

CS 354

Lighting

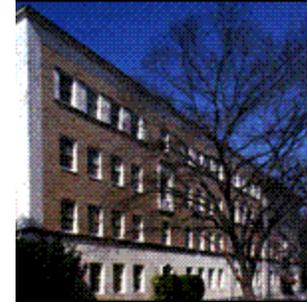
Mark Kilgard
University of Texas
February 28, 2012

Today's material

- In-class quiz
- Lecture topics
 - *Lighting surfaces*
- Course work
 - Homework
 - Turn in Homework #3 today
 - Homework #4 to be assigned next class, due March 6th
 - Reading
 - Chapter 7, 388-403 on Programmable Shading
 - Project #2 on texturing, shading, & lighting is coming
 - Will be due after Spring Break
 - **Remember: Midterm in-class on March 8**

My Office Hours

- Tuesday, before class
 - Painter (PAI) 5.35
 - 8:45 a.m. to 9:15
- Thursday, after class
 - ACE 6.302
 - 11:00 a.m. to 12:00



Last time, this time

- Last lecture, we discussed
 - Texture mapping
 - What's involved in a texel fetch operation?
- This lecture
 - How do we simulate how light interacts with visible surfaces in computer graphics?

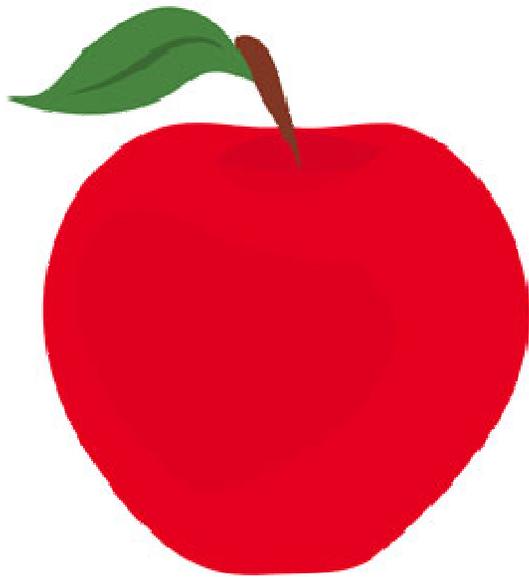
Daily Quiz

On a sheet of paper

- Write your EID, name, and date
- Write #1, #2, #3, #4 followed by its answer

1. How many texture coordinate components are used to access a cube map?
2. In conventional texture mapping, elements of the texture (texels) store color values. What is stored in the texel values for shadow mapping?
3. If bilinear filtering involves interpolating between 4 texel samples, mipmap bilinear filtering involves filtering how many texels?
4. **Multiple choice:** The interval between the Texel Selection and Texel Combinations stages of a texture fetch is meant to hide what?
 - a) texture seams
 - b) memory latency
 - c) cache hits
 - d) filtering artifacts
 - e) all of the above

What makes an apple red?



Child's view:
“an apple is red”

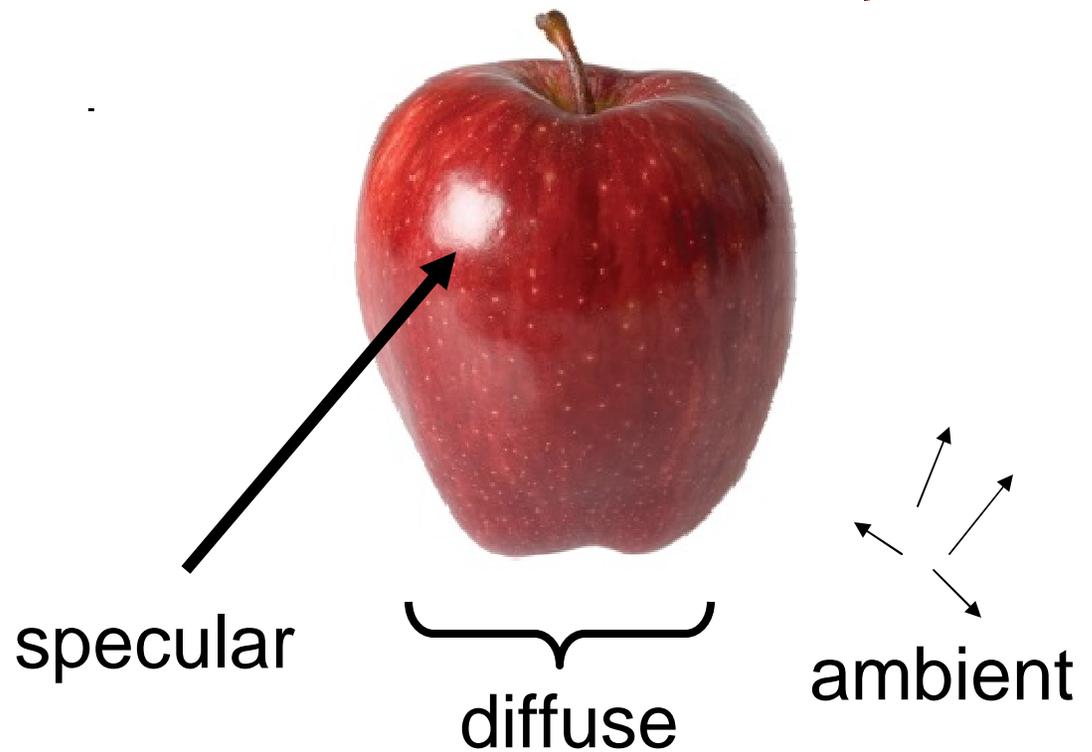


Image synthesis view:
“light, surface, and material interact to reflect light perceived as color, modeled via simplifying assumptions”

Complex Realistic Lighting



iray



OptiX

Why lighting?

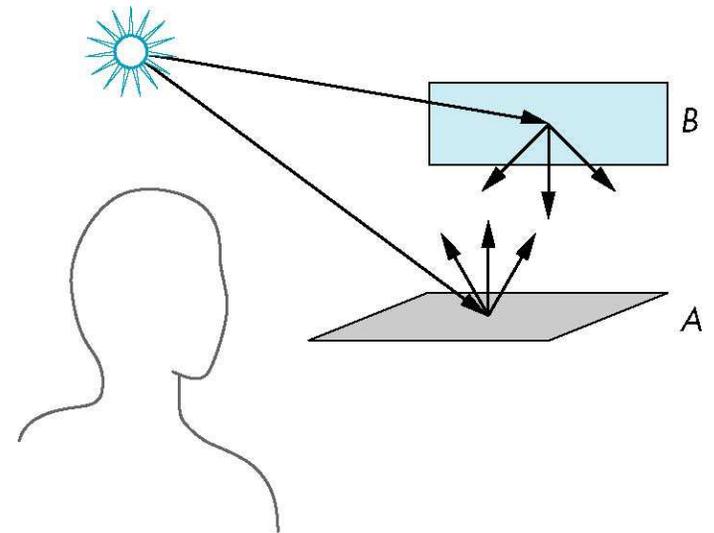
- Convey sense of surface roughness/smoothness
- Convey sense of material
 - Visual appearance of plastic is different than copper or ebony
- Realism
- Emotion

Teapots rendered with different material parameters

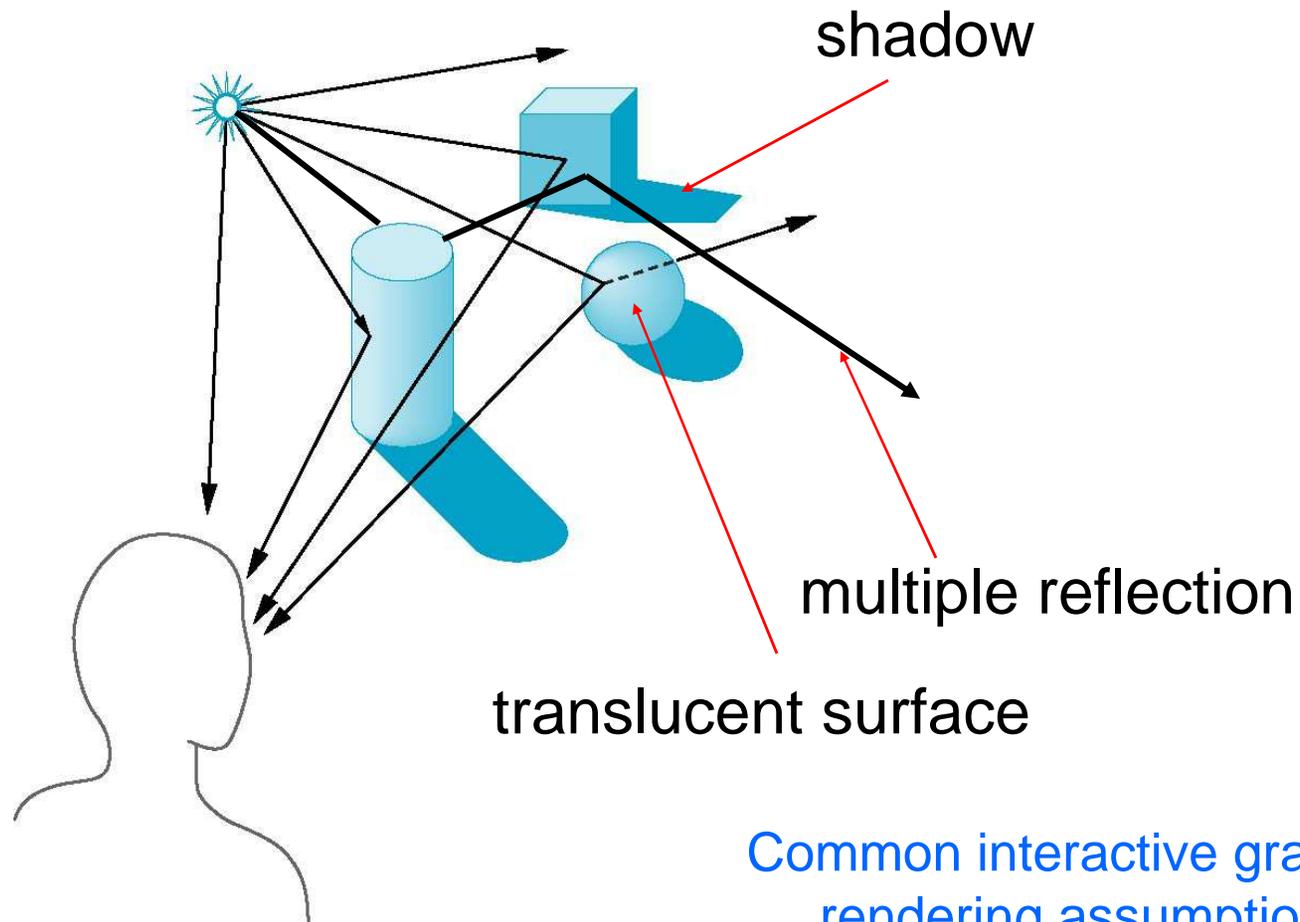


Light Scatters

- Light strikes A
 - Some scattered
 - Some absorbed
- Some of scattered light strikes B
 - Some scattered
 - Some absorbed
- Some of this scattered light strikes A and so on
 - Inherently recursive
 - Massively parallel
 - Points to the [ray tracing](#) methodology for rendering



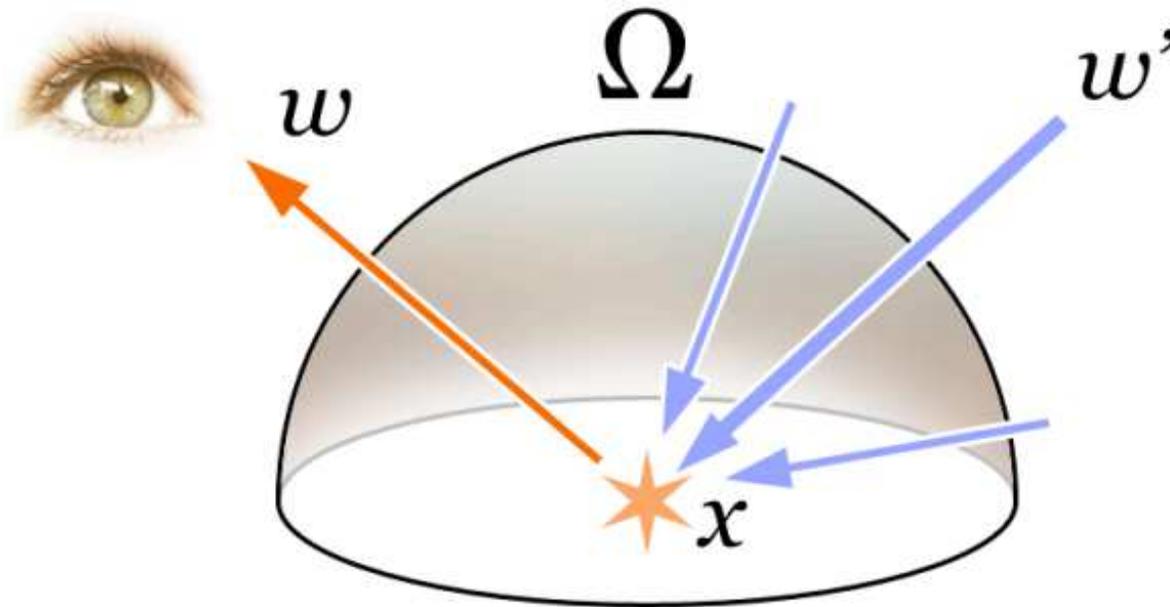
Global Lighting Effects



Common interactive graphics
rendering assumption:
only model local lighting effects

Rendering Equation

- Theory for light-surface interactions



$$L_o(\mathbf{x}, \omega, \lambda, t) = L_e(\mathbf{x}, \omega, \lambda, t) + \int_{\Omega} f_r(\mathbf{x}, \omega', \omega, \lambda, t) L_i(\mathbf{x}, \omega', \lambda, t) (-\omega' \cdot \mathbf{n}) d\omega'$$

Rendering Equation Parts

$$L_o(\mathbf{x}, \omega, \lambda, t) = L_e(\mathbf{x}, \omega, \lambda, t) + \int_{\Omega} f_r(\mathbf{x}, \omega', \omega, \lambda, t) L_i(\mathbf{x}, \omega', \lambda, t) (-\omega' \cdot \mathbf{n}) d\omega'$$

- L_o = outgoing light from \mathbf{x} in direction ω
- \mathbf{x} = point on a surface
- ω = normalized outgoing light vector
- λ = wavelength of light
- t = time
- L_e = emitted light at \mathbf{x} going towards from ω
- \mathbf{n} = surface normal at \mathbf{x}

Rendering Equation Integral

- \int = integrate over a region
- Ω = region of a hemisphere
 - Together: “integrate overall the incoming directions for a hemisphere at a point \mathbf{x} ”
- ω' = normalized outgoing light vector
- L_i = incoming light at \mathbf{x} coming from ω'
- \mathbf{n} = surface normal at \mathbf{x}
- $(-\omega' \cdot \mathbf{n})$ = cosine of angle between incoming light and surface normal
- $d\omega'$ = differential of incoming angle

Bidirectional Reflectance Distribution Function

- Ratio of differential outgoing (reflected) radiance to differential incoming irradiance

$$f_r(\mathbf{x}, \omega', \omega, \lambda, t) = \frac{dL_r(\mathbf{x}, \omega, \lambda, t)}{dE_i(\mathbf{x}, \omega', \lambda, t)} = \frac{dL_r(\mathbf{x}, \omega, \lambda, t)}{dL_i(\mathbf{x}, \omega', \lambda, t) (-\omega' \cdot \mathbf{n}) d\omega'}$$

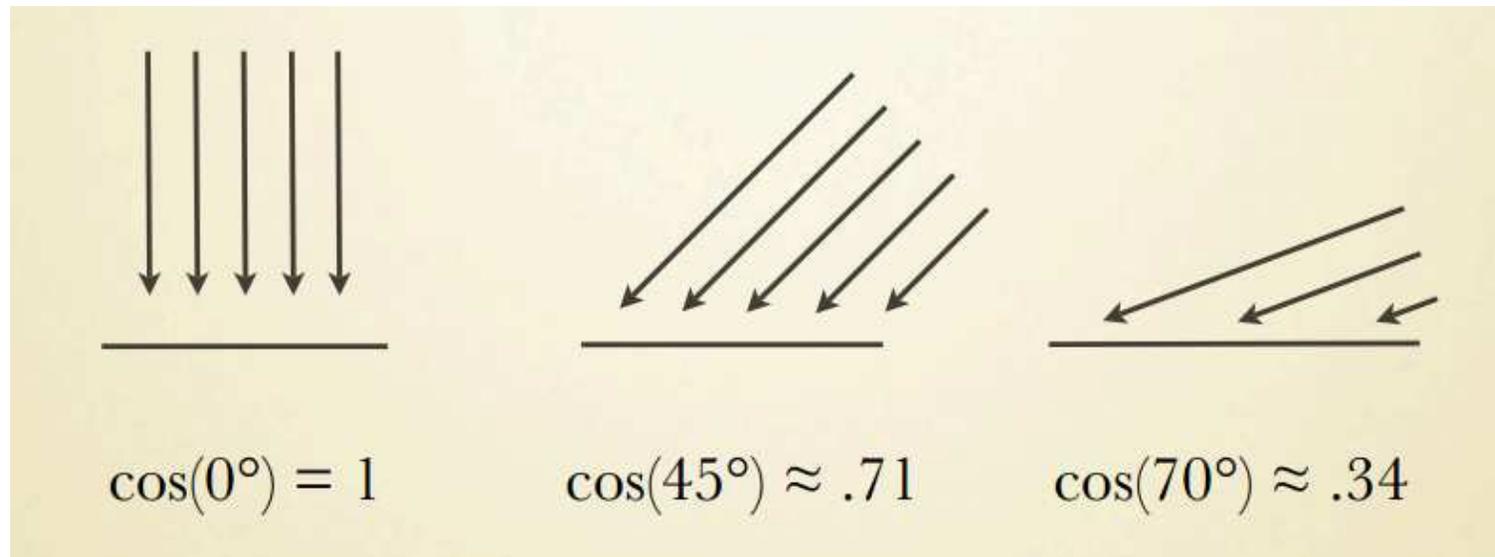
- Physically based BRDF properties

L = radiance

E = irradiance

- Must be non-negative
- Must be reciprocal
 - Swap incoming & reflected directions generates same ratio
- Conserves energy
 - Integrating over entire hemisphere must be ≤ 1

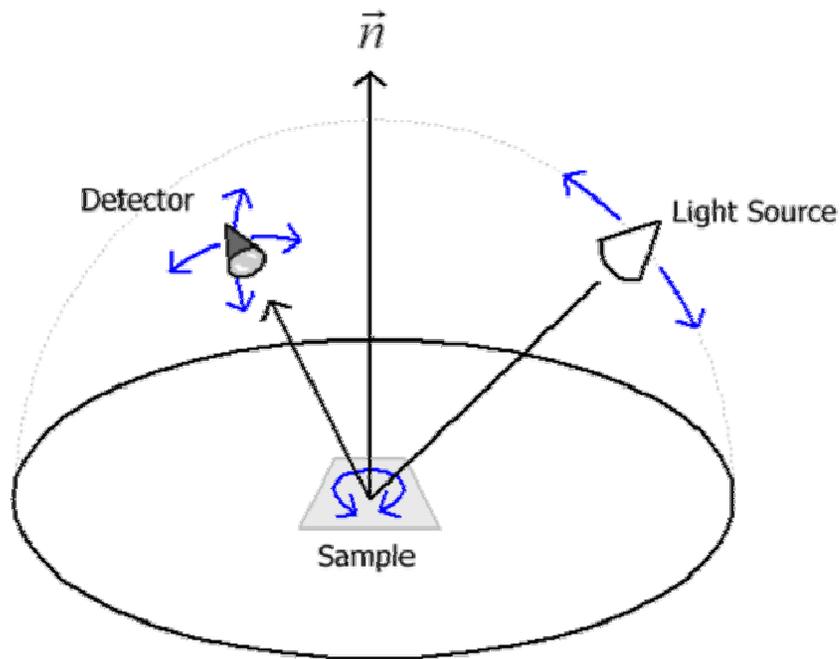
Cosine Weighting for Irradiance



- At shallow angles, incoming light spreads over a wider area of the surface
 - Thus spreading the energy
 - Thus dimming the received light

Measuring BRDFs Empirically

- Gonioreflectometer is device to measure BRDFs



Rendering Equation Interpretation

- Recursive equation

$$L_o(\mathbf{x}, \omega, \lambda, t) = L_e(\mathbf{x}, \omega, \lambda, t) + \int_{\Omega} f_r(\mathbf{x}, \omega', \omega, \lambda, t) L_i(\mathbf{x}, \omega', \lambda, t) (-\omega' \cdot \mathbf{n}) d\omega'$$

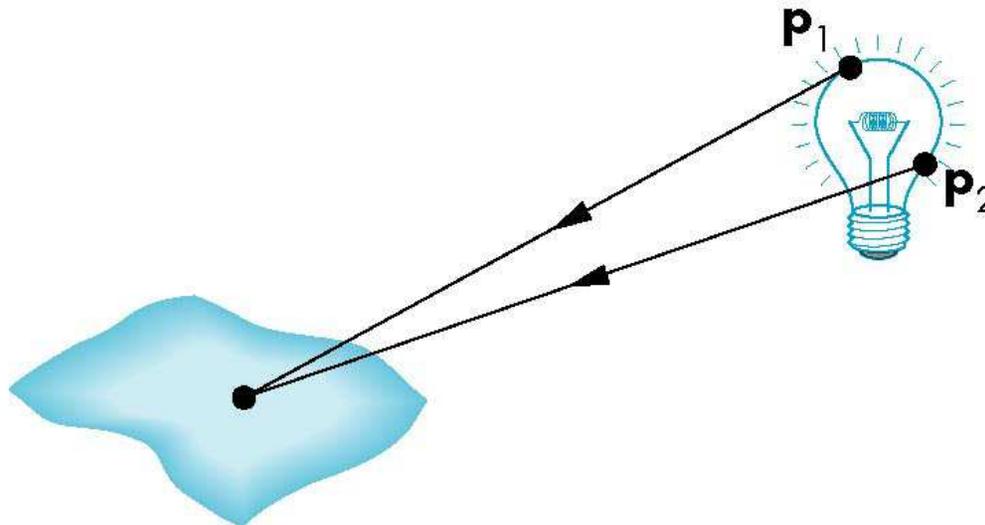
*outgoing light “sums up” all incoming light
for all surfaces*

- Limitations

- Treats wavelengths all independent
- Treats time in Newtonian way
- Ignores volumetric and subsurface scattering
- More complex models can incorporate these aspects of light*
- Impractical for actual computer graphics rendering

Light Sources

- Real lights have “emissive regions”
 - Simplifying assumption treats lights as points
 - Model area lights source as “lots of points”
 - Lights have color and attenuation
 - Fall-off, spotlight, and shadow attenuation



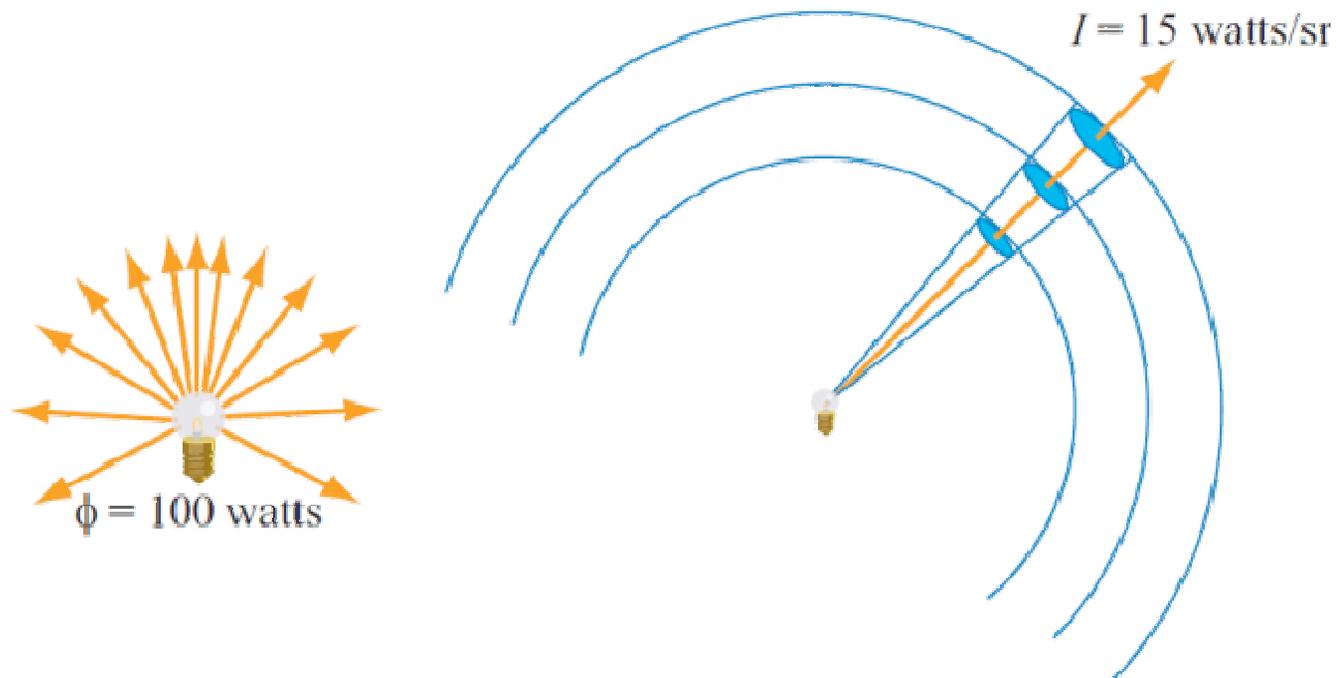
Area Light Source Modeled as Many Points

12 point light sources model an area light



Light Attenuation

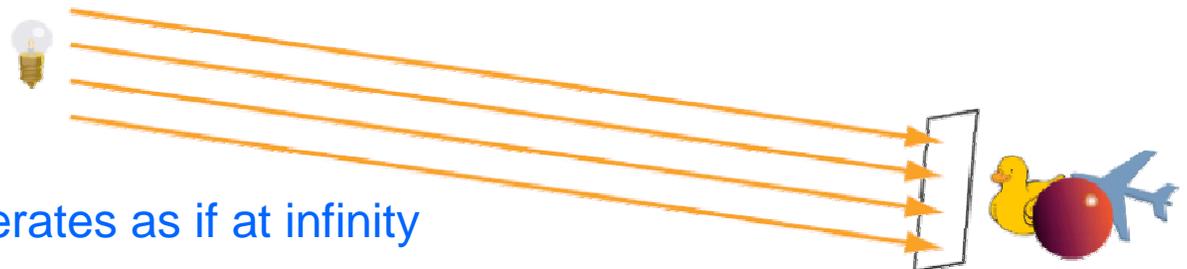
- Energy of light in watts
 - But energy received over a distance surface falls off with the inverse square of the distance from the light



Steradian (st) is measure of solid angle

Simple Light Source Types

- Positional (point) light sources
 - $(x,y,z,1)$
- Directional (infinite) light sources
 - Represented with homogeneous coordinates
 - $(x,y,z,0)$
- Spot light
 - Directional lighting
 - Often a cone or cone with fall-off function
- Linear light sources
- Ambient light
 - Useful as an error term

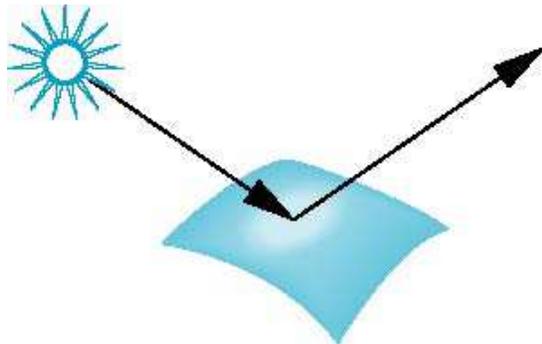


Directional light operates as if at infinity

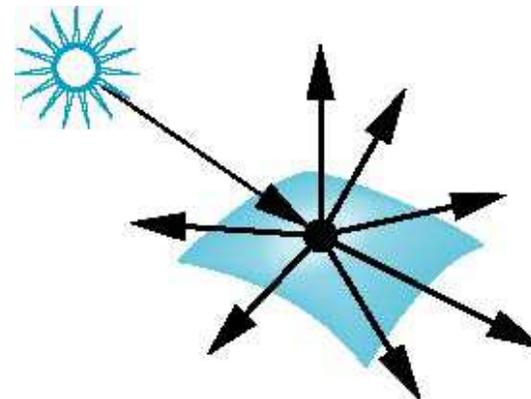
Rays are essentially parallel—sun would be such a light

Surface Types

- Smooth surfaces reflect like mirrors
- Rough surfaces scatter light
- *Real surfaces act as mixture of both*



**Smooth reflective
surface**



**Rough diffuse
surface**

Emissive Light

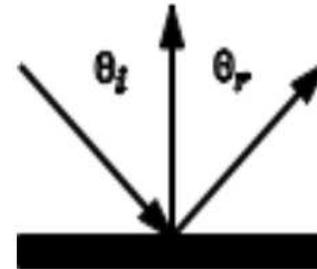
- Certain objects emit light
 - Various reasons: incandescence, burning, fluorescence, phosphorescence
 - Typically models as a emissive color



Types of Reflected Light

■ Mirror reflection

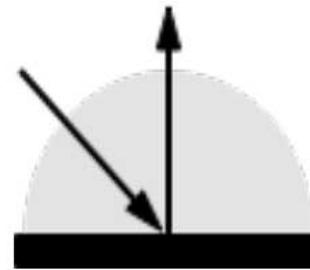
- Ideal reflection
- Reflection Law



Phong
lighting

■ Diffuse reflection

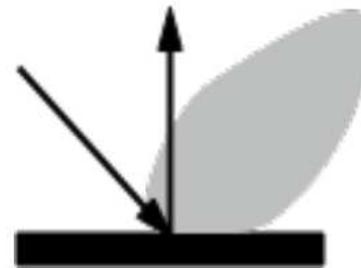
- Matte, flat finish
- Lambert's Law



Lambertian
lighting

■ Specular

- Highlights and gloss
- Micro-facet model

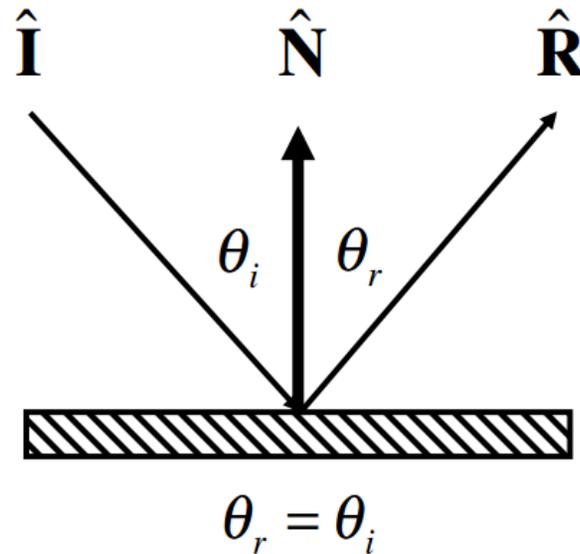


Blinn
lighting

Mirror: Law of Reflection



Planar reflectance



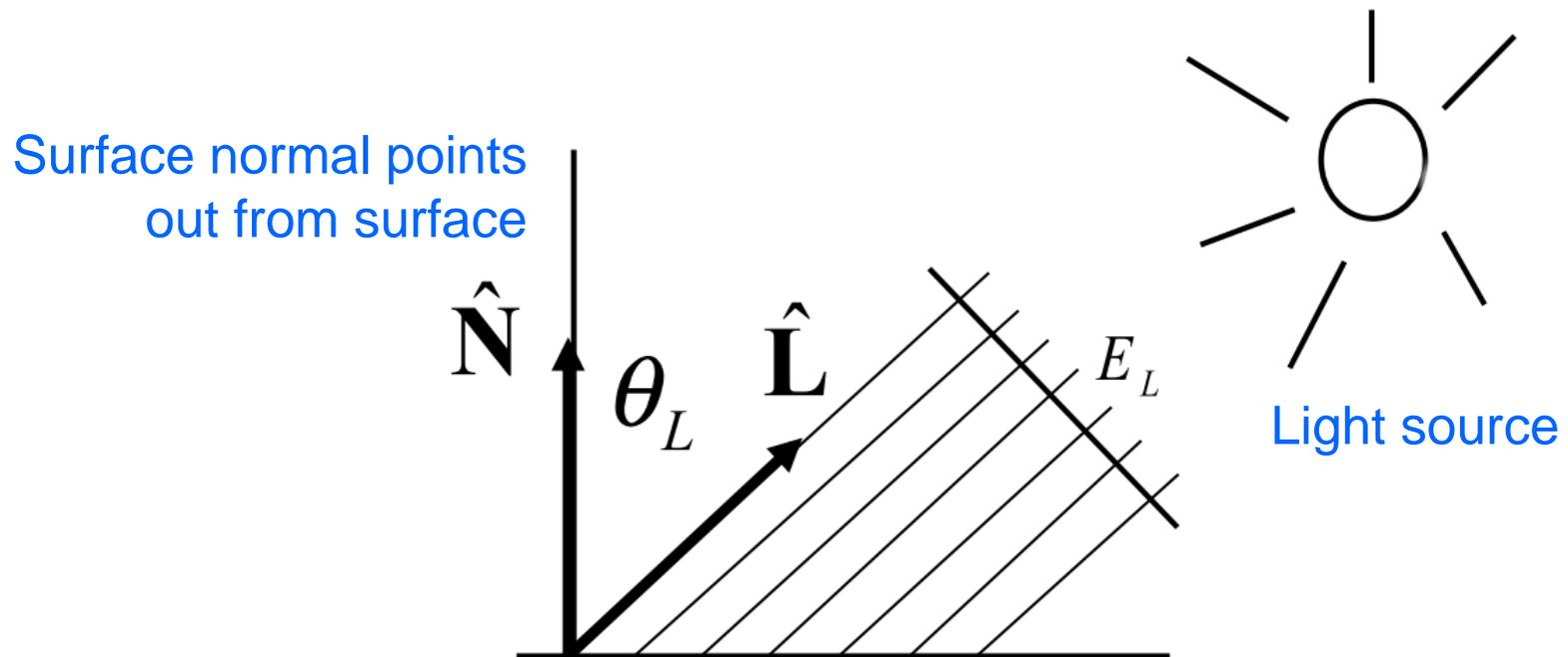
Spherical reflectance

$$\hat{\mathbf{R}} + (-\hat{\mathbf{I}}) = 2 \cos \theta_i \hat{\mathbf{N}} = -2(\hat{\mathbf{I}} \cdot \hat{\mathbf{N}}) \hat{\mathbf{N}}$$

$$\hat{\mathbf{R}} = \hat{\mathbf{I}} - 2(\hat{\mathbf{I}} \cdot \hat{\mathbf{N}}) \hat{\mathbf{N}}$$

Angle of incidence = angle of reflection

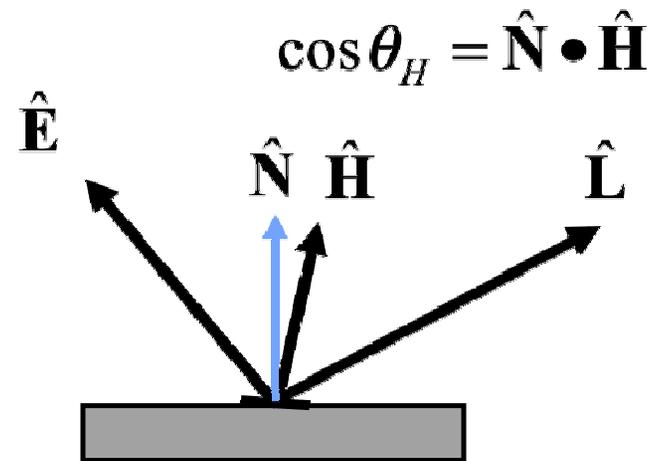
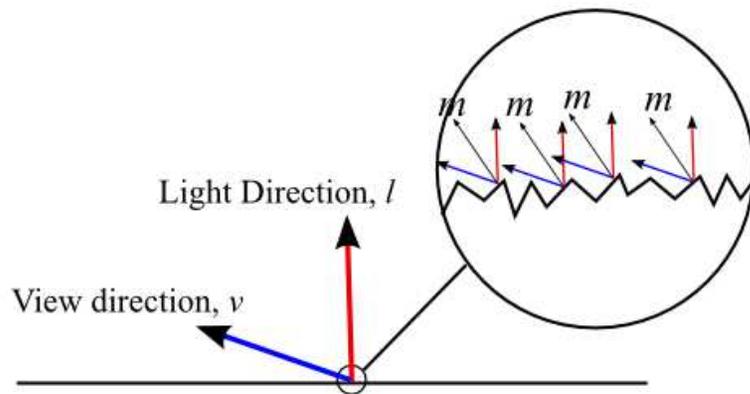
Lambert's Law, or Diffuse Illumination



$$\begin{aligned} \text{diffuse} &= \max(0, \hat{\mathbf{N}} \cdot \hat{\mathbf{L}}) \\ &= \cos \theta_L \end{aligned}$$

Micro-facet Specular

- Think of the surface as have a statistical distribution of facet orientations
 - OpenGL's fixed-function specular model



$$\cos \theta_H = \hat{\mathbf{N}} \cdot \hat{\mathbf{H}}$$

Halfway vector
$$\hat{\mathbf{H}} = \frac{\hat{\mathbf{L}} + \hat{\mathbf{E}}}{|\hat{\mathbf{L}} + \hat{\mathbf{E}}|}$$

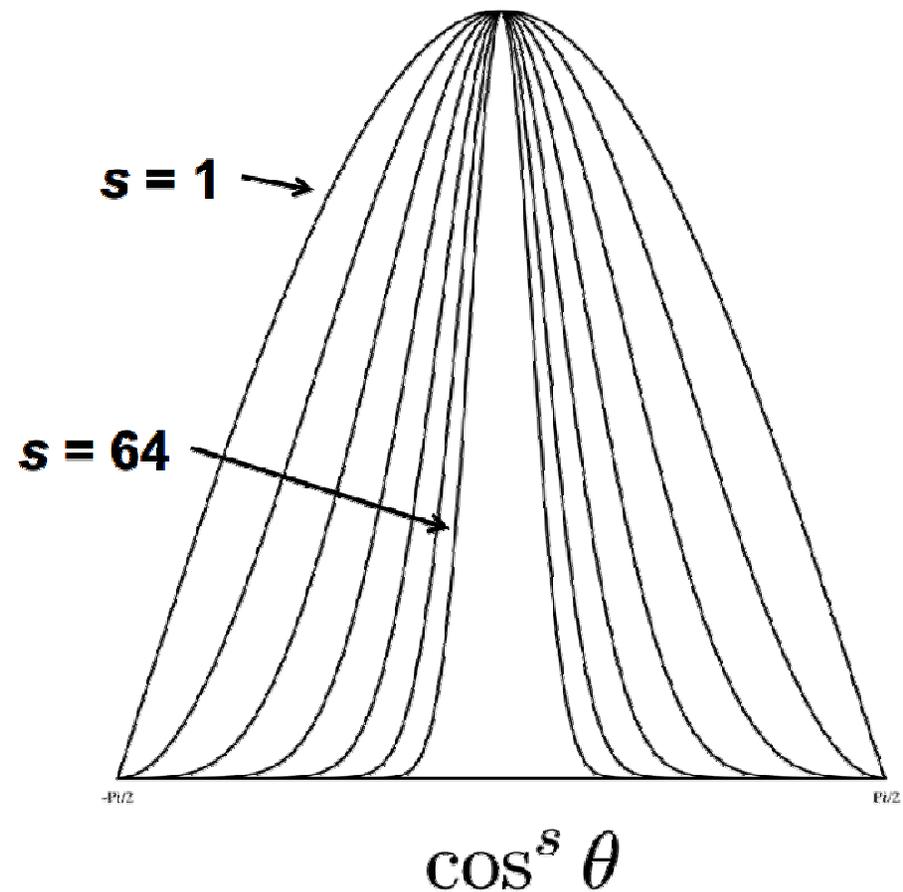
Microfacet distribution

$$(\cos \theta_H)^s = (\hat{\mathbf{N}} \cdot \hat{\mathbf{H}})^s$$

Shininess s

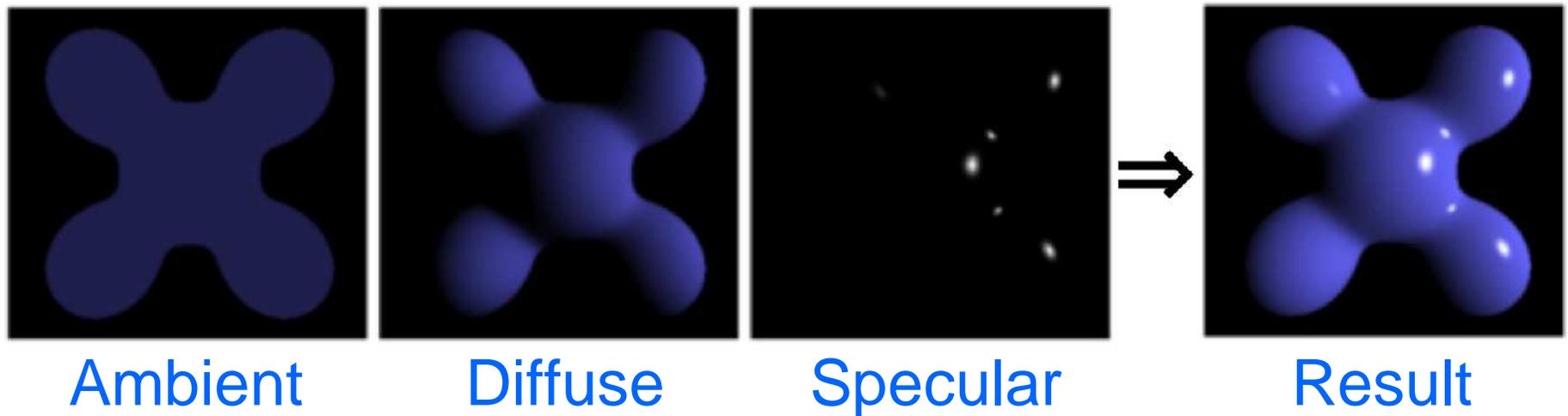
Shininess

- Modeled as exponential fall-off function
 - Larger **shininess** exponent values means a tighter specular highlight
 - Complement of shininess is **roughness**
- Exponential fall-off also use to model
 - Spotlight attenuation
 - Fog effects



Combining Lighting Contributions

- Contributions can be summed
 - Since RGB color components are additive
 - Illumination color modulated by material color



Summing Multiple Lights

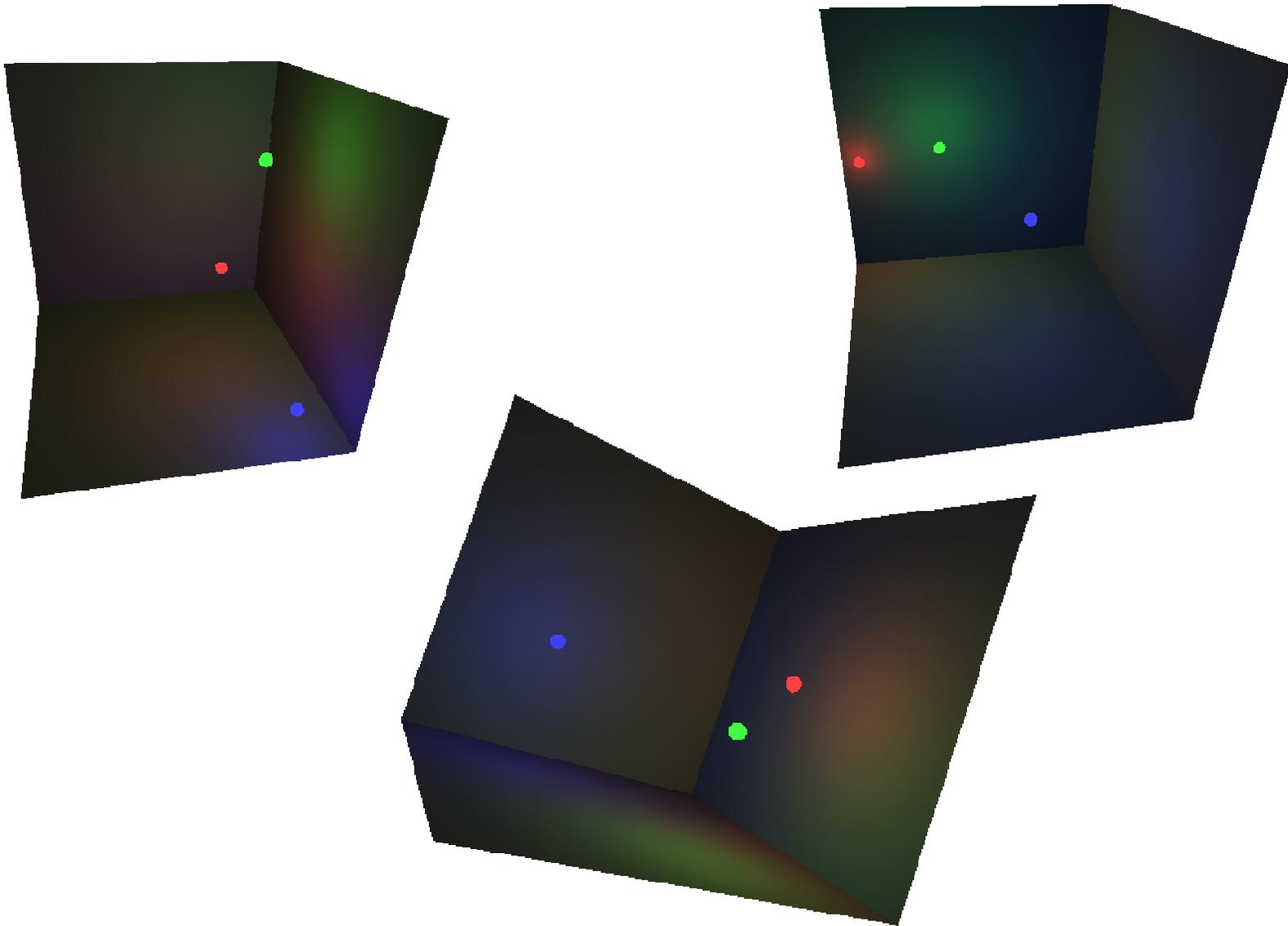
- For each light source and each color component, the Blinn or Phong model can be written (without the distance terms) as

$$I = k_d I_d \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^\alpha + k_a I_a \quad \text{Phong}$$

$$I = k_d I_d \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{h} \cdot \mathbf{n})^\alpha + k_a I_a \quad \text{Blinn}$$

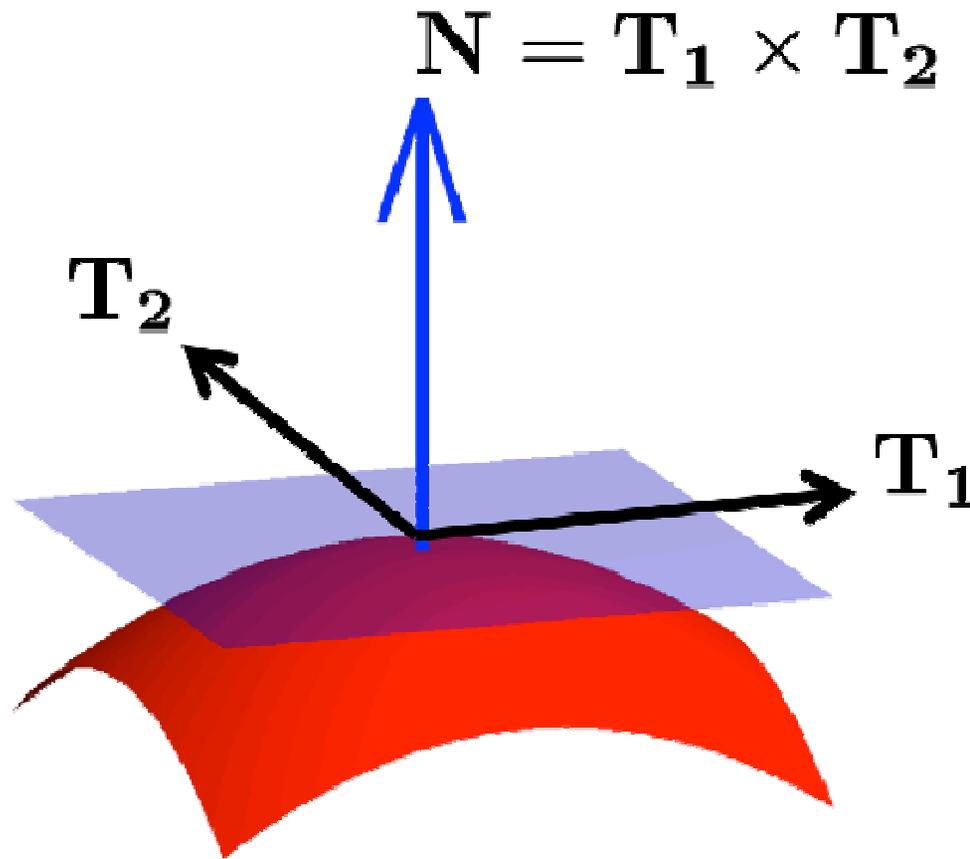
- For each color component, we add contributions from all sources

Multiple Animated Lights



Surface Normal

- Surface normal = cross product of surface gradients



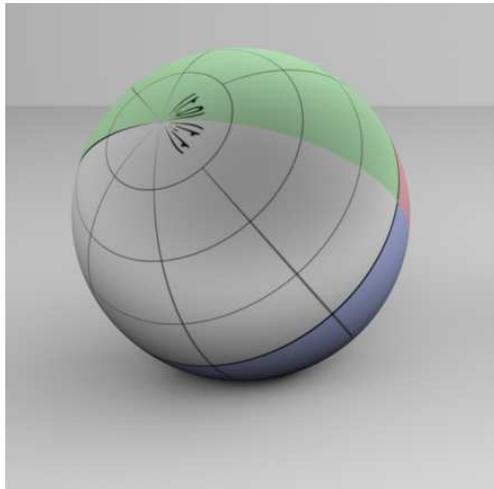
$$(x, y, z) = F(u, v)$$

$$\mathbf{T}_1 = \frac{\partial F}{\partial u}$$

$$\mathbf{T}_2 = \frac{\partial F}{\partial v}$$

Example: Normals for a Sphere

- Parameterize sphere with latitude and longitude



$$x = \sin \theta \cos \phi$$

$$y = \sin \theta \sin \phi$$

$$z = \cos \theta$$

$$\vec{\mathbf{P}} = (x, y, z)$$

$$\vec{\mathbf{N}} = \frac{\partial \vec{\mathbf{P}}}{\partial \theta} \times \frac{\partial \vec{\mathbf{P}}}{\partial \phi}$$

$$\frac{\partial \vec{\mathbf{P}}}{\partial \theta} = (\cos \theta \cos \phi, \cos \theta \sin \phi, -\sin \theta)$$

$$\frac{\partial \vec{\mathbf{P}}}{\partial \phi} = (-\sin \theta \sin \phi, \sin \theta \cos \phi, 0)$$

Problem: What happens at the poles?

When θ is +90 or -90 degrees

Transforming Normals

- **Recall:** vertex and vector transformation

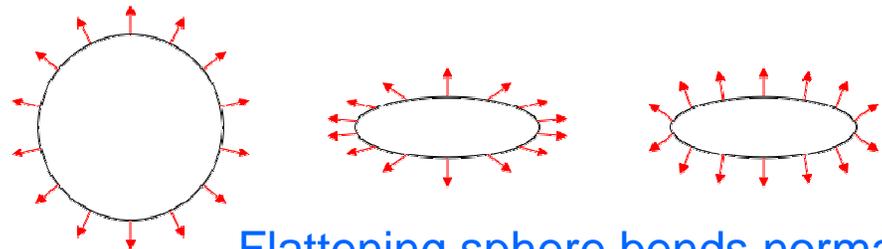
- Multiply transform matrix by column vector
- $v' = M v$

- How to transforming normals correctly

- **Desire:** $n' \cdot v' = n \cdot v = n^T v$

- **Rationale:** dot product should be invariant under coordinate space transformation

- $I = M^{-1} M$
- $n^T I v = n^T M^{-1} M v$
- $n' = n^T M^{-1}, v' = M v$
- $n' = M^{-T} n$

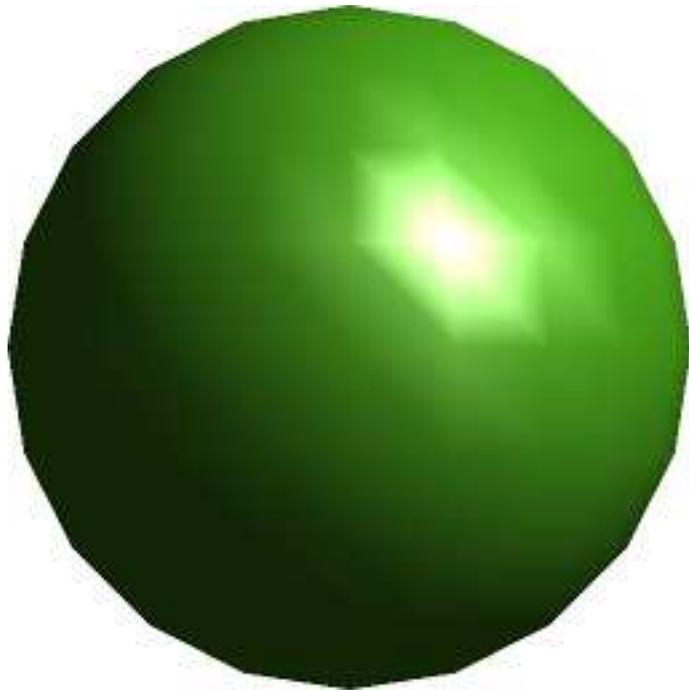


Flattening sphere bends normals up

- **Implication:** column normal transformed by inverse transpose of transform matrix used for vectors

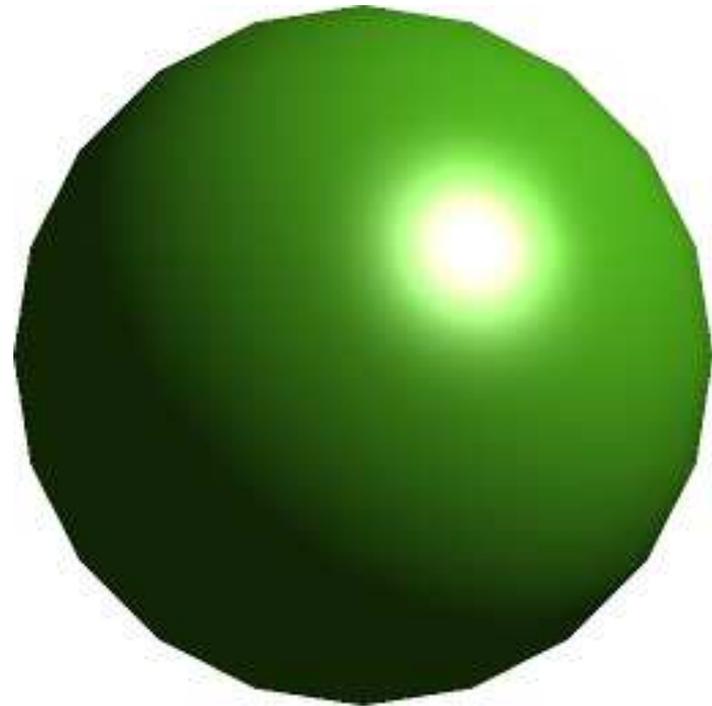
Frequency of Lighting Computations

■ Per-vertex



Specular highlight “wobbles”
under animation

■ Per-fragment



Specular highlight has
stable structure

Fixed-function OpenGL

Per-vertex Lighting

- OpenGL API includes lighting model
 - Operates at the per-vertex level
 - Implements a Blinn-Phong lighting model
- Example usage
 - `glEnable(GL_LIGHTING);`
 - `glEnable(GL_LIGHT0);`
 - `glLightfv(GL_LIGHT0, GL_POSITION, position);`
 - `glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);`
 - `glLightfv(GL_LIGHT0, GL_SPECULAR, specular);`
- Relies on
 - Per-vertex normal value sent with `glNormal3f`
 - And per-vertex material properties sent with `glMaterialfv`

OpenGL Light Parameters

- Per-light state set with `glLightfv` command

Token Name	Meaning	Type	Initial Value
<code>GL_AMBIENT</code>	Ambient light color	RGBA, 4 floats	(0,0,0,1)
<code>GL_DIFFUSE</code>	Diffuse light color	RGBA, 4 floats	(1,1,1,1) for light 0, (0,0,0,1) for other lights
<code>GL_SPECULAR</code>	Specular light color	RGBA, 4 floats	(1,1,1,1) for light 0, (0,0,0,1) for other lights
<code>GL_POSITION</code>	Light location	Homogeneous position, 4 floats	(0,0,1,0), +Z axis direction
<code>GL_SPOT_DIRECTION</code>	Spot light direction	Direction vector, 3 floats	(0,0,-1)
<code>GL_SPOT_EXPONENT</code>	Spot light exponential fall-off	Non-negative scalar exponent	0
<code>GL_SPOT_CUTOFF</code>	Angle of spot light cut off	Angle in degrees	180, uniform distribution
<code>GL_CONSTANT_ATTENUATION</code>	Light attenuation inverse constant	Scalar float	1
<code>GL_LINEAR_ATTENUATION</code>	Inverse linear attenuation	Scalar float	0
<code>GL_QUADRATIC_ATTENUATION</code>	Inverse quadratic attenuation	Scalar float	0

OpenGL Material Parameters

- Per-light state set with `glLightfv` command
 - Surface has `GL_FRONT` and `GL_BACK` versions of these materials
 - For two-sided lighting, depending on how polygon faces

Token Name	Meaning	Type	Initial Value
<code>GL_AMBIENT</code>	Ambient material color	RGBA, 4 floats	(0.2, 0.2, 0.2, 1)
<code>GL_DIFFUSE</code>	Diffuse material color	RGBA, 4 floats	(0.8, 0.8, 0.8, 1)
<code>GL_SPECULAR</code>	Specular material color	RGBA, 4 floats	(0, 0, 0, 1)
<code>GL_EMISSION</code>	Emissive material color	RGBA, 4 floats	(0, 0, 0, 1)
<code>GL_SHININESS</code>	Specular exponent of material	Scalar float	0

OpenGL Fixed-function Lighting Equation

surface
result
color

$$\begin{aligned}
 \mathbf{c} &= \mathbf{e}_{cm} \leftarrow \text{emissive} \\
 &+ \mathbf{a}_{cm} * \mathbf{a}_{cs} \leftarrow \text{global ambient} \\
 &+ \sum_{i=0}^{n-1} (att_i)(spot_i) [\mathbf{a}_{cm} * \mathbf{a}_{cli} \leftarrow \text{per-light ambient} \\
 &\quad + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} \\
 &\quad + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}]
 \end{aligned}$$

for each light source

diffuse

specular

$$f_i = \begin{cases} 1, & \mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli} \neq 0, \\ 0, & \text{otherwise,} \end{cases}$$

diffuse squashes specular

OpenGL Lighting Equation Terms

half-angle

$$\mathbf{h}_i = \begin{cases} \overrightarrow{\mathbf{VP}}_{pli} + \overrightarrow{\mathbf{VP}}_e, & v_{bs} = \text{TRUE}, \\ \overrightarrow{\mathbf{VP}}_{pli} + (0 \ 0 \ 1)^T, & v_{bs} = \text{FALSE}, \end{cases}$$

local viewer assumption

infinite viewer assumption

distance
attenuation

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\mathbf{VP}_{pli}\| + k_{2i}\|\mathbf{VP}_{pli}\|^2}, & \text{if } \mathbf{P}_{pli}'s \ w \neq 0, \\ 1.0, & \text{otherwise,} \end{cases}$$

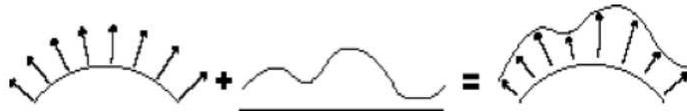
inverse square fall-off

spotlight
attenuation

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli})^{s_{rli}}, & c_{rli} \neq 180.0, \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli} \geq \cos(c_{rli}), \\ 0.0, & c_{rli} \neq 180.0, \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli} < \cos(c_{rli}), \\ 1.0, & c_{rli} = 180.0. \end{cases}$$

Displacement and Bump Mapping

- Use surface offsets stored in texture
 - Perturb or displace the surface
 - Shade on the resulting surface normals



$$\mathbf{P}(u, v)$$

$$\mathbf{S}(u, v) = \frac{\partial \mathbf{P}(u, v)}{\partial u} \quad \mathbf{T}(u, v) = \frac{\partial \mathbf{P}(u, v)}{\partial v}$$

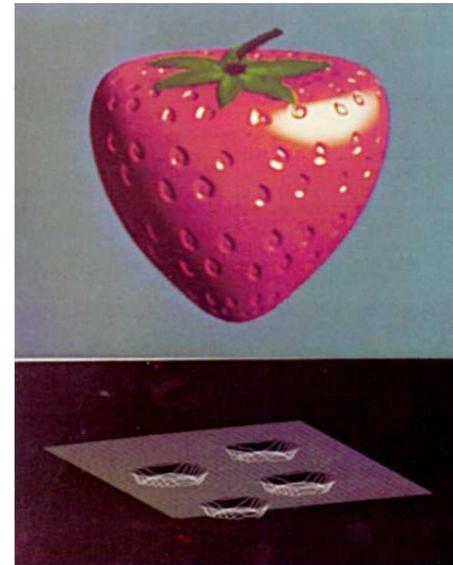
$$\mathbf{N}(u, v) = \mathbf{S} \times \mathbf{T}$$

■ Displacement

$$\mathbf{P}'(u, v) = \mathbf{P}(u, v) + h(u, v)\mathbf{N}(u, v)$$

■ Perturbed normal

$$\begin{aligned} \mathbf{N}'(u, v) &= \mathbf{P}'_u \times \mathbf{P}'_v \\ &= \mathbf{N} + h_u(\mathbf{T} \times \mathbf{N}) + h_v(\mathbf{S} \times \mathbf{N}) \end{aligned}$$



From Blinn 1976

Normal Mapping

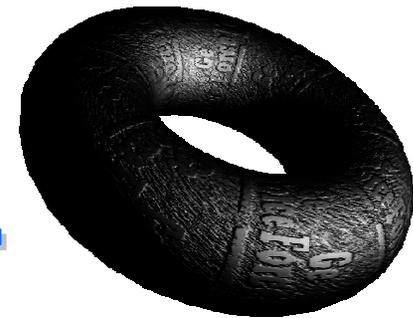
- Bump mapping via a normal map texture
 - Instead of a height field
 - The normal map is generated from the height field



diffuse



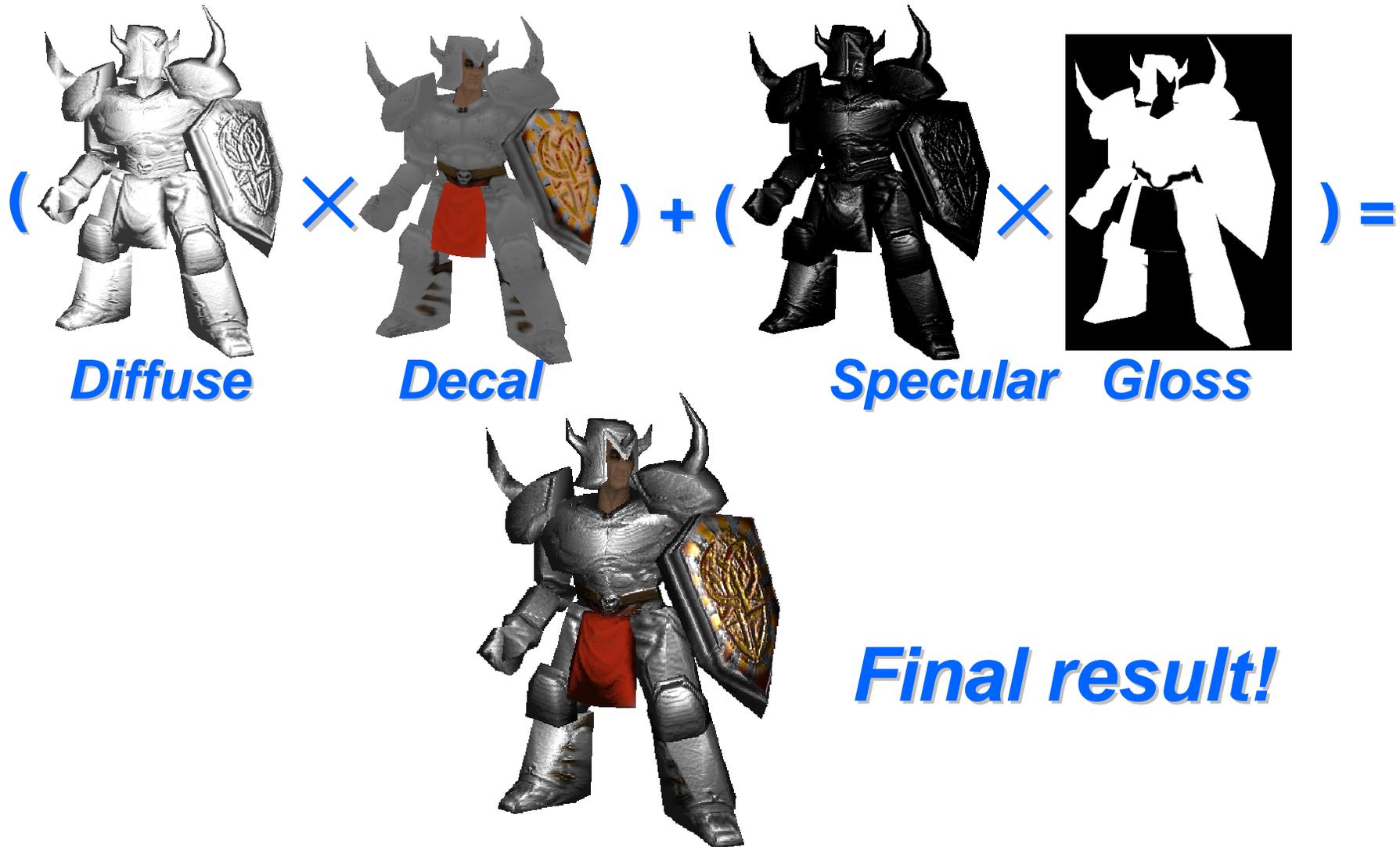
decal



specular



Combination of Normal Map Lighting + Texturing



Environment Mapping as Omni-directional Lighting



**Access texture
by surface reflection
vector**

Other Fancy Lighting Effects



Caustic patterns

Light is “focused” as it refracts through interfaces



Diffraction

Small slits act as a diffraction grating, separating wavelengths

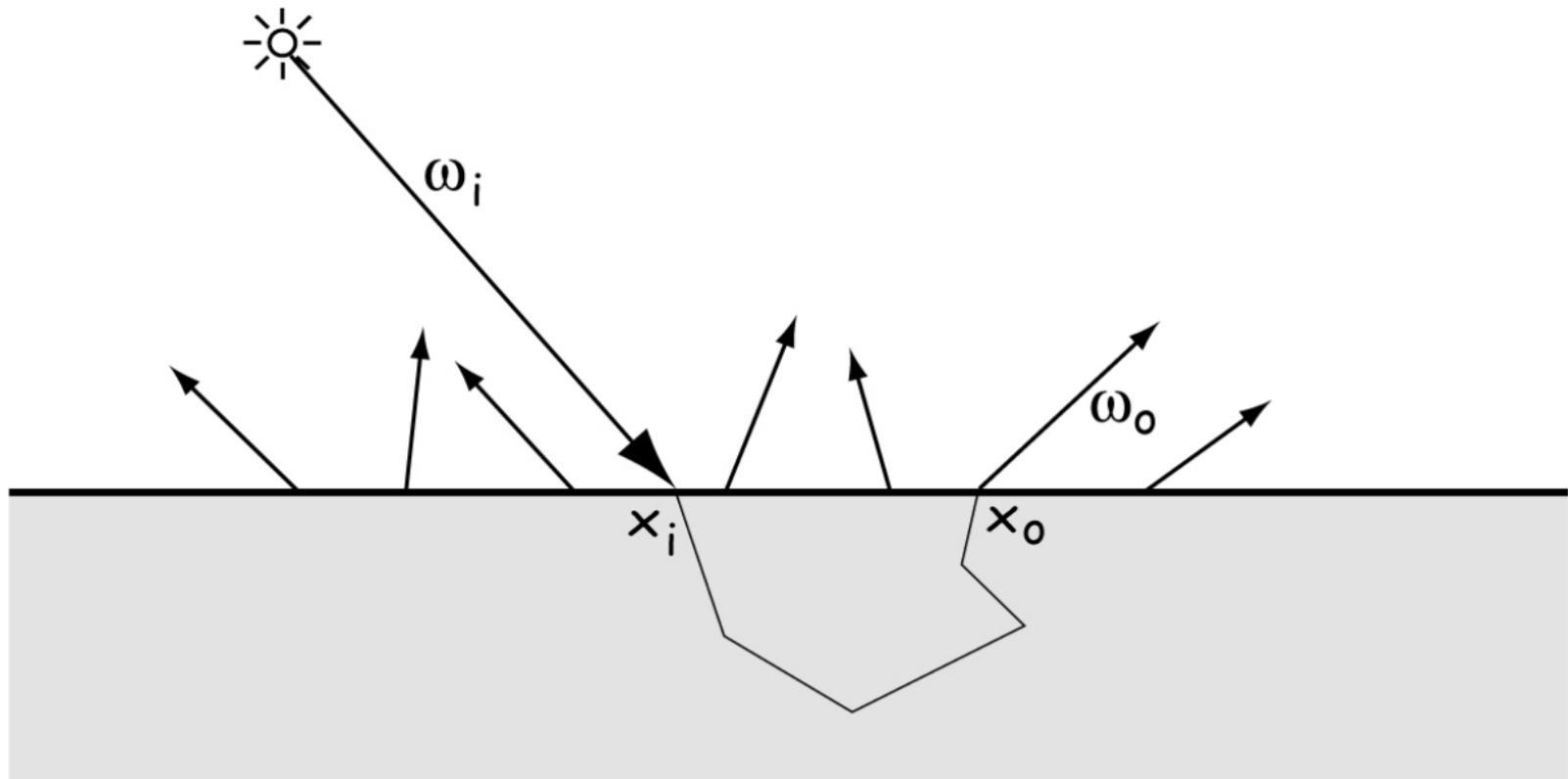
Subsurface Scattering

- Light bounces around within an object
 - Translucency



Paths of Scattered Photons

- Trace photons statistically as they bounce and eventually exit the material



Accurate Skin Needs to Model Subsurface Scattering



Skin with subsurface scattering
Looks natural

Credit: Stephen Stahlberg

Without
Looks lifeless



Take Away Information

- Rendering equation
 - Good theory for modeling realistic lighting
 - Basis for approximation
- Interactive computer graphics typically uses a local lighting model
 - Simple but effective
 - Models lighting as ambient, diffuse, and specular interactions
- Lighting can be implemented in programmable shaders
 - At either the per-vertex or per-fragment level

Next Lecture

- Programmable shading
 - *Combine texture, lighting, and shading under application control*
 - *In a high-level language*

- As usual, expect a short quiz on today's lecture

- Assignments
 - Be sure to schedule your Project 1 demos with Randy
 - Reading
 - Chapter 7, 388-403 on Programmable Shading
 - Homework #4 to be assigned next class, due March 6th
 - Project #2 coming soon
 - Will be due after Spring Break
 - **Remember: Midterm in-class on March 8**

Credits and Appreciation

- Some images from
 - Dr. Pat Hanrahan, Stanford University
 - Ed Angel & David Shriner, textbook
 - Stephen Stahlberg
 - Christian Miller, UT CS