Crafting Physically Motivated Shading Models for Game Development

Naty Hoffman

Activision

Talk Outline

- Motivation and Infrastructure
- Making an Ad-hoc Game Shading Model Physically Plausible
- Environmental And Ambient Light
- Fine-Tuning and Future Directions

Previous Talk

- Covered some similar ground, but this talk goes at it from a different angle, with slightly different results
- Interesting to see different approaches to physically-based shading

Motivation and Infrastructure

Why Physically-Based?

- Easier to achieve photorealism / hyperrealism
- Consistent under lighting and viewing changes
- Less tweaking and “fudge factors”
- Simpler material interface for artists
- Easier to troubleshoot
- Easier to extend

Infrastructure

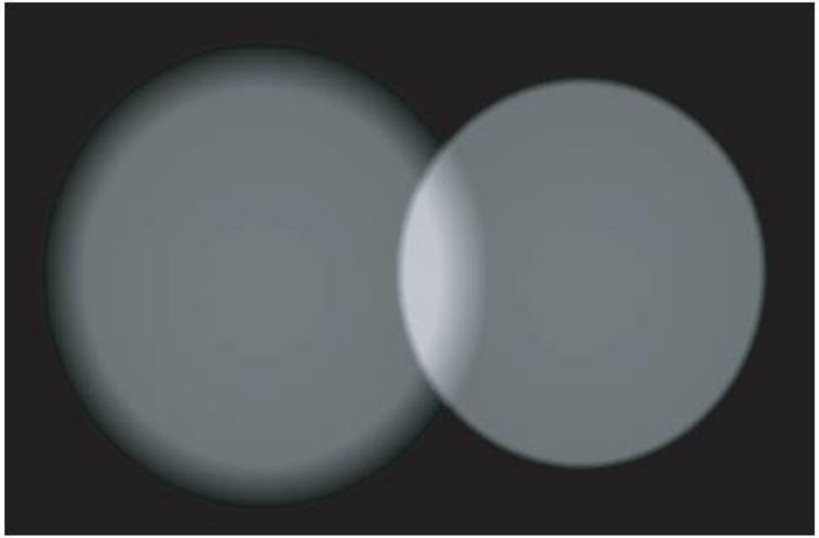
- To get the most benefit from physically-based shaders, your game first needs the basics:
  - Gamma-correct rendering
  - Support for HDR values
  - Good tone mapping (ideally filmic; see course on Tuesday, *Color Enhancement and Rendering for Film and Game Production*)

Gamma-Correct Rendering

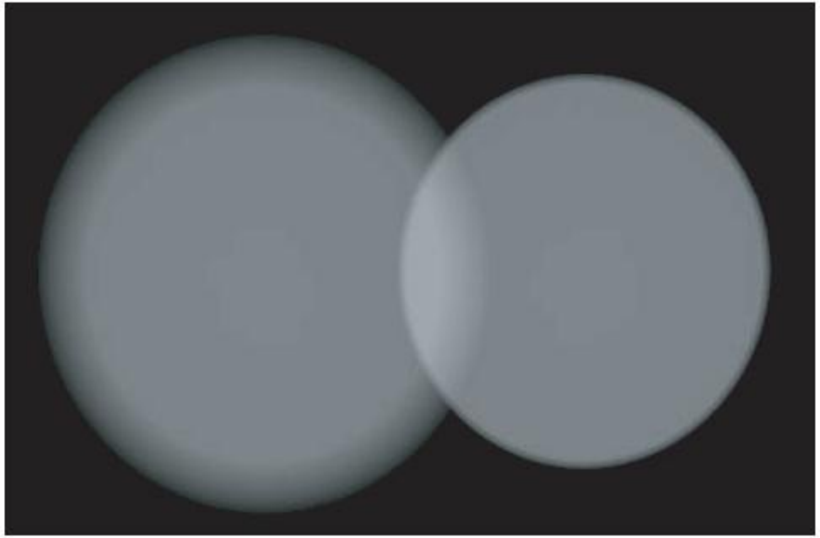
- Shading inputs (textures, light colors, vertex colors, etc.) naturally authored, previewed and (often) stored with nonlinear (gamma) encoding
- Final frame buffer also uses nonlinear encoding
- This is done for good reasons
  - Perceptually uniform(ish) = efficient use of bits
  - Legacy reasons (tools, file formats, hardware)

Shading Defaults to Gamma Space

- Incorrect; yields “1+1=3” effects



Adding lights in gamma space



Adding lights in linear space

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

High Dynamic Range (HDR) Values

- Realistic rendering requires handling values much higher than display white (1.0)
- Before shading: light intensities, lightmaps, environment maps
- Shading produces highlights that affect bloom, fog, DoF, motion blur, etc.
- Cheap solutions exist; details in course notes

Making an Ad-hoc Game Shading Model Physically Plausible

History

- Fixed-function HW shading was used in games before programmable GPU shaders
- Developers, accustomed to the fixed models, used them as a starting point for more complex shaders enabled by newer hardware

Common Game Shading Model

- Straightforward Phong
- Equation for single punctual light (game will have multiple lights, ambient, envmaps, etc.)

$$L_o(\mathrm{v}) = \left(\mathbf{c}_{\mathrm{diff}} \underline{(\mathbf{n} \cdot \mathbf{l}_{\mathbf{c}})} + \begin{cases} \mathbf{c}_{\mathrm{spec}} \underline{(\mathbf{r}_{\mathbf{v}} \cdot \mathbf{l}_{\mathbf{c}})}^{\alpha_p}, & \text{if } (\mathbf{n} \cdot \mathbf{l}_{\mathbf{c}}) > 0 \\ 0, & \text{otherwise} \end{cases} \right) \otimes \mathbf{c}_{\mathrm{light}}$$

- Underbar denotes clamping to 0: $\underline{x} = \max(x, 0)$

First, Remove Conditional

- Intended to remove specular when light below the surface
- But conditional doesn't make physical sense and (more importantly) can introduce artifacts

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

Multiply Specular by Cosine Term

- This makes sense since this cosine term is not part of the BRDF, but of the rendering equation
- Punctual light equation from background talk:

$$L_o(\mathrm{v}) = \pi f(\mathbf{l_c}, \mathbf{v}) \otimes \mathbf{c}_{\text{light}} \underline{(\mathbf{n} \cdot \mathbf{l_c})}$$

- We'll skip the "$L_o(\mathrm{v})=$" part from now on

Multiply Specular by Cosine Term

- Simpler than conditional, faster, no artifacts

$$\left(\mathbf{c}_{\mathrm{diff}} + \mathbf{c}_{\mathrm{spec}}\underline{(\mathbf{r_v} \cdot \mathbf{l_c})}^{\alpha_p}\right) \otimes \mathbf{c}_{\mathrm{light}}\underline{(\mathbf{n} \cdot \mathbf{l_c})}$$

- From here, we'll focus only on the specular term

What's With This Reflection Vector?

- Specular doesn't look like microfacet theory – what is the physical meaning of $(\mathbf{r_v} \bullet \mathbf{l_c})$?
- Blinn-Phong is very similar, but uses the more physically meaningful half-vector – recall:

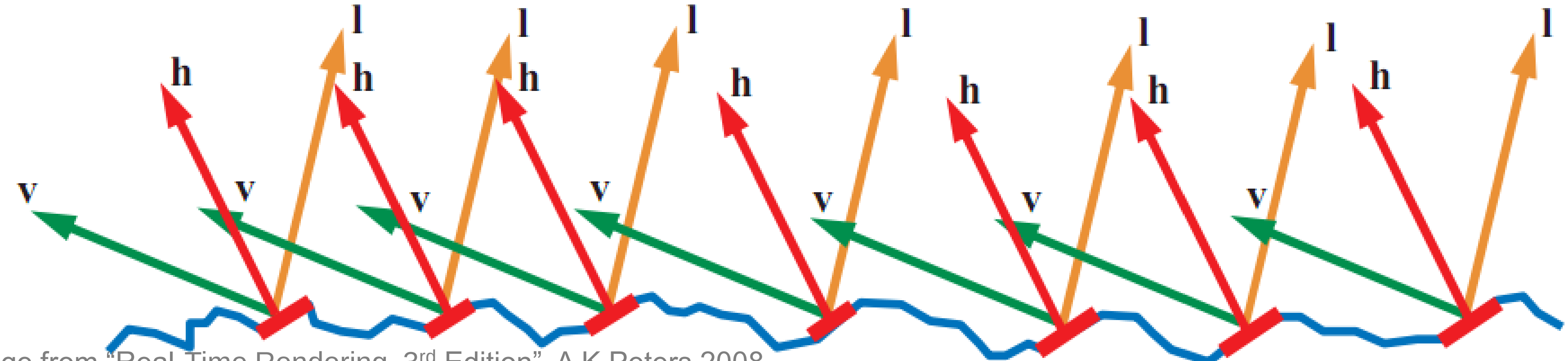


Image from "Real-Time Rendering, 3rd Edition", A K Peters 2008

Change to Blinn-Phong

- It makes more physical sense, but is it better?

$$\underline{(\mathbf{n} \cdot \mathbf{h})}^{\alpha_p} \mathbf{c}_{\text{spec}} \otimes \mathbf{c}_{\text{light}} \underline{(\mathbf{n} \cdot \mathbf{l}_\mathbf{c})}$$

Visual Comparison

Phong | **Blinn-Phong**

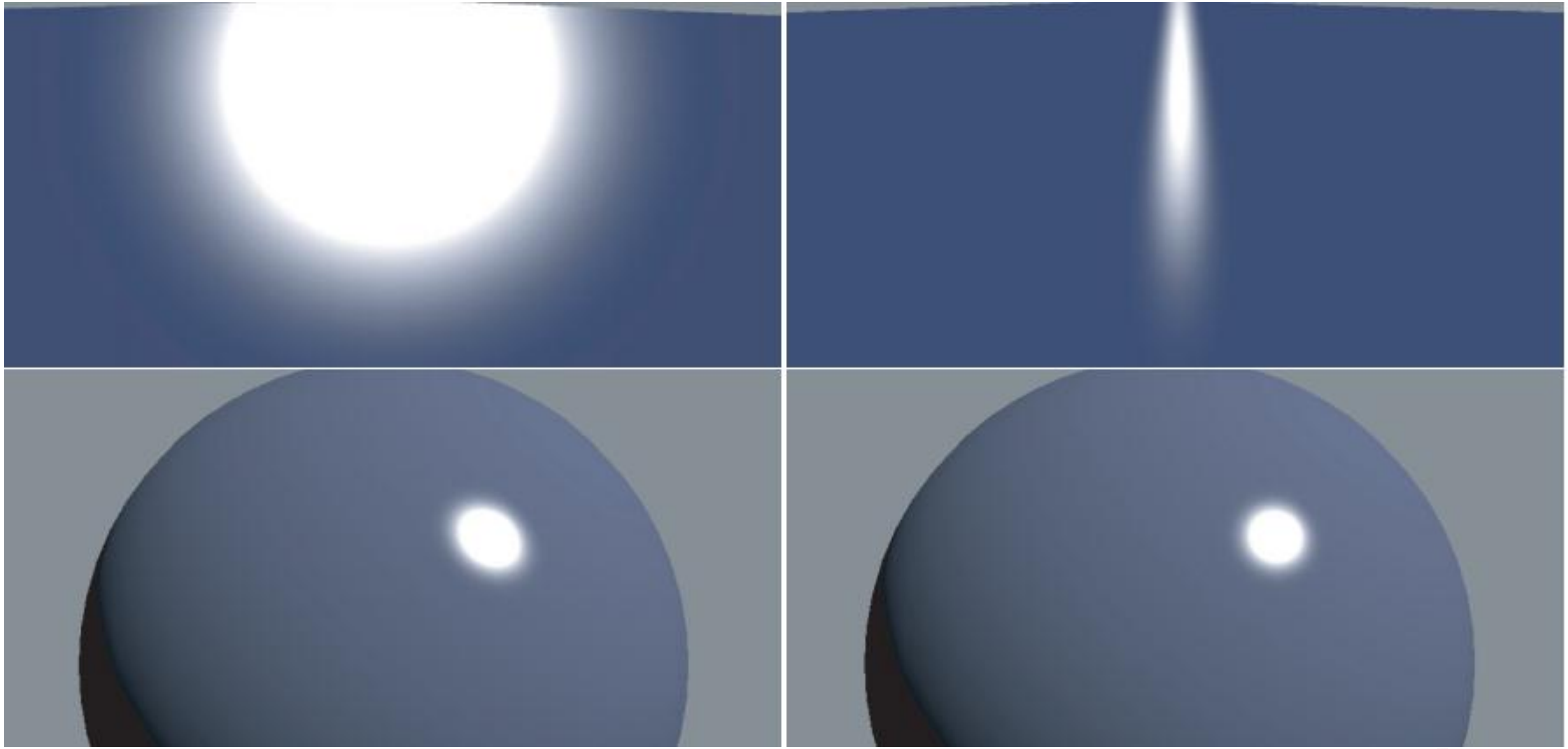


Image from "Real-Time Rendering, 3rd Edition", A K Peters 2008

Visual Comparison

- The two look close for round objects, but for lights glancing off flat surfaces like floors, they are very different
  - Phong has a round highlight
  - Blinn-Phong has a stretched highlight
- Which is more realistic?

Blinn-Phong is the Clear Winner

Image from “Real-Time Rendering, 3rd Edition”, A K Peters 2008; photographer: Elan Ruskin

More Microfacet Theory

- Applying a little bit of microfacet theory was a win, let’s try some more.
- Let’s compare our specular equation to that for a microfacet BRDF lit by a punctual light

Comparing to Microfacet BRDF

$$\pi \frac{D(\mathbf{h})G(\mathbf{l_c}, \mathbf{v}, \mathbf{h})}{4\underline{(\mathbf{n}\cdot\mathbf{l_c})}\,\underline{(\mathbf{n}\cdot\mathbf{v})}} F(\mathbf{l_c}, \mathbf{h}) \otimes \mathbf{c}_{\text{light}}\underline{(\mathbf{n}\cdot\mathbf{l_c})}$$

$$\underline{(\mathbf{n}\cdot\mathbf{h})}^{\alpha_p}\mathbf{c}_{\text{spec}} \otimes \mathbf{c}_{\text{light}}\underline{(\mathbf{n}\cdot\mathbf{l_c})}$$

Converting to a Simple Microfacet BRDF

- Correctly normalized, the cosine power term becomes a microfacet distribution:

$$D(\mathbf{m}) = \frac{\alpha_p + 2}{2\pi} \underline{(\mathbf{n} \cdot \mathbf{m})^{\alpha_p}}$$

Converting to a Simple Microfacet BRDF

- Then replace $\mathbf{c}_{\text{spec}}$ with $F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, \mathbf{l}, \mathbf{h})$
- Last talk detailed advantages of correct Fresnel
- Some ways Fresnel is incorrectly used in games
  - Darkening specular color towards middle rather than interpolating it to white on edges
  - Using the wrong angle ($\mathbf{n}\bullet\mathbf{v}$ instead of $\mathbf{l}\bullet\mathbf{h}$)
  - Adding parameters instead of just using $\mathbf{c}_{\text{spec}}$

What About Remaining Term?

- Geometry (shadowing / masking) term divided by foreshortening factors
- We call these combined terms the *visibility term*

$$\frac{G(\mathbf{l_c}, \mathbf{v}, \mathbf{h})}{\underline{(\mathbf{n} \cdot \mathbf{l_c})}\,\underline{(\mathbf{n} \cdot \mathbf{v})}}$$

Simplest Possible Visibility Term

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

- Equivalent to:

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

- Which is a plausible shadowing / masking term

Resulting Microfacet Shading Model

$$\frac{\alpha_p + 2}{8}\underline{(\mathbf{n}\cdot\mathbf{h})^{\alpha_p}}F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, \mathbf{l}_{\mathbf{c}}, \mathbf{h}) \otimes \mathbf{c}_{\text{light}}\underline{(\mathbf{n}\cdot\mathbf{l}_{\mathbf{c}})}$$

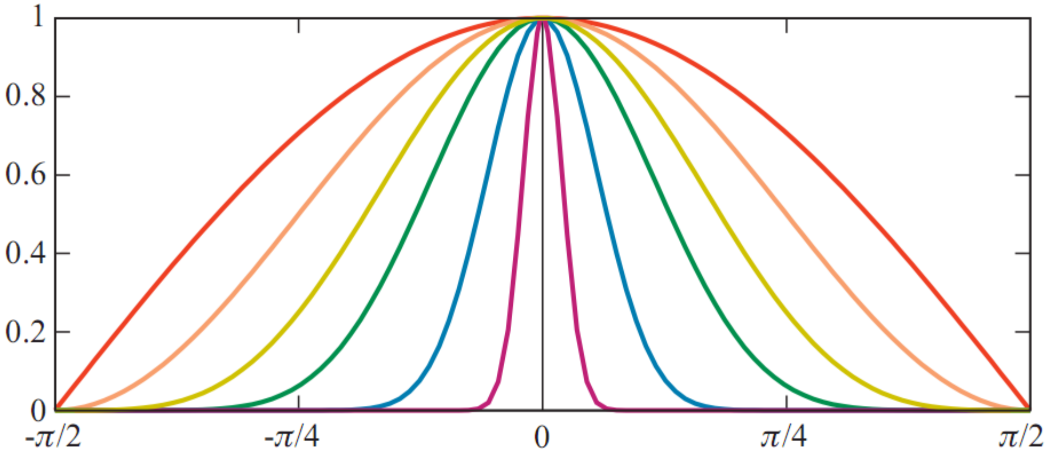
- Besides the Fresnel term (which advantages have been discussed) the primary difference is the $(\alpha_p+2)/8$ normalization factor

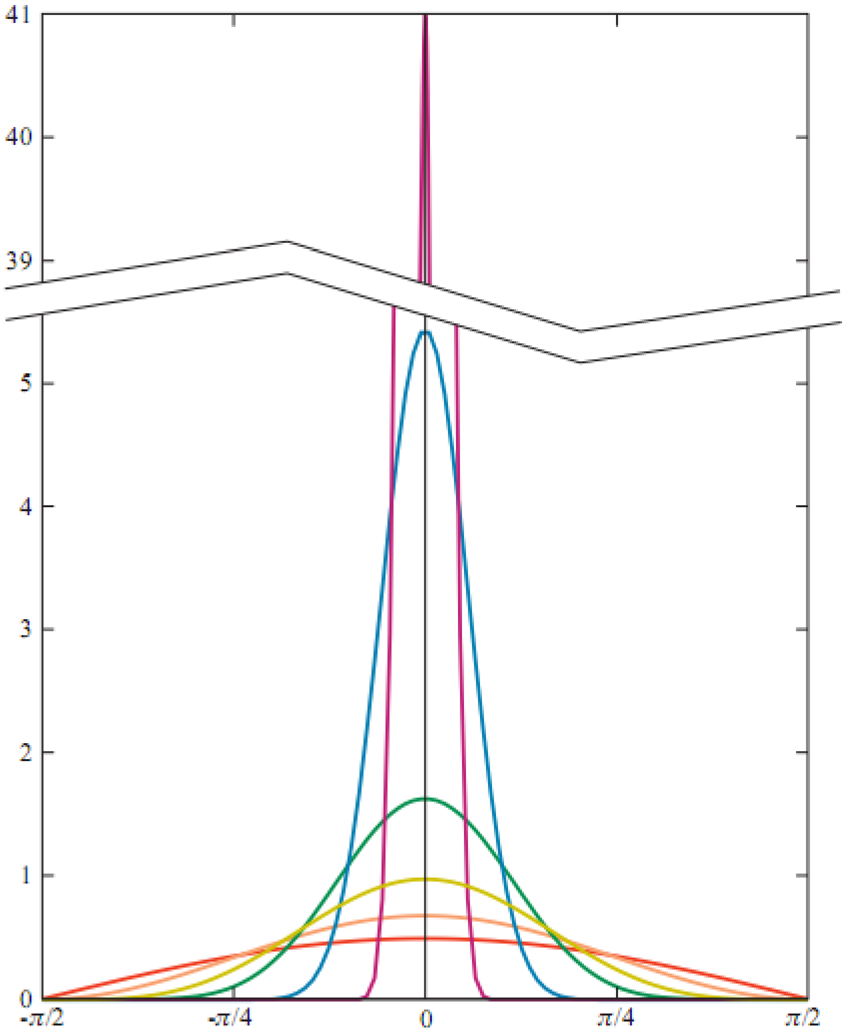
Normalization Factor Hugely Important

- Without it, specular brightness is anywhere from 4 times too bright to thousands of times too dark, depending on the value of α_p
  - Error so large, Fresnel factor becomes irrelevant
- No normalization makes it very hard to create realistic-looking materials, especially when α_p varies per-pixel

Normalized vs. Original

- (not to same scale – note y-axis)

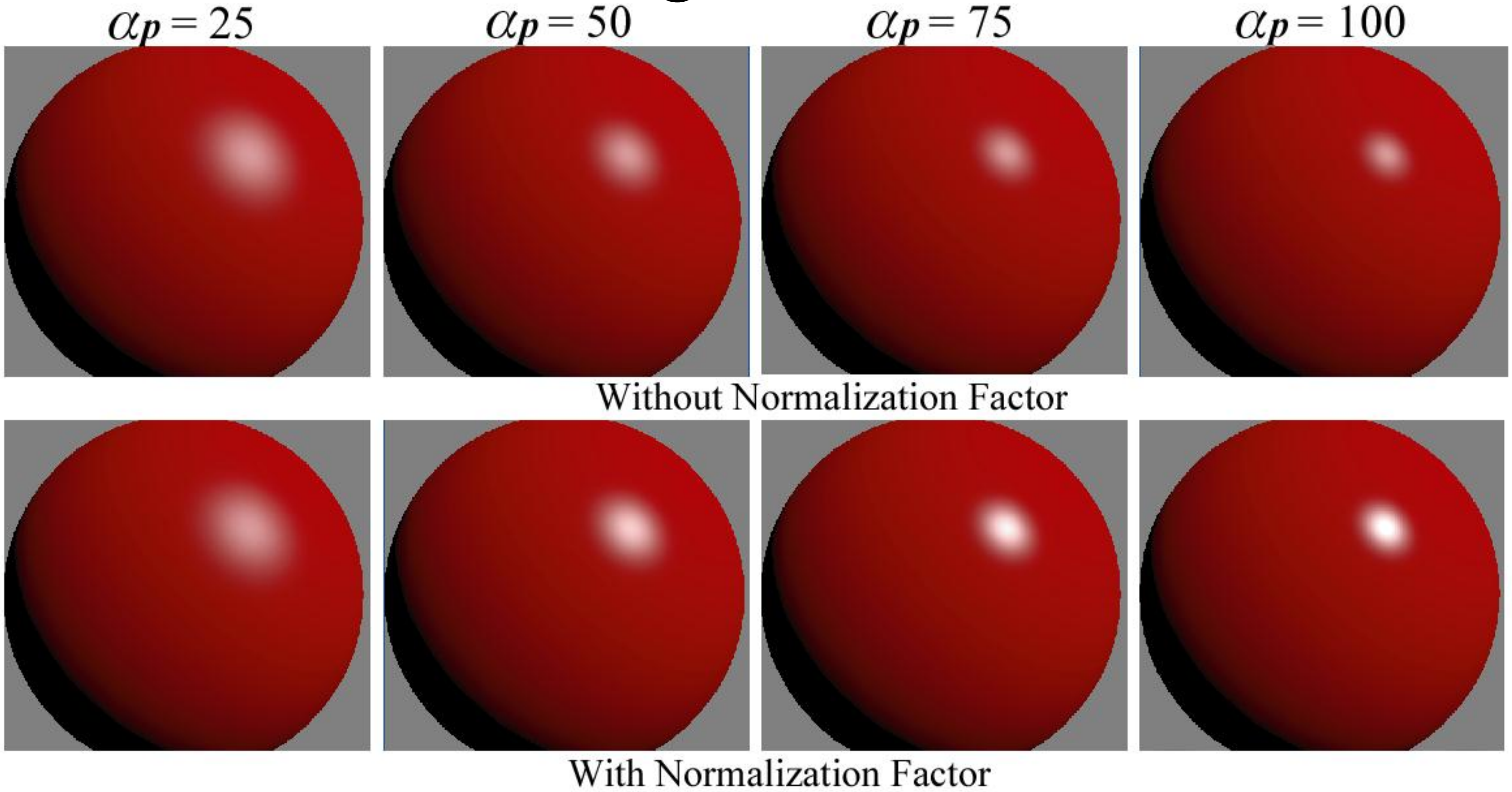




Images from “Real-Time Rendering, 3rd Edition”, A K Peters 2008

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

Normalized vs. Original



Normalization: Better Material Interface

- Normalization clearly separates surface <u>substance</u> ($\mathbf{c}_{\mathrm{spec}}$) from <u>roughness</u> (α_p)
- Per-pixel roughness in a texture map is a very effective way to vary surface appearance
  – Roughness varies highlight width and intensity, as opposed to just width as in non-normalized shader
- Can use real-world $F(0°)$ values for $\mathbf{c}_{\mathrm{spec}}$

Normalization: Better Material Interface

- From the $F(0°)$ tables earlier in the course, recall that the vast majority of real-world materials (anything not metal or gems) have $F(0°)$ values in a very narrow range (~0.02 - 0.06)
- Changes in roughness will be far more noticeable, so for many materials you can just set $\mathbf{c}_{\mathrm{spec}}$ to a constant value (around 0.04)

Normalization: Better Material Interface

- For "advanced" materials with exposed metal, artists should take care in painting $\mathbf{c}_{\text{spec}}$ values
  - As pointed out in the previous talk, easy to get wrong
  - Artists should refer to tables of known values
- No such thing as "no specular"
  - "Matte" surfaces: $\mathbf{c}_{\text{spec}} \approx$ 0.02 - 0.06 , $\alpha_p \approx$ 0.1 - 2.
  - At glancing angles, all "matte" surfaces have specular

Roughness Map

- All surfaces should have roughness maps with small-scale detail from scratches, wear, etc.
  - Closely tied to normal map
  - For best results, stores a nonlinear function of specular power; e.g. $\alpha_p = (\alpha_{max})^s$ where s is a 0-1 value read from the texture

Environmental And Ambient Light

(Cube) Environment Maps

- Very important when using physical reflectance, especially for metals
  - Consider having them on everything

Environment Map Content

- Don’t need to be exact reflections
  - Exception: player’s car in racing game
- Do need same average RGB as diffuse ambient
  - Can ensure this by “normalizing” envmaps in tools (dividing them by their average) and later multiplying by average diffuse ambient

Shading With Environment Maps

- Specular: same color, different Fresnel term
  - $F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, \mathbf{v}, \mathbf{n})$
  - instead of $F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, \mathbf{l}, \mathbf{h})$ (or $F_{\text{Schlick}}(\mathbf{c}_{\text{spec}}, \mathbf{v}, \mathbf{h})$; same)
- Diffuse: prefilter into:
  - Separate lowres env map
  - Bottom MIP of env map
  - Spherical Harmonics coefficients

Use Roughness Values to Blur Envmap

- Preblur (using full HDR values) when generating MIPs (use ATI's CubeMapGen library)
- At runtime, select LOD based on roughness
- Very effective combined w. per-pixel roughness

Use Roughness Values to Blur Envmap

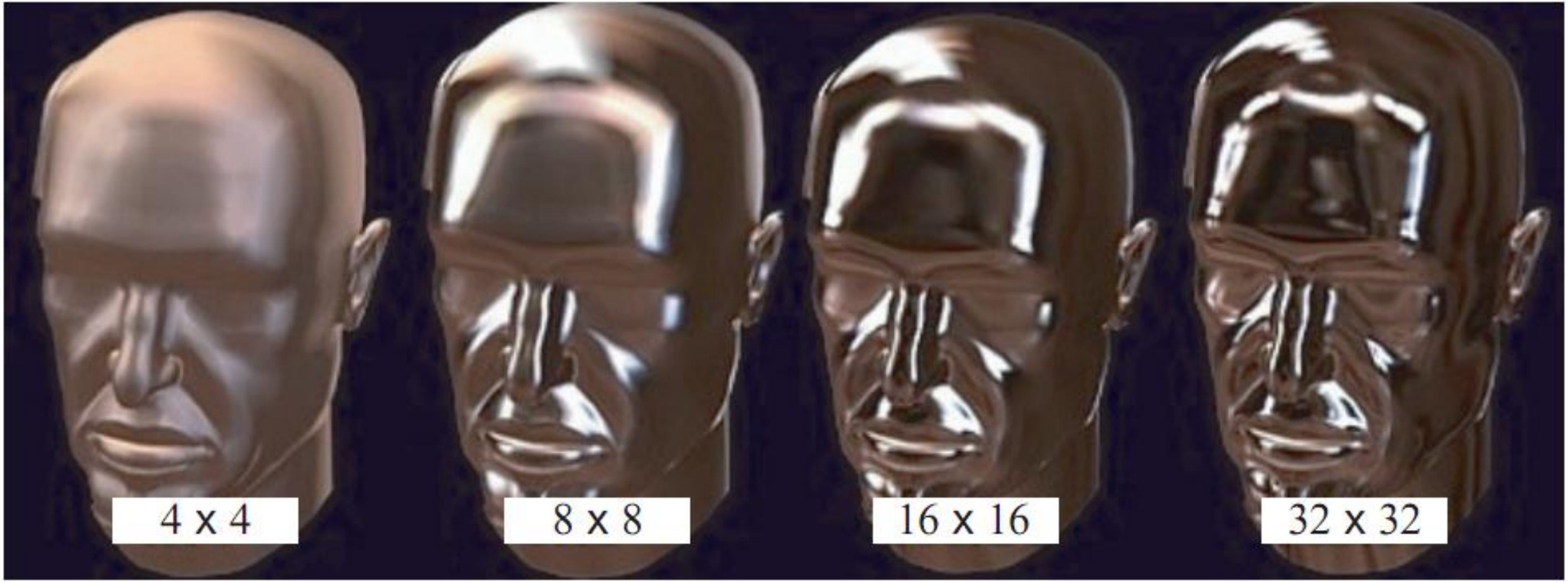


Image from “Real-Time Rendering, 3rd Edition”, A K Peters 2008 (CubeMapGen image used with permission from AMD)

Specular Shading with Ambient / SH

- I‘ve only done diffuse SH myself
- The previous talk described a good method for arbitrary BRDFs with ambient
- See also Bungie’s presentation, *Lighting and Material of Halo 3*

Fine-Tuning and Future Directions

Overbright Specular

- When switching over to more correct models, you will often hear complaints about the specular now being too bright
- Two main reasons:
  - Fresnel defeating bump occlusion
  - Overdark diffuse + overexposure

Fresnel Defeating Bump Occlusion

- Few engines have bump self-shadowing support, so some occlusion often painted into specular and diffuse color maps
- But Fresnel will brighten the darkest specular color at glancing angles
- Causing bright highlights from within deep cracks

Ambient Occlusion Textures

- If you have a separate occlusion map, apply this to specular after Fresnel
- Yeah, it’s not correct to apply AO to direct lighting, but in this case it’s better than the alternative
- You might want to reduce AO contrast when using it for this purpose

Overdark Diffuse Colors

- I used to think you could just eyeball the diffuse colors, but experience taught me otherwise
- If you are not careful, easy for material artists to make diffuse colors too dark, lighting artists overexpose to compensate, and carefully tuned physically correct specular looks too bright

Correct Exposure

- HDR exposure should be set using well-known principles like the Ansel Adams zone system
- Basically a lit diffuse white surface should expose to a little under full white
  - Leave some room for specular highlights

Ensuring Correct Diffuse

- Calibrate photo reference (divide out lighting)
- For stuff painted from scratch make sure artists are viewing textures as they will be displayed in the game
  - See Sony Pictures Imageworks' OpenColorIO project for relevant workflow examples (there is a "Birds of a Feather" session on Wednesday, also mentioned in *Color Enhancement and Rendering for Film and Game Production* course on Tuesday)

Unsolved Problems / Future Work

- Fresnel term for prefiltered envmaps
  - Need to integrate over a range of microfacet normals
- Tiny punctual highlights on smooth surfaces
  - Need to account for light size somehow
  - Perhaps cheap version of ILM's solution?
- "Blinn-Phong-style" reflections from envmaps
- Try out more Geometry terms

Acknowledgements

- A K Peters for permission to use RTR3 images
- Paul Edelstein, Yoshiharu Gotanda and Dimitar Lazarov for thought-provoking discussions
- Elan Ruskin for photographs
- AMD for CubeMapGen image

Backup Slides

Gamma-Correct Rendering Details

In Theory, Just Need To:

- Convert shader inputs to linear before shading
- Convert shader output to gamma at end
- “Free” (pre-convert constants & vertex colors, HW converts from textures / to frame buffer)
- In practice this works if you never do shading operations in the frame buffer

Complications

- Some HW does gamma blending incorrectly
  - Bad for multipass / deferred shading, transparencies
- Some HW filters gamma textures incorrectly
  - But you can at least generate MIP maps the right way
- Actual nonlinear space supported by HW varies
  - Especially bad for consoles

Unintended Consequences

- Changes light distance falloff, Lambert falloff, soft shadow edges, vertex interpolation, etc.
  - May require artist adjustment / retraining
  - In some cases (like vertex interpolation) it might make sense to fix in the shader

High Dynamic Range Details

HDR Values – Lightmaps & Envmaps

- HDR, but don't need huge range, precision
  - With careful management of lighting and exposure, don't need more than about 25-100X display white
  - In gamma space this is just ~4-8, can scale and store in 10/10/10 textures (8/8/8/8 or even DXT in a pinch)
- Artist exposure control often works better than automatic approaches

HDR Values – Shader Outputs

- In simple case (opaque objects, no multipass or deferred rendering) tone map at end of shader
  - Many benefits of HDR w/o HDR frame buffer
  - But post effects don't account for HDR
  - Transparent objects also incorrect
- Or use one of many HDR encoding options
  - fp16, fp11/11/10, RGBE/M, LogLuv, logRGB, etc.

Environment Map Details

Environment Map Range

- Since $\mathbf{c}_{\mathrm{spec}} \geq$ 0.02-0.05, envmap goes to 20-50X display white before saturating (more for bloom)
- If you're doing the "normalization" trick from the last slide, you may need a bit more range since diffuse ambient may darken it
- In gamma space this reduces to ~4-6X display white, LDR formats with scaling work fine

Selecting Envmap MIP

- If $\alpha_p = (\alpha_{max})^s$, making desired MIP level a linear function of s works well
  - Validate: "one superbright pixel" envmap vs. highlight
  - Important for highlights and envmap to be similarly blurry across the roughness range

Selecting Envmap MIP

- Compare desired MIP to the automatic MIP level and choose the lower-resolution of the two
- How does shader know automatic MIP level?
  - XB360 (and I think newer D3D): has instruction
  - Others: store MIP level in cubemap, do extra read
    - Separate one-channel cubemap (same resolution)
    - Or in alpha of environment map (can then use extra RGB for “double reflection” effect, e.g. metallic car paint)

Geometry Factor Details

Other Geometry Factors

- The "implicit geometry factor" $(\mathbf{n} \bullet \mathbf{l})(\mathbf{n} \bullet \mathbf{v})$ goes to zero too quickly compared to real materials
  - Causes edge reflections to be slightly too dark
- Often not a problem in practice; if you want more accurate reflections there are a few options

Kelemen-Szirmay-Kalos Geometry Factor

- A very cheap approximation to the entire Cook-Torrance visibility term:

$$\frac{G_{\mathrm{CT}}(\mathbf{l_c}, \mathbf{v}, \mathbf{h})}{\underline{(\mathbf{n} \cdot \mathbf{l_c})}\;\underline{(\mathbf{n} \cdot \mathbf{v})}} \approx \frac{1}{(\mathbf{l_c} \cdot \mathbf{h})^2}$$

- Just divide by square of the same dot product you need to compute for Schlick anyway

Smith Shadowing Term

- More correct in principle than Cook-Torrance, since it takes account of surface roughness
- The approximation in RTR3 is not the right one – The paper *Microfacet Models for Refraction through Rough Surfaces* has a more correct one
  - I haven’t used it, but Imageworks has; Adam Martinez’s talk later on will discuss it