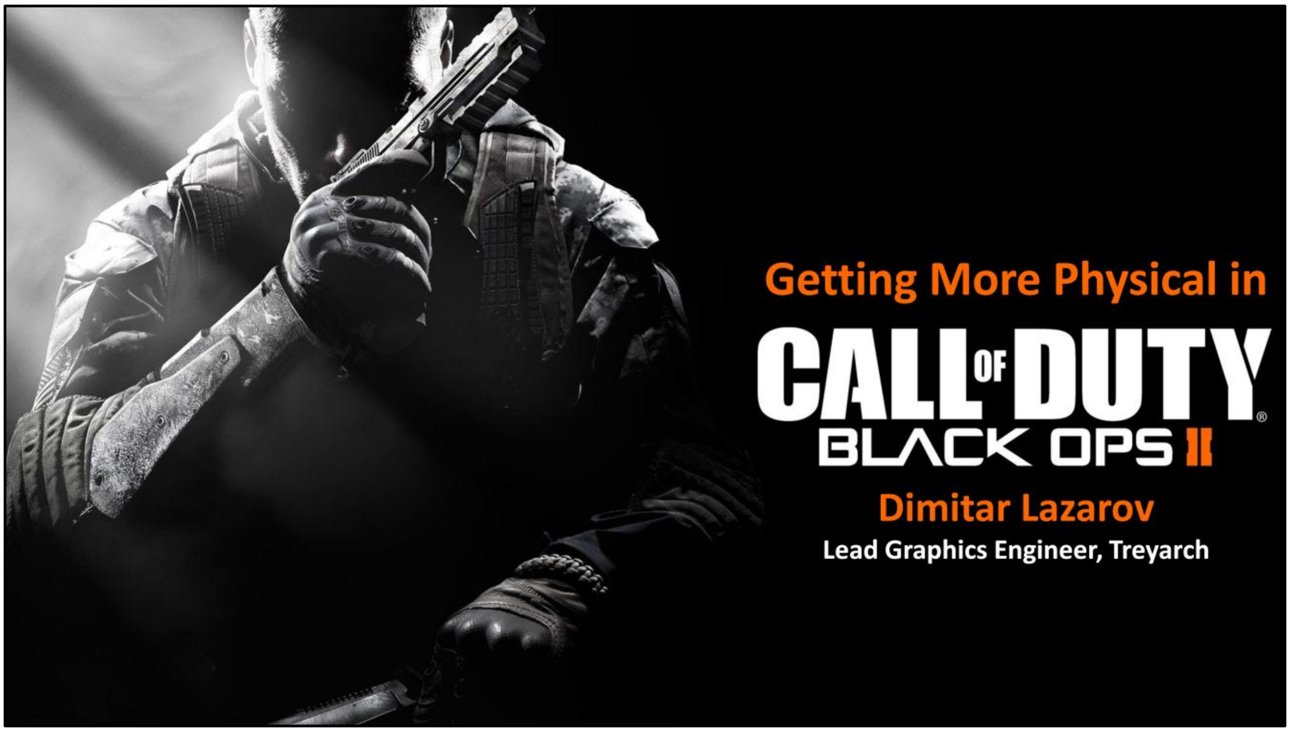
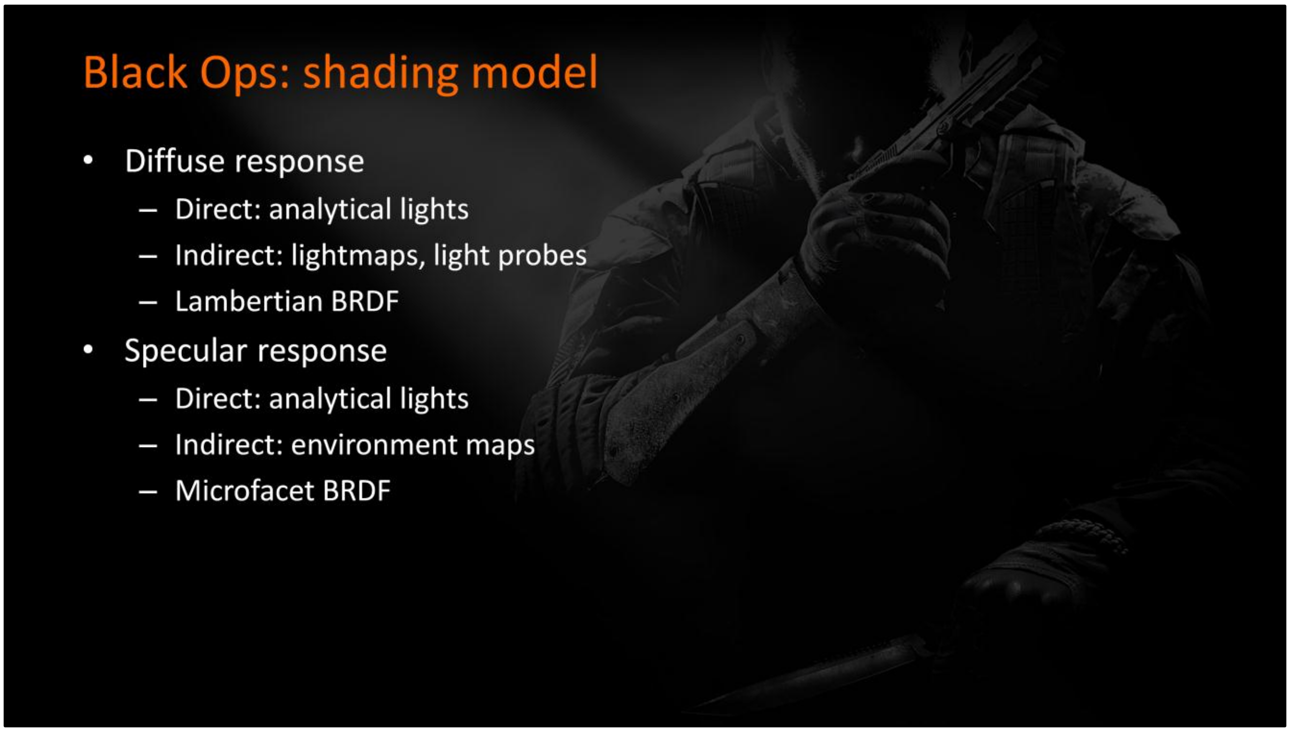


Getting More Physical in CALL OF DUTY BLACK OPS II

Dimitar Lazarov
Lead Graphics Engineer, Treyarch

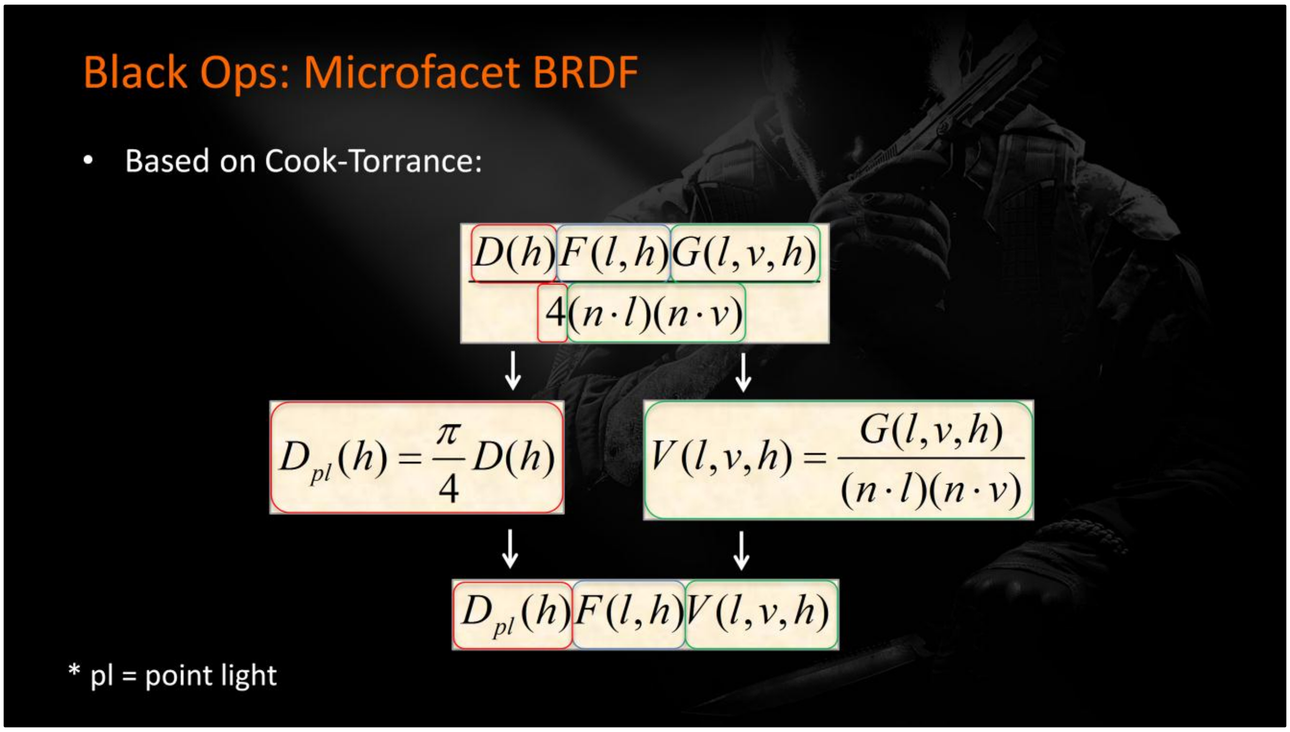
We started pursuing Physically Based Shading during Black Ops, details of which I presented at SIGGRAPH 2011, as part of the "Advances in Real-Time Rendering" course.

Here I'll present a brief summary of the important aspects and then dive into improvements we made on Black Ops II.



Black Ops: shading model

- Diffuse response
  - Direct: analytical lights
  - Indirect: lightmaps, light probes
  - Lambertian BRDF
- Specular response
  - Direct: analytical lights
  - Indirect: environment maps
  - Microfacet BRDF



Black Ops: Microfacet BRDF

- Based on Cook-Torrance:

$$\frac{D(h)F(l,h)G(l,v,h)}{4(n\cdot l)(n\cdot v)}$$

$$\downarrow \qquad\qquad \downarrow$$

$$D_{pl}(h)=\frac{\pi}{4}D(h) \qquad V(l,v,h)=\frac{G(l,v,h)}{(n\cdot l)(n\cdot v)}$$

$$\downarrow \qquad\qquad \downarrow$$

$$D_{pl}(h)F(l,h)V(l,v,h)$$

* pl = point light

D = normal distribution function (NDF)

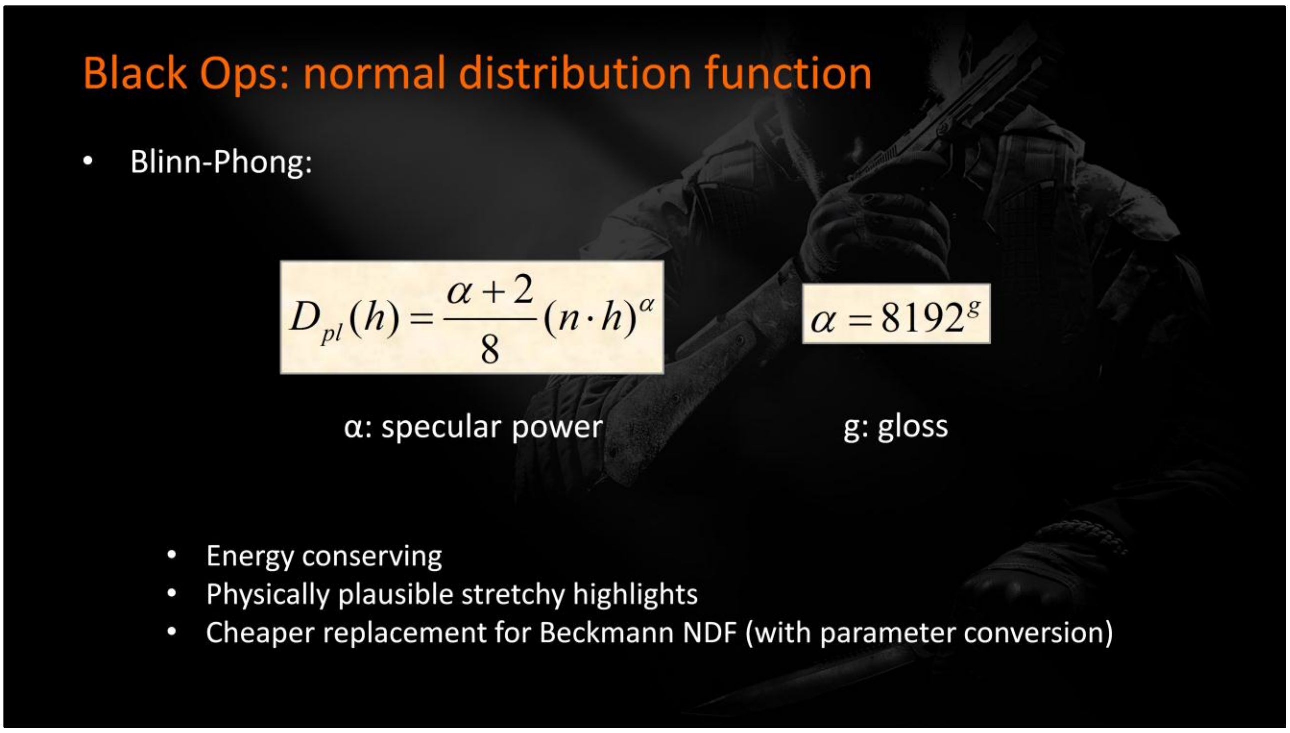
F = reflectance function (Fresnel function)

G = shadowing-masking function (Geometry function)

Grouped into three “lego” pieces for easier experimentation with different formulations and approximations.

D_pl = NDF for point lights grouped with pi/4.

V = visibility function: shadowing-masking grouped with foreshortening terms.



Black Ops: normal distribution function

- Blinn-Phong:

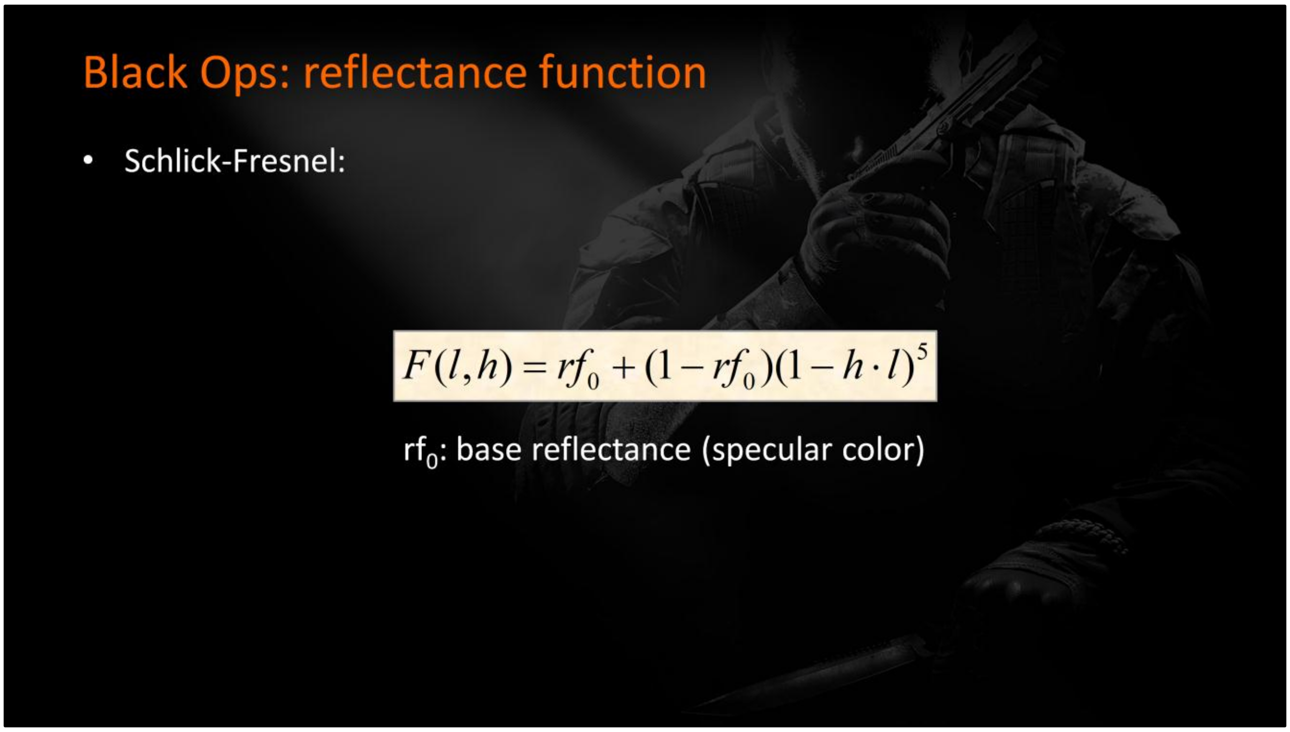
$$D_{pl}(h) = \frac{\alpha + 2}{8}(n \cdot h)^{\alpha}$$

$$\alpha = 8192^{g}$$

α: specular power

g: gloss

  - Energy conserving
  - Physically plausible stretchy highlights
  - Cheaper replacement for Beckmann NDF (with parameter conversion)



Black Ops: reflectance function

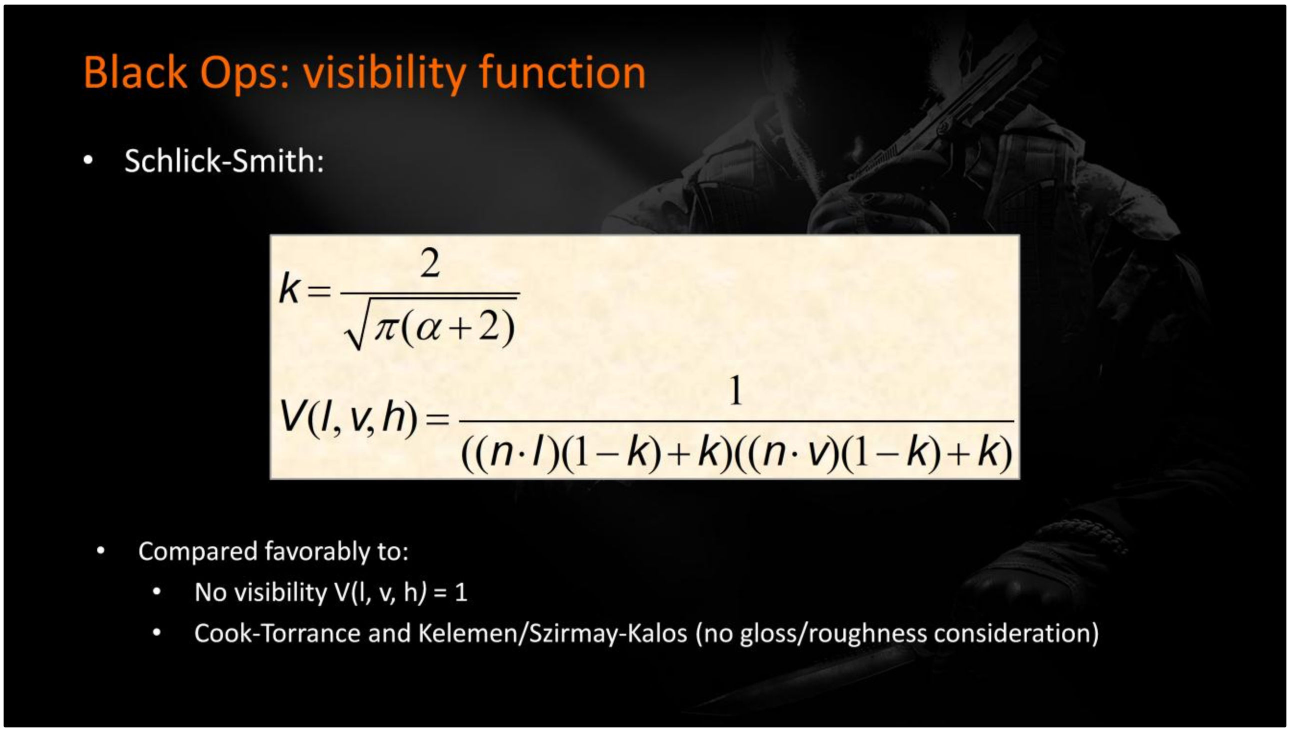
- Schlick-Fresnel:

$$F(l,h) = rf_0 + (1 - rf_0)(1 - h \cdot l)^5$$

rf_0: base reflectance (specular color)

Microfacet H-dot-L-based formulation.

Everyone seems to use this.



Black Ops: visibility function

- Schlick-Smith:

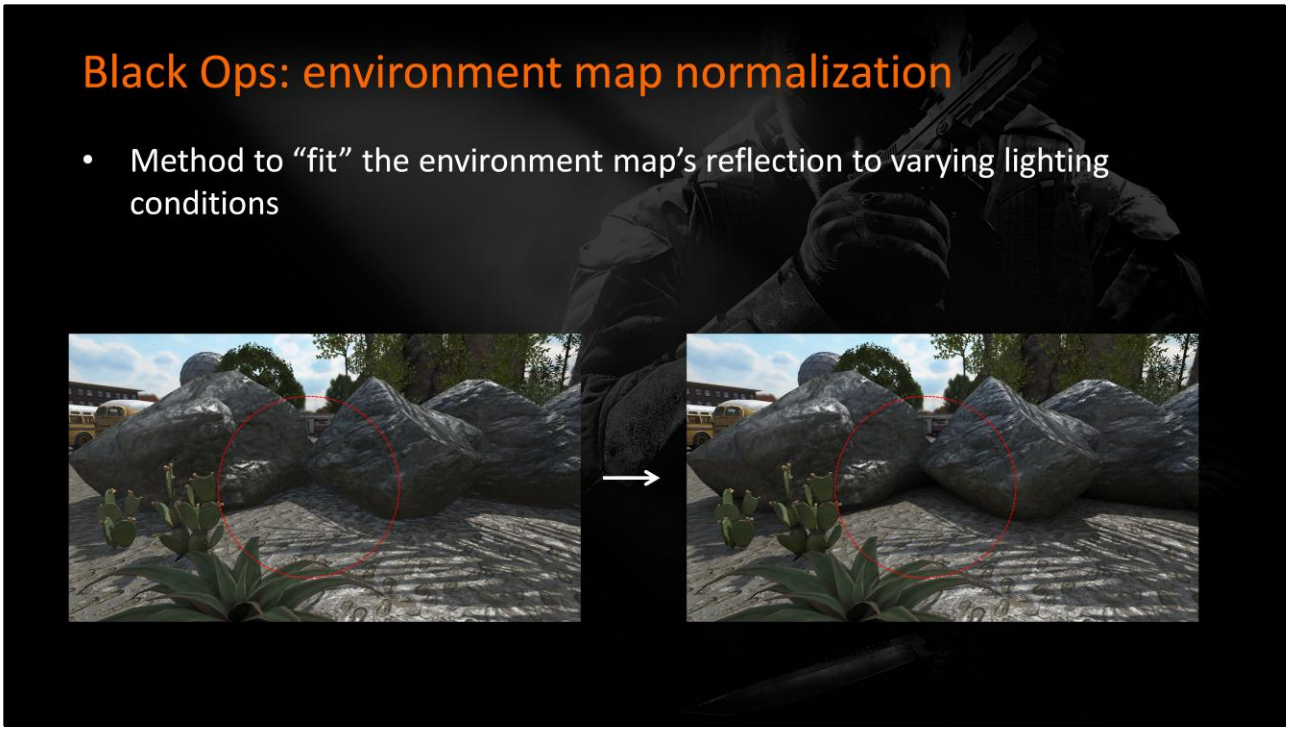
$$k = \frac{2}{\sqrt{\pi(\alpha+2)}}$$

$$V(l,v,h) = \frac{1}{((n \cdot l)(1-k)+k)((n \cdot v)(1-k)+k)}$$

- Compared favorably to:
  - No visibility V(l, v, h*)* = 1
  - Cook-Torrance and Kelemen/Szirmay-Kalos (no gloss/roughness consideration)

Schlick-Smith gave the most plausible (albeit the most expensive) specular response.

Lesson learned: very important to have high quality shadow-masking for PBS.



Black Ops: environment map normalization

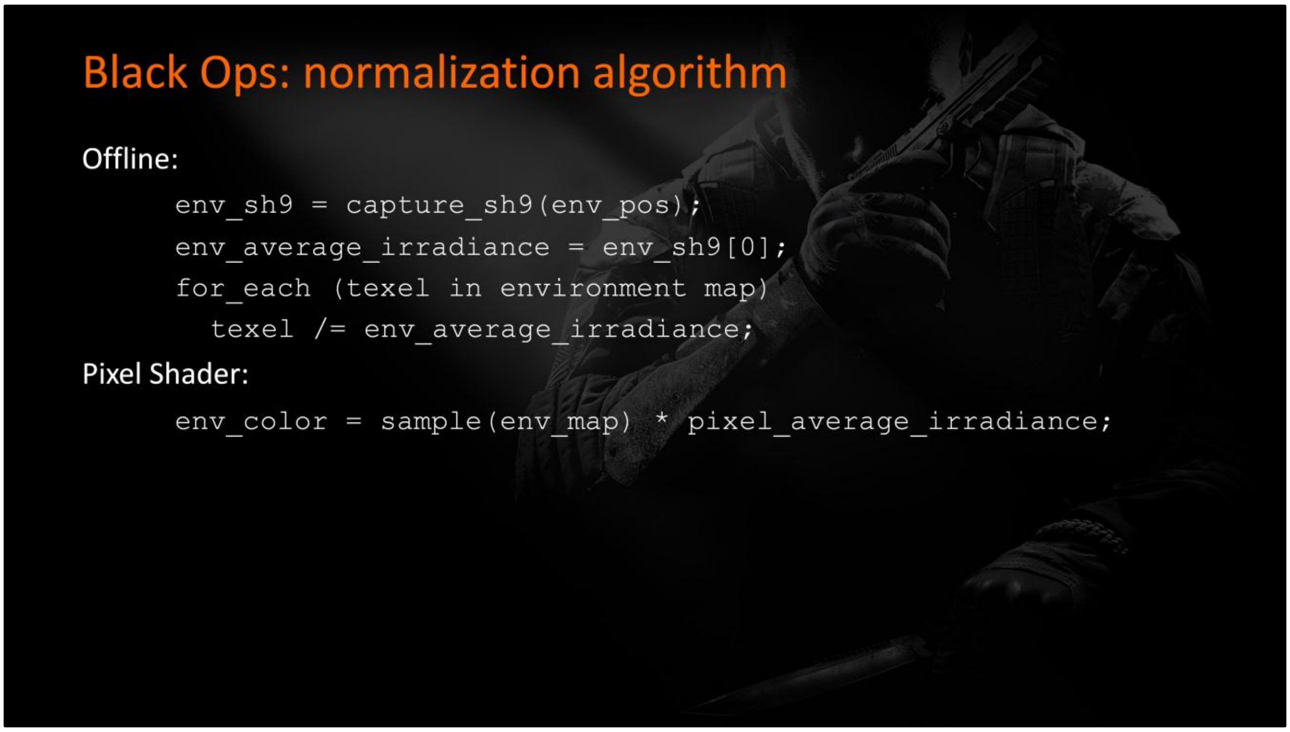
- Method to "fit" the environment map's reflection to varying lighting conditions

Switching gears: indirect specular via environment maps.

We capture environment maps at artist-selected locations in the level.

We wanted to use as few and as high resolution environment maps as possible.

We came up with a method that lets us better "fit" and hence "reuse" the same environment map in very different lighting environments compared to where the environment map was captured.



Black Ops: normalization algorithm

Offline:

```
env_sh9 = capture_sh9(env_pos);
env_average_irradiance = env_sh9[0];
for_each (texel in environment map)
  texel /= env_average_irradiance;
```

Pixel Shader:

```
env_color = sample(env_map) * pixel_average_irradiance;
```

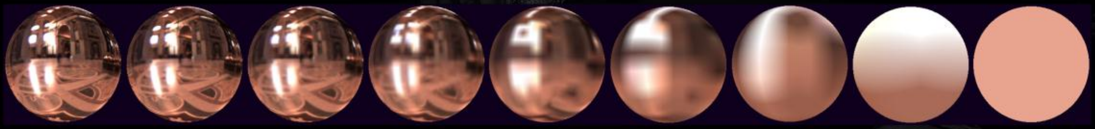
env_sh[0] is the DC term, which is equivalent to evaluating the SH with a (0, 0, 0) direction.

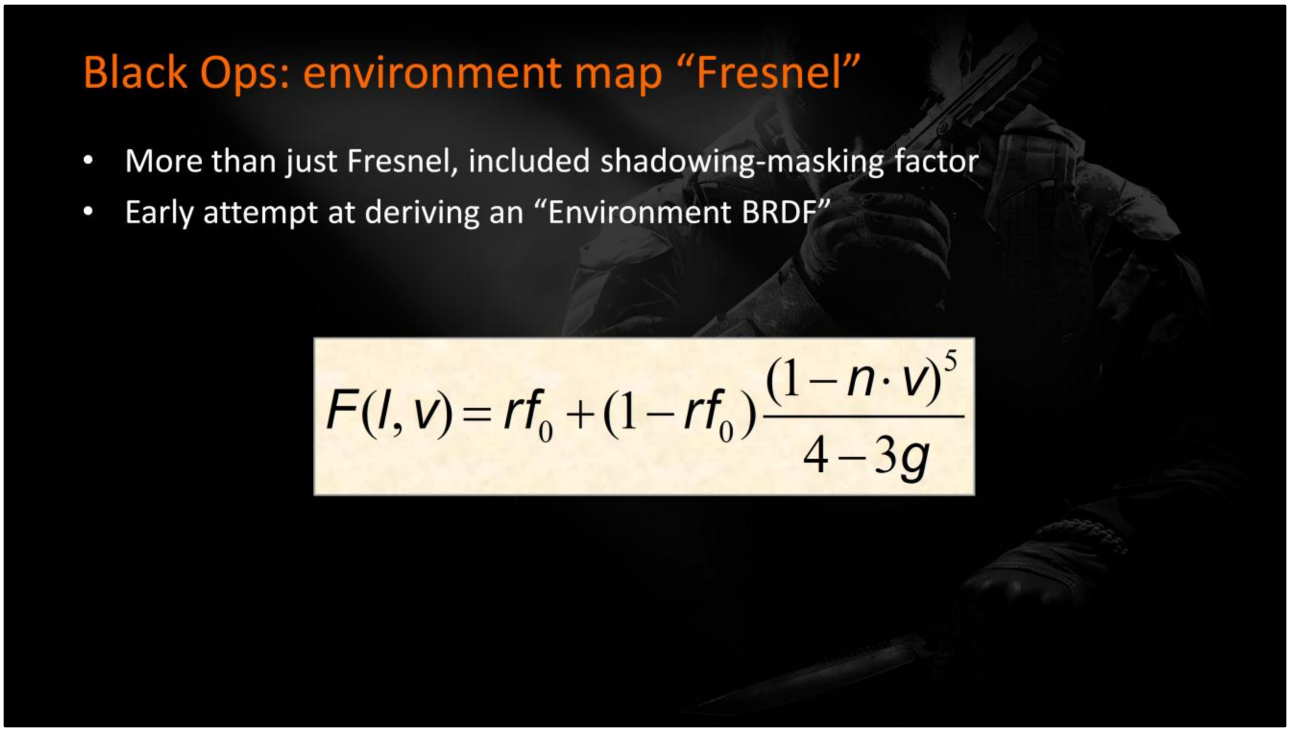
pixel_average_irradiance can usually be calculated as a cheap byproduct from most forms of light baking.

Black Ops: environment map pre-filtering

- Offline, CubeMapGen
  - Angular Gaussian filter
  - Edge fixup
- Pixel shader selects mip as a linear function of gloss:

```
        texCUBElod(uv, float4(R, nMips - gloss * nMips));
```





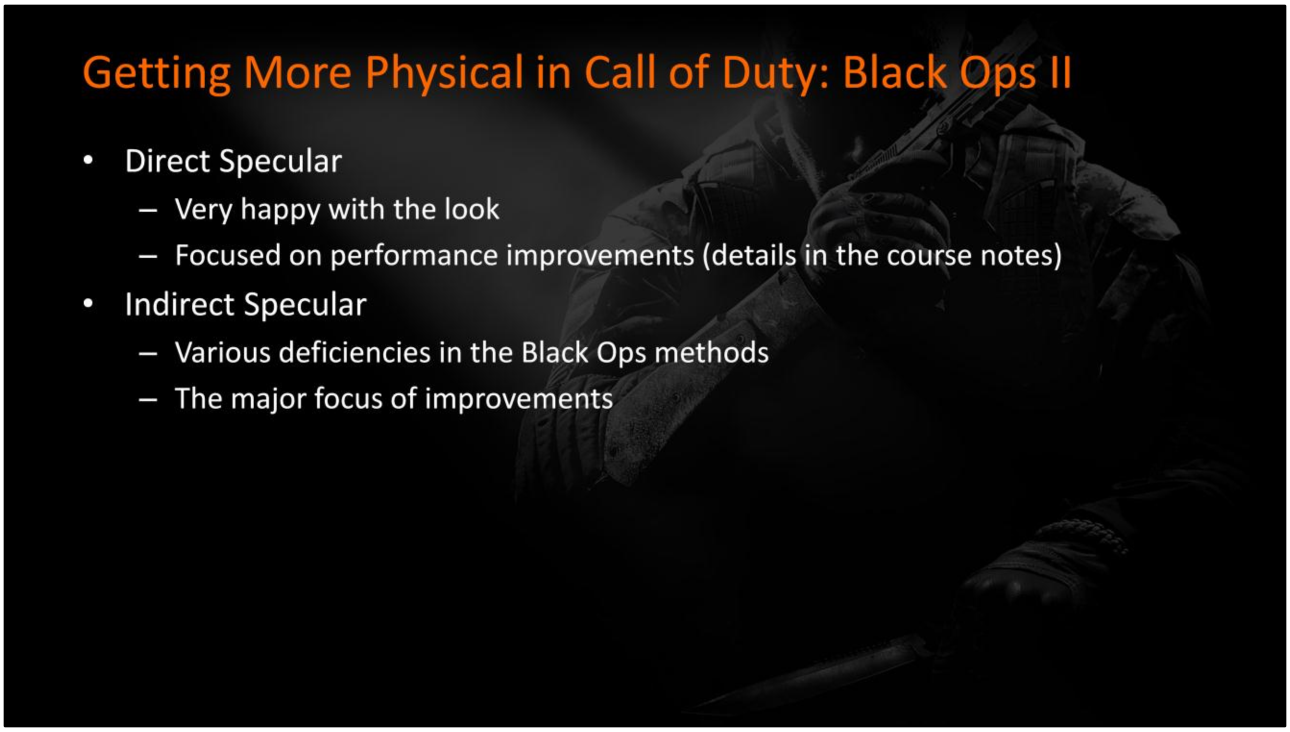
Black Ops: environment map "Fresnel"

- More than just Fresnel, included shadowing-masking factor
- Early attempt at deriving an "Environment BRDF"

$$F(l,v) = rf_0 + (1 - rf_0)\frac{(1 - n \cdot v)^5}{4 - 3g}$$

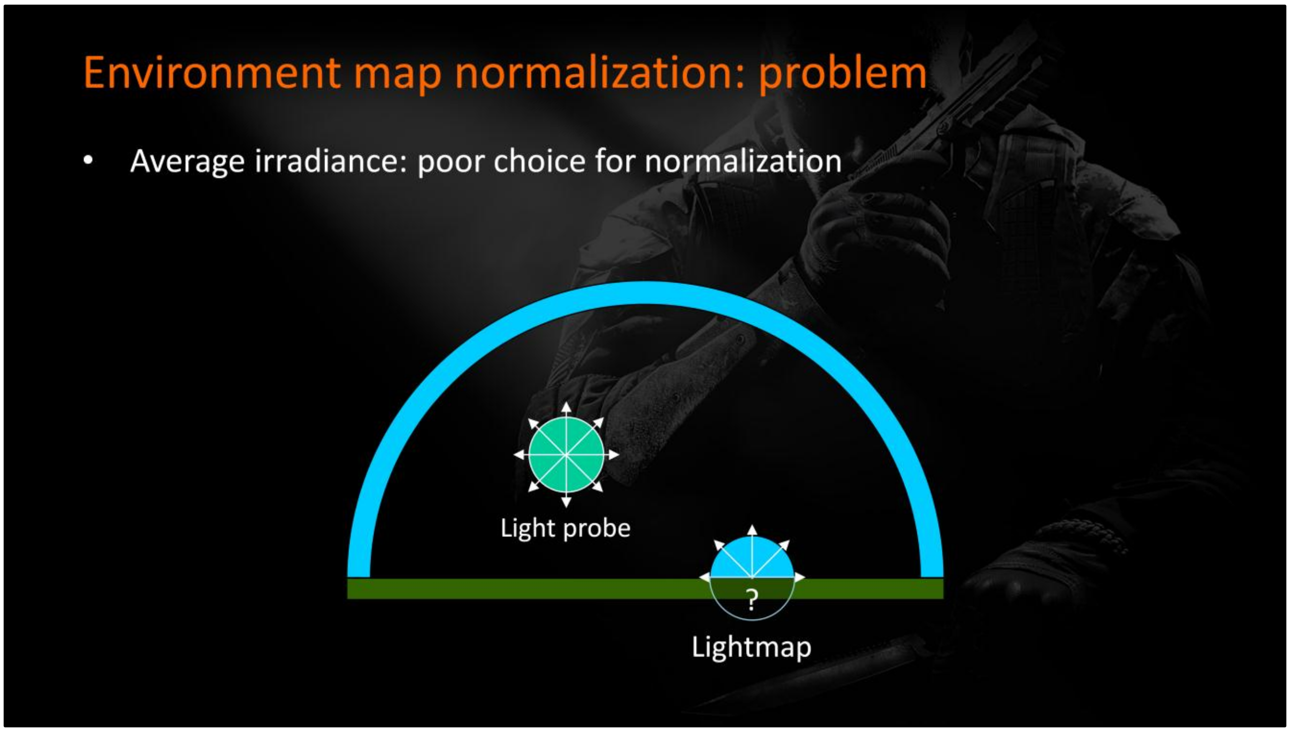
Here we're using the N-dot-V form of the Fresnel formulation.

1/(4 – 3g) acts as a shadowing-masking factor



Getting More Physical in Call of Duty: Black Ops II

- Direct Specular
  - Very happy with the look
  - Focused on performance improvements (details in the course notes)
- Indirect Specular
  - Various deficiencies in the Black Ops methods
  - The major focus of improvements



Environment map normalization: problem

- Average irradiance: poor choice for normalization

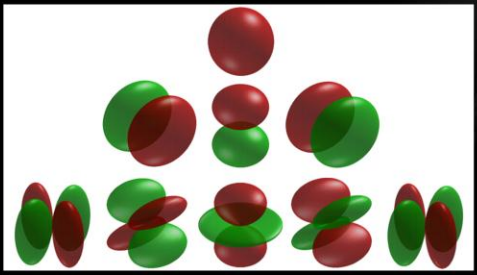
The picture shows conceptually the problem through the use of colors. In practice, the problems were mostly intensity related and less often about color.

The environment map is normalized from a light probe sample (full sphere of lighting information).

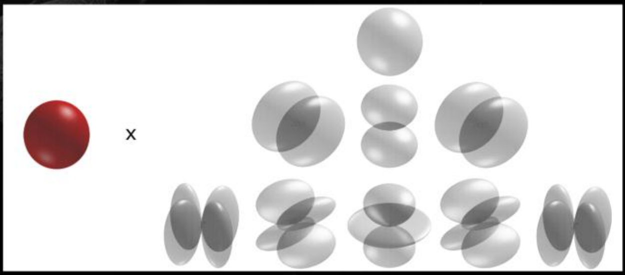
However, lightmap bakes often store only a hemisphere worth of information, hence when we de-normalize the environment map we can’t compensate for the missing hemisphere.

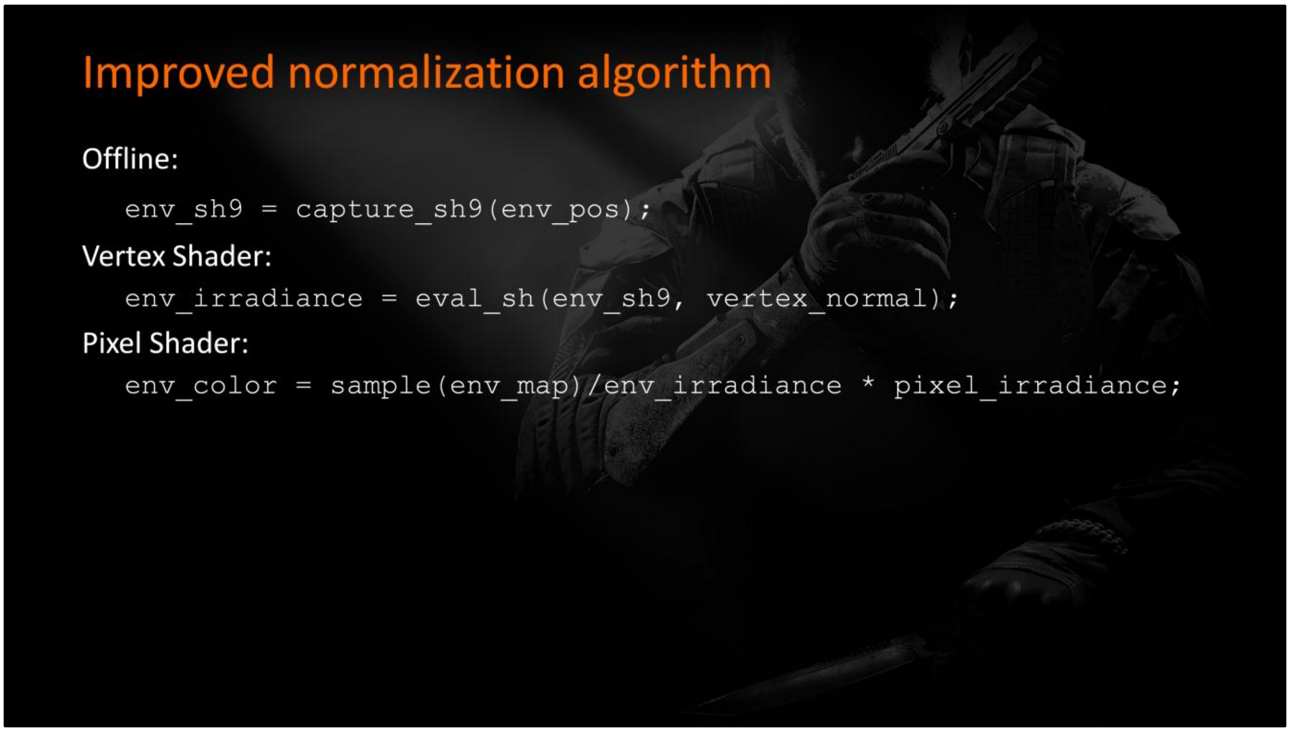
Environment map normalization: new idea

- Normalize with irradiance
  - Can't bake normalization offline
  - Pass environment map's directional irradiance to run-time (used tinted scalar 3rd-order Spherical Harmonics)



$\longrightarrow$





Improved normalization algorithm

Offline:

```
env_sh9 = capture_sh9(env_pos);
```

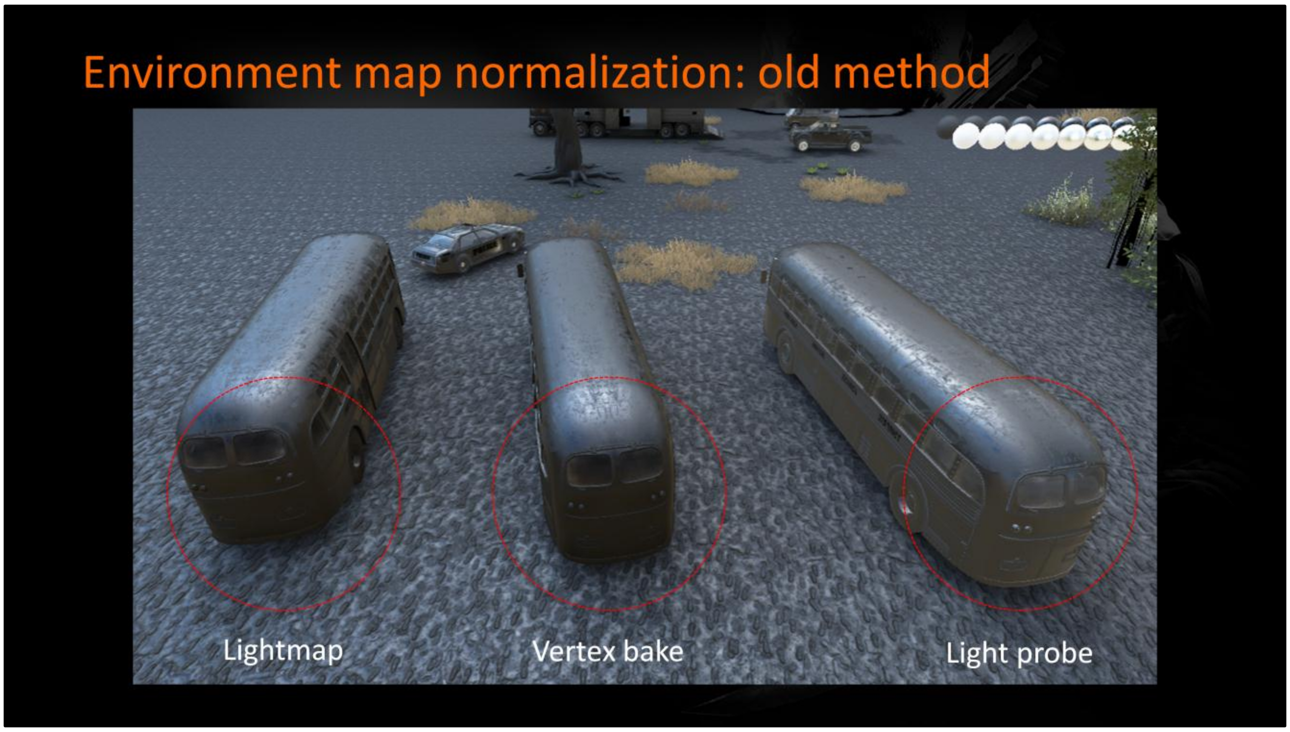
Vertex Shader:

```
env_irradiance = eval_sh(env_sh9, vertex_normal);
```

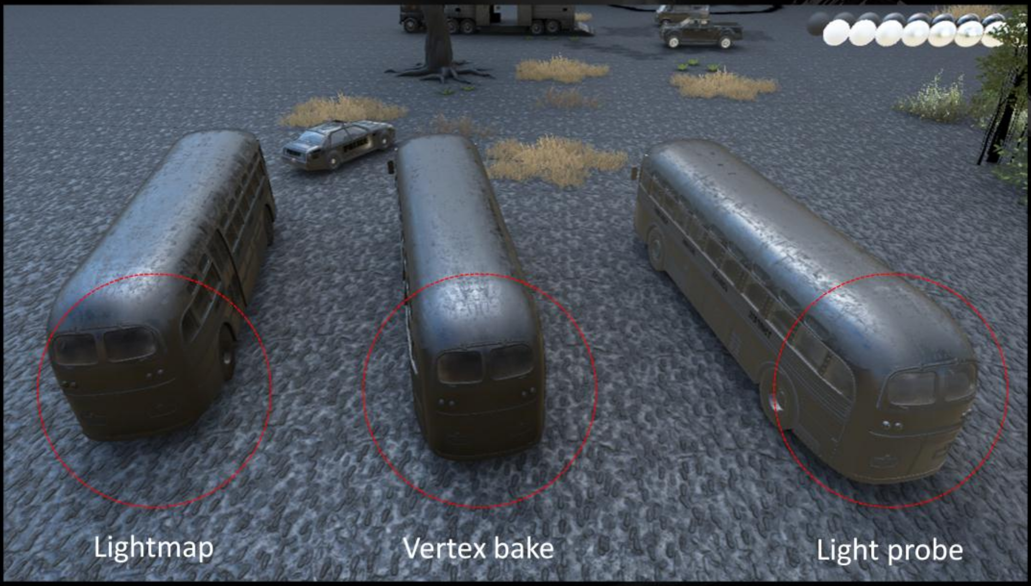
Pixel Shader:

```
env_color = sample(env_map)/env_irradiance * pixel_irradiance;
```

pixel_irradiance can usually be calculated as a cheap byproduct of most forms of light baking. It’s virtually guaranteed to exist considering geometric normal irradiance is the most important direction to reproduce correctly.

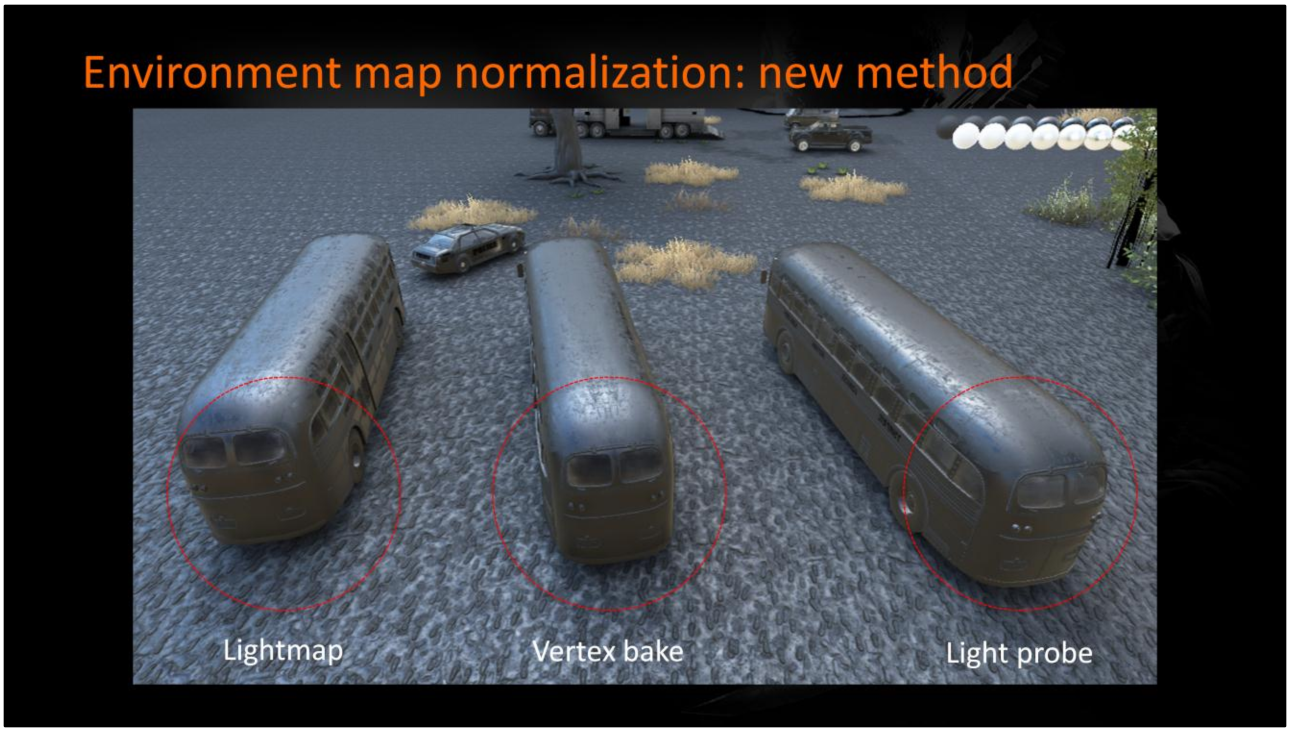


Environment map normalization: old method

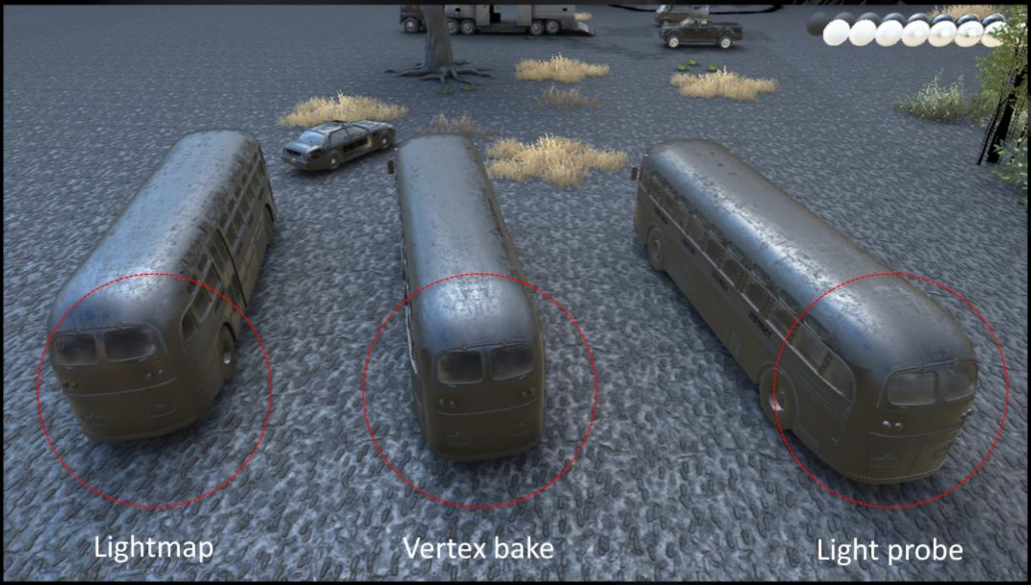


The same mesh with three different light bakes.

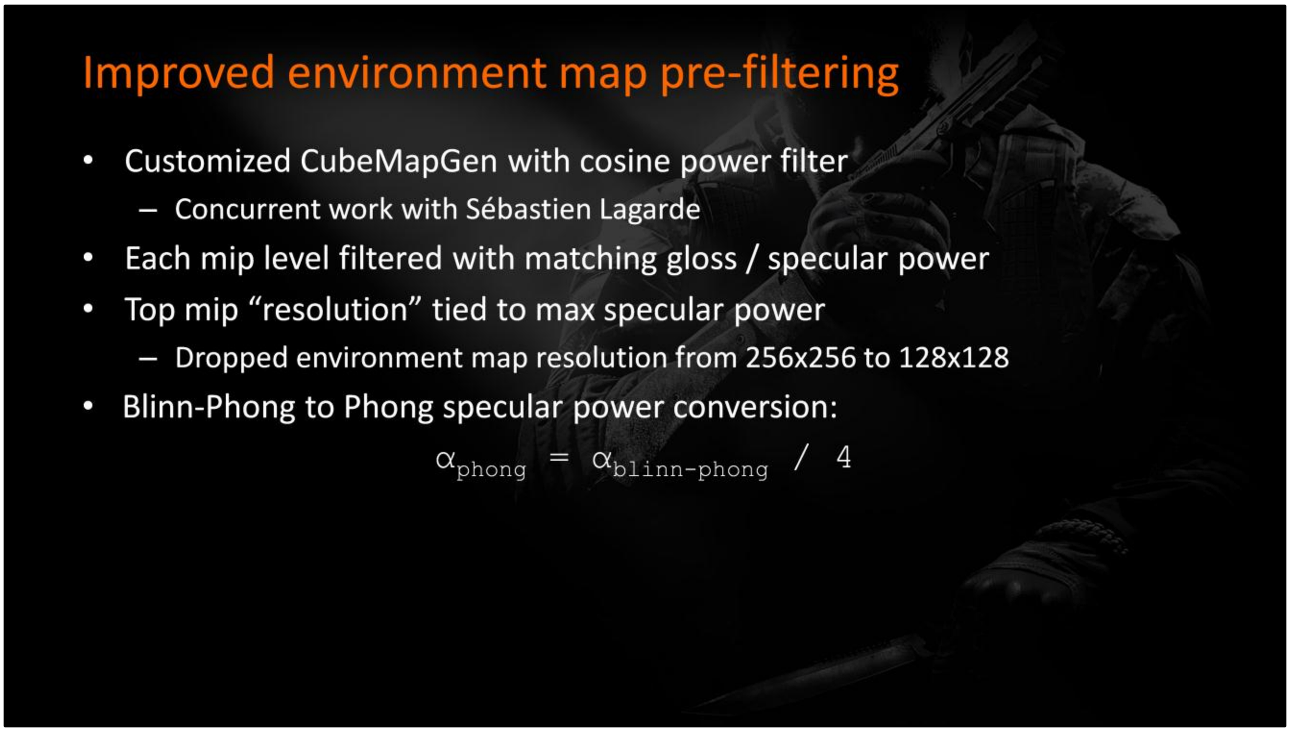
The image shows environment specular, boosted 3 stops for easier comparison.



Environment map normalization: new method



Notice how the specular from the different light bakes is a lot closer now.



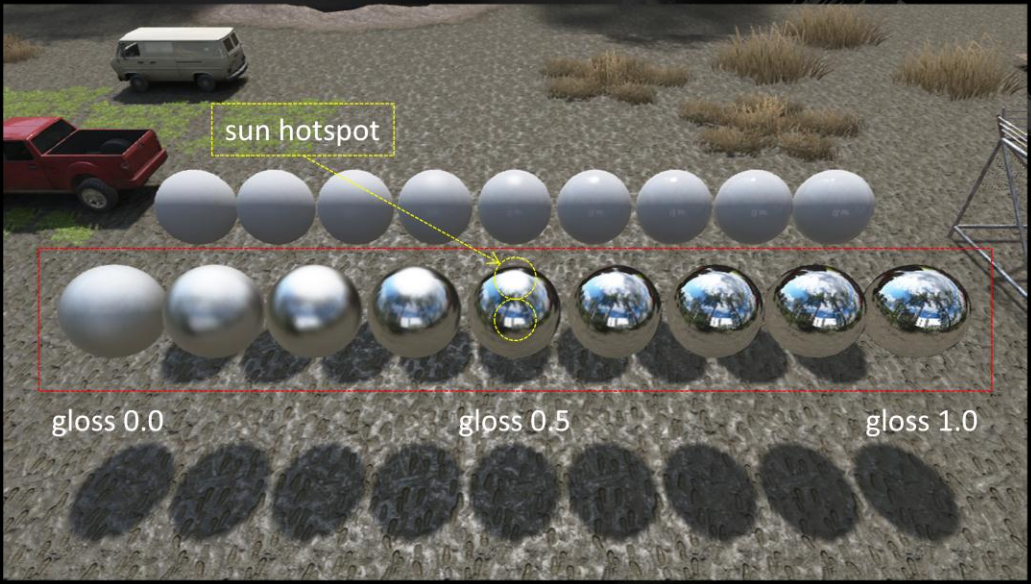
Improved environment map pre-filtering

- Customized CubeMapGen with cosine power filter
  - Concurrent work with Sébastien Lagarde
- Each mip level filtered with matching gloss / specular power
- Top mip "resolution" tied to max specular power
  - Dropped environment map resolution from 256x256 to 128x128
- Blinn-Phong to Phong specular power conversion:

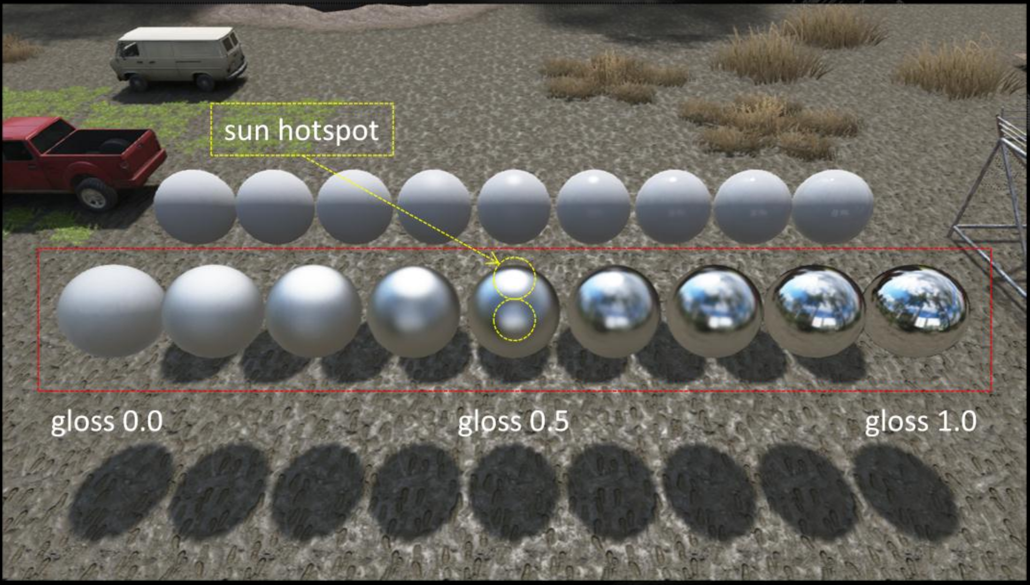
$$\alpha_{phong} = \alpha_{blinn\text{-}phong} / 4$$

The Black Ops pre-filtering used a Gaussian, which was not an exact match to our cosine power distribution function.

Environment map pre-filtering: old method

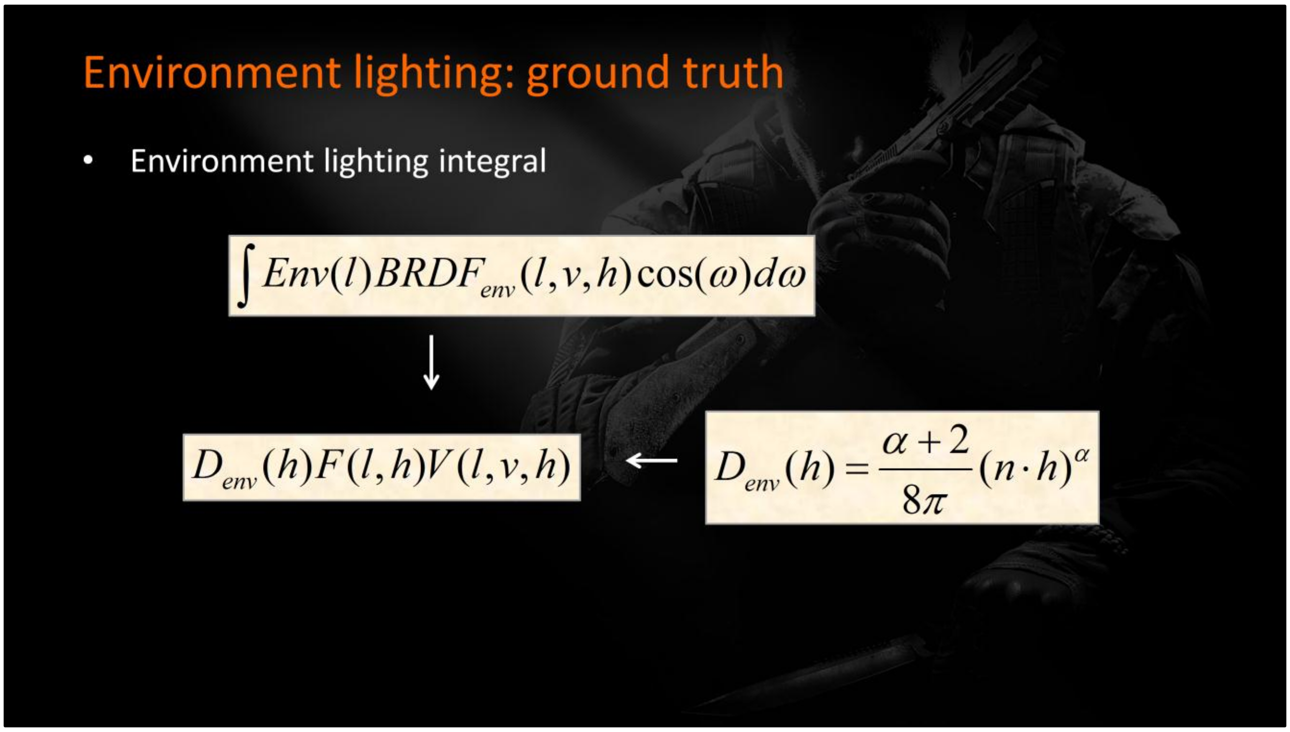


Environment map pre-filtering: new method



Notice how the environment map blurs much more linearly across the gloss range.

Also notice how the environment blur is a closer match to the sun hotspot diffusion.



Environment lighting: ground truth

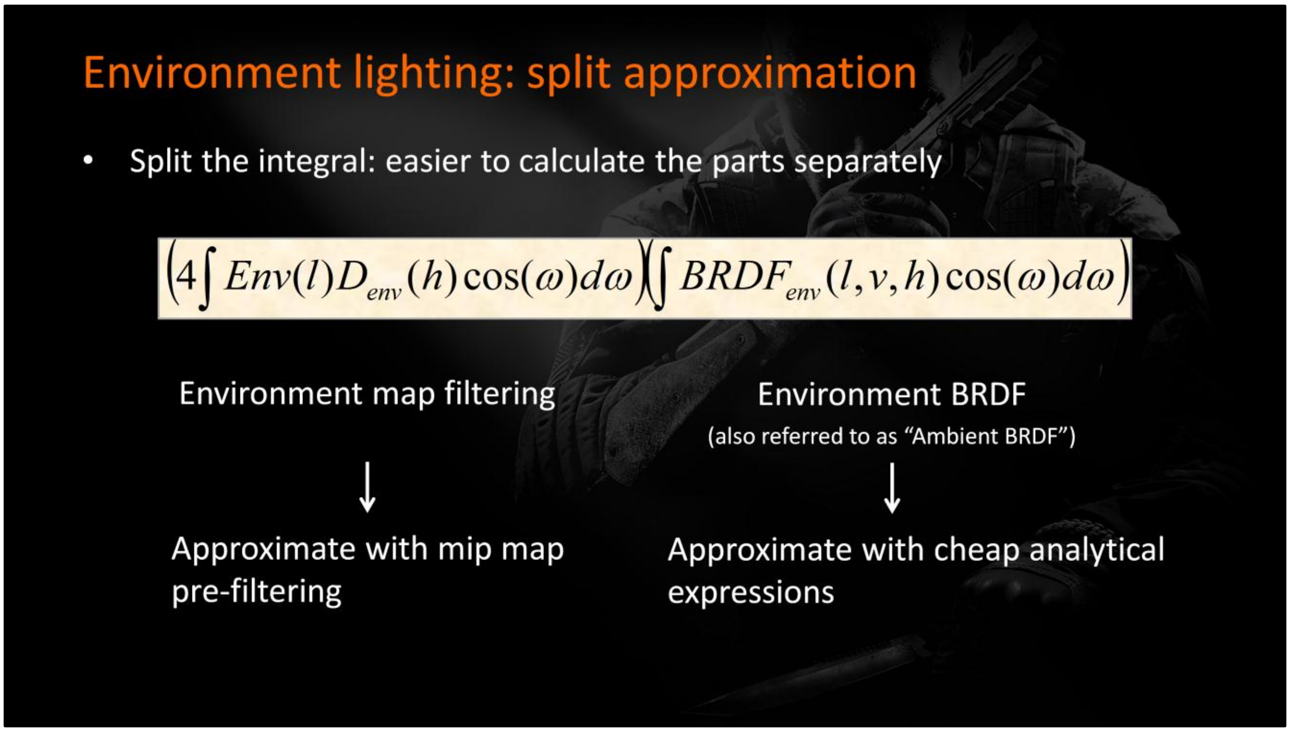
- Environment lighting integral

$$\begin{tikzcd}
\int Env(l)BRDF_{env}(l,v,h)\cos(\omega)d\omega \arrow[d] & \\
D_{env}(h)F(l,h)V(l,v,h) & D_{env}(h)=\frac{\alpha+2}{8\pi}(n\cdot h)^{\alpha} \arrow[l]
\end{tikzcd}$$

Before we talk about how we improved the Environment BRDF, let’s talk about what we’re trying to solve first.

One option for solving this integral is to importance sample the environment map and the BRDF in the pixel shader. This is very expensive, but it’s a good ground truth.

Note that the environment NDF has a division by pi, as opposed to the point-light NDF which does not.



Environment lighting: split approximation

- Split the integral: easier to calculate the parts separately

$$\left(4\int Env(l)D_{env}(h)\cos(\omega)d\omega\right)\left(\int BRDF_{env}(l,v,h)\cos(\omega)d\omega\right)$$

Environment map filtering

$\downarrow$

Approximate with mip map pre-filtering

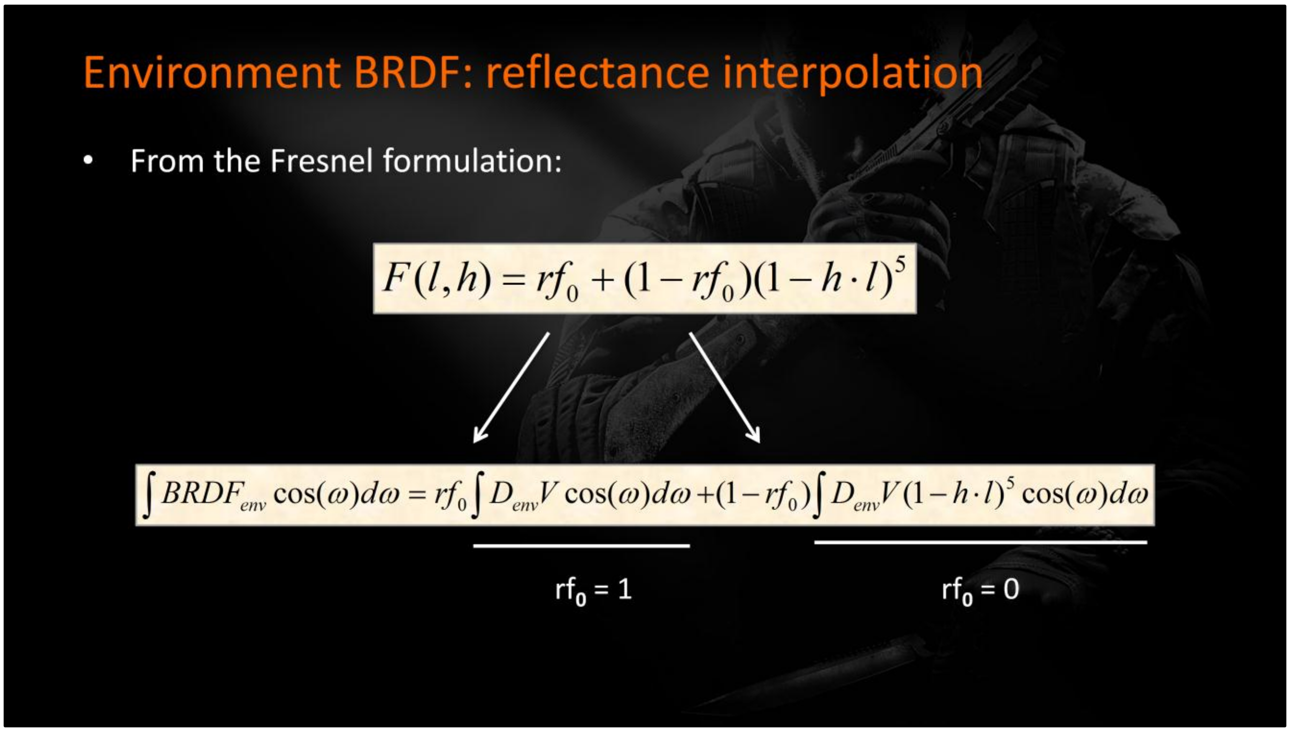
Environment BRDF

(also referred to as "Ambient BRDF")

$\downarrow$

Approximate with cheap analytical expressions

Environment BRDF: conceptually you can think of it as the BRDF result for a white environment map.



Environment BRDF: reflectance interpolation

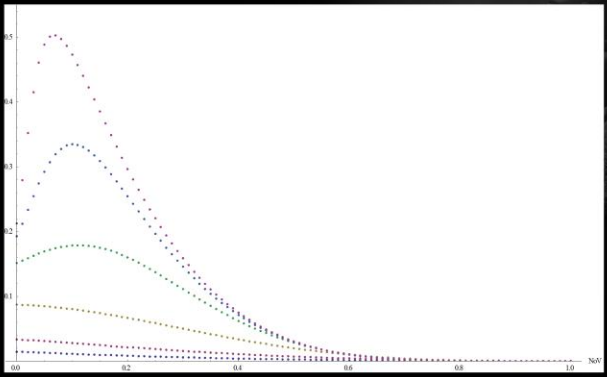
- From the Fresnel formulation:

$$F(l,h) = rf_0 + (1 - rf_0)(1 - h \cdot l)^5$$

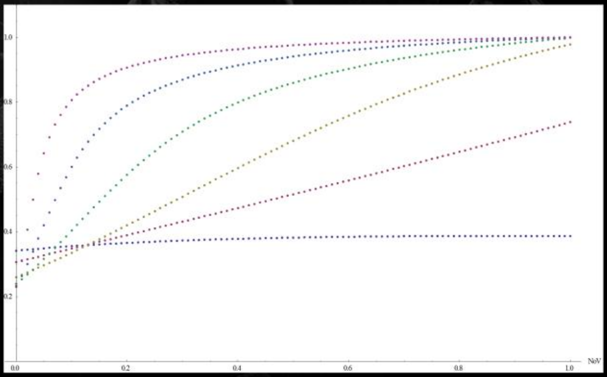
$$\int BRDF_{env} \cos(\omega) d\omega = rf_0 \underbrace{\int D_{env} V \cos(\omega) d\omega}_{rf_0 = 1} + (1 - rf_0) \underbrace{\int D_{env} V (1 - h \cdot l)^5 \cos(\omega) d\omega}_{rf_0 = 0}$$

Numerical integration in *Mathematica*

- Plotted two sets of ground-truth curves for $rf_0 = 0$ and $rf_0 = 1$
- Each set contained curves for gloss values 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0

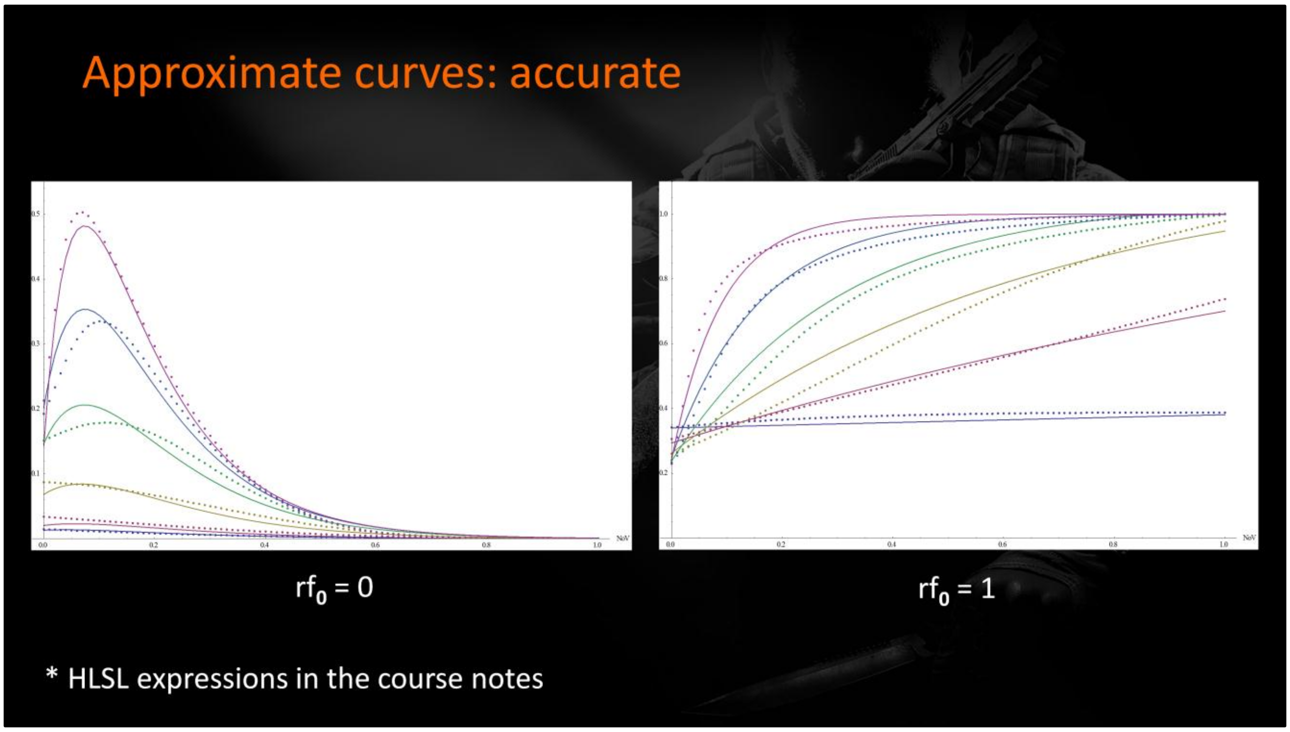


$rf_0 = 0$

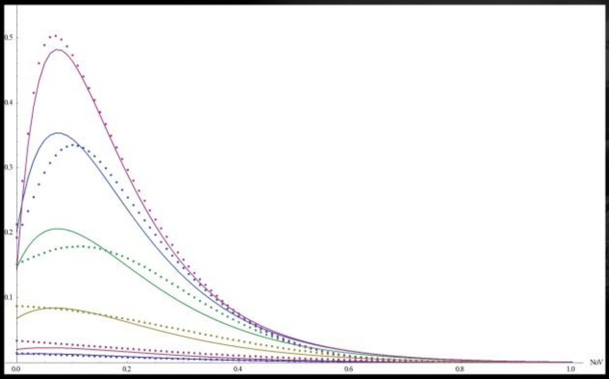


$rf_0 = 1$

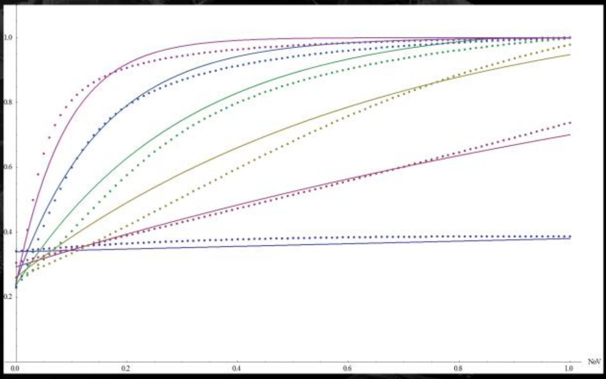
On the x-axis we plot N-dot-V, where rf0 is to the right, glancing angles are to the left.



Approximate curves: accurate



$rf_0 = 0$

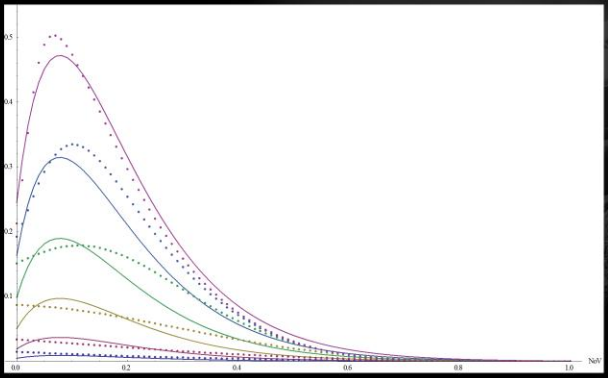


$rf_0 = 1$

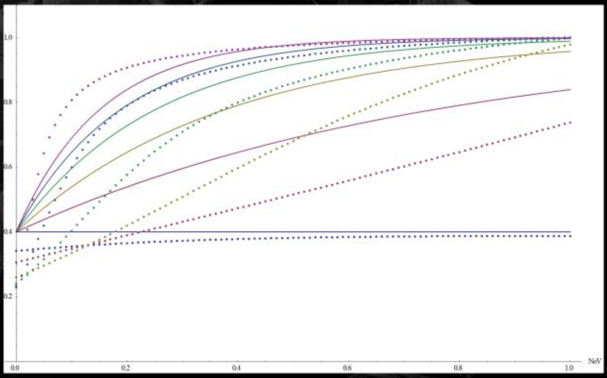
* HLSL expressions in the course notes

We fitted curves by trial and error, using Mathematica to guide the process.

Approximate curves: cheaper



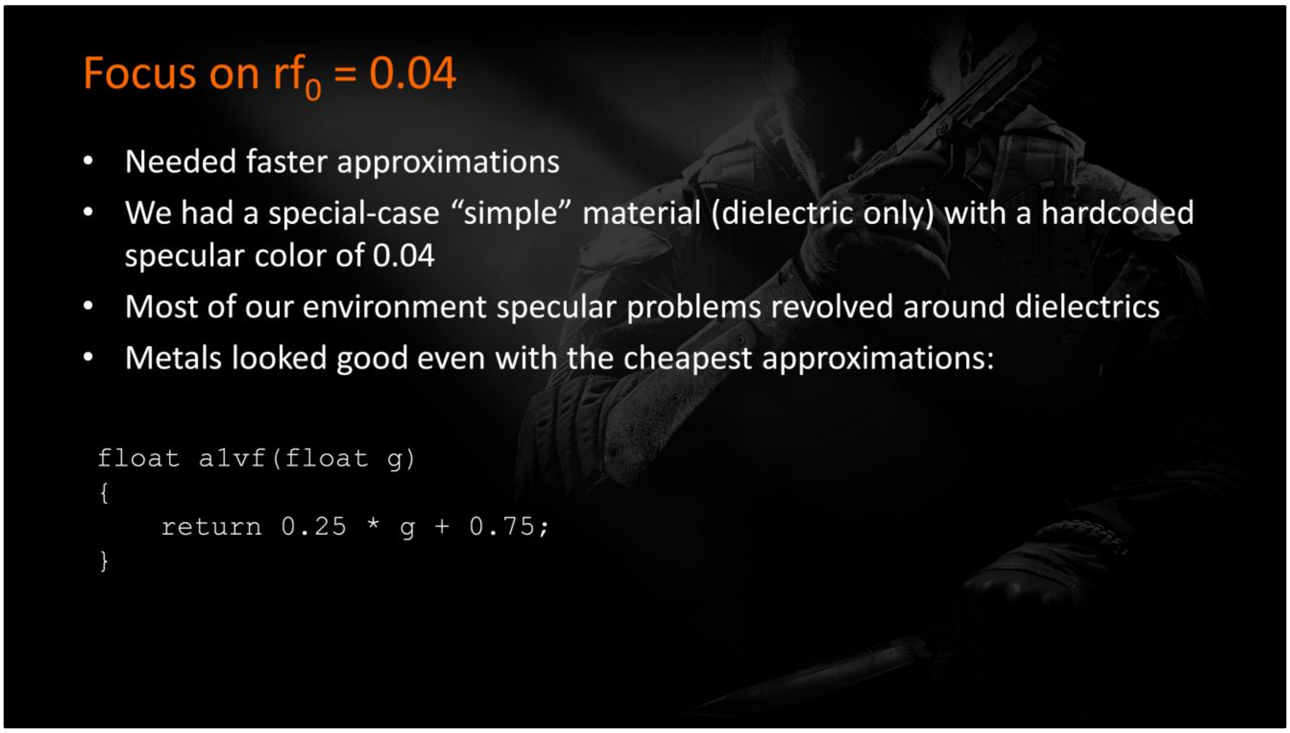
$rf_0 = 0$



$rf_0 = 1$

* HLSL expressions in the course notes

We deployed the “cheaper” curves, with good visual results. However, we needed even cheaper expressions.



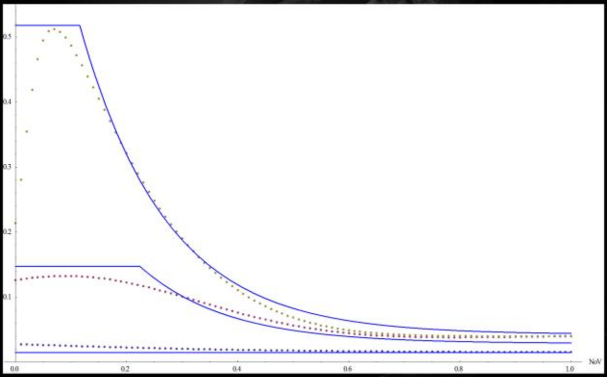
Focus on $rf_0 = 0.04$

- Needed faster approximations
- We had a special-case “simple” material (dielectric only) with a hardcoded specular color of 0.04
- Most of our environment specular problems revolved around dielectrics
- Metals looked good even with the cheapest approximations:

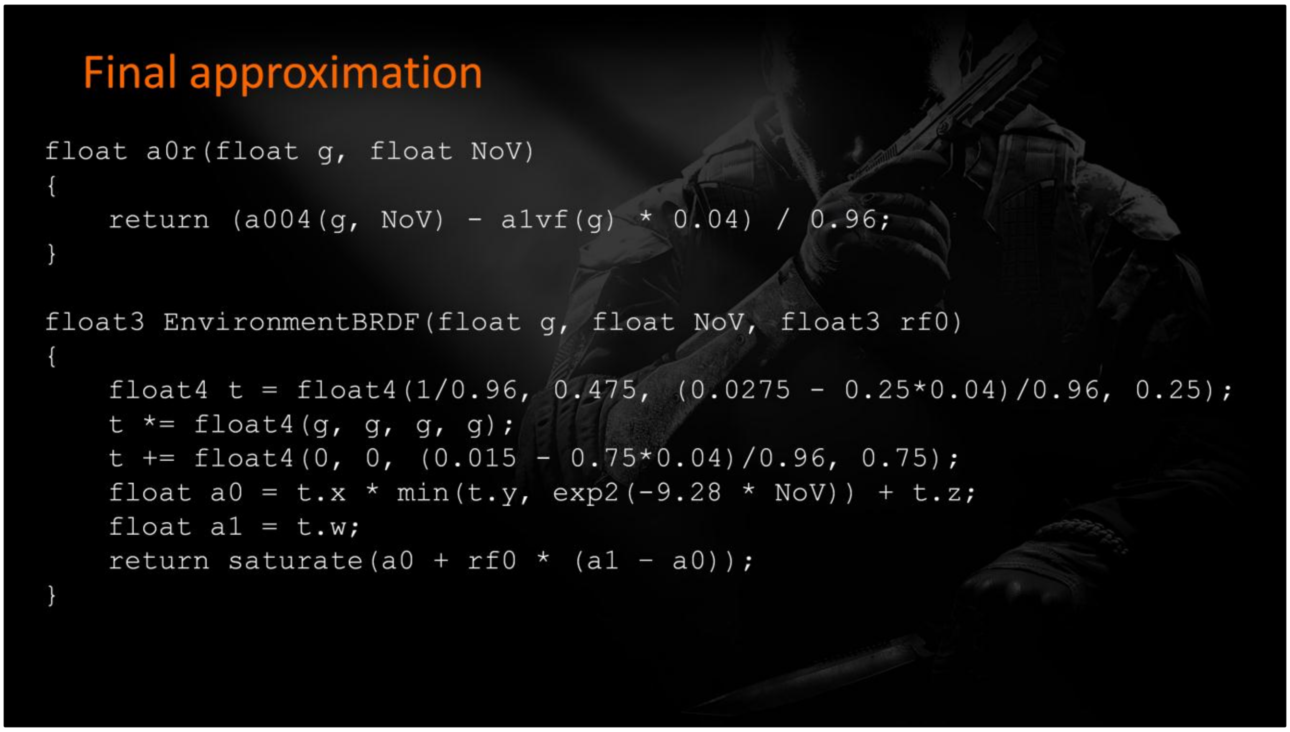
```
float a1vf(float g)
{
    return 0.25 * g + 0.75;
}
```

Approximate curves: $rf_0 = 0.04$

```
float a004(float g, float NoV)
{
    float t = min(0.475 * g, exp2(-9.28 * NoV));
    return (t + 0.0275) * g + 0.015;
}
```



g = 0.0, 0.5, 1.0



Final approximation

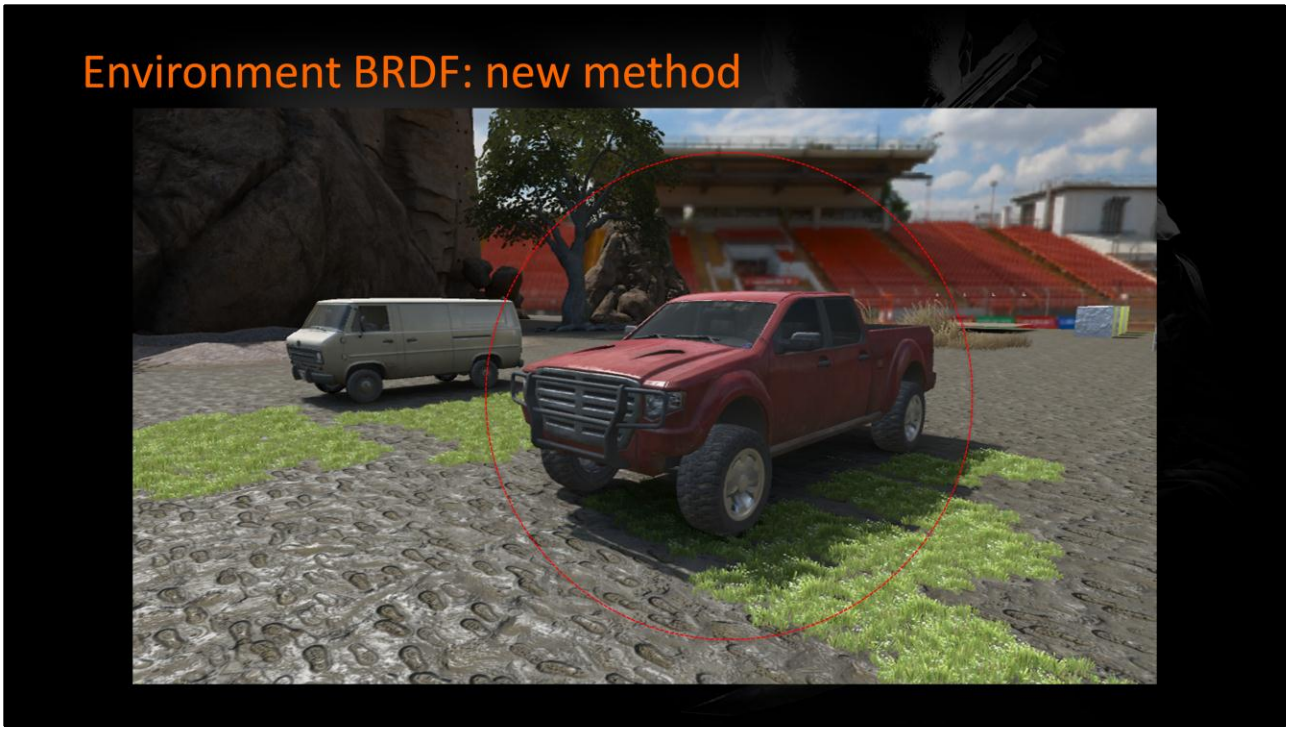
```
float a0r(float g, float NoV)
{
    return (a004(g, NoV) - a1vf(g) * 0.04) / 0.96;
}

float3 EnvironmentBRDF(float g, float NoV, float3 rf0)
{
    float4 t = float4(1/0.96, 0.475, (0.0275 - 0.25*0.04)/0.96, 0.25);
    t *= float4(g, g, g, g);
    t += float4(0, 0, (0.015 - 0.75*0.04)/0.96, 0.75);
    float a0 = t.x * min(t.y, exp2(-9.28 * NoV)) + t.z;
    float a1 = t.w;
    return saturate(a0 + rf0 * (a1 - a0));
}
```

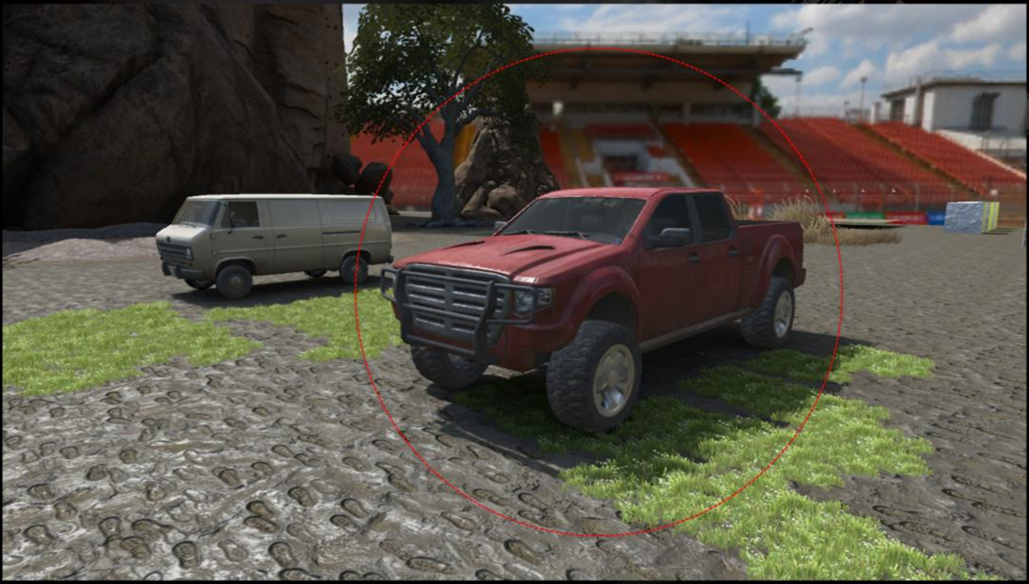
This is what we shipped with.

It was actually faster than the Black Ops Environment “Fresnel”.

Environment BRDF: old method



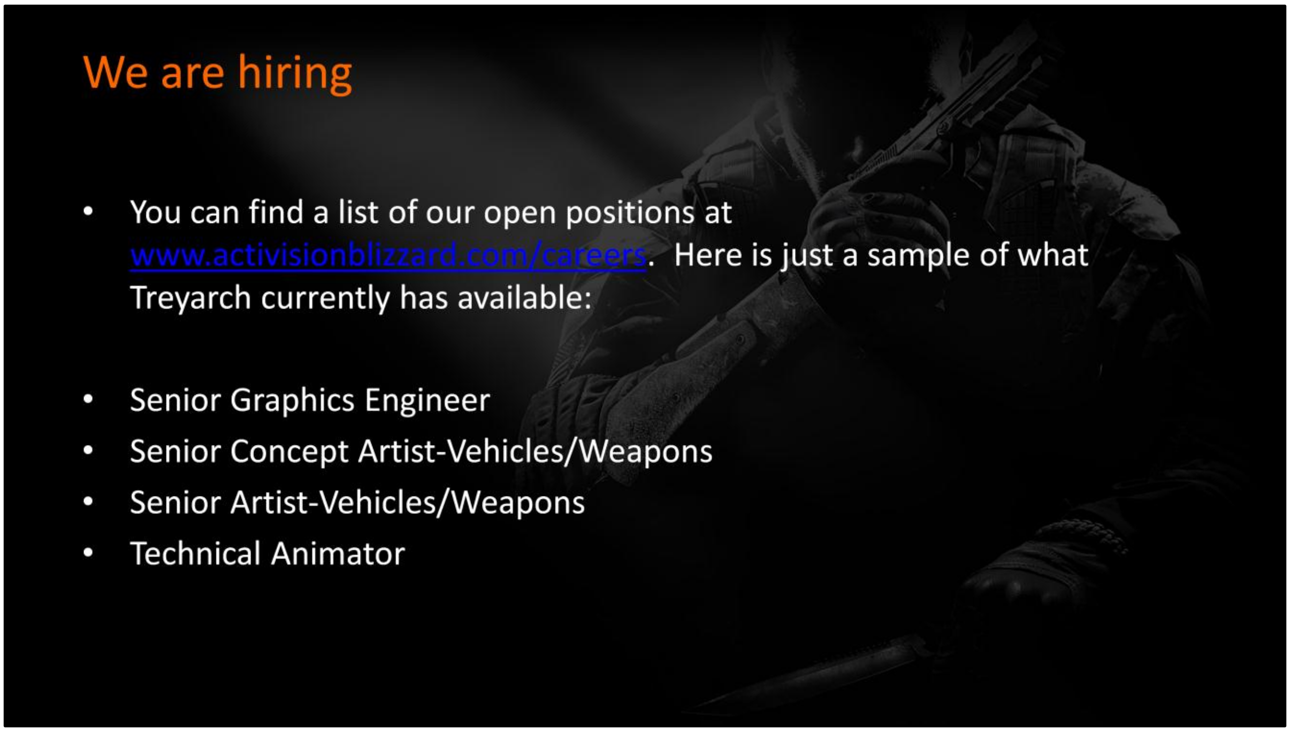
Environment BRDF: new method



Notice the much more natural looking specular on the shadowed side of the truck.
The car paint is a dielectric material (non-metallic paint).

Acknowledgments

- Naty Hoffman
- Marc Olano
- Jorge Jimenez
- Sébastien Lagarde
- Stephen Hill & Stephen McAuley
- The team at Treyarch

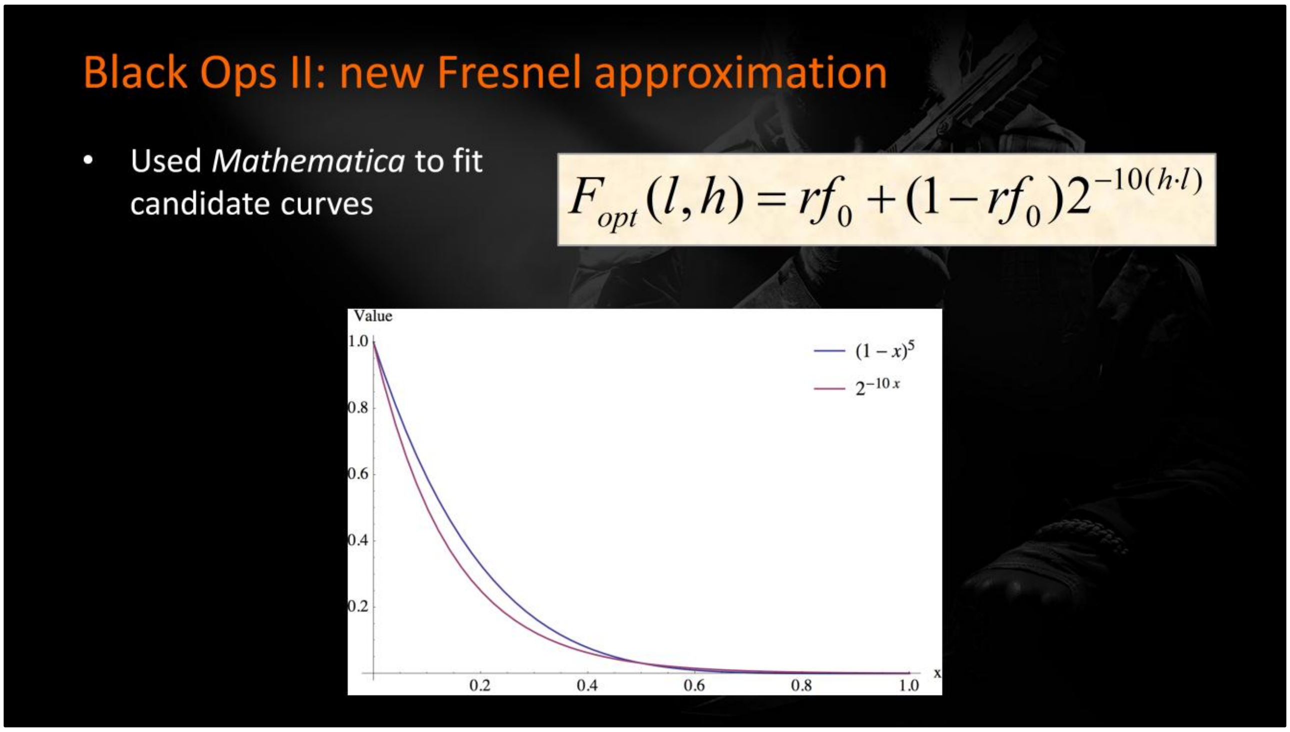


We are hiring

- You can find a list of our open positions at www.activisionblizzard.com/careers. Here is just a sample of what Treyarch currently has available:

- Senior Graphics Engineer
- Senior Concept Artist-Vehicles/Weapons
- Senior Artist-Vehicles/Weapons
- Technical Animator

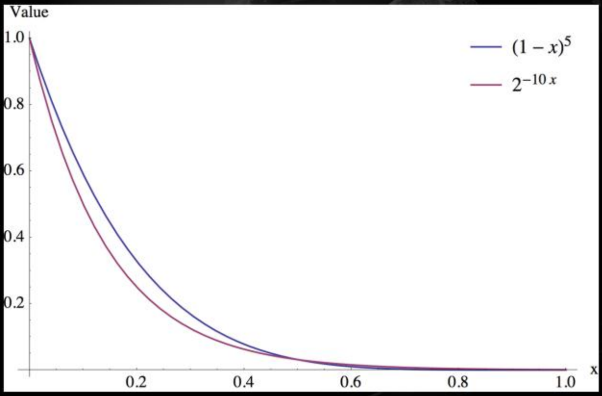
Bonus slides



Black Ops II: new Fresnel approximation

- Used *Mathematica* to fit candidate curves

$$F_{opt}(l,h) = rf_0 + (1 - rf_0)2^{-10(h \cdot l)}$$



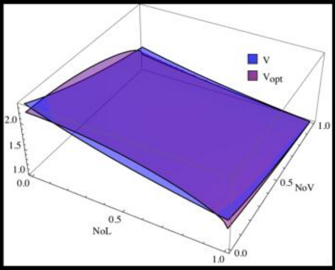
Note: this is to approximate the (1 – H-dot-L)^5 part of Schlick’s Fresnel function.

Black Ops II: new visibility function approximation

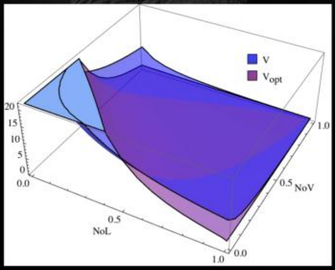
- Visually matched in game
  (not an exact fit, but much faster)

$$k = \min(1.0, g + 0.545)$$

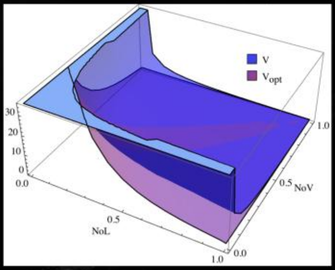
$$V_{opt}(v,h) = \frac{1}{k(v \cdot h)^2 + (1-k)}$$



g = 0.0



g = 0.5



g = 1.0