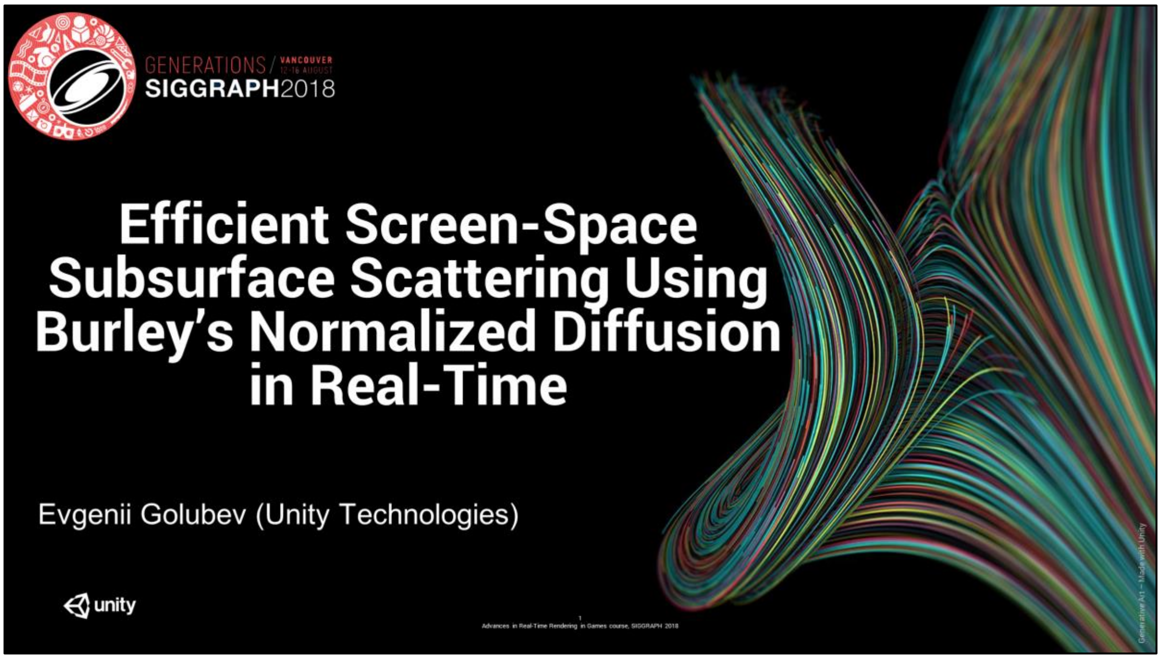


GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

Efficient Screen-Space Subsurface Scattering Using Burley's Normalized Diffusion in Real-Time

Evgenii Golubev (Unity Technologies)

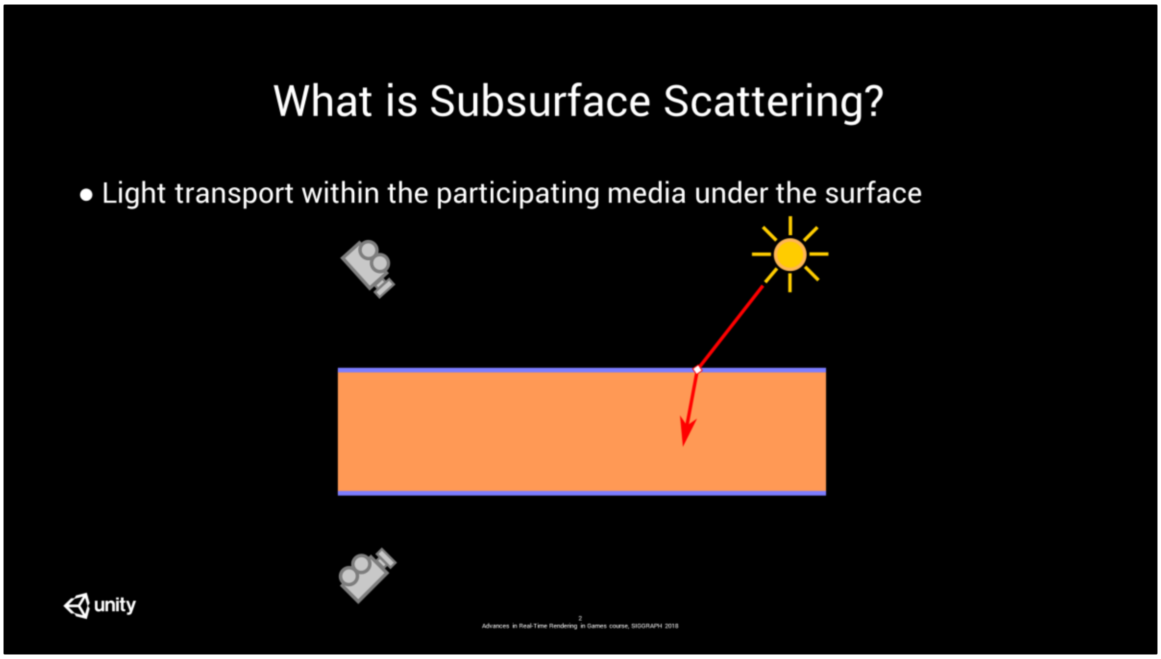
unity

Generative Art – Made with Unity

Hello again, and thanks for sticking around.

I will give an overview of how subsurface scattering (or SSS for short) is implemented in Unity's High Definition Render Pipeline.

Due to time constraints, I will be brief. You can check the slides for more details later.



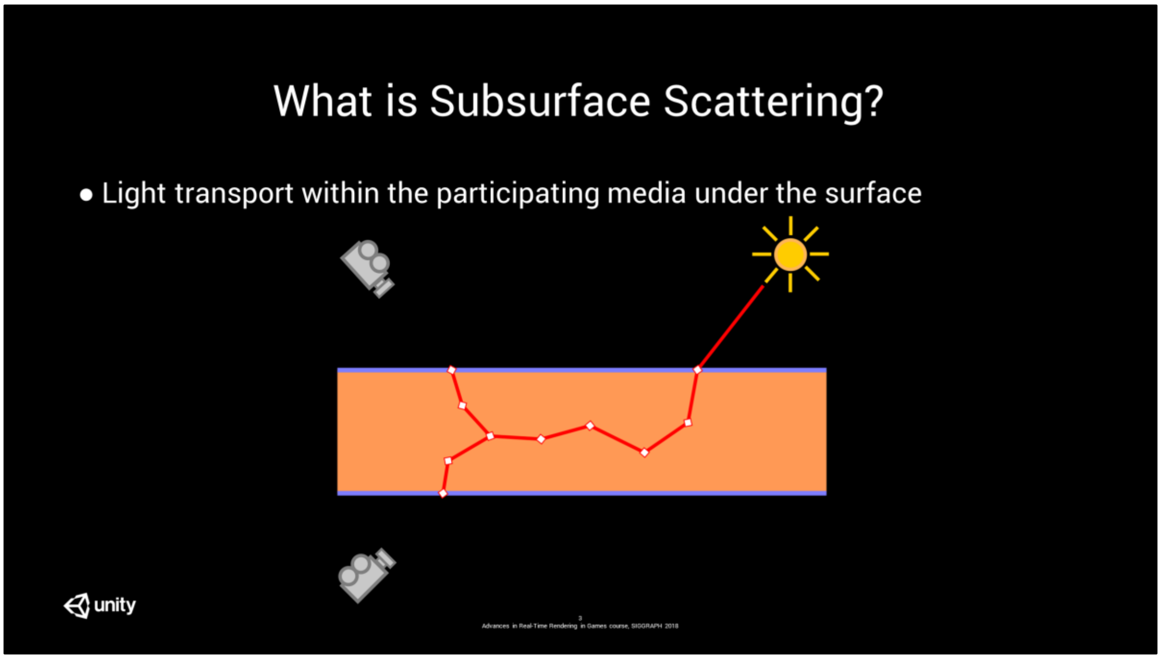
What is Subsurface Scattering?

- Light transport within the participating media under the surface

Let’s start by trying to understand what **is** subsurface scattering.

By definition, it is light transport within the participating media under the surface.

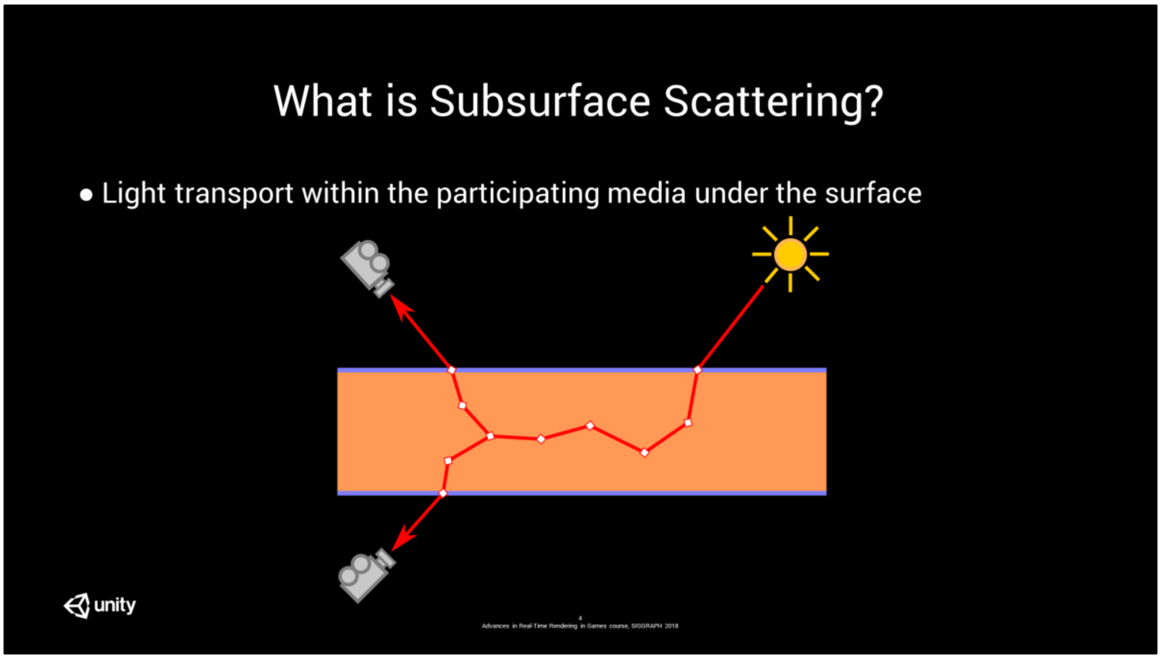
Light is refracted at the dielectric boundary, ...



What is Subsurface Scattering?

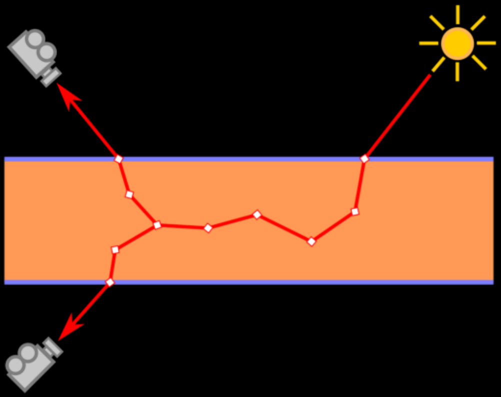
- Light transport within the participating media under the surface

... scattered within the participating media multiple times, ...

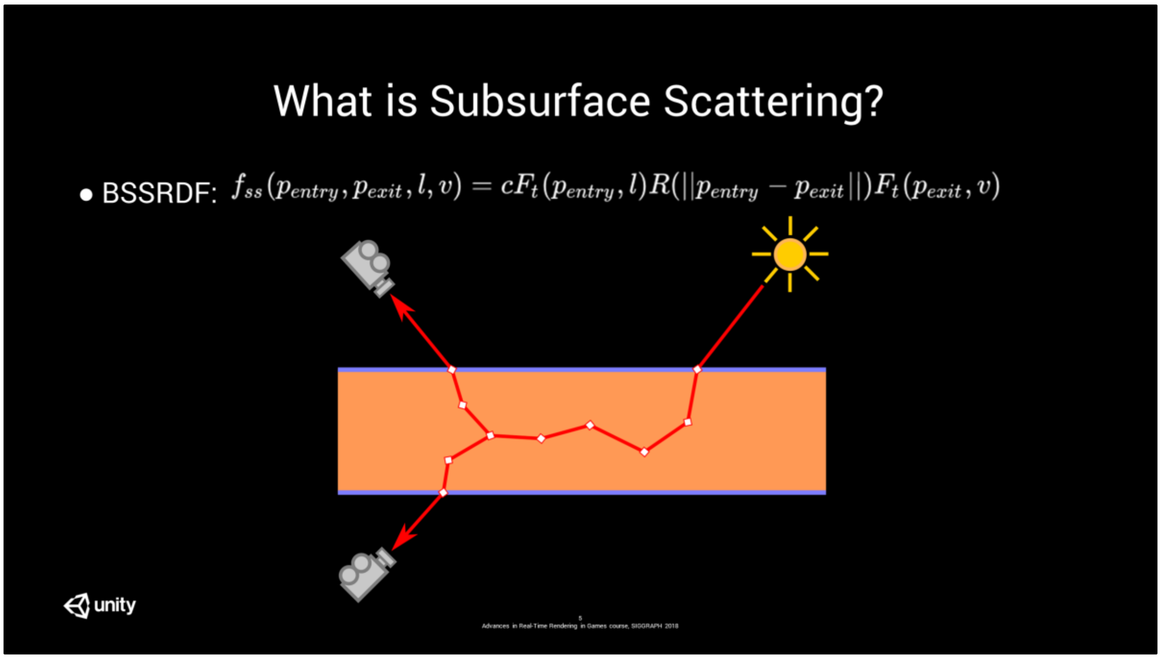


What is Subsurface Scattering?

- Light transport within the participating media under the surface



... and refracted outside again.



What is Subsurface Scattering?

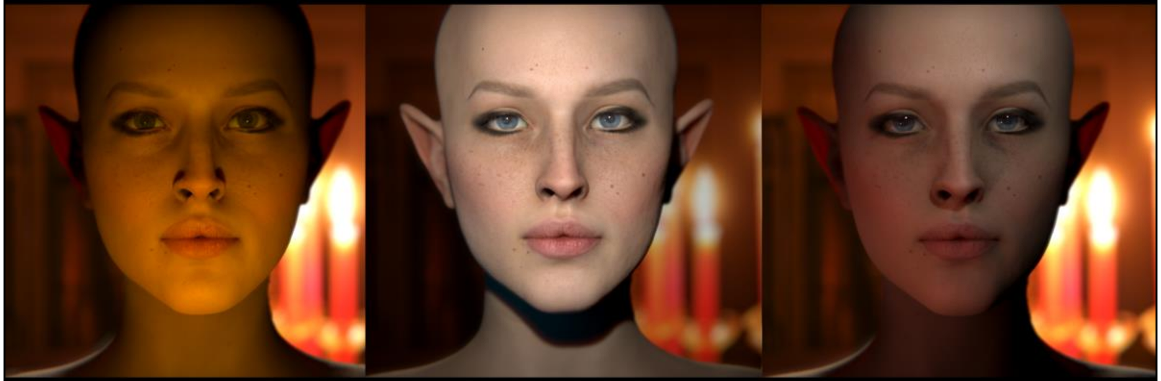
- BSSRDF: $f_{ss}(p_{entry}, p_{exit}, l, v) = cF_t(p_{entry}, l)R(||p_{entry} - p_{exit}||)F_t(p_{exit}, v)$

unity

Scattering is typically assumed to be isotropic (which is a reasonable assumption for multiple scattering), and is therefore modelled with a radially symmetric diffusion profile [Jensen 2001].

This leads us to the following definition of the bidirectional subsurface scattering reflectance distribution function (or BSSRDF for short), where R is the diffusion profile, F_t is Fresnel transmission and C is the normalization constant.

What is Subsurface Scattering?



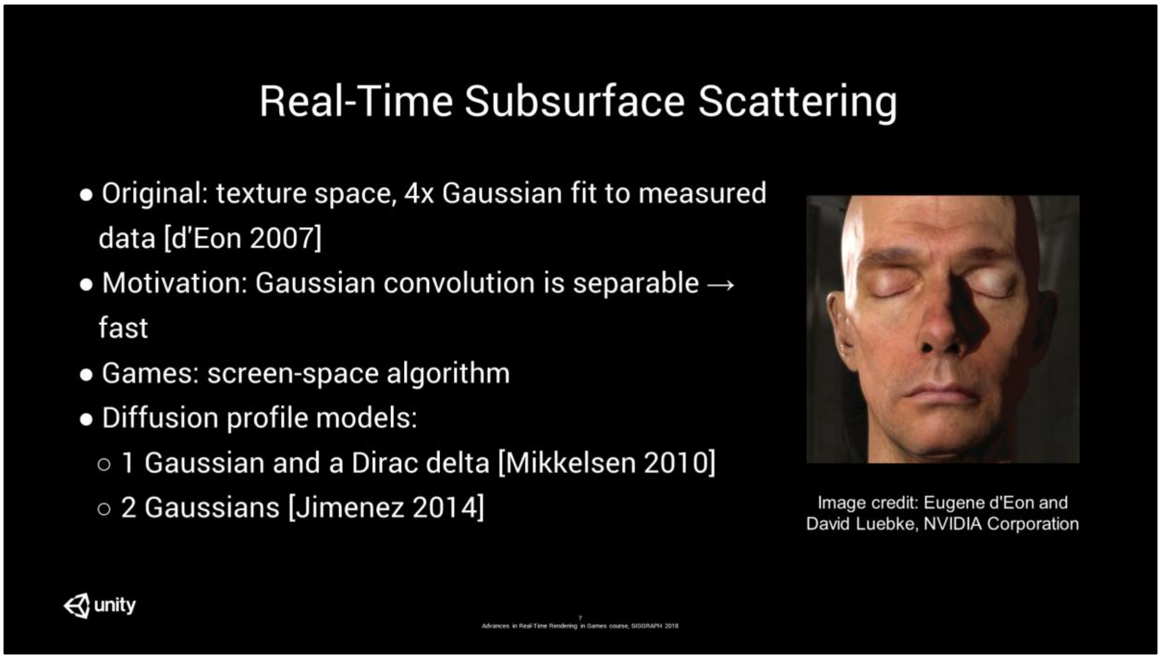
Model credit: Dan Roarty

unity

What this means in practice is that materials such as skin typically look smooth and organic rather than plasticy. <here is an example of a model showing subsurface scattering on skin and eyes rendered with HDRP in real-time>

You also get some color bleeding around areas where illumination changes abruptly.

It appears that there's a certain misconception that with SSS you get pronounced color bleeding near all shadow boundaries. In fact, it is very subtle if the shadow is soft.



Real-Time Subsurface Scattering

- Original: texture space, 4x Gaussian fit to measured data [d'Eon 2007]
- Motivation: Gaussian convolution is separable → fast
- Games: screen-space algorithm
- Diffusion profile models:
  - 1 Gaussian and a Dirac delta [Mikkelsen 2010]
  - 2 Gaussians [Jimenez 2014]

Image credit: Eugene d'Eon and David Luebke, NVIDIA Corporation

unity

How is subsurface scattering typically implemented?

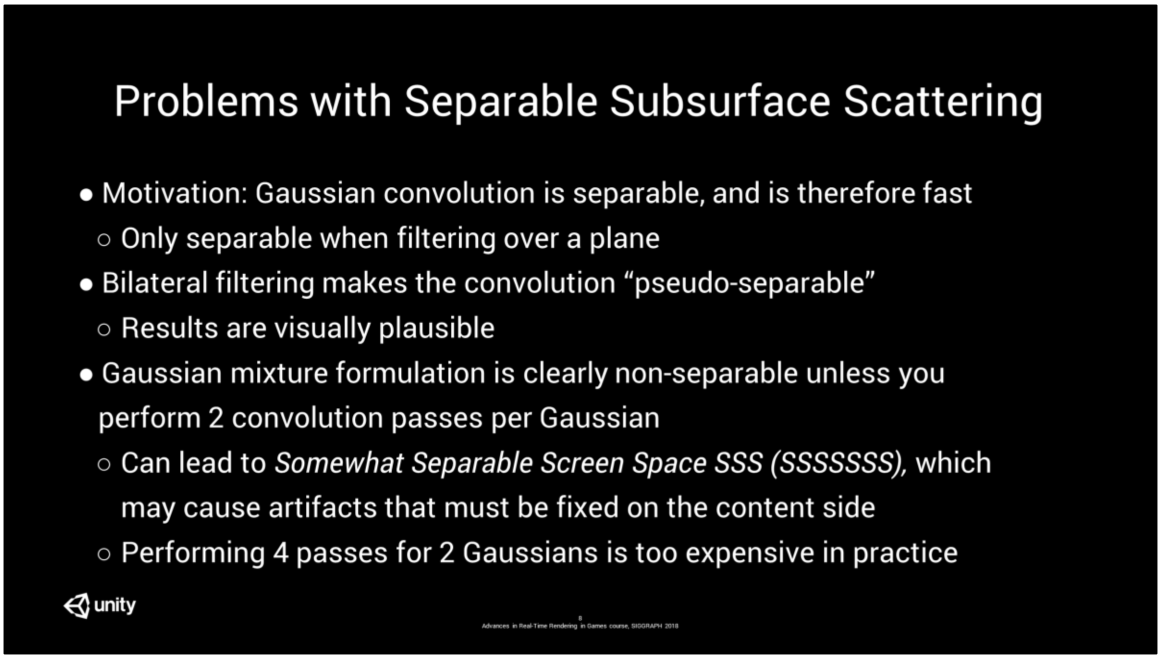
The first real-time SSS approach was proposed by Eugene d'Eon, and used a mixture of Gaussians (4, to be specific) to create a numerical fit for a measured diffusion profile.

Gaussian was chosen for several reasons:

- A mixture of Gaussians is a convenient target for a numerical fit.
- Additionally, convolution with a Gaussian is separable, and so it has a linear (rather than quadratic) complexity in the number of samples.
- And finally, repeated convolutions with a "smaller" Gaussian are equivalent to a single convolution with a "larger" Gaussian.

This approach yields fantastic results, but it is still too expensive for games, even today. Therefore, real-time constraints force us to implement SSS in the screen space.

The filter is usually composed of 1 or 2 Gaussians and a Dirac delta function which, in practice, means that you interpolate between the original and the blurred image.

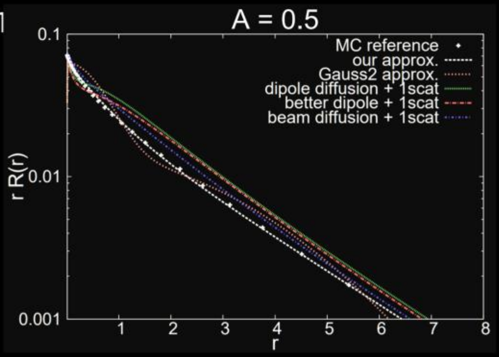


Problems with Separable Subsurface Scattering

- Motivation: Gaussian convolution is separable, and is therefore fast
  - Only separable when filtering over a plane
- Bilateral filtering makes the convolution “pseudo-separable”
  - Results are visually plausible
- Gaussian mixture formulation is clearly non-separable unless you perform 2 convolution passes per Gaussian
  - Can lead to *Somewhat Separable Screen Space SSS (SSSSSSS),* which may cause artifacts that must be fixed on the content side
  - Performing 4 passes for 2 Gaussians is too expensive in practice

Problems with Separable Subsurface Scattering

- 2 Gaussians only provide a partial fit for the measured data [Burley 201



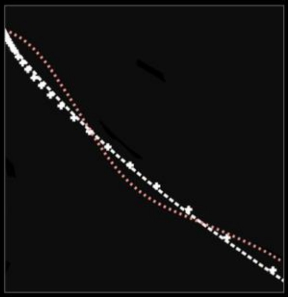


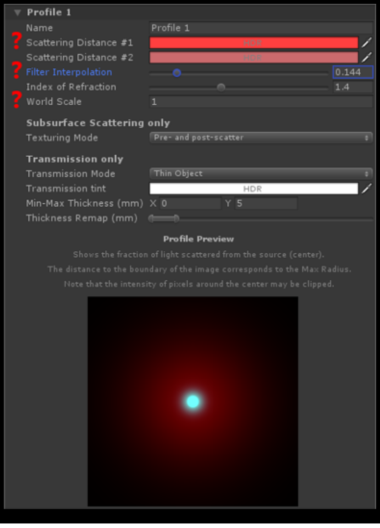
Image credit: Per H. Christensen and Brent Burley, Pixar & Walt Disney Animation Studios

A single Gaussian does a pretty good job at approximating multiple scattering. However, even a mix of 2 Gaussians struggles to accurately represent the combination of single and multiple scattering.

On the image, you can see the reference results in white, and the dual Gaussian approximation in red.

Problems with Separable Subsurface Scattering

- Gaussian mixtures are not artist-friendly
  - Gaussian is just a math concept
  - Artist is forced to either use the measured data or "eyeball" the numerical fit
    - Too many degrees of freedom (7)
    - Most combinations are nonsense



Profile 1

Name	Profile 1
? Scattering Distance #1	HDR
Scattering Distance #2	HDR
? Filter Interpolation	0.144
Index of Refraction	1.4
? World Scale	1

Subsurface Scattering only

Texturing Mode	Pre- and post-scatter

Transmission only

Transmission Mode	Thin Object
Transmission tint	HDR
Min-Max Thickness (mm)	X 0 Y 5
Thickness Remap (mm)	

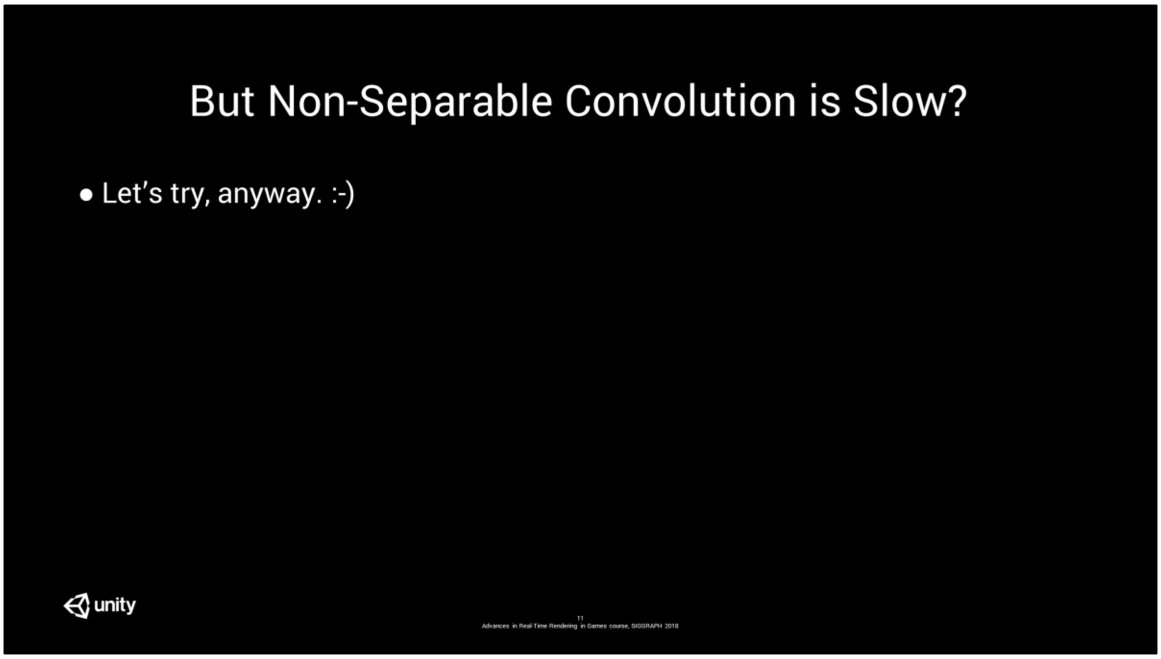
Profile Preview

Shows the fraction of light scattered from the source (center).
The distance to the boundary of the image corresponds to the Max Radius.
Note that the intensity of pixels around the center may be clipped.

After implementing the Gaussian mix model, we have encountered another problem - it is not artist-friendly.

A Gaussian is just a mathematical concept, and it has no physical meaning in the context of SSS. And when you have two of them with a lerp parameter, that's 7 degrees of freedom, and it's not exactly clear how to set them up so that the resulting combination makes sense.

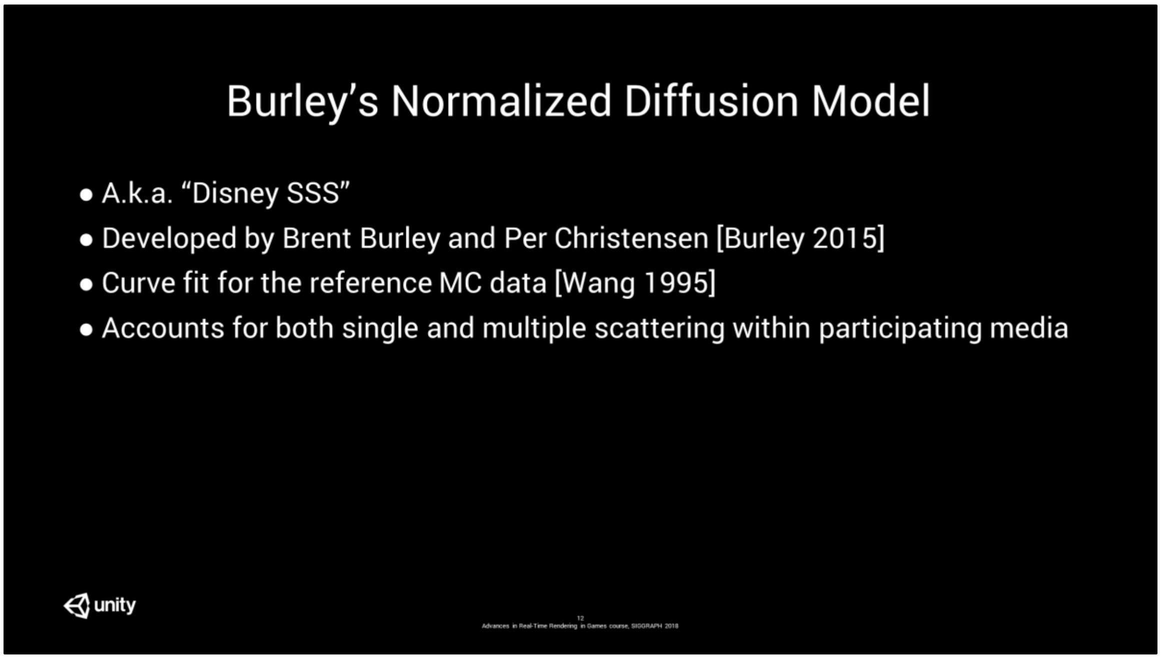
It is worth noting that a skilled artist can still achieve great results with the Gaussian mix model. However, this can be a lengthy and complicated process, which is not a right fit for Unity.



But Non-Separable Convolution is Slow?

- Let’s try, anyway. :-)

unity



Burley's Normalized Diffusion Model

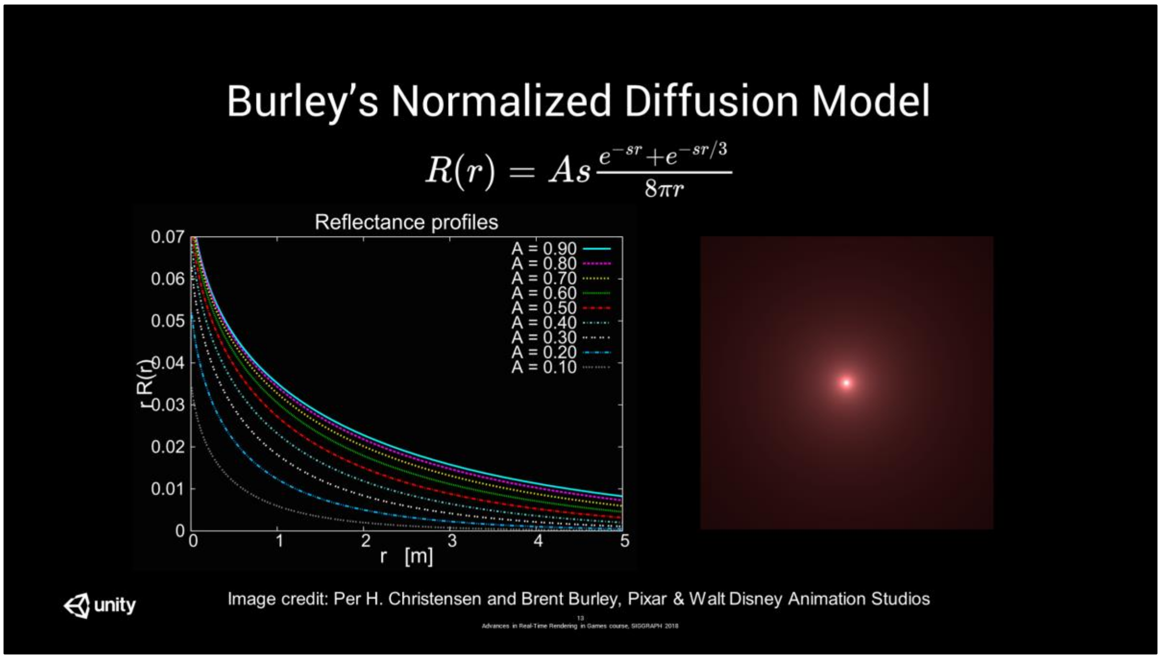
- A.k.a. "Disney SSS"
- Developed by Brent Burley and Per Christensen [Burley 2015]
- Curve fit for the reference MC data [Wang 1995]
- Accounts for both single and multiple scattering within participating media

unity

So, is there a better solution? Well, it is in the title of this talk. :-)

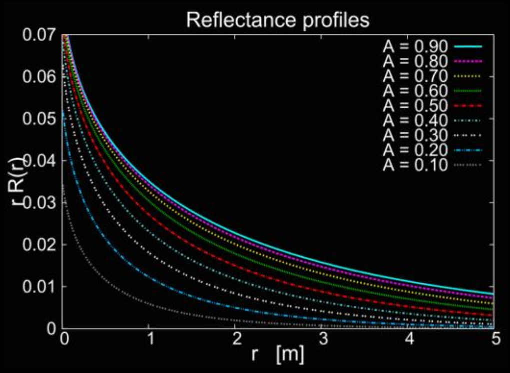
We implemented Burley's normalized diffusion model, which we call the Disney SSS.

It provides an accurate fit for the reference data obtained using Monte Carlo simulation.
Naturally, that means accounting for both single and multiple scattering.



Burley's Normalized Diffusion Model

$$R(r) = As\frac{e^{-sr} + e^{-sr/3}}{8\pi r}$$



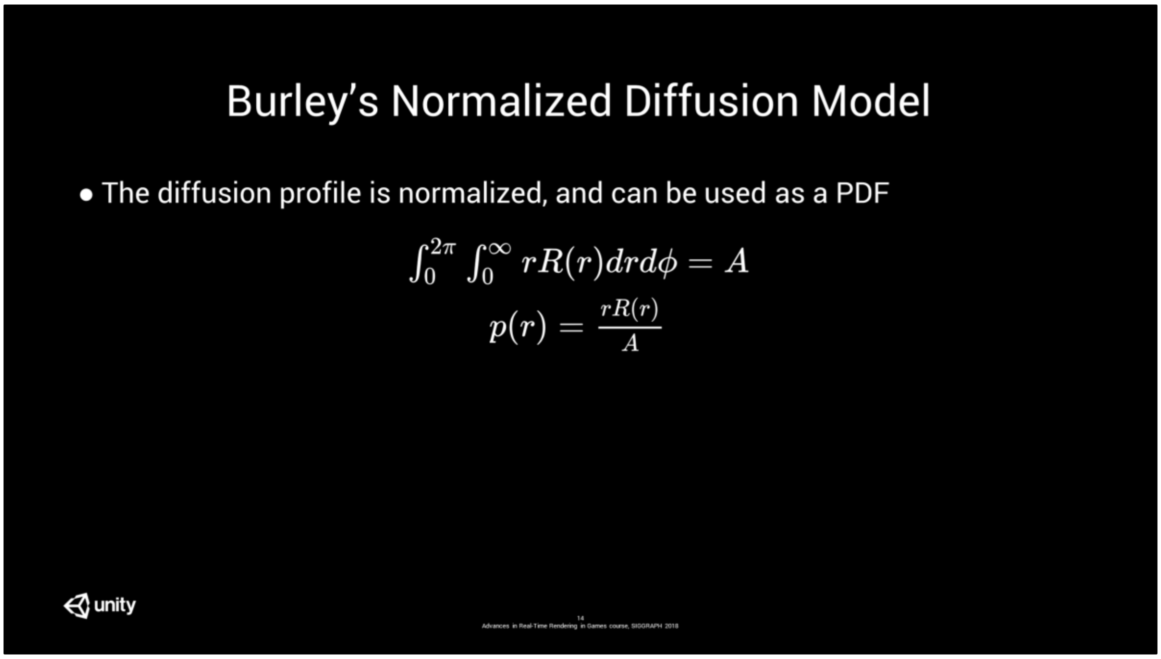
unity

Image credit: Per H. Christensen and Brent Burley, Pixar & Walt Disney Animation Studios

It has only two parameters: the volume albedo **A**, and the shape parameter **s**. Both of them can be interpreted as colors.
The shape parameter is inversely proportional to the scattering distance, and this is what we expose in the UI.

If we look at the graph of the resulting profiles, two things become obvious:

1. The sharp peak and the long tail cannot be modeled with a single Gaussian
2. The resulting filter is clearly non-separable



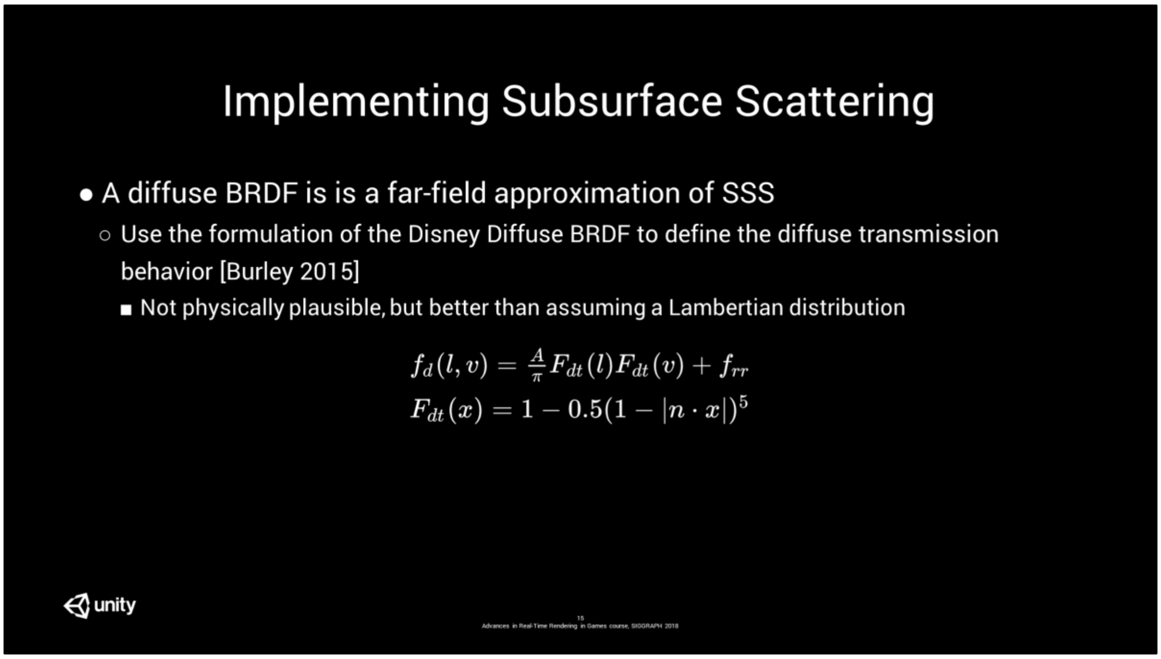
Burley’s Normalized Diffusion Model

- The diffusion profile is normalized, and can be used as a PDF

$$\int_0^{2\pi} \int_0^{\infty} rR(r) dr d\phi = A$$

$$p(r) = \frac{rR(r)}{A}$$

The diffusion profile is normalized, and can be directly used as a probability density function (or PDF for short). This is a useful property for Monte Carlo integration, as we’ll see in a bit.



Implementing Subsurface Scattering

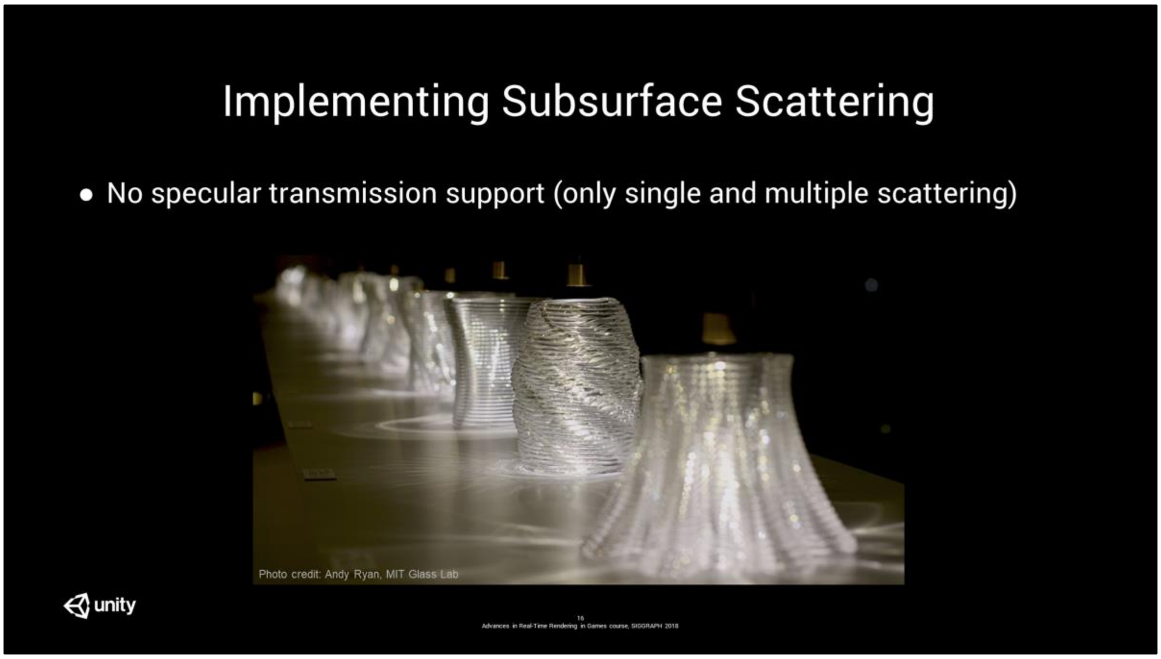
- A diffuse BRDF is is a far-field approximation of SSS
  - Use the formulation of the Disney Diffuse BRDF to define the diffuse transmission behavior [Burley 2015]
    - Not physically plausible, but better than assuming a Lambertian distribution

$$f_d(l,v) = \frac{A}{\pi} F_{dt}(l) F_{dt}(v) + f_{rr}$$
$$F_{dt}(x) = 1 - 0.5(1 - |n \cdot x|)^5$$

unity

A diffuse BRDF approximates SSS when the scattering distance is within the footprint of the pixel.
What this means is that both shading models should have the same visuals past a certain distance.

In order to achieve that, we use the formulation of the Disney Diffuse BRDF to define the diffuse transmission behavior at the dielectric boundary.
While it doesn’t match transmission defined by the GGX model, it’s still better than assuming a Lambertian distribution.



Implementing Subsurface Scattering

- No specular transmission support (only single and multiple scattering)

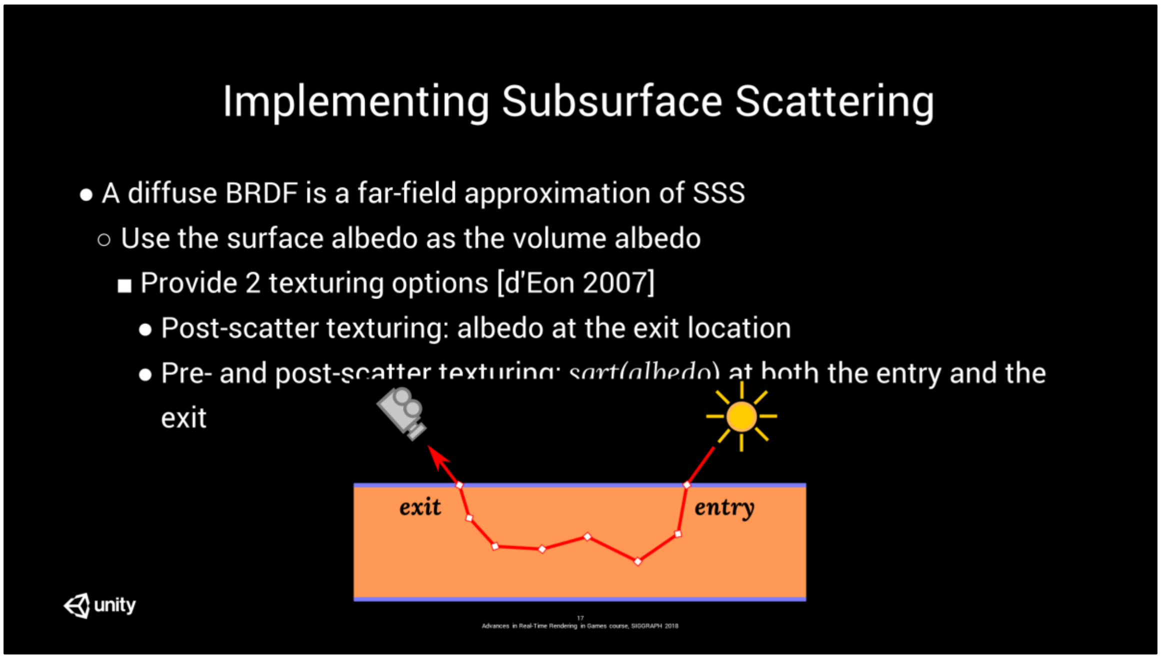
Photo credit: Andy Ryan, MIT Glass Lab

unity

16
Advances in Real-Time Rendering in Games course, SIGGRAPH 2018

We do not model specular transmission, and our model does not generalize to low-albedo materials like glass.

Note: transmission may be a confusing term. Specular transmission refers to the Fresnel transmission at a smooth dielectric boundary.
Diffuse transmission models transmission through a rough dielectric boundary, and typically assumes a high amount of multiple scattering. Often, it's just the Lambertian term.



Implementing Subsurface Scattering

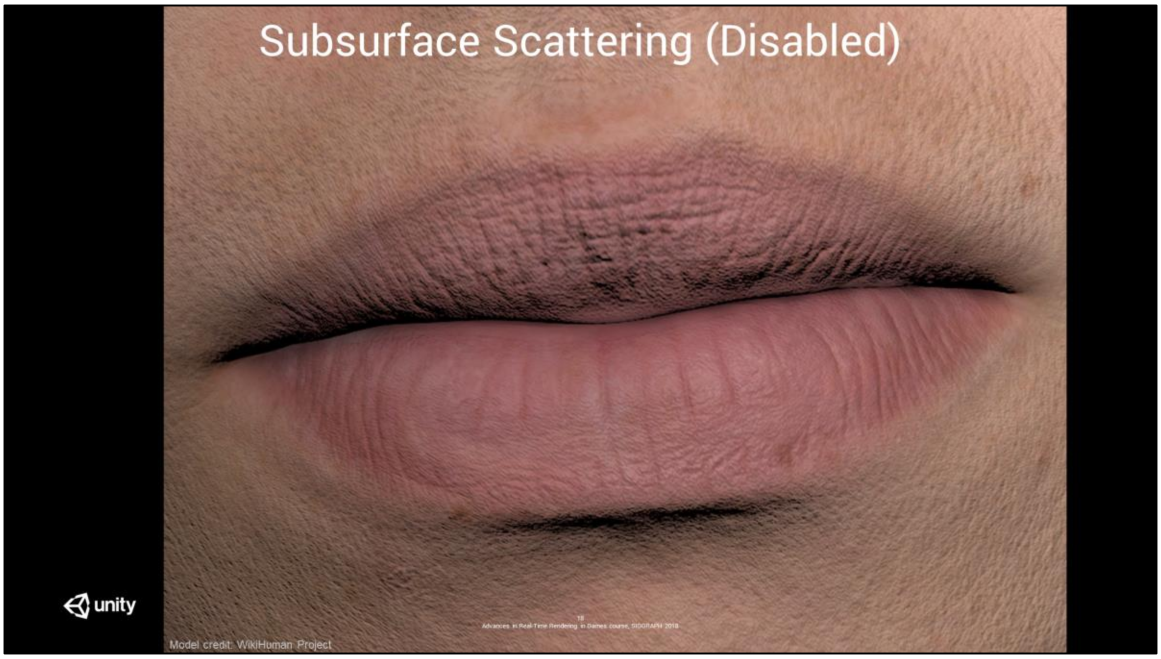
- A diffuse BRDF is a far-field approximation of SSS
  - Use the surface albedo as the volume albedo
    - Provide 2 texturing options [d'Eon 2007]
      - Post-scatter texturing: albedo at the exit location
      - Pre- and post-scatter texturing: *sqrt(albedo)* at both the entry and the exit

exit
entry
unity

To enforce visual consistency, we also directly use the surface albedo as the volume albedo.

We offer 2 albedo texturing options:

1. Post-scatter texturing should be used when the albedo texture already contains some color bleed due to SSS. That is the case for scans and photographs. In this mode, we only apply the albedo once, at the exit location.
2. Pre- and post-scatter texturing effectively blurs the albedo, which can result in a softer, more natural look, which is desirable in certain cases.



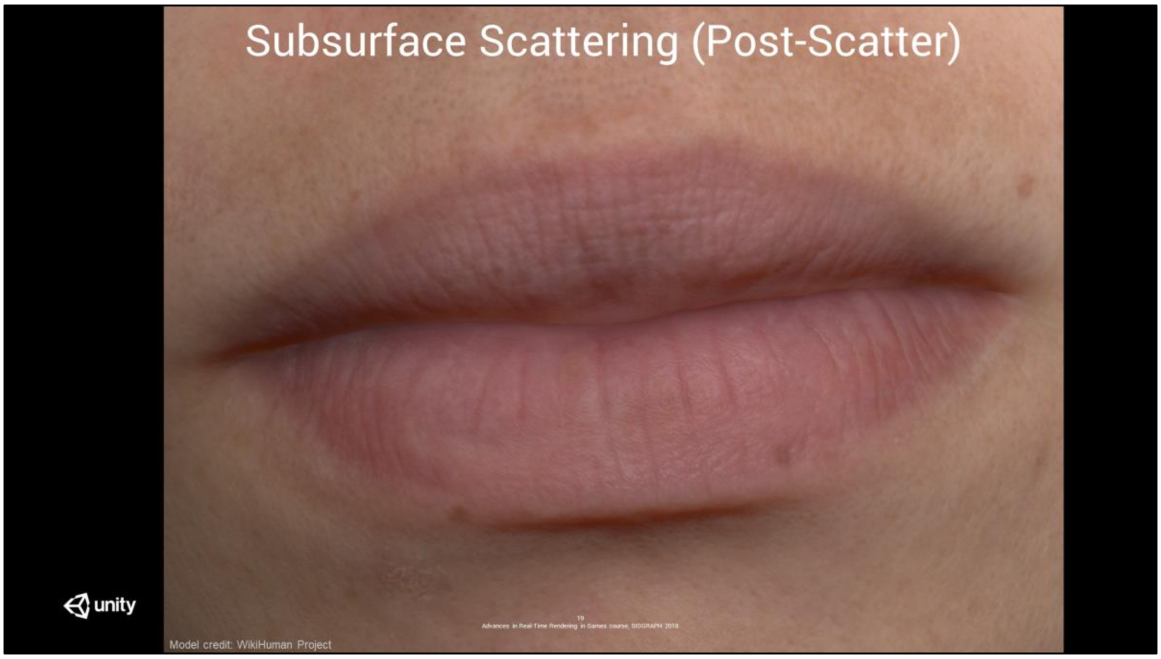
Subsurface Scattering (Disabled)

unity

Model credit: WikiHuman Project

Let’s look at some examples of both texturing modes.
We’ll start with the original model of Emily kindly shared by the WikiHuman project.

This picture clearly needs some SSS...



Subsurface Scattering (Post-Scatter)

Model credit: WikiHuman Project

unity

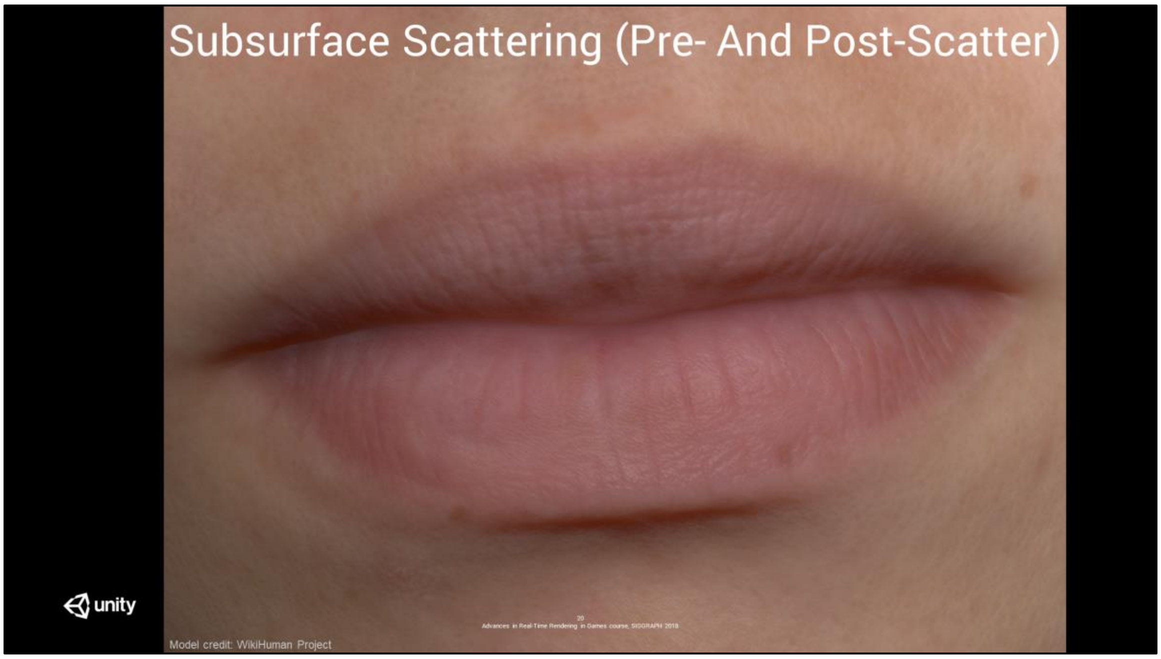
19

Advances in Real-Time Rendering in Games course, SIGGRAPH 2018

So here it is, with post-scatter texturing enabled.

(flip back and forth)

You may notice that the post-scatter option does a better job at preserving detail, as expected.
The difference is subtle, so I encourage you to look at the slides on a good display later.



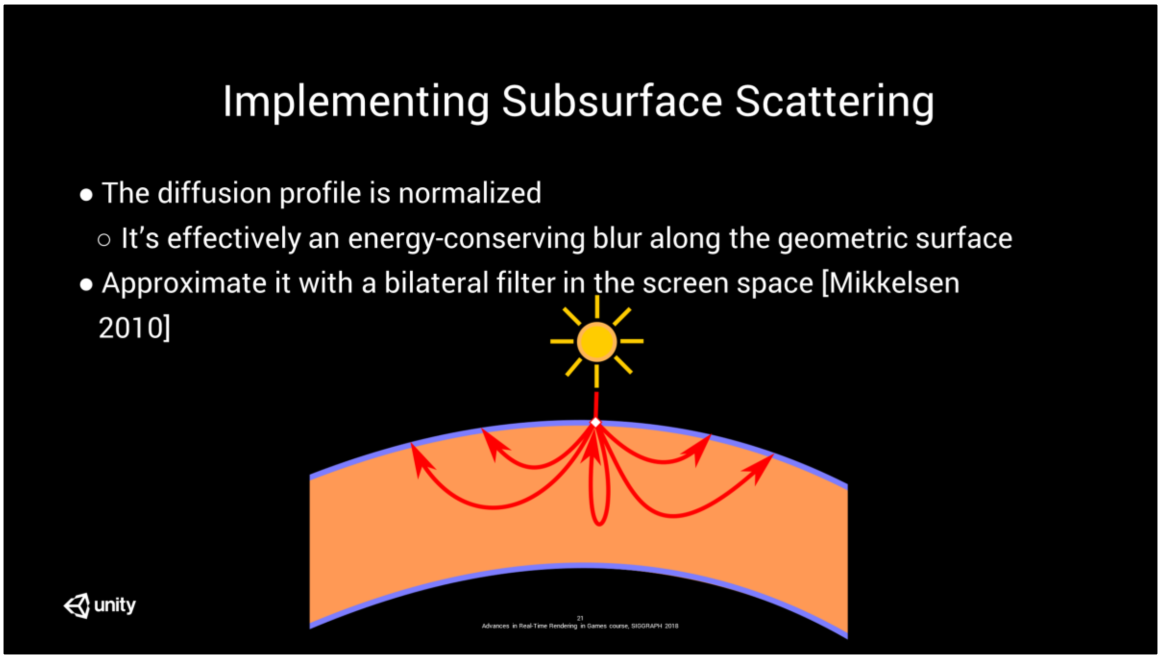
Subsurface Scattering (Pre- And Post-Scatter)

Model credit: WikiHuman Project

And for comparison, this is pre- and post-scatter.

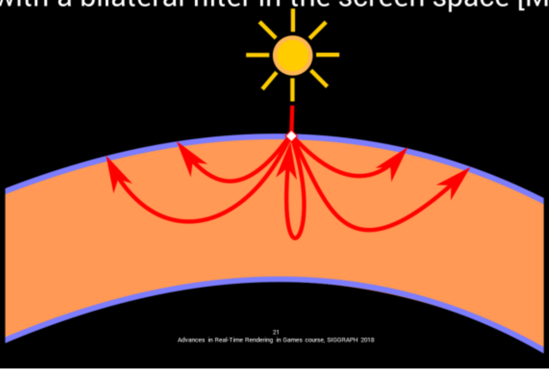
(flip back and forth)

You may notice that the post-scatter option does a better job at preserving detail, as expected.
The difference is subtle, so I encourage you to look at the slides on a good display later.



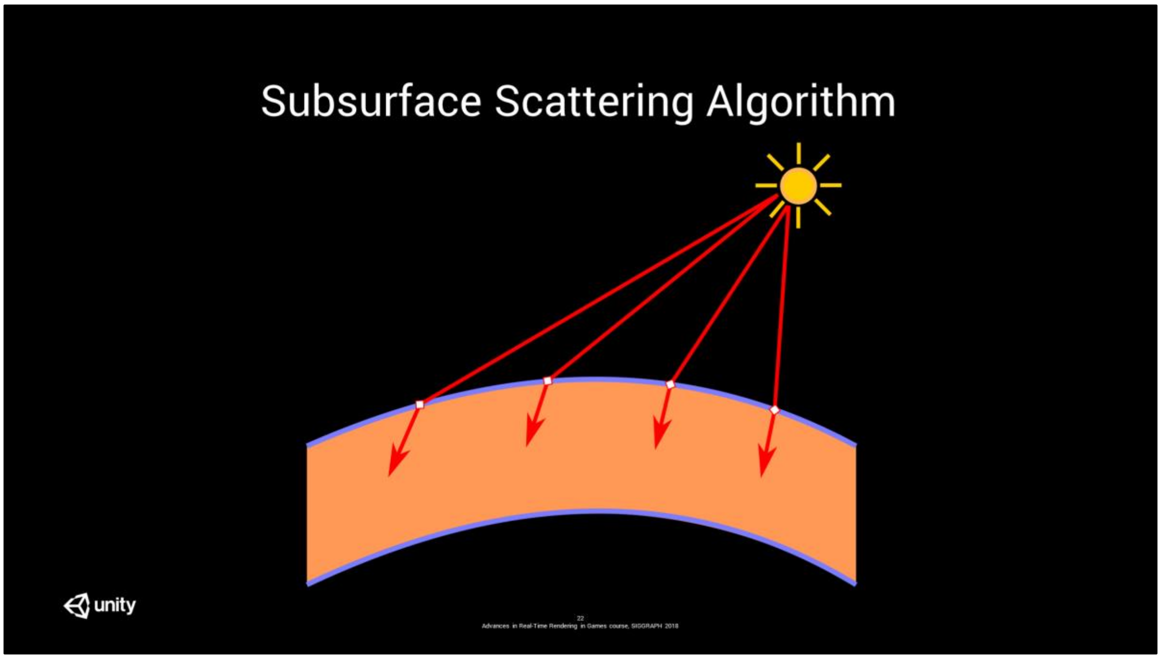
Implementing Subsurface Scattering

- The diffusion profile is normalized
  - It's effectively an energy-conserving blur along the geometric surface
- Approximate it with a bilateral filter in the screen space [Mikkelsen 2010]



Since Burley's diffusion profiles are normalized, SSS can be implemented as an energy-conserving blur filter.
You can imagine light energy being redistributed from the point of entry across the surrounding surface...

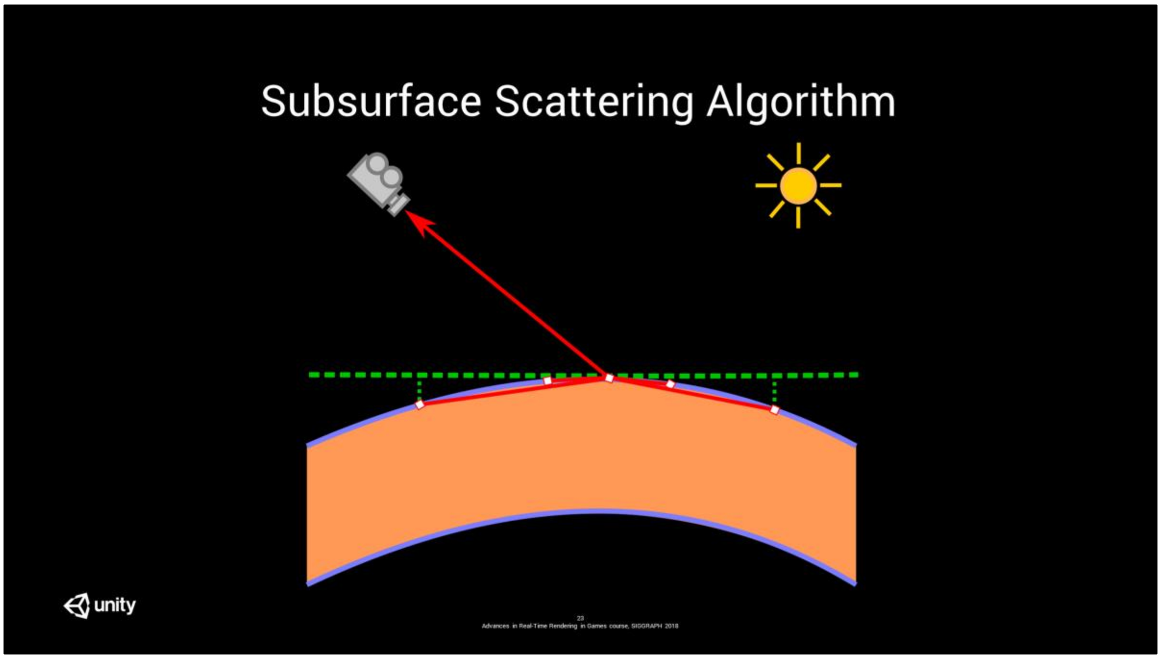
Similarly to the previous approaches, we perform convolution in the screen space, and use the depth buffer to account for the surface geometry.



Subsurface Scattering Algorithm

Let’s have a visual overview of the entire algorithm.

Lighting pass:
Compute incident radiance at the entry point of the surface
Perform diffuse transmission from the light direction into the surface
Apply the entry point albedo depending on the texturing mode
Record transmitted radiance in a render target



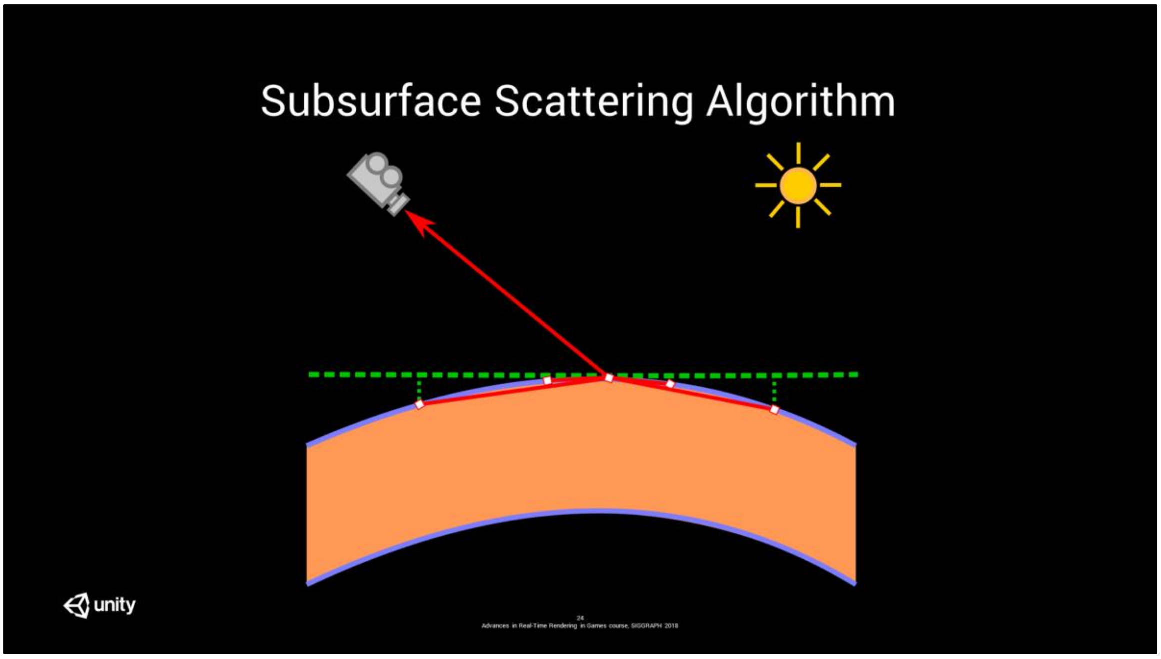
Subsurface Scattering Algorithm

SSS pass:

Perform bilateral filtering of the radiance buffer with the diffusion profile around the exit point

Apply the exit point albedo depending on the texturing mode

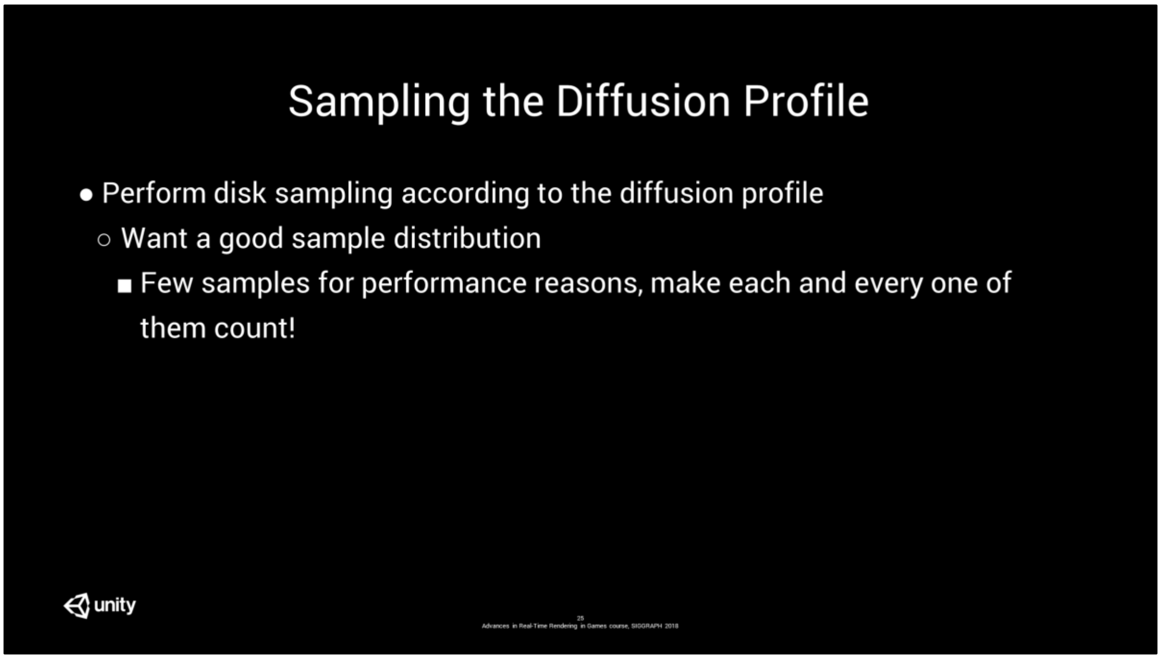
Perform diffuse transmission outside of the surface in the view direction



Subsurface Scattering Algorithm

Ideally, we should directly sample the surface of the object*, but that is too expensive for real-time applications.
Instead, we use a set of samples which we precompute offline, and (conceptually) place them on a disk, pictured here with a dashed green line.
As the picture demonstrates, this planar approximation is not necessarily a good match for the shape of the surface.
What’s arguably worse, since projection of samples from the disk onto the surface distorts distances, projected samples end up within a different distribution. We’ll have to do something about that, but first…

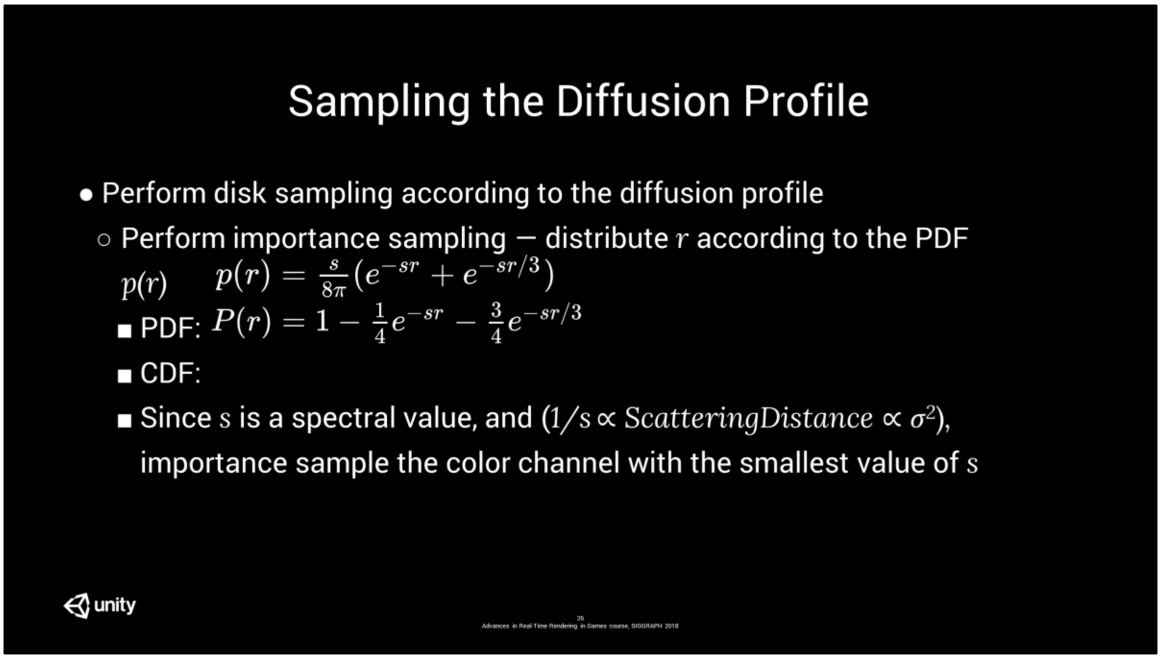
Note: that’s basically SSS using photon mapping.



Sampling the Diffusion Profile

- Perform disk sampling according to the diffusion profile
  - Want a good sample distribution
    - Few samples for performance reasons, make each and every one of them count!

... let’s talk about disk sampling.
We can only take a few samples for performance reasons, so we need to make each and every one of them count.

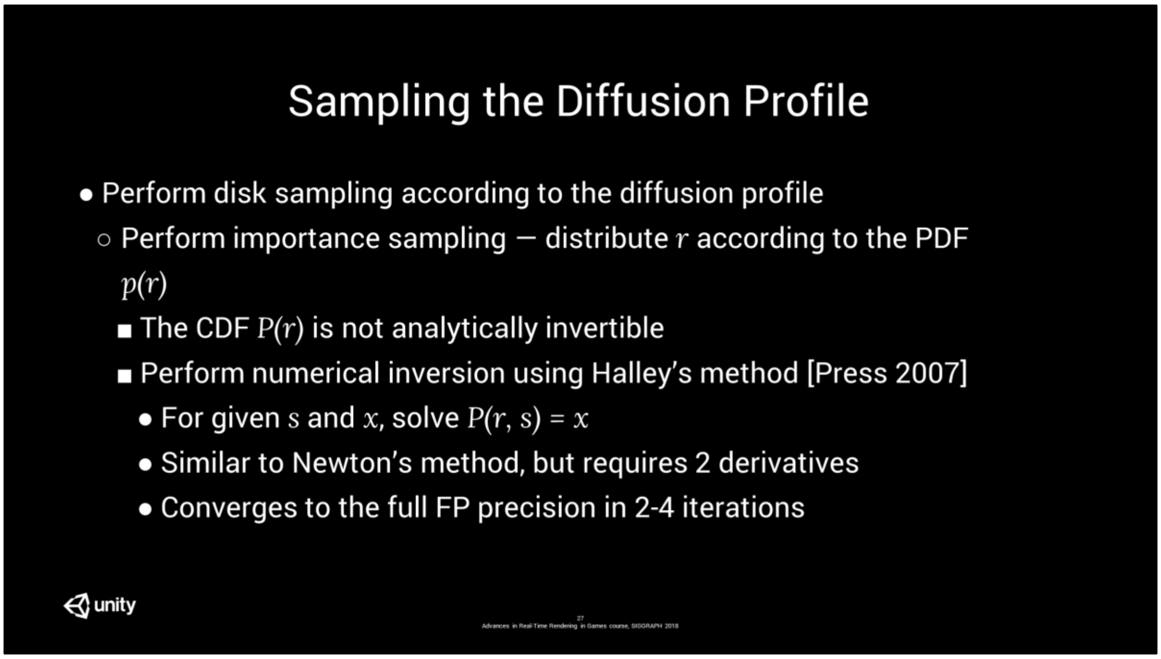


Sampling the Diffusion Profile

- Perform disk sampling according to the diffusion profile
  - Perform importance sampling — distribute r according to the PDF $p(r)$ $p(r) = \frac{s}{8\pi}(e^{-sr} + e^{-sr/3})$
    - PDF: $P(r) = 1 - \frac{1}{4}e^{-sr} - \frac{3}{4}e^{-sr/3}$
    - CDF:
    - Since s is a spectral value, and $(1/s \propto ScatteringDistance \propto \sigma^2)$, importance sample the color channel with the smallest value of s

Therefore, we importance sample radial distances - we distribute them according to the PDF of the diffusion profile.

The shape parameter **s** is a spectral value, and it is inversely proportional to the scattering distance, which itself is proportional to variance.
Since we want to achieve the best possible variance reduction, we choose to importance sample the color channel which corresponds to the largest scattering distance (for skin, it is the red channel).



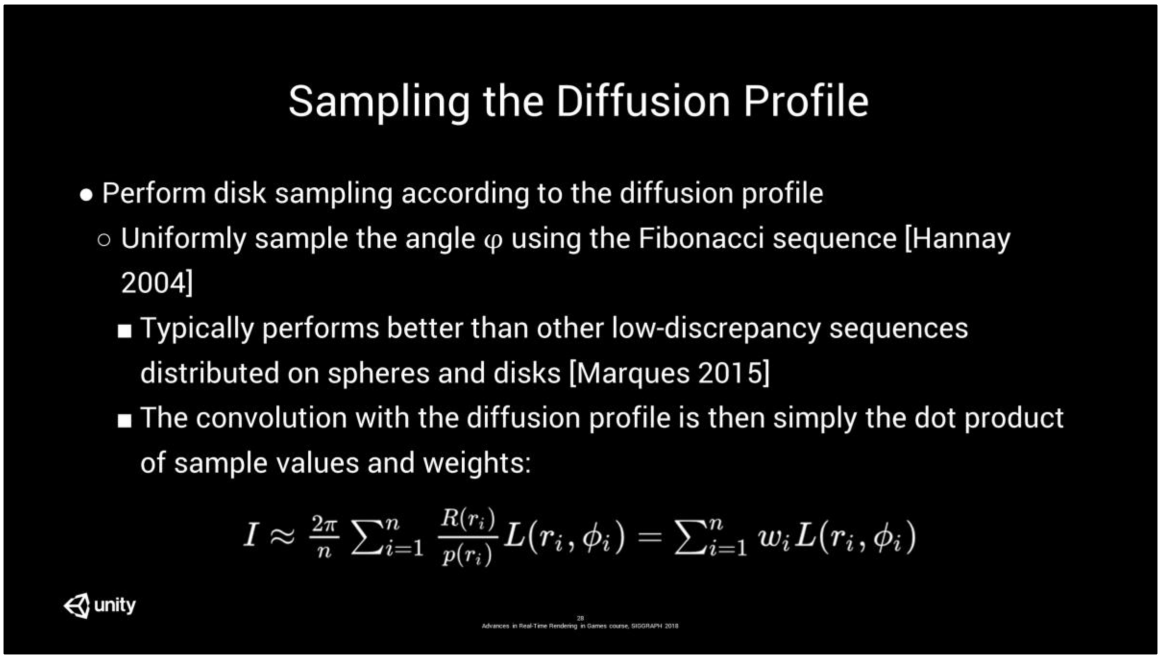
Sampling the Diffusion Profile

- Perform disk sampling according to the diffusion profile
  - Perform importance sampling — distribute r according to the PDF $p(r)$
    - The CDF $P(r)$ is not analytically invertible
    - Perform numerical inversion using Halley's method [Press 2007]
      - For given s and x, solve $P(r, s) = x$
      - Similar to Newton's method, but requires 2 derivatives
      - Converges to the full FP precision in 2-4 iterations

unity

For importance sampling, it is necessary to invert the cumulative distribution function (or CDF for short).
Unfortunately, the CDF is not analytically invertible, so we resort to numerical inversion using Halley's method, which is, nonetheless, quite efficient in practice.

Note: Halley's method is the second algorithm in the class of Householder's methods, after Newton's method.



Sampling the Diffusion Profile

- Perform disk sampling according to the diffusion profile
  - Uniformly sample the angle φ using the Fibonacci sequence [Hannay 2004]
    - Typically performs better than other low-discrepancy sequences distributed on spheres and disks [Marques 2015]
    - The convolution with the diffusion profile is then simply the dot product of sample values and weights:

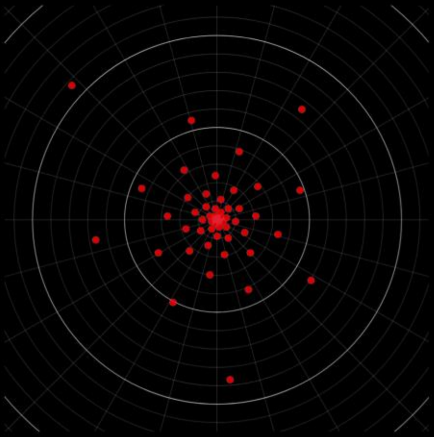
$$I \approx \frac{2\pi}{n} \sum_{i=1}^{n} \frac{R(r_i)}{p(r_i)} L(r_i, \phi_i) = \sum_{i=1}^{n} w_i L(r_i, \phi_i)$$

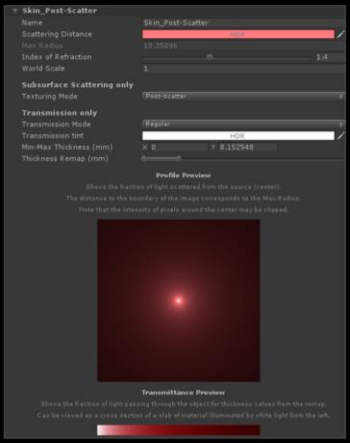
unity

Finally, we use the Fibonacci sequence to uniformly sample the polar angle. It yields a good sample distribution on spheres and disks, and is useful in many contexts, such as reflection probe filtering.

We use the Monte Carlo integration method to perform convolution across the disk. It simply boils down to the dot product of sample values and weights.

Sampling the Diffusion Profile





Skin_Post-Scatter

Name	Skin_Post-Scatter
Scattering Distance	HDR
Max Radius	10.35096
Index of Refraction	1.4
World Scale	1
Subsurface Scattering only	
Texturing Mode	Post-scatter
Transmission only	
Transmission Mode	Regular
Transmission tint	HDR
Min-Max Thickness (mm)	X 0 Y 8.152548
Thickness Remap (mm)	

Profile Preview

Shows the fraction of light scattered from the source (center).
The distance to the boundary of the image corresponds to the Max Radius.
Note that the intensity of pixels around the center may be clipped.

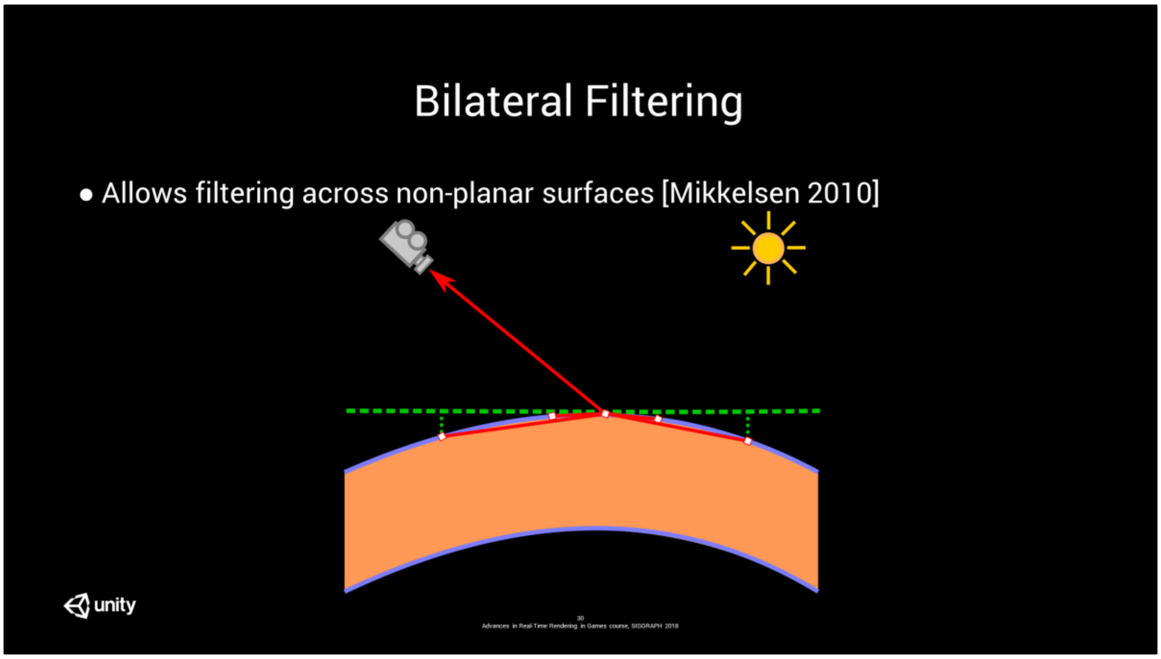
Transmittance Preview

Shows the fraction of light passing through the object for thickness values from the remap.
Can be viewed as a cross section of a slab of material illuminated by white light from the left.

unity

The importance sampling process results in a precomputed sample pattern which can look like this.

Notice that we sample more densely near the origin since most of the energy is concentrated there.

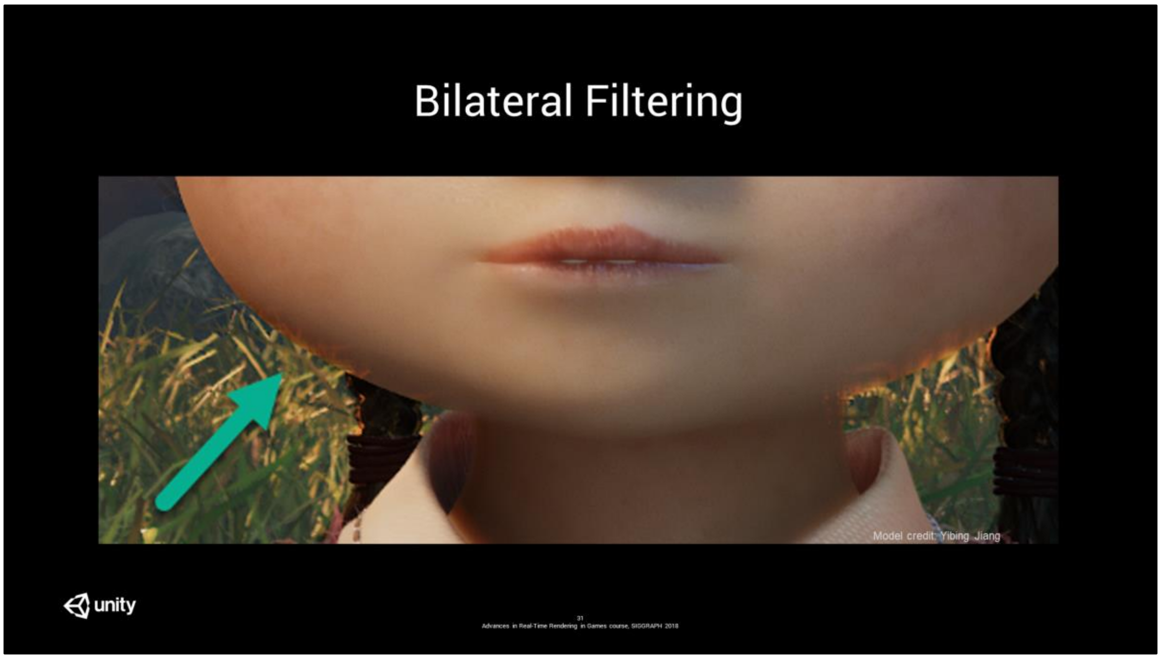


Bilateral Filtering

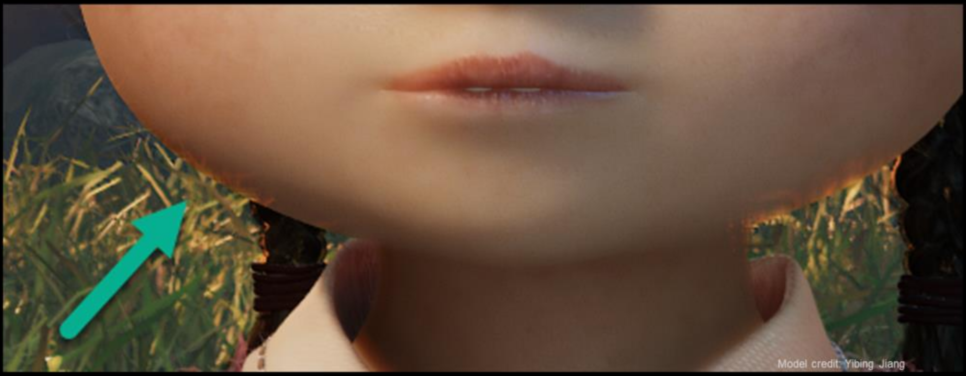
- Allows filtering across non-planar surfaces [Mikkelsen 2010]

Now that we know how to precompute the samples, let’s return to our planar approximation...

Since the surface geometry is not aligned to the disk, and we use precomputed samples, our only option is to re-weight the samples somehow. This process is called bilateral filtering.

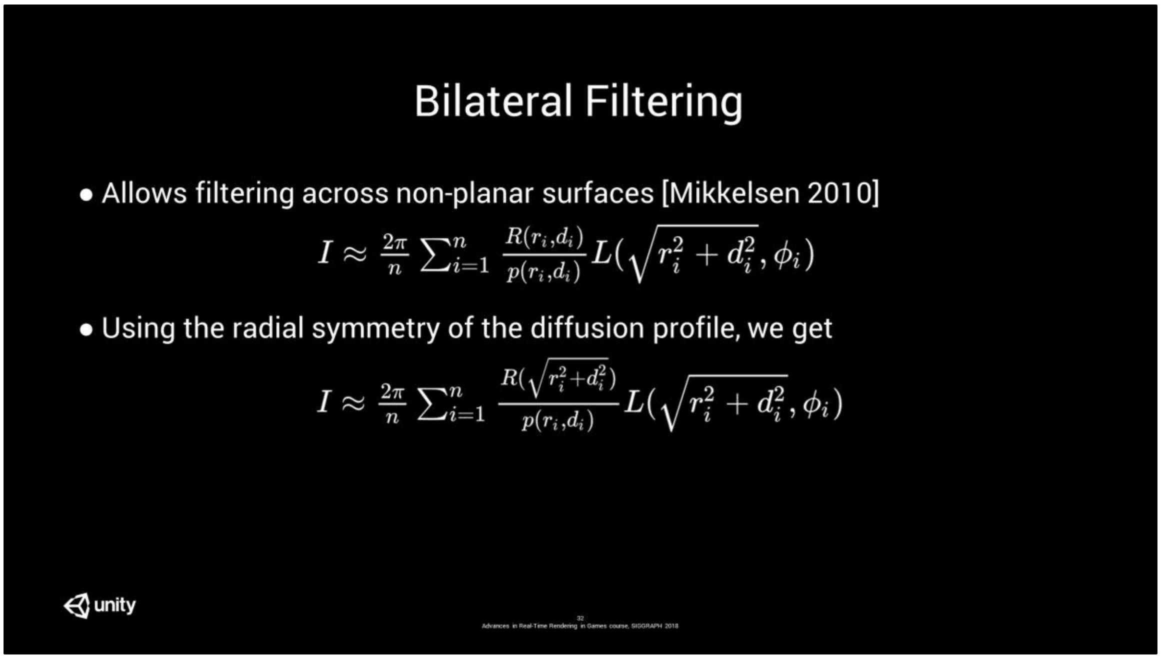


Bilateral Filtering



unity

Bilateral filtering makes convolution take depth into account.
Getting this right is very important, not just for a quality boost, but also to avoid the background bleeding onto the foreground, and vice versa.



Bilateral Filtering

- Allows filtering across non-planar surfaces [Mikkelsen 2010]

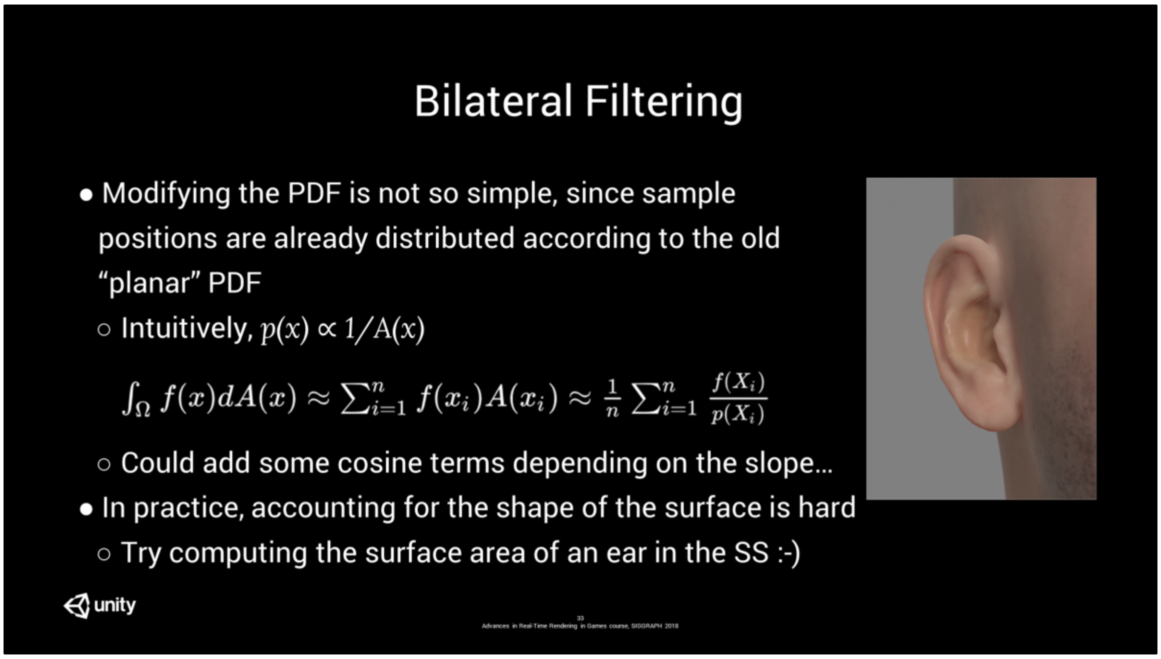
$$I \approx \frac{2\pi}{n} \sum_{i=1}^{n} \frac{R(r_i, d_i)}{p(r_i, d_i)} L\left(\sqrt{r_i^2 + d_i^2}, \phi_i\right)$$

- Using the radial symmetry of the diffusion profile, we get

$$I \approx \frac{2\pi}{n} \sum_{i=1}^{n} \frac{R\left(\sqrt{r_i^2 + d_i^2}\right)}{p(r_i, d_i)} L\left(\sqrt{r_i^2 + d_i^2}, \phi_i\right)$$

Using the Monte Carlo formulation of convolution, sample weights are defined as the ratio between the value of the function and the PDF.
Modifying the value of the function is easy - we just evaluate the profile using the actual Euclidean distance between the entry and the exit points.

Note: for the ease of exposition, the math assumes a right triangle and thus uses the Pythagorean theorem. Generally speaking, you should account for the perspective distortion as well [Mikkelsen 2010].



Bilateral Filtering

- Modifying the PDF is not so simple, since sample positions are already distributed according to the old "planar" PDF
  - Intuitively, $p(x) \propto 1/\mathrm{A}(x)$

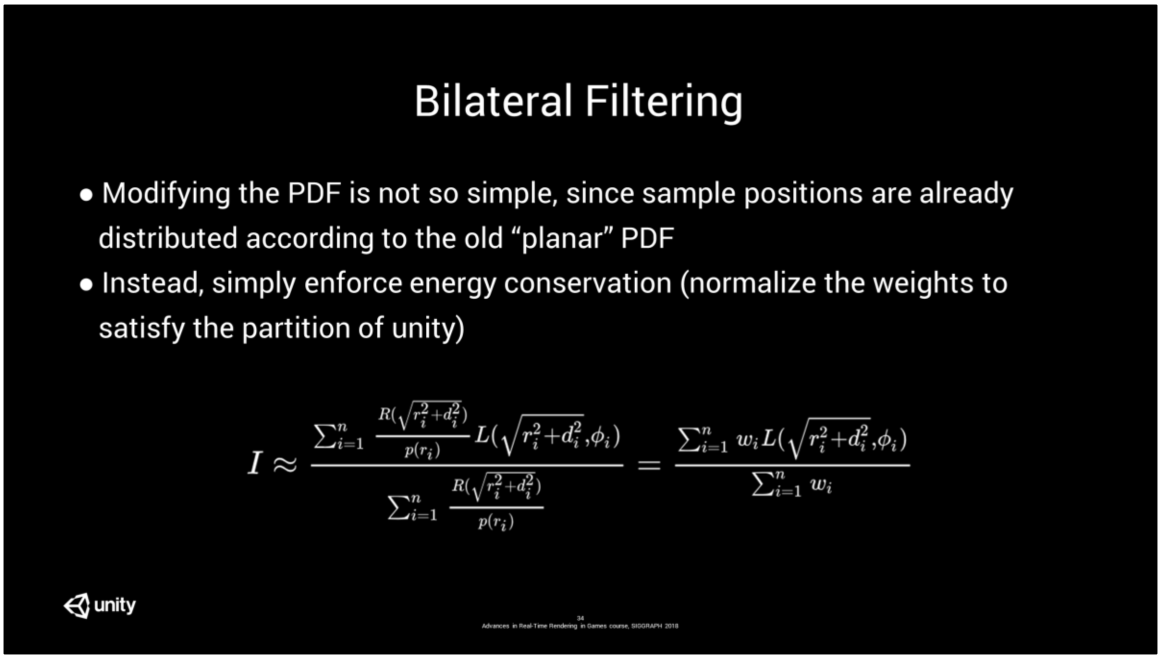
$$\int_\Omega f(x)dA(x) \approx \sum_{i=1}^{n} f(x_i)A(x_i) \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}$$

  - Could add some cosine terms depending on the slope...
- In practice, accounting for the shape of the surface is hard
  - Try computing the surface area of an ear in the SS :-)

Unfortunately, we cannot do the same for the PDF, since our sample positions are already distributed according to this old "planar" PDF.

If we make a connection between the formula for Monte Carlo integration and the quadrature formula for integration over area, we can see that the PDF value is inversely proportional to the area associated with each sample.

But how do you compute this area? It's possible to make certain assumptions about the surface and add some cosine factors to account for slopes...
However, solving this problem in a general and robust way is **hard**, especially due to the limited information in the screen space.

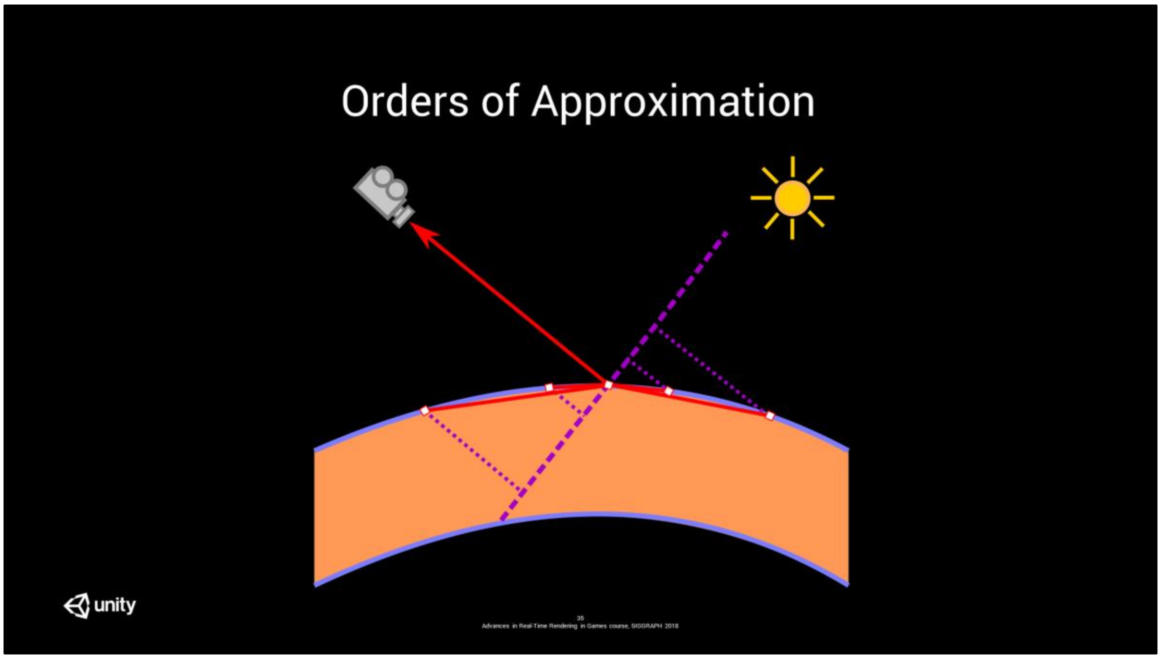


Bilateral Filtering

- Modifying the PDF is not so simple, since sample positions are already distributed according to the old "planar" PDF
- Instead, simply enforce energy conservation (normalize the weights to satisfy the partition of unity)

$$I \approx \frac{\sum_{i=1}^{n} \frac{R(\sqrt{r_i^2+d_i^2})}{p(r_i)} L(\sqrt{r_i^2+d_i^2},\phi_i)}{\sum_{i=1}^{n} \frac{R(\sqrt{r_i^2+d_i^2})}{p(r_i)}} = \frac{\sum_{i=1}^{n} w_i L(\sqrt{r_i^2+d_i^2},\phi_i)}{\sum_{i=1}^{n} w_i}$$

Instead, we can simply utilize the fact that our filter is meant to be energy conserving, and normalize the weights to sum up to 1.

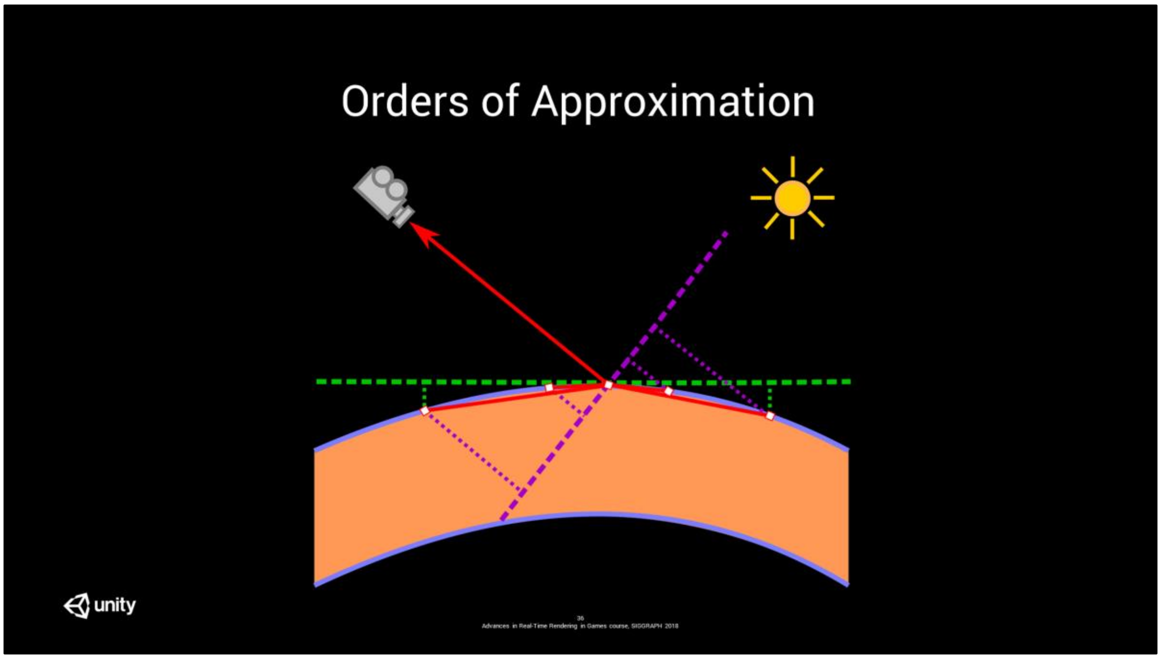


Orders of Approximation

Now that we know how to perform bilateral filtering across the disk, let’s consider how to place and orient this disk.
The disk is naturally placed at the intersection of the camera ray with the geometry, in the world or the camera space.

But what about the orientation of the disk? We have several options.

The 0 order approximation is to align the disk parallel to the screen plane, which directly corresponds to a disk in the screen space.
It’s simple and fast, but can result in poor sample distribution for geometry at oblique angles.

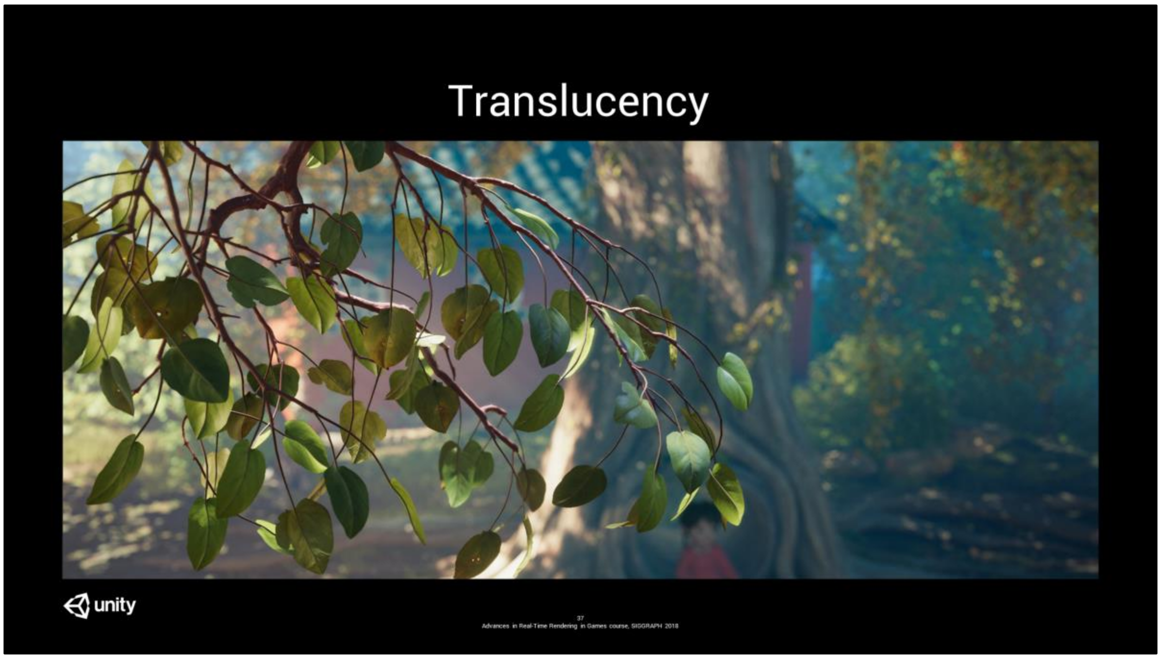


Orders of Approximation

A better solution is to align the disk with the tangent plane of the neighbourhood of the surface point (*1st order approximation*).

This can be challenging because the G-Buffer typically does not contain the interpolated vertex normal.
And, unfortunately, using the shading normal can result in artifacts, as it often has little in common with the geometry around the surface point, and can even be back-facing.

It's worth noting that even the naive method performs quite well in practice.



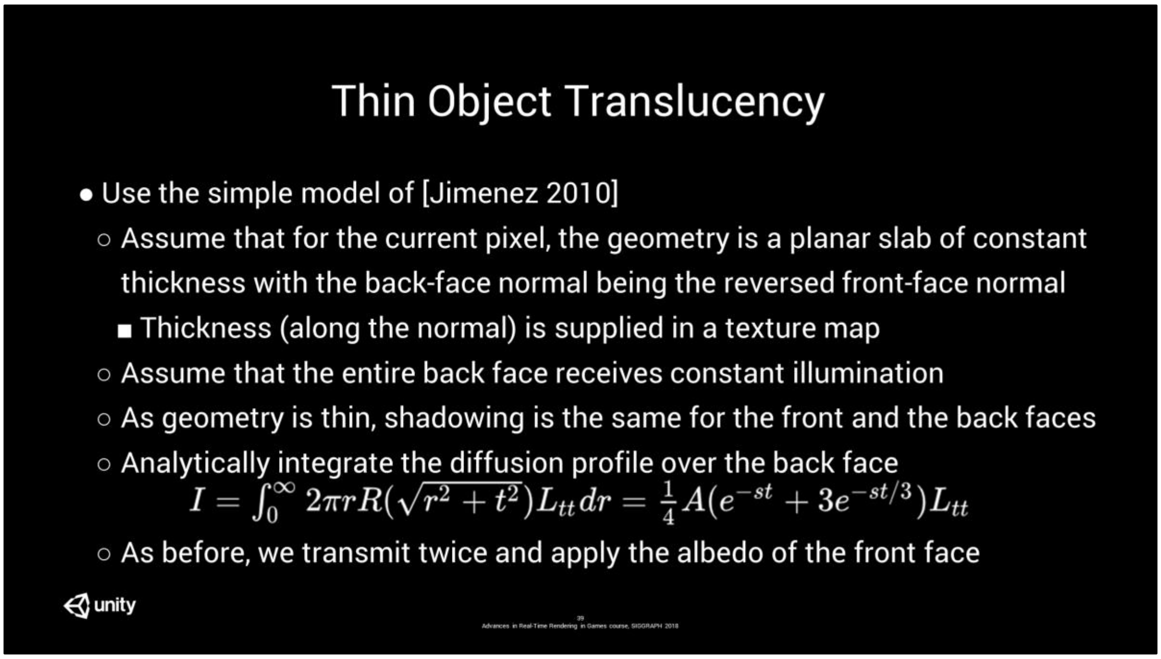
Translucency

SSS is also responsible for the translucent look of back-lit objects.

Translucency

While the underlying physical process is exactly the same, for efficiency reasons, we handle this effect in a more simple way.

We implemented 2 different approaches.
The first one only works with thin objects (and is commonly used for foliage), and the second one attempts to handle the more general translucency case.
The primary difference between the two is geometric thickness, which forces us to handle shadows in two different ways.

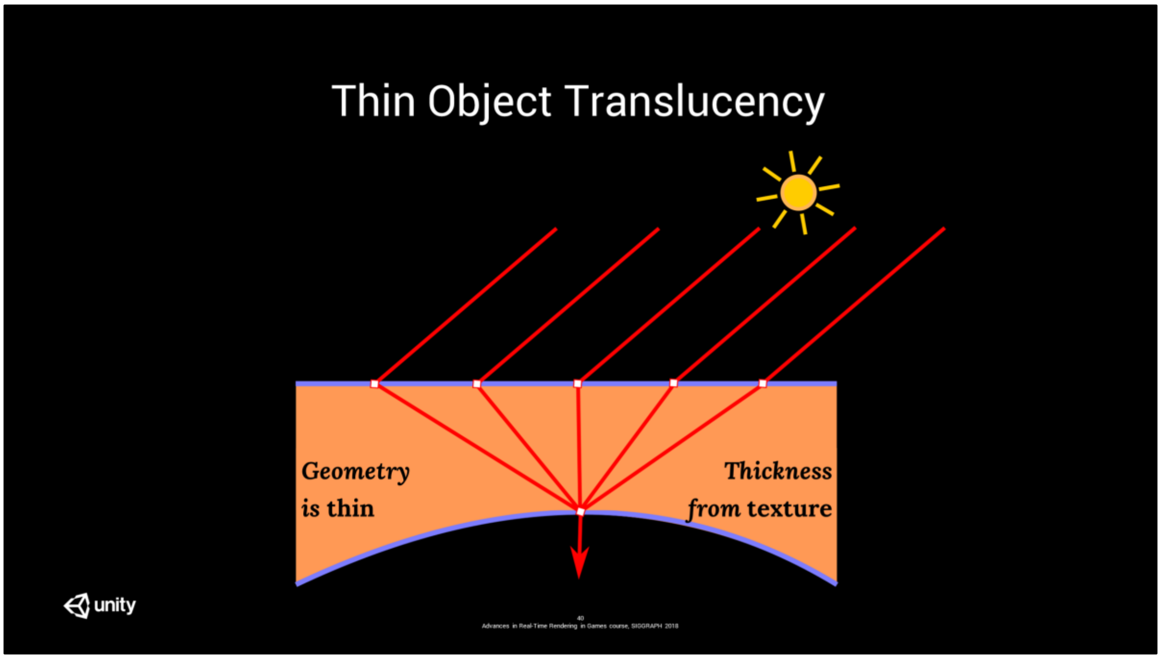


Thin Object Translucency

- Use the simple model of [Jimenez 2010]
  - Assume that for the current pixel, the geometry is a planar slab of constant thickness with the back-face normal being the reversed front-face normal
    - Thickness (along the normal) is supplied in a texture map
  - Assume that the entire back face receives constant illumination
  - As geometry is thin, shadowing is the same for the front and the back faces
  - Analytically integrate the diffusion profile over the back face

$$I = \int_0^\infty 2\pi r R(\sqrt{r^2 + t^2}) L_{tt} dr = \frac{1}{4} A(e^{-st} + 3e^{-st/3}) L_{tt}$$

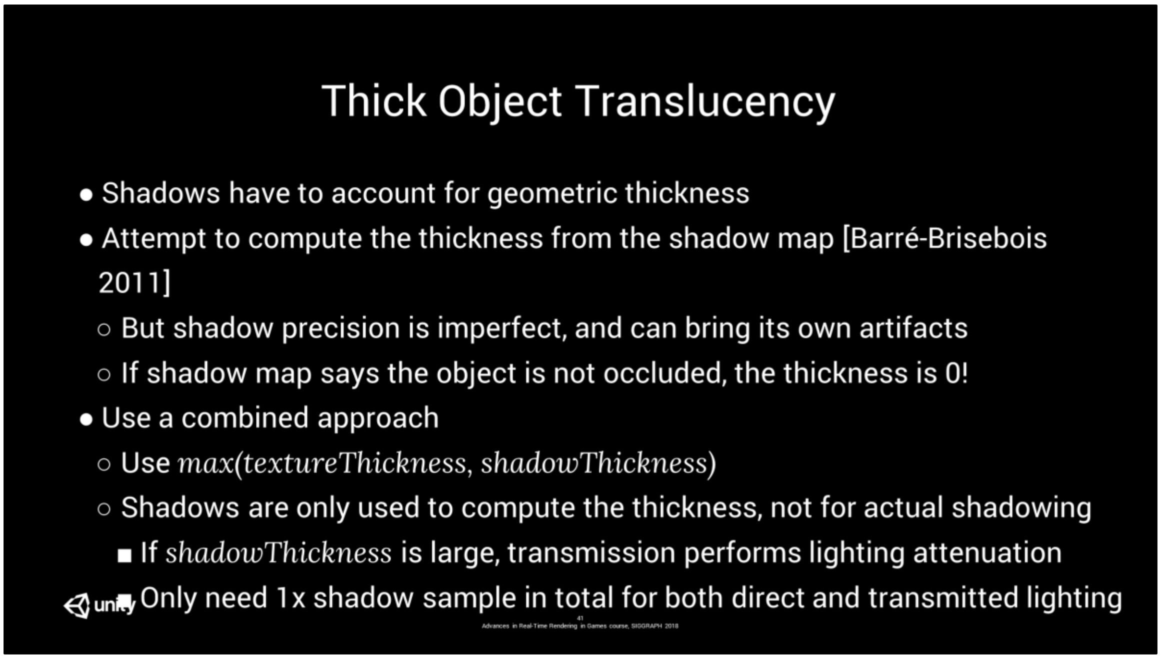
  - As before, we transmit twice and apply the albedo of the front face



Thin Object Translucency

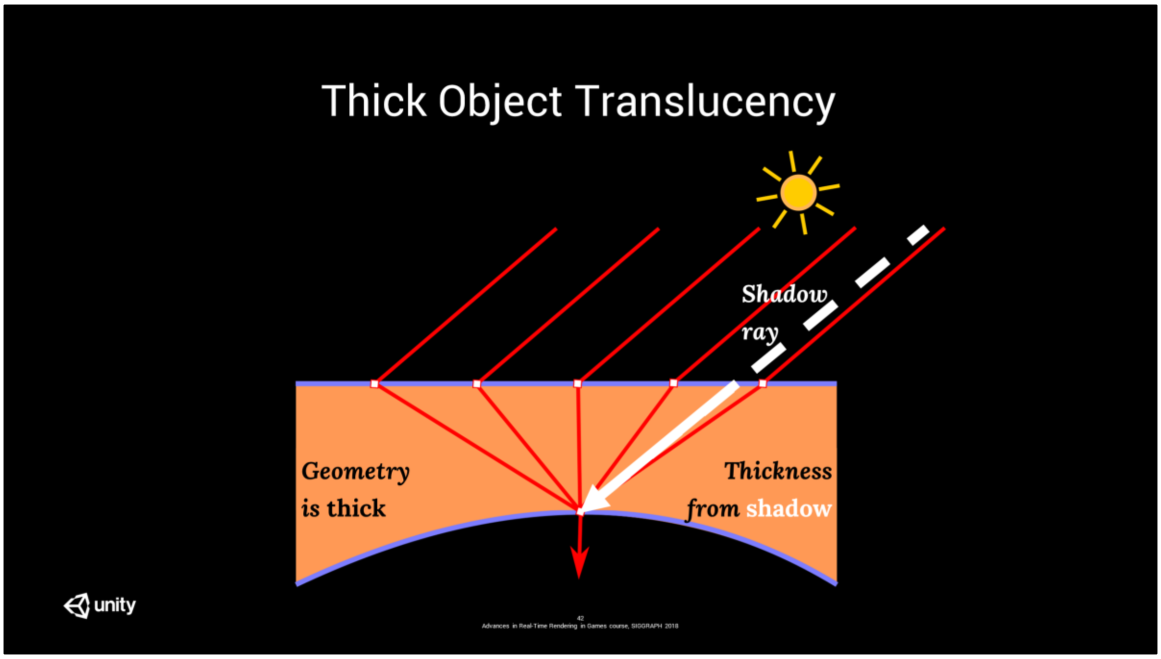
unity

For thin object translucency, we use the simple model proposed by Jorge Jimenez.
We assume that for the current pixel, the geometry is a planar slab of constant thickness with the back-face normal being the reversed front-face normal.
Thickness is provided in an artist-authored texture map.
Additionally, we assume that the entire back face receives constant illumination.
As geometry itself is thin, shadowing is the same for the front and the back faces, so we can share a single shadow map fetch.
Given this simplified setup, it's possible to analytically integrate the contribution of the diffusion profile over the back face.
And as before, we transmit twice and apply the albedo of the front face.



Thick Object Translucency

- Shadows have to account for geometric thickness
- Attempt to compute the thickness from the shadow map [Barré-Brisebois 2011]
  - But shadow precision is imperfect, and can bring its own artifacts
  - If shadow map says the object is not occluded, the thickness is 0!
- Use a combined approach
  - Use $max(textureThickness, shadowThickness)$
  - Shadows are only used to compute the thickness, not for actual shadowing
    - If $shadowThickness$ is large, transmission performs lighting attenuation
    - Only need 1x shadow sample in total for both direct and transmitted lighting

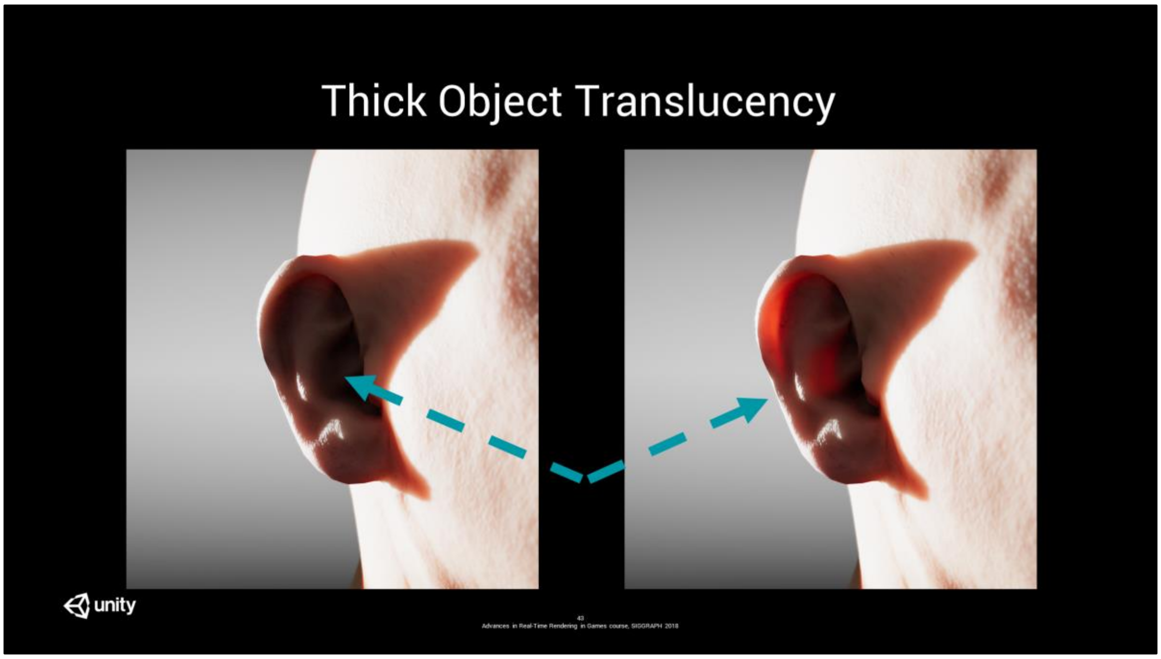


Thick Object Translucency

For thicker objects, reusing the shadowing status of the front face obviously does not work (since the back face may shadow the front face).

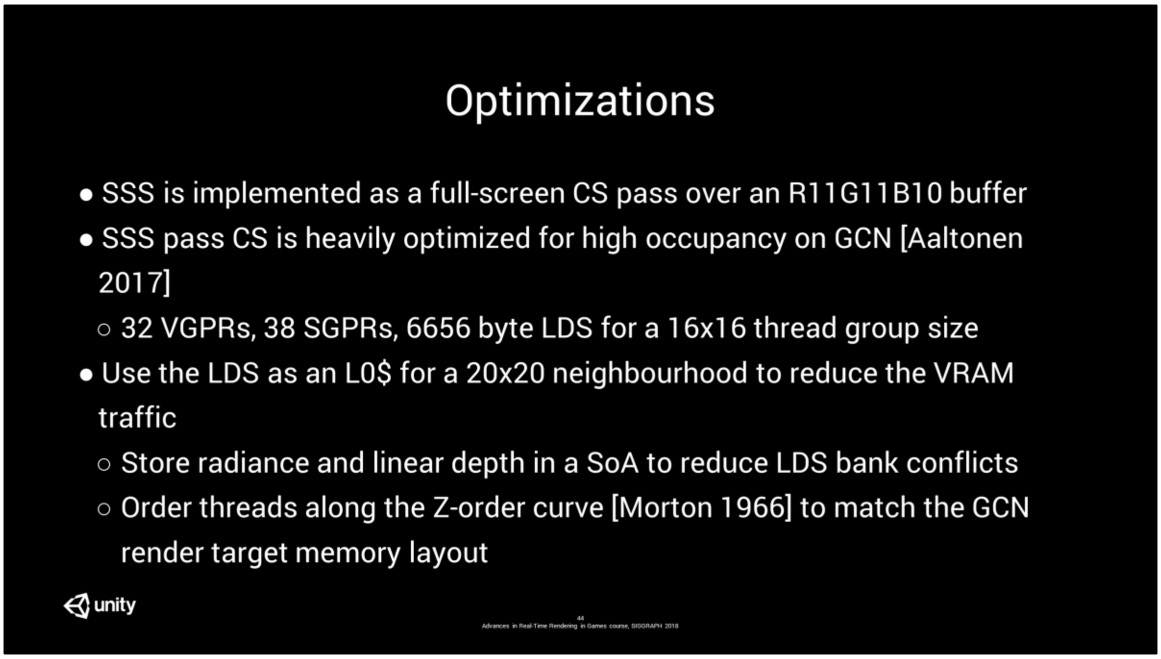
Initially, we attempted to compute thickness at runtime solely using the distance to the closest occluder given by the shadow map.
It quickly became apparent that this approach does not work well for fine geometric features due to the limited precision of shadow maps.

Instead, we opted for a combined approach. We compute thickness using both methods, and take the maximum value, which gives an artist an opportunity to work around shadow mapping issues using the “baked” thickness texture.



Thick Object Translucency

The method admittedly requires some tweaking, but with some effort it is possible to achieve plausible results.


Optimizations

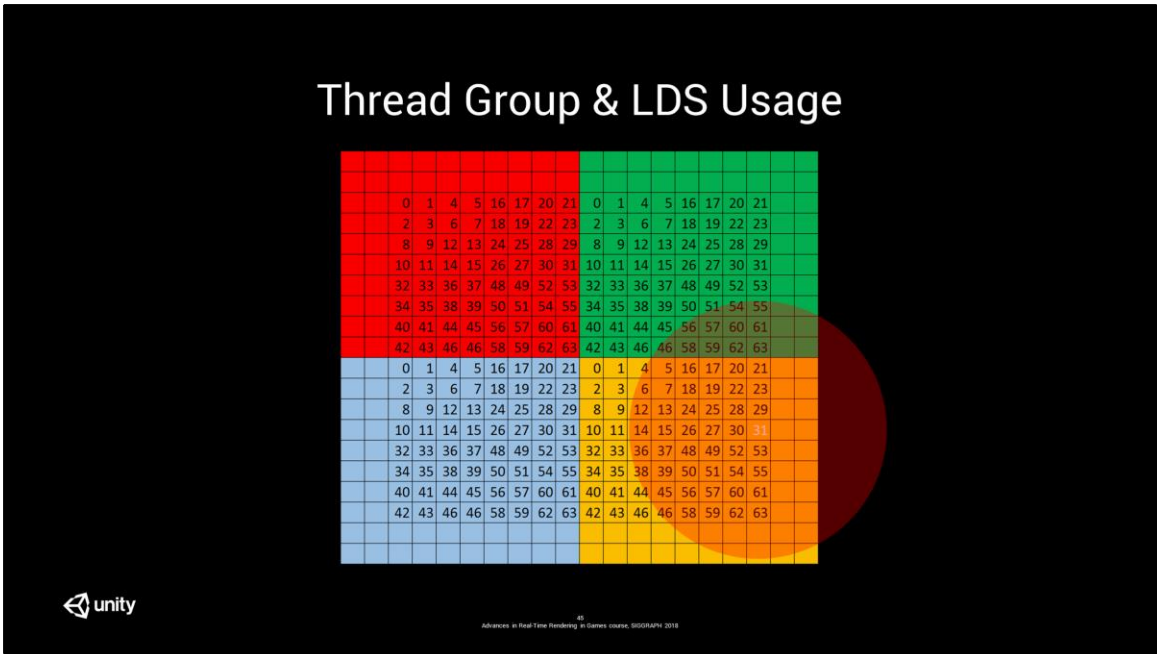
- SSS is implemented as a full-screen CS pass over an R11G11B10 buffer
- SSS pass CS is heavily optimized for high occupancy on GCN [Aaltonen 2017]
  - 32 VGPRs, 38 SGPRs, 6656 byte LDS for a 16x16 thread group size
- Use the LDS as an L0$ for a 20x20 neighbourhood to reduce the VRAM traffic
  - Store radiance and linear depth in a SoA to reduce LDS bank conflicts
  - Order threads along the Z-order curve [Morton 1966] to match the GCN render target memory layout

unity

A few words about implementation details, and how to make it efficient…

As I mentioned earlier, we importance sample offline, and use a set of precomputed samples at runtime.

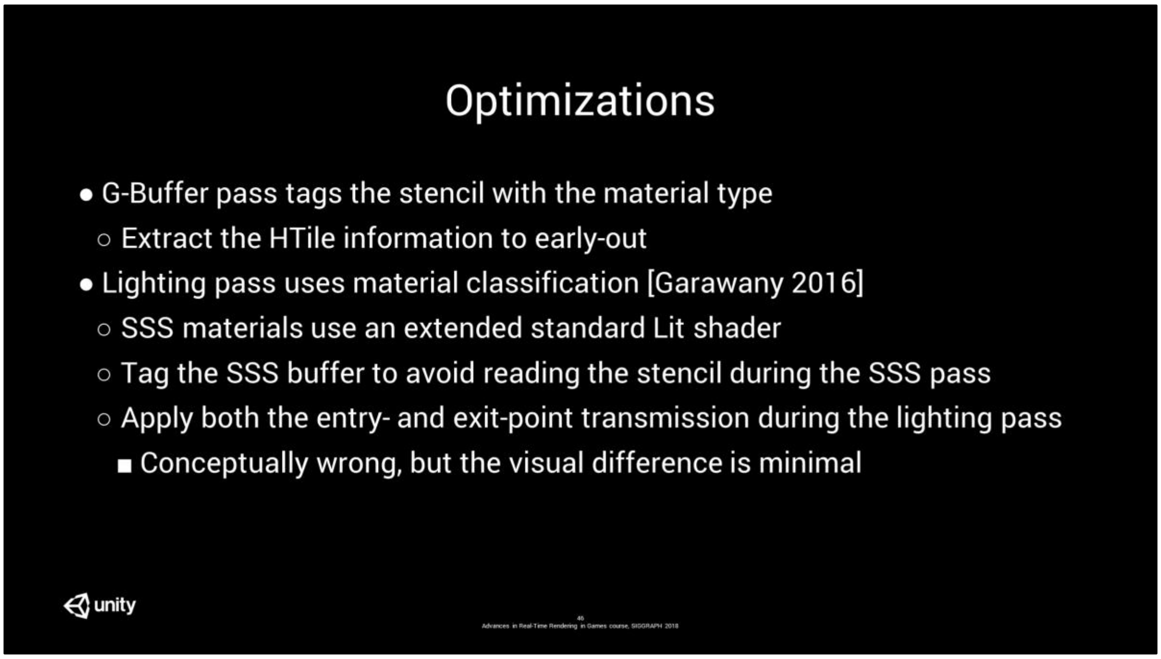
The SSS pass itself is implemented as a full-screen compute shader. It is bandwidth-heavy and makes heavy use of LDS to reduce off-chip memory traffic.



Thread Group & LDS Usage

0	1	4	5	16	17	20	21	0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23	2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29	8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31	10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53	32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55	34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61	40	41	44	45	56	57	60	61
42	43	46	46	58	59	62	63	42	43	46	46	58	59	62	63
0	1	4	5	16	17	20	21	0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23	2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29	8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31	10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53	32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55	34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61	40	41	44	45	56	57	60	61
42	43	46	46	58	59	62	63	42	43	46	46	58	59	62	63

The thread group (shown as numbers) is composed of 4 wavefronts, with individual threads ordered along the Z-order curve for improved data locality.
The LDS cache (shown as color blocks) contains radiance and linear depth values, and has a 2 texel border so that each pixel has at least a small cached neighbourhood.

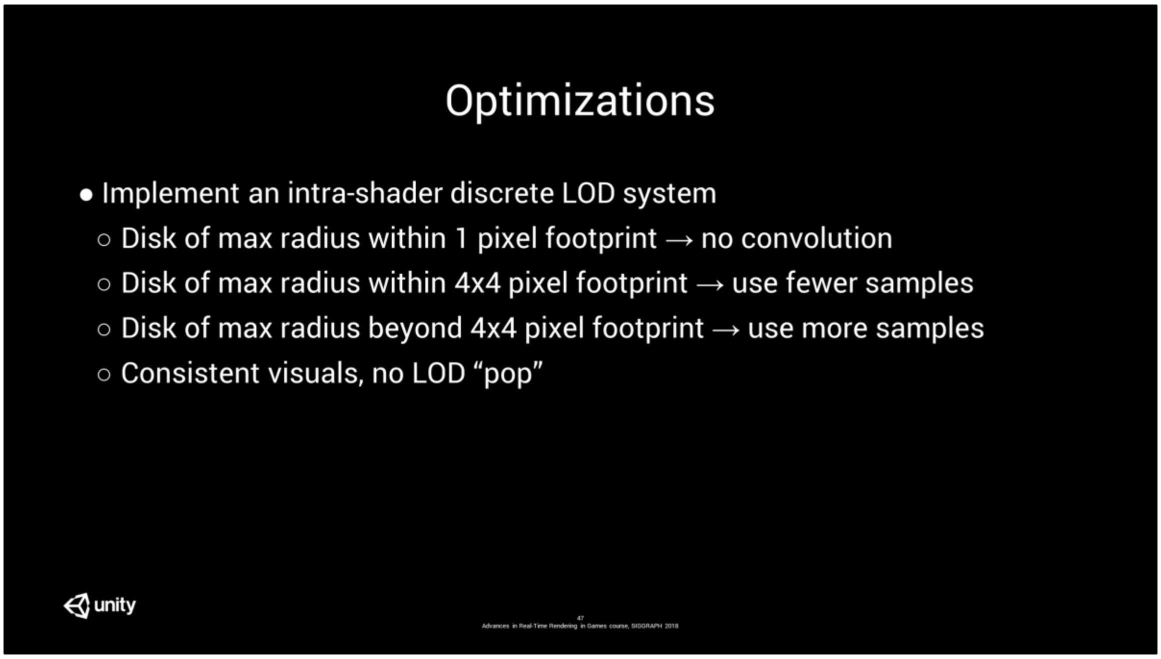


Optimizations

- G-Buffer pass tags the stencil with the material type
  - Extract the HTile information to early-out
- Lighting pass uses material classification [Garawany 2016]
  - SSS materials use an extended standard Lit shader
  - Tag the SSS buffer to avoid reading the stencil during the SSS pass
  - Apply both the entry- and exit-point transmission during the lighting pass
    - Conceptually wrong, but the visual difference is minimal

unity

We tag the stencil with the material type during the G-Buffer pass. This allows us to create a hierarchical stencil representation, and discard whole pixel tiles during the SSS pass.
During the lighting pass, we also tag the subsurface lighting buffer in order to avoid performing per-pixel stencil test during the SSS pass.
We also evaluate both transmission events during the lighting pass. While this is conceptually wrong, the visual difference is very small, and it allows us to avoid reading the normal buffer during the SSS pass.

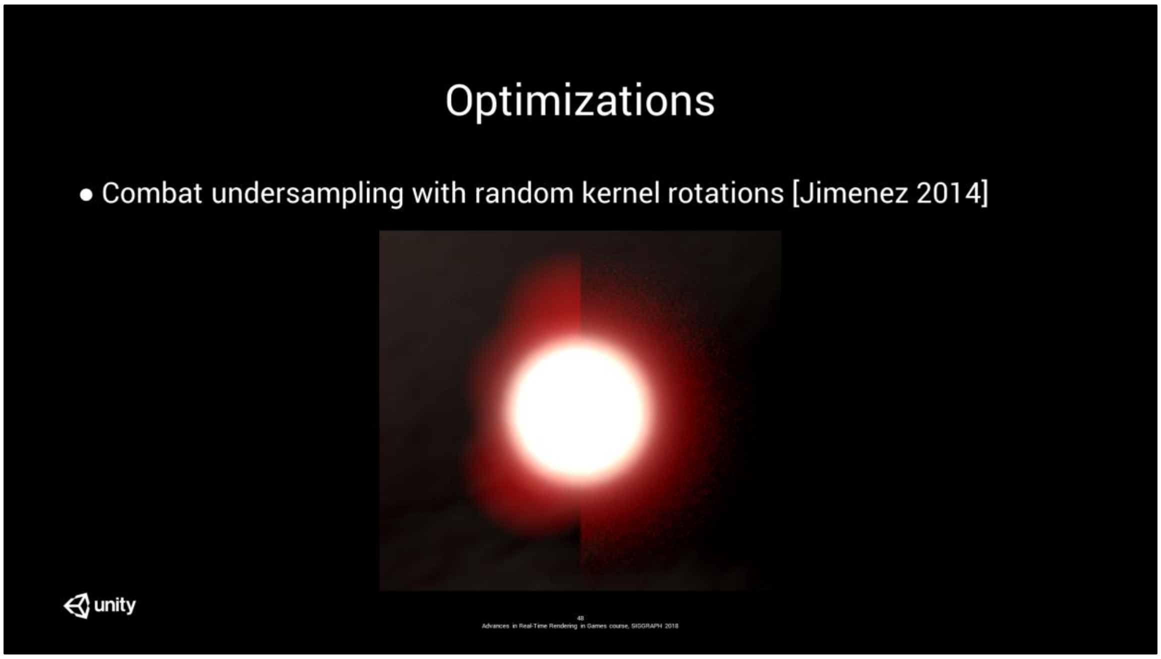


Optimizations

- Implement an intra-shader discrete LOD system
  - Disk of max radius within 1 pixel footprint → no convolution
  - Disk of max radius within 4x4 pixel footprint → use fewer samples
  - Disk of max radius beyond 4x4 pixel footprint → use more samples
  - Consistent visuals, no LOD “pop”

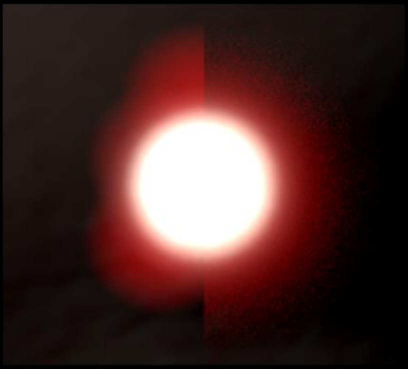
We also implemented a basic LOD system.

We change the number of samples depending on the screen-space footprint of the filter in 3 discrete steps: we disable filtering for sub-pixel sized disks, we use 21 samples for medium sizes, and 55 otherwise.
Visuals remain consistent, and LOD transitions are invisible.

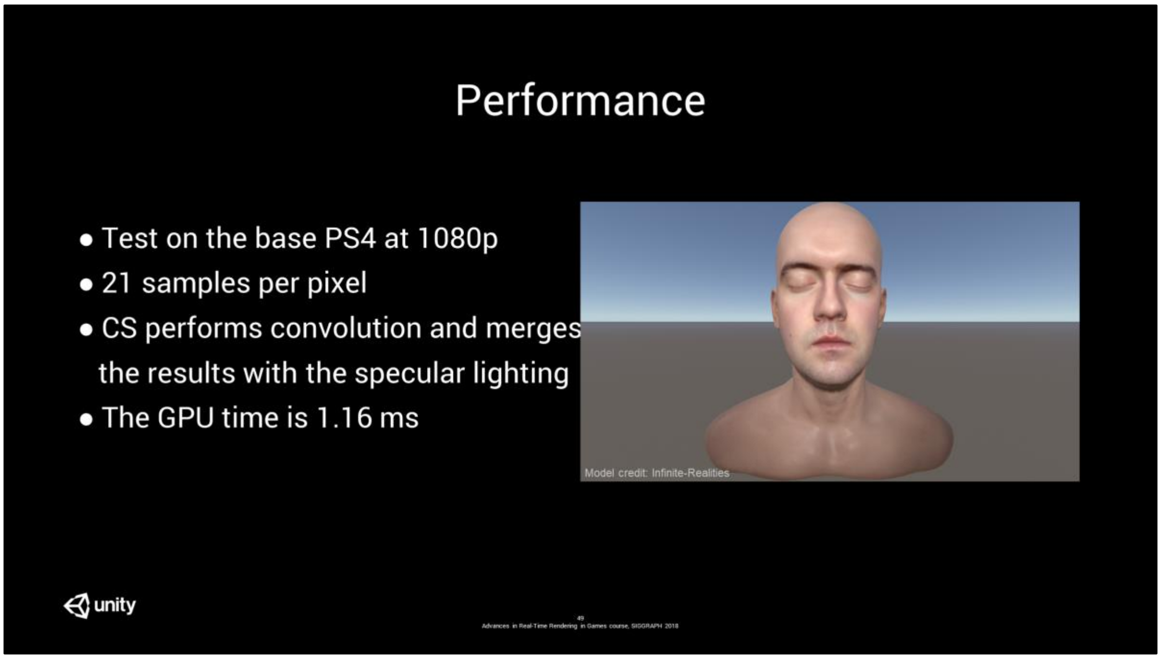


Optimizations

- Combat undersampling with random kernel rotations [Jimenez 2014]



We also found it important to perform random per-pixel rotations of the sample distribution. This allows us to trade structured undersampling artifacts for less objectionable noise.



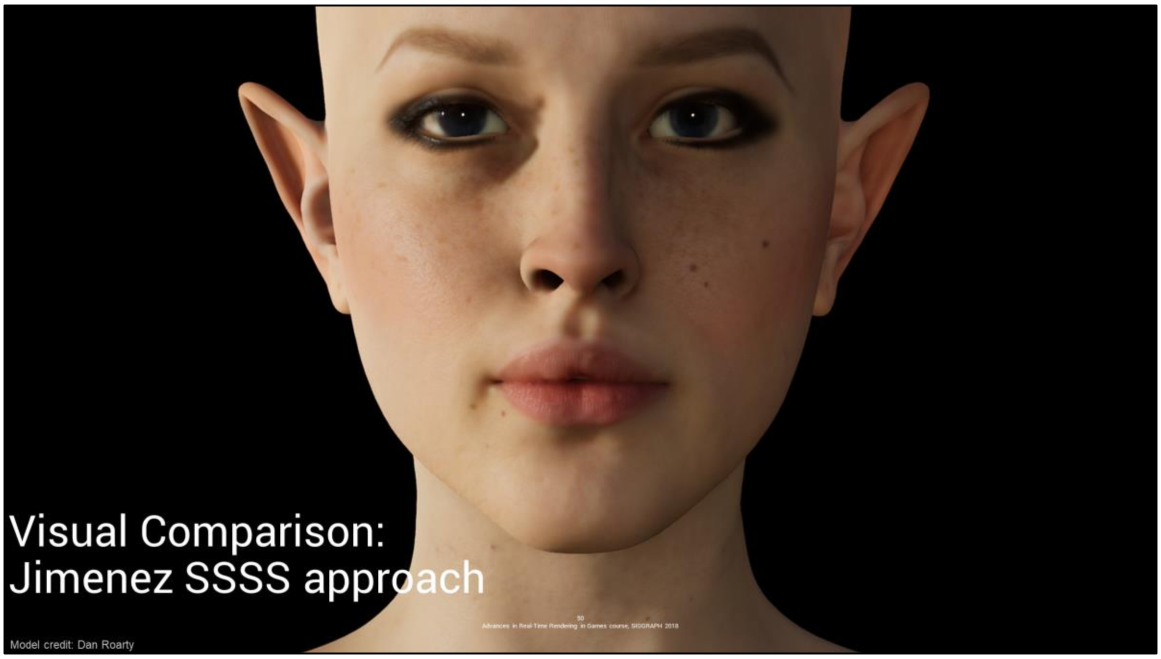
Performance

- Test on the base PS4 at 1080p
- 21 samples per pixel
- CS performs convolution and merges the results with the specular lighting
- The GPU time is 1.16 ms

Model credit: Infinite-Realities

On PS4, we restrict ourselves to 21 samples per pixel.

With the setup you see on the screen, the compute shader which performs convolution and merges diffuse and specular lighting buffers takes 1.16 milliseconds to execute.



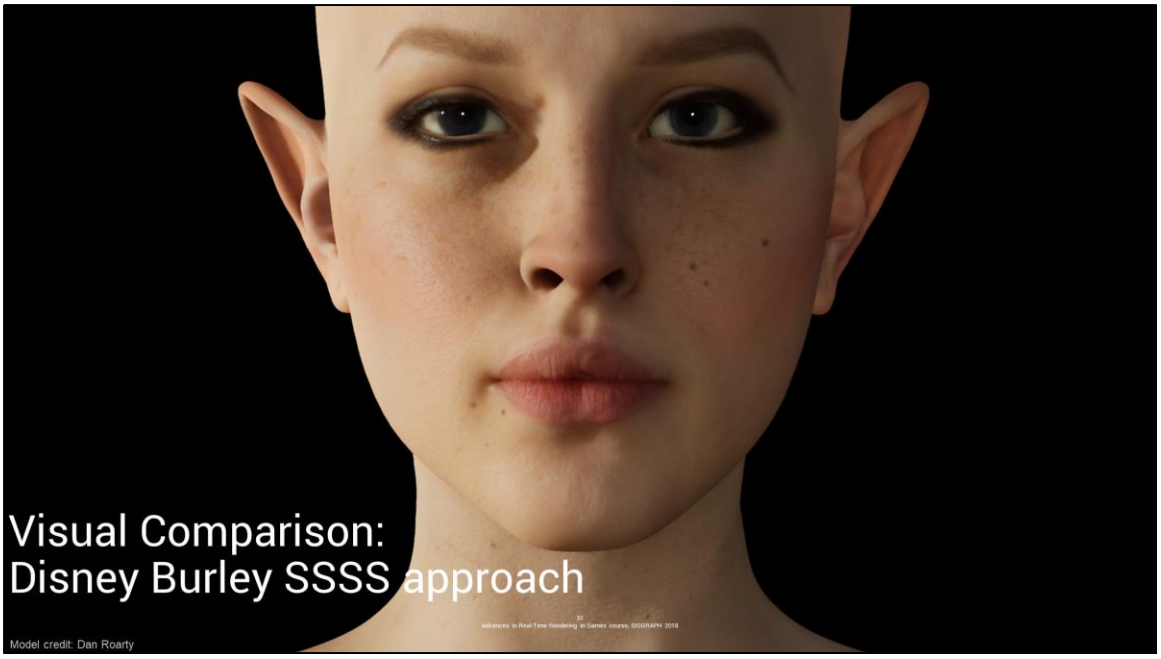
Visual Comparison: Jimenez SSSS approach

Model credit: Dan Roarty

We also implemented the latest Gaussian mix model of Jorge Jimenez for comparison.
If you can see the difference, you don’t need glasses. :-)

flip back and forth

With a single parameter, the Disney model is very easy to control, and it’s possible to achieve good results in a matter of minutes.



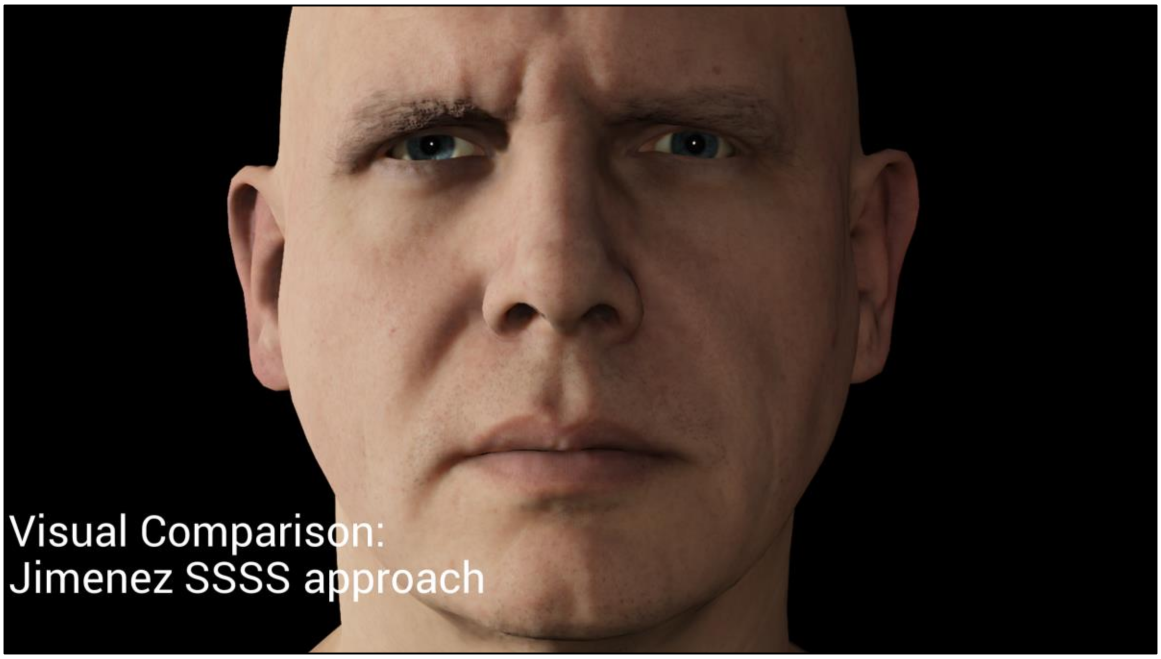
Visual Comparison: Disney Burley SSSS approach

Model credit: Dan Roarty

We also implemented the latest Gaussian mix model of Jorge Jimenez for comparison.
If you can see the difference, you don’t need glasses. :-)

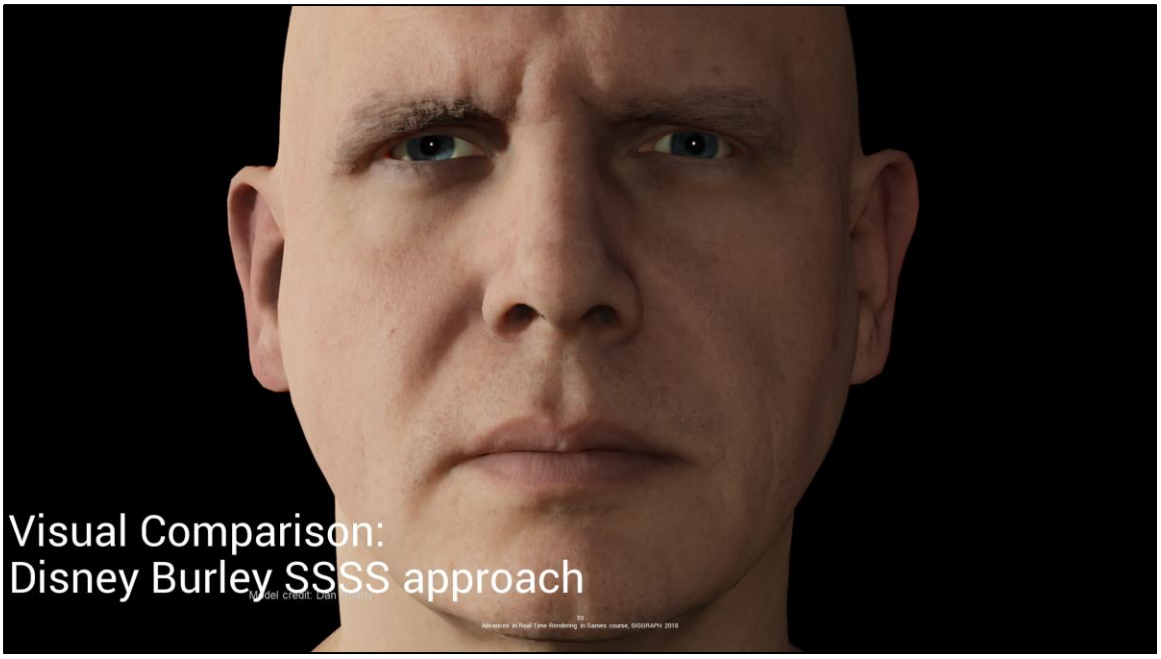
flip back and forth

With a single parameter, the Disney model is very easy to control, and it’s possible to achieve good results in a matter of minutes.



The primary advantage of the Disney model is sharper visuals, which preserve more normal mapping details given the same scattering distance.

Many thanks to my colleague Sebastien Lachambre for allowing us to use the scan of his head.

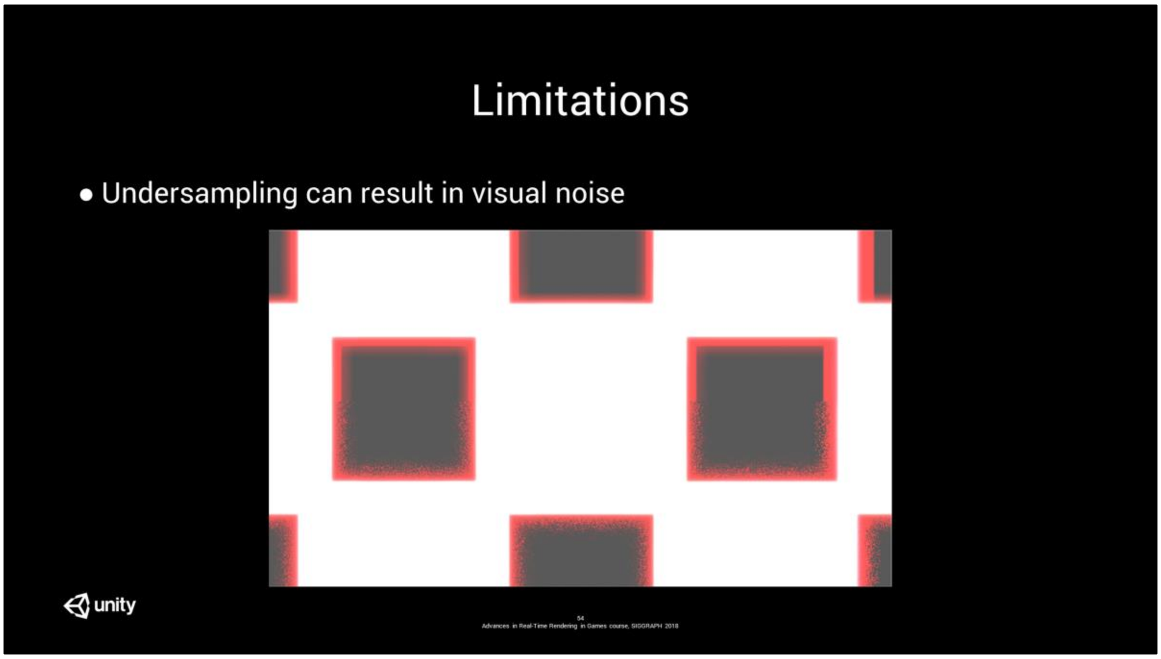


Visual Comparison: Disney Burley SSSS approach

Model credit: Dan Roarty

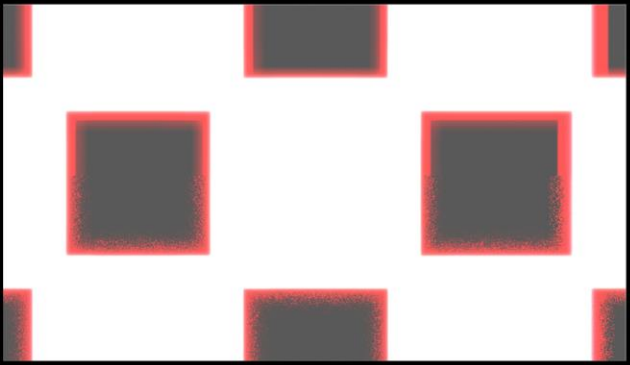
The primary advantage of the Disney model is sharper visuals, which preserve more normal mapping details given the same scattering distance.

Many thanks to my colleague Sebastien Lachambre for allowing us to use the scan of his head.



Limitations

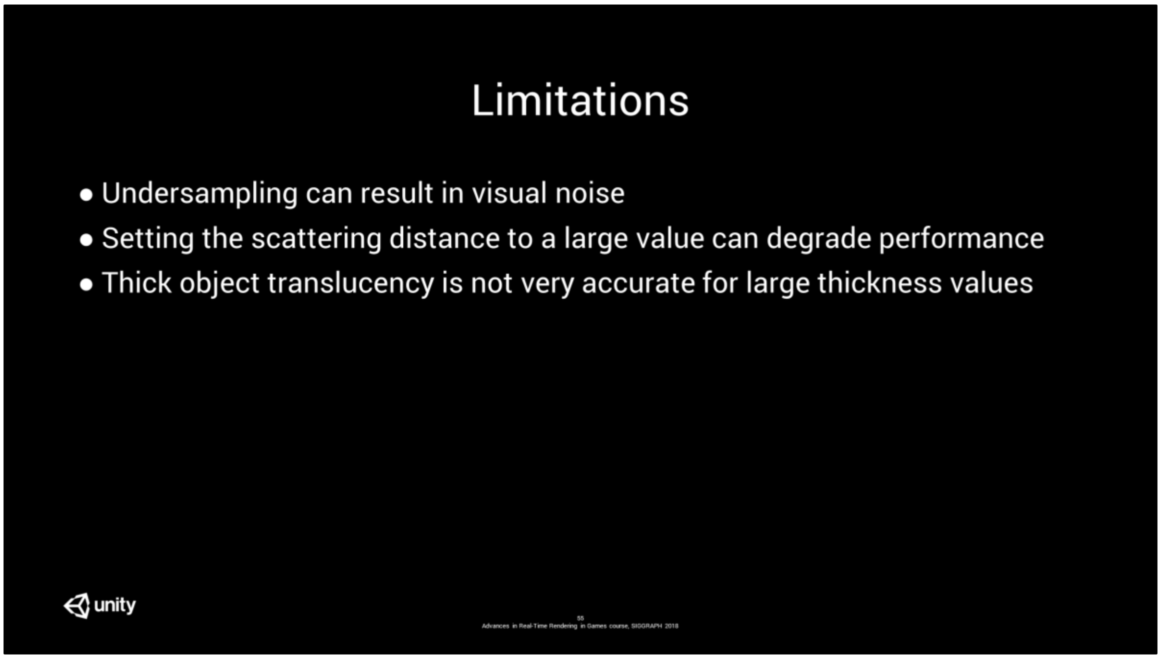
- Undersampling can result in visual noise



unity

A few words about limitations...

Subsurface scattering is implemented as a convolution of the subsurface lighting buffer with the diffusion profile.
The diffusion profile is importance sampled, while lighting is not. Therefore, undersampling of the lighting signal can result in the typical stochastic noise.
Under sufficient illumination, there are usually no visible problems. However, artificial lighting conditions can cause issues.
For example, a simple checkerboard pattern of alternating lit and completely unlit patches is, technically speaking, not a bandlimited signal.
We use temporal antialiasing to reduce the amount of noise.

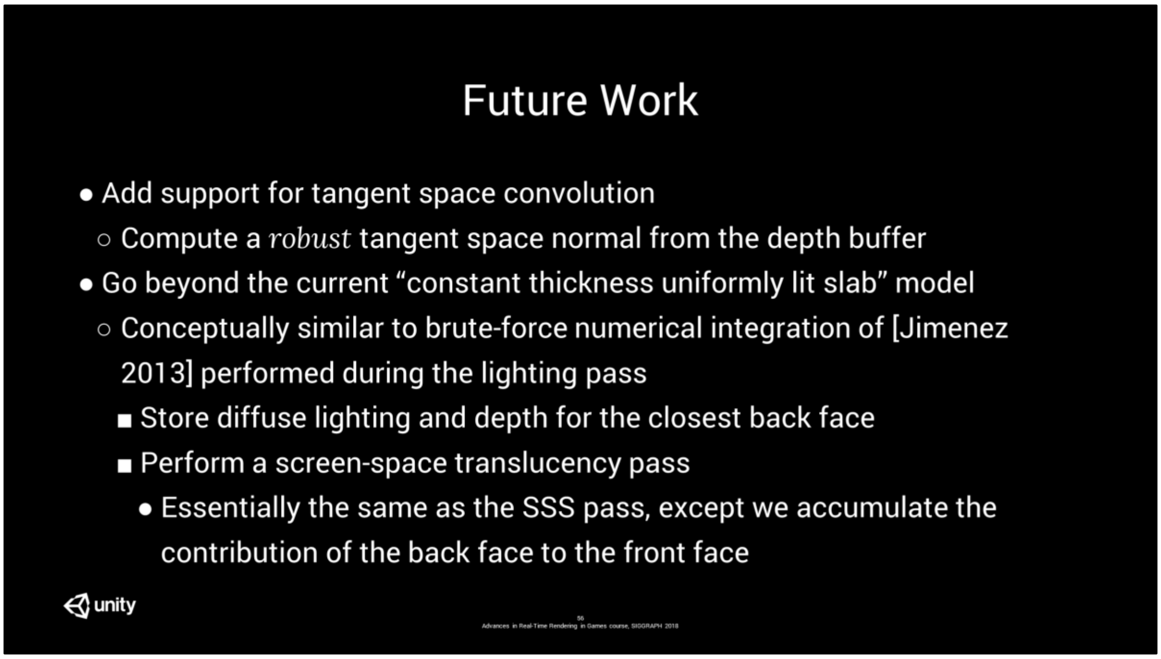


Limitations

- Undersampling can result in visual noise
- Setting the scattering distance to a large value can degrade performance
- Thick object translucency is not very accurate for large thickness values

Another limitation of our method is kernels with very large screen footprint. Samples end up being very far apart, trashing the texture cache and degrading performance.

Finally, thick object translucency makes too many assumptions to produce accurate results for complex thick objects.



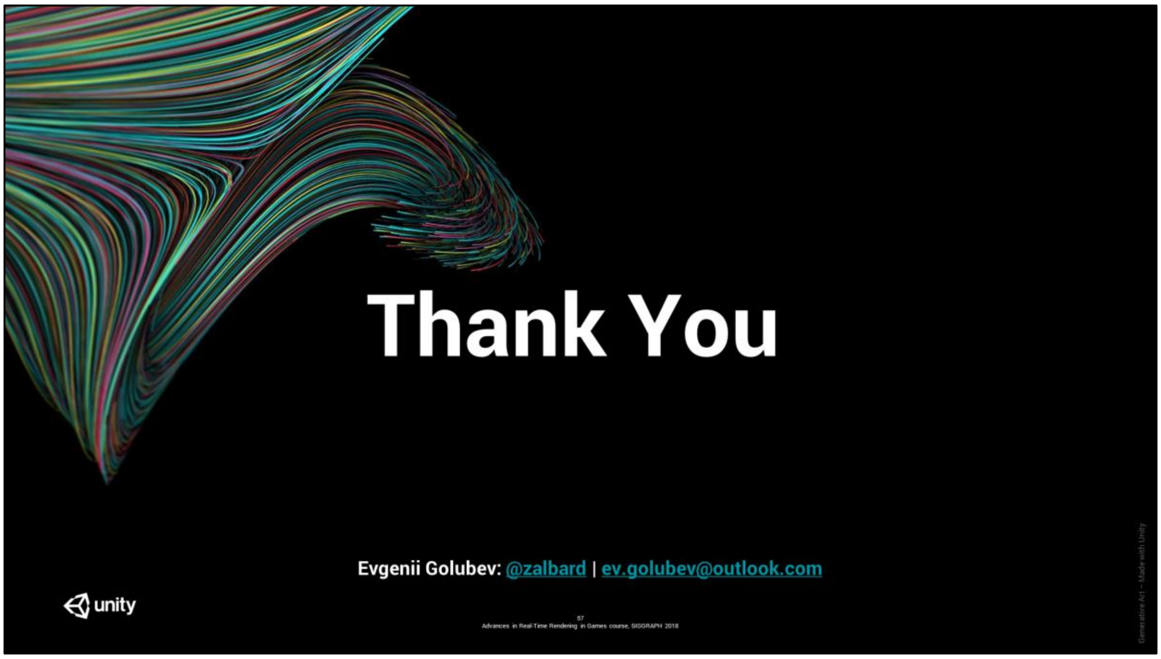
Future Work

- Add support for tangent space convolution
  - Compute a *robust* tangent space normal from the depth buffer
- Go beyond the current “constant thickness uniformly lit slab” model
  - Conceptually similar to brute-force numerical integration of [Jimenez 2013] performed during the lighting pass
    - Store diffuse lighting and depth for the closest back face
    - Perform a screen-space translucency pass
      - Essentially the same as the SSS pass, except we accumulate the contribution of the back face to the front face

unity

In the future, we would like to extend our implementation to properly support tangent-space integration.
Since interpolated vertex normals are not available in the G-Buffer, our plan is to compute a robust tangent space normal from the depth buffer.

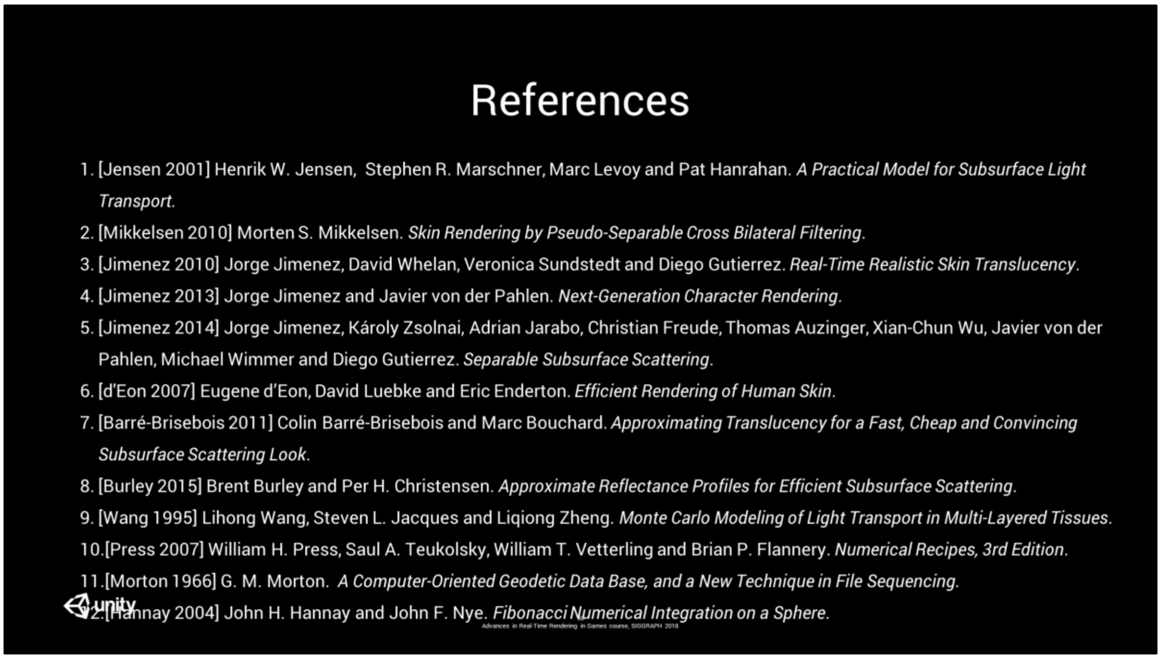
For translucency, we would like to go beyond the current “constant thickness uniformly lit slab” model.
This requires diffuse shading of the closest back face, and another screen-space pass to integrate it with front-face SSS.



Thank You

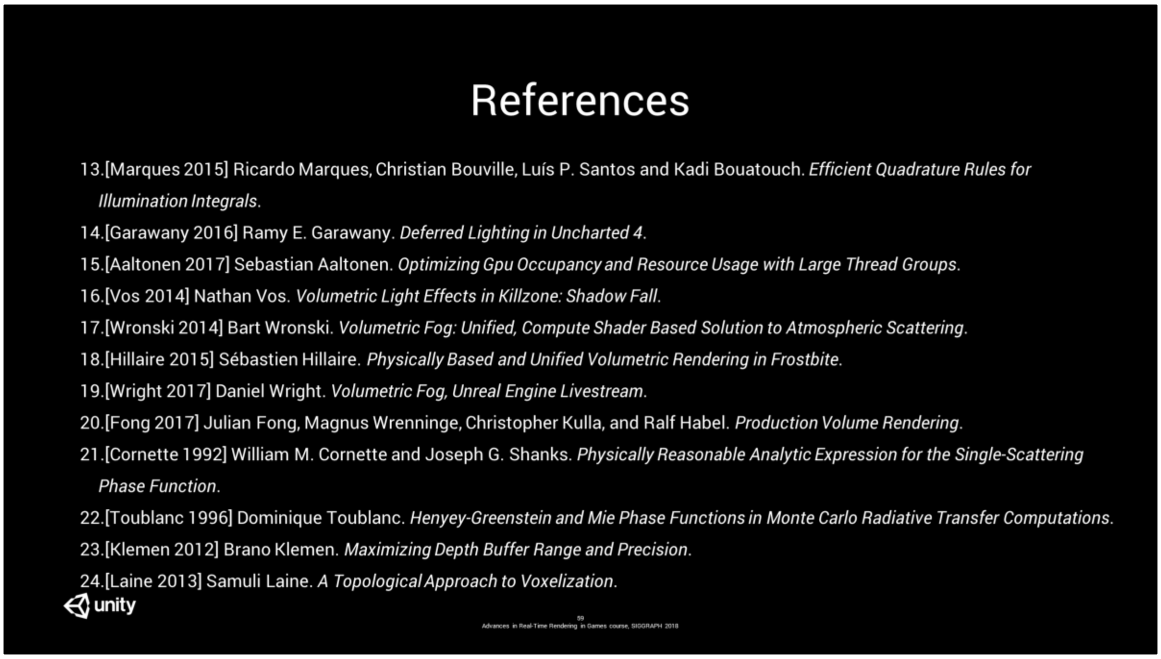
Evgenii Golubev: @zalbard | ev.golubev@outlook.com

Thank you.



References

1. [Jensen 2001] Henrik W. Jensen, Stephen R. Marschner, Marc Levoy and Pat Hanrahan. *A Practical Model for Subsurface Light Transport.*
2. [Mikkelsen 2010] Morten S. Mikkelsen. *Skin Rendering by Pseudo-Separable Cross Bilateral Filtering*.
3. [Jimenez 2010] Jorge Jimenez, David Whelan, Veronica Sundstedt and Diego Gutierrez. *Real-Time Realistic Skin Translucency*.
4. [Jimenez 2013] Jorge Jimenez and Javier von der Pahlen. *Next-Generation Character Rendering*.
5. [Jimenez 2014] Jorge Jimenez, Károly Zsolnai, Adrian Jarabo, Christian Freude, Thomas Auzinger, Xian-Chun Wu, Javier von der Pahlen, Michael Wimmer and Diego Gutierrez. *Separable Subsurface Scattering*.
6. [d'Eon 2007] Eugene d'Eon, David Luebke and Eric Enderton. *Efficient Rendering of Human Skin*.
7. [Barré-Brisebois 2011] Colin Barré-Brisebois and Marc Bouchard. *Approximating Translucency for a Fast, Cheap and Convincing Subsurface Scattering Look*.
8. [Burley 2015] Brent Burley and Per H. Christensen. *Approximate Reflectance Profiles for Efficient Subsurface Scattering*.
9. [Wang 1995] Lihong Wang, Steven L. Jacques and Liqiong Zheng. *Monte Carlo Modeling of Light Transport in Multi-Layered Tissues*.
10. [Press 2007] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. *Numerical Recipes, 3rd Edition*.
11. [Morton 1966] G. M. Morton. *A Computer-Oriented Geodetic Data Base, and a New Technique in File Sequencing.*
12. [Hannay 2004] John H. Hannay and John F. Nye. *Fibonacci Numerical Integration on a Sphere*.



References

13.[Marques 2015] Ricardo Marques, Christian Bouville, Luís P. Santos and Kadi Bouatouch. *Efficient Quadrature Rules for Illumination Integrals*.

14.[Garawany 2016] Ramy E. Garawany. *Deferred Lighting in Uncharted 4*.

15.[Aaltonen 2017] Sebastian Aaltonen. *Optimizing Gpu Occupancy and Resource Usage with Large Thread Groups*.

16.[Vos 2014] Nathan Vos. *Volumetric Light Effects in Killzone: Shadow Fall*.

17.[Wronski 2014] Bart Wronski. *Volumetric Fog: Unified, Compute Shader Based Solution to Atmospheric Scattering*.

18.[Hillaire 2015] Sébastien Hillaire. *Physically Based and Unified Volumetric Rendering in Frostbite*.

19.[Wright 2017] Daniel Wright. *Volumetric Fog, Unreal Engine Livestream*.

20.[Fong 2017] Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. *Production Volume Rendering*.

21.[Cornette 1992] William M. Cornette and Joseph G. Shanks. *Physically Reasonable Analytic Expression for the Single-Scattering Phase Function*.

22.[Toublanc 1996] Dominique Toublanc. *Henyey-Greenstein and Mie Phase Functions in Monte Carlo Radiative Transfer Computations*.

23.[Klemen 2012] Brano Klemen. *Maximizing Depth Buffer Range and Precision*.

24.[Laine 2013] Samuli Laine. *A Topological Approach to Voxelization*.

References

25.[Veach 1997] Eric Veach. *Monte Carlo Methods for Light Transport Simulation.*

26.[Dutré 2006] Philip Dutré, Kavita Bala and Philippe Bekaert. *Advanced Global Illumination, 2nd Edition.*

27.[Novák 2018] Jan Novák, Iliyan Georgiev, Johannes Hanika and Wojciech Jarosz. *Monte Carlo Methods for Volumetric Light Transport Simulation.*

28.[Heitz 2014] Eric Heitz. *Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs.*

29.[Heitz 2016] Eric Heitz, Johannes Hanika, Eugene d'Eon and Carsten Dachsbacher. *Multiple-Scattering Microfacet BSDFs with the Smith Model.*

30.[Heitz 2017] Eric Heitz and Stephen Hill. *Real-Time Line- and Disk-Light Shading.*

31.[Lagarde 2014] Sébastien Lagarde and Charles de Rousiers. *Moving Frostbite to PBR.*

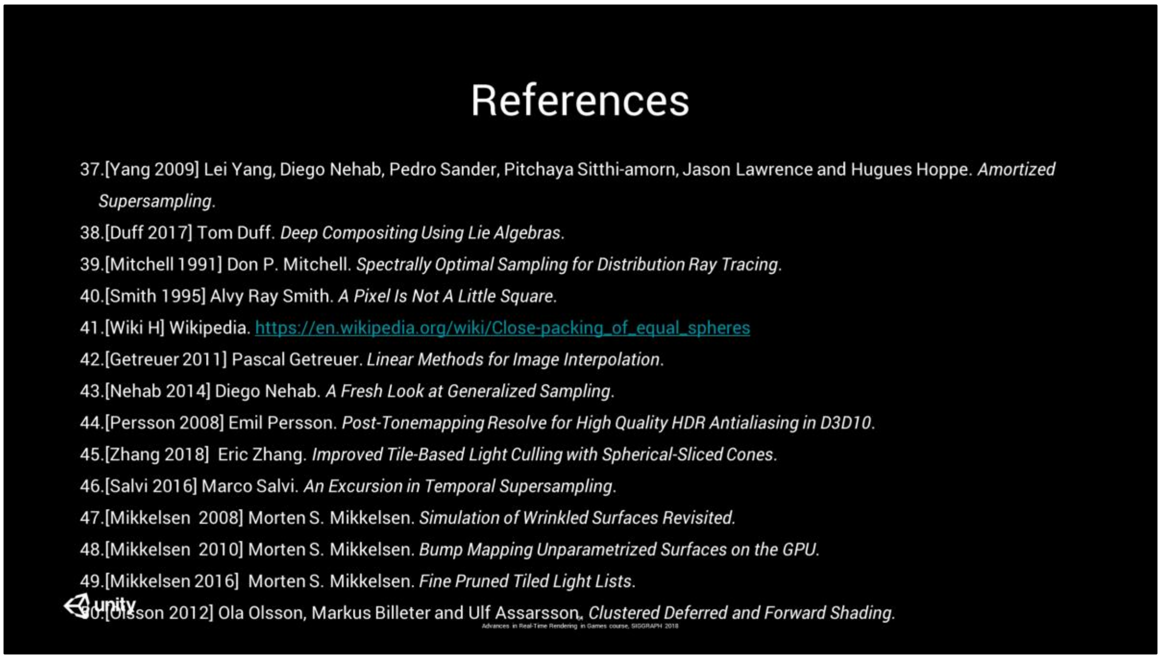
32.[Lagarde 2016] Sébastien Lagarde. *An Artist-Friendly Workflow for Panoramic HDRI.*

33.[Brisebois 2012] Colin Barré-Brisebois and Stephen Hill. *Blending in Detail.*

34.[Belcour 2017] Laurent Belcour and Pascal Barla. *A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence.*

35.[Kulla 2011] Christopher Kulla and Marcos Fajardo. *Importance Sampling of Area Lights in Participating Media.*

36.[Kulla 2017] Christopher Kulla and Alejandro Conty. *Revisiting Physically Based Shading at Imageworks.*



References

37.[Yang 2009] Lei Yang, Diego Nehab, Pedro Sander, Pitchaya Sitthi-amorn, Jason Lawrence and Hugues Hoppe. *Amortized Supersampling*.

38.[Duff 2017] Tom Duff. *Deep Compositing Using Lie Algebras*.

39.[Mitchell 1991] Don P. Mitchell. *Spectrally Optimal Sampling for Distribution Ray Tracing*.

40.[Smith 1995] Alvy Ray Smith. *A Pixel Is Not A Little Square*.

41.[Wiki H] Wikipedia. https://en.wikipedia.org/wiki/Close-packing_of_equal_spheres

42.[Getreuer 2011] Pascal Getreuer. *Linear Methods for Image Interpolation*.

43.[Nehab 2014] Diego Nehab. *A Fresh Look at Generalized Sampling*.

44.[Persson 2008] Emil Persson. *Post-Tonemapping Resolve for High Quality HDR Antialiasing in D3D10*.

45.[Zhang 2018] Eric Zhang. *Improved Tile-Based Light Culling with Spherical-Sliced Cones*.

46.[Salvi 2016] Marco Salvi. *An Excursion in Temporal Supersampling*.

47.[Mikkelsen 2008] Morten S. Mikkelsen. *Simulation of Wrinkled Surfaces Revisited.*

48.[Mikkelsen 2010] Morten S. Mikkelsen. *Bump Mapping Unparametrized Surfaces on the GPU.*

49.[Mikkelsen 2016] Morten S. Mikkelsen. *Fine Pruned Tiled Light Lists*.

50.[Olsson 2012] Ola Olsson, Markus Billeter and Ulf Assarsson. *Clustered Deferred and Forward Shading.*

References

37.[Burley 2012] Brent Burley. *Physically-Based Shading at Disney*.
38.[McAuley 2015] Steve McAuley. The rendering of far cry 4.
39.[Revie 2011] Donald Revie, Implementing Fur Using Deferred Shading.
40.[Lagarde 2011] Sébastien Lagarde. Feeding a physically based shading model.
41.[Lagarde 2013] Sébastien Lagarde. Memo on Fresnel equations.
42.[Brinck 2016] Waylon Brinck and Andrew Maximov. Technical art of Uncharted 4.
43.[Sousa 2016] Tiago Sousa and Jean Geffroy. The devil is in the details: idTech 666.
44.[Cantlay 2007] Iain Cantlay. High-Speed, Off-Screen Particles
45.[Coffin 2011] Christina Coffin, SPU-Based Deferred Shading in BATTLEFIELD 3 for Playstation 3
46.[Hill 2016] Stephen Hill. LTC Fresnel Approximation.