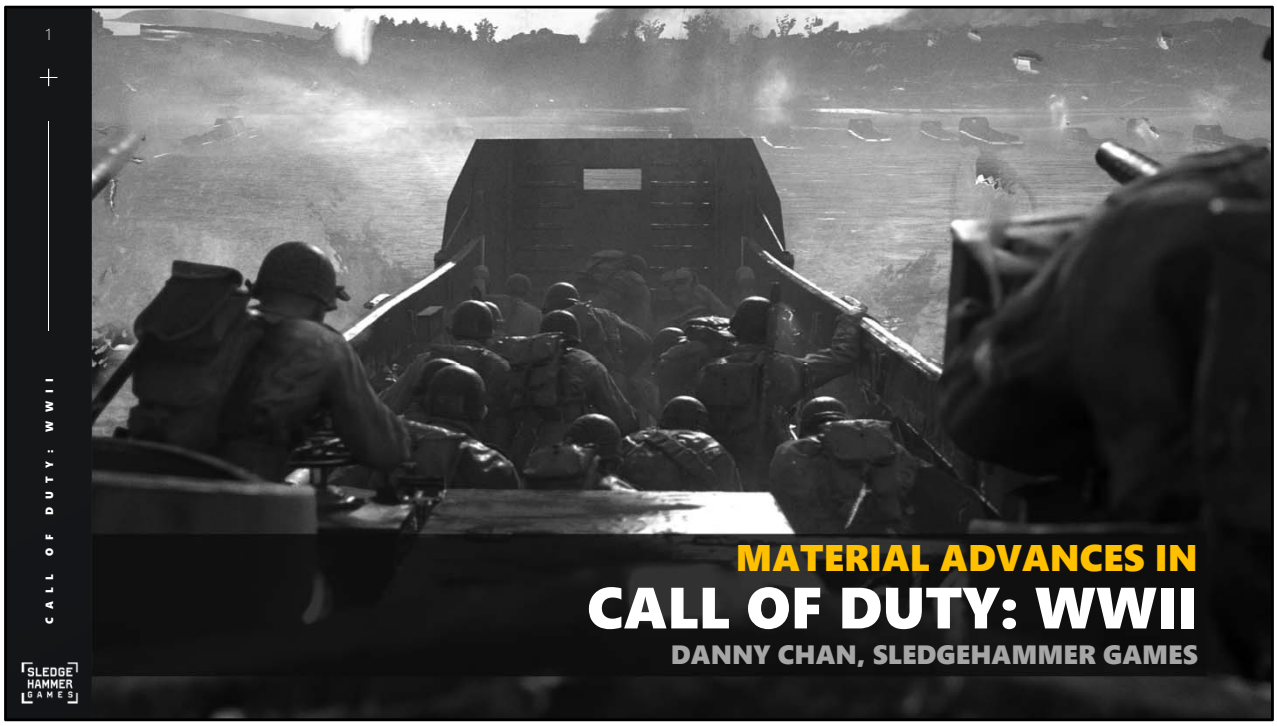
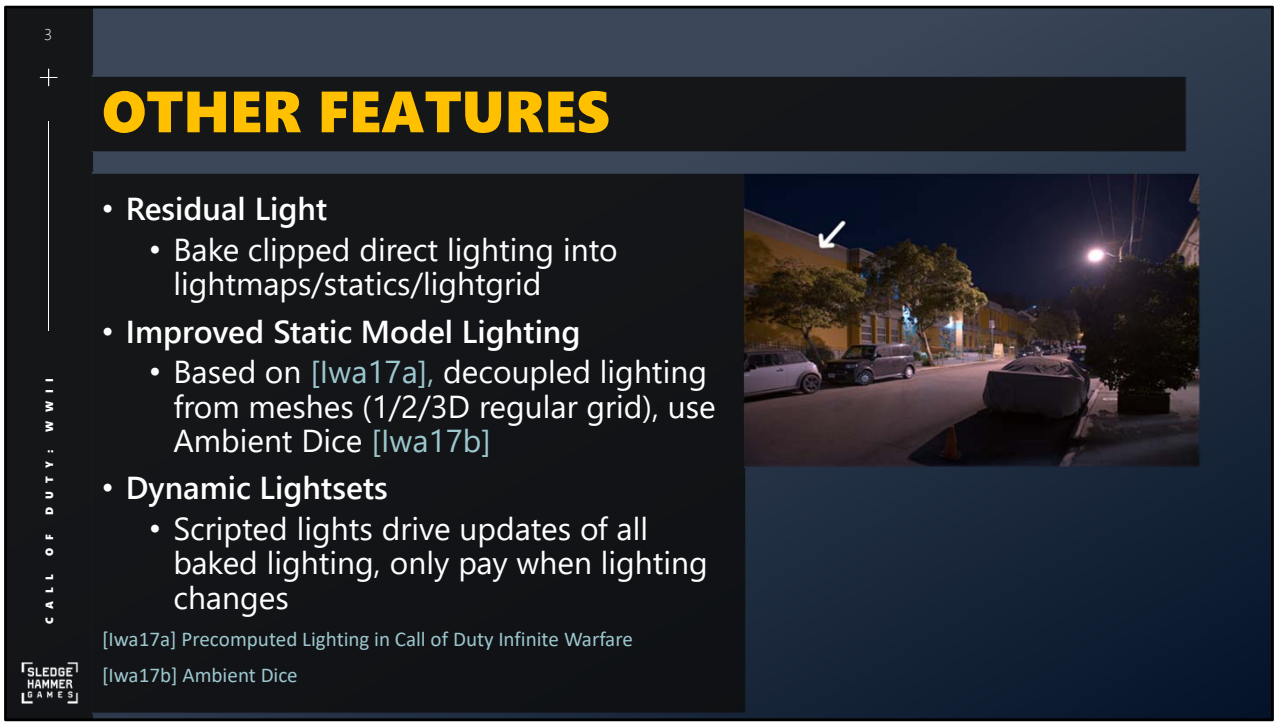
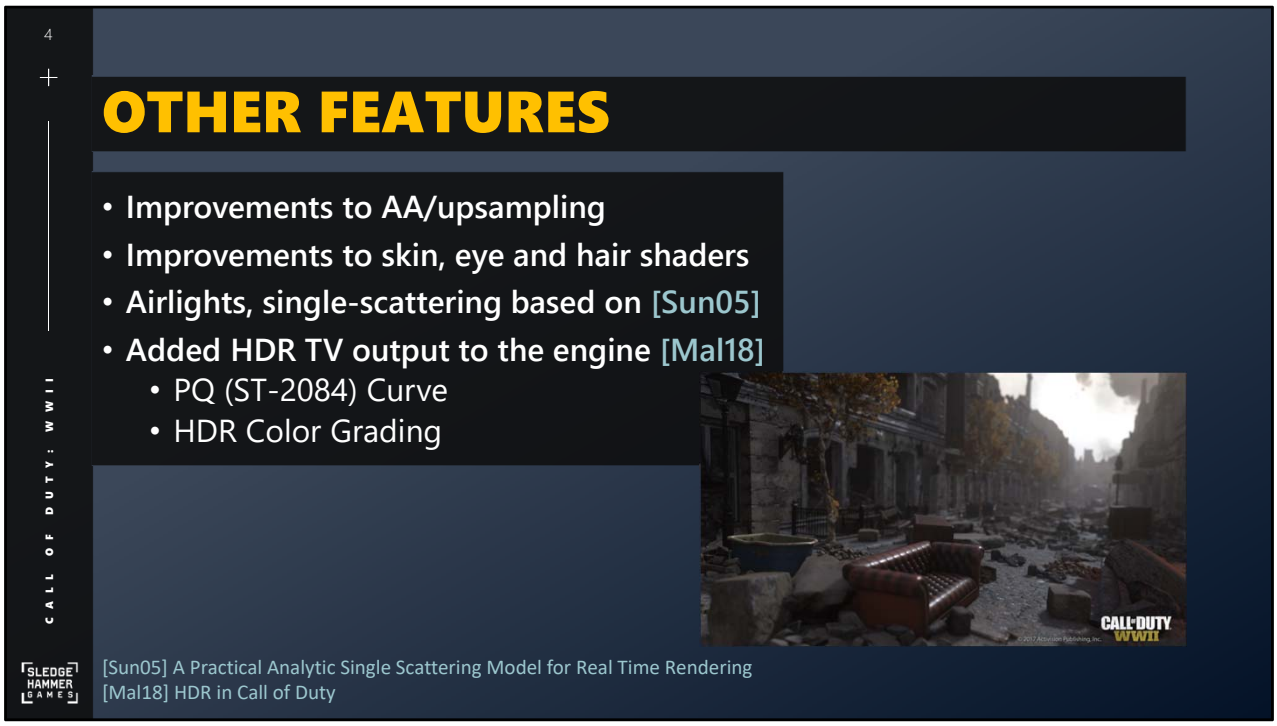

1
CALL OF DUTY: WWII
SLEDGE HAMMER GAMES
MATERIAL ADVANCES IN
CALL OF DUTY: WWII
DANNY CHAN, SLEDGEHAMMER GAMES

Welcome

When we started WWII, our art directors wanted our game to render during magic hour.
A lot of talk centered around wanting to achieve the look that the Revenant had by relying on soft lighting.
Our engine was great at bright sunlit scenes but the tech we developed for AW was not quite enough to achieve a rich ambient look.
We needed to refine our material and lighting models if we wanted to achieve that look.
Here is an early version of Normandy rendered in-game.
We were able to capture some of that mood we were after.



Before we begin, here's a quick run-down of some of the other features that we don't have time to talk about.
Residual Lighting is based on the idea that punctual light sources don't act like real-world lights.
Punctual lights falloff to zero at some radius, and are usually confined to a cone.
Real worlds lights are not limited in these ways.
We noticed there is a long tail of lighting that is very visible, especially at night.
And light spills in all directions.
The photo to the right demonstrates this, where a single streetlamp is able to light the entire street,
even the top of the far wall of the building on the left.
Residual lighting is this long tail of lighting that is not represented by our punctual light sources.
We bake this residual lighting into our lightmaps and lightgrid.
We also made changes to the representation of indirect lighting for models.
We switched to Ambient Dice, an alternative to SH.
We also added Dynamic Lightsets, providing a way to update baked lighting based on scriptable lights.



We made improvements to anti-aliasing, upsampling, our skin, eye and hair shaders.
EDIT ADDED AFTER THE COURSE We added Airlights, single scattering based on a model introduced in SIGGRAPH 2005.
And finally, we added HDR TV support to the engine.

[Sun05] A Practical Analytic Single Scattering Model for Real Time Rendering
http://www.cs.columbia.edu/~bosun/sig05.htm

[Mal18] HDR in Call of Duty
https://research.activision.com/t5/Publications/HDR-in-Call-of-Duty/ba-p/10744846

5

CALL OF DUTY: WWII

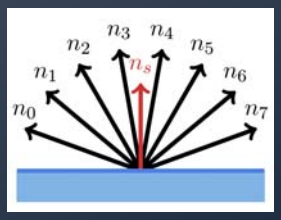
SLEDGEHAMMER GAMES

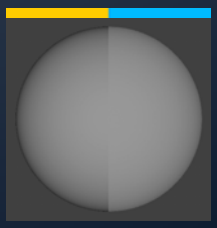
MATERIAL ADVANCES IN CALL OF DUTY: WWII

(1) Normal and Gloss Mipmapping
- Rational Function Fitting
- Combining Detail Gloss

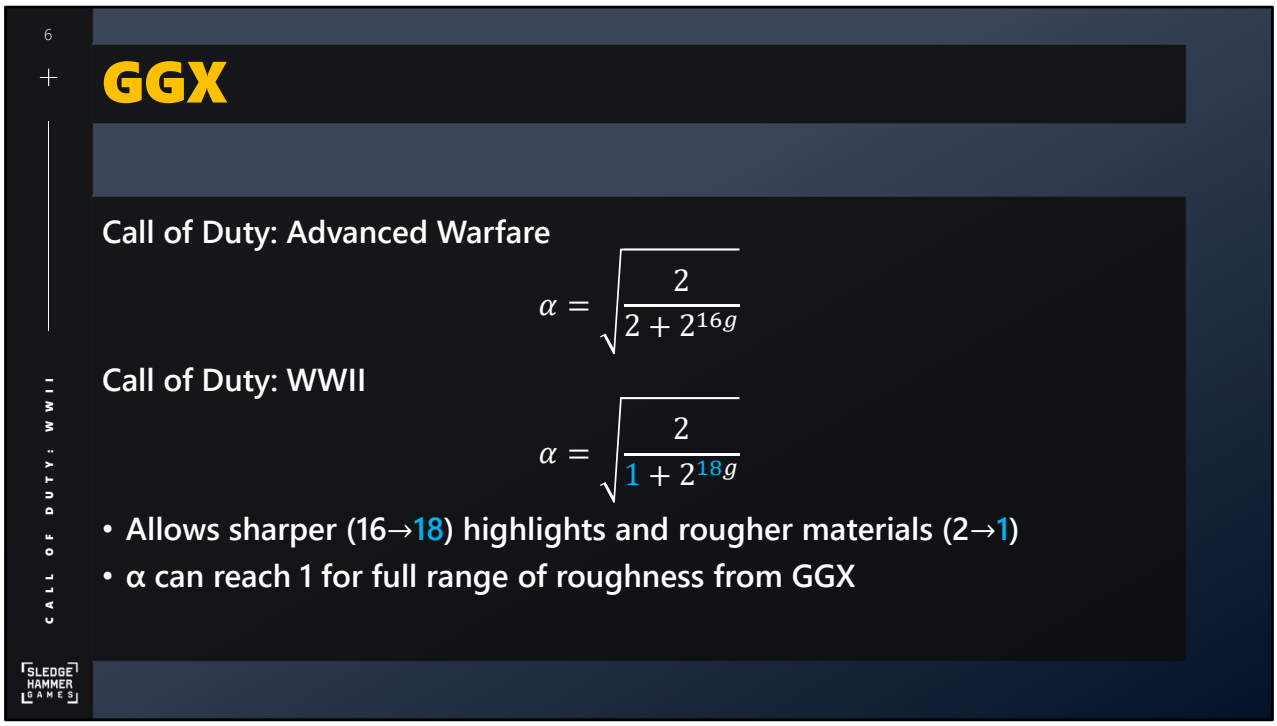
(2) Material Surface Occlusion

(3) Multiscattering Diffuse BRDF
- Energy Conserving Diffuse



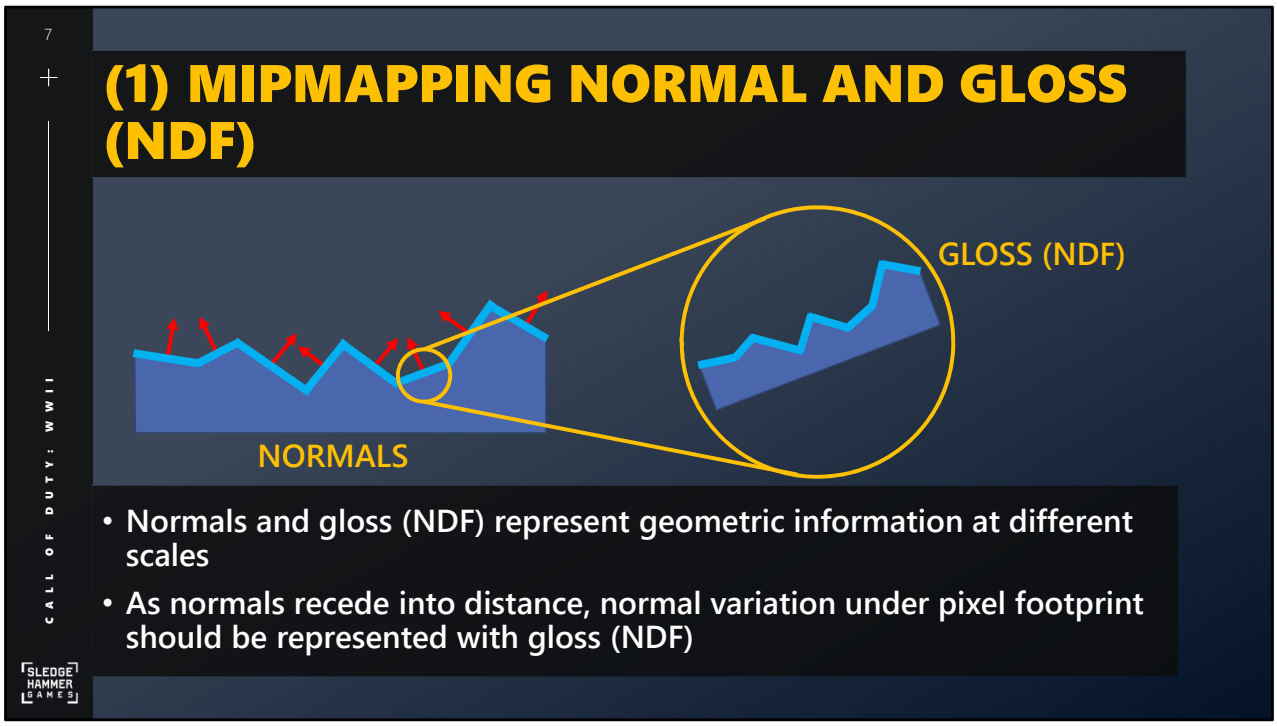


What we'll be talking about in detail about are material changes that we made during the project to help achieve our art direction goals.
First we'll talk about normal and gloss mipmapping with some side-detours into rational function fitting and combining gloss values.
Then, we'll talk about Material Surface Occlusion, which occurs because of microgeometry defined by normal maps.
Finally, we'll go over a Multiscattering Diffuse BRDF that gives us more material differentiation through our gloss range.

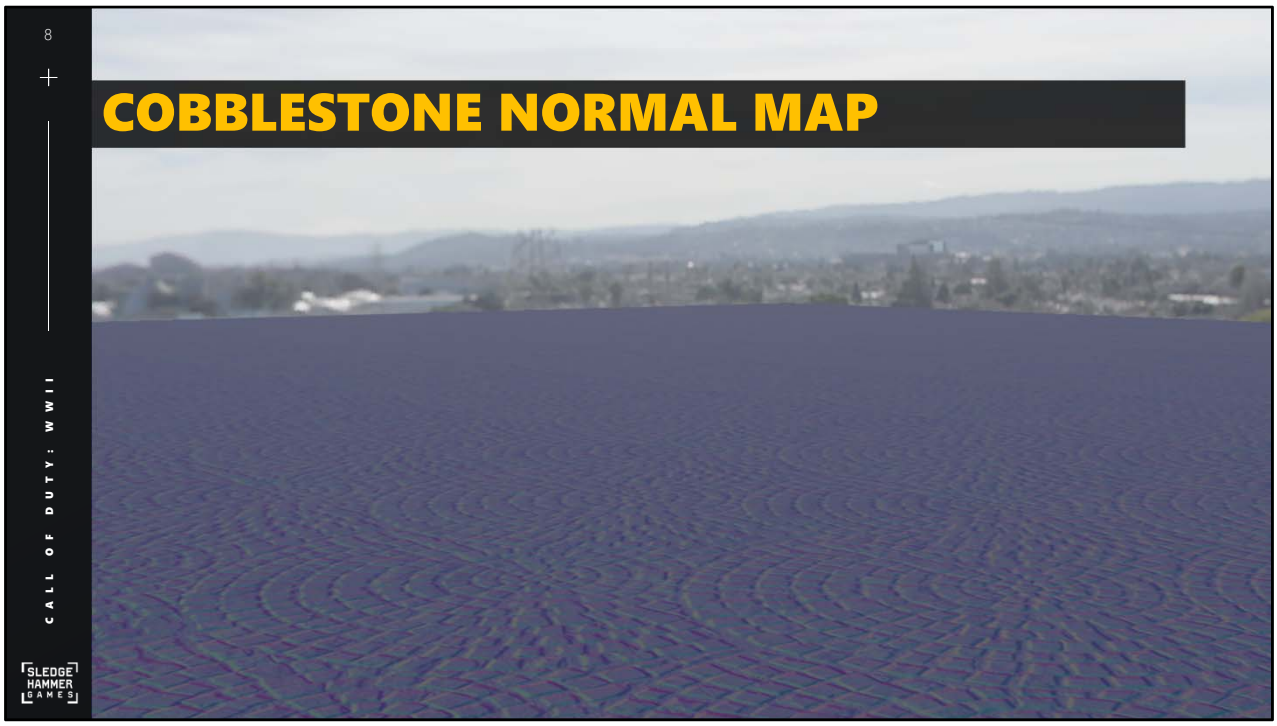


Before we dive in, I want to mention we are using a new parameterization of gloss.
We've changed from the one we used on Advanced Warfare.
Full range of roughness, also glossier.
This is primarily motivated by the need to create glossier materials and
it's also important to achieve full range of our new Multiscattering Diffuse BRDF.

* EDIT ERRATA * Original slides mistakenly had Advanced Warfare: alpha=sqrt(2/(2+16g))
and WWII: alpha=sqrt(2/(1+18g))

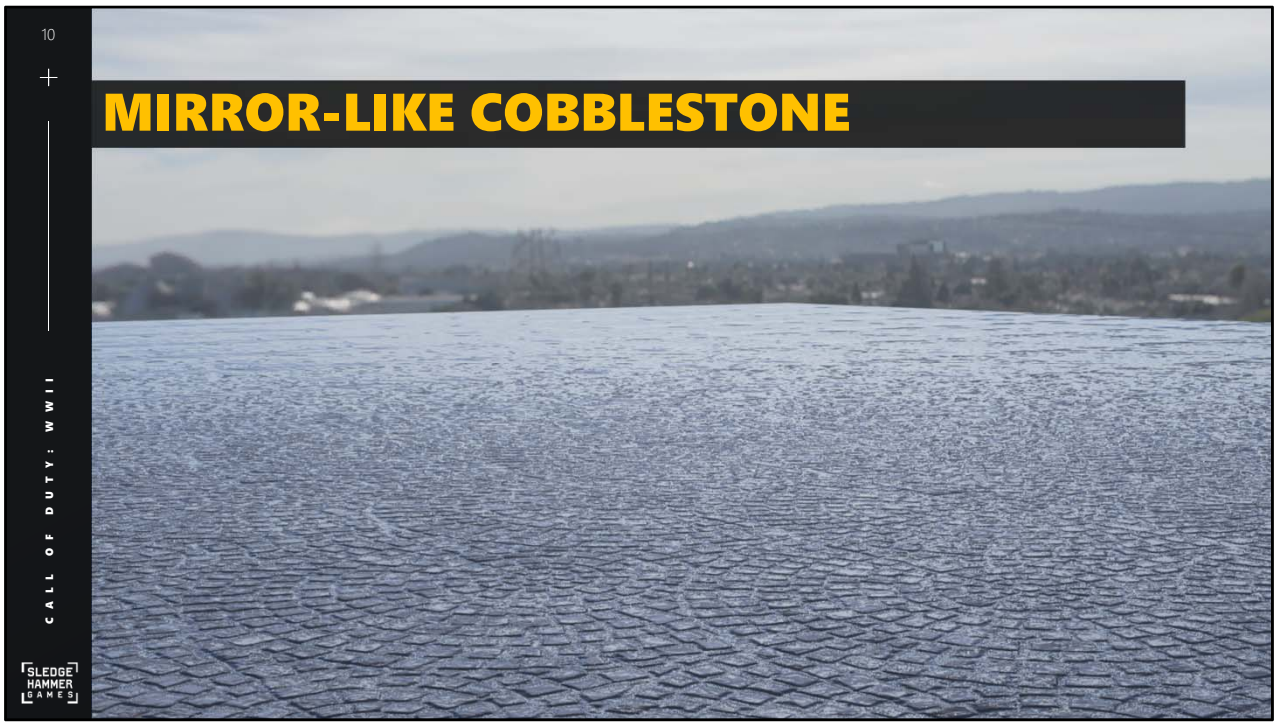


We use a microfacet model for material shading, where the microfacet details are described by a gloss map.
For the first section of the talk, I'll be using the term gloss, but this is interchangeable with smoothness or roughness, and it really describes the underlying NDF of a pixel.
Since gloss represents a normal distribution function, a normal map and a gloss map represent similar geometric information, just at different scales.
As a normal map recedes into the distance, the normal variation under the rendered pixel footprint should be represented with gloss.

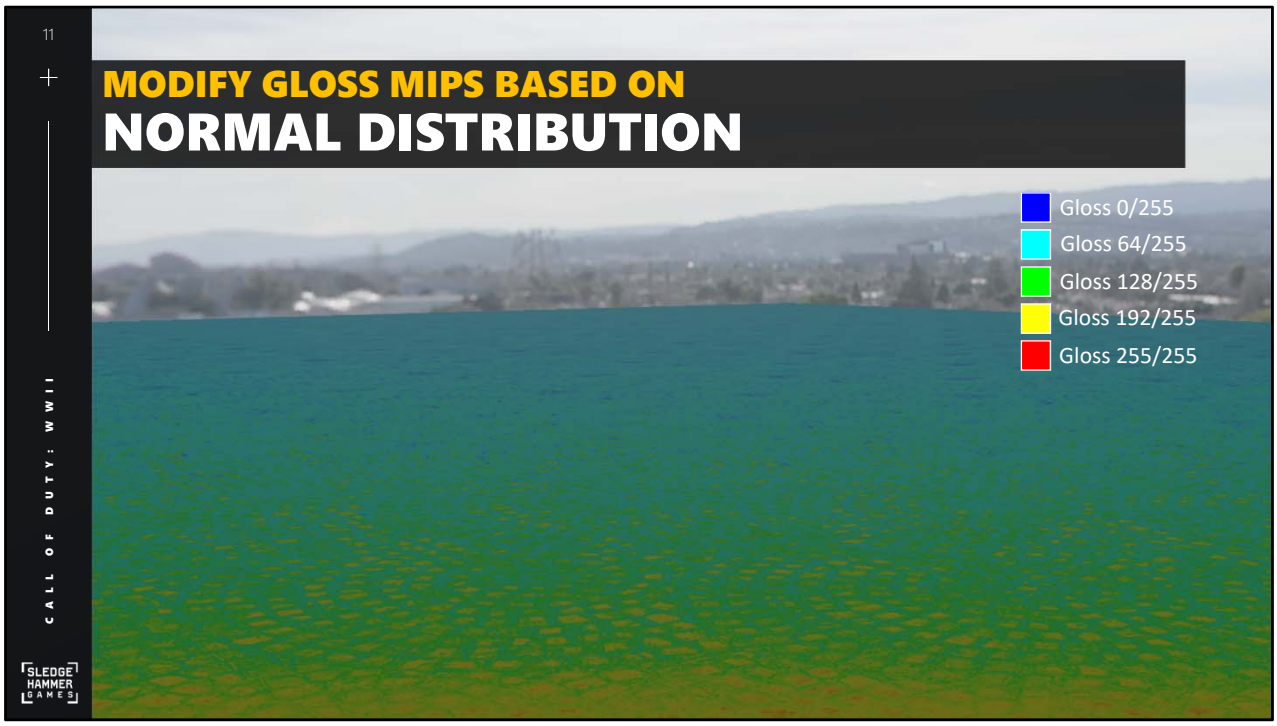


Let’s go over a simple example of how standard mipmapping would result in less than ideal results.
This is the normal map of a cobblestone material. Notice in the distance, the normal map mipmaps into a flatter, less bumpy surface.

Here is a very glossy gloss map.
There isn't a cobblestone material in the game with a constant gloss of 255, but I've exaggerated the gloss of this example to show off the effect that standard mipmapping can have on materials.
This is standard mipmapping of 255 gloss.

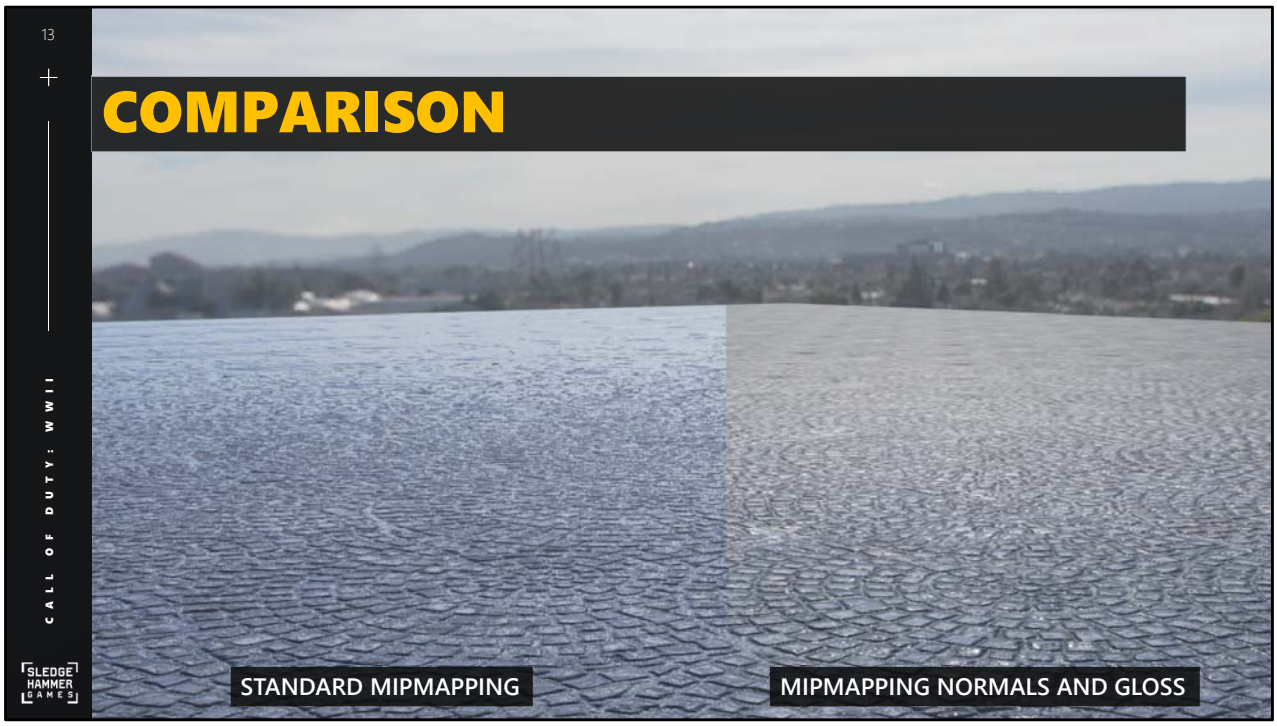


And this results in a mirror-like cobblestone rendering.
If you were to move the camera around, you would see a shimmering from specular aliasing.
Shimmering is probably the more noticeable side-effect of standard mipmapping.

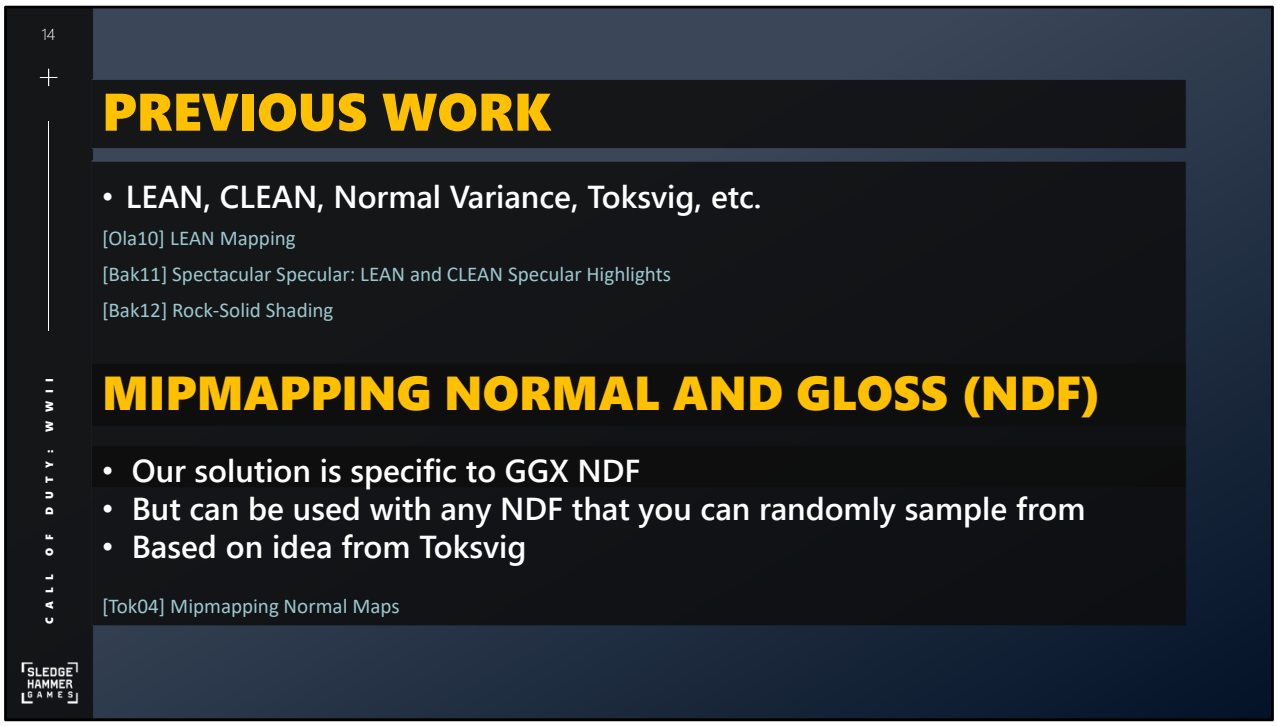


Here we will modify our gloss mips based on the underlying normal distribution of our normal maps.
Notice in the distance, the gloss value goes lower to account for the variation in normals under the pixel footprint.

And this is the result.
The surface is no longer mirror-like in the distance.



Here's a comparison between standard mipmapping and our technique for mipmapping normals and gloss.



A lot of techniques have been developed to address these issues of specular aliasing and maintaining normal variation across scales.
For a good rundown, please refer to the SIGGRAPH 2012 presentation in this course by Dan Baker and Stephen Hill.

Our technique is specific to any NDF that you can importance sample from.
In our case our NDF is GGX.
The foundation of the technique is based on an idea from a 2004 paper by Toksvig.

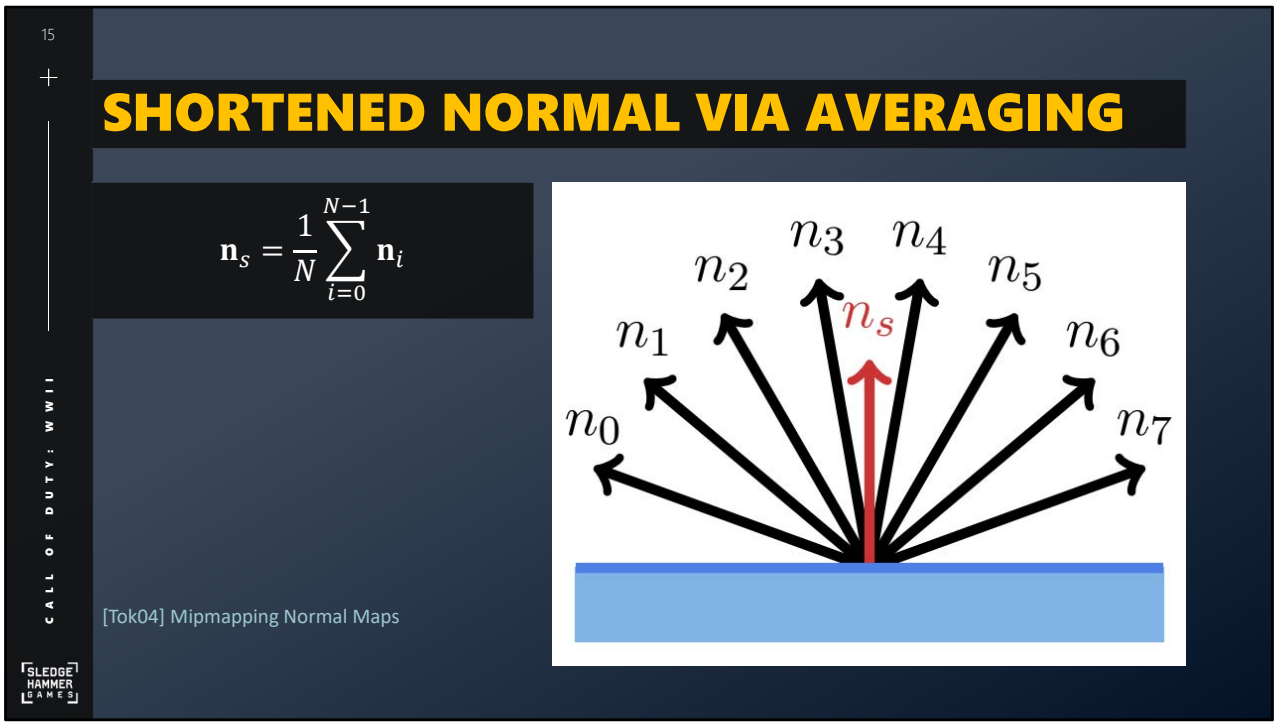
[Ola10] LEAN Mapping
https://www.csee.umbc.edu/~olano/papers/lean/

[Bak11] Spectacular Specular: LEAN and CLEAN Specular Highlights
https://www.gdcvault.com/play/1014557/Spectacular-Specular-LEAN-and-CLEAN

[Bak12] Rock-Solid Shading
http://advances.realtimerendering.com/s2012/

[Tok04] Mipmapping Normal Maps
https://developer.download.nvidia.com/whitepapers/2006/Mipmapping_Normal_Maps.pd

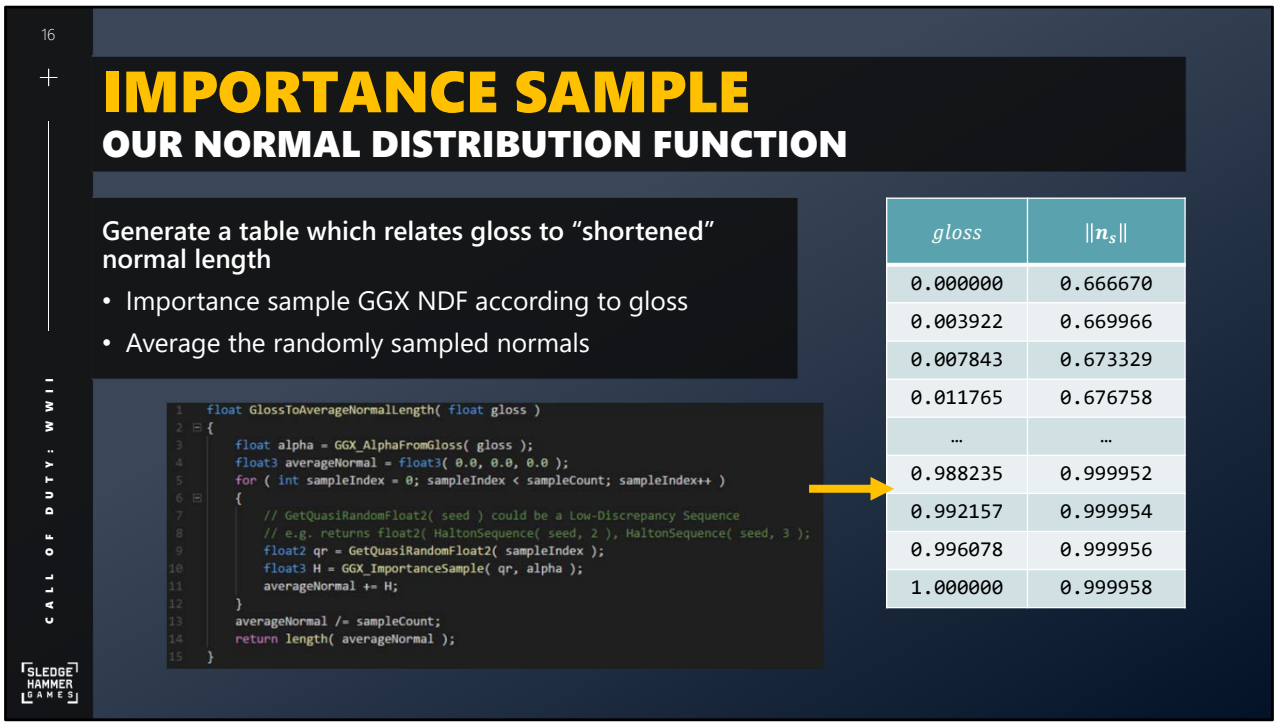
f



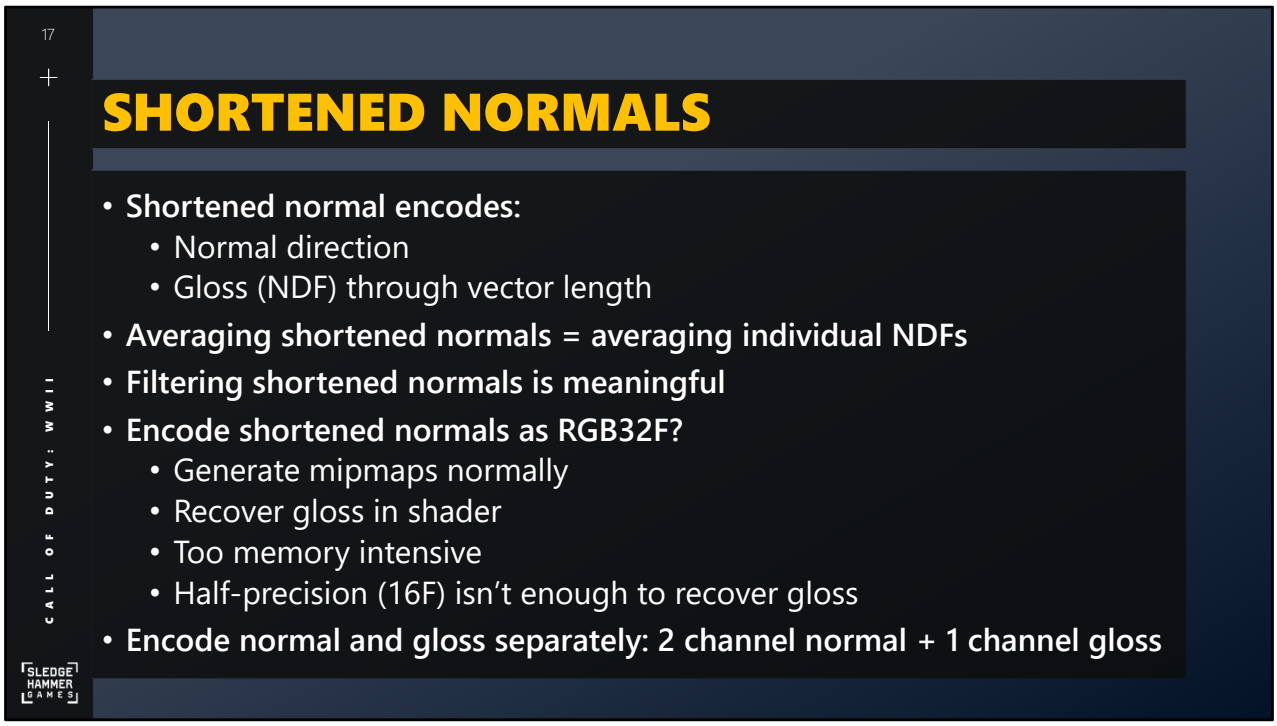
The Toksvig paper describes the concept of an average (shortened) normal.
To calculate a shortened normal, you just average a bunch of normals.
That's what the red vector is in the diagram, an average of the eight normal vectors.
This shortened normal is a representation that can encode both direction and NDF, through the normal vector length.
The Toksvig paper relates shortened normal length to variance of normal direction and uses this to scale down specular shininess in a Blinn-Phong model.

We're a little different.
We ended up relating shortened normal length *directly* to gloss in our microfacet model with our specific NDF.
This lets us bypass the intermediate variance representation.

[Tok04] Mipmapping Normal Maps
https://developer.download.nvidia.com/whitepapers/2006/Mipmapping_Normal_Maps.pdf



The term importance sampling is normally used in the context of Monte Carlo simulations.
Here we can also think of it as randomly sampling normals from the normal distribution function.
We're going to loop through 255 gloss values from 0 to 1 and for every gloss value
We're going to average a lot of random normals for that gloss and
Calculate the length of that average normal.
Now we have our table relating gloss to shortened normal length.
At this point, we can convert freely between the two.
This conversion table will be used in our mipmapping process.



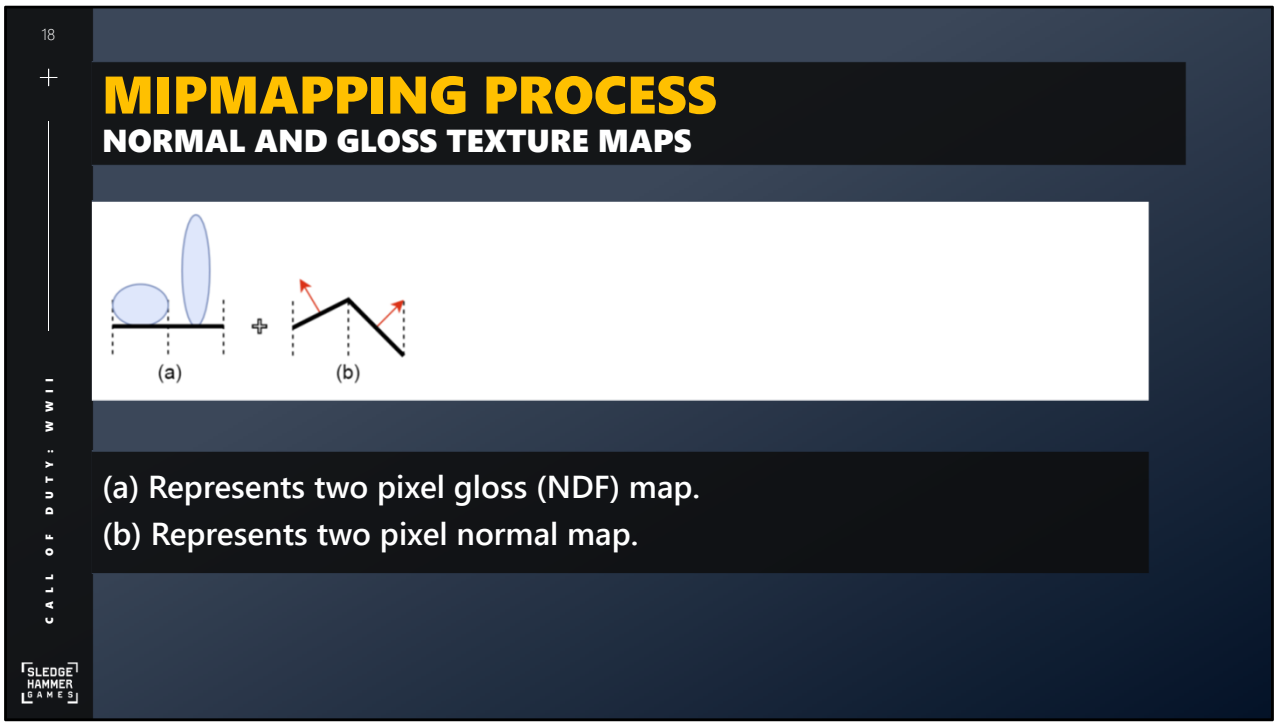
Shortened normals encode both the normal direction and gloss.
The key property of shortened normals is that averaging them is equivalent to averaging the individual NDFs.
This leads to the nice property that filtering shortened normals is meaningful.

So we could encode normal and gloss information into a single 3-channel floating point texture.
Then we generate mipmaps using whatever filter we choose.
The filtering of shortened normals will just "do the right thing".
In our shaders, we would recover gloss from the length of the filtered normal vector.
But this is too memory intensive and half-precision normals don't give enough precision to recover gloss accurately.

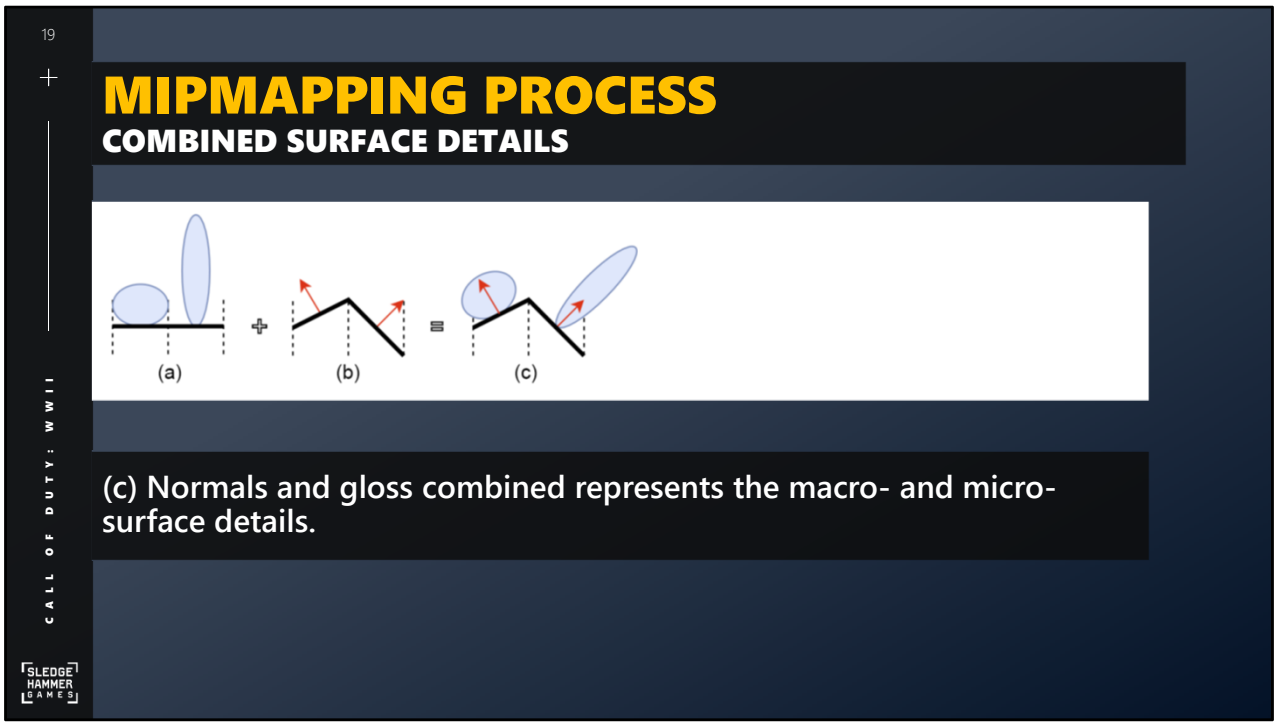
Instead we encode normal and gloss separately.
We have a two-channel normal and one-channel of gloss.

[Question from Q&A session after the talk]
Why not use 16 bit signed integer values to represent normals, instead of RGB32F? Would that have worked?

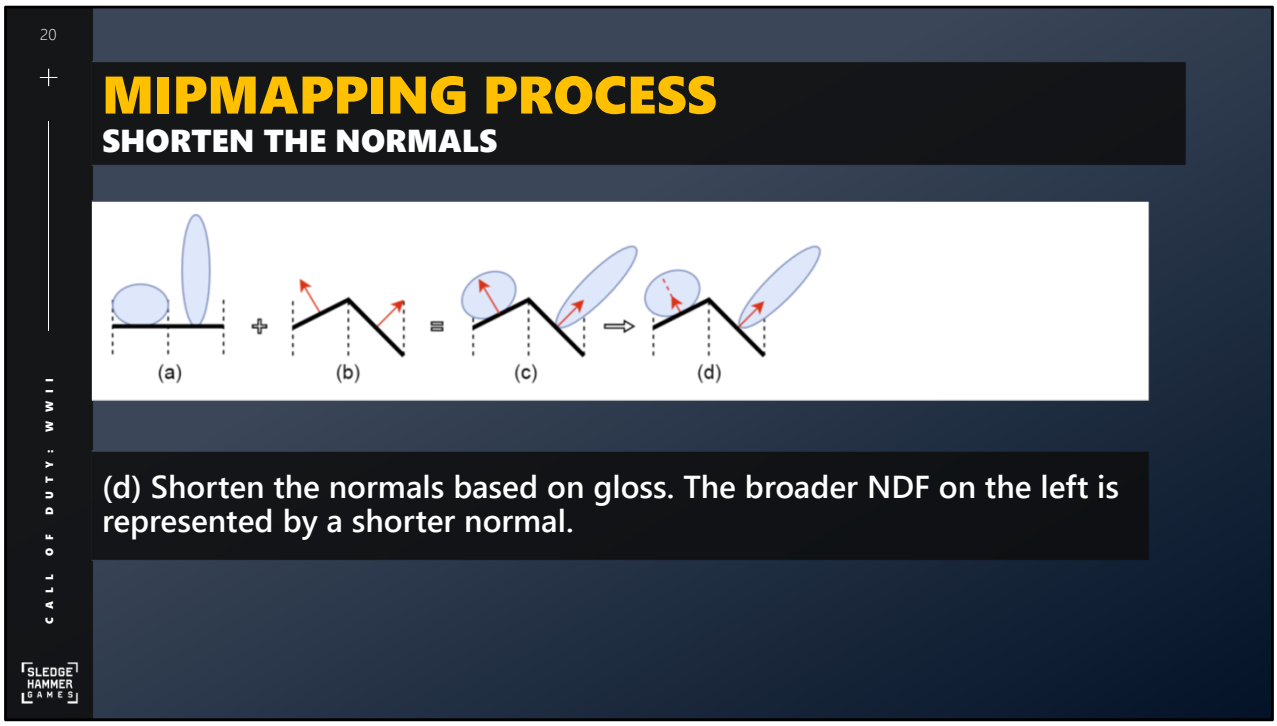
- During the Q&A, I didn’t remember if we considered this but suspected it still wouldn’t have the precision required since we need at least five digits of base-10 precision. We actually need closer to 6 digits of base-10 precision (see previous slide for table) when calculating the *length* of the shortened normal vector, which means the vector would need to be represented to even higher precision. Also, we only considered 16F because BC6H provides a block-coded format at 8bpp. There is no block-coded format that does 16bit SNORM yet. An uncompressed 3 channel 16 bit SNORM would still be too memory intensive.



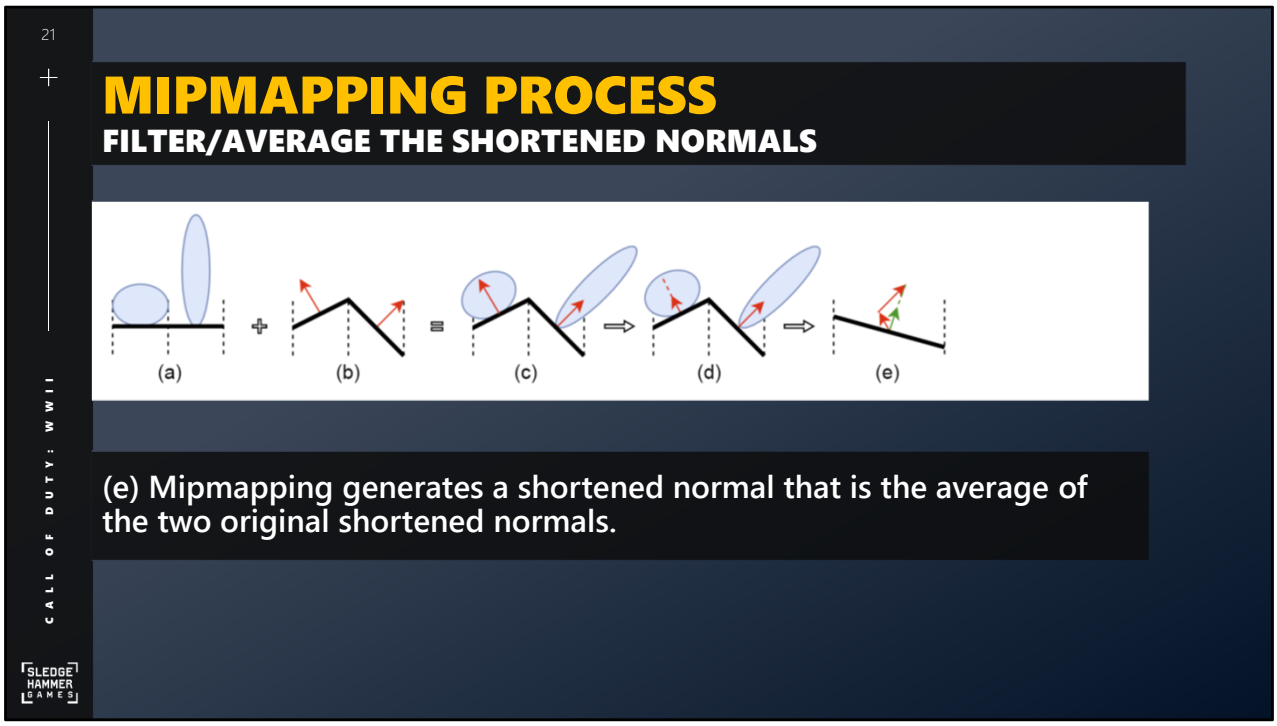
Now, we'll go over the simple process of mipmapping normals and gloss using our technique.
Imagine a two pixel gloss map and normal map, viewed conceptually from the side
(a) Represents the gloss map
(b) Represents the normal map



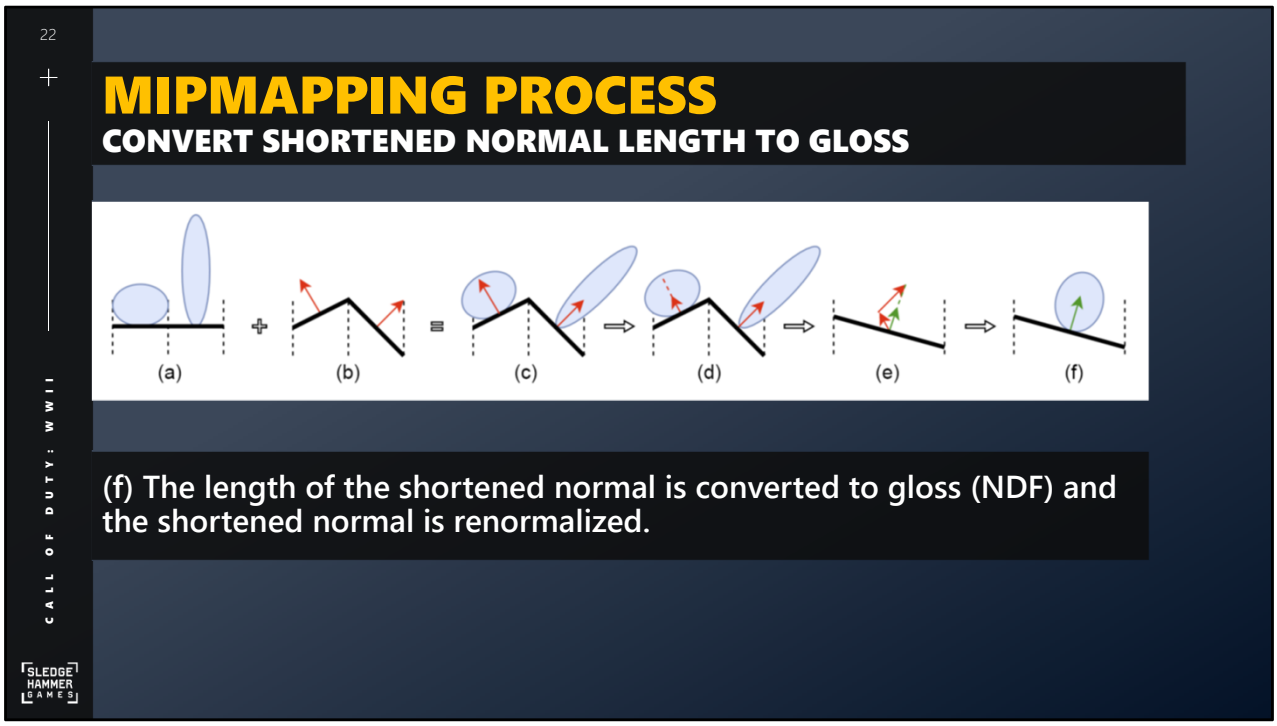
(c) Normals and gloss combine to represent the macro- and micro-surface details.



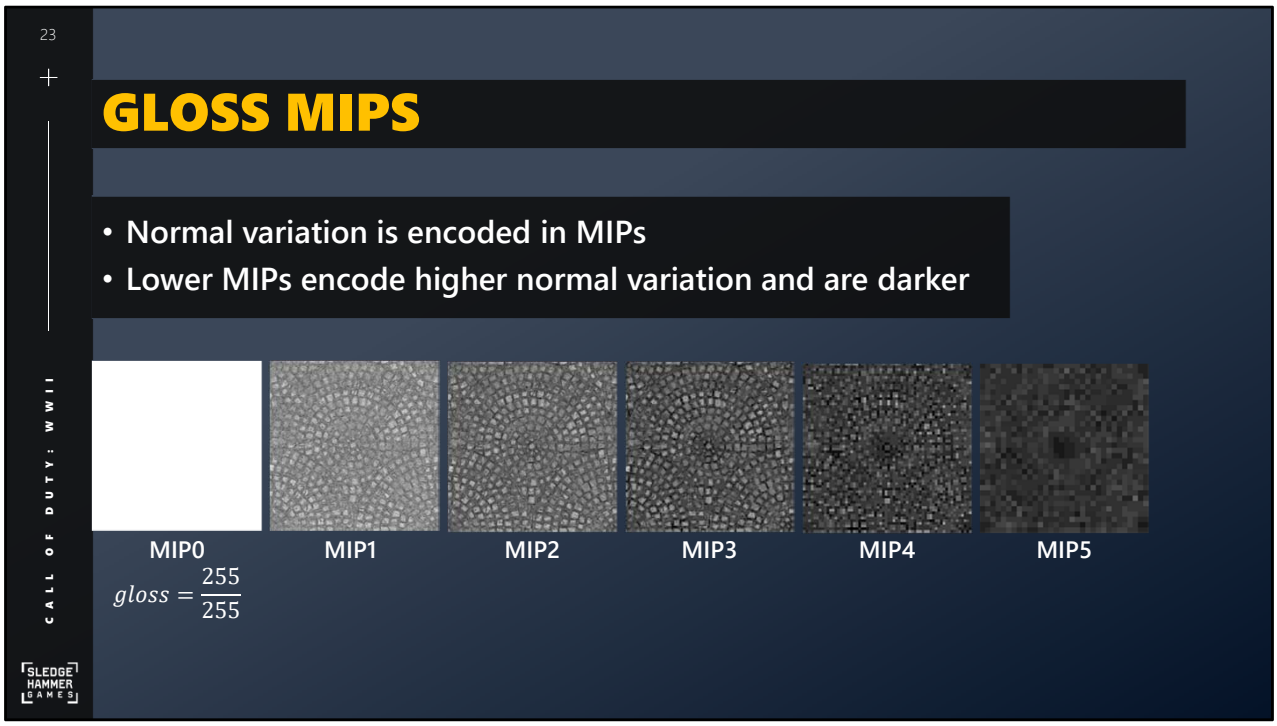
(d) Now, we shorten the normals based on gloss. The broader NDF on the left is represented by a shorter normal.



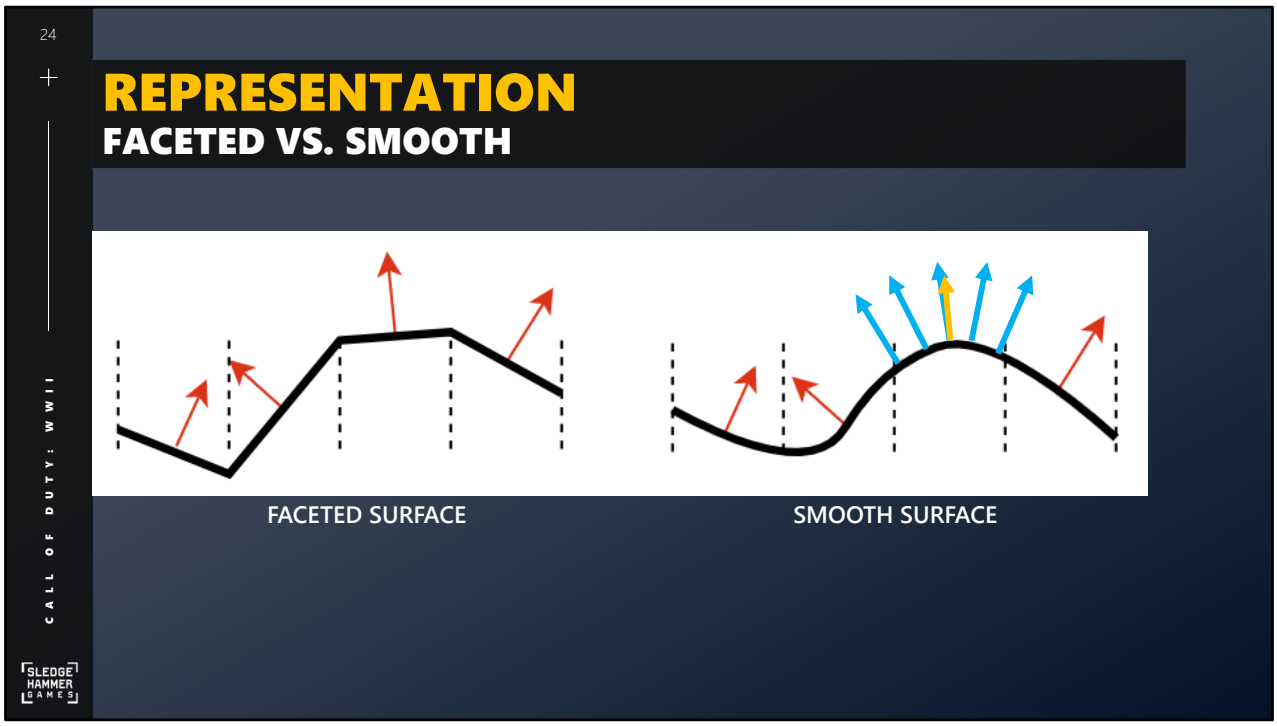
(e) Mipmapping generates a shortened normal, in green, that is the average of the two original shortened normals.



The length of the shortened normal is converted to gloss (NDF) and the shortened normal is renormalized resulting in a single pixel of normal and gloss.

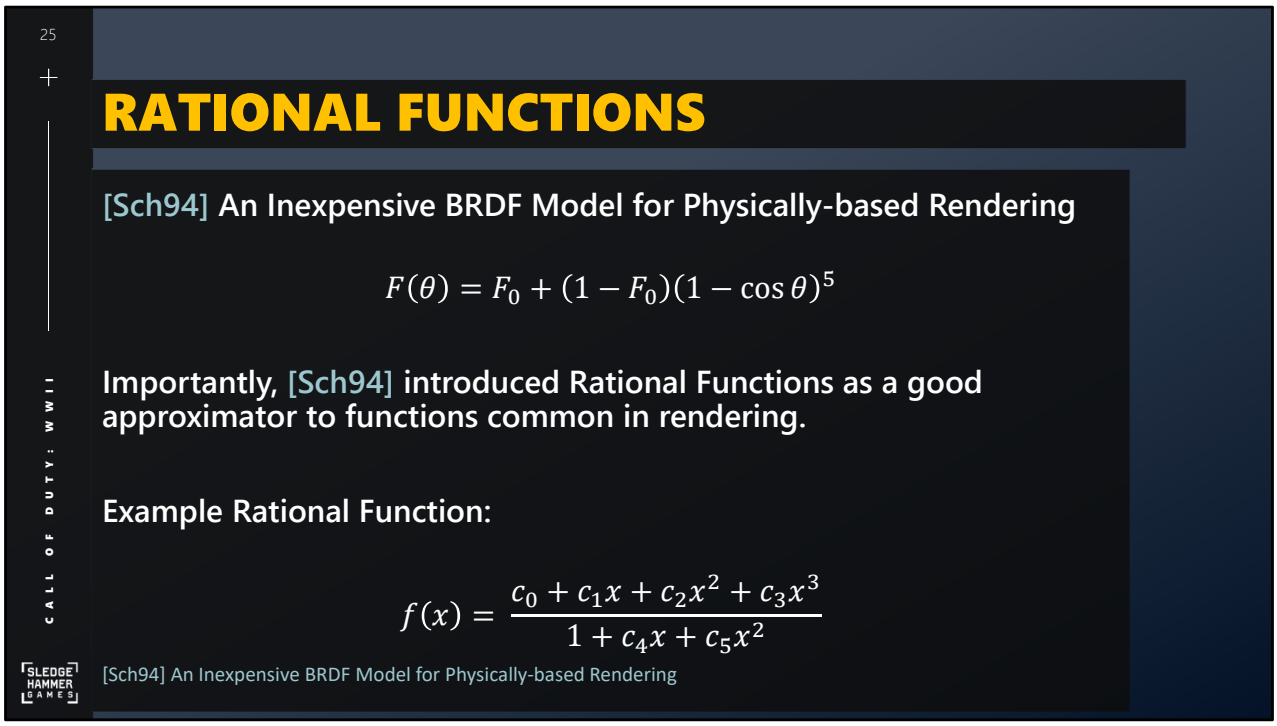


On the left is the pure white gloss map.
As we mipmap gloss, we pull in information from the normal map and introduce the normal variation within the pixel footprint of the normal map and adjust our gloss.
Notice as we get to the lower MIPs, the gloss map is darker, representing higher normal variation.



Question comes up: Should we modify gloss of mip0, and change what was authored?
Depends on how we interpret normals.
What exactly do they represent?
A faceted surface or a smooth continuous one?
Or some combination of both?
If we think of it as a faceted surface, then we should not change mip0 gloss. That's what we do in WWII to respect the wishes of artists.
For a continuous surface we could average the normals over the pixel footprint [click]
to get a shortened normal from which we can extract the implicit gloss of the pixel. [click]
Then this implicit gloss from the normal map would need to be combined with the corresponding authored gloss map.
We'll talk about how to combine gloss values a bit later.

One thing to note about the smooth surface scenario, is that under magnification this idea falls apart without some further modifications.
Under magnification the rendered pixel footprint might not cover the full extent of the source normal and gloss map so the underlying normal distribution function wouldn't be as broad.



Brief detour into Rational Functions:

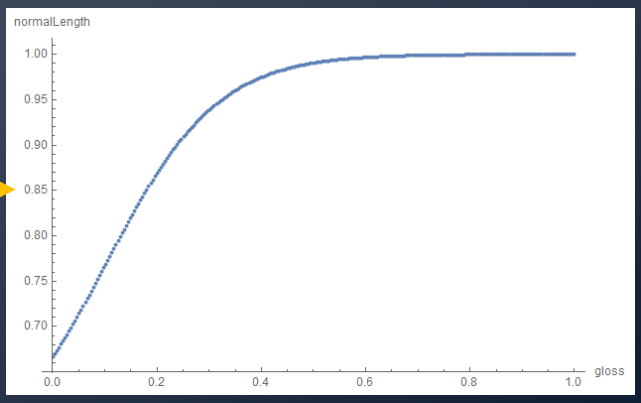
Most physically-based shaders use the Schlick approximation to Fresnel.
In the paper where this approximation was introduced, rational functions were described as good approximators to many functions we encounter in rendering.
A rational function is algebraic fraction of polynomials. See example.
Operation-wise it is a series of MADDs and a divide.

[Sch94] An Inexpensive BRDF Model for Physically-based Rendering
http://www.cs.virginia.edu/~jdl/bib/appearance/analytic%20models/schlick94b.pdf

FUNCTION PLOT

$gloss$	$\|\mathbf{n}_s\|$
0.000000	0.666670
0.003922	0.669966
0.007843	0.673329
0.011765	0.676758
…	…
0.988235	0.999952
0.992157	0.999954
0.996078	0.999956
1.000000	0.999958

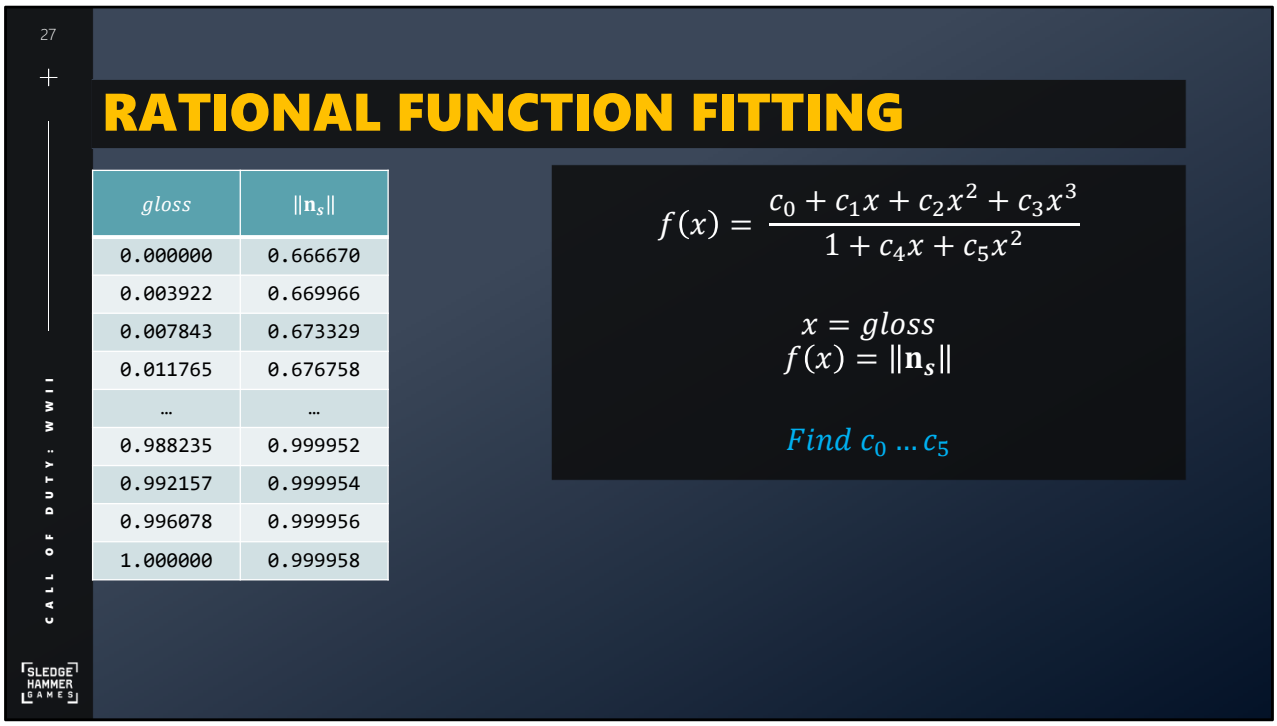
PLOT



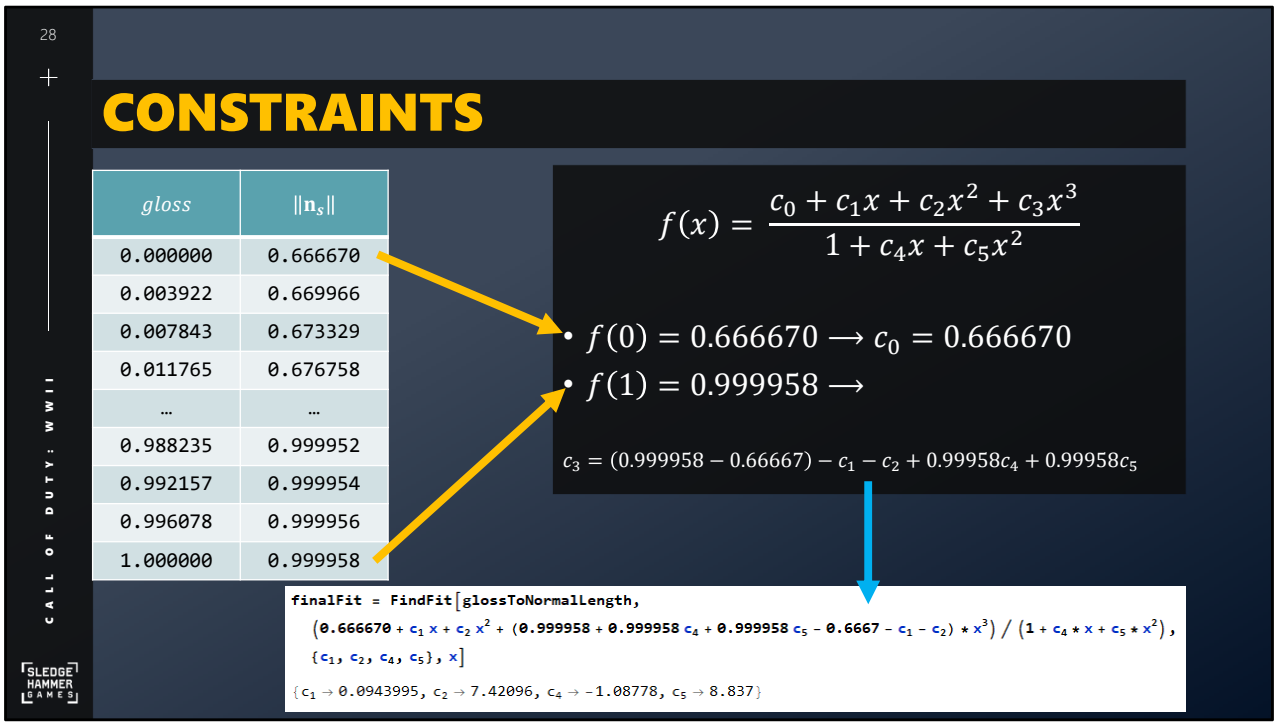
CALL OF DUTY: WWII

SLEDGE HAMMER GAMES

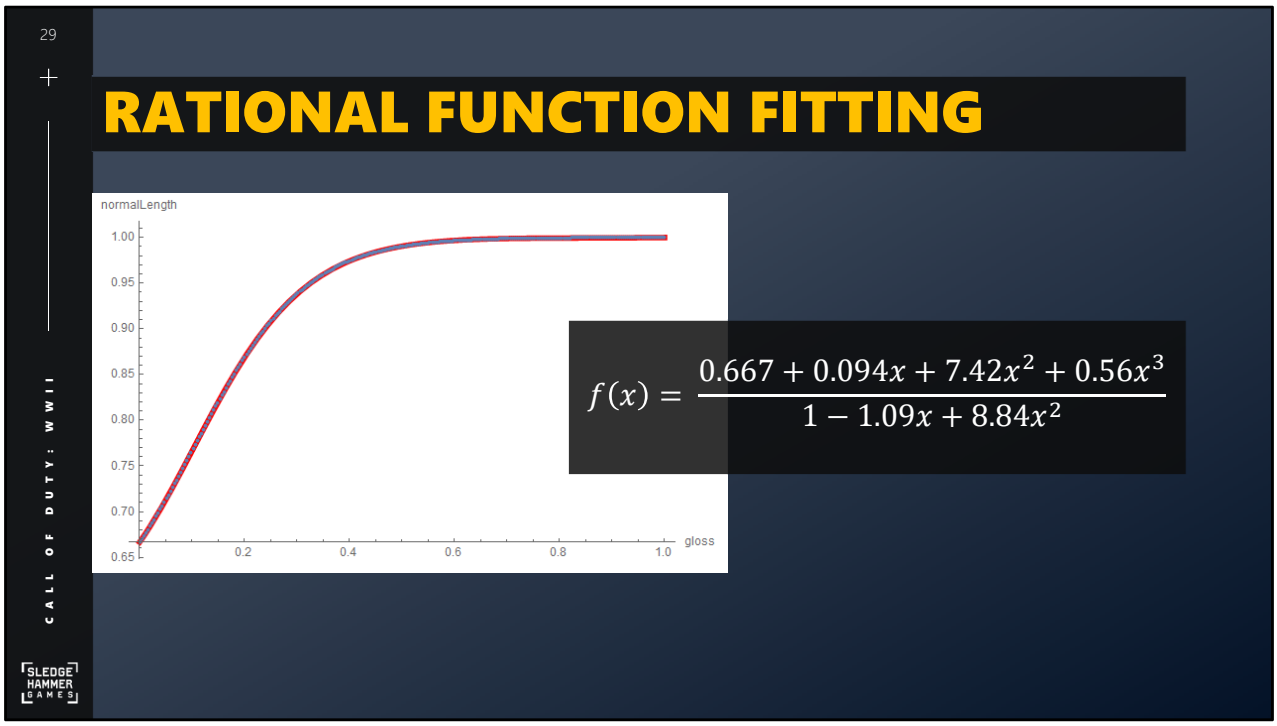
Let’s revisit the gloss to average normal length table we generated earlier.
Here we’ll plot the values that we want to find a rational function for.



Our goal is to find the coefficients c_0 through c_5, such that the Rational Function approximates our table.



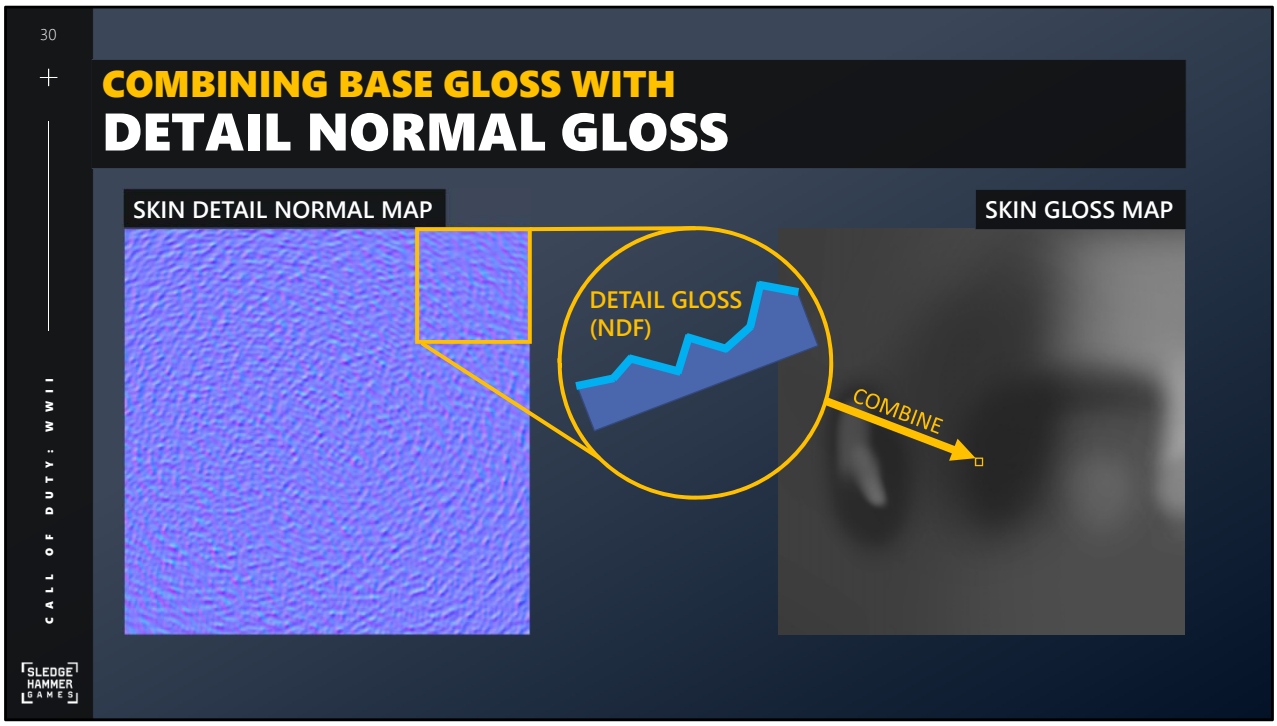
Let's introduce a couple constraints.
The first constraint is simple. We know the value of the function when x=0. So c_0 = 0.66667.
We are, in effect, fixing the start point of the curve. This removes one degree of freedom.
We wouldn't necessarily apply this constraint since we can accept some error at this end of the curve, but we are doing this purely as a demonstration.
The second constraint is important since small errors at this end of the curve can lead to wildly different results.
With the second constraint, we are fixing the end point of the curve.
With some rearranging we can remove c_3. This is an arbitrary choice, we could remove any of the other coefficients.
For this simple example, we'll just use Mathematica's FindFit function to find the best coefficients for our rational function.



Here, the red is the rational function, overlaid on top of the blue which is our table values. It's a really good fit.
Once again this was done purely as a demonstration but
In later sections, we'll see other examples where rational functions act as good approximators as well.

[Question from the Q&A session]
Why use a rational function when the divide is so expensive? Why not just a polynomial?

- It's true the divide is the most expensive part of the rational function (shader code-wise), but it's an important ingredient and allows canceling of some of the polynomial "waviness" of the numerator, using the denominator. This allows you to represent "cliffs", "hills" and "plateaus" more accurately. A much higher-order polynomial would be needed to achieve the same error for many of these shapes.



We apply detail normal maps on top of some of our materials, like skin.
When generating mipmaps for detail normal maps, we push some normal variation into a detail gloss (NDF) channel.
If you imagine that detail normal map mipmapped to a 4x4 pixel texture, we need to represent the normal variation in that top-right pixel as gloss.
We need to combine this detail gloss [click]
with the base gloss of our material in our pixel shaders at run-time.
How do we combine gloss?
We're going to see this is matter of combining many normals of one NDF with the normals from the other NDF.

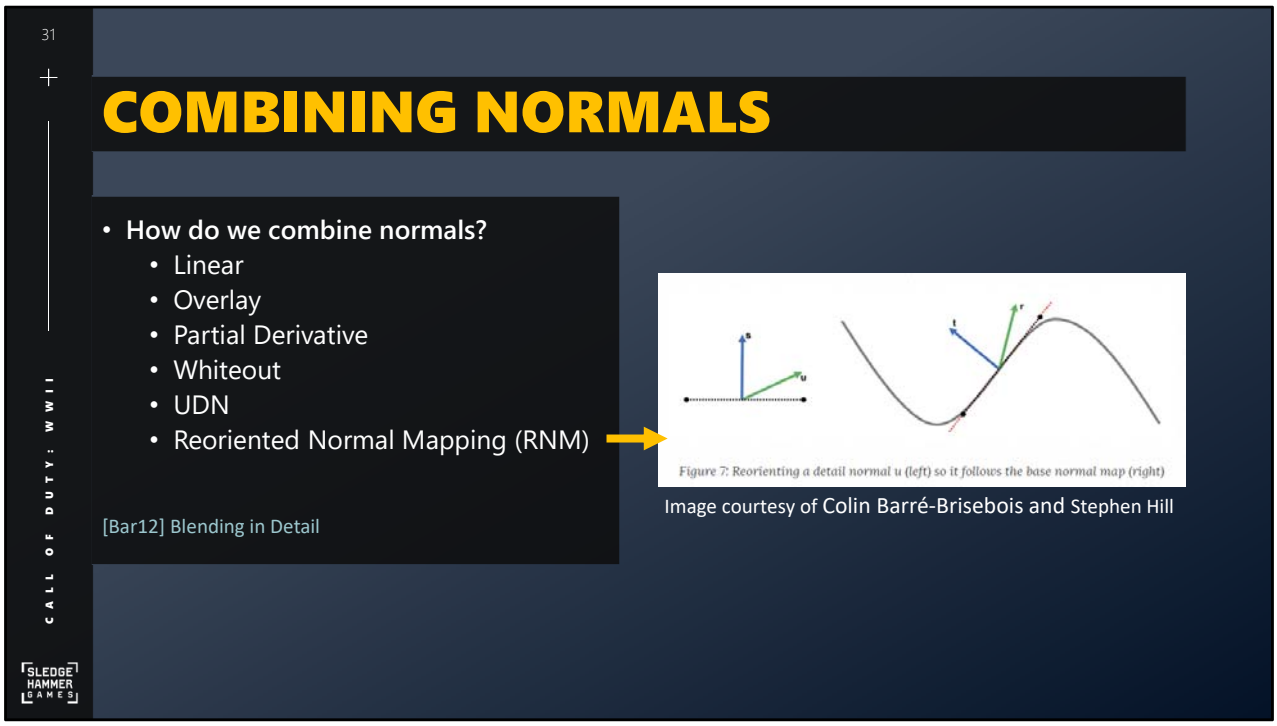
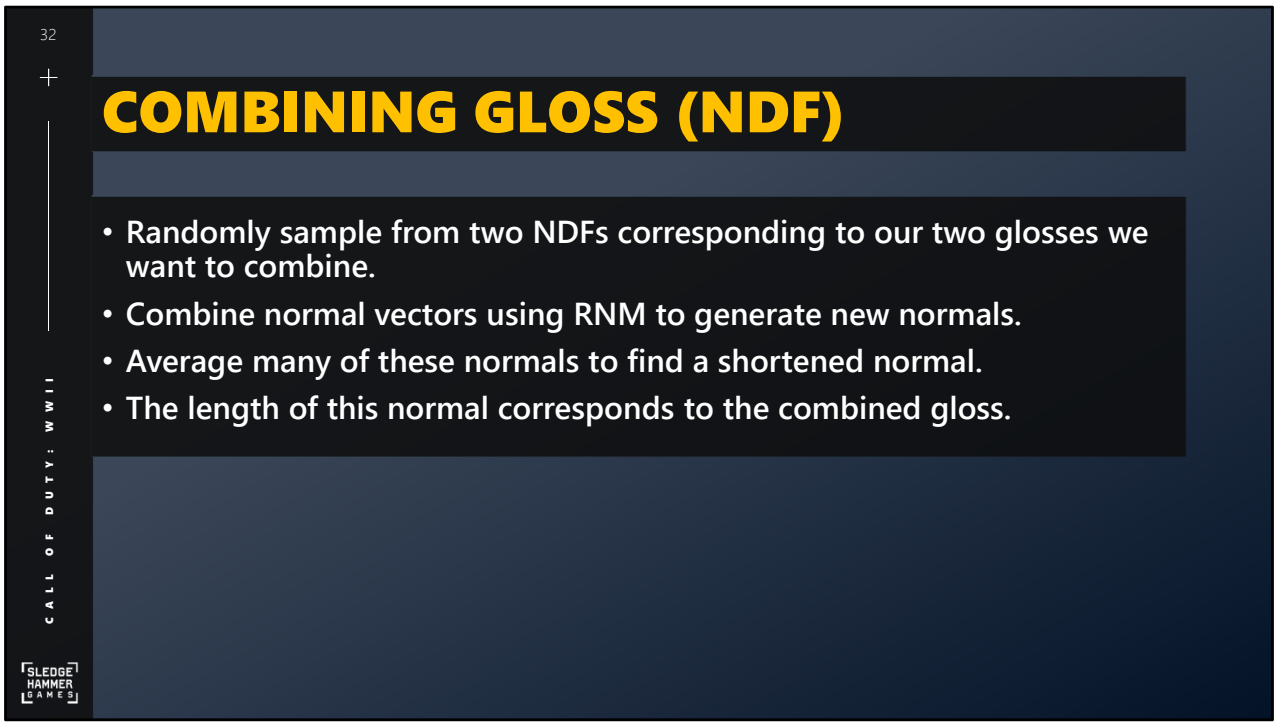


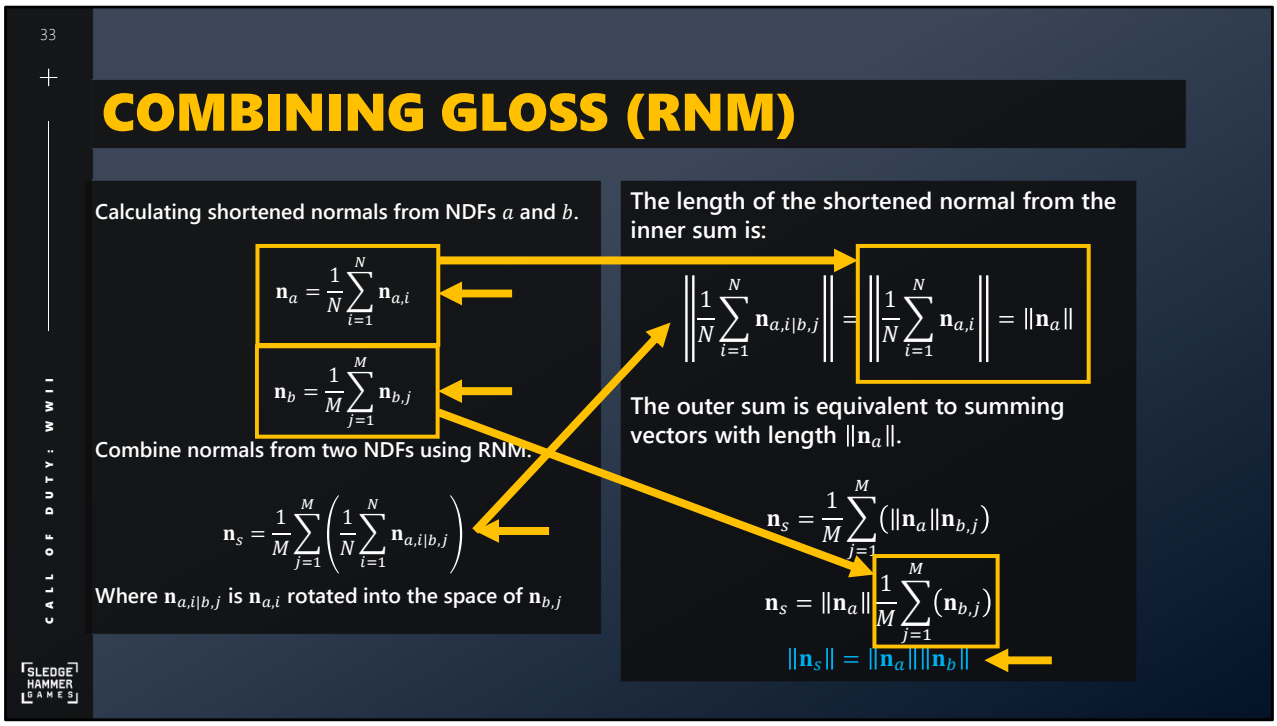
Image courtesy of Colin Barré-Brisebois and Stephen Hill

The first step to combining gloss is determining what method we use to combine normals. A great reference is a blog post titled “Blending in Detail”. In that post, there is a survey of the many techniques that can be used.
The normal combining method we choose is important since how we combine randomly chosen normals from two NDFs is going to determine how our glosses combine.
What we are going to do is combine normal vectors using Reoriented Normal Mapping. In Reoriented Normal Mapping, the detail normal is rotated into the space of the base normal.

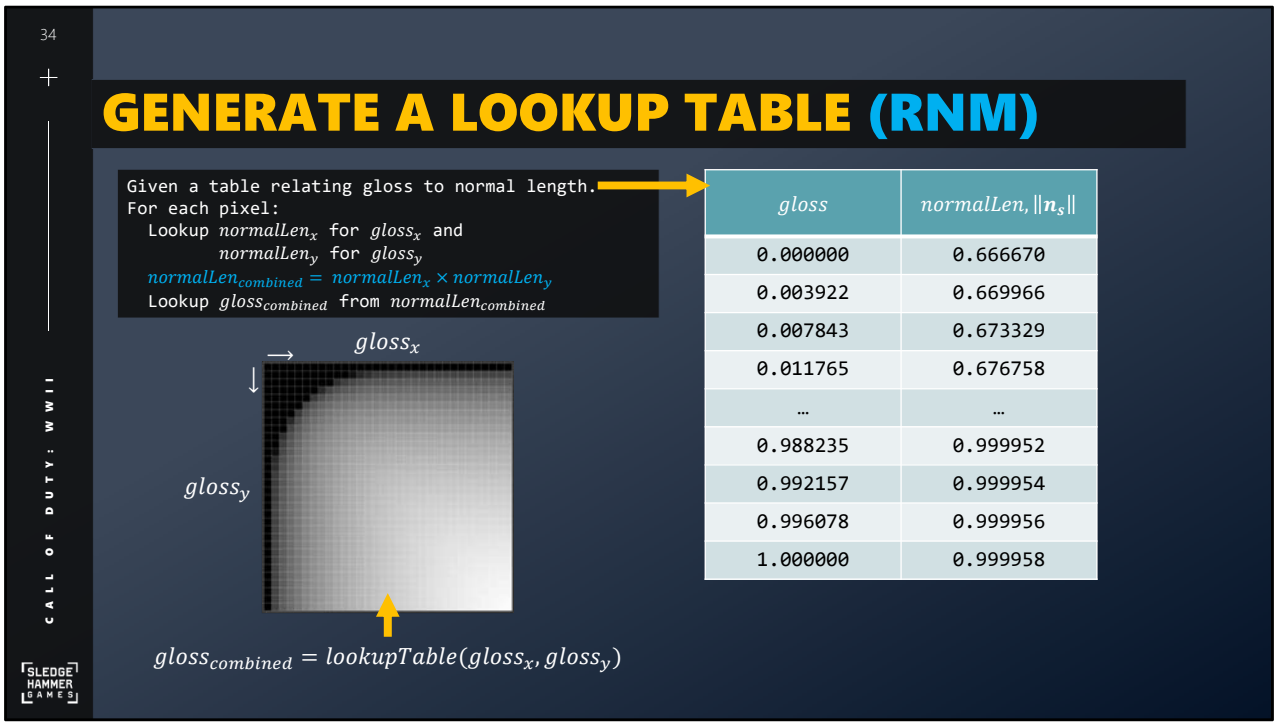
[Bar12] Blending in Detail
http://blog.selfshadow.com/publications/blending-in-detail/



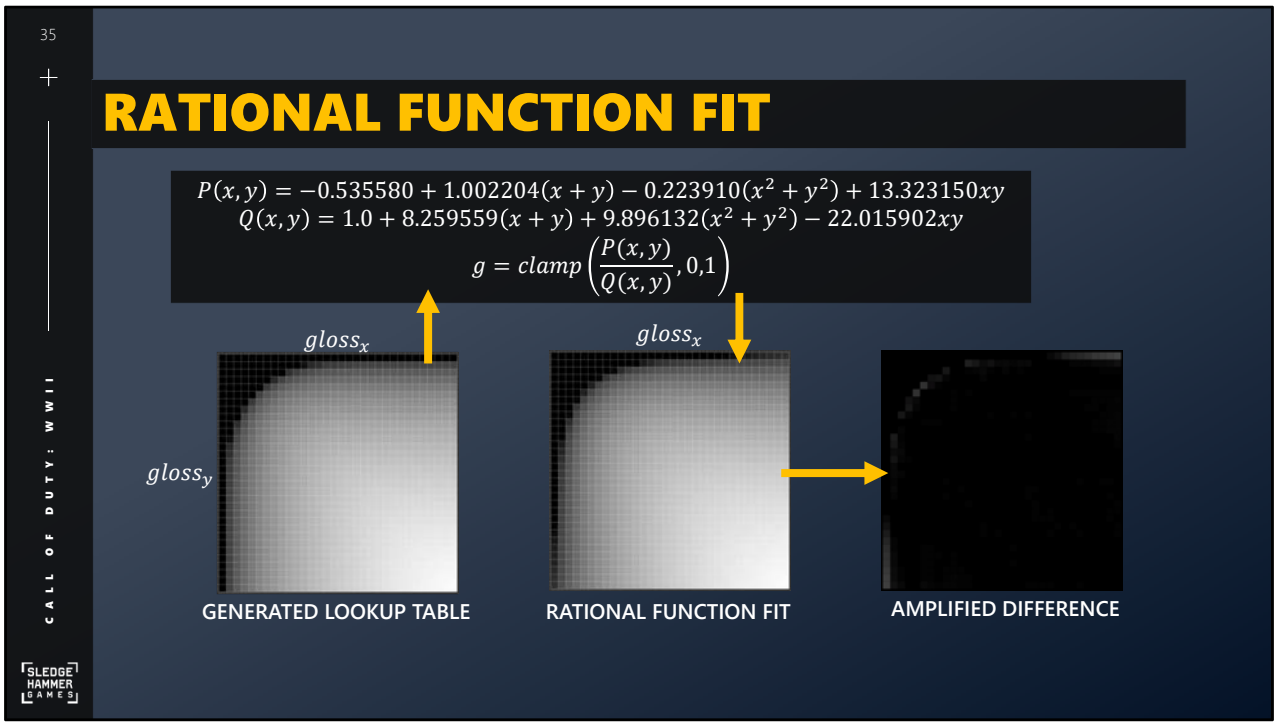
Now that we've decided on a method to combine normals, we can extend this to combine glosses (or NDFs)
We are going to sample randomly from the two normal distribution functions and combine the normal vectors using RNM.
Then we average a bunch of these combined normals to find a shortened normal.
The length of this shortened normal corresponds to the combined gloss.
Going through a process like this is one way to calculate how to combine gloss.



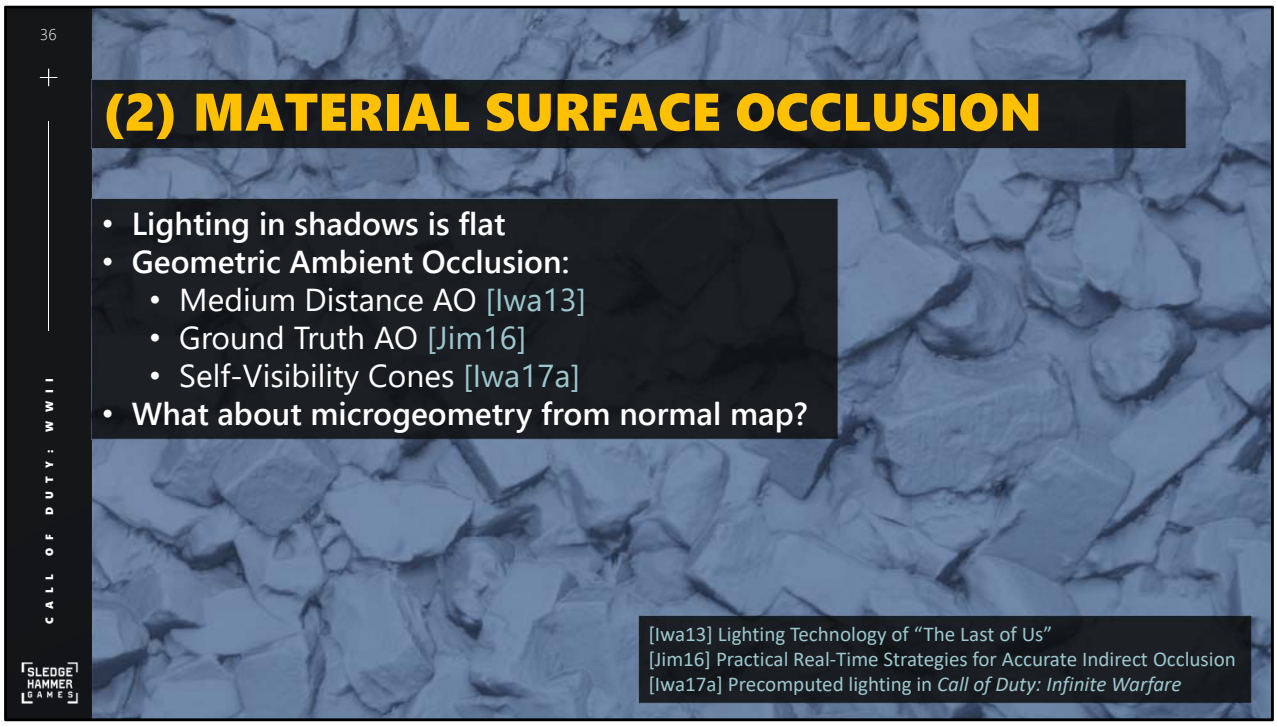
But we can also reason mathematically about it and come up with a short-cut to the previous process.
Once again, if we use Reoriented Normal Mapping, then we are rotating one normal into the space of the other.
To start,
[click] we are going to describe how we calculate shortened normals from two NDFs: a and b. We just average a collection of random samples from the NDFs.
[click] Then, we combine normals from the two NDFs using RNM.
There's an inner and outer sum here and we can just imagine combining every normal from one distribution with every normal from the other.
[click] This makes the inner sum equivalent to the length of the shortened normal in one of the distributions we are combining.
[click] Then applying the outer sum is equivalent to multiplying the lengths of the shortened normals in the two distributions.
[click]



On this next slide, I want to make clear, there are two lookup tables. The one to the right is going to be used to generate the one on the left.
Combining gloss values boils down to a few lookups and a multiply.
To generate our gloss combining lookup table, we just iterate over every pixel (or entry) and we lookup two normal lengths for our gloss values.
Then we multiply the normal lengths to get our combined normal length, which we convert back to gloss.



Once we have our generated lookup table, we can fit a rational function to it.
Notice we can reduce the degrees of freedom in the rational function since the function is symmetric with respect to its arguments.
In this case, the rational function is a very good approximation to the original function.
We use this rational function in our pixel shader to combine detail gloss with base gloss.

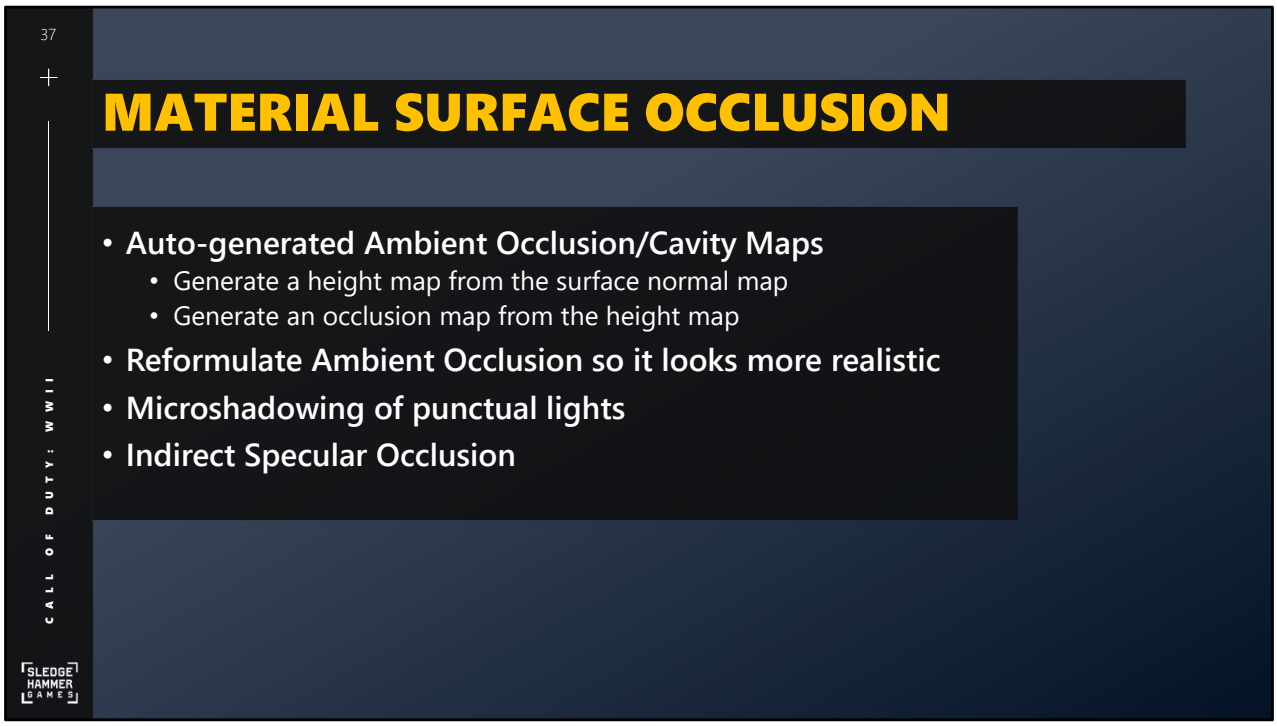


A common complaint from artists is that shadow areas look too flat.
And this was something we wanted to address in WWII since one of our art directives was to render during magic hour, like the movie Revenant.
We have several types of Geometric Ambient Occlusion in our engine: Medium Distance AO, Ground Truth AO, Self-Visibility Cones.
But we needed a solution for occlusion from microgeometry described by the normal map.
We call this material surface occlusion.

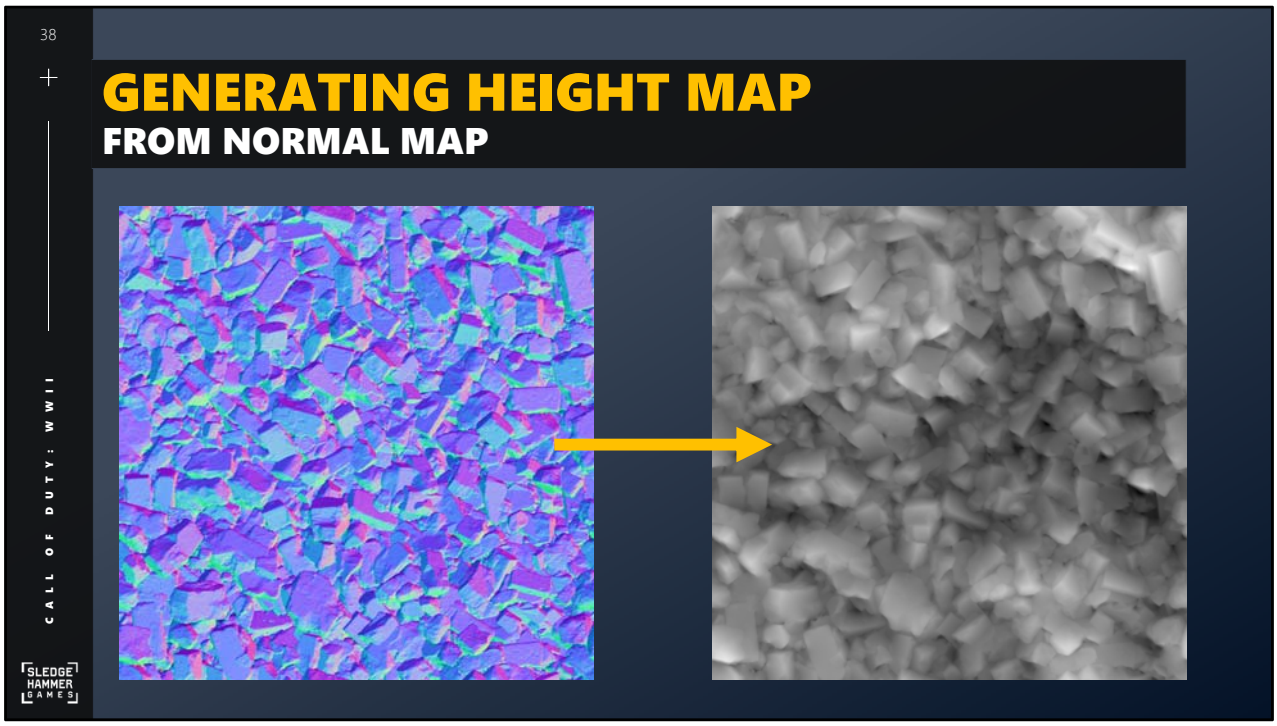
[Iwa17a] Precomputed Lighting in Call of Duty Infinite Warfare
http://advances.realtimerendering.com/s2017/

[Jim16] Practical Real-Time Strategies for Accurate Indirect Occlusion
http://blog.selfshadow.com/publications/s2016-shading-course/

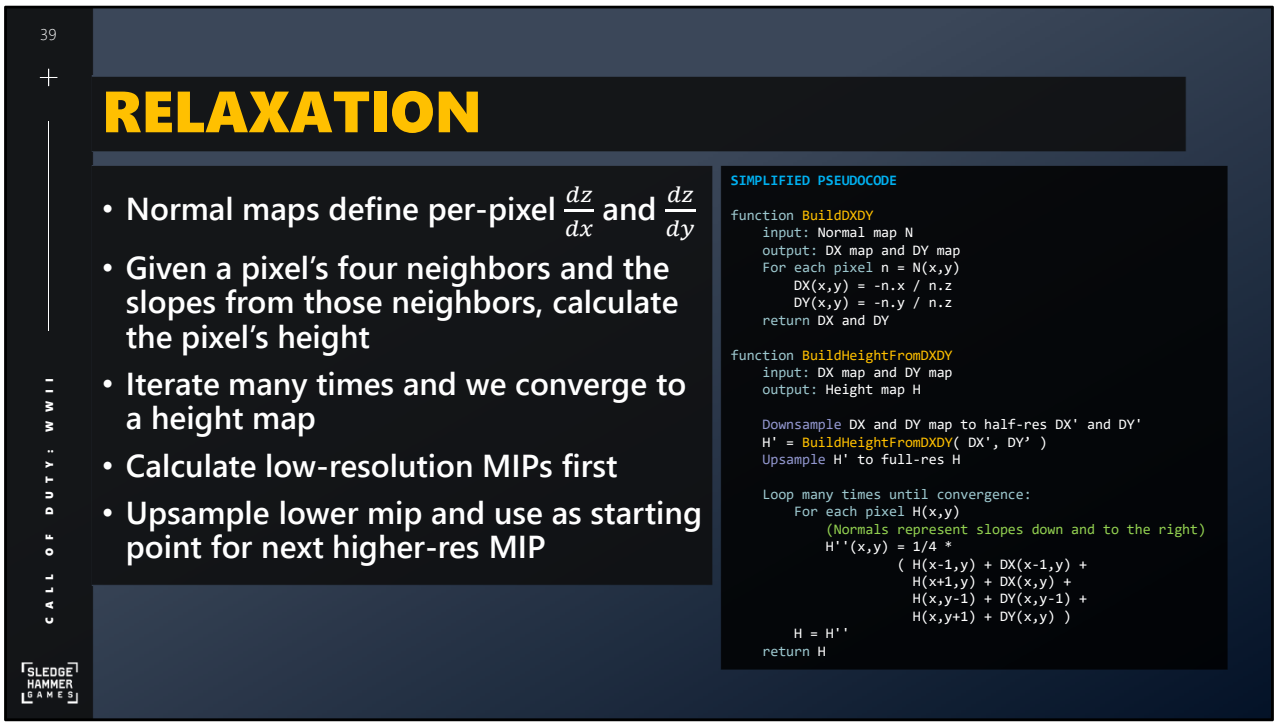
[Iwa13] Lighting Technology of "The Last of Us"
http://miciwan.com/SIGGRAPH2013/Lighting%20Technology%20of%20The%20Last%20Of%20Us.pdf



These are the topics we'll cover in this section.
The first two cover how we auto-generate the ambient occlusion or cavity map.
Then we are going to rethink how we interpret ambient occlusion so we don't get overly dark results.
Next, we'll discuss how we use occlusion values to shadow punctual light sources.
Finally, we'll talk about how we deal with indirect specular occlusion, which isn't just limited to the effect of cavity maps,
we also apply it using geometric ambient occlusion.

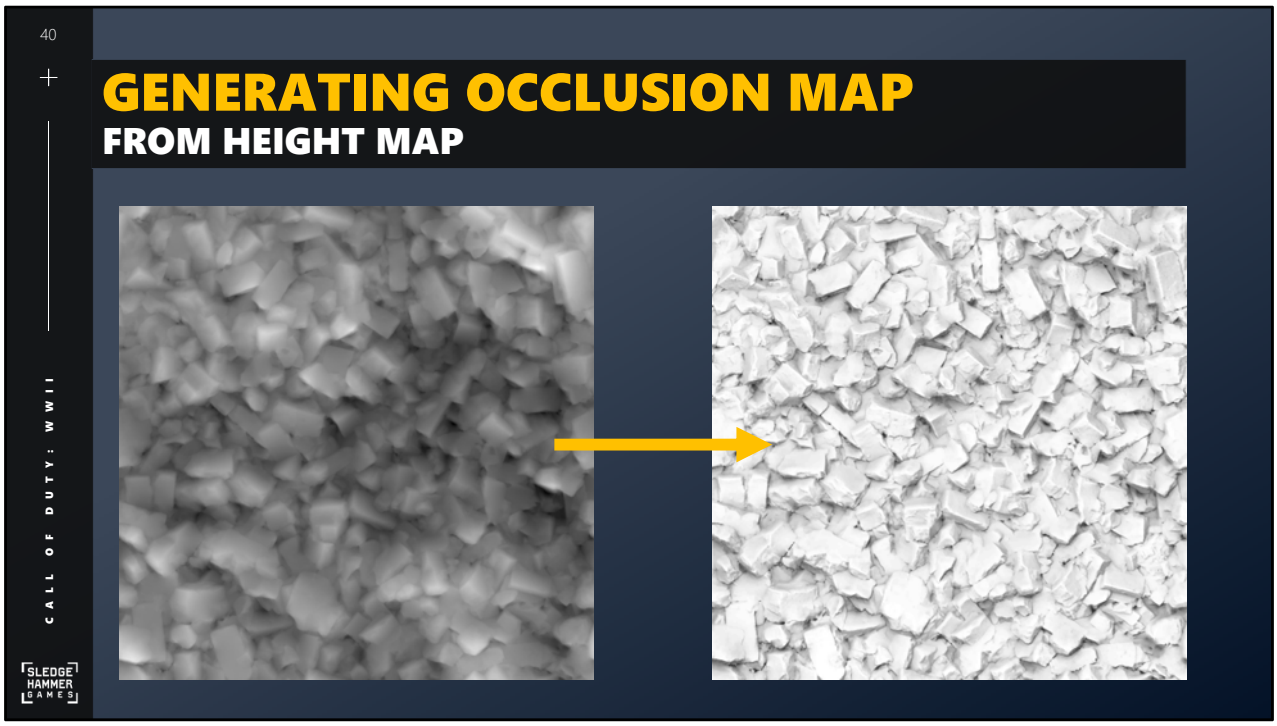


The first step in generating the cavity map is generating a height map from the normal map.

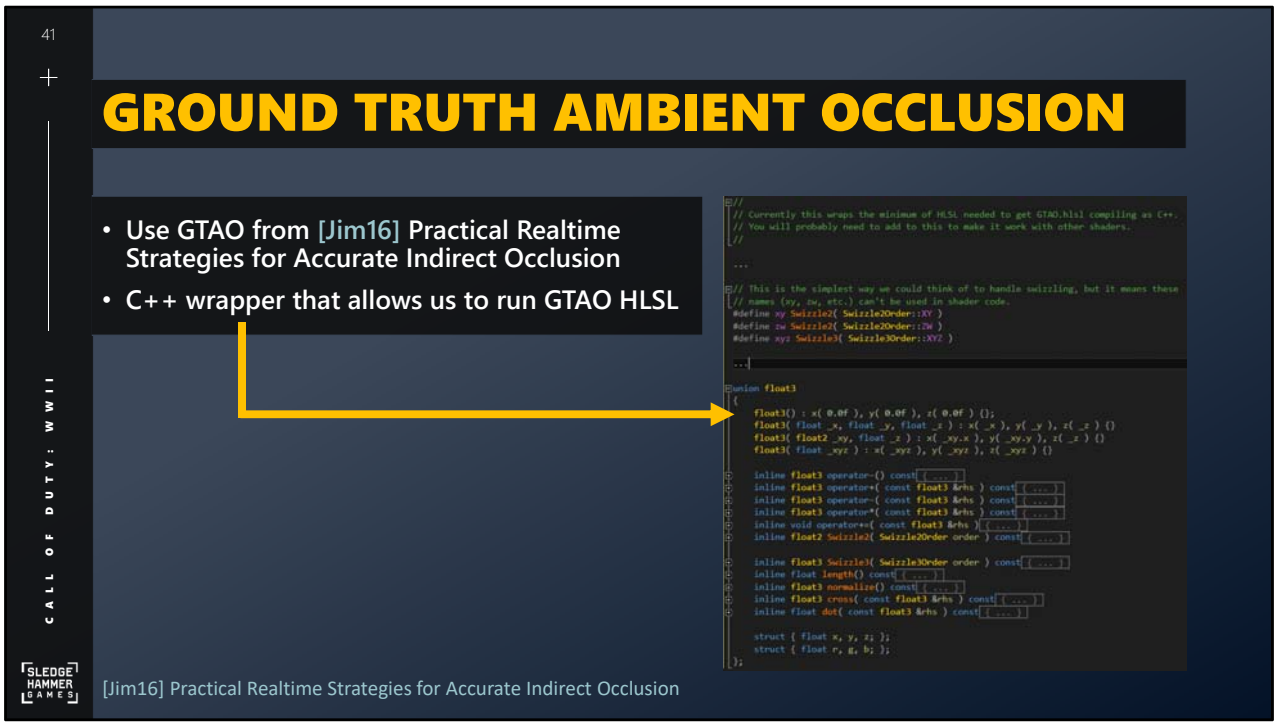


We do this using relaxation.
Normal maps define per-pixel change in height from neighboring pixels.
So if we have the neighbors of a pixel and the slopes from those neighbors we can calculate the pixel's height.
If we iterate many times, we converge to a height map.
Our process calculates low-resolution MIPs first, then upsamples those results and uses that as the starting point for next MIP up.

NOTE: Simplified pseudocode assumes normal represents slopes down and to the right, but you can also reformulate with normals at pixel center.



Next, we generate the cavity or occlusion map from the height map.

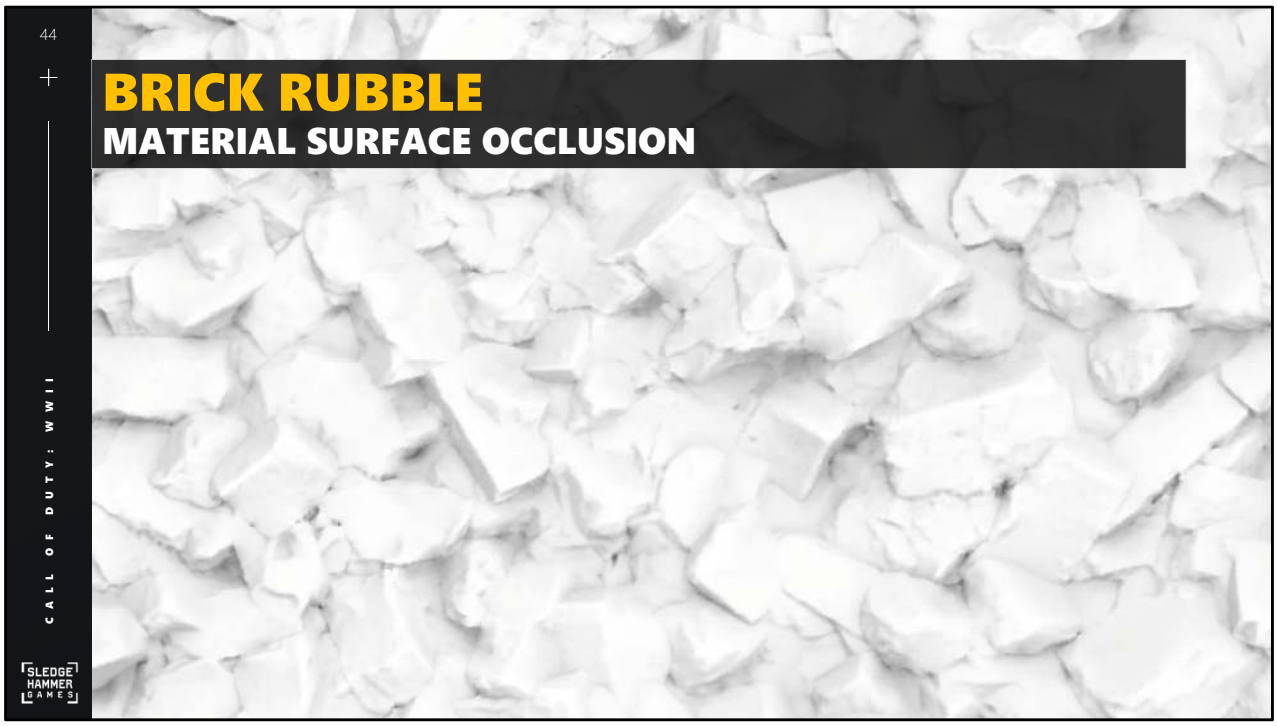


Doing this is fairly straightforward.
We used Ground Truth Ambient Occlusion, which is a screen-space ambient occlusion technique, applied to the height map.
Jorge Jimenez modified it slightly to account for an orthographic projection.
Then we wrote a C++ wrapper that allowed HLSL code to be executed within our texture converters.
All surfaces with normal maps have an occlusion map generated for them.

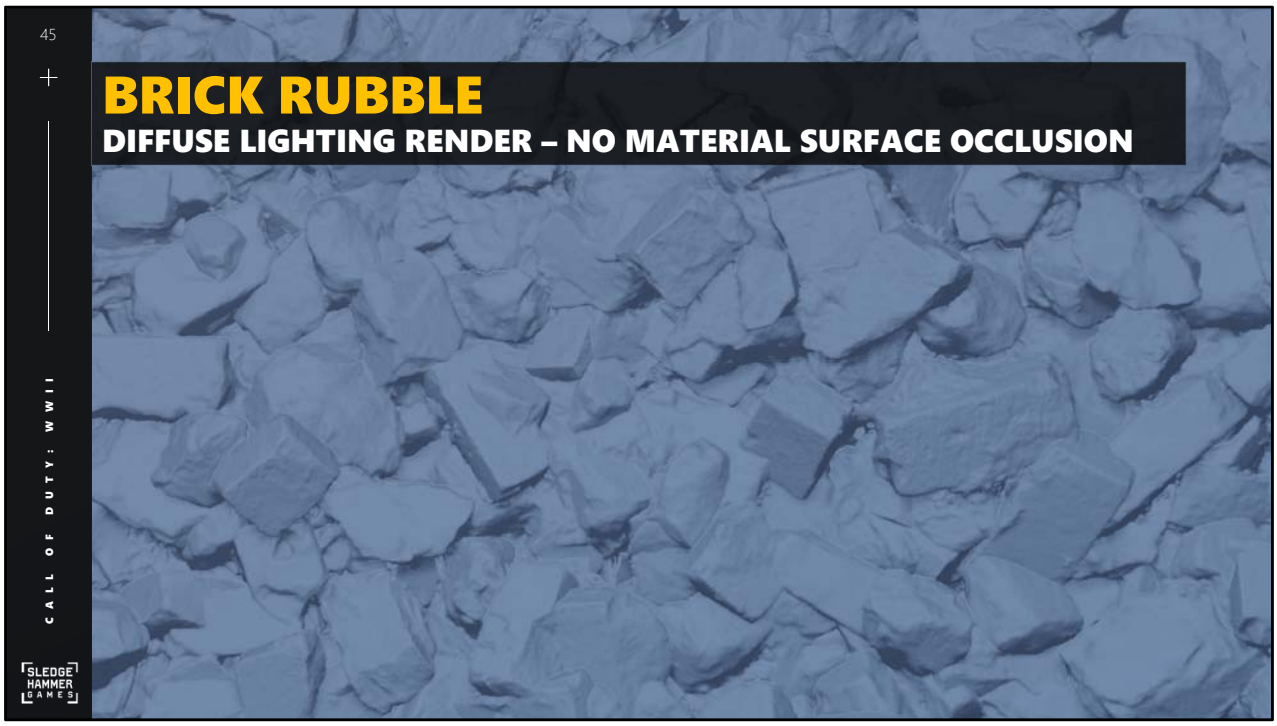
[Jim16] Practical Realtime Strategies for Accurate Indirect Occlusion
http://blog.selfshadow.com/publications/s2016-shading-course/

This is the diffuse albedo texture for a brick rubble material.

This is the normal map for the material.

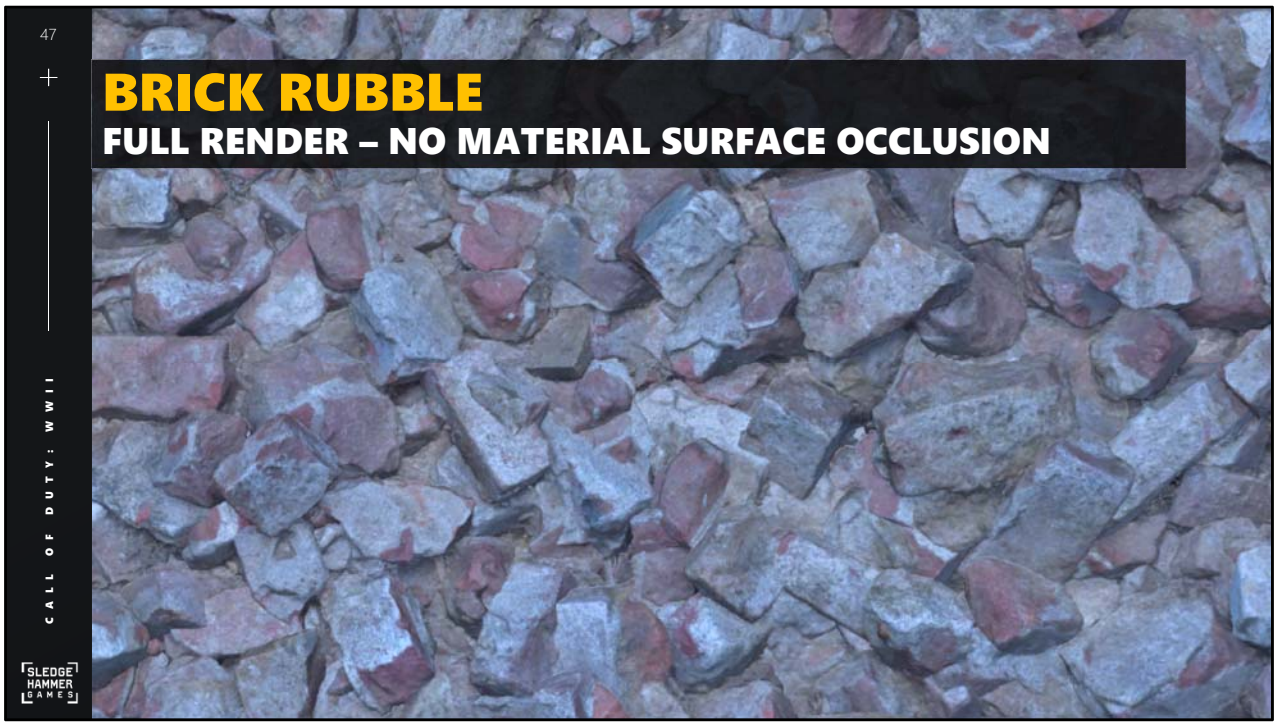


Here is the material surface occlusion map generated from the normal map.



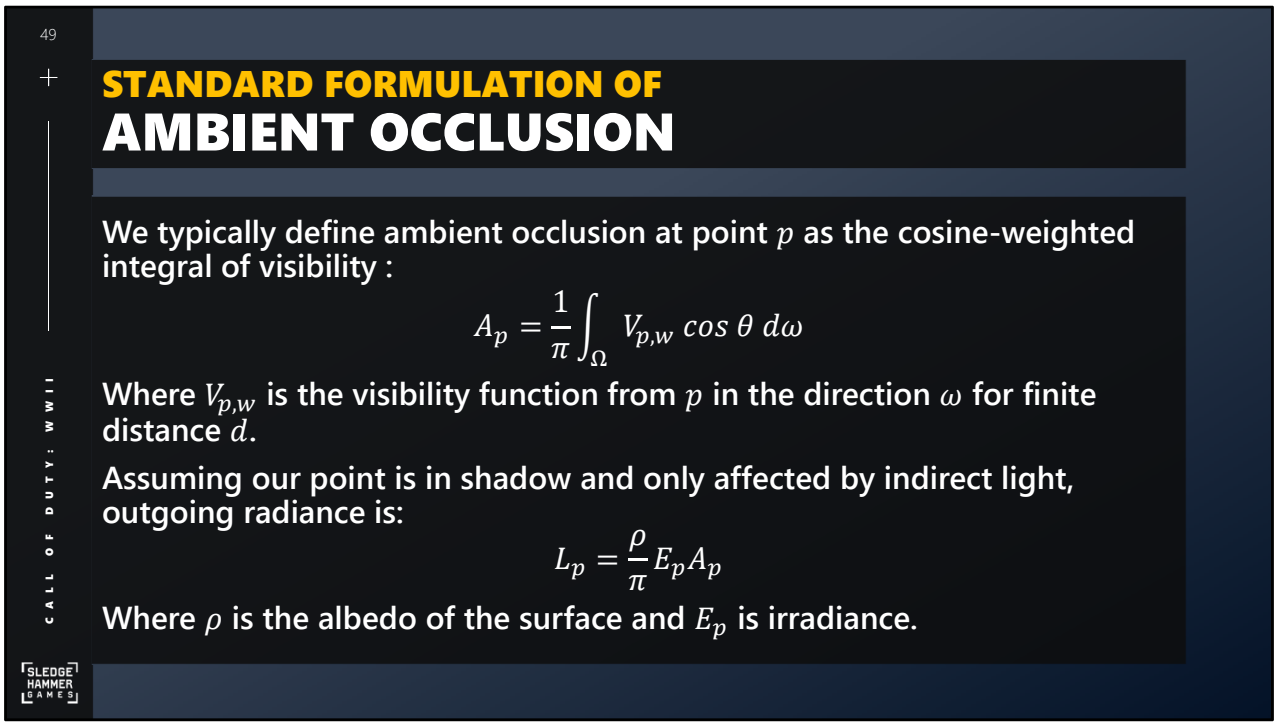
This is diffuse lighting render with no occlusion applied.

Here is material surface occlusion applied.
[Toggle with last frame]

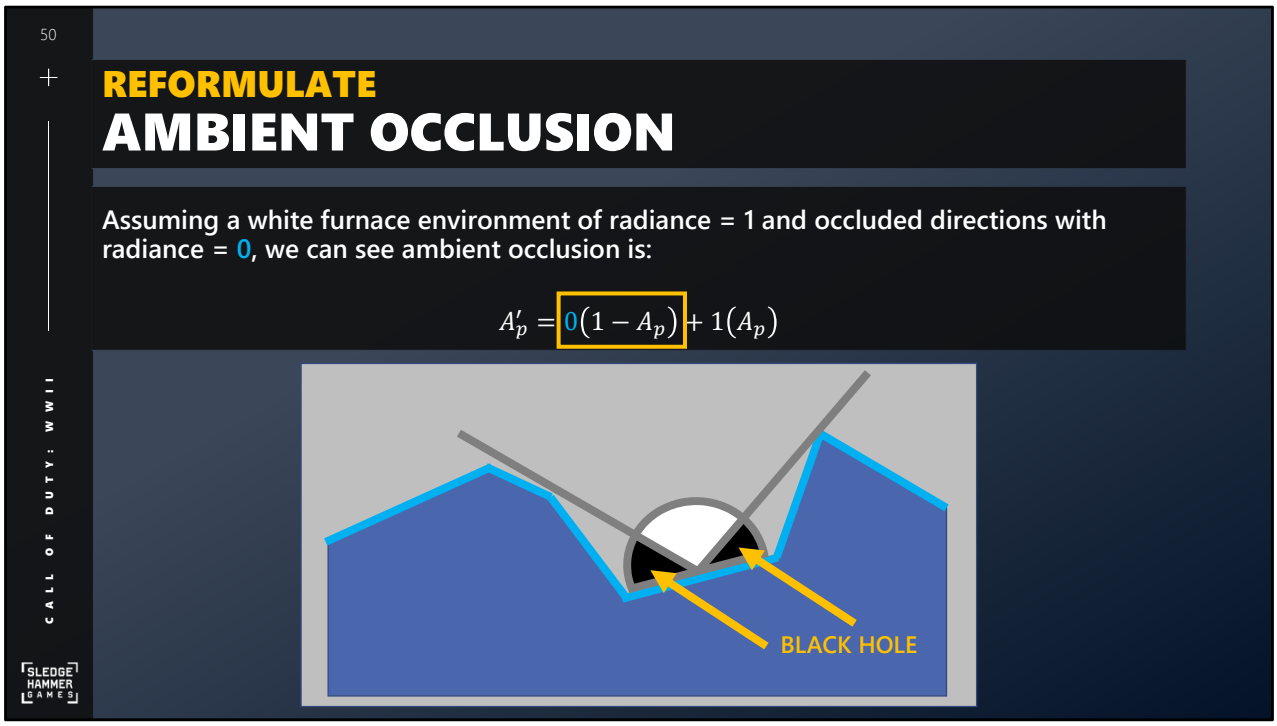


Now, we have a full render. This has no material surface occlusion.

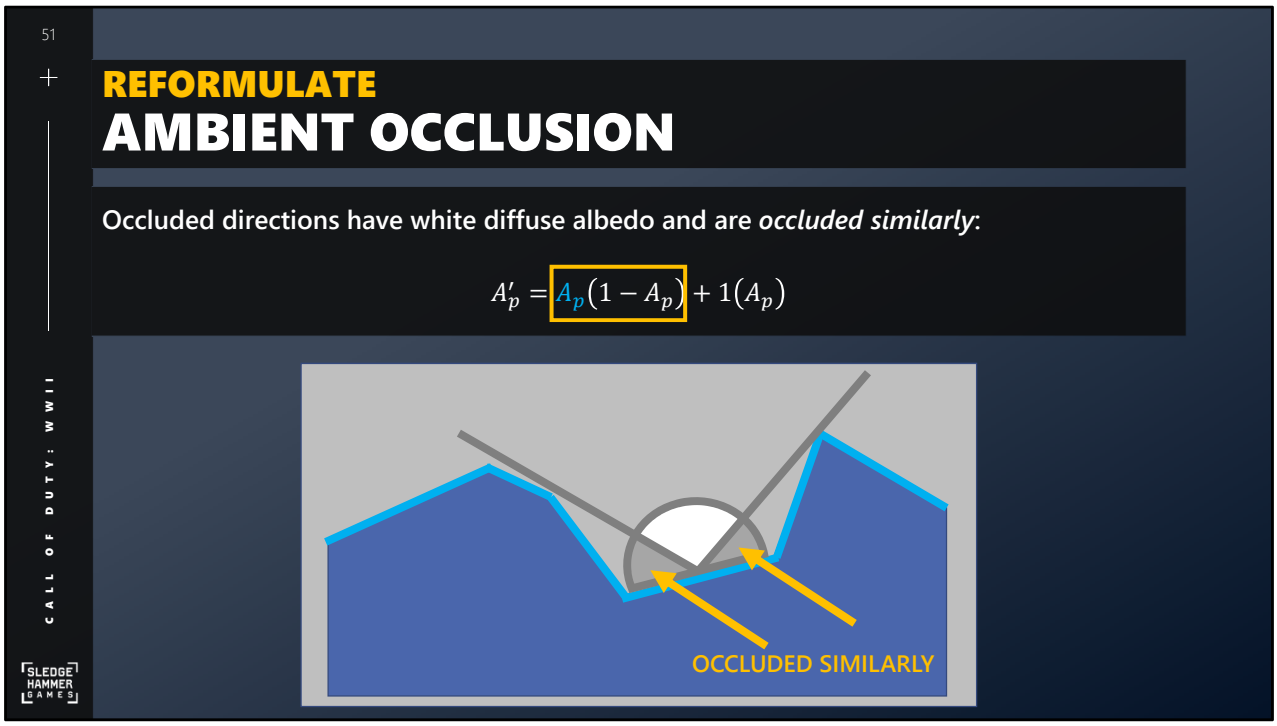
This is a full render with material surface occlusion applied.



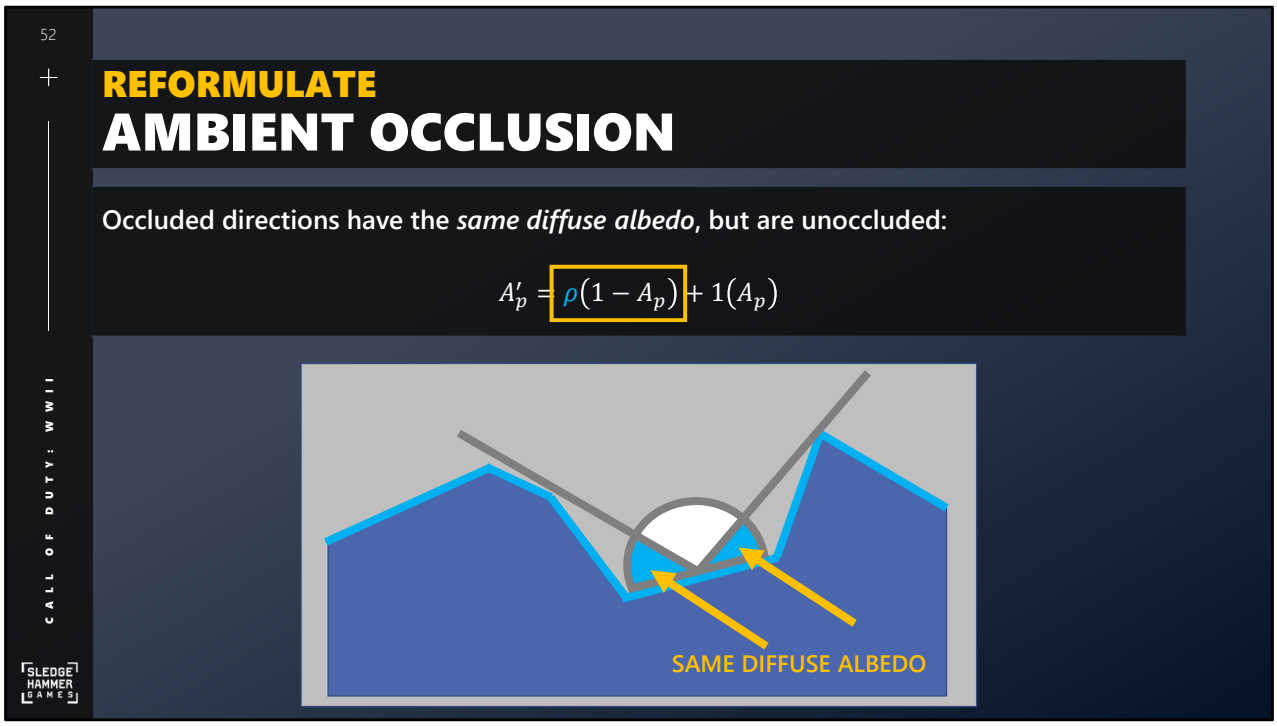
Ambient occlusion is usually defined as the cosine-weighted integral of visibility.
So indirect lighting becomes a simple attenuation of irradiance using our ambient occlusion value.
This is the typical technique for applying ambient occlusion, but it usually leads to overly dark results.



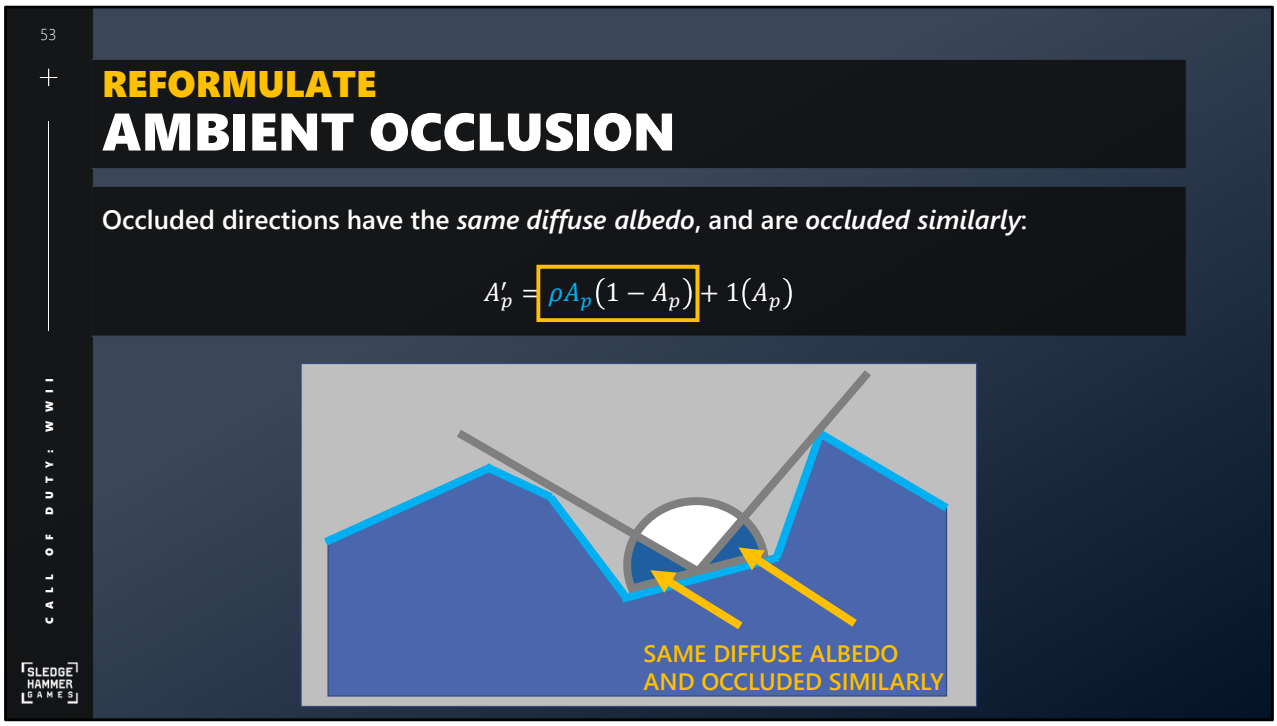
If we think of ambient occlusion as a cosine-weighted integral of radiance in a white furnace environment, we can see that occluded directions contribute zero radiance.
But this isn't what happens in the real-world.
Occluded directions are not just black holes.



We're going to reformulate ambient occlusion with some better assumption about our occluded directions.
The first of our examples assumes occluded directions have white diffuse albedo and are occluded similarly.



Our next example assumes occluded directions have the same diffuse albedo but are unoccluded.



And finally, we can assume occluded directions have the same diffuse albedo and are occluded similarly.
This is a reasonable assumption since pixels are likely to be similar to neighboring pixels.

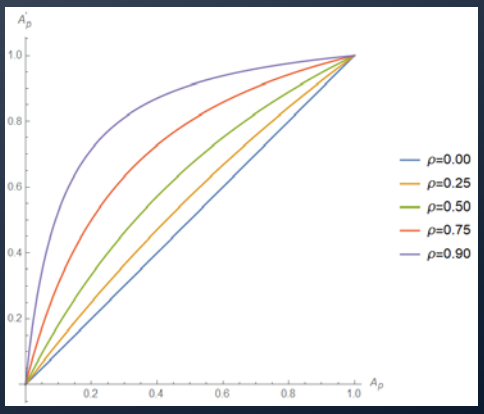
54

INTERREFLECTION MODEL OF AMBIENT OCCLUSION

Occluded directions have same radiance as outgoing diffuse radiance of pixel being shaded:

$$A'_p = \frac{A_p}{1 - \rho + \rho A_p}$$

[Ste96] Towards Accurate Recovery of Shape from Shading under Diffuse Lighting



CALL OF DUTY: WWII

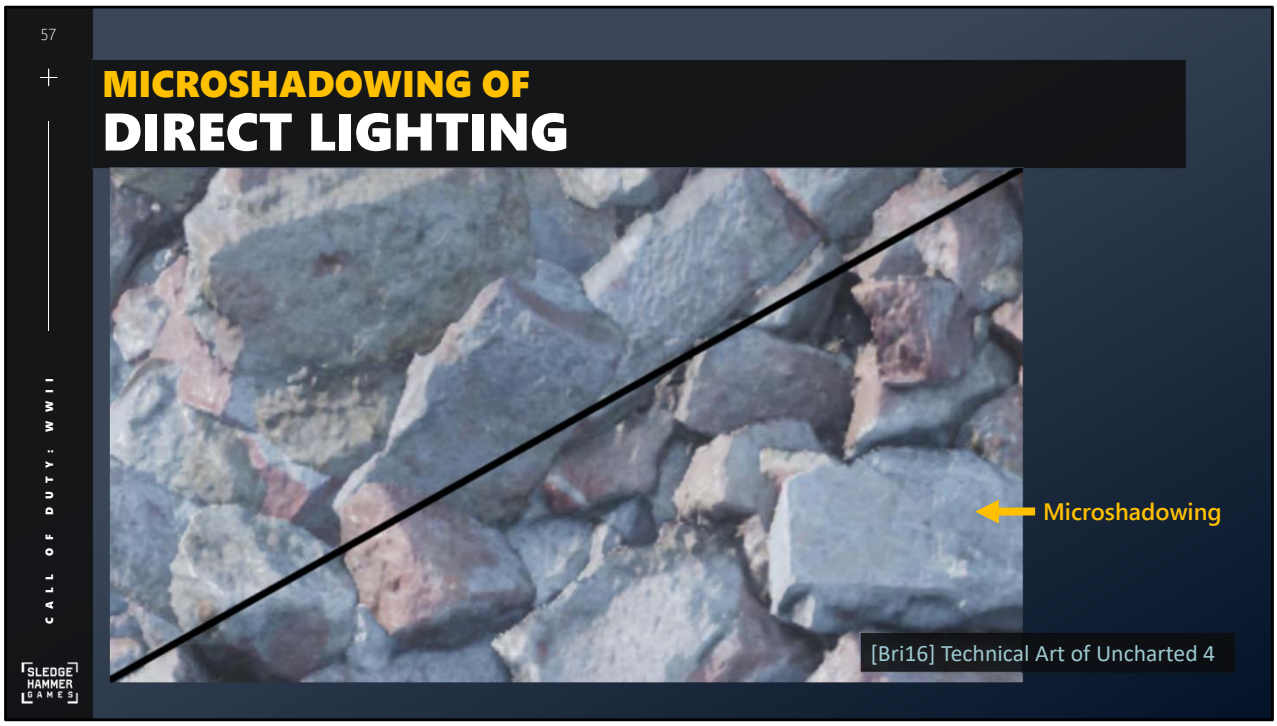
SLEDGEHAMMER GAMES

Here, we go even further with our assumption.
Thanks to Jorge Jimenez, who suggested the Interreflection Model of Ambient Occlusion based on conversations with Naty Hoffman.
Now, occluded directions have the same radiance as the outgoing diffuse radiance of the pixel being shaded.
On the graph to the right, we can see the effect that albedo has on adjusting our ambient occlusion value.
High albedo materials will reflect more light in the occluded directions and our adjusted ambient occlusion value will tends towards 1.

[Ste96] Towards Accurate Recovery of Shape from Shading under Diffuse Lighting
http://www.cim.mcgill.ca/~langer/MY_PAPERS/Stewart-Langer-CVPR96.pdf

Here is a render of raw Material Surface Occlusion values.
Applying the Interreflection Model of Ambient Occlusion is a subtle change. [click]

Not only is the occlusion brighter, but there is a little bit of color coming from the underlying red brick albedo texture.
This slide should look just a little warmer than the previous.
[Toggle with last frame]
This lends just a touch of saturation to our occluded areas.



Next, we'll going to discuss Microshadowing of Direct Lighting.
Another version of this technique from talk in SIGGRAPH 2016: Technical Art of Uncharted 4

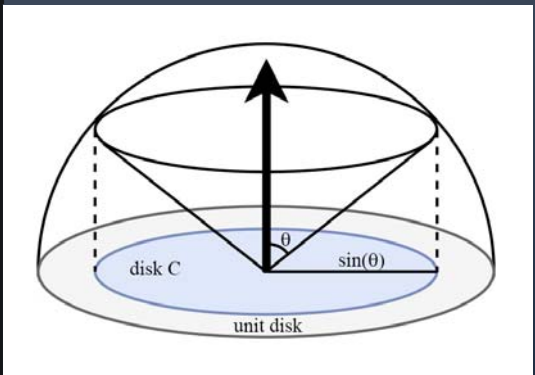
We are going to adopt the same term Naughty Dog used, "Microshadowing", for the effect of attenuating direct lighting using our occlusion map.

[Bri16] Technical Art of Uncharted 4
http://advances.realtimerendering.com/other/2016/naughty_dog/index.html

58

CONVERTING OCCLUSION TO EQUIVALENT CONE ANGLE

$$V = A'_p$$



$$\frac{cos - weighted\ visibility}{hemisphere\ visibility} = \frac{area\ of\ disk\ C}{area\ of\ unit\ disk}$$

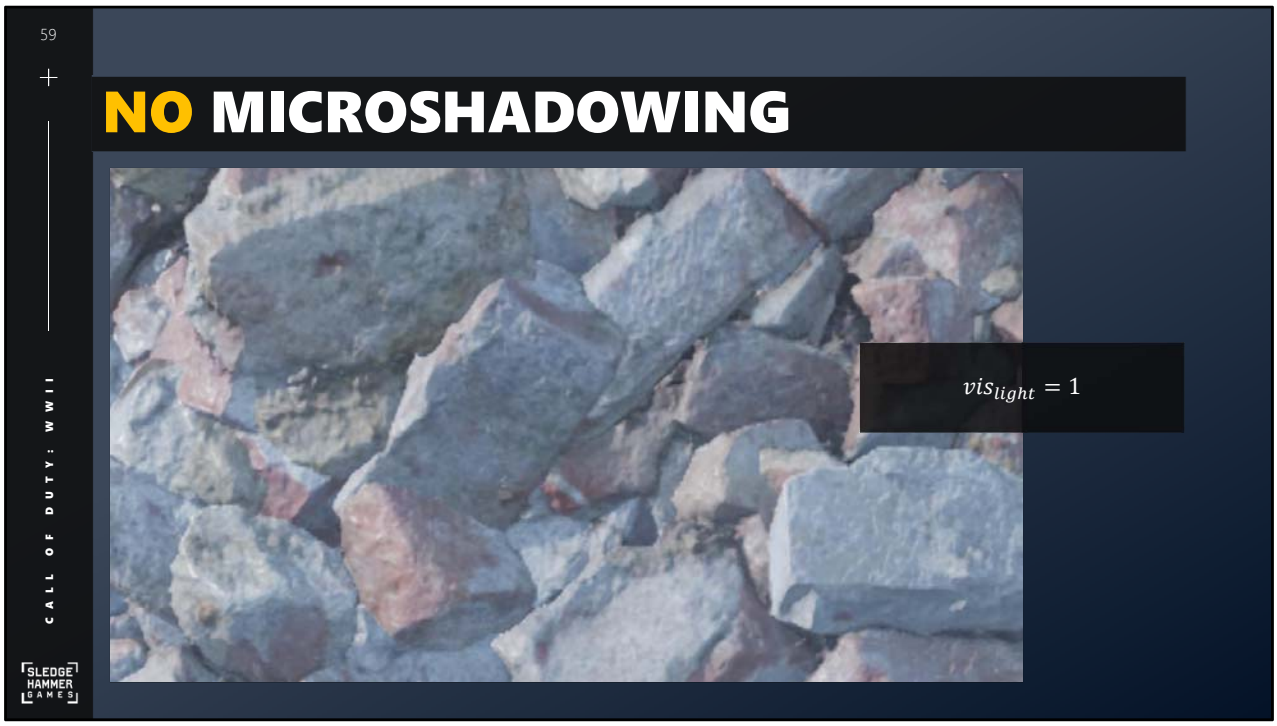
$$\frac{V}{1} = \frac{\pi r^2}{\pi}$$

$$V = r^2$$
$$r = \sin\theta$$
$$V = sin^2\theta$$
$$V = 1 - cos^2\theta$$
$$\cos\theta = \sqrt{1 - V}$$

CALL OF DUTY: WWII

SLEDGEHAMMER GAMES

Now, we're going to change from calling our value an Ambient Occlusion value to calling it Visibility, which is more appropriate since a value of 1 means fully visible.
The first step is to convert visibility to an equivalent cone angle.
We are going to reason about this via the Nusselt Analog.
The area of disk C divided by the area of the unit disk is equal to the cosine-weighted visibility.
From this, we can relate occlusion to a cone angle.

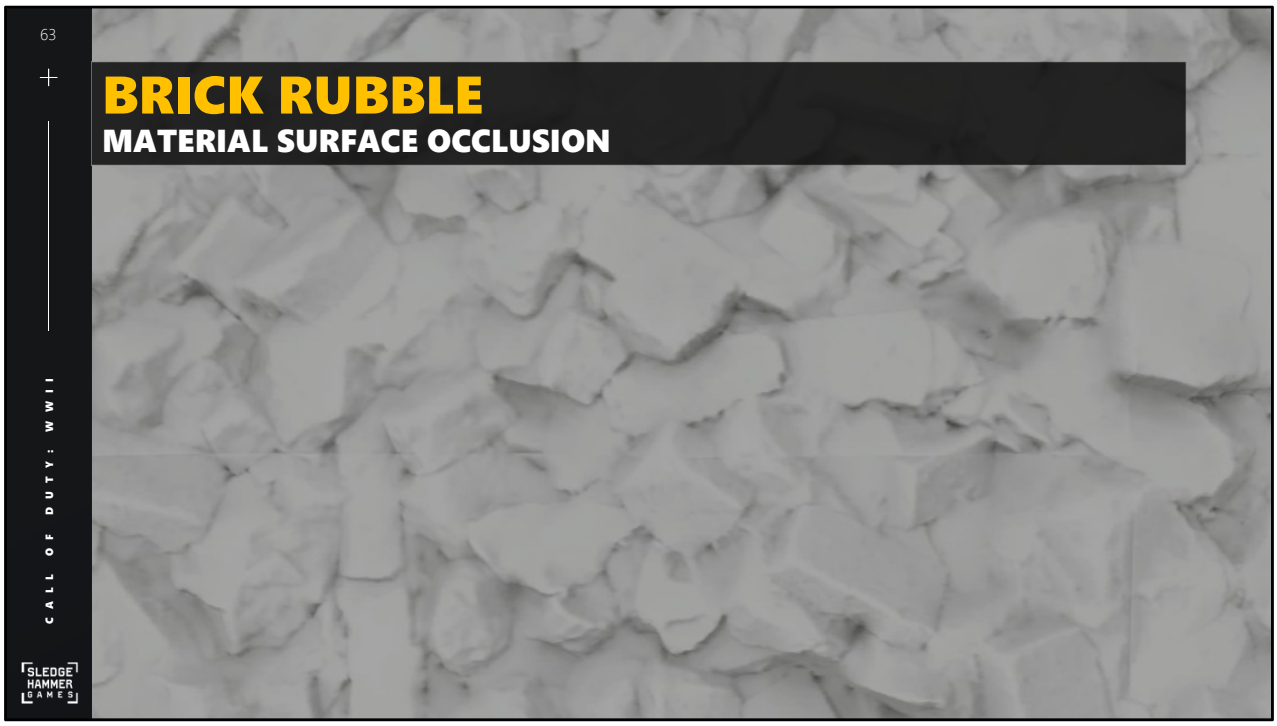


We can use the cone in many different ways to apply microshadows.
Here's a render with no microshadowing.

If we apply a threshold, where the light direction must lie within the equivalent cone, we get this hard splotchy effect.
The equivalent cone is a very rough approximation to actual visibility, so this isn't a surprise.

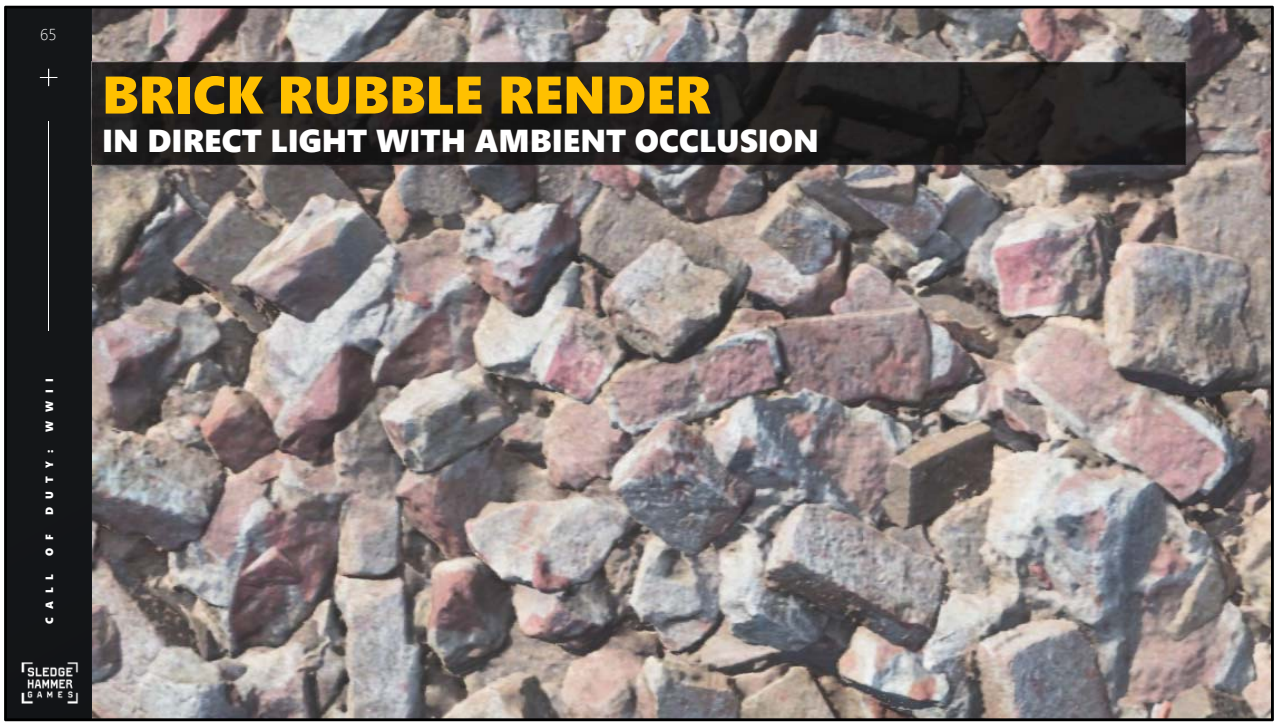
We can apply an ad-hoc adjustment instead.
Here, we smooth the falloff so it isn't a step function.
This looked a little too soft.

So we applied a sharper transition, which is what we shipped with in-game.



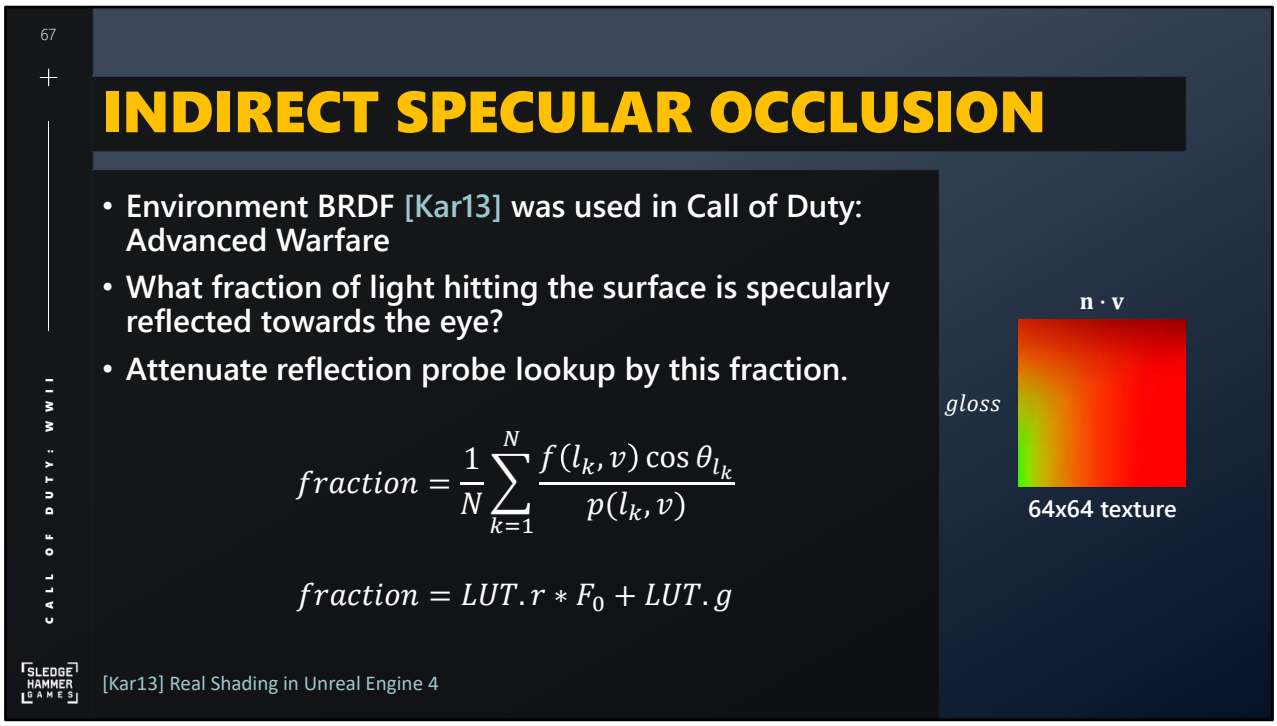
Here’s another example of microshadowing at work.
This is the generated Material Surface Occlusion texture.

Here is a render in direct light, with no ambient occlusion nor microshadowing applied.



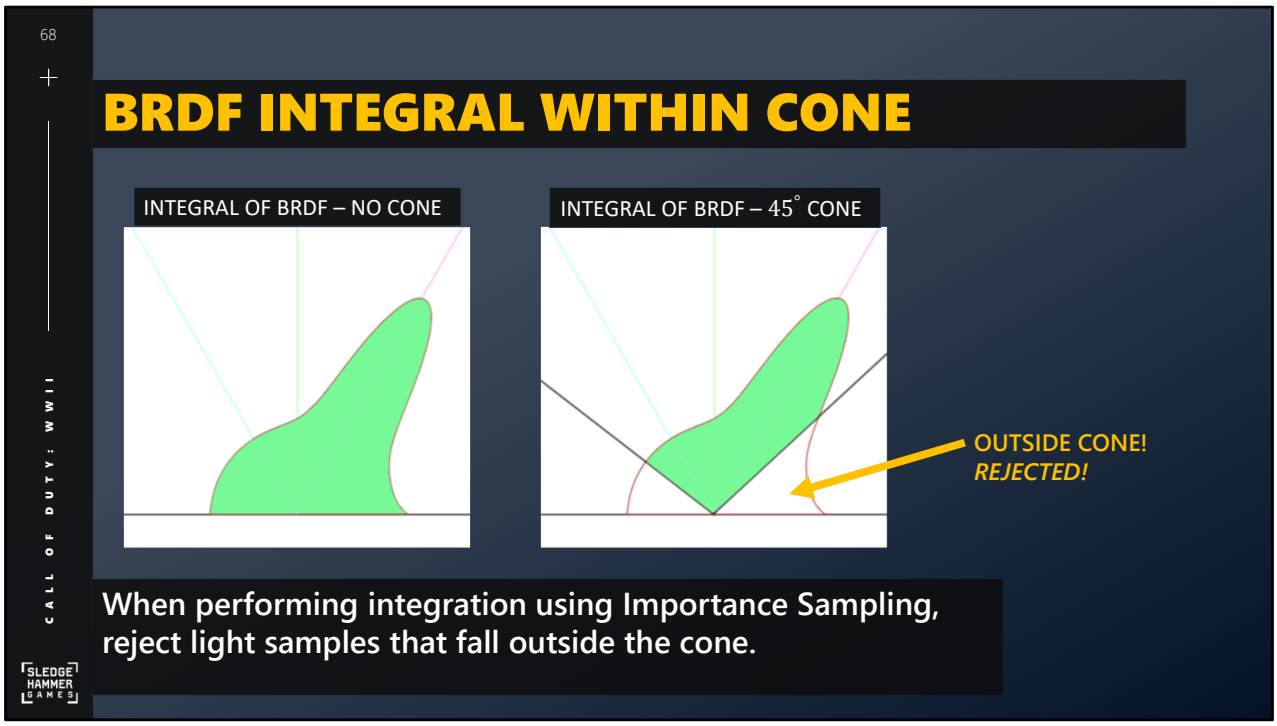
Ambient occlusion is pretty subtle when we are in direct lighting.

Now we apply microshadowing for a more pronounced effect.
This is a rendered quad and all the lighting variation comes from the texture maps.
[Toggle with last frame]

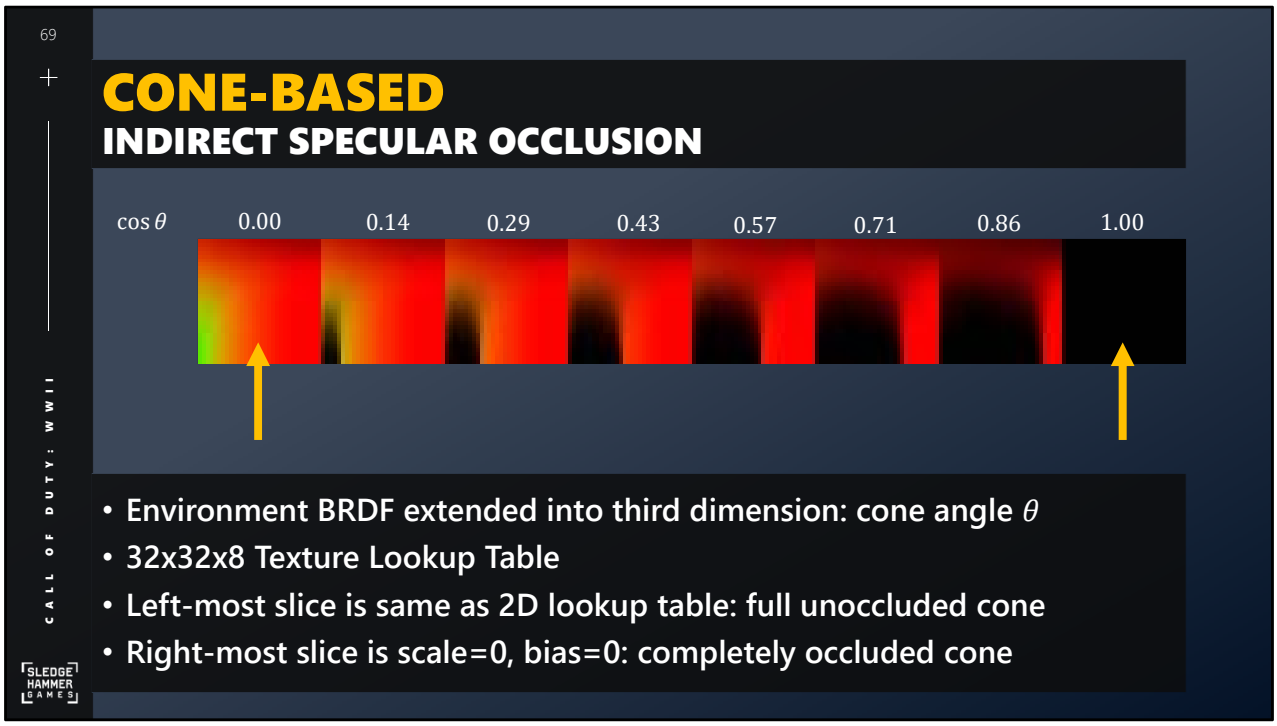


An Environment BRDF lookup table was used in Advanced Warfare.
This lookup table answers the question:
What fraction of light hitting the surface is specularly reflected towards the eye?
Not going to go into detail about this, please refer to Brian Karis' presentation from SIGGRAPH 2013.
But we generate the table by integrating our specular BRDF over the hemisphere, using importance sampling.
We can lookup this fraction using the lookup table which encodes a scale and bias to apply against F0.

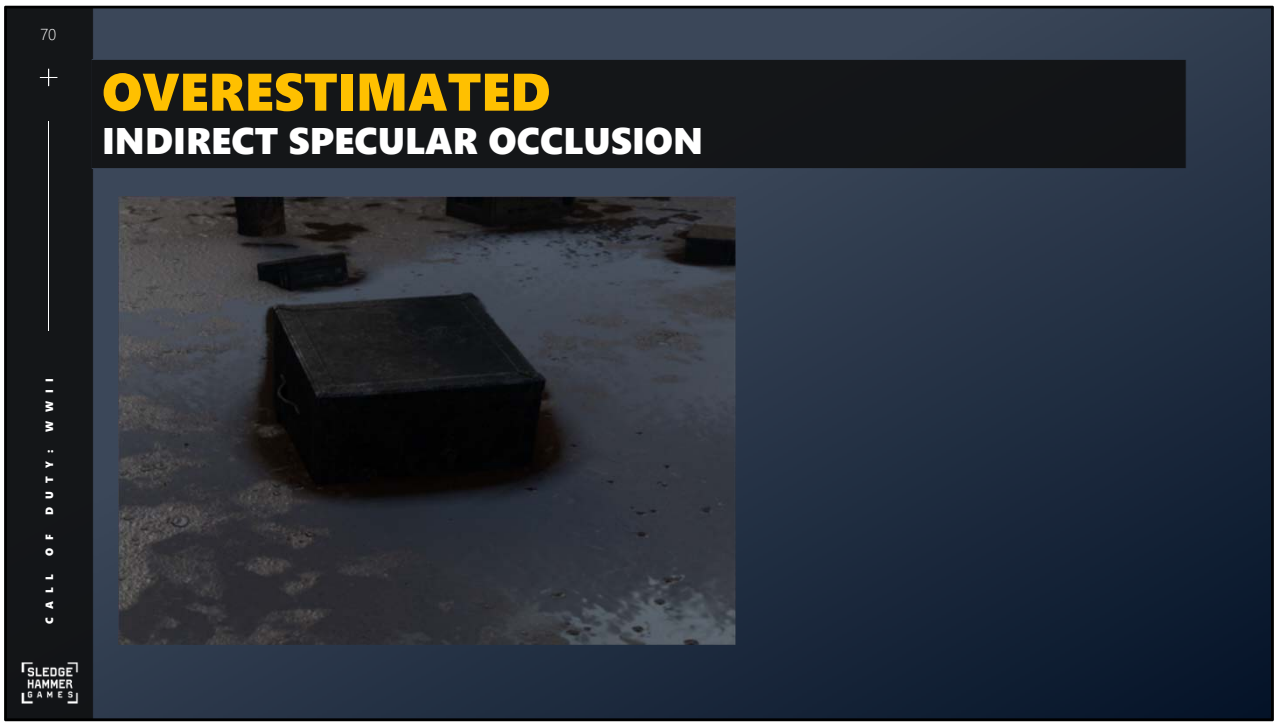
[Kar13] Real Shading in Unreal 4, by Brian Karis, Unreal Games
https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf



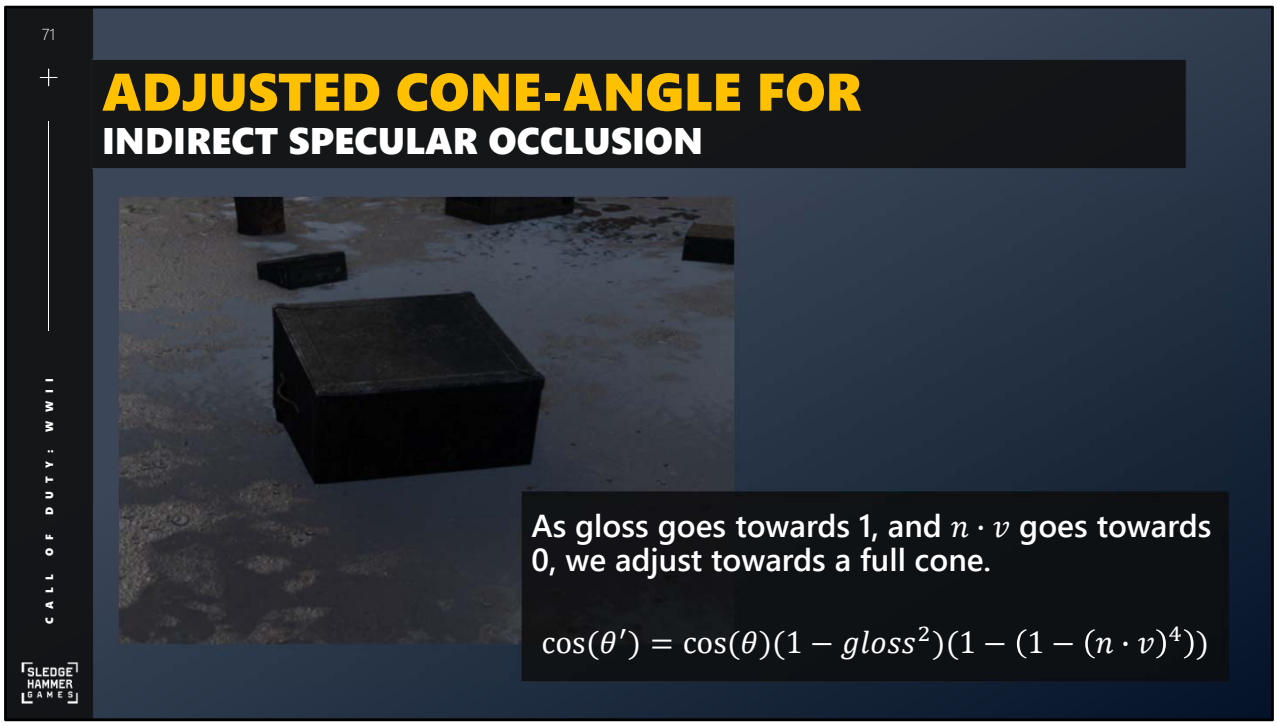
We have a cone-representation of occlusion, so we can apply this cone when integrating the BRDF.
As we integrate the BRDF, we reject light directions that lie outside the cone.



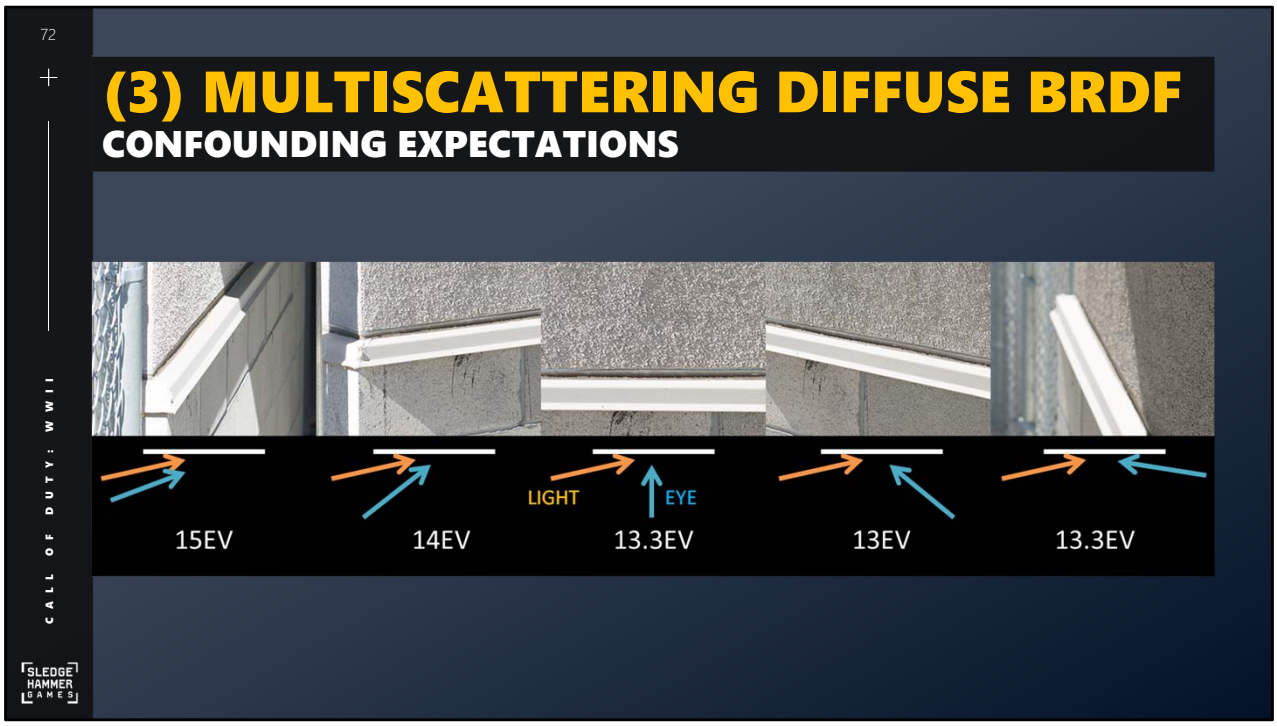
And then we modify our EnvBrdfLut to be 3D, where the 3rd dimension is cone angle.
We also change the resolution of the texture from 64x64 to 32x32x8.
Notice the left-most slice [click]
is exactly the same as the 2D lookup table we saw before.
And the right-most slice is black, [click]
which represents a scale and bias of zero.
This represents a completely occluded cone.



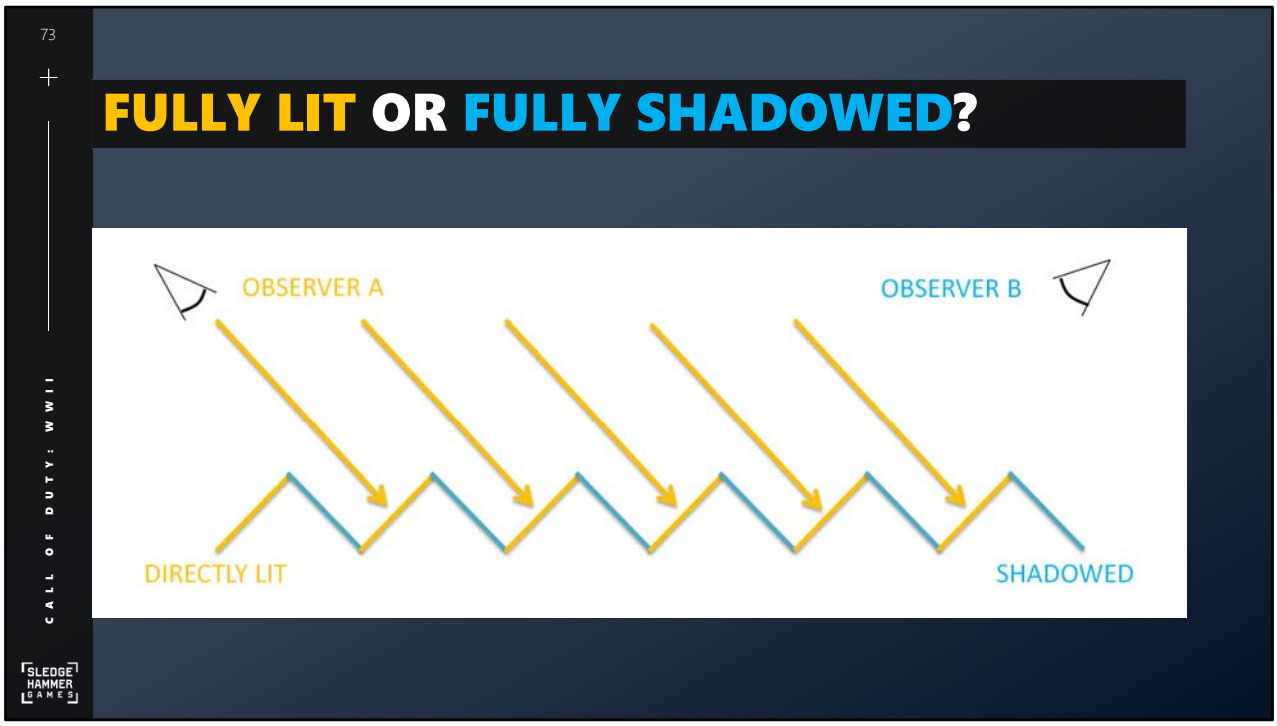
The problem with using cone-based occlusion is that the cone may be a poor approximation for actual visibility.
And this approximation gets worse when the specular lobe is sharper.
There is much more chance for the sharp lobe to fall outside the cone, especially at grazing angles.
This leads to an overestimation of specular occlusion and you get halos around objects that are surrounded by glossy materials, like water.



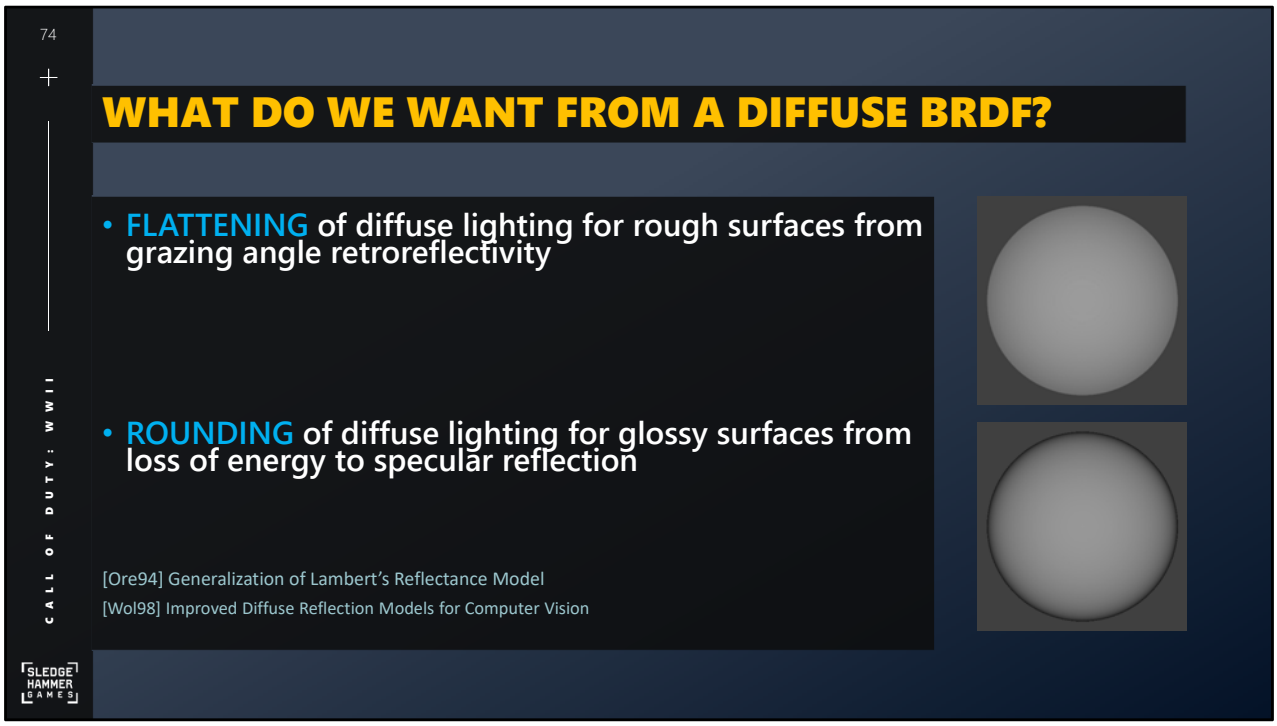
We use an ad-hoc adjustment to our cone-angle and which open up the cone as gloss goes towards 1 and as view angle becomes more grazing.
This generally works pretty well, but
Here we have an underestimation of specular occlusion because of this adjustment.
It's basically turned off.
Still the adjustment was necessary because the player's guns were losing too much specular reflection, since most of the surfaces of the gun were viewed at grazing angles.
In the future, we'd like to reason about this probabilistically, or add more directional information into the occlusion value to resolve these issues.



These pictures are from the early days of Advanced Warfare.
We noticed that when taking area spot meter measurements of bumpy surfaces, our measurements confounded our expectations.
Why is the left most image brighter than the right most image?
This seems to respond differently to what we'd expect.
Normally specular response would make rightmost image brightest.
But that's not what is happening here.



Our diffuse model did not take into account the bumpiness of the surface.
Bumpiness can cause a diffuse surface to look different to two different observers.
We already have some of this accounted for in our specular microfacet models, but not in our diffuse.
In this extreme example, observer A sees a fully lit surface, while observer B sees a shadowed surface.



What do we want from a Diffuse BRDF?

Many papers show a flattening of diffuse lighting for rough surfaces.
Think of the moon as an example of this phenomenon.
And BRDFs such as Oren-Nayar attempt to model this.

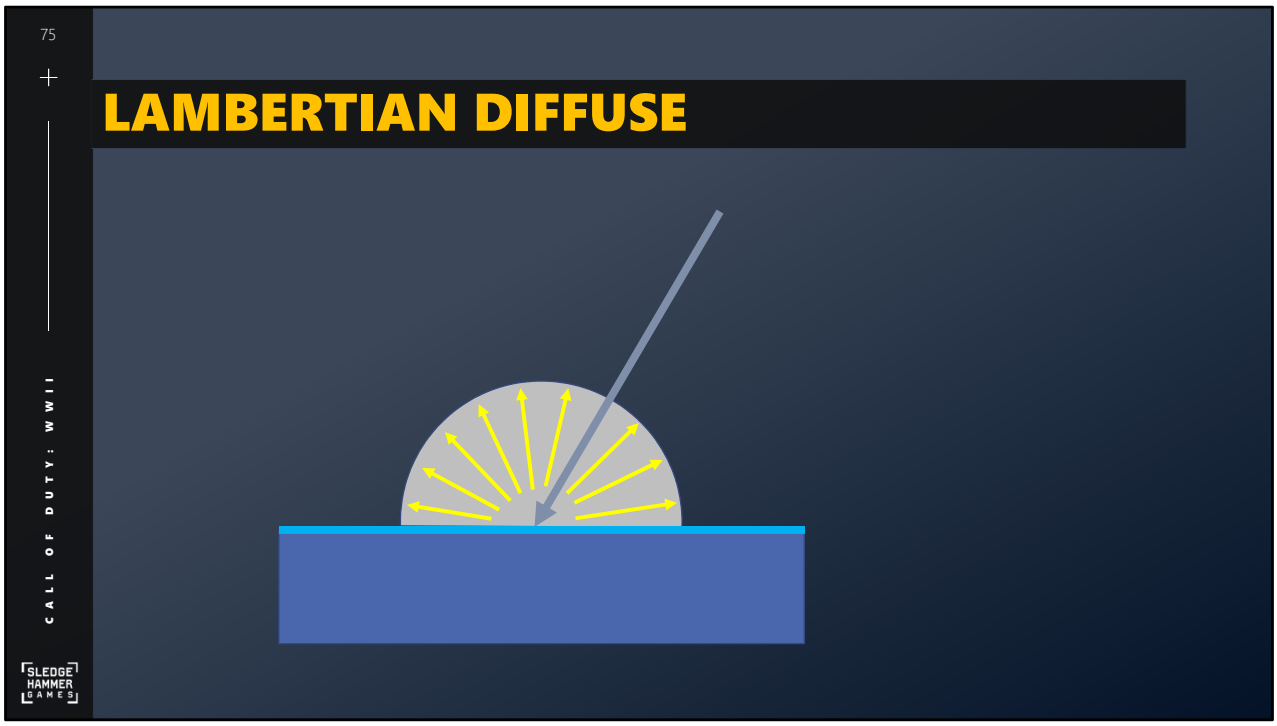
Then, there are observations in a couple other papers that describe a rounding of lighting for smooth surfaces.
This is opposite of the flattening effect.
With what we are calling a "rounding" of lighting, the falloff of light is more extreme along the silhouette of the sphere below.
This can be explained by energy loss to specular.

In images to the right, light and view are the same.

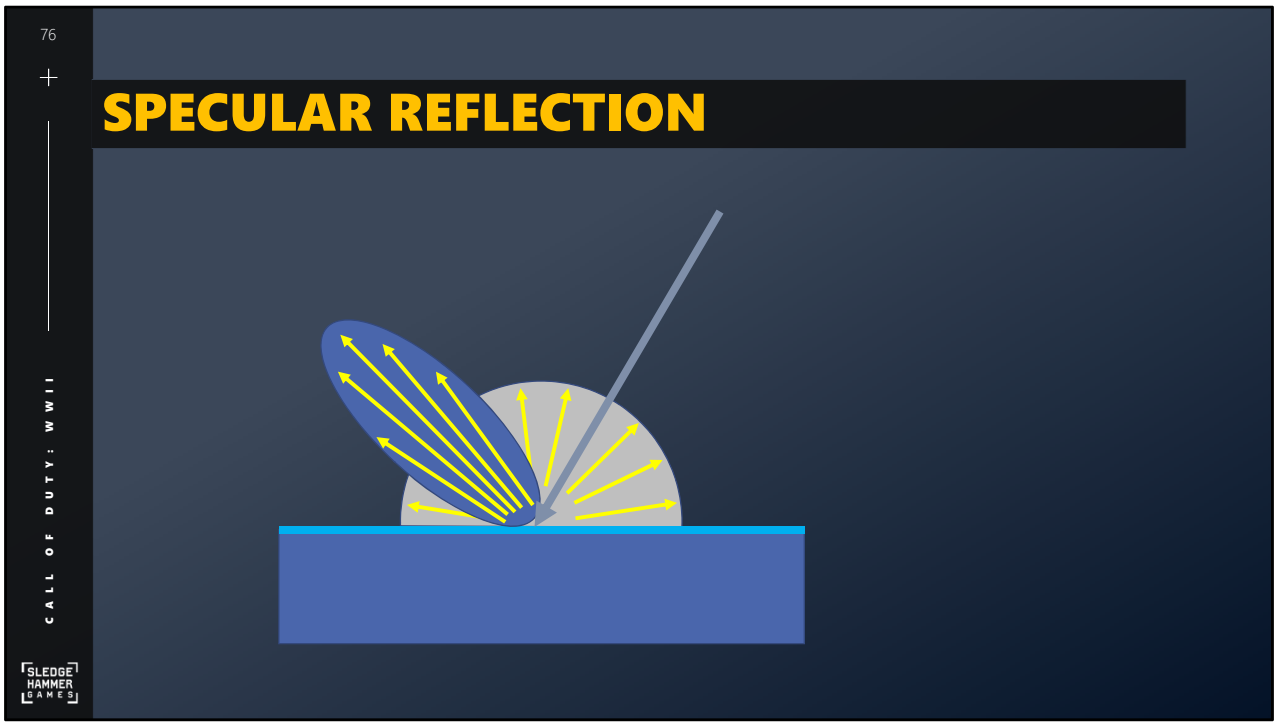
Most solutions focus on the flattening of lighting (Oren-Nayar), but we'll see later that the rounding of lighting is important as well and provides better material differentiation.

[Ore94] Generalization of Lambert's Reflectance Model
http://www1.cs.columbia.edu/CAVE/publications/pdfs/Oren_SIGGRAPH94.pdf

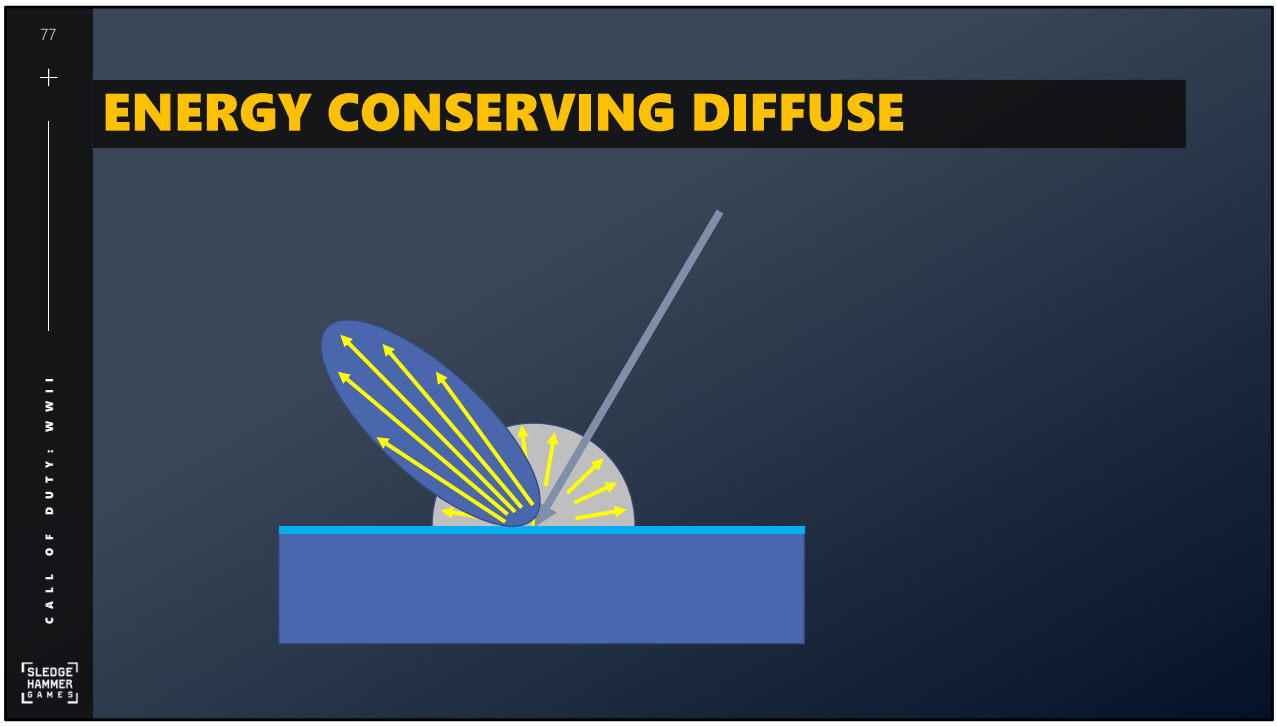
[Wol98] Improved Diffuse Reflection Models for Computer Vision
https://link.springer.com/article/10.1023%2FA%3A1008017513536



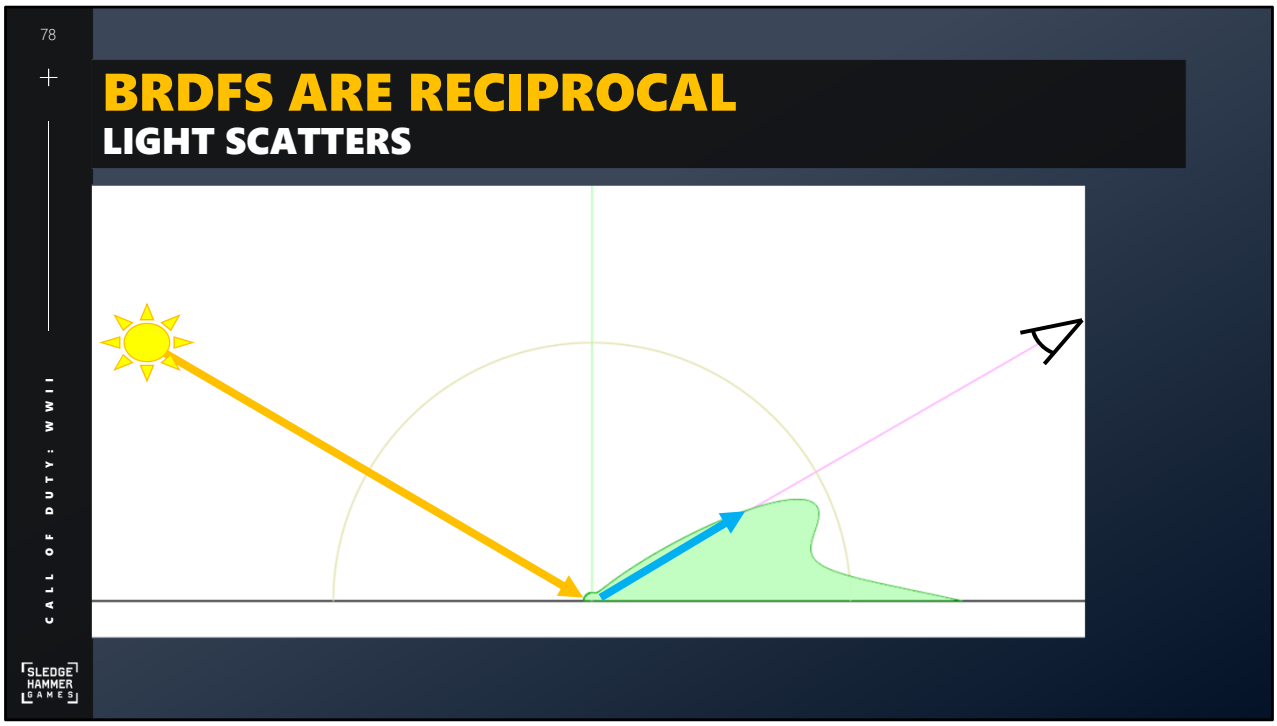
Lambertian diffuse has been the standard for diffuse response in real-time graphics, at least until recently.
If a surface is only lit with Lambertian diffuse, that surface will appear the same brightness from any angle.



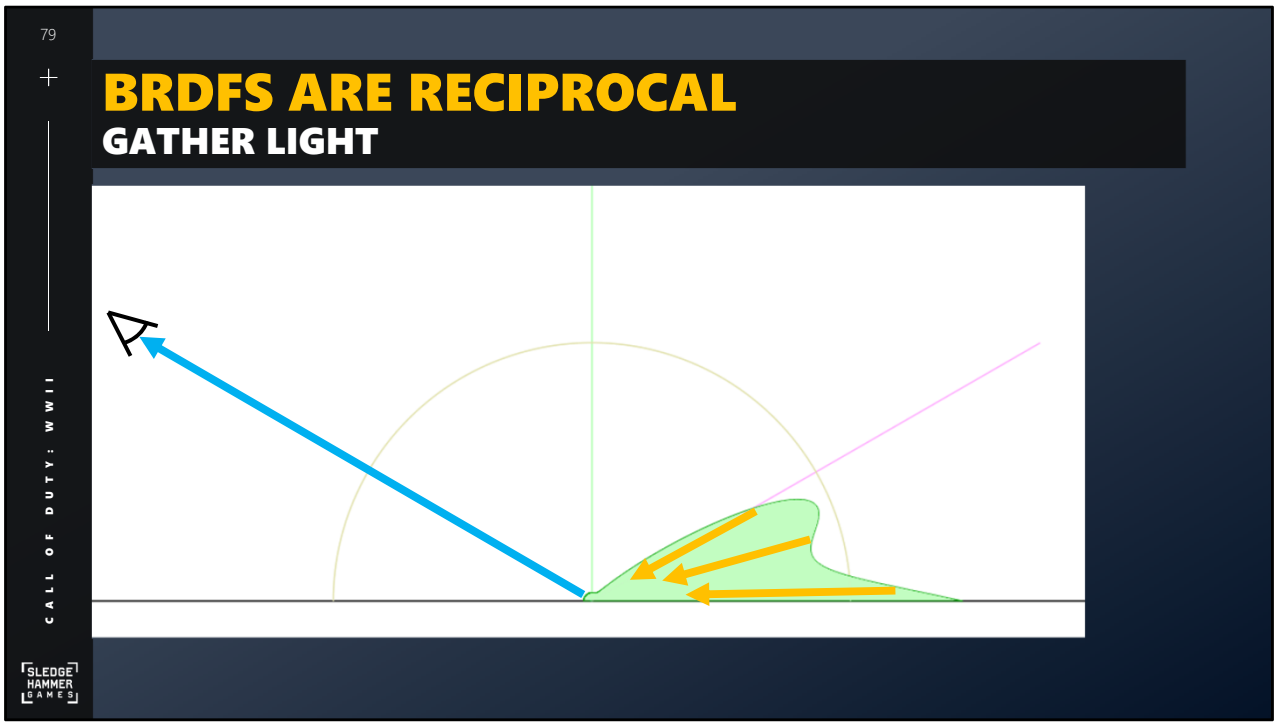
Not too long ago, we would model specular reflection and just add this specular contribution on top of the diffuse.
But really, in order to be energy conserving, the energy reflected specularly would not be available to reflect diffusely.



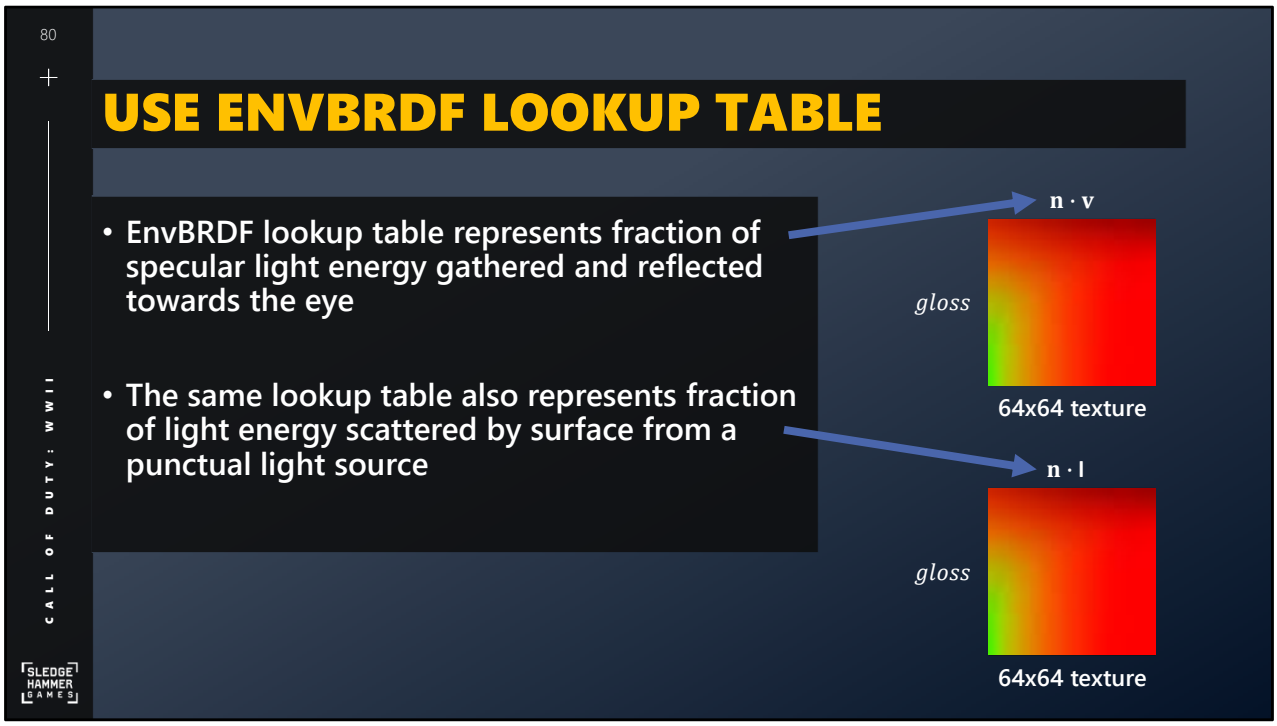
Diffuse reflectance is diminished by the amount that is specularly reflected.
So the question is: how much is specularly reflected?



An important concept to understand is that BRDFs are reciprocal.
By that, we mean light will travel along the same path in both directions.
That means we can swap the light vector and the view vector for a BRDF function and it should still be equivalent.
You can try this with the microfacet equation, swap the light vector (L) and the view vector (V). Notice, that the half-angle vector (H) is unaffected by swapping them.
When we think about punctual light sources, we typically imagine light traveling with a single path towards the point being shaded
And then we evaluate how much reflectance is bounced towards the eye.
At least that's the way I used to think of punctual lights.
Notice there is reflected light scattering in many directions, defined by the green lobe, that never makes it to the eye.



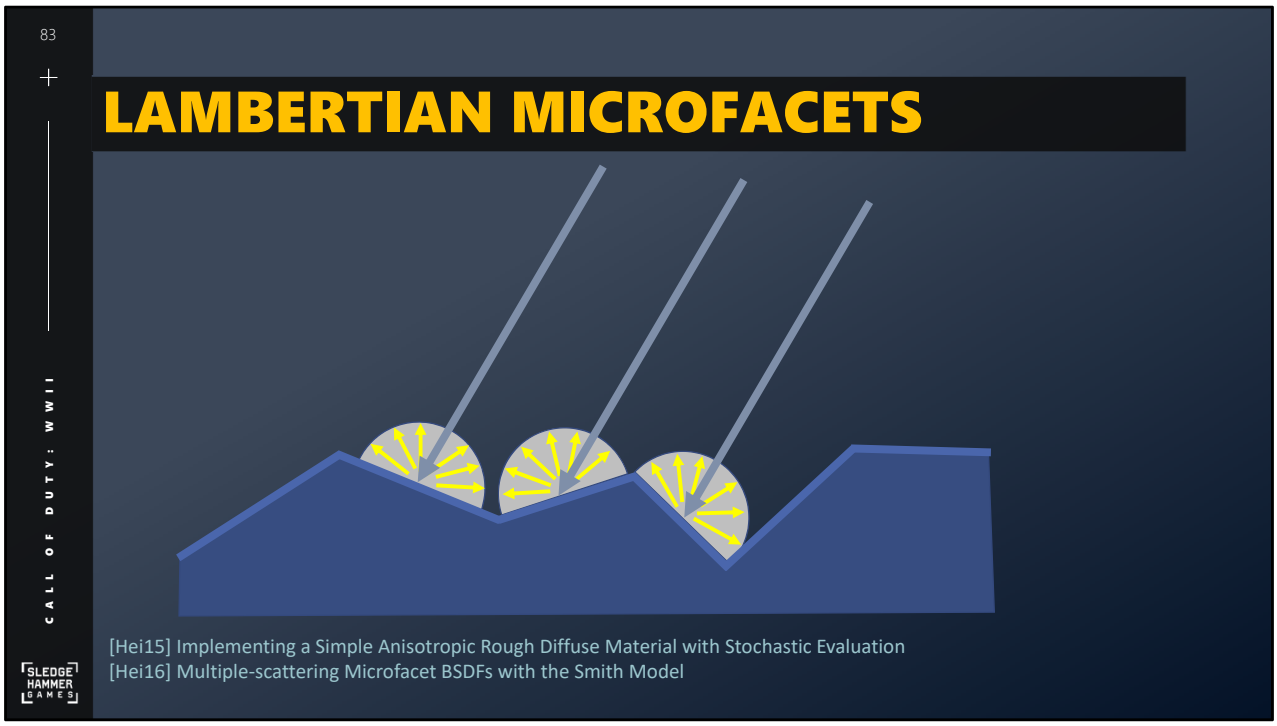
But we can just as easily imagine it the other way.
Notice the BRDF lobe remains unchanged but our eye takes the place of the light source.
What light is reflected towards is the eye, is determined by the light *gathered* in the directions defined by the BRDF lobe.
This is typically how we imagine gathering indirect specular reflection, through reflection probes or environment maps.



Now, we are going to bring up the EnvBRDF lookup table again.
It was used previously to calculate what fraction of specular light energy should be gathered and reflected towards the eye.
But it can just as easily be used to calculate the light energy scattered by a surface from a punctual light source.
This is the energy reflected specularly, that is unable to participate in diffuse reflectance.

Here is a glossy material using Lambertian diffuse.

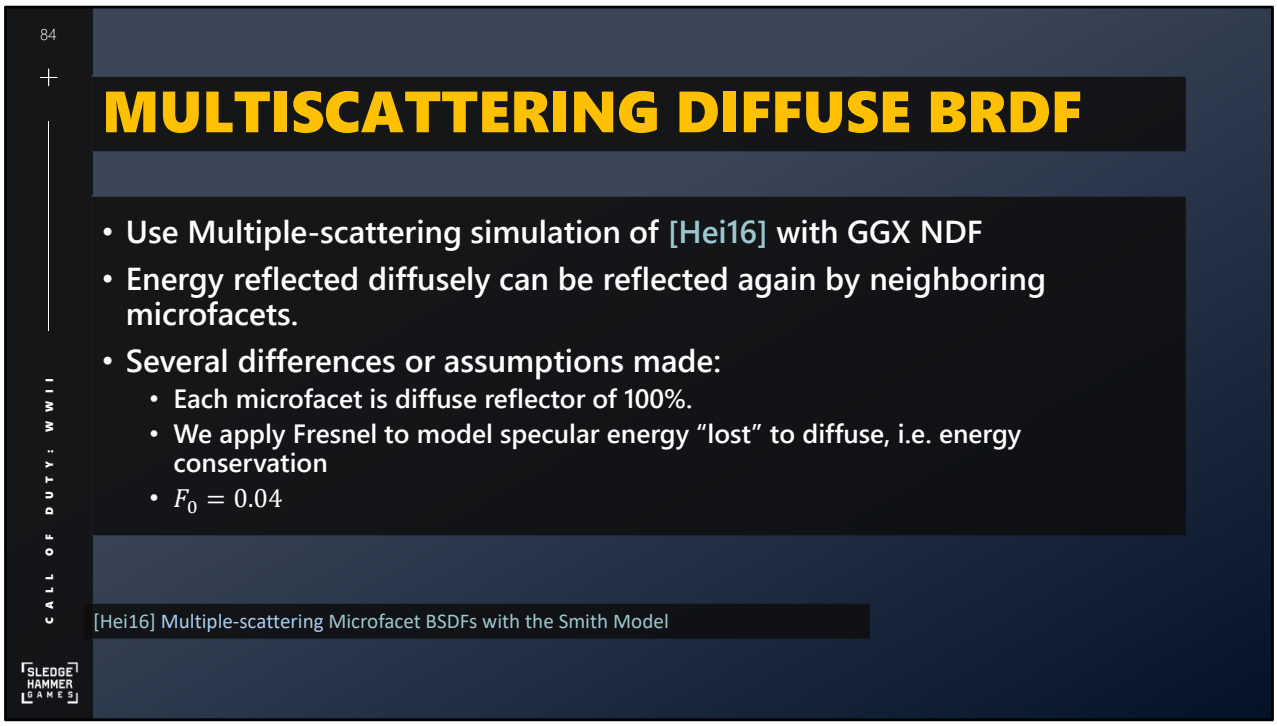
Here we are reducing the amount of light energy available for diffuse reflection using the EnvBrdf lookup table.
[Toggle with last frame]
It's important to note, by just diminishing the energy available for Diffuse reflectance, we are creating a BRDF that potentially violates reciprocity.
Part of future work is to resolve this by introducing multiscattering specular and tying it all together.
We didn't ship with this technique though because we have a built-in energy conservation in the next method that we are going to discuss.



Next, we looked into to some recent work by Heitz and his coauthors.
The first paper talks about modeling a diffuse BRDF using Lambertian microfacets.
The second paper gives a general solution to multiple scattering with microfacets.
Lambertian microfacets sounded intriguing to us since it keeps a consistent physical representation between our diffuse and specular models.
There is an argument to be made against modeling diffuse reflectance in this way: depending upon scale, diffusion distance means the receiving microfacet may not be the re-emitting microfacet of a photon.
But as we'll see next, nice properties fall out of this model.

[Hei15] Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation
https://drive.google.com/file/d/0BzvWIdpUpRx_M3ZmakxHYXZWaUk/view

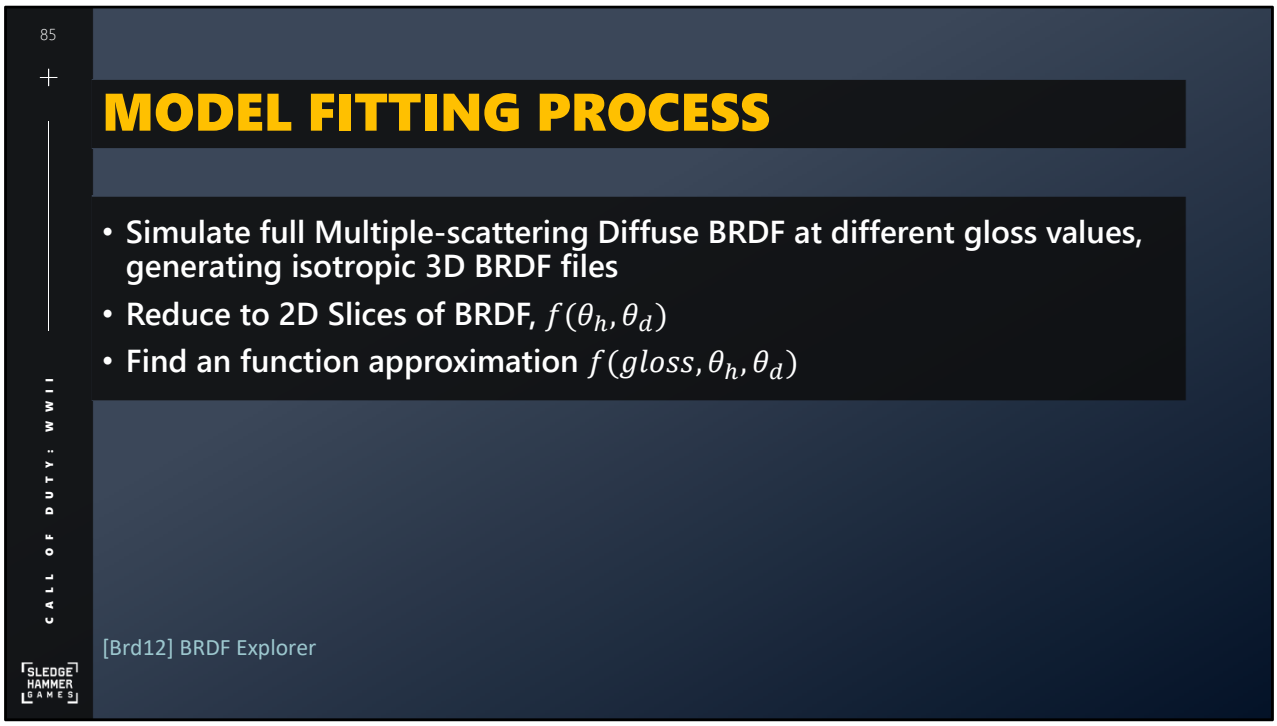
[Hei16] Multiple-scattering Microfacet BSDFs with the Smith Model
https://eheitzresearch.wordpress.com/240-2/



Used Heitz Multiscattering simulator with GGX NDF.
Using a multiscattering model means energy reflected diffusely can be reflected again by neighboring microfacets.
We made some additions and some assumptions to his model:
1) We treat each microfacet as a diffuse reflector of 100%.
2) We apply Fresnel to determine energy reflected diffusely.
3) We use implicit F0 = 0.04 to determine energy reflected specularly and unable to participate in diffuse reflectance.

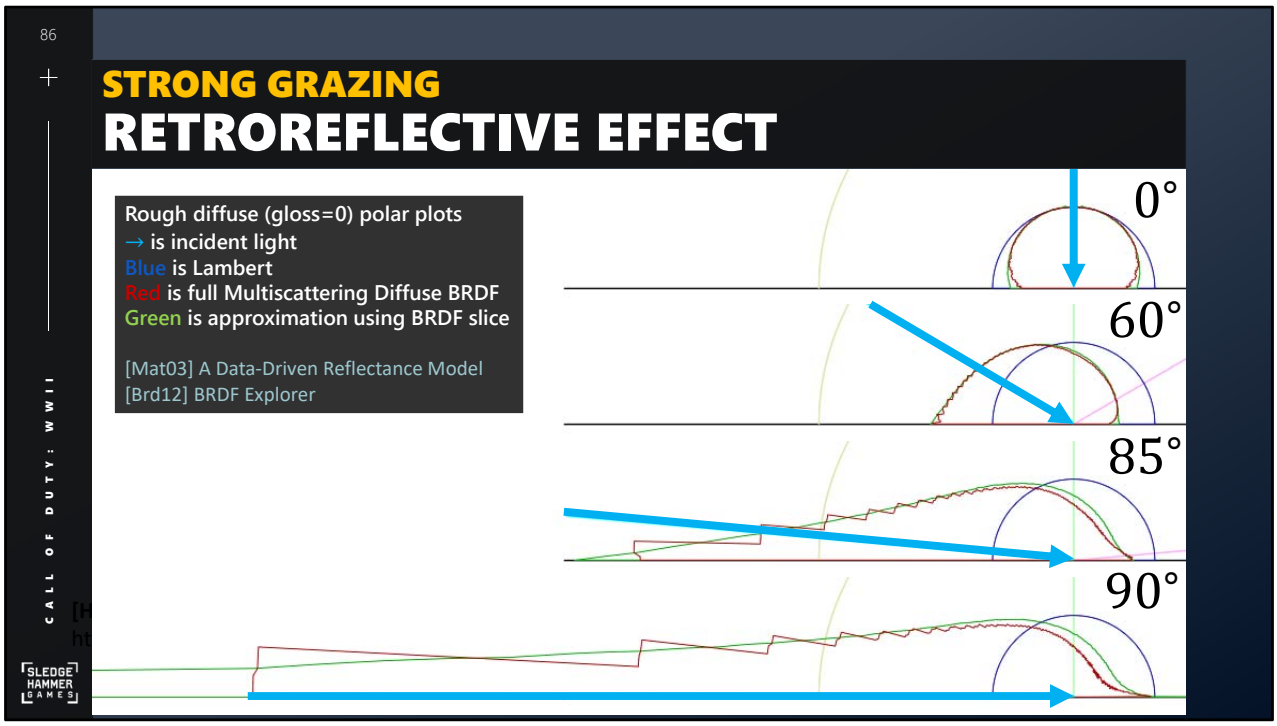
Interestingly, the properties we want fall out of this model automatically:
Flattening for rough surfaces from strong grazing retroreflective response,
Rounding for smooth surfaces from energy loss to specular.

[Hei16] Multiple-scattering Microfacet BSDFs with the Smith Model
https://eheitzresearch.wordpress.com/240-2/



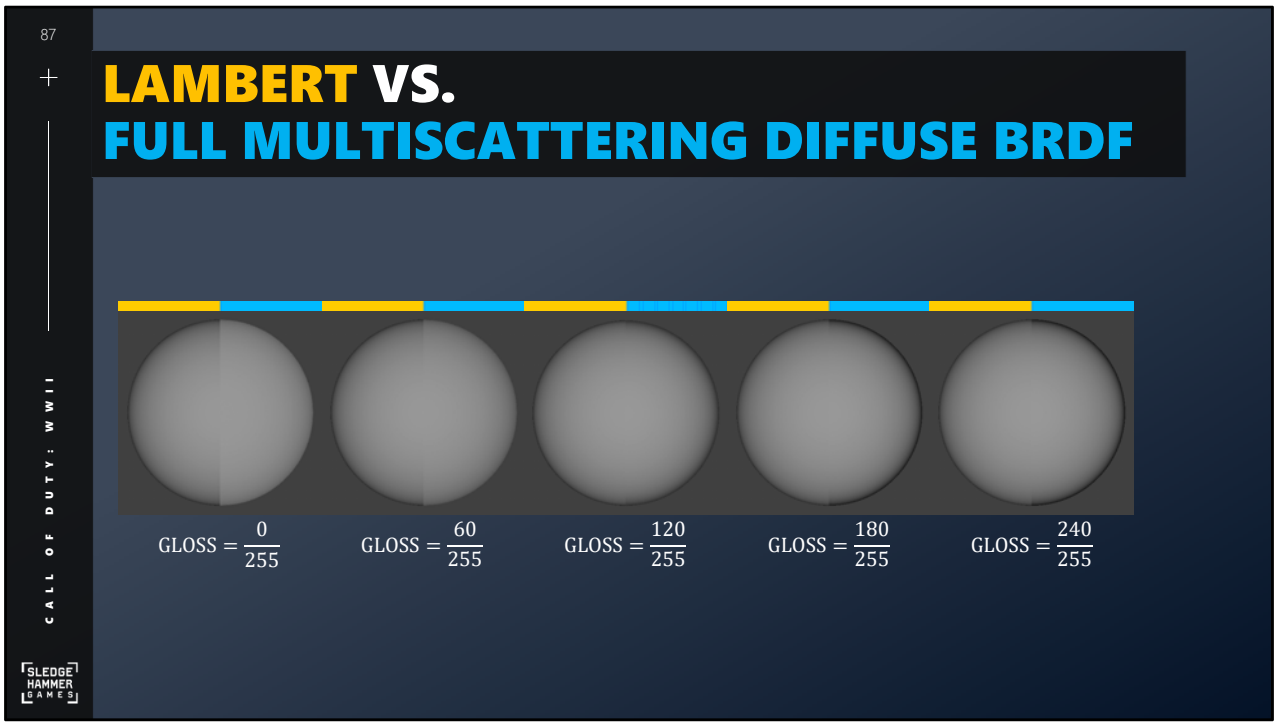
Here's a general outline of our fitting process.
First we simulate the Multiscattering Diffuse BRDF at many different gloss values, generating isotropic 3D BRDF files that can be loaded in BRDFExplorer.
Then we reduce to a 2D slice of the BRDF.
Finally, we find a function approximation that includes gloss as a parameter.

[Brd12] BRDF Explorer
https://www.disneyanimation.com/technology/brdf.html

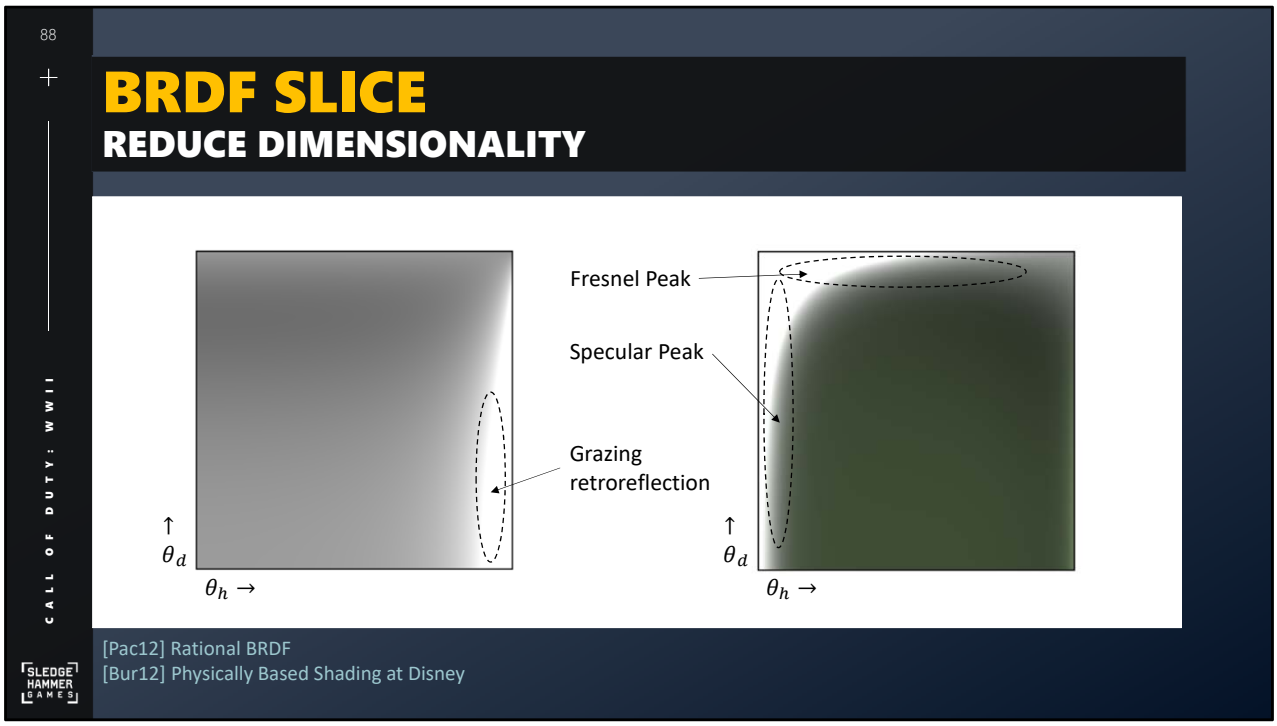


We simulated many bounces of photons using Heitz' Multiple-Scattering Microfacet model with our modifications to generate a table of diffuse response over the hemisphere.
This was processor intensive but easily parallelizable, so we distributed this task over machines in our network.
We generated files in MERL database format so that BRDFExplorer could open and inspect them.
These are polar plots from BRDFExplorer, showing the strong grazing retroreflective effect from rough surfaces using our Diffuse model.

[Brd12] BRDF Explorer
https://www.disneyanimation.com/technology/brdf.html



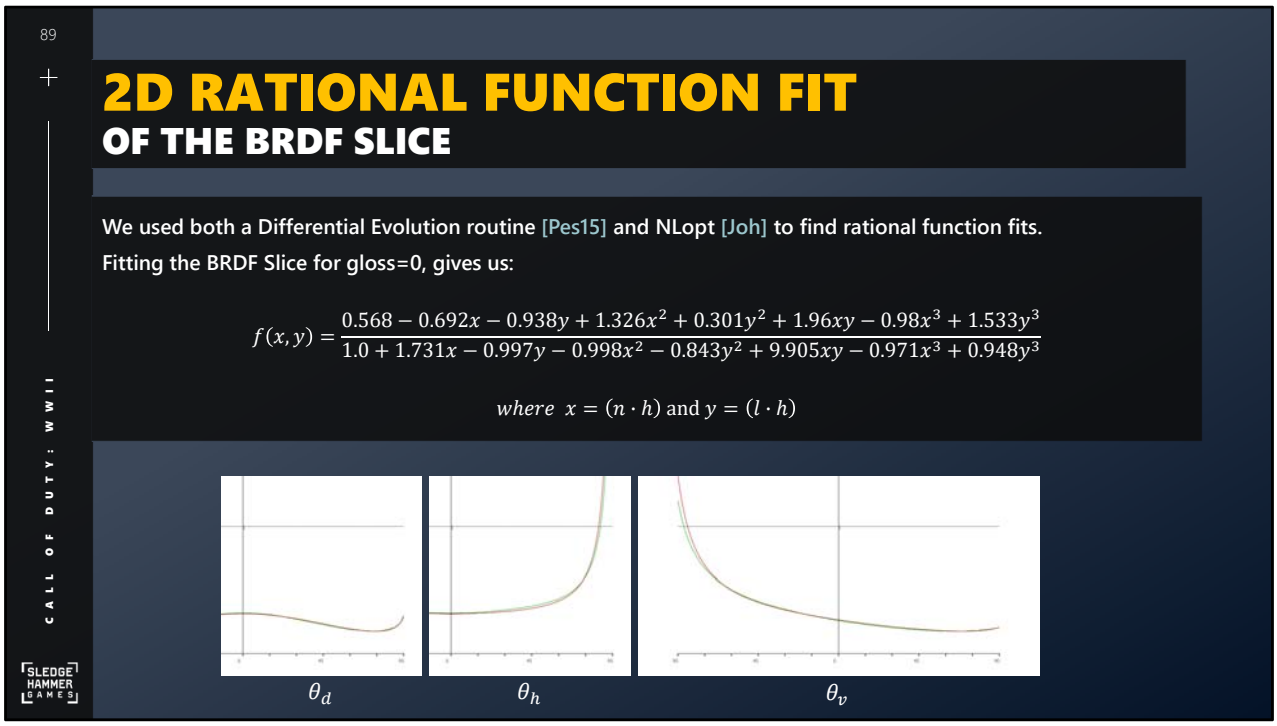
Here is a dead-on view of a lit sphere.
The left half of the spheres are lit using lambert,
While the right half are lit using the fully simulated Multiscattering Diffuse BRDF.
Notice the flat lighting of Multiscattering Diffuse in the left-most sphere compared to lambert.
In the right-most sphere, Multiscattering Diffuse results in a darkening in the silhouette.



The full isotropic BRDF has 3-dimensions.
There's a method to reduce the dimensionality of the problem.
Take a 2D slice out of the 3D volume.
It turns out a 2D slice is usually pretty good at representing the full BRDF.
Another nice thing about 2D slices, it lends itself to visual interpretation.
Here are two examples of BRDF slices.
The left one represents a rough material using our Multiscattering Diffuse model.
The right one is a standard green glossy material.
What we do now is we generate BRDF slices for different gloss values.
The first thing we tried was fitting these slices using 2D Rational Functions, which is describe next.

[Pac12] Rational BRDF
https://hal.inria.fr/hal-00678885

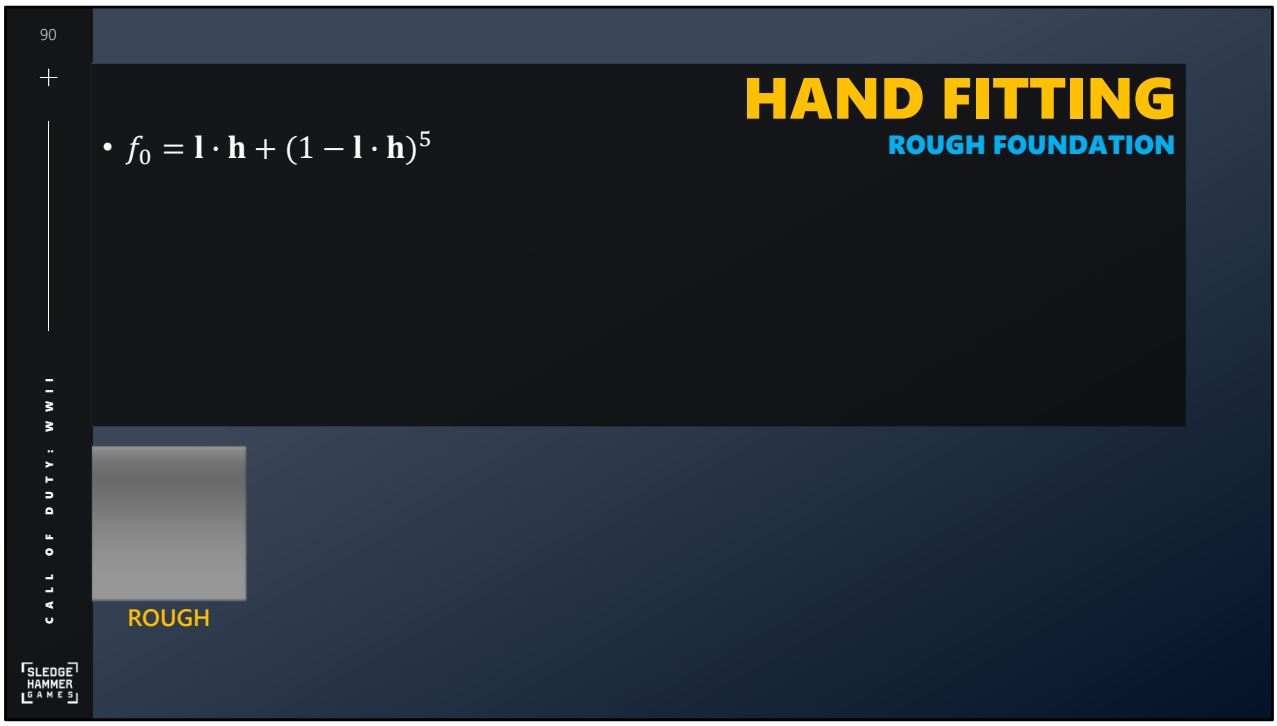
[Bur12] Physically Based Shading at Disney
http://blog.selfshadow.com/publications/s2012-shading-course/



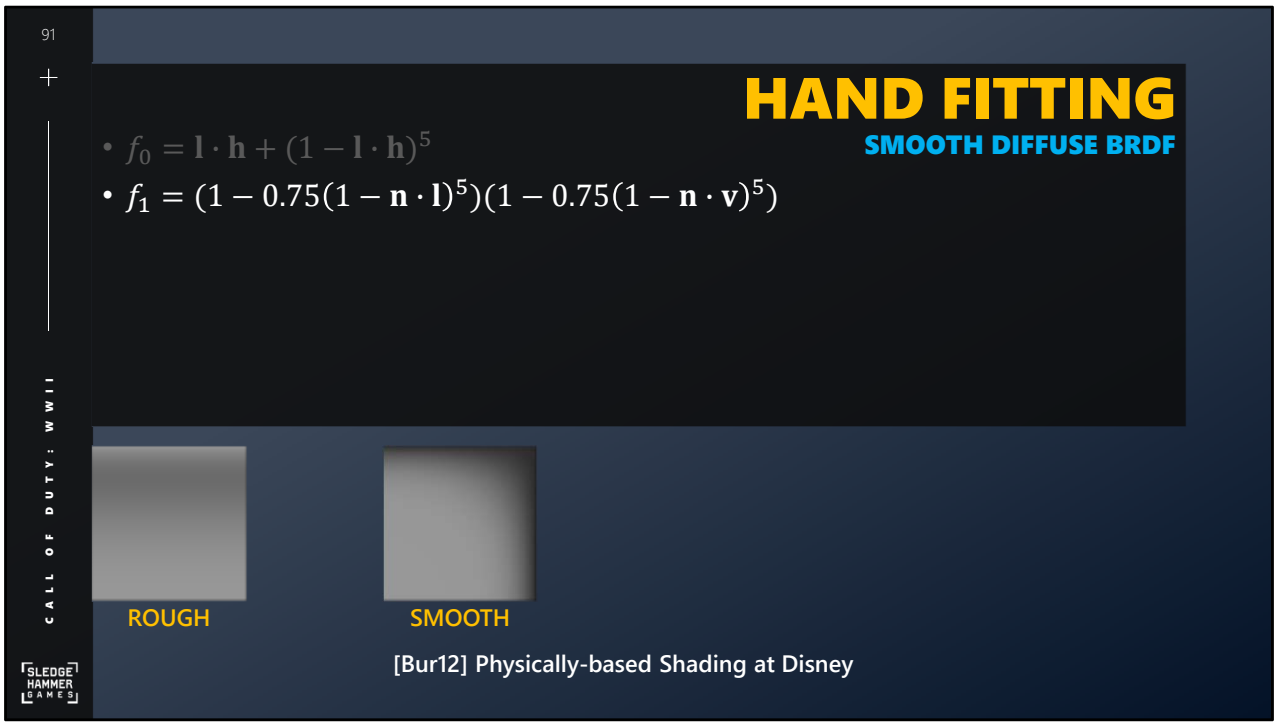
We found a good 15-coefficient rational function fit for the BRDF Slice when gloss is 0.
Ultimately, we need to be able to render the entire gloss range from 0 to 1.
We tried extending to a 3D rational function by adding gloss as another parameter,
But this required many more coefficients.

[Pes15] Approximate Models for Physically Based Rendering
http://blog.selfshadow.com/publications/s2015-shading-course/

[Joh] NLopt
https://nlopt.readthedocs.io/en/latest/

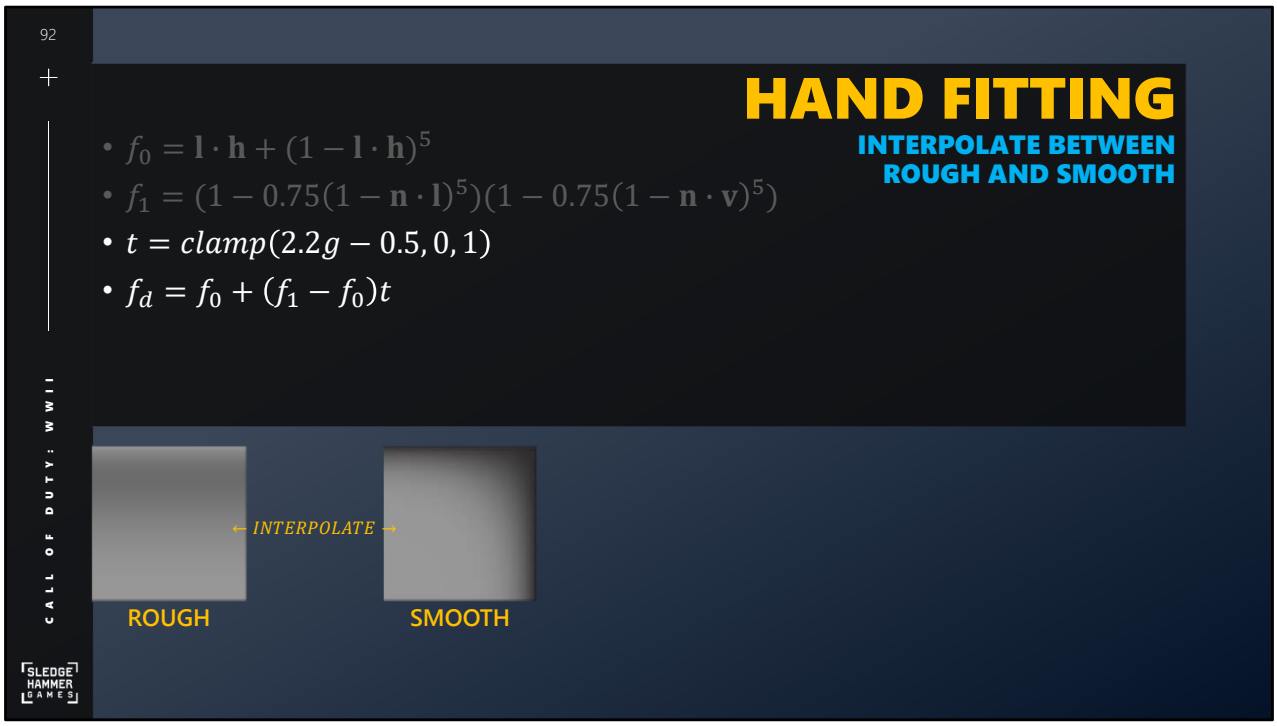


At this point, we decided to try to hand-fit the BRDF.
With hand-fitting we could ensure salient characteristics, like the grazing retroreflective effect that we see, are preserved.
We did this using Disney's BRDFExplorer.
As mentioned before, we modified BRDFExplorer to read our generated BRDF files, both the full 3D isotropic BRDF and the 2D slice.
By analyzing the BRDF using the different graphs provided in BRDFExplorer, we broke the BRDF down into a manageable number of parts, that could be combined to form the full result.
The first part is what we call the Rough Foundation.
At the bottom, there is a 2D BRDF Slice representing this Rough Foundation.

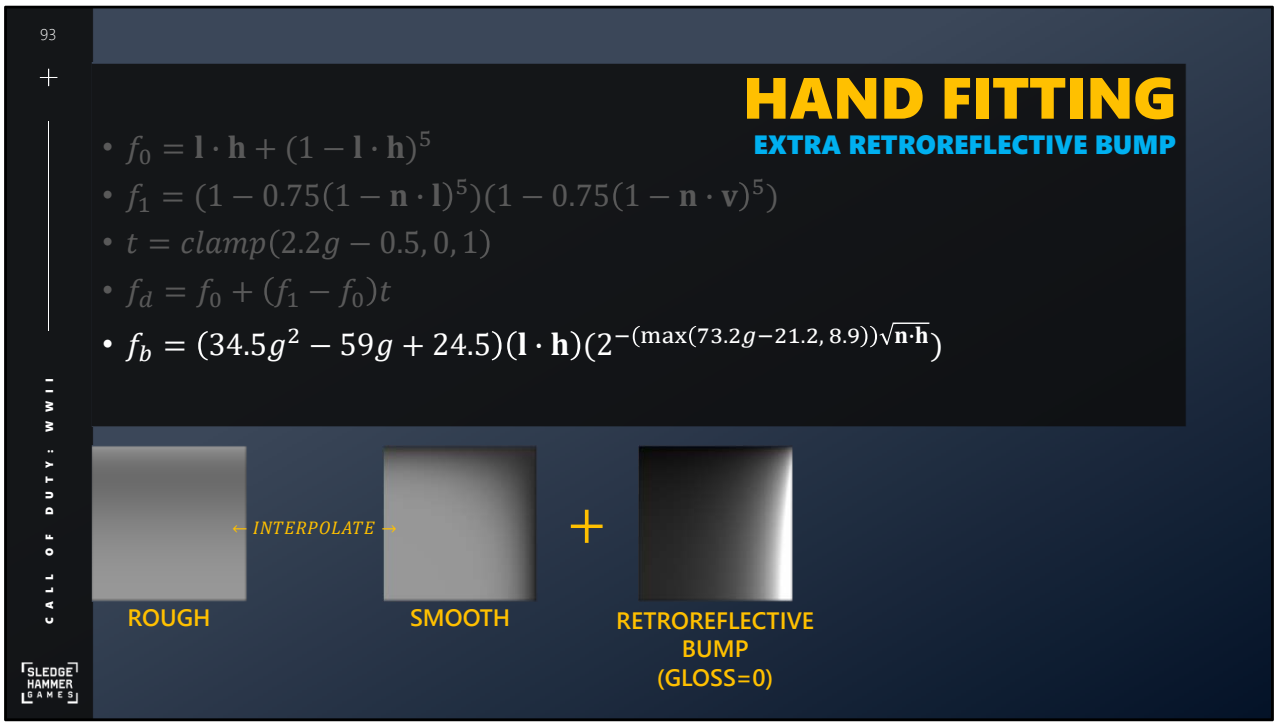


The second part is the Smooth Diffuse BRDF.
We found that Disney's Diffuse model from SIGGRAPH 2012 had a very good approximation to what we saw with Multiscattering Diffuse when gloss = 1.

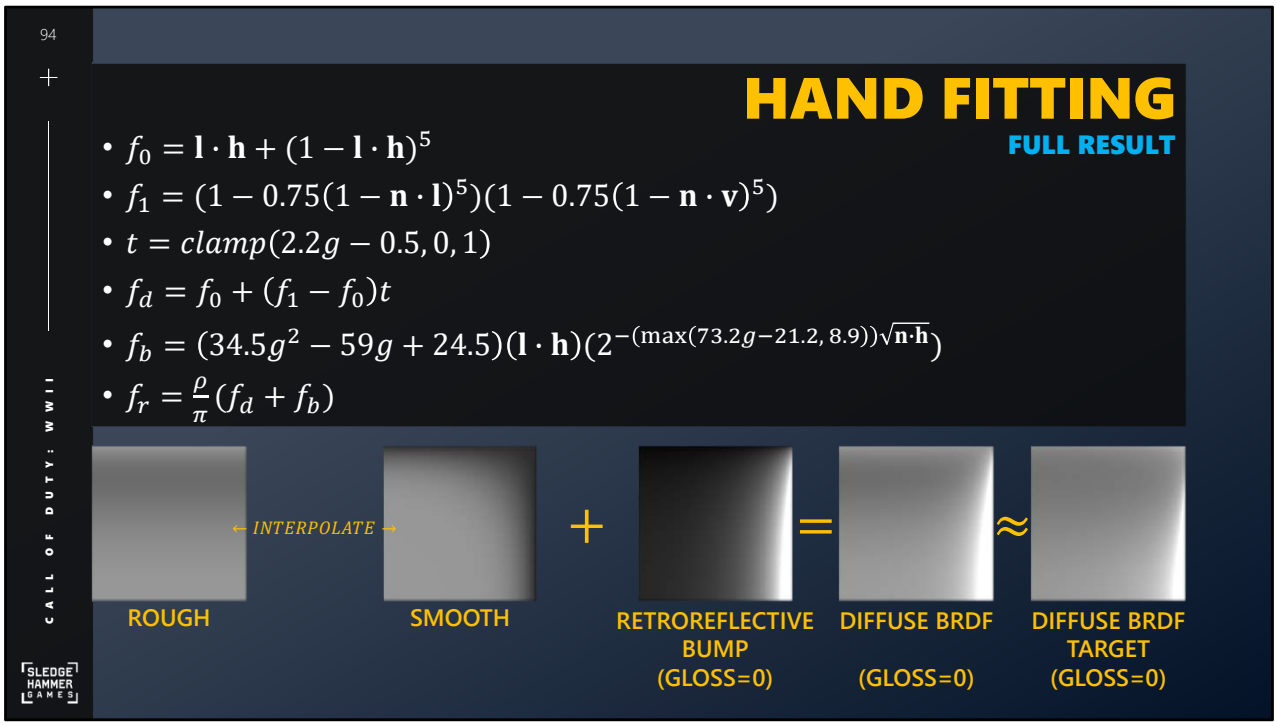
[Bur12] Physically-based Shading at Disney
http://blog.selfshadow.com/publications/s2012-shading-course/



Then we interpolate between our Rough and Smooth models based on gloss.

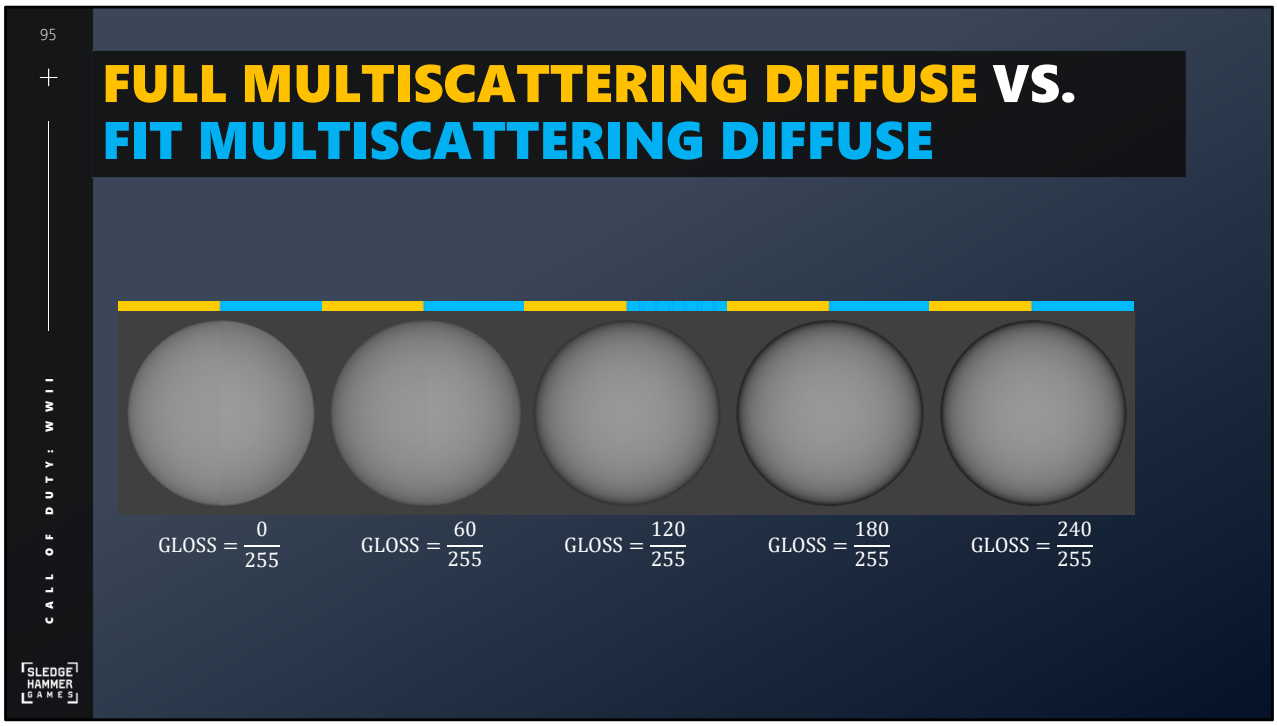


We also add a grazing angle retroreflective component, which is the most important ingredient for rough materials.



And we arrive at our final result.
This is what shipped in WWII.

* EDIT * Original slides omitted rho/PI for f_r. Rho is diffuse albedo.



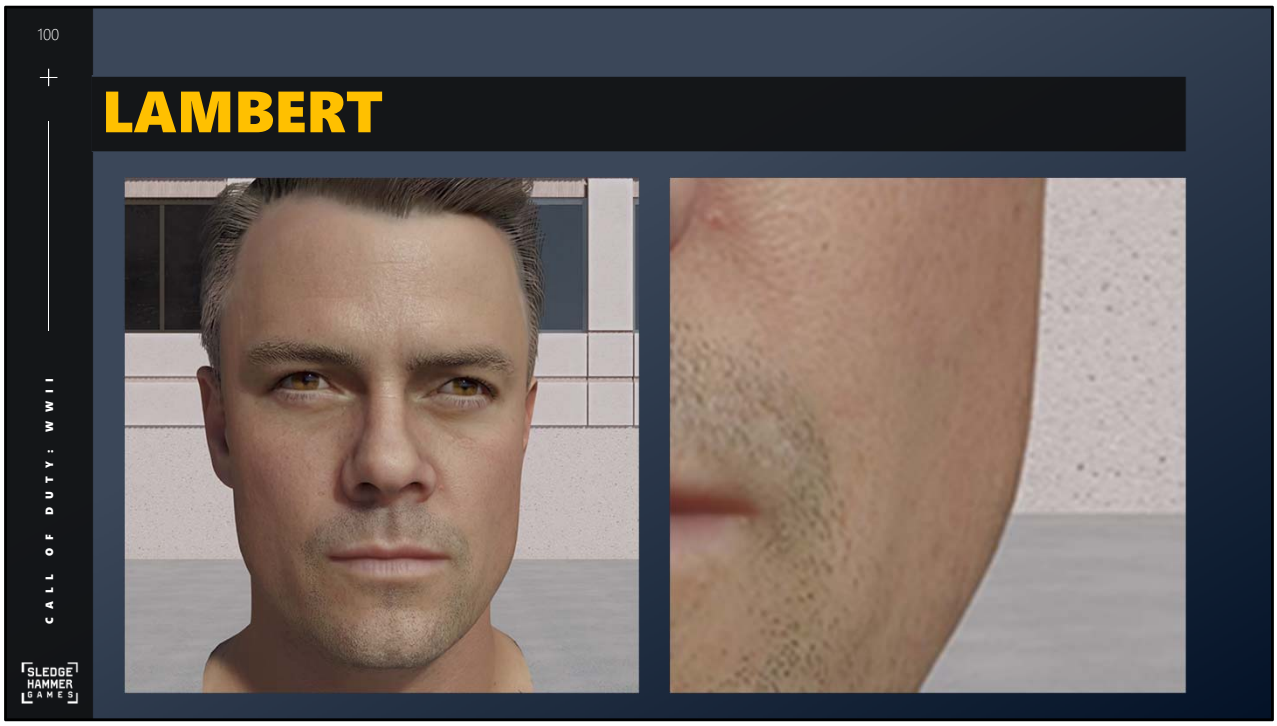
The left half of each sphere is the full simulated multiscattering BRDF.
The right half is our final hand-fit multiscattering model.
As mentioned earlier, the full model was first reduced to 2D by taking a BRDF Slice, before finding a fit.

Here is a material with gloss of 0, using Lambertian diffuse and GGX specular.

This is using the new Multiscattering Diffuse BRDF.
Notice the added brightness in the bumpy concrete floor from using a Multiscattering Diffuse BRDF.
This aligns with our measurements of bumpy concrete at the beginning of this section.
There is a subtle flattening of the lighting for this material.
[Toggle last frame]

This is a material with gloss = 1 using Lambertian diffuse and GGX specular.

This is using the new Multiscattering Diffuse BRDF.
One thing to keep in mind is that all the intermediate glossinesses provide an interpolation between flat and round diffuse lighting.
This results in better material differentiation based on gloss maps.
[Toggle last frame]
Notice the increased darkening of the silhouette of the right side of the object, the so-called “rounding” of lighting.
There’s a bit more richness to the material.



Here’s an interesting point.
You wouldn’t normally consider using Multiscattering Diffuse on skin which is soft and has subsurface scattering, but our character artists were first to ask to use it when it was an experimental feature hidden in a Debug menu.
This is Lambertian diffuse on our star character head.

Here is Multiscattering Diffuse applied.
The artists preferred the reduction of the dark silhouette when using Multiscattering Diffuse. In future work, we intend to validate this against photographs.
[Toggle with last frame]

LIGHTING STUDY

LIGHTING STUDY

There was a large team that worked on WWII.
It's the work of many people.

REFERENCES

[Bak11] Dan Baker, Spectacular Specular: LEAN and CLEAN Specular Highlights, GDC 2011

[Bak12] Dan Baker and Stephen Hill, Rock-Solid Shading, Advances in RTR Course SIGGRAPH 2012

[Bar12] Colin Barré-Brisebois and Stephen Hill http://blog.selfshadow.com/publications/blending-in-detail/ 2012

[Brd12] BRDF Explorer https://www.disneyanimation.com/technology/brdf.html 2012

[Bri16] Waylon Brinck and Andrew Maximov, Technical Art of Uncharted 4, SIGGRAPH 2016

[Bur12] Brent Burley, Physically Based Shading at Disney, PBR Course SIGGRAPH 2012

[Hei15] Eric Heitz and Jonathan Dupuy, Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation, 2015

[Hei16] Eric Heitz, Johannes Hanikia, Eugene d'Eon, Carsten Dachsbacher, Multiple-scattering microfacet BSDFs with the Smith model, SIGGRAPH 2016

[Iwa13] Michał Iwanicki, Lighting Technology of "The Last of Us", SIGGRAPH 2013

[Iwa17a] Michał Iwanicki and Peter-Pike Sloan, Precomputed Lighting in Call of Duty Infinite Warfare, Advances in RTR Course SIGGRAPH 2017

[Iwa17b] Michał Iwanicki and Peter-Pike Sloan, Ambient Dice, EGSR EI&I track 2017

[Jim16] Jorge Jiménez, Xian-Chun Wu, Angelo Pesce, Adrian Jarabo, Practical Real-Time Strategies for Accurate Indirect Occlusion, PBR Course SIGGRAPH 2016

[Joh] Steven G. Johnson, The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt

[Kar13] Brian Karis, Real Shading in Unreal Engine 4, PBR Course SIGGRAPH 2013

[Mal18] Paul Malin, HDR in Call of Duty 2018

[Mat03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, Leonard McMillan, A Data-Driven Reflectance Model, SIGGRAPH 2003

[Ola10] Marc Olano and Dan Baker, LEAN Mapping, I3D 2010 https://www.csee.umbc.edu/~olano/papers/lean/ 2010

[Ore94] Michael Oren and Shree K. Nayar, Generalization of Lambert's Reflectance Model, SIGGRAPH 1994

[Pac12] Romain Pacanowski, Oliver Salazar Celis, Christophe Schlick, Xavier Granier, Peirre Poulin, Annie Cuty, Rational BRDF, IEEE TVCG 18(11), 2012

[Pes15] Angelo Pesce and Michał Iwanicki, Approximate Models for Physically Based Rendering, PBR Course SIGGRAPH 2015

[Sch94] Christophe Schlick, An Inexpensive BRDF Model for Physically-based Rendering, CGF 13(3) 1994

[Ste96] James Stewart and Michael S Langer, Towards Accurate Recover of Shape from Shading under Diffuse Lighting, IEEE PAMI 19(9), 1997

[Sun05] Bo Sun, Ravi Ramamoorthi, Srinivasa Narasimhan, and Shree Nayar, A Practical Analytic Single Scattering Model for Real Time Rendering, SIGGRAPH 2005

[Tok04] Michael Toksvig, Mipmapping Normal Maps http://www.nvidia.com/object/mipmapping_normal_maps.html 2004

[Wol98] Lawrence B. Wolff, Shree K. Nayar, Michael Oren, Improved Diffuse Reflection Models for Computer Vision, IJCV 30(1), 1998