Materials

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

Cook, Robert L., and Kenneth E. Torrance, *ACM Transactions on Graphics (TOG)* 1982.

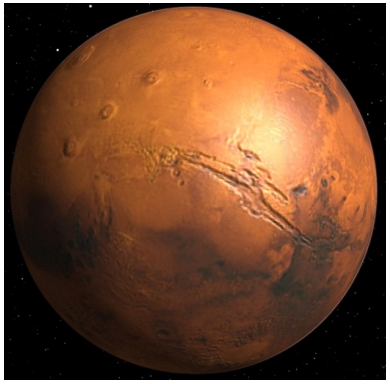
Announcements

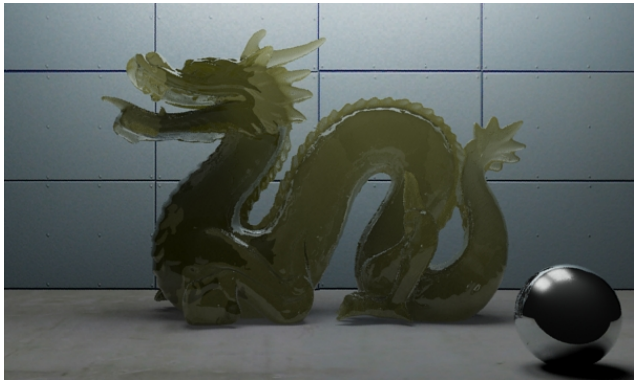
- Project proposal due this Thursday **at 1:30 PM**
  - Check website for formats and a guideline
- Use the VM !
  - HW3 is due this Thursday too!
- **Minjae Lee** has new OH on Wednesdays: **Gates 210: Wednesdays: 10:00 - 12:00 PM**
- **Top 3 Extra Credit** Solutions will be showcased
  - Let us know if you don't want your EC to be shown in class.

Intersection of Science and Art

Interplay between:

- **Efficiency** of rendering
- Desired **artistic** outcome
- **Physicality** of material



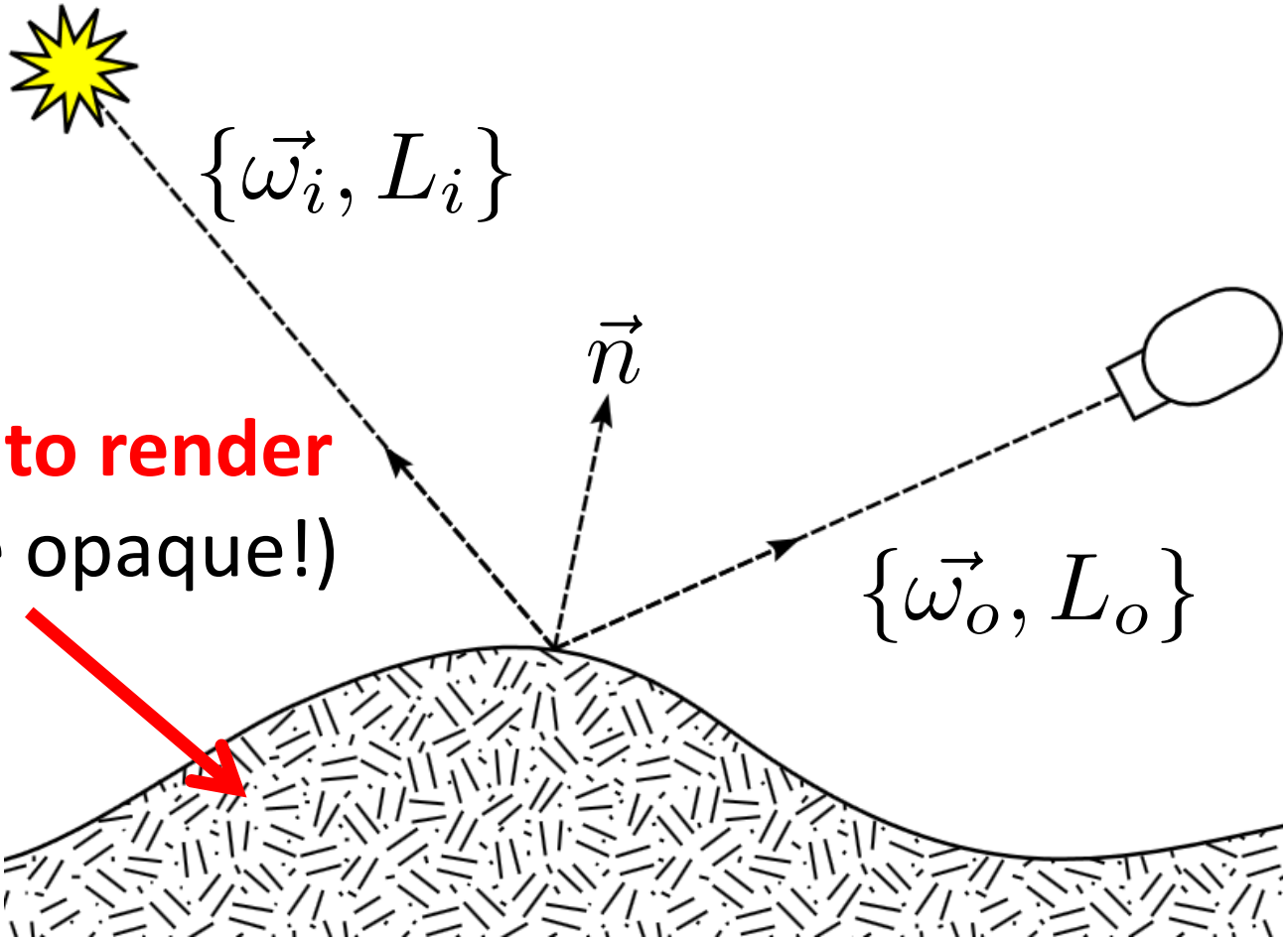


http://zanir.wz.cz/programming/dx81/img/012_Cube_Mapping.jpg
http://www.mrbluesummers.com/wp-content/uploads/2010/07/mental-ray-parti-volume-scatter-color.jpg
http://www.lepp.cornell.edu/~seb/celestia/images/mars_spec_ogl2.jpg

Problem at Hand

How do physical materials interact with light?

Basic Shading Model

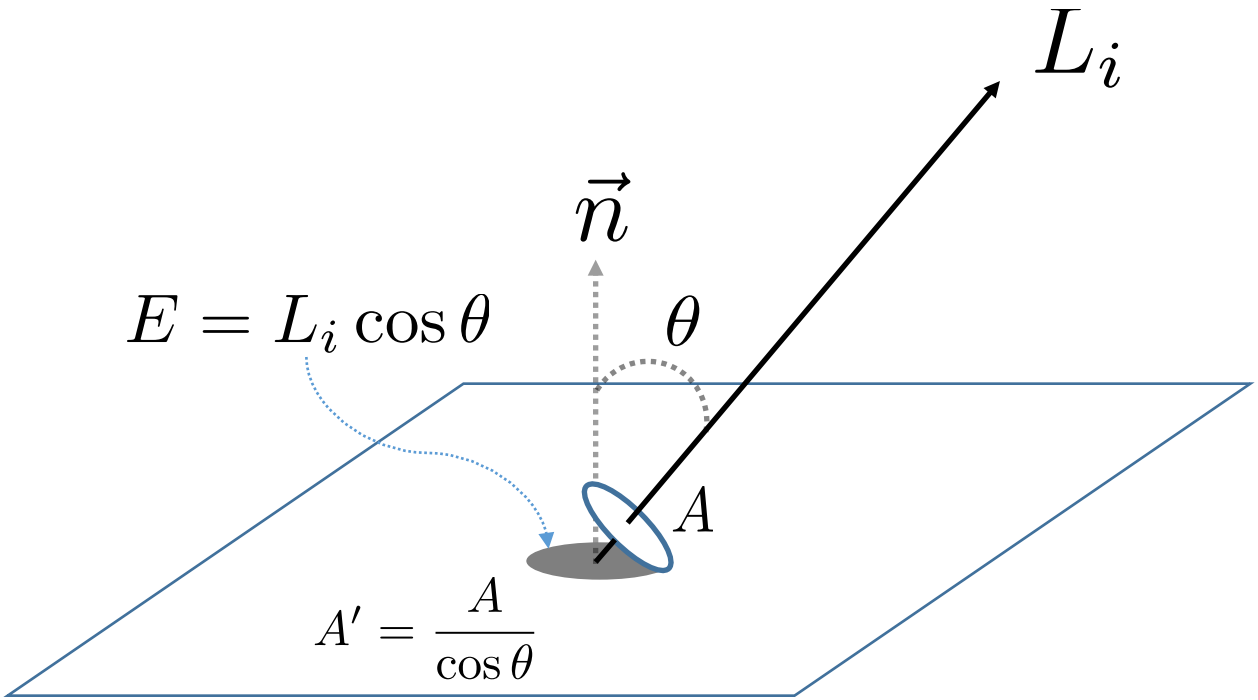


Surface to render

(assume opaque!)

http://upload.wikimedia.org/wikipedia/commons/e/ed/BRDF_Diagram.svg

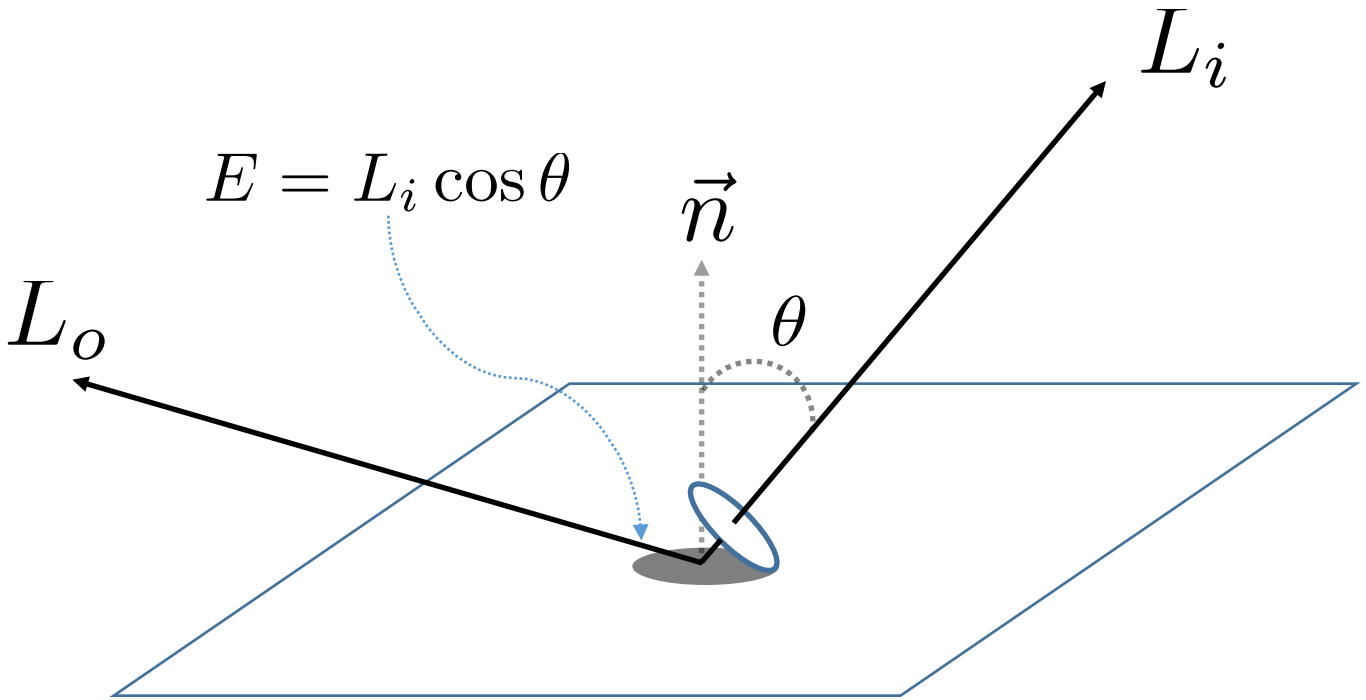
Basic Radiometry



E = Irridiance: Light energy per unit area

L = Radiance: Light energy at a direction (a ray)

Basic Radiometry



$$L_o \propto E$$

$$L_o = f(\ldots) L_i \cos\theta$$

BRDF

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right)$$

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = \frac{L_o}{E} = \frac{L_o}{L_i \cos\theta}$$

Fraction of incoming irridiance transported to outgoing radiance

Bidirectional Reflectance Distribution Function

BRDF Properties

- Positive

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) \geq 0$$

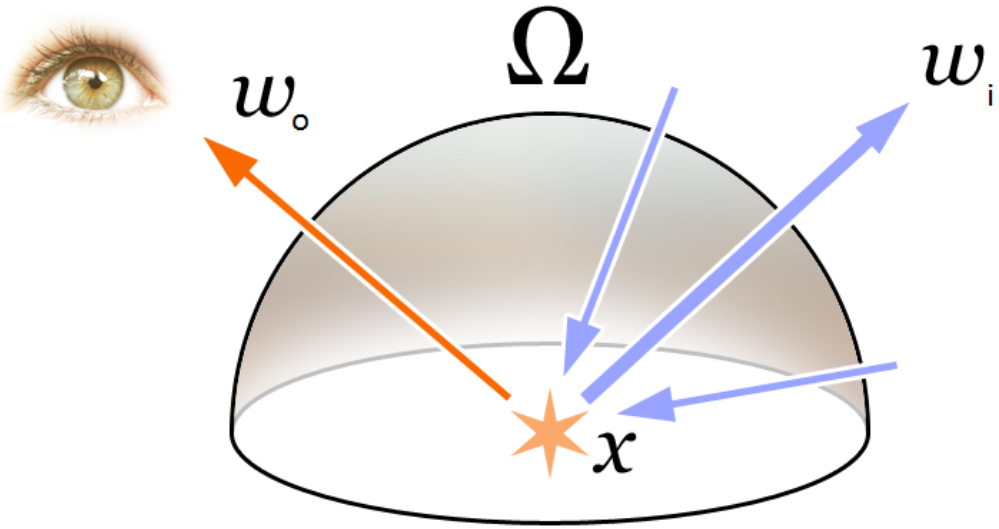
- Obeys Helmholtz Reciprocity

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = f\left(\vec{\omega_o}, \vec{\omega_i}\right)$$

- Conserves Energy

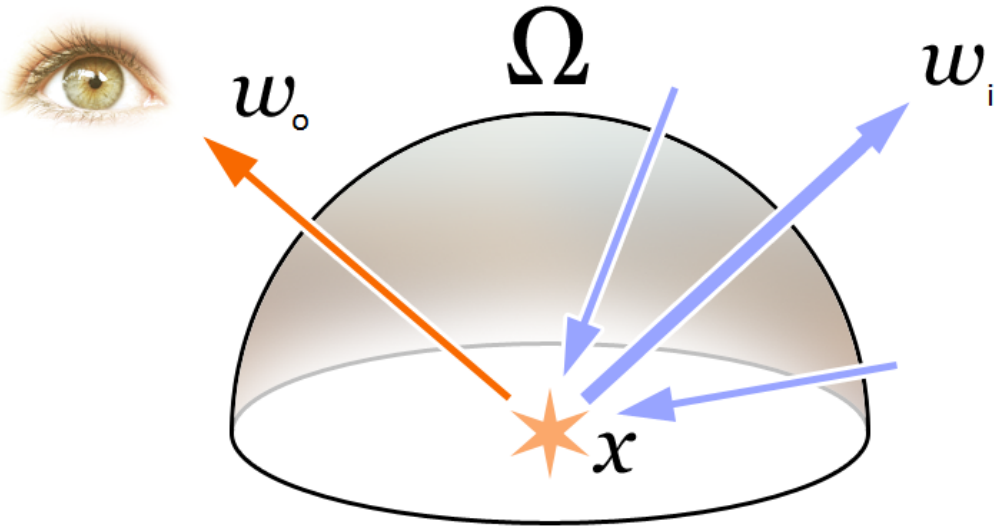
$$\int_{\Omega} f\left(\vec{\omega_i}, \vec{\omega_o}\right)\left(\vec{n} \cdot \vec{w_i}\right) d\omega_i \leq 1$$

Rendering Equation



$$L_o(\vec{\omega}_o) = L_e(\vec{\omega}_o) + \int_{\Omega} f\left(\vec{\omega}_i, \vec{\omega}_o\right) L_i(\vec{\omega}_i) \underbrace{(\vec{n} \cdot \vec{w}_i)}_{\cos\theta} d\vec{\omega}_i$$

Rendering Equation



$$L_o(\vec{\omega}_o) = L_e(\vec{\omega}_o) + \int_{\Omega} f\,(\vec{\omega}_i, \vec{\omega}_o)\, L_i(\vec{\omega}_i)\, \underbrace{(\vec{n} \cdot \vec{w}_i)}_{\cos\theta}\, d\vec{\omega}_i$$

Sum over all incoming light

BRDF: History

- First to introduce Rendering Equation in Computer Graphics
  - James T. Kajiya, *SIGGRAPH*'86
  - David S. Immel, Michael F. Cohen, Donald P. Greenberg, *SIGGRAPH*'86

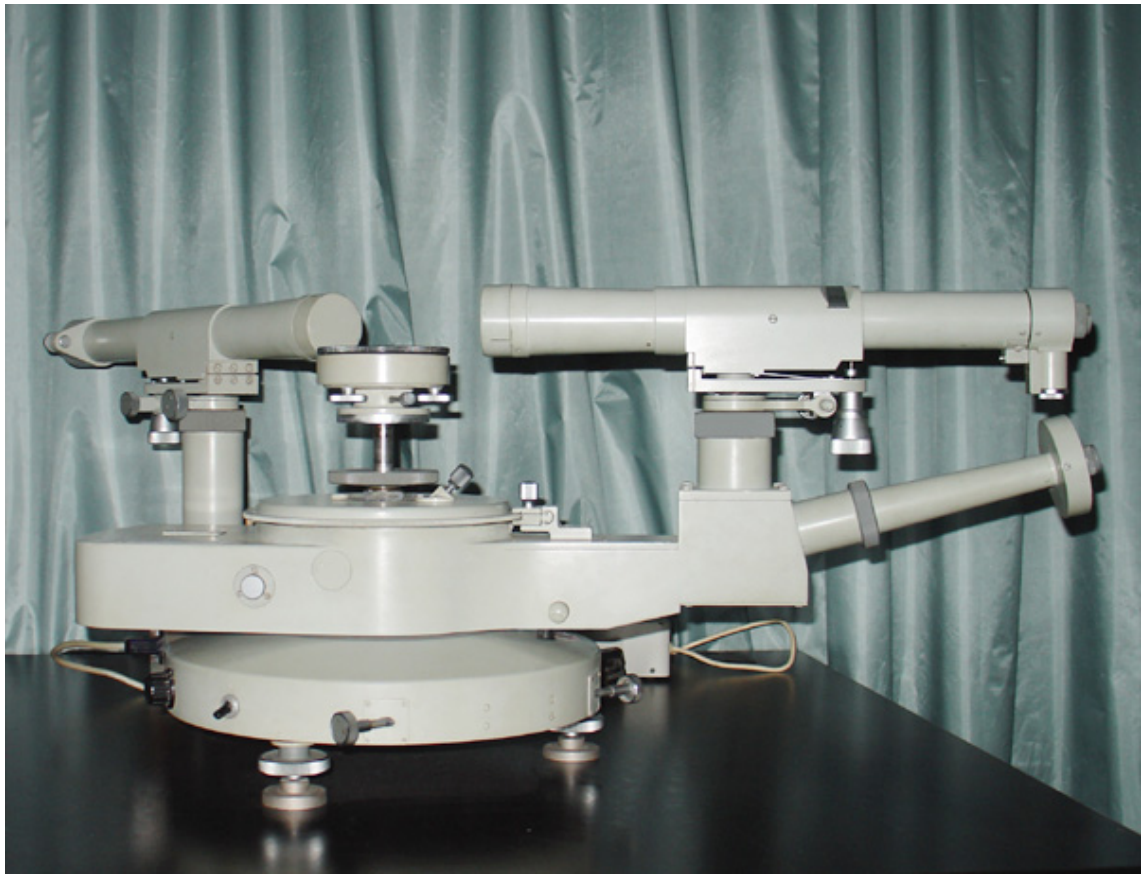
THE RENDERING EQUATION

James T. Kajiya
California Institute of Technology
Pasadena, Ca. 91125

A RADIOSITY METHOD FOR NON-DIFFUSE ENVIRONMENTS

David S. Immel, Michael F. Cohen,
Donald P. Greenberg
Cornell University
Ithaca, N.Y. 14853

Measuring BRDFs



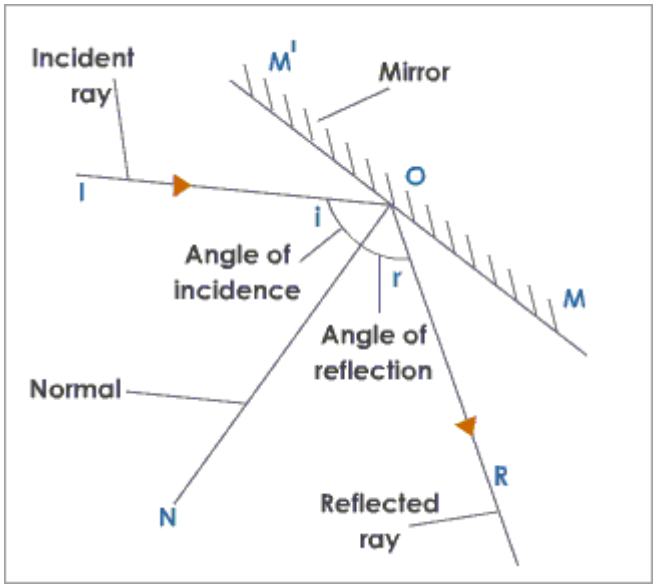
Goniophotometer

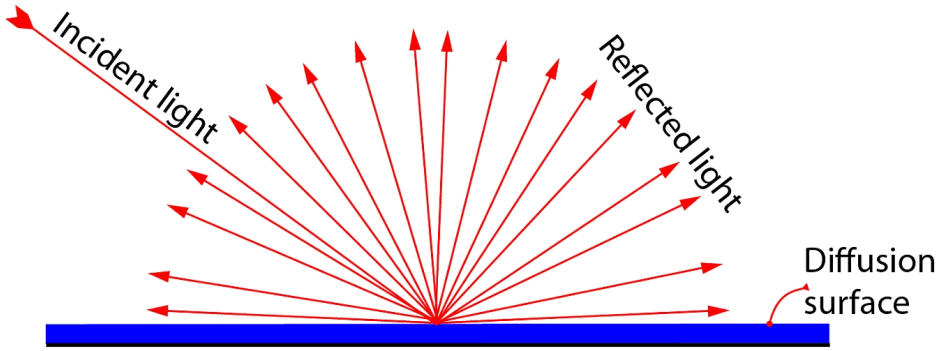
http://www.lcoptical.com/images/goniophotometer.jpg

For Fast Rendering

We sometimes have closed form solutions for L

Two *L*'s We Already Covered





Reflection **Diffuse**

Diffuse BRDF

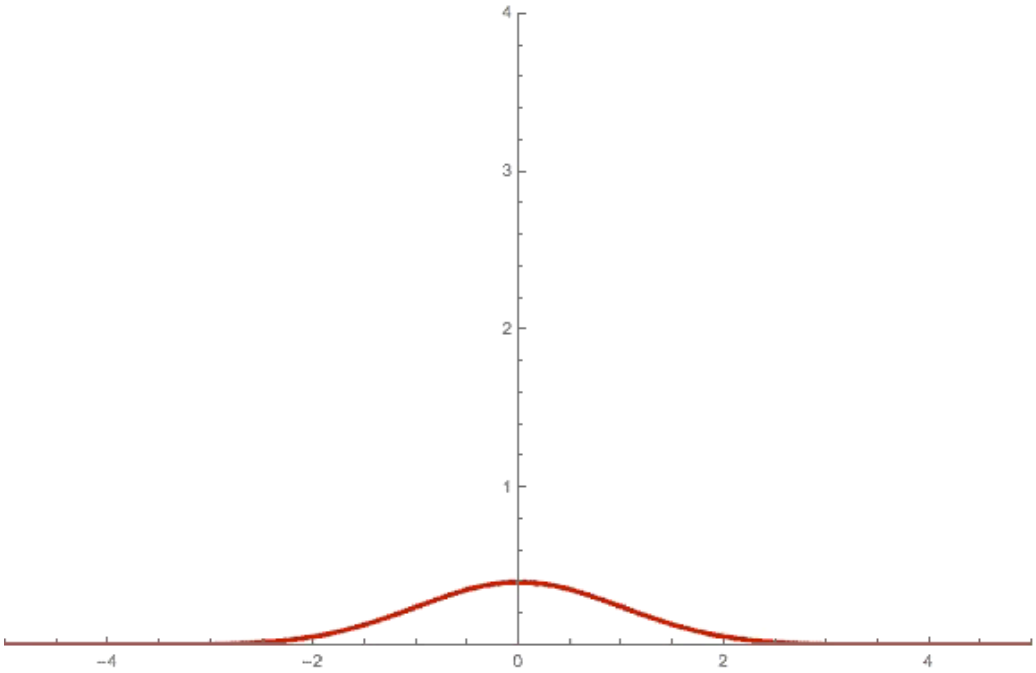
$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = C$$

$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega_i}, \vec{\omega_o}\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

For a single point light source

$$L_i(\vec{\omega_i}) = L\delta(\vec{\omega_i} - \vec{d})$$

Dirac-Delta Function



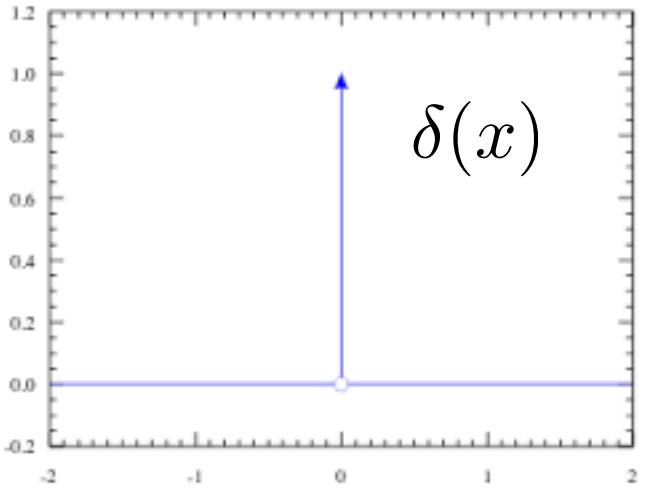
$$\delta(x) = \lim_{\sigma \to 0} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

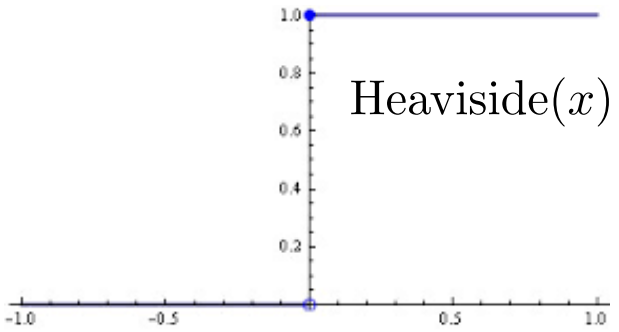
Dirac-Delta Function: Properties

$$\int_{-\infty}^{\infty} \delta(x)dx = 1$$

$$\int_{-\infty}^{\infty} f(x)\delta(x-t)dx = f(t)$$

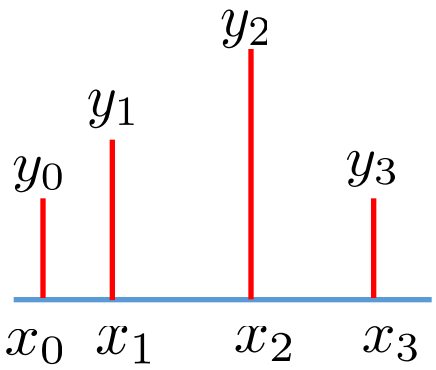
$$\frac{d}{dx}\text{Heaviside}(x) = \delta(x)$$



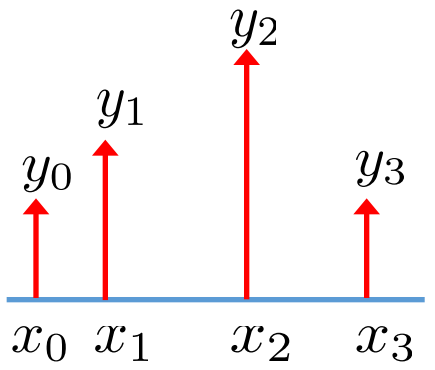


Dirac-Delta: Discrete Functions

Discrete Function



Continuous Function



$$\{(x_0, y_0), (x_1, y_1) \ldots\}$$

$$f(x) = \sum_i y_i \delta(x - x_i)$$

Can be used to describe a discrete function in continuous form

Diffuse BRDF

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = C$$

$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega_i}, \vec{\omega_o}\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

For a single point light source $L_i(\vec{\omega_i}) = L\delta(\vec{\omega}_i - \vec{d})$

$$L_o(\vec{\omega}_o) = C \int_\Omega L_i \delta\left(\vec{w}_i - \vec{d}\right)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

$$= CL\left(\vec{n} \cdot \vec{d}\right)$$

Diffuse BRDF

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = C$$

$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega_i}, \vec{\omega_o}\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

For a single point l

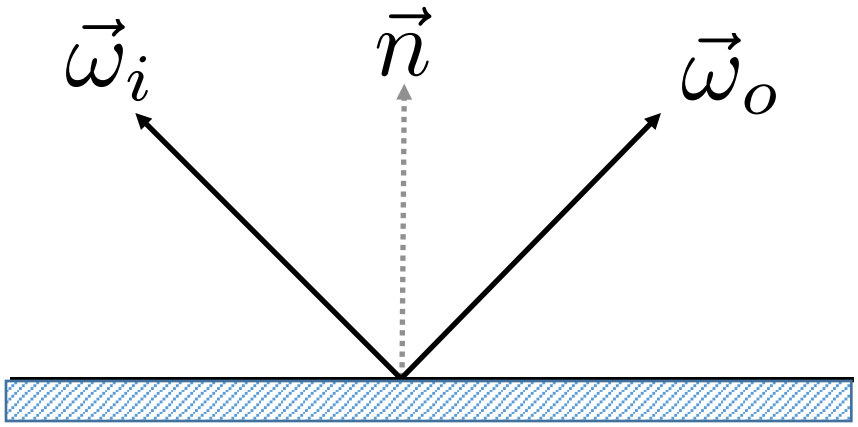
$$= L_o(\vec{\omega}_i - \vec{d})$$

Lambertian !

$$L_o(\vec{\omega}_o) = C \int_\Omega L_i \delta\left(\vec{w}_i - \vec{d}\right)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

$$= CL\left(\vec{n} \cdot \vec{d}\right)$$

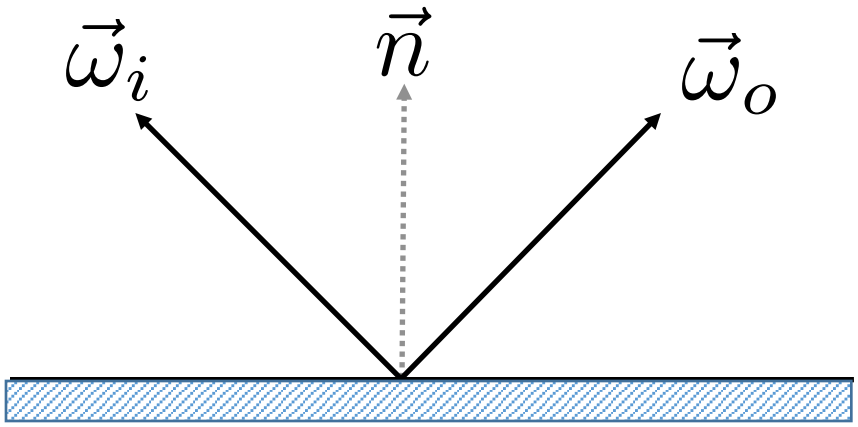
Reflection BRDF



$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = ?$$

$$r(\vec{\omega}_i) : \text{Reflection of vector } \vec{\omega}_i$$
$$= 2\left(\vec{\omega}_i \cdot \vec{n}\right)\vec{n} - \vec{\omega}_i$$

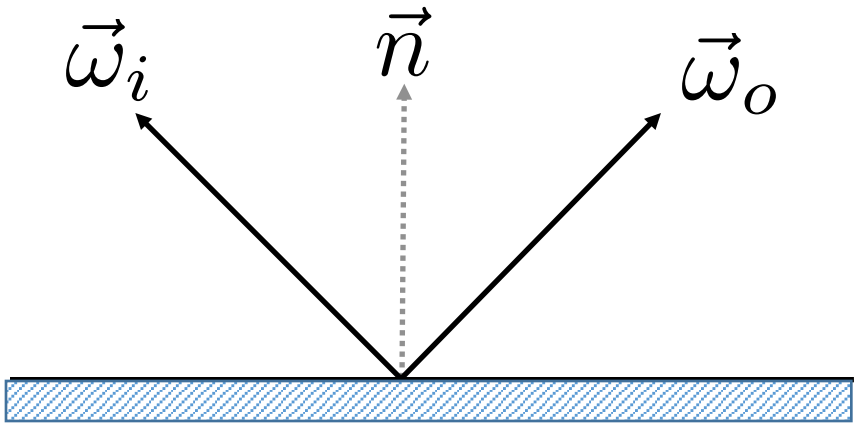
Reflection BRDF



$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega_i}, \vec{\omega_o}\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = \frac{\delta\left(\vec{\omega_i} - r(\vec{\omega_o})\right)}{\vec{n} \cdot \vec{\omega_i}}$$

Reflection BRDF



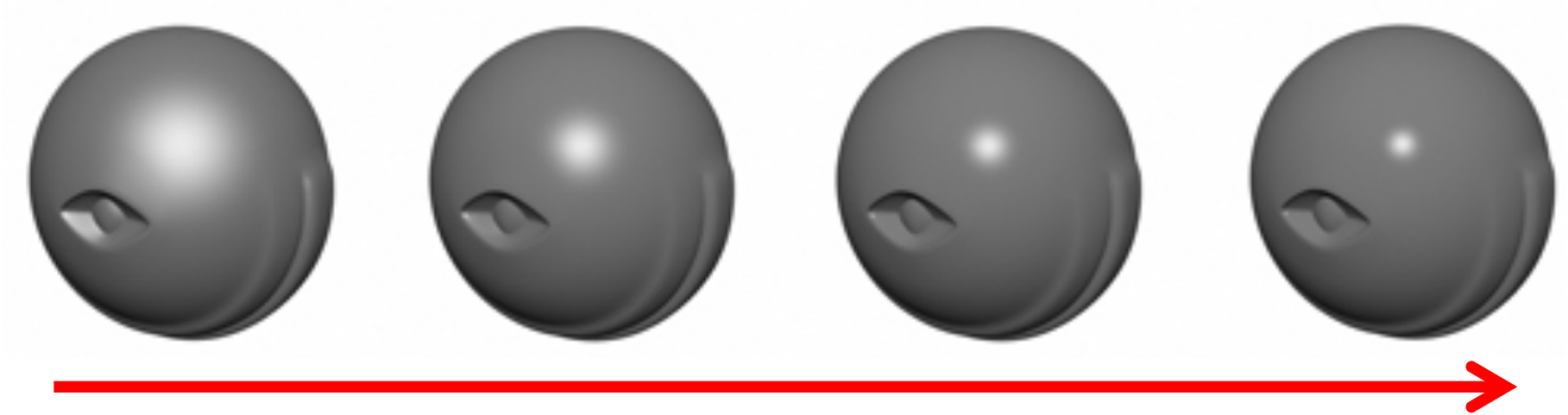
$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega}_i, \vec{\omega}_o\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = \frac{\delta\left(\vec{\omega}_i - r(\vec{\omega}_o)\right)}{\vec{n} \cdot \vec{\omega}_i}$$

$$L_o\left(\vec{\omega}_o\right) = L_i\left(r(\vec{\omega}_o)\right)$$

Phong Specular Term

$$L_o = L \cdot (r(\vec{\omega}_i) \cdot \vec{\omega}_o)^n$$



Increasing n

http://http.developer.nvidia.com/CgTutorial/elementLinks/fig5_12.jpg

Phong Specular Term

$$L_o = L \cdot (r(\vec{\omega}_i) \cdot \vec{\omega}_o)^n$$

Increasing n

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = ?$$

http://http.developer.nvidia.com/CgTutorial/elementLinks/fig5_12.jpg

Ambient BRDF ?

$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega}_i, \vec{\omega}_o\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

Ambient BRDF ?

$$L_o(\vec{\omega}_o) = \int_\Omega f\,(\vec{\omega}_i, \vec{\omega}_o)\, L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

Doesn't depend on light position/direction

Ambient BRDF ?

$$L_o(\vec{\omega}_o) = \int_\Omega f\left(\vec{\omega}_i, \vec{\omega}_o\right) L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

Doesn't

Full integral for the scene will create ambiance !

tion

Adding BRDFs

$$f\left(\vec{\omega_i}, \vec{\omega_o}\right) = f_1\left(\vec{\omega_i}, \vec{\omega_o}\right) + f_2\left(\vec{\omega_i}, \vec{\omega_o}\right) + \ldots$$

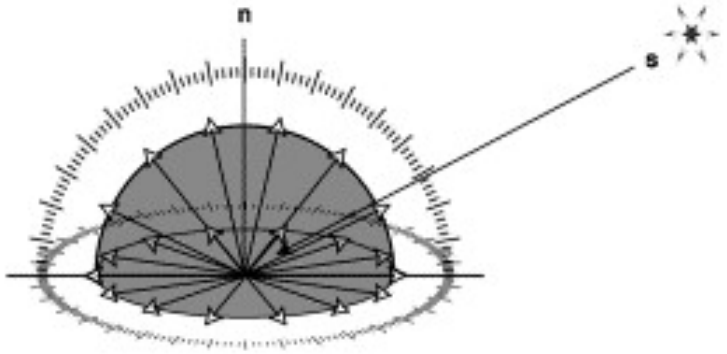
e.g. Diffuse Terms + Specular Term

Revisit Phong Reflection Model

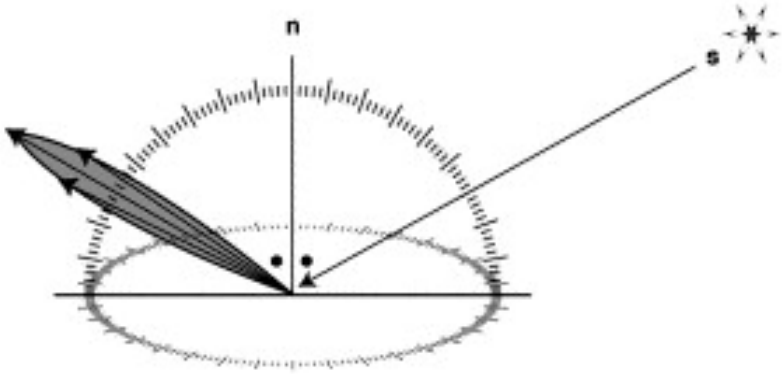
$$L_o(\vec{\omega}_o) = L_e(\vec{\omega}_o) + \int_\Omega f\,(\vec{\omega}_i, \vec{\omega}_o)\, L_i(\omega_i)(\vec{n} \cdot \vec{w}_i) d\vec{\omega}_i$$

$$\text{Phong Model} : L_o(\vec{\omega}_o) = \text{Ambient} + \underbrace{\text{Emission}}_{optional} + \text{Diffuse} + \text{Specular}$$

Visualizing BRDFs

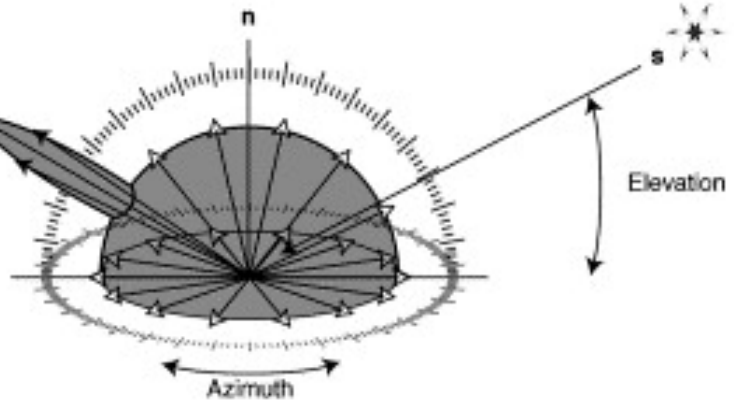


Lambertian



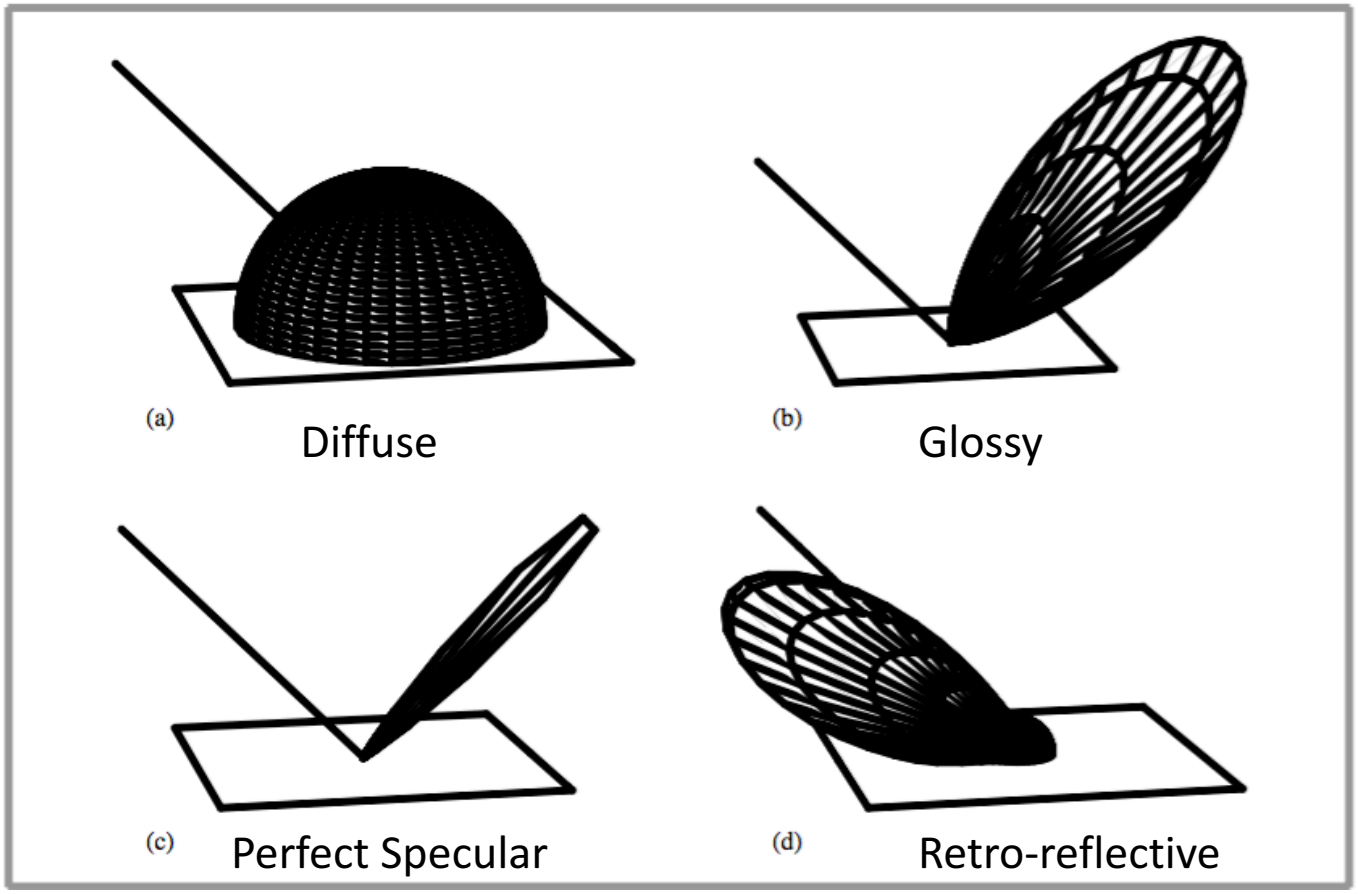
Specular

Combined



http://www.sciencedirect.com/science/article/pii/S0001691805000909

Types of Material



Chapter 8: M. Pharr and G. Humphreys, PBRT v2

Revisiting OpenGL 1.x

Enable a Light

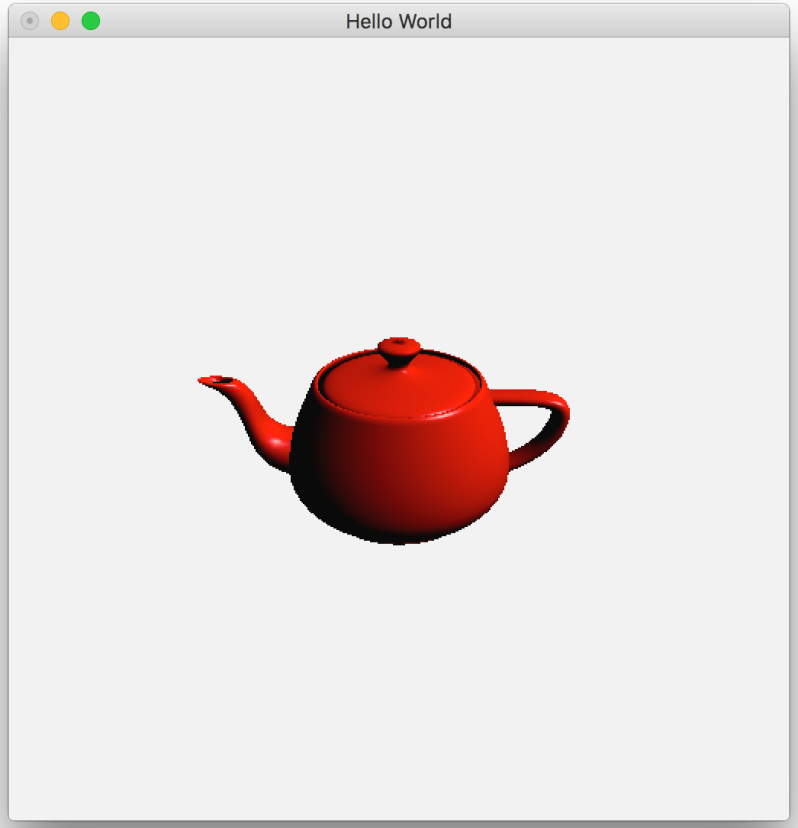
```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

float lightDir[] = {1,1,1,0};
glLightfv(GL_LIGHT0,GL_POSITION,lightDir);
```

Setup material

```
float color[] = {1,0,0,1};
float specular[] = {1,1,1,1};
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,  color);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 128);

glutSolidTeapot(1);
```

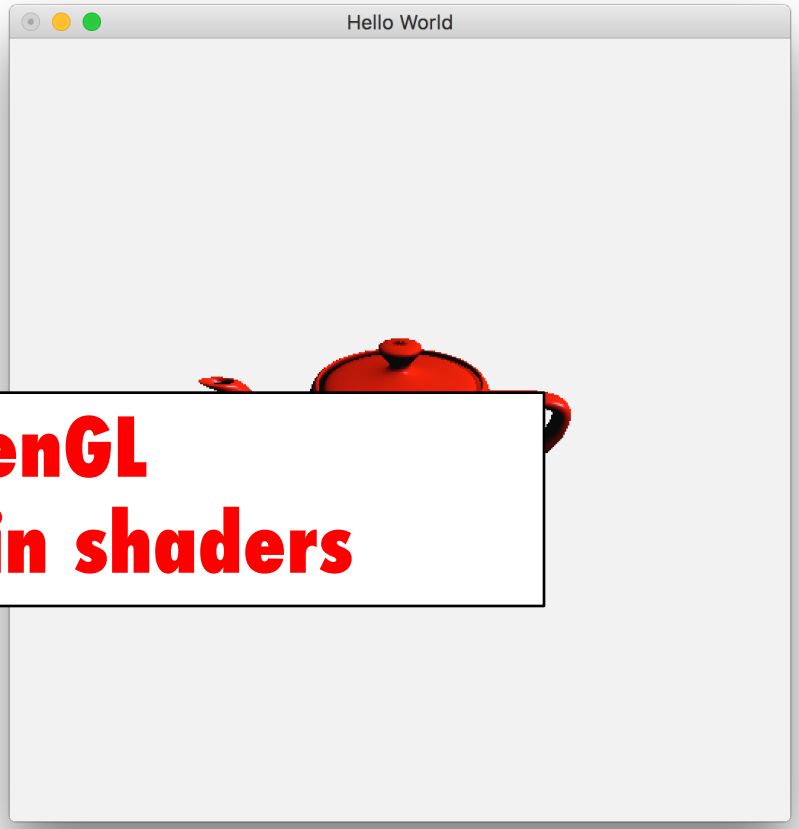


Revisiting OpenGL 1.x

Enable a Light

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

float lightDir[] = {1,1,1,0};
glLightfv(GL_LIGHT0,GL_POSITION,lightDir);
```



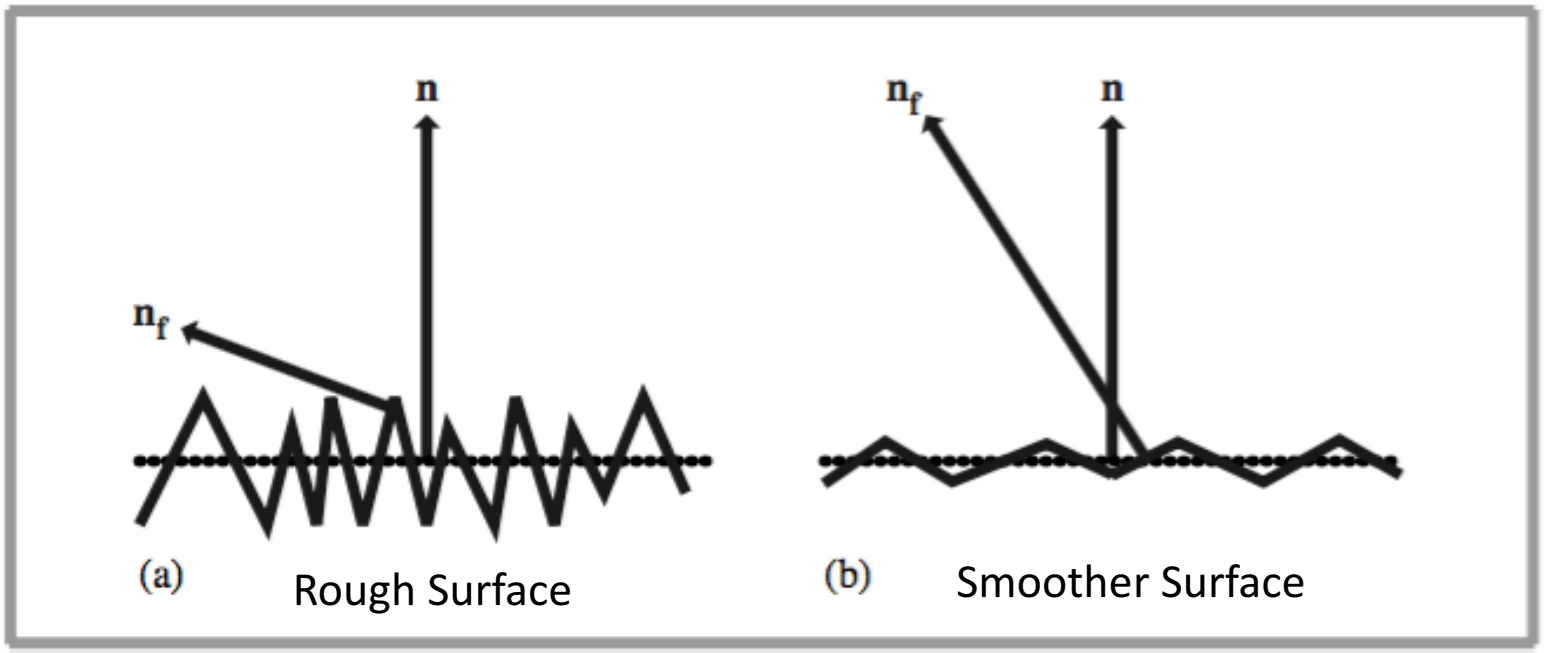
Setup

Modern OpenGL
everything done in shaders

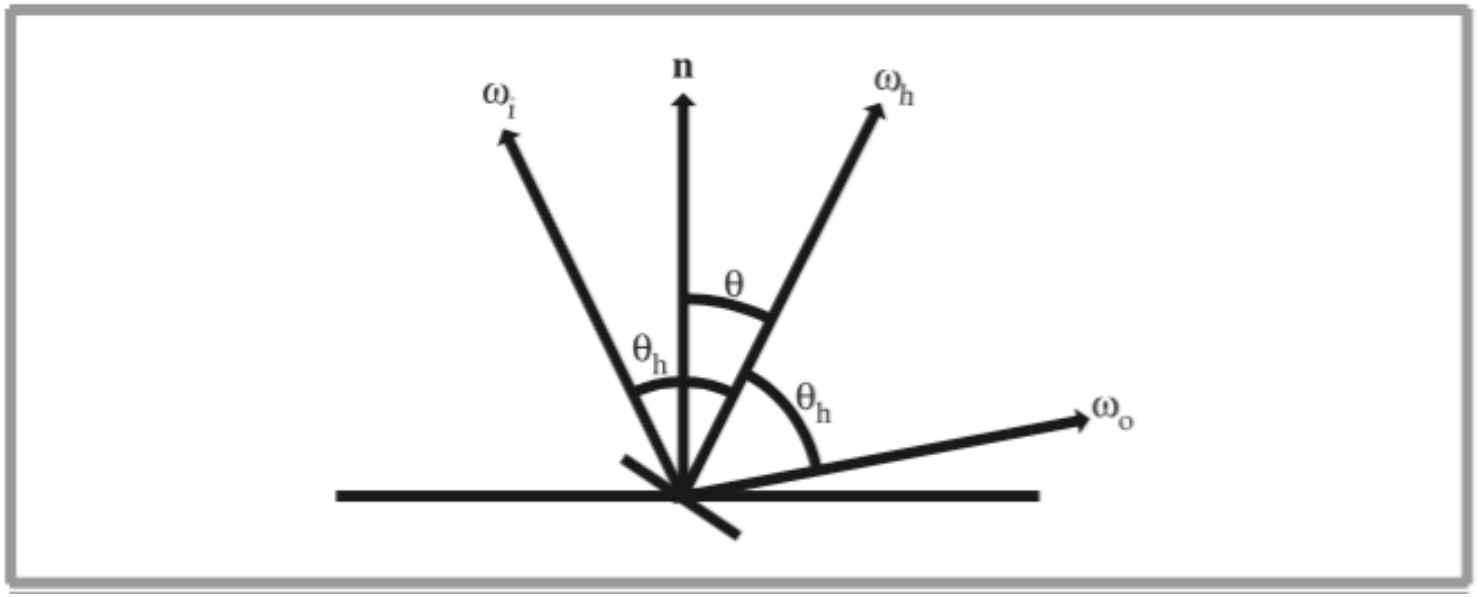
```
float c
float specular[] = {1,1,1,1};
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,  color);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 128);

glutSolidTeapot(1);
```

Microfacet based BRDF



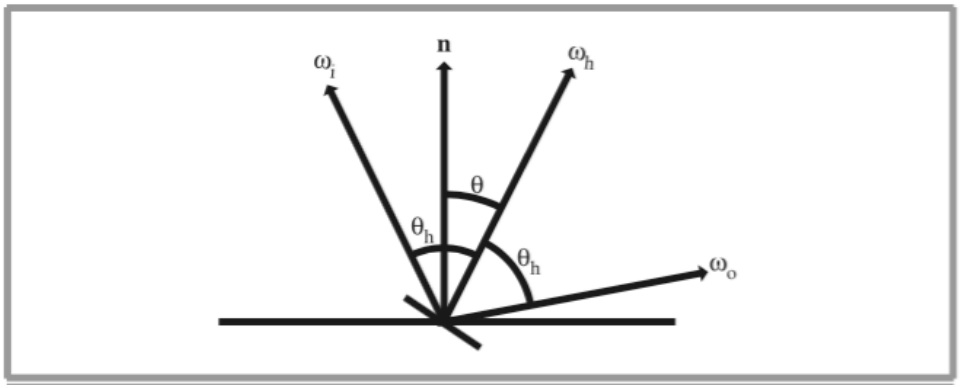
Microfacet: Torrance-Sparrow



Each microfacet is perfectly reflective

For a given $\vec{\omega}_i, \vec{\omega}_o$, all the microfacets with orientation $\vec{\omega}_h = \underbrace{\widehat{\vec{\omega}_i + \vec{\omega}_o}}_{\text{half angle}}$ will reflect

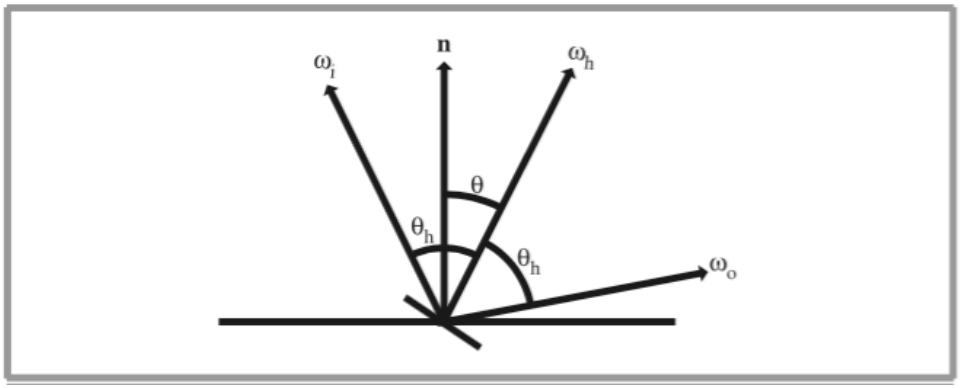
Microfacet: Torrance-Sparrow



Key idea:

For a given $\vec{\omega}_i, \vec{\omega}_o$, all the microfacets with orientation $\vec{\omega}_h = \underbrace{\widehat{\vec{\omega}_i + \vec{\omega}_o}}_{\text{half angle}}$ will reflect

Microfacet: Torrance-Sparrow

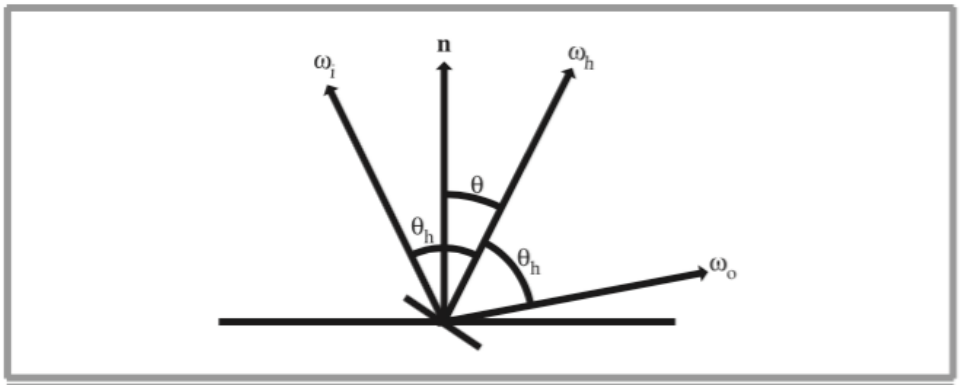


Key idea:

For a given $\vec{\omega}_i, \vec{\omega}_o,$ all the microfacets with orientation $\vec{\omega}_h = \underbrace{\widehat{\vec{\omega}_i + \vec{\omega}_o}}_{\text{half angle}}$ will reflect

$D\left(\vec{\omega}_h\right)$: Distribution of microfacets with orientation $\vec{\omega}_h$

Microfacet: Torrance-Sparrow



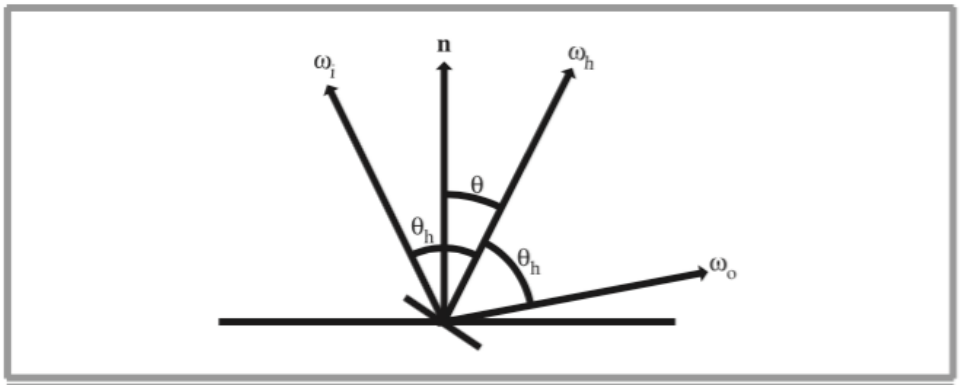
Key idea:

For a given $\vec{\omega}_i, \vec{\omega}_o,$ all the microfacets with orientation $\vec{\omega}_h = \underbrace{\widehat{\vec{\omega}_i + \vec{\omega}_o}}_{\text{half angle}}$ will reflect

$$D\left(\vec{\omega}_h\right) : \text{Distribution of microfacets with orientation } \vec{\omega}_h$$

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = \frac{\mathrm{D}(\vec{\omega}_h)}{4\left(\vec{n} \cdot \vec{\omega}_o\right)\left(\vec{n} \cdot \vec{\omega}_i\right)}$$

Microfacet: Torrance-Sparrow



Key idea:

For a given $\vec{\omega}_i, \vec{\omega}_o,$ all the microfacets with orientation $\vec{\omega}_h = \underbrace{\widehat{\vec{\omega}_i + \vec{\omega}_o}}_{\text{half angle}}$ will reflect

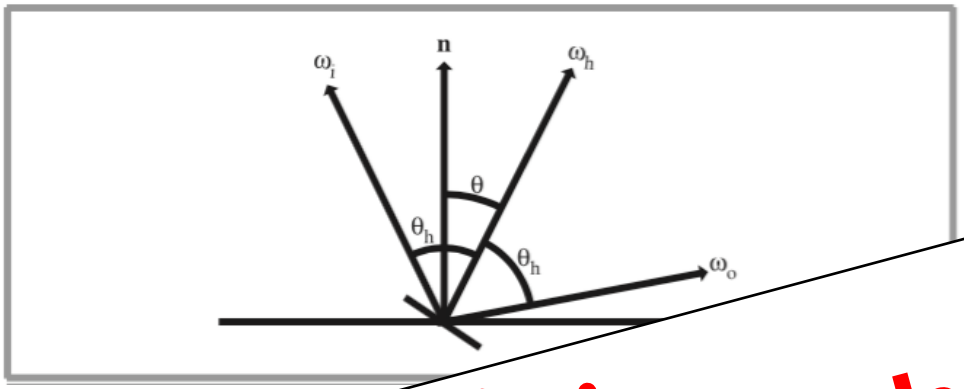
$D\left(\vec{\omega}_h\right)$: Distribution of microfacets with orientation $\vec{\omega}_h$

$$f\left(\vec{\omega}_i, \vec{\omega}_o\right) = \frac{\mathrm{D}(\vec{\omega}_h)}{4\left(\vec{n}\cdot\vec{\omega}_o\right)\left(\vec{n}\cdot\vec{\omega}_i\right)}$$

$$D\left(\vec{\omega_h}\right) = \frac{e+2}{2\pi}\left(\vec{\omega_h}\cdot\vec{n}\right)^e$$

Blinn's Distribution

Microfacet: Torrance-Sparrow



Key idea:

For a given [illegible] $\underbrace{\vec{\omega}_i + \vec{\omega}_o}$ will reflect
half angle

[illegible] microfacets with orientation $\vec{\omega}_h$

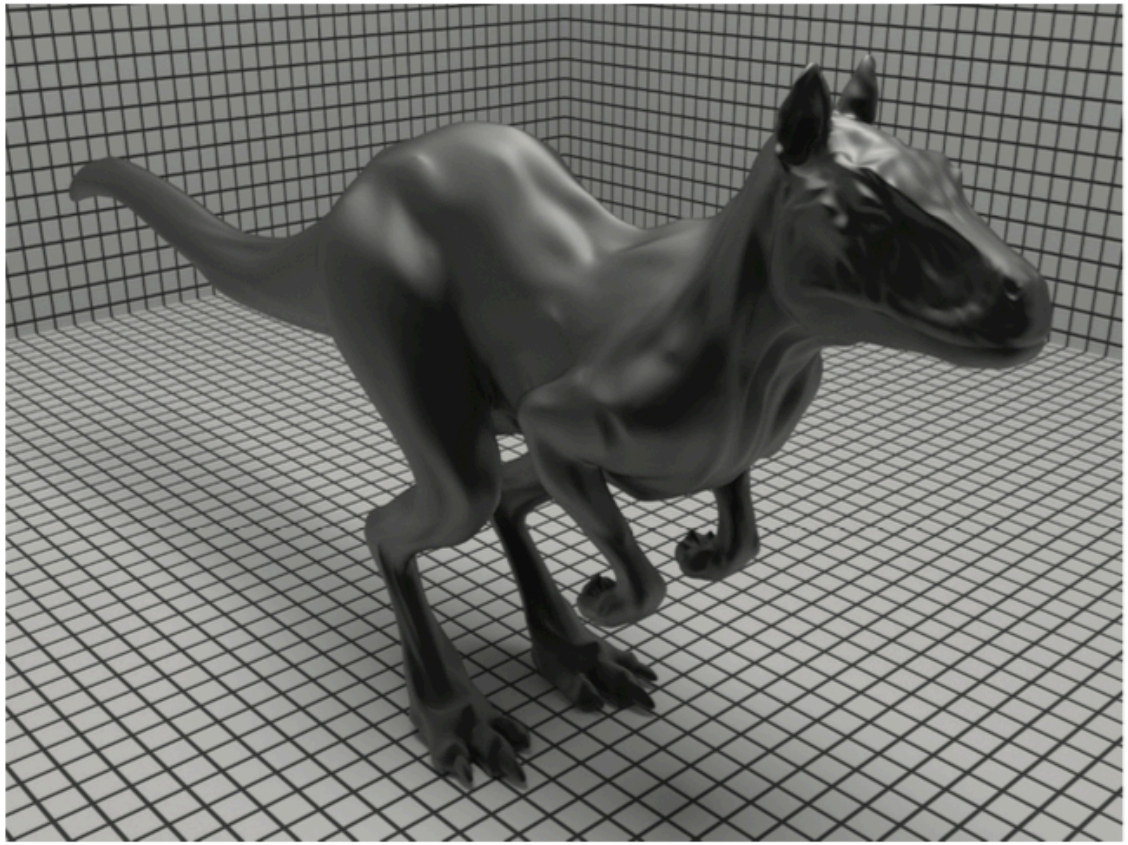
Exercise:
Derive shading equation for a single light ray for OpenGL implementation

$$f(\vec{\omega}_i, \vec{\omega}_o) = \frac{\mathrm{D}(\vec{\omega}_h)}{4(\vec{n} \cdot \vec{\omega}_o)(\vec{n} \cdot \vec{\omega}_i)}$$

$$D(\vec{\omega}_h) = \frac{e+2}{2\pi}(\vec{\omega}_h \cdot \vec{n})^e$$

Blinn's Distribution

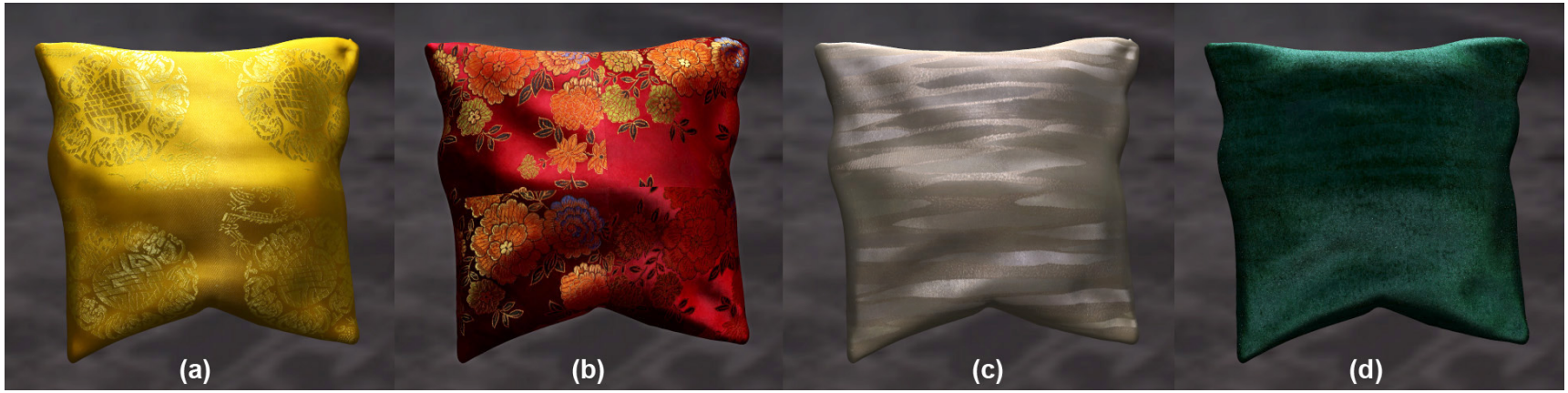
Microfacet: Torrance-Sparrow



With Blinn's distribution

Has a nice parameter to vary from "smooth" to "rough"

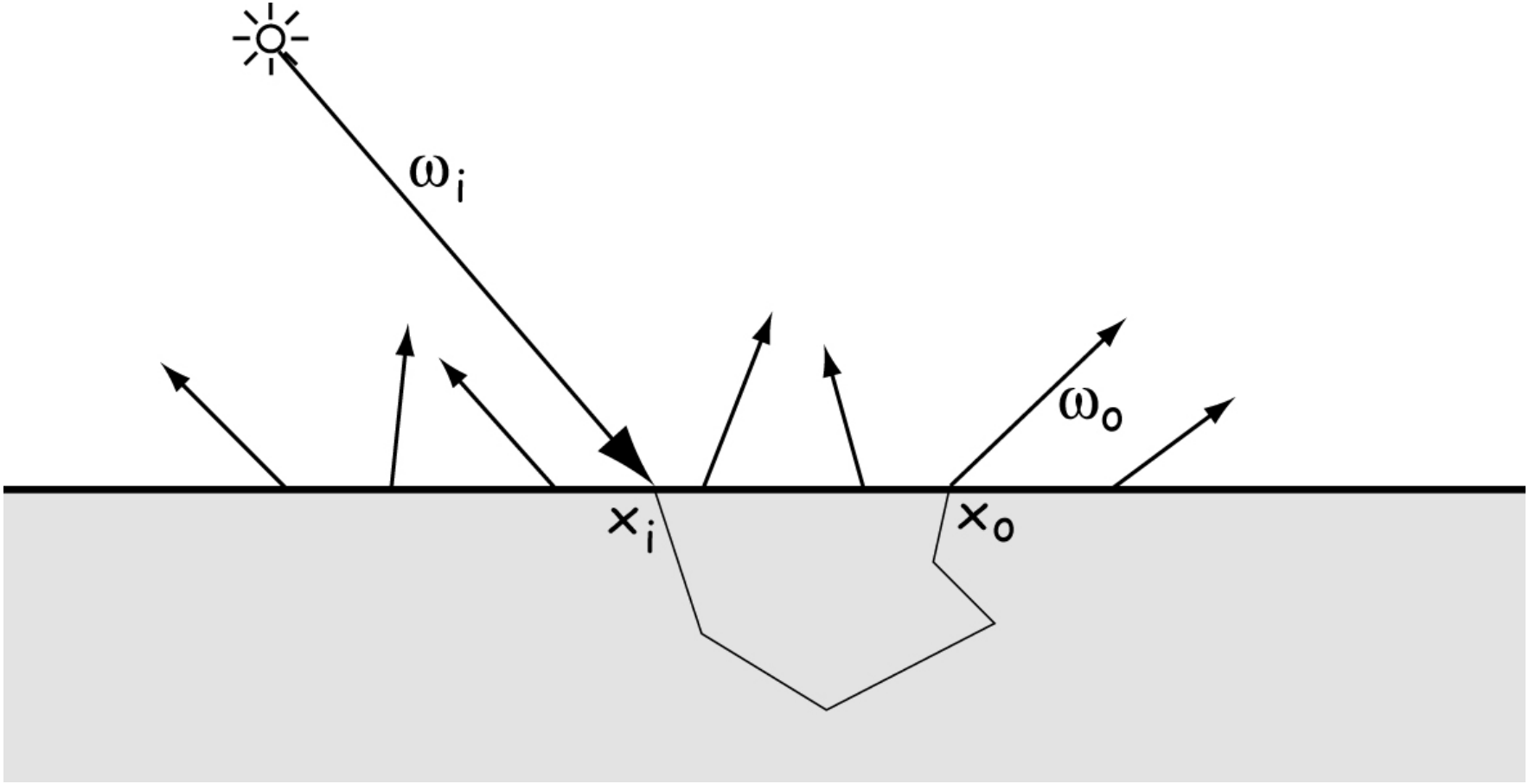
Extensions of BRDFs



Spatially-varying BRDF (SVBRDF)

http://research.microsoft.com/en-us/um/people/johnsny/images/aniso_pillows.bmp

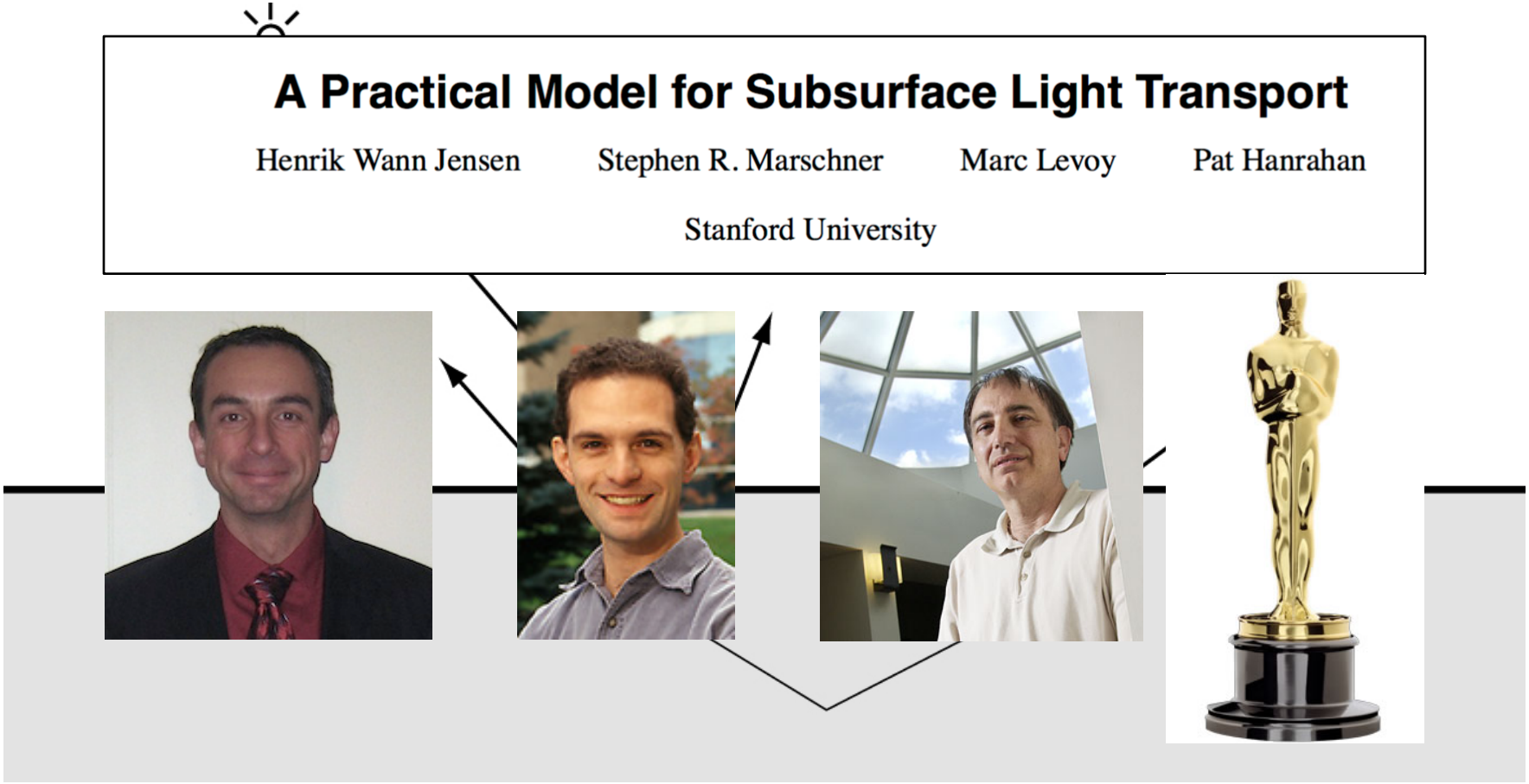
Extensions of BRDFs



B Subsurface Scattering RDF (BSSRDF)

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

Extensions of BRDFs



B Subsurface Scattering RDF (BSSRDF)

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

Extensions of BRDFs

B Subsurface Scattering RDF (BSSRDF)

http://graphics.ucsd.edu/~henrik/images/imgs/diana_bssrdf.jpg

Extensions of BRDFs

B Subsurface Scattering RDF (BSSRDF)

http://graphics.ucsd.edu/~henrik/images/imgs/face_brdf.jpg

Extensions of BRDFs

B Subsurface Scattering RDF (BSSRDF)

http://graphics.ucsd.edu/~henrik/images/imgs/milk.jpg

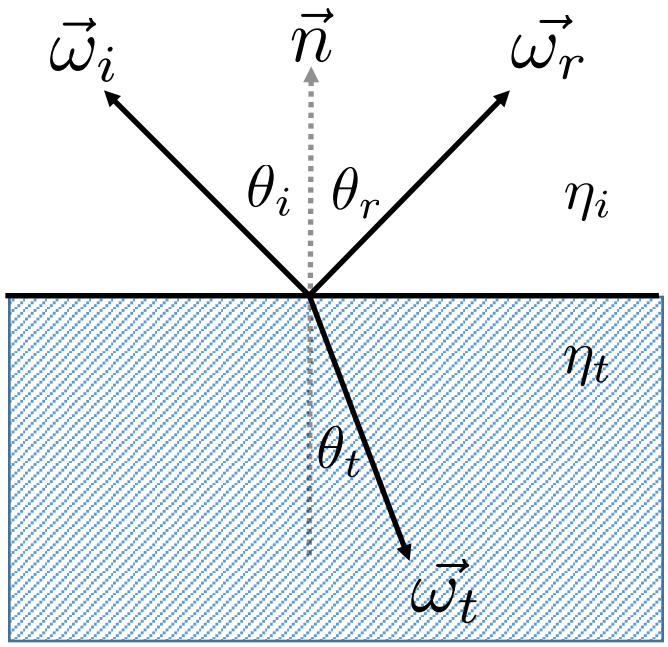
Extensions of BRDFs

B Subsurface Scattering RDF (BSSRDF)

http://graphics.ucsd.edu/~henrik/images/imgs/milk.jpg

Fresnel Term

Fresnel Term



For Dielectrics

$$r_{\parallel} = \frac{\eta_t \cos(\theta_i) - \eta_i \cos(\theta_t)}{\eta_t \cos(\theta_i) + \eta_i \cos(\theta_t)}$$

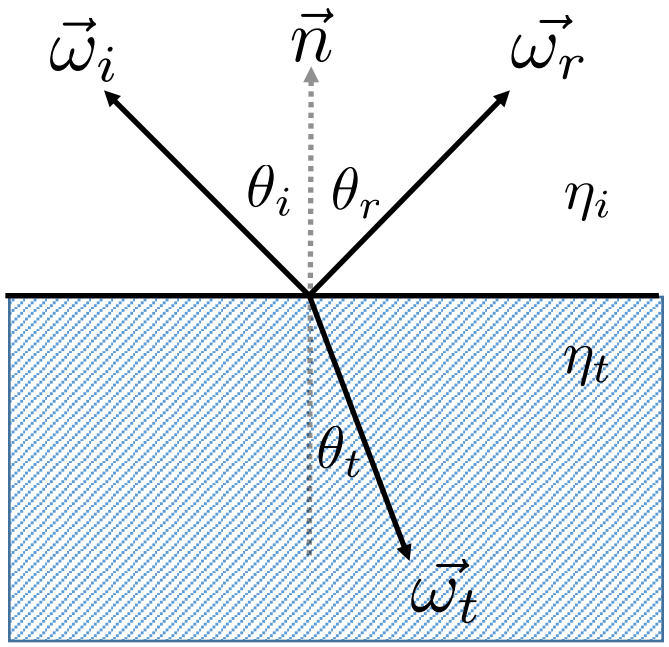
$$r_{\perp} = \frac{\eta_i \cos(\theta_i) - \eta_t \cos(\theta_t)}{\eta_i \cos(\theta_i) + \eta_t \cos(\theta_t)}$$

For unpolarized light

$$F_r = \frac{1}{2}\left(r_{\parallel}^2 + r_{\perp}^2\right)$$

$$F_t = 1 - F_r$$

Fresnel Term



For Dielectrics

$$r_{\parallel} = \frac{\eta_t \cos(\theta_i) - \eta_i \cos(\theta_t)}{\eta_t \cos(\theta_i) + \eta_i \cos(\theta_t)}$$

$$r_{\perp} = \frac{\eta_i \cos(\theta_i) - \eta_t \cos(\theta_t)}{\eta_i \cos(\theta_i) + \eta_t \cos(\theta_t)}$$

For unpolarized light

$$F_r = \frac{1}{2}\left(r_{\parallel}^2 + r_{\perp}^2\right)$$

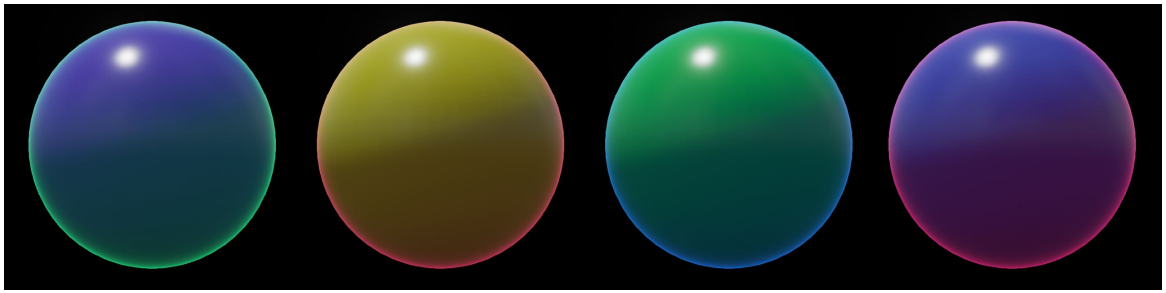
$$F_t = 1 - F_r$$

Schlick's Approximation

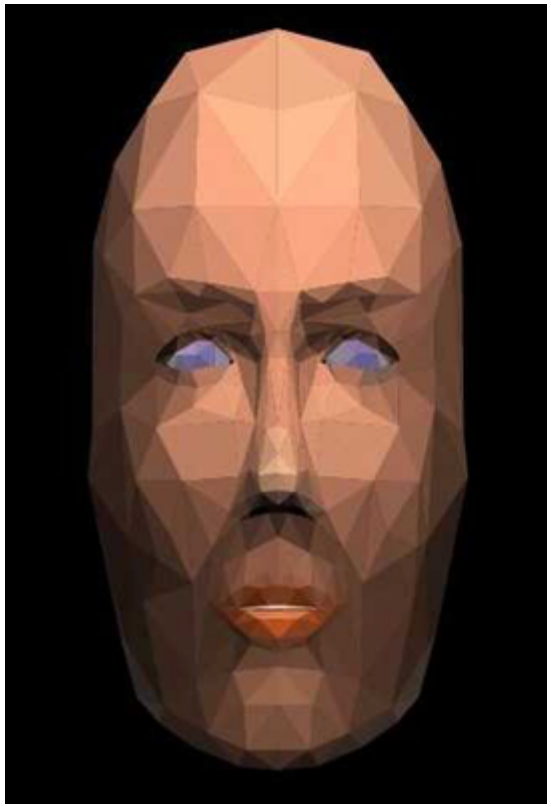
$$F_r = F_0 + (1 - F_0)(1 - \cos\theta_i)^5$$

$$F_0 = \left(\frac{\eta_t - \eta_i}{\eta_t + \eta_i}\right)^2$$

Fresnel Term: Game Effects



Algorithms for Shading



Algorithm:

1. **Calculate color *c* of triangle.**
2. **Fill whole triangle with *c***

Flat shading: One color per triangle

http://www.svgopen.org/2005/papers/Converting3DFaceToSVG/Figure_10.png

Algorithms for Shading

Algorithm:

1. **Calculate color c_i of each vertex.**
2. **Interpolate pixel colors using barycentric weights.**

Gouraud shading: One color per vertex

http://www.svgopen.org/2005/papers/Converting3DFaceToSVG/Figure_10.png

Algorithms for Shading

Algorithm:

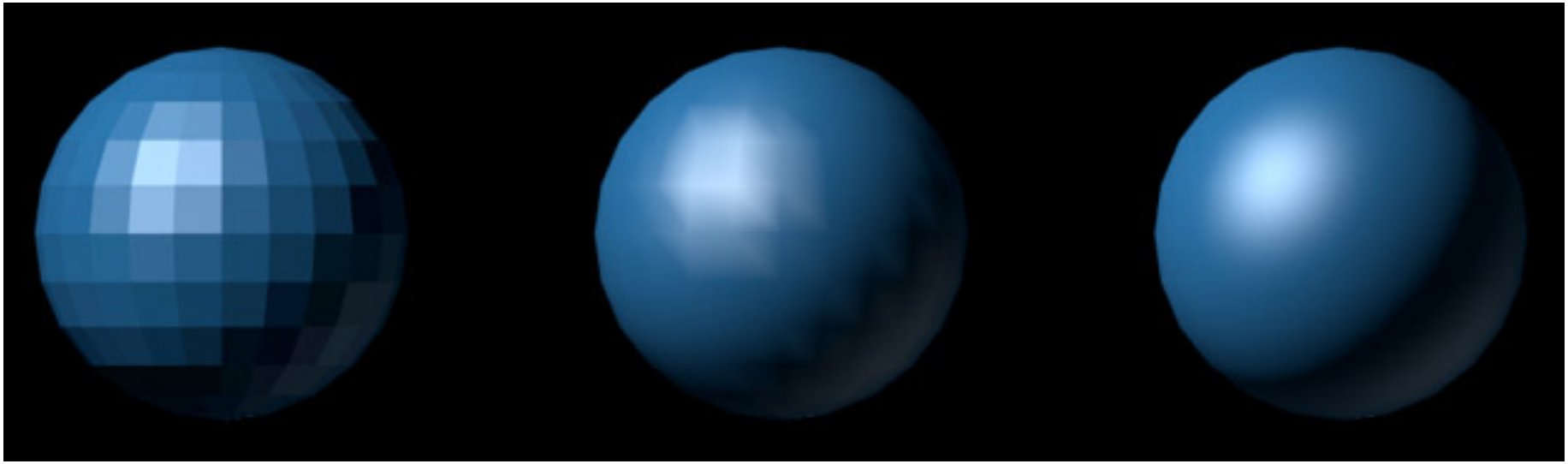
1. Calculate normal N_i of each vertex.
2. Interpolate normal using barycentric weights, normalize, and use it for shading.

Phong shading: One color per fragment

a.ka. Per Pixel Lighting

http://www.cs.cmu.edu/~ph/nyit/facet_gouraud_phong.jpeg

Comparison



Which is which?

http://www.inf.pucrs.br/~pinho/CG/Aulas/Iluminacao/Gouraud_Phong_01.jpg

Materials

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

Cook, Robert L., and Kenneth E. Torrance, *ACM Transactions on Graphics (TOG)* 1982.