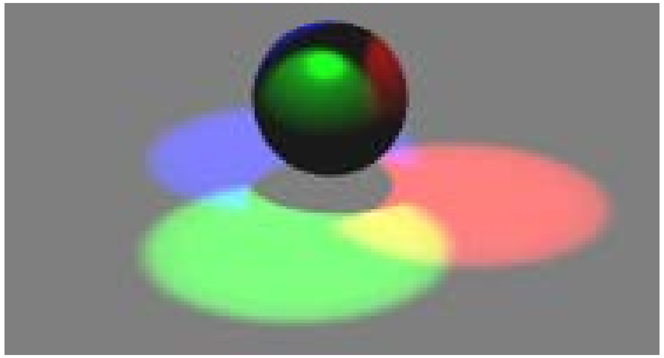
CS380: Computer Graphics
Illumination and Shading

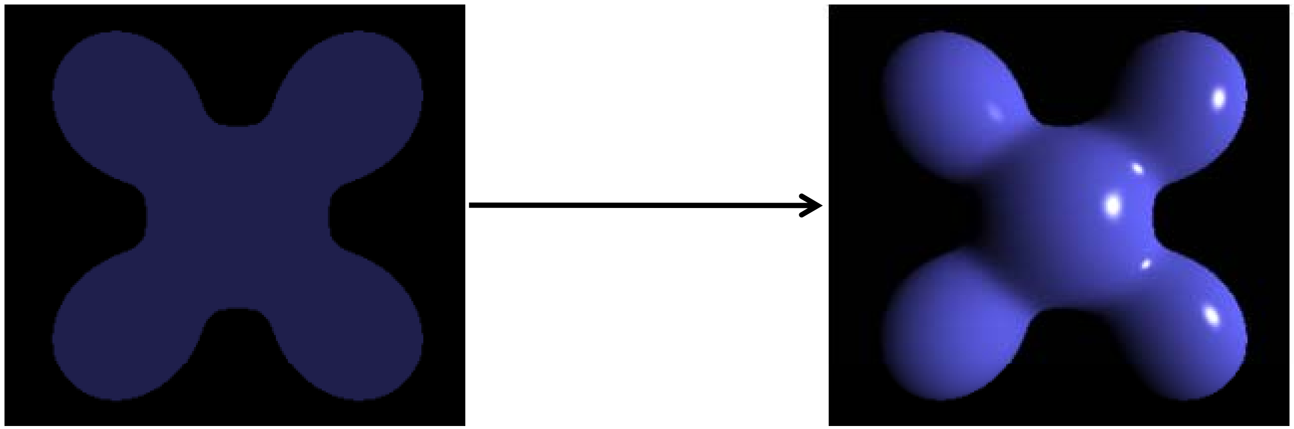
Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/CG/

Course Objectives

- **Know how to consider lights during rendering models**
  - **Light sources**
  - **Illumination models**
  - **Shading**
  - **Local vs. global illumination**

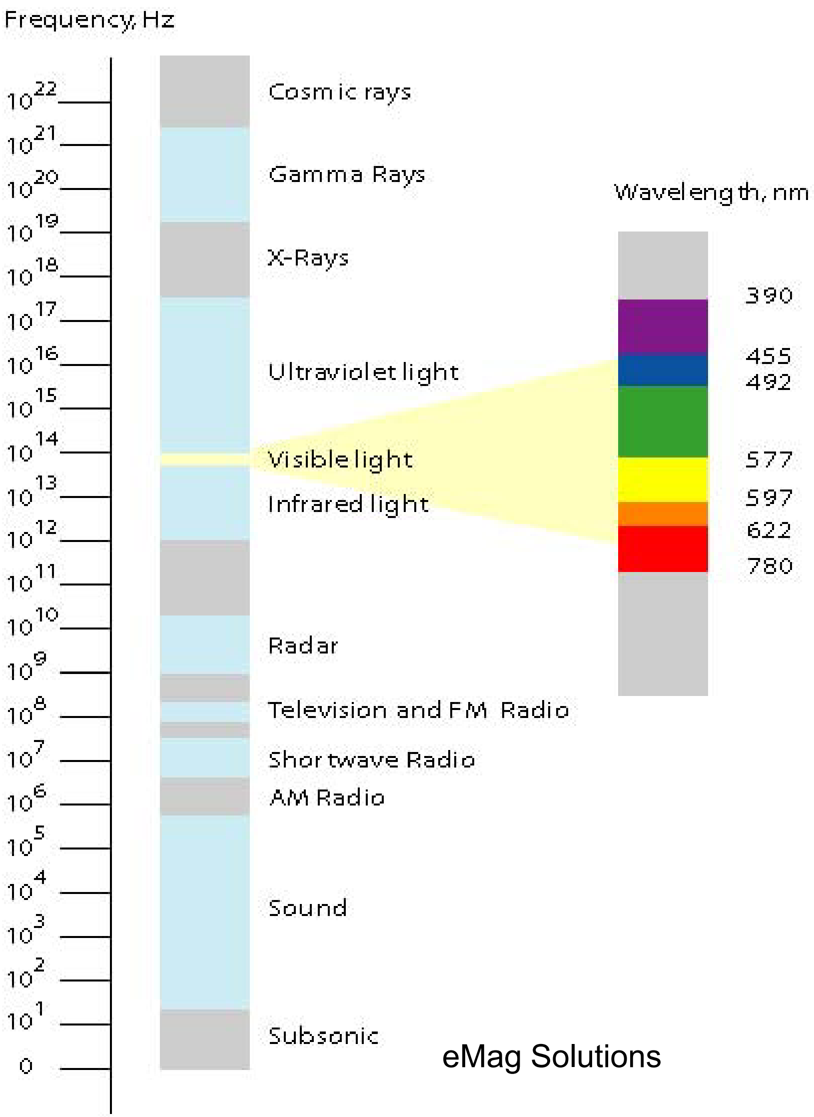




Question: How Can We See Objects?

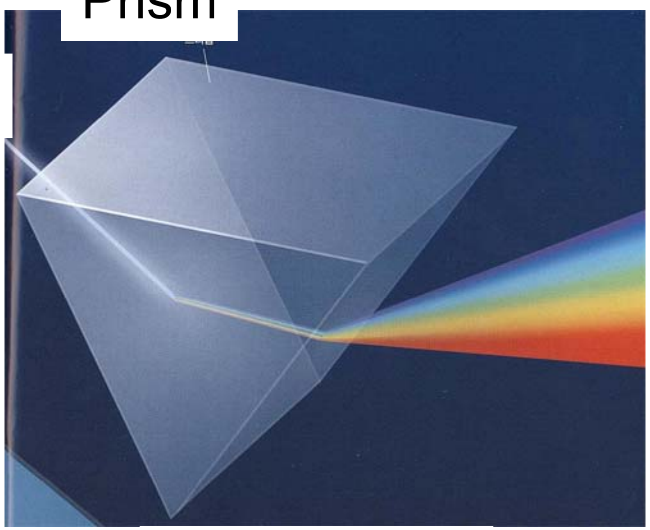
- **Emission and *reflection*!**

Question: How Can We See Objects?



eMag Solutions

Light
(sub-class of electromagnetic waves)

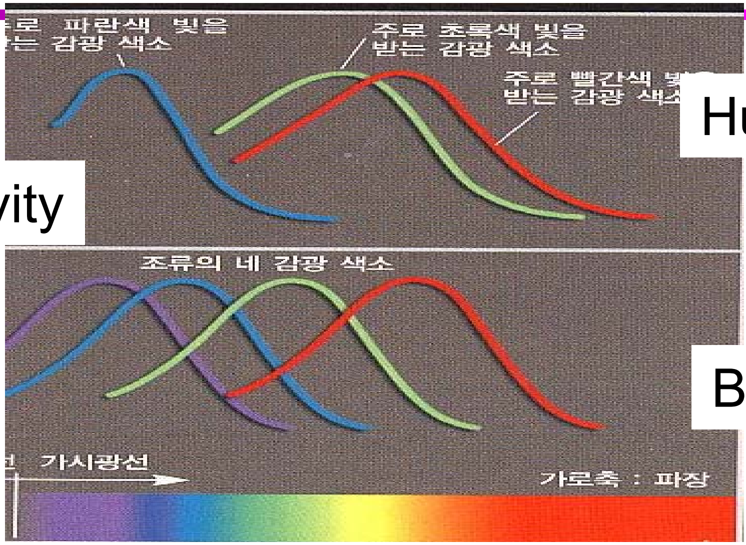


From Newton magazine

KAIST

Question: How Can We See Objects?

Sensitivity



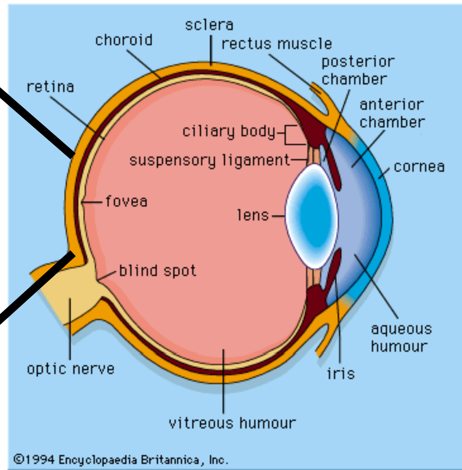
Human

Birds

Light
(sub-class of
electromagnetic waves)

Rod and cone

From Newton magazine

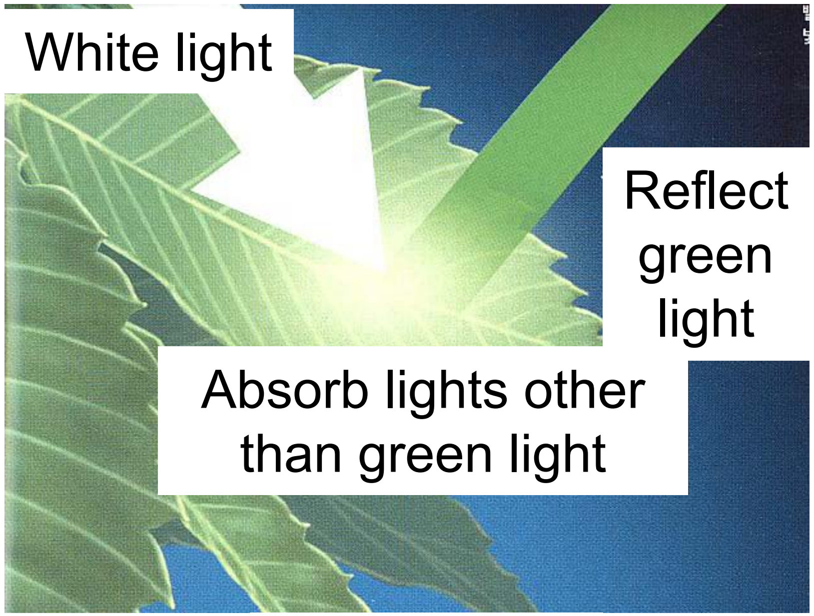


Eye

Question: How Can We See Objects?

- **Emission and *reflection*!**

Light
(sub-class of electromagnetic waves)



From Newton magazine

Eye

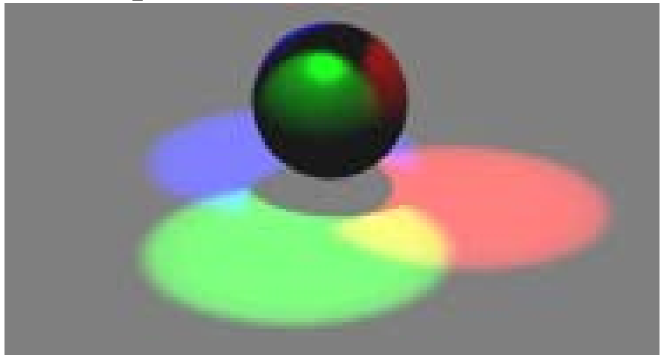
- **How about mirrors and white papers?**

Illumination Models

- Physically-based
  - Models based on the actual physics of light's interactions with matter
- Empirical
  - Simple formulations that approximate observed phenomenon

Two Components of Illumination

- **Light sources:**
  - **Emittance spectrum (color)**
  - **Geometry (position and direction)**
  - **Directional attenuation**



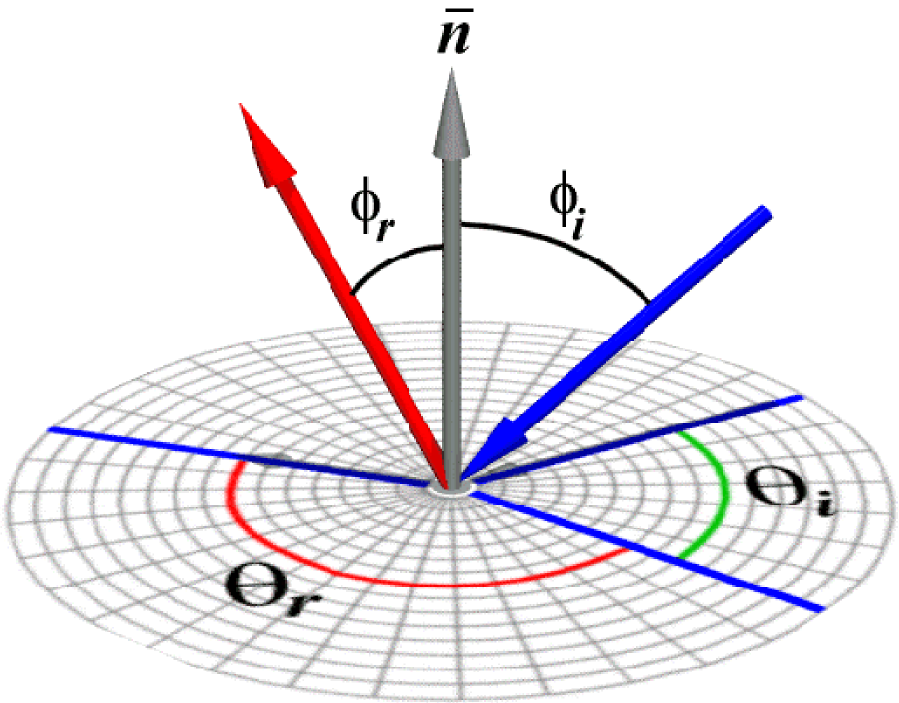
- **Surface properties:**
  - **Reflectance spectrum (color)**
  - **Geometry (position, orientation, and micro-structure)**
  - **Absorption**

KAIST

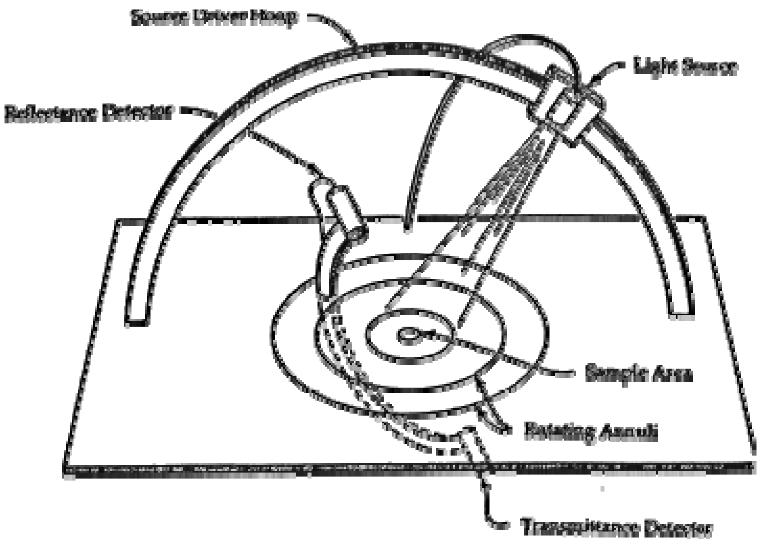
Bi-Directional Reflectance Distribution Function (BRDF)

- **Describes the transport of irradiance to radiance**

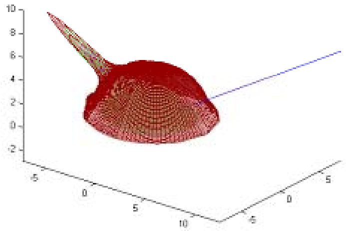
$\rho(\theta_r, \phi_r, \theta_i, \phi_i)$



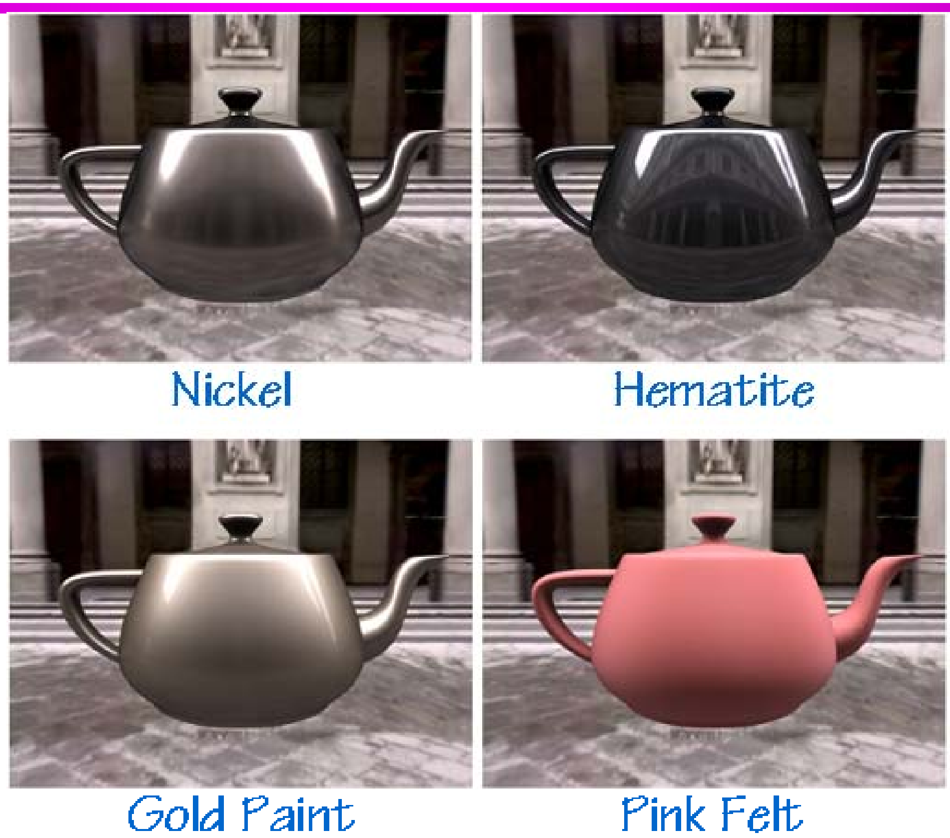
Measuring BRDFs



- **Goniophotometer**
  - **One 4D measurement at a time (slow)**



How to use BRDF Data?



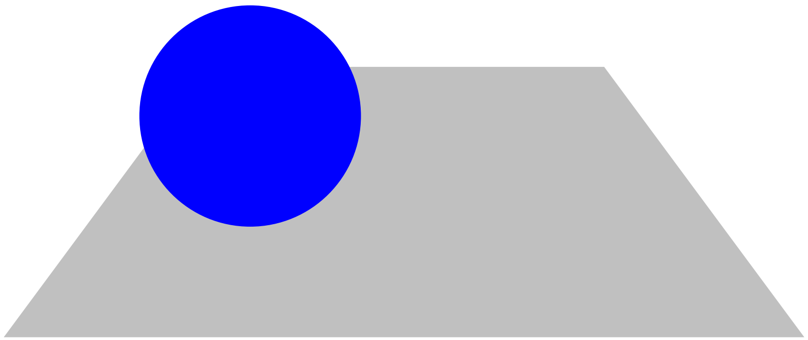
One can make direct use of acquired BRDFs in a renderer

Two Components of Illumination

- Simplifications used by most computer graphics systems:
  - Compute only direct illumination from the emitters to the reflectors of the scene
  - Ignore the geometry of light emitters, and consider only the geometry of reflectors

Ambient Light Source

- A simple <u>hack</u> for indirect illumination
  - Incoming ambient illumination ($I_{i,a}$) is constant for all surfaces in the scene
  - Reflected ambient illumination ($I_{r,a}$) depends only on the surface's ambient reflection coefficient (k_a) and not its position or orientation $I_{r,a} = k_a I_{i,a}$
  - These quantities typically specified as (R, G, B) triples

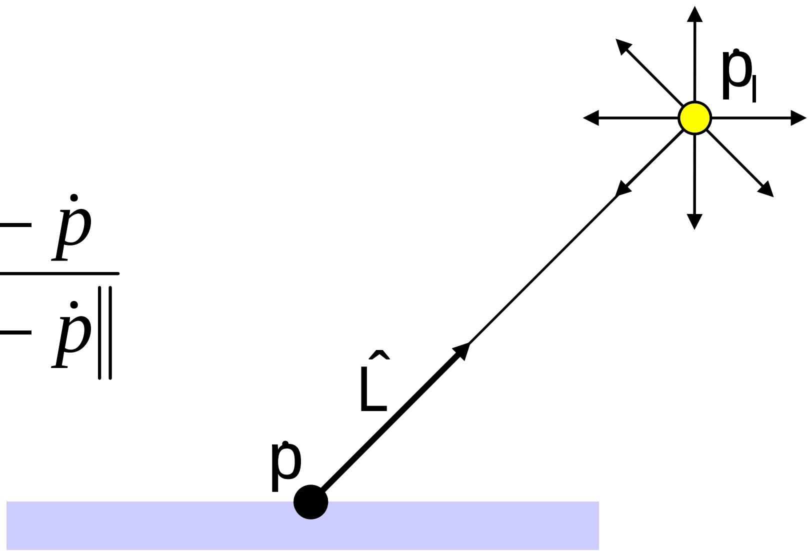


KAIST

Point Light Sources

- **Point light sources emit rays from a single point**
  - **Simple approximation to a local light source such as a light bulb**

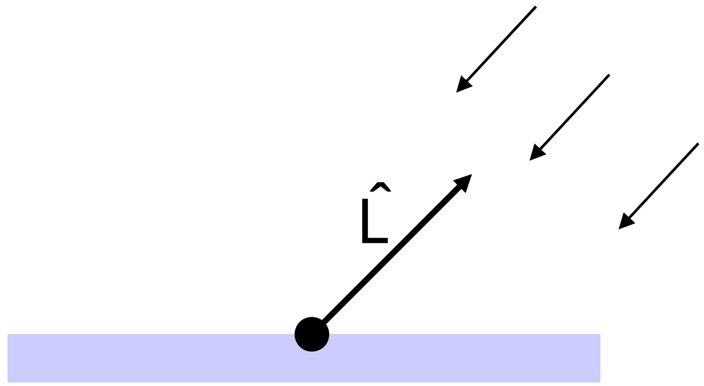
$$\hat{L} = \frac{\dot{p}_l - \dot{p}}{\left\| \dot{p}_l - \dot{p} \right\|}$$



- **The direction to the light changes across the surface**

Directional Light Sources

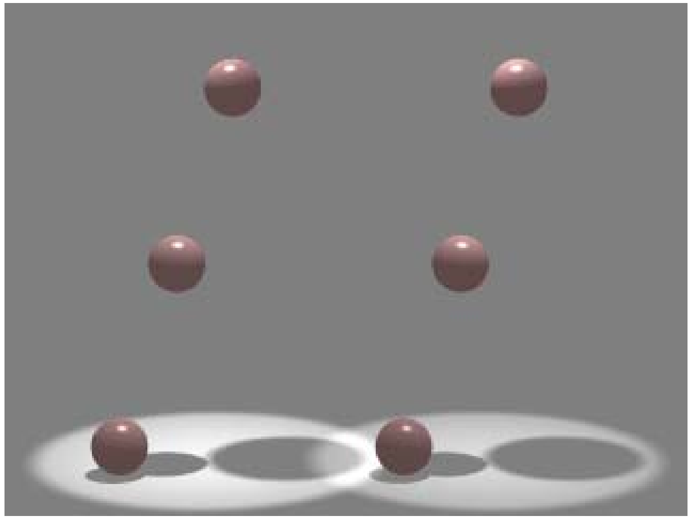
- Light rays are parallel and have no origin
  - Can be considered as a point light at infinity
  - A good approximation for sunlight

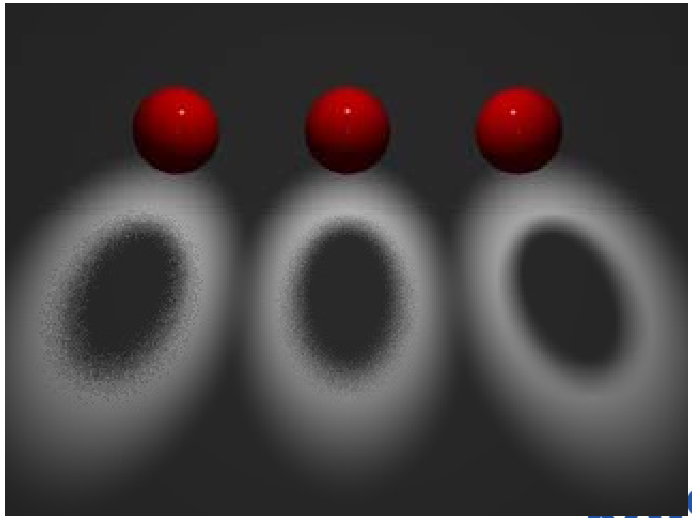


- The direction to the light source is constant over the surface
- How can we specify point and directional lights?

Other Light Sources

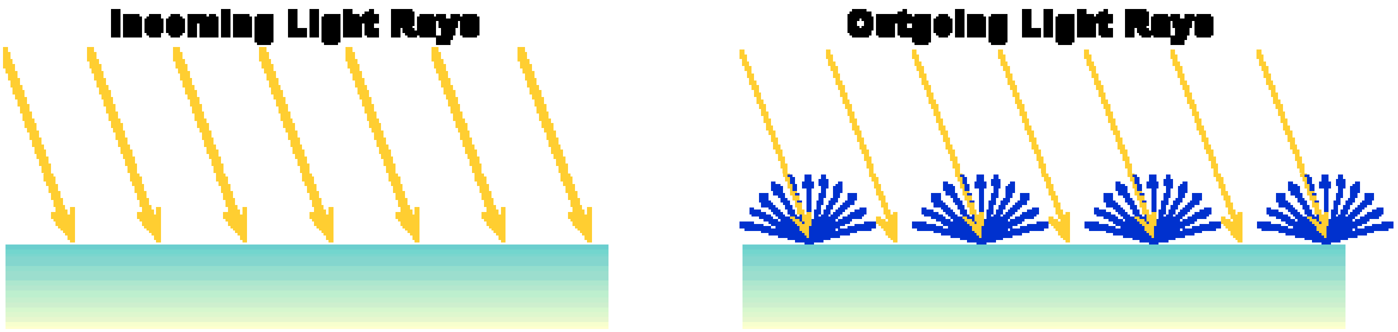
- Spotlights
  - Point source whose intensity falls off away from a given direction
- Area light sources
  - Occupies a 2D area (e.g. a polygon or a disk)
  - Generates *soft* shadows





Ideal Diffuse Reflection

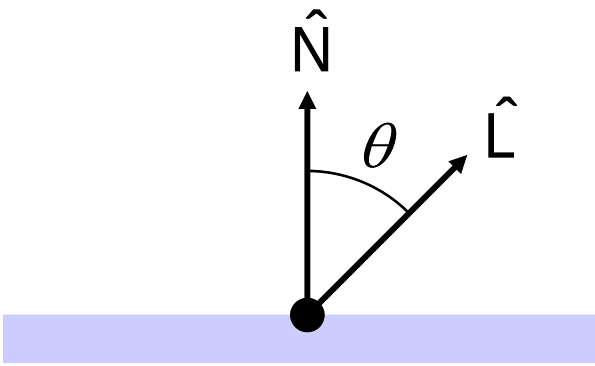
- Ideal diffuse reflectors (e.g., chalk)
  - Reflect uniformly over the hemisphere
  - Reflection is view-independent
  - Very rough at the microscopic level
- Follow Lambert's cosine law

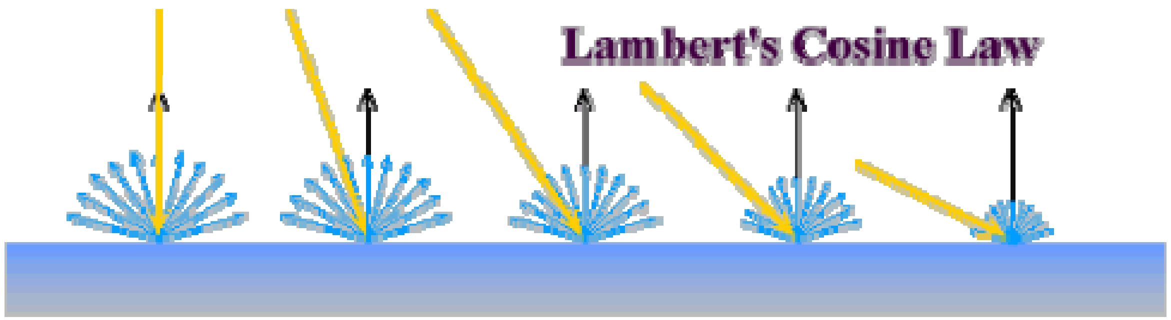


Lambert's Cosine Law

- **The reflected energy from a small surface area from illumination arriving from direction $\hat{L}$ is proportional to the cosine of the angle between $\hat{L}$ and the surface normal**

$$I_r \approx I_i \cos\theta$$
$$\approx I_i(\hat{N} \bullet \hat{L})$$



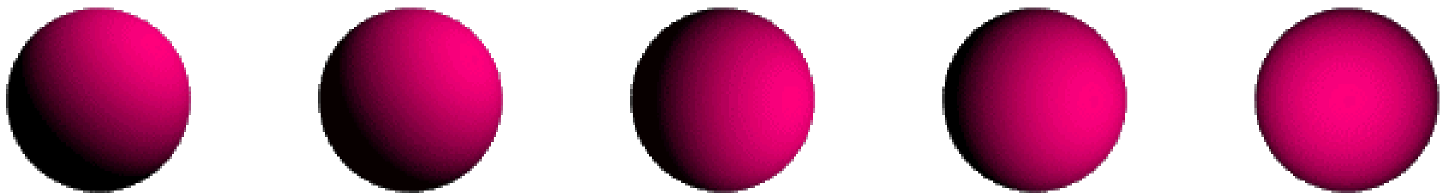


Computing Diffuse Reflection

- **Constant of proportionality depends on surface properties**

$$I_{r,d} = k_d I_i (\hat{N} \bullet \hat{L})$$

  - **The constant k_d specifies how much of the incident light I_i is diffusely reflected**



Diffuse reflection for varying light directions

  - **When $(\hat{N} \cdot \hat{L}) < 0$ the incident light is blocked by the surface itself and the diffuse reflection is 0**

Specular Reflection

- **Specular reflectors have a bright, view dependent highlight**
  - **E.g., polished metal, glossy car finish, a mirror**
  - **At the microscopic level a specular reflecting surface is very smooth**
  - **Specular reflection obeys Snell's law**

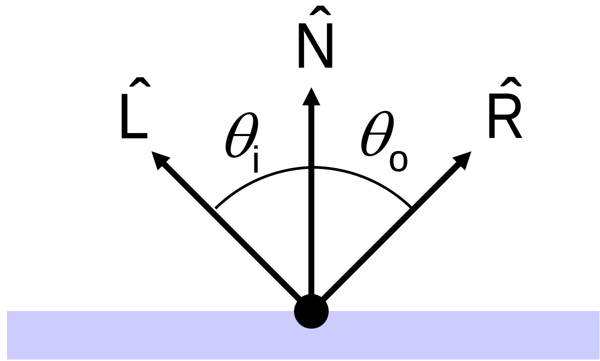
Image source: astochimp.com and wiki

KAIST

Snell's Law

- The relationship between the angles of the incoming and reflected rays with the normal is given by:

$$n_i \sin\theta_i = n_o \sin\theta_o$$



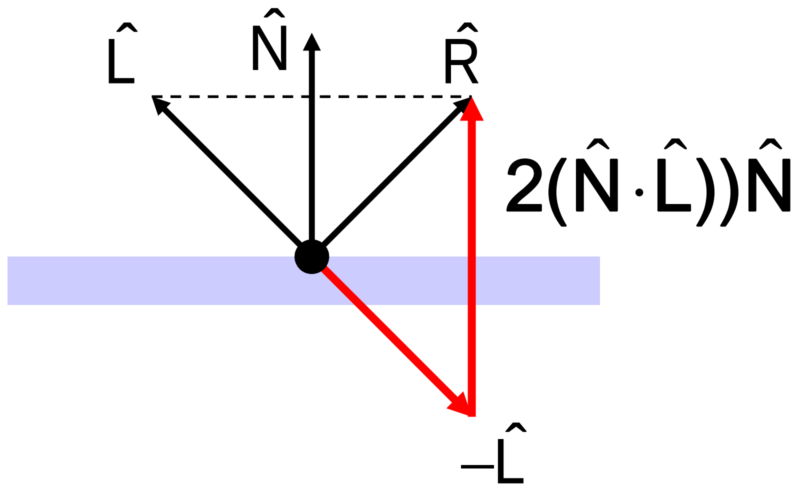
  - n_i and n_o are the indices of refraction for the incoming and outgoing ray, respectively
  - Reflection is a special case where $n_i = n_o$ so $\theta_o = \theta_i$
  - The incoming ray, the surface normal, and the reflected ray all lie in a common plane

Computing the Reflection Vector

- **The vector R can be computed from the incoming light direction and the surface normal as shown below:**

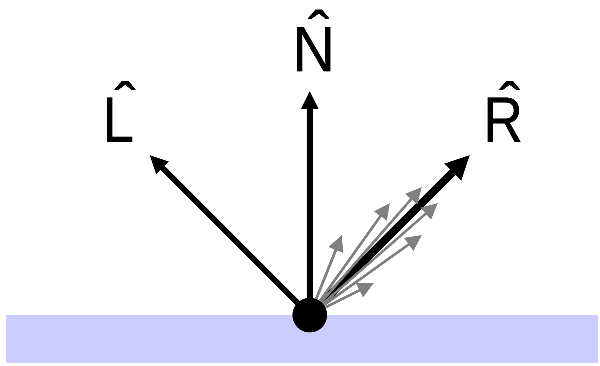
$$\hat{R} = (2(\hat{N} \cdot \hat{L}))\hat{N} - \hat{L}$$

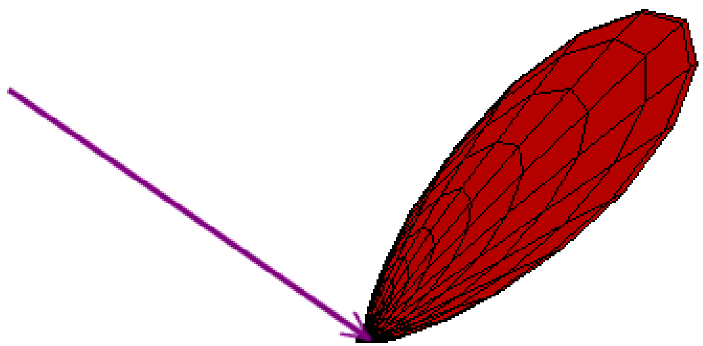
- **How?**



Non-Ideal Reflectors

- **Snell's law applies only to *ideal* specular reflectors**
  - **Roughness of surfaces causes highlight to "spread out"**
  - **Empirical models try to simulate the appearance of this effect, without trying to capture the physics of it**

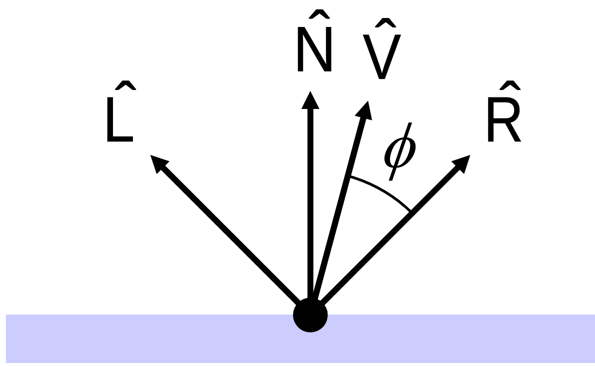




Phong Illumination

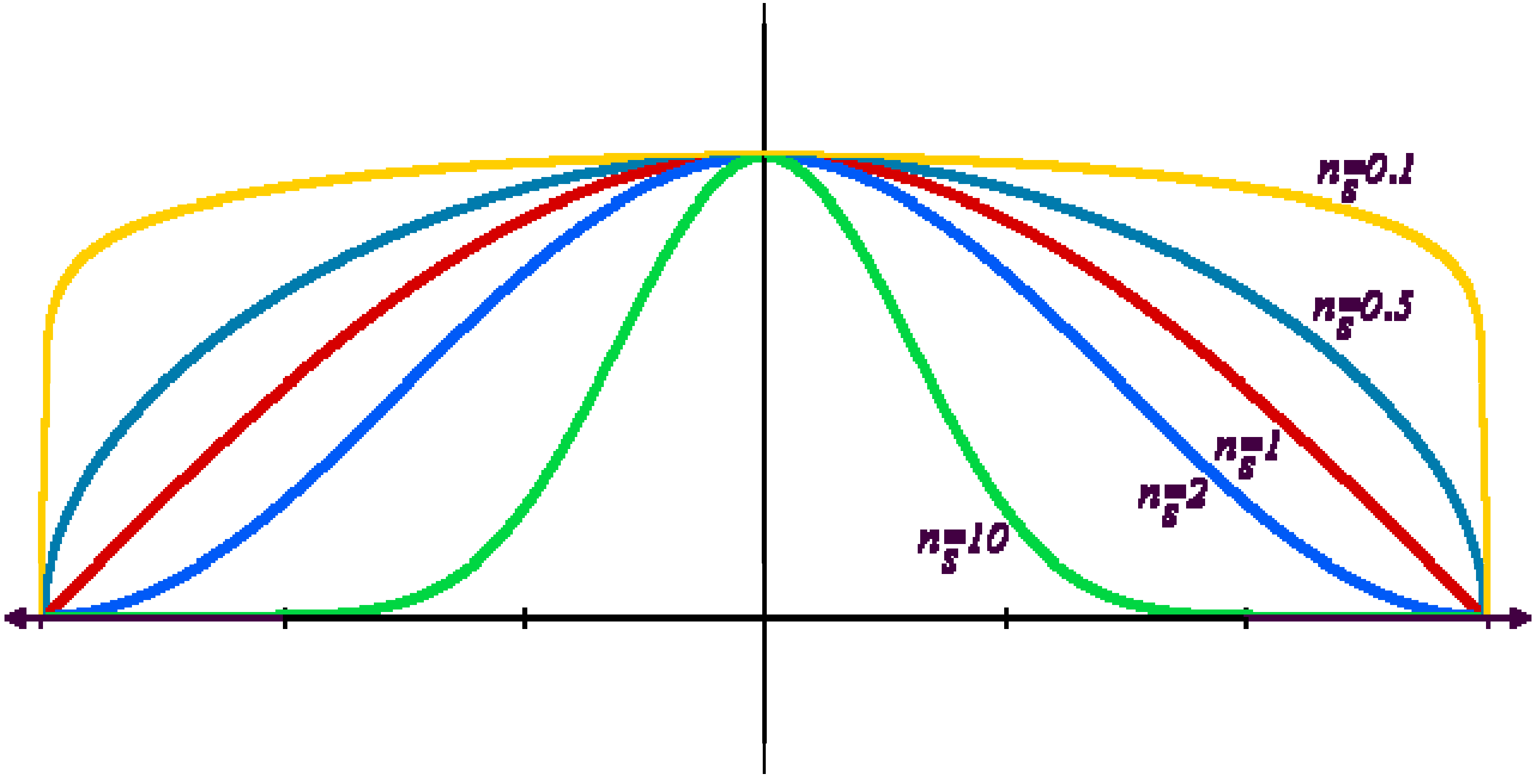
- **One of the most commonly used illumination models in computer graphics**
  - **Empirical model and does not have no physical basis**

$$I_r = k_s I_i (\cos\phi)^{n_s}$$
$$= k_s I_i (\hat{V} \bullet \hat{R})^{n_s}$$



- **$(\hat{V})$ is the direction to the viewer**
  - **$(\hat{V} \bullet \hat{R})$ is clamped to [0,1]**
  - **The specular exponent n_s controls how quickly the highlight falls off**

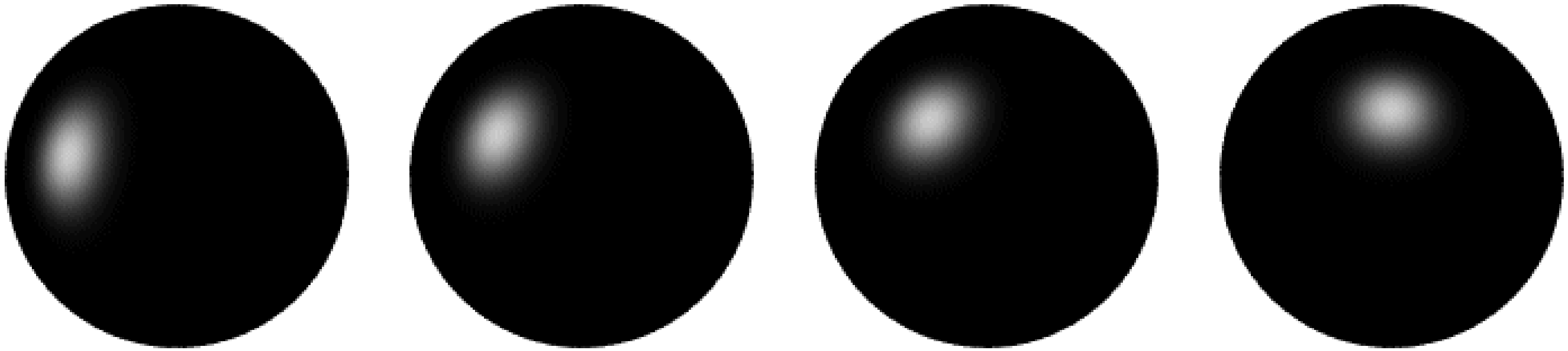
Effect of Specular Exponent



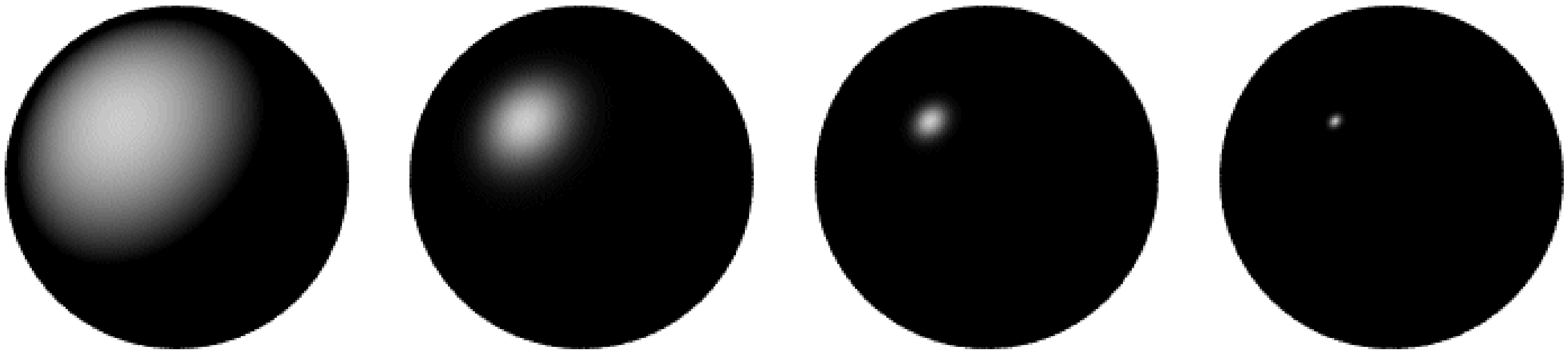
- **How the shape of the highlight changes with varying n_s**

KAIST

Examples of Phong



varying light direction



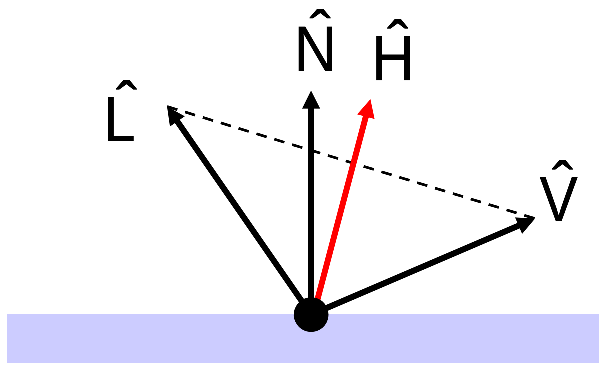
varying specular exponent

Blinn & Torrance Variation

- **Jim Blinn introduced another approach for computing Phong-like illumination based on the work of Ken Torrance:**

$$\hat{H} = \frac{\hat{L} + \hat{V}}{\left|\hat{L} + \hat{V}\right|}$$

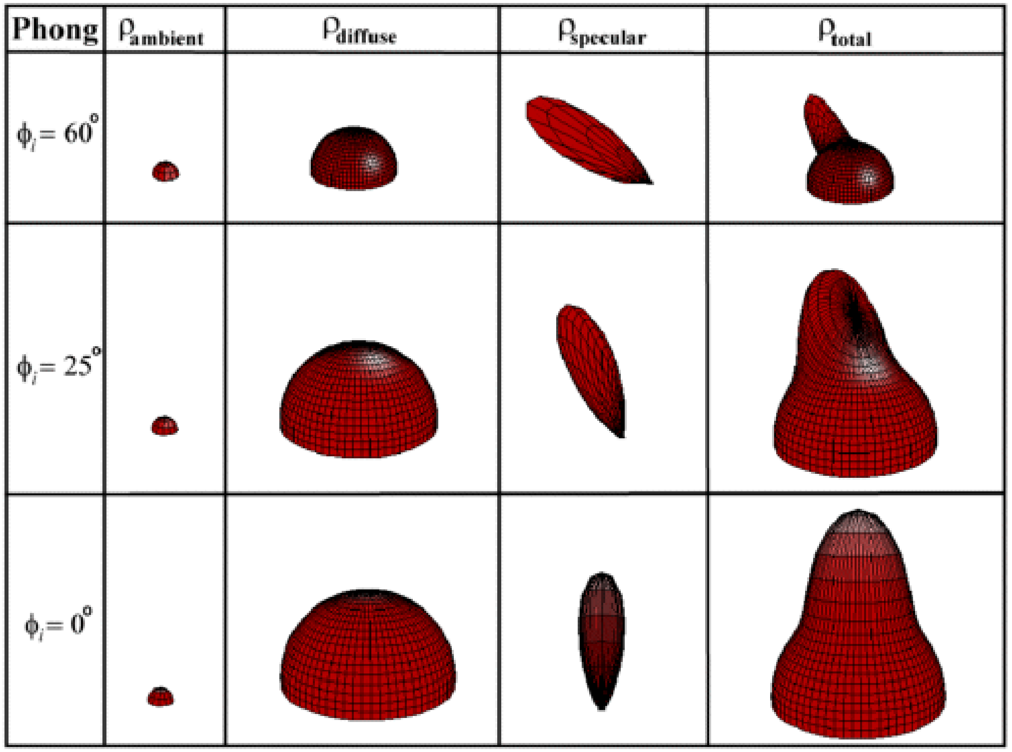
$$I_{r,s} = k_s I_i (\hat{N} \cdot \hat{H})^{n_s}$$



- $\hat{H}$ **is the half-way vector that bisects the light and viewer directions**

Putting it All Together

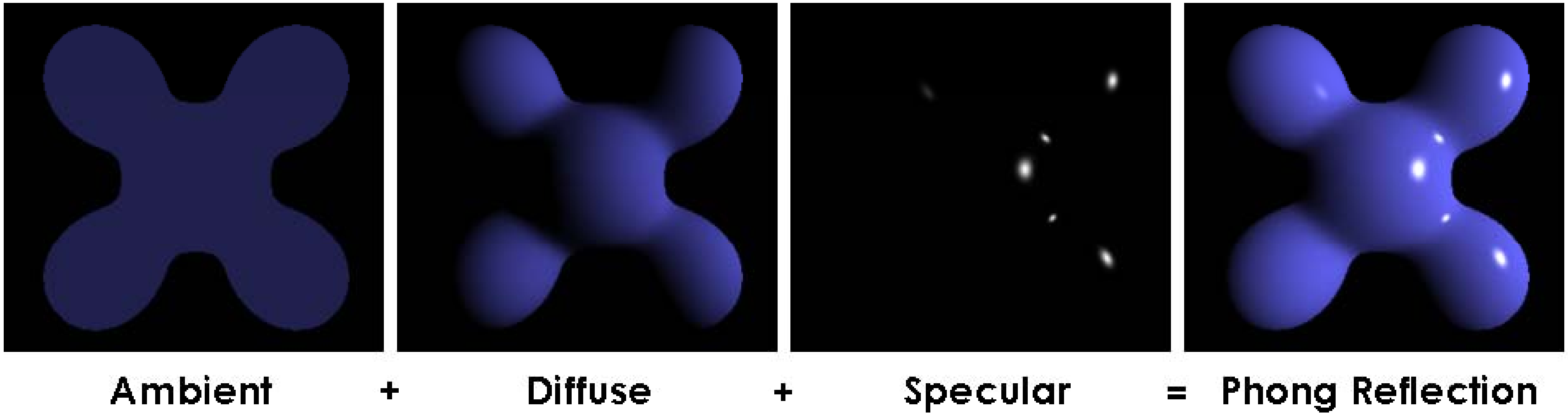
$$I_r = \sum_{j=1}^{numLights} \left(k_a^j I_a^j + k_d^j I_d^j \max((\hat{N} \bullet \hat{L}_j),0) + k_s^j I_s^j \max((\hat{V} \bullet \hat{R}),0)\right)^{n_s}$$



KAIST

Putting it All Together

$$I_r = \sum_{j=1}^{numLights} \left(k_a^j I_a^j + k_d^j I_d^j \max((\hat{N} \bullet \hat{L}_j),0) + k_s^j I_s^j \max((\hat{V} \bullet \hat{R}),0)\right)^{n_s}$$



From Wikipedia

KAIST

OpenGL's Illumination Model

$$I_r = \sum_{j=1}^{numLights} (k_a^j I_a^j + k_d^j I_d^j \max((\hat{N} \bullet \hat{L}_j),0) + k_s^j I_s^j \max((\hat{V} \bullet \hat{R}),0))^{n_s}$$

- **Problems with empirical models:**
  - **What are the coefficients for copper?**
  - **What are k_a, k_s, and n_s?**
    Are they measurable quantities?
  - **Is my picture accurate? Is energy conserved?**

KAIST

Lights in OpenGL

- **Light positions are specified in homogeneous coordinates**
  - **They are transformed by the current modelview matrix**
- **Directional light sources have w=0**

Lights in OpenGL

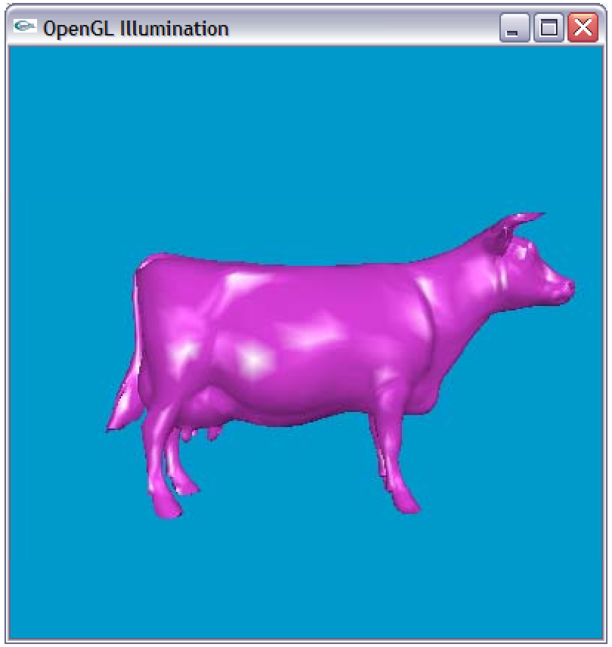
```
# define a directional light
lightDirection = [1, 1, 1, 0]
glLightfv(GL_LIGHT0, GL_POSITION, lightDirection)
glEnable(GL_LIGHT0)

# define a point light
lightPoint = [100, 100, 100, 1]
glLightfv(GL_LIGHT1, GL_POSITION, lightPoint)
glEnable(GL_LIGHT1)

# set up light's color
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientIntensity)
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseIntensity)
glLightfv(GL_LIGHT0, GL_SPECULAR, specularIntensity)
```

OpenGL Surface Properties

```
glMaterialfv(GL_FRONT, GL_AMBIENT, ambientColor)
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseColor)
glMaterialfv(GL_FRONT, GL_SPECULAR, specularColor)
glMaterialfv(GL_FRONT, GL_SHININESS, nshininess)
```



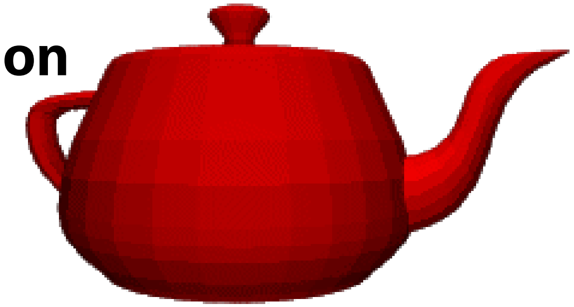
Illumination Methods

- Illumination can be expensive
  - Requires computation and normalizing of vectors for multiple light sources

- Compute illumination for faces, vertices, or pixels with increasing realism and computing overhead
  - Correspond to flat, Gouraud, and Phong shading respectively

Flat Shading

- **The simplest shading method**
  - **Applies only one illumination calculation per face**



- **Illumination usually computed at the centroid of the face:**

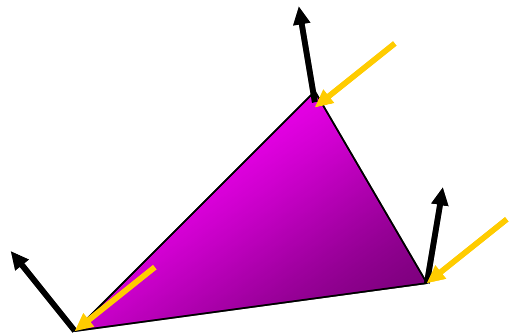
$$centroid = \frac{1}{n}\sum_{i=1}^{n} p_i$$

- **Issues?**

KAIST

Gouraud Shading

- **Performs the illumination model on vertices and interpolates the intensity of the remaining points on the surface**

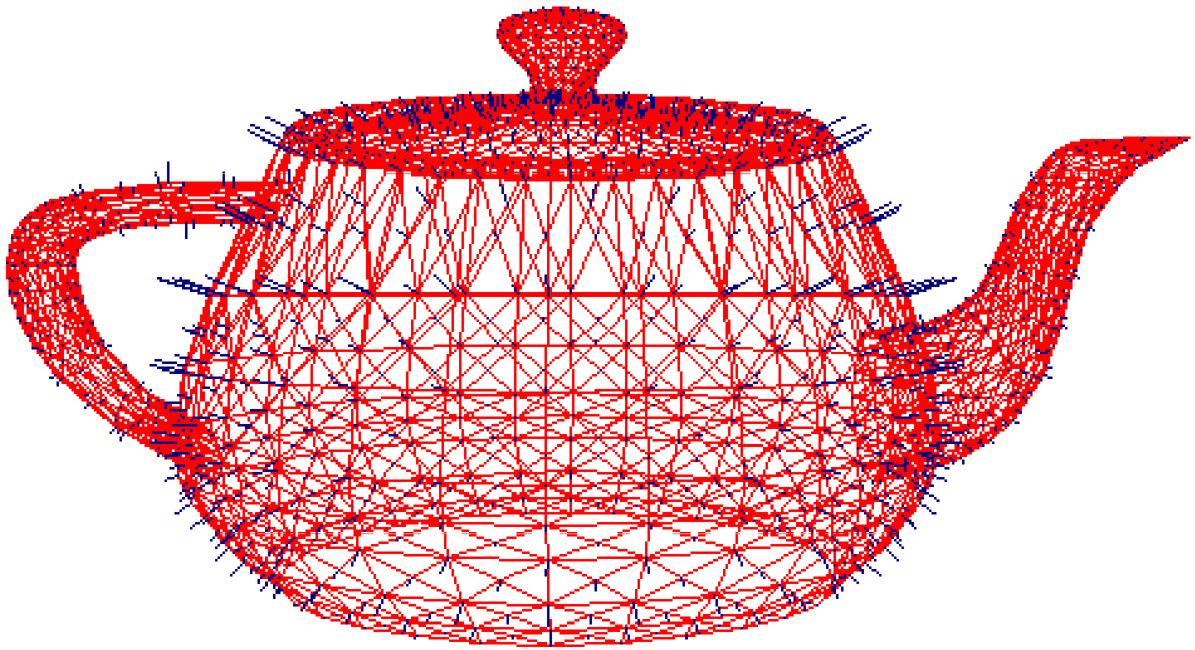


Notice that facet artifacts are still visible

Vertex Normals

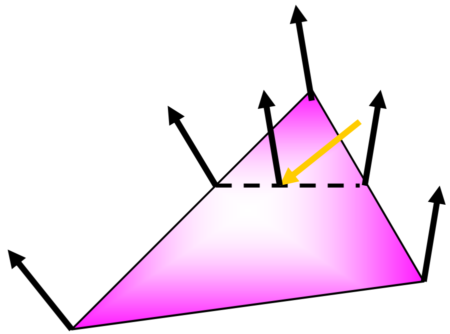
If vertex normals are not provided they can often be approximated by averaging the normals of the facets which share the vertex

$$\vec{n}_v = \sum_{i=1}^{k} \vec{n}_{face,i}$$



Phong Shading

- **Surface normal is linearly interpolated across polygonal facets, and the illumination model is applied at every point**
  - **Not to be confused with Phong's illumination model**



- **Phong shading will usually result in a very smooth appearance**
  - **However, evidence of the polygonal model can usually be seen along silhouettes**

Local Illumination

- **Local illumination models compute the colors of points on surfaces by considering only local properties:**
  - **Position of the point**
  - **Surface properties**
  - **Properties of any light affect it**
- **No other objects in the scene are considered neither as light blockers nor as reflectors**
- **Typical of immediate-mode renders, such as OpenGL**

Global Illumination

- **In the real world, light takes indirect paths**
  - **Light reflects off of other materials (possibly multiple objects)**
  - **Light is blocked by other objects**
  - **Light can be scattered**
  - **Light can be focused**
  - **Light can bend**
- **Harder to model**
  - **At each point we must consider not only every light source, but and other point that might have reflected light toward it**

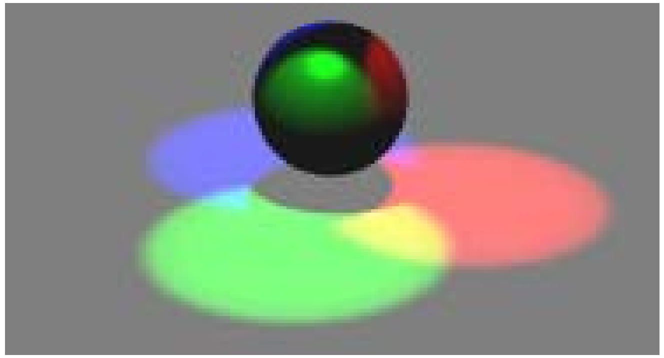
Various Effects using Physically-based Models

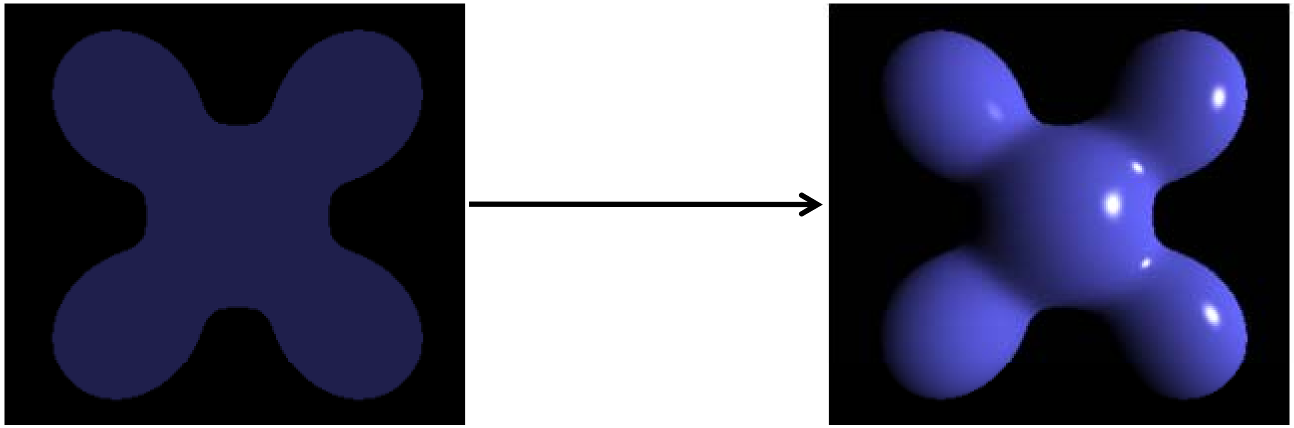
From slides of Pat Hanrahan

- **There are still many open problems to accurately represent various natural materials and efficiently render them**

Course Objectives

- **Know how to consider lights during rendering models**
  - **Light sources**
  - **Illumination models**
  - **Shading**
  - **Local vs. global illumination**





KAIST

Reading Homework

- **Read a chapter of “Texture Mapping”**

Next Time

- **Texture mapping**