SIGGRAPH2011

Physically Based Lighting in *Call of Duty: Black Ops*

Dimitar Lazarov, Lead Graphics Engineer, Treyarch

Advances in Real-Time Rendering in Games

SIGGRAPH2011

Agenda

- Physically based lighting and shading
- in the context of evolving Call of Duty’s graphics
- and what lessons we learned

Performance

- Shapes all engine decisions and direction
- Built on two principles
  - Constraints
  - Specialization

Constrained rendering choices

- Forward rendering, 2x MSAA
- Single pass lighting
- All material blending inside the shader
- Almost all transparencies either alpha tested (foliage, fences) or blended but with simple shading (pre-lit particles)

Forward rendering

SIGGRAPH2011

- Forward rendering has traditional issues when it comes to lighting:
  - Exponential shader complexity
  - Multi-pass
  - Wasteful on large meshes
- Unless:

Lighting constraints

SIGGRAPH2011

- One primary light per surface!

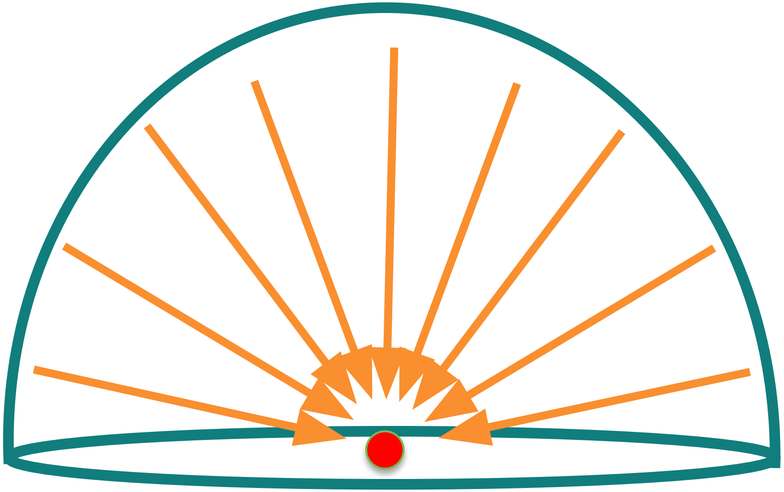
Lighting constraints

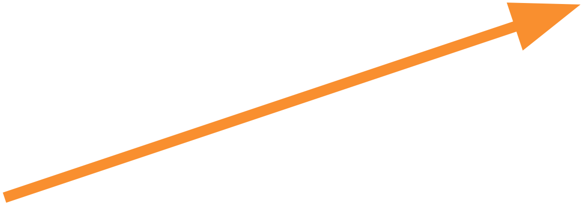
- However:
  - unlimited secondary (baked) lights
  - small number of effect lights per scene:
    - 4 diffuse-only omni lights (gun flashes etc)
    - 1 spot light (flashlight)

Baked lighting

- Performed offline in a custom global illumination (raytracing) tool, stored in three components:
- Lightmaps
- Lightgrid
- Environment Probes

Radiance vs. irradiance





Radiance (L)

Irradiance (E)

$$E = \int_{\Omega} L(\mathbf{l})(\mathbf{n} \cdot \mathbf{l})d\mathbf{l}$$

Run-time lighting

- All Primary lighting is computed in the shader
- A run-time shadowmap per primary overrides the baked shadow in a radius around the camera
- As a result:
  - Primary can change color and intensity, move and rotate to a small extent and still look correct
  - Static and dynamic shadows integrate well together

Run-time lighting: diffuse

- Primary Diffuse
  - Classic Lambert term
  - Modulated by the shadow and the diffuse albedo
- Secondary Diffuse
  - Reconstructed from lightmap/lightgrid secondary irradiance with per-pixel normal, modulated by the diffuse albedo

Run-time lighting: specular

- Primary Specular
  - Microfacet BRDF
  - Modulated by the shadow and the “diffuse” cosine factor
- Secondary Specular
  - Reconstructed from environment probe with per-pixel normal and Fresnel term, also tied to secondary irradiance
  - Based on same BRDF parameters as primary specular

Why Physically-Based

- *Crafting Physically Motivated Shading Models for Game Development* (SIGGRAPH 2010):
  - Easier to achieve photo/hyper realism
  - Consistent look under different lighting conditions
  - Just works - less tweaking and “fudge factors”
  - Simpler material interface for artists
  - Easier to troubleshoot and extend

Why Physically-Based continued

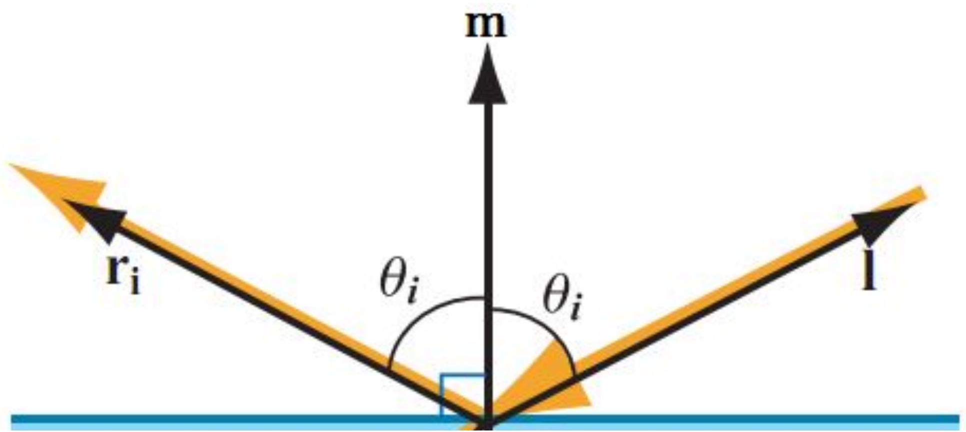
- *Call of Duty: Black Ops* objectives:
  - Maximize the value of the one primary light
  - Improve realism, lighting consistency (move to linear/HDR lighting, improve specular lighting)
  - Simplify authoring (remove per material tweaks for Fresnel, Environment map etc)

Some prerequisites

- Gamma correct pipeline
  - Used gamma 2.0, mix of shader & GPU conversion
- HDR lighting values
  - Limited range (0 to 4), stored in various forms
- Exposure and tone-mapping
  - Art-driven, applied at the end of every shader
  - Filmic curve part of final color LUT

Microfacet theory

- Theory for specular reflection; assumes surface made of *microfacets* – tiny mirrors that reflect incoming light in the mirror direction around the microfacet normal $\mathbf{m}$



The half vector

SIGGRAPH2011

- For given $\mathbf{l}$ and $\mathbf{v}$ vectors, only microfacets which happen to have their surface normal $\mathbf{m}$ oriented exactly halfway between $\mathbf{l}$ and $\mathbf{v}$ ($\mathbf{m} = \mathbf{h}$) reflect any visible light

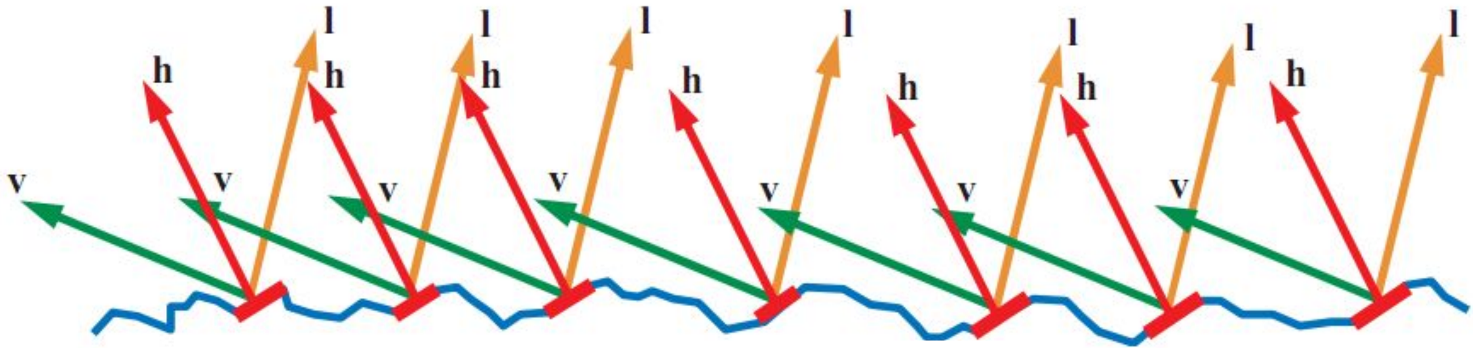
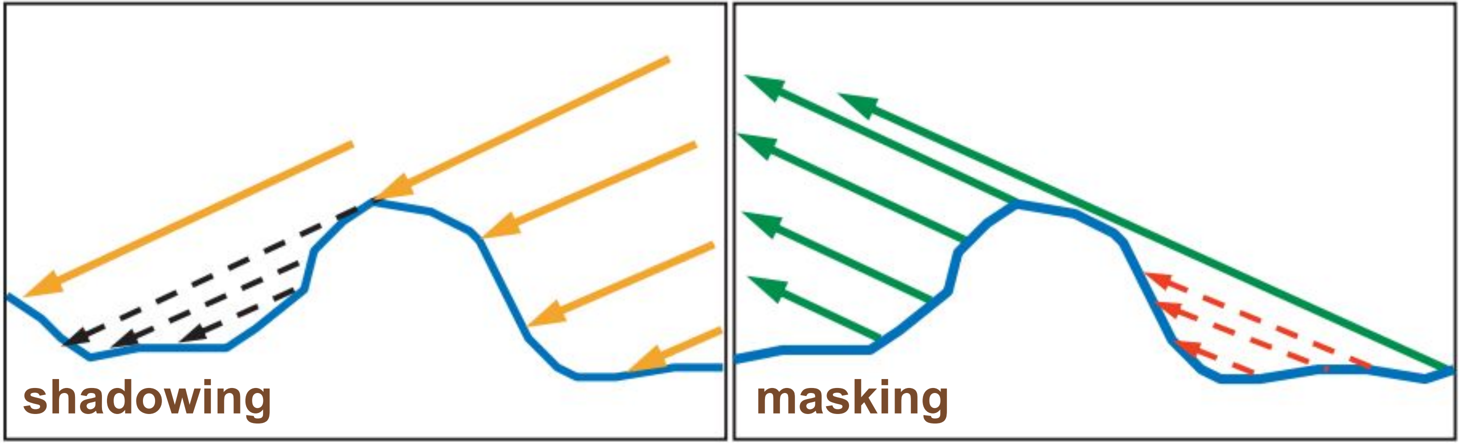


Image from "Real-Time Rendering, 3rd Edition", A K Peters 2008

Shadowing and masking

SIGGRAPH2011

- Not all microfacets with $\mathbf{m} = \mathbf{h}$ contribute; some blocked by other microfacets from $\mathbf{l}$ (*shadowing*) or $\mathbf{v}$ (*masking*)



Images from “Real-Time Rendering, 3rd Edition”, A K Peters 2008

Microfacet BRDF

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

Microfacet BRDF - D

SIGGRAPH2011

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

Microfacet BRDF - F

SIGGRAPH2011

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

Microfacet BRDF - G

SIGGRAPH2011

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

Microfacet BRDF – the rest

SIGGRAPH2011

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

Modular approach

- Early experiments used Cook-Torrance
- We then tried out different options to get a more realistic look and better performance
- Since each part of the BRDF can be chosen separately, we tried out various “lego pieces”

Shading with microfacet BRDF

SIGGRAPH2011

- Useful to factor into three components
  - Distribution function times constant: $\frac{\pi}{4} D(\mathbf{h})$
  - Fresnel: $F(\mathbf{l}, \mathbf{h})$
  - Visibility function:

$$V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{G(\mathbf{l}, \mathbf{v}, \mathbf{h})}{(\underline{\mathbf{n} \cdot \mathbf{l}})(\underline{\mathbf{n} \cdot \mathbf{v}})}$$

Distribution functions

- Beckmann:

$$D_B(\mathbf{h}) = \frac{e^{\left(\frac{(\underline{\mathbf{n}\cdot\mathbf{h}})^2-1}{m^2(\underline{\mathbf{n}\cdot\mathbf{h}})^2}\right)}}{\pi m^2(\underline{\mathbf{n}\cdot\mathbf{h}})^4}$$

- Read roughness m from an LDR texture (range 0 to 1)

Distribution functions continued

SIGGRAPH2011

- Phong lobe NDF (Blinn-Phong):

$$D_P(\mathbf{h}) = \frac{n+2}{2\pi}(\underline{\mathbf{n}\cdot\mathbf{h}})^n$$

- Specular power n in the range (1, 8192)
- Encode log in gloss map: $n = (8192)^g = 2^{(13g)}$

Distribution functions comparison

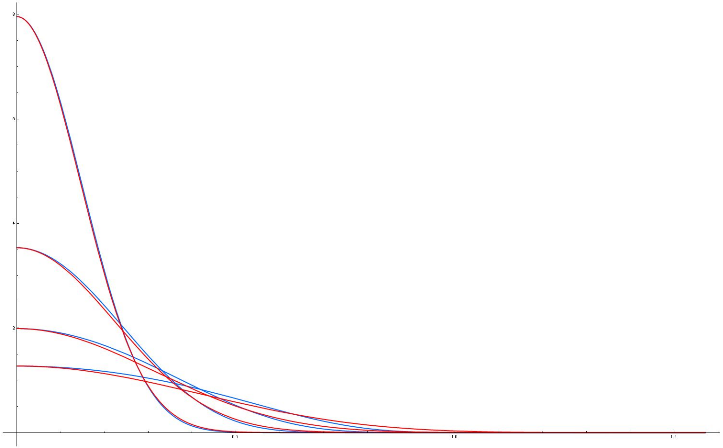
- Beckmann, Phong NDFs very similar in our gloss range
- Blinn-Phong is cheaper to evaluate and the gloss representation seems visually more intuitive
- It is easy to switch between the two if needed:

$$m = \sqrt{\frac{2}{n+2}}$$

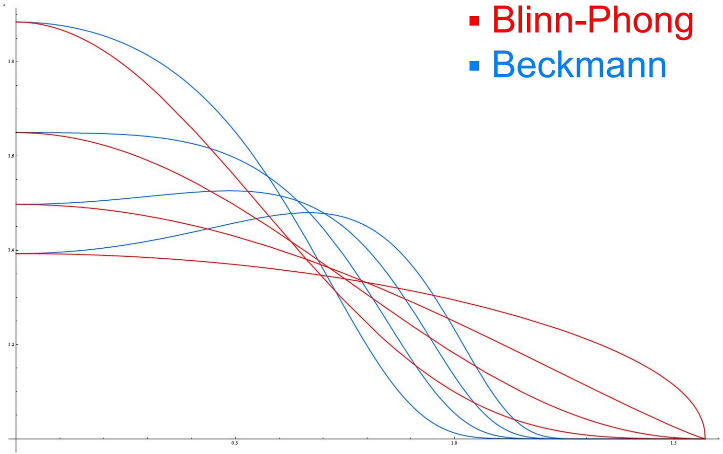
Beckmann Distribution function

Blinn-Phong Distribution function

Distribution functions comparison



m = 0.2, 0.3, 0.4, 0.5



m = 0.6, 0.7, 0.8, 0.9

Fresnel functions

- Schlick's approximation to Fresnel

$$F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, x) = \mathbf{c}_{\text{spec}} + (1 - \mathbf{c}_{\text{spec}})(1 - x)^5$$

  - Original (mirror reflection) definition: x= $(\mathbf{n}\bullet\mathbf{l})$ or $(\mathbf{n}\bullet\mathbf{v})$
  - Microfacet form: x= $(\mathbf{h}\bullet\mathbf{l})$ or $(\mathbf{h}\bullet\mathbf{v})$ (no clamp needed)
  - Better not to have highlight Fresnel at all rather than use the "wrong" mirror form for highlights

No Fresnel

Correct Fresnel

Incorrect Fresnel

Visibility functions

- No visibility function:

$$V(\mathbf{l},\mathbf{v},\mathbf{h}) = \frac{G(\mathbf{l},\mathbf{v},\mathbf{h})}{(\underline{\mathbf{n}\cdot\mathbf{l}})(\underline{\mathbf{n}\cdot\mathbf{v}})} = 1$$

- Shadowing-masking function is effectively:

$$G(\mathbf{l},\mathbf{v},\mathbf{h}) = (\underline{\mathbf{n}\cdot\mathbf{l}})(\underline{\mathbf{n}\cdot\mathbf{v}})$$

Visibility functions continued

- Kelemen-Szirmay-Kalos approximation to Cook-Torrance visibility function:

$$V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{4}{(\mathbf{v} + \mathbf{l}) \cdot (\mathbf{v} + \mathbf{l})}$$

Visibility functions continued

- Schlick's approximation to Smith's Shadowing Function

$$a = m\sqrt{\frac{2}{\pi}} = \frac{1}{\sqrt{\frac{\pi}{4}n + \frac{\pi}{2}}}$$

$$V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{1}{\left((\underline{\mathbf{n} \cdot \mathbf{l}})(1 - a) + a\right)\left((\underline{\mathbf{n} \cdot \mathbf{v}})(1 - a) + a\right)}$$

Visibility functions comparison

SIGGRAPH2011

- Having no Visibility function makes the specular too dark, but costs nothing
- Kelemen-Szirmay-Kalos is too bright and does not account for roughness/gloss, but costs little and is a pretty good approximation to the Cook-Torrence Shadow-Masking function
- Schlick-Smith gives excellent results, albeit costs the most

No Visibility function

SIGGRAPH2011

Schlick-Smith Visibility function

Kelemen Visibility function

Cook-Torrance Visibility function

Schlick-Smith Visibility function

SIGGRAPH2011

Kelemen Visibility function

SIGGRAPH2011

Environment maps

SIGGRAPH2011

- Traditionally we had dozens of environment probes to match lighting conditions
  - Low resolution due to memory constraints
  - Transition issues, specular pops, continuity on large meshes
- For *Black Ops* we wanted to address these issues and also have higher resolution environment maps to match our high specular power

Environment maps: normalization

SIGGRAPH2011

- The solution:
  - Normalize – divide out environment map by average diffuse lighting at the capture point
  - De-normalize – multiply environment map by average diffuse lighting reconstructed per pixel from lightmap/lightgrid

Environment maps: normalization

- The normalization allows environment maps to fit better in different lighting conditions
- Outdoor areas can get away with as little as one environment map
- Indoor areas need more location specific environment maps to capture secondary specular lighting

Environment map: prefiltering

- Mipmaps are prefiltered and generated with AMD/ATI's CubeMapGen
  - HDR angular extent filtering
  - Face edges fixup

Environment maps: blurring

- The mip is selected based on the material gloss

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

- For very glossy surfaces this could cause texture trashing
- Some GPUs have an instruction to get the hardware selected mip

Environment maps: Fresnel

SIGGRAPH2011

- Fresnel is based on the angle between the view/light vector and the surface normal
  - Mirror reflections: surface normal well defined ($\mathbf{n}$)
  - Microfacet highlights: surface normal well defined ($\mathbf{h}$)
  - Glossy reflections: average over many different microfacet normals – which Fresnel to use?

Fresnel for glossy reflections

- A full solution would involve multiple samples from the environment map and BRDF together
- We can't do that, so we fit a cheap curve to the integral of the BRDF over the hemisphere
  - Multiply it by the value read from the prefiltered cube map
  - Isn't only Fresnel, also has the shadowing/masking term

Fresnel for glossy reflections

SIGGRAPH2011

- Environment map "Fresnel"

$$F_{\mathrm{Glossy}}(\mathbf{c}_{\mathrm{spec}}, x) = \mathbf{c}_{\mathrm{spec}} + (1 - \mathbf{c}_{\mathrm{spec}})\frac{(1-x)^5}{4-3g}$$

- In this case x = $(\mathbf{n} \bullet \mathbf{v})$

Environment maps continued

Environment maps continued

Too much specular …

SIGGRAPH2011

Too much specular …

- Initial suspects:
  - Fresnel can boost up the material specular color for both the procedural light and the environment map
  - Any non trivial Visibility function can also amplify the specular color at certain angles

Too much specular …

- The real culprit:
  - Normal map mipping will make large distant surfaces behave like giant mirrors

Normal Variance

- Variance maps can directly encode the lost information from mipping normal maps (see also “LEAN Mapping” from I3D 2010)
- Variance maps need high precision and cost extra to store, read and decode in the shader
- What if we combine them with the gloss maps offline?

Normal Variance continued

SIGGRAPH2011

- Extract projected variance from the normal map, always from the top mip, preferably with a NxN weighted filter:

$$v = \frac{1}{K}\sum_{i}^{K}\left(1 - (\mathbf{n}_i \cdot \overline{\mathbf{n}})^2\right)$$

Normal Variance continued

- Add in the authored gloss, converted to variance:

$$v' = v + \frac{1}{n+1} = v + \frac{1}{(n_{\max})^{g} + 1}$$

Normal Variance continued

SIGGRAPH2011

- Convert variance back to gloss:

$$v'' = \mathbf{clamp}\left(v', \frac{1}{n_{\mathrm{max}} + 1}, \frac{1}{n_{\mathrm{min}} + 1}\right)$$

$$g = \frac{\log\left(\frac{1}{v''} - 1\right)}{\log\left(n_{\mathrm{max}}\right)}$$

Normal Variance continued

- This method solved the majority of our specular intensity issues
- Tends to anti-alias the specular as well
- Minimizes the chance for texture trashing when gloss-controlling the mips of the environment map

Without Variance-to-Gloss

With Variance-to-Gloss

SIGGRAPH2011

Without Variance-to-Gloss

With Variance-to-Gloss

The Art perspective

- Even with all techniques properly implemented the "ease of authoring" still elusive
- Artists had trouble adjusting to the new concepts and the slight loss of (specular) control
- Education and good examples are essential
- Pre-existing notions and workflow need to be re-examined

Diffuse textures

SIGGRAPH2011

- Using amateur photos as diffuse maps no longer works well
- Diffuse textures can and should be carefully calibrated (can be directly captured through cross polarization)
- It takes more effort but it pays off later when lighting "just works"

Specular textures

- Specular maps no longer control the maximum specular effect
- Ambient occlusion maps can control it but they have to be used judiciously
- Specular maps less important than gloss maps

Gloss textures

- Perhaps the most important yet most difficult maps to author
- It takes time to build an intuition on how to paint them. WYSIWYG tools can help tremendously
- It might be possible to directly capture from real surfaces

Special cases

- With Physically Based Shading, material specular color can be roughly separated in two groups:
  - Metals – colored specular above 0.5 linear space
  - Non-metals – monochrome specular between 0.02 and 0.04 linear space
- What if we create a material/shader that takes advantage of this?

Special cases continued

- Pure metal shader
  - No diffuse texture and no diffuse lighting
- “Simple” shader (non-metals)
  - No specular texture (hardcoded to 0.03 in shader)
  - Specular lighting calculations can be scalar instead of vector

Performance

- Physically Based Shading is relatively more expensive (average 10-20% more ALU)
- Using special case shaders helps
- For texture bound shaders the extra ALU cost can be hidden
- Still a good idea to have a fast Lambert shader for select cases

Conclusions

SIGGRAPH2011

- Physically Based Shading is totally worth it! It will make your specular truly “next gen”
- Be prepared to put a decent amount of effort on both the Engineering and Art side to get the benefits
- It is a package deal – difficult or impossible to skip certain parts of the implementation
- Don’t go overboard

Conclusions

Thanks

- Natalya Tatarchuk
- Naty Hoffman
- Paul Edelstein
- The Call of Duty: Black Ops Team

Contact info

SIGGRAPH2011

- Email me at dlazarov@treyarch.com

Bonus slides

Multiple surface bounces

- In reality, blocked light continues to bounce; some will eventually contribute to the BRDF
- Microfacet BRDFs ignore this – assume all blocked light is lost

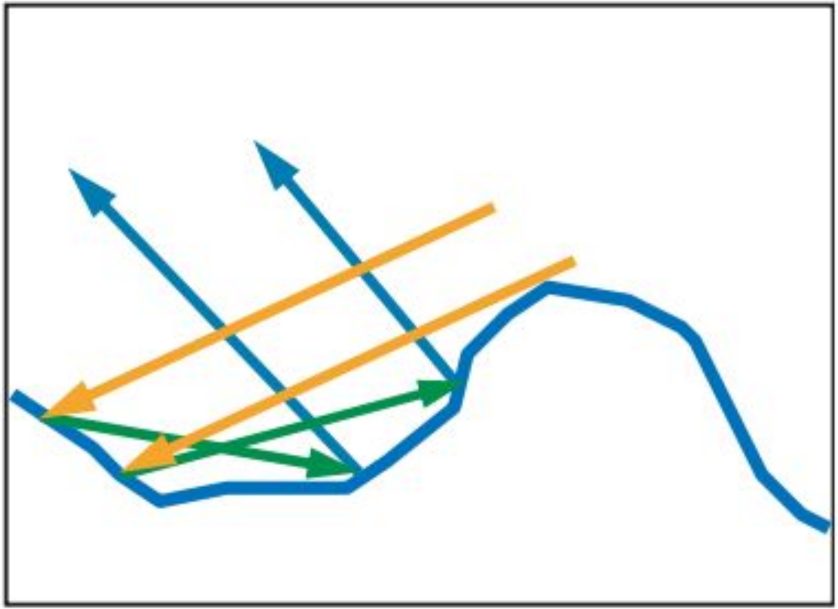


Image from “Real-Time Rendering, 3rd Edition”, A K Peters 2008

Blinn-Phong normalization

- Some games use $(n+8)$ instead of $(n+2)$
- The $(n+8)$ "Hoffman-Sloan" normalization factor first appeared in "Real-Time Rendering, 3rd edition"
- Result of normalizing entire BRDF rather than just NDF
- Compensates for overly dark visibility function
- More accurate to use $(n+2)$ with better visibility function

Ambient Occlusion

- Materials with AO maps can suppress secondary diffuse, primary and secondary specular
- Suppressing primary specular is not entirely correct yet not entirely wrong if we consider AO as microfacet self-shadowing
- AO will mip to below white and compensate (somewhat) against the normal map mipping

Primary lighting selection

SIGGRAPH2011

- Static world surfaces (BSP) are split offline to resolve primary lighting conflicts
- Static objects pick a primary based on their (adjustable) lighting origin
- Dynamic objects pick a primary every time they move
- Other lighting (direct from secondary light sources and indirect bounce from primary & secondary) is baked

BSP

BSP + static objects

BSP + static and dynamic objects

Metalness method

SIGGRAPH2011

- Two textures: color and metalness
- If metalness is 1 then color is treated as specular color and diffuse color is assumed to be black
- If metalness is 0 then color is treated as diffuse color and specular color is assumed to be 0.03 linear
- This works for non binary values of metalness as well

Metalness method continued

- Great idea, but it doesn’t work well in practice
- Artists will have hard time figuring out the concept
- The resulting shader will actually be more expensive than a traditional shader
- There is no storage advantage when textures are DXT compressed
- No advantage when using forward rendering either