Hi everyone, my name is Sébastien Lagarde and, together with Charles de Rousiers, we will be presenting our work on moving the Frostbite engine to Physically Based Rendering (PBR).

Acknowledgments

- Contributions from ***many*** people
- This talk and the course notes are about:
  1. Summarizing all of the steps to move an engine to PBR
  2. Using the state of the art in our base implementation
  3. Small improvements in quality

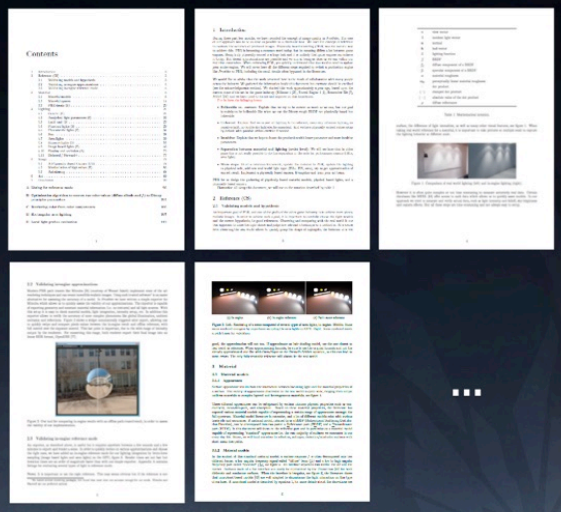
First, I would like to mention that this work has been made possible thanks to many people – not just within the Frostbite team, but also from the global rendering community.

Our base implementation follows the state of the art for PBR, as most other engines are doing it.

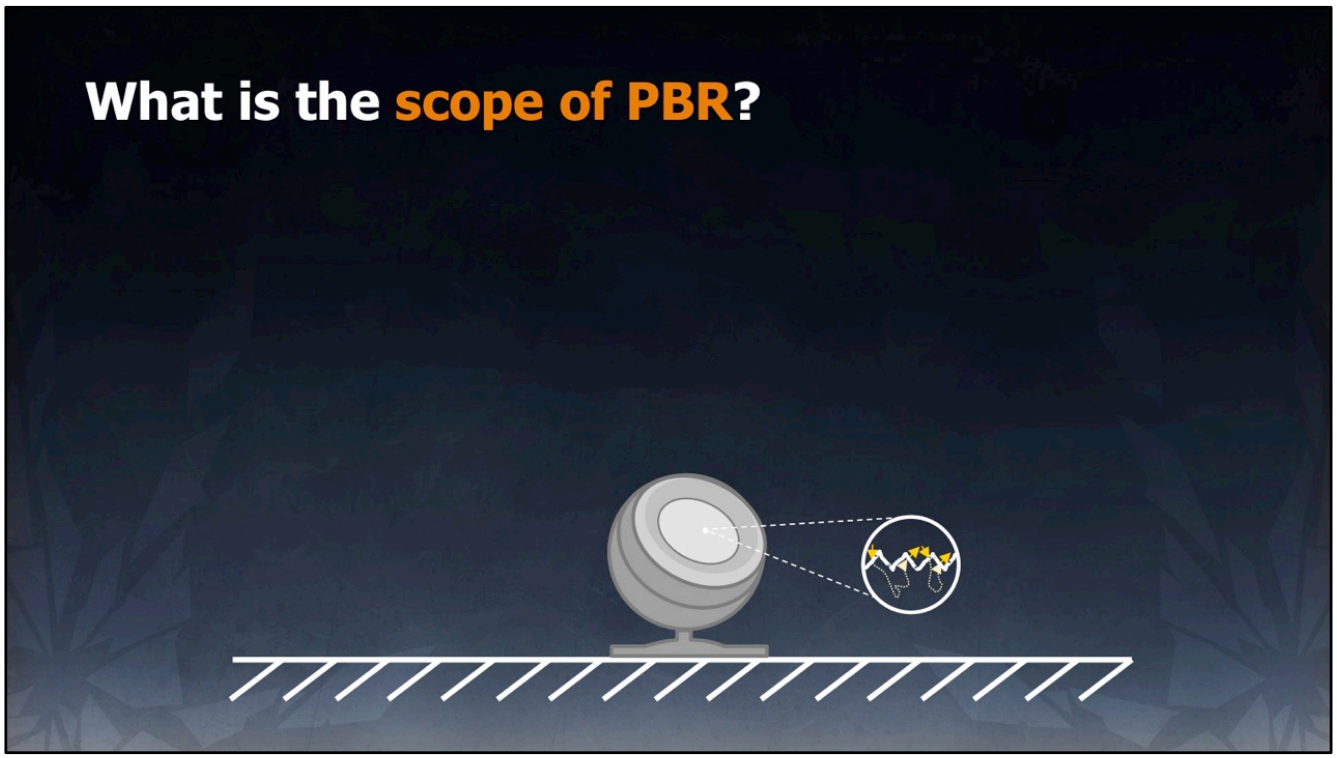
We have also made some small quality improvements.

Disclaimer

- This presentation is about:
  - A high level overview
  - Steps to move to PBR
- Courses notes come with full details and code

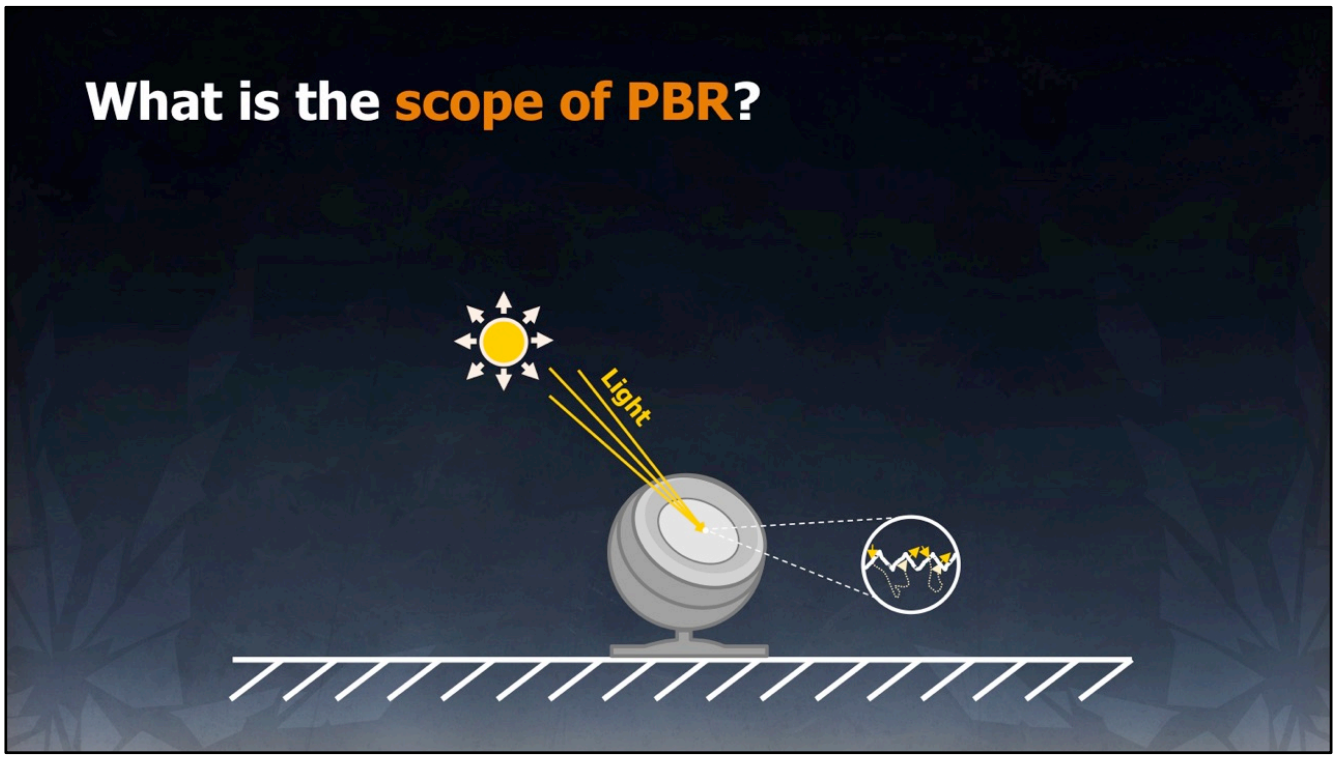


This presentation gives a high-level overview of our work and covers a few of the steps in moving to PBR. All of the details – including code – are available in the course notes.

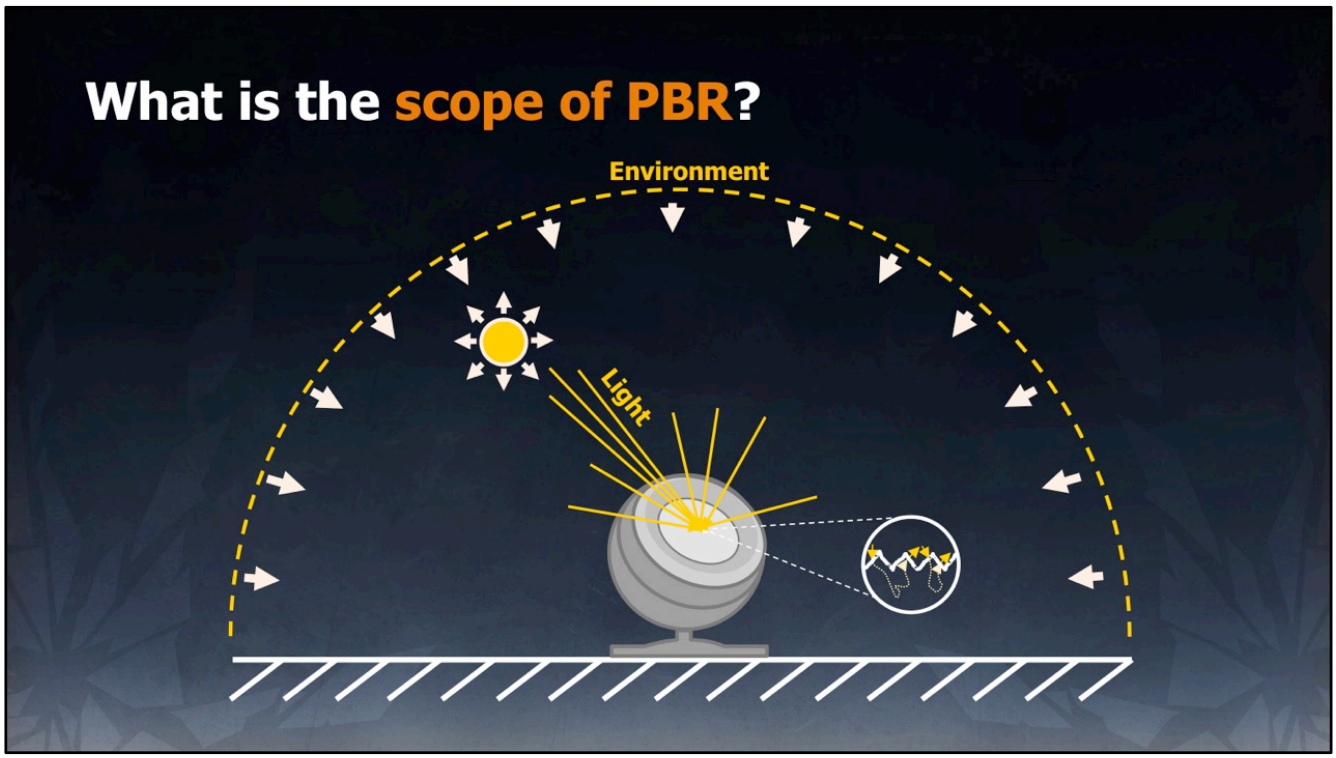


What is the scope of PBR for us?

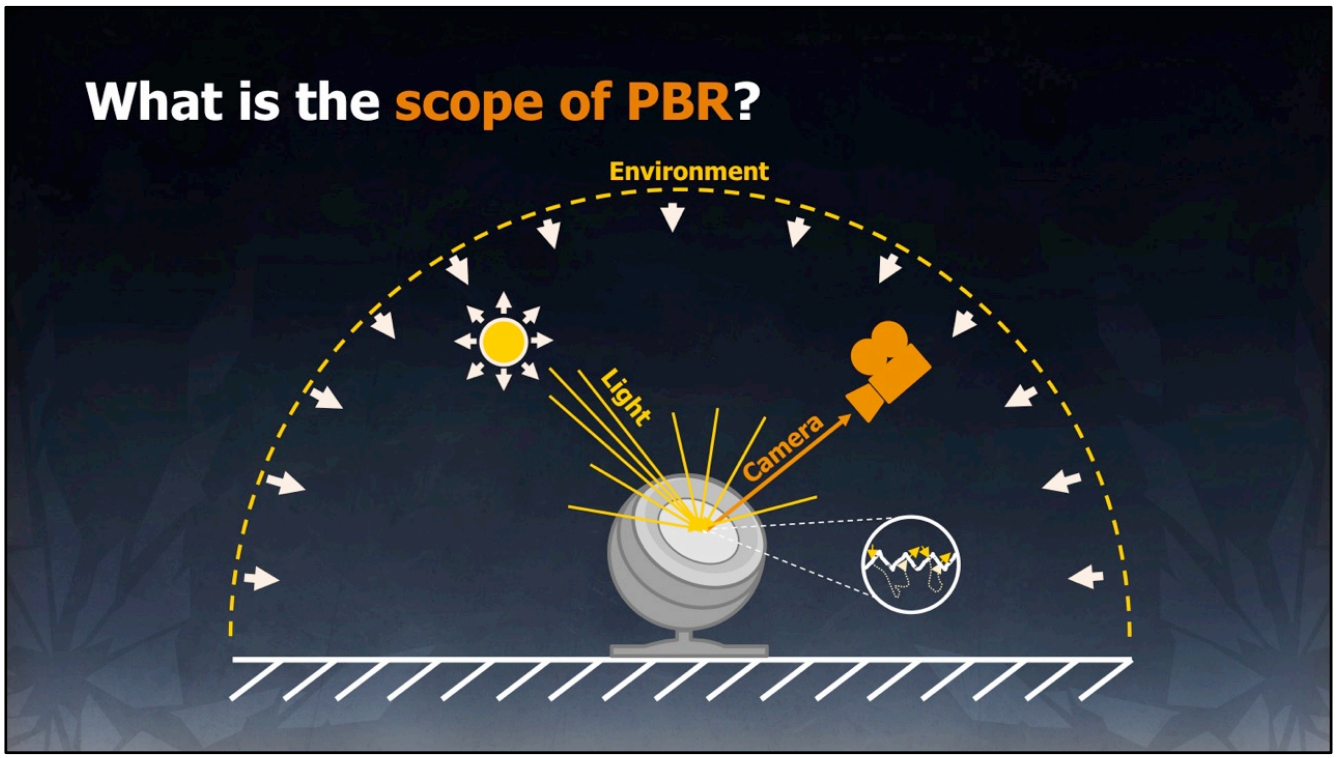
Well, we have matter…



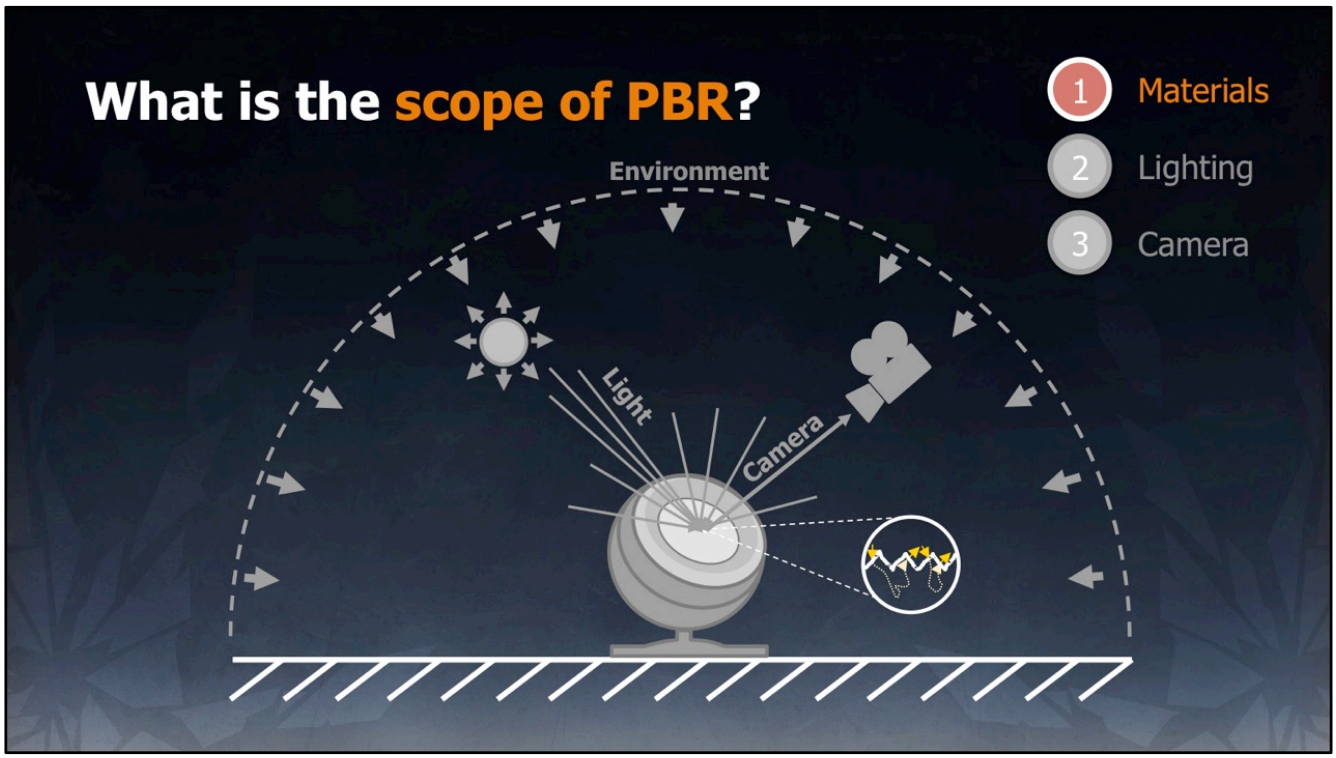
A light interacting with the matter…



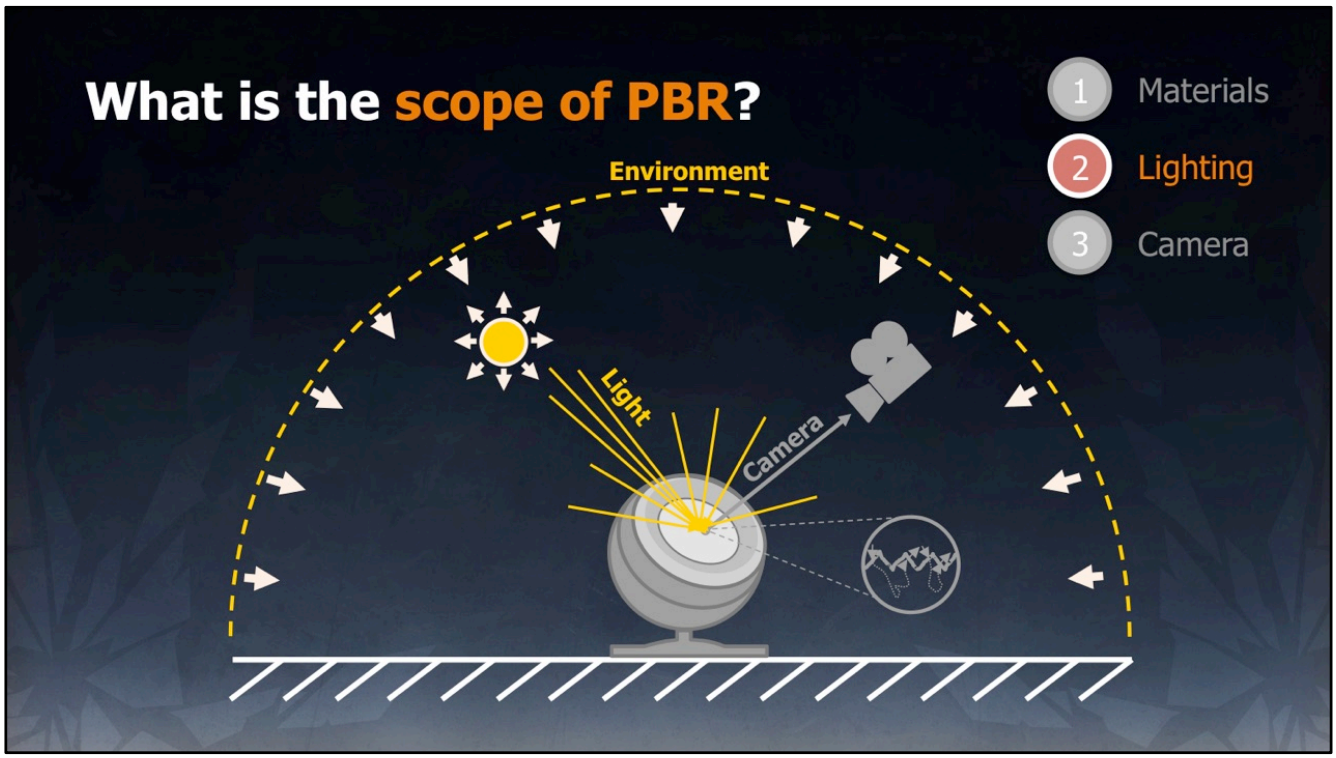
Environmental lighting – from the sky and other elements in the scene…



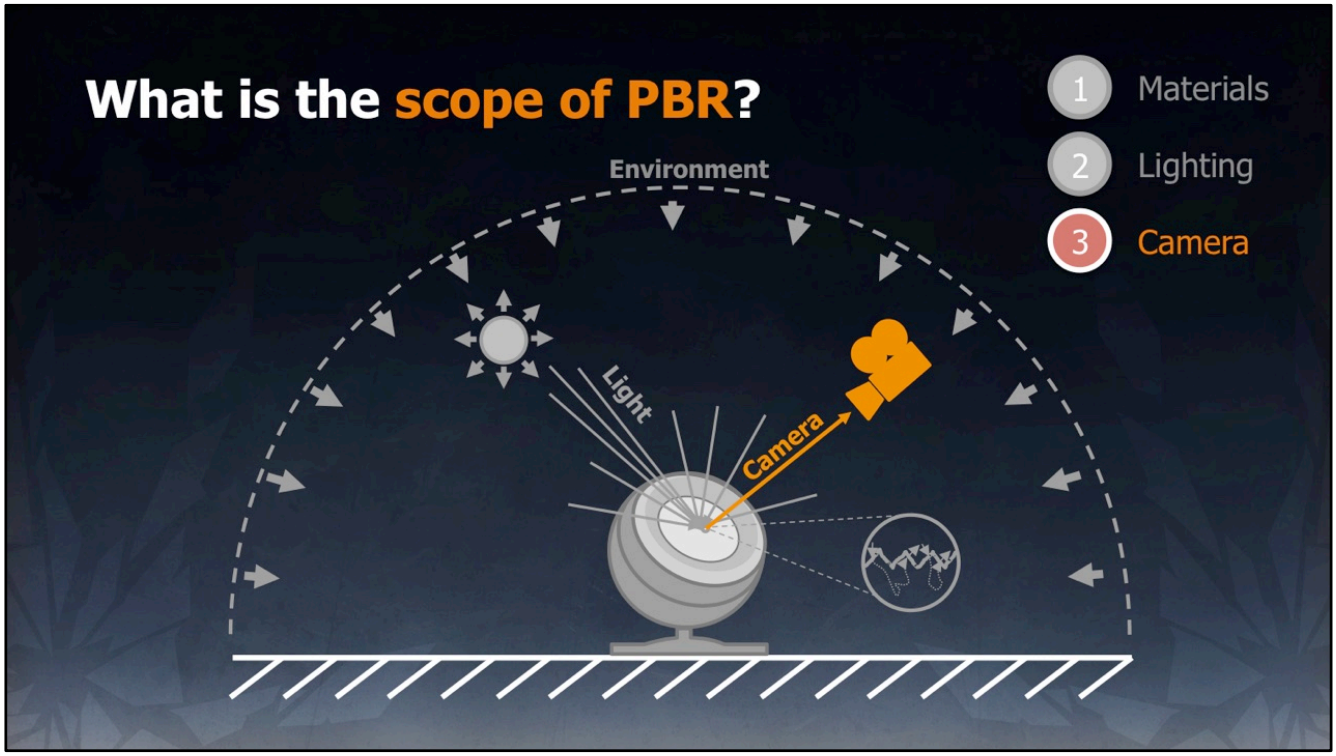
…and an observer.



For us, PBR means these three components: Materials…


What is the scope of PBR?
1 Materials
2 Lighting
3 Camera
Environment
Light
Camera

…lighting…

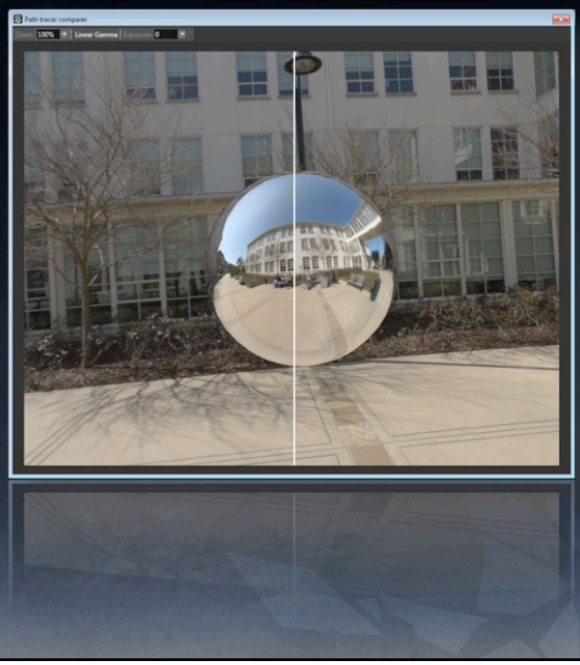


…and the camera.

But before talking about these components, the first thing to do when working with PBR is to develop…

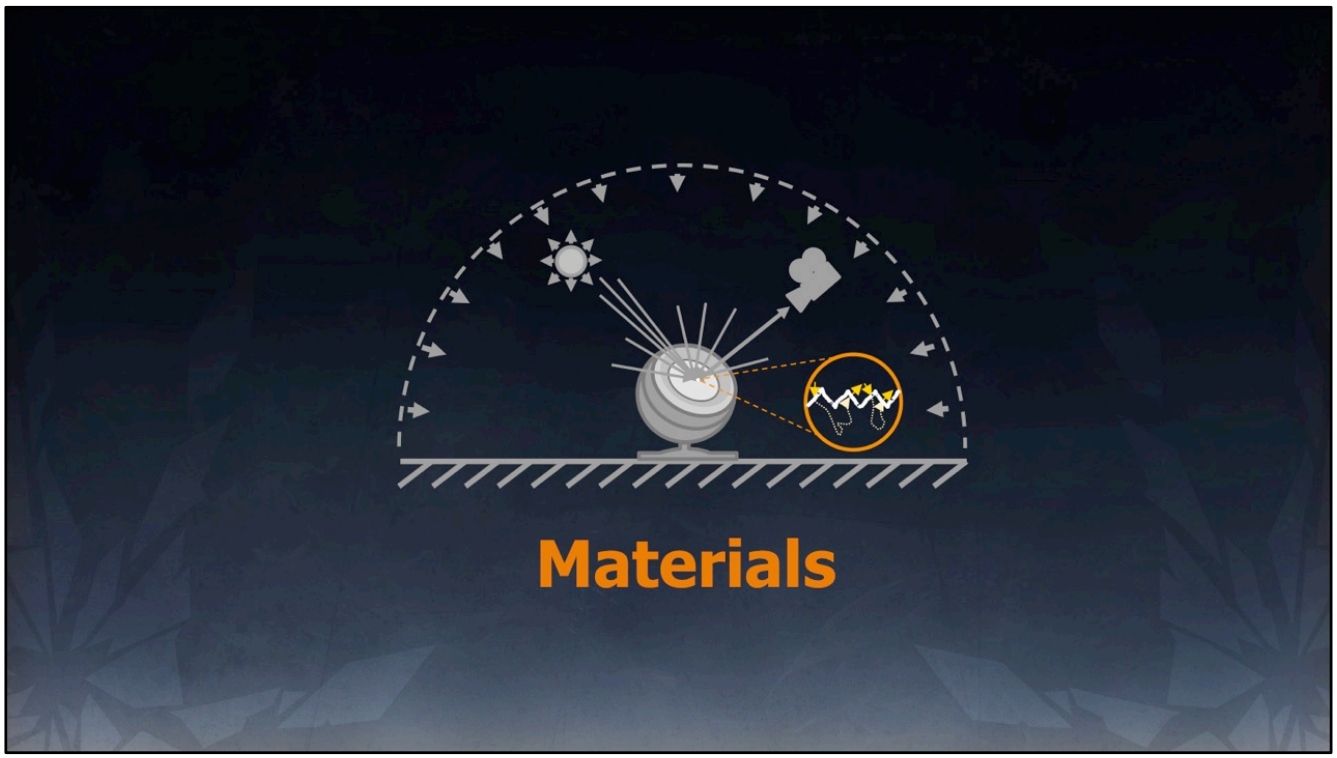
Reference framework

- Need good reference
  - Export to offline path-tracer
  - In-engine reference



…a reference framework.

From the beginning, we tried to find a way to compare our results.
We built an exporter from Frostbite to Mitsuba, which is an offline path tracer.
But the process was too slow. So, to rapidly validate our rendering, we have built an in-engine reference mode.



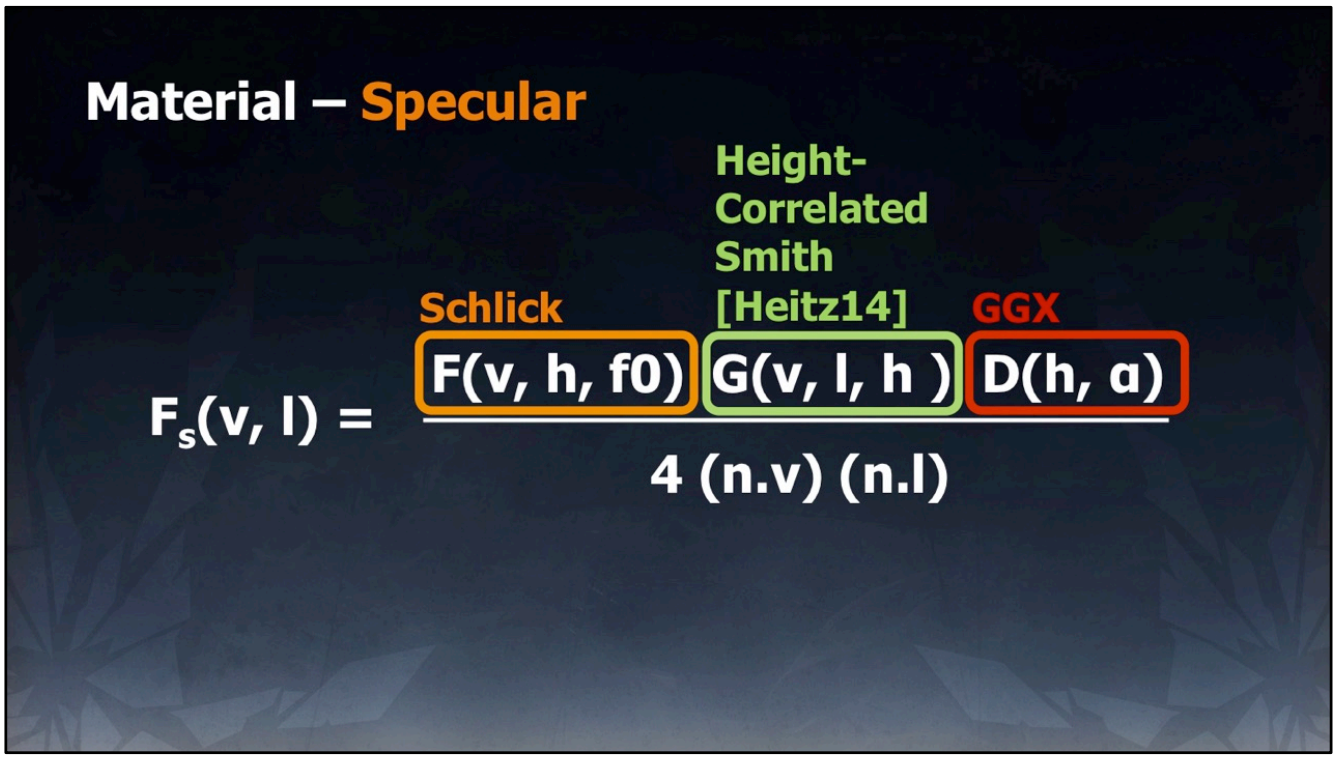
So, materials…

Material – Standard Model

- **Standard** material
  - **80% of appearance types**
  - **Model**
    - **Specular**: Microfacet model with GGX NDF
    - **Diffuse**: Disney's model
- **Other** material types
  - Subsurface material
  - Single layer coated material

Our standard material is composed of a diffuse term and a specular term – nothing new there. It allows us to represent 80% of our cases.

We support other kinds of material through additional BRDFs – such as clear coating or subsurface scattering – but we will not discuss them here.

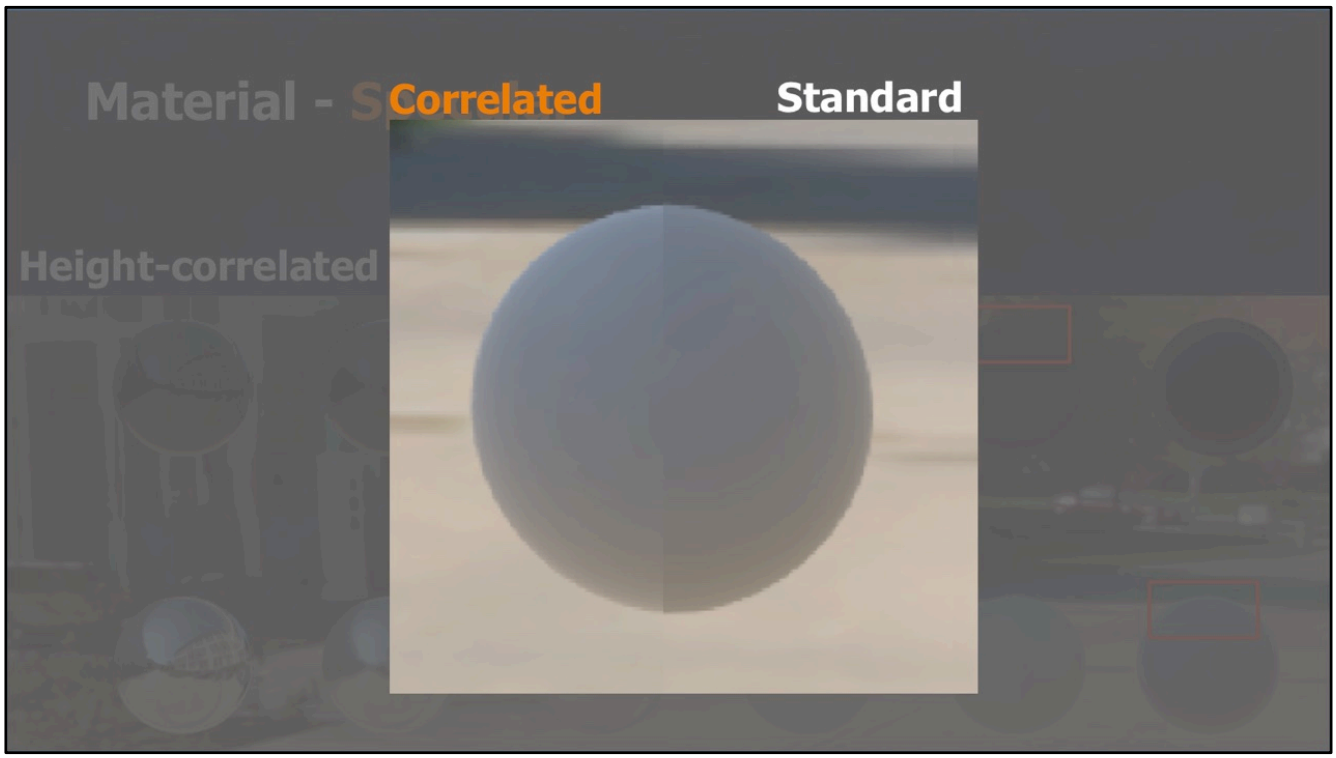


Our specular term is a traditional microfacet model, using Schlick's approximation of the Fresnel function, and the GGX Normal Distribution Function (NDF).

One small deviation from common practice is the geometric term, which does not use the regular (uncorrelated) Smith function. Instead we use a height-correlated version, as it is "theoretically" more correct – see Eric Heitz' course notes for a thorough discussion.

Here is a simple comparison. First the uncorrelated Smith G term, for a dielectric material with black diffuse (top) and a metal material (bottom).

And here the height correlated version. The difference is subtle but noticeable for high roughness values (on the right).



Here’s a side-by-side comparison for high roughness.

Material – Diffuse

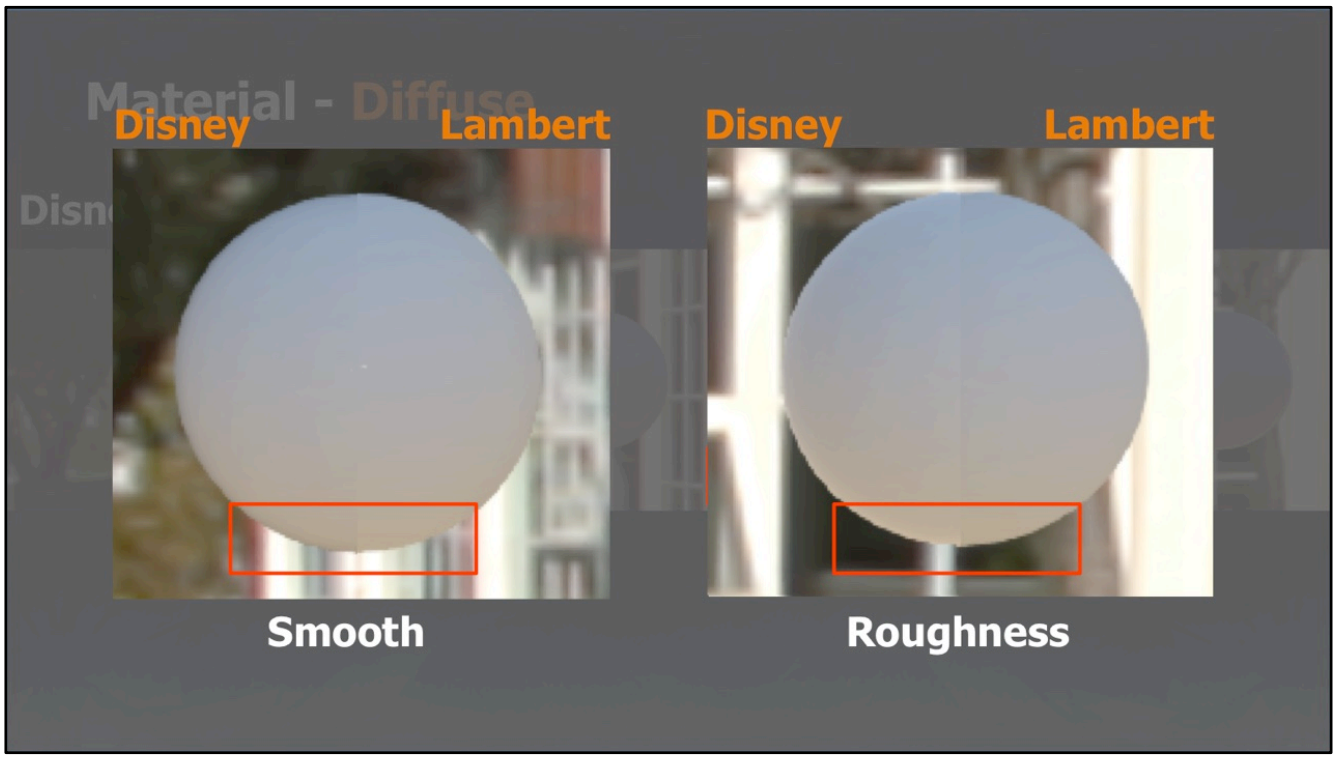
- Disney Diffuse [Burley12]
  - Coupled roughness between diffuse and specular
  - Retro-reflection

For our diffuse term, we wanted to couple the roughness of the specular term and the one from the diffuse term. We also wanted to get retro-reflective behavior when the roughness is high.

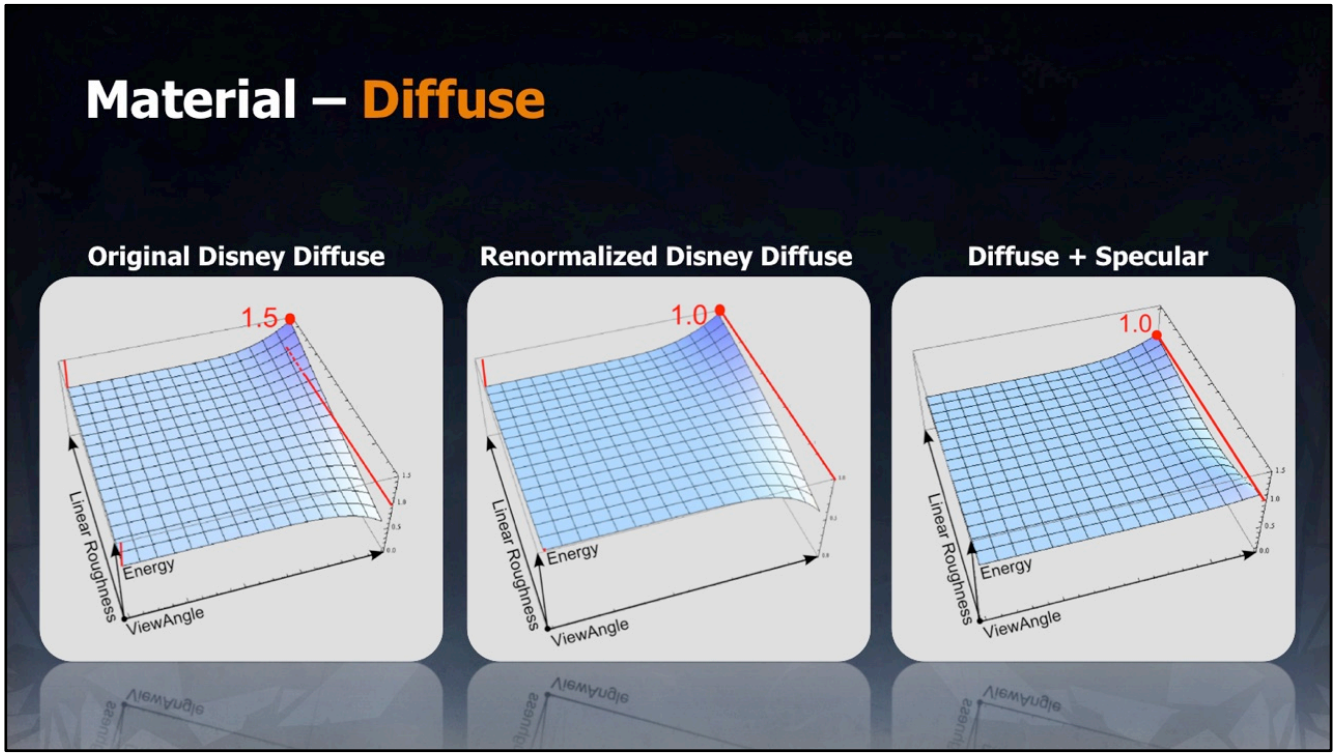
We have investigated proper derivations from the GGX distribution as [Gotanda14], but we went for Disney's diffuse model (from [Burley12]) as it is simple and good enough.

As a quick reminder of Disney's diffuse model, here is a comparison. This is Lambertian diffuse, which is the same for all roughness values.

Here is Disney's diffuse, which is a bit darker at low roughness, and brighter at high roughness. Subtle, but it can make a difference.



Here is a side-by-side comparison, with low roughness (left), and high roughness (right).



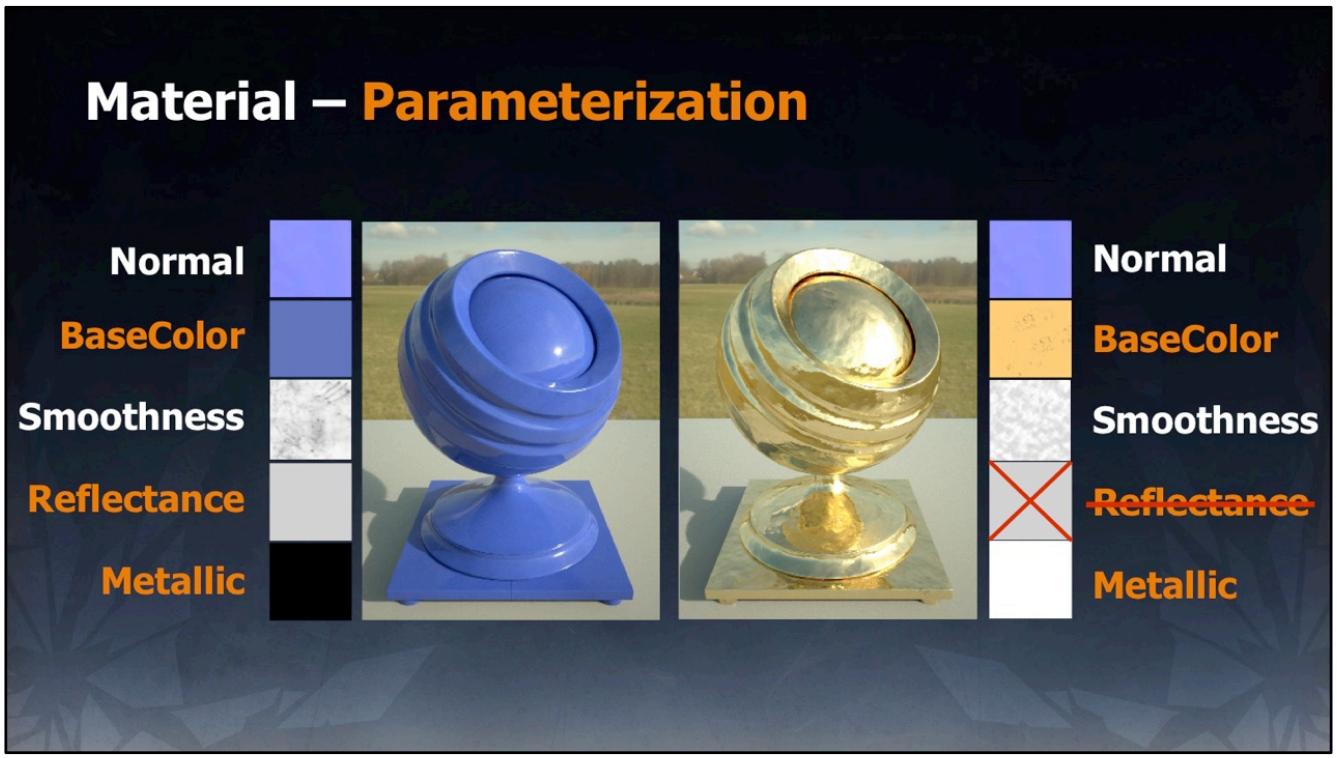
One issue we had with the original Disney diffuse term is that it does not respect energy conservation: in some cases, the reflected light can be higher than the incoming illumination.

We have applied a simple linear correction to ensure that the hemispherical-directional reflectance is below 1 when we add the specular and the diffuse terms together.

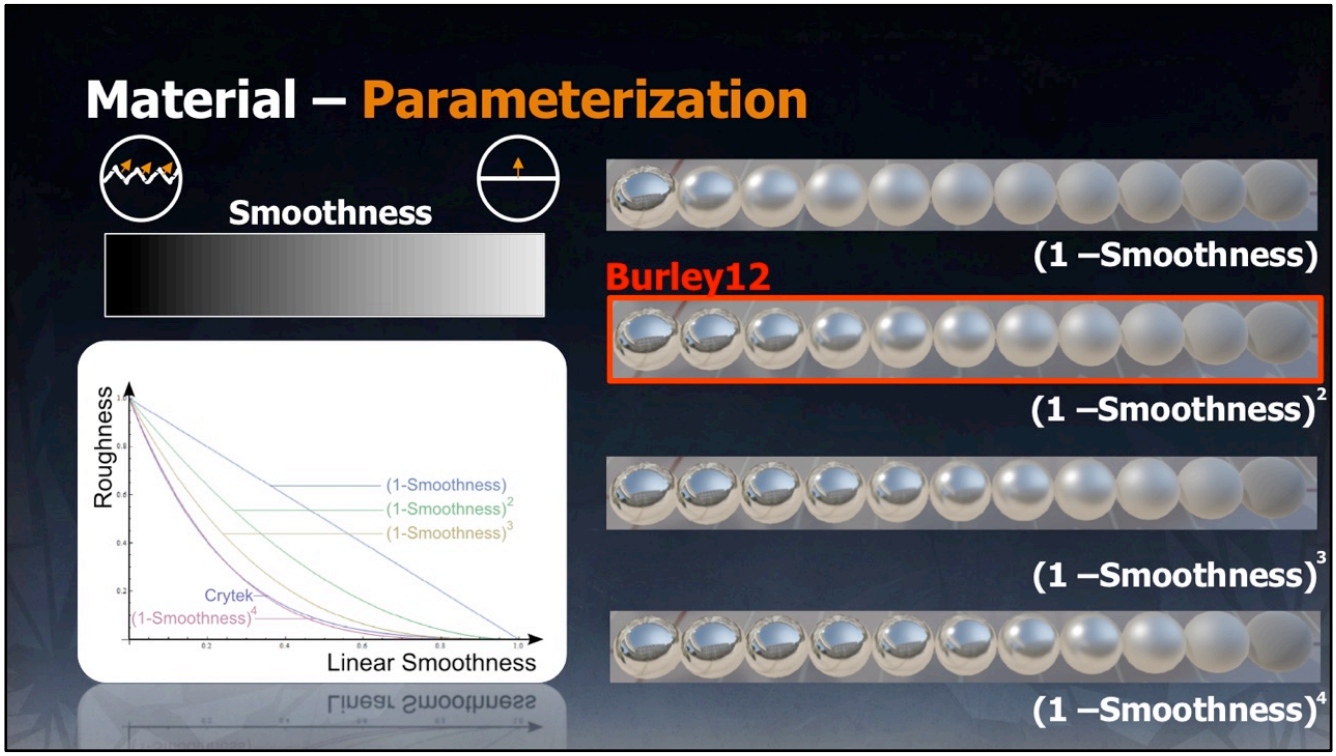
For comparison, here is the un-normalized term…

…and here's the renormalized version.

Notice that the model appears darker, particular for high roughness values (right), where there is more retroreflection.

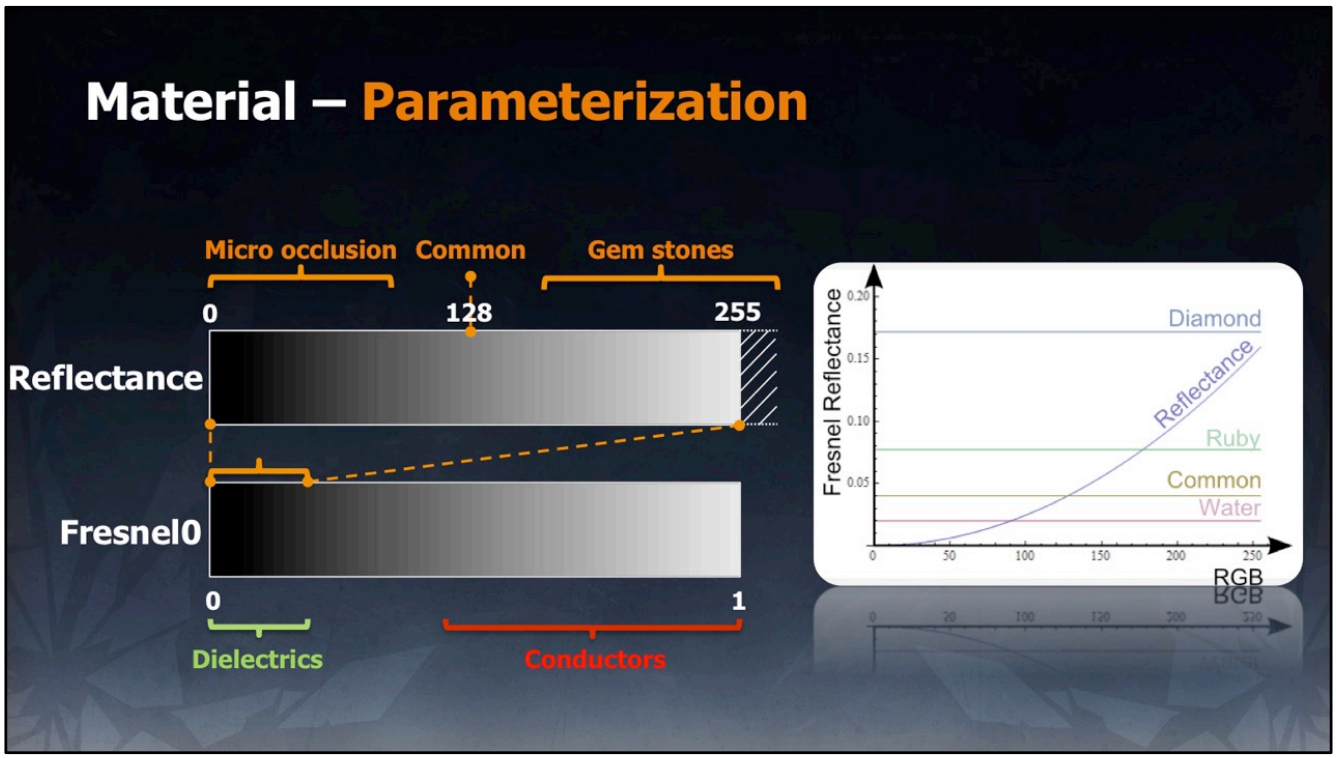


For the input values of the diffuse and specular terms, we have again followed Burley's approach, which decouples dielectrics and metals for easier authoring.



We chose to expose ‘smoothness’ to the artists rather than roughness, because white = smooth is more intuitive for them.

We also experimented with various remapping functions in order to get ‘perceptually linear’ roughness. Again, we ended up using Burley’s approach of squaring the roughness.

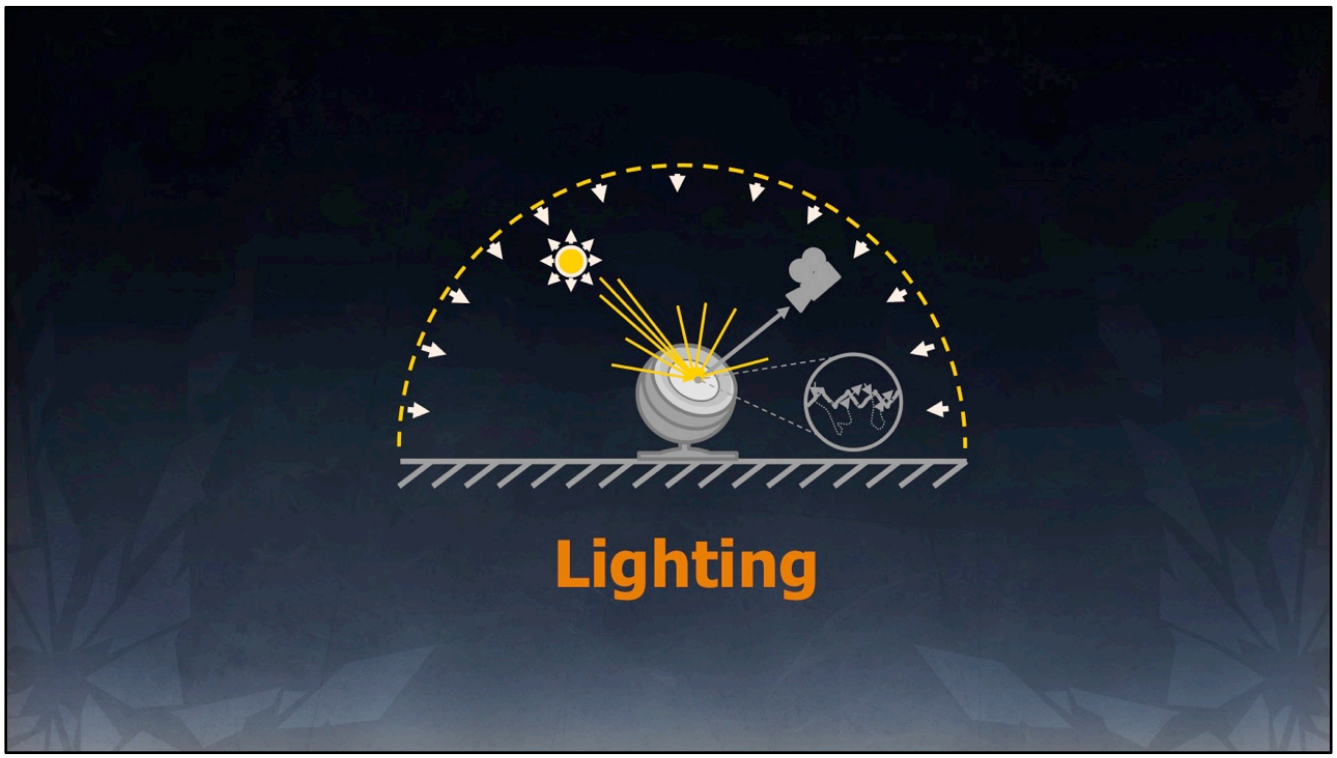


'Reflectance' controls the Fresnel value for dielectric materials.

It represents a subset of the full f0 range, in order to get more precision.

We also wanted to fit a couple of semantics into it.
- The bottom range is dedicated to specular micro occlusion.
- The mid-value is remapped on the common 4% of reflectivity
- The top range is dedicated to gem stones. We have bounded it to 16%, even if some gemstones like diamond are around 17%. The goal was to keep the decoding of f0 as simple as possible.

An example of in-engine results.



That covered the material section. Now let's take a look at how light interacts with these material properties.

Lighting

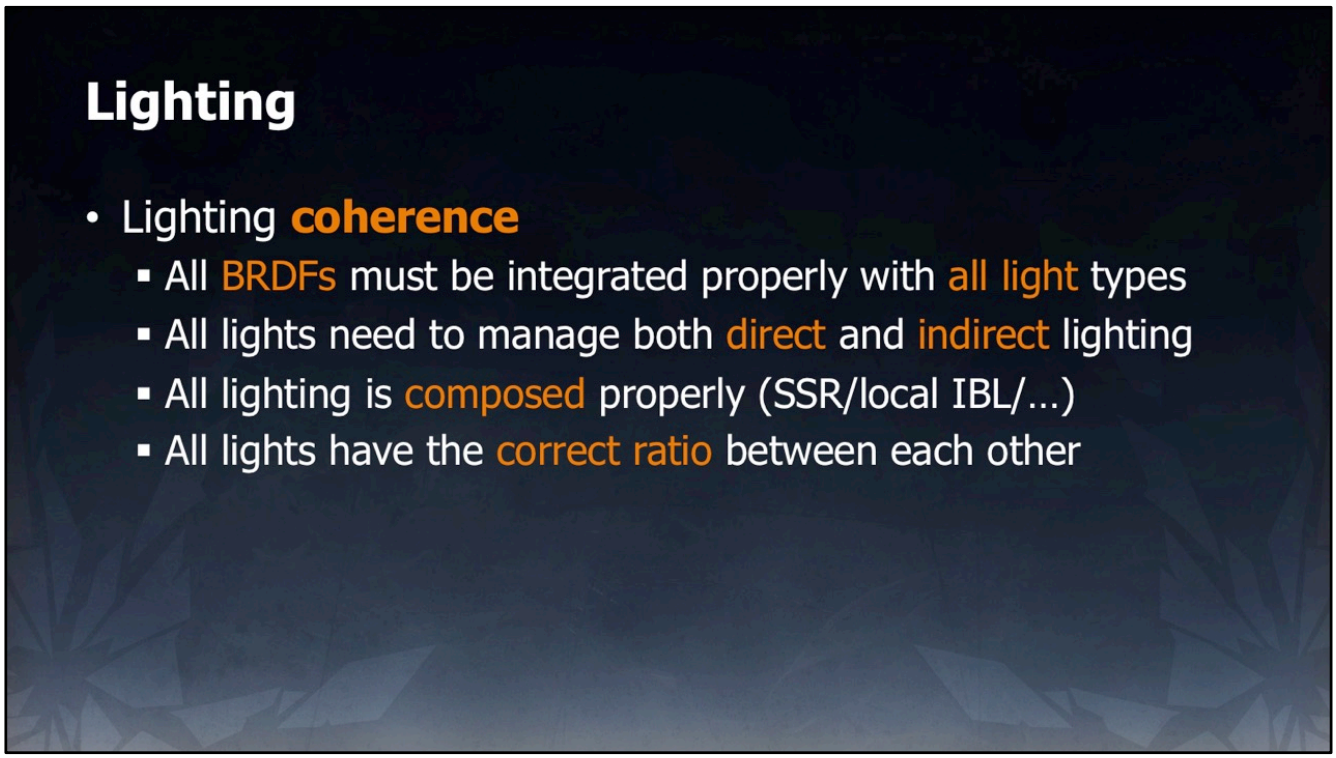
In the real world, an object will receive lighting from everywhere.

Analytical lights surrounding the space.

Outside, from the sun and the sky.

But also the entire surroundings.

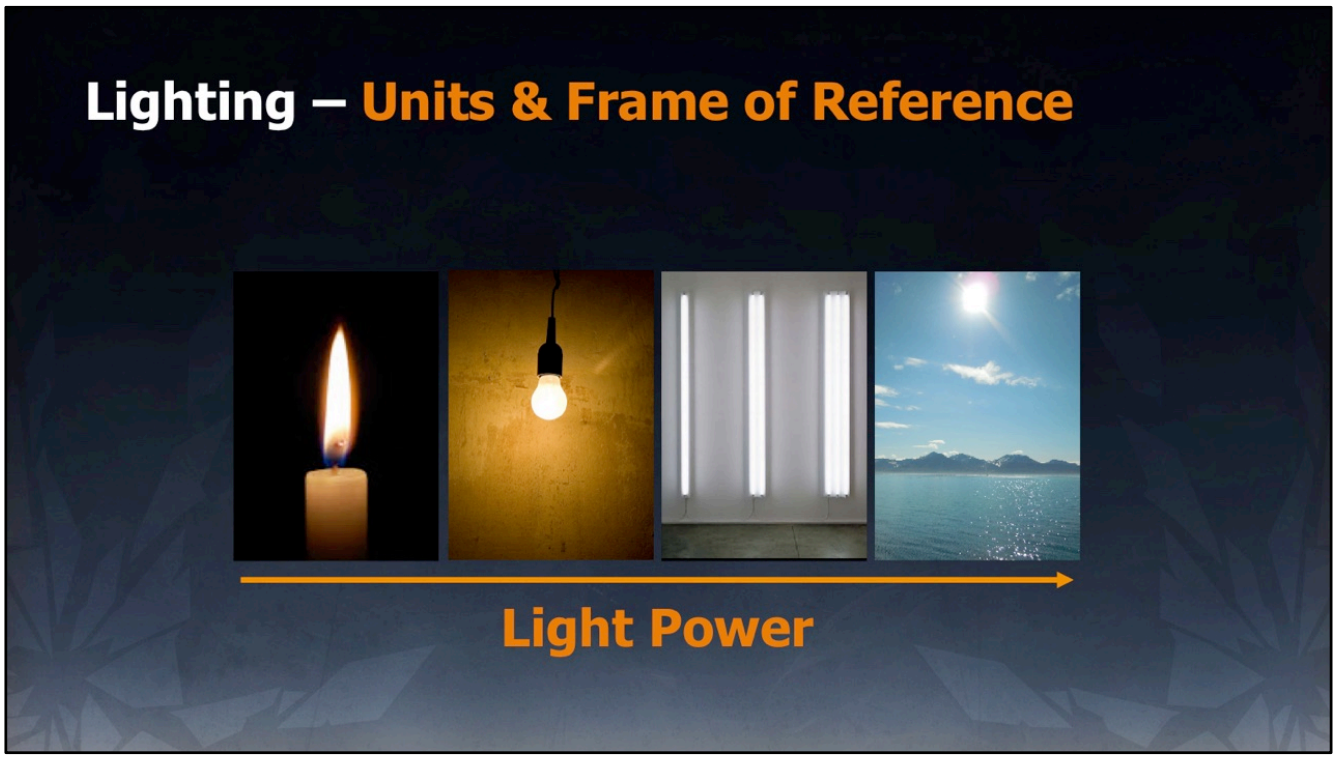
One important point to note here is that all lights are visible at the same time, with different intensity ratios.



This highlights that coherence in lighting is key for achieving realistic images.

Coherence in an engine needs to be respected in several places:
- All of the BRDFs must be integrated properly with all light types – this includes area lights and image-based lights (IBLs).
- All lights need to affect both direct and indirect lighting. For instance, any new area light type must also be added to the radiosity system.
- When lighting with different techniques like screen-space reflections (SSR) or light probes, all lighting must be combined properly – to avoid 'double counting', for example – and it must transition smoothly.

In addition, the intensity ratios between all of the lights needs to be correct.

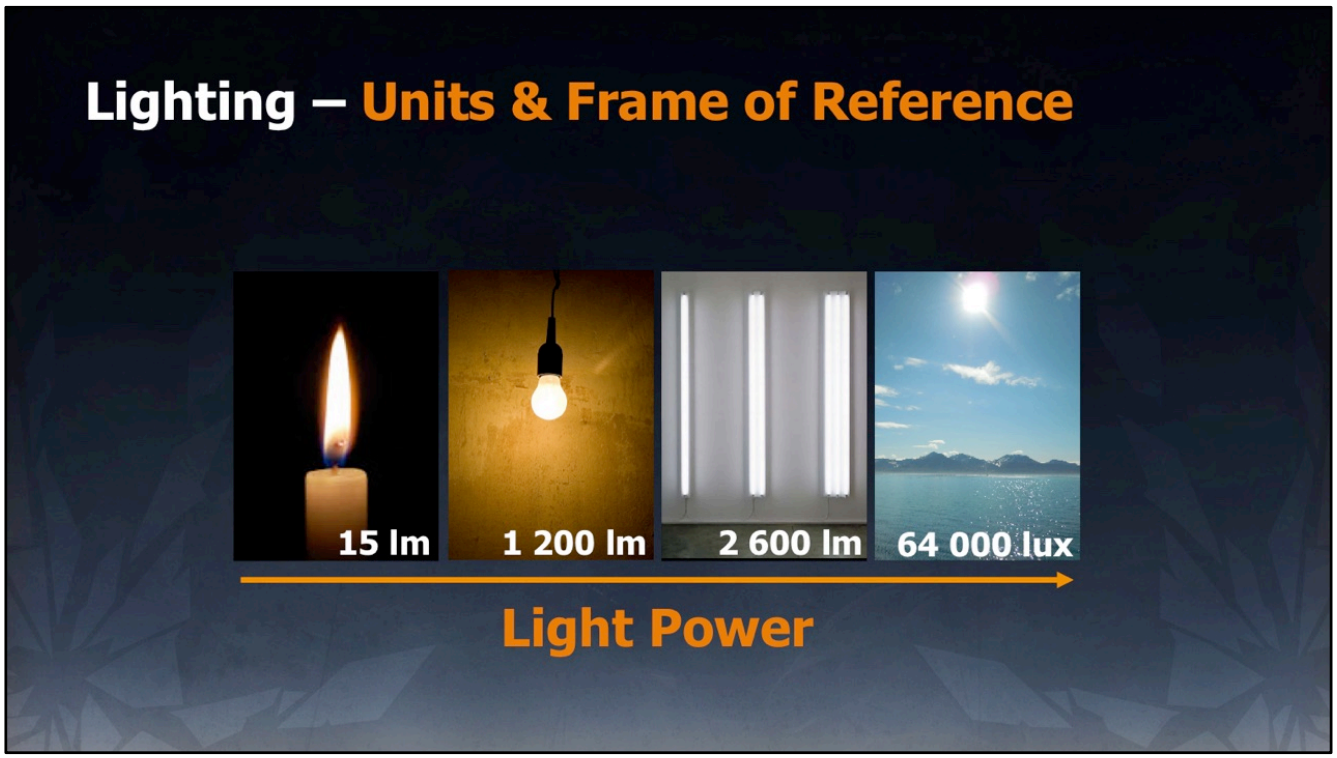


We know that light sources span a wide range of values: a candle emits little light, the sun is super bright.

In the past, artists have typically defined an arbitrary value for the strongest light of a scene. Then they define the intensity of other lights relative to this key light.

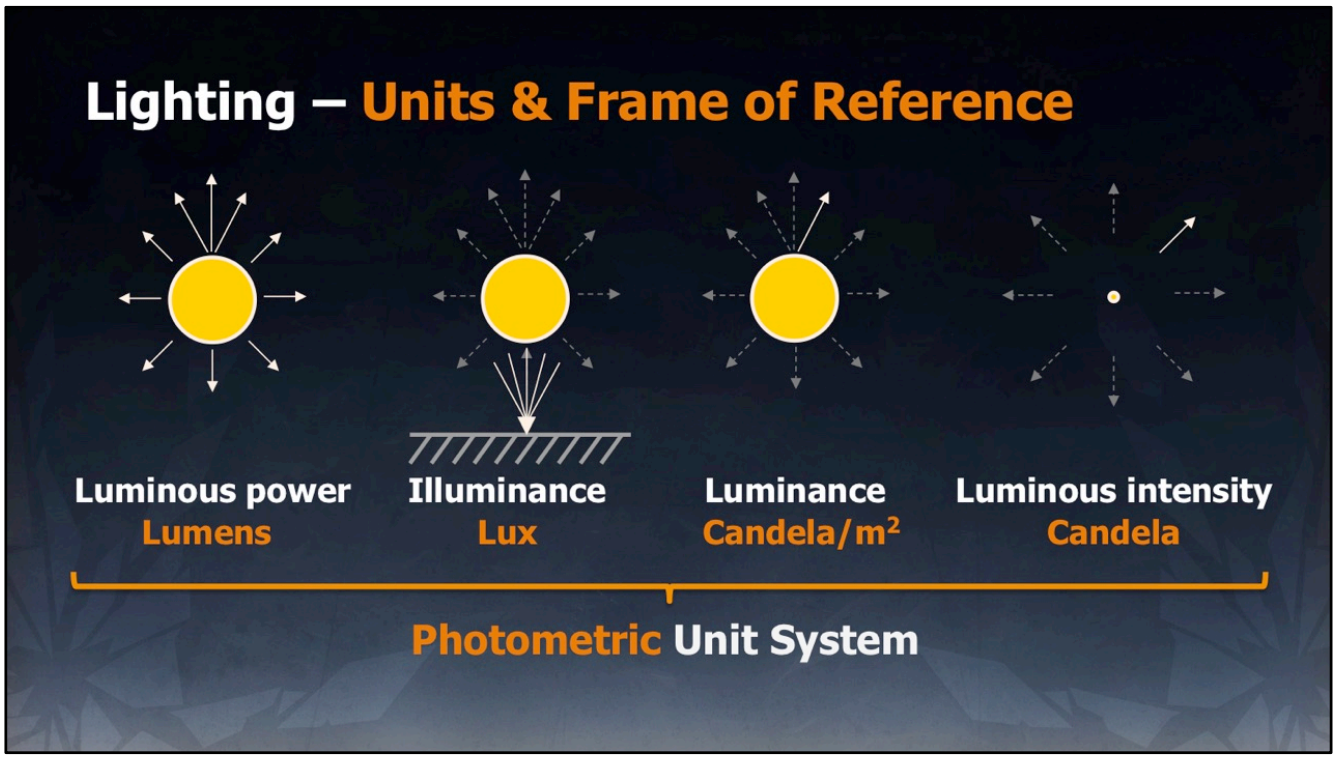
This can make it difficult to tweak the lighting, or reuse light setups.

A better approach is to use a common frame of reference…



By using a consistent unit system for light intensities – photometry for instance – one can determine accurate (physical) values for any given light source.

By having a single frame of reference, we can now ensure consistency for all our lighting, and we can reuse some lighting rigs on different scenes.



Frostbite's entire lighting pipeline is in photometric units.

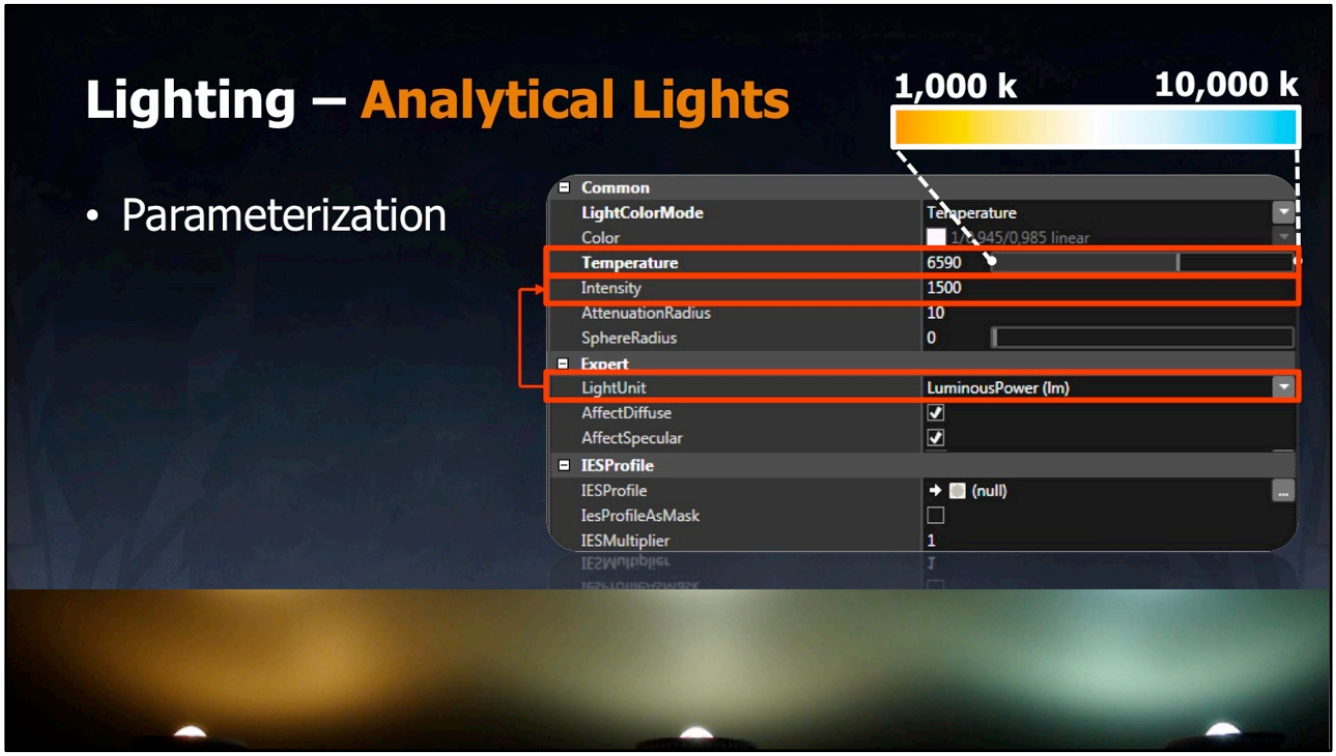
We use four types of units.

Lighting – Units & Frame of Reference

Punctual	Luminous power (lm)
Photometric	Luminous intensity (cd)
Area	Luminous power (lm), Luminance (cd/m^2), or EV
Emissive	Luminance (cd/m^2) or EV
Sun	Illuminance (lux)

These are the various analytical light types found in Frostbite, with their corresponding units.

We will go through all of them.



For our analytical lights, we have separated our light settings into intensity and color.

Artists use the appropriate units, depending on the light type (see previous slide).

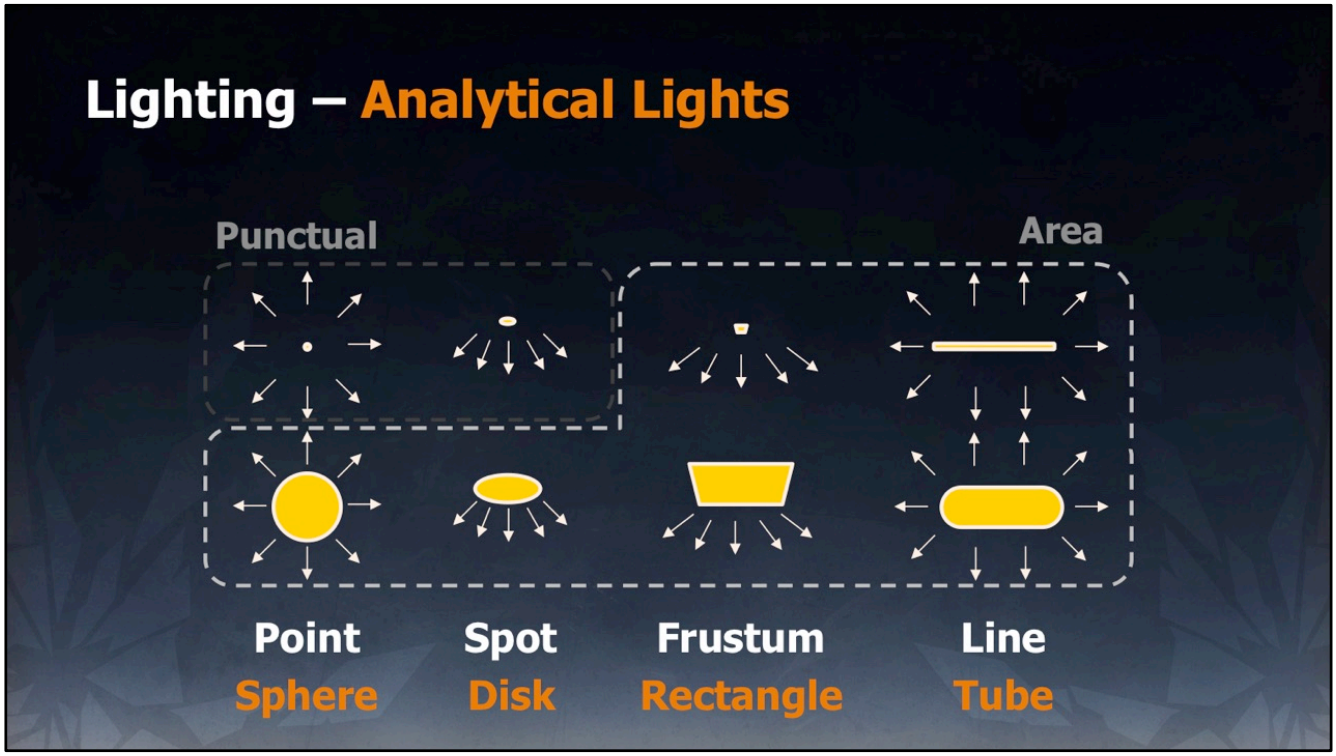
For the color, they can choose to derive it from a color temperature.

Lighting – Units & Frame of Reference

Such settings are convenient because they can be taken from light bulb packages or gathered over the internet.

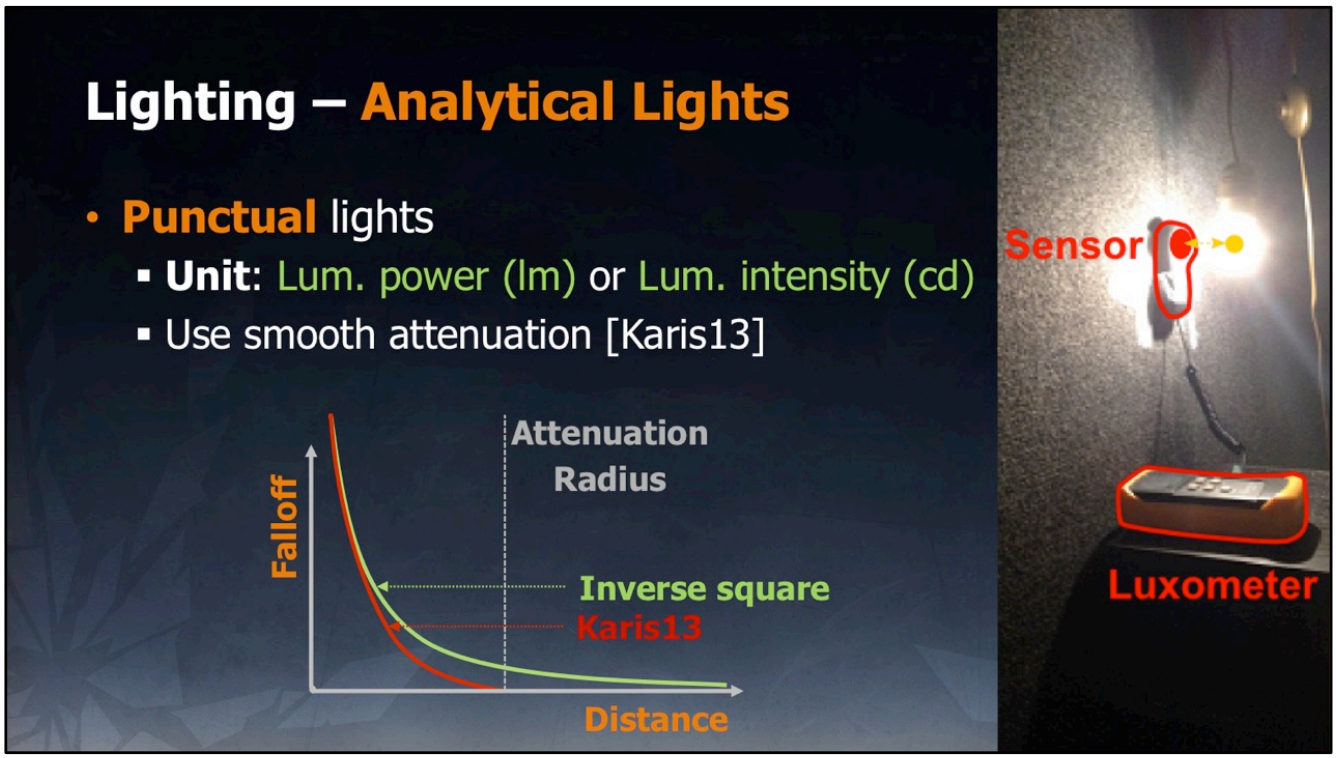
Lighting – Analytical Lights

Real-world lighting is made up of area lights of various shapes.



To cover this variety, we support four different shapes: sphere, disk, rectangle and tube.

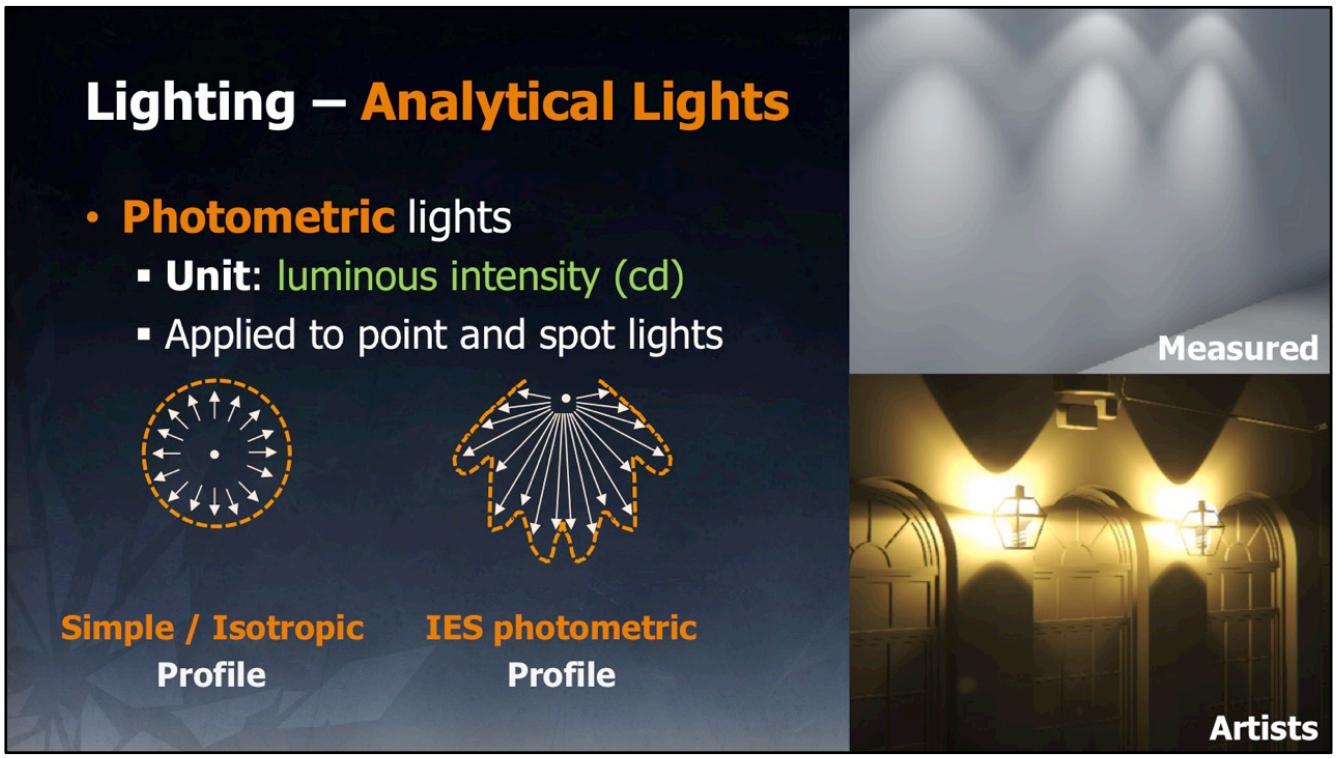
Each of these lights could have a simpler version, but only point and spot use a punctual light path, as they are more frequent due to their low cost.



Our punctual lights are pretty common. They follow the well-known inverse square law, and for performance reasons they attenuate to 0 when they reach a given attenuation radius.

For this we have adopted the smooth attenuation function presented by Brian Karis – see his 2013 course notes for details.

To validate our in-engine lighting equation, we performed real world measurements of a light bulb, and it turns out that the physics is correct!

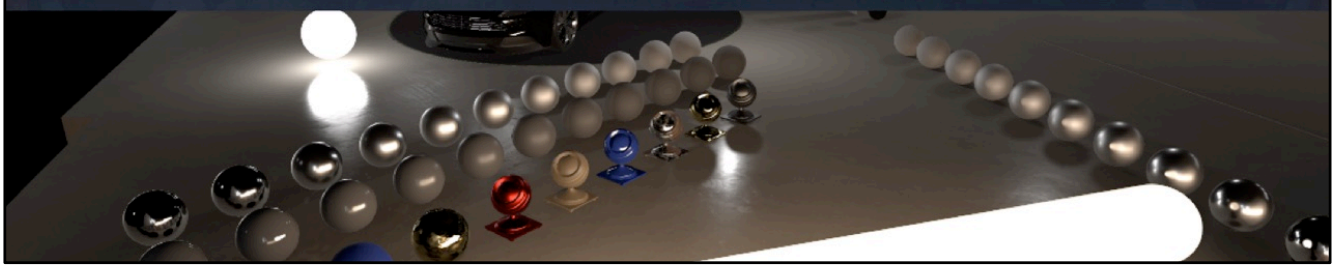


An IES profile can be attached to our punctual lights.

The IES profiles have different intensity depending on the direction. They can come from real-world measurements – available from manufacturers' websites – or they can be authored by artist in a creative way.

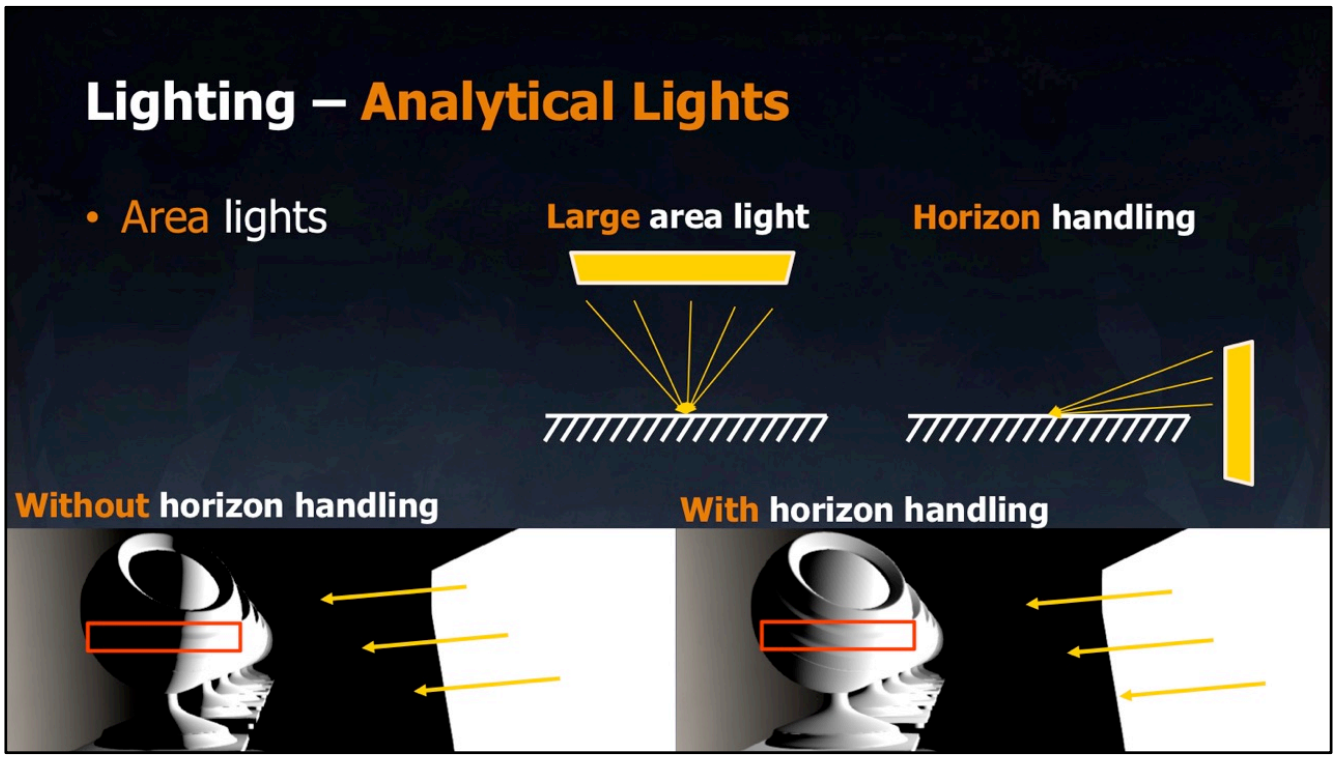
Lighting – Analytical Lights

- **Area** lights
  - **Unit**: Luminous power (lm), Luminance (cd/m^2 or EV)
  - Separate diffuse and specular evaluation



Area lights should be ’first-class citizens’ of a PBR engine. Sadly, they are quite complex to implement and in Frostbite they are currently not ready for production.

We have separated the evaluation of area lights into diffuse and specular components.



Correctly handling the diffuse part is key to obtaining the soft lighting provided by a large area light.

When lit with large area lights, objects should exhibit wrapped lighting. To have correct lighting intensity in this wrapped region, it is important to consider the horizon.

This means that when the light starts to cross the plane defined by a given shading point and normal, the intensity must decrease. In Frostbite, all of our diffuse area lights account for this.

Lighting – Analytical Lights

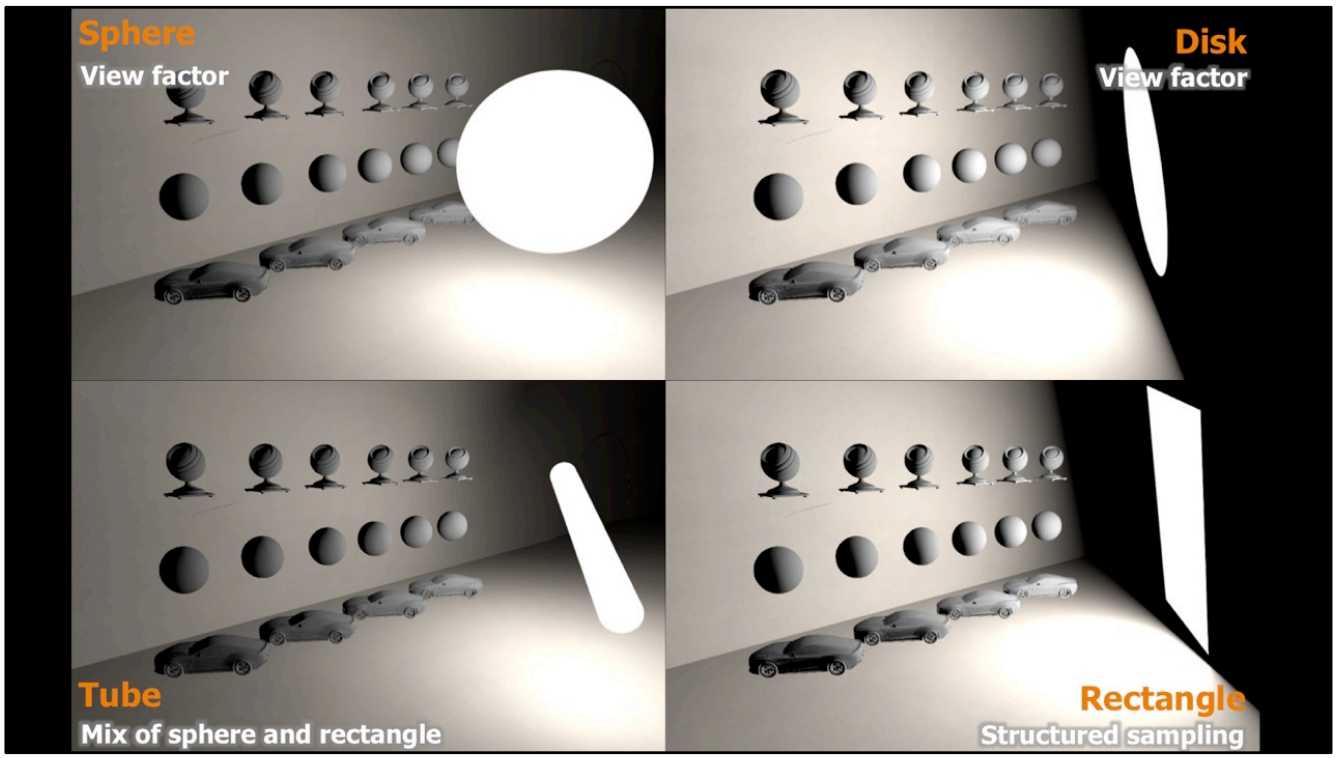
- **Diffuse** area lights
  - 3 integration techniques:
    - Analytic
      Form factors (radiosity) / view factors (heat transfer)
    - MRP
      Solid angles x Most Representative Point lighting [Drobot14]
    - Structured sampling of light shape
      Solid angles x average cos

We have explored three integration techniques for diffuse area lighting.

The first one is based on analytic integration. The well-known ‘form factor’ from radiosity provides the exact result of the diffuse integration. There is an equivalent formulation referred to as the ‘view factor’ in heat transfer science.

The second relies on the ‘most representative point’ (MRP) method presented by Michał Drobot in his GPU Pro 5 article.

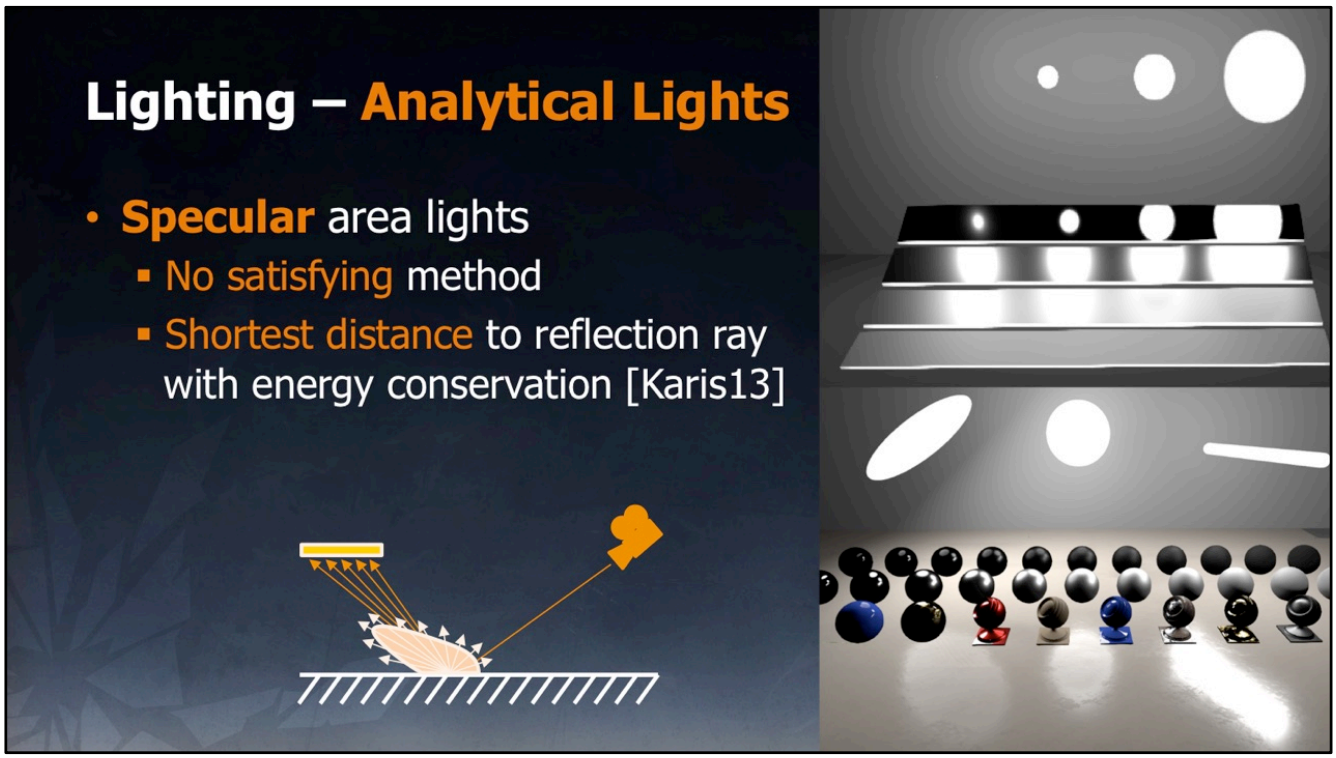
The third is something we call ‘structured sampling of the light shape’, which has been developed internally. The details and the code are in the course notes.



Here are some in-engine results.

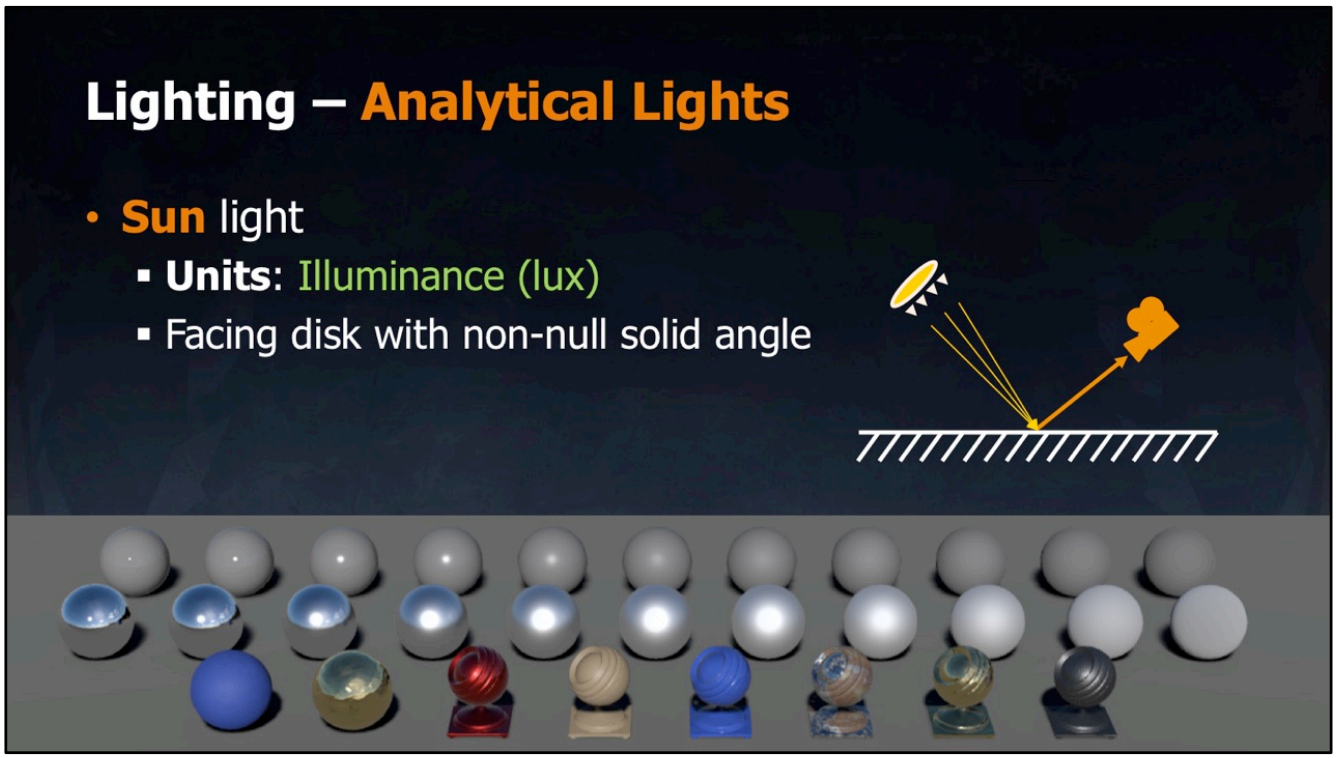
We use either the MRP or the light position for the Disney diffuse term evaluation, depending on the light type.

(Note: these screenshots are with Lambert.)



For the specular area lighting, we haven't found anything satisfying. We have experimented with our own method, as well as the MRP from Michał Drobot, and the 'shortest distance to reflection ray' approach presented by Brian Karis last year. However, none of them stand up to comparison with the reference.

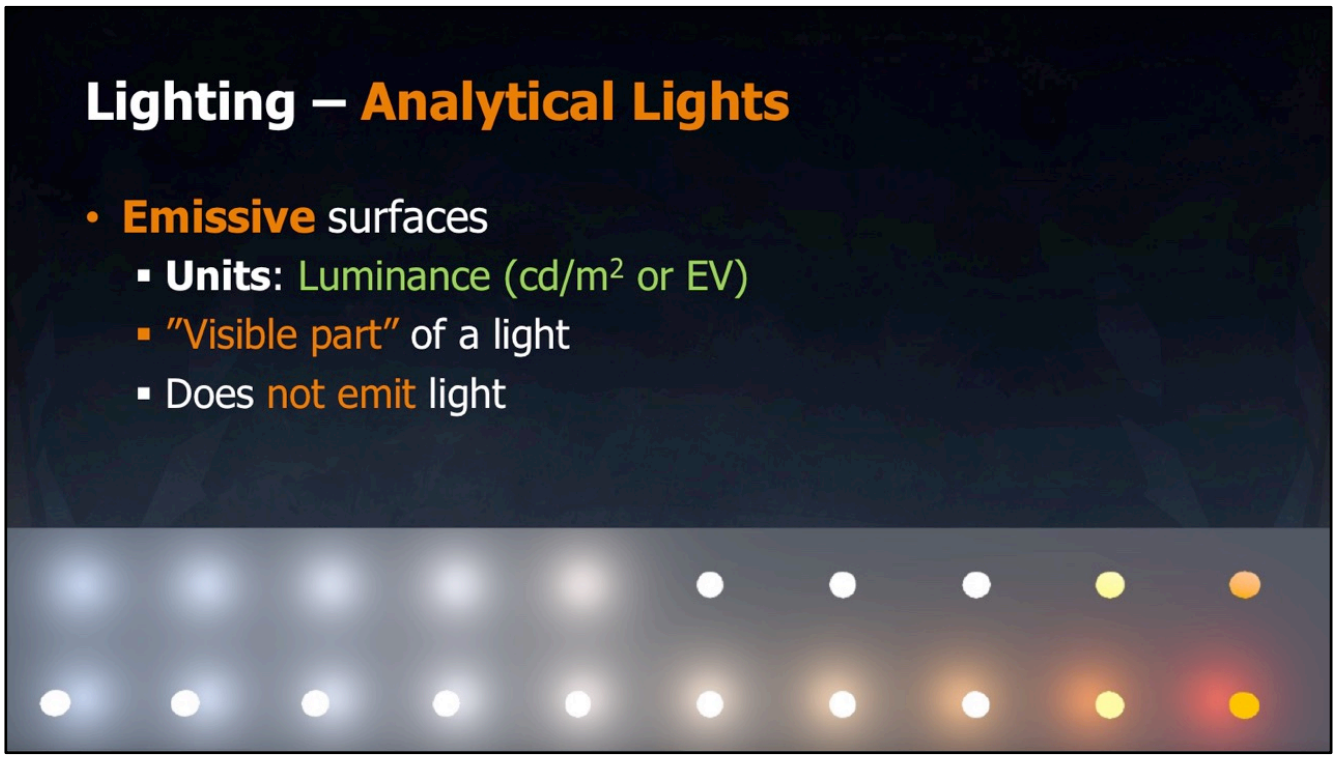
We have chosen to use the approach presented by Karis because it is fast and still has good-looking results. However, we have not found a good energy-conserving term for disk and rectangle area lights.



For the sun, the diffuse term is simply approximated with a single direction.

However, for the specular term, we use an oriented disk approximation. This has two benefits: firstly, it produces the actual solar disk shape for smooth materials, and secondly it helps to reduce specular aliasing (versus to a directional light).

We have also made some real-world measurements to determine valid ranges for sun (in lux). We found it to be above 100,000.



Emissive surfaces are complementary to area lights in Frostbite. They represent the visible part of area light sources.

They don't emit light, but they use physical light units to ensure that their look matches the amount of light that is cast by the source.

Lighting – Image-Based Lights

- Types of IBLs
  - Distant light probe
  - Local light probes
  - Screen-space reflections
  - Planar reflections

Traditionally, 'IBL' has been used to refer to specular light probes (typically cube maps). However, we use the term to collectively refer to any form of image-based lighting.

This covers a few types:
- A distant light probe for parallax-free far lighting
- Local light probes, for local lighting, with parallax
- Screen space reflections, for close-range lighting (supporting glossy reflection)
- Planar reflections as an alternative to SSR

Lighting – Image-Based Lights

- Types of IBLs
  - Distant light probe
  - Local light probes
  - Screen-space reflections
  - Planar reflections

Focus

We will focus on the first two items in this presentation.

Lighting – Image-Based Lights

- **Units**: Luminance (cd/m^2 or EV)
- Source for the distant light probe
  - HDRI
  - Procedural sky

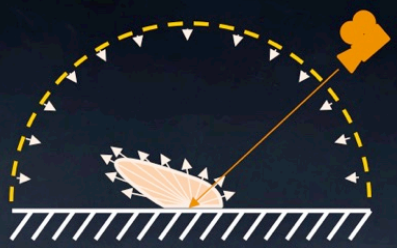
For distant light probes, we support two types of sources: the light can be an actual HDR image acquired by artists, or it can come from a procedural sky.

Be careful when using HDR images, because they often don't contain proper luminance values.

Most of the HDR images we've seen on the internet have bad ranges, and therefore don't combine well with other light types.

Lighting – Image-Based Lights

- Light probe lighting: Integral[Env. lighting x BRDF]



- Pre-integration by separating:
  - Integral of Lighting x NDF, for V = N
  - Integral of BRDF, for all V & roughness values

Specular [Karis13] & Diffuse

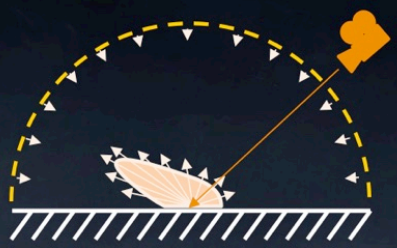
When dealing with light probes, we need to compute the integral of the product: lighting times BRDF.

This is a costly operation, so we rely on pre-integration and approximations.

(Click).

Lighting – Image-Based Lights

- Light probe lighting: Integral[Env. lighting x BRDF]



- Pre-integration by separating:

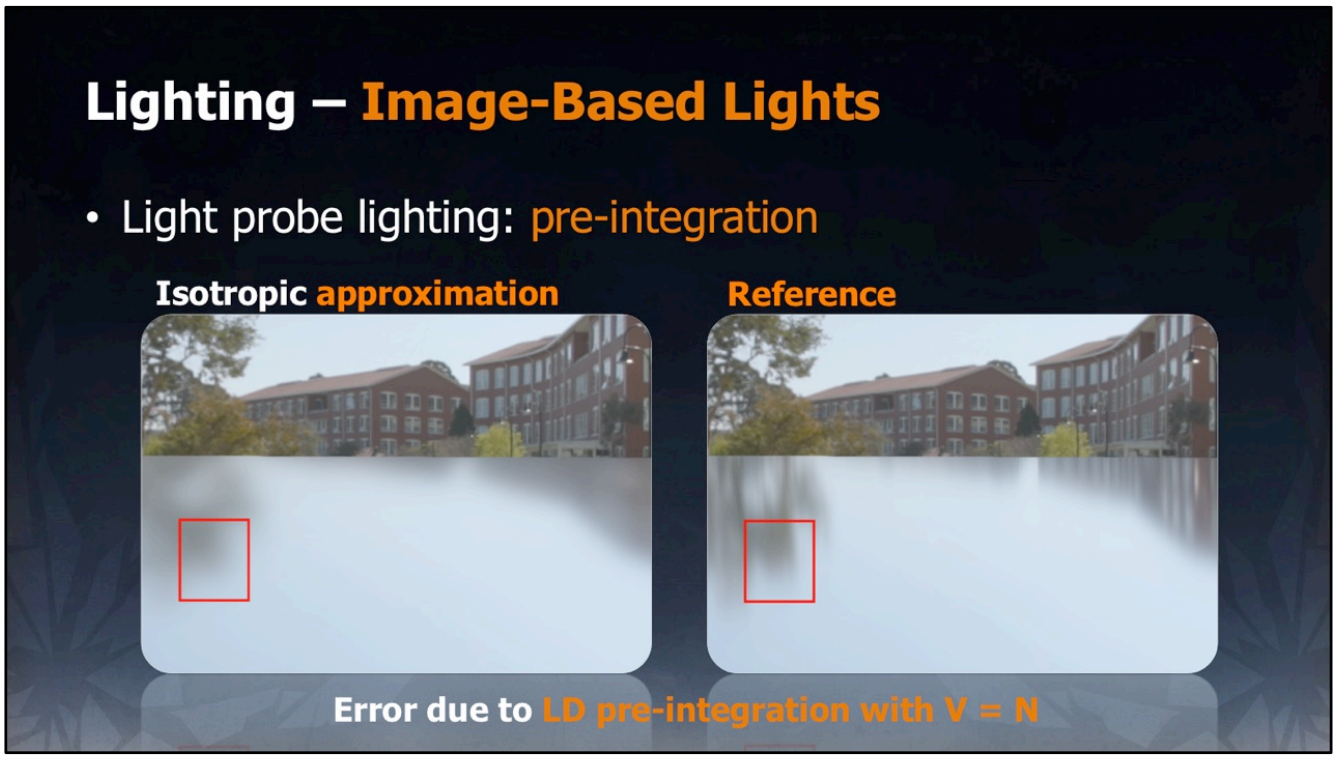
LD • Integral of Lighting x NDF, for V = N

DFG • Integral of BRDF, for all V & roughness values

Specular [Karis13] & Diffuse

We used the decomposition covered by Brian Karis last year, which decouples the pre-convolved lighting (LD) from the BRDF (DFG).

We have also used the same decomposition for the Disney diffuse model.



A remark here: the separation makes some significant approximations. The biggest of these is in the LD pre-integration, where the view vector is assumed to be aligned with the normal vector.
This results in a loss of anisotropy at grazing angles, which is quite noticeable on flat surfaces.

This is something we would like to address in the future.

Lighting – Image-Based Lights

- Light probe lighting: pre-integration
  - LD needs to be computed each time the lighting changes
  - Needs to be fast (real-time capture / refresh)
  - Deals with HDR source
- Integration method for **LD**
  - Importance sampling
  - Multiple importance sampling
  - Filtered importance sampling

The LD pre-integration can be baked offline, but in our case, we need to also bake it at runtime as we support dynamic light probes.
This means that the computation needs to be fast, but it still needs to be robust enough to deal with high-contrast (HDR) illumination.

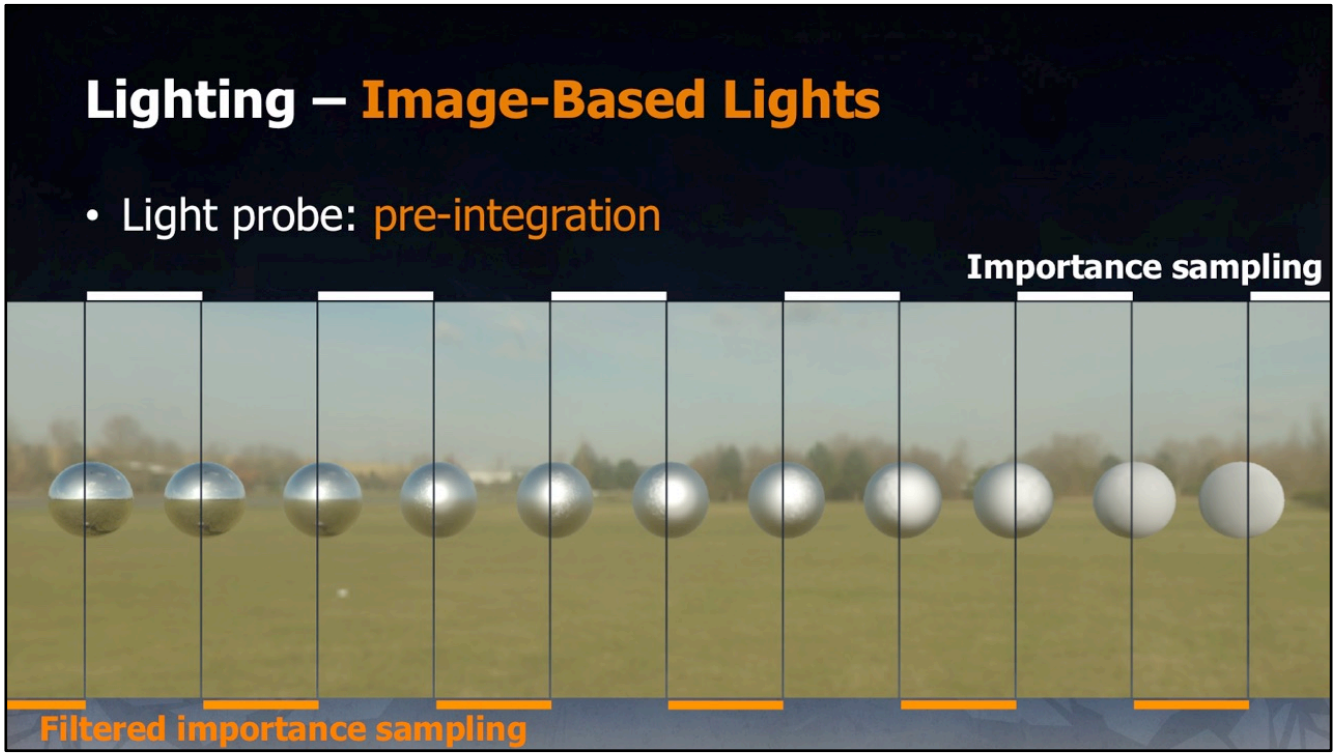
For this we tried importance sampling, but it was not enough, since importance sampling just the BRDF – not the lighting – can lead to noise in the case of high-contrast sources. We tried multiple importance sampling (MIS) to take account of both the BRDF and the lighting, but even if the convergence was faster, the shader cost was higher.

(Click)

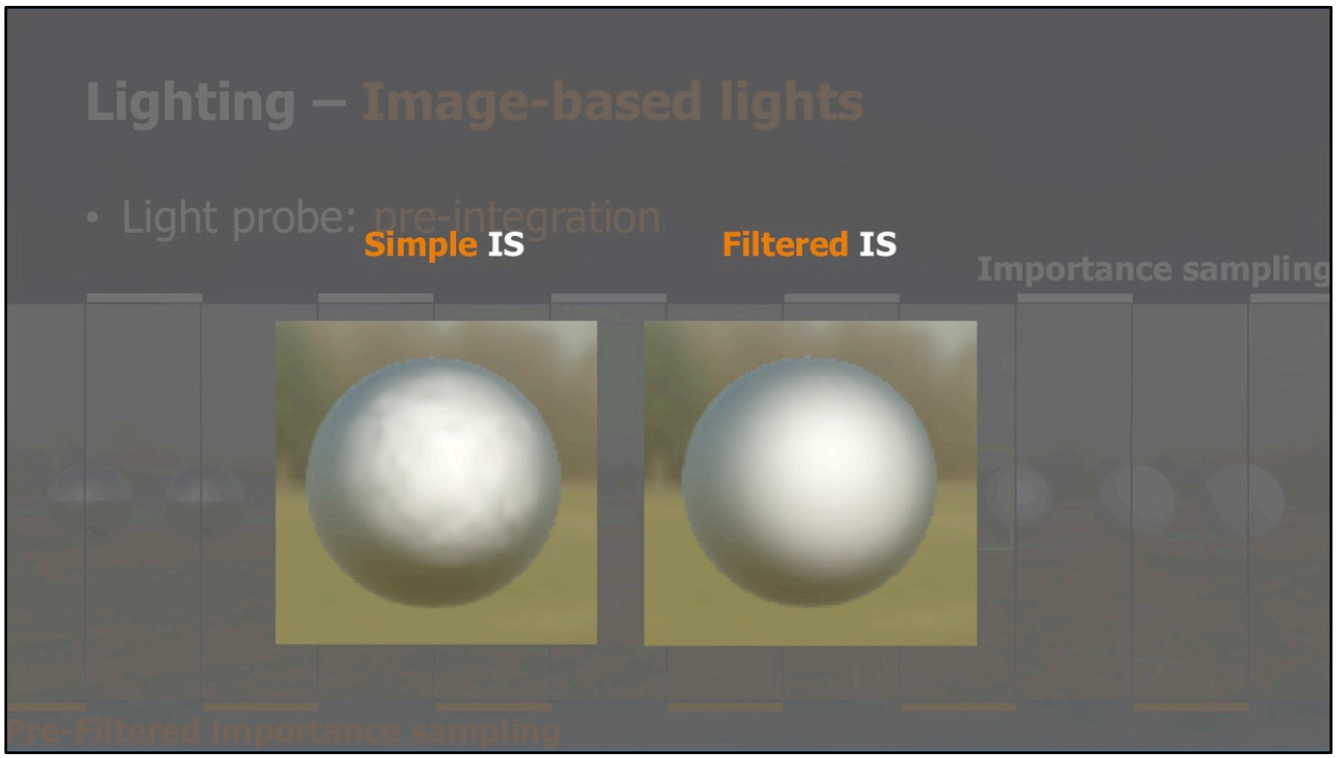
Lighting – Image-Based Lights

- Light probe lighting: pre-integration
  - LD needs to be computed each time the lighting changes
  - Needs to be fast (real-time capture / refresh)
  - Deals with HDR source
- Integration method for **LD**
  - Importance sampling
  - Multiple importance sampling
  - Filtered importance sampling **Faster convergence**

Instead we use Filtered Importance Sampling (FIS), which employs mipmapped versions of the lighting in regions of lower sampling density. This introduces some bias, but it gives us faster convergence.



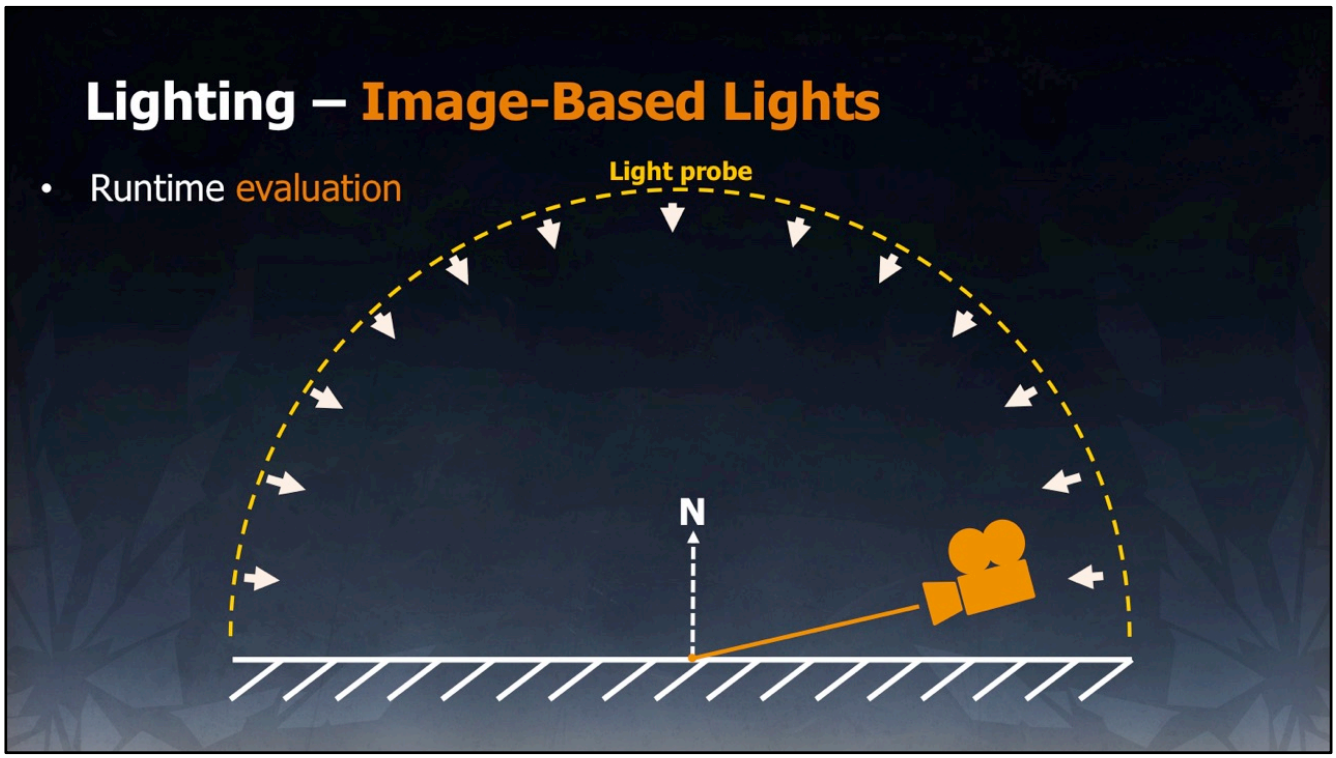
Here is a comparison of the IS and the filtered approach.



And close up.

In practice we use 32 samples.

We also don’t process the first MIP of the LD texture, in order to get a perfect specular mirror integration.

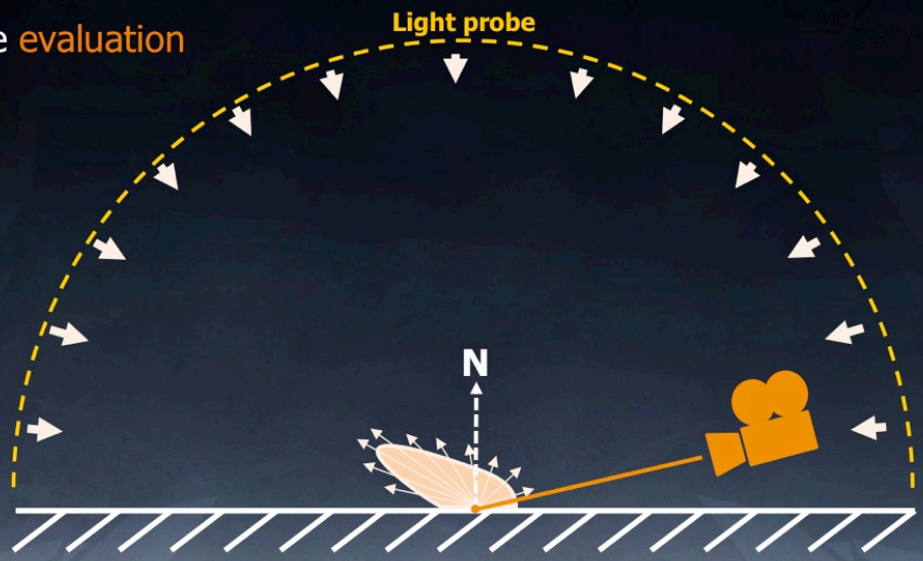


At runtime, to evaluate the light probe for a view direction.

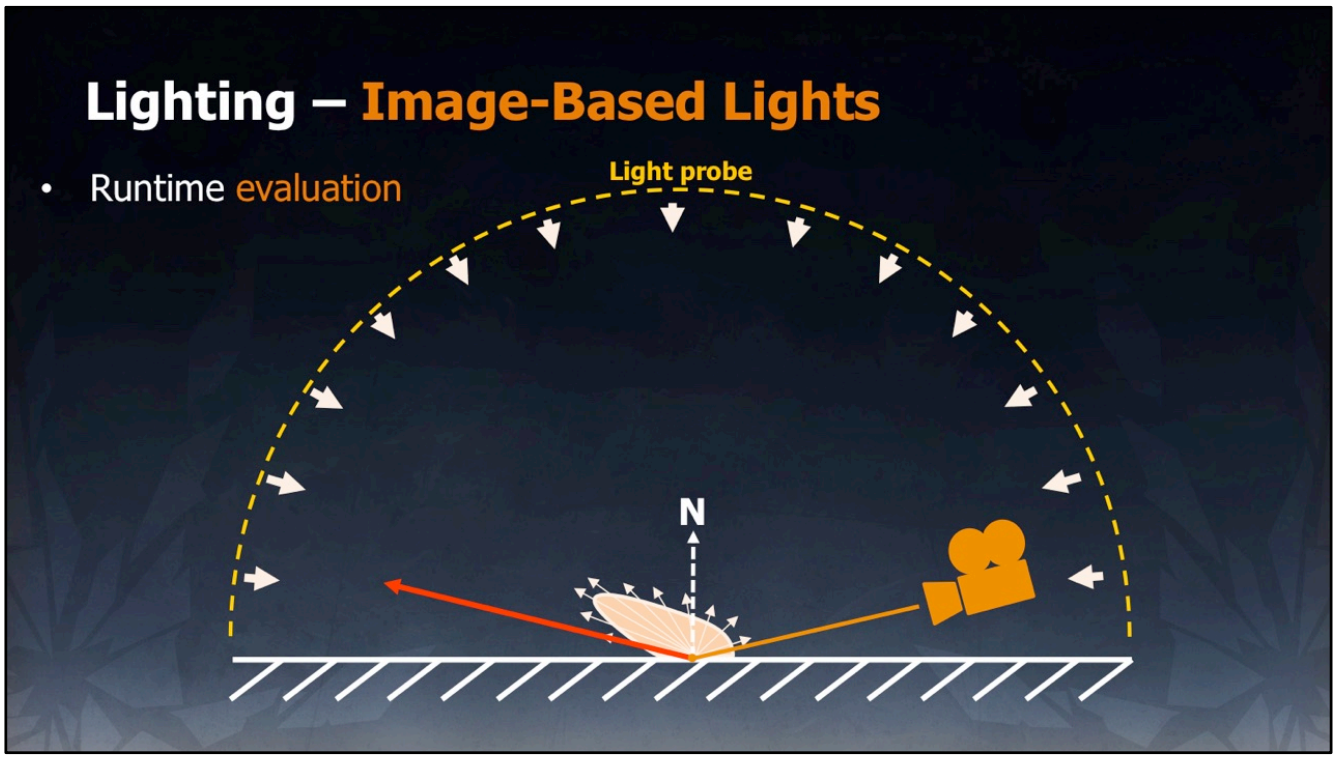
Given a surface...

Lighting – Image-Based Lights

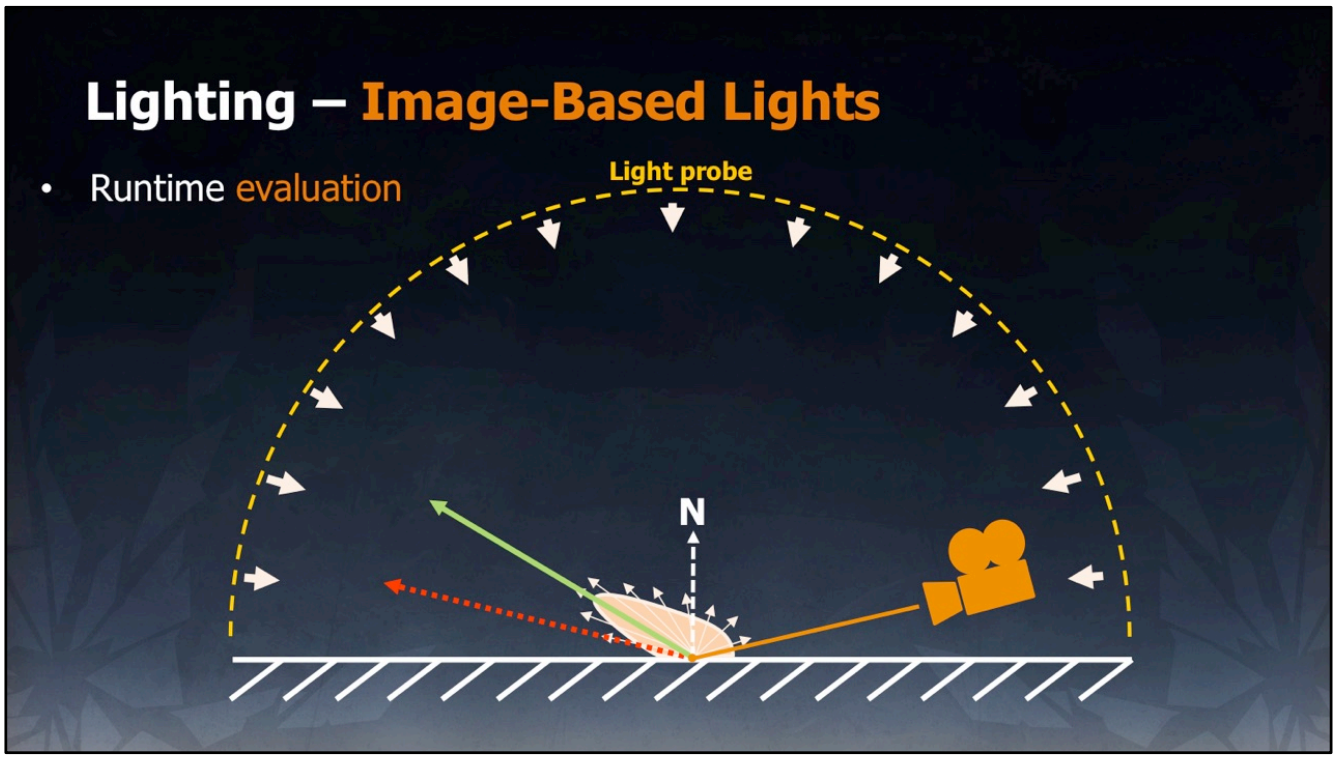
- Runtime evaluation



…we evaluate a material.



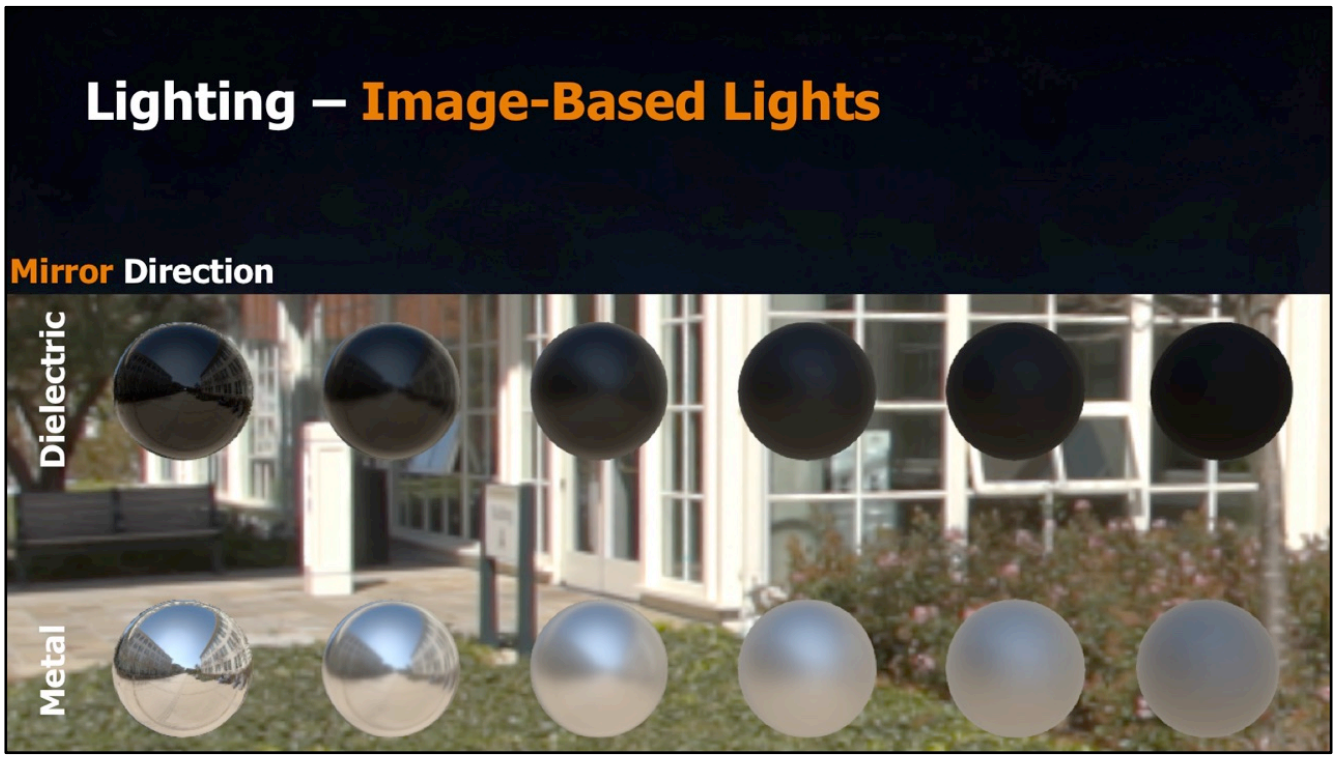
…and we generally use the mirror direction to fetch the lighting from the cube map…



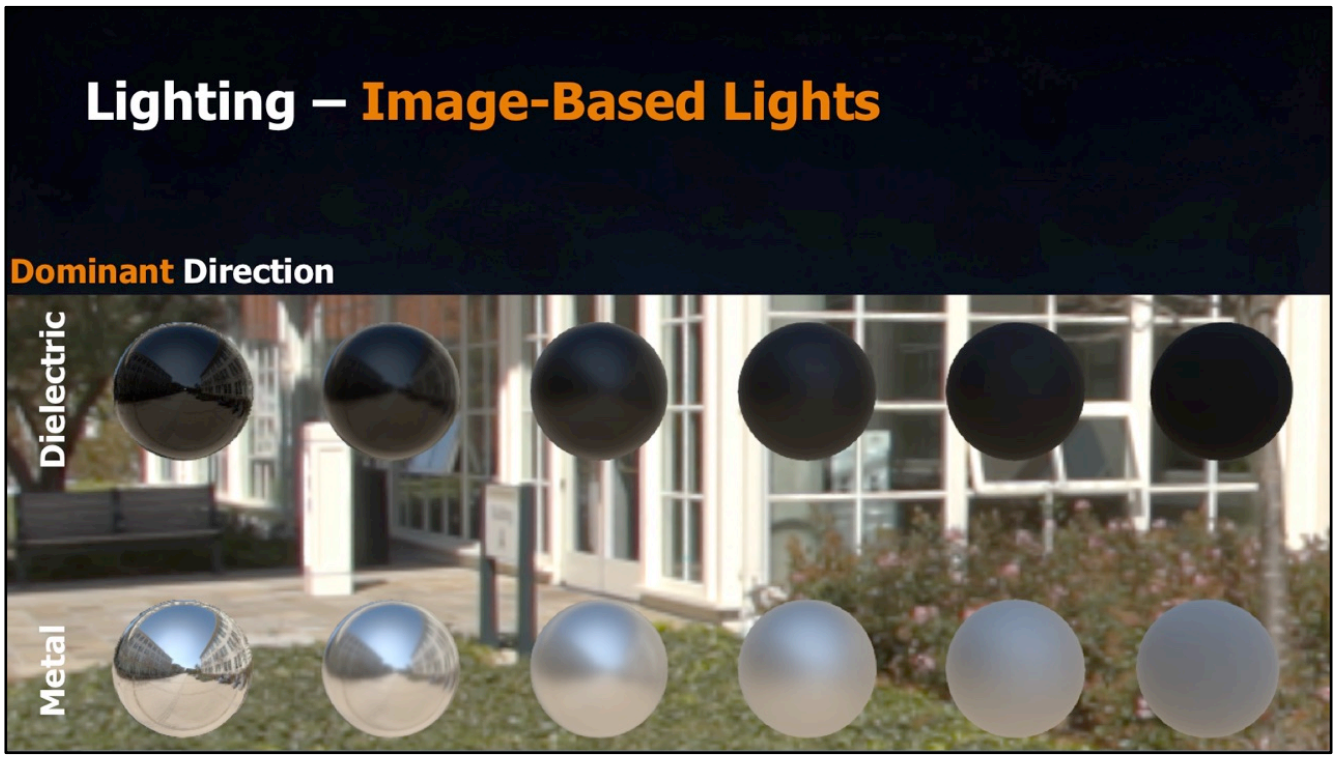
…but the dominant direction of the BRDF is slightly shifted compared to the mirror direction.

Using this dominant direction instead of the mirror direction helps to improve the accuracy of the integration approximation.

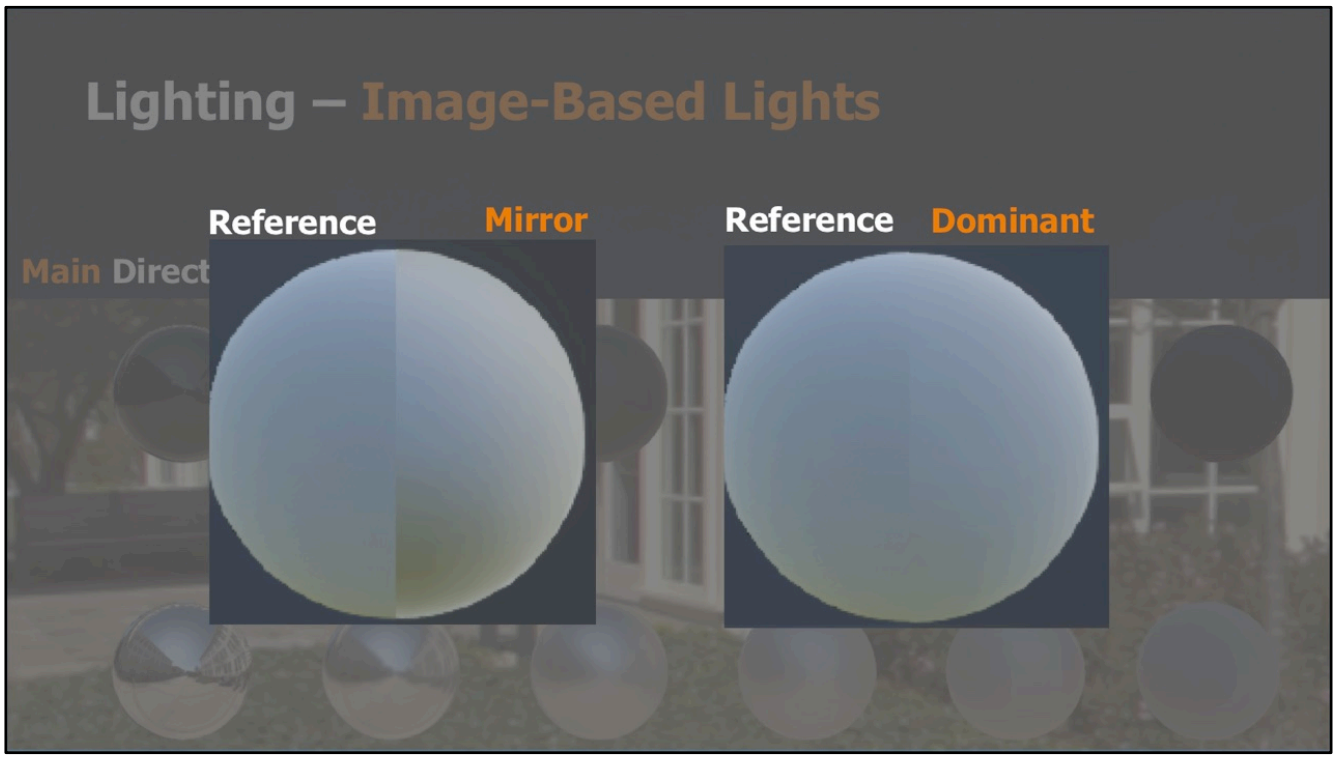
We provide a formula to convert the mirror direction to the dominant direction in the course notes.



Here is an example.



The improvement for rough materials is quite noticeable.

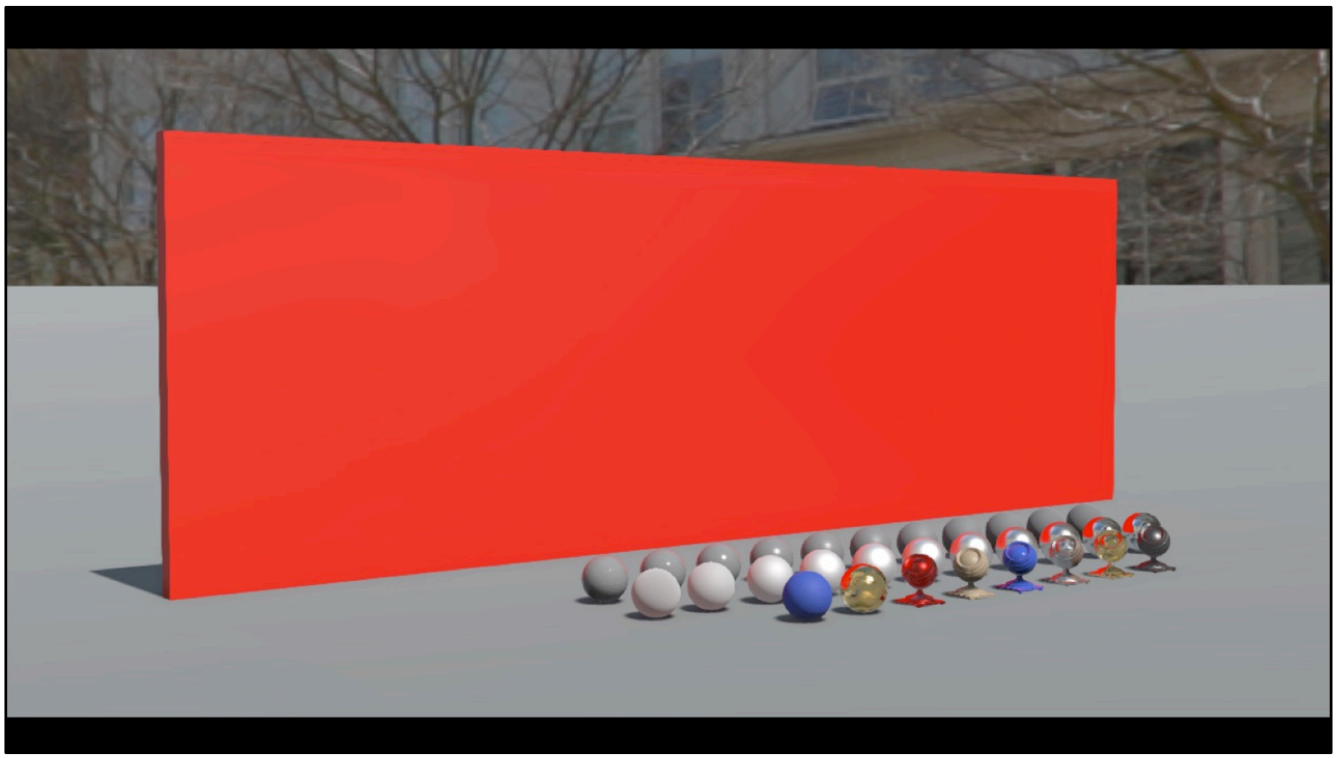


In this close-up, we can see that using the dominant direction is a better approximation when compared to the reference.

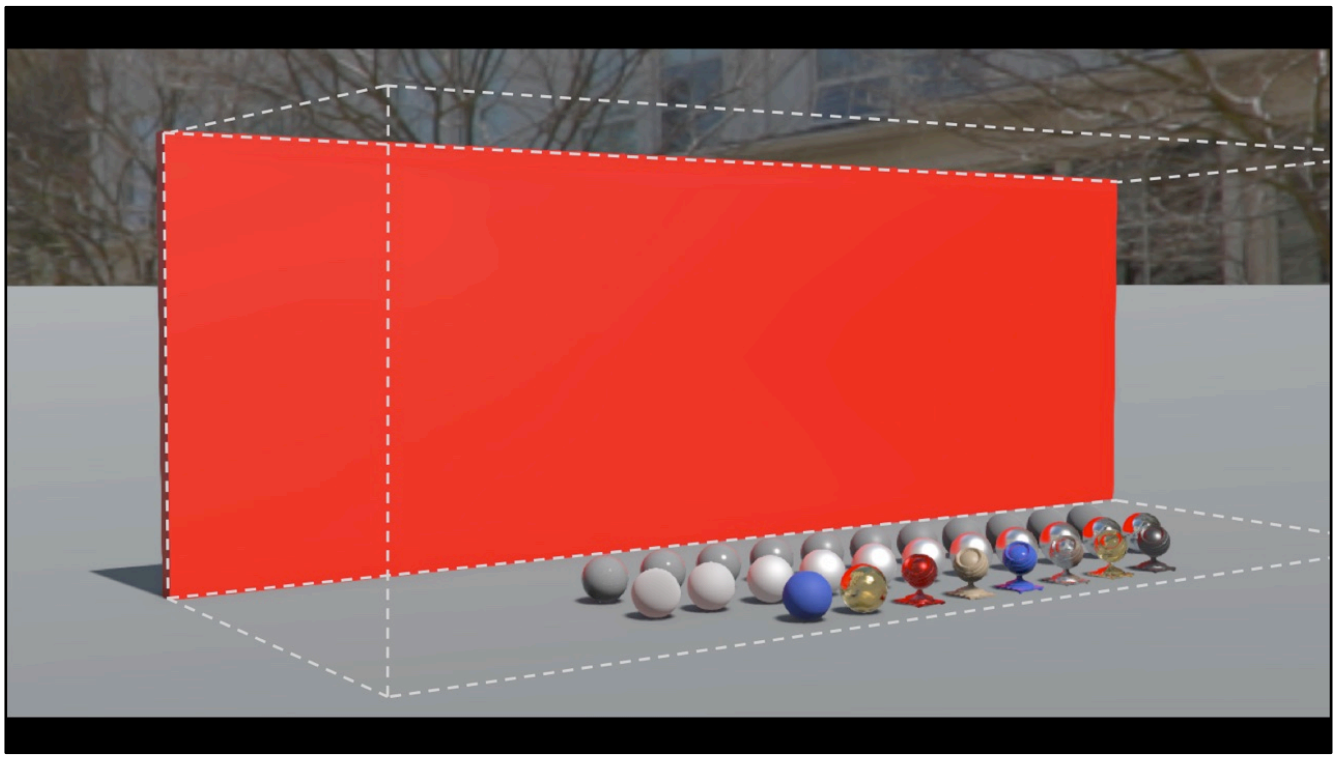
Lighting – Image-Based Lights

- Local light probes
  - Acquire surrounding geometry
  - Approximate local parallax: box & sphere proxy [Lagarde12]

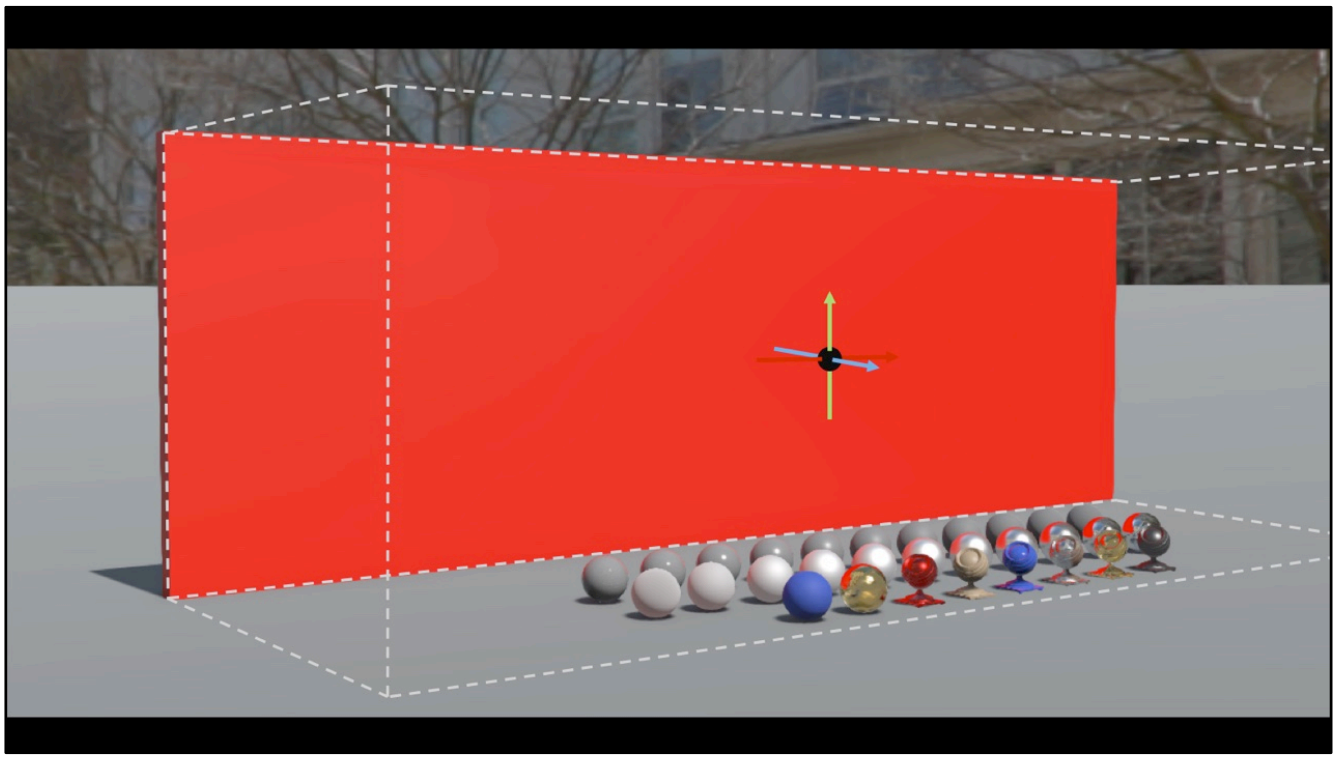
The local light probes capture the surrounding environment and use parallax correction via box and sphere proxies.



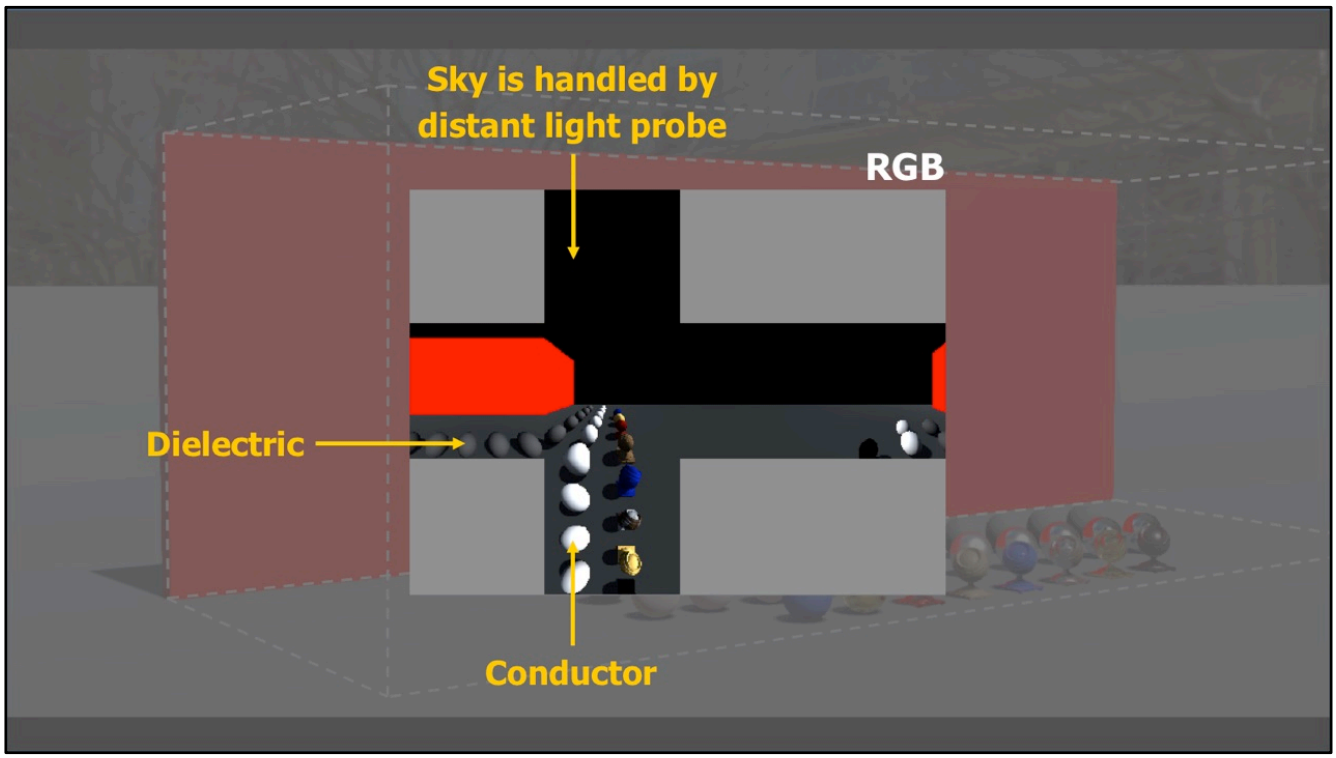
So we have a scene…



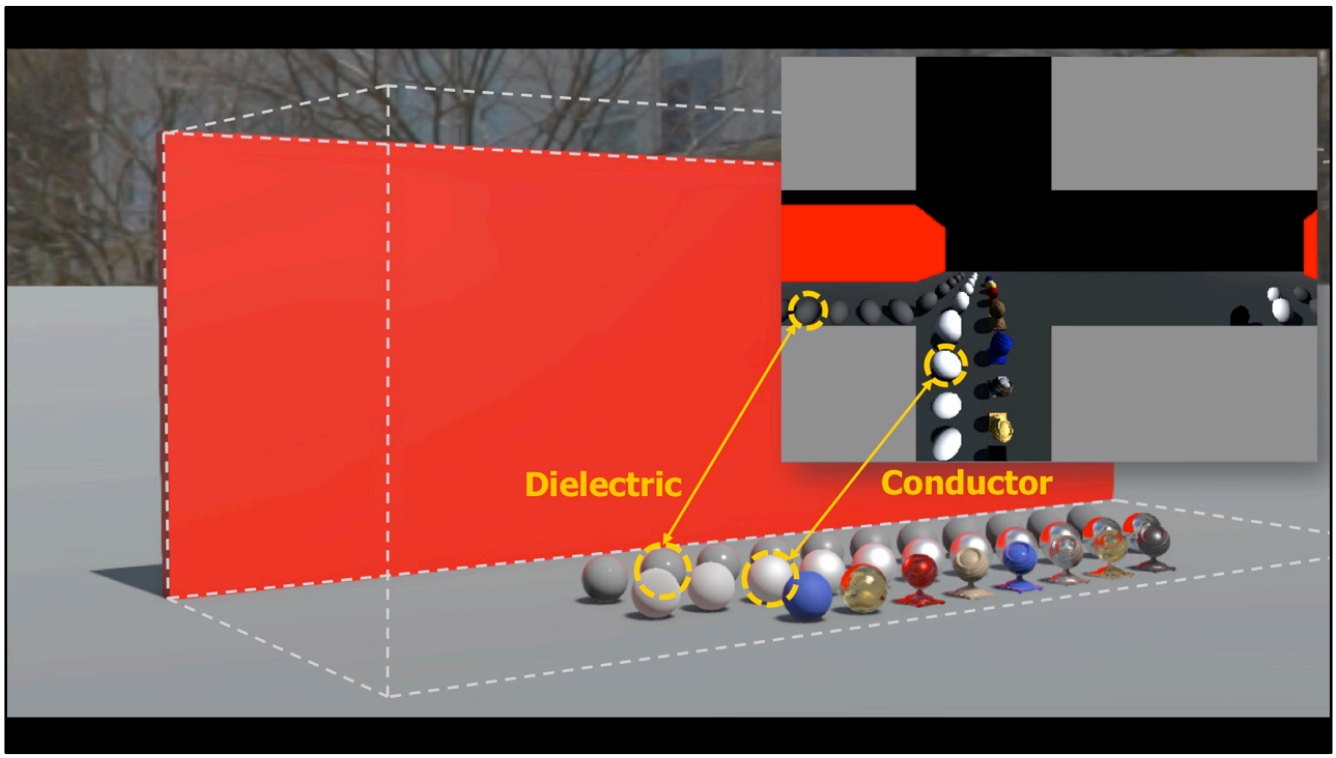
…we place proxy geometry, which approximates the surrounding geometry…



…then we acquire the incident lighting at the center of this proxy…



…which provides us with incident lighting stored in a cube map that we can reproject onto the proxy.

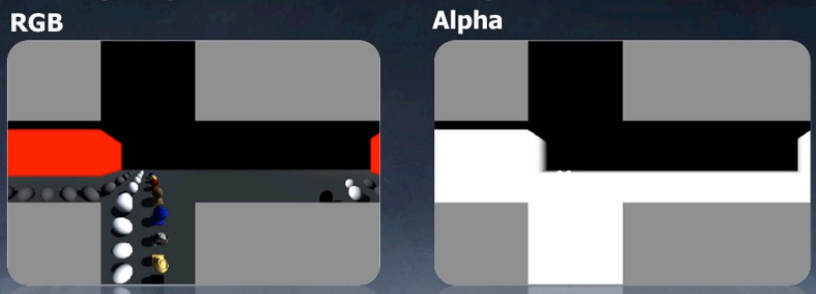


The incident capture contains only the diffuse component, ignoring any view-dependent effects such as specular reflections.

For conductors, as they don’t have any diffuse term, we use the f0 term as diffuse color. This ensures that we have some sort of surface response in dynamic cube maps, where recursive rendering of reflections would be prohibitively expensive.

Lighting – Image-Based Lights

- Distant & local light probes composition
  - Lots of local light probes across level
  - Local light probes can overlap each other
  - Distant light probe contains sky information



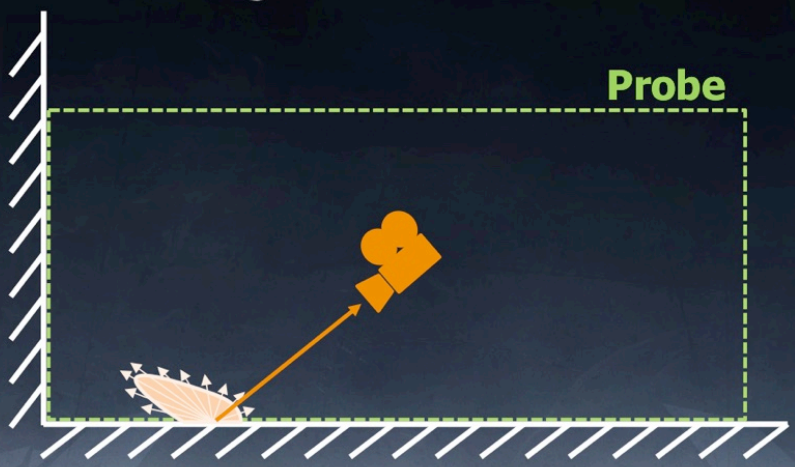
These local light probes are used all over the game world and can be placed in a nested fashion.

When we capture the incident lighting, we also capture the visibility of the surrounding objects and store it in the alpha channel of the cube map.
That's why the sky is black in our cube map.

By using this visibility information, we are able to fall back to the distant light probe when no object occludes it.
The distant light probe contains the sky, and it can be of higher resolution and even contain animated clouds.

Lighting – Image-Based Lights

- Distance-based roughness

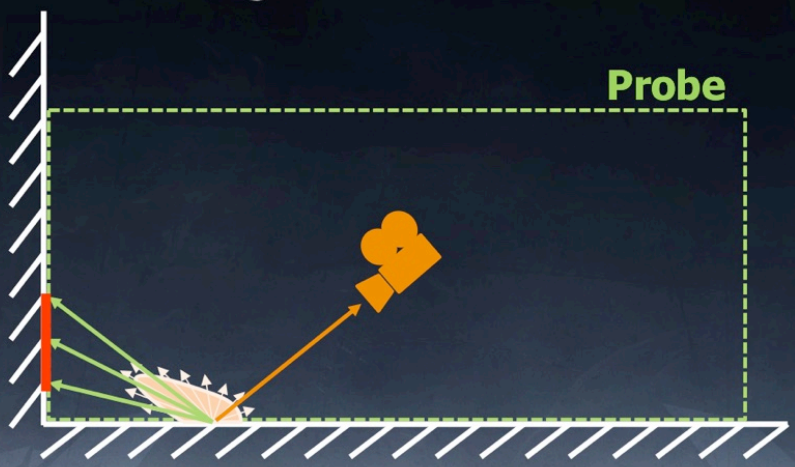


An extra piece of information for local light probes…

When we evaluate the local light probe, we must take into account the distance of the proxy geometry from the shaded point.

Lighting – Image-Based Lights

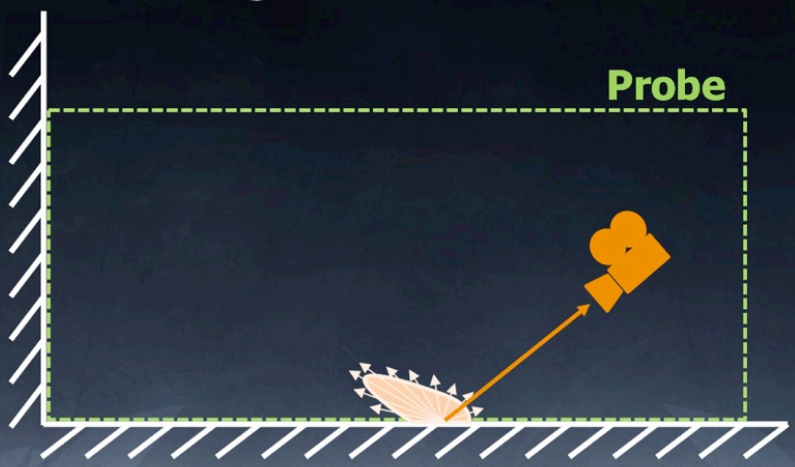
- Distance-based roughness



The BRDF can be represented by a cone which reflects a small portion of the wall.

Lighting – Image-Based Lights

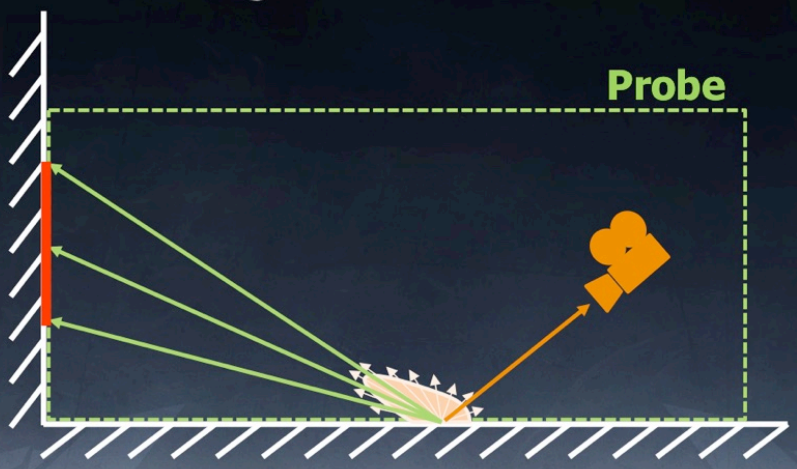
- Distance-based roughness



But for a distance a bit further away…

Lighting – Image-Based Lights

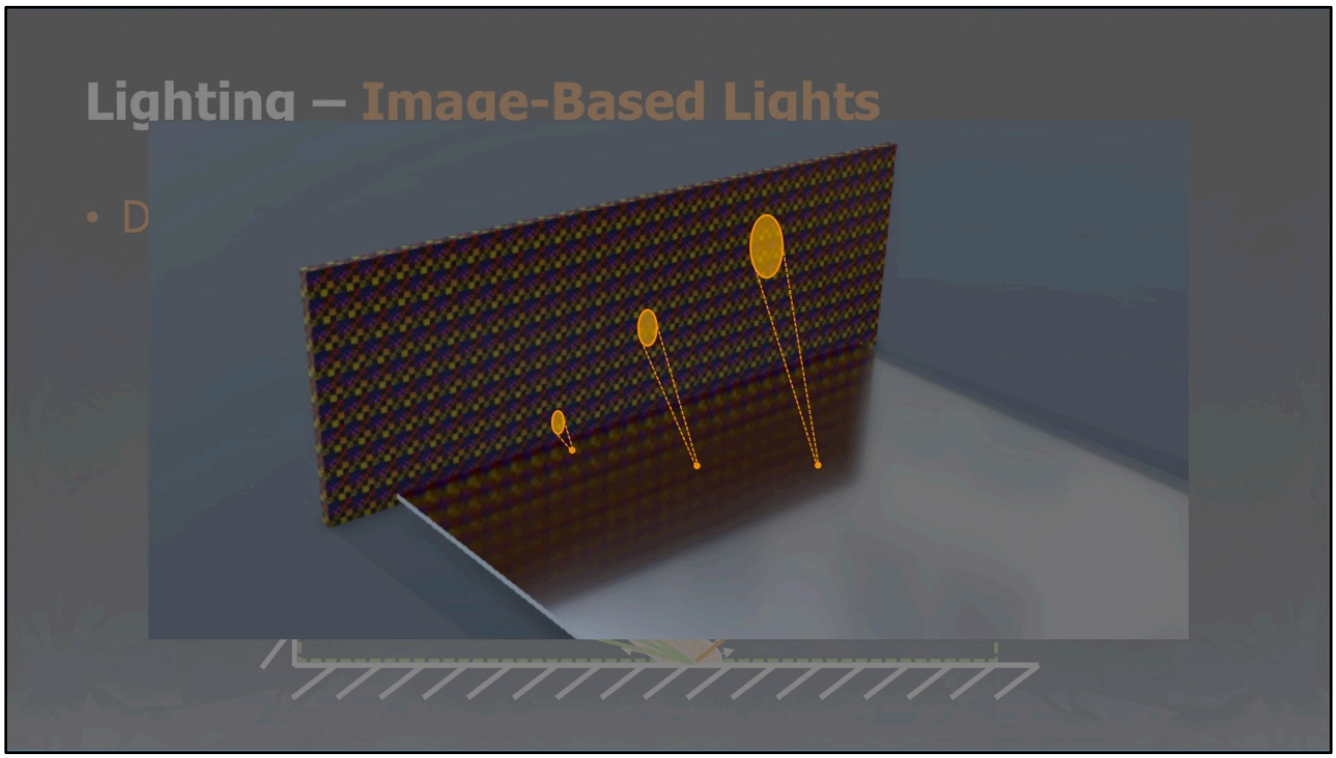
- Distance-based roughness



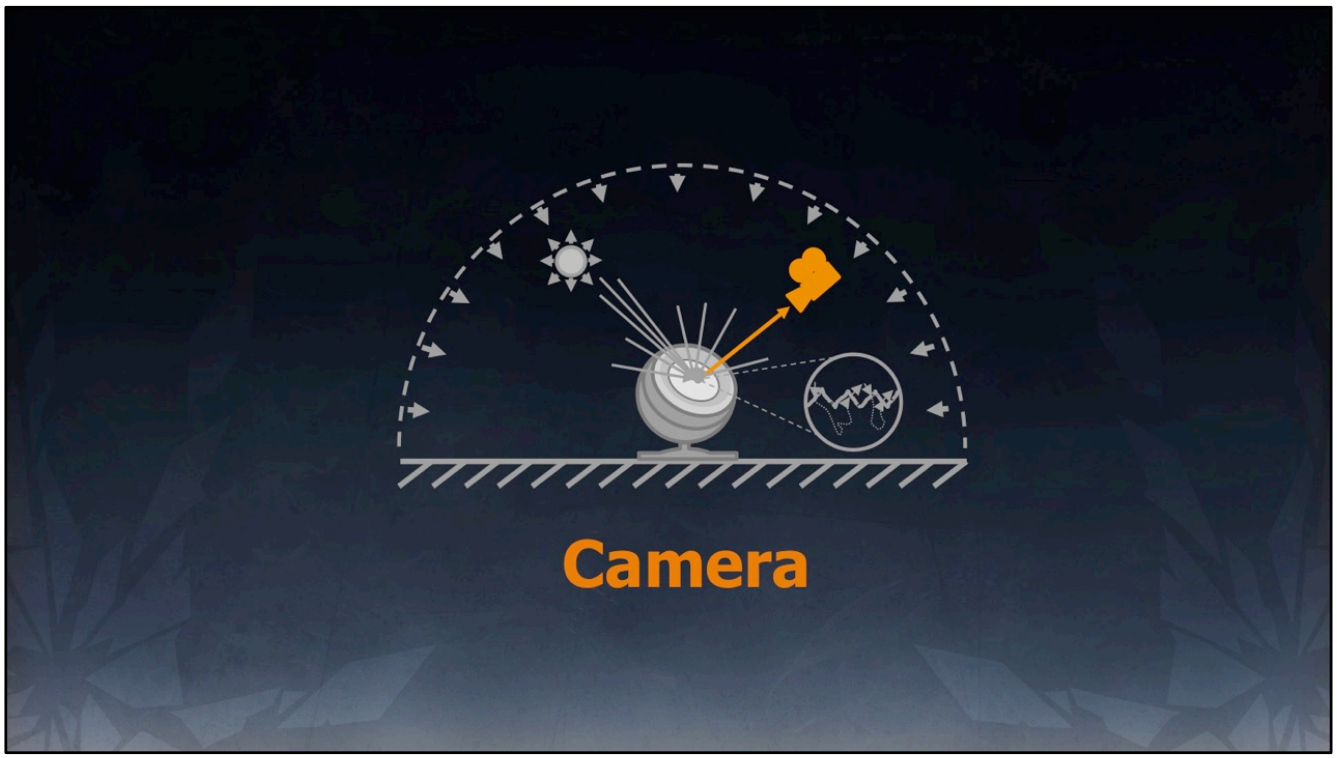
The cone will span a larger portion of the wall.

To take this phenomenon into account, we adjust the roughness. This is what we call distance-based roughness.

Taking this into account will help to transition from SSR to local light probes.



Here’s an example showing distance-based roughness in action.

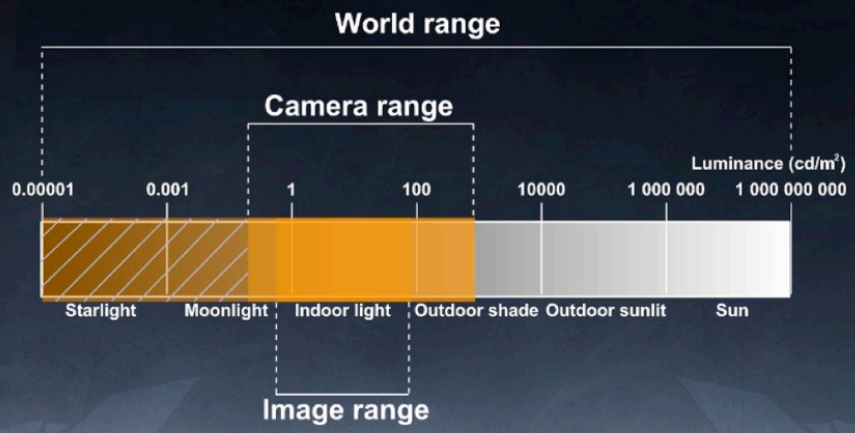


This concludes the lighting section.

Finally, the camera part.

Camera – Physically Based Camera

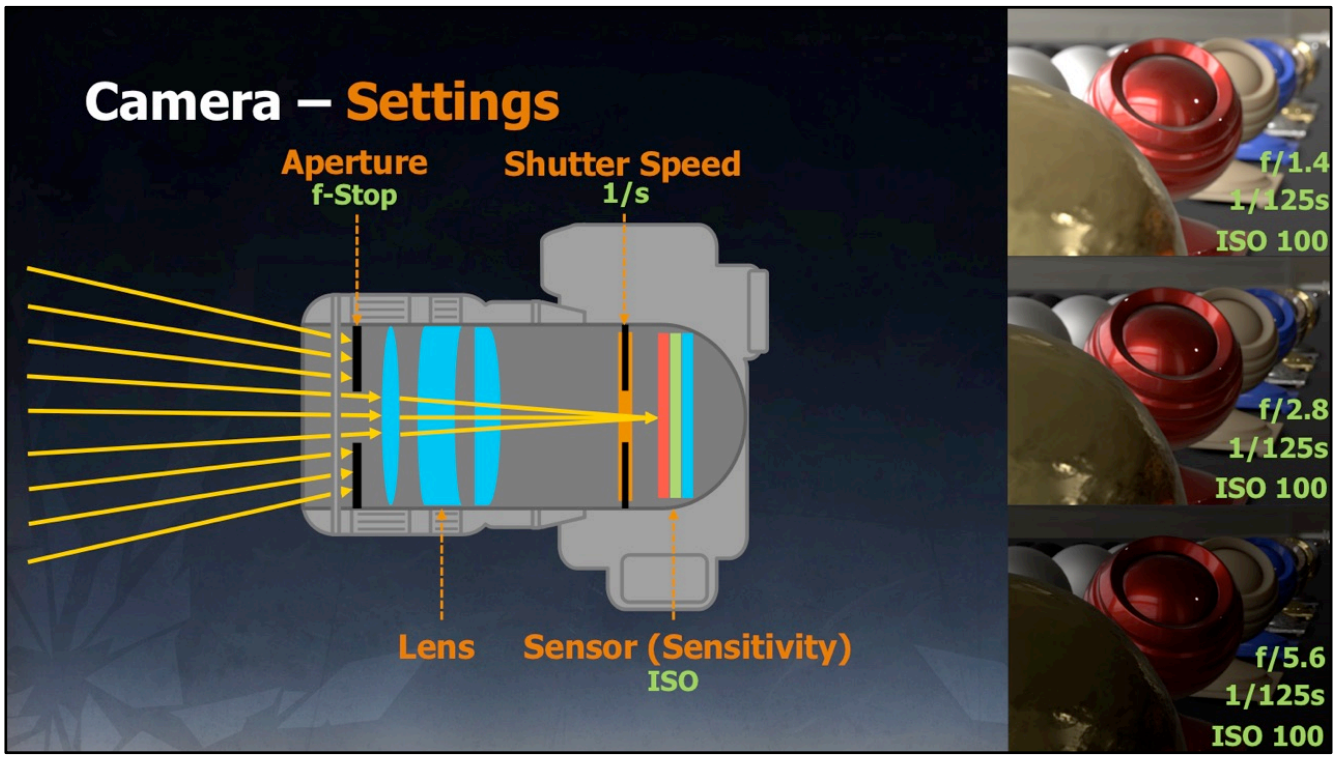
- Transforming scene luminance to pixel value



As mentioned earlier, all of the lighting computations in Frostbite are made using real-world luminance values.

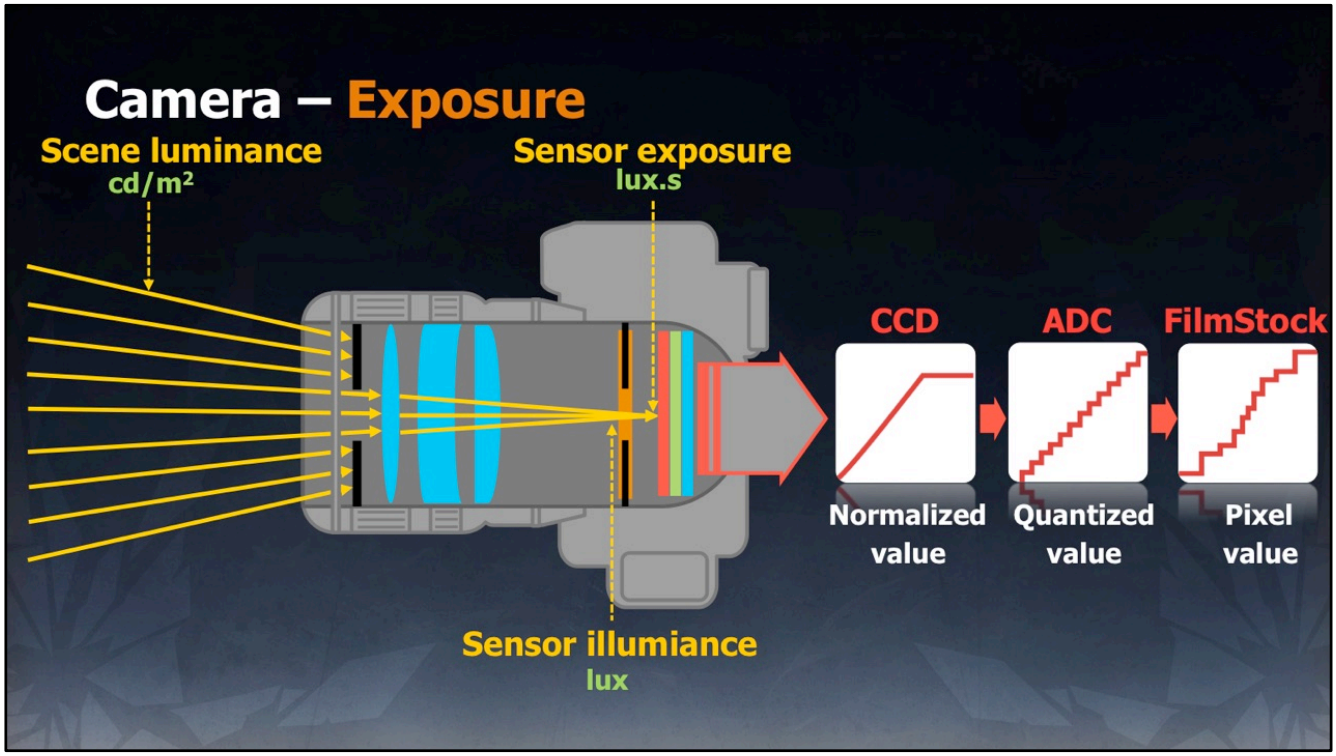
We need to transform this incident luminance into a pixel value.

It's not possible to capture the entire range of incident luminance due to its extremely high dynamic range, so the camera captures only a small section by 'exposing' the scene…

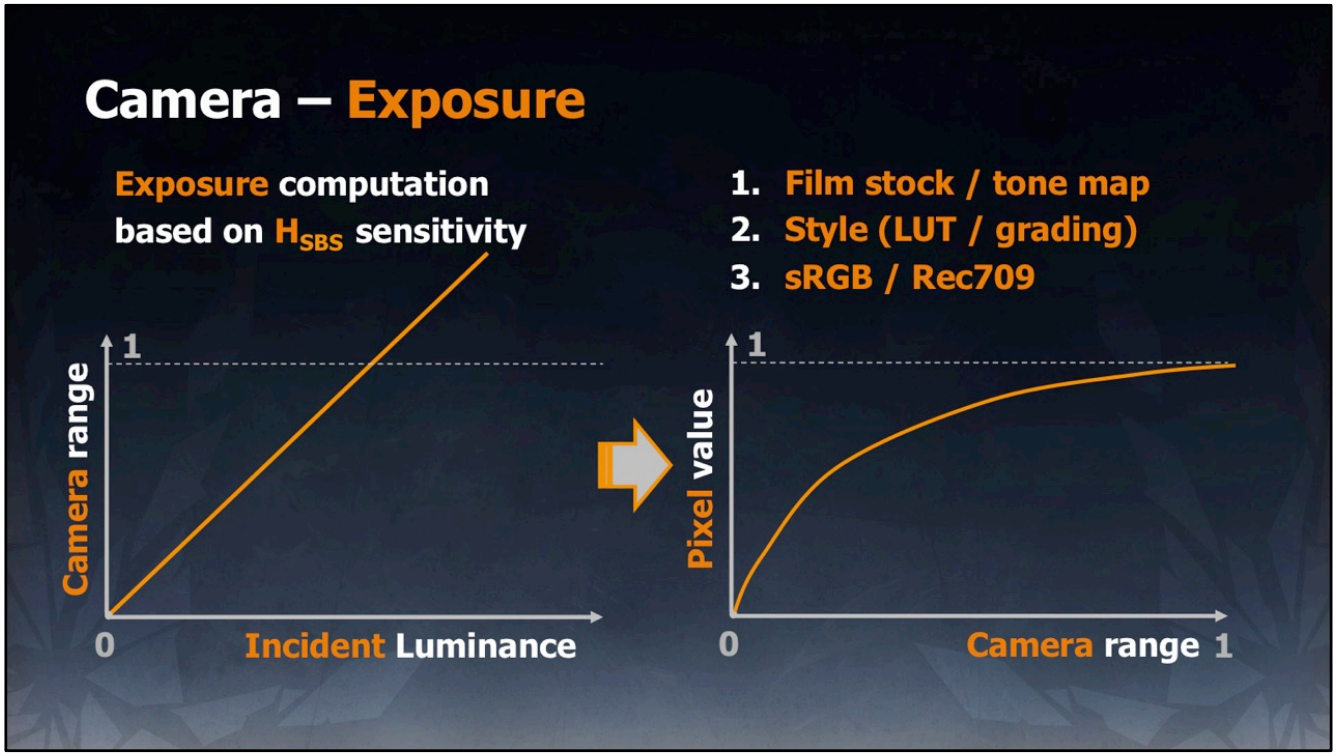


The exposure is controlled by three settings manipulated by the artists: the aperture, the shutter speed, and the sensitivity.

These settings can be automated depending on the art/game requirements.

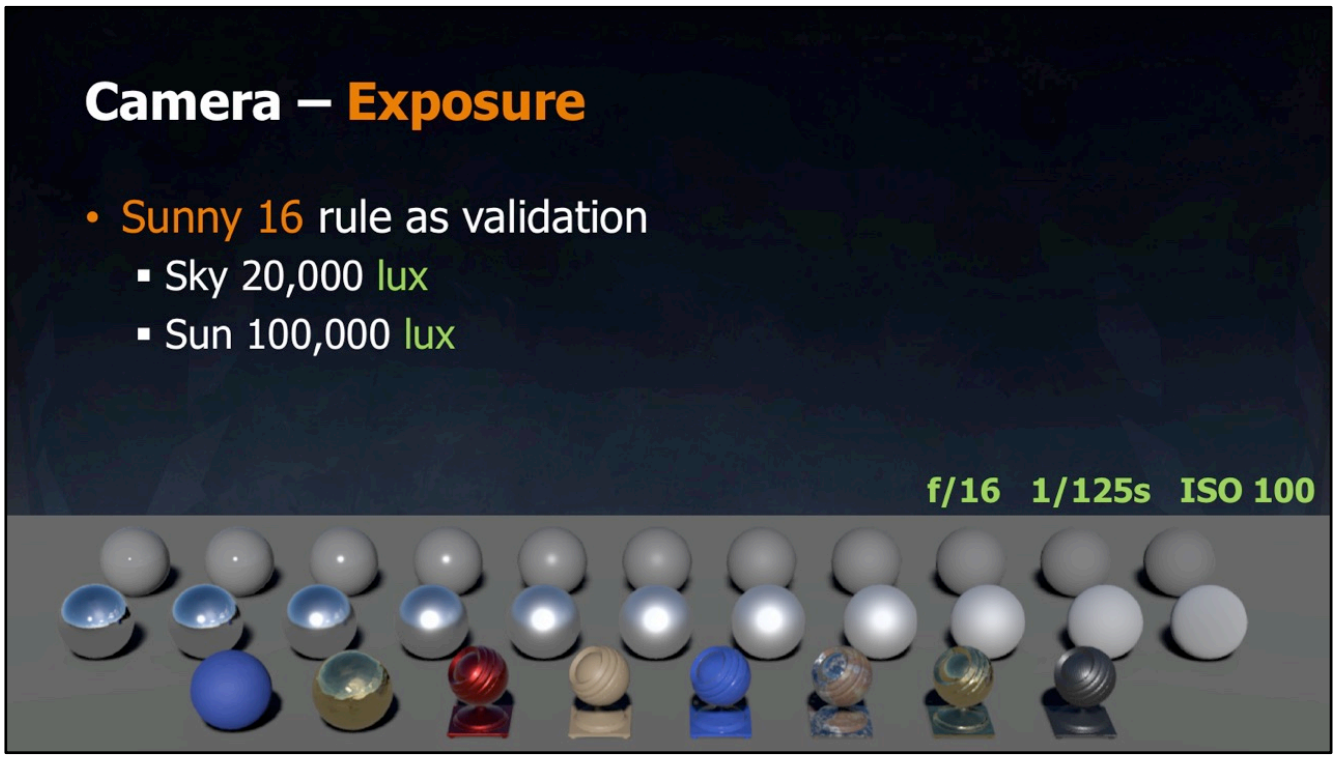


The amount of scene luminance reaching the sensor will be determined by the exposure, which is then converted into a pixel value.



The conversion from the exposed value to the pixel value uses the common color pipeline, with the usual tonemapping, color grading, gamma correction, etc.

The exposure computation relies on a physically based approach. We are using the standard named Saturation-Based Sensitivity.
It defines a maximum luminance based on the camera settings and divides the scene luminance by this maximum value. Again, details and code are provide in the course notes.

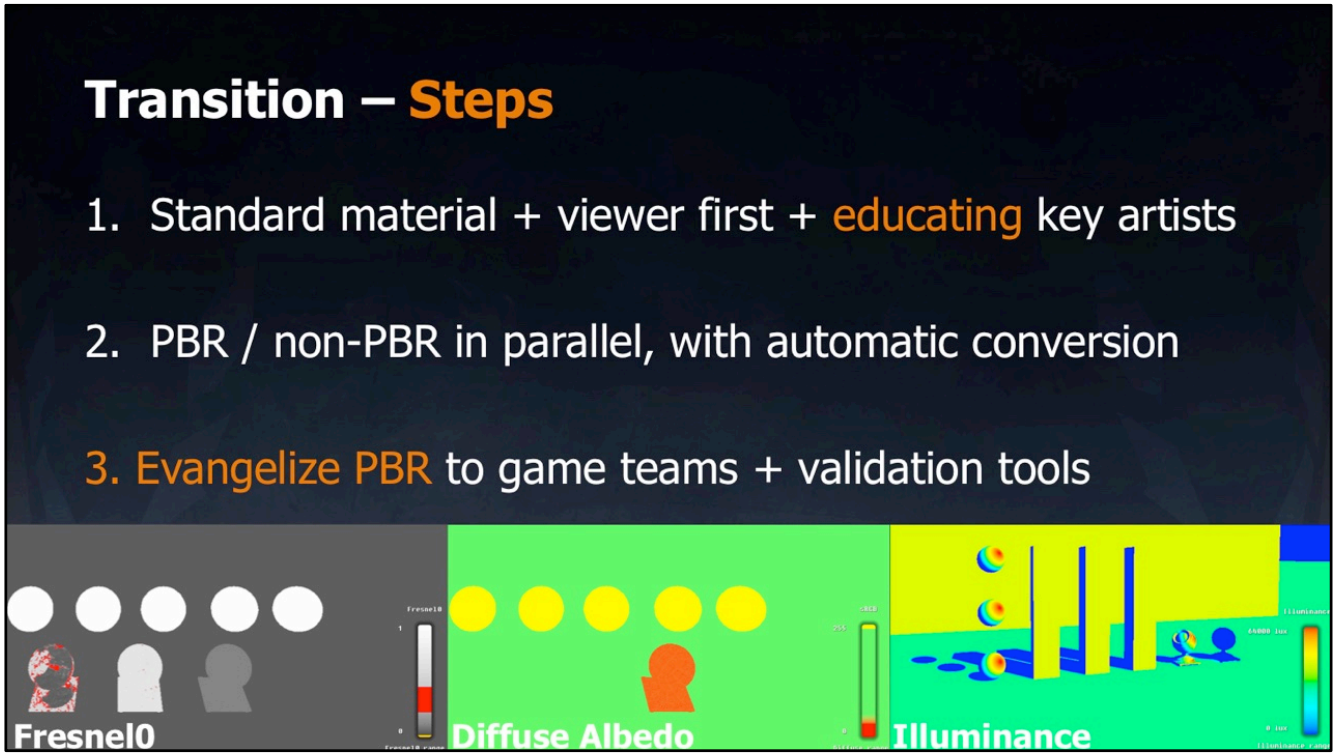


To validate that we correctly implemented our luminance-to-pixel conversion and that our sun values are in a good range, we employed the ‘Sunny 16’ rule used by photographers.

We set the camera and sun parameters with the expected values of Sunny 16 and checked that we got a well-exposed image.

After all of this technical stuff, we'd like to conclude with some practical advice, drawing from our recent experience.

We have taken a few steps to try to ensure a smooth transition to PBR at EA Frostbite…



The first step was to code our standard material and to develop a viewer with a distant light probe, since there were no tools such as Marmoset available at the time.

We then gathered a group of key technical artists from various game teams and started to train them. Education was done in parallel to other PBR feature development.

Once we achieved a minimal set of PBR features, we introduced them into the production engine and we maintained a PBR and a non-PBR version side-by-side.

We set up an automatic conversion system for textures and lights, to be able to load existing levels. This wasn't a high-quality conversion – particularly for the lighting – but it was sufficient for our needs as converted assets were not expected to ship.

Finally, we evangelized PBR to our game teams with the help of our freshly trained technical artists. We also provided a set of validation tools and ported our shaders to various authoring tools, such as Substance and Mari.

Acknowledgements

- EA Frostbite - Alex Fry, Christian Bense, Noah Klabunde, Henrik Fernlund, the rendering team
- EA DICE - Yasin Uludag, Arne Schober
- Lucasfilm: Lutz Latta, Cliff Ramshaw, Rodney Huff, Rogers Cordes
- Graphics community: Michał Drobot, Benjamin Rouveyrol, Eric Heitz, Juan Cañada, Ondra Karlík, Tomasz Stachowiak, Brian Karis
- Stephen Hill & Stephen McAuley

We would like to thanks all of the people who made this presentation possible.

QUESTIONS?

Sébastien Lagarde
Lagardese@hotmail.fr
Twitter: @seblagarde

Charles de Rousiers
Charles.derousiers@frostbite.com
Twitter: @kiwaiii

References

- [Burley12] Brent Burley, "Physically Based Shading at Disney", SIGGRAPH'12, PBR Course
- [Karis13] Brian Karis, "Real Shading in Unreal Engine 4", SIGRRAPH'13, PBR Course
- [Drobot14] Michal Drobot, "Physically Based Area Lights", GPU Pro 5
- [Heitz14] Eric Heitz, "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs", JCGT, 2014
- [Lagarde12] Sébastien Lagarde, "Local Image-based Lighting With Parallax-Corrected Cubemaps", SIGGRAPH'12