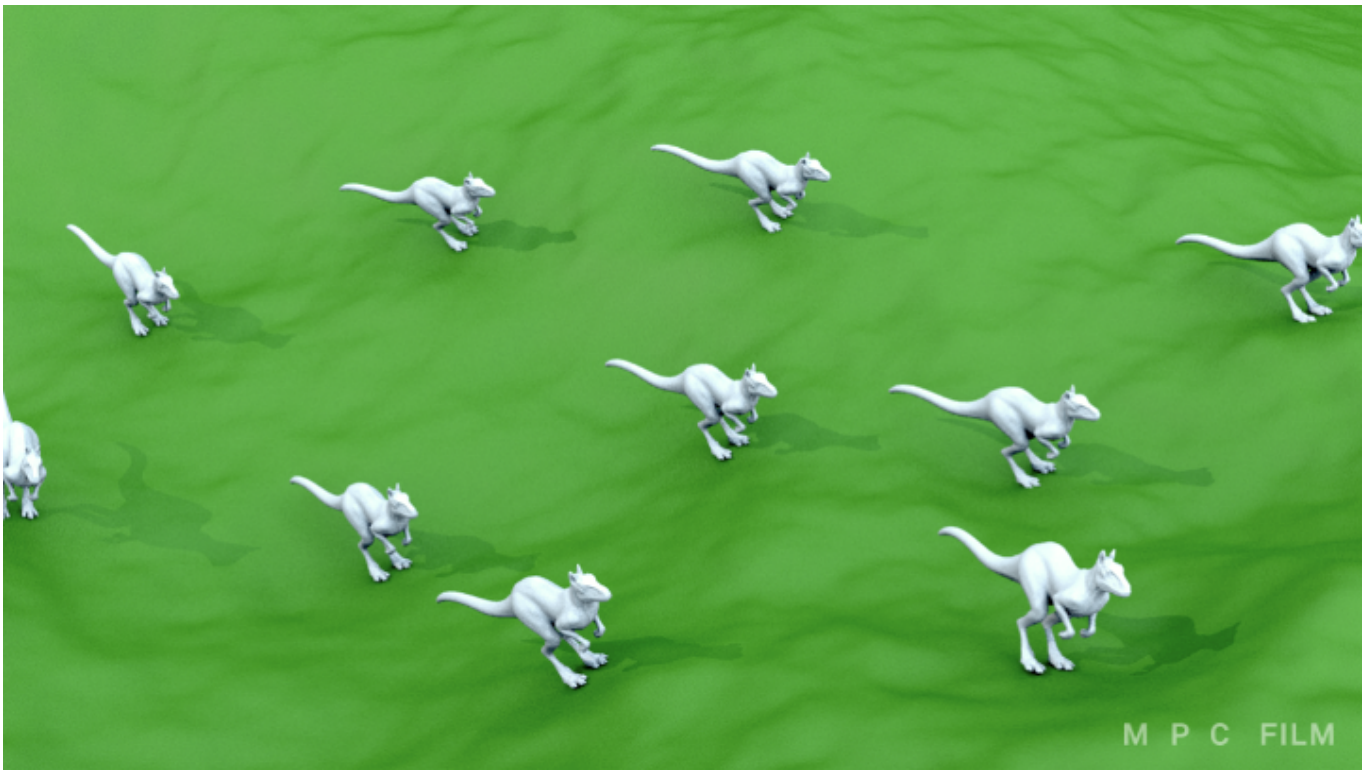


In the early 2000s, MPC was already using image-based lighting to help our CG content integrate more naturally into the filmed content.



We wrote tools which would decompose light probes into a series of directional lights via median cut and similar algorithms. So now we've got dozens of lights per shot, each of which may have several shadow maps. Ultimately this transformed the role of a Lighter into a data wrangler needing to manage thousands of shadow passes *per-frame*.

In short, a data management nightmare.



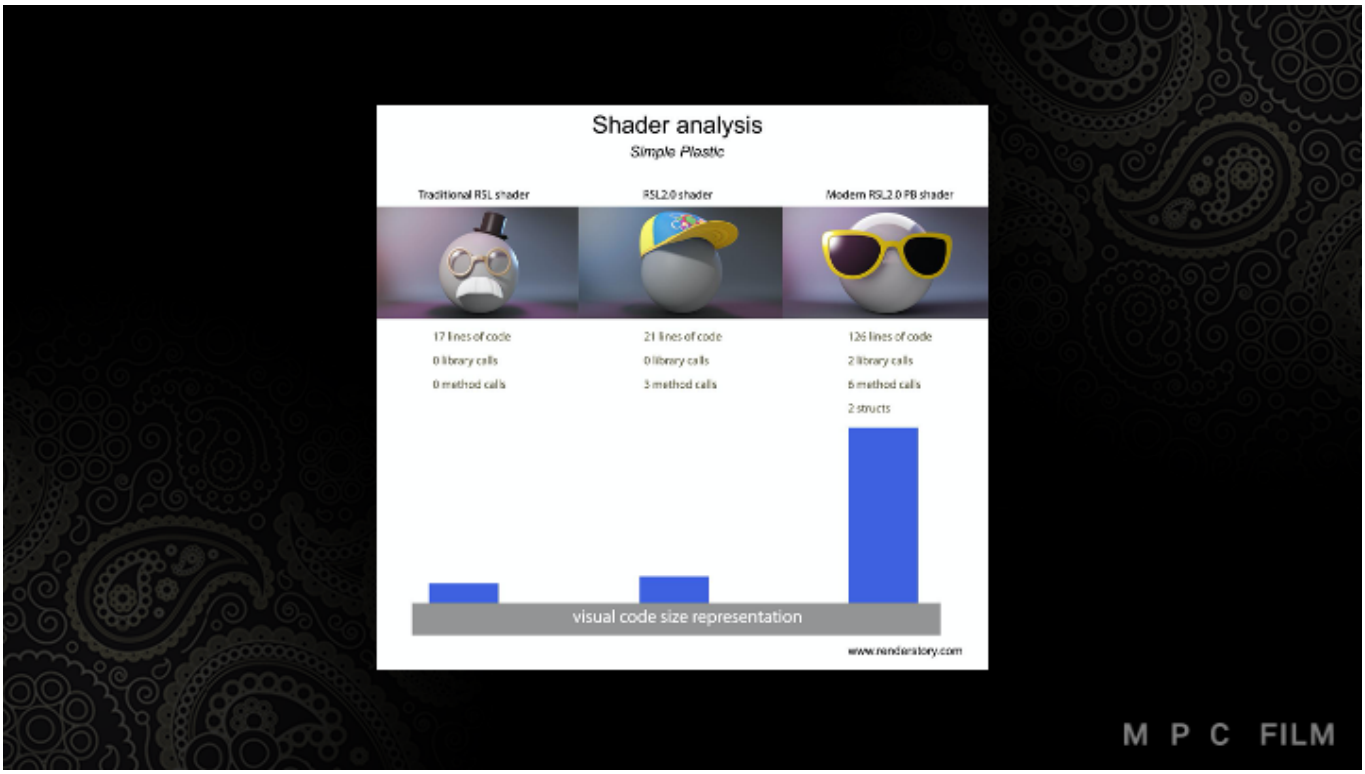
But at least the images looked prettier.

Further complicating the life of our Lighters were the bespoke surface shaders with little to no grounding in real-world physics. This meant they needed to invest heavily in shot-specific Look-Dev to ensure not only that individual assets felt believable in isolation, but also in relation to neighbouring assets. A tree that looked beautiful in one shot might appear to be glowing in a different shot. Similarly, a character standing beside the tree might be getting darker at the same time the tree was getting lighter.

Our road to salvation started 15 years ago, when we got our first taste of ray tracing. It also let us start dabbling with ray-traced shadows and reflections, although in all honesty for many many years we still found ourselves using a lot of shadow maps for assorted artistic control and performance reasons, and just being the devil we already knew.

The most fundamental shift in our approach to rendering came in the form of the Physically Based Shading movement around 2010 or maybe a few years earlier. In preparation for this presentation, I spent a lot of time trying to figure out how to best summarise the impact, and one thing that I kept coming to is that it made us more disciplined and able to reason about our materials and shading. We started to think more about the area of lights, about energy … really about representing the physical quantities of the real world.

In theory this should make the lives of our artists simpler, and one of our first efforts was to pursue materials that were energy conserving, and introduced surface shading based on work by Ashikhmin and Shirley, complemented with “albedo pump-up” based on work by Neumann et al. This was complemented by an in-house framework for importance-sampled materials (where the number of reflection/occlusion rays was driven by surface roughness, for example). We first used this on 2010’s *Clash of the Titans* and talked about it that year in an Importance Sampling course at SIGGRAPH. I’d encourage you to read the notes from that course if you’re interested, and I’m not going to spend more time on it here.



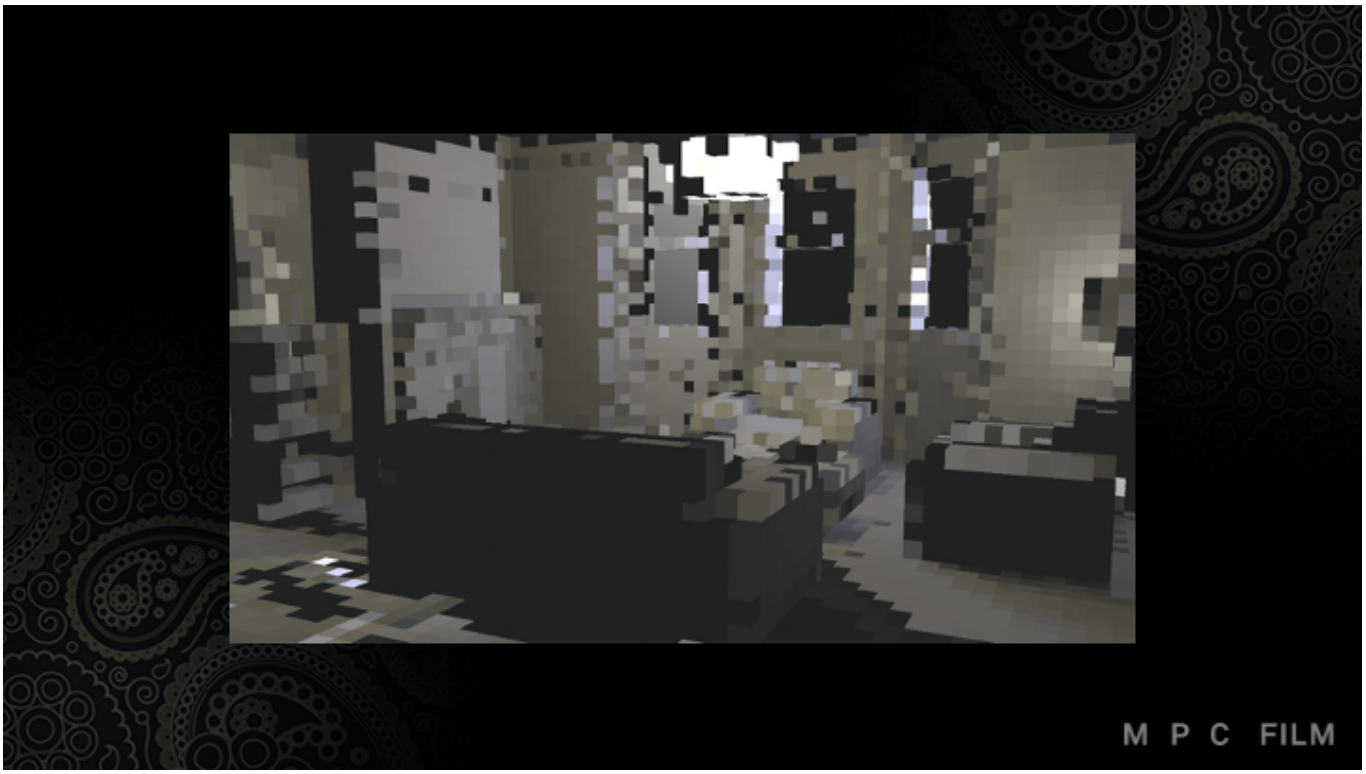
Not long after we started our efforts on PBS, Pixar released their own framework for physically plausible shaders in RenderMan, and we transitioned to use it. This continued our trend to embrace ray tracing, and started a movement away from bespoke materials and towards more general-purpose ones.

Of course nothing good comes for free. Reality is rather complicated, and that complexity ends up manifesting itself in many places, such as in our shaders. Our main production shader at this time went up to about 4,000 lines of code … just for the surface-light interaction; all the pattern generation code was in separate co-shaders.

And we ended up trading one data management problem for another. RenderMan still wasn't really designed at the lowest level to be the world's simplest or fastest ray tracer at the time, so we had to find ways to get the benefits of indirect illumination without brute-force tracing a bajillion rays.

So we’d render direct illumination passes ...

... bake that illumination out into point clouds ...



... convert those into brick maps ...

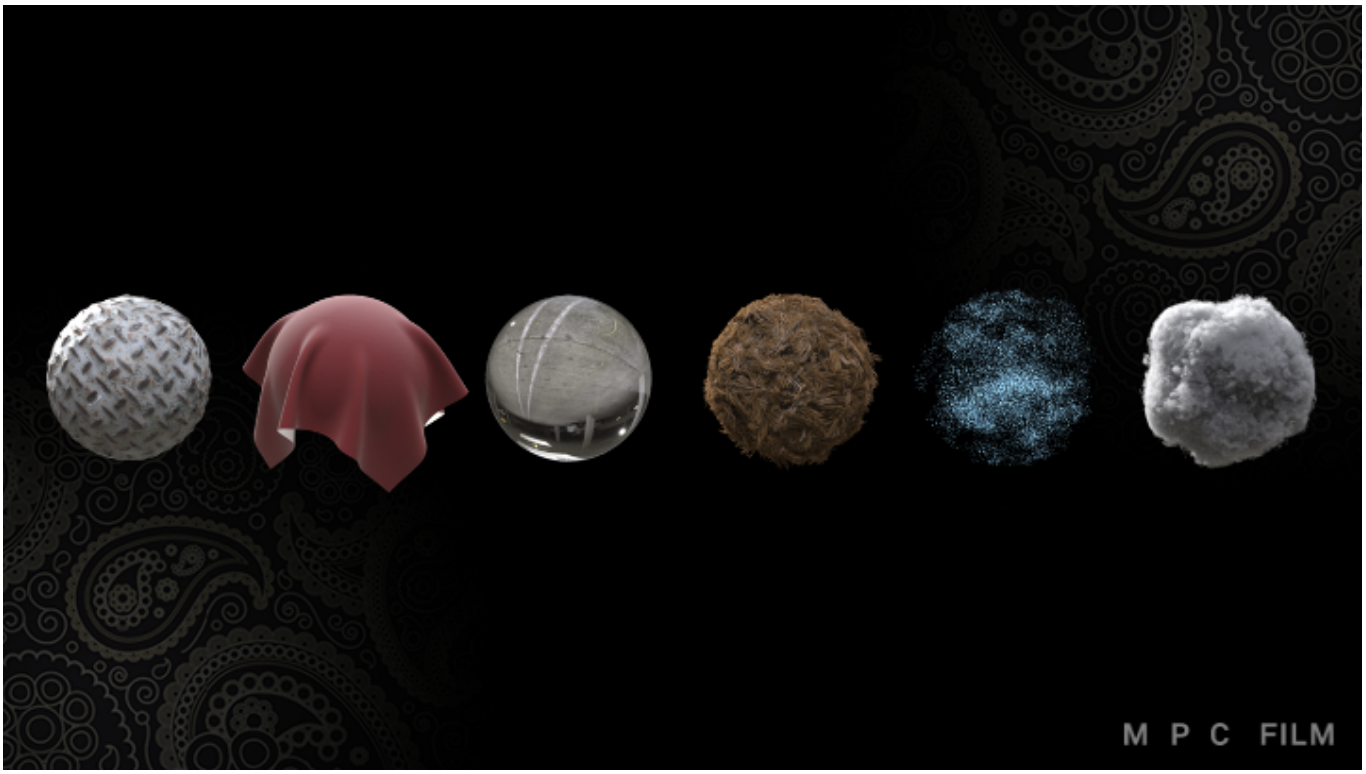
... And then render again, tracing secondary rays against these caches for the bounce lighting. And the result is pretty darn good. Especially if we compare this image ...

... against the previous one.

But we still have a multi-pass render, with point cloud and brick map artefacts along the way.

~5 years ago we had another major paradigm shift in our rendering as Pixar added the RIS framework into RenderMan. We buried our shadow maps and point clouds and brick maps deep in the ground, and embraced the world of single-pass path tracing.

This also meant throwing away our entire shading library and, for better or worse, starting again from scratch.



Initially we started by building a fairly small but distinct set of BXDFs for different material types. We had separate plugins for general hard surfaces, cloth, glass, hair, particles, volumes, and eyes (which I left off this slide because a photorealistic eyeball is a bit gross to look at). These plugins were a combination of custom code and work from the RenderMan team.

Over time we decided to consolidate our materials (for example pulling cloth and glass into our general hard surface). This increased the complexity of the remaining BXDFs, but made it easier to think about material layering and blending, a topic we'll talk about later.

While our general trend recently has been to move towards vendor-provided materials (for example our use of RenderMan’s PxrSurface), the glaring exception has been our continued investment in custom hair shading.

One of the very first films we rendered using RIS was Disney's 2016 movie - *The Jungle Book*. We delivered this project using a custom hair shader based on the popular Marschner model although it was largely successful, but had a number of limitations that we would like to improve.

Thankfully a new huge show was coming and it was a request from production to further improve our fur and hair shading.

For Disney's *The Lion King*, we needed something that would make the show distinctive in terms of the fur look; the progress in shading had to be noticeable. The close-up performances meant our curves could no longer look like ribbons, and the pale lion fur meant we needed to properly investigate the physical properties of hair and colour attenuation.

There's been lots of great research into fur rendering in the last decade, especially in the last five-ish years.

We spent time to investigate all the works which were done in the last time related to Fur/Hair shading and decided to implement quickly what was written in the PBRT supplementary chapter dedicated to that aspect of the shading (https://www.pbrt.org/hair.pdf).

Initially we were also very much inspired by the work *A Practical and Controllable Hair and Fur Model for Production Path Tracing (Chiang)* from Disney explained in their 2016 paper. But as soon as prototyping was done and the main points of that paper were implemented, we understood that we need something more physically plausible in terms of parameterization, as hyper photorealism and physically plausibility of the shading were the main requests.

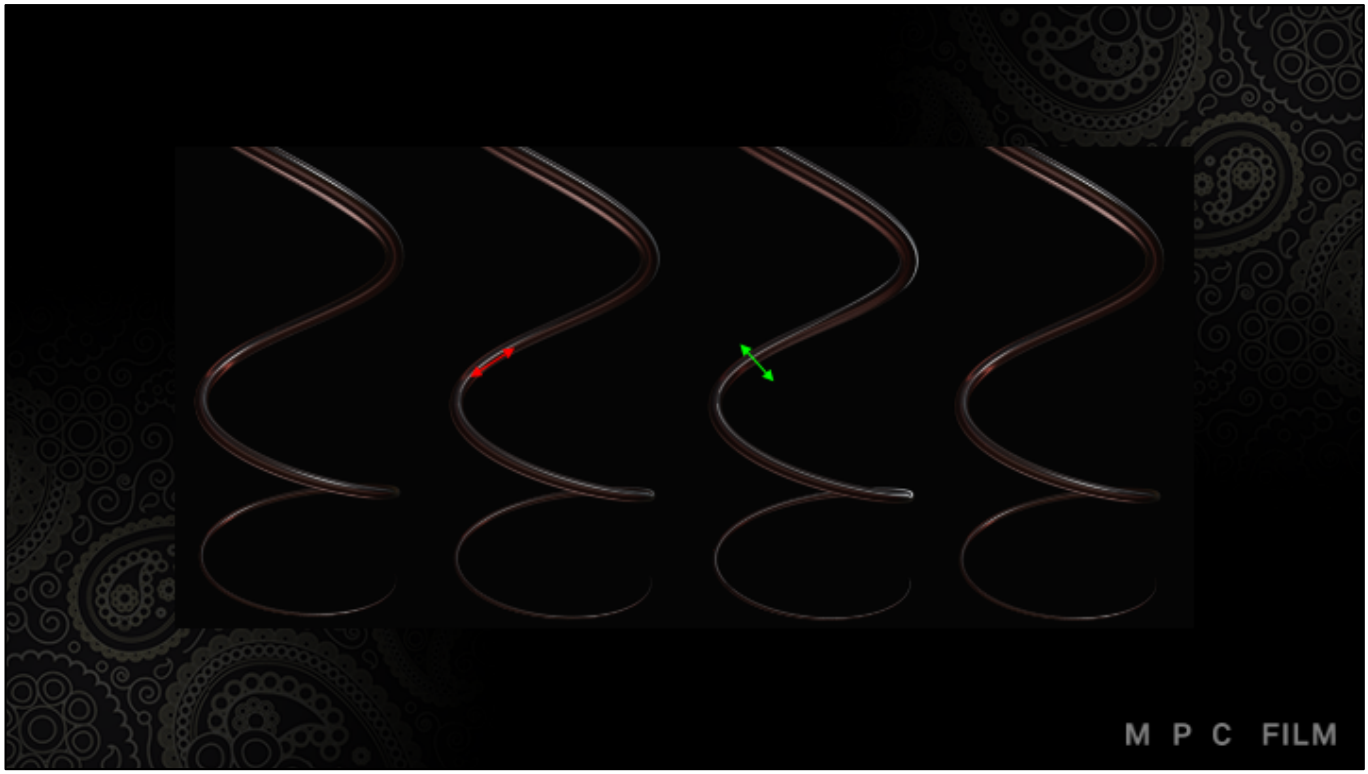
So we diverged from many aspects of Disney's work and were mostly focused on the *Physically Accurate Fur Reflectance: Modeling, Measurement and Rendering, 2015* and *An Efficient and Practical Near and Far Field Model, 2017* papers from Yan, as these series of papers give a deeper explanation of how fur works and what kind of parameterization should be implemented. The provided tables with measured values of different species of animals would give us an initial and approximated parameterization which we could then extend in future internal lookdev iterations.

{ VIDEO DIALOGUE }

There's been lots of great research into fur rendering in the last decade, especially in the last five-ish years.

Initially we were also very much inspired by the work *A Practical and Controllable Hair and Fur Model for Production Path Tracing (Chiang)* from Disney explained in their 2016 paper.

We diverged from many aspects of Disney's work and were mostly focused on the *Physically Accurate Fur Reflectance: Modeling, Measurement and Rendering, 2015* and *An Efficient and Practical Near and Far Field Model, 2017* papers from Yan, as these series of papers give a deeper explanation of how fur works and what kind of parameterization should be implemented.



Ensuring that our curves had a cylindrical (rather than flat) look became one of the main request. The coming show supposed to have a lot of close ups where we would see individual strands of fur and hair.

Longitudinal/Azimuthal roughness became a default request to the new shader and was the main means by which we could control all the lobes. By this I mean there were no extra roughness values for each lobe; all the properties of the consequent lobes were computed automatically and had to be be physically plausible.

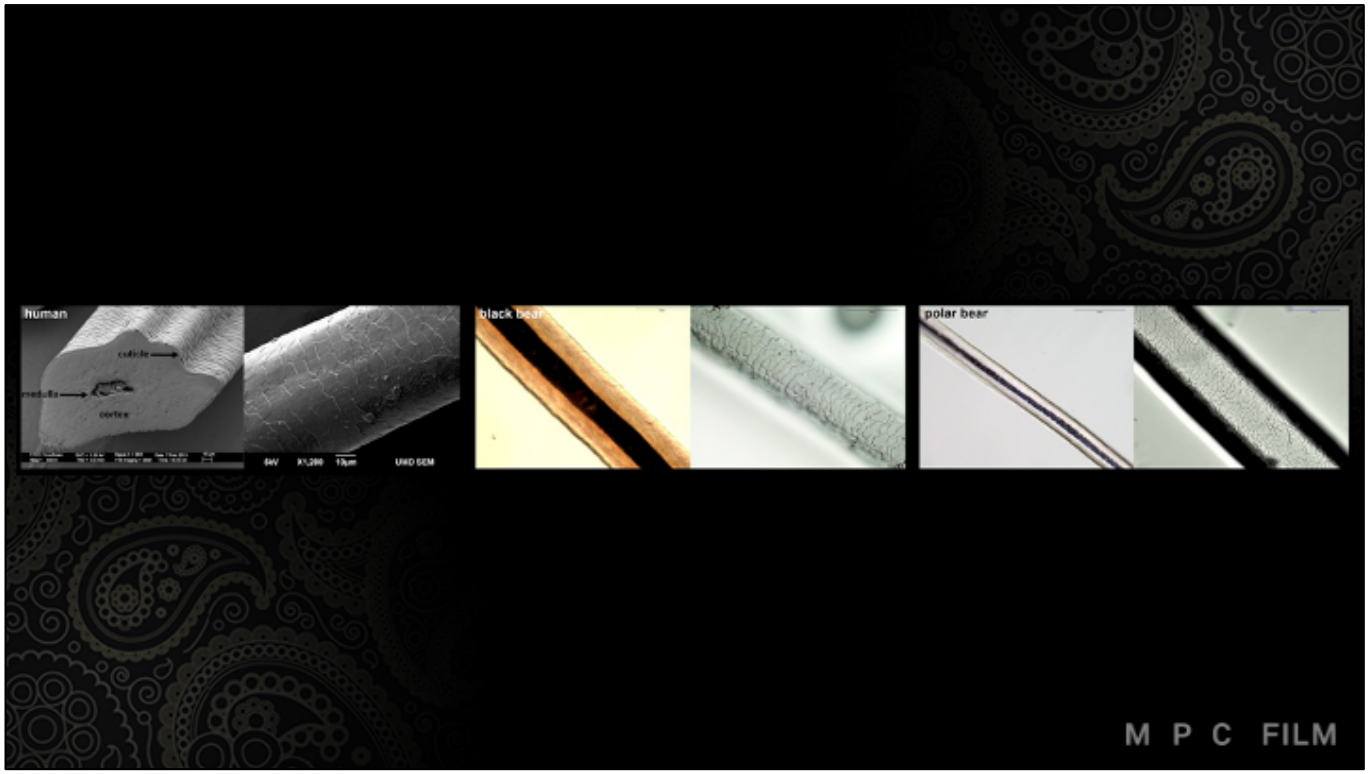
Another goal was to make the controls as minimal and easy to use as possible. But this couldn't extend to the point where our artist's creativity was hampered as we had to be able to achieve the look of all possible species of animals.

Natural light scattering and a proper color parametrization was requested as well. Using the diffuse lobe as the main lobe to set the color of the groom was not acceptable; the color had to be accumulated due to the multiple bounces across the groom.

All the lobes supposed to be well balanced in terms of energy conservation and, of course, the performance of the shader had to be as good or better than what we had before.

As mentioned, we'd started our prototype based on PBRT supplementary chapter and some of the aspects from Disney's 2016 work.

But we started to get deeper into the subject of fur and began analysing what we could improve in terms of physical parameterization to get even more realism. We started by looking deeper into the study of fur structure and how fur is physically composed.



Fur vs Hair structure

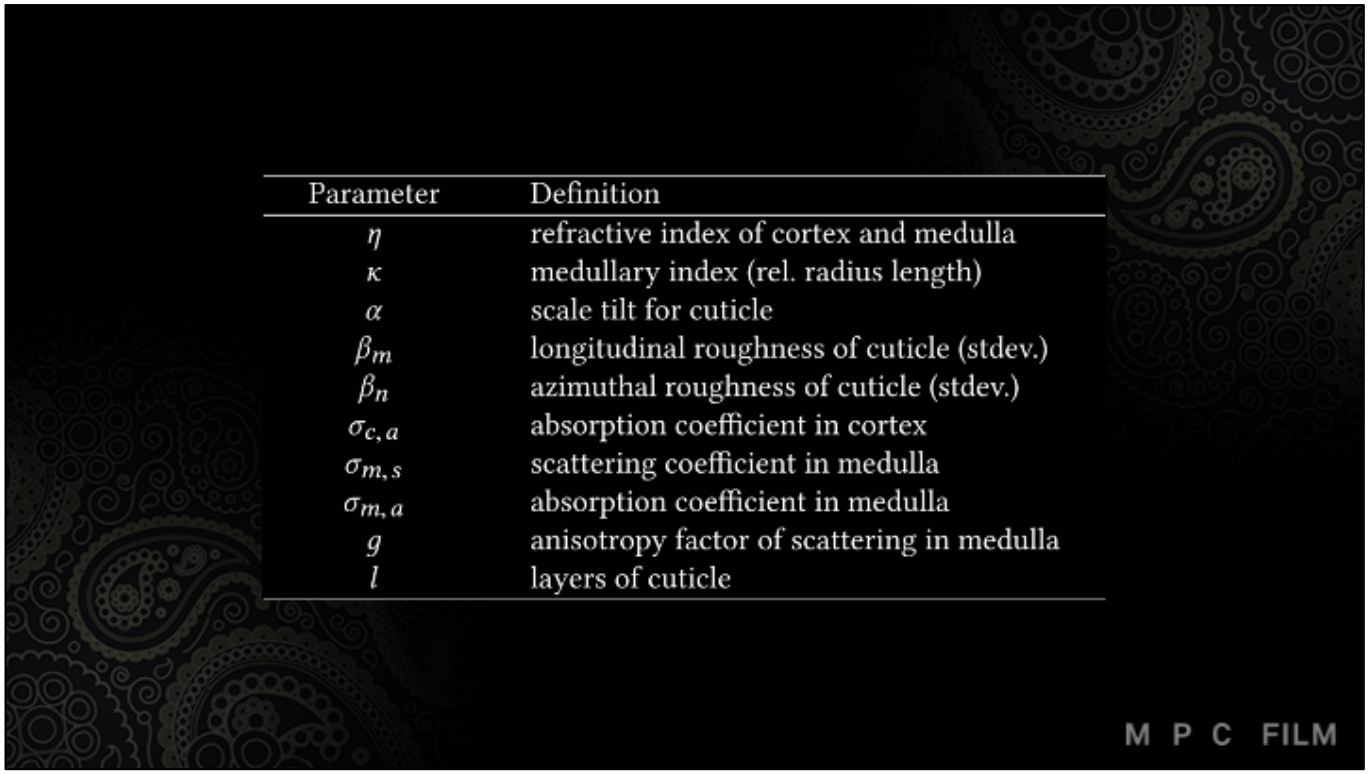
This series of images contrasts human hair against the fur of animals.

The two left images show the internal structure and cuticle layer for human hair; the four right images highlight the medulla and cuticle layer of animal fur fibres.

Hair and fur share some common structures. They are often cylindrical with some eccentricity and, in both cases, there are three main structural layers:

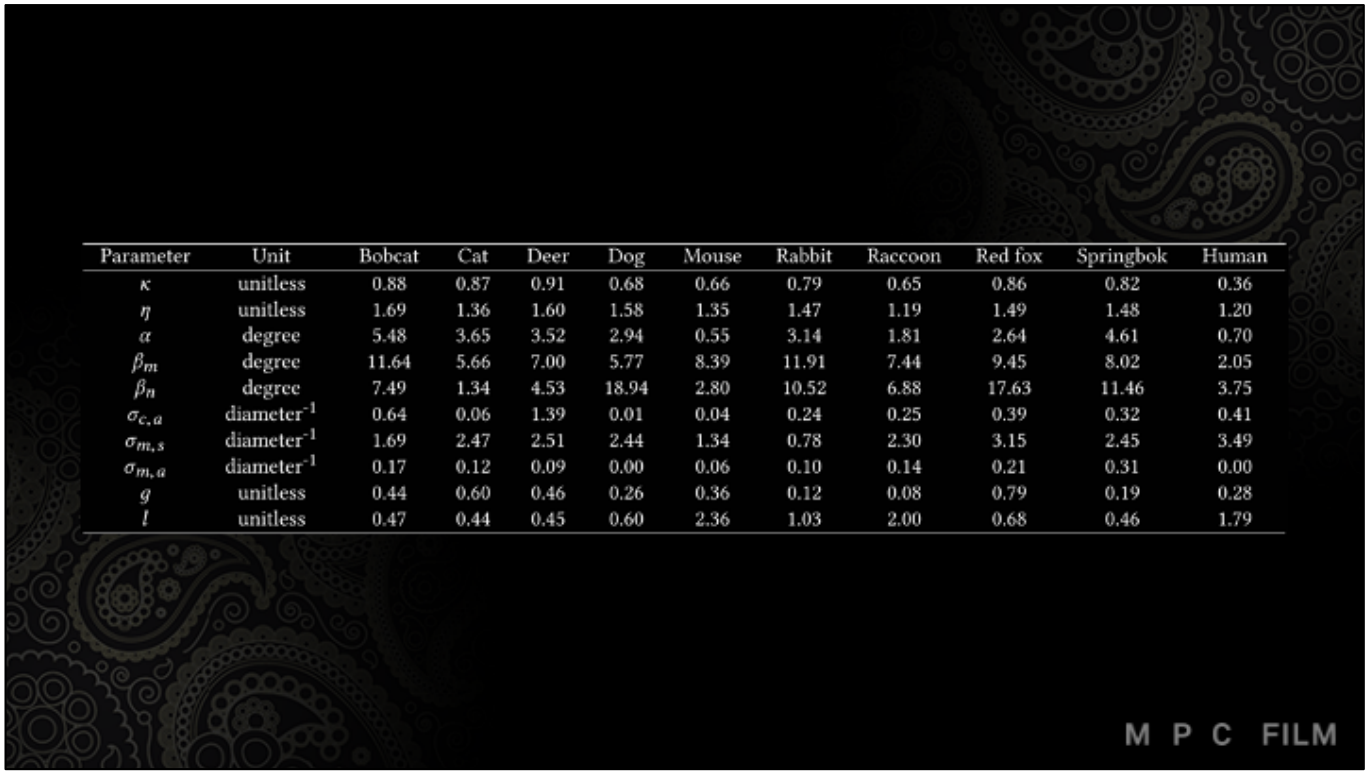
- the cuticle, which covers fibres with inclined scales
- the cortex which contains nearly all the colored pigments within the fibre
- and the medulla which lies in the centre of the fibre with complex internal structure that scatters light

There are also some notable differences. I'll draw your attention to both the inner medulla core (which is quite significant in fur but very small in human hair), and the outer cuticle layer (which has more complex structural detail in fur fibres).



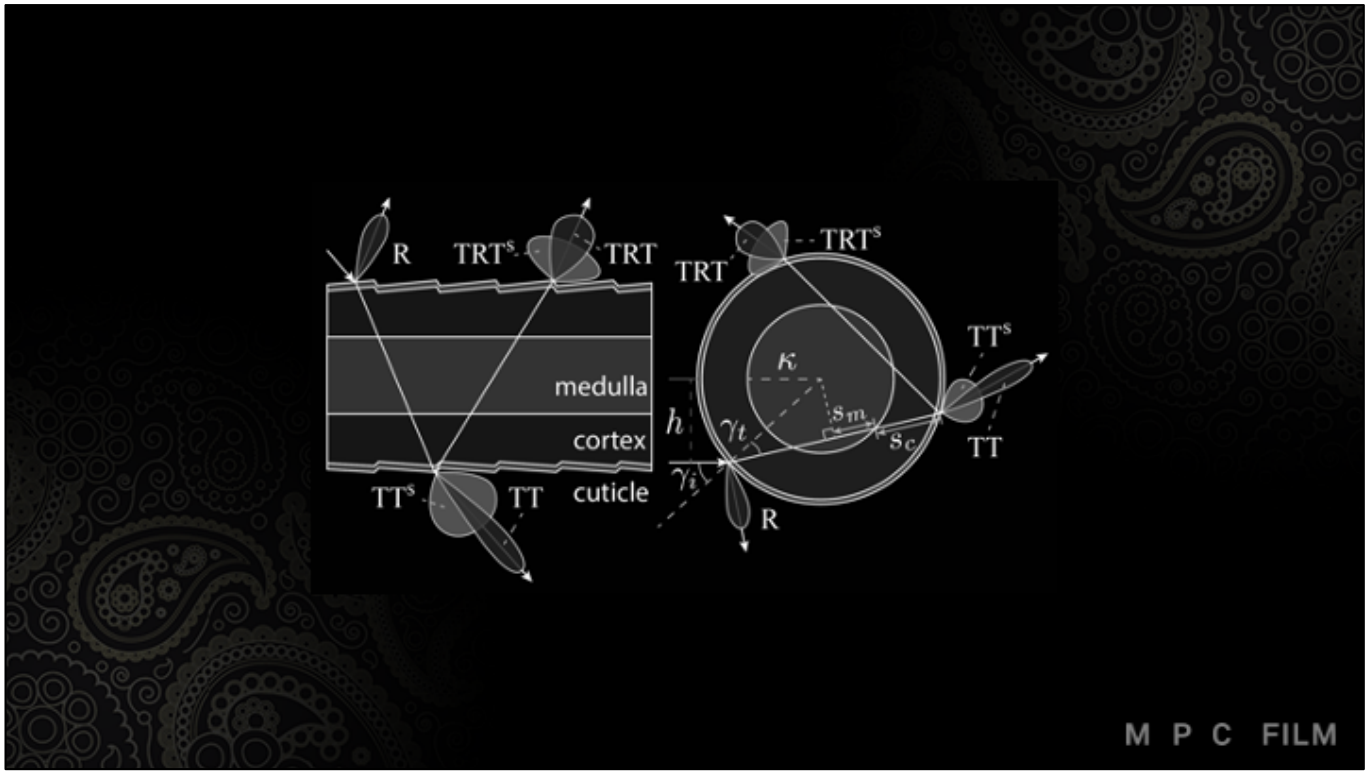
Physically Based Parameterization

We decided to get as close as it possible to the parametrization described in the *An Efficient and Practical Near and Far Field Fur Reflectance Model, 2017* paper by Yan and parametrize our medulla as a volume based on absorption, scattering coefficients and a phase function.



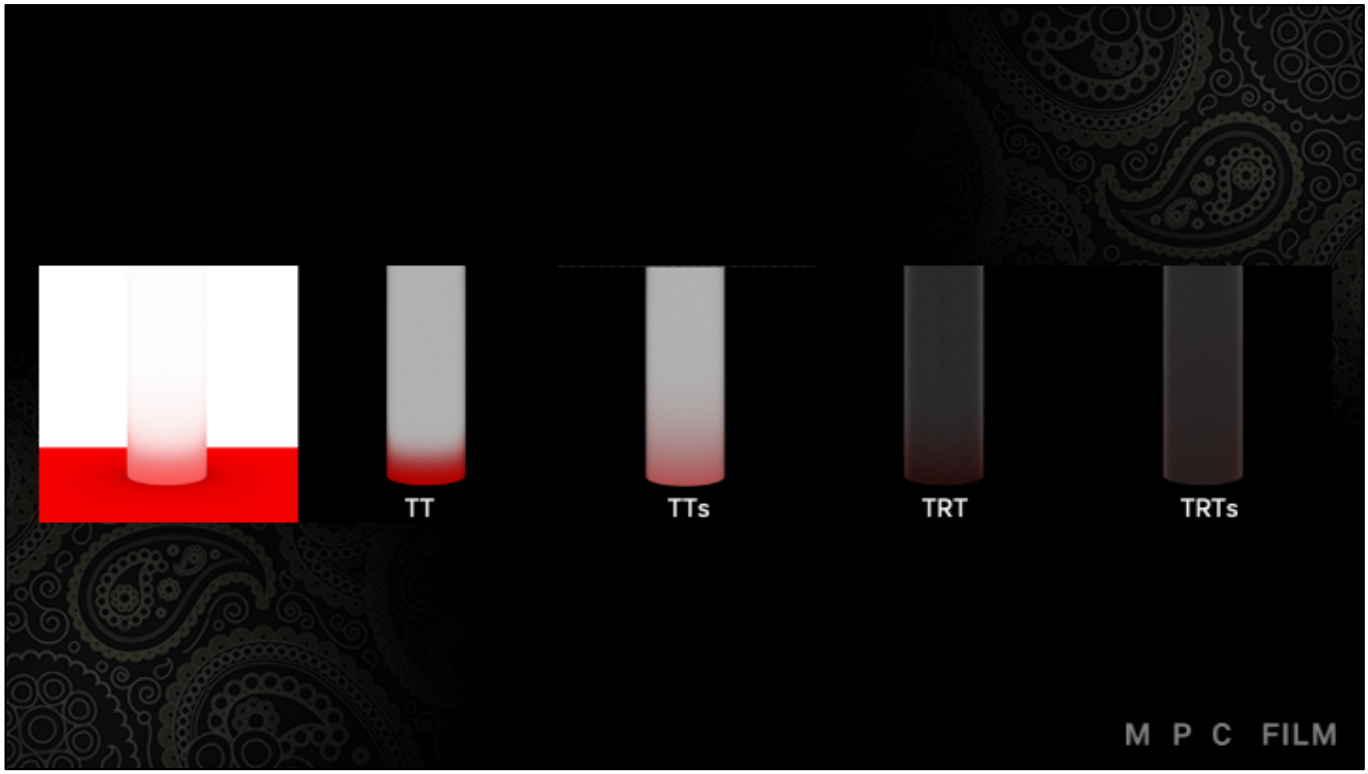
Physically Based Parameterization

We wanted to be able to understand in numbers how fur behaves for different species of animals. The table supposed to be a starting point for LookDev to further tune.



Our custom fur shader went through two major iterations, which you'll see in a few slides. While they both considered the structure of fur to be based off of the previously mentioned components (cuticle, cortex, medulla), one of the major differences between them was the parameterization of the medulla.

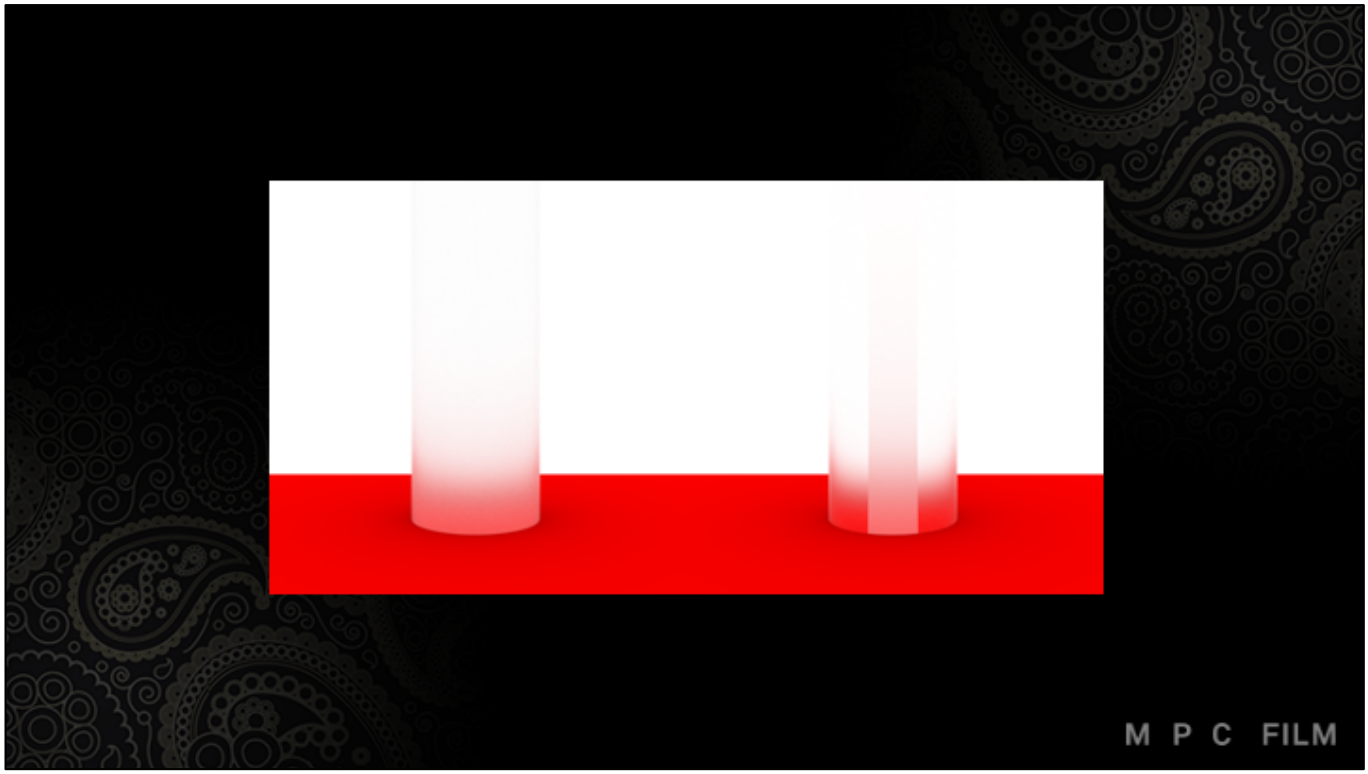
At the beginning for our first iteration, which we call Fur v1, we made an assumption about medulla based on a schema from the paper *Physically Accurate Fur Reflectance: Modeling, Measurement and Rendering, 2015* by *Yan* paper. We decided to add two extra lobes (TTs and TRTs) which had a wider angle of sampling; this allowed us to control the scattering over the groom and make look more diffusive when it is needed.



Fur v1

Here we can see the different lobes we got from Fur v1.

But the medulla contribution to the final look was weighted by an extra parameter, something like “Medulla Intensity”, which had a non-physical meaning compared to what we really wanted which was a “Medulla Radius”.



Fur v1 vs Fur v2

And this led to Fur v2, which you can see in the image on the right.

As mentioned, we wanted to control medulla contribution not as a weight of medulla lobes, but literally controlling the size or radius of the medulla, as that would gave us much more correct color attenuation. For this goal we wanted our shader to emulate a second cylinder, simulating the medulla core inside the cortex, acting as a scattering medium with actual radius. This is known as the Double Cylinder model and was suggested by Yan in his 2015 paper.

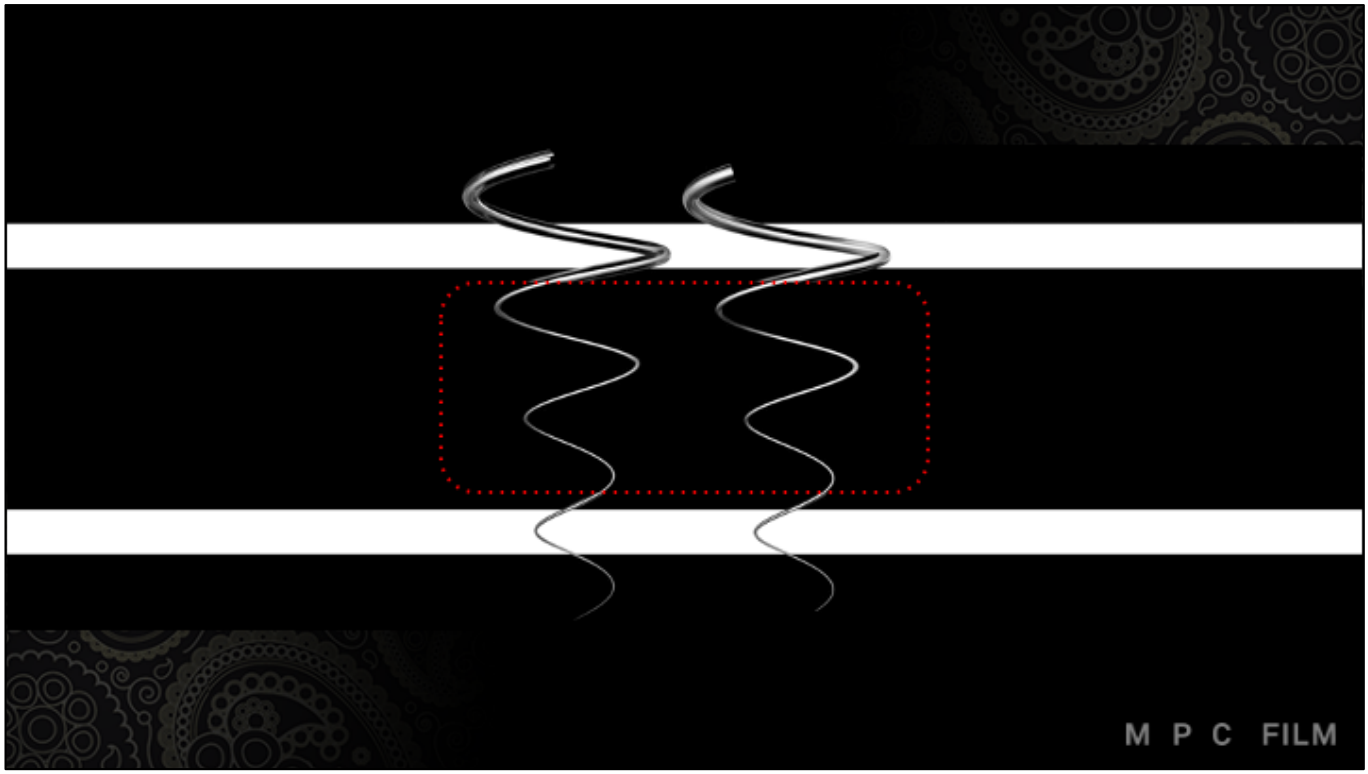
We also wanted to control the light scattering, by specifying it as forward, isotropic, or backward. Ultimately, the goal was to get closer to the parametrization provided in the Yan 2017 paper.

Linking both cylinders back the the structure of fur:

The outer cylinder can vary with a cuticle layers parameter, adjusting the ratio between the reflected and refracted light.

The inner cylinder represents the medulla. It doesn't absorb light significantly, but scatters light when light travels inside.

And between these two cylinders is the cortex, which simply absorbs light.

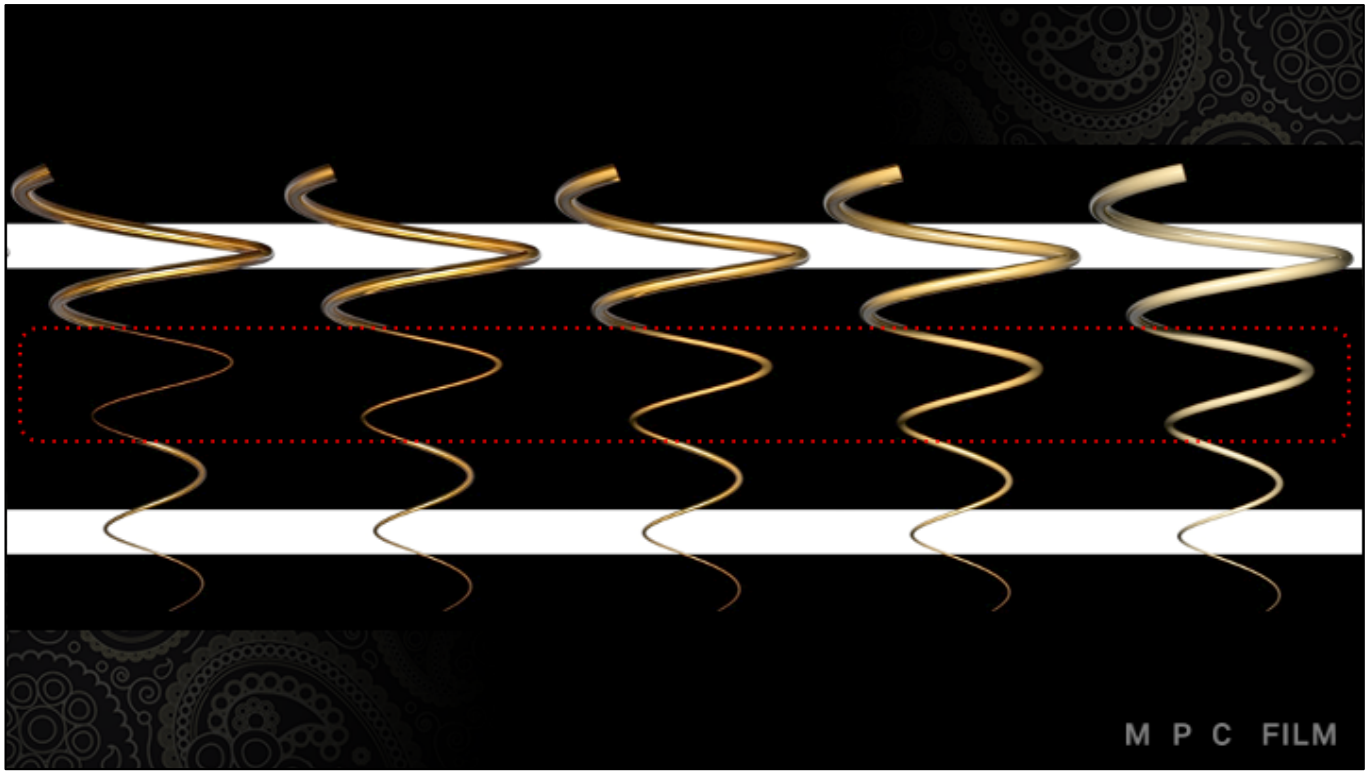


During the development of our shader, it was important to match the shading to a real geometric ground truth.

On the left is actual modeled mesh geometry with a glass shader assigned to it. There are actually two nested geometries; the inner one representing the medulla.

On the right is a curve with our custom shader applied. Hopefully you agree that the look matches pretty closely.

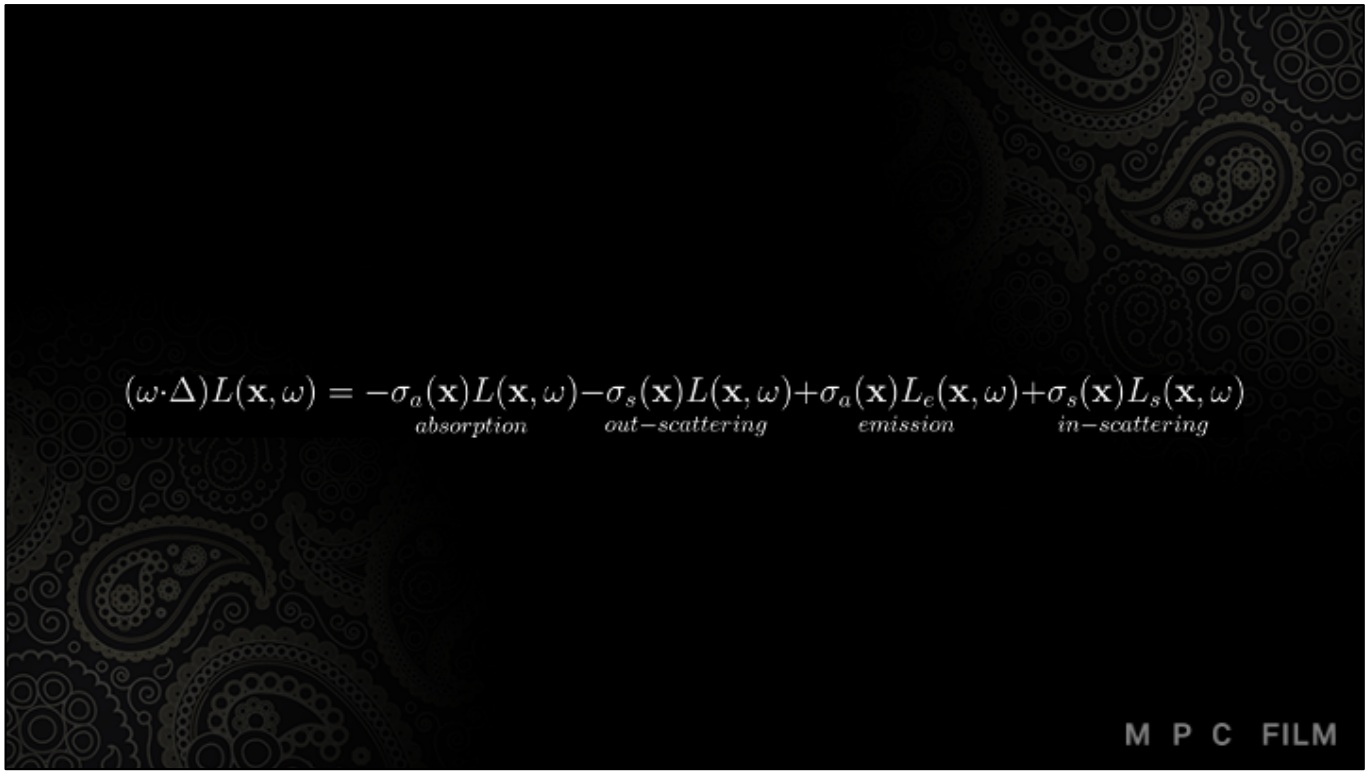
Inside the red box we discard the outer shell and only visualise the inner medulla. It’s subtle on this slide but this isolation tells an interesting story in the next slide.



Medulla Radius

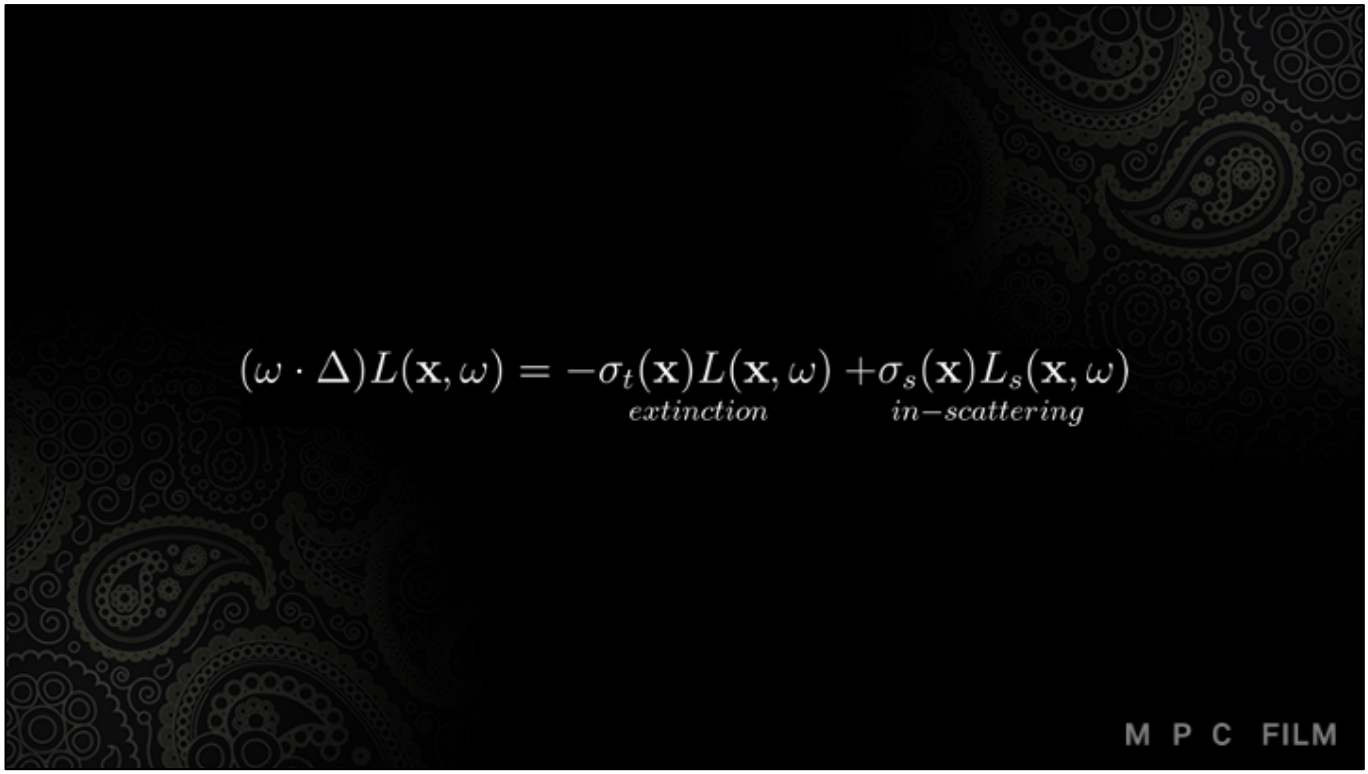
Being able to control the medulla radius gave us the ability to change the look quite dramatically. Here you can see the impact of increasing the radius from left to right, again highlighting just the medulla in the red box. On the left we have a very narrow medulla; on the right the medulla takes up almost the entire curve.

According to Yan 2015, 2017 research papers medulla behaves very much like a volumetric medium.



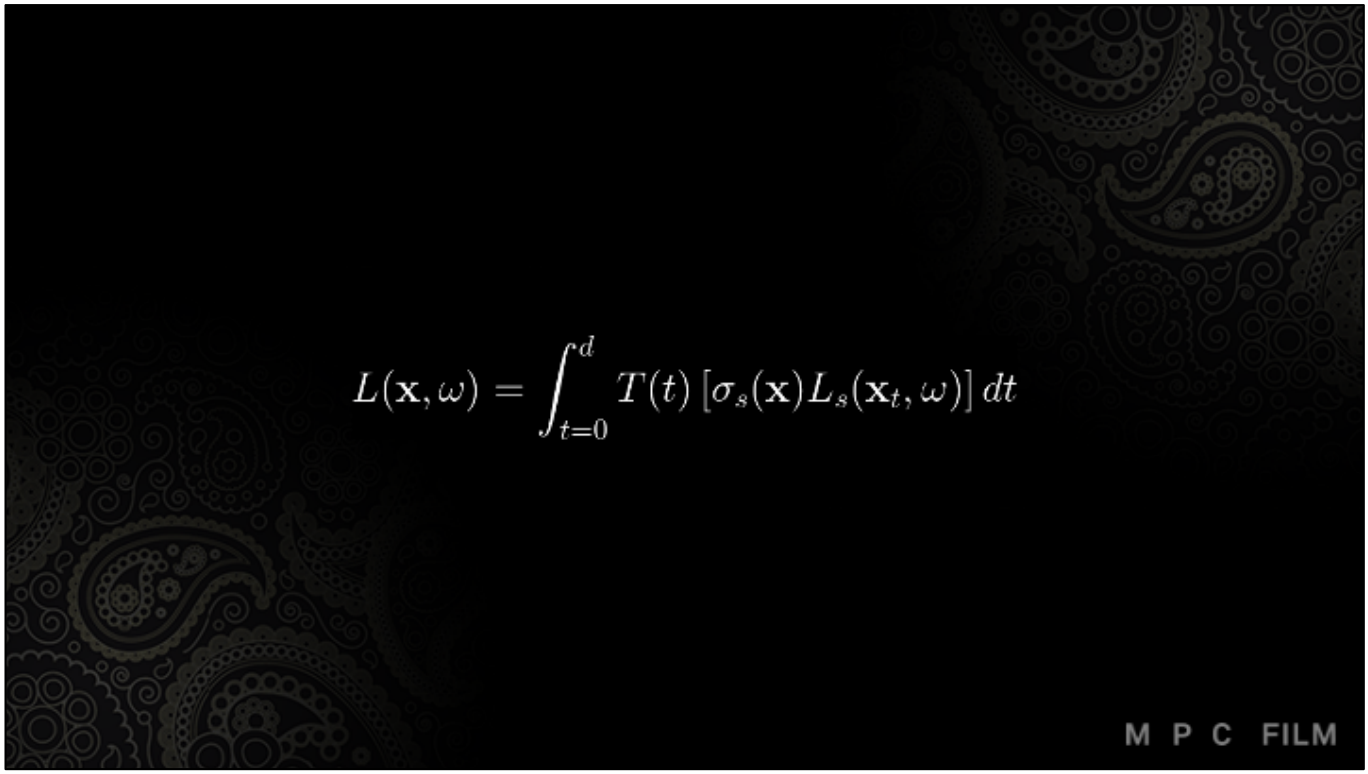
Light Propagation in Volumes - Radiative Transport Equation

Let’s consider the radiative transport equation, which represents the distribution of radiance in volumes and try to understand which parts of it could be considered for medulla simulation.

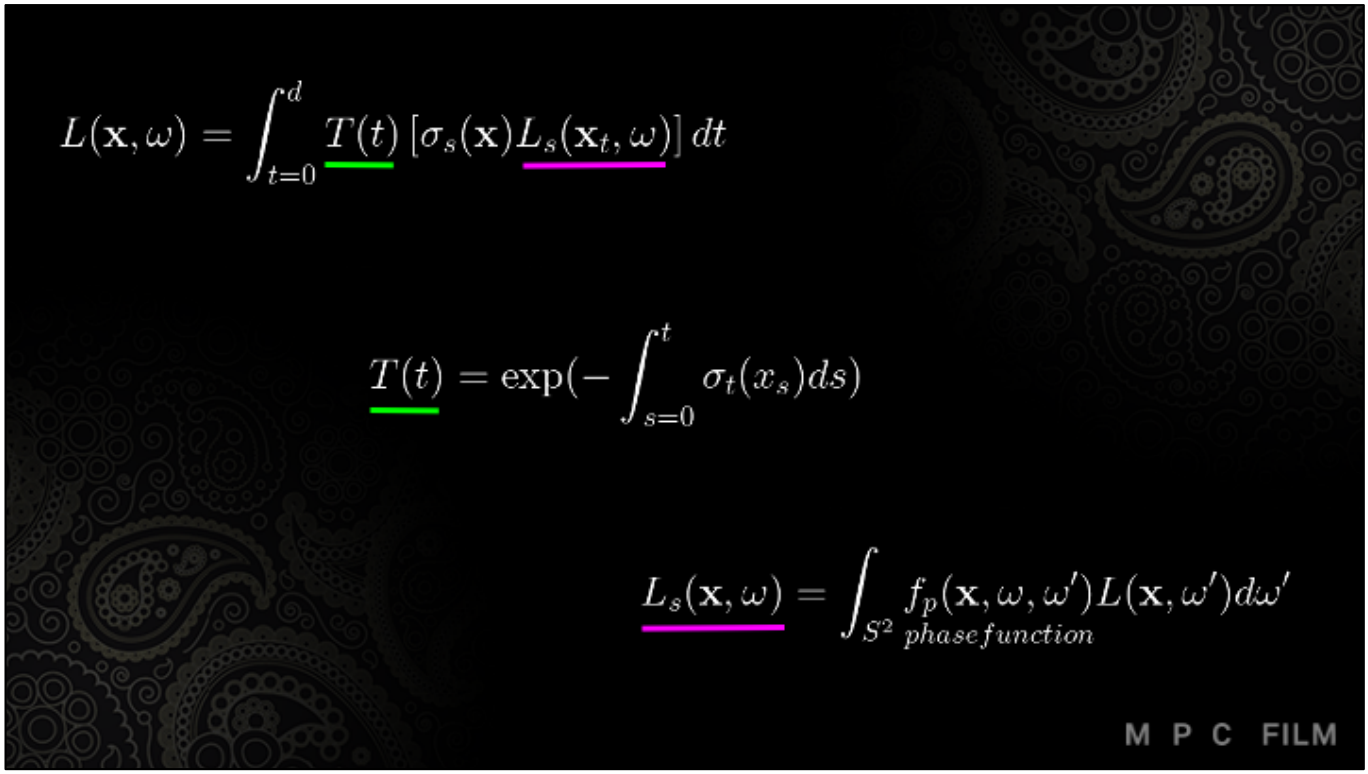


Light Propagation in Volumes - Radiative Transport Equation

We got rid of emission, and merged together absorption and out-scattering into the extinction coefficient.



Now, let's represent this in the **Volumetric Rendering Equation** form.

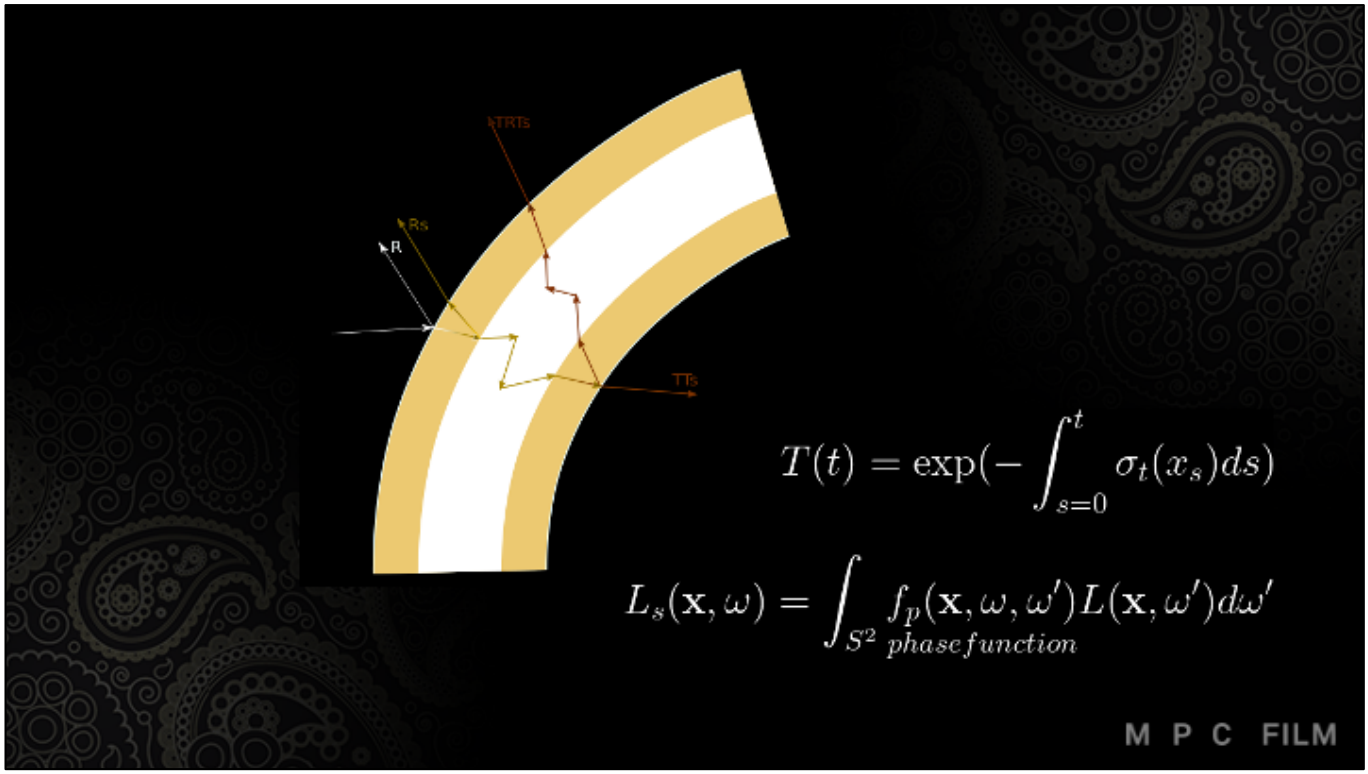


Volume Rendering Equation

Here we have two important components:

The first exponential member of the integral is Transmittance which is responsible for light attenuation and includes both absorption and scattering coefficients called also extinction factor.

And the second member which includes phase function is Scattering.



Looking at this diagram we have to apply a transmittance formula for each part of the path for each lobe keeping in mind that cortex and medulla has different absorption coefficients and cortex doesn't scatter the light mostly behaving like a glass.

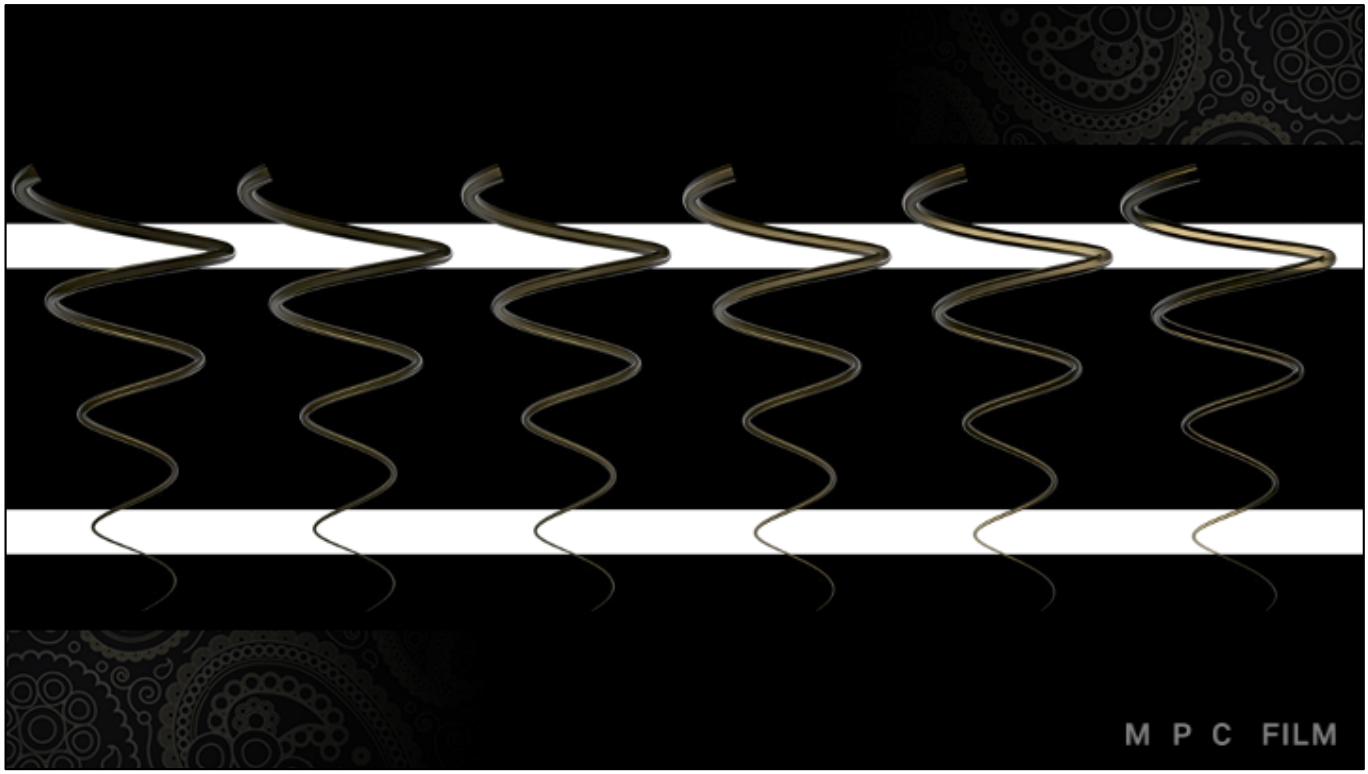
Medulla behaves like a volume, that means that the radius of medulla and it is absorption coefficient as well as the width of curve affect the final color.

And if absorption coefficients is something that is based on the input color and then converted into absorption coefficients for both cortex and medulla as the initial values.

In-scattering component is quite tricky to simulate.

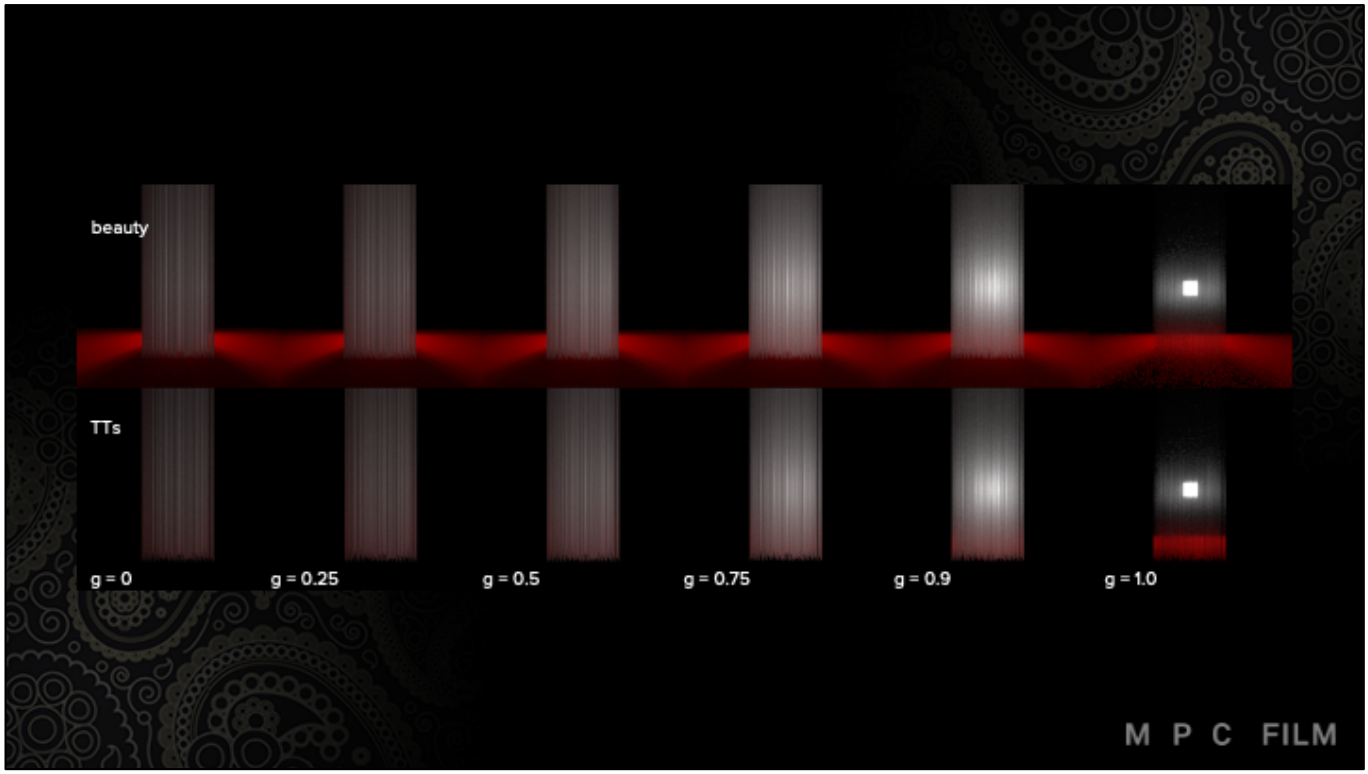
Although scattering is happening on the particle level and phase function is the angular distribution of light intensity scattered by a particle at a given wavelength due to the particle/wave duality of the light we decided to generalize Henyey-Greenstein phase function to the scale of curve and don’t take into consideration multiple scattering inside the medulla.

That kind of approximation became the main idea in our medulla shading.



Phase Function - [0; 1)

So here you can see how the phase function affects the look of the curve. The value changes from 0 to 1 from left to right. For fur medulla the phase function is positive, so the scattering is either isotropic or has a forward direction.



Phase Function (g) - [0; 1)

Here we similarly demonstrate how the medulla’s phase function affects bunches of hair. As g increases, more light goes straight through.

Although there is no scattering at the particle level, there is still a scattering on the broader groom level between fibres which we can control using phase function.

Local Scattering - [0; 1]

There are some difficulties to introduce visual density of a groom which would made a groom less scattered and more diffusive. In the volume case, we can set the scattered distance shorter (the distance between particles), make it denser (when the scattering distance is ~0 the volume behaves diffuse-like), but it is quite undesirable to increase the density of the groom as it will change the look of the character and increase memory consumption and we can't control scattering on particle level due to our generalisation.

That is why we introduced Local Scattering parameter which is a blend between phase function and diffuse. Such a combination is supposed to simulate local scattering properties of the medulla, making it more dense visually. We can use it in both cases: when groom should look more denser and when scattering distance should be shorter.

Here you can see how look changes when we change a Local Scattering of the medulla from pure scattering behaviour - left, to pure diffuse - right.

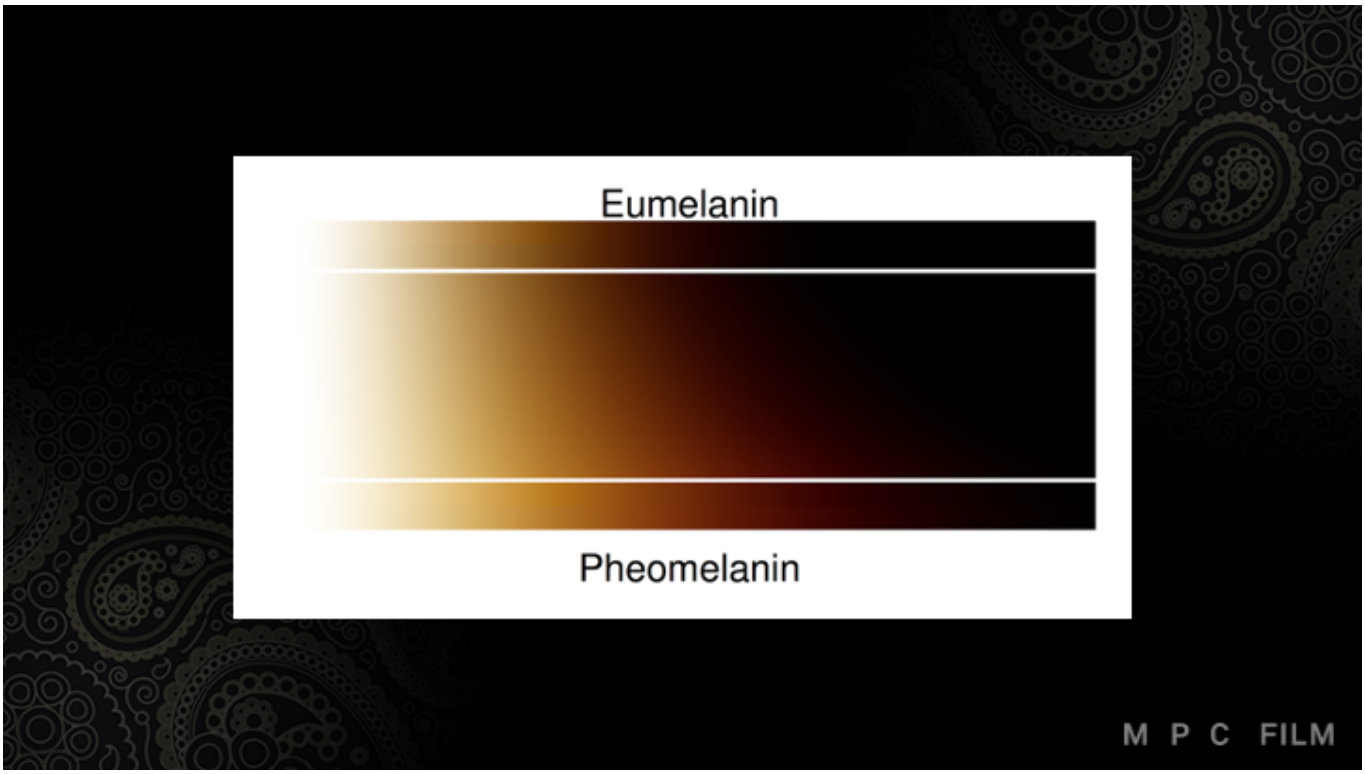
{ VIDEO DIALOGUE }

Here you can see how look changes when we change a Local Scattering of the medulla from pure scattering behaviour - left, to pure diffuse - right.

Curve thickness and overall fur density affects the transmittance and scattering and, by extension, the final colour. Our artists may want to decouple the geometry and shading, controlling each individually, and so we exposed a curve width parameter in the shader itself. The image here demonstrates, from left to right, the effect of emulating increasing width via the shader while leaving the geometry as-is. You can see shifts in colour, saturation and brightness.

In our new shader, colour is no longer picked up by a diffuse reflection but, rather, accumulates as the result of scattering events within the medulla inside the fur. In short, it’s gone from a surface effect to a volumetric one.

The colour we achieve comes from absorption in cortex and scattering in the medulla across the multiple bounces.

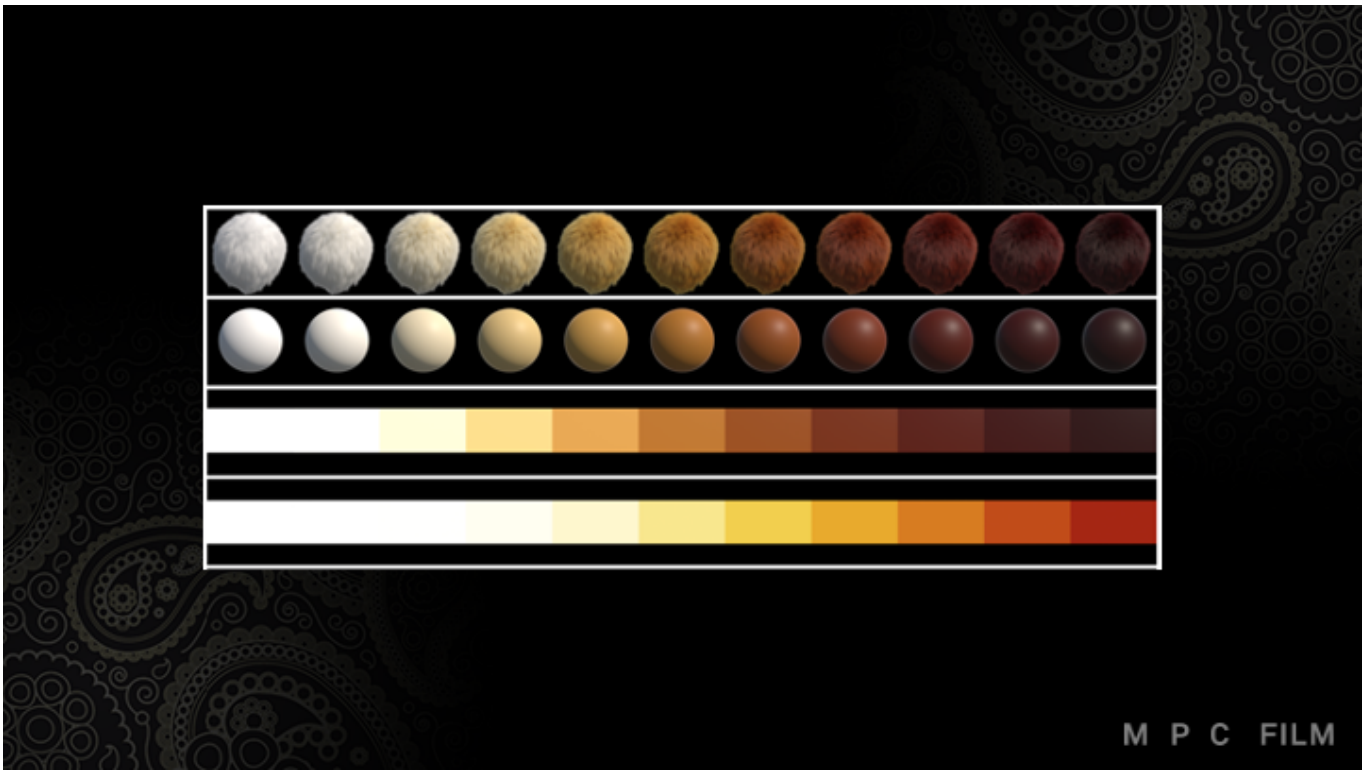


Specifically, we get colour in real hair because of the presence of a substance called melanin mostly in the cortex layer.

Eumelanin causes hair to be brown/black. Pheomelanin causes hair to be red.

Rather than letting artists pick random colours for fur, we provided them with a swatch like this, which let them specify a colour appropriate to an actual melanin concentration value ...

But that brings us back to this single-strand vs fur-ball comparison, and the obviously different appearance of each. Our artists knew what colour they wanted the final result to look like but, given that it resulted from complex volumetric scattering in a multi-bounce lighting situation, it wasn't obvious how all the equations should be seeded.



We’re not the first to observe this. The 2016 paper by Disney basically describes the same thing. The artists at Disney wanted to match the colour of fur to the surface colour of spheres by specifying a common input colour - shown in the third row. The team worked out the math that internally would generate the colours in the fourth row which, when used to drive their fur shader, would give the visually desired result.

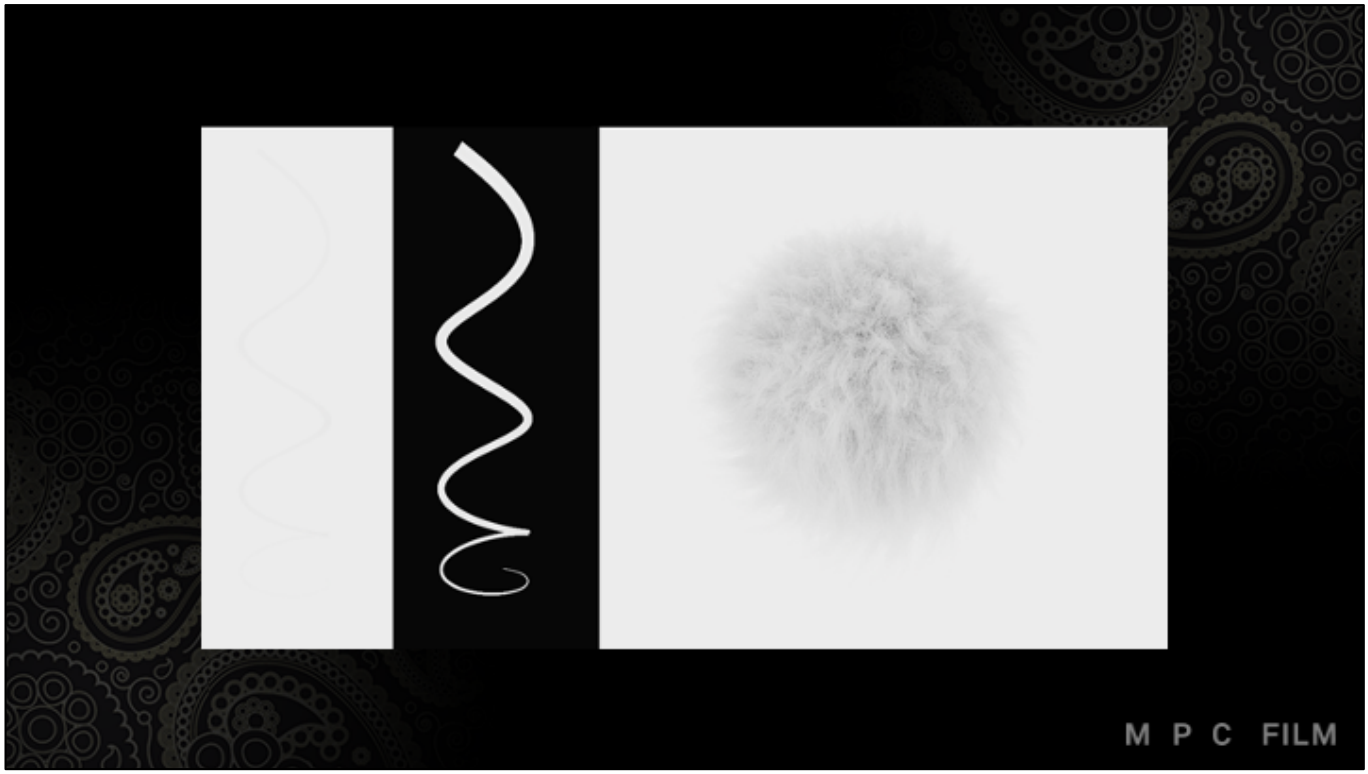
We went through a similar process with our fur shader. And it’s complicated ... there are a number of factors that affect the colour of the fur volume. Fur density can drive rich saturated colours or pale muted colours.

The number of light bounces affects the final colour too. In this image we have 0, 2, 4, 6, 8, 10, 12 bounces from left to right. Not only does the fur get brighter in the right images, it also gets more reddish-orange.

In the end, we rendered the bulk of the show including *The Lion King* with 8 to 10 bounces of light, and calibrated our fur shader accordingly.

We were really concerned about performance and energy conservation.

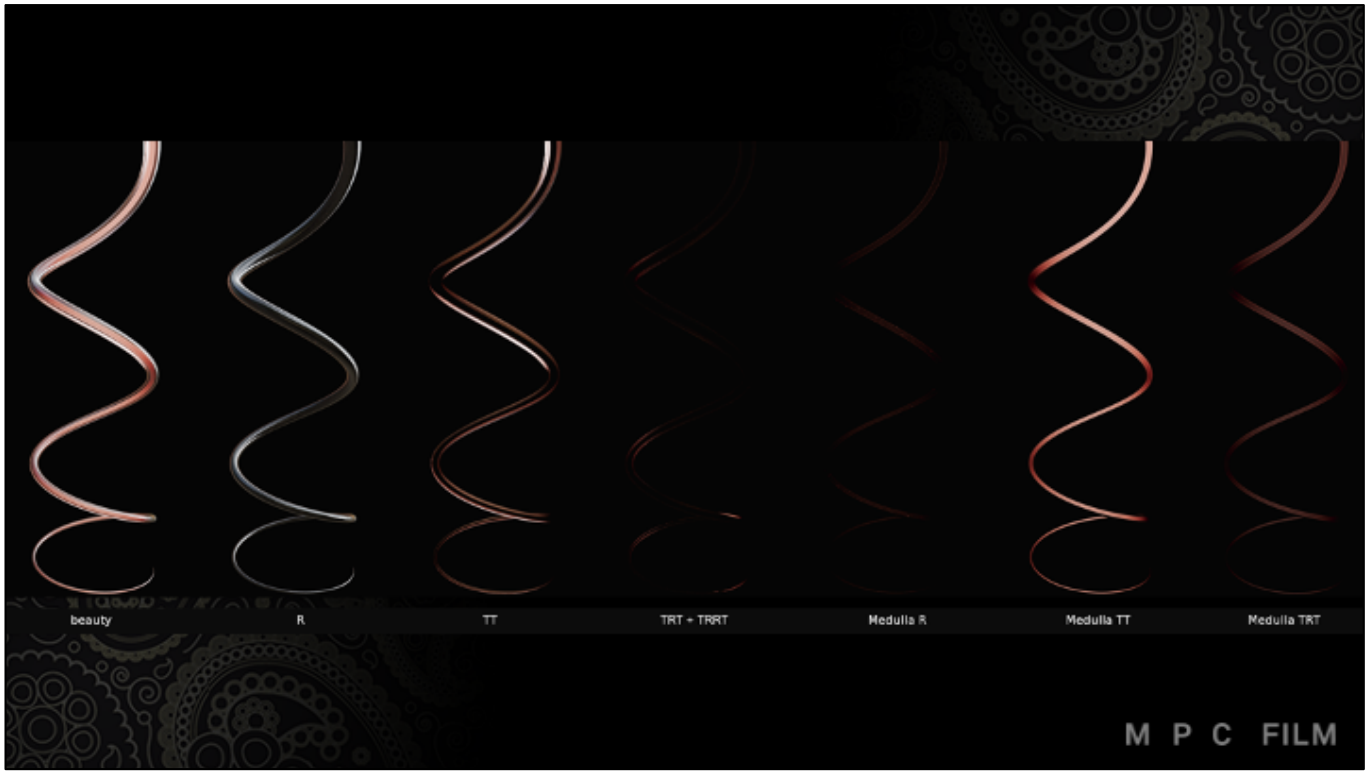
We utilized a multiple importance sampling technique based on each lobe’s contribution, and the performance in general was 15%-30% better than a similar look with PxrMarschner, depending on shading parametrization and lighting environment.



A huge amount of attention was paid to energy conservation during the development of our shader. This was one of, if not the most important improvements we made, ensuring that the sum of all the lobes was normalised and well balanced.

On the left there is a curve which matches a white background almost perfectly. Just to the right of it is the same curve but isolated against a black background. And on the right is a furry sphere ball where tips of the curves perfectly matches to the background and some occlusion happens towards the roots. This is the typical furnace test and shows we're not getting any inappropriate glowing or darkening.

As a result we've got an ability to render white fur without any difficulties. Our lookdev team were really excited about this for another show we were working on at the same time. Achieving this effect was quite hard before using Marschner model where the color was set by diffuse lobe.



Overall we have 7 fiber lobes R, TT, TRT, TRRT, Rs , TTs, TRTs which allow us to render Fur physically correct and in energy conserving manner. We output them to AOVs using LPEs.

Besides that diffuse lobe was introduced a top of this to simulate dust and dirt

Extra specular lobe to simulate wetness

And extra input for iridescence to render feathers

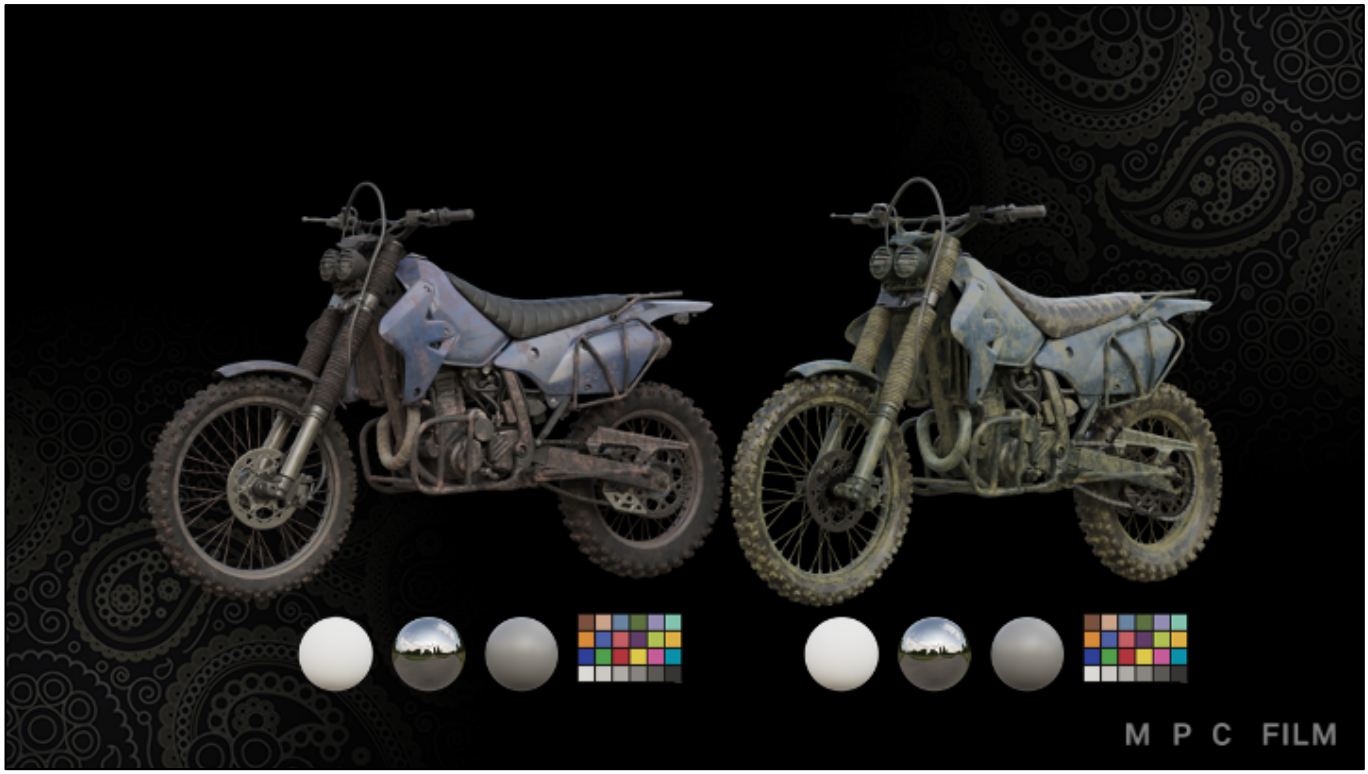
Here’s the full evolution of our fur shaders over the last 5 years. On the left is the flat ribbon-like curve shader that we used on *The Jungle Book;* in the middle is our Fur v1 shader that we used on *The Lion King*, with the nice cylindrical look; finally, on the right is our Fur v2 shader that has recently been rolled out into production, complete with the medulla scattering core.

And I'll close by showing the same comparison in a slightly more interesting context than a single fur strand. Again, the left-most image is using flat ribbon curves, the second image uses Fur v1, the third image uses Fur v2, and the rightmost image is Fur v2 with textured albedo.

And that takes us to where we are today.

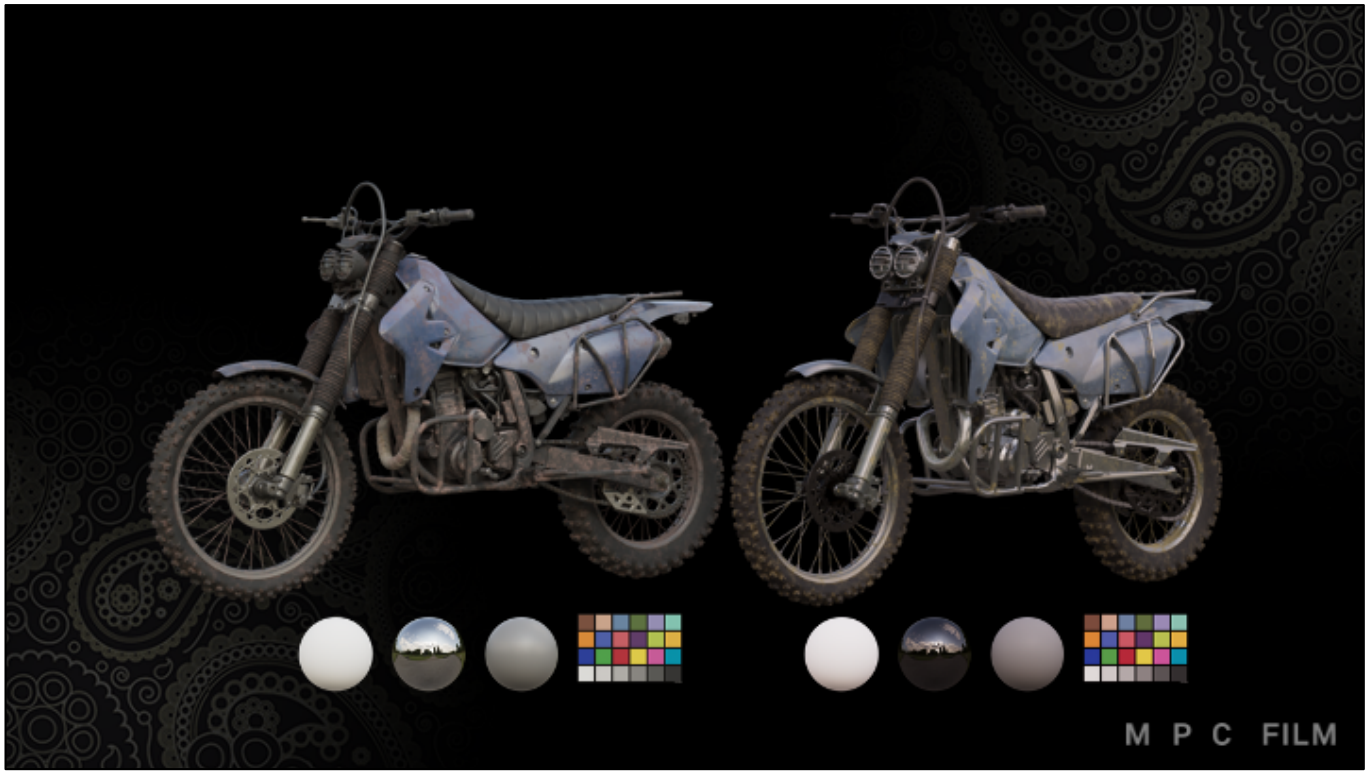
Allowing look-dev artists to work with ad-hoc shading networks gave us the ability to do really interesting things with lookdevelopment.

It also meant we were often re-inventing the wheel on every show, even for commonly used materials such as skin, cloth and hair.



It also led to inconsistencies when having to incorporate that work into final lighting.

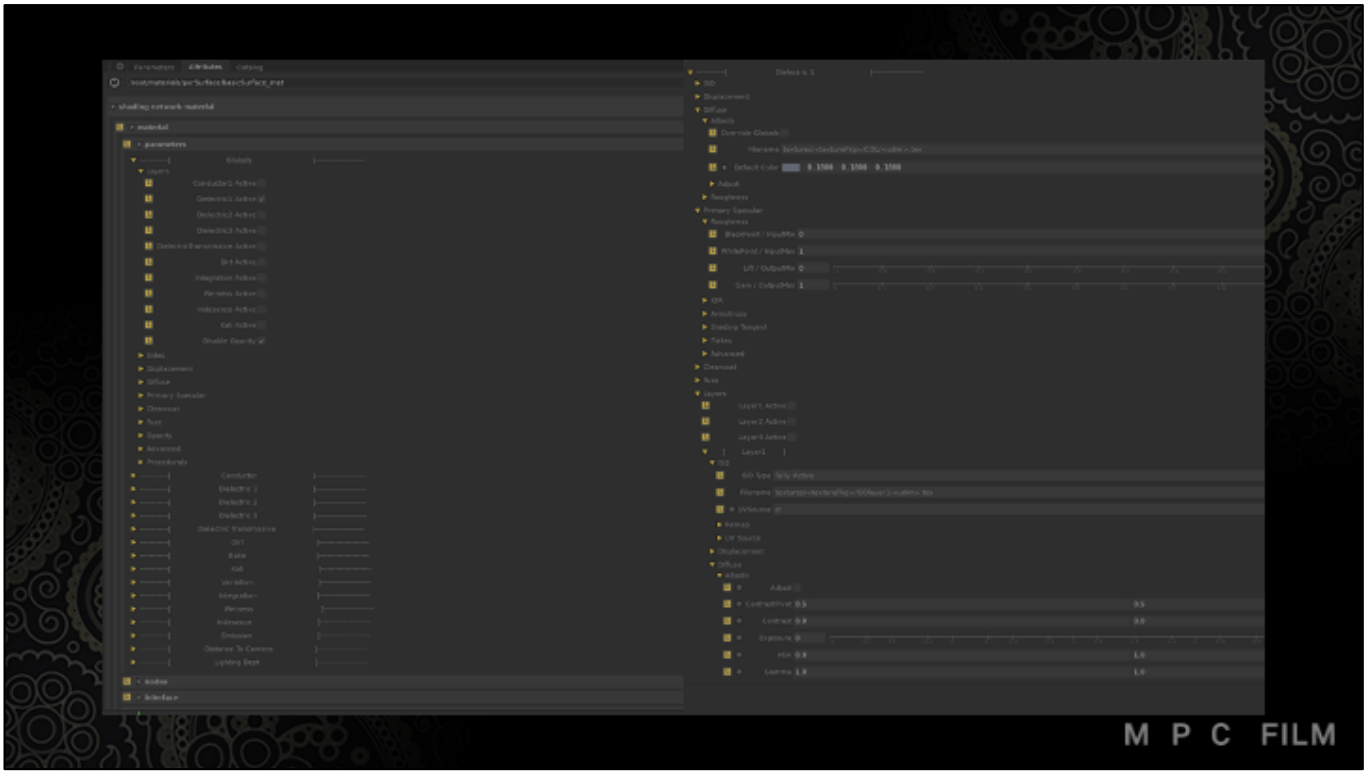
This is illustrated above where the motorcycle on the right is suppose to look like the one of the left.



Using the tools available to them, lighters would often make shading and lighting adjustments within their scene to compensate.

Here I've used light-linking along with various adjustments to the hue, saturation and even the specular contribution of the light in order to get a closer match on the two motorcycles. You can see the impact of these lighting adjustments in the reference spheres.

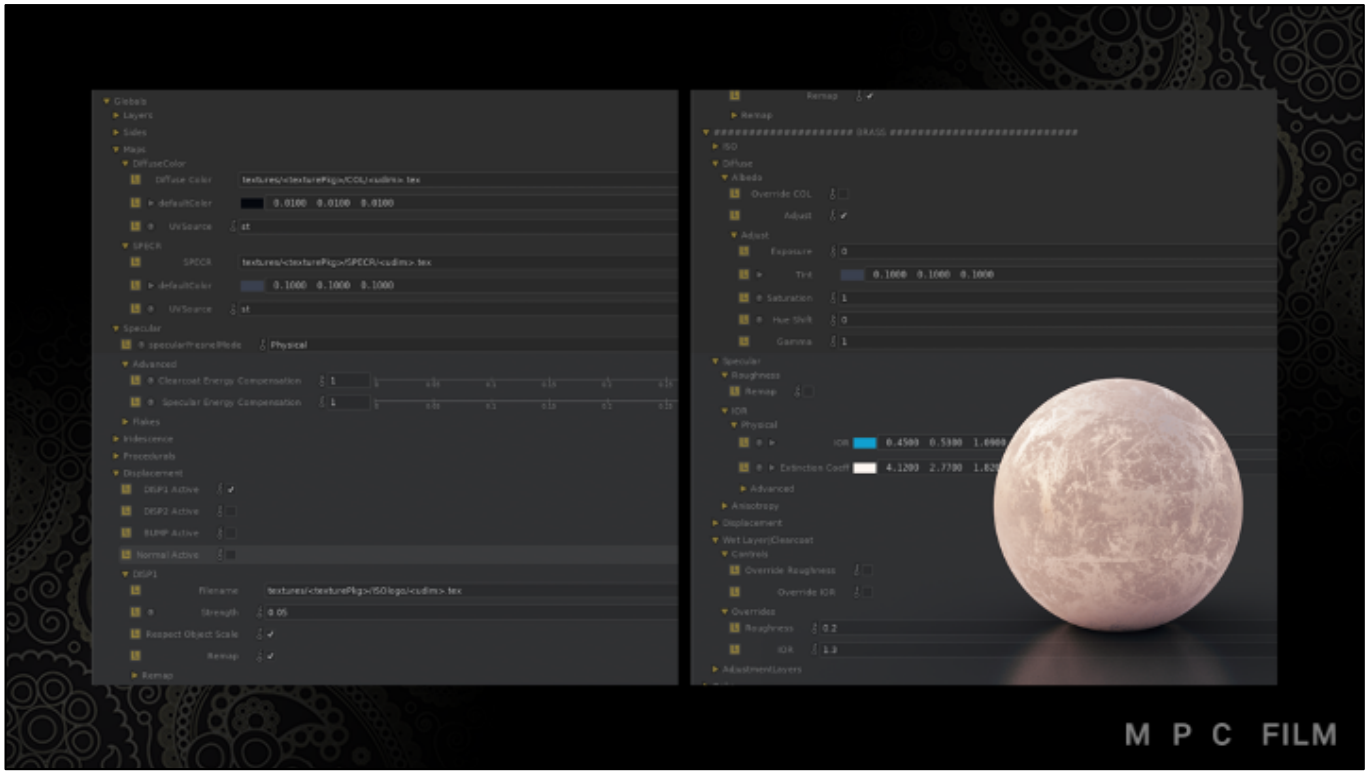
Over the last few years we've built up an all encompassing uber-shader we've referred to internally as the "Asset Shader". Originally inspired by Disney's principled shading work, we set out to create a general purpose material which could handle nine out of ten of all our look-dev needs.



While the foundation for this uber-shader is a shading network, the user experience is presented within Katana as a hierarchy of parameters. Utilizing all of the available lobes within PxrSurface and offering several layers of overrides from which to drive the appearance of multi-layered materials.

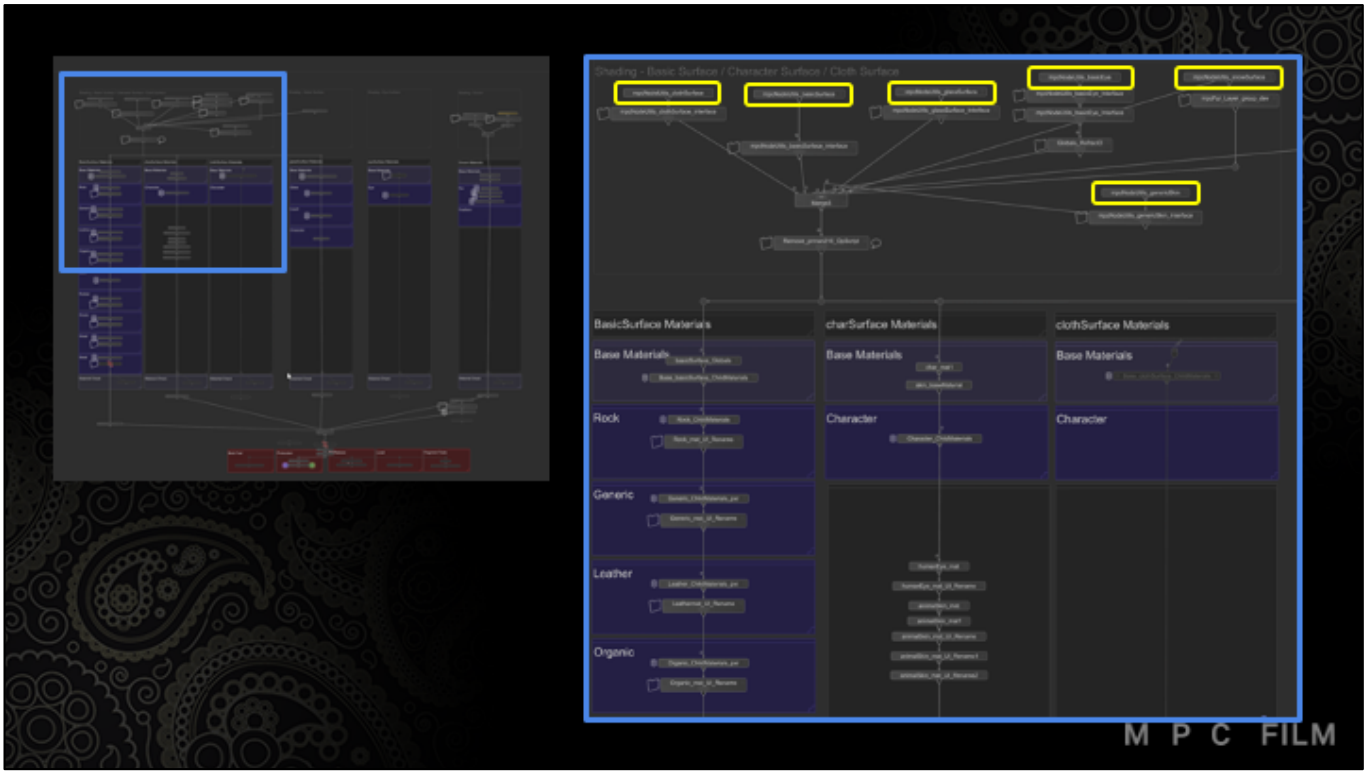
We additionally incorporated material presets, building up a series of parent and child materials from which we could automatically apply look-dev and get production-ready results quickly.

This automatic system relied on two inputs being well defined prior to the look-dev stage, first the tagging of objects with a material type and the second being appropriately painted texture maps for that material. For example in order to reveal a metal layer under a painted layer, the material would look for a predefined texture map used to mask the painted layer.

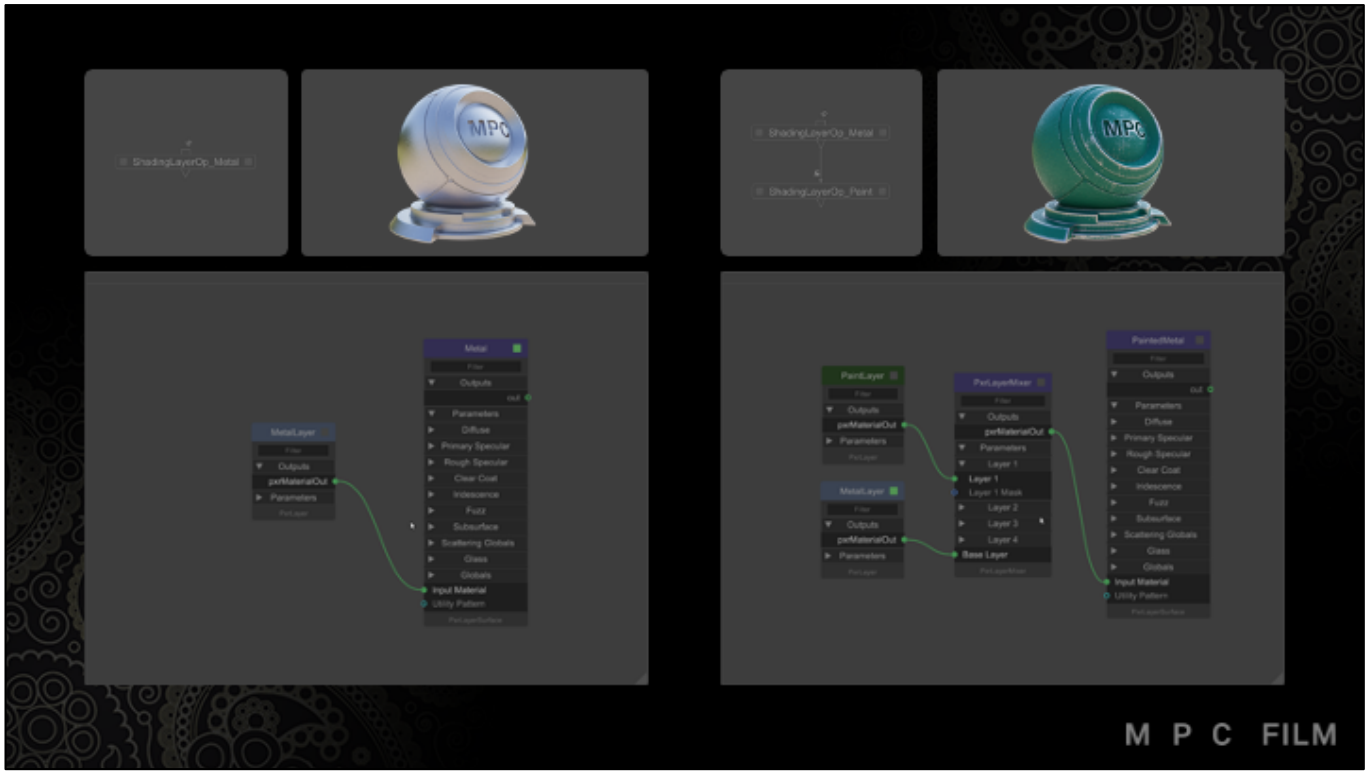


The use of the Asset Shader and its presets helped unify a lot of our look-dev in lighting scenes, but its complexity and associated compute cost became problematic - while the user-interface could be simplified with conditional widgets that could hide shading parameters that were not being used, we still ended up a large number of parameters for even relatively simple single-layered materials.

Further concern was that large portions of shading network itself were being always being evaluated even in these simple materials. Attempts to optimize the logic of the shading network didn't result in significant improvements and in production cases where the compute cost became problematic, we optimized materials with scripted operations that disconnected sections of the shading network or replaced the troublesome shaders with heavily simplified versions of those materials.



One of the main problems with version one of the Asset Shader was its top-down approach to building the shader. The following screenshot showing the node graph used to create the material presets, the nodes highlighted in yellow contain the shading networks that make up the foundation of the Asset Shader.

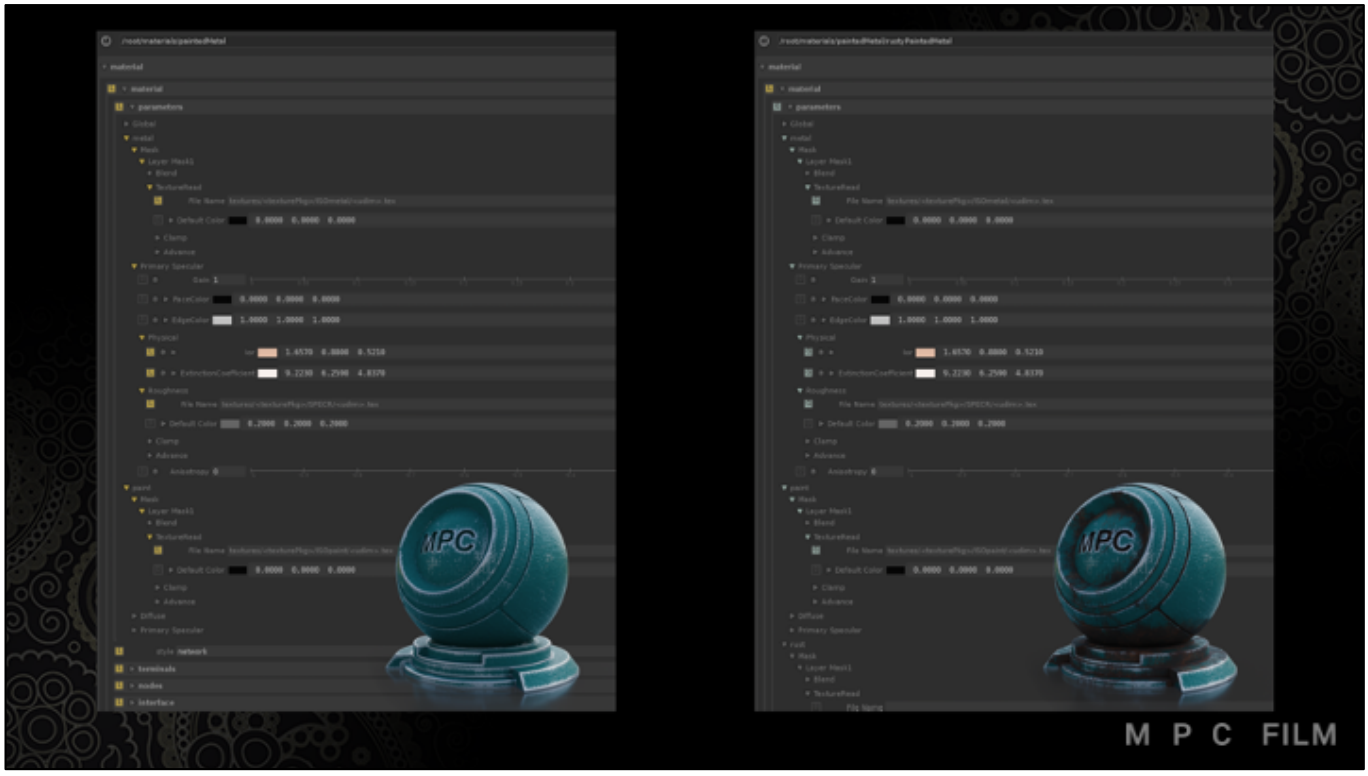


Version two was designed to address this, to build the shader from the ground-up in a more dynamic and procedural way while still maintaining a consistent structure and user interface when it came to layered materials.

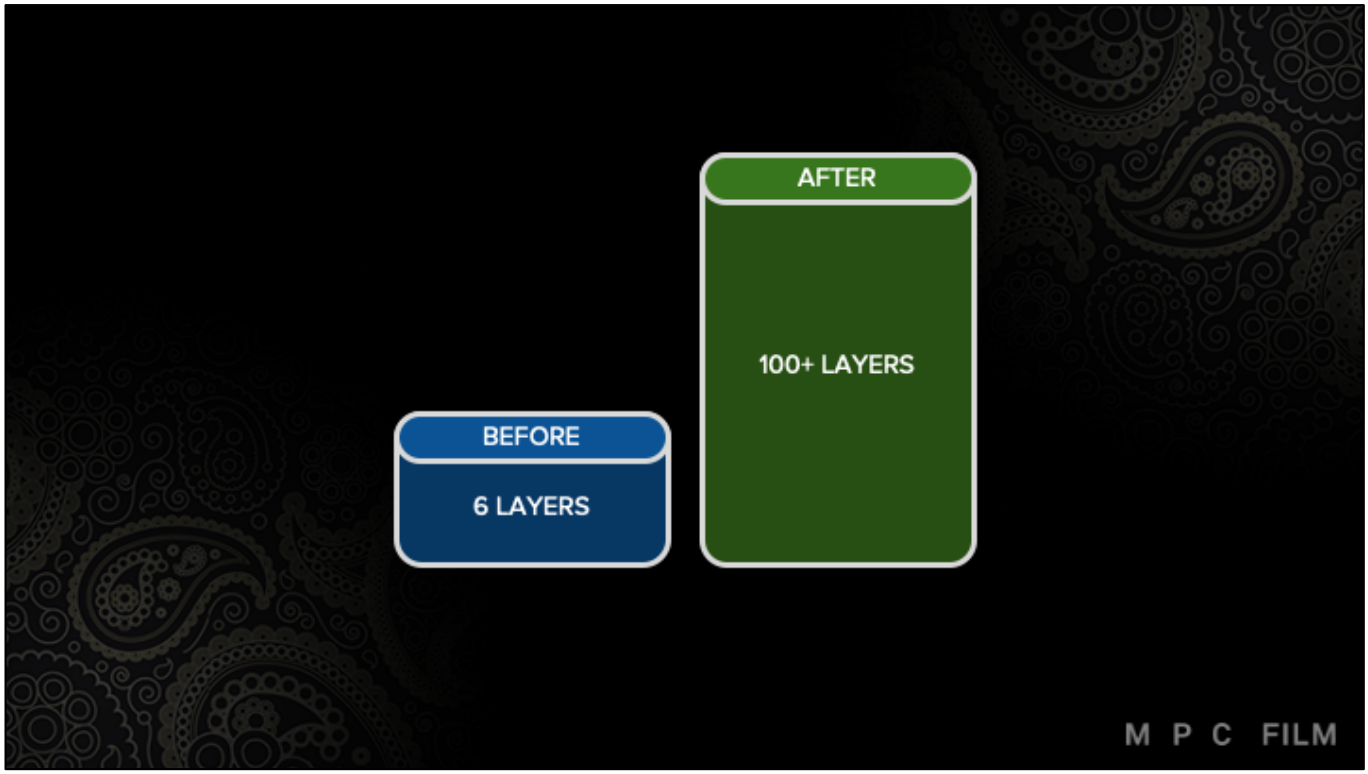
Within Katana we developed a supertool (seen along the top) that would allow a look-dev artist to dynamically build materials, creating both the shading network and the parameter interface for the artist. The user never directly interacting with the shading network itself.

This now allows look-dev artists to create individual shading layers as presets that can be combined together with masks, here the layering operation is purely horizontal and utilizes a fixed vertical stack of lobes.

Shaders can still inherit from parent shaders - allowing users to define child shaders which can add or subtract layers as needed, making it easy to add layered variants to existing materials.



The resulting user interface also becoming a lot more straightforward and easy to manage - even when increasing the number of layers.



This new system does allows artists to create larger number of layers than we previously allowed in the Asset Shader.

In practice we're hopeful that lookdev artists will want to work with fewer layers and that the overall number of shading nodes that make up an asset will go down.

As we start rolling version two into productions, the render times comparisons we've done between version one and version two have reduced by around 20%.

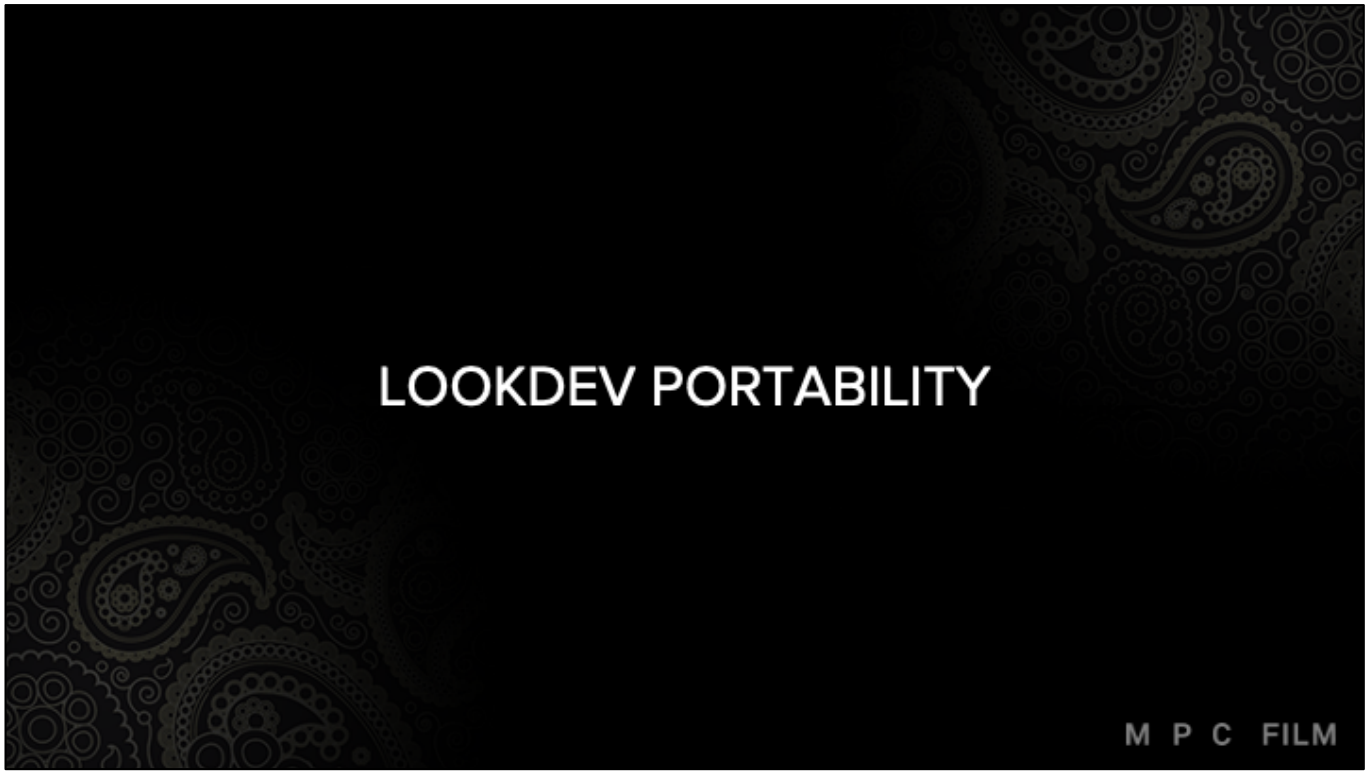
In the best case examples we've seen render times drop from around six hours to two hours - a two-thirds reduction in rendering cost.

Looking ahead, we continue to push for more physically accurate (or at least physically motivated) effects in our shading and rendering.

We continue to have internal discussions about layered materials, especially where it concerns energy conservation and balancing light contribution among substrate layers.

And although we've moved away from additional BXDFs such as cloth, glass and particle, cloth is one area we are likely to revisit soon, especially as clothing becomes more realistically modelled as fibres rather than as textured subdivision surfaces.

Lastly, with increased confidence in our materials, we're taking an increased interest in how we describe the lights that illuminate our world, and the cameras and sensors that consume this illumination.



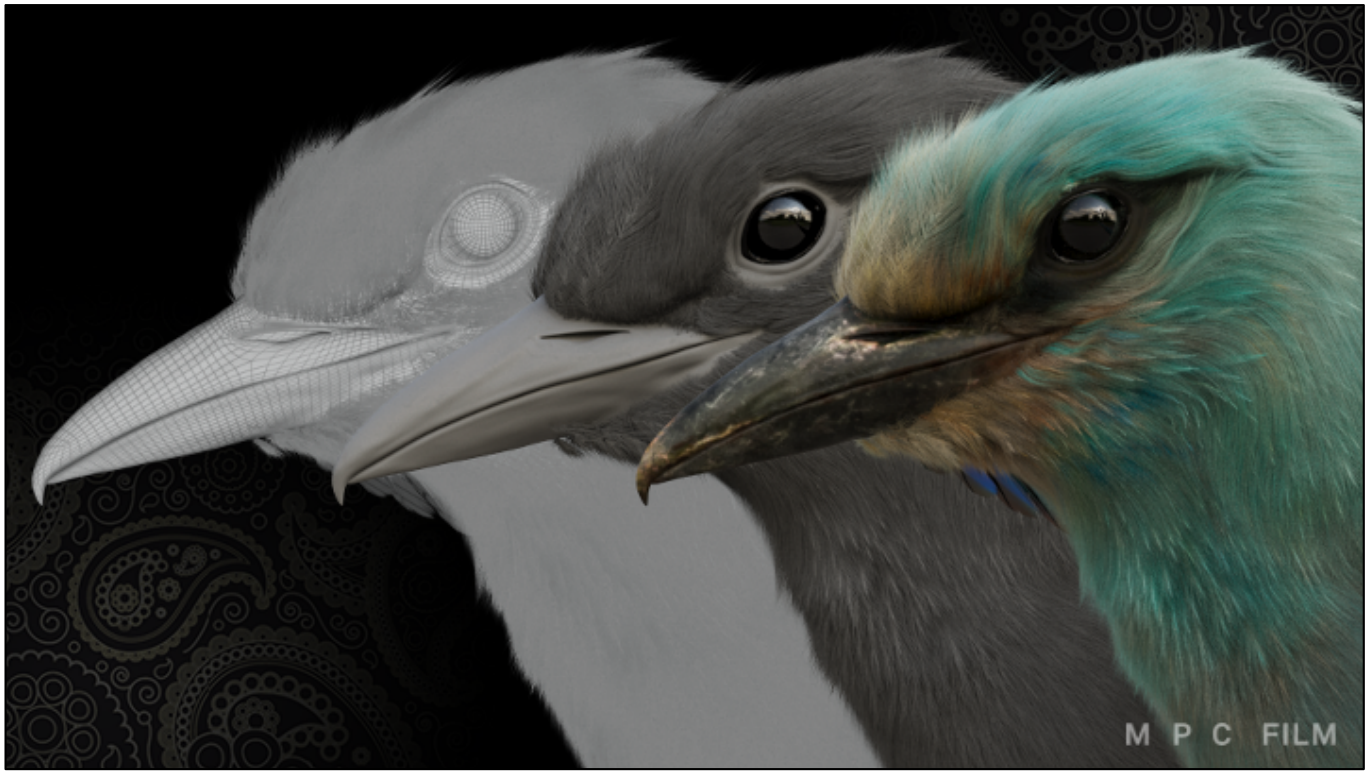
Over the last few years we've worked on a few films which have required the exchange of assets between visual effects studios. While model and texture data is often straightforward and easy to deal with - the exchange of look-dev data is often little more than visual references such as renders and movies from which to try match the look. This can often be complex and time-consuming process to translate artistically.

This translation is frustrating as the language of look-dev among studios and renderers is more akin to different regional dialects than it is a number of entirely different languages - the common building blocks for processing signals in lookdev have long been well established in shading languages such as OSL, where terms such as gamma, exposure and remap are the same wherever you go.

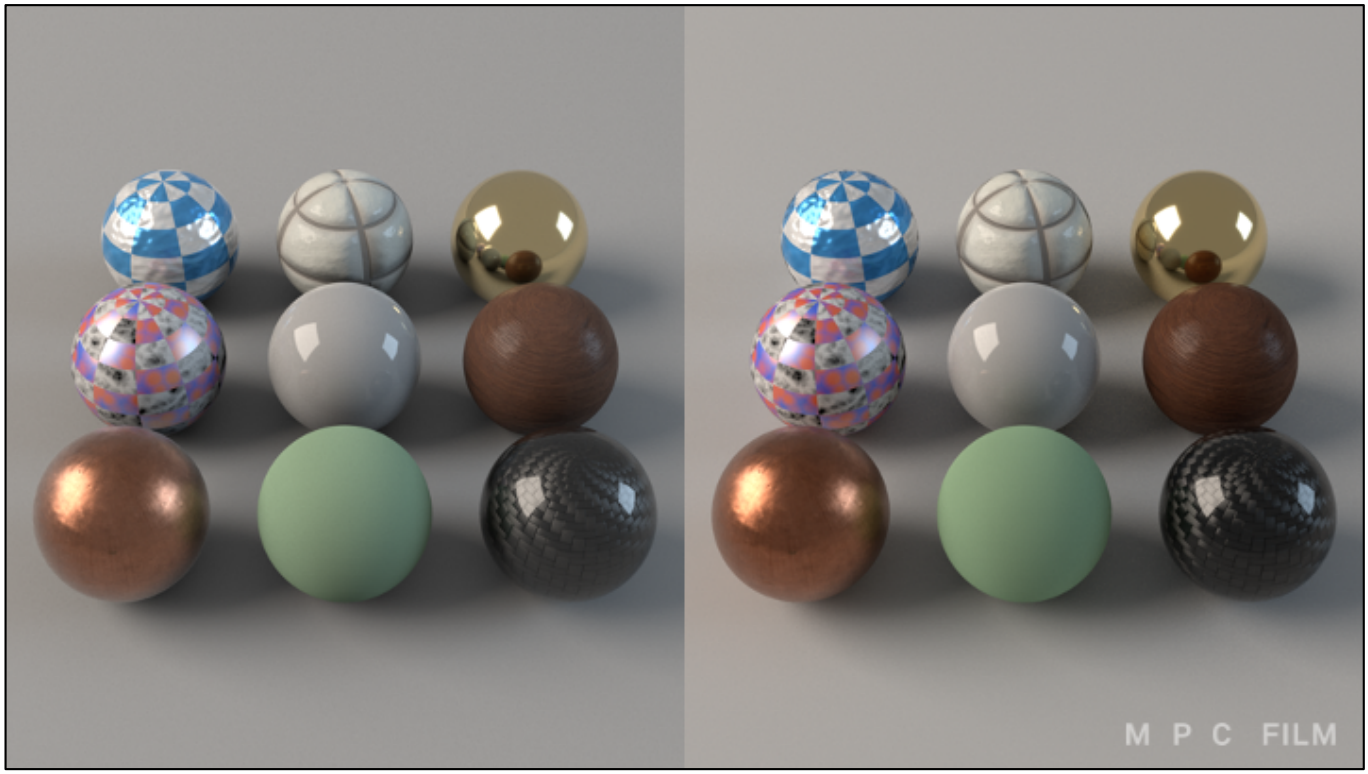
Even when discussing BXDFs where the differences in implementation are stronger, the language of diffuse, specular and subsurface is pretty common.

The example shown is a translation of an Arnold scene (left) into a RenderMan scene (right). The end result is remarkably similar and gives us increased confidence that sharing of look-dev will soon become as straightforward as sharing geometry and textures.

Realistic head scan courtesy of Infinite Realities via Creative Commons.
https://ir-ltd.net/portfolio/infinite/



This portability doesn’t just influence external exchange, often internally within the studio we have cases where doing the look-dev early and being able to use it under multiple contexts has huge benefits. One of the major goals in our virtual production work is to get lighting and materials representation closer in approximation to the final frame - allowing filmmakers to make more up front decisions about the work when filming.



We are forming an opinion regarding technologies such as MaterialX and MDL. The above example illustrating MDL materials available with V-Ray (left) and our work on an MDL implementation in RenderMan (right).

In the last year, we've heavily discussed not just how physically based shading plays a part in our work - but also how lights, lenses and cameras play a part in the verisimilitude of our physically-based rendering.

Outside a handful of rare situations, we've traditionally avoided rendering with lens effects such as distortion or depth of field, instead utilizing compositing techniques to artistically reproduce the look and feel of a real lens. In the rare cases we have rendered lens effects we've used standard thin-lens models to represent those effects.

Recognizing that a real-lens can have both a strong and subtle impact on the look and feel of a photographed image and that a lot of very specialized engineering and design goes into a real lens - we were keen to explore what a more accurate lens model would give us over the thin-lens model and so earlier this year we implemented "Sparse high-degree polynomials for wide-angle lenses" (Emanuel Schrade, Johannes Hanika and Carsten Dachsbacher, 2016) as a projection plugin in RenderMan.

The results we got were certainly impressive and much of the lens effects produced would be difficult to achieve with current compositing tools. The biggest concerns here are both practical and artistic - needing additional render time in order to adequately sample the effect and also being very difficult to artistically control. In the image presented we see a render through a toy lens, while this nicely portrays the characteristics of that particular lens, it is accurate to a fault in that no part of the image is sharply in focus. For realistic integration of cg elements into a live-action plate this may be desirable use of a real lens - but less desirable if the creative intention is to capture the feeling of the lens rather than all it's physical characteristics.

{ VIDEO DIALOGUE }

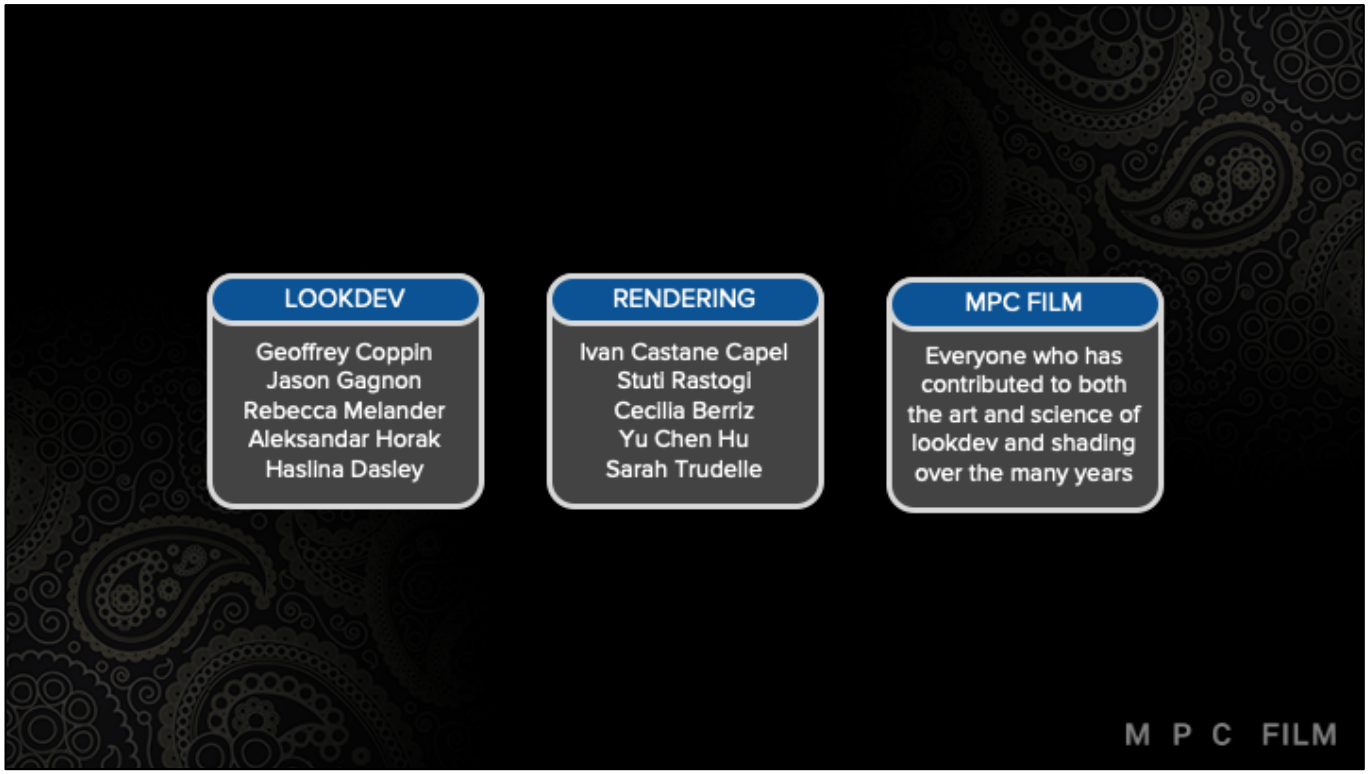
Outside a handful of rare situations, we've traditionally avoided rendering with lens effects such as distortion or depth of field, instead utilizing compositing techniques to artistically reproduce the look and feel of a real lens.

We were keen to explore what a more accurate lens model would give us over the thin-lens model and so earlier this year we implemented "Sparse high-degree polynomials for wide-angle lenses" (Emanuel Schrade, Johannes Hanika and Carsten Dachsbacher, 2016) as a projection plugin in RenderMan.

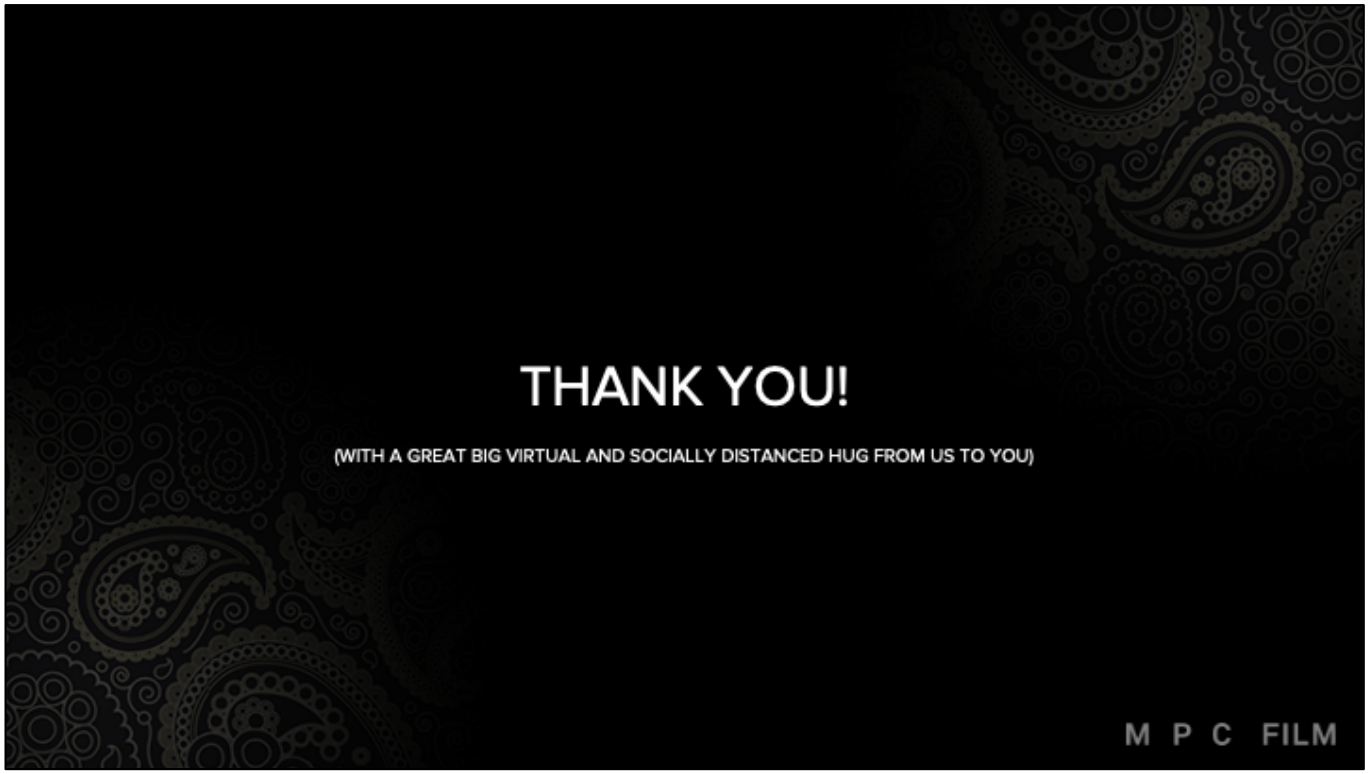
In the image presented we see a render through a toy lens, while this nicely portrays the characteristics of that particular lens, for realistic integration of cg elements into a live-action plate this may be desirable use of a real lens - but less desirable if the creative intention is to capture the feeling of the lens rather than all it's physical characteristics.

The switch to physically based shading has meant over the years our lighting has become more spatially accurate to the geometry of live-action sets and locations. However all of our illumination values are still relative to the live-action plates we use rather than absolute photometric units.

Spectral rendering is also an area where we can see obvious benefits and improvements to our shading and rendering, particularly in the case of digital humans where even subtle improvements in shading can dramatically increase the realism and our emotional connection to the resulting image. While we've yet to cross that threshold the arguments for pursuing more physically based illumination and spectral precision in production rendering are gradually becoming stronger.



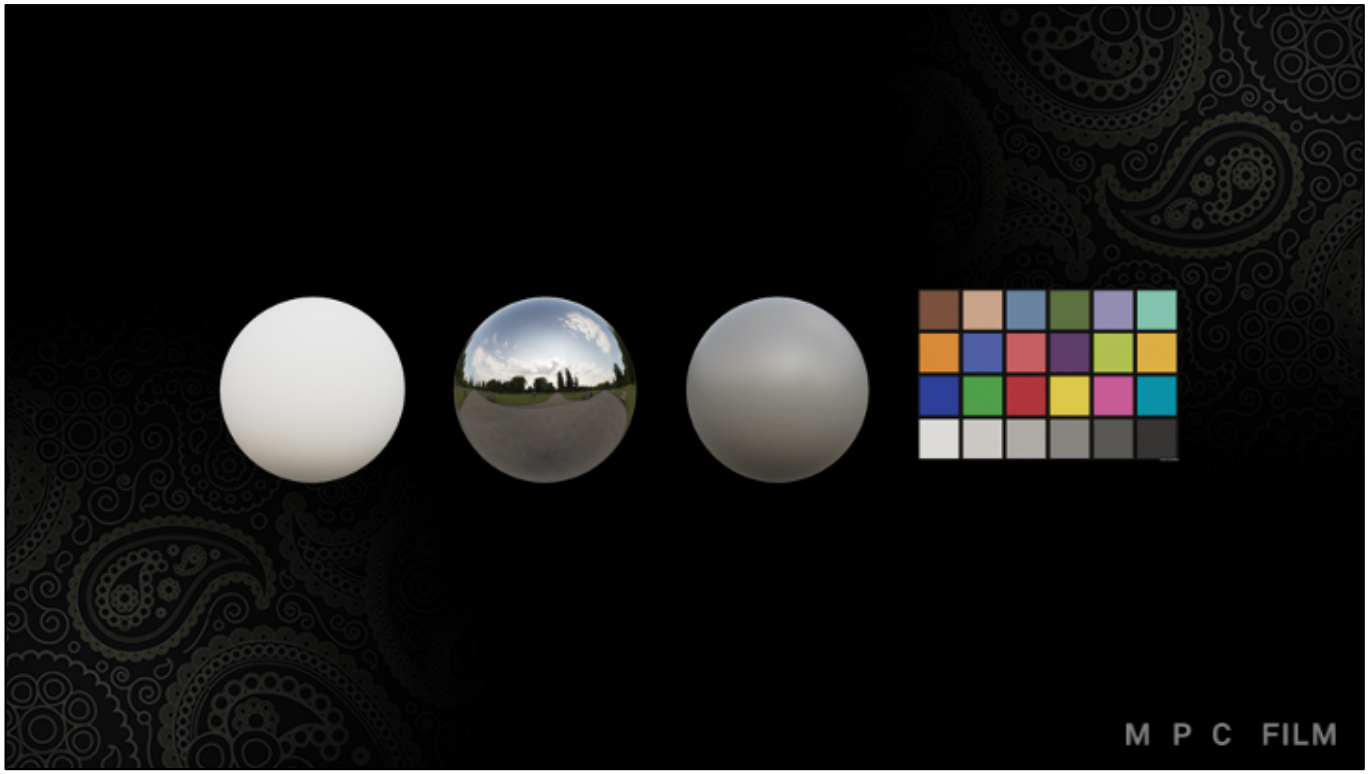
This work is always a team effort - special thanks to the following people for their contributions to the world of physically based shading at MPC Film in the last year and all of those who have contributed to lookdev and shading at MPC Film in years prior.



And thanks to *you* for your attention. It’s always wonderful for us to have the opportunity to share a bit of the history and magic that goes into MPC’s movie making process, and I hope you enjoyed the story.

(From us all - good morning, afternoon, evening or night)

Keep safe!



THE DELETED SLIDES

During the process of producing the video presentation for this talk we made some edits in order to keep within 30 minutes.

Throughout the speaker notes you'll see mention of VIDEO DIALOGUE, this contains both the narration used in the final video along with the original intended narration.

This edit also resulted in some slides being cut completely from the final video edit. We have included these slides here with context to where they originally appeared.

Because the width as well as the density of the groom affects the final result a lot, we have absorption scaling for both cortex and medulla as well as a global width scaling which affects overall groom.

{ This slide originally appeared after slide #51 }

We support layered materials with Fur2 through the use of OSL which allows us simulate undercoat layers - we are just starting to roll this out now.

When it comes to furry characters, there are aspects of the texturing workflow that we feel could be improved. While we currently transform fur colours into melanin concentrations within Fur2, we'd like to directly paint concentration values inside Mari - rather than colours picked from a swatch - so artists can get a better idea of the shaded result while painting.

With recent improvements in interactive rendering, we'd like to also look at bringing Fur2 directly to our texture artists in Mari, offering them the ability to see the shaded results while painting.

We'd like to fully sample the medulla scattering as a volumetric ground truth and compare against our own approximated model

and any future research in fur shading.

{ This slide originally appeared after slide #79 }

With physically based rendering, we still occasionally get requests to bend the laws of physics in order to make a shot work - be it changing what object is reflected in a window or making the shadows go in the wrong direction.

We've also seen an industry trend with productions now utilizing non-photoreal rendering technique in order to create unique graphical styles of animation and is an area we've taken a mixed-media approach towards investigation and how we would render different art mediums such as pencil line work, clay modelling and watercolours.

{ This slide originally appeared after slide #83 }

The use of real lenses also brings up questions about what level of control we offer: is it simply the physical controls over aperture, focal distance and in the case of a zoom lens control over focal length?

Or do we open it up and allow end-users to modify existing lenses or even create their own?

Even without a background in optical engineering, given the right tools and the ability to make interactive rendering iterations it may be possible for a user to create interesting lens effects on their own that may serve a creative visual effect that isn't based in reality.

{ This slide originally appeared after slide #85 }