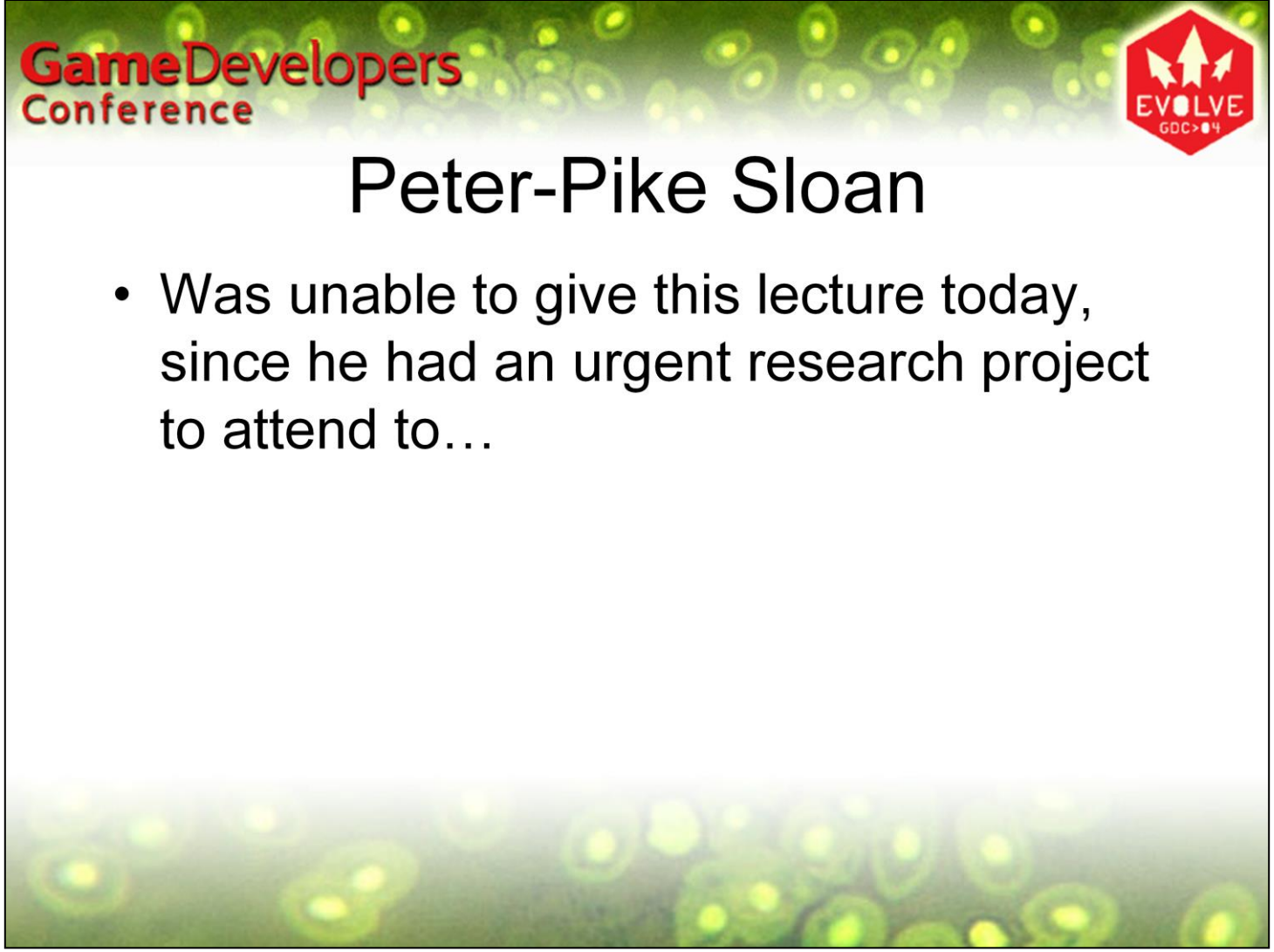
Advanced Real-Time Reflectance

Dan Baker
Microsoft
danbaker@windows.microsoft.com

Naty Hoffman
Naughty Dog, Inc.
naty@naughtydog.com

Peter-Pike Sloan
Microsoft
ppsloan@windows.microsoft.com



Peter-Pike was not able to come to GDC to give the talk because…

GameDevelopers Conference

EVOLVE GDC>04

Kai Michael James Sloan

…of his most recent research project.

Advanced Real-Time Reflectance

- Motivation (Naty Hoffman)
- Surface Types (Naty Hoffman)
- Reflectance Theory (Naty Hoffman)
- Reflection Models (Naty Hoffman)
- Implementation (Dan Baker)
- Production Issues (Naty Hoffman)

Motivation and Theory

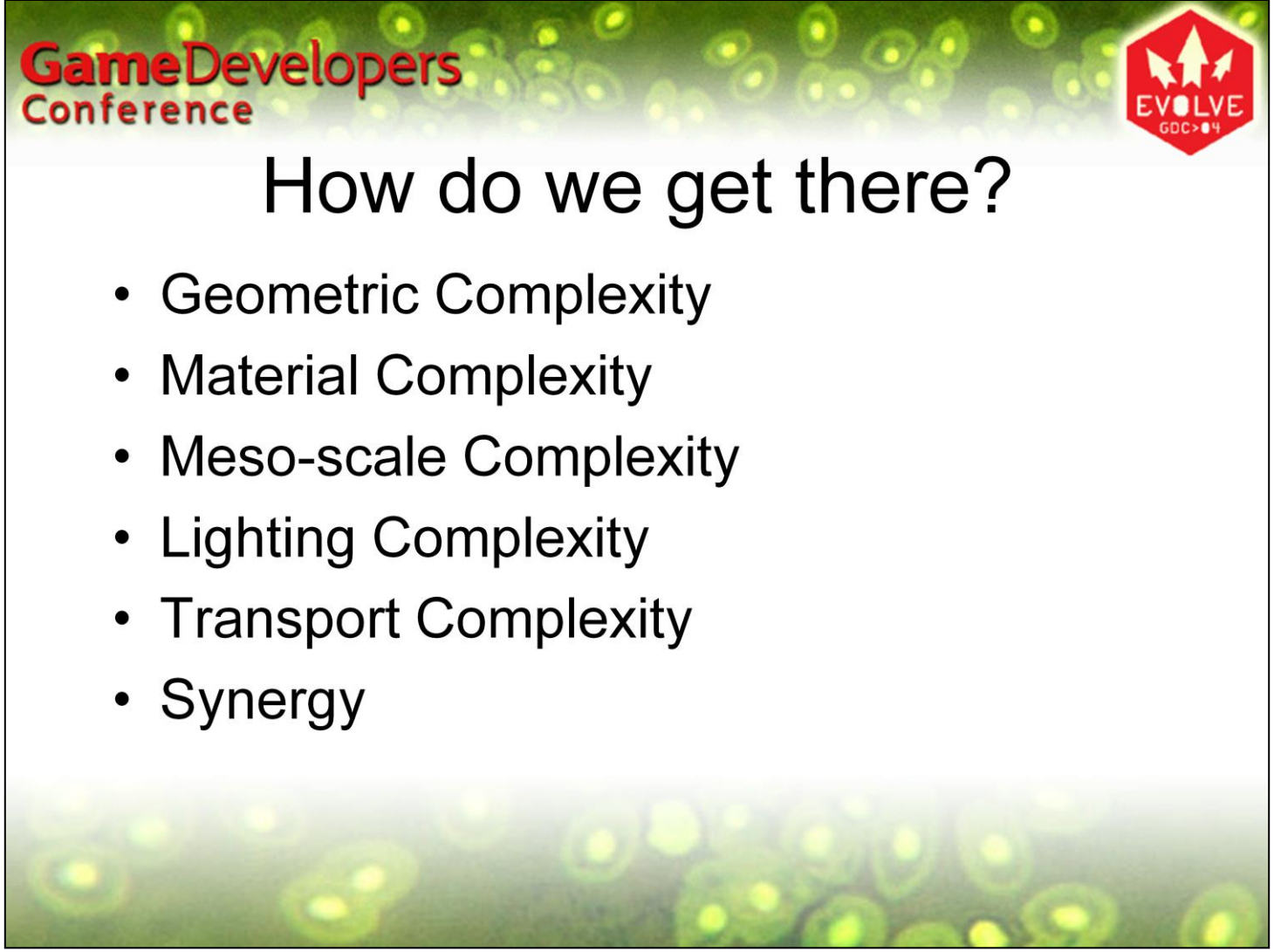
Naty Hoffman

Motivation

We see here an example of a real-world scene which has a lot of visual complexity and richness. Generating synthetic images that come close to this is an extremely challenging problem.

Having them animate and respond to a users control is even more daunting.

We will discuss some of the issues that need to be addressed to meet this challenge.



There are many types of scene complexity which operate individually and in synergy with each other to generate the visual complexity of the resulting image.

Geometric Complexity

- Real-world scenes have a lot of large-scale detail

Material Complexity

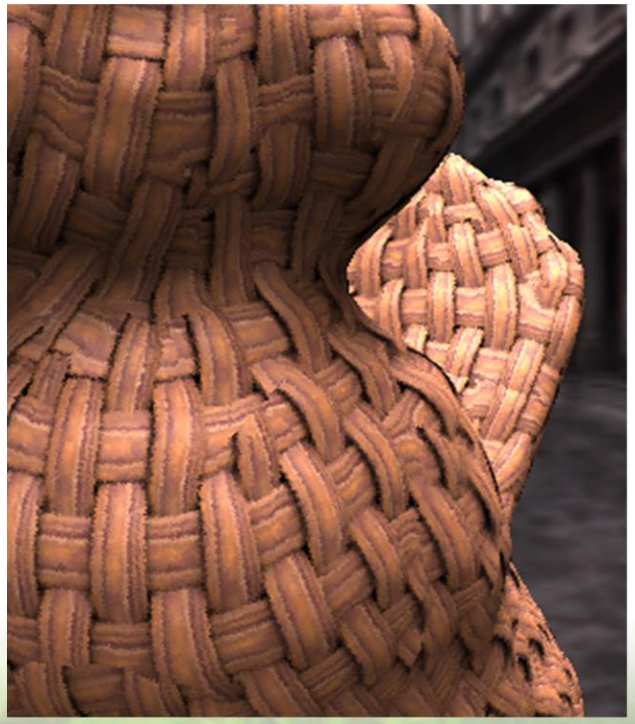
- Models how light interacts with a surface
  - Assumes the “structure” of the material is below the visible scale

GameDevelopers Conference

EVOLVE GDC>04

Meso-Scale Complexity

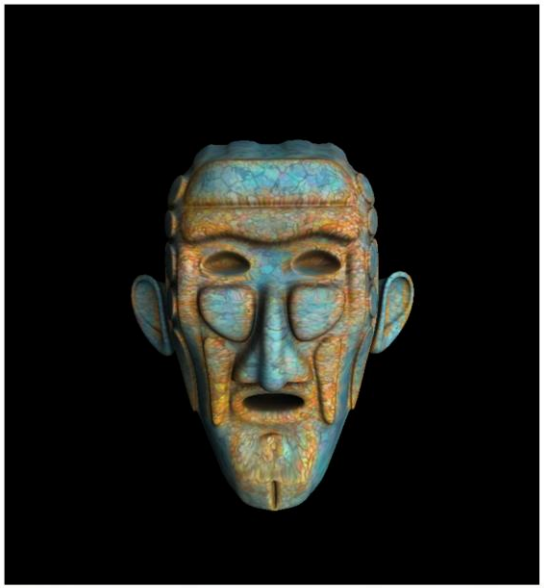
- Variations at a visible scale - not geometry
  - Bump/Roughness/ Twist maps
  - Parallax Mapping/BTF's extreme examples of this



Medium-scale, or meso-scale, lies between invisible details which are handled as materials and larger details which are handled as geometry.

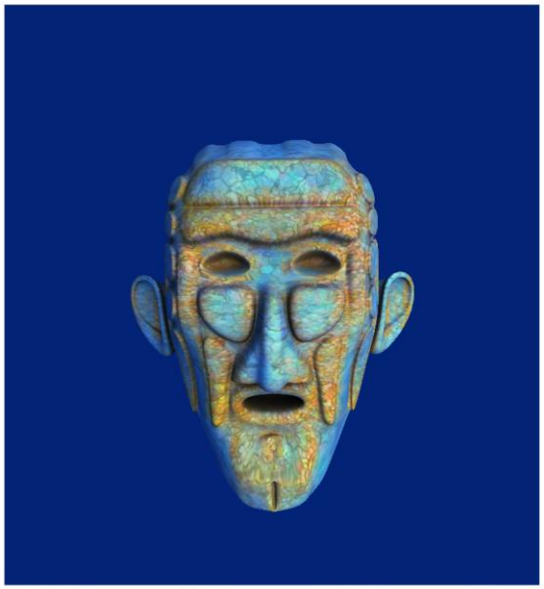
Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general



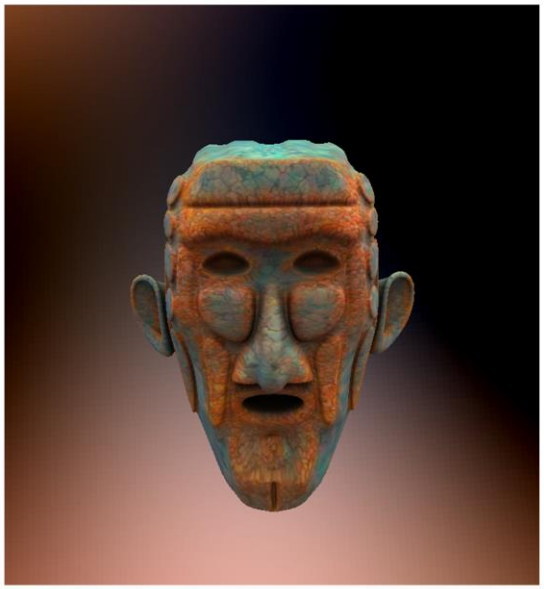
Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general



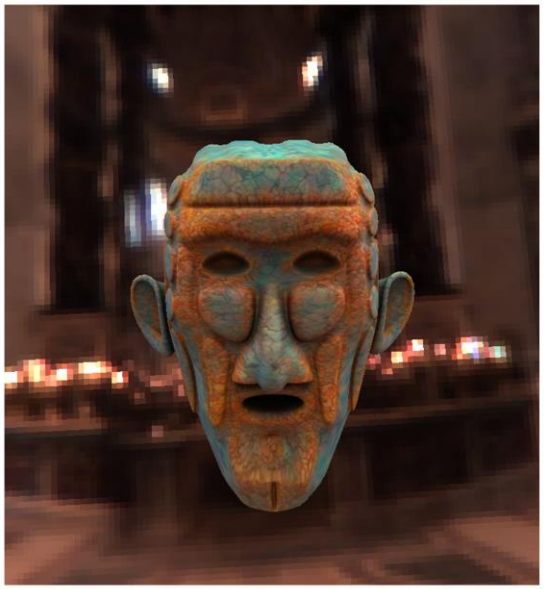
Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - "Smooth" (low frequency) lighting
  - Completely general



Lighting Complexity

- What kind of lighting environment is an object in?
  - Directional/point lights
  - Directional + ambient
  - “Smooth” (low frequency) lighting
  - Completely general

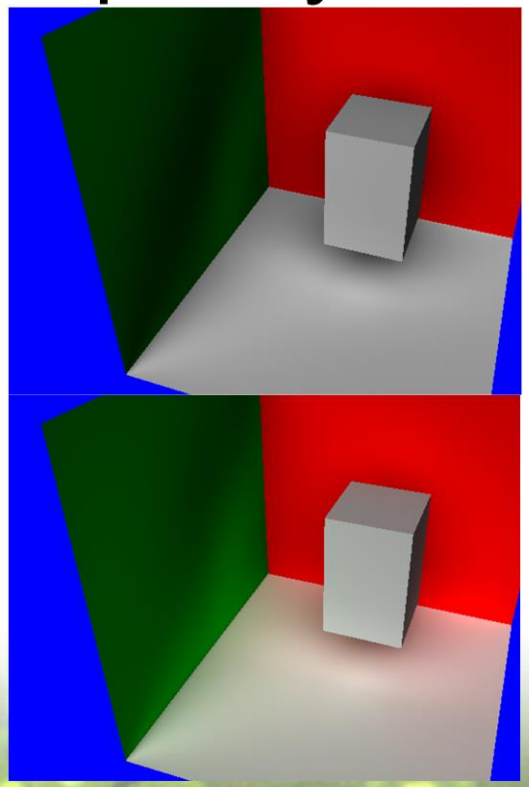


Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)

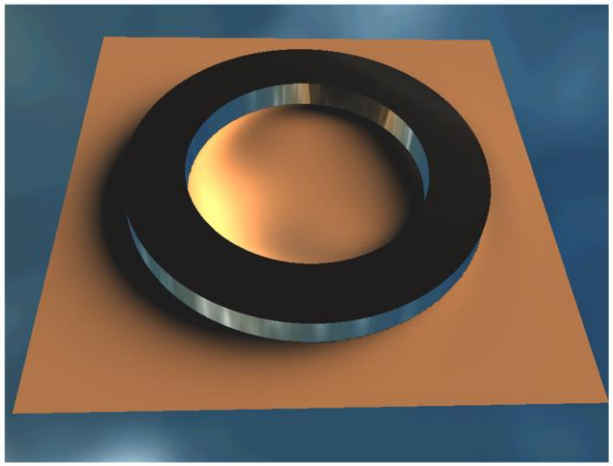
Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



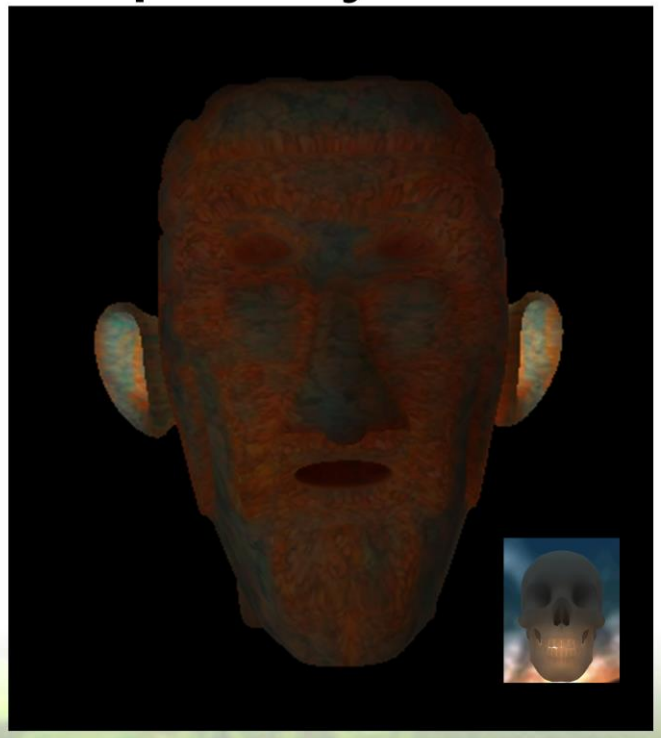
Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



Transport Complexity

- How light interacts with objects/scene at a visible scale
  - Shadows
  - Inter-reflections
  - Caustics
  - Translucency (subsurface scattering)



Some of all of this

- Real scenes have all of these forms of complexity
- High realism on one is not necessarily interesting without the others
  - Complex materials lit by single point light
  - Complex lighting environments on diffuse surfaces with no shadows

In This talk

- Focus is on material complexity
- Also talk a little bit about
  - Meso-scale complexity
  - Light complexity
- As they affect materials
  - No parallax mapping / BTF's

In This Talk

- Translucency
  - Only at microscopic scales, not at visible scales (where light leaves from a different location than it enters)
- No transport complexity

GameDevelopers Conference

EVOLVE GDC>04

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics

The material models in this talk are based on geometric optics (light treated as particles.) Quantum and Wave optics model light in a different manner. Some common visual effects can only be correctly modeled with one or the other:

GameDevelopers Conference

EVOLVE GDC>04

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics
  - Diffraction

Diffraction patterns (bright colors you see when you shine a light on a CD) are due to the physical structure of the material (pits in the CD) being at the scale of the wavelength of light.

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics
  - Diffraction
  - Iridescence

Iridescence from soap bubbles or some insect wings is caused by thin films on the surface and requires a wave model of light.

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics
  - Diffraction
  - Iridescence
  - Polarization

Polarization refers to light waves being preferentially oriented in a certain direction.

GameDevelopers Conference

EVOLVE GDC>04

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics
  - Diffraction
  - Iridescence
  - Polarization
  - Fluorescence

Fluorescence is the changing of the wavelength of light that leaves the surface (from the wavelength that arrives) – “black lights” are an example of this (certain materials will reflect light that is incoming in a part of the spectrum that is not visible into light in the visible spectrum.)

GameDevelopers Conference

EVOLVE GDC>04

In This Talk

- Materials discussed are purely based on Geometric Optics
- No quantum or wave optics
  - Diffraction
  - Iridescence
  - Polarization
  - Fluorescence
- Steady state physics
  - No transient effects such as phosphorescence

We will also assume that the physics happens at a steady state – energy is not stored and released later (like “glow in the dark” toys.)

GameDevelopers Conference

EVOLVE GDC>04

Surface Types



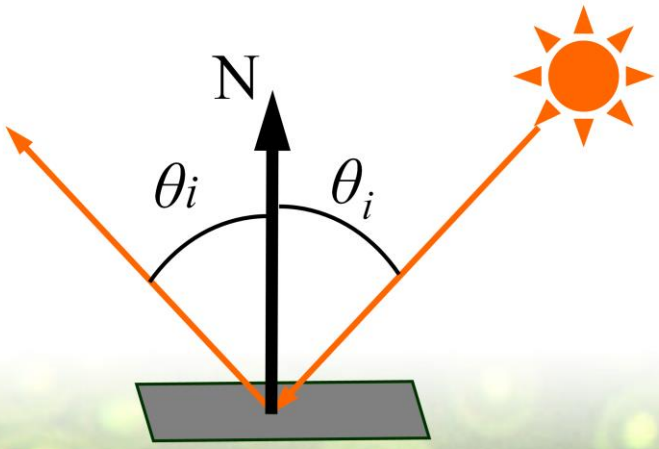





In this section we will go over the basic material types that can be seen on common objects, and qualitatively describe the physical phenomena underlying their salient visual characteristics.

Smooth (Mirror)

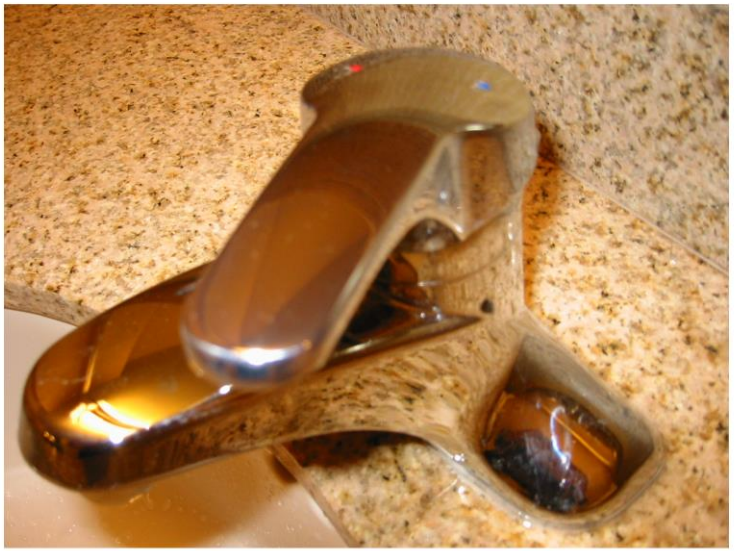
- Perfectly smooth surface
- Incident ray of light only reflected in one direction



The reflection direction is the light direction mirrored about the surface normal. Note that the light direction is the direction to the light, not the direction the light is going. This is a little confusing but it is accepted practice and works better for implementation as well.

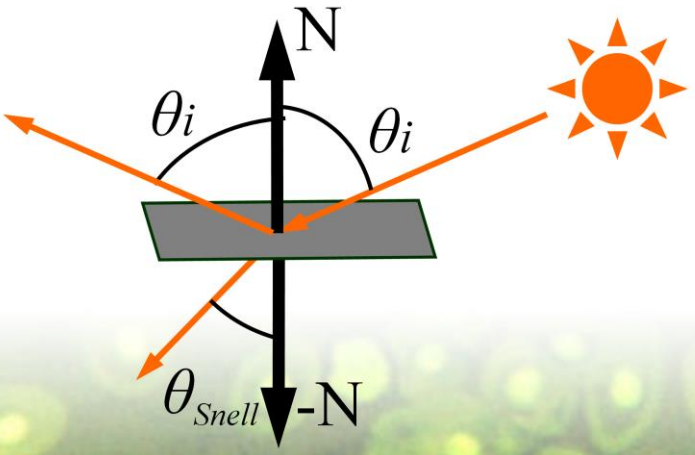
Smooth (Mirror)

- Lights reflected as tiny bright spots
- Fine environment details can be seen in reflection



Smooth (Mirror)

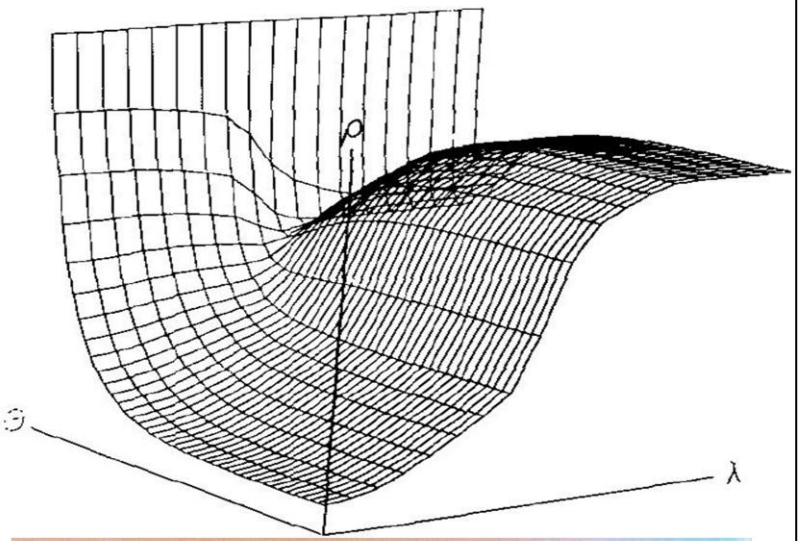
- Light which is not reflected is *refracted*
- Refraction direction obeys *Snell's law*
  - Depends on refractive index of material



Note that the refractive index depends on the light's wavelength as well as on the material.

Smooth - Fresnel

- Reflectance depends on
  - Incidence angle (goes to 100%)
  - Refractive index (depends on wavelength)
  - Polarization

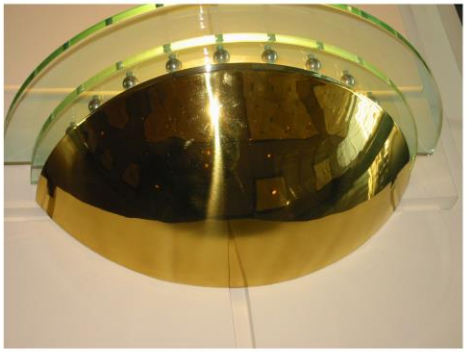


IMAGES BY R. COOK AND K. TORRANCE

The proportion of reflected vs. refracted light obeys the Fresnel equations. As the incidence angle increases (going to more glancing angles), the reflectance increases until it is 100% at all wavelengths. Note that this is not monotonic, there is a dip before it goes up (causing the shift to blue just before it goes white) but the main trend is upwards. Most CG ignores polarization, but it can be significant in some cases (skylight is polarized).

Smooth Metal

- Refracted light absorbed; converted to heat
- Very reflective (50%+ at visible wavelengths)
- Some metals have almost constant reflectance over visible spectrum
  - Steel ~55%, Silver and Aluminum >95%
- Other metals have strong wavelength dependence at normal incidence
  - Gold, Copper
  - Usually less reflective at short (blue) wavelengths



Most colored metals have a yellow or red tint. All reflectance value are at normal incidence.

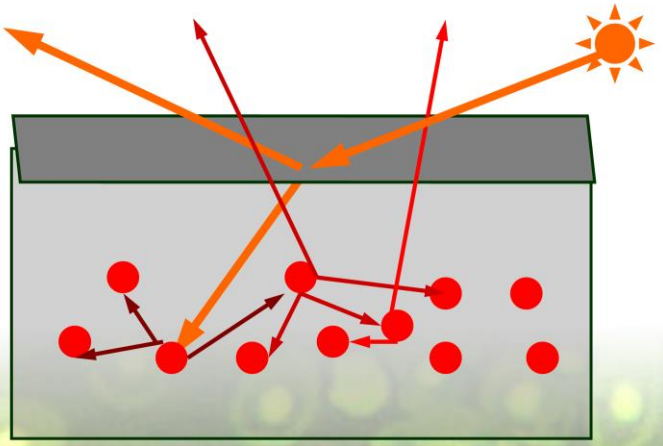
Smooth Dielectric

- Low reflectance (water, glass, plastic, ceramic, etc. ~5%)

Like other reflectance numbers, these are at normal incidence – at glancing angles it is higher.

Smooth Dielectric

- Refracted light continues inside the material, being scattered by impurities until it is absorbed or re-exits the surface



Smooth Dielectric - Scattering

- In some materials (milk, wax, etc.), much of the scattered light will exit at some distance from the original point of entry. Techniques to render these *translucent* materials are outside the scope of this talk

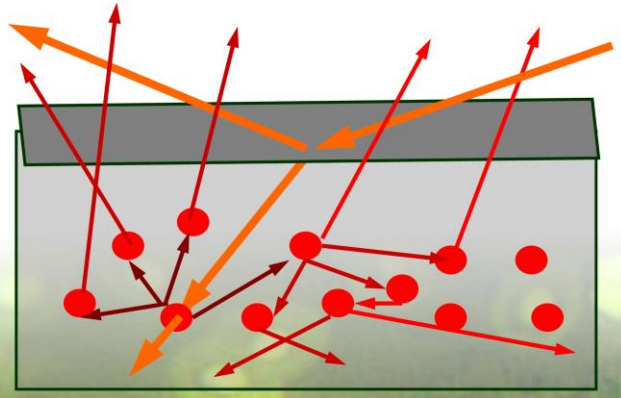


IMAGE BY H. W. JENSEN

GameDevelopers Conference

EVOLVE GDC>04

Smooth Dielectric - Scattering

- For many materials, point of exit is close enough to point of entry to be treated as the same point.

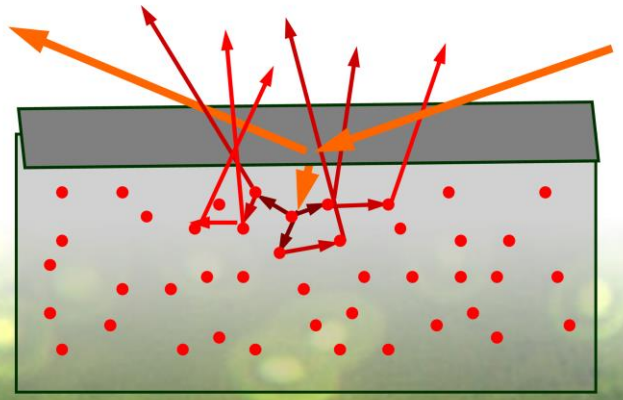


IMAGE BY H. W. JENSEN

Note that the scale is important for issues of subsurface scattering (and many other issues related to reflectance). A tiny marble figurine can have appreciable translucency, where a large statue made of the same material can be treated as non-translucent.

Scattering and Scale

- The validity of this approximation depends on the relative scale of material and geometry

360cm Marble Statue

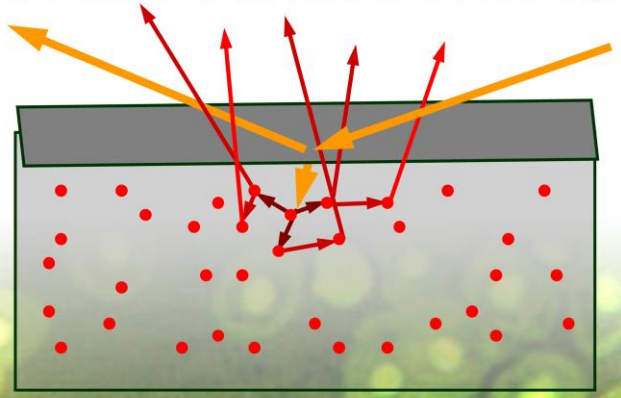
30cm Marble Statue

6cm Marble Statue

IMAGES BY H. W. JENSEN

Smooth Dielectric - Scattering

- The impurities in dielectrics have optical properties unrelated to the surface boundary. They impart to the scattered light a color which is different from that of the specular surface reflection



For example a plastic where red pigment particles are suspended in a clear substrate. In this case the reflected light will be of the same color as the incident light, where the scattered light will be tinted red.

Diffuse / Specular Tradeoff

- Only light which was not reflected is available for scattering
- At glancing angles diffuse reflectance decreases, specular reflectance increases

Diffuse / Specular Tradeoff

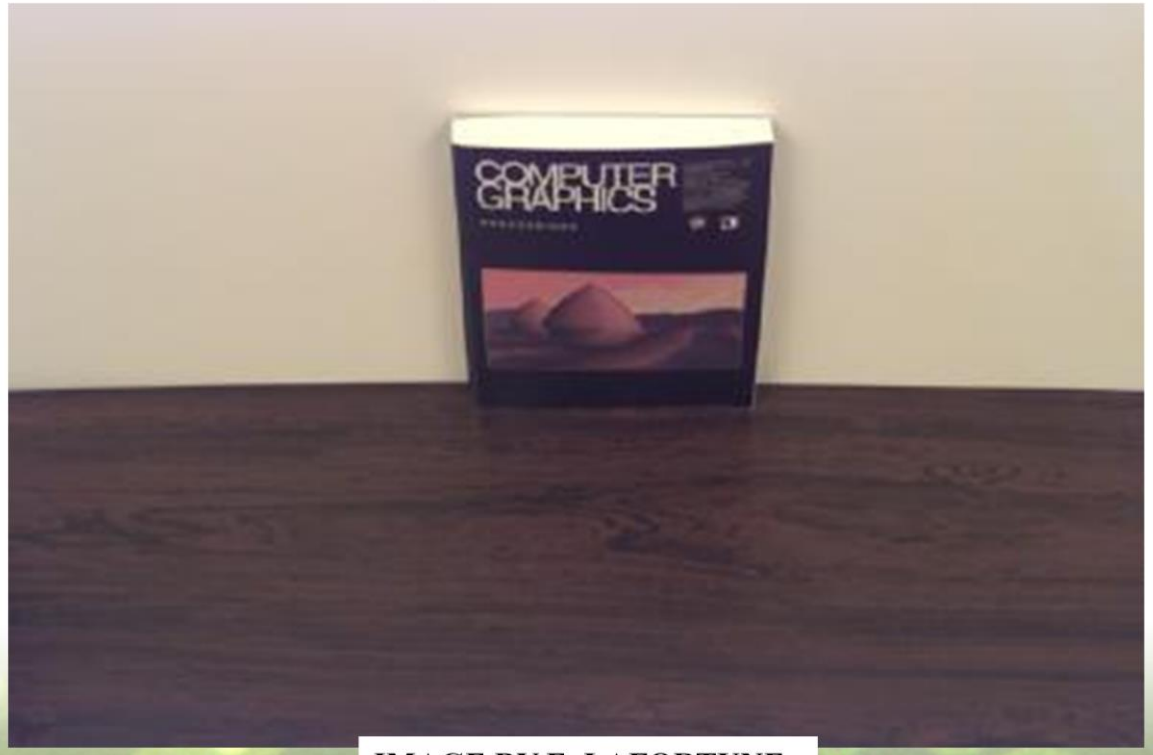


IMAGE BY E. LAFORTUNE

Diffuse / Specular Tradeoff

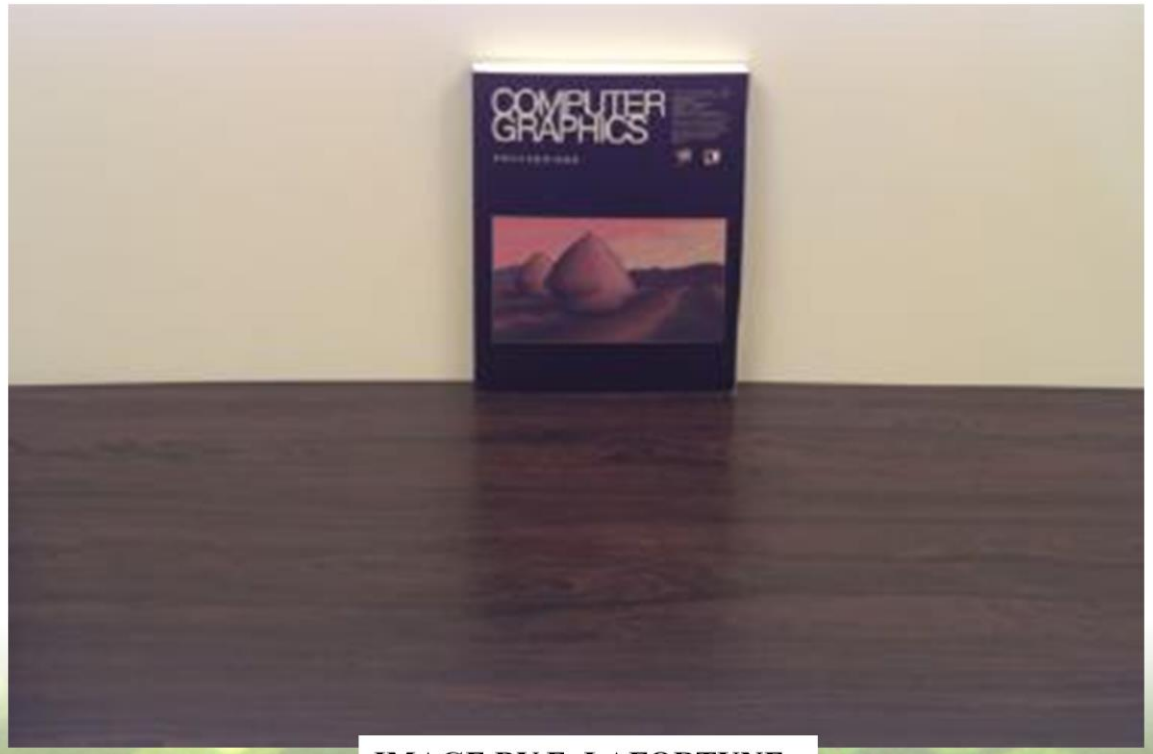


IMAGE BY E. LAFORTUNE

Diffuse / Specular Tradeoff

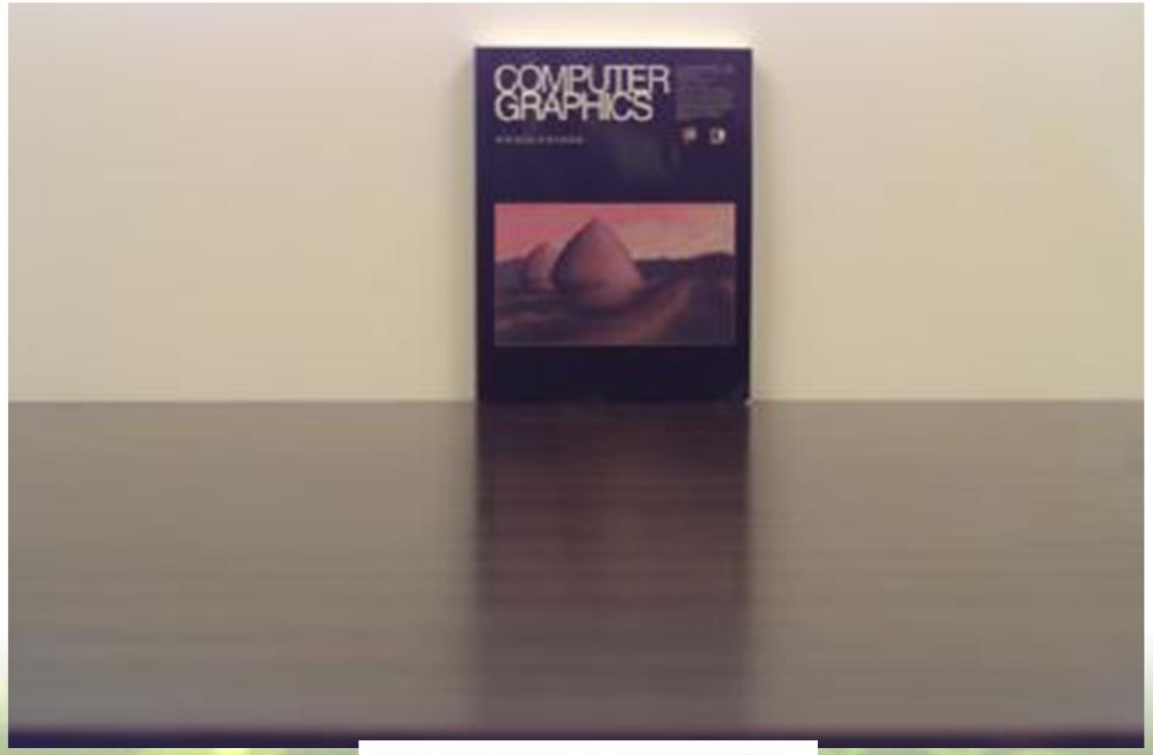


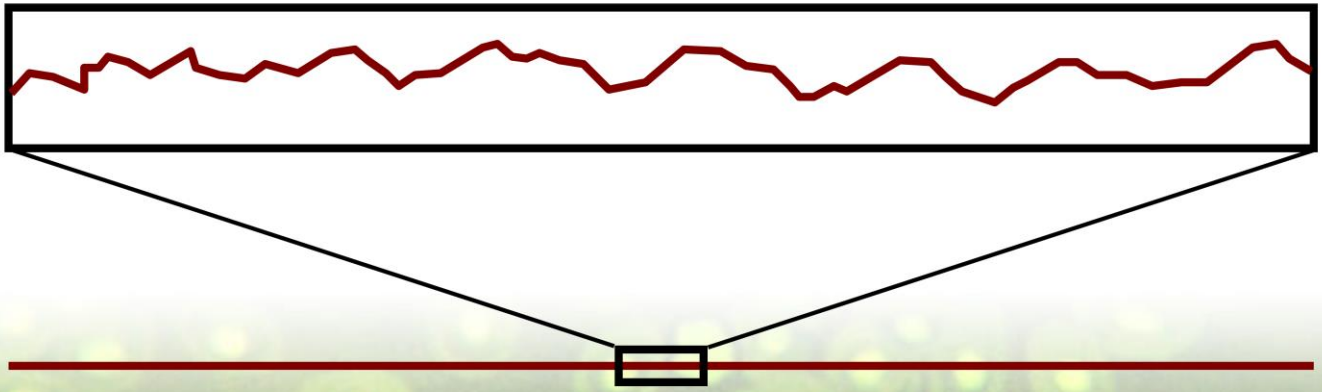
IMAGE BY E. LAFORTUNE

Semi-Rough (Glossy)

- Most surfaces are not flat at all scales

Semi-Rough (Glossy)

- Most surfaces are not flat at all scales
  - Many surfaces which appear flat at visible scales have complex *microscale* structure



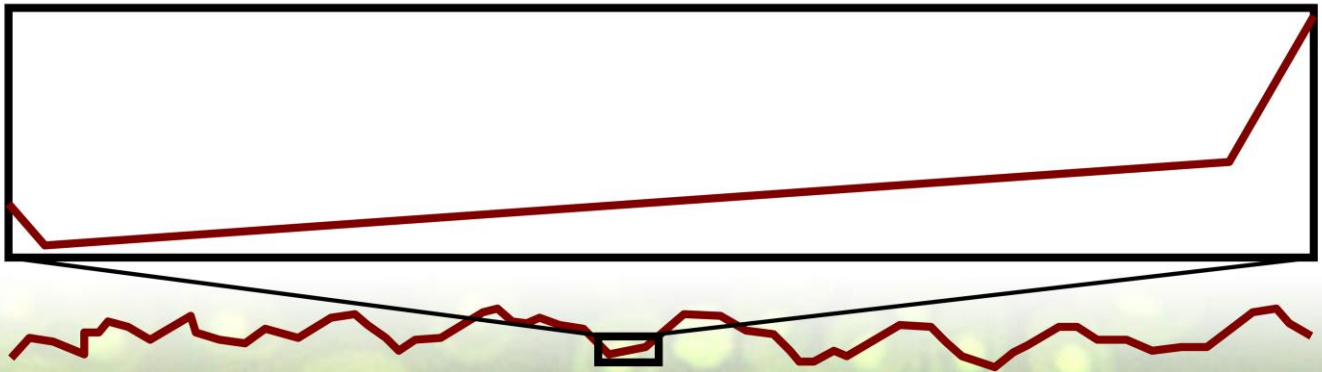
Semi-Rough (Glossy)

- Most surfaces are not flat at all scales
  - Many surfaces which appear flat at visible scales have complex *microscale* structure

If the smallest features are not much larger than a wavelength of visible light (0.4-0.7 micrometer) then *physical optics* comes into play, which is beyond the scope of this talk.

Semi-Rough (Glossy)

- Most surfaces are not flat at all scales
  - Many surfaces which appear flat at visible scales have complex *microscale* structure
  - At smallest scale, can often treat as flat again



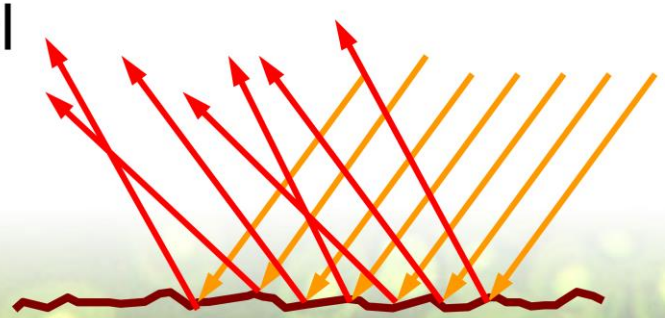
If the surface is locally flat at wavelength scale, then we can treat it as flat (a Fresnel mirror) at that scale.

GameDevelopers Conference

EVOLVE GDC>04

Semi-Rough (Glossy)

- A surface patch contains micro-facets with continuously distributed normals
- Light reflects off facets, ‘spreads out’
- In ‘semi-rough’ surfaces distribution of micro-normals biased to macro-normal



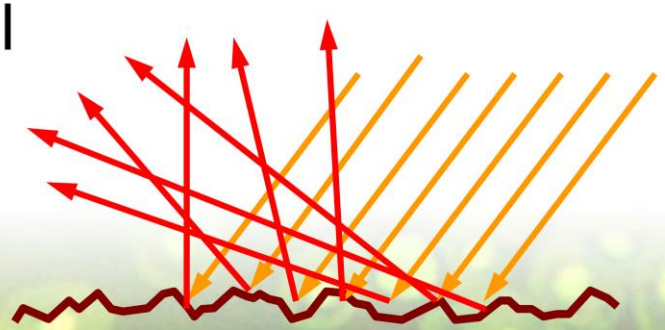
This definition of a ‘semi-rough’ surface is not a standard one in CG, but we find it useful to describe surfaces. Semi-rough surfaces can exhibit a continuum of roughness, here we see a relatively smooth surface where the reflected light is only spread out a little bit.

GameDevelopers Conference

EVOLVE GDC>04

Semi-Rough (Glossy)

- A surface patch contains micro-facets with continuously distributed normals
- Light reflects off facets, ‘spreads out’
- In ‘semi-rough’ surfaces distribution of micro-normals biased to macro-normal

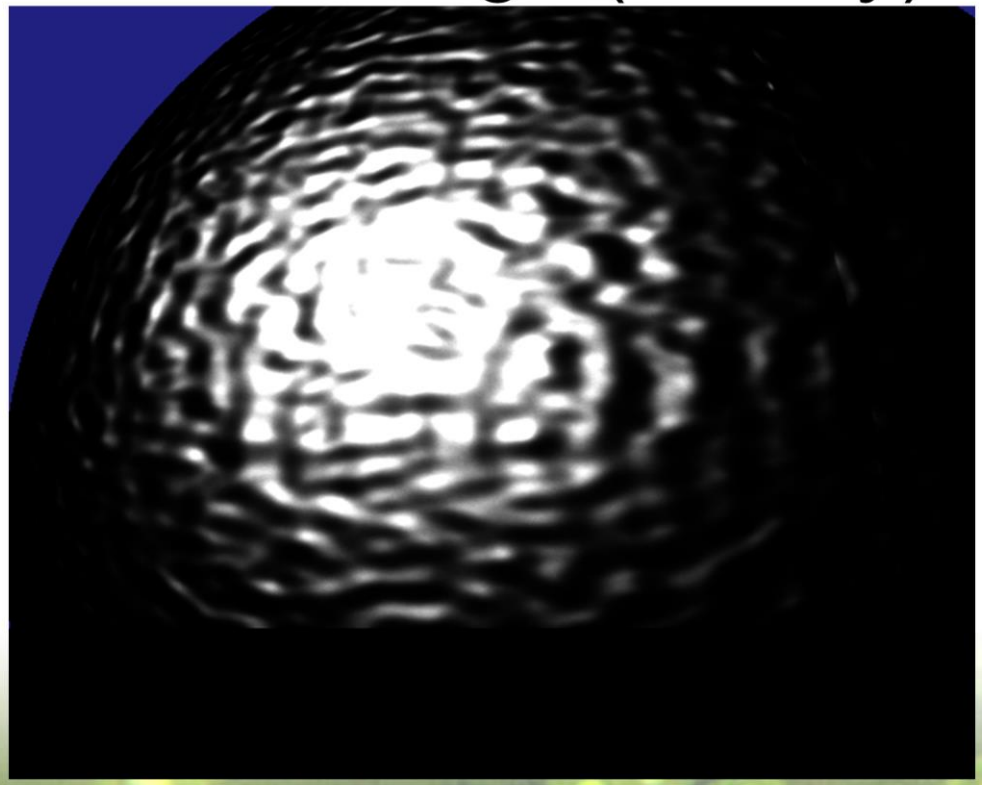


And here we see a rougher surface where the reflected light is spread out more, but it is not completely random.

GameDevelopers Conference

EVOLVE GDC>04

Semi-Rough (Glossy)

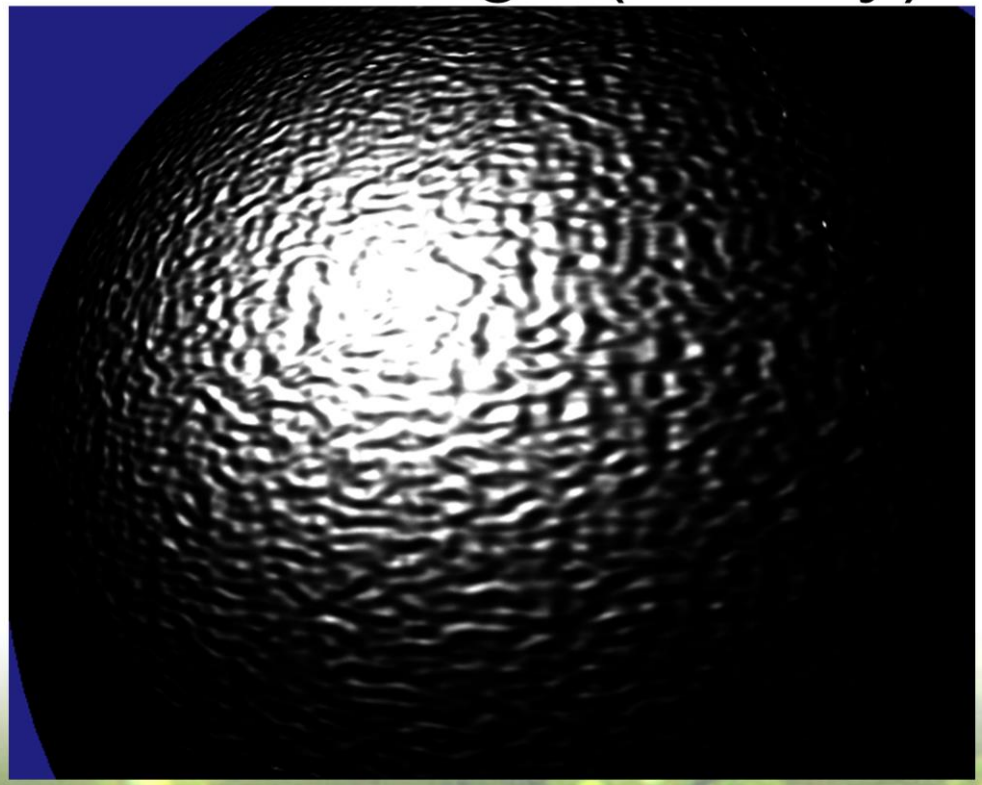


Another way to think about this is to look at a bumpy surface which is locally smooth. Each bump has its own little highlight.

GameDevelopers Conference

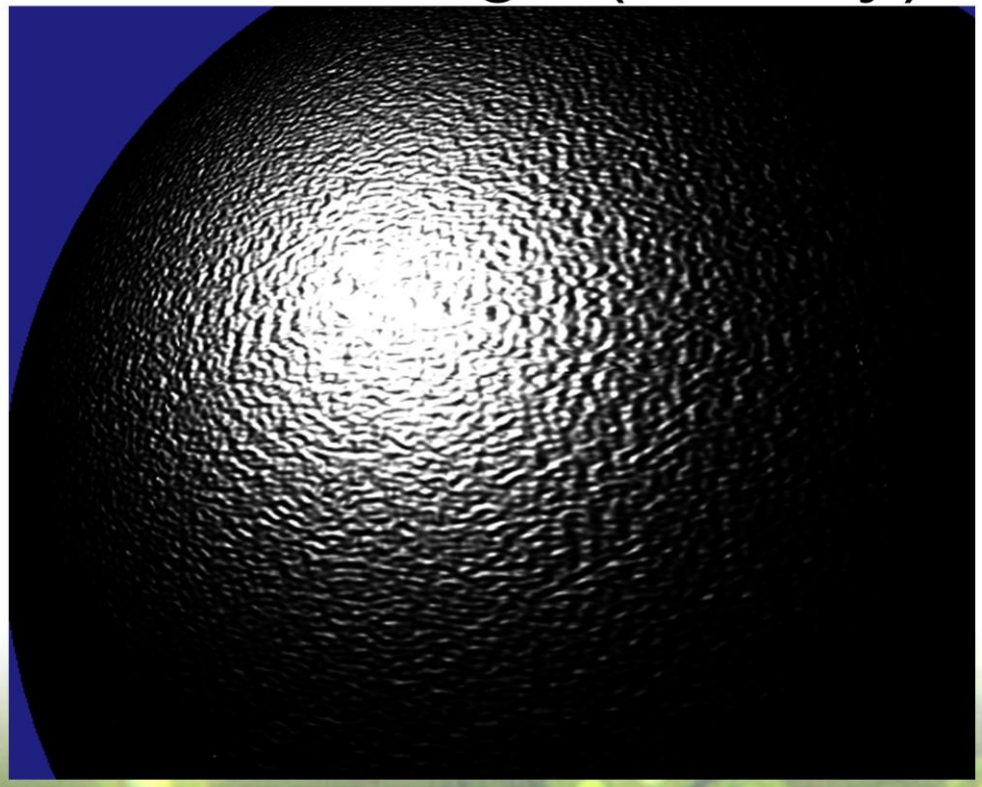
EVOLVE GDC>04

Semi-Rough (Glossy)



Keeping the same normal distribution, we'll make the bumps smaller…

Semi-Rough (Glossy)

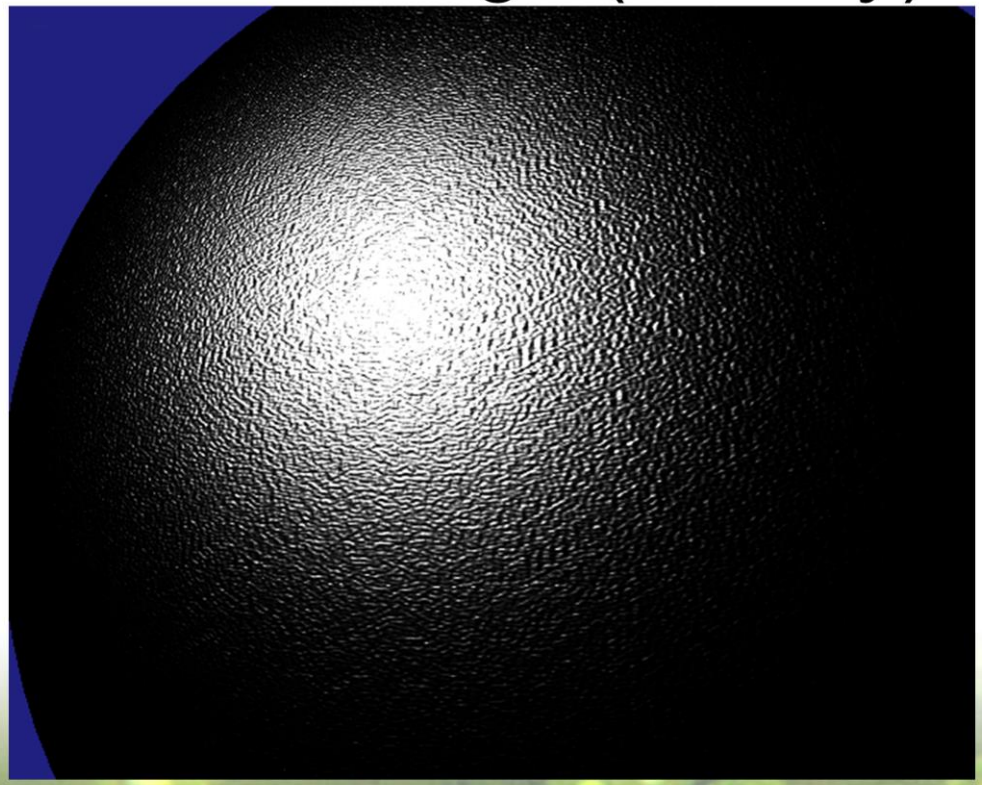


…and smaller…

GameDevelopers Conference

EVOLVE GDC>04

Semi-Rough (Glossy)

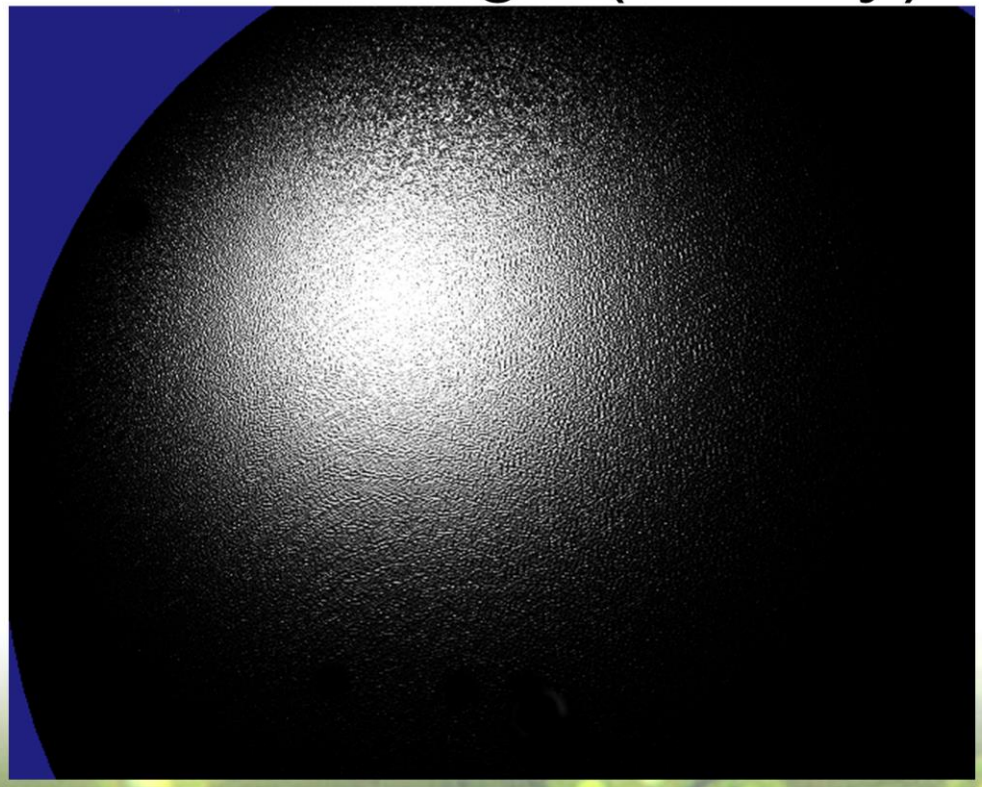


…and smaller…

GameDevelopers Conference

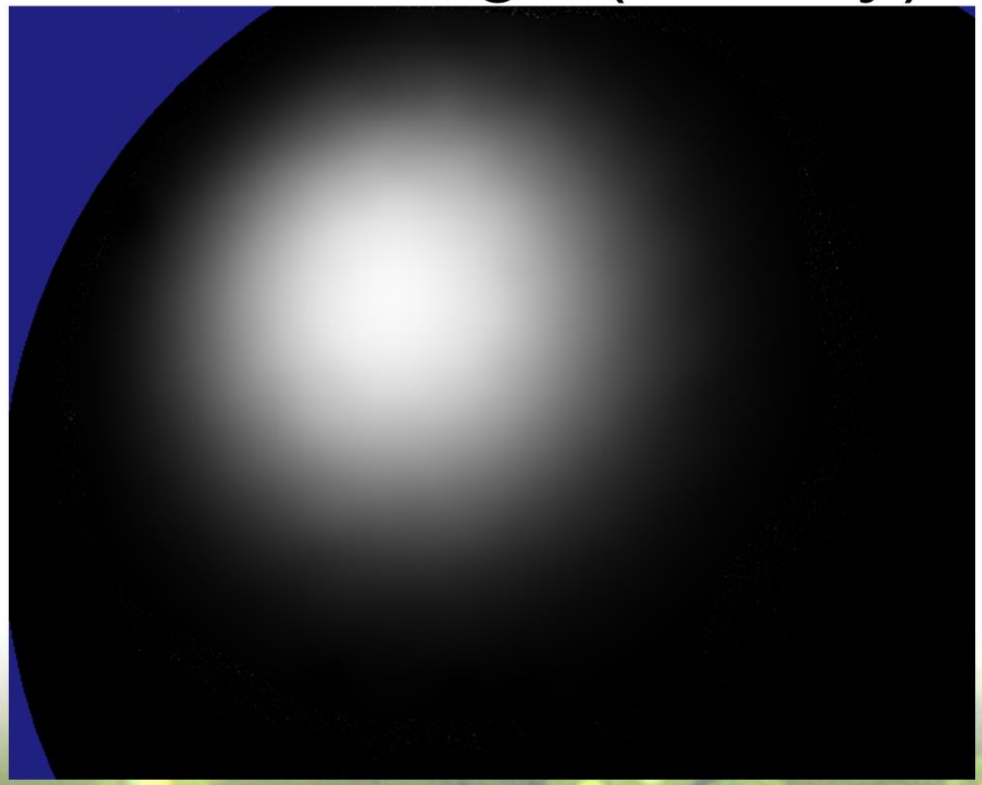
EVOLVE GDC>04

Semi-Rough (Glossy)



…and smaller…

Semi-Rough (Glossy)



…until eventually the bumps become too small to see—but they still affect the appearance of the surface.

Semi-Rough (Glossy)

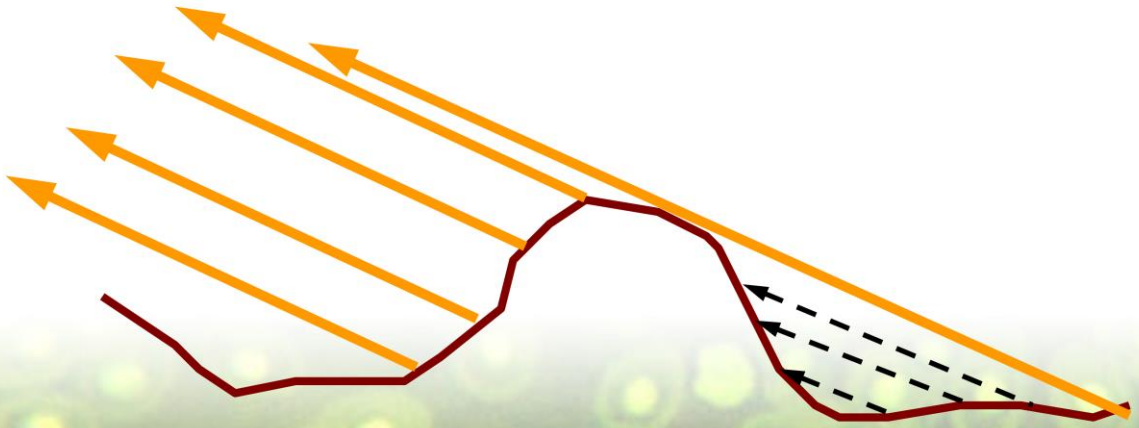
- Micro-geometry also exhibits:
  - Shadowing

Besides having a continuous distribution of normals (rather than a single surface normal), the micro-geometry affects the reflectance in a few other ways.

Shadowing is some micro-facets blocking light from others. This depends on the exact shape of the micro-geometry, but not directly on the normal distribution. The area with the black dashed arrows is shadowed.

Semi-Rough (Glossy)

- Micro-geometry also exhibits:
  - Shadowing
  - Masking



Masking is some micro-facets ‘hiding’ others from the view position. This also depends on properties of the microgeometry other than the normal distribution. The area with the black dashed arrows is shadowed.

GameDevelopers Conference

EVOLVE GDC>04

Semi-Rough (Glossy)

- Micro-geometry also exhibits:
  - Shadowing
  - Masking
  - Interreflections

And finally the light that was masked doesn't just go away – some of it is reflected again and bounces off the surface. Light may undergo several bounces before it reaches the eye.

Semi-Rough (Glossy)

- Lights reflect as highlights
- Blurry reflection of environment

Here we define the highlight as the single-bounce reflection – this is not a standard use of the term but is useful in discussing reflectance. The highlights size and shape mostly depends on the distribution of microfacet normals, and their color is that of the surface material boundary.

Semi-Rough (Glossy)

- The reflectance of a surface is a function both of its surface physics and its microscale structure

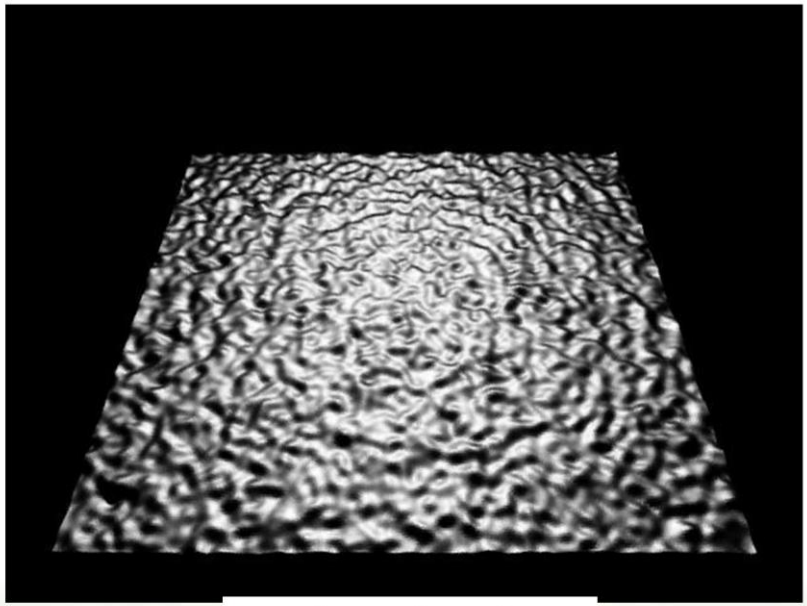


IMAGE BY S. WESTIN

Semi-Rough Metal

- Reflectance is dominated by the highlight
- Multiple-bounce reflections create an additional diffuse reflection
  - More strongly colored (each bounce increases saturation)

Semi-Rough Dielectric

- Highlight tend to be weaker in dielectric surfaces due to lower reflectance
  - For the same reason, multiple-bounce reflections are less noticeable
  - Diffuse mostly due to subsurface scattering

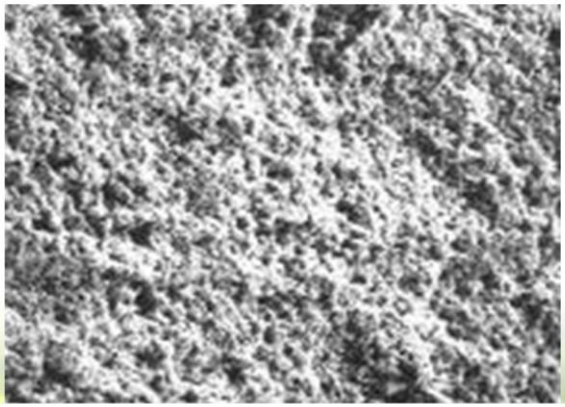
Like dielectric smooth surfaces, semi-rough dielectric surfaces also exhibit a tradeoff between diffuse and specular reflection at glancing angles (the only difference is that the specular reflection is less sharp).

GameDevelopers Conference

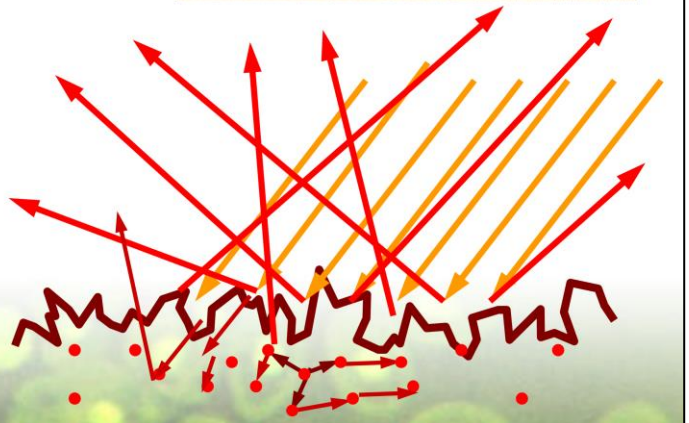
EVOLVE GDC>04

Rough Dielectric

- Normal distribution random
- Diffuse with some retroreflection



PHOTOMICROGRAPH BY T. HIMLAN

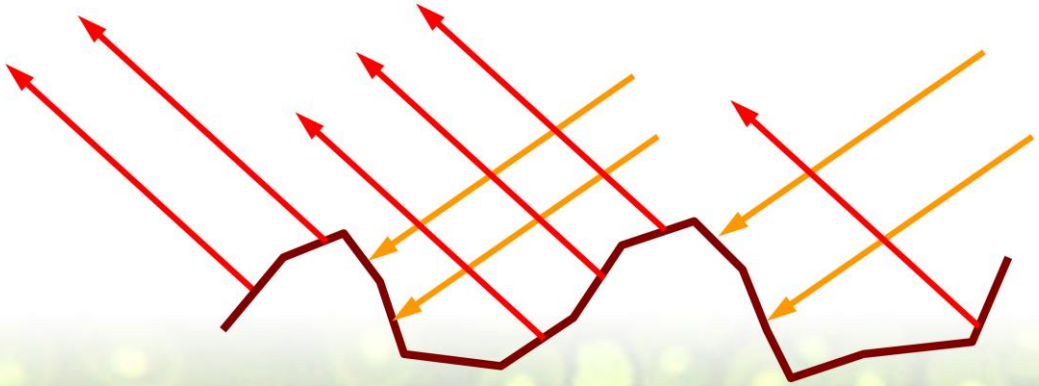


Of course, we can also have rough metal surfaces, but they are less interesting in themselves and can be treated as an extreme case of semi-rough metals.

Rough dielectrics include surfaces such as dust, rough clay. These are surfaces that we think of as “matte” or “diffuse”. Such surfaces appear flat (example; the full moon) and contrary to expectations, do not obey Lambert’s law (the retroreflection tends to counteract the cosine falloff). Here we have a combination of extreme surface roughness and sub-surface scattering.

Rough Dielectric

- Retroreflective tendencies caused by foreshortening of shadowed parts of surface when eye near light



If we imagine each facet being essentially Lambertian (which is a reasonable approximation to what is going on in these surfaces), then when the light and view directions are very different the surfaces we can see are the ones which are more dimly lit.

Rough Dielectric

- Retroreflective tendencies caused by foreshortening of shadowed parts of surface when eye near light

When the light and view directions are similar the surfaces we can see are the ones which are more strongly lit.

Rough Dielectric

- Lights reflect diffusely
- No environment details visible

Structured Surfaces

- So far the surfaces we have seen have been flat or had random, unstructured microgeometry
- Surfaces with regular, structured microgeometry exhibit interesting reflective characteristics

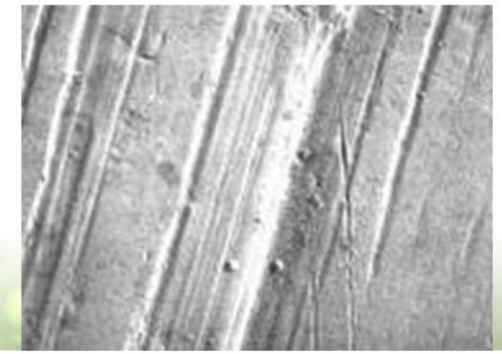
These surfaces may be metallic or dielectric; since the microstructure is the interesting thing about them we will not distinguish the two cases.

GameDevelopers Conference

EVOLVE GDC>04

Anisotropic Surface

- When the surface microscale is *anisotropic* (not the same in all directions), then the reflectance exhibits directionality
- Examples: surfaces such as wood, brushed metal which have an oriented structure



PHOTOMICROGRAPH BY T. HIMLAN

One way in which the microgeometry can be structured is if it is not the same in all directions – if it is *anisotropic*. This causes the reflection to exhibit directional behavior.

Anisotropic Surface

- Highlights are stretched
- Environment is directionally blurred

GameDevelopers Conference

EVOLVE GDC>04

Retroreflective Surface

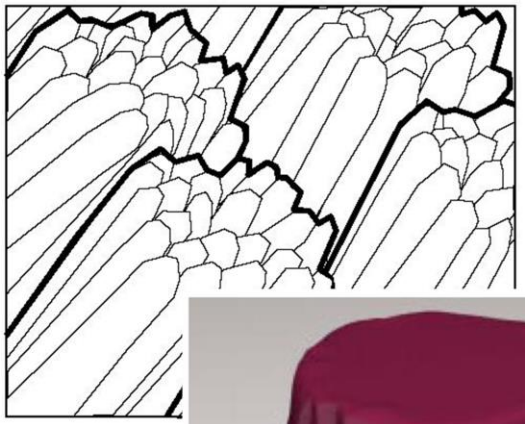
- Microgeometry is constructed to reflect most of the light back in the incident direction

Most retroreflective materials are artificial materials designed for use in street signs, projection screens, etc. Some rare natural surfaces also exhibit strong retroreflective behavior (such as cat's eyes).

GameDevelopers Conference

EVOLVE GDC>04

Other Structured Surfaces



IMAGES BY M. ASHIKHMIN, S. PREMOŽE AND P. SHIRLEY

Fabrics are usually structured surfaces, and many of them exhibit interesting reflective properties.

Reflectance Theory

- Radiometric Theory
- The BRDF

Here we will lay the theoretical groundwork needed to understand reflectance from a physical standpoint.

Radiometric Theory

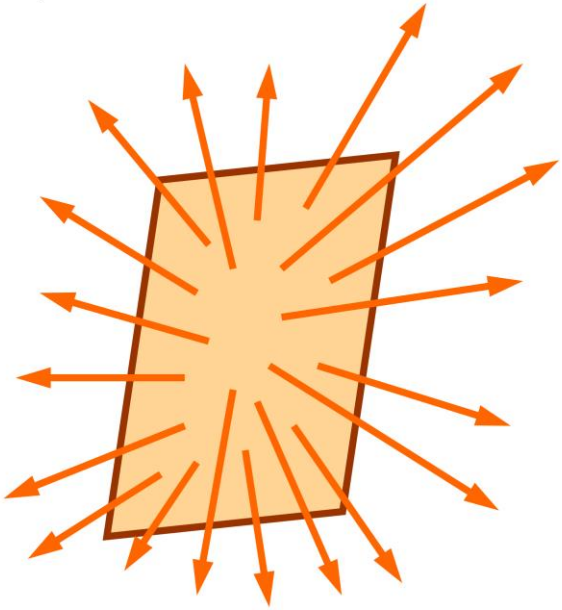
- Radiometry
  - The measurement of radiant energy
  - Specifically electromagnetic radiant energy
  - In CG, specifically energy in the visible portion of the EM spectrum (~380nm-780nm)

GameDevelopers Conference

EVOLVE GDC>04

Radiometric Quantities

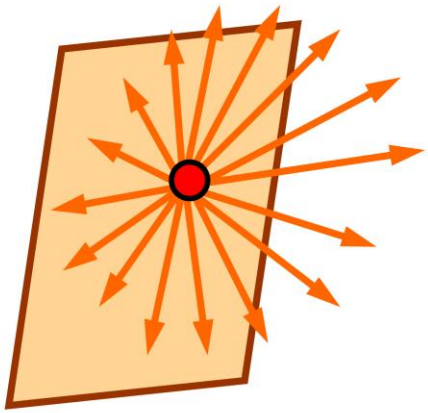
- Radiant Flux Φ
  - Total radiant power
  - Power
  - Watts



In this example, we are looking at a window. The power of all the light pouring through the window (in all directions) is radiant flux.

Radiometric Quantities

- Radiant Exitance (Radiosity) B
  - Radiant power *exitant* from a point
  - Power per surface area
  - Watts / meter^2



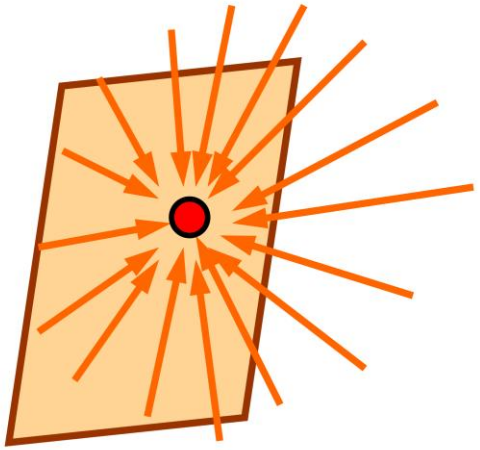
The power *exitant* (coming out of) a single point on the window is radiant exitance (also called radiosity) – it can be thought of as the surface density of excitant radiant flux.

GameDevelopers Conference

EVOLVE GDC>04

Radiometric Quantities

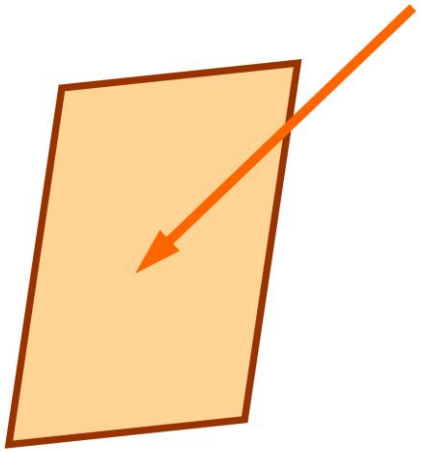
- Irradiance E
  - Radiant power *incident* to a point
  - Power per surface area
  - Watts / meter^2



Irradiance is very similar to radiosity, but it measures *incident* (incoming) light rather than exitant light.

Radiometric Quantities

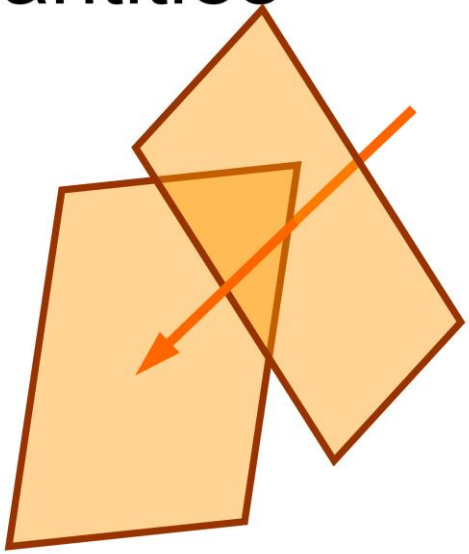
- Radiance L
  - Radiant power in a single ray
  - Power per (projected) surface area per solid angle
  - Watts / (meter^2 steradian)



Unlike radiosity and irradiance, radiance is not tied to a surface, but to a ray. Radiance is important because the final pixel RGB color is basically derived directly from the radiance in the rays through the pixel sample locations.

Radiometric Quantities

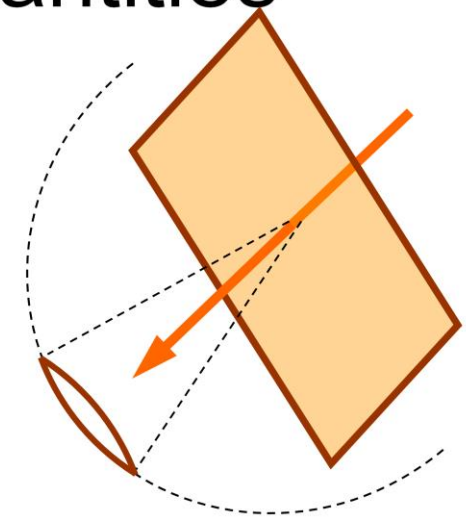
- Radiance L
  - Radiant power in a single ray
  - Power per (**projected**) surface area per solid angle
  - Watts / (meter^2 steradian)



The surface area here (unlike the other quantities) is projected surface area, or surface area perpendicular to the ray direction.

Radiometric Quantities

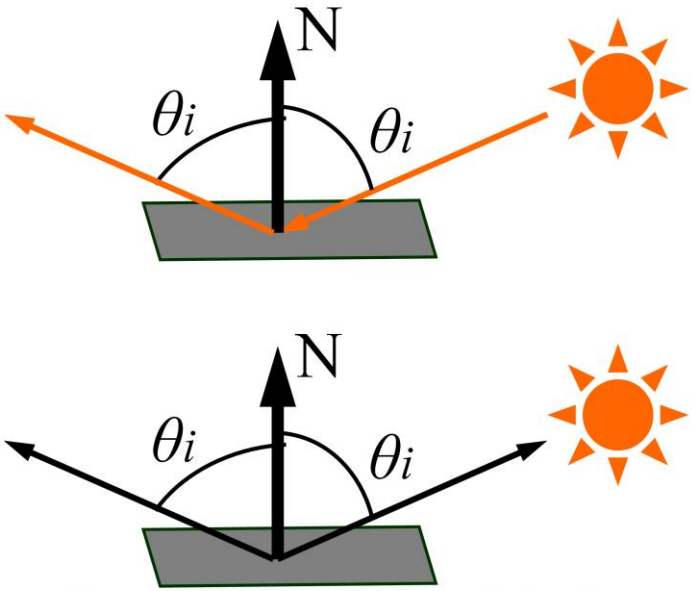
- Radiance L
  - Radiant power in a single ray
  - Power per (projected) surface area per **solid angle**
  - Watts / (meter^2 steradian)



A solid angle is a 3D extension of the concept of an angle – it is a sheaf of directions in 3-space. Just as an angle can be measured in radians as the length of an arc on a unit circle, a solid angle can be measured as the area of a patch on a unit sphere. Solid angles are measured in steradians, of which there are 4π in a sphere.

Light Direction

- Until this slide, the light arrows have pointed in the direction the light is going
- From now on, we will use ‘light direction’ vectors which point towards the light



Until now, we have been showing the physics of light bouncing around. From now on, we are talking about the math and implementation and there it is both customary and convenient to have the ‘light direction’ pointing TO the light.

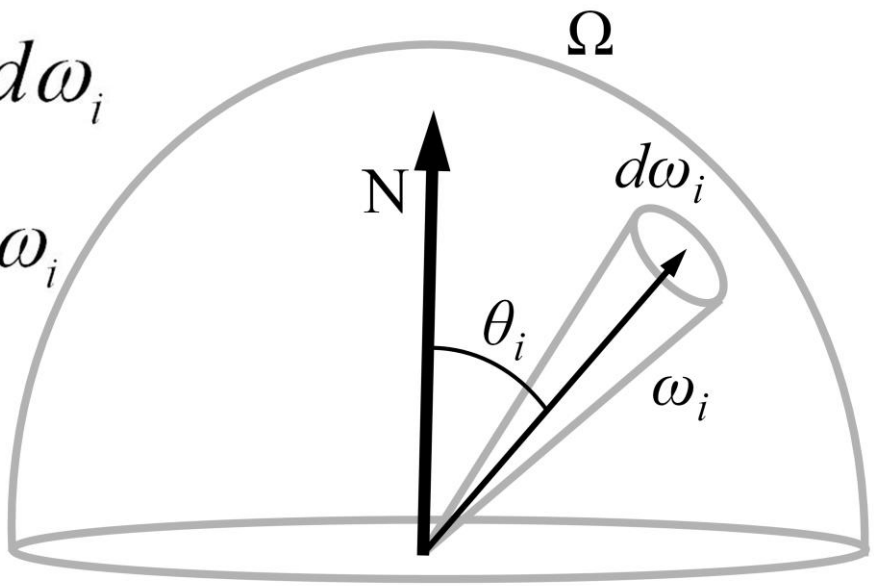
GameDevelopers Conference

EVOLVE GDC>04

Radiance and Irradiance

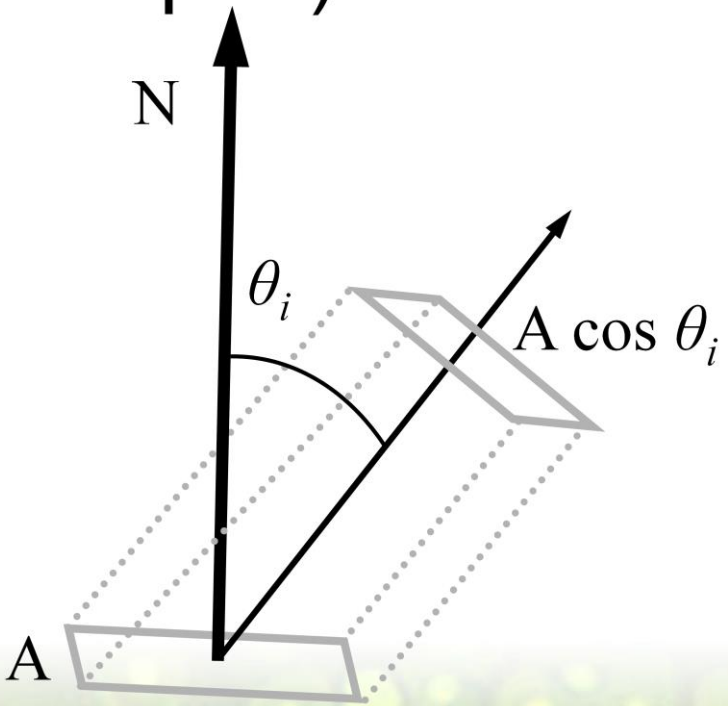
$$dE = L_i \cos\theta_i d\omega_i$$

$$E = \int_{\Omega} L_i \cos\theta_i d\omega_i$$



Given a patch of incident directions with solid angle *dωi* (so small it can be represented by a single incident direction *ωi*), then radiance incident from *ωi*, times *dωi*, times cosine of *θi* (angle between *ωi* and the normal N) gives the patch's irradiance contribution. Integrating over the hemisphere Ω (centered on N) gives total irradiance. Note that to illuminate (contribute irradiance) a light source needs non-zero radiance and non-zero solid angle. Note: the cosine is only valid over the hemisphere and needs to be clamped to 0 outside it (true for all cosine terms in this talk).

The (Clamped) Cosine Factor

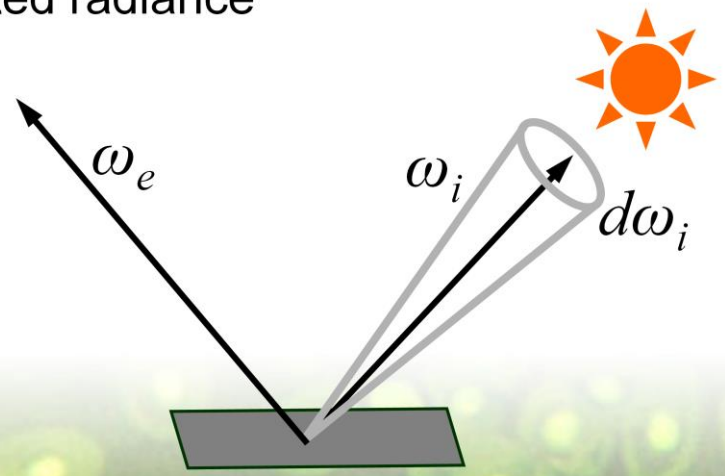


Where did the (clamped) cosine factor come from? The cosine is there because radiance is defined relative to an area perpendicular to the ray, and irradiance is defined relative to an area parallel to the surface. Another way of looking at it is that the same irradiance, coming in at a more oblique angle, contributes a smaller amount to the irradiance because it is ‘spread out’ more. We can also see here why it is clamped – if theta-I is under the surface then there is no contribution.

The BRDF

- Bidirectional Reflectance Distribution Function
  - Ratio of irradiance to reflected radiance

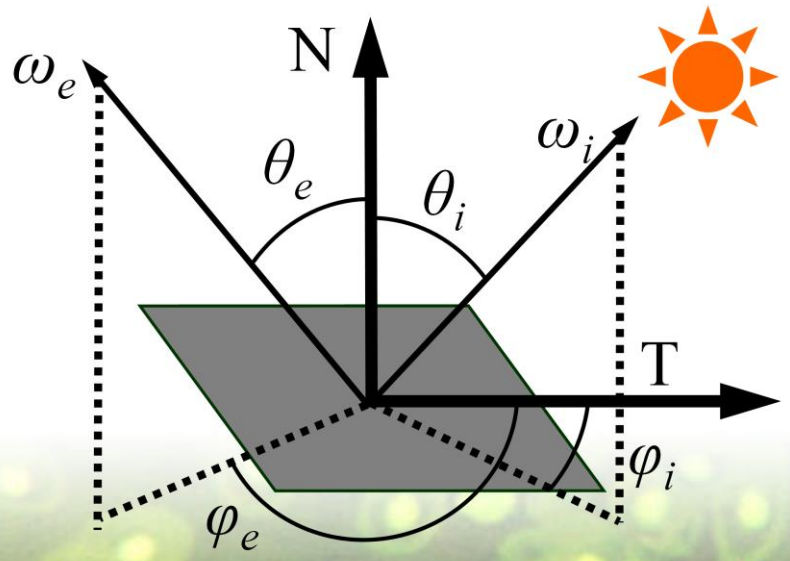
$$f_r(\omega_i, \omega_e) = \frac{dL_e(\omega_e)}{dE(\omega_i)}$$



ωi is the direction to the incident irradiance, and *ωe* is the direction to the exitant reflected radiance. For every such pair of directions, the BRDF gives us the ratio between incident irradiance and exitant radiance. Since the incident direction and the excitant direction are both 2D quantities (a common parameterization is to use two angles: elevation *θ* relative to the surface normal and rotation *φ* about the normal), the BRDF is a 4D function. Note that the incident and exitant directions are usually defined in the *local frame* of the surface.

The BRDF

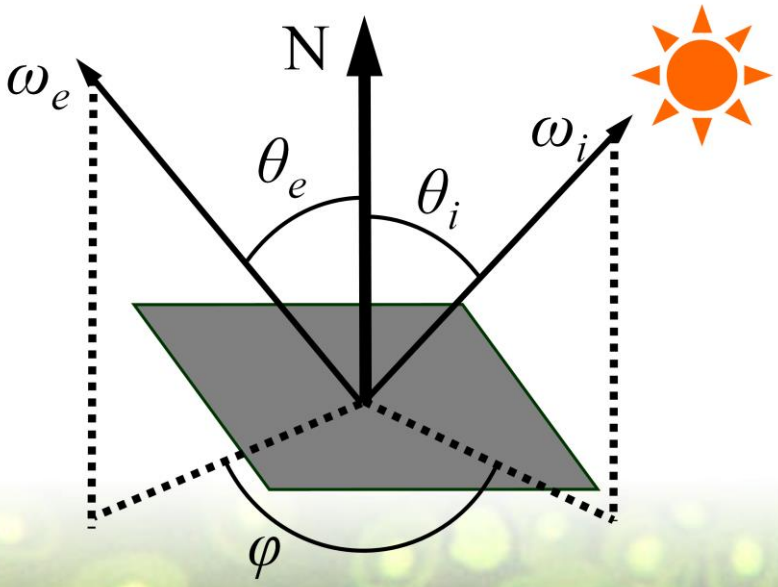
- Incident, excitant directions defined in the surface's *local frame* (4D function)



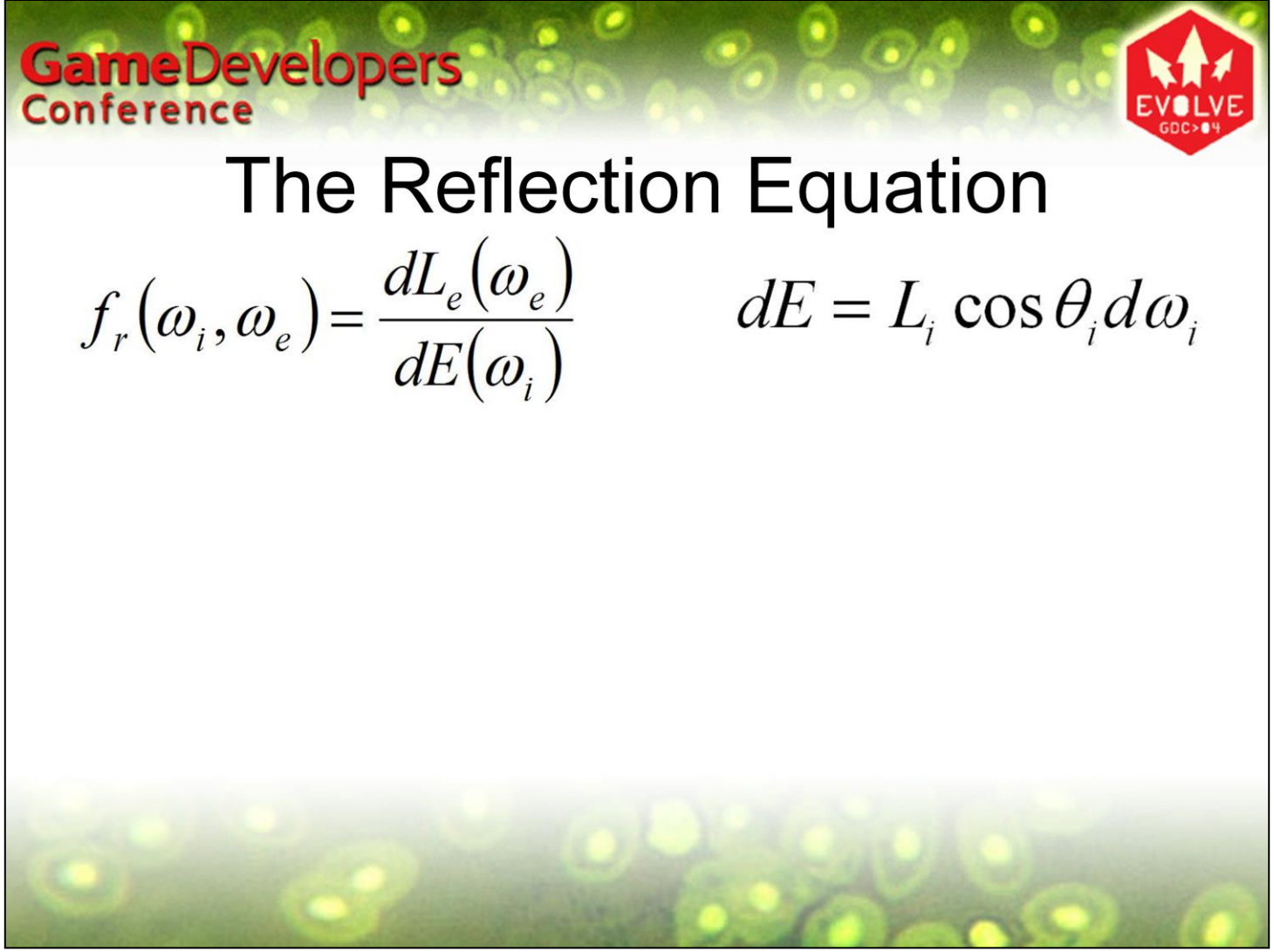
Directions defined relative to the local surface normal and tangent.

The BRDF

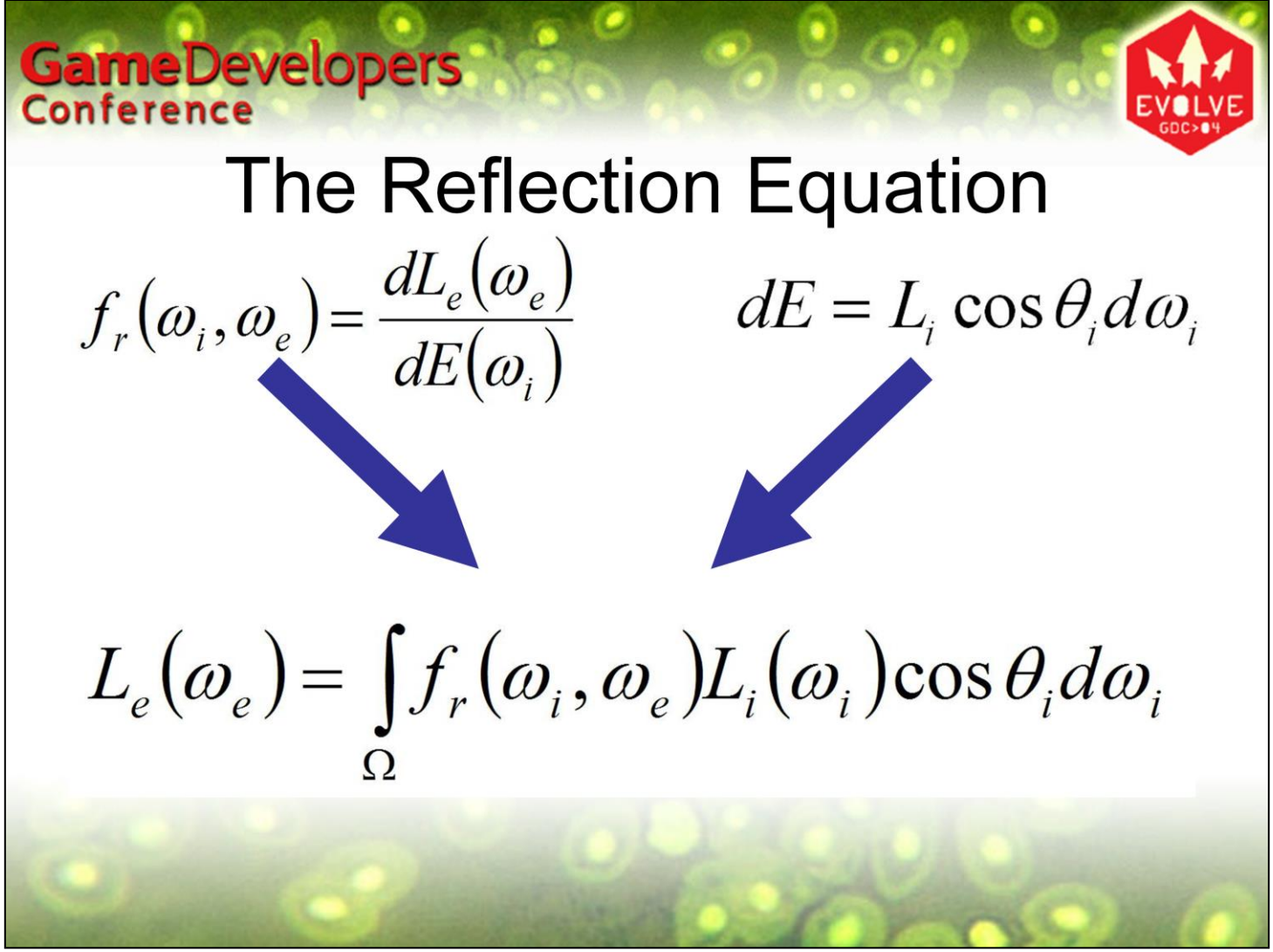
- Isotropic BRDFs are 3D functions



For most surfaces, the relation of the incident and exitant directions to a local surface tangent doesn't matter (these surfaces are isotropic and have no local preferred direction). So instead of the rotations between each of these two directions and a tangent vector, the BRDF depends on the rotation between the two directions, which removes one degree of freedom.



From the definition of a BRDF and the relation between irradiance and radiance, we get:



The reflection equation. This means to get the exitant radiance in a direction *ωe*, we need to integrate the incident radiance, times the BRDF, times the cosine of the angle with the normal, over all incoming directions in the hemisphere around the surface normal.

The BRDF

- The laws of physics impose certain constraints on a BRDF
- To be *physically plausible*, it must be:
  - Reciprocal
  - Energy-Conserving

Reciprocity

- More properly, *Helmholtz Reciprocity*

$$f_r(\omega_i, \omega_e) = f_r(\omega_e, \omega_i)$$

All this means is that the surface must reflect light the same way in both directions – if incoming and outgoing directions are changed, the reflectance must remain the same.

Energy Conservation

- Directional-Hemispherical Reflectance

$$R(\omega_i) = \frac{dB}{dE(\omega_i)} = \int_{\Omega} f_r(\omega_i, \omega_e) \cos\theta_e \, d\omega_e$$

- $R(\omega_i)$ must be less / equal to 1 for all ω_i

The directional-hemispherical reflectance is the ratio of differential exitant radiance to differential irradiance. It tells us how much of the incoming radiant energy is absorbed and how much is reflected.

The reflected light energy cannot be more than the incident light energy, which means that the directional-hemispherical reflectance must be less or equal to 1. 1 means a perfectly reflective surface with no absorption at the given incident angle.

Bihemispherical Reflectance

- Also known as albedo

$$\rho = \frac{B}{E} = \frac{1}{\pi} \int_{\Omega} R(\omega_i) \cos\theta_i d\omega_i =$$

$$\frac{1}{\pi} \int_{\Omega} \int_{\Omega} f_r(\omega_i, \omega_e) \cos\theta_i \cos\theta_e d\omega_i d\omega_e$$

Bihemispherical reflectance (albedo) is the ratio between exitant radiance and irradiance. Like directional-hemispherical reflectance, it's between 0 (full absorption) and 1 (no absorption). It's an overall measure of reflectivity—how much radiant flux (from all angles) is reflected vs. absorbed. It can also be seen as the cosine-weighted average of the directional-hemispherical reflectance. It is computed by integrating the BRDF over all incident and exitant directions. The normalization factor 1/pi shows up a lot, since integrating cosine over the hemisphere yields pi.

Reflectance

- Reflectance values such as R and ρ are restricted to a range of 0 to 1
  - 0 is perfectly absorbent
  - 1 is perfectly restrictive
- This does not apply to the BRDF!
  - For example, in the center of a tight highlight the BRDF value will be quite high

Wavelength Dependence

- Note that the BRDF and reflectance values are wavelength-dependent
- In practice this means that they are all RGB triples

In high-fidelity offline rendering many more than three spectral samples are often used which would require reflectance quantities to be represented as long vectors, but we wont worry about that for this talk.

The Reflection Equation

- Real-time applications use a simplified form of $L_i(\omega_i)$
  - A constant ambient L_{iA} from all directions
  - Except for directional/point lights
    - Each light characterized by $(L_i d\omega_i)$ and ω_i

$$L_e(\omega_e) = \sum_l f_r(\omega_{il}, \omega_e)(L_i d\omega_i)_l \cos\theta_{il} + L_{iA} \int_\Omega f_r(\omega_i, \omega_e) \cos\theta_i d\omega_i$$

In real-time rendering, a simplified model of incident radiance is used (environment maps allow complex incident radiance, but they are a special case). In this model the incident radiance is equal to an ambient constant *LiA* from all directions, except for a small number of directions in which we have directional or point lights. Those lights are characterized by radiance times solid angle and direction. This converts the integral in the reflection equation into a sum over the lights, with a separate ambient term.

The Reflection Equation

$$L_e(\omega_e) = \sum_l f_r(\omega_{il}, \omega_e)(L_i d\omega_i)_l \cos\theta_{il} + L_{iA} \int_\Omega f_r(\omega_i, \omega_e) \cos\theta_i d\omega_i$$

If we look at the ambient term, we see that it is the same as the directional-hemispherical reflectance, except that the incident and exitant directions are switched. Due to reciprocity, this doesn't make a difference, so we can write the reflection equation in a slightly simplified form:

The Reflection Equation

$$L_e(\omega_e) = \sum_l f_r(\omega_{il}, \omega_e)(L_i d\omega_i)_l \cos\theta_{il} + L_{iA} R(\omega_e)$$

Note that here we use the directional-hemispherical reflectance with the exitant direction instead of the incident direction. It is tempting, but somewhat incorrect, to call this the hemispherical-directional reflectance.

Finally, it is common to assume that the directional-hemispherical reflectance is constant. This is generally incorrect, but it does simplify the equation to the form which is usually used in real-time applications:

The Reflection Equation

$$L_e(\omega_e) = \sum_l f_r(\omega_{il}, \omega_e)(L_i d\omega_i)_l \cos\theta_{il} + L_{iA}R$$

This is the commonly used form. For each light, we evaluate the BRDF at the light and view direction, multiply by a light intensity term and the cosine term, sum the results and add an ambient term.

This constant ambient term tends to negatively affect image realism. In the diffuse case this is often improved by modulating it with an ambient occlusion term. Later in this talk we will discuss other ways to improve this term.

GameDevelopers Conference

EVOLVE GDC>04

Scale and the BRDF

- Reflectance Models are intimately tied to scale
- The complexity of many BRDFs originates in statistically modeled subpixel structure

We've seen a little bit of this in the earlier discussion of semi-rough materials.

For example, the BRDF of an object like a tree varies depending on whether we are looking at a single leaf, an entire tree, or a forest. In each case, the structure covered by a single pixel is very different.

Scale and the BRDF

- The same features may be modeled as
  - Geometry
  - Bump maps
  - BRDFs
- Depending on scale
- Smooth transitions between these representations pose interesting issues

Scale and the BRDF

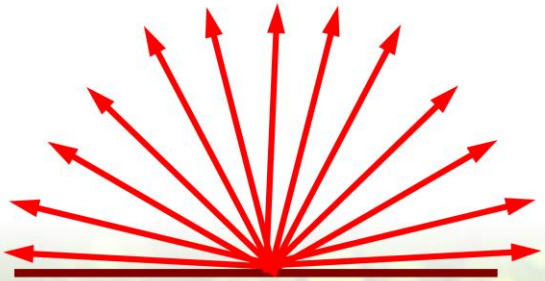
- Filtering
  - In theory, when moving off into the distance, bumps should be filtered into bumps + BRDFs
  - Similar problems apply for other BRDF parameters

Reflection Model Foundations

- Lambert
- Fresnel Equations
- Microfacet theory

Lambert

- Constant BRDF
- Pure Lambert impossible
- Usually used for dielectric subsurface scattering terms



The same radiance reflected in all directions. Remember that the well-known Lambertian cosine factor is actually part of the reflectance equation and not the BRDF. A perfectly Lambertian surface would violate physical laws.

Lambert

- The Lambertian BRDF is a constant value over all incident and exitant directions
- BRDF equal to bihemispherical reflectance (albedo) over π

$$\rho = \frac{1}{\pi}\int_{\Omega}\int_{\Omega} f_{rLambert}\cos\theta_i \cos\theta_e d\omega_i d\omega_e = f_{rLambert}\pi$$

$$f_{rLambert} = \frac{\rho}{\pi}$$

The Lambertian BRDF is a constant, what is it's value? We can calculate the bihemispherical reflectance, or albedo, and see that the BRDF is equal to the bihemispherical reflectance over pi. This is useful, since the bihemispherical reflectance is an intuitive number, between 0 and 1, which tells us how reflective the surface is overall. Note since the bihemispherical radiance is wavelength-dependent, it is an RGB triple, usually thought of as the material's diffuse color.

Light Intensities

- 'Game engine' Lambert $L_e = \rho \, \mathrm{I} \cos\theta_i$
- Radiometric Lambert $L_e = \frac{\rho}{\pi} L_i d\omega_i \cos\theta_i$

$$\mathrm{I} = \frac{L_i d\omega_i}{\pi}$$

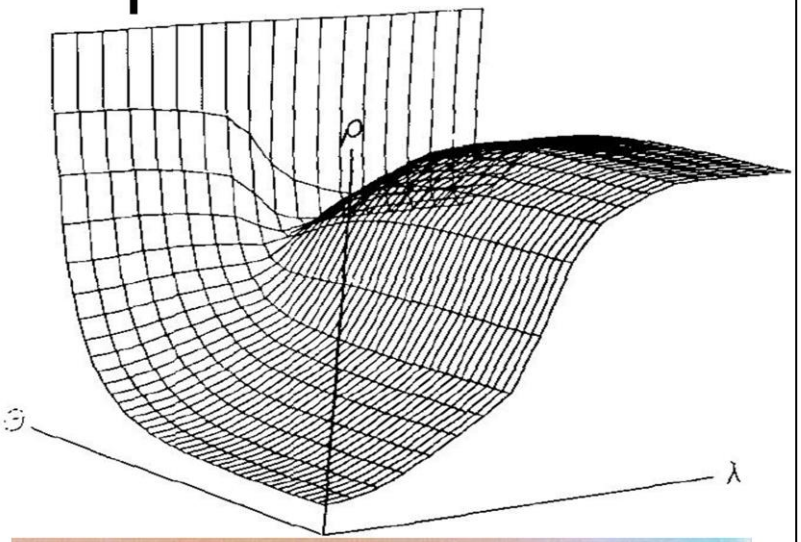
In game engines, the light intensity value is typically multiplied by diffuse color (albedo) and cosine factor to get exitant radiance (result pixel color). Comparing to the radiometric version, we see that game engine light intensity corresponds to radiance times solid angle over pi. When using a BRDF in a game, you need to either change light intensity to be changed to something more meaningful like radiance times solid angle, or multiply the BRDF by pi. This multiplication will tend to cancel out the "one over pi" normalization factors which are common in many BRDFs.

GameDevelopers Conference

EVOLVE GDC>04

Fresnel Equations

- Reflectance depends on
  - Incidence angle (goes to 100%)
  - Refractive index (depends on wavelength)
  - Polarization



IMAGES BY R. COOK AND K. TORRANCE

This is a recap of a previous slide. Note here only that the main effect is to gradually change from the spectral reflectance at normal incidence to 100% reflectance at all wavelengths, as the incidence angle goes from normal to glancing. The shift is not monotonic however, which causes the blue shift seen on the left just before it goes to white.

Fresnel Equations

- Full equations expensive to compute, need (possibly complex) refractive index
- Schlick approximation

$$R_F(\theta_i) = R_F(0) + (1 - R_F(0))(1 - \cos\theta_i)^5$$

- $R_F(0)$ is the directional-hemispherical reflectance when incident direction and surface normal coincide

The Schlick approximation is accurate to within a few percent, is much cheaper to compute, and has a much more intuitive parameter: RF(0), which is the reflectance at normal incidence. Note that the reflectances here are all directional-hemispherical.

RF(0) is commonly thought of as the materials' specular color. It is relatively high for metals, and low for dielectrics. This cosine factor is also clamped to zero.

Fresnel Equations

- $R_F(0)$ is high for metals
  - Steel ~0.55, Silver, Aluminum ~0.95
  - Colored metals
    - Gold: from ~0.6 for blue to ~0.9 for red
    - Copper: from ~0.4 for blue to ~0.85 for red
- $R_F(0)$ lower for dielectrics
  - Water, glass, plastic, etc. ~0.05
  - Diamond ~0.15

Colorless metals have RF(0) which is almost constant over the visible spectrum. Colored metals tend to have higher RF(0) for longer wavelengths. Dielectrics are usually colorless and have low RF(0).

Microfacet Theory

- For semi-rough surfaces
- Models some effects of microgeometry
  - single-bounce reflection
  - Shadowing
  - masking
- Doesn't model
  - interreflections
  - Scattering (usually combined with Lambert)

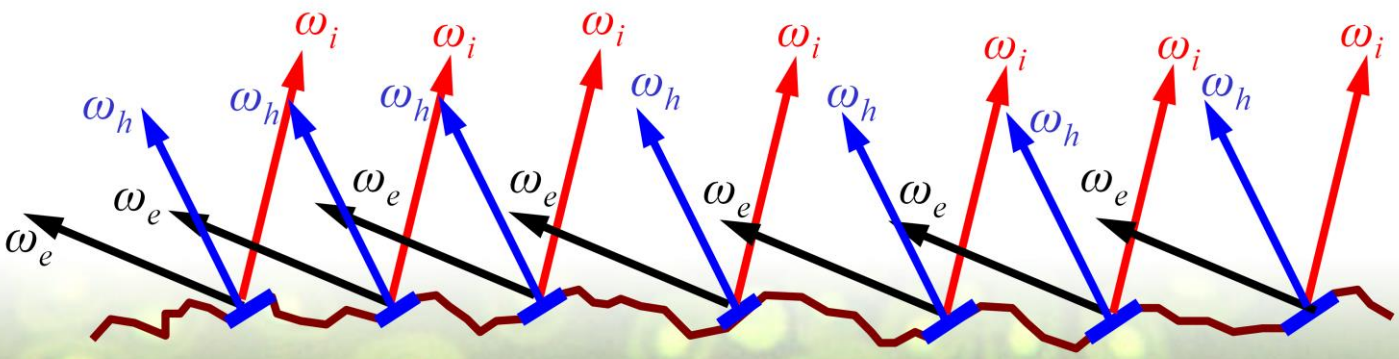
Microfacet Theory

- Surface modeled as flat *microfacets*
  - Each microfacet is a Fresnel mirror
- Normal Distribution Function (NDF) $p(\omega)$
  - $p(\omega)d\omega$ is fraction of facets which have normals in the solid angle $d\omega$ around ω
  - ω is defined in the surfaces' local frame
  - There are various different options for $p(\omega)$

We will look at various options for modeling the NDF later.

Microfacet Theory

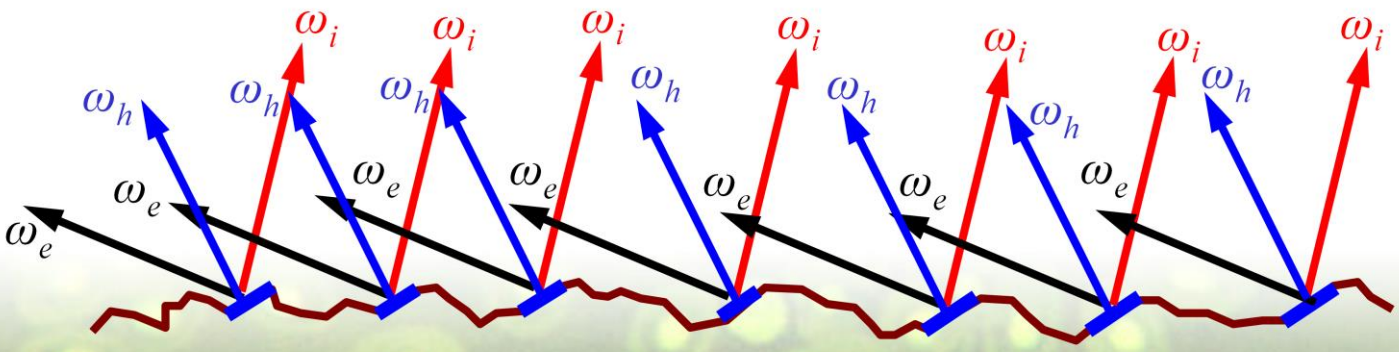
- For given ω_i and ω_e, only facets oriented to reflect ω_i into ω_e are active
- These facets have normals at ω_h



Since each facet is a perfectly smooth mirror, it has to be oriented exactly right to reflect ωi into ωe to participate in the reflectance at all. ωh is the half-angle vector, which is the vector exactly half-way between ωi and ωe.

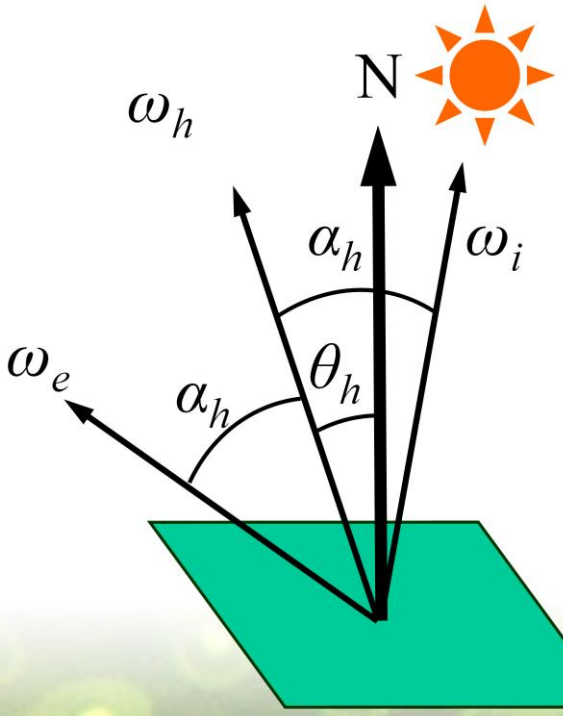
Microfacet Theory

- Fraction of active microfacets is $p(\omega_h)d\omega$
- Active facets have reflectance $R_F(\alpha_h)$
- α_h is the angle between ω_i (or ω_e) and ω_h

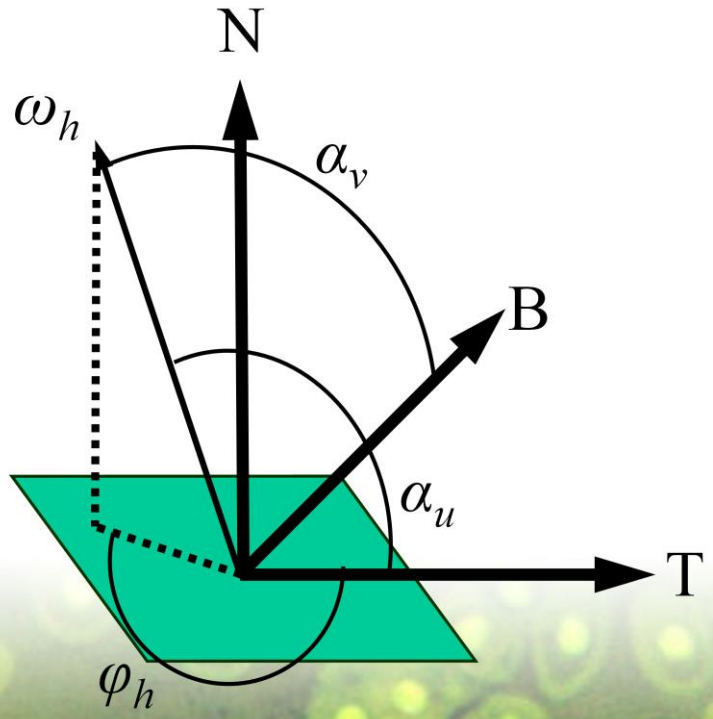


Since each facet is a perfectly smooth mirror, it has to be oriented exactly right to reflect ωi into ωe to participate in the reflectance at all. ωh is the half-angle vector, which is the vector exactly half-way between ωi and ωe.

Half-angle Vector



Half-angle Vector



Microfacet Theory

$$f_r(\omega_i, \omega_e) = \frac{p(\omega_h) G(\omega_i, \omega_e, \omega_h) R_F(\alpha_h)}{4 K_p \cos\theta_i \cos\theta_e}$$

- $G(\omega_i, \omega_e, \omega_h)$ (the *geometry factor*) is the fraction of microfacets which are not masked or shadowed
- K_p is a constant with a value dependent on the microgeometry

From the previously seen relations and equations, we can derive this form for the microfacet BRDF (detailed derivation in the proceedings paper of this talk). We will see a few different options for modeling the geometry factor later in this talk. Note that the geometry factor may contain parts that will cancel out or modify the cosine factors in the denominator.

Microfacet Theory

- $p(\omega)$ is the most important parameter
- Controls roughness, anisotropy, exact shape of highlight
- A large variety of different functions can be used
- It can even be painted into a texture
  - Analogous to painting the highlight

The NDF is the most important parameter. We will learn more about hand-painted NDFs later in the talk.

Reflection Models

- Blinn-Phong
- Cook-Torrance
- Banks
- Ward
- Ashikhmin-Shirley
- Lafortune
- Oren-Nayar

Blinn-Phong

- Commonly used as:

$$f(\mathbf{V},\mathbf{L}) = k_d\, dot(\mathbf{N},\mathbf{L}) + k_s\, dot(\mathbf{N},\mathbf{H})^n$$

- Recast as BRDF with our notation:

$$f_r(\omega_i,\omega_e) = k_d + \frac{k_s}{\cos\theta_i}(\cos\alpha_h)^n$$

The top version is the one commonly used in game engines. It is not in the form of a BRDF (among other things, it includes the clamped cosine factor), so we will rewrite the same function as a BRDF using the previous notation from this talk. Note that we included the one over pi factor for going from “game engine” lighting values to radiometric values.

Blinn-Phong

$$f_r(\omega_i, \omega_e) = k_d + \frac{k_s}{\cos\theta_i}(\cos\alpha_h)^n$$

- Not reciprocal
- No shadowing / masking
- No specular/diffuse tradeoff at glancing angles
- Not normalized!
  - Hard to ensure energy conservation
  - Hard to ensure material is bright enough

The absence of reciprocity, Fresnel, etc. can be minor issues depending on the material we are trying to simulate. The lack of normalization is more of a problem.

BRDF Normalization

- Why is this important?
  - Needed for global illumination algorithms to converge, which is usually not a concern for us
- Important for proper "HDR BRDFs"
- Normalized terms make it possible to directly control a surface's reflectance
  - Range of valid parameters is clear
- With non-normalized terms, risk of
  - Making the surface 'glow'
  - Making the surface too dark

Making the surface too dark is what commonly happens.

BRDF Normalization

- For each un-normalized BRDF term (diffuse, specular)
  - Find C_{R} such that $R(\omega_i) \leq C_{\mathrm{R}}$ for all ω_i
  - Divide term by C_{R}
  - And multiply it by a reflectance parameter like ρ_d, $R_F(0)$, or $R_F(\alpha_h)$

If the term already has a reflectance parameter included, you may need to remove it before computing the upper bound CR.

The upper bound needs to be conservative, to enforce conservation of energy. However it also needs to be as tight as possible, to ensure that the total reflectance of the material is as close to the user-set parameters as possible. There usually isn't an analytical form for an exact bound, so some experimentation and judgment is needed.

Normalized Blinn-Phong

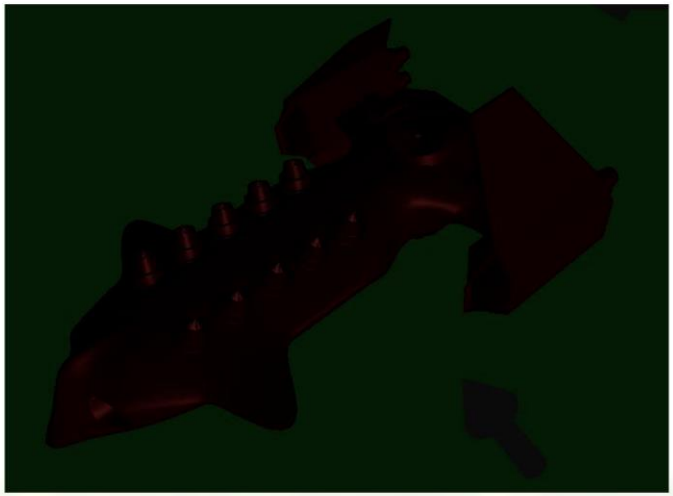
$$f_r(\omega_i, \omega_e) = \frac{\rho_d}{\pi} + \frac{(n+4)R_F(0)}{8\pi \cos\theta_i}(\cos\alpha_h)^n$$

- Now diffuse and specular reflectance can be set directly
- Conservation of energy: $\rho_d + R_F(0) \leq 1$

Actual reflectance will be just a little bit lower than the reflectance parameter due to the conservative normalization factor, but it will be close.

Blinn-Phong

Non-normalized
5% specular color, 32 power



Normalized
5% specular color, 32 power

GameDevelopers Conference

EVOLVE GDC>04

Cook-Torrance

- Microfacet BRDF

$$f_r(\omega_i, \omega_e) = (1-s)\frac{\rho_d}{\pi} + s\frac{p(\omega_h)G(\omega_i, \omega_e, \omega_h)R_F(\alpha_h)}{\pi \cos\theta_i \cos\theta_e}$$

- Reciprocal
- Shadowing / masking
- Specular/diffuse tradeoff
  - Specular reflectance increases at glancing angles, but diffuse reflectance doesn't decrease
  - Not energy-conserving
- Not well-normalized

We can see this is in the same form as the microfacet BRDF we saw earlier. S is a factor between 0 and 1 that controls the relative intensity of the specular and diffuse reflection.

Quite a bit of energy is lost via the geometry factor which is not compensated for – the actual reflectance is quite a bit lower than the parameters would indicate.

Cook-Torrance

- The Cook-Torrance paper recommends using the Beckmann NDF:

$$p(\omega_h) = \frac{1}{m^2 \cos^4 \theta_h} e^{-\left\{\frac{\tan^2 \theta_h}{m^2}\right\}}$$

M is a parameter which controls roughness of the surface.

But it is suggested that a variety of other NDFs can work.

Normal Distribution Functions

- Gaussian
  - (not normalized)

$$p(\omega_h) = e^{-(c_G \theta_h)^2}$$

- Phong

$$p(\omega_h) = \frac{n+1}{2\pi} \cos^n \theta_h$$

- Trowbridge-Reitz
  - (not normalized)

$$p(\omega_h) = \left(\frac{{c_{TR}}^2}{\cos^2 \theta_h \left({c_{TR}}^2 - 1 \right) + 1} \right)^2$$

Phong effectively can be seen as using a normal distribution function. These various functions have subtle differences – worth trying them out, especially the cheap ones. The non-normalized NDFs need to be multiplied by a constant so they are true probability distribution functions (integrate to 1). Note that a normalized NDF doesn't guarantee a normalized BRDF – the BRDF has other factors.

Cook-Torrance

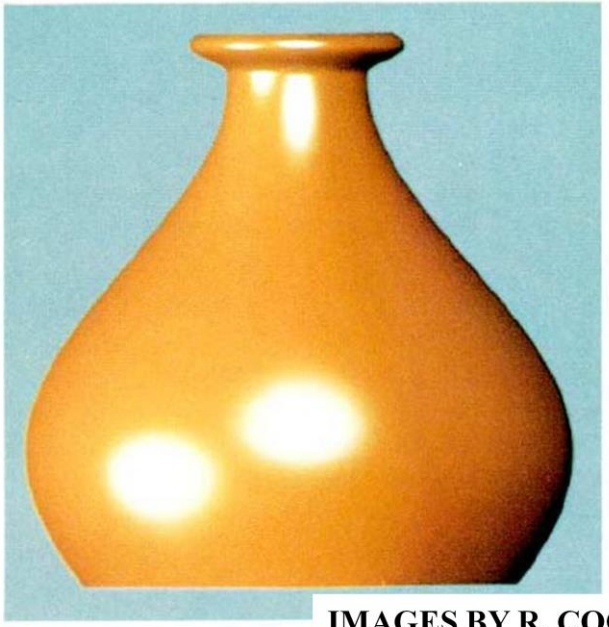
- Geometry Term:

$$G(\omega_i, \omega_e, \omega_h) = \min\left\{1, \frac{2\cos\theta_h \cos\theta_e}{\cos\alpha_h}, \frac{2\cos\theta_h \cos\theta_i}{\cos\alpha_h}\right\}$$

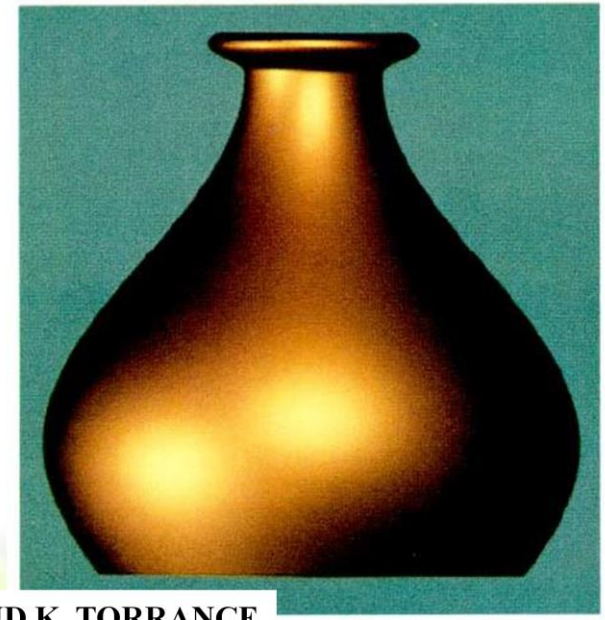
This geometry term is based on a shadowing/masking model of long thin V-shaped grooves. It is not very consistent given that it it is supposed to be used on isotropic surfaces.

Cook-Torrance

Plastic



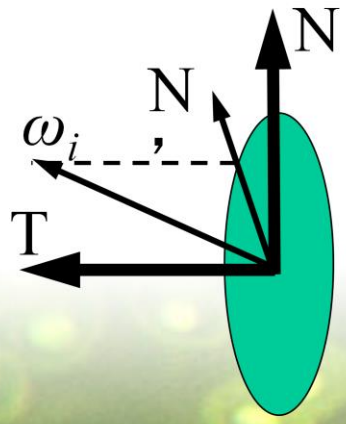
Metal

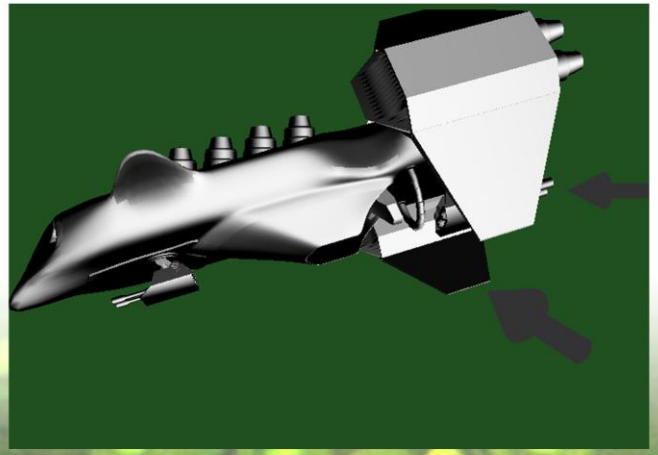


IMAGES BY R. COOK AND K. TORRANCE

Banks

- Anisotropic version of Blinn-Phong
  - Surface model composed of threads
  - Uses projection of light vector onto tangent plane instead of surface normal





Aside from the new normal vector, this is almost exactly the same as Blinn-Phong. One more difference is that when the light is not in the hemisphere about the original surface normal, the lighting is set to 0 (“self-shadowing” term). This is the only effect the surface normal has on this model.

Ward - Isotropic

$$f_r(\omega_i, \omega_e) = \frac{\rho_d}{\pi} + R_F(0) \frac{e^{-\left\{\frac{\tan^2\theta_h}{m^2}\right\}}}{4\pi m^2 \sqrt{\cos\theta_i \cos\theta_e}}$$

- No shadowing / masking
- No specular/diffuse tradeoff at glancing angles
- Reciprocal, conserves energy, well normalized

This is another “pseudo-microfacet model”. It uses a Beckmann NDF, but no explicit Fresnel or masking / shadowing. It is well normalized however.

Ward - Anisotropic

$$f_r(\omega_i, \omega_e) = \frac{\rho_d}{\pi} + R_F(0) \frac{e^{-\left\{\tan^2\theta_h \left(\frac{\cos^2\phi}{m_u^2} + \frac{\sin^2\phi}{mv^2} \right)\right\}}}{4\pi\, m_u m_v \sqrt{\cos\theta_i \cos\theta_e}}$$

The only change is in the NDF, which is now anisotropic.

Ward - Anisotropic

$$f_r(\omega_i,\omega_e)=\frac{\rho_d}{\pi}+R_F(0)\frac{e^{-2\left\{\frac{\left(\frac{\cos\alpha_u}{m_u}\right)^2+\left(\frac{\cos\alpha_v}{m_u}\right)^2}{1+\cos\theta_h}\right\}}}{4\pi\, m_u m_v\sqrt{\cos\theta_i\cos\theta_e}}$$

This approximation is much faster to compute since it uses mostly dot-products.

GameDevelopers Conference

EVOLVE GDC>04

Ward - Anisotropic

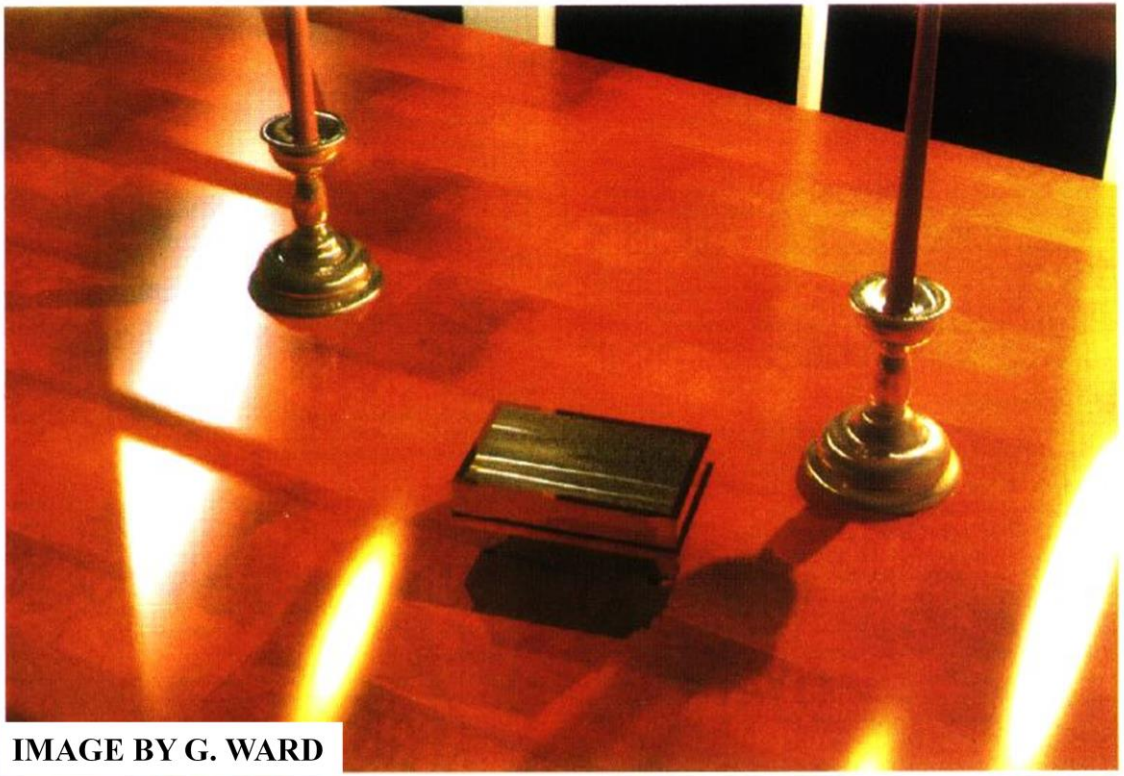


IMAGE BY G. WARD

Although it is missing some of the features of the physically-based BRDFs, the images still look quite nice.

Ashikhmin-Shirley

- Specular and diffuse terms

$$f_r(\omega_i, \omega_e) = f_d(\omega_i, \omega_e) + f_s(\omega_i, \omega_e)$$

- No shadowing / masking
- Specular/diffuse tradeoff at glancing angles
- Reciprocal, conserves energy, well normalized

Another ‘pseudo-microfacet’ model

Ashikhmin-Shirley

- Diffuse term

$$f_d(\omega_i,\omega_e)=\frac{28\rho_d}{23\pi}(1-R_F(0))\left(1-\left(1-\frac{\cos\theta_i}{2}\right)^5\right)\left(1-\left(1-\frac{\cos\theta_e}{2}\right)^5\right)$$

- Trades off reflectance with the specular term at glancing angles
- Without losing reciprocity or energy conservation

Ashikhmin-Shirley

- Specular term

$$f_s(\omega_i, \omega_e) = \frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi} \frac{\cos\theta_h^{\,n_u \cos^2\phi + n_v \cos^2\phi}}{\cos\alpha_h \max(\cos\theta_i, \cos\theta_e)} R_F(\alpha_h)$$

- Note $\max(\cos, \cos)$ term in denominator

Ashikhmin-Shirley

- Specular term (implementation-friendly form)

$$f_s(\omega_i,\omega_e)=\frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi}\frac{\cos\theta_h^{\frac{n_u\cos^2\alpha_u+n_v\cos^2\alpha_u}{1-\cos^2\theta_h}}}{\cos\alpha_h\max(\cos\theta_i,\cos\theta_e)}R_F(\alpha_h)$$

Ashikhmin-Shirley

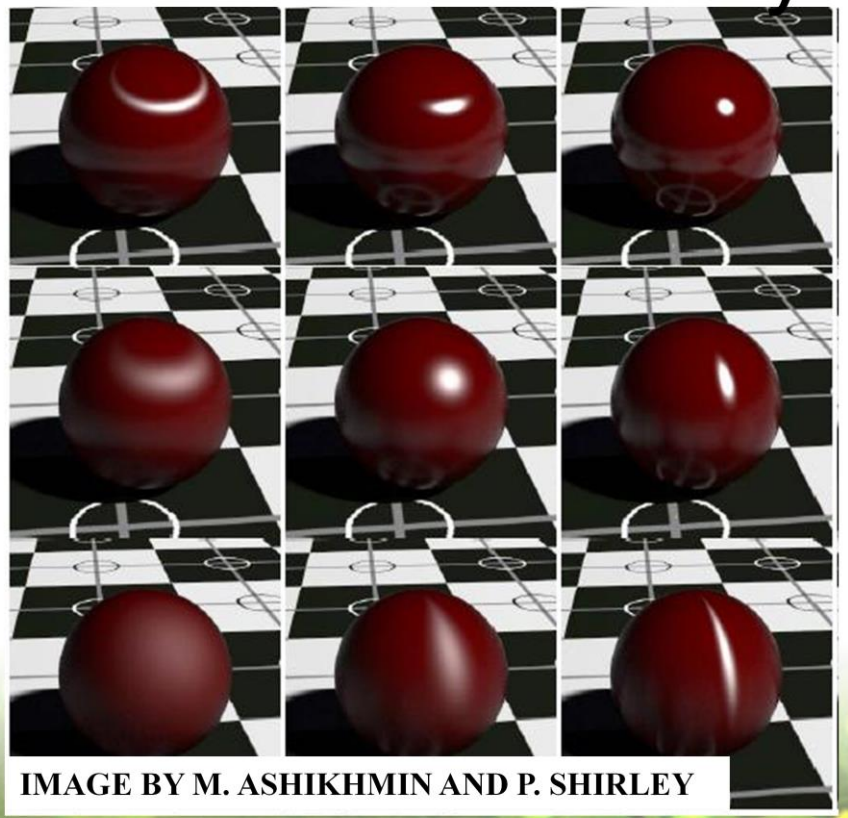


IMAGE BY M. ASHIKHMIN AND P. SHIRLEY

Ashikhmin-Shirley

IMAGE BY M. ASHIKHMIN AND P. SHIRLEY

Lafortune

- Generalization of reciprocal version of original Phong (not Blinn-Phong) specular term:

$$f_s(\omega_i, \omega_e) = \frac{(n+1)R_F(0)}{2\pi} dot(\mathbf{V}, \mathbf{R})^n$$

Lafortune

- In local frame, reflection operator is just multiplying x and y components by -1:

$$f_s(\omega_i, \omega_e) = \frac{(n+1)R_F(0)}{2\pi}\left((-1)\mathbf{V}_x\mathbf{L}_x + (-1)\mathbf{V}_y\mathbf{L}_y + 1\mathbf{V}_z\mathbf{L}_z\right)^n$$

This requires that the light and view direction are in the local frame of the surface (which the BRDF definition assumes anyway, it's just that with many BRDFs you can get away with not actually transforming them into that space).

Lafortune

- Generalize to one spectral (RGB) constant and four scalar constants per term, add several terms (lobes) :

$$f_s(\omega_i,\omega_e)=\sum R\left(C_x\mathbf{V}_x\mathbf{L}_x+C_y\mathbf{V}_y\mathbf{L}_y+C_z\mathbf{V}_z\mathbf{L}_z\right)^n$$

Lafortune

- Lambertian:
  - $R = \rho_d/\pi$, $n = 0$
- Non-Lambertian diffuse:
  - $R = \rho_d$, $C_x = C_y = 0$, $C_z = (n+2)/2\pi$
- Off-specular reflection:
  - $C_z < -C_x = -C_y$
- Retro-reflection:
  - $C_x > 0$, $C_y > 0$, $C_z > 0$
- Anisotropy:
  - $C_x \neq C_y$

Besides the standard cosine lobe, this can handle many other cases.

Lafortune

- Very general, inexpensive to compute, but has very unintuitive parameters
- Best used to fit measured data or some other model with more intuitive parameters

Lafortune

IMAGE BY D. MCALLISTER, A. LASTRA AND W. HEIDRICH

Lafortune

IMAGE BY D. MCALLISTER, A. LASTRA AND W. HEIDRICH

Oren-Nayar

- Lambertian Microfacet model

$$f_r(\omega_i, \omega_e) = \frac{\rho}{\pi}(A + B\cos\varphi)\sin\alpha\tan\beta$$

$$A = 1.0 - 0.5\frac{\sigma^2}{\sigma^2 + 0.33} \qquad B = 0.45\frac{\sigma^2}{\sigma^2 + 0.09}$$

$$\alpha = \max(\theta_i, \theta_e)$$

$$\beta = \min(\theta_i, \theta_e)$$

Sigma is a roughness parameter. 0 is smooth (Lambertian), and increasing the number increases the roughness.

Oren-Nayar

- Normalized, reciprocal, physically based

Hand-painted BRDFs

- N dot L / N dot V
- NDF mapping

Implementation

Dan Baker

Where do we evaluate the BRDF?

- Usually, BRDFs are evaluated at a per pixel or per vertex level.
- Both the BRDF and the local frame it is evaluated in can vary
- We need to sample the BRDF function at a frequency high enough to prevent aliasing

A basic Framework

```
float4 Shader(float4 Normal : TEXCOORD, float3 Tangent :
    TEXCOORD1, uniform LightDir, uniform ViewDir ) : COLOR
{
    float4 Out = 0;
    Out.xyz = saturate(dot(Normal, LightDir))
                * BRDF(ViewDir, LightDir);
    return Out;
}
```

All BRDFS are multiplied by <N,L>

A BRDF only has 2 inputs, View Dir and Light Dir

Adding More lights

```
float4 Shader(float4 Normal : TEXCOORD, float3 Tangent : TEXCOORD1,
    uniform LightDir[MaxLights], uniform ViewDir ) : COLOR
{
    float4 Out = 0;
    for(int i = 0; i < numLights;i++)
    {
        Out.xyz += BRDF(ViewDir, LightDir[i], i);
    }

    Out *= saturate(dot(Normal, LightD
    Out += EnviromentLight();

    return IntoGamma(Out);
}
```

Great place to add low frequency lighting such as environment mapping or PRT.

If we are rasterizing to the Screen we need to convert into gamma space!

Looping in Hardware

- ps_3_0 allows limited looping in the hardware
  - Cannot index samplers
  - Cannot index constants
- Samplers would have to be resolved by the compiler unrolling the loops.
- Can compile multiple version of shader for the different lighting scenarios
- Can also use dynamic conditionals.

An example BRDF

- Fixed function hardware does Blinn-Phong at a per vertex level, so we've grown accustomed to this model.
- In HLSL, the Blinn-Phong model looks like:

```
float3 BlinnPhong(float3 L, float3 V)
 {
    float3 H = normalize(V + L);
    return Ks*pow(dot(H, float3(0,0,1)), Power) + Kd*dot(N,L);
 }
```

How do we evaluate a BRDF?

- Direct Evaluation
  - Make an ALU program in the GPU
- Texture Evaluation
  - Perform a texture lookup
- A combination of the 2.
  - Factor some things into textures
  - Do others with ALU

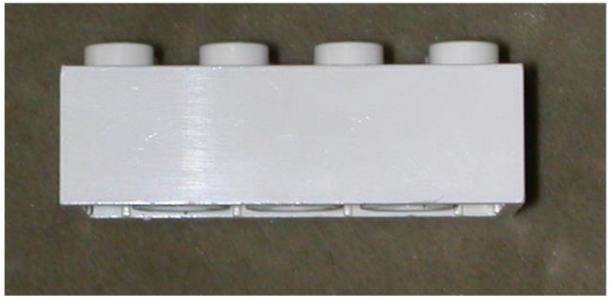
BRDF Costs, no Bumpmapping

Model	Texture costs	ALU costs
Blinn-Phong Direct	0	7
Blinn-Phong factored	1	2
Banks Direct	0	12
Banks Factored	1	5
Ashikhmin/Shirley	0	40
Ashikhmin/Shirley factored	4	10
Lafortune Direct	0	10 + 5 *n
Cook-Torrance	0	35

BRDF Costs with Bumpmapping

Model	Texture costs	ALU costs
Blinn-Phong Direct	1	15
Blinn-Phong factored	2	10
Banks Direct	1	25
Banks Factored	2	18
Ashikhmin/Shirley	2	50 (60)*
Ashikhmin/Shirley factored	6	30
Lafortune Direct	2	30 + 5*Lobes
Cook-Torrance	1	40

How Do We Represent Material Variation?



- Homogonous objects are rare. Some of the varaition is to do meso-geometric effects (like micro scratches), while some is do to smaller scale structure differences. We usually adjust the orientation of the BRDF for mesogeometry, and the parameters for microgeometry.

Getting Variation

- Could sample real materials to generate a BRDF map [McAllister SBRDF].
- This is hard and in many cases infeasible.
- Most of the time, variation will be done by an artist.
- Sometimes we get variation by using an understanding of mesogeometry
- Sometimes it is done by making changes in the assumption of the microgeometry

Adding Variation

```
float3 BlinnPhong(float3 L, float3 V, float2 texCrd, sampler
    GlossMap, sampler ColorMap)
{
    float4 Gloss = tex2D(GlossMap, texCrd);
    float  Power = Gloss.w;
    float3  Kd = tex2D(ColorMap, texCrd);
    float3 H = normalize(V + L);

    return Gloss*Ks*pow(dot(H, float3(0,0,1)), Power)
    + Kd*dot(N,L);
}
```

This adjusts the reflection color of the specular component, and changes its power

Gives our object a diffuse color at each point. The Traditional way we light objects.

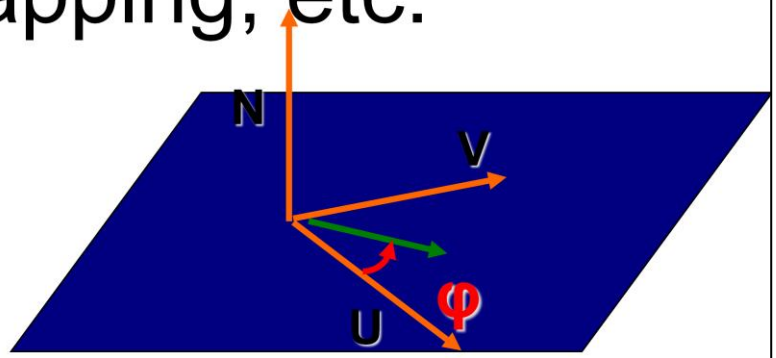
Gamma space

- BRDFs Operate in Linear space, not gamma space
- Most albedo textures are authored in gamma space, due to the fact they are made to look right on a monitor.
- This is ok, but need to convert them into linear space in the BRDF (and convert them back into gamma space).
- Can use SRGB to convert gamma albedo textures into linear space. This gives us more precision where we want it and acts as a kind of compression scheme.

Bump Mapping

Twist mapping, etc.

- Could also store the tangent (twist)
- Can store an Angle
- Sometimes, convenient just to store the first two components of the Tangent, and assume z is 0



Implementing Tangents and Normals

- Must rotate all vector data into per-pixel local fame.
- This frame isn't the (per-vertex) tangent space, but rather a per-pixel space above tangent space.
- Can implement all BRDFs such that Normal can be assumed to be (0,0,1), and the tangents (1,0,0) and (0,1,0) respectively.
- The matrix that takes us from this space to tangent space is:

$$\begin{pmatrix} \text{Tangent} \\ \text{Tangent X Normal} \\ \text{Normal} \end{pmatrix} \qquad \begin{pmatrix} \text{Tangent} \\ \text{Tangent X Normal} \\ \text{Normal} \end{pmatrix}^{T}$$

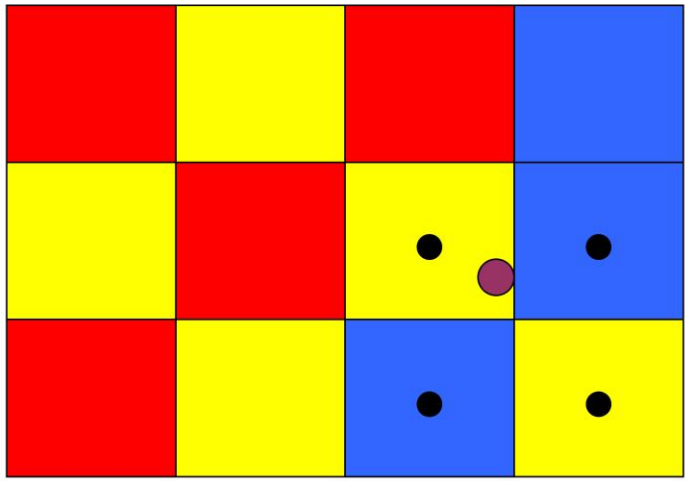
Adding Tangent Support

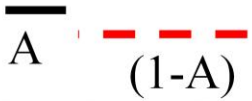
```
float4 Shader(float2 TexCrd: TEXCOORD0,float3 LightDir : TEXCOORD2, float3
    ViewDir : TEXCOORD3, sampler NormalMap, sampler TwistMap ) : COLOR
{
    float4 Out = 0;
    float3 Normal  = tex2D(NormalMap, TexCrd);
    float3 Tangent = tex2D(TwistMap, TexCrd);
    float3x3 reverse = float3x3( Tangent,
                                 cross(Normal,Tangent),
                                 Normal);
    reverse = tranpose(reverse);

    Out = saturate(dot(Normal, LightDir)) / PI
                * BRDF(mul(ViewDir,reverse), mul(-
    return
}
```

LightDir and ViewDir must be in tangent space.

This will cause filtering problems

Texture Filtering Review





Sample Color =

A*B * +

A*(1-B) * +

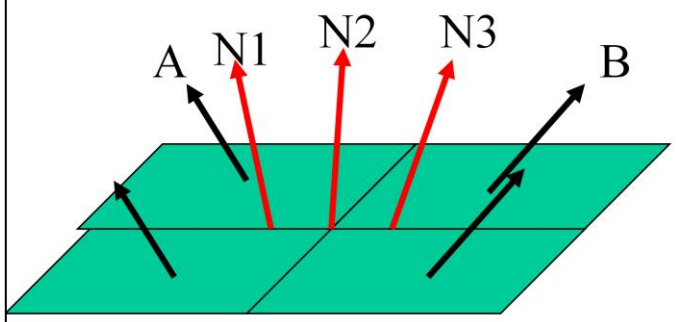
(1-A)*B * +

(1-A)*(1-B)* +

Aliasing

- BRDF aliasing is one of the most significant issues in realtime graphics.
- Shows up as ‘sparkles’ when we rotate a bump mapped object in the world
- Very difficult problem to solve – and little research on this subject.

Why we get the sparklies



A pixel could go from black, To bright green, and back to black as it's exact center moves across the texture (which occurs as objects move in screen space). This easily happens with high specular powers

But, Filtering works ok for diffuse

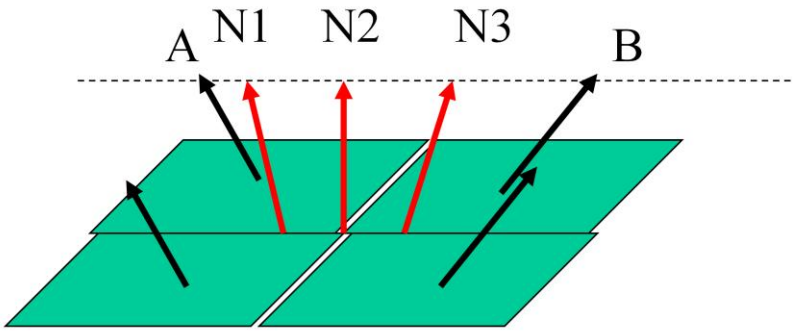
- Observe that this is linear (except for the clamp to 0), where Wx, Wy represent distance from the texel centers.

$$Wx * Wy \langle N_{11}, L \rangle + (1-Wx) * Wy \langle N_{12}, L \rangle + Wx * (1-Wy) \langle N_{21}, L \rangle + (1 - Wx) * (1 - Wy) * \langle N_{22}, L \rangle$$

$$=$$

$$\langle Wx * Wy * N_{11} + (1-Wx) * Wy N_{12} + Wx * (1-Wy) * N_{21} + (1-Wx) * (1 - Wy) * N_{22}, L \rangle$$

Diffuse Filtering

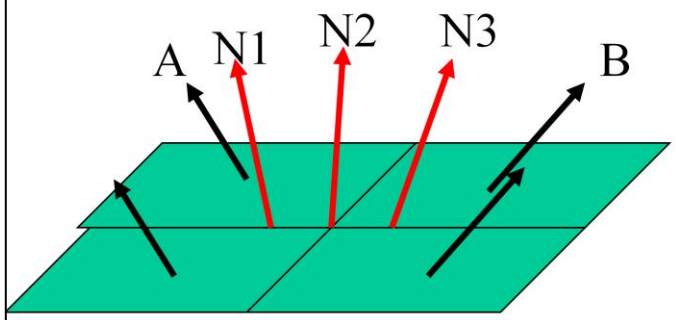


By allowing the normal to remain nonnormalized, we prevent intermediate pixels in the diffuse case from exceeding the brightness of the neighboring texels.

How to decrease Aliasing

- Smoothly moving through a BRDF's parameters doesn't guarantee there won't be significant aliasing. BRDFs are often very non linear. As we've seen before, the texture filtering will not help much.
- A better approach would be to evaluate the BRDF at the resolution of the texture and filter the outputed samples.
- In DX9, shader model 2_x and 3_0 give us all the information to do this.

Manual Filtering



If we evaluated the BRDF for each texel center, then we remove this kind aliasing. Notice all the samples in between the texel center are all the same.

Manual Filtering

```
float4 main(float2 tex : TexCoord, sampler NormalMap, uniform float2 texSize,
            float3 View : TEXCOORD1, uniform float3 Light)
{
        x = 1-frac(tex.x*dim.x - halftexsize.x*.5f);
        y = 1-frac(tex.y*dim.y - halftexsize.y*.5f);

        float3 norm1 = tex2D(NormalMap, texCoord);
        float3 norm2 = tex2D(NormalMap,              texCoord + float2(texsize.x, 0));
        float3 norm3 = tex2D(NormalMap, texCoord + float2(0, texsize.y));
        float3 norm4 = tex2D(NormalMap, texCoord + texsize);


        float4 out = 0;
        out += x*y*BlinnPhong(norm1, View, Light);
        out += (1-x)*y*BlinnPhong(norm2, view, Light);
        out += x*(1-y)*BlinnPhong(norm3, view, Light);
        out += (1-x)*(1-y) BlinnPhong(norm4, view, Light);

        return out;

}
```

We perform 4 Manual Operations and use point filtering.

Filter using the blend weights

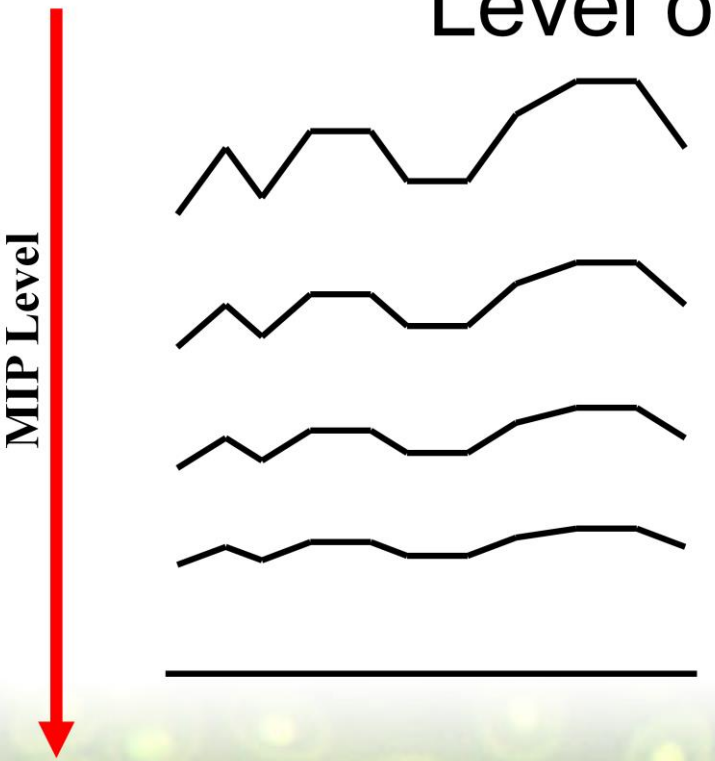
A robust approach

- Can do ‘texture rendering’, a more robust approach to aliasing
- Rather then render the scene in eye space, render the a texture from the point of view of the texture itself.
- Compute the lighting for each texel – only works if each texel lives in only one places in the world.
- Would work for a uniquely textured object.

Cheaper filtering

- Doing Explicit filtering is expensive – so often we must make do without. Take care to make sure the parameters (including) don't vary so quickly as to cause serious aliasing.
- MIP levels are trickier – a linear filter of parameters won't work very well. We want a BRDF parameterization which best fits the previous MIP's BRDF.
- Common fix is simply not to allow high frequency normal maps.
- Additionally, reduce the variance of the parameters on the lower MIP levels. For instance, shift Normals to point straight up.

Level of Detail



By lowering the amplitude Of the displacement, we are effectively decreasing to a less complex model as the model moves further away. In this case, we do this to prevent alias, not for a performance boost.

Texture Evaluation

- Will it be faster to store values in textures and look them up?
- Most pixel shader are ALU limited, not texture limited.
- However, each time we make a new texture it increases the chances of stalls, puts more into the texture caches, etc.
- Our texture caches are very small – it would be very easy to blow them

Function level Factorizing

- Should look for candidates for factorization
- Preferably, functions with a texture friendly domain (i.e. 0 to 1), and a friendly range. Obviously, can do some transforms to the right domain, but each transform adds ALU costs which we want to avoid.
- Small Ranges. Don't want to pollute the cache.

Function Level Factorizing

- Precision Friendly. Precision of domain and range are important.
- Will be relatively local from pixel to pixel. Locality of references of a texture is very important for performances. With small caches, misses are easy and could cause performance to crawl.
- Moral : Don't go overboard – In a heavy ALU shader it is wise to factorize into textures (they will be almost free), but if a large number of textures are being used, it will cause everything to run slowly.

Artistic Factorization

- For given half vector, what's the intensity?
- Could be stored in cube map, but half the map would be unused, parabolic parameterization or sphere map works better.
- Can also make N*V maps, which can be used to paint Fresnel Effects

Examples of texture evaluation

```
float4 AnisoPS( VS_OUTPUT Input ) : COLOR
{
   float3 Normal = (tex2D(NormalMap, Input.TexCoord) - .5f)*2 ;

  float2 tex1 = float2(Input.Half.x*.5 + .5, dot(Normal, Input.Half));
  float2 tex2 = float2(Input.Half.y*.5 + .5, dot(Normal, Input.Half));

   float3 Color = tex2D(UPowerMap,
     tex1)*tex2D(VPowerMap,tex2)*tex1D(FresnelMap,dot(Input.Half,Input.LightVec)
     )  + Rd*tex1D(Diffuse,dot(Input.Half, Input.LightVec));

     Color *= saturate(dot(Normal, Input.LightVec));

    return float4(Color,0);
}
```

A version of the Ashikhmin-Shirley BRDF that has been written to use texture lookups. This version can compile to ps_1_4.

Environment map prefiltering

- Except for Phong, simple environment map lookups don't work.
- For Radially Symmetric BRDFs, we can prefilter, or blur the environment map
- MIP bias is an effective technique – as specular power decreases, blur the enviromentmap.
- Might use a Isotropic BRDF representation for Environment (no high frequancy lights in environment), and use Anisotropic for direct lights.

Example of MIP Bias

```
float3 Enviroment(uniform sampler Envmap, float3
    TexCoord, float Power)
{
    float MipScale = sqrt((MAXPOWER – power)/
                                    MAXPOWER)*MIPBIAS - 1;
    return texCUBEbias(Envmap, float4(TexCoord,
    MipScale);
}
```

The w component controls the MIP bias. Increasing this value will implicitly lower the frequency of lighting

<Show Demo>

Lafortune

- From the implementation standpoint, Lafortune is one of the easiest.
- But function operates in tangent space. It assumes the normal is (0,0,1) (actually, its symmetric so it could be (0,0,-1) as well)
- So ViewDir and LightDir must be transformed into tangent space. This might be expensive (~20 instructions)
- But Otherwise very cheap (~6 cylces)

```
float BRDF(ViewDir, LightDir)
{
    return dot(LightDir * float3(-Cx,-Cy, Cz) , ViewDir), Power) * (2 +
    N)/PI;
}
```

Lafortune Parameters

- Parameters are difficult to deal with
  Best used to fit another BRDF – either sampled or another analytic model.
- Can model a few things that others can't. E.G. Retroreflection can be modeled by inverting Nz.

Production Issues

Naty Hoffman

Production Issues

- Artist-friendly reflectance parameters
- Authoring reflectance shaders
- Managing reflectance shaders

Reflectance Parameters

- Important to expose shader parameters that the artists can understand and tweak easily
- BRDFs such as Ashikhmin-Shirley have an advantage in this area over those like Lafortune
- Though it is sometimes possible to expose intuitive parameters on top of an unintuitive BRDF

Artist-Friendly Parameters

- If any parameters are in textures, they need to be easy to paint and visualize
- This is at least partially a tools issue

Authoring Reflectance Shaders

- Black Box
  - Shaders authored by programmers
  - Artist selects one from a toolbox and tweaks ‘knobs’
- Configurable
  - Artist wires together a set of ‘building blocks’
- Write from scratch
  - Artist / programmer writes shaders

Managing Reflectance Shaders

- Material shaders occupy a twilight zone between code and data
- Can be handled as either to some extent
- Advantages to both, needs to be decided on a per-project basis

Conclusions

- A good understanding of reflectance physics will help choose reflectance models to make your game look better
- More ‘correct’ is not necessarily better
  - But in some cases it can make a big difference
- Implementing complex BRDF models doesn’t have to be expensive or difficult

Acknowledgements

- Thanks to Michael Ashikhmin, Henrik Wann Jensen, David McAllister, Eric Lafortune, Michael Oren, Kenneth Torrance, Gregory Ward, and Stephen Westin for permission to use images for this talk
- Thanks to Jason Sandlin for help with the demos