SONY PICTURES

imageworks

25TH ANNIVERSARY

30 JULY – 3 AUGUST Los Angeles

SIGGRAPH2017

Revisiting Physically Based Shading at Imageworks

Christopher Kulla & Alejandro Conty

SIGGRAPH 2017

Thank you for the introduction. My name is Chris Kulla and I will be presenting, with Alex, our latest take on physically based shading at Imageworks.

Outline

- Motivation
- Energy-Preserving BSDFs
- Subsurface Scattering
- Coatings
- Putting it all Together
- Future Directions

So here is a quick overview of what we will be discussing today. We've tried to pack as much information as possible into this talk, so I'll jump right in.

Motivation

First let me go over our basic motivations for this work.

We have been using physically based shading models for quite a while now, we participated in this course back in 2010, 2012 and 2013. So what has changed since then?

Motivation

- Previous shader libraries presented a long list of lobes to users
- Very flexible but parameter space too large
- Hard to dial simple cases correctly
- Too easy to dial non-physical values (e.g. wood with IOR=100!)
- Too easy to create energy, particularly at grazing angles

Our previous shader libraries were built around the idea of individual lobes. This gave artists lots of flexibility, but also made it hard to hit some simple cases since many knobs needed to be turned “just right”.

When looking at production data, we occasionally found cases where the artist had achieved a nice look using physically nonsensical values. For example we saw a case where wood had an IOR over 100. The end result looked fine because of how the lobes where scaled, but it suggested to us the parameter space had grown too large.

What was more concerning, however, was that adding multiple independent lobes was leading to excess energy being created, particularly at grazing angles.

Motivation

- Inspired by Disney BRDF/BSDF
- Encourage a more physical description of materials
- Enforce energy conservation, even in layered cases
- Energy preservation
- Enforce reciprocity
- Unify SSS and volume rendering

Our biggest inspiration came from the Disney BRDF models. They encourage a more physical description of materials by cleanly separating dielectrics and conductors. They also distilled controls to a small set of intuitive parameters.

We also wanted to ensure that our materials would always be energy conserving, even in layered cases, but also energy *preserving* by which I mean that materials that scatter all energy should have an albedo close to 1.

We had a few BRDFs that were non-reciprocal and wanted to find alternatives to them where possible.

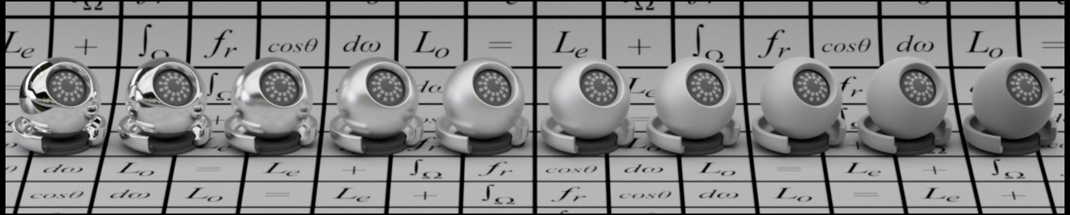
Finally, like some of the other presenters, we felt it was time to try and unify subsurface scattering and volume rendering.

Energy-Preserving BSDFs

I’ll begin by discussing energy *preservation* for the basic BSDFs that make up our shading model.

Microfacet Specular

- GGX + height-correlated masking & shadowing
- Single-scattering assumption introduces darkening at high roughness
- Close to 50% energy missing for $\mathrm{rough} = 1$



As you might expect, we picked the GGX distribution. This is slowly becoming an industry standard, so we definitely wanted to follow that trend.

The biggest problem with current microfacet models is that while they *conserve* energy (in other words: they don't create any) they don't *preserve* energy at high roughness. This is due to the single-scattering assumption they make.

Here I'm showing an idealized metal where the microfacets have no Fresnel. It loses close to half the incoming energy at roughness 1.

Microfacet Energy Compensation - Previous Work

- Kelemen et al., 2001: Generic solution via precomputed tables
  - Did not address textured materials
  - No treatment of transmission

We're not the first to notice this problem of course.

Kelemen in 2001 published a very generic solution that relied on precomputed tables. His paper only discusses plastics, even though the technique is more broadly applicable. However, the paper didn't directly address varying BRDF parameters or transmission.

Microfacet Energy Compensation - Previous Work

- Kelemen et al., 2001: Generic solution via precomputed tables
  - Did not address textured materials
  - No treatment of transmission
- Jakob et al., 2014: Similar solution in a more general framework
  - Framework is complex
  - Too heavy for textured materials

Wenzel Jakob and colleagues developed a very comprehensive solution that solves many cases and includes layering, however they do so within a very complex framework that is still too heavy for textured materials.

Microfacet Energy Compensation - Previous Work

- Kelemen et al., 2001: Generic solution via precomputed tables
  - Did not address textured materials
  - No treatment of transmission
- Jakob et al., 2014: Similar solution in a more general framework
  - Framework is complex
  - Too heavy for textured materials
- Heitz et al., 2016: Found “ground truth” but only as a stochastic model
  - Requires many random numbers to sample *and* evaluate
  - Poor fit for our rendering architecture

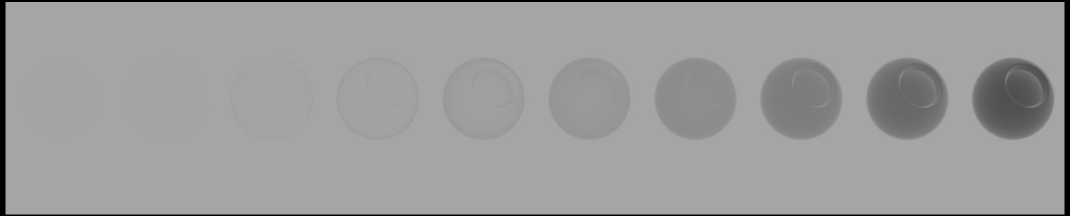
Last year, Eric Heitz presented a ground-truth result for multiple scattering between microfacets. It requires no precomputation at all, but the model is stochastic, meaning that random numbers are required to both sample *and* evaluate the BRDF. This was a poor fit for our rendering architecture.

Microfacet Energy Compensation

Energy reflected for a particular viewing direction:

$$E(\mu_o) = \int_0^{2\pi} \int_0^1 f(\mu_o, \mu_i, \phi) \mu_i d\mu_i d\phi$$

Expect $E(\mu_o) = 1$ if microfacets are purely reflective



GGX Furnace Test

We can measure energy loss by integrating the cosine-weighted BRDF.

Notice that I'm using the Greek letter μ for the $\cos\theta$ term here. This is just to avoid too much trigonometry in the slides and its actually closer to how the formulas are actually implemented.

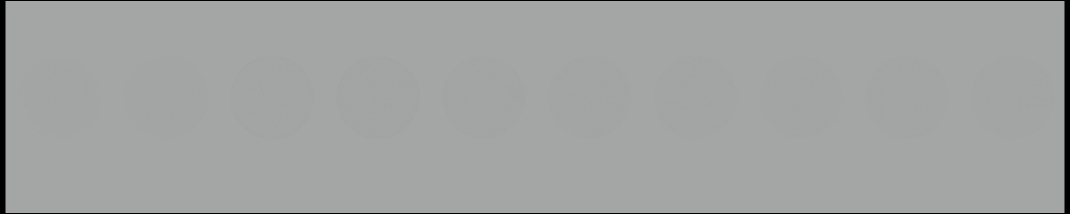
This energy E is sometimes referred to as the directional albedo. It varies from 0 to 1 for a given viewing direction.

It's also sometimes called the "furnace test" because it's equivalent to a lighting integral against a constant background.

Microfacet Energy Compensation - Technique

We can compensate for missing energy using:

$$f_{\mathrm{ms}}(\mu_o, \mu_i) = \frac{(1 - E(\mu_o))(1 - E(\mu_i))}{\pi(1 - E_{\mathrm{avg}})}, E_{\mathrm{avg}} = 2\int_0^1 E(\mu)\,\mu\,d\mu$$


GGX + Energy Compensation Furnace Test

The technique presented in the Kelemen paper can be distilled down to the following formula.

We can create a new BRDF lobe out of the function *E* we just saw. When we add this lobe to our original BRDF, we'll always get a perfectly energy conserving result.

The denominator here is a normalization term that involves the cosine-weighted average of *E*. You can think of this as the average amount of energy loss for any possible viewing direction.

Also notice that this BRDF is reciprocal. So exchanging the view and light direction cosines doesn't change the result.

Microfacet Energy Compensation - Technique

Why does this work? Let's compute the directional albedo of f_{ms}:

$$\begin{aligned}
E_{\mathrm{ms}}(\mu_o) &= \int_0^{2\pi}\int_0^1 f_{\mathrm{ms}}(\mu_o,\mu_i,\phi)\mu_i d\mu_i d\phi \\
&= 2\pi \int_0^1 \frac{(1-E(\mu_o))(1-E(\mu_i))}{\pi(1-E_{\mathrm{avg}})}\mu_i d\mu_i \\
&= 2\frac{1-E(\mu_o)}{1-E_{\mathrm{avg}}}\int_0^1 (1-E(\mu_i))\ \mu_i\, d\mu_i \\
&= \frac{1-E(\mu_o)}{1-E_{\mathrm{avg}}}(1-E_{\mathrm{avg}}) \\
&= 1-E(\mu_o)
\end{aligned}$$

The directional albedo of f_{ms} exactly complements f!

It's worth spending a few minutes going through the math to understand why this works. This formula is stated without proof in the Kelemen paper.

In fact the derivation is very straightforward. We are going to compute the directional albedo of the new BRDF we just introduced. Lots of terms can be pulled out of the integral right away.

The integral that is leftover is in fact just the definition of E_{avg}, which cancels out the denominator. We're left with just $1-E$ evaluated with the cosine of the viewing direction.

In other words, the directional albedo of this new BRDF exactly accounts for the energy missing from the original BRDF. Also notice that I haven't made any particular assumption about the BRDF. This method always works.

Microfacet Energy Compensation - Technique

Computing *E* requires integration. We want to be able to spatially vary:

- roughness
- anisotropy
- IOR

At first glance, tabulating all possibilities seems prohibitive.

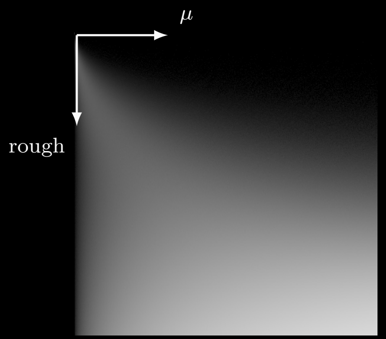
Now the only catch as you have probably realized is that the function *E* we have been using is expressed as an integral. And unfortunately for the BRDFs we care about this integral doesn't have a closed-form solution.

Kelemen's paper simply tabulated this function assuming all parameters stay constant. However, we want to be able to spatially vary roughness, anisotropy and IOR. In the case of metals, IOR is itself defined by several parameters.

So tabulating all possibilities doesn't seem practical.

Microfacet Energy Compensation - Roughness

Precompute $1 - E(\mu)$ for $\mu \in [0,1]$ and $\mathrm{rough} \in [0,1]$:



How much resolution is required?

Let's try anyway. We can start with roughness since its the most important.

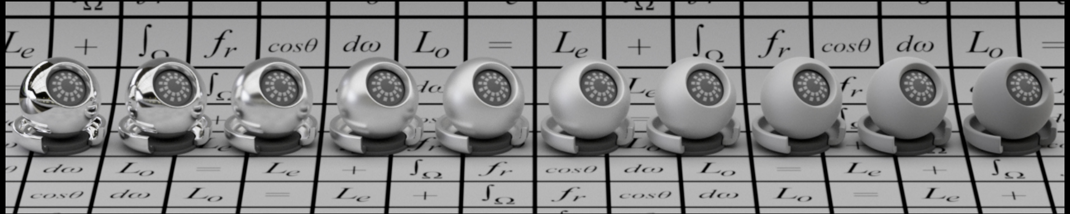
Precomputing $1 - E$ for all possible viewing cosines and roughness values we get the following result. It looks very smooth, particularly at high roughness values where it's most important.

So how much resolution do we need for the method to work?

Microfacet Energy Compensation - Roughness

It turns out a 32^2 table is sufficient. This is “just” 4Kb if stored as floats.

Analytical fits possible as well but less precise.



GGX only

It turns out that 32^2 is sufficient. Stored as floats this is just 4Kb which is the approach we chose for simplicity. We also store a 1D table for E_{avg} with just 32 entries indexed by the roughness.

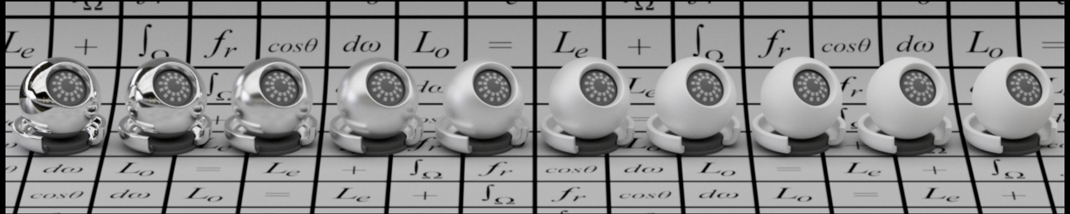
We did also find analytical fits, but they’re not quite as precise. We’ll have some more details in the course notes.

Now our row of microfacet spheres with increasing roughness from before…

Microfacet Energy Compensation - Roughness

It turns out a 32^2 table is sufficient. This is “just” 4Kb if stored as floats.

Analytical fits possible as well but less precise.



GGX + Energy Compensation

…gives the expected result at high roughness. Notice how much energy is recovered even in the center of the roughness range.

Energy Comp ON

Microfacet Energy Compensation - Fresnel

If the surface absorbs or transmits energy, $E(\mu) < 1$. We drew inspiration from Jakob et al., 2014:

Multiply scattered energy is diffused, so make use of the average Fresnel:

$$F_{\text{avg}} = 2\int_0^1 F(\mu)\mu d\mu$$

We know the overall missing energy is $1 - E_{\text{avg}}$ so we can compute the response from successive bounces against the microfacets:

$$F_{\text{avg}}\left(1 - E_{\text{avg}}\right)\sum_{k=0}^{\infty} F_{\text{avg}}^k E_{\text{avg}}^k = \frac{F_{\text{avg}}\left(1 - E_{\text{avg}}\right)}{1 - F_{\text{avg}}E_{\text{avg}}}$$

This factor is a simple multiplier to f_{ms} computed earlier.

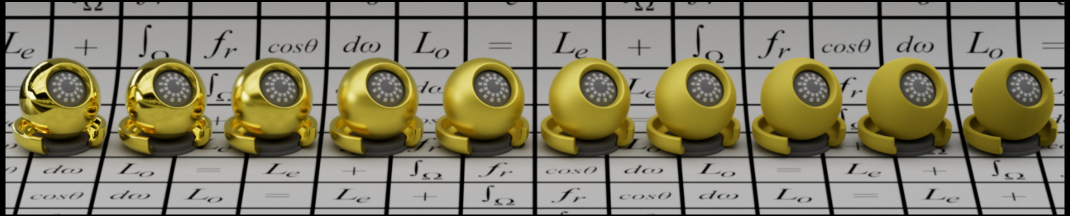
So the next factor to discuss is the Fresnel term. Once we add this to our BRDF, we no longer expect an albedo of 1. Some of the energy must be absorbed.

The solution Jakob and colleagues found makes the following assumption: The multiply scattered energy is diffused, so we can roughly model it with the cosine-weighted average Fresnel as well as the energy loss E_{avg} we used before. In this context, E_{avg} corresponds to how much energy is accounted for by a single microfacet bounce.

Turning that into a small geometric series, we get the following expression, which can just be multiplied against our energy compensation lobe.

Microfacet Energy Compensation - Fresnel

As in previous work, color is maintained across the roughness range, with a slight increase in saturation.


L_e $+$ $\int_\Omega$ f_r $cos\theta$ $d\omega$ L_o $=$

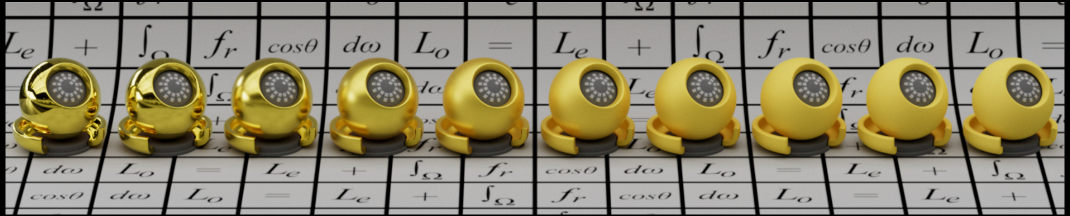
Energy Compensation: Off

Here are the results of the Fresnel compensation applied with a gold material. Without any compensation, the material becomes quite dull looking at high roughness.

Energy Comp OFF

Microfacet Energy Compensation - Fresnel

As in previous work, color is maintained across the roughness range, with a slight increase in saturation.



Energy Compensation: On

And with energy compensation, we maintain the color we expect, with a tiny bit of extra saturation caused by multiple scattering.

Energy Comp ON

Microfacet Energy Compensation - Fresnel

How can we compute F_{avg} efficiently?

- Jakob et al. suggested a Gaussian quadrature.
- Only two to four points are needed for good accuracy
- The following numerical fits are a bit faster...

Again, I've glossed over how we can compute this average Fresnel term. It's defined as yet another integral.

Jakob's paper suggested using Gaussian quadrature, which is actually a very reasonable choice. Even just two points give surprisingly good accuracy, because most Fresnel curves are very smooth.

But we can do a bit better by finding numerical fits to the integral...

Microfacet Energy Compensation - Fresnel

Dielectric Fresnel[*]

$$
\begin{aligned}
g &= \sqrt{\eta^2 + \mu^2 - 1} \\
F(\eta, \mu) &= \frac{1}{2}\left(\frac{g-\mu}{g+\mu}\right)^2 \left(1 + \left(\frac{\mu(g+\mu)-1}{\mu(g-\mu)+1}\right)^2\right) \\
F_{\text{avg}}(\eta) &\approx \frac{\eta - 1}{4.08567 + 1.00071\eta}, 1 < \eta < 400 \\
F_{\text{avg}}(\eta) &\approx 0.997118 + 0.1014\eta - 0.965241\eta^2 - 0.130607\eta^3, 0 < \eta < 1
\end{aligned}
$$

Max Error $\sim 0.65\%$ and $\sim 0.29\%$ respectively.

[*]See Aronson, "Boundary conditions for diffusion of light", 1995 for analytical solutions.

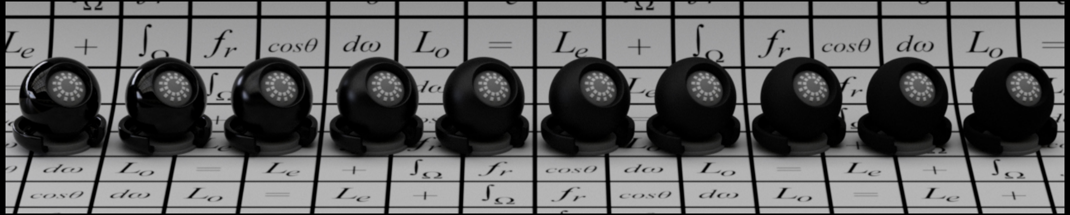
I'm going to go through these quickly, but don't worry: the slides will be posted online.

Dielectric Fresnel is the simplest because it just has one argument: the IOR.

These fits have less than 1% error, which is sufficient. They are also valid over a very wide range - probably more than necessary, but it could be helpful if you have legacy data with strange IOR values.

I've included the fit for IORs less than 1 since we will need it for glass in a few slides.

Microfacet Energy Compensation - Fresnel

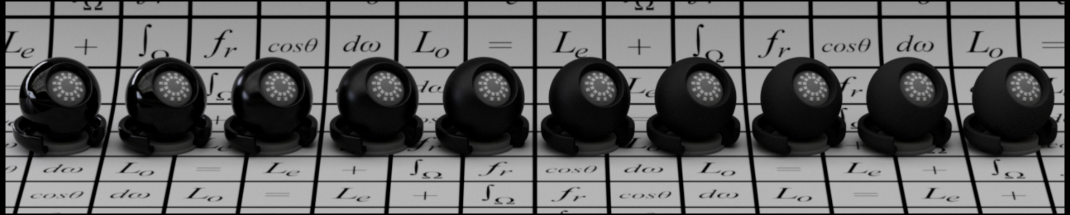


Dielectric Fresnel
Energy Compensation: Off

Here is the result for dielectric Fresnel. In this case the Fresnel effect is very strong and the energy compensation tweak is very subtle.

Energy Comp OFF

Microfacet Energy Compensation - Fresnel



Dielectric Fresnel
Energy Compensation: On

You might not be able to see this tweak from the back, its mostly visible on the right at higher roughness values.

Energy Comp ON

Microfacet Energy Compensation - Fresnel

Conductor Fresnel - Gulbrandsen, 2014 - $r = \text{reflectance}, g = \text{edgetint}$

$$\begin{aligned} F(r,g,\mu) &= \text{see paper...} \\ F_{\text{avg}}(r,g) &\approx 0.087237 + 0.0230685g - 0.0864902g^2 + 0.0774594g^3 \\ &+0.782654r - 0.136432r^2 + 0.278708r^3 \\ &+0.19744gr + 0.0360605g^2r - 0.2586gr^2 \end{aligned}$$

Max error: $\sim 2\%$

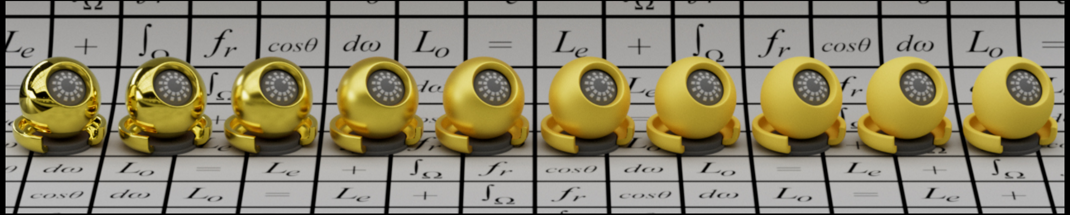
Avg error: $\sim 0.25\%$

Here is the fit for metallic Fresnel. I've used the artist-friendly parameterization by Ole Gulbrandsen, which is how we present the parameters to our users.

The max error is a bit higher in this case, but the average error is still below 1%.

I should say we haven't observed any issues in practice from these fits compared to more precise Gaussian quadrature results. In fact as long as the fits don't go outside the range of 0 to 1, no energy can be created.

Microfacet Energy Compensation - Fresnel


Conductor Fresnel (Physical)

Here are the results for gold that we saw before.

Microfacet Energy Compensation - Fresnel

Artist Friendly - Schlick inspired - $r = \mathrm{reflectance}, g = \mathrm{edgetint}, p = \mathrm{falloff}$

$$F(r,g,p,\mu) = r + (g-r)(1-\mu)^{\frac{1}{p}}$$

$$F_{\mathrm{avg}}(r,g,p) = \frac{2gp^2 + r + 3pr}{1 + 3p + 2p^2}$$

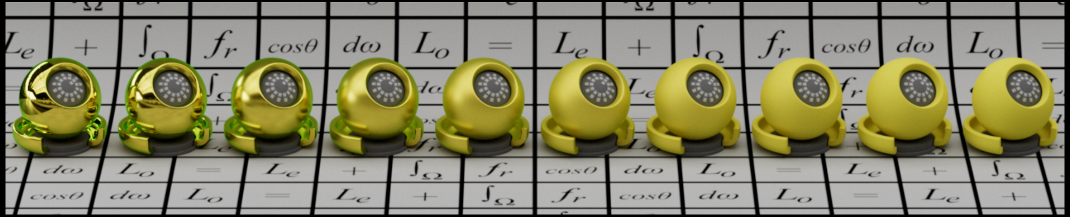
Exact solution. Because the F_{avg} integral is simple, many more creative options can easily be supported.

And one more. This is an even more artist-friendly variant of metals that gives really precise control over the edge color as well as its falloff. The physical Fresnel always goes to 1 at the edge and the edge tint is really subtle.

There are actually lots of physical effects that aren't really modeled by the real Fresnel equations. For example car paints or anodized metals. Rather than get into much fancier physics, we can just let the artists pick the colors they want.

In this case the Fresnel formula is simple, so the average has an exact solution. If you have your own flavor of this, chances are you can probably derive an average Fresnel curve for it as well.

Microfacet Energy Compensation - Fresnel



Conductor Fresnel (Artistic)

Here is an example of the artistic mode with a green tint at the edges.

I promise that our artists use this much better than me.

Microfacet Energy Compensation - Anisotropy

Finally for anisotropy, we use:

$$\begin{aligned} \alpha_x &= \mathrm{rough}^2(1 + \mathrm{aniso}) \\ \alpha_y &= \mathrm{rough}^2(1 - \mathrm{aniso}) \end{aligned}$$

Using E driven by rough alone maintained energy conservation. Intuitively, we are increasing roughness in one direction and decreasing in the other, so the average result is similar.

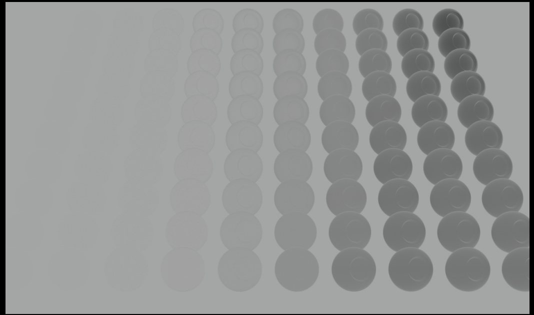
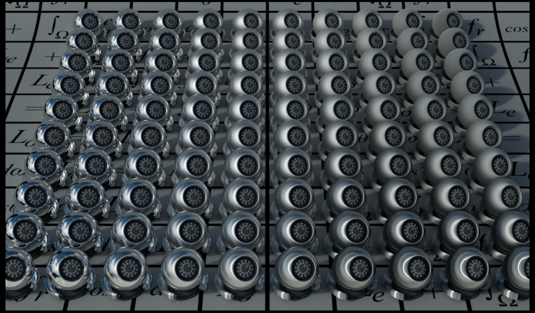
Only $\sim 95\%$ preserving in the worst case, but better than nothing or building larger tables.

The last parameter to talk about is anisotropy. We represent this with a single float that increases the roughness in x while decreasing the roughness in y.

This is very similar to the Disney model, but this version can make perfectly sharp anisotropic highlights, which our artists found helpful.

Rather than try to introduce an extra dimension of tabulation, we decided to just ignore anisotropy and drive energy compensation just from the original roughness. Its not perfect, but 95% conserving is still better than 50% or so we get if doing nothing.

Microfacet Energy Compensation - Anisotropy

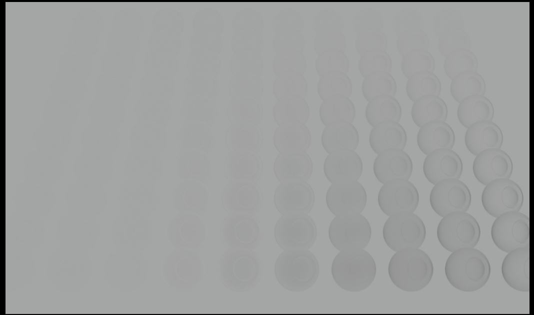
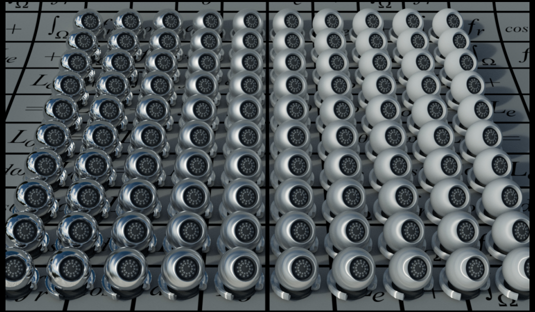



Energy Compensation: Off

Here is a wedge of all possible roughness and anisotropy values, both parameters range from 0 to 1.

Energy Comp OFF

Microfacet Energy Compensation - Anisotropy

Energy Compensation: On

Again the energy compensation term helps recover most of the missing energy.

Energy Comp ON

Microfacet Specular Energy Preservation

Our specular term approximates multiple scattering between microfacets to *preserve* energy as well as *conserve* it.

We reduced tabulation requirements by:

- Roughness: small 4Kb table
- Fresnel: finding analytical fits to F_{avg}
- Anisotropy: using a parameterization that lets us ignore it!

So just to recap, we now have a simple BRDF lobe that approximates the multiple-scattering effects missing from the microfacet lobe. We mostly followed the technique from Kelemen's paper but we reduced tabulation requirements by:

Using a tiny table for roughness.

Using analytical fits for the average Fresnel.

And just ignoring anisotropy completely!

Microfacet Dielectrics - Diffuse Term

For dielectrics, we need a tinted diffuse term as a first approximation to internal scattering. Simply adding a constant diffuse term is not energy conserving at grazing angles, where $F = 1$.

We follow the approach of Kelemen et al., 2001, this time we build another table for the missing energy after reflection by specular. In this case we need to account for both roughness and IOR, so we need a 3D table. We found that 16^3 was sufficient.

That was specular, now let's discuss the diffuse term. This is actually what Kelemen's paper was originally about.

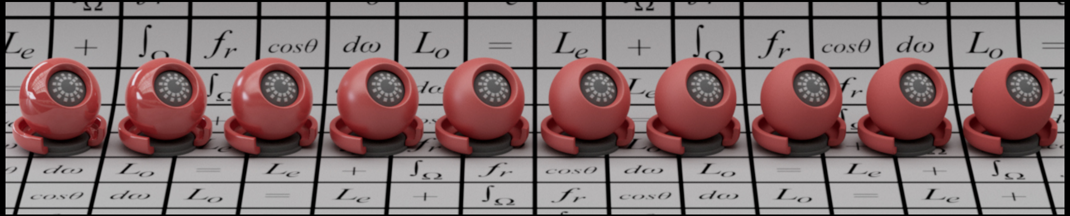
Just adding a constant diffuse term can never be energy conserving because Fresnel goes to 1 at the edges.

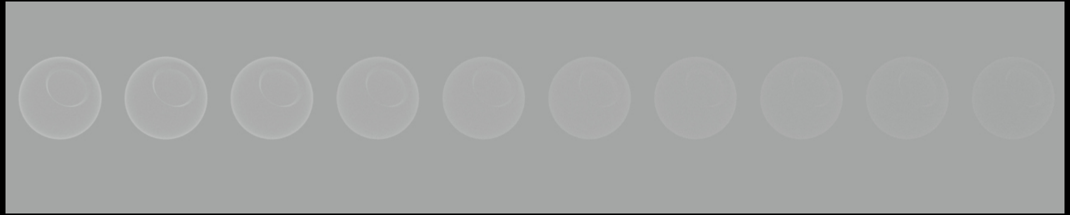
But we can reuse the same technique that helped us get energy *preservation* on just the specular term to get energy *conservation* for the specular/diffuse combination.

This time we need to record the directional albedo of the entire BRDF, so we need to tabulate both roughness and IOR. Together with the incoming angle cosine, this makes a 3D table.

The good news is that the energy loss function is even smoother than before. We found we could get away with 16 entries for each dimension, which is 16Kb stored as floats.

Microfacet Dielectrics - Diffuse Term



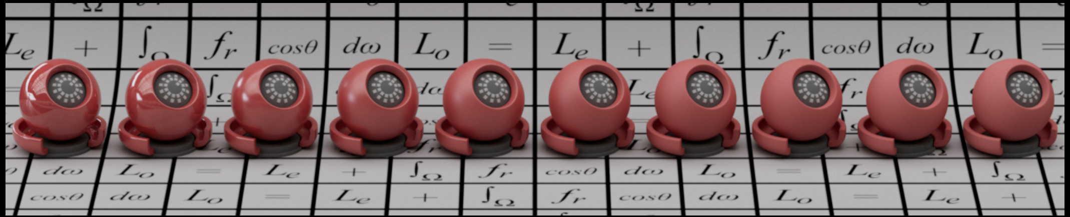


Energy Conservation: Off

Here are the results. Again just naively add a microfacet specular model to a constant diffuse lobe, we get extra energy at grazing angles. This is more pronounced at low roughness but still happens at high roughness.

Energy comp OFF

Microfacet Dielectrics - Diffuse Term

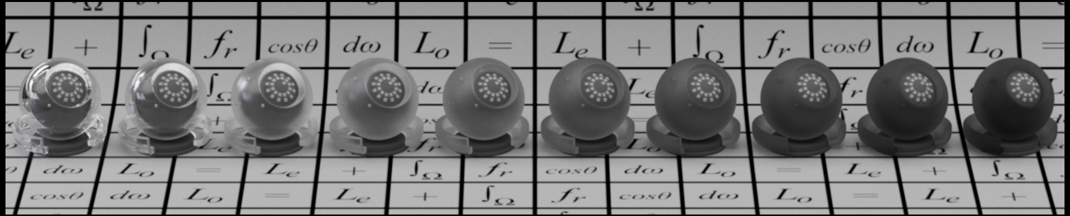


Energy Conservation: On

And when we modulate diffuse by the energy compensation curves, we get a perfectly energy conserving result.

Energy comp ON

Microfacet Dielectrics - Transmission



For transmission, we need to take care to obey the reciprocity condition:

$$f(\omega_o, \omega_i) = \eta^2 f(\omega_i, \omega_o)$$

We tabulate the energy loss E over the sphere for $\eta \in [1, 3]$ and $\eta \in \left[\frac{1}{3}, 1\right]$ in 16^3 tables, for varying roughness values and incident angles.

So now we arrive at the most complicated case, which is glass.

This is also where the energy loss issue is most severe because it typically takes several bounces to leave the surface.

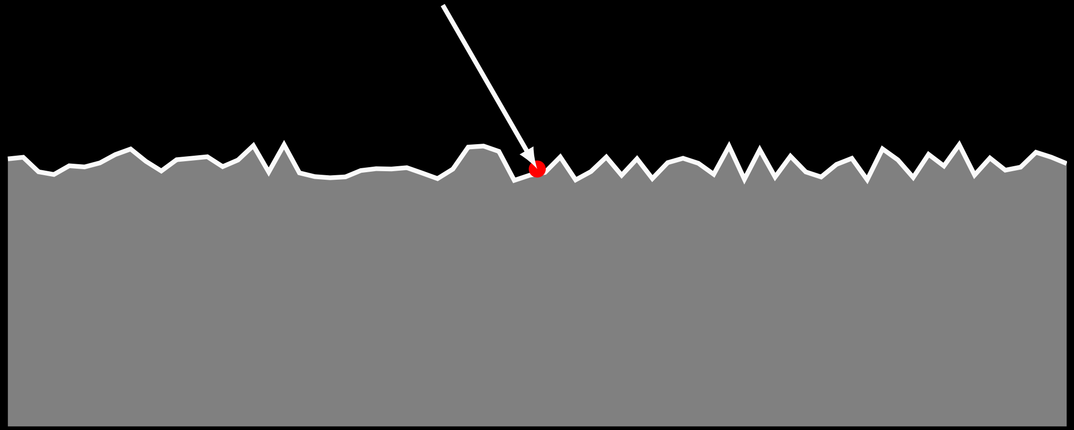
Another complicating factor is the reciprocity condition. Exchange view and light directions when changing mediums requires scaling by the square of the IOR. This is all explained in Eric Veach's thesis and needs to be accounted for properly when using bidirectional methods.

Just like in the dielectric diffuse case, both roughness and IOR matter here. We again use 3D tables, but this time integrated over the whole sphere.

Luckily 16 entries per dimension seem to work fine here as well since the functions are very smooth.

Microfacet Dielectrics - Transmission

Entering IOR 1 $\rightarrow \eta$

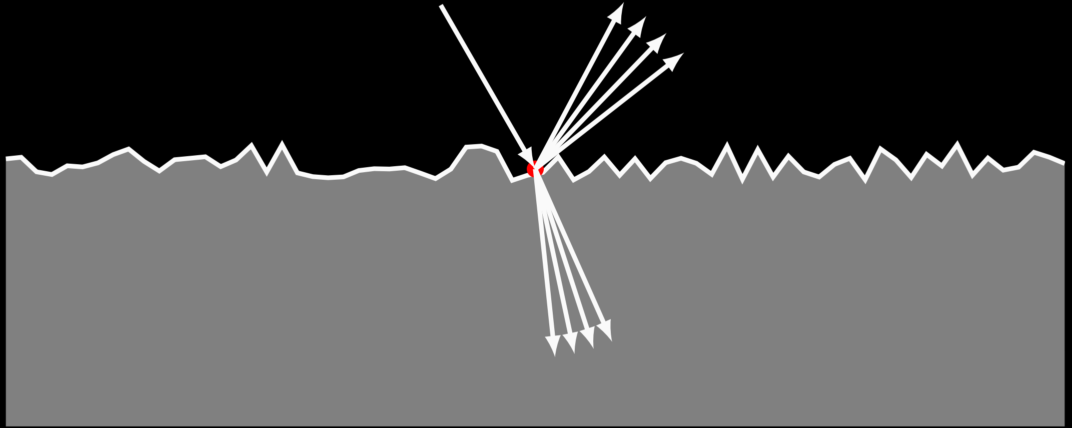


I'm going to run through an example here to help build up the compensation lobes for reflection and transmission.

We'll start with a ray that enters a surface.

Microfacet Dielectrics - Transmission

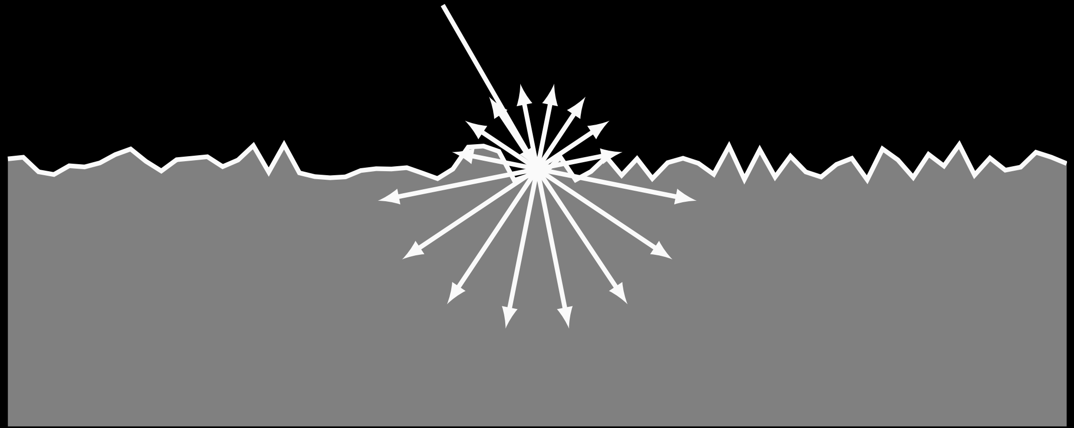
Microfacets scatter $E(\eta, \mu_o)$



The microfacets scatter light both above and below the surface.

Microfacet Dielectrics - Transmission

Missing energy is $1 - E(\eta, \mu_o)$



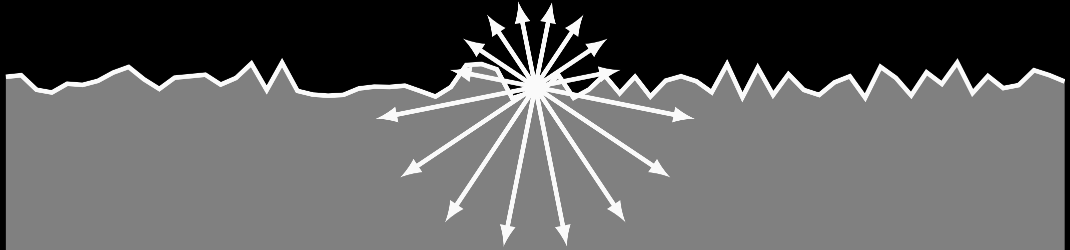
Some energy is lost because of the single-scattering assumption.

We want to build an energy compensation term that will recover the missing energy into reflection and transmission in the right amounts.

The formulas are going to look really similar to the BRDF case, but I'll introduce them term by term to explain what each one does.

Microfacet Dielectrics - Transmission

$$f_{\mathrm{msR}} = \qquad (1 - E(\eta, \mu_o))$$

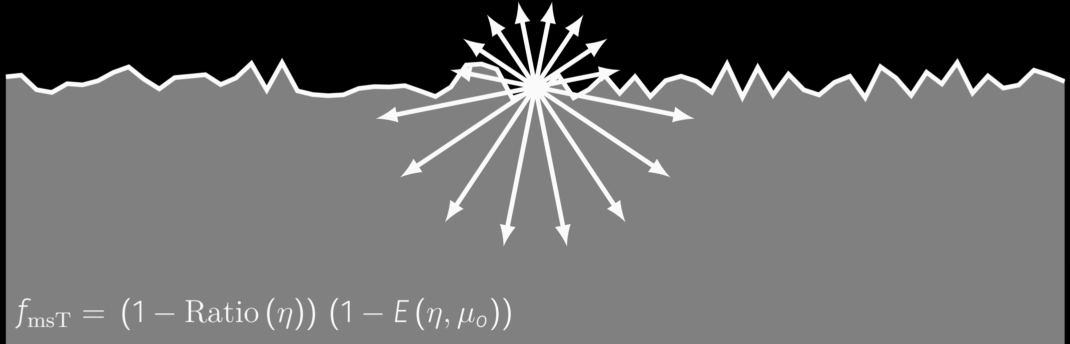


$$f_{\mathrm{msT}} = \qquad (1 - E(\eta, \mu_o))$$

The first is $1 - E$ evaluated with the viewing cosine. That represents the amount of energy we are trying to recover. Remember from the earlier slide that this is the term that was left after everything else cancels out.

Microfacet Dielectrics - Transmission

$$f_{\mathrm{msR}} = \mathrm{Ratio}(\eta)\,(1 - E(\eta, \mu_o))$$

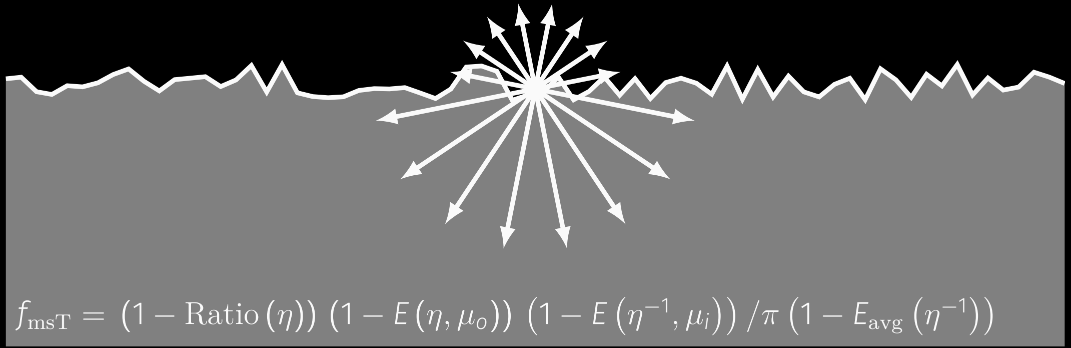


$$f_{\mathrm{msT}} = (1 - \mathrm{Ratio}(\eta))\,(1 - E(\eta, \mu_o))$$

Because we have reflection and transmission lobes now, we need to split the energy by some ratio, which I'll define in a minute.

Microfacet Dielectrics - Transmission

$$f_{\mathrm{msR}} = \mathrm{Ratio}\left(\eta\right)\,\left(1 - E\left(\eta, \mu_o\right)\right)\,\left(1 - E\left(\eta, \mu_i\right)\right)/\pi\left(1 - E_{\mathrm{avg}}\left(\eta\right)\right)$$



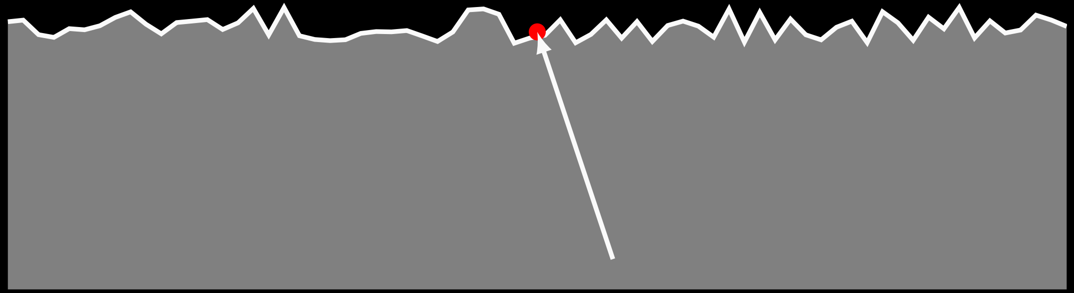
$$f_{\mathrm{msT}} = \left(1 - \mathrm{Ratio}\left(\eta\right)\right)\,\left(1 - E\left(\eta, \mu_o\right)\right)\,\left(1 - E\left(\eta^{-1}, \mu_i\right)\right)/\pi\left(1 - E_{\mathrm{avg}}\left(\eta^{-1}\right)\right)$$

And finally we add the remaining terms, which are designed to integrate to 1. The reflection term works just like before, but for transmission, notice that I am using the energy curve tabulated from the opposite side.

This convention comes from the reciprocity requirement. Exchanging directions also means inverting the IOR.

Microfacet Dielectrics - Transmission

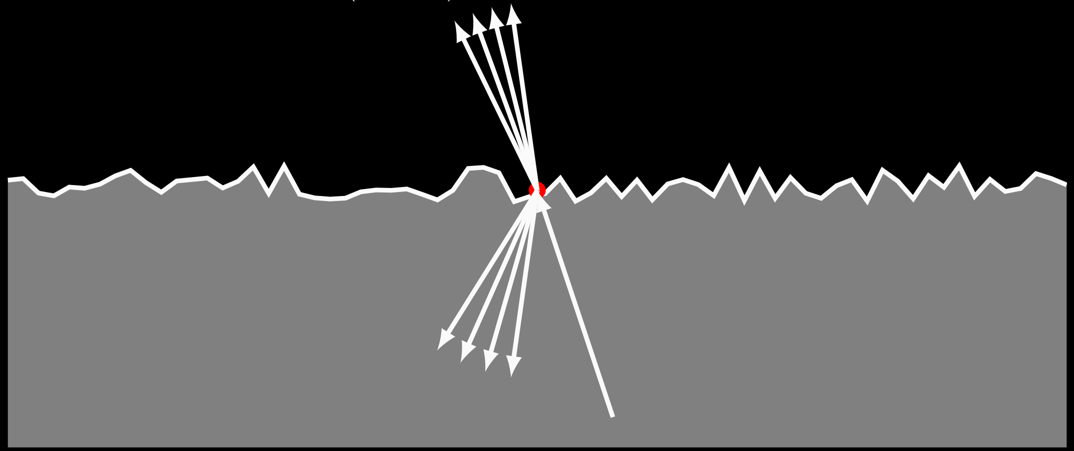
Leaving IOR $\eta \to 1$



I'll just quickly show the reverse case where a ray leaves the surface.

Microfacet Dielectrics - Transmission

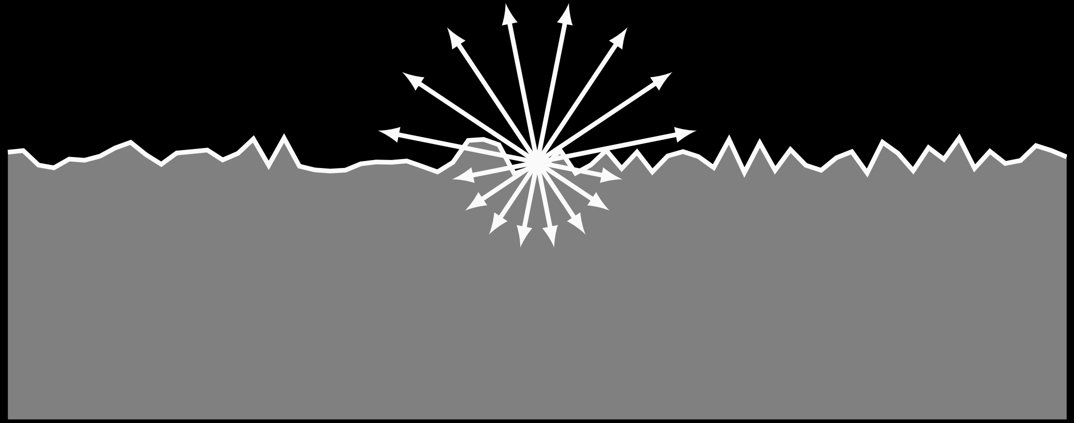
Microfacets scatter $E\left(\eta^{-1}, \mu_o\right)$



Again the microfacet model scatters only a fraction of the available energy.

Microfacet Dielectrics - Transmission

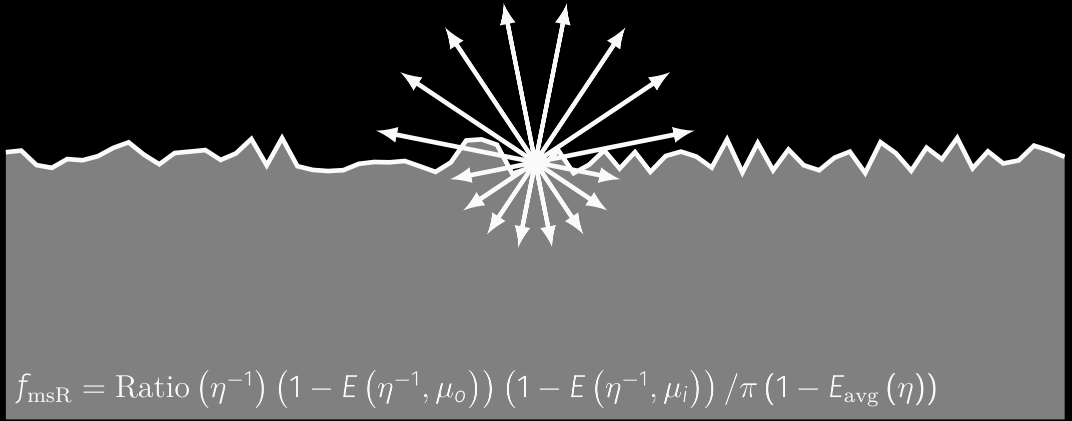
Missing energy is $1 - E\left(\eta^{-1}, \mu_o\right)$



Notice that we are evaluating E with the inverse IOR now.

Microfacet Dielectrics - Transmission

$$f_{\mathrm{msT}} = \left(1 - \mathrm{Ratio}\left(\eta^{-1}\right)\right)\left(1 - E\left(\eta^{-1}, \mu_o\right)\right)\left(1 - E\left(\eta, \mu_i\right)\right)/\pi\left(1 - E_{\mathrm{avg}}\left(\eta\right)\right)$$



$$f_{\mathrm{msR}} = \mathrm{Ratio}\left(\eta^{-1}\right)\left(1 - E\left(\eta^{-1}, \mu_o\right)\right)\left(1 - E\left(\eta^{-1}, \mu_i\right)\right)/\pi\left(1 - E_{\mathrm{avg}}\left(\eta\right)\right)$$

The reflection and refraction terms are just like before, but with the IOR flipped. The transmission term on the top is the one that uses the *E* function from the IOR and its inverse.

Glass Energy Conservation - Reciprocity

The BRDF lobes are already reciprocal.

The BTDF lobes have reciprocal varying terms by design. Need to choose Ratio such that constant terms match:

$$\frac{1-\mathrm{Ratio}\left(\eta\right)}{1-E_{\mathrm{avg}}\left(\eta^{-1}\right)} = \frac{1-\mathrm{Ratio}\left(\eta^{-1}\right)}{1-E_{\mathrm{avg}}\left(\eta\right)}\eta^{2}$$

One equation, two unknowns: $\mathrm{Ratio}\left(\eta\right)$ and $\mathrm{Ratio}\left(\eta^{-1}\right)$.

I've given the overall form of the multiple-scattering lobes. Now we just need to make sure they are truly reciprocal.

The part of the formula that depends on the view and light cosines is reciprocal by design. So we just need to make sure the overall scaling factors match.

I didn't specify what the ratio was yet, so we'll use that degree of freedom to ensure things are equal.

But we just have one equation and two unknowns: the ratio when entering the surface and the ratio when leaving the surface.

Glass Energy Conservation - Reciprocity

Let's make an educated guess that $\mathrm{Ratio}(\eta) \approx F_{\mathrm{avg}}(\eta)$. Now we just need to solve for x that makes both sides match:

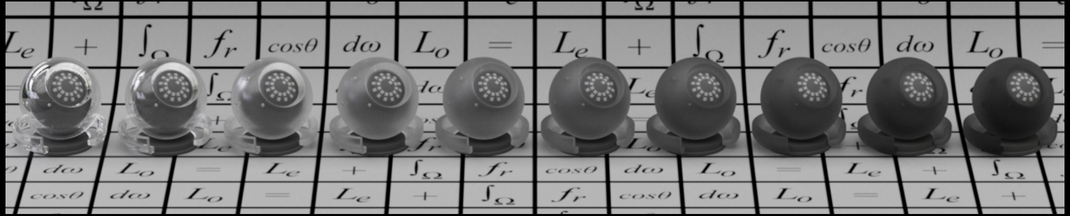
$$x \frac{1 - F_{\mathrm{avg}}(\eta)}{1 - E_{\mathrm{avg}}(\eta^{-1})} = (1 - x) \frac{1 - F_{\mathrm{avg}}(\eta^{-1})}{1 - E_{\mathrm{avg}}(\eta)} \eta^2$$

This gives us the final ratio between BRDF and BTDF lobes, completing the model.

To reduce this a bit more, we make the guess that the ratio will be close to the average Fresnel. Then we can just solve for a small scaling factor *x* that equalizes both sides.

That lets us define the final ratio of reflection to transmission, which completes the model.

Glass Energy Conservation - Result

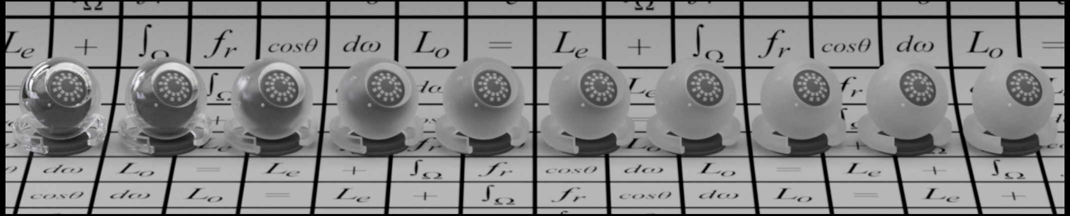

Energy Preservation: Off

That was a lot of math, so lets see if it was worth it.

Here is the row of glass spheres without any energy preservation…

Energy comp OFF

Glass Energy Conservation - Result



Energy Preservation: On

…and now adding our extra lobe.

So just like before, this compensation makes a huge difference to the look - even for moderate roughness values.

Energy comp ON

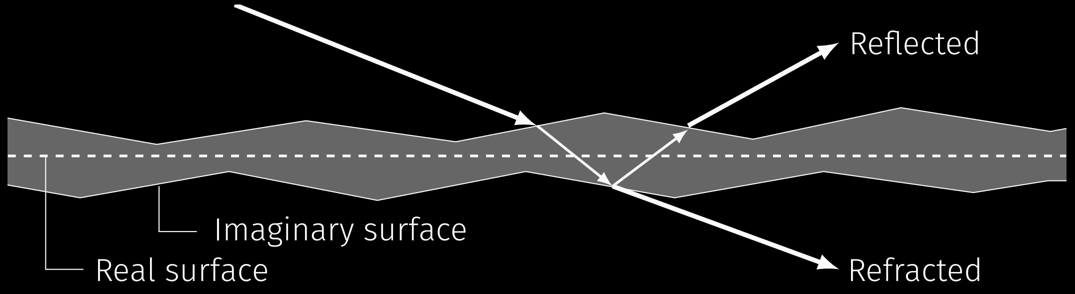
Thin Surface BSDF

I’m now going to hand the talk to Alex who will describe the *thin* version of glass.

We also have dielectric thin surface simulation

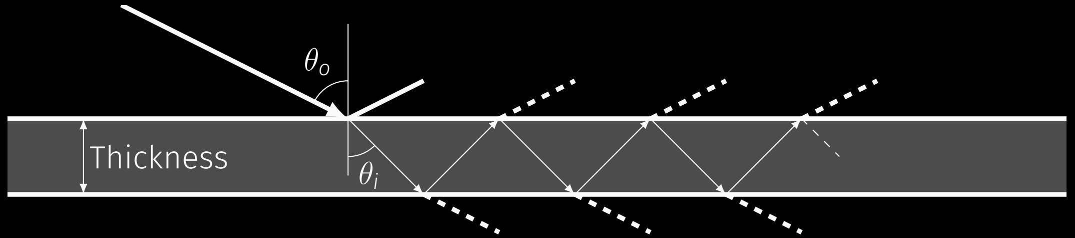
Thin Surface

Thin surface is thought of as a double refraction event that happens at a surface that is actually modeled as a single surface. It serves as a cheap way of rendering windows in large buildings, leaves, or butterfly wings.



Thin surface lets us fake a fully modeled thin object using only a single surface. It is very useful for windows, leaves, or butterfly wings. The kind of object that would give numerical problems if modeled with its two sides.

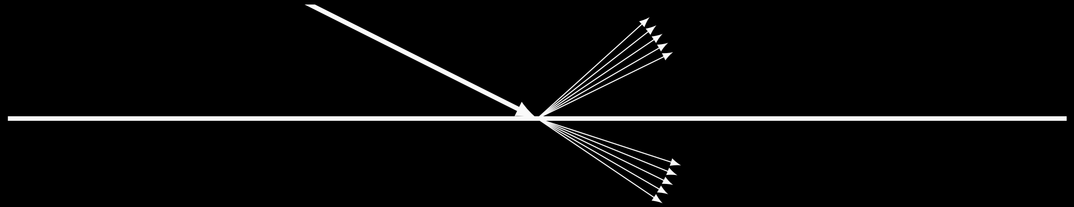
Thin reflect/refract exit summation



If $\sin\theta_o = \sin\theta_i \cdot \eta$, then $F_o = fresnel(\cos\theta_o, \eta), F_i = fresnel(\cos\theta_i, 1/\eta)$ are the required Fresnel terms and $A = e^{-t\ \sigma_a/cos\theta_i}$ is the absorption from bounce to bounce inside the layer, where t is the thickness. All the exits have to be summed up.

We take into account the two refractions, the reflections, Fresnel terms and multiple bounces so all the energy is preserved. We just assume the surfaces is locally flat.

Thin Surface



At the macro level, and due to roughness we have two blurred lobes that are a result of the geometric series summation, $F_o + (1 - F_o)(1 - F_i)\frac{F_i A^2}{1 - F_i^2 A^2}$ for reflection and $(1 - F_o)A(1 - F_i)\left(1 + \frac{F_i^2 A^2}{1 - F_i^2 A^2}\right)$ for refraction, where F_o is the Fresnel factor for the outside of the layer, F_i the inside Fresnel and A the absorption along the traveled distance within the layer.

We end up with two lobes, reflection and refraction, with weights coming from the series summation of the bounces and the associated absorption.

Thin roughness mismatch

We are using a **mirrored reflection** to fake double refraction and the roughness behaves differently because reflection and refraction have **different *Jacobian* transformations**.

We can either find what roughness would give us the same peak PDF with the different *Jacobians* or assume $\mathrm{Var}(\omega_h) \simeq r^4$ and scale it by the square derivative of the transformation, which is roughly the inverse *Jacobian*.

We get to the same relationship either way!

We use a mirrored reflection for the refraction lobe, but the roughness impact is different due to the distinct Jacobians of both transformations. We got hints as how this difference applies by relating how the two Jacobians transform the ray PDF.

Roughness mapping (equal peak PDF method)

For viewing, half, refracted and mirrored vectors $\omega_o, \omega_h, \omega_r, \omega_m$...

$$\begin{aligned}
\left\| \frac{\partial \omega_h}{\partial \omega_r} \right\| D_{max}(r) &= \left\| \frac{\partial \omega_h}{\partial \omega_m} \right\| D_{max}(r_{thin}) \\
\frac{|\omega_r \cdot \omega_h|\, \eta^2}{((\omega_r \cdot \omega_h)\eta + (\omega_o \cdot \omega_h))^2} \frac{1}{\pi r^4} &= \frac{1}{4(\omega_o \cdot \omega_h)} \frac{1}{\pi r_{thin}^4} \\
\sqrt{\frac{\eta - 1}{2\eta}}\, r &= r_{thin} \quad \text{(Assuming } (\omega_r \cdot \omega_h) \simeq -(\omega_o \cdot \omega_h))
\end{aligned}$$

But in the thick case we get two refraction events, so there is still a missing roughness scale factor $S(\eta)$ that in practice is not a constant. We found $S(\eta) = \frac{2\eta - 1}{\eta}\sqrt{1.7}$ to work well.

We basically equate the maximum possible PDF parametrized by the roughness using the different Jacobians on each side and find what the needed roughness scaling would be to satisfy the constraint. We get a similar result with a rough estimation of the variance. And finally we manually adjusted to account for the fact that we have two refraction events

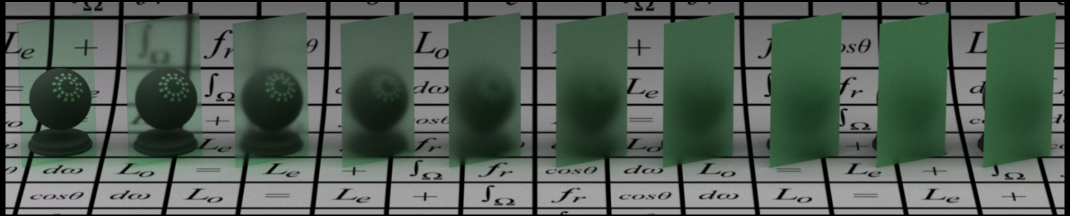
Roughness mapping (final curve)

- We made a lot of assumptions to get here but...
  - The transformed roughness vanishes as $\eta \to 1$ as expected
  - Converges to a finite value as $\eta \to \infty$
  - The remaining scale due to double refraction is hard to predict and was found empirically going from $\eta = 1$ to a maximum of 2.6
- The final curve goes from $0\times$ to $1.84\times$ and is

$$r_{thin} = r\sqrt{\frac{3.7\,(\eta - 1)\,(\eta - 0.5)^2}{\eta^3}}$$

So all this partially obscure math may be a bit lousy but it gives a good starting curve that we could shape to out needs and has the expected behavior. And it's giving us a good mapping.

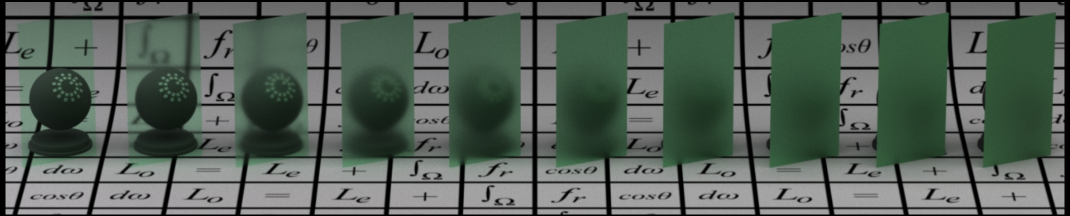
Thin Surface



Thick surface with glass shader (reference)

This is a render with modeled thickness and our glass BSDF

Thin Surface


Thin surface approximation

We get a very close match to the roughness distribution, and color is approximately preserved. An exact match is not required, since thin and thick surfaces are rarely interchanged.

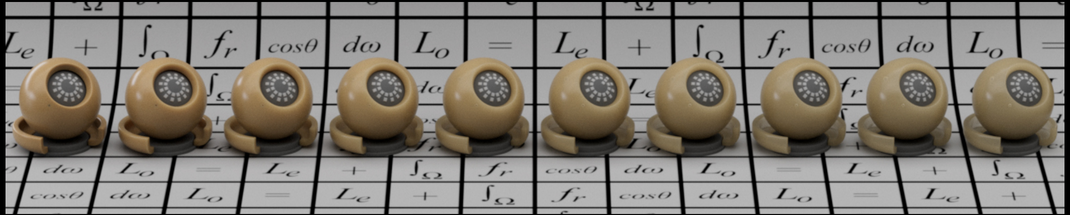
And this is a single surface using our thin layer. The roughness matches very well and the color from absorption is close enough. They normally don't switch from one to the other, but we wanted to be as close as possible.

Subsurface Scattering

That covers all the surface scattering models - now we can talk about subsurface scattering. I'll hand it back to Chris.

Subsurface Scattering

We define the density of the medium by its mean free path:



$$\sigma_t = \frac{1}{\text{mfp}}, \sigma_s = \frac{\alpha}{\text{mfp}}$$

We describe subsurface scattering as a homogeneous medium of constant density that we reach through a dielectric boundary. In other words, its represented as a constant volume inside glass.

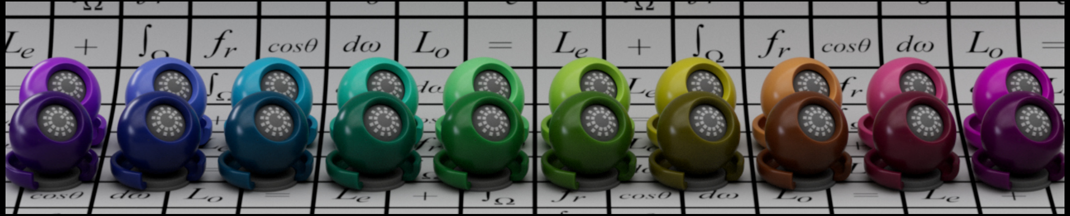
This morning I explained how we allow surface shaders to define their own volumetric interior with a priority based system. Now I want to also describe how the artists control the appearance.

We define the overall density of particles by a mean free path or scattering radius which lets us control the amount of bleed.

And then the color is defined by the albedo α of each particle.

Subsurface Scattering - Color control

Single-scattering albedo α predicts overall color poorly:



We adopted an approximation from van de Hulst, 1980*:

$$C = \frac{(1-s)(1-0.139s)}{1+1.17s}, s = \sqrt{\frac{1-\alpha}{1-\alpha g}}$$

*See d'Eon, *A Hitchhiker's Guide to Multiple Scattering*, 2016 for more details

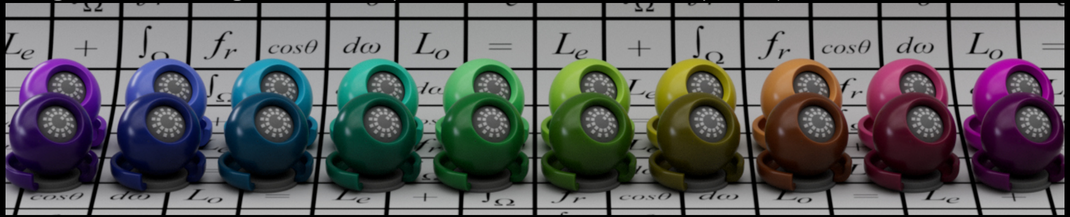
Conceptually α has the same role diffuse color, but setting them to the same value produces very different results. Here the front row uses subsurface scattering while the back row is a diffuse set to the same color.

This was already discussed in the previous talks, but I'll just mention that we use the following formula, which comes from the astrophysics literature.

This formula predicts what color C comes out for given values of α and g (the phase function eccentricity). The assumptions this makes is that we have a semi-infinite half space lit uniformly from all directions. This is never exactly what we have, but for typical scattering distances it works really well.

Subsurface Scattering - Color control

Single-scattering albedo α predicts overall color poorly:



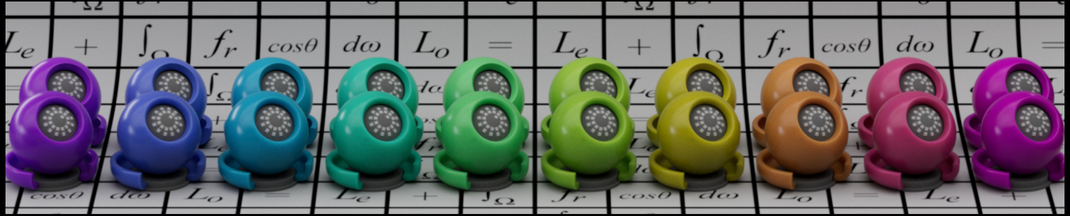
We actually need the inverse formula:

$$
\begin{aligned}
s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\
\alpha &= \frac{1 - s^2}{1 - gs^2}
\end{aligned}
$$

Of course, in our case we want the inverse of this formula because we want to know what value of α to use to achieve a given color C.

Subsurface Scattering - Color control

Applying the correction makes SSS color track better with diffuse:



We actually need the inverse formula:

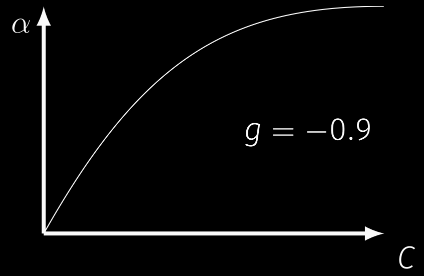
$$\begin{aligned} s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\ \alpha &= \frac{1 - s^2}{1 - gs^2} \end{aligned}$$

And here is the result of this remapping.

Obviously this makes the model a lot more user-friendly for artists.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

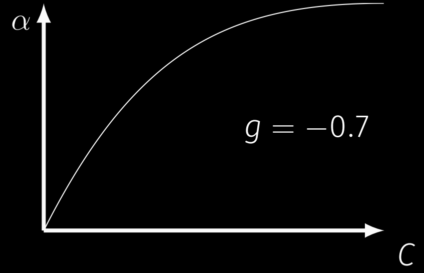


Mapping accounts for influence of g

I just want to show a plot of the curve of the equation here, to give a sense of what it's doing. I'm going to be changing g from backward scattering towards forward scattering...

Subsurface Scattering - Albedo

$$
\begin{aligned}
s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\
\alpha &= \frac{1 - s^2}{1 - gs^2}
\end{aligned}
$$

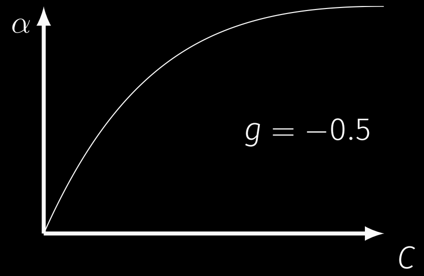


Mapping accounts for influence of g

…

Subsurface Scattering - Albedo

$$\begin{aligned} s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\ \alpha &= \frac{1 - s^2}{1 - gs^2} \end{aligned}$$

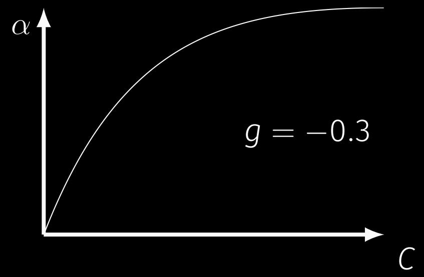


Mapping accounts for influence of g

...

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

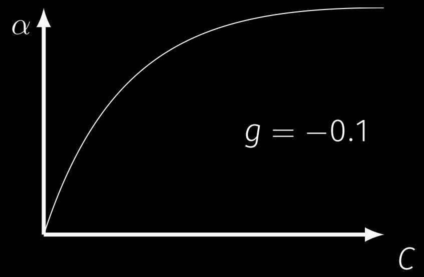


Mapping accounts for influence of g

…so the curve bends more and more…

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

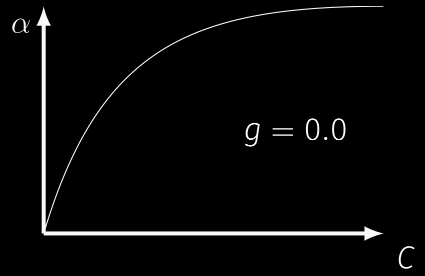


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

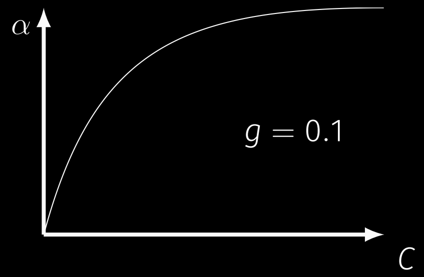


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

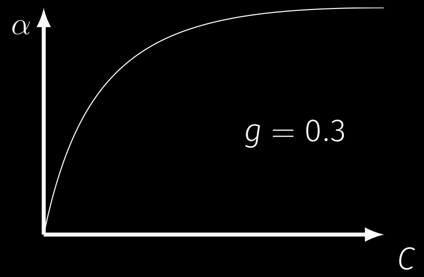


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$
\begin{aligned}
s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\
\alpha &= \frac{1 - s^2}{1 - gs^2}
\end{aligned}
$$

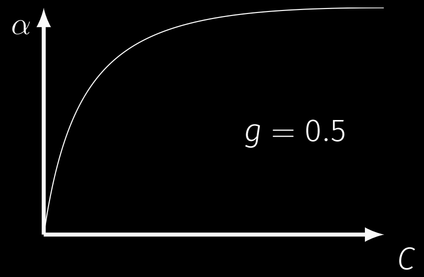


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

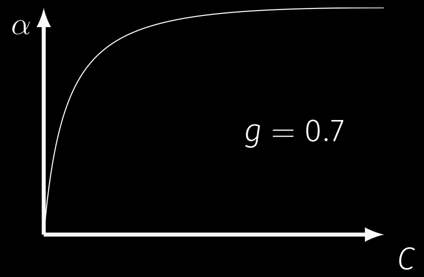


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

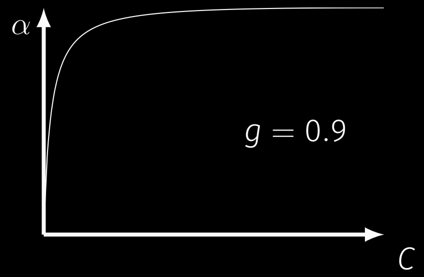


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$s = 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2}$$
$$\alpha = \frac{1 - s^2}{1 - gs^2}$$

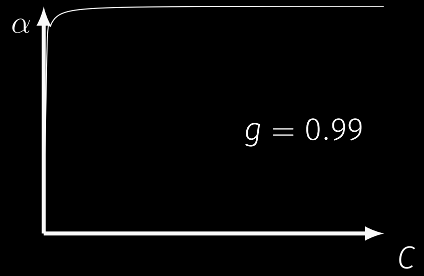


Mapping accounts for influence of g

This is because it takes a brighter albedo to reflect the light back towards the boundary.

Subsurface Scattering - Albedo

$$\begin{aligned} s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\ \alpha &= \frac{1 - s^2}{1 - gs^2} \end{aligned}$$



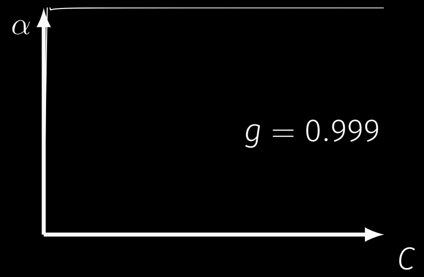
Mapping becomes unstable as $g \to 1$

Once we get to a purely forward scattering phase function, the curve becomes unstable.

This makes sense, because the model assumes we have to get back to the entry point to return any color, which is more and more unlikely.

Subsurface Scattering - Albedo

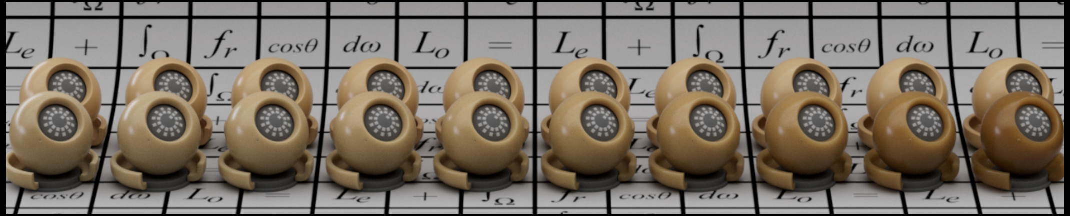
$$
\begin{aligned}
s &= 4.09712 + 4.20863C - \sqrt{9.59217 + 41.6808C + 17.7126C^2} \\
\alpha &= \frac{1 - s^2}{1 - gs^2}
\end{aligned}
$$



Mapping becomes unstable as $g \to 1$

Of course in practice we don't really need to go this far, so we clamp the range of g to avoid these extreme values.

Subsurface Scattering - Eccentricity



Albedo remapping with $g = 0$

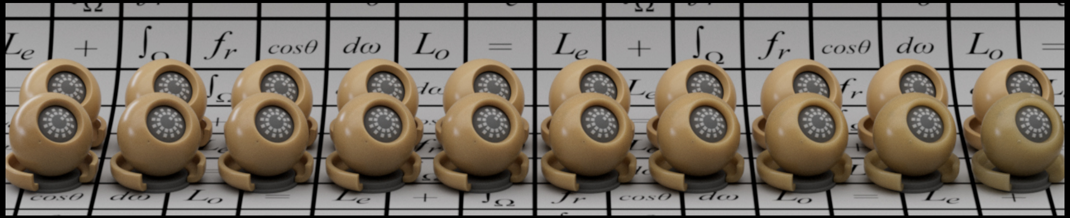
Here is the result showing the influence of g changing from -1 to 1 from left to right. Again I put a diffuse sphere in the back for comparison.

Remapping the albedo but ignoring g causes an intensity change.

This is mostly visible on the right where the material is forward scattering. It gets dimmer because most of the light is pushed away from the viewer.

A bit more subtle, but on the left the spheres are brighter because backward scattering pushes light back out very quickly.

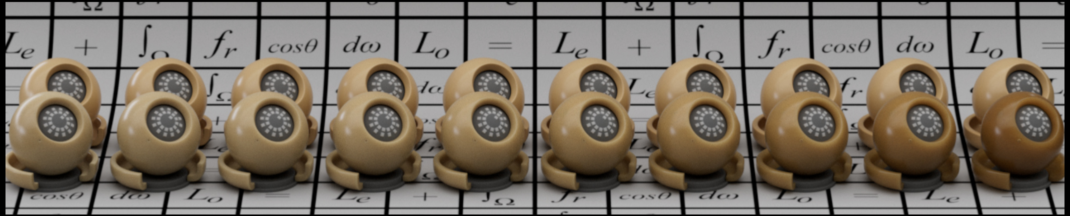
Subsurface Scattering - Eccentricity



Albedo remapping using *g*

Using *g* in the mapping helps equalize the color across the whole range. Although the more forward scattering cases are still a bit difficult to match exactly.

Subsurface Scattering - Eccentricity



Albedo remapping with $g = 0$

Some artists preferred g to change the intensity!

However it turns out that our artists didn't always like this automatic remapping, so we leave it optional in our shaders. The quantities passed to the renderer are just the raw volume parameters, and we leave all the remapping logic in the shader. So its very easy to offer different parameterizations for different cases.

Its also worth pointing out that the fact that this remapping works at all motivates the field of similarity theory. In many cases, there is a set of parameters with isotropic scattering that mimics the overall effect of a forward scattering material.

Subsurface Scattering

Efficient techniques for volumetric scattering discussed in “Production Volume Rendering” course from this morning.

To make brute-force *SSS* practical, caustics from the refractive boundary must be resolved somehow. We wanted SSS to rely on the same solution as other cases, to keep the renderer consistent and predictable.

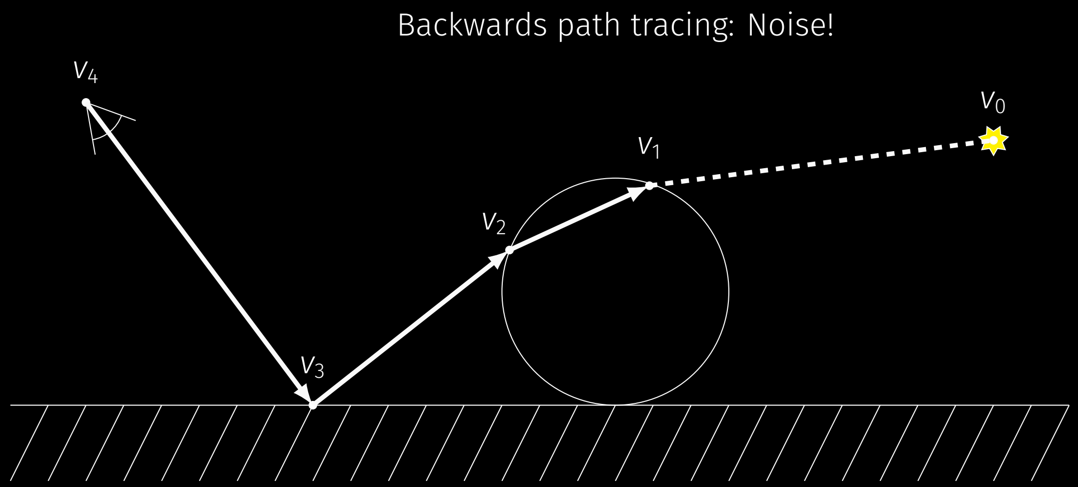
Techniques for efficient volume integration were discussed in detail this morning.

I am now going to hand the talk to Alex who will discuss the important detail of how we approximate the influence of the boundary on subsurface scattering. In fact, this is more broadly related to how we approximate caustics in general in our path tracer.

And then Alex will also talk about the remaining features of our shading model.

Caustic Path Avoidance

The caustic issue in path tracing

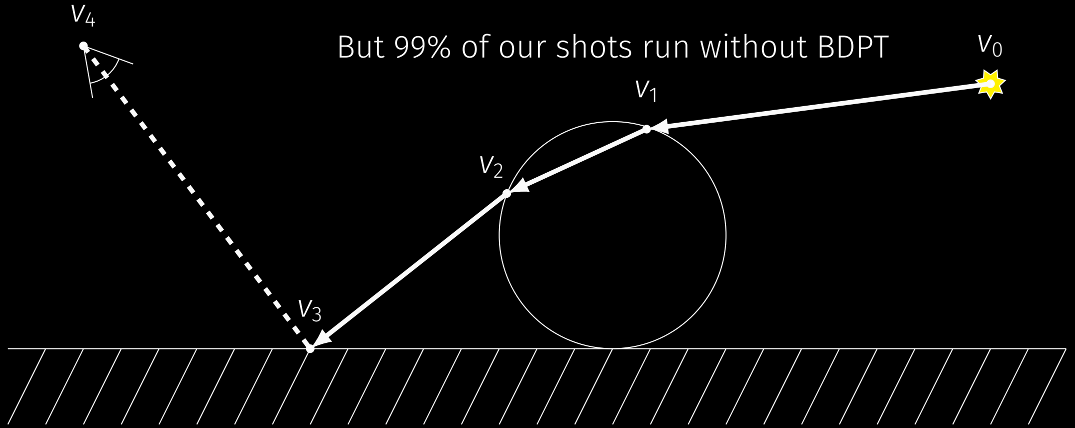


This is a typical caustic case, a sphere concentrating light coming from a point. Regular backwards path tracing will have a really hard time finding the light source by BSDF sampling, and next event estimation is not possible since light is blocked by the glass. It produces a lot of noise.

The caustic issue in path tracing

Forward tracing from light: Good!

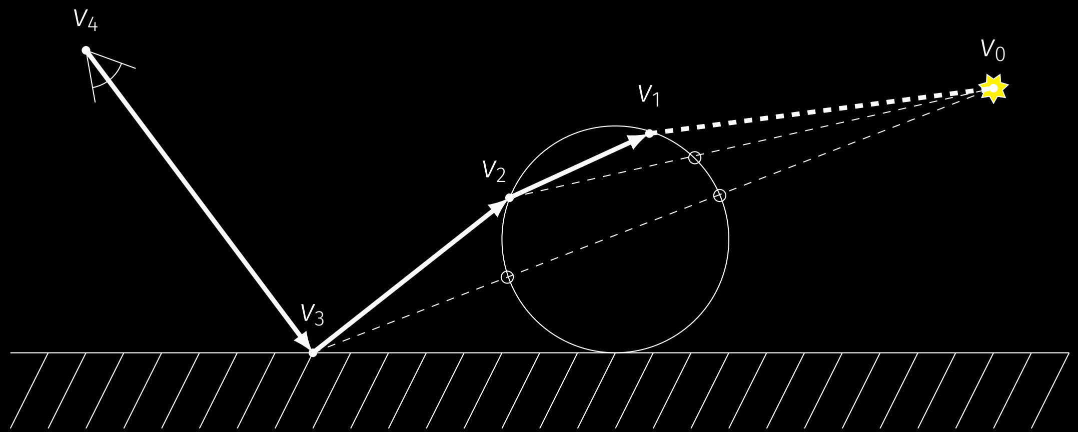
But 99% of our shots run without BDPT



Forward path tracing solves this problem. But most of the work we do doesn’t use these fancy integrators because of performance reasons. And most of the time we don’t care about fancy light shapes and lens effects; most of the time we just want light to go through a glass window.

The caustic issue in path tracing

Fake caustics via transparent shadows



Our solution: transparent shadows. Just pretend the glass is transparent and light doesn’t bend. Unless the glass is rough, in which case we could afford the real thing. But this means we have to deal with a couple of issues.

Making path tracing caustic noise resistant

1. Clamp roughness as we bounce off surfaces
   - Roughness always grows along the path
   - Smooths out caustics, but also dissolves them
   - Solves the noise
2. Apply transparent shadows
   - Fills in missing energy from transparent geometry
3. Solve double lighting
   - Path tracing and transparent shadows add up
   - We need to weight them to compensate

First we have to make sure our indirect lighting doesn't throw in noisy caustics. We do this clamping the roughness. If you bounce off a diffuse surface all glass will appear rough for the rest of the walk. Then we apply transparent shadows to fill in the missing energy from smooth glass. And finally, we have to deal with double lighting. Note that both indirect lighting and transparent shadows try to guess the same thing and they add up.

BDPT-based weighting

In a BDPT context, backwards path tracing is only one more sampling method, and for caustics it is outweighted by the forward tracing one.

$$PTweight = 1/\left(1 + \frac{P(v_{n-1} \leftarrow v_{n-2})\dots P(v_2 \leftarrow v_1)P(v_1 \leftarrow v_0)}{P(v_{n-1} \rightarrow v_{n-2})\dots P(v_0)}\dots\right)$$

There are many other terms in the denominator, but usually one will become big to bring down *PTweight*. Assuming $P(v_i \rightarrow v_{i+1}) \simeq P(v_{i-1} \leftarrow v_i)$, the size of the winning term will be related to $\frac{P(v_i \leftarrow v_{i+1})}{P(v_i \rightarrow v_{i+1})}$ at the "caustic connection".

We cannot compute the exact BDPT weight because we are missing information, but we can use a heuristic based on subsequent bounces and their PDFs, which are directly related to the surface roughness.

To avoid the double lighting we weight our two sources of light: transparent shadows and the indirect sampling, so their weights add up to one. We got inspired by BDPT weighting, where the method with the highest probability wins. We have to simplify it because we are missing a lot of information, but it all comes down to the roughness difference between one scattering event and the next.

A roughness-based heuristic

We have two subsequent path vertices A and B. B might be found by indirect ray sampling B_p, or as a transparent hit in a shadow ray B_t. Depending on the roughness, we will choose the weights $w(B_p)$ and $w(B_t)$.

- If $rough(A) \gg rough(B) \rightarrow w(B_t) \gg w(B_p)$ (favor transparent shadows)
- If $rough(A) \leq rough(B) \rightarrow w(B_t) \ll w(B_p)$ (favor indirect lighting)
- $w(B_t) = \max\{rough(A) - rough(B), 0\}$, $w(B_p) = 1 - w(B_t)$
- Recalculate using power heuristic: $w'(B_t) = \frac{w(B_t)^2}{w(B_t)^2 + w(B_p)^2}$

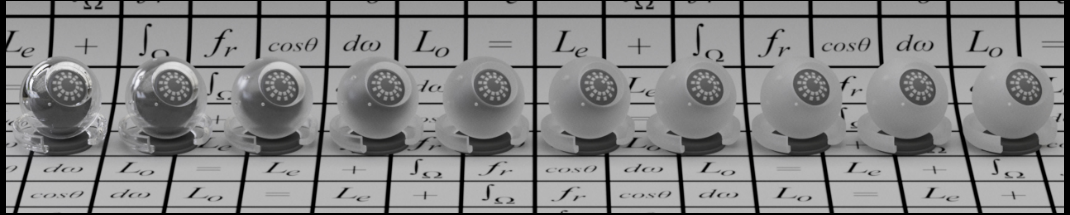
Long story short: if the surface is rough and the occluder is smooth, transparent shadows are the best approach and their weight will be high. Otherwise if the surface is smooth, whatever comes next is better treated by regular path tracing and transparent shadows are weighted down. We came up with this simple formula that has been giving us good results in production.

Mixing in the transparent shadows

- When computing lighting through refraction, we look at the previous vertex roughness and scale by $w(B_p)$
- When tracing transparent hits in shadows, we scale the transparency by $w(B_t)$
- The method is not perfect and the error becomes more visible with many bounces inside glass objects, so we scale down Total Internal Reflection with $w(B_p)$ too
    - Not a real loss since $w(B_p)$ is only low after rough bounces

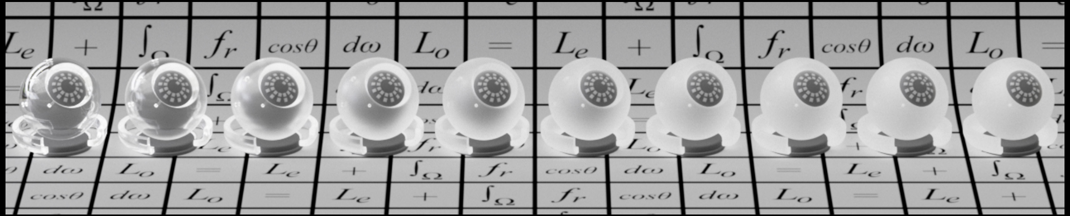
And it's very simple to apply. During path tracing or shadow tracing we look at the previous event and, comparing roughness, we compute the weights for the lighting. It is not perfect, and we have to be careful with internal bounces, but you'll see it is looking very decent for its simplicity.

Caustic Avoidance


Reference (BDPT)

This is BDPT, which we take as the ground-truth reference. A glass object with increasing roughness.

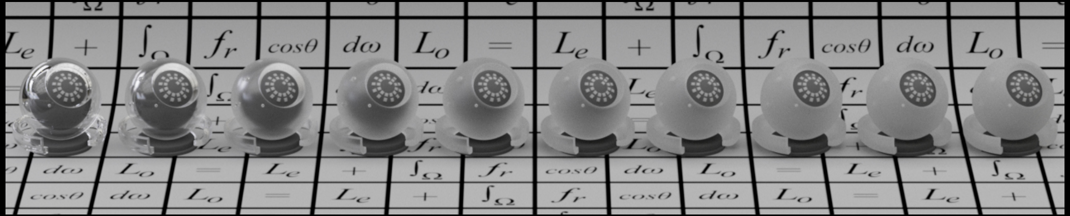
Caustic Avoidance



Transparent Shadows

This is path tracing combining roughness-clamped indirect with transparent shadows. Notice how the energy blows up because of the double lighting.

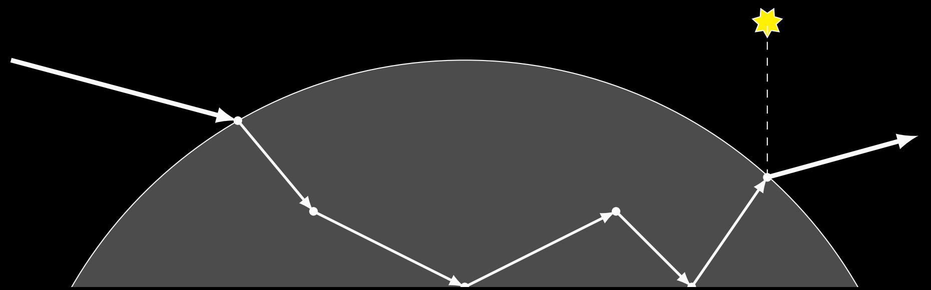
Caustic Avoidance



Transparent Shadow + Roughness Weighting

And this is path tracing with our special weighting for indirect and transparent shadows. It’s very close to the BDPT result but much cheaper.

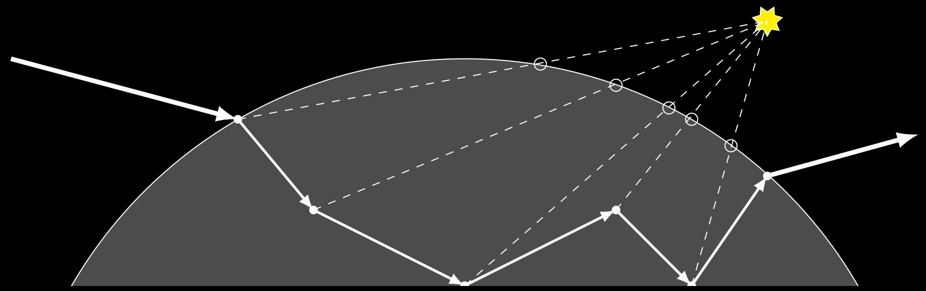
Brute-force subsurface scattering



For rough interfaces, we can afford a random walk looking for an exit and then connect to the light source. This is what our transparency/indirect weighting produces for these cases. But for a smooth interface, the result would be too noisy - even worse than surface caustics.

With a rough interface connections to the light at the exit point are trouble-free, so our weighting will favor this approach.

Brute-force subsurface scattering



If the interface is smooth, lighting at the exit point will be weighted down and replaced by lighting in the interior via transparent shadows. We still get indirect lighting from the environment after the exit point.

For the smooth case we resort to transparent shadows and weight down the lighting at the exit point, but the path will continue to get indirect lighting.

Coatings

Now I will go over how we put coatings or layers on top of our generic BSDFs.

Coatings

To create more complex layered looks, we support multiple coating types:

- Clearcoat
- Sheen
- Thin Film

For shader writers, coats are created in the OSL code on top of any closure combination. We could even put coats on top of other coats.

```
Ci = coat(closure1(...) + closure2(...) ...);
```

We have just a few handy coatings that can be combined over any BSDF. In fact, you could stack them indefinitely one over the other, and you can see here what it looks like in OSL code.

Coats and energy conservation

Assuming we know the energy response for a particular coating $E(\mu)$, if we want to combine it over another arbitrary BSDF, we can do as the already seen technique:

$$f_s' = (1 - E(\mu_o)) \cdot (1 - E(\mu_i)) \cdot f_s(\mu_o, \mu_i, \phi),$$

where f_s is the underlying BSDF passed by the OSL shader. But in this case E_{avg} is unknown and we cannot renormalize to avoid the energy loss due to $(1 - E(\mu_i))$, so to reduce this loss we instead scale f_s as

$$f_s' = \min\{(1 - E(\mu_o)), (1 - E(\mu_i))\} \cdot f_s(\mu_o, \mu_i, \phi),$$

which is still reciprocal and less darkening.

Like Chris explained in the dielectric/diffuse energy compensation, we try to compose these coats in a way that maintains the right albedo. But since the underlying BSDF can be anything - not just diffuse - we have no way to renormalize it. We can only scale it in a way to avoid energy gain while losing as little energy as possible.

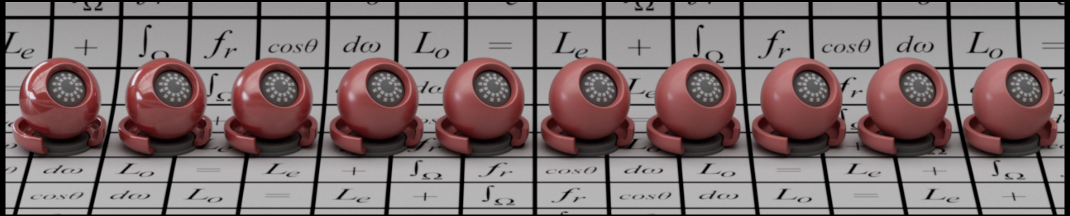
Clearcoat

L_e
$+$
$\int_\Omega$
f_r
$\cos\theta$
$d\omega$
L_o
$=$

Coating off

For instance, we can take this BSDF…

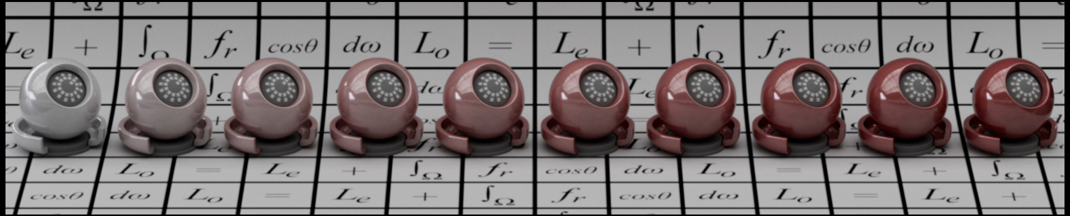
Clearcoat



Clearcoat is a dielectric layer that uses the same 3D table seen before to compute $E(\mu)$. It can simulate wet or varnish without incurring energy gain.

And add a clearcoat like a varnish layer without energy gain.

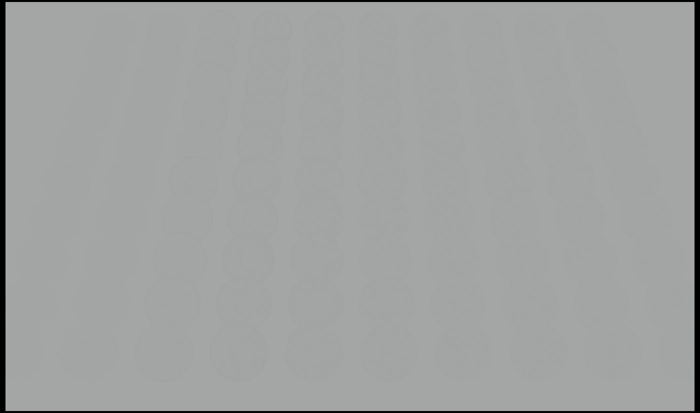
Clearcoat



Coating with varying thickness + tint. We can also define absorption on the clearcoat layer, so coloring appears as the artist increases the imaginary thickness. Grazing angles are more saturated as rays travel longer within the coat.

The clearcoat has an imaginary thickness and can produce nice coloring effects using absorption.

Clearcoat



Furnace test (Coating Off)

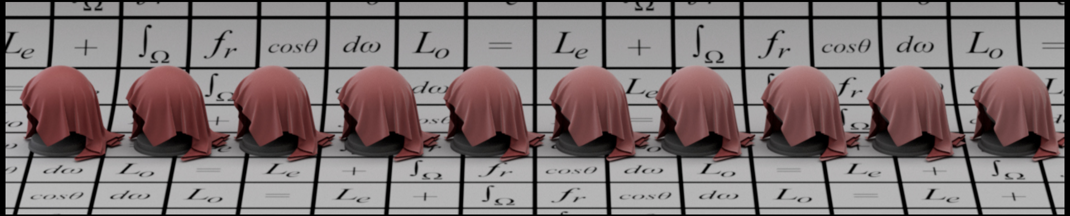
Finally, here are furnace tests, which are ideal with the coating off.

Clearcoat

Furnace test (Coating On)

And with the coating on, we have a little loss, but nothing that would get us any complaints from the artists.

Sheen

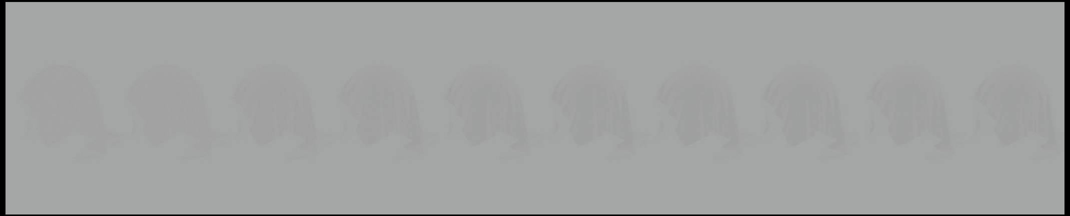


Sheen with increasing roughness. Our sheen model is a microfacet reflection just like the clearcoat, except the microfacet density is

$$D(m) = \frac{(2 + 1/r)\sin^{1/r}\theta}{2\pi}$$

We also have our own sheen model. We call it the “charlie” sheen. It is a microfacet model whose distribution is shown in the slide, and it’s inspired by normal-oriented microcylinders. We get to tweak the roughness to change its appearance, as you can see.

Sheen



Sheen energy conservation: we store the albedo of the sheen lobe in a 16×16 2D table indexed by roughness and incident angle, since we don't take Fresnel into account. With our $\min\{(1 - E(\mu_o)), (1 - E(\mu_i))\}$ scaling we avoid energy gain and suffer only subtle energy loss.

And it is composed just like the other coatings so we don't gain energy, as you can see in this furnace test.

Sheen Shadowing/Masking function

We compute this numerically given the chosen distribution of microfacets. Instead of using a lookup table, we found a good fit:

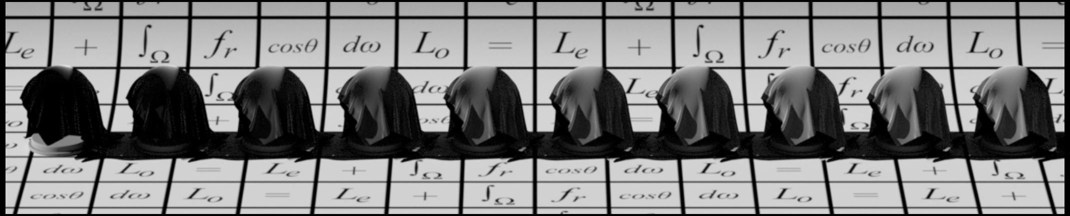
$$\Lambda(\theta) = \begin{cases} e^{L(\cos\theta)} & \text{if } \cos\theta < 0.5 \\ e^{2L(0.5) - L(1-\cos\theta)} & \text{otherwise,} \end{cases}$$

where $L(x) = a/(1 + bx^c) + dx + e$, and the a, b, c, d, e values were found for roughness $r = 0$ and $r = 1$, then we interpolate using $(1 - r)^2$.

r	a	b	c	d	e
0.0	25.3245	3.32435	0.16801	−1.27393	−4.85967
1.0	21.5473	3.82987	0.19823	−1.97760	−4.32054

Its shadowing/masking term cannot be derived analytically, so we did it numerically and then we found a very good fit. So in the end, the whole model is physically plausible, and if you are interested there is more information in the supplementary material.

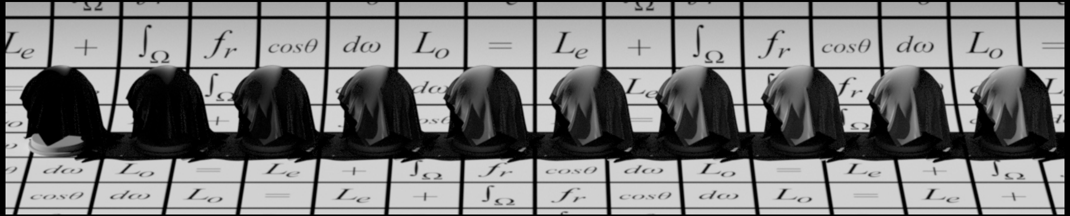
Sheen Terminator softening



Shadowing/masking term derived from microfacet distribution. Light terminator is abrupt.

As engineers we were very happy with the results, but the artists wanted an even softer terminator.

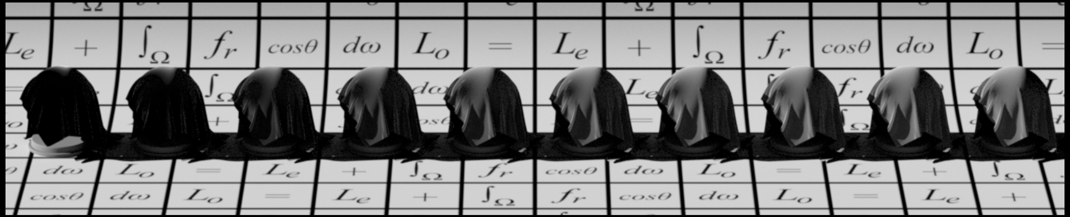
Sheen Terminator softening



Modified shadowing/masking term to smooth out the terminator. We modify it so $\Lambda'(\theta) = \Lambda(\theta)^{1+2(1-\cos\theta)^8}$. **Reciprocal** but too dark.

So we had to go and creatively modify the shadowing function to make it softer, but you can see that it kills the highlights on the edges.

Sheen Terminator softening



Modified shadowing term to smooth out the terminator only in the light direction. **Non-reciprocal** but visually pleasing.

We ended up applying this softening only for the shadowing part. This makes the model non-reciprocal, but hey, the customer is always right!

Putting it all together

Thank you Alex. I just want to wrap things up by giving you a sense of how we turned all this into shaders for our artists.

Putting it all together

We restructured shaders around parameter blending:

- Shaders track three "lobes" internally: Metal, Plastic, Glass
- Parameter blending within each lobe
- Avoids contamination in cases like dirt on glass
- Implementation detail: artists have total freedom to mix anything

Generalized our previous `PatternCreate` tool into `MaterialCombine`. Materials can be textured, combined, tweaked or coated within a graphical interface.

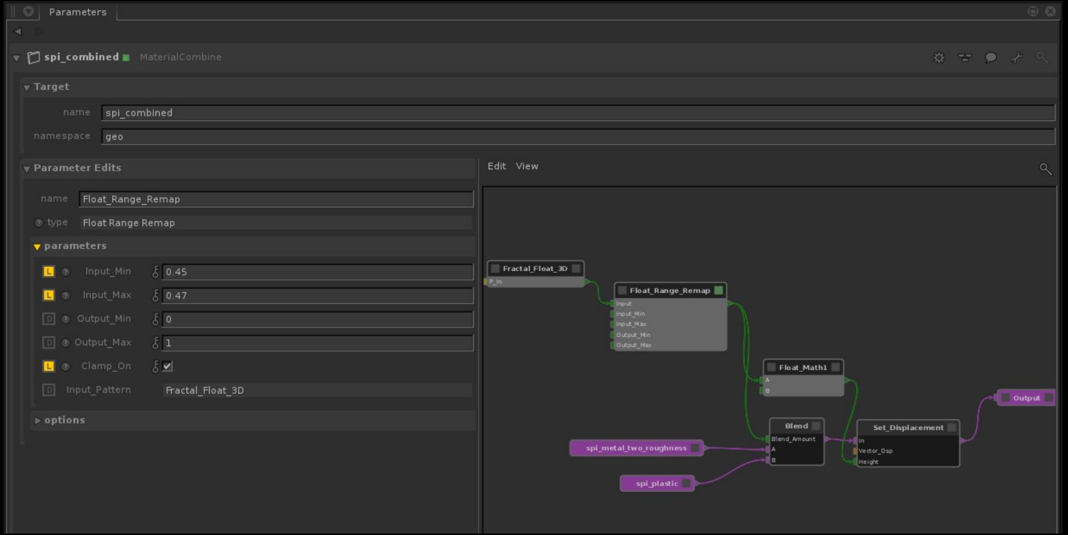
The system we deployed to production is based on the idea of parameter blending, again inspired by the Disney model. The shaders keep track of three basic lobes – for metal, plastic and glass – and do parameter blending within each lobe.

This is just an implementation detail, but it helps to avoid confusing parameters when doing things like dirt over glass. We wouldn't want the IOR set on the dirt to have any impact on the refraction of the glass.

Of course, as far as the artist is concerned, any material can be mixed with any other type.

The tool we called Pattern Create - for texturing specific parameters – evolved into Material Combine, where artists can blend entire materials together.

Material Combine



Parameters

spi_combined MaterialCombine

Target

name: spi_combined

namespace: geo

Parameter Edits

name: Float_Range_Remap

type: Float Range Remap

parameters

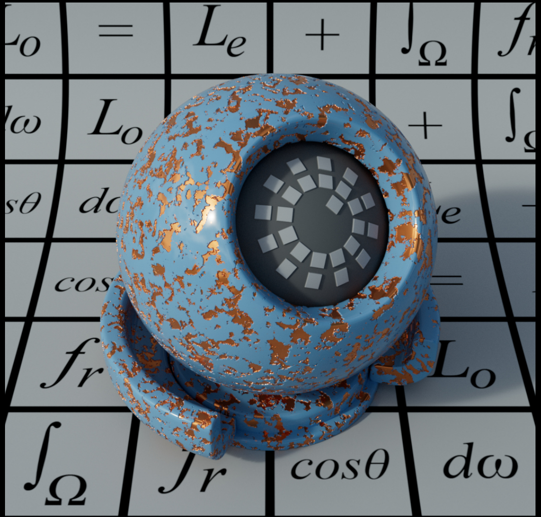
Input_Min	0.45
Input_Max	0.47
Output_Min	0
Output_Max	1
Clamp_On	✓
Input_Pattern	Fractal_Float_3D

options

This is a screen capture of a small material graph that's blending between two existing materials. Of course those materials can themselves be graphs.

This is a very simple example that just uses a fractal node to blend between two materials and also drive displacement.

Material Combine


$L_o = L_e + \int_\Omega f_r$
$d\omega$ L_o $+$
$\cos\theta$ $d\omega$
$\cos$ $=$
f_r L_o
$\int_\Omega f_r \cos\theta\, d\omega$

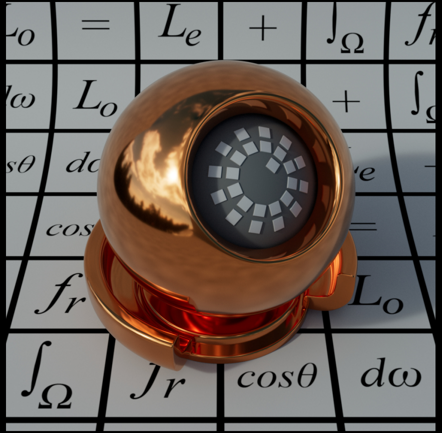
Easy to create new materials out of existing ones.
Leverages Katana's deferred scene processing to allow late edits to the material graph.

This is the result of that graph.

The tool is embedded inside Katana, which is our lighting and lookdev tool. This means we can do these kind of edits to material graphs at very late stages of the pipeline.

For example, an artist can very easily add paint drops driven by FX onto any material without needing to know how the base material was constructed.

Double Specular



Rough = 0.1

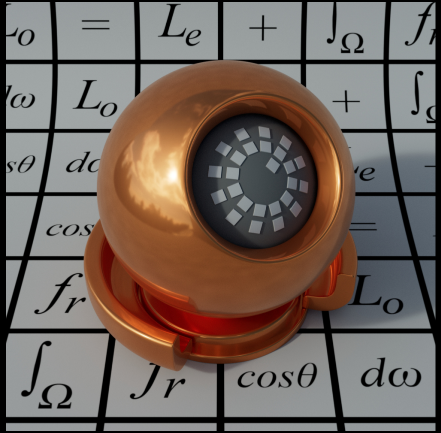
GGX does not match all measured data.

I want to mention another small but important detail of our shaders.

It's well known that the GGX distribution doesn't fit well to all measured data. Several researchers have proposed extending the distribution itself to give control over its tail, for instance the GTR or STD distributions.

Unfortunately these make the masking and shadowing functions much more complicated and exact importance sampling more difficult.

Double Specular



Rough = 0.1 (50%), 0.5 (50%)

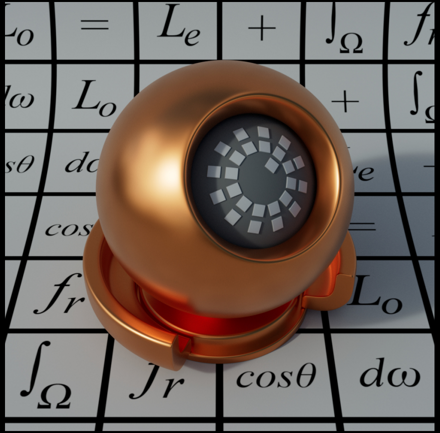
GGX does not match all measured data.

We expose two roughness values to help achieve the “hazy” look of a longer-tailed distribution, at low cost.

Instead we chose a much simpler solution, which is to just expose two roughness values and a blend amount.

This is easy to implement and importance sample, and works seamlessly with our energy compensation scheme because it just involves a linear combination of energy-preserving BRDFs.

Double Specular



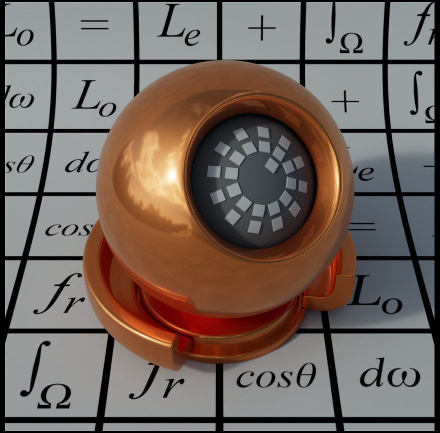
Rough = 0.3

GGX does not match all measured data.

We expose two roughness values to help achieve the “hazy” look of a longer-tailed distribution, at low cost.

Here you can see how a roughness of 0.3 is very different to a 50/50 blend of roughness 0.1 and 0.5.

Double Specular



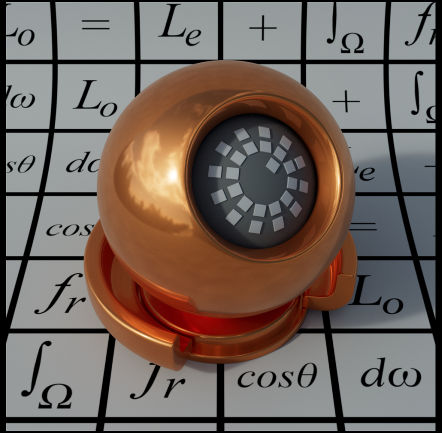
Rough = 0.1 (50%), 0.5 (50%)

GGX does not match all measured data.

We expose two roughness values to help achieve the “hazy” look of a longer-tailed distribution, at low cost.

Here you can see how a roughness of 0.3 is very different to a 50/50 blend of roughness 0.1 and 0.5.

Double Specular



Rough = 0.1 (50%), 0.5 (50%)

GGX does not match all measured data.

We expose two roughness values to help achieve the "hazy" look of a longer-tailed distribution, at low cost.

Vangorp et al., "The perception of hazy gloss", 2017

In fact, just this year Peter Vangorp and colleagues did a perceptual study that suggests there is some merit to this approach.

Future Work

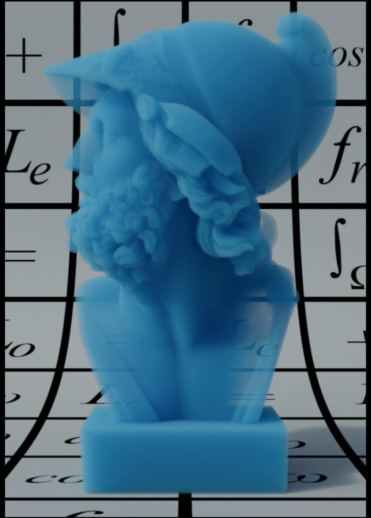
I’ll now briefly talk about some of the work we still would like to do going forward.

Future Work - Diffuse

Is Lambertian diffuse a good proxy for SSS?

Lambertian diffuse is assumed to be a reasonable stand-in for SSS. But is it really?

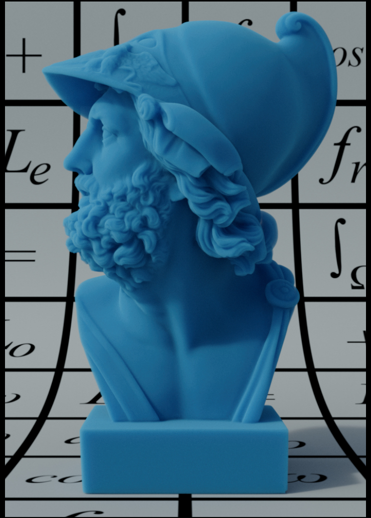
Future Work - Diffuse



Mean Free Path = 1 / 4

Here I am rendering the mesh as a volume with shrinking mean free path.

Future Work - Diffuse



Mean Free Path = 1 / 16

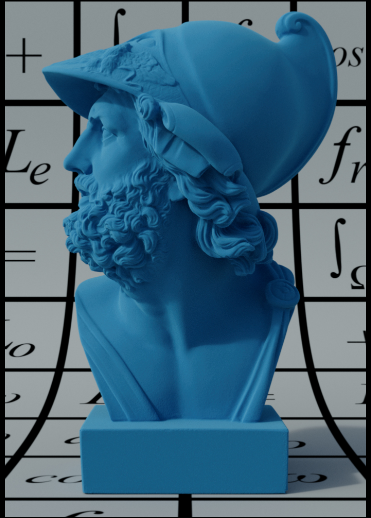
Here I am rendering the mesh as a volume with shrinking mean free path.

Future Work - Diffuse

Mean Free Path = 1 / 64

Once the mean free path is small enough, the result looks like it can be expressed as a BRDF.

Future Work - Diffuse



Mean Free Path = 1 / 256

Once the mean free path is small enough, the result looks like it can be expressed as a BRDF.

Future Work - Diffuse

+
L_e
f_r
$\int_\Omega$
=
os

Chandrasekhar BRDF

Matches volume scattering exactly

In fact there is a closed-form solution to this exact problem that has been derived a long time ago. Here I am showing that this BRDF matches the volume simulation exactly.

Future Work - Diffuse

Diffuse BRDF is very different!

This has already been explored by previous work. Result should also depend on the microfacet roughness and IOR.

Our diffuse term accounts for roughness and IOR but does not match the limit behavior of SSS.

On the other hand, it doesn't match the Lambertian result at all!

Some other researchers have explored this, the Disney BSDF model in particular tries to get a seamless match between subsurface and diffuse, but we feel there is more work to be done in this area.

Future Work - Diffuse

Other interesting recent work:

- Meneveaux et al., “Rendering Rough Opaque Materials with Interfaced Lambertian Microfacets”, 2017
- Hammon Jr., *PBR Diffuse Lighting for GGX+Smith Microsurfaces*, 2017
- Holzschuch et al., “A Two-Scale Microfacet Reflectance Model Combining Reflection and Diffraction”, 2017

I am just pointing out a few recent papers that have proposed improved diffuse models.

However, none of them are trying to exactly match volume scattering results.

Future Work - Energy Preservation

Was our focus on energy preservation worthwhile?

- Helpful for glass and metals. Works well with caustic avoidance scheme and roughness clamping.
- Better understand how our construction differs from ground-truth and measured data.
- Analytical formulas would be preferable to tabulation for real-time applications.

Finally, was all this focus on energy preservation worthwhile?

We feel that it's a very clear win for metals and glass. It works really well with the caustic avoidance scheme because making a material rougher doesn't lead to less energy being transmitted.

But we'd like to do more comparisons to measured data.

And of course for real-time applications, having formulas would be better than tabulating. We are very close for some of the 2D tables, but the 3D cases seem more challenging.

Production Use (mixed with older shaders)

Alice Through the Looking Glass
It's Time For a Little Madness May 27
From the studio that delivered The Lego Movie
Storks
Find your flock
september 23
Will Smith Jared Leto Margot Robbie Joel Kinnaman Viola Davis
August 5
Suicide Squad
Worst Heroes Ever
They were never alone
Smurfs
The Lost Village
In Cinemas 2017

I've kept these slides intentionally free of production examples so we could share all the slides online. But these are the productions that first used some of this technology.

For these shows, we just added new lobes to our old shaders. And we also introduced a new glass shader that got used a lot…

Production Use (all new shaders)

…and these are the first few shows that used our new parameter blended shaders with volumetric subsurface.

Of course everything in production going forward is using these brand new shaders as well.

Acknowledgements

- Shading Team
    - Lee Kerley
    - Ole Gulbrandsen
- Lookdev Artists
    - Brian Kloc
    - Brian Steiner
    - Craig Feifarek
    - David Conlon
    - Kurt Judson
    - Maung Maung Hla Win
    - and many more …

Finally I just want to acknowledge the hard work of our Shading team, in particular Lee Kerley and Ole Gulbrandsen who were instrumental in working with us to assemble all of this into an artist-friendly package.

I also want to thank all the lookdev artists who participated in the early discussions and testing of these models.

SONY PICTURES imageworks 25TH ANNIVERSARY

Thank You! Questions?

30 JULY – 3 AUGUST Los Angeles SIGGRAPH2017

To hear more about our renderer please see:

- Talk: "Importance Sampling of Many Lights With Adaptive Tree Splitting" - Monday 3:45PM, Room 402AB
- Course: "Path Tracing in Production - Part 2: Making Movies", Wednesday 2:00PM, Room 408AB

That concludes our talk.

Thank you very for your attention, and Alex and I will be happy to take any questions.

References I

- Raphael Aronson. “Boundary conditions for diffusion of light”. In: *Journal of the Optical Society of America A* (1995).
- Eugene d’Eon. *A Hitchhiker’s Guide to Multiple Scattering*. `http://www.eugenedeon.com/hitchhikers`. 2016.
- Ole Gulbrandsen. “Artist Friendly Metallic Fresnel”. In: *Journal of Computer Graphics Techniques (JCGT)* (2014).
- Earl Hammon Jr. *PBR Diffuse Lighting for GGX+Smith Microsurfaces*. `http://gdcvault.com/play/1024478/PBR-Diffuse-Lighting-for-GGX`. 2017.

References II

- Eric Heitz. *A Simpler and Exact Sampling Routine for the GGX Distribution of Visible Normals.* Research Report. Unity Technologies, Apr. 2017. URL: `https://hal.archives-ouvertes.fr/hal-01509746`.
- Eric Heitz et al. “Multiple-scattering Microfacet BSDFs with the Smith Model”. In: *ACM Transactions on Graphics* (2016).
- Nicolas Holzschuch and Romain Pacanowski. “A Two-Scale Microfacet Reflectance Model Combining Reflection and Diffraction”. In: *ACM Transactions on Graphics* (2017).

References III

- Wenzel Jakob et al. “A Comprehensive Framework for Rendering Layered Materials”. In: *ACM Transactions on Graphics* (2014).
- Csaba Kelemen and Laszlo Szirmay-Kalos. “A Microfacet Based Coupled Specular-Matte BRDF Model with Importance Sampling”. In: *Eurographics - Short Presentations* (2001).
- Daniel Meneveaux et al. “Rendering Rough Opaque Materials with Interfaced Lambertian Microfacets”. In: *IEEE Transaction on Visualization and Computer Graphics* (2017).
- Hendrik van de Hulst. *Multiple Light Scattering*. Academic Press, 1980.

References IV

Peter Vangorp, Pascal Barla, and Roland W. Fleming. “The perception of hazy gloss”. In: *Journal of Vision* (2017).