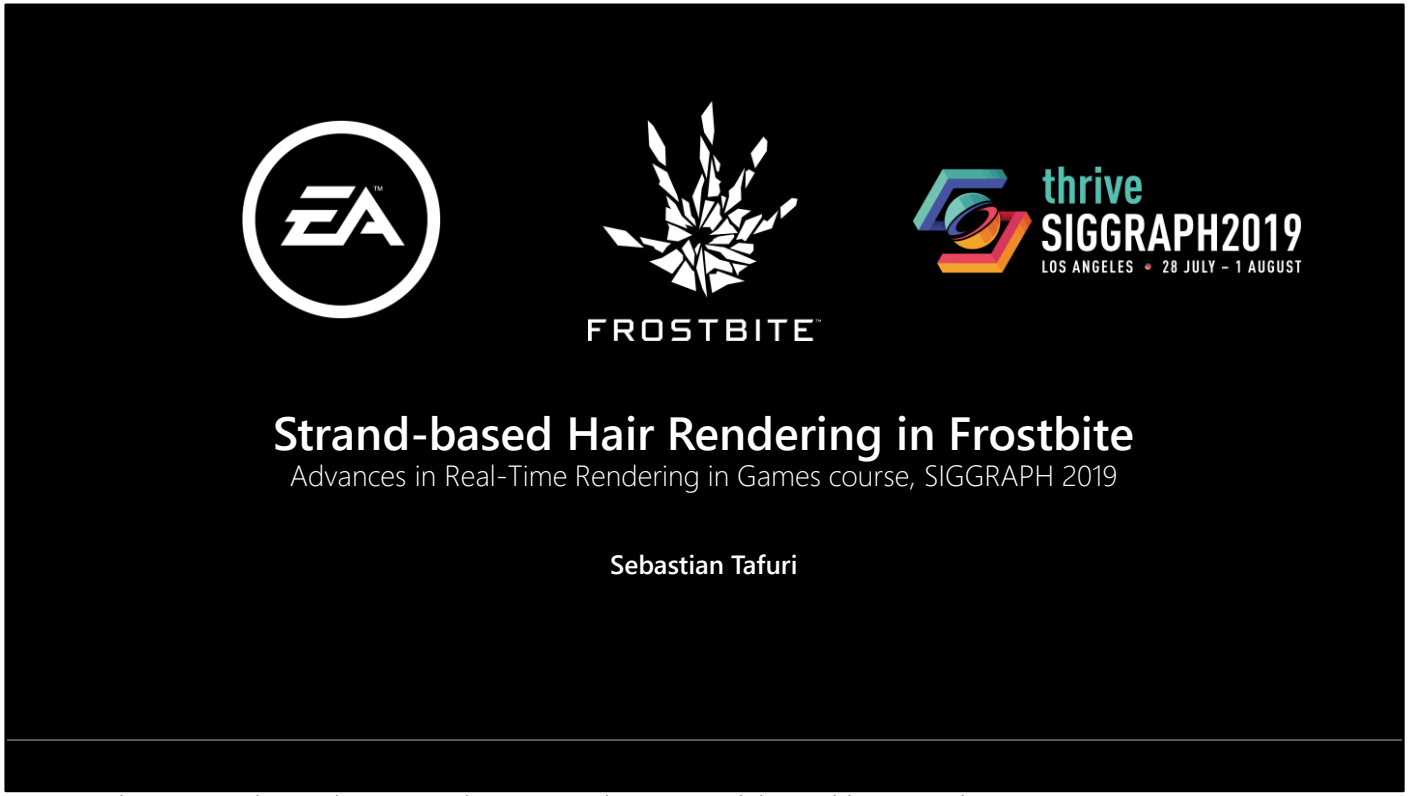
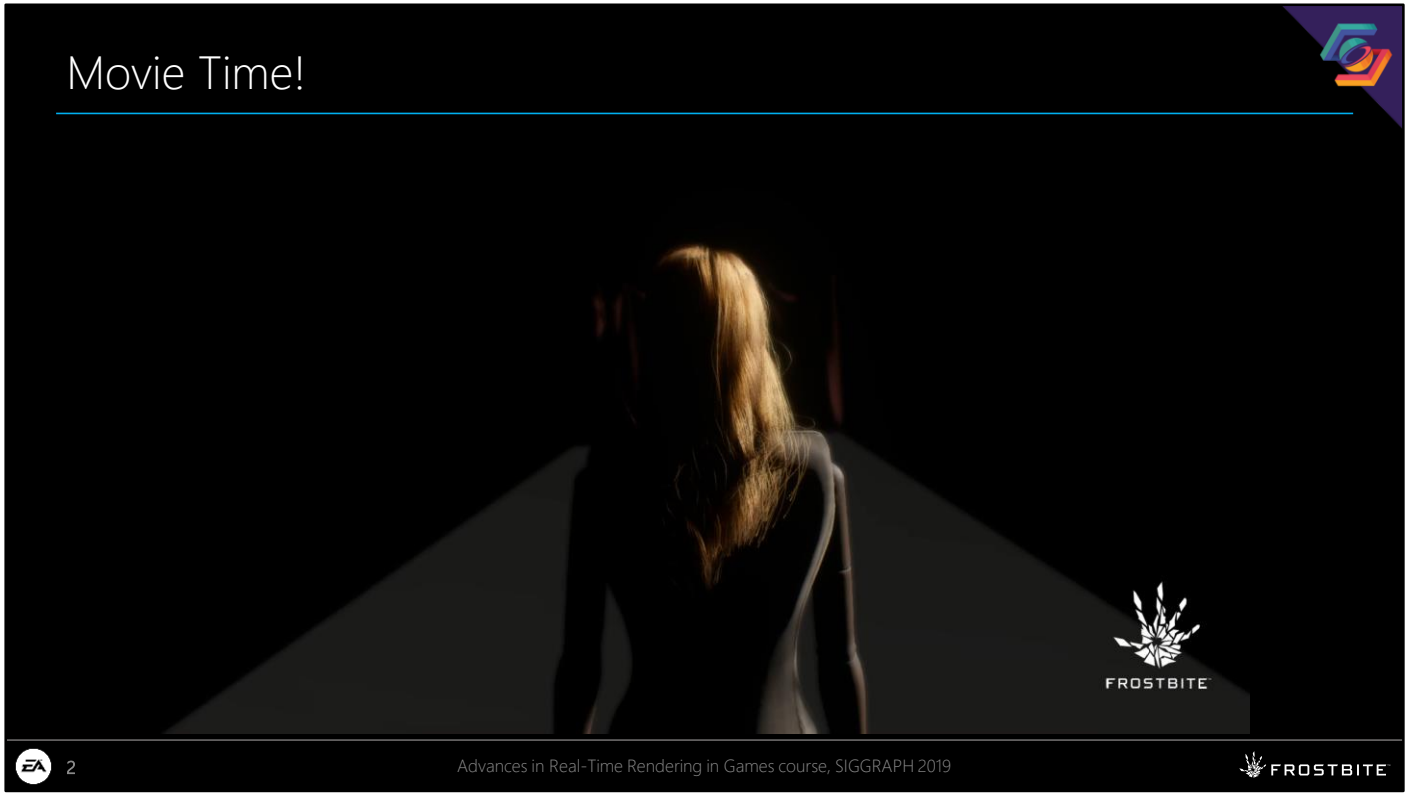


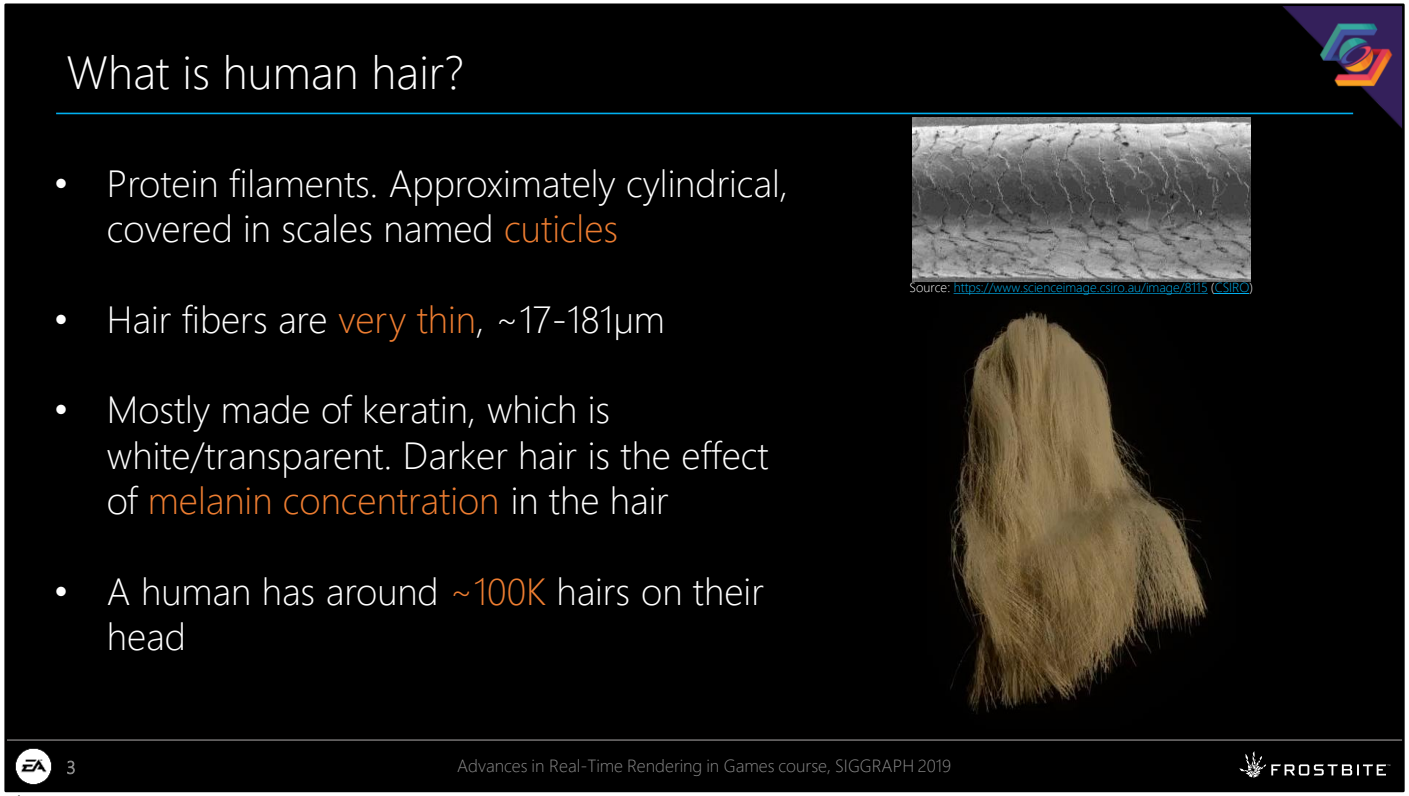
Today I will be presenting the research made at Frostbite regarding strand-based hair rendering.

This is a project aimed at seeing what is possible regarding hair rendering and simulation on current or near-future hardware for use in games.

Keep in mind that this work is all still very much WIP and nothing presented is final.



But first, movie time!
This video was posted on the Frostbite developer blog earlier this year and it shows our hair technology on this slightly creepy manikin.



So, what is human hair?

They are protein filaments, approximately cylindrical and covered in small scales named cuticles, which you can see in the image at the top.

They are very thin, approximately 17-180 micrometers.

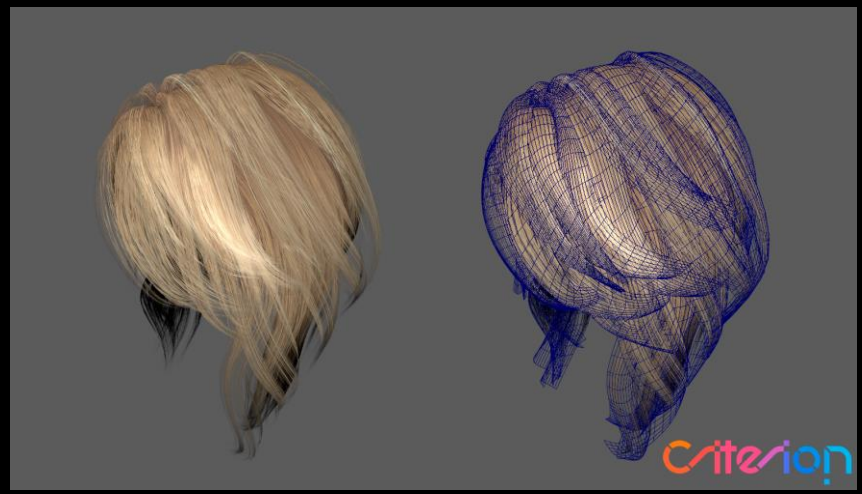
The are mostly made of keratin with melanin pigments which are what controls the hair color and darkness.
The animation to the right shows the effect of varying the melanin concentration.

A human head has around 100’000 hair strands on their head.

Motivation

Games usually resort to hair cards or shells

- ☹ Costly authoring
- ☹ Limited hairstyles
- ☹ Realistic simulation
- ☹ Simplified shading



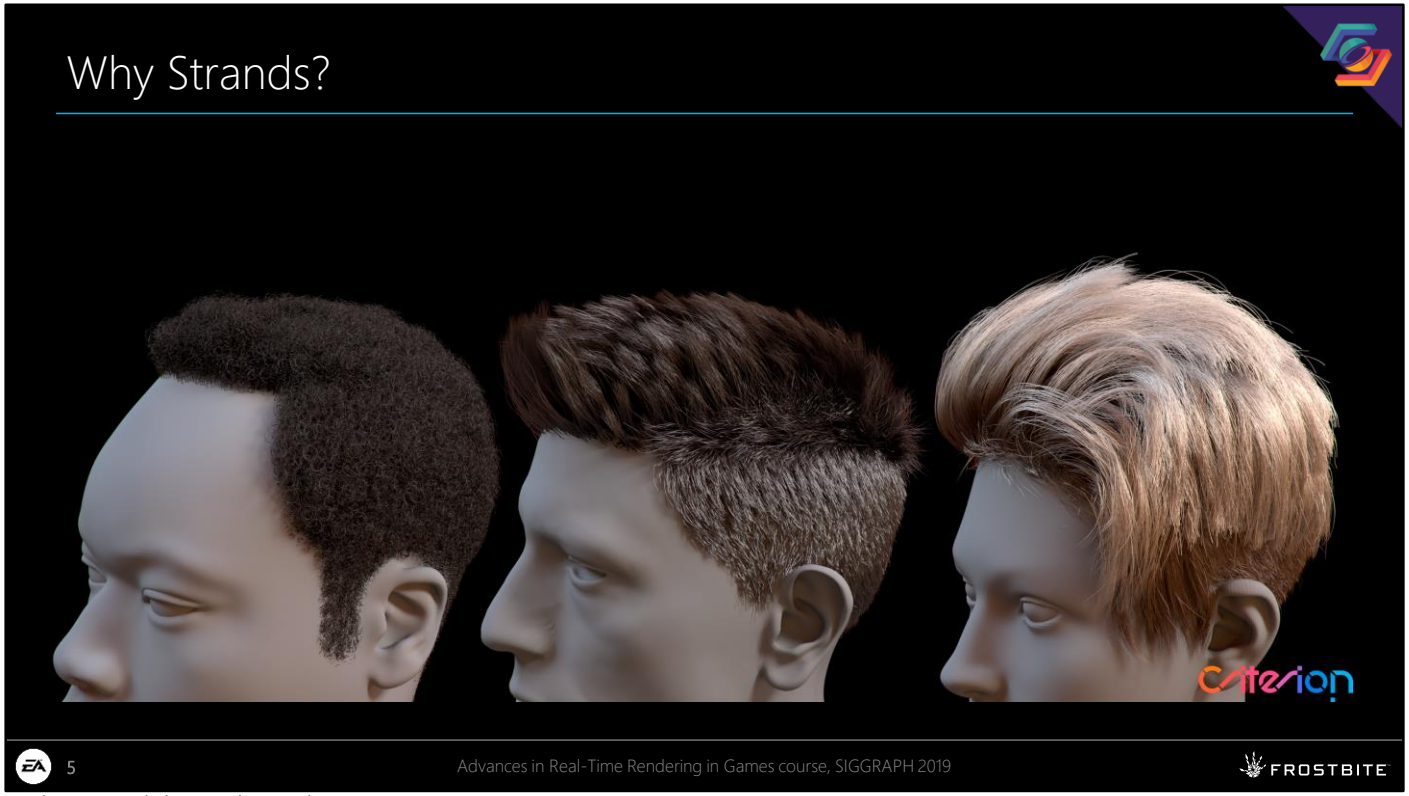
In games it is currently very common to model human hair using hair cards or shells where multiple hair fibers are clumped together as a 2d surface strips.
These cards are generated using a combination of tooling and manual authoring by artists.

This work can be very time consuming and it can be very difficult to get right.

If done well these can look good, but they do not work equally well for all hairstyles and therefore limit possible visual variety.

And since it is a very coarse approximation of hair there are also limits on how realistic any physics simulation or rendering can be.

It is also common to use very simplified shading models which do not capture the full complexity of lighting in human hair, this can be especially noticeable when shading lightly colored hair.



So, what is better with strand-based rendering?

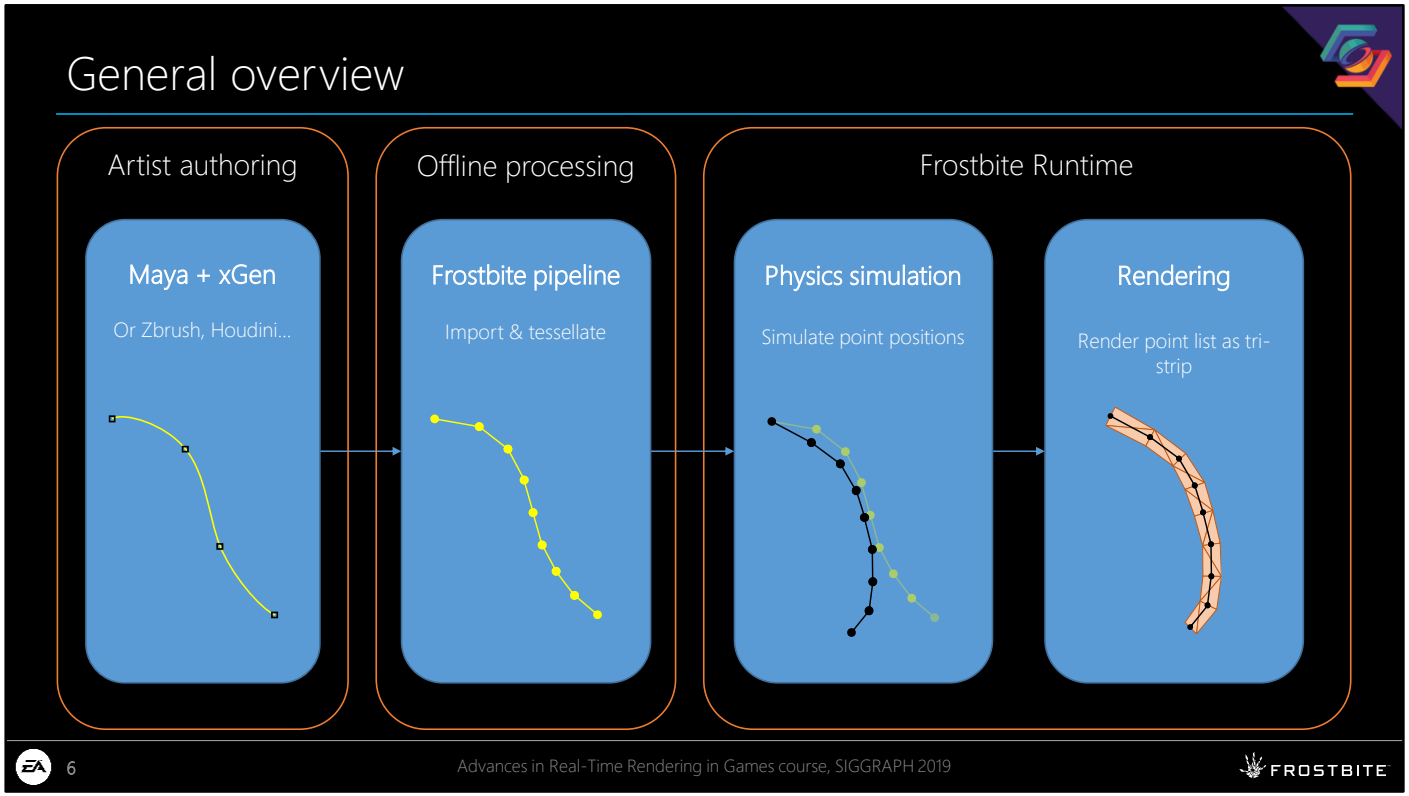
Strand based rendering, where hair fibers are modelled as individual strands, or curves, is the current state of the art when rendering hair offline. And it can look something like this image generated using Arnold.

Modelling hair as strands has a lot of benefits for physics and rendering since it is better model of real human hair.

It also requires less authoring time than hair cards since you do no longer need to create them. And since the process of creating hair cards usually also involves
creating hair strands for projecting onto the hair cards, you basically save time on that whole step.

With hair-strands it is also possible to do more granular culling for both physics and simulation, and easier to generate automatic LODs via decimation.

The goal of this project is to get as close as possible to movie quality hair, using-hair strands, while still achieving real time frame rates.



Here is an overview of the whole system

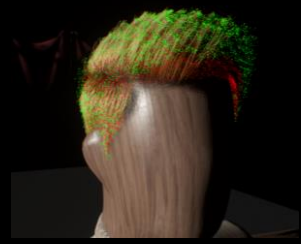
An artist creates a hairstyle in some tool and exports it as a collection of NURBs.

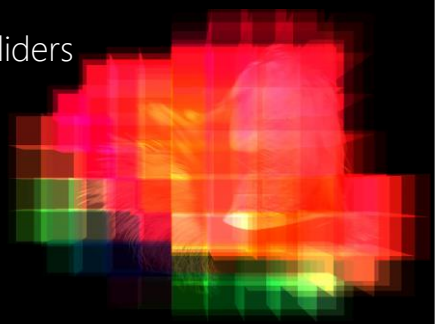
These are then processed and converted into series of points and constraints.

During execution, these points are then loaded and simulated before being rendered as triangle strips.

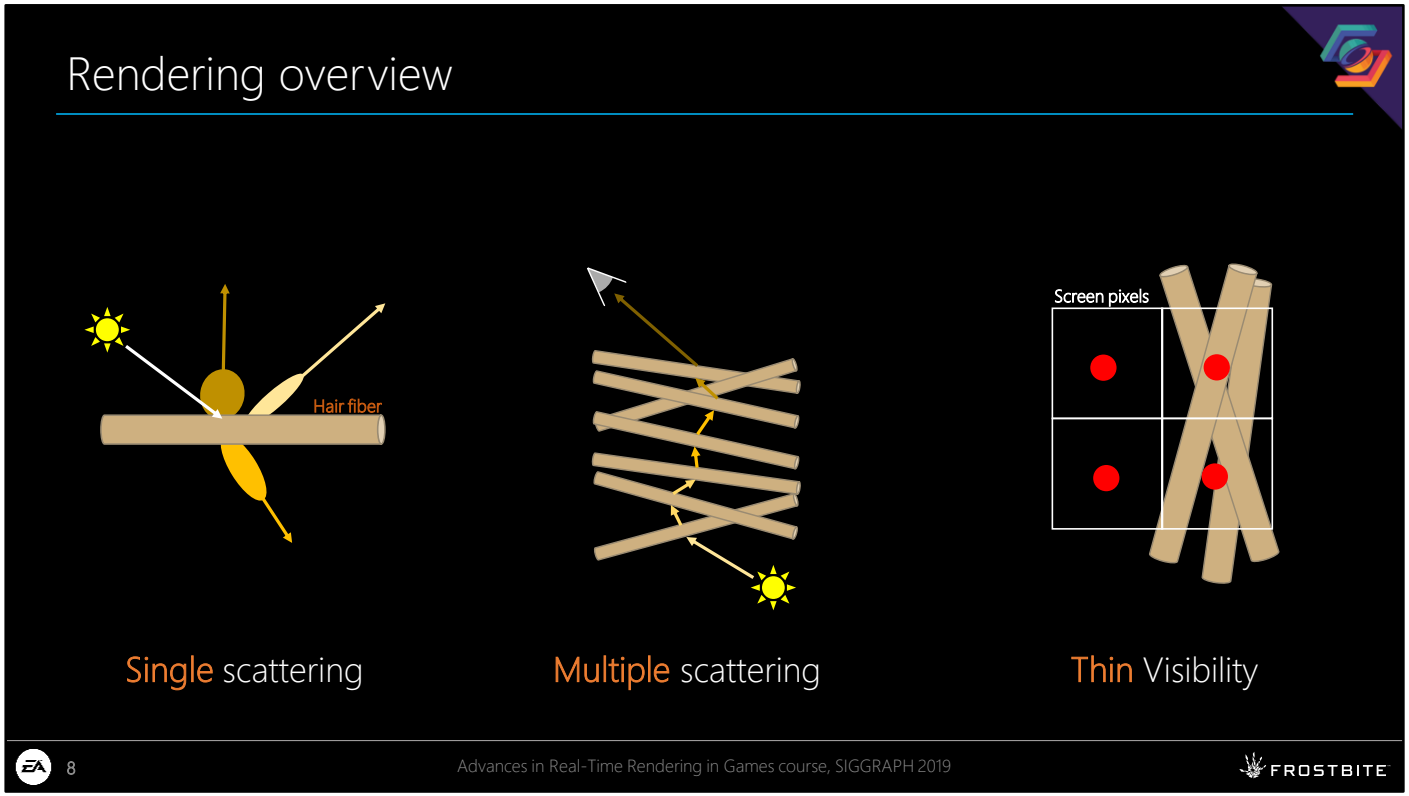
Hair simulation

- Strands are simulated as a series of points with constraints
- Combination of *Eulerian* and *Lagrangian* simulations
- Strand-strand interactions calculated using a grid
  - Friction, volume preservation, aerodynamics
- Integration is done on each point individually
- Then points are solved by iterating over constraints and colliders
- More details in future presentations and the frostbite blog





A physically plausible simulation is very important to make rendered hair look right.
The hair strands are simulated as a series of points with constraints
The simulation is a combination of Eulerian, so simulation on a grid, and Lagrangian simulation on the points
The strand interactions, like friction, volume preservation and aerodynamics, are calculated using a grid, to the bottom right you can see a color-coded visualization of the friction.
The actual integration is done on each point individually and solving by iterating over all constraints and colliders
I will not go into more detail here, but for those interested we will share more details in future presentations and on the frostbite blog



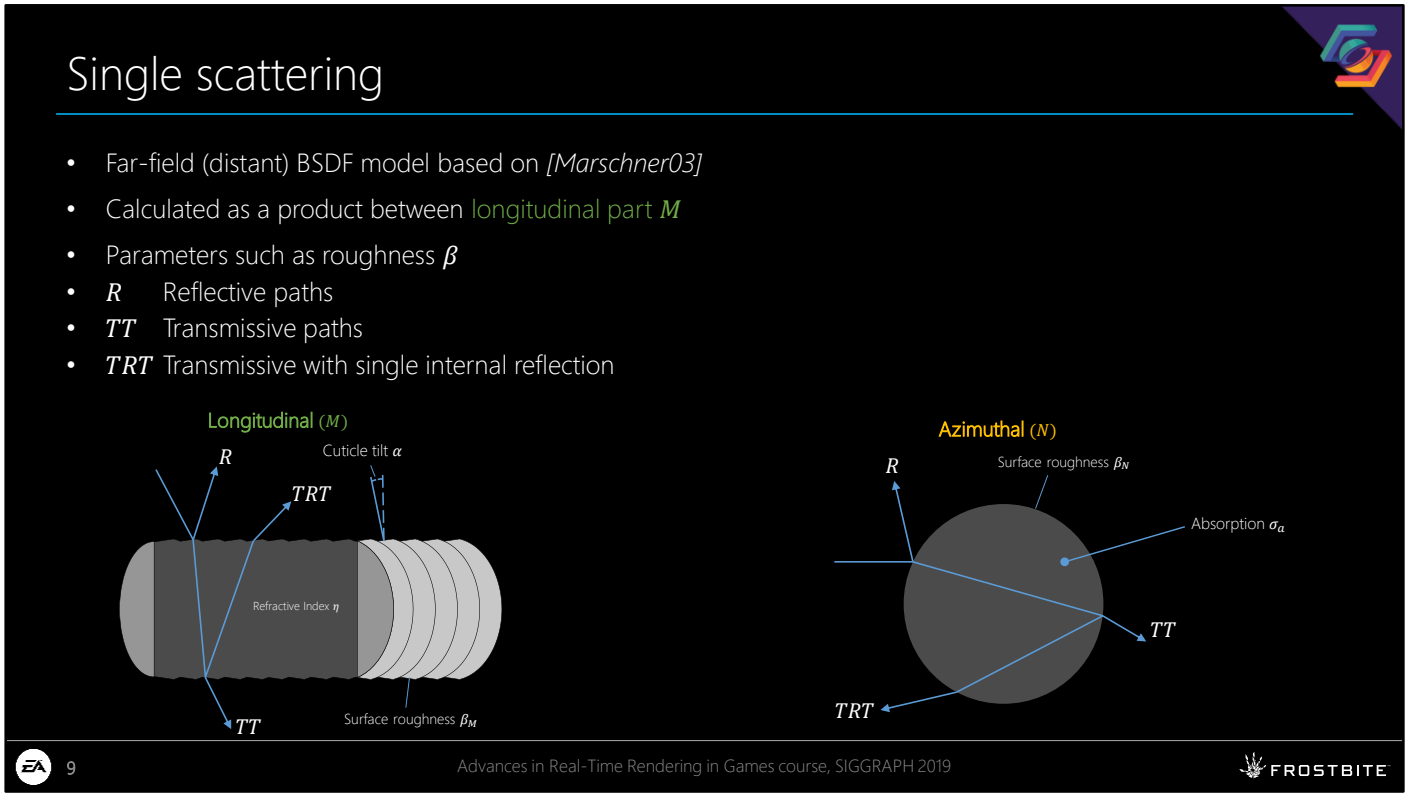
Rendering of hair can be split up into three main problems we need to solve, which are all important to get plausible appearance.

We have single scattering, which is the problem of how light interacts with a single fiber. This is actually a very complex problem and while there has been a lot of research done in this area, much of is not directly suitable for real time rendering.

After single scattering we have multiple scattering, which is the problem of how light interacts with multiple hair fibers.

The final problem relates to the rendering and how to do it in a way that is performant and does not introduce a lot of aliasing due the thin and numerous nature of hair.

I will go through these problems in order



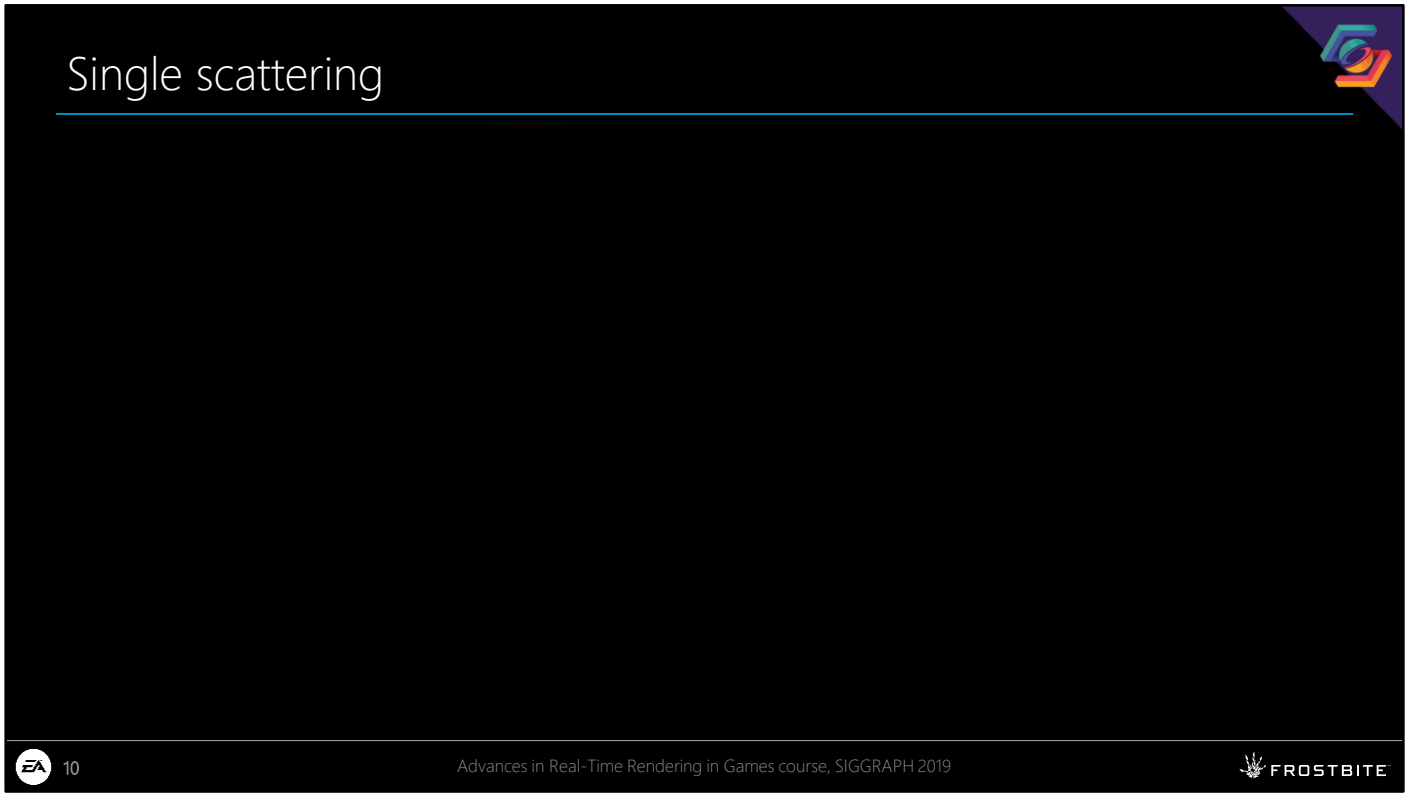
For single scattering we use a BSDF based on an original model for hair created by Marschner et.al. in 2003. The model is a far-field model which means that it is meant to model the visual properties of hair when seen from a distance, not for single fiber closeups.
This model was later improved for path-tracing by Disney and Weta Digital and approximated for real time use by Karis, parts of which work we also incorporate.

The model is split up as a product between a longitudinal part M and an azimuthal part N

It contains parameters such as surface roughness, absorption and cuticle tilt angle.

Different types of light paths are evaluated separately and added together.
These are R , which are reflective paths
TT which is transmission through the fiber,
and TRT which is transmission with a single internal reflection.
These paths are also enumerated as p 0, p 1 and p 2
One can of course add longer paths as well, but we only include these three paths for now.

Single scattering



Now I will try to visualize the contribution of the different light paths
First, we have the R path which contain direct reflections of light
We then have the TT path which mostly contributes when we have light from behind
And then we have the TRT path which shows more saturated colors due to absorption.
With all of them together it looks like this

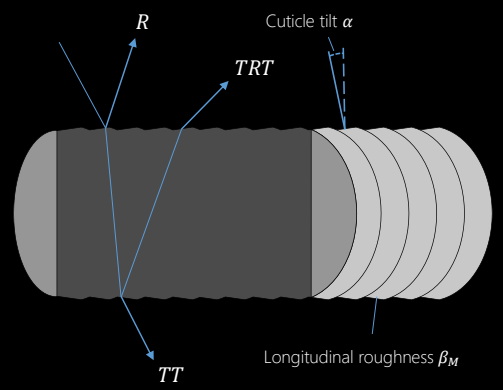
Here is a comparison showing the effect of changing the hair absorption with increasing values from the left to the right.

And here is a comparison showing the effect of increasing surface roughness value instead.

As you can see the way in which these parameters affect the perceived hair color is not trivial, and we need to capture this complexity to get realisticly looking hair.

Longitudinal Scattering M_p

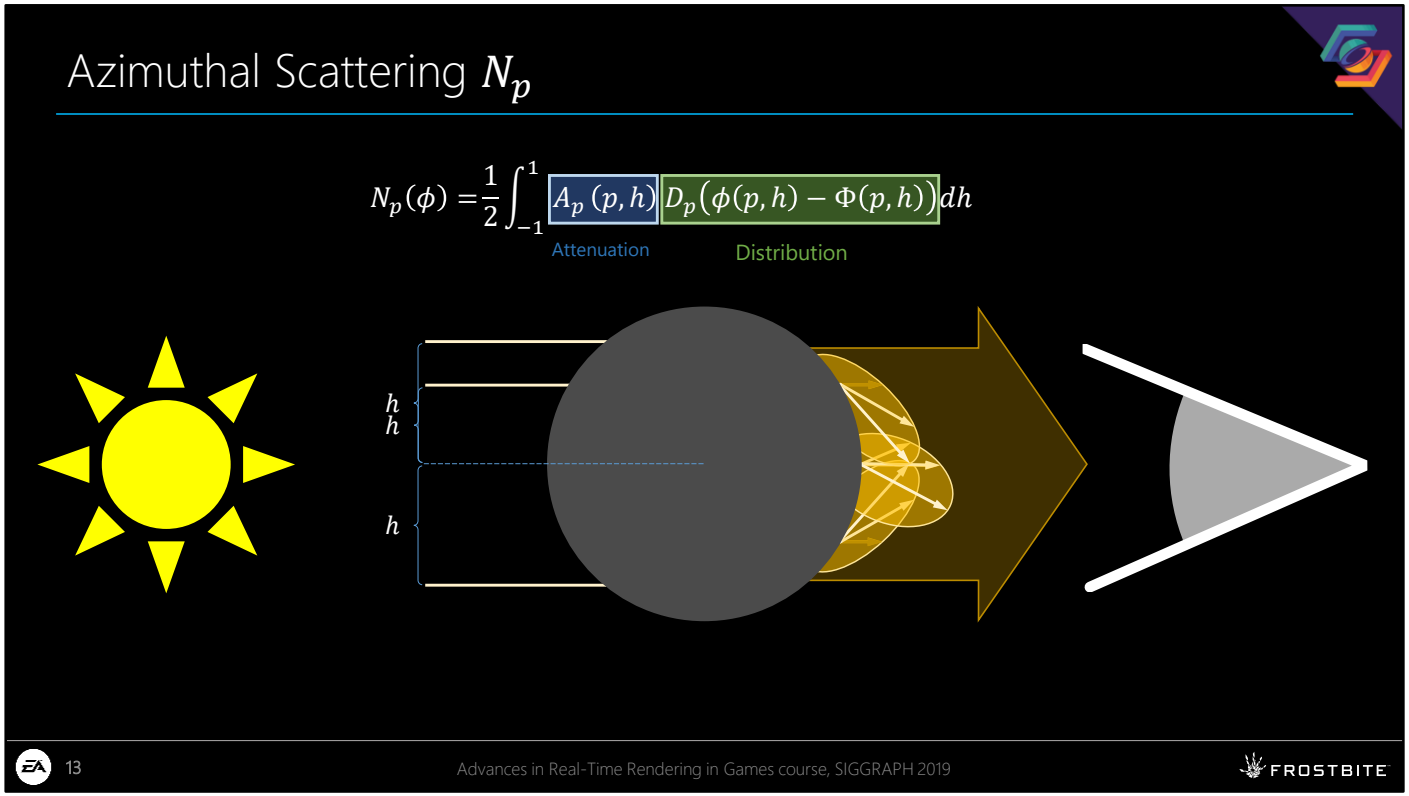
- Modelled using a single gaussian lobe per path
  - $M_R(\theta_h) = g(\beta_M; \theta_h - \alpha)$
  - $M_{TT}(\theta_h) = g\left(\frac{\beta_M}{2}; \theta_h + \frac{\alpha}{2}\right)$
  - $M_{TRT}(\theta_h) = g\left(2\beta_M; \theta_h + \frac{3\alpha}{2}\right)$
- θ_h is the longitudinal half vector



Now on the actual model, specifically the longitudinal scattering

For the longitudinal scattering M, each path type is modelled using a single gaussian lobe with the parameters depending on the longitudinal roughness and the cuticle tilt angle.

The motivations for these equations are explained in more detail in the Marschner original paper



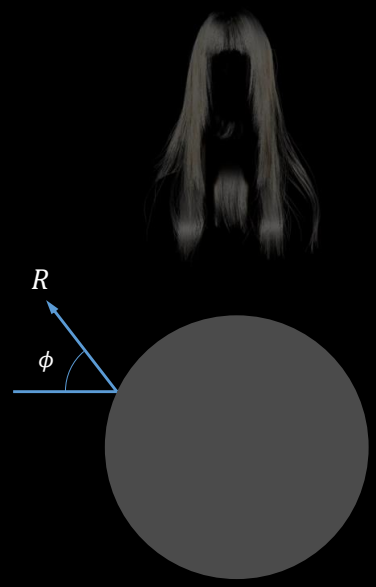
The azimuthal scattering is split up into an Attenuation factor A, which accounts for Fresnel reflection and absorption in the fiber, and a distribution D which is the lobe modelling how the light scatters when it is reflected or exits the fiber.

To properly simulate surface roughness for transmitted paths, this product is then integrated over the width the hair strand to get the total contribution in a specific outgoing direction.

Numerically integrating this equation for every shading point and light path is of course not feasible to do in real-time. So we have to approximate.

Azimuthal Scattering N_R

- Distribution same as *[Karis16]*
  - $D_R(\phi) = \frac{1}{4}\cos\frac{\phi}{2}$
- Attenuation is Schlick's Fresnel
  - $A_R(\phi) = F\left(\eta, \sqrt{\frac{1}{2}(1 + \omega_i \cdot \omega_r)}\right)$
  - ω_i and ω_r are the 3D incident and reflected directions



The reflective path R, is still pretty simple though, and we use the simplifications from Karis presentation.

The distribution looks like this and the attenuation is just the Fresnel term, keep in mind though that we use the original 3D vectors here

Azimuthal scattering N_p

- Previous approximations by Karis works well for smooth hair fibers

$\beta_M = 0.3$
$\beta_N = 0.3$

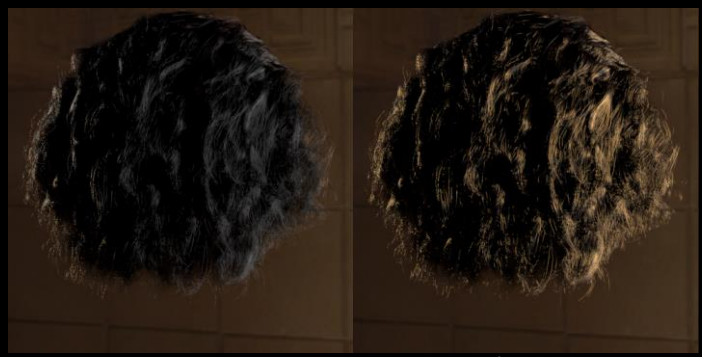
[Karis16] Reference [Chiang16]

In Karis presentation he also proposed some approximations for the TT and TRT paths. The approximations work pretty well with smooth hair fibers.

Azimuthal scattering N_p

- But they do not work as well with higher roughness values

$\beta_M = 0.3$
$\beta_N = 0.9$



[Karis16]

Reference
[Disney16]

But the approximations do not work as well when the roughness of the hair fibers is increased, and the appearance is getting more dominated by internal absorption.

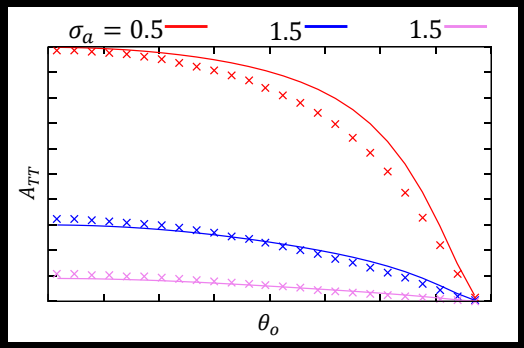
We wanted to improve on these approximations

Lets start with the attenuation term A

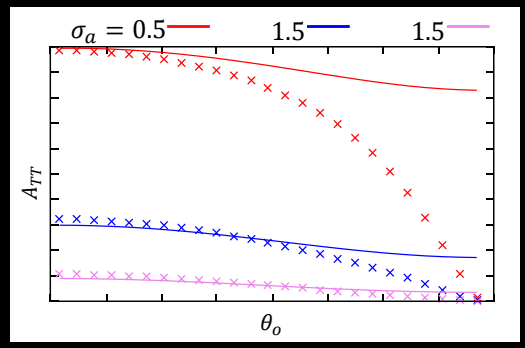
Approximating A_{TT}

- Transmission straight through fiber is dominant contributor
- Can approximate A_{TT} at $h_{TT} = 0$

Frostbite



[Karis16]



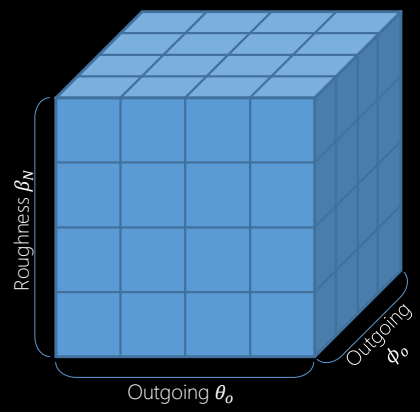
By analyzing the attenuation term for transmission one can note that the dominant contribution comes from the light that is transmitted straight through the middle of the fiber. So when the h parameter is 0.
If we compare the attenuation value only calculated at h zero, and the full integral, we can see that this is actually a pretty ok approximation.

Here is a plot showing this approximation for three different absorption values with the reference integral drawn as crosses and the approximation drawn as a solid line.

And here is a plot showing how the approximation Karis used stacks up and one can see that it has some problems with grazing angles, especially with more translucent, brighter hair.

Approximating D_{TT}

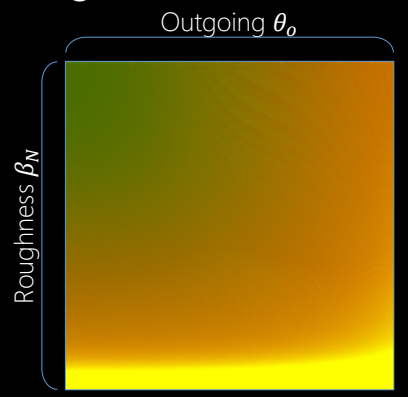
- $\int D_p$ approximated using a LUT
- Depends on roughness β_N, outgoing angles ϕ_o and θ_o



For the distribution we use a LUT.
The distribution depends on roughness, azimuthal outgoing angle and the longitudinal outgoing angle which means that the LUT becomes be three dimensional.

Approximating D_{TT}

- We reduce to 2D by fitting a gaussian to each ϕ_o slice
  - Gaussian $g(\phi_o) = ae^{-(\phi_o - b)^2}$
- Parameters a and b then gives us the final 2D LUT

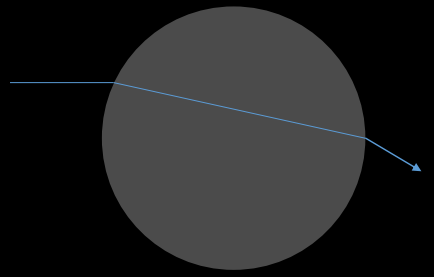


But instead of storing this whole 3D texture, we instead reparametrize it down to 2D by fitting a gaussian function to each azimuthal angle slice.

So the parameters a and b, in the gaussian, are fitted to the integral and we then store them in a two-channel 2D texture.

Azimuthal scattering N_{TT}

- Distribution approximated with LUT
  - $D_{TT} = g(\mathrm{LUT}_{D\,TT}[\theta_o, \beta_N], \phi_o)$
- Attenuation approximated with
  - $A_{TT} = A_p(1, h_{TT})$
  - Where $h_{TT} = 0$

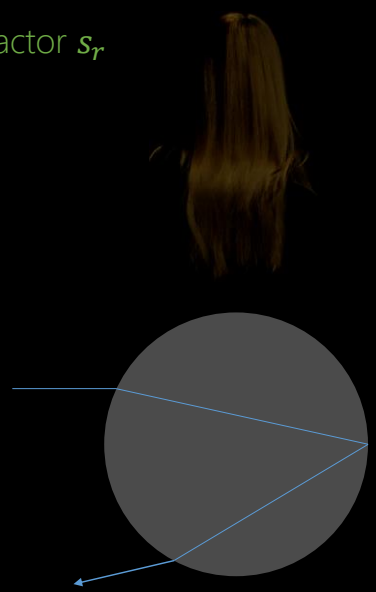


So to summarize the transmitted path

The distribution is approximated using our LUT, where we look up the parameters for the gaussian and then evaluate using the azimuthal angle.
And the attenuation term A is approximated using a constant value of h=0

Azimuthal scattering N_{TRT}

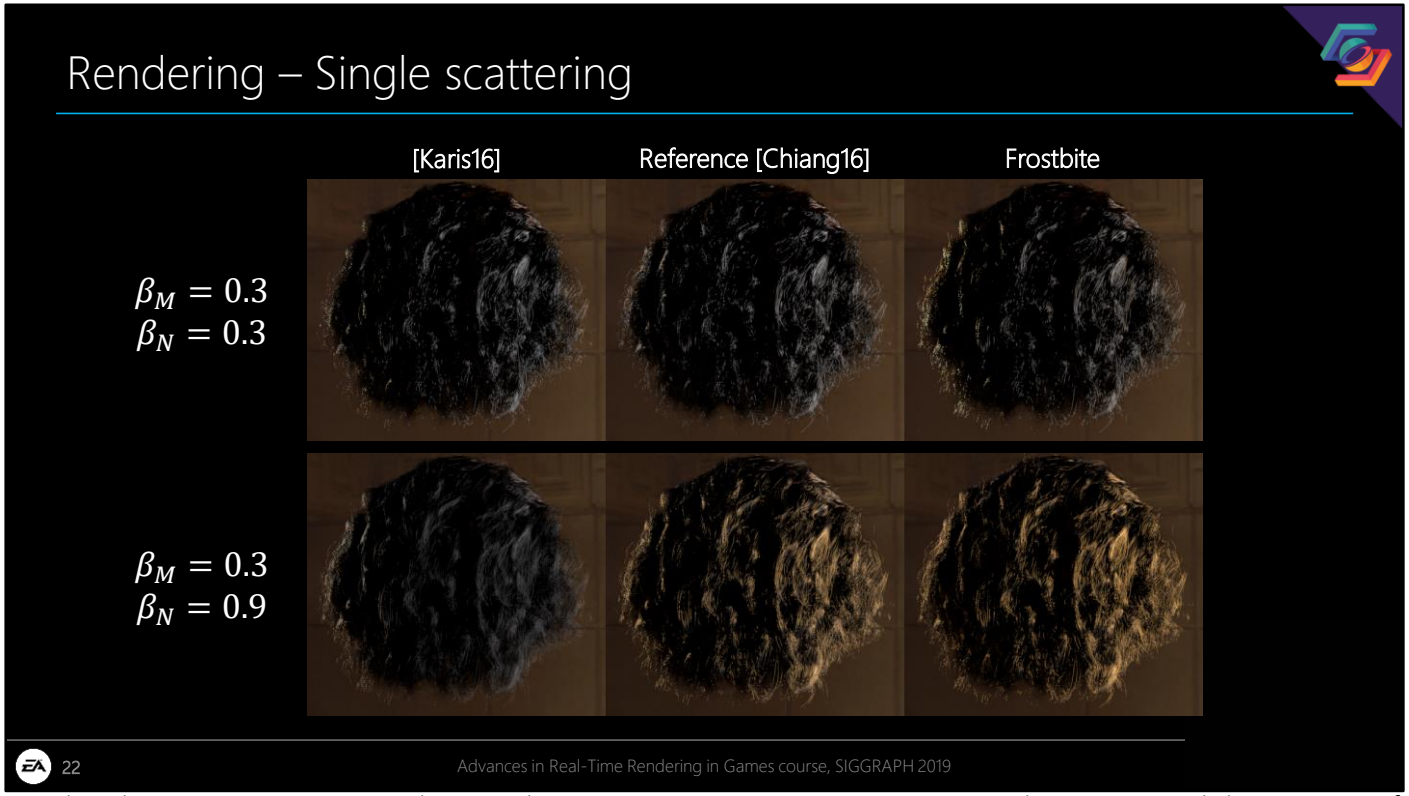
- Distribution improved Karis approximation using a scale factor s_r
  - $D_{TRT} = s_r e^{s_r(17\cos\phi - 16.78)}$
  - Manually fitted to approximate effect of roughness β_N
  - $s_r = \text{clamp}\left(1.5(1-\beta_N)\right)$
- Attenuation approximated with
  - $A_{TRT} = A_p(2, h_{TRT})$
  - Where $h_{TRT} = \frac{\sqrt{3}}{2}$ [Karis16]



And now for the final TRT path.

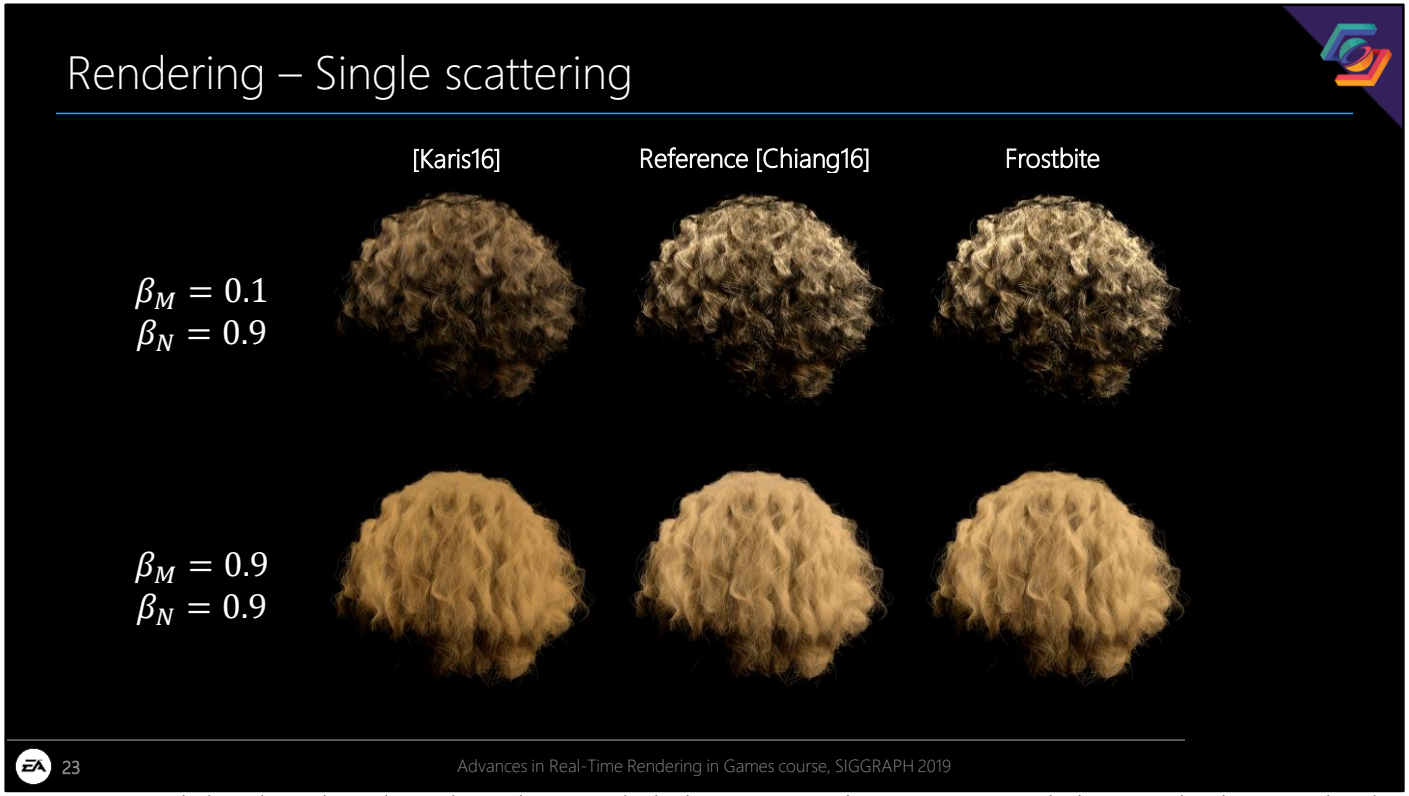
For the distribution we improved upon Karis approximation by adding a scale factor s_r which you can see highlighted in the equation here.
This scale factor was manually adapted to approximate the effect of surface roughness, like this.
This approximation is, however, still quite pretty coarse and may need some more work to improve the visual quality in some cases.

The attenuation term we approximate in the same way we did for the transmissive path, but here instead we use an h value of square-root of three divided by 2. Which is the same constant used in Karis approximation.



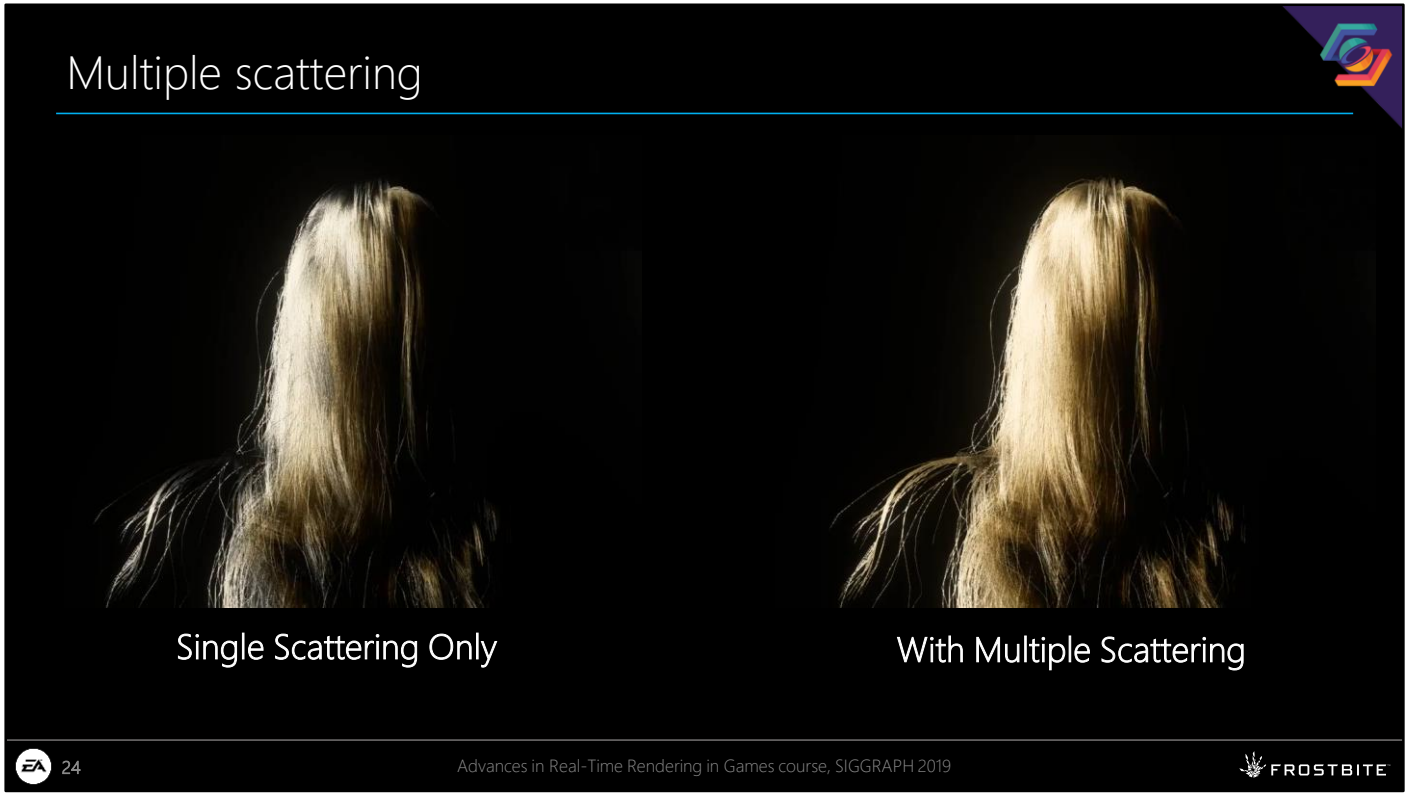
Putting all of this together, here is a comparison showing how our approximations compare with to Karis and the Disney reference.

And as you can see our changes to the TT and TRT better captures the changes to especially, the color saturation that you get when you use hair with higher roughness values.



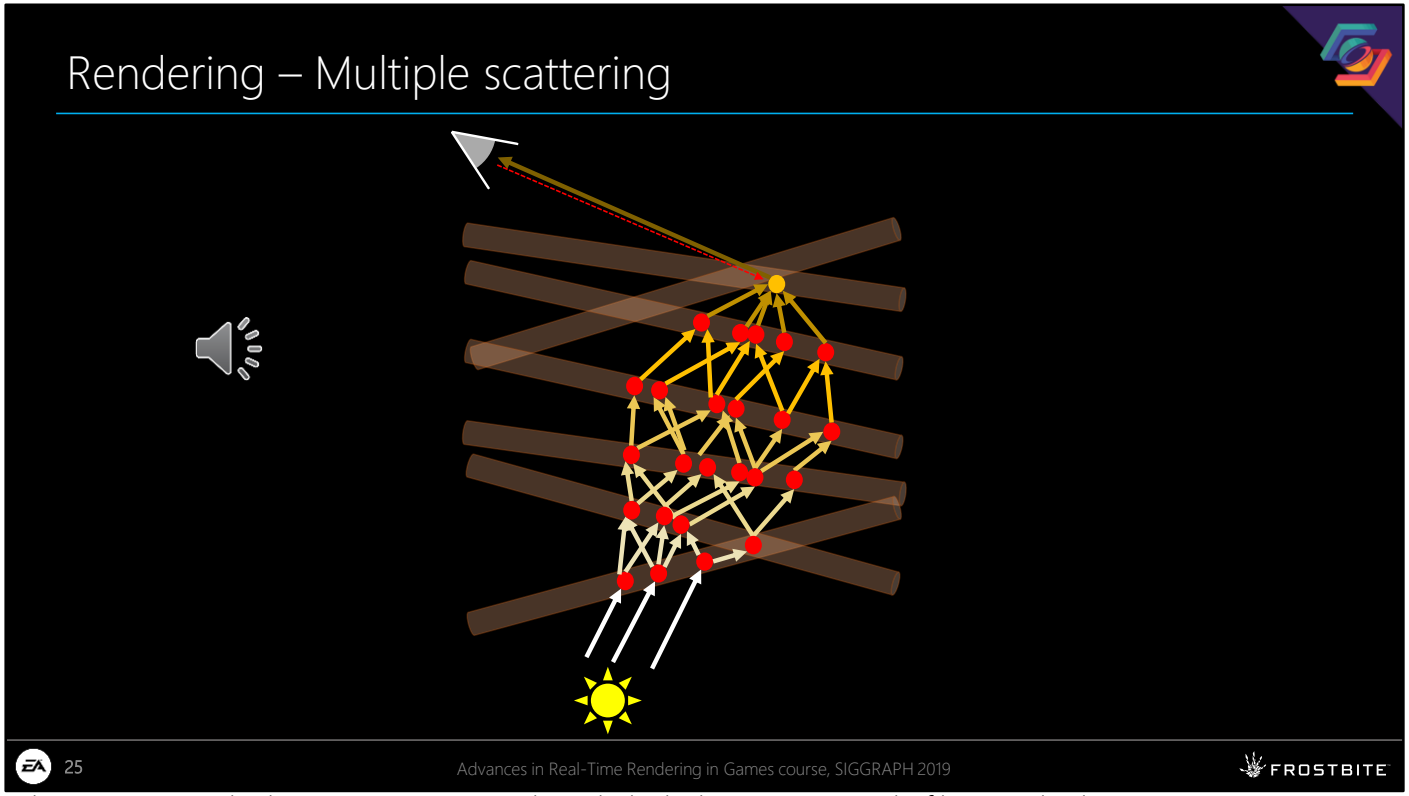
Here is another comparison with brighter hair, lit only with a single light source, where we instead change the longitudinal roughness value.

Here we can see improvements in both the highlights, for the shiny hair in the top row, and color in the bottom row.

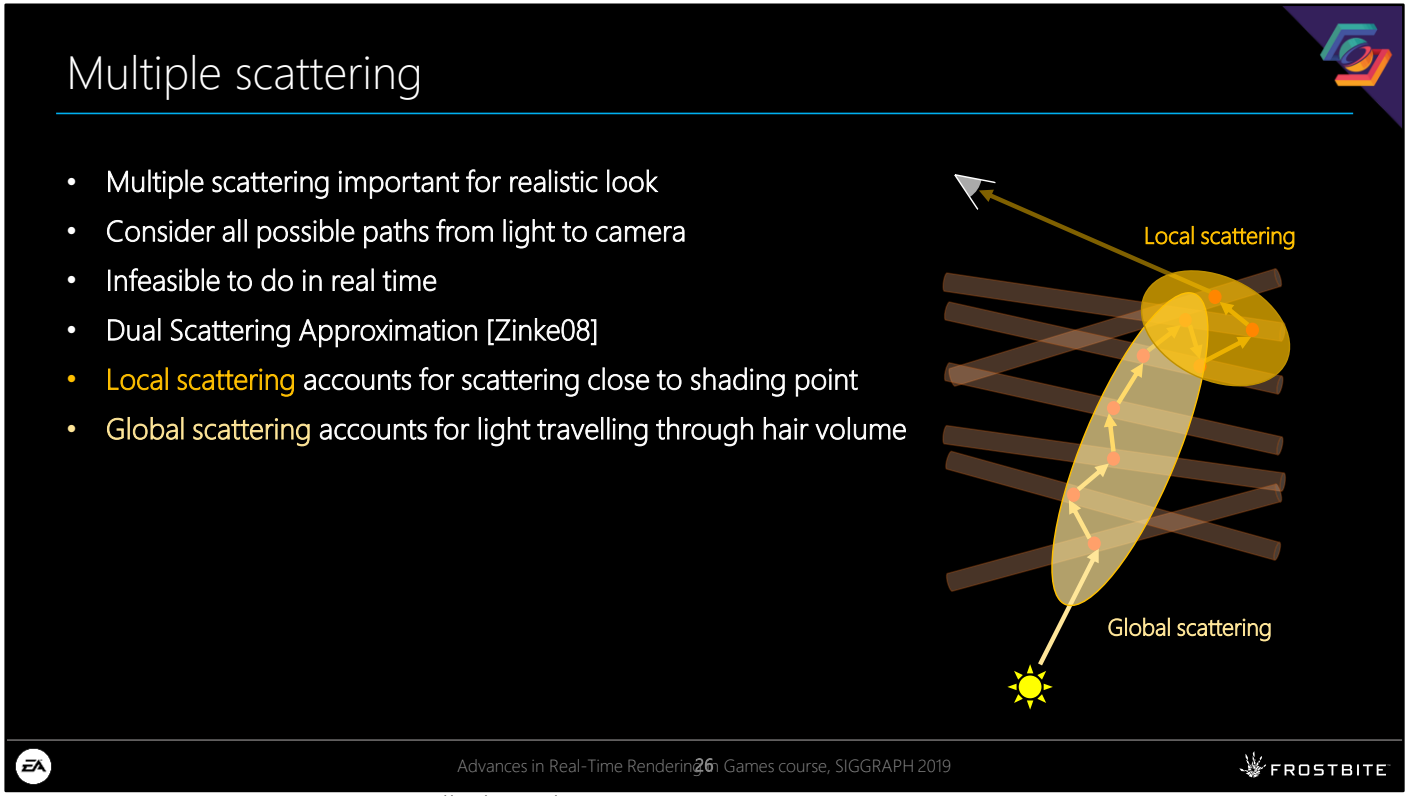


Now onto multiple scattering
Multiple scattering is an important effect to capture to get realistic color saturation and sense of depth/volume, especially for lightly colored hair.

So if you look at the movie to the right, you will hopefully be able to see that it looks more saturated and that the lighting in the hair volume looks smoother and more natural.



In contrast with single scattering, which aims at capturing how light behaves in a single fiber, multiple scattering tries to model the effect when light travels through many fibers.
This means that we need to evaluate multiple paths that the light travel between a light source and the camera.
This is of course not feasible for real-time rendering, so we need to approximate this effect as well.



In our implementation we use an approximation called Dual Scattering

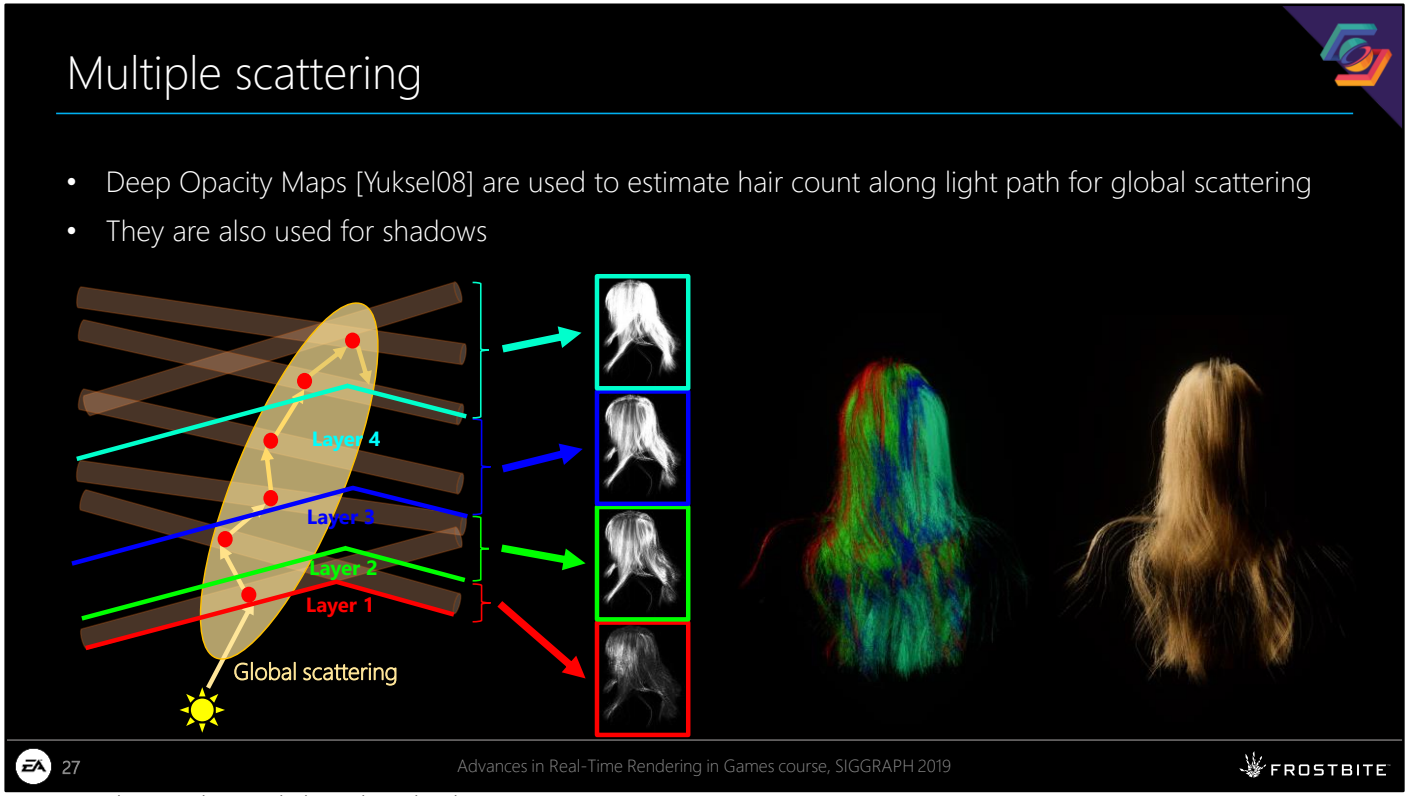
The point of dual scattering is to approximate multiple scattering as a combination of two components.

Local scattering and Global scattering

Local scattering accounts for scattering in the neighborhood of the shading point and accounts for a lot of the visible hair coloring.
Global scattering is meant to capture the effect of outside light travelling through the hair volume.

The reason that the dual scattering approximation works well for hair is because most light is only scattered in a forward direction. So basically because we have more contribution from TT than TRT.

Global scattering is estimated by only considering scattering along a shadow path, or light direction. Therefore we need some way of estimating the amount of hair between two points in the hair-volume in the light direction.



We do this the same way the authors did in the dual scattering paper; we use Deep Opacity Maps.

Deep opacity maps are similar to Opacity shadow maps, a technique where shadow maps for a volumetric object is generated in a lot of slices over the object.
The benefit of deep opacity maps is that it require a lot fewer layers and it does not suffer from banding artifacts common with opacity shadow maps.

We use 4 deep opacity map layers and 4 layers to be able to accumulate the hair transmittance.

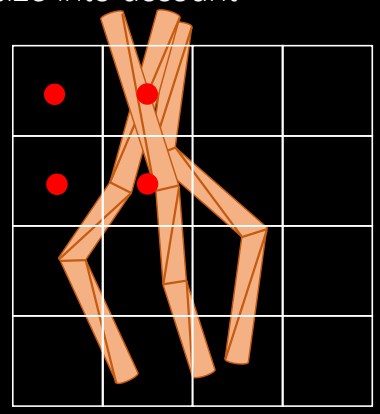
The attenuation due to the scattering is then calculated, averaged and stored into a LUT. The deep opacity maps are also used to determine shadows.

As a lower quality fallback one can also estimate the attenuation using a hair density constant and the Beer-Lambert law. But this will of course not adapt with the actual changes of the hair volume.

I recommend anyone that is interested in more details regarding dual scattering or deep opacity maps to read the original papers.

Rendering

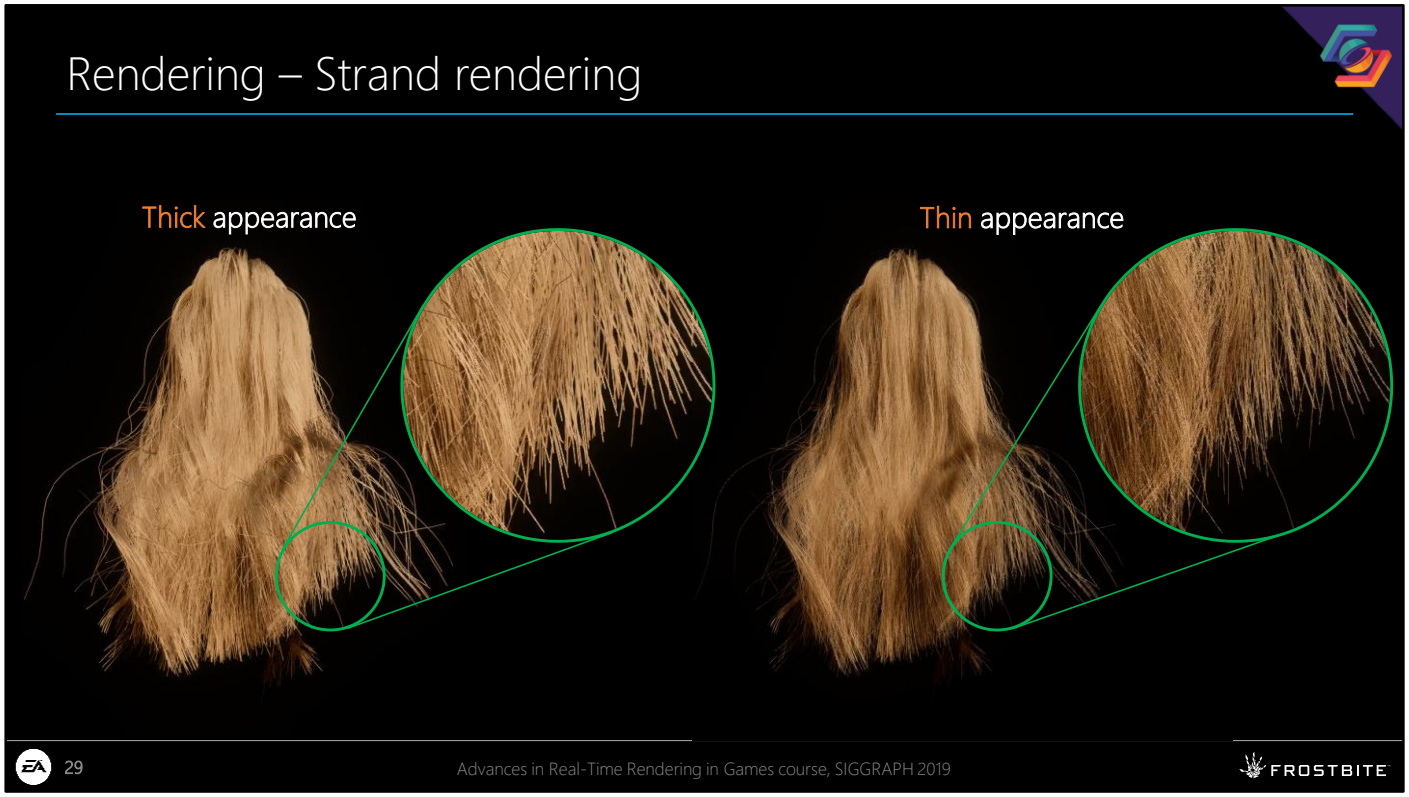
- Hair is tessellated as triangle strips
- Width is usually less than pixel size
- Tessellation must take pixel size into account



The hair-strands are tessellated and rendered as triangle strips so we must take special care to properly handle aliasing.

Since the strands are very thin, they will usually have a width that is less than that of a screen pixel.

We therefore need to take the pixel size into account when tessellating, and increase the width appropriately, or we will risk getting missing or broken up hair strands.



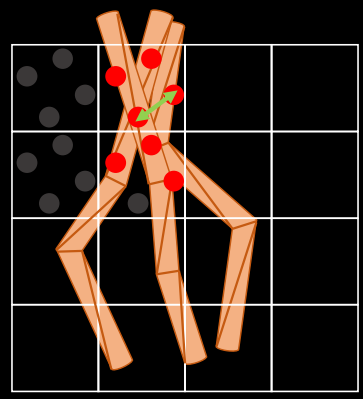
Unfortunately, this will have another not so nice side effect which can cause the hair to look too thick and more like thicker spaghetti or straw.

Another problem is that the amount of overdraw which will be massive and hurt performance a lot.

Rendering – Strand Rendering

Just enabling MSAA does not really help

+ Less aliasing
+ Thin appearance



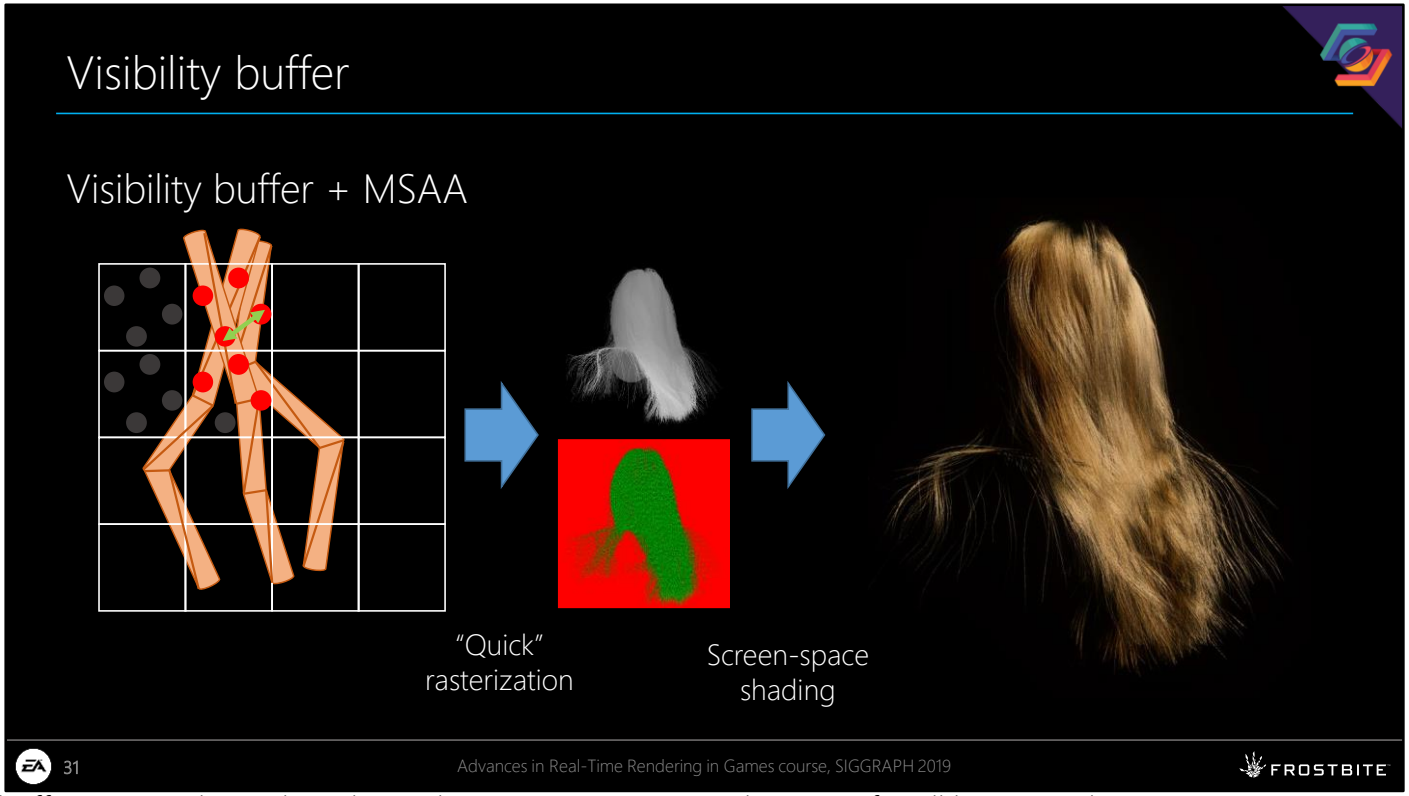
\- A lot of overdraw

Just enabling MSAA does unfortunately not solve all problems.

While it does improve on aliasing issues, by taking more samples per pixel, and therefore allows us to keep the thin hair appearance.

It will suffer an even bigger performance hit due to overdraw, because there will be more of it.

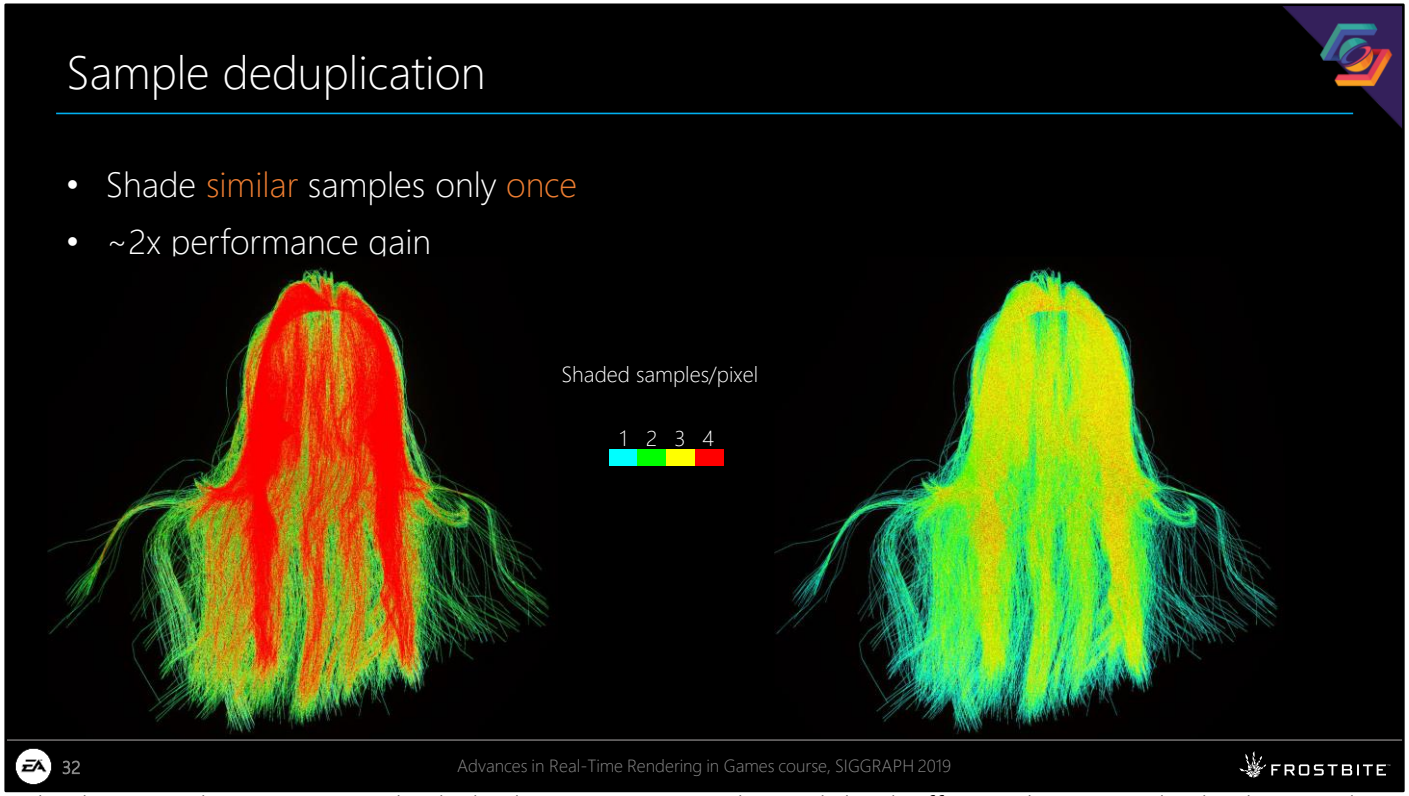
To reduce the amount of overdraw we use a visibility buffer



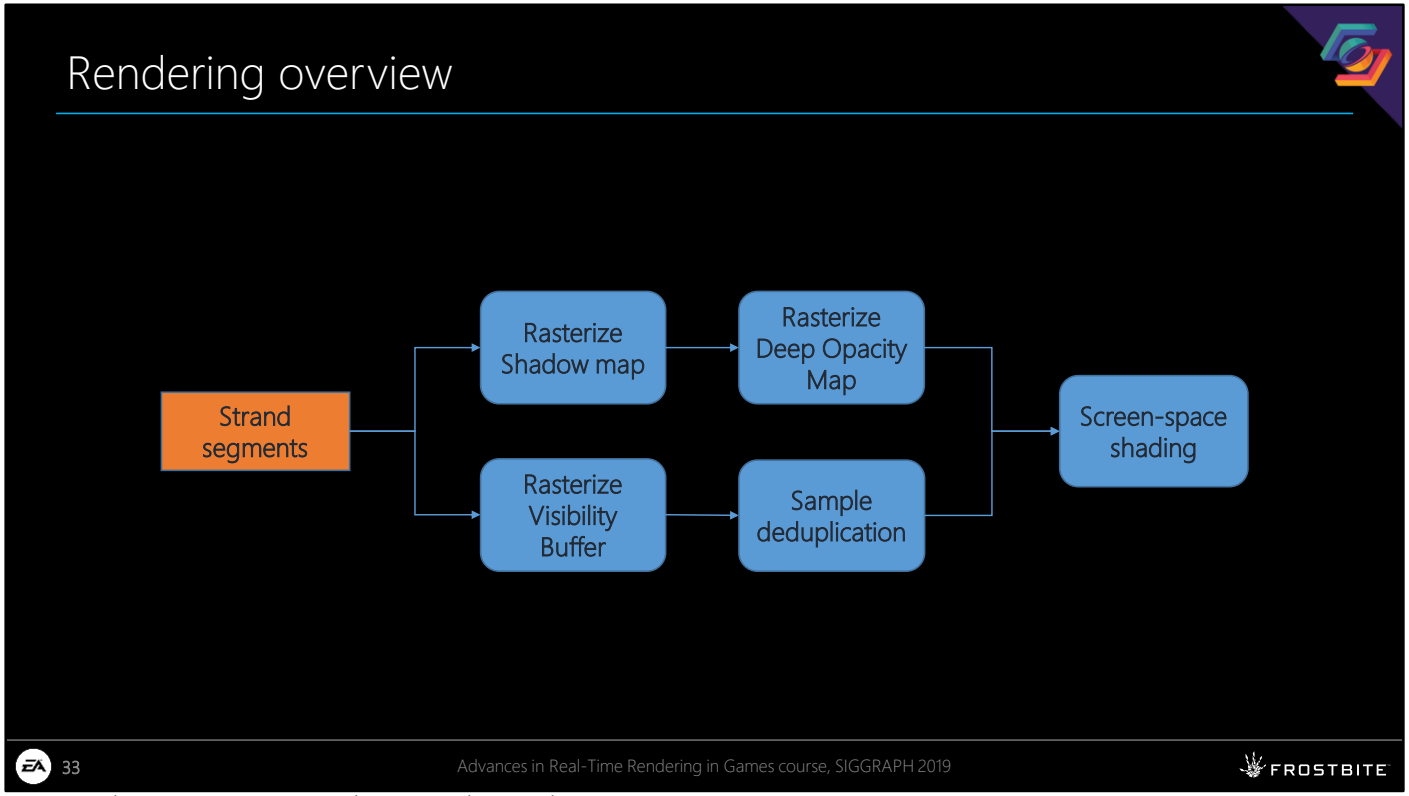
With the visibility buffer we can do a relatively quick rasterization pass, with MSAA, for all hair strands.

We can then use that information to do a screen-space shading pass to get the final antialiased render.

There is still, however, some unnecessary shading going on because we may be shading the same strand multiple times per pixel.

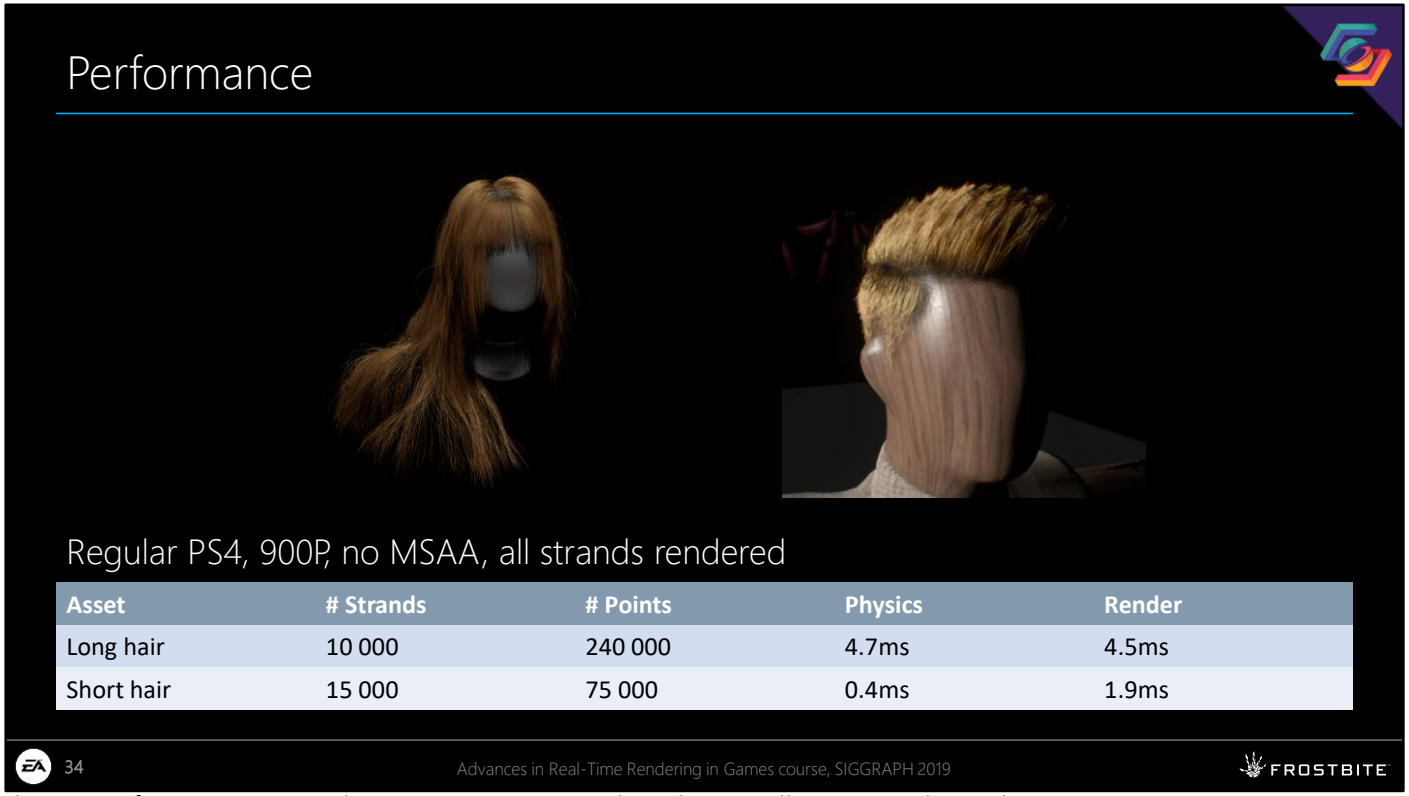


To reduce this over shading we also run a sample deduplication pass on the visibility buffer so that we only shade samples within a pixel when they are considered different.
This reduces the number of pixel-shader invocations greatly and it gave us roughly a 2 times performance increase compared to just using the visibility buffer.



Here is an overview over the steps we use when rendering hair.

We start with the simulated strand segments; these are rasterized into a shadow maps and the deep opacity maps. We also rasterize the visibility buffer run the sample deduplication pass before finally doing a screens-space shading pass.



So before talking about performance I need to again point out that this is still very much work in progress.
While we have been looking into performance it has mostly been to keep the project in the realm of possibility. As one example we have always rendered and simulated all the strands in our test models, which is probably not something someone would do in a game.

At Frostbite we usually work very close with the game teams to ease the introduction of new tech. And when they have it, they are usually very good at finding ways to get more with less.

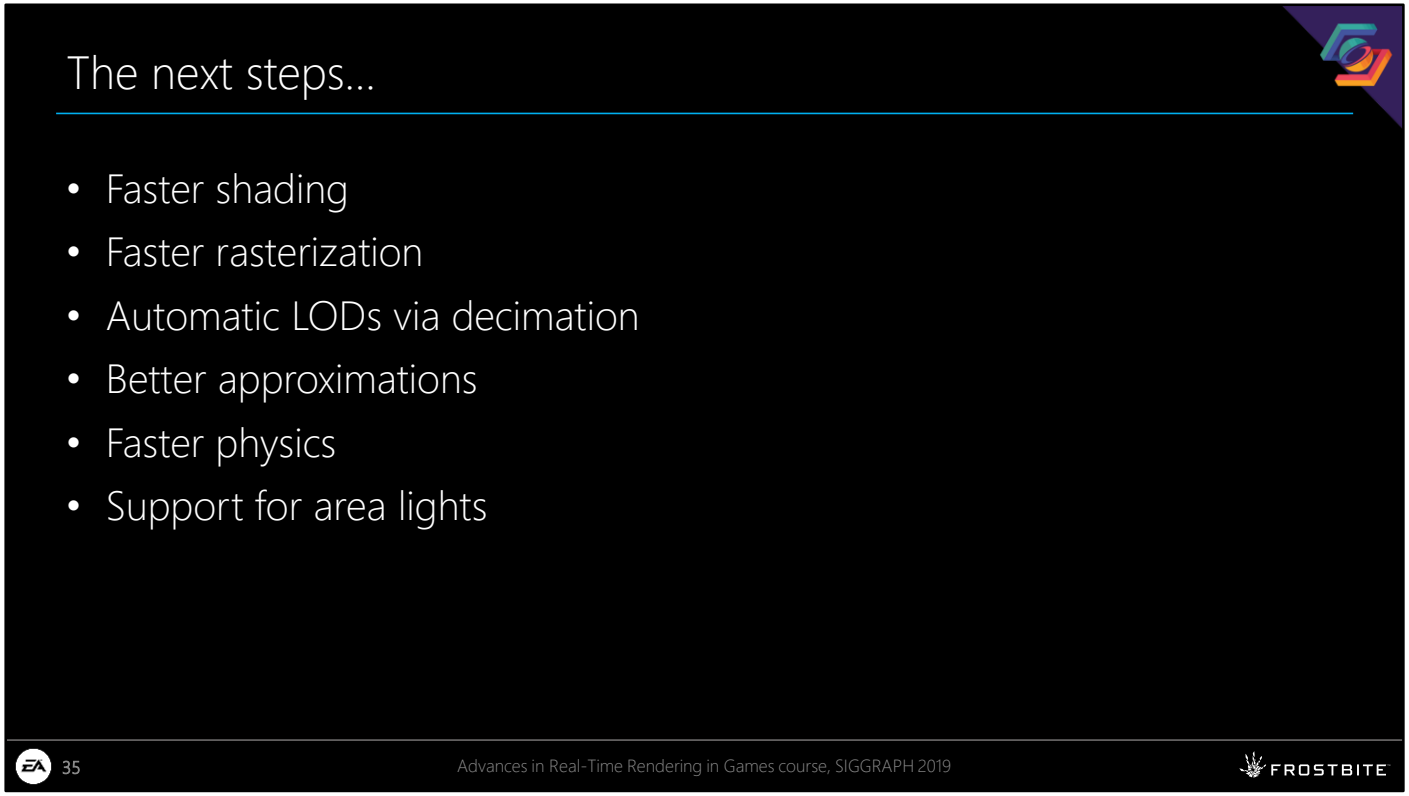
In any case, here are some numbers showing what the performance is currently like on a regular PS4 at 900p resolution, without MSAA, with the hair covering a big chunk of the screen.

So for the long-haired asset, which contains about 10'000 strands and a total of 240'000 points, the physics currently take around 4.7ms and the rendering takes around 4.5 ms.
The short-haired asset, which contains about 15'000 strands and a total of 75'000 points, the physics instead take around 0.4ms and rendering about 1.9ms.

The main reason for the long render times are currently that our GPU utilization is very low, something we are currently investigating different ways

to improve.

Keep in mind again that this is with all strands rendered and simulated every frame. In comparison some of the alternative hair simulation system only simulate about 1% of all hair strands. Early experiments show that we can get a 5x performance boost by simulating only 1/10th of all strands and interpolating the results.



So what are next steps

We definitely need to improve the performance more, especially shading.

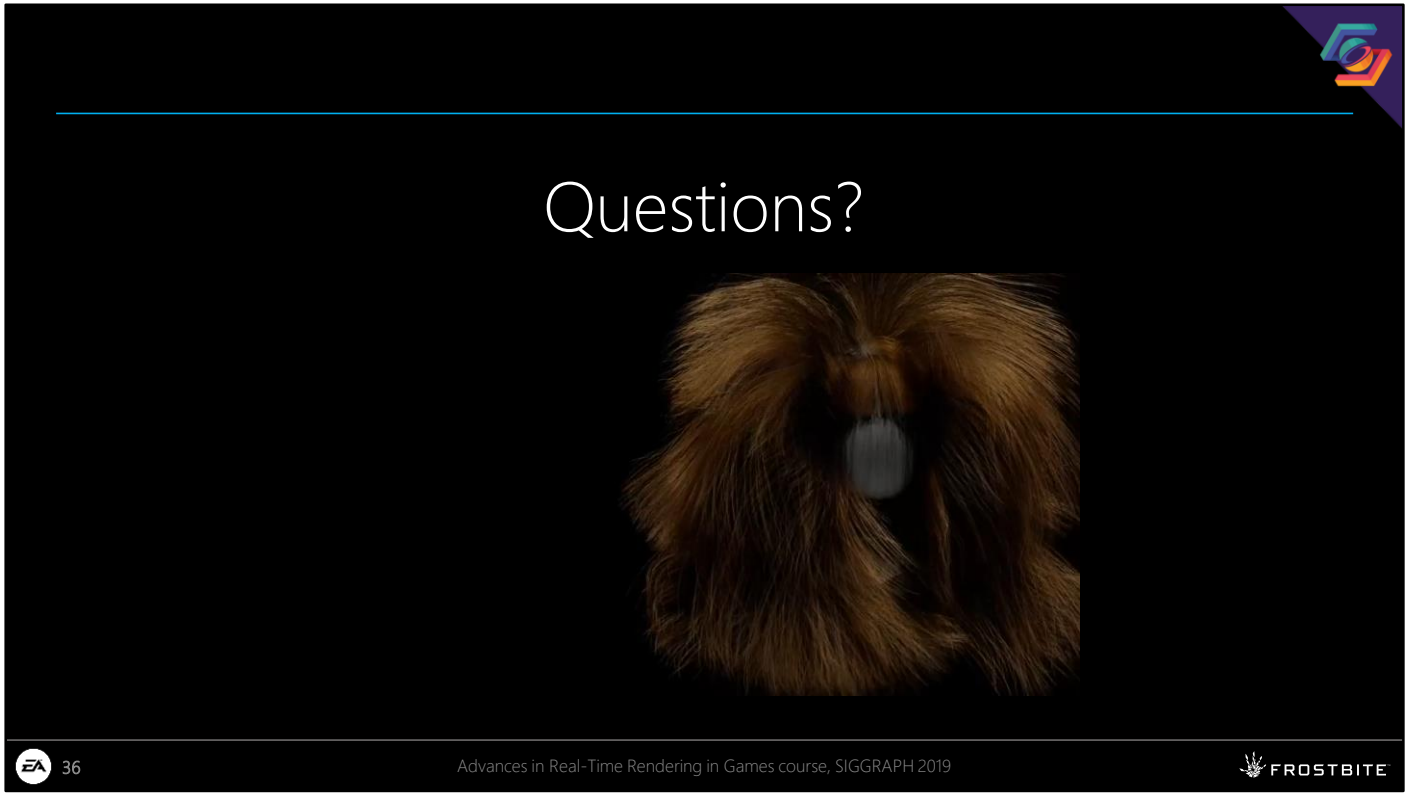
We are also investigation methods to get faster rasterization to improve GPU utilization

Related to that we plan to introduce automatic LOD generation via decimation.

We also would like to further improve the quality of the approximations

We also need to improve the performance of physics more and work more on simulating only a fraction of all strands.

And we want to investigate how to incorporate area lights into the shading model.


Questions?
36
Advances in Real-Time Rendering in Games course, SIGGRAPH 2019
FROSTBITE


Thank You!
www.ea.com/frostbite/news
37
Advances in Real-Time Rendering in Games course, SIGGRAPH 2019
FROSTBITE

References

[Marschner03] Light Scattering from Human Hair Fibers

[Yuksel08] Deep Opacity Maps

[Zinke08] Dual Scattering Approximation for Fast Multiple Scattering in Hair

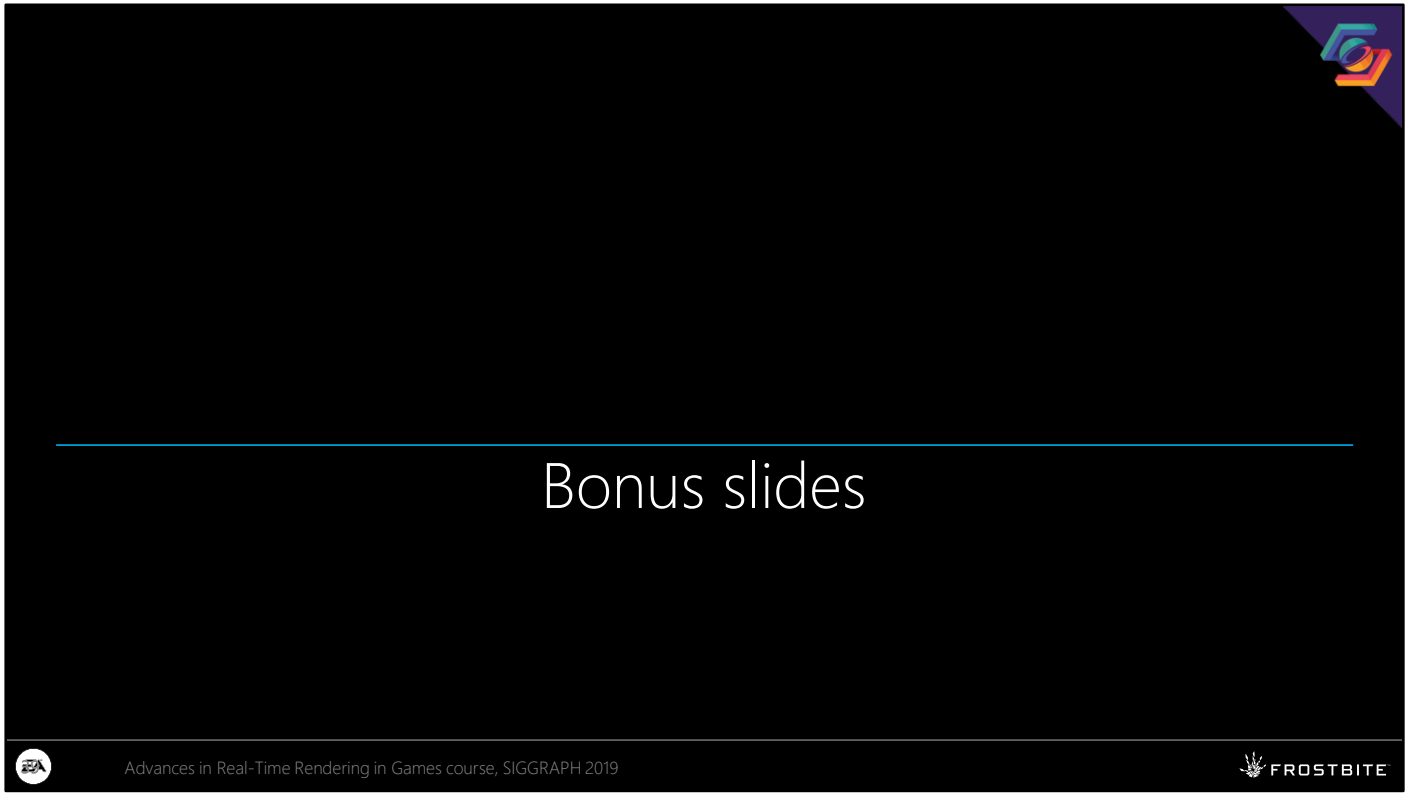
[d'Eon11] An Energy-Conserving Hair Reflectance Model

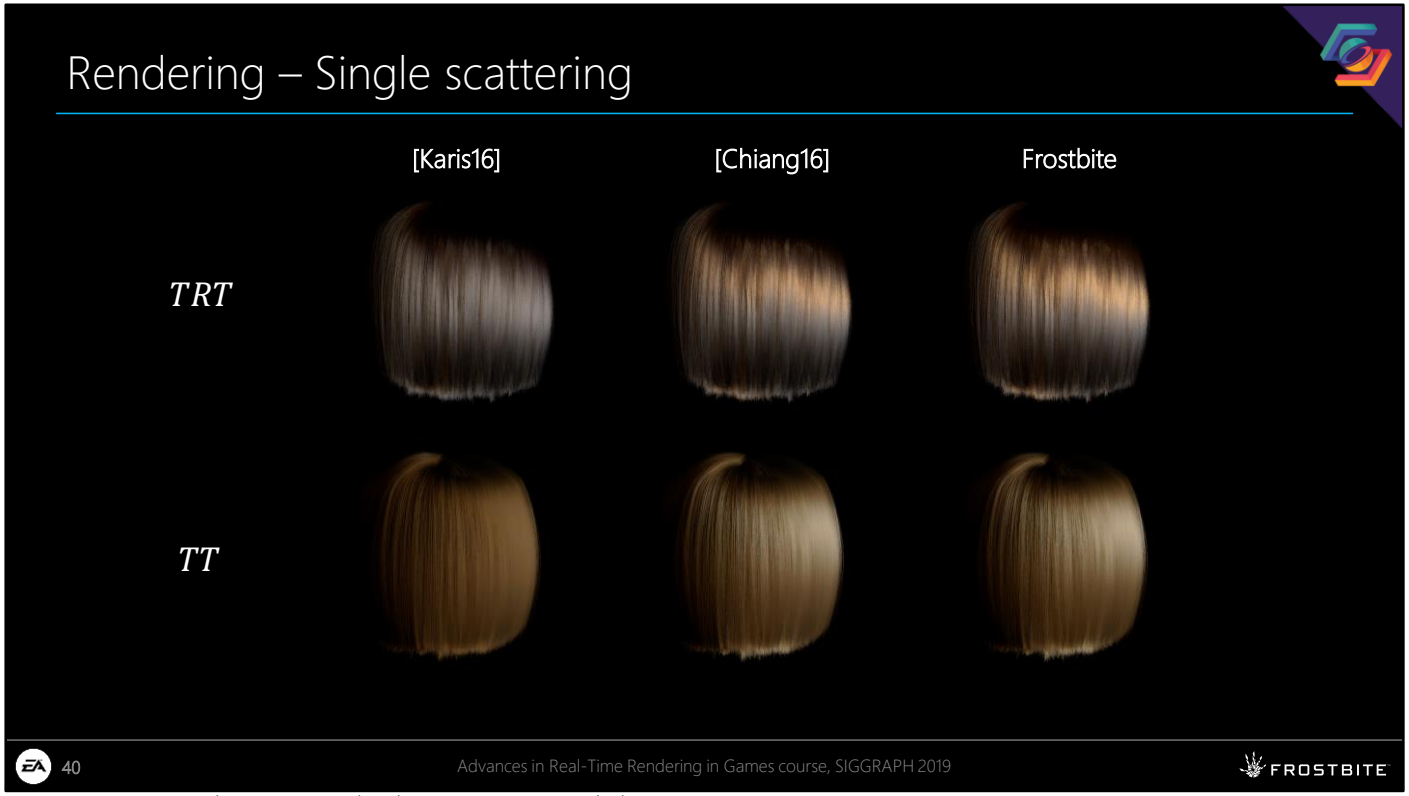
[Burns13] The Visibility Buffer: A Cache-Friendly Approach to Deferred Shading

[Pekelis15] A Data-Driven Light Scattering Model for Hair

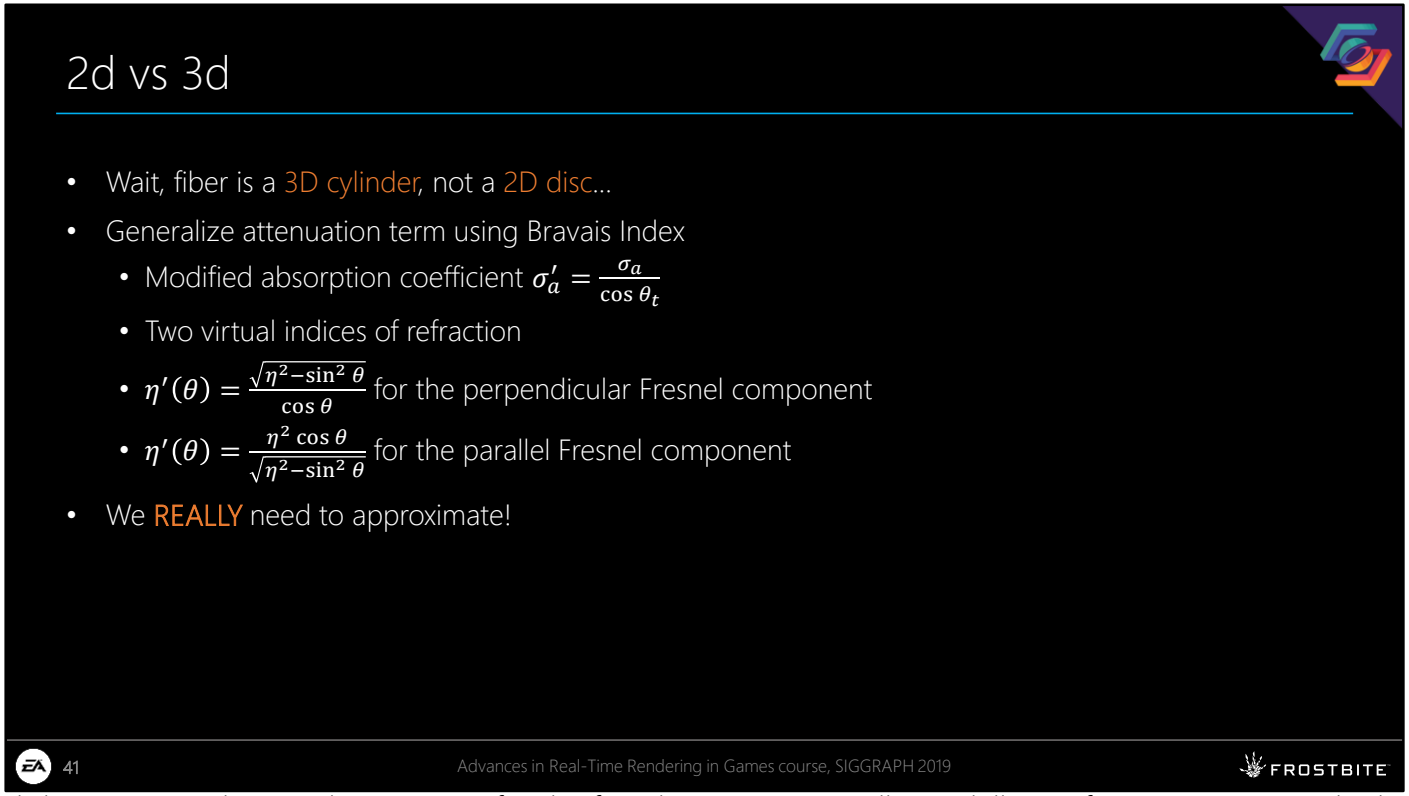
[Chiang16] A Practical and Controllable Hair and Fur Model for Production Path Tracing

[Karis16] Physically Based Hair Shading in Unreal


Bonus slides
Advances in Real-Time Rendering in Games course, SIGGRAPH 2019
FROSTBITE



And here is another comparison showing only the TT term and the TRT term



To make things slightly worse we also need to account for the fact that we are actually modelling refraction on a 3D cylinder and not a 2D disc.

Applying Bravais index it turns out we can fully account for this by modifying the absorption coefficient
And by using two virtual indices of refraction. The first one, eta-prime is used for the perpendicular part of the Fresnel equation and the second one, eta-double prime is used for the parallel Fresnel part.

We REALLY need to approximate!