Real Shading in Unreal Engine 4

by Brian Karis, Epic Games

Figure 1: UE4: Infiltrator demo

Introduction

About a year ago, we decided to invest some time in improving our shading model and embrace a more physically based material workflow. This was driven partly by a desire to render more realistic images, but we were also interested in what we could achieve through a more physically based approach to material creation and the use of material layering. The artists felt that this would be an enormous improvement to workflow and quality, and I had already seen these benefits first hand at another studio, where we had transitioned to material layers that were composited offline. One of our technical artists here at Epic experimented with doing the layering in the shader with promising enough results that this became an additional requirement.

In order to support this direction, we knew that material layering needed to be simple and efficient. With perfect timing came Disney's presentation [2] concerning their physically based shading and material model used for *Wreck-It Ralph*. Brent Burley demonstrated that a very small set of material parameters could be sophisticated enough for offline feature film rendering. He also showed that a fairly practical shading model could closely fit most sampled materials. Their work became an inspiration and basis for ours, and like their "principles," we decided to define goals for our own system:

Real-Time Performance

- First and foremost, it needs to be efficient to use with many lights visible at a time.

Reduced Complexity

- There should be as few parameters as possible. A large array of parameters either results in decision paralysis, trial and error, or interconnected properties that require many values to be changed for a single intended effect.

- We need to be able to use image-based lighting and analytic light sources interchangeably, so parameters must behave consistently across all light types.

Intuitive Interface

- We prefer simple-to-understand values, as opposed to physical ones such as index of refraction.

Perceptually Linear

- We wish to support layering through masks, but we can only afford to shade once per pixel. This means that parameter-blended shading must match blending of the shaded results as closely as possible.

Easy to Master

- We would like to avoid the need for technical understanding of dielectrics and conductors, as well as minimize the effort required to create basic physically plausible materials.

Robust

- It should be difficult to mistakenly create physically implausible materials.

- All combinations of parameters should be as robust and plausible as possible.

Expressive

- Deferred shading limits the number of shading models we can have, so our base shading model needs to be descriptive enough to cover 99% of the materials that occur in the real world.

- All layerable materials need to share the same set of parameters in order to blend between them.

Flexible

- Other projects and licensees may not share the same goal of photorealism, so it needs to be flexible enough to enable *non*-photorealistic rendering.

Shading Model

Diffuse BRDF

We evaluated Burley's diffuse model but saw only minor differences compared to Lambertian diffuse (Equation 1), so we couldn't justify the extra cost. In addition, any more sophisticated diffuse model would be difficult to use efficiently with image-based or spherical harmonic lighting. As a result, we didn't invest much effort in evaluating other choices.

$$f(\mathbf{l}; \mathbf{v}) = \frac{\mathbf{c}_{diff}}{\pi} \tag{1}$$

Where $\mathbf{c}_{diff}$ is the diffuse albedo of the material.

Microfacet Specular BRDF

The general Cook-Torrance [5, 6] microfacet specular shading model is:

$$f(\mathbf{l}; \mathbf{v}) = \frac{D(\mathbf{h})\, F(\mathbf{v}; \mathbf{h})\, G(\mathbf{l}; \mathbf{v}; \mathbf{h})}{4\,(\mathbf{n} \cdot \mathbf{l})\,(\mathbf{n} \cdot \mathbf{v})} \tag{2}$$

See [9] in this course for extensive details.

We started with Disney's model and evaluated the importance of each term compared with more efficient alternatives. This was more difficult than it sounds; published formulas for each term don't necessarily use the same input parameters which is vital for correct comparison.

Specular D

For the normal distribution function (NDF), we found Disney's choice of GGX/Trowbridge-Reitz to be well worth the cost. The additional expense over using Blinn-Phong is fairly small, and the distinct, natural appearance produced by the longer "tail" appealed to our artists. We also adopted Disney's reparameterization of $\alpha = \text{Roughness}^2$.

$$D(\mathbf{h}) = \frac{\alpha^2}{\pi((\mathbf{n} \cdot \mathbf{h})^2(\alpha^2 - 1) + 1)^2} \tag{3}$$

Specular G

We evaluated more options for the specular geometric attenuation term than any other. In the end, we chose to use the Schlick model [19], but with $k = \alpha/2$, so as to better fit the Smith model for GGX [21]. With this modification, the Schlick model exactly matches Smith for $\alpha = 1$ and is a fairly close approximation over the range $[0, 1]$ (shown in Figure 2). We also chose to use Disney's modification to reduce "hotness" by remapping roughness using $\frac{\text{Roughness}+1}{2}$ before squaring. It's important to note that this adjustment is only used for analytic light sources; if applied to image-based lighting, the results at glancing angles will be much too dark.

$$k = \frac{(\text{Roughness}+1)^2}{8}$$

$$G_1(\mathbf{v}) = \frac{\mathbf{n} \cdot \mathbf{v}}{(\mathbf{n} \cdot \mathbf{v})(1 - k) + k}$$

$$G(\mathbf{l}; \mathbf{v}; \mathbf{h}) = G_1(\mathbf{l})\, G_1(\mathbf{v}) \tag{4}$$

Specular F

For Fresnel, we made the typical choice of using Schlick's approximation [19], but with a minor modification: we use a *Spherical Gaussian* approximation [10] to replace the power. It is slightly more efficient to calculate and the difference is imperceptible. The formula is:

$$F(\mathbf{v}; \mathbf{h}) = F_0 + (1 - F_0)\,2^{(-5.55473(\mathbf{v} \cdot \mathbf{h}) - 6.98316)(\mathbf{v} \cdot \mathbf{h})} \tag{5}$$

Where F_0 is the specular reflectance at normal incidence.

Figure 2: Schlick with $k = \alpha/2$ matches Smith very closely

Image-Based Lighting

To use this shading model with image-based lighting, the radiance integral needs to be solved, which is often done using importance sampling. The following equation describes this numerical integration:

$$\int_H L_i(l) f(l; v) \cos\theta_l \, dl \approx \frac{1}{N} \sum_{k=1}^{N} \frac{L_i(l_k) f(l_k; v) \cos\theta_{l_k}}{p(l_k; v)} \tag{6}$$

The following HLSL code shows how to do this with our shading model:

```
ˆŽ‘ƒ−j  •'‘"−ƒ•…‡  ƒ•'Ž‡  ˆŽ•ƒ−i ‹Y ˆŽ‘ƒ− '—‰Š•‡•ˆŽYƒ−j  €
‡
    ˆŽ‘ƒ−ƒ  °  '—‰Š•‡•• ‹  '—‰Š•‡••Z

    ˆŽ‘ƒ−  Š‹ ° i ‹    ‹  ‹\šZ
    ˆŽ‘ƒ−  '• Š‡−ƒ  ℓ"" • •h ^  ‹\›€  u • h «  •ƒ‹ƒ ^ h€ ‹  ‹\› € €Z
    ˆŽ‘ƒ−  ‹• Š‡−ƒ  ℓ"" • h ^  '• Š‡−ƒ ‹  '• Š‡−ƒ €Z

    ˆŽ‘ƒ−j  Z
    \š ° ‹• Š‡−ƒ .‹.'• Š‹ €Z
    \› ° ‹• Š‡−ƒ •‹‹•• Š‹ €Z
    \œ ° '• Š‡−ƒZ

    ˆŽ‘ƒ−j  ' ‡…−'" ƒ„• \œ€ ³ e\ppp Žcƒ−je YeYh€ˆℓ‘ƒ−jh YeYe€Z
    ˆŽ‘ƒ−j  ƒ•‰‡•− •°"•ƒŽ‹œ‡..‹'•• ' ‡…−'"Y  € €Z
    ˆŽ‘ƒ−j  ƒ•‰‡•− .°"'•• Y  ƒ•‰‡•− €Z
    uu  ƒ•‰‡•− −' ™‹"Ž† •'ƒ…‡
    "‡—" • ƒ•‰‡•− ‹ \š «  ƒ•‰‡•− ‹ \› «  ‹ \œZ
ˆ

ˆŽ‘ƒ−j  '‡…—Žƒ" ˆŽ•ƒ−j '‡…—Žƒ" 'ŽˆŽYƒ− '—‰Š•‡•ˆŽYƒ−j YˆŽ‘ƒ−j  €
‡
    ˆŽ‘ƒ−j  '‡…—Žƒ" ‹‰Š−‹•‰ ° eZ

    …'•− −‹•−−• ƒ•'Ž‡• ° heikZ
    ˆ‘"• →•− ‹ ° eZ ‹ ³ —• ƒ•'Ž‡•Z ‹««« €
    ‡
        ˆŽ‘ƒ−i ‹ ° ƒ••‡"•Ž‡‡›• ‹Y —• ƒ•'Ž‡• €Z
```

4

```
ˆŽ‘ƒ–j  °  •'‘"–ƒ•…‡ ƒ•'Ž‡   •  ‹Y  '—‰Š•‡••Y   €Z
ˆŽ‘ƒ–j  °  i‹ †'•  Y  € ‹  ^  Z

ˆŽ‘ƒ– '  °  •ƒ——"ƒ•‡‘•  Y  € €Z
ˆŽ‘ƒ– '  °  •ƒ——"ƒ•‡‘•  Y  € €Z
ˆŽ‘ƒ– '  °  •ƒ——"ƒ•‡‘•  Y  € €Z
ˆŽ‘ƒ– '  °  •ƒ——"ƒ•‡‘•  Y  € €Z

‹ˆ•  '  ´  e €
‡
    ˆŽ‘ƒ–j ƒ•'Ž‡ 'Ž‘"  °  •˜ ƒ'\ ƒ•'Ž‡ ‡˜‡Ž• •˜ ƒ' ƒ•'Ž‡"Y  Y e €\"‰„Z

    ˆŽ‘ƒ–  °  O •‹–Š• '—‰Š•‡••Y ' Y '  €Z
    ˆŽ‘ƒ– … °'‘TM h ^ ' Y I €Z
    ˆŽ‘ƒ–j  °  •h ^ …€ ‹ '‡…—Žƒ" 'Ž‘" « …Z

    u u •…‹†‡•– Ž‹‰Š– ° ƒ•'Ž‡ 'Ž‘" ‹ '
    u u ‹…"'ˆ…‡– •'‡…—Žƒ" ° ‹ ‹ u •k‹ ' ‹ ' €
    u u †ˆ° ‹ ' u •k‹ ' €
     '‡…—Žƒ" ‹‰Š–‹•‰ «° ƒ•'Ž‡ 'Ž‘" ‹ ‹ ‹ ' u • ' ‹ ' €Z
    ˆ
  ˆ

  "‡—"• '‡…—Žƒ" ‹‰Š–‹•‰ u —• ƒ•'Ž‡•Z
ˆ
```

Even with importance sampling, many samples still need to be taken. The sample count can be reduced significantly by using mip maps [3], but counts still need to be greater than 16 for sufficient quality. Because we blend between many environment maps per pixel for local reflections, we can only practically afford a single sample for each.

Split Sum Approximation

To achieve this, we approximate the above sum by splitting it into two sums. Each separate sum can then be precalculated. This approximation is exact for a constant $L_i(l)$ and fairly accurate for common environments.

$$\frac{1}{N}\sum_{k=1}^{N}\frac{L_i(l_k)f(l_k;v)\cos\theta_k}{p(l_k;v)} \approx \left(\frac{1}{N}\sum_{k=1}^{N}L_i(l_k)\right)\left(\frac{1}{N}\sum_{k=1}^{N}\frac{f(l_k;v)\cos\theta_k}{p(l_k;v)}\right) \tag{7}$$

Pre-Filtered Environment Map

We pre-calculate the first sum for different roughness values and store the results in the mip-map levels of a cubemap. This is the typical approach used by much of the game industry [1, 9]. One minor difference is that we convolve the environment map with the GGX distribution of our shading model using importance sampling. Since it's a microfacet model, the shape of the distribution changes based on viewing angle to the surface, so we assume that this angle is zero, i.e. $\mathbf{n} = \mathbf{v} = \mathbf{r}$. This isotropic assumption is a second source of approximation and it unfortunately means we don't get lengthy reflections at grazing angles. Compared with the split sum approximation, this is actually the larger source of error for our IBL solution. As shown in the code below, we have found weighting by $\cos\theta_k$ achieves better results [R].

[R] h?Bb r2iBQM; Bb MQi T`2b2Mi BM 1[m iBQM d- r?B+? Bb H27i BM  bBKTH2` 7Q`K

‡

Environment BRDF

The second sum includes everything else. This is the same as integrating the specular BRDF with a solid-white environment, i.e. $L_i(l_k) = 1$. By substituting in Schlick's Fresnel: $F(v;h) = F_0 + (1 - F_0)(1 - v \cdot h)^5$, we find that F_0 can be factored out of the integral.

$$\int_H f(l;v) \cos \theta_l \, dl = F_0 \int_H \frac{f(l;v)}{F(v;h)}\left(1 - (1 - v \cdot h)^5\right) \cos \theta_l \, dl + \int_H \frac{f(l;v)}{F(v;h)}(1 - v \cdot h)^5 \cos \theta_l \, dl \tag{8}$$

This leaves two inputs (*Roughness* and $\cos \theta_p$) and two outputs (a scale and bias to F_0), all of which are conveniently in the range $[0, 1]$. We precalculate the result of this function and store it in a 2D look-up texture[k] (LUT).

Figure 3: 2D LUT

---

[k] q2 m b2M hm h7nQ`K i- bBM+2 r2 7QmM/ T`2+BbBQM iQ #2 BKTQ`i MiX

After completing this work, we discovered both existing and concurrent research that lead to almost identical solutions to ours. Whilst Gotanda used a 3D LUT [8], Drobot optimized this to a 2D LUT [7], in much the same way that we did. Additionally—as part of this course—Lazarov goes one step further [11], by presenting a couple of analytical approximations to a similar integral[j].

```
ˆŽ'ƒ–i  •–‡‰"ƒ–‡   ˆŽ•'ƒ– '—‰Š•‡•·•ŽYƒ–'   €
‡
    ˆŽ'ƒ–j  Z
    \š ° •""—• h\eˆ ^  '  ‹  '   €Zu •‹•
    \› ° eZ
    \œ°  ' Z                         u u  …'•

    ˆŽ'ƒ–  ° eZ
    ˆŽ'ƒ–  ° eZ

    …'•– —‹•——• ƒ•'Ž‡• ° heikZ
    ˆ'"• —+–‹ ° eZ ‹ ³  —• ƒ•'Ž‡•Z ‹«« €
    ‡
        ˆŽ'ƒ–i ‹ °  ƒ••‡"•Ž‡›• ‹Y  —• ƒ•'Ž‡• €Z
        ˆŽ'ƒ–j  °  •''"–ƒ•…‡ ƒ•'Ž‡  •  ‹Y '—‰Š•‡••Y    €Z
        ˆŽ'ƒ–j  ° i‹ †'–• Y  € ‹  ^  Z

        ˆŽ'ƒ– '  ° •ƒ——"ƒ–‡\œ €Z
        ˆŽ'ƒ– '  ° •ƒ——"ƒ–‡\œ €Z
        ˆŽ'ƒ– '  ° •ƒ——"ƒ–‡'–• Y   € €Z

        ‹ˆ• '  ´ e €
        ‡
            ˆŽ'ƒ–  ° O •‹–Š• '—‰Š•‡••Y ' Y '  €Z

            ˆŽ'ƒ– O ‹• °  ‹ ' u•' ‹ ' €Z
            ˆŽ'ƒ– … °''TM• h ^ ' Y I €Z
              «°•h ^ …€ ‹ O ‹•Z
              «° … ‹ O ‹•Z
        ˆ
    ˆ

    "‡—"•ˆŽ'ƒ–•i Y  € u —• ƒ•'Ž‡•Z
ˆ
```

Finally, to approximate the importance sampled reference, we multiply the two pre-calculated sums:

```
ˆŽ'ƒ–j ''"'š‹•ƒ–‡ '‡…—Žƒ"ˆŽ'ƒ–j '‡…—Žƒ" 'ŽˆŽYƒ–'—‰Š•‡•·•ŽYƒ–j YˆŽ'ƒ–j €
‡
    ˆŽ'ƒ– '  ° •ƒ——"ƒ–'–• Y   € €Z
    ˆŽ'ƒ–j  ° i‹ †'–• Y € ‹  ^  Z

    ˆŽ'ƒ–j "‡ˆ‹Ž–‡"‡† 'Ž'" ° "‡ˆ‹Ž–‡" •˜ ƒ'• '—‰Š•‡••Y   €Z
    ˆŽ'ƒ–i •˜   ° •–‡‰"ƒ–‡  • '—‰Š•‡••Y ' €Z

    "‡—"• "‡ˆ‹Ž–‡"‡† 'Ž'" ‹• '‡…—Žƒ" 'Ž'" ‹ •˜   \š « •˜   \› €Z
ˆ
```

[j] h?2B` b? /BM2HKMQb2b /Dz2M2GM7mM+iBQMbX
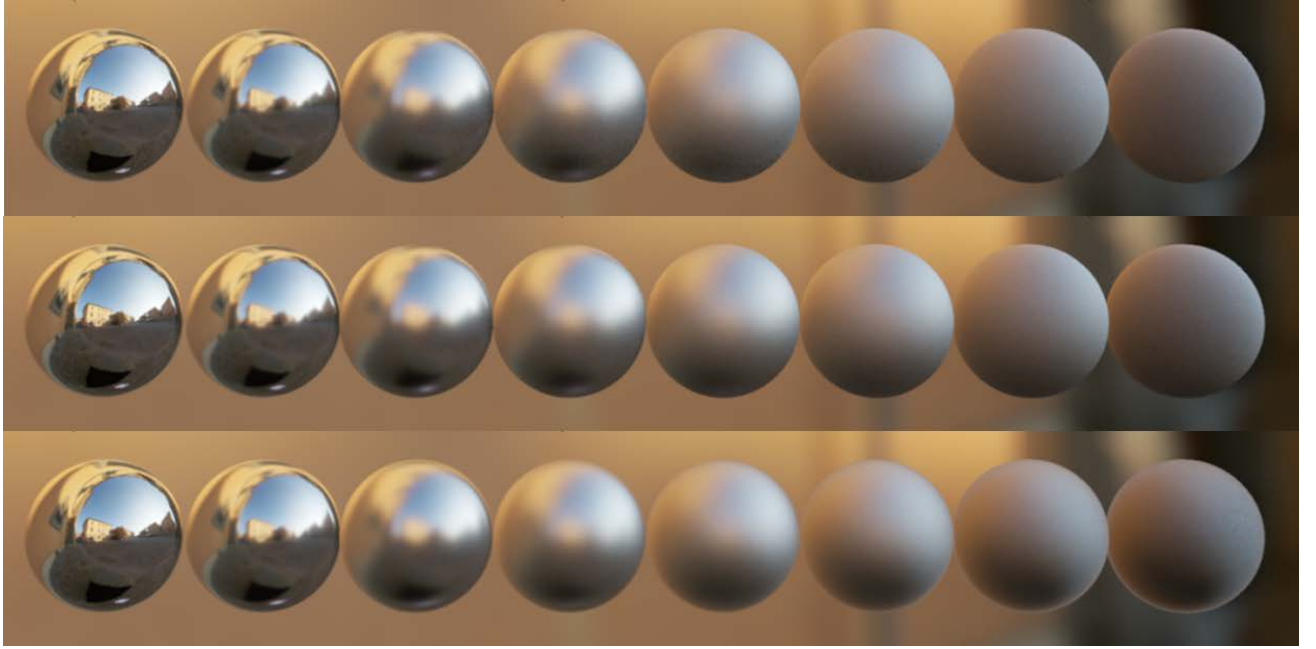
Figure 4: Reference at top, split sum approximation in the middle, full approximation including $\mathbf{n} = \mathbf{v}$ assumption at the bottom. The radially symmetric assumption introduces the most error but the combined approximation is still very similar to the reference.
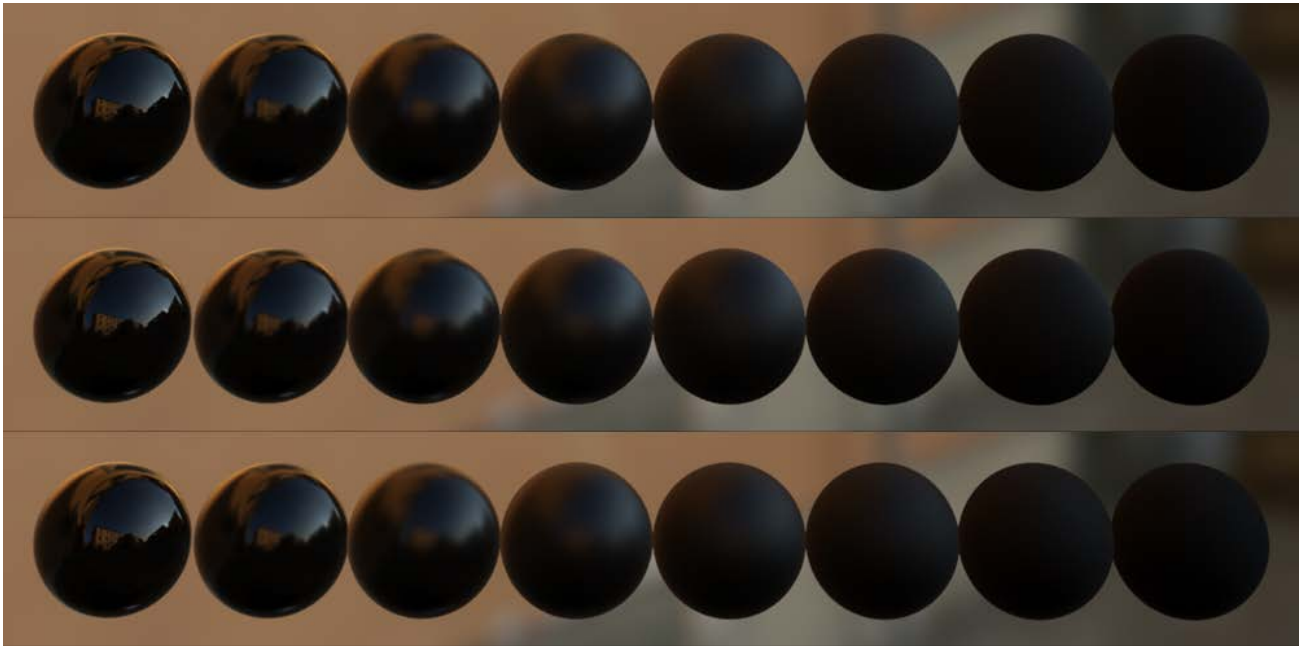
Figure 5: Same comparison as Figure 4 but with a dielectric.

Material Model

Our material model is a simplification of Disney's, with an eye towards efficiency for real-time rendering. Limiting the number of parameters is extremely important for optimizing G-Buffer space, reducing texture storage and access, and minimizing the cost of blending material layers in the pixel shader.

The following is our base material model:

BaseColor	Single color. Easier concept to understand.
Metallic	No need to understand dielectric and conductor reflectance, so less room for error.
Roughness	Very clear in its meaning, whereas *gloss* always needs explaining.
Cavity	Used for small-scale shadowing.

BaseColor, *Metallic*, and *Roughness* are all the same as Disney's model, but *Cavity* parameter wasn't present, so it deserves explanation. *Cavity* is used to specify shadowing from geometry smaller than our run-time shadowing system can handle, often due to the geometry only being present in the normal map. Examples are the cracks between floor boards or the seams in clothing.

The most notable omission is the *Specular* parameter. We actually continued to use this up until the completion of our *Infiltrator* demo, but ultimately we didn't like it. First off, we feel "specular" is a terrible parameter name which caused much confusion and was somewhat harmful to the transition from artists controlling specular *intensity* to controlling *roughness*. Artists and graphics programmers alike commonly forgot its range and assumed that the default was 1, when its actual default was Burley's 0.5 (corresponding to 4% reflectance). The cases where *Specular* was used effectively were almost exclusively for the purpose of small scale shadowing. We found variable index of refraction (IOR) to be fairly unimportant for nonmetals, so we have recently replaced *Specular* with the easier to understand *Cavity* parameter. F_0 of nonmetals is now a constant 0.04.

The following are parameters from Disney's model that we chose not to adopt for our base material model and instead treat as special cases:

Subsurface	Samples shadow maps differently
Anisotropy	Requires many IBL samples
Clearcoat	Requires double IBL samples
Sheen	Not well defined in Burley's notes

We have not used these special case models in production with the exception of *Subsurface* which was used for the ice in our *Elemental* demo. Additionally we have a shading model specifically for skin. For the future we are considering adopting a hybrid deferred/forward shading approach to better support more specialized shading models. Currently with our purely deferred shading approach, different shading models are handled with a dynamic branch from a shading model id stored in the G-Buffer.

Experiences

There is one situation I have seen a number of times now. I will tell artists beginning the transition to using varying roughness, "Use *Roughness* like you used to use *SpecularColor*" and soon after I hear with excited surprise: "It works!" But an interesting comment that has followed is: "*Roughness* feels inverted." It turns out that artists want to see the texture they author as brighter *texels* equals brighter *specular highlights*. If the image stores roughness, then *bright* equates to *rougher*, which will lead to less intense highlights.

A question I have received countless times is: "Is *Metallic* binary?," to which I'd originally explain the subtleties of mixed or layered materials. I have since learned that it's best to just say "Yes!" The reason is that artists at the beginning were reluctant to set parameters to absolutes; very commonly I would find metals with *Metallic* values of 0.8. Material layers—discussed next—should be the way to describe 99% of cases where *Metallic* would not be either 0 or 1.

We had some issues during the transition, in the form of materials that could no longer be replicated. The most important set of these came from *Fortnite*, a game currently in production at Epic. *Fortnite* has a non-photorealistic art direction and purposefully uses complementary colors for diffuse and specular reflectance, something that is not physically plausible and intentionally cannot be represented in our new material model. After long discussions, we decided to continue to support the old *DiffuseColor/SpecularColor* as an engine switch in order to maintain quality in *Fortnite*, since it was far into development. However, we don't feel that the new model precludes non-photorealistic rendering as demonstrated by Disney's use in *Wreck-It Ralph*, so we intend to use it for all future projects.

Material Layering

Blending material layers which are sourced from a shared library provides a number of benefits over our previous approach, which was to specify material parameters individually with the values coming from textures authored for each specific model:

- Reuses work across a number of assets.

- Reduces complexity for a single asset.

- Unifies and centralizes the materials defining the look of the game, allowing easier art and tech direction.

To fully embrace this new workflow, we needed to rethink our tools. Unreal Engine has featured a node graph based material editor since early in UE3's lifetime. This node graph specifies inputs (textures, constants), operations, and outputs, which are compiled into shader code.

Although material layering was a primary goal of much of this work, surprisingly very little needed to be added tool-side to support the authoring and blending of material layers. Sections of node graphs in UE4's material editor could already be grouped into functions and used from multiple materials. This functionality is a natural fit for implementing a material layer. Keeping material layers inside our node-based editor, instead of as a fixed function system on top, allowed layers to be mapped and combined in a programmable way.

To streamline the workflow, we added a new data type, material attributes, that holds all of the materials output data. This new type, like our other types, can be passed in and out of material functions as single pins, passed along wires, and outputted directly. With these changes, material layers can be dragged in as inputs, combined, manipulated and output in the same way that textures were previously. In fact, most material graphs tend to be simpler since the adoption of layers as the primary things custom to a particular material are how layers are mapped and blended. This is far simpler than the parameter specific manipulation that used to exist.

Due to having a small set of perceptually linear material parameters, it is actually practical to blend layers entirely within the shader. We feel this offers a substantial increase in quality over a purely offline composited system. The apparent resolution of texture data can be extremely high due to being able to map data at different frequencies: per vertex or low frequency texture data can be unique, layer blend masks, normal maps and cavity maps are specified per mesh, and material layers are tiled across the surface of the mesh. More advanced cases may use even more frequencies. Although

Figure 6: Simple material layering in the UE4 material editor

we are practically limited in the number of layers we can use due to shader cost, our artists have not yet found the limitation problematic.

An area that is cause for concern is that the artists in cases have worked around the in-shader layering limitations by splitting a mesh into multiple sections, resulting in more draw calls. Although we expect better draw call counts in UE4 due to CPU side code optimizations, this seems like it may be a source of problems in the future. An area we haven't investigated yet is the use of dynamic branching to reduce the shader cost in areas where a layer has 100% coverage.

So far, our experience with material layers has been extremely positive. We have seen both productivity gains and large improvements in quality. We wish to improve the artist interface to the library of material layers by making it easier to find and preview layers. We also intend to investigate an offline compositing/baking system in addition to our current run-time system, to support a larger number of layers and offer better scalability.

Figure 7: Many material layers swapped and blended with rust.

Figure 8: Material layering results exploiting multiple frequencies of detail.

Lighting Model

As with shading, we wished to improve our lighting model by making it more physically based. The two areas we concentrated on were light falloff and non-punctual sources of emission—commonly known as area lights.

Improving light falloff was relatively straightforward: we adopted a physically accurate inverse-square falloff and switched to the photometric brightness unit of lumens. That said, one minor complication we had to deal with is that this kind of falloff function has no distance at which it reaches zero. But for efficiency—both in real-time and offline calculations—we still needed to artificially limit the influence of lights. There are many ways to achieve this [4], but we chose to window the inverse-square function in such a way that the majority of the light's influence remains relatively unaffected, whilst still providing a soft transition to zero. This has the nice property whereby modifying a light's radius doesn't change its effective brightness, which can be important when lighting has been locked artistically, but a light's extent still needs to be adjusted for performance reasons.

$$\textbf{falloff} = \frac{\text{saturate}(1 - (\text{distance/lightRadius})^4)^2}{\text{distance}^2 + 1} \tag{9}$$

The 1 in the denominator is there to prevent the function exploding at distances close to the light source. It can be exposed as an artist-controllable parameter for cases where physical correctness isn't desired.

The quality difference this simple change made, particularly in scenes with many local light sources, means that it is likely the largest bang for buck takeaway.

Figure 9: Inverse square falloff achieves more natural results

Area Lights

Area light sources don't just generate more realistic images. They are also fairly important when using physically based materials. We found that without them, artists tended to intuitively avoid painting very low roughness values, since this resulted in infinitesimal specular highlights, which looked unnatural. Essentially, they were trying to reproduce the effect of area lighting from punctual sources [9].

Unfortunately, this reaction leads to a coupling between shading and lighting, breaking one of the core principles of physically based rendering: that materials shouldn't need to be modified when used in a different lighting environment from where they were created.

Area lights are an active area of research. In offline rendering, the common solution is to light from many points on the surface of the light source—either using uniform sampling or importance sampling [12] [20]. This is completely impractical for real-time rendering. Before discussing possible solutions, here were our requirements:

- Consistent material appearance

  { The amount of energy evaluated with the Diffuse BRDF and the Specular BRDF cannot be significantly different.

- Approaches point light model as the solid angle approaches zero

  { We don't want to lose any aspect of our shading model to achieve this.

- Fast enough to use everywhere

  { Otherwise, we cannot solve the aforementioned "biased roughness" issue.

---

[9] h ? B b i H H B Q # b B ï ß i B Q M b 7 ` Q K  Q i ? 2 ` / 2 p 2 H Q T 2 ` b (d - 3) X

Billboard Reflections

Billboard reflections [13] are a form of IBL that can be used for discrete light sources. A 2D image, which stores emitted light, is mapped to a rectangle in 3d space. Similar to environment map pre-filtering, the image is pre-filtered for different-sized specular distribution cones. Calculating specular shading from this image can be thought of as a form of cone tracing, where a cone approximates the specular NDF. The ray at the center of the cone is intersected with the plane of the billboard. The point of intersection in image space is then used as the texture coordinates, and the radius of the cone at the intersection is used to derive an appropriate pre-filtered mip level. Sadly, while images can express very complex area light sources in a straightforward manner, billboard reflections fail to fulfill our second requirement for multiple reasons:

- The image is pre-filtered on a plane, so there is a limited solid angle that can be represented in image space.

- There is no data when the ray doesn't intersect the plane.

- The light vector, $\mathbf{l}$, is unknown or assumed to be the reflection vector.

Cone Intersection

Cone tracing does not require pre-filtering; it can be done analytically. A version we experimented with traced a cone against a sphere using Oat's cone-cone intersection equation [15], but it was far too expensive to be practical. An alternative method presented recently by Drobot [7] intersected the cone with a disk facing the shading point. A polynomial approximating the NDF is then piece-wise integrated over the intersecting area.

With Drobot's recent advancements, this seems to be an interesting area for research, but in its current form, it doesn't fulfill our requirements. Due to using a cone, the specular distribution must be radially symmetric. This rules out stretched highlights, a very important feature of the microfacet specular model. Additionally, like billboard reflections, there isn't a defined light vector required by the shading model.

Specular D Modification

An approach we presented last year [14] is to modify the specular distribution based on the solid angle of the light source. The theory behind this is to consider the light source's distribution to be the same as $\mathsf{D}(\mathbf{h})$ for a corresponding cone angle. Convolving one distribution by the other can be approximated by adding the angles of both cones to derive a new cone. To achieve this, convert α from Equation 3 into an effective cone angle, add the angle of the light source, and convert back. This α' is now used in place of α. We use the following approximation to do this:

$$\alpha' = \text{saturate}\left(\alpha + \frac{\text{sourceRadius}}{3 \cdot \text{distance}}\right) \tag{10}$$

Although efficient, this technique unfortunately doesn't fulfill our first requirement, as very glossy materials appear rough when lit with large area lights. This may sound obvious, but the technique works a great deal better when the specular NDF is compact—Blinn-Phong, for instance—thereby better matching the light source's distribution. For our chosen shading model (based on GGX), it isn't viable.

Figure 10: Reference on the left, specular D modification method on the right. The approximation is poor due to the spherical shape getting lost on grazing angles and glossy materials, such as the polished brass head, look rough.

Representative Point

If for a specific shading point we could treat all light coming from the area light as coming from a single *representative point* on the surface of the light source, our shading model could be used directly. A reasonable choice is the point with the *largest contribution*. For a Phong distribution, this is the point on the light source with the smallest angle to the reflection ray.

This technique has been published before [16] [22], but energy conservation was never addressed. By moving the origin of emitted light, we effectively increased the light's solid angle but haven't compensated for the additional energy. Correcting for it is slightly more complex than dividing by the solid angle, since the energy difference is dependent on the specular distribution. For instance, changing the incoming light direction for a rough material will result in very little change in energy, but for a glossy material the change in energy can be massive.

Sphere Lights

Irradiance for a sphere light is equivalent to a point light if the sphere is above the horizon [18]. Although counter-intuitive, this means we only need to address specular lighting if we accept inaccuracies when the sphere dips below the horizon. We approximate finding the point with the smallest angle to the reflection ray by finding the point with the smallest distance to the ray. For a sphere this is straightforward:

$$
\begin{aligned}
\mathbf{centerToRay} &= \mathbf{L} - (\mathbf{L} \cdot \mathbf{r})\,\mathbf{r} \\
\mathbf{closestPoint} &= \mathbf{L} + \mathbf{centerToRay} \cdot \mathrm{saturate}\left(\frac{\mathsf{sourceRadius}}{|\mathbf{centerToRay}|}\right) \\
\mathbf{l} &= \|\mathbf{closestPoint}\|
\end{aligned}
\tag{11}
$$

Here, $\mathbf{L}$ is a vector from the shading point to the center of the light and $\mathsf{sourceRadius}$ is the radius of the light sphere and $\mathbf{r}$ is the reflection vector. In the case where the ray intersects the sphere, the

15

point calculated will be the closest point on the ray to the center of the sphere. Once normalized, it is identical.

By moving the origin of emitted light to the sphere's surface, we have effectively widened the specular distribution by the sphere's subtended angle. Although it isn't a microfacet distribution, this can be explained best using a normalized Phong distribution:

$$I_{\text{point}} = \frac{p+2}{2}\cos^p\theta_r \tag{12}$$

$$I_{\text{sphere}} = \begin{cases} \dfrac{p+2}{2} & \text{if } \theta_r < \theta_s \\[2mm] \dfrac{p+2}{2}\cos^p(\theta_r - \theta_s) & \text{if } \theta_r \geq \theta_s \end{cases} \tag{13}$$

Here θ_r is the angle between $\mathbf{r}$ and $\mathbf{L}$ and θ_s is half the sphere's subtended angle. I_{point} is normalized, meaning the result of integrating it over the hemisphere is 1. I_{sphere} is clearly no longer normalized and depending on the power p, the integral can be far larger.

Figure 11: Visualization of the widening effect explained with Equation 13

To approximate this increase in energy, we apply the same reasoning used by the specular D modification described earlier, where we widened the distribution based on the solid angle of the light. We use the normalization factor for that wider distribution and replace the original normalization factor. For GGX the normalization factor is $\frac{1}{\alpha^2}$. To derive an approximate normalization for the representative point operation we divide the new widened normalization factor by the original:

$$\text{SphereNormalization} = \left(\frac{\alpha}{\alpha_0}\right)^2 \tag{14}$$

The results of the representative point method meet all of our requirements. By correctly addressing energy conservation, materials behave the same regardless of light source size. Glossy materials still produce sharp-edged specular highlights, and since it only modifies the inputs to the BRDF, our shading model is unaffected. Lastly, it is efficient enough to use wherever our artists like.

Tube Lights

Where sphere lights are useful to represent light bulbs, tube lights (capsules) can be useful to represent florescent lights which are quite common in the real world. To start, we solve for tube lights with a length but zero radius, also known as linear lights. Irradiance for a line segment can be integrated analytically so long as the segment is above the horizon [16, 17]:

$$\int_{\Gamma_0}^{\Gamma_1} \frac{\mathbf{n}\cdot\mathbf{L}}{|\mathbf{L}|^3}d\mathbf{L} = \frac{\frac{\mathbf{n}\cdot\mathbf{L}_0}{|\mathbf{L}_0|} + \frac{\mathbf{n}\cdot\mathbf{L}_1}{|\mathbf{L}_1|}}{|\mathbf{L}_0||\mathbf{L}_1| + (\mathbf{L}_0\cdot\mathbf{L}_1)} \tag{15}$$

Where $\mathbf{L}_0$ and $\mathbf{L}_1$ are vectors from the shading point to the end points of the segment.

Figure 12: Reference on the left, representative point method on the right. Although the energy conservation isn't perfect, our approximation convincingly matches the reference.

We modify this equation to prevent negative irradiance, divide-by-zero, and to match our point light falloff when the length is zero:

$$\text{irradiance} = \frac{2 \cdot \text{saturate}\left(\frac{\mathbf{N} \cdot \mathbf{L}_0}{2|\mathbf{L}_0|} + \frac{\mathbf{N} \cdot \mathbf{L}_1}{2|\mathbf{L}_1|}\right)}{|\mathbf{L}_0||\mathbf{L}_1| + (\mathbf{L}_0 \cdot \mathbf{L}_1) + 2} \tag{16}$$

For linear light specular we need to solve for t in the following set of equations:

$$\begin{aligned}
\mathbf{L}_d &= \mathbf{L}_1 - \mathbf{L}_0 \\
l &= |\mathbf{L}_0 + \text{saturate}(t)\mathbf{L}_d|
\end{aligned} \tag{17}$$

Picott [16] found t for the smallest angle to $\mathbf{r}$:

$$t = \frac{(\mathbf{L}_0 \cdot \mathbf{L}_d)(\mathbf{r} \cdot \mathbf{L}_0) - (\mathbf{L}_0 \cdot \mathbf{L}_0)(\mathbf{r} \cdot \mathbf{L}_d)}{(\mathbf{L}_0 \cdot \mathbf{L}_d)(\mathbf{r} \cdot \mathbf{L}_d) - (\mathbf{L}_d \cdot \mathbf{L}_d)(\mathbf{r} \cdot \mathbf{L}_0)} \tag{18}$$

Like the sphere case, we approximate smallest angle and solve instead for the shortest distance:

$$t = \frac{(\mathbf{r} \cdot \mathbf{L}_0)(\mathbf{r} \cdot \mathbf{L}_d) - (\mathbf{L}_0 \cdot \mathbf{L}_d)}{|\mathbf{L}_d|^2 - (\mathbf{r} \cdot \mathbf{L}_d)^2} \tag{19}$$

There are edge cases that aren't properly handled such that this won't always find the closest point but it is a little cheaper to calculate and seems to produce just as reasonable of results as Equation 18.

It is important to note, that since both Equation 18 and 19 treat $\mathbf{r}$ as a line instead of ray, neither solution properly handles rays pointing away from the line segment. This can cause an abrupt change from one end point to the other even for perfectly flat surfaces. This happens when the reflection rays transition from pointing towards the light to pointing away. We could address this by choosing between the calculated point and each end point but this is expensive. At the moment we simply accept the artifact.

To conserve energy we apply the same concept used for sphere lights. The specular distribution has been widened by the light's subtended angle but this time only in one dimension so we use

the anisotropic version of GGX [2]. The normalization factor for anisotropic GGX is $\frac{1}{\alpha_x \alpha_y}$ where $\alpha_x = \alpha_y = \alpha$ in the isotropic case. This gives us:

$$\text{LineNormalization} = \frac{\alpha}{\alpha_0} \qquad (20)$$

Because we are only changing the light's origin and applying an energy conservation term these operations can be accumulated. Doing so with a line segment and a sphere, approximates a convolution of the shapes and models fairly well the behavior of a tube light. Tube light results are shown in Figure 13
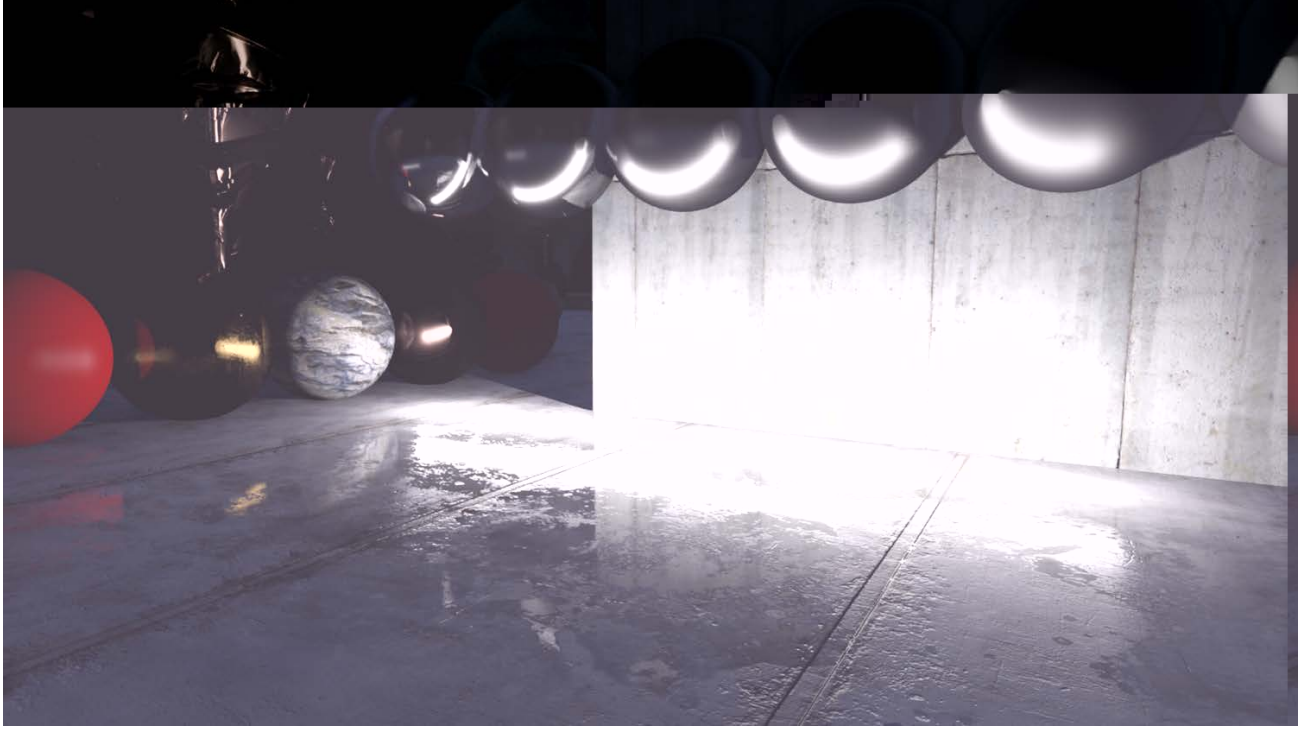
Figure 13: Tube light using the representative point method with energy conservation

We have found the representative point method with energy conservation to be effective for simple shapes and would like to apply it in the future to other shapes beyond spheres and tubes. In particular, we would like to apply it to textured quads to express more complex and multi-colored light sources.

Conclusion

Our move to more physically based implementations, in the areas of shading, materials and lighting, has proved to be very successful. It contributed greatly to the visuals of our latest Infiltrator demo and we plan to use these improvements for all future projects. In fact, in cases where it is practical these changes have been integrated into *Fortnite*, a project well into development before this work began. We intend to continue to improve in these areas with the goals of greater flexibility and scalability with the goal that all variety of scenes and all levels of hardware can take advantage of physically based approaches.

Acknowledgments

Bibliography

[1] AMD, CubeMapGen: Cubemap Filtering and Mipchain Generation Tool. Š−−'[u u † ‡ ˜ ‡ Ž ' ' ‡ " \ ƒ • † \
… ' • u " ‡ • ' — " … ‡ • u ƒ " … Š ‹ ˜ ‡ u ƒ " … Š ‹ ˜ ‡ † ^ − ' ' Ž • u ‰ ' — ^ − ' ' Ž • ^ ƒ " … Š ‹ ˜ ‡ u … — „ ‡ • ƒ ' ‰ ‡ • u

[2] Burley, Brent, "Physically-Based Shading at Disney", part of "Practical Physically Based Shading in Film and Game Production", *SIGGRAPH 2012 Course Notes*. Š−−'[u u „ Ž ' ‰ \ • ‡ Ž ˆ • Š ƒ † ' ™ \ … ' •
' — „ Ž ‹ … ƒ − ' • • u • i e h i ^ • Š ƒ † ‹ • ‰ ^ … ' — " • ‡ u

[3] Colbert, Mark, and Jaroslav Krivanek, "GPU-based Importance Sampling", in Hubert Nguyen, ed., *GPU Gems 3*, Addison-Wesley, pp. 459–479, 2007. Š−−'[u u Š−−'\ † ‡ ˜ ‡ Ž ' ' ‡ " \ • ˜ ‹ † ‹ ƒ \ … ' •
‡ • • j u ‰ ' — ‰ ‡ • • j O … Š i e \ Š − • Ž

[4] Coffin, Christina, "SPU Based Deferred Shading in Battlefield 3 for Playstation 3", *Game Developers Conference*, March 2011. Š−−'[u u ™ ™ ™ \ • Ž ‹ † ‡ • Š ƒ " ‡ \ • ‡ − u   — — † ‹ ' u • ' — „ ƒ • ‡ † ^
• Š ƒ † ‹ • ‰ ^ ‹ • ^ „ ƒ − − Ž ‡ ˆ ‹ ‡ Ž † ^ j ^ ' " ^ ' Ž ƒ ƒ › • − ƒ − ‹ ' • ^ j

[5] Cook, Robert L., and Kenneth E. Torrance, "A Reflectance Model for Computer Graphics", *Computer Graphics (SIGGRAPH '81 Proceedings)*, pp. 307–316, July 1981.

[6] Cook, Robert L., and Kenneth E. Torrance, "A Reflectance Model for Computer Graphics", *ACM Transactions on Graphics*, vol. 1, no. 1, pp. 7–24, January 1982. Š−−'[u u ‰ " ƒ ' Š ‹ … • \ ' ‹ š ƒ " \
… ' • u Ž ‹ „ ƒ " › u ‡ ˆ Ž ‡ … − ƒ • … ‡ ' † ‡ Ž u

[7] Drobot, Michał, "Lighting Killzone: Shadow Fall", *Digital Dragons*, April 2013. Š−−'[u u ™ ™ ™ \
‰ — ‡ " " ‹ Ž Ž ƒ ^ ‰ ƒ • ‡ • \ … ' • u ' — „ Ž ‹ … ƒ − ' • • u

[8] Gotanda, Yoshiharu, "Practical Implementation of Physically-Based Shading Models at tri-Ace", part of "Physically-Based Shading Models in Film and Game Production", *SIGGRAPH 2010 Course Notes*. Š−−'[u u " ‡ • † ‡ " ™ ' • • \ … ' • u ' — „ Ž ‹ … ƒ − ' • • u • i e h e ^ • Š ƒ † ‹ • ‰ ^ … ' — " • ‡ u

[9] Hoffman, Naty, "Background: Physics and Math of Shading", part of "Physically Based Shading in Theory and Practice", *SIGGRAPH 2013 Course Notes*. Š−−'[u u „ Ž ' ‰ \ • ‡ Ž ˆ • Š ƒ † ' ™ \ … ' •
' — „ Ž ‹ … ƒ − ' • • u • i e h j ^ • Š ƒ † ‹ • ‰ ^ … ' — " • ‡ u

[10] Lagarde, Sebastien, "Spherical Gaussian approximation for Blinn-Phong, Phong and Fresnel", June 2012. Š−−'[u u • ‡ „ Ž ƒ ‰ f " † ‡ \ ™ ' " † ' " ‡ • • \ … ' u i e h i u e m u e j u • ' Š ‡ " ‹ … ƒ Ž ^ ‰ ƒ — • • ‹ ‡
^ ' " ^ „ Ž ‹ • • ^ ' Š ‹ • ‰ ^ ' Š ‹ • ‰ ^ ƒ • † ^ ˆ " ‡ • • Ž u

[11] Lazarov, Dimitar, "Getting More Physical in Call of Duty: Black Ops II", part of "Physically Based Shading in Theory and Practice", *SIGGRAPH 2013 Course Notes*. Š−−'[u u „ Ž ' ‰ \
• ‡ Ž ˆ • Š ƒ † ' ™ \ … ' • u ' — „ Ž ‹ … ƒ − ' • • u • i e h j ^ • Š ƒ † ‹ • ‰ ^ … ' — " • ‡ u

[12] Martinez, Adam, "Faster Photorealism in Wonderland: Physically-Based Shading and Lighting at Sony Pictures Imageworks", part of "Physically-Based Shading Models in Film and Game Production", *SIGGRAPH 2010 Course Notes.* Š−−'[u u " ‡ • † ‡ " ™ ' • • \ … ' • u ' — „ Ž ‹ … ƒ – ‹ ' • • u • i e h e ^ • Š … ' — " • ‡ u

[13] Mittring, Martin, and Bryan Dudash, "The Technology Behind the DirectX 11 Unreal Engine Samaritan Demo", *Game Developer Conference 2011.* Š−−'[u u — † • \ ‡ ' ‹ … ‰ ƒ • ‡ • \ … ' • u Š " ‡ ‡ u ' Š " ‡ ‡ u ‹ " ‡ … – h h ‡ • † ‡ " ‹ • ‰ u ƒ " – ‹ • O   h h O   h h O ' " ‡ • ‡ • – ƒ – ‹ ' • \ ' † ˆ

[14] Mittring, Martin, "The Technology Behind the Unreal Engine 4 Elemental demo", part of "Advances in Real-Time Rendering in 3D Graphics and Games Course", *SIGGRAPH 2012.* Š−−'[u u ™ ™ ™ \ — • " ‡ ƒ Ž ‡ • ‰ ‹ • ‡ \ … ' • u ˆ ‹ Ž ‡ • u • ‹ • … u Š ‡ O ‡ … Š • ' Ž ' ‰ › O ‡ Š ‹ • † O – Š ‡ O Ž ‡ • ‡ •

[15] Oat, Chris, "Ambient Aperture Lighting", *SIGGRAPH 2006.* Š−−'[u u † ‡ ˜ ‡ Ž ' ' ‡ " \ ƒ • † \ … ' • u ™ ' " † ' " ‡ • • u • ‡ † ‹ ƒ u i e h i u h e u ƒ – ^ • „ ‹ ‡ • – ' ‡ – — " ‡ ‹ ‰ Š – ‹ • ‰ \ ' † ˆ

[16] Picott, Kevin P., "Extensions of the Linear and Area Lighting Models", *Computers and Graphics*, Volume 12 Issue 2, March 1992, pp. 31-38. Š−−'[u u † š \ † ‹ ‹ \ ' " ‰ u h e \ h h e p u j o \ h i k i o m

[17] Poulin, Pierre, and John Amanatides, "Shading and Shadowing with Linear Light Sources", *IEEE Computer Graphics and Applications*, 1991. Š−−'[u u ™ ™ ™ \ … • ‡ \ › ' " • — \ … ƒ u L ƒ • ƒ • ƒ u " ‡ • ‡

[18] Quilez, Inigo, "Spherical ambient occlusion", 2006. Š−−'[u u ™ ™ ™ \ ‹ " — ‹ Ž ‡ œ Ž ‡ • \ ' " ‰ u ™ ™ ™ u • ' Š ‡ " ‡ ƒ ' u • ' Š ‡ " ‡ ƒ ' \ Š − •

[19] Schlick, Christophe, "An Inexpensive BRDF Model for Physically-based Rendering", *Computer Graphics Forum*, vol. 13, no. 3, Sept. 1994, pp. 149–162. Š−−'[u u † ‡ ' – ^ ‹ • ˆ ' \ Ž ƒ „ " ‹ \ — ^ „ ' " † ‡ ƒ — š \ ˆ L • … Š Ž ‹ … • u u ‡ — " i \ Š − • Ž

[20] Snow, Ben, "Terminators and Iron Men: Image-based lighting and physical shading at ILM", part of "Physically-Based Shading Models in Film and Game Production", *SIGGRAPH 2010 Course Notes.* Š−−'[u u " ‡ • † ‡ " ™ ' • • \ … ' • u ' — „ Ž ‹ … ƒ – ‹ ' • • u • i e h e ^ • Š ƒ † ‹ • ‰ ^ … ' — " • ‡ u

[21] Walter, Bruce, Stephen R. Marschner, Hongsong Li, Kenneth E. Torrance, "Microfacet Models for Refraction through Rough Surfaces", *Eurographics Symposium on Rendering (2007)*, 195–206, June 2007. Š−−'[u u ™ ™ ™ \ … • \ … ' " • ‡ ‹ ‹ \ ‡ † — u L • " • u ' — „ Ž ‹ … ƒ – ‹ ' • • u e n ^ „ – † ˆ \ Š − • Ž

[22] Wang, Lifeng, Zhouchen Lin, Wenle Wang, and Kai Fu, "One-Shot Approximate Local Shading" 2006.

Real Shading in Unreal En

Brian Karis brian.karis@epicgames.com

Goals

‡More realistic image

‡Material layering

   ±Better workflow

   ±Blended in shader

‡Timely inspiration from Disney

   ±Presented in this course last year

Overview

‡Shading model

‡Material model

‡Lighting model

Shading Model

Diffuse BRDF

‡Lambert

±Saw little effect of more sophisticated models

Lambert   Burley

Specular BRDF

‡Generalized microfacet model

±Compared many options for each term

±Use same input parameters

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

Specular distribution

‡Trowbridge-Reitz (GGX)

±Fairly cheap

±Longer tail looks much more natural

GGX   Blinn-Phong

Geometric shadowing

‡ Schlick

± Matched to Smith

± Cheaper, difference is minor

± Uses the same remapping

Fresnel

‡Schlick

±Approximate the power

Identical for all practical purposes

Image-based lighting : Proble

‡Only use single sample per environment map

‡Match importance-sampled reference

‡Same as P L split the sum

‡Pre-calculate both parts

$$\frac{s}{0}\sum_{\triangleright @ 5}^{Ç}\frac{\cdot(\check{Z})\ B(\check{Z}á)\tilde{}\ \dots\mathbf{\dot{j}\ddot{o}}\ à}{L(\check{Z}á)R} \approx \left(\frac{s}{0}\sum_{\triangleright @ 5}^{Ç}\cdot(\check{Z})\right)\left(\frac{s}{0}\sum_{\triangleright @ 5}^{Ç}\frac{B:\check{Z}á\tilde{}\ ;\ \dots\mathbf{\dot{j}\ddot{o}}\ à}{L:\check{Z}á\tilde{}\ ;}\right)$$

Pre-filtered environment ma

‡1st sum stored in cubemap mips

±Pre-LOWHU IRU VSHFLILF URXJKQHV

±Fixed distribution, assume

±Loses stretched highlights

$$\frac{s}{0} \quad \text{...}$$

Environment BRDF

‡2nd sum stored in 2D lookup texture (LUT)

Roughness

Importance-sampled reference

Split sum approximation

Complete approximation (n=v)

Importance-sampled reference

Split sum approximation

Complete approximation (n=v)

Material Model

Material model

‡BaseColor
  ±Single color
‡Metallic
  ±Less chance of error
‡Roughness
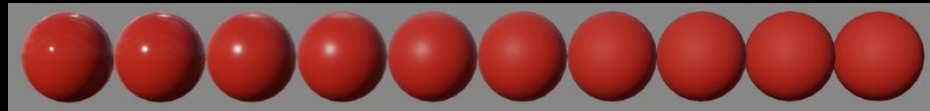 ±Very clear in its meaning
‡Cavity
  ±Used for small scale shadowing

Metallic 0 to 1

Metal with roughness 0 to 1

Nonmetal with roughness 0 to 1

Material model lessons

‡Specular parameter is confusing

±Not really needed

±Replaced with Cavity

Samaritan

DiffuseColor
SpecularColor
SpecularPower

Infiltrator

BaseColor
Metallic
Specular
Roughness

Now

BaseColor
Metallic
Roughness
Cavity
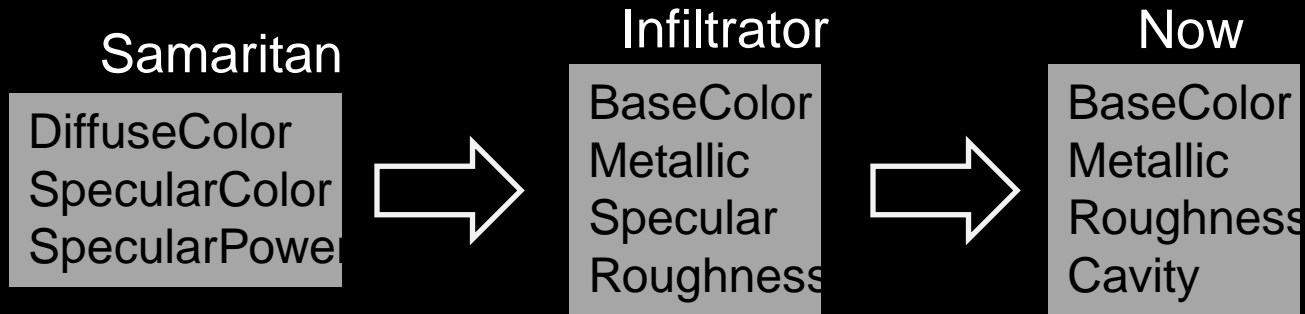
Material layering

‡TODO≥another image

Material layering tools

‡Added layers to our node graph based material e

±Layers use existing material function feature

±Added material attributes struct

‡Layer workflow similar to previous texture workflo

Material layering

Lighting Model

Inverse square falloff

Old falloff   Inverse square

Area Lights

Area light requirements

‡Consistent material appearance

±Energy evaluated with diffuse BRDF and specular BRDF s

‡Approaches point light model as solid angle ap

± ' R Want to lose any aspect of our shading model

‡Fast enough to use everywhere

±Otherwise artists will bias roughness

Specular D modification

‡: L G H Q   V S H F X O D U   G L V V W U L E

±We presented this last year

‡Problems

± * O R V V \   V X U I D F H V anymore G R Q I W   O R R N   J O

Reference

Specular D modifica

Representative point

‡Pick one representative point on light source s

‡Shading model can be used directly

‡Point with largest contribution is a good choice

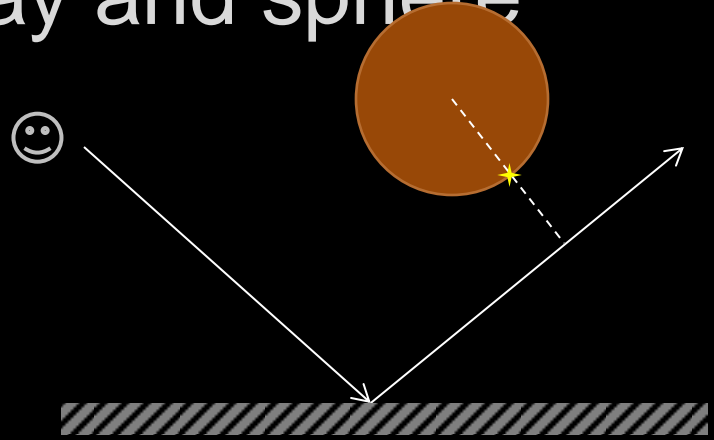‡Approximate using smallest angle to reflection

Sphere lights

‡Irradiance identical to point light

±If sphere above horizon

‡Closest point between ray and sphere

±Approximates smallest angle ☺

Sphere light energy conserva

‡ 6 S H F X O D U   G L V W U L E X W L R Q   K [

± : H   D O U H D G \   K D Y H   D Q   D S S U R [L P D W L R Q

± Only use normalization term

± Divide out original normalization, multiply in new

☺

Reference   Representative poir

Representative point applied to Tube Lights

In the course notes

‡Tons of extra stuff

±Importance sampling code

±Area light formulas

±Lots of math

Thanks

‡Epic

  ±Rendering team

  ±All the artists making me look good

‡Special thanks to Sébastien Lagarde

‡Stephen Hill and Stephen McAuley for valuabl