CS380: Computer Graphics
Illumination and Shading

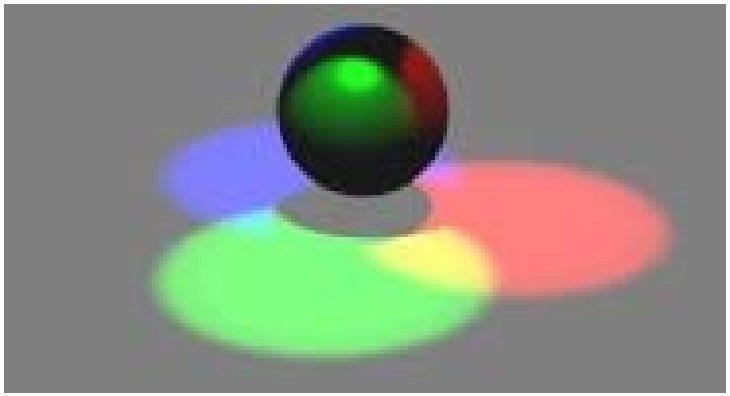
Sung-Eui Yoon
(윤성의)

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

KAIST

Course Objectives (Ch. 8)

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

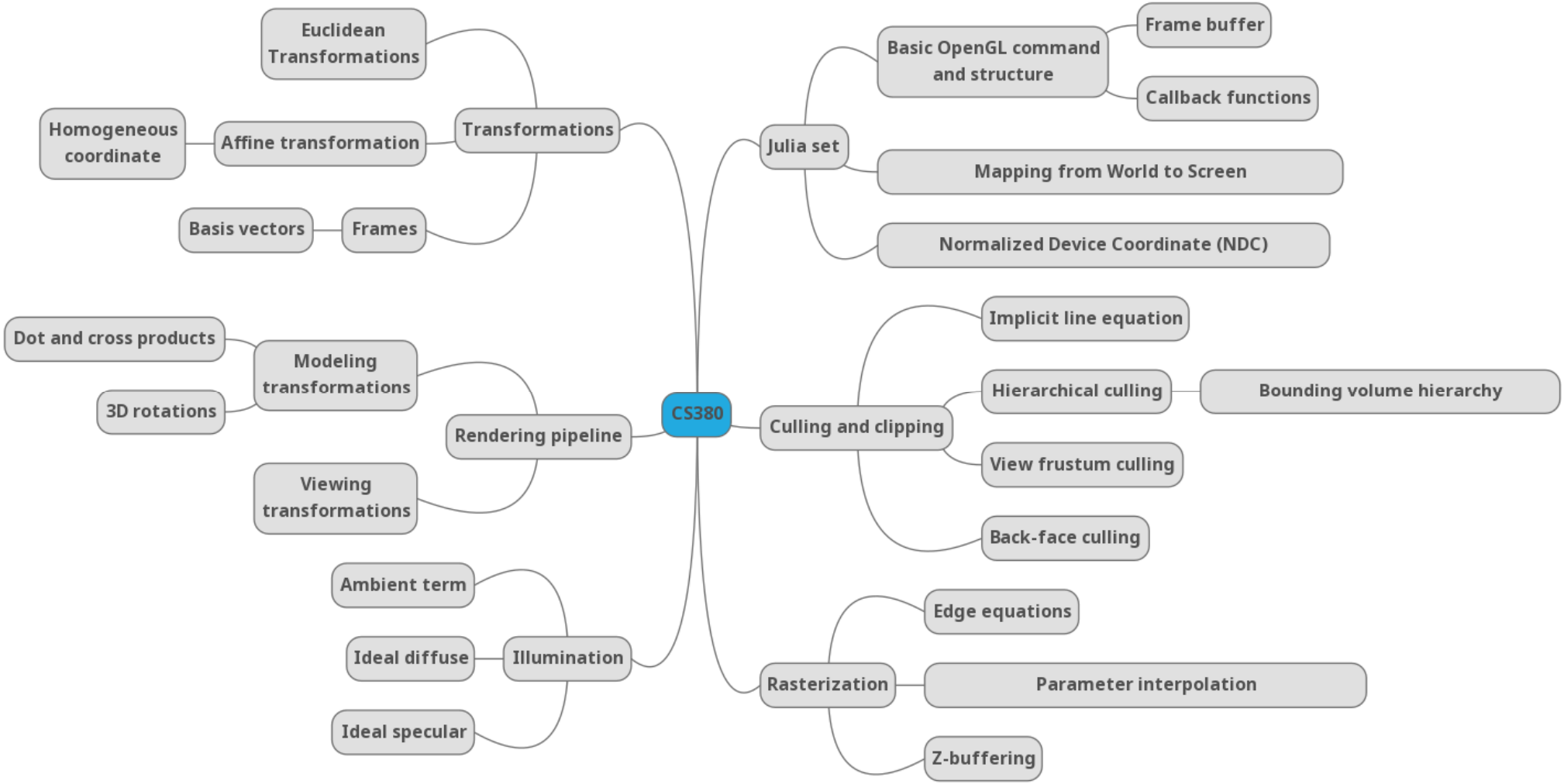


- **At the last class:**
  - **Ambient and diffuse terms**
  - **Specular term**

Questions

- **This question is about formula in Lambert's Cosine Law.**
  Since energy of light must be included (with assumption that surface does not absorb energy), The sum of outgoing ray's energy(or brightness) should be equal to incoming ray's energy. Does the formula here satisfies this fact?

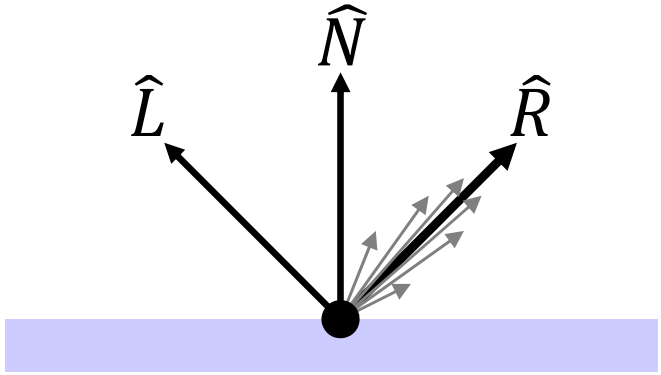
Summary so far

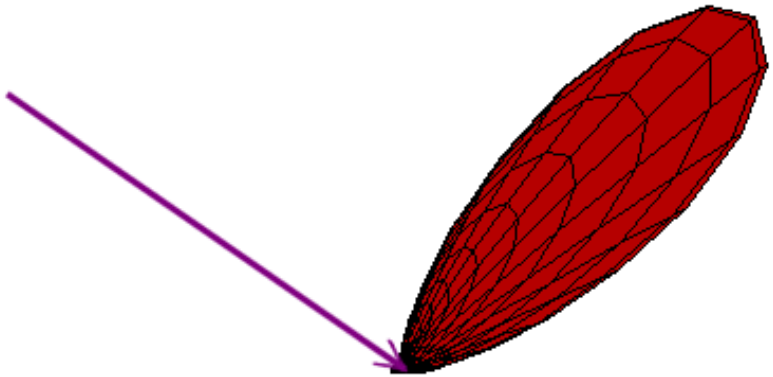


KAIST

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**



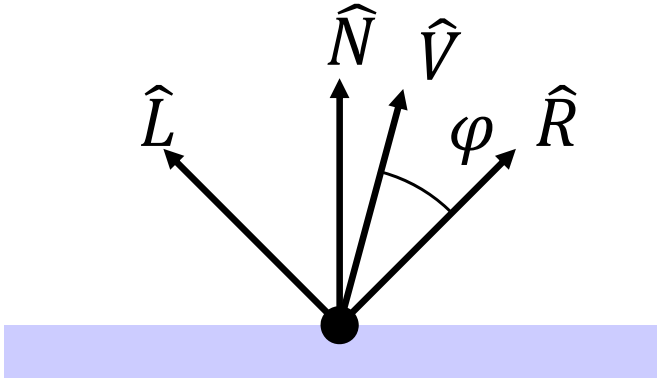


KAIST

Phong Illumination

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

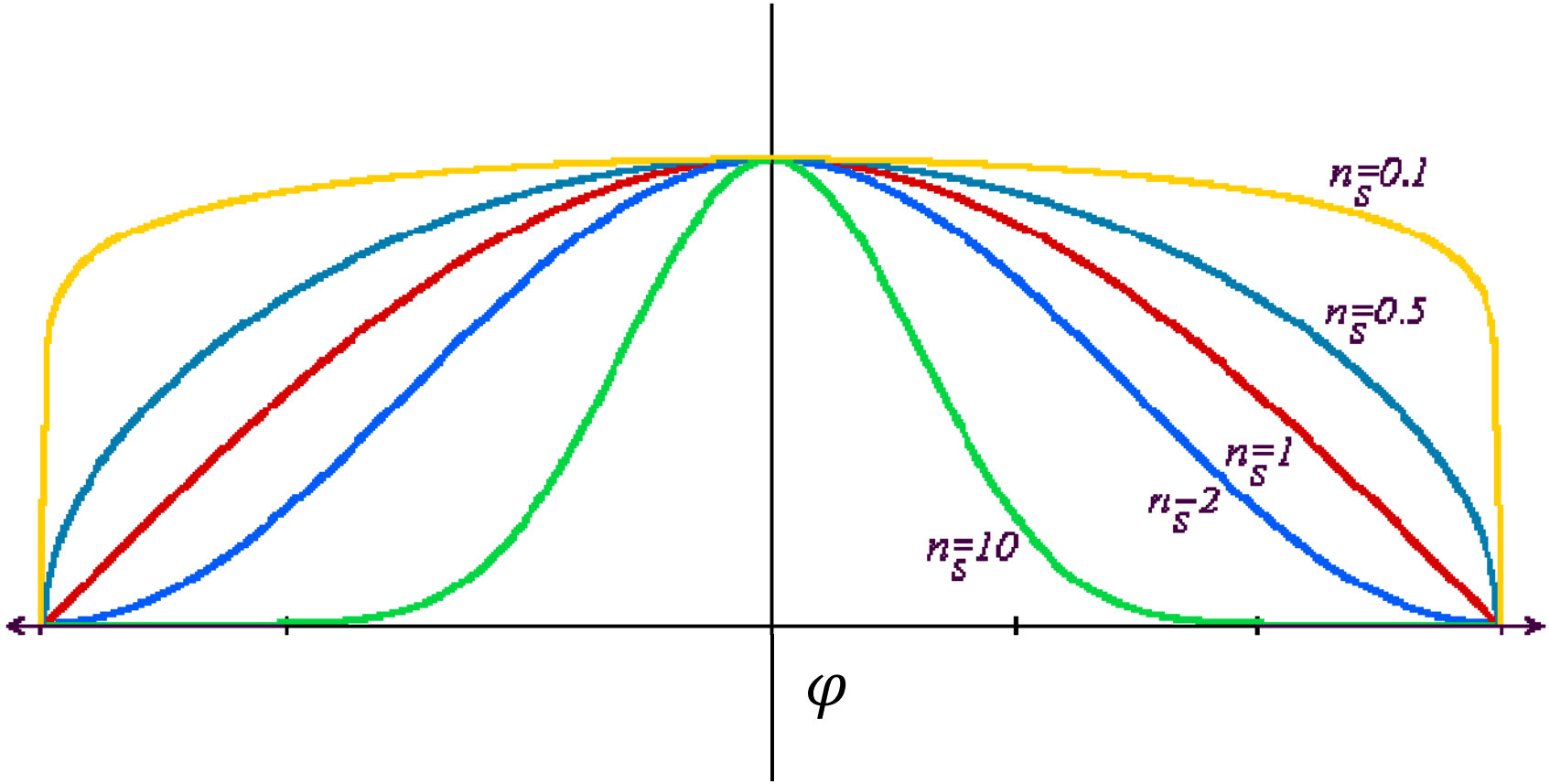
$$I_r = k_s I_i (\cos\varphi)^{n_s}$$
$$= k_s I_i (\hat{V} \bullet \hat{R})^{n_s}$$



- $\hat{V}$ **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**

KAIST

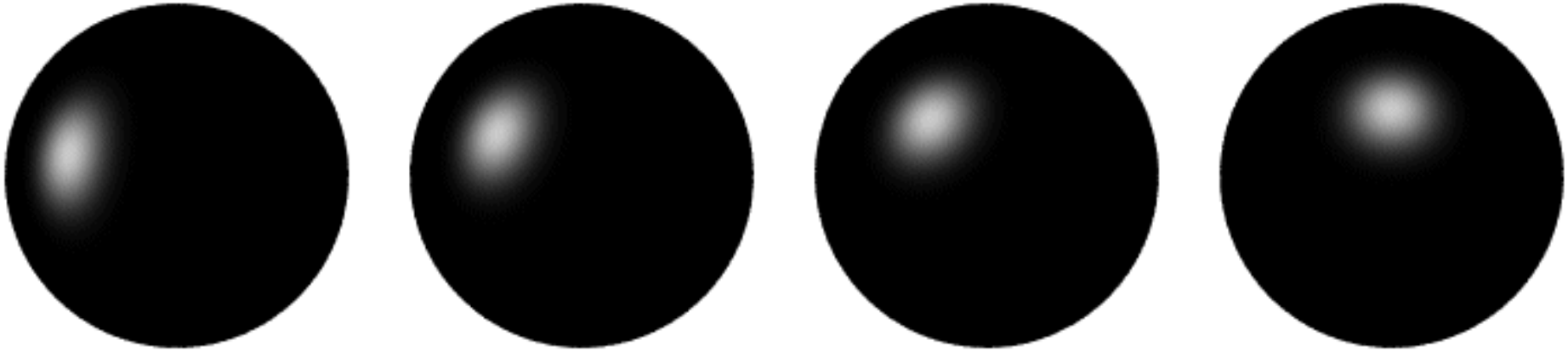
Effect of Specular Exponent



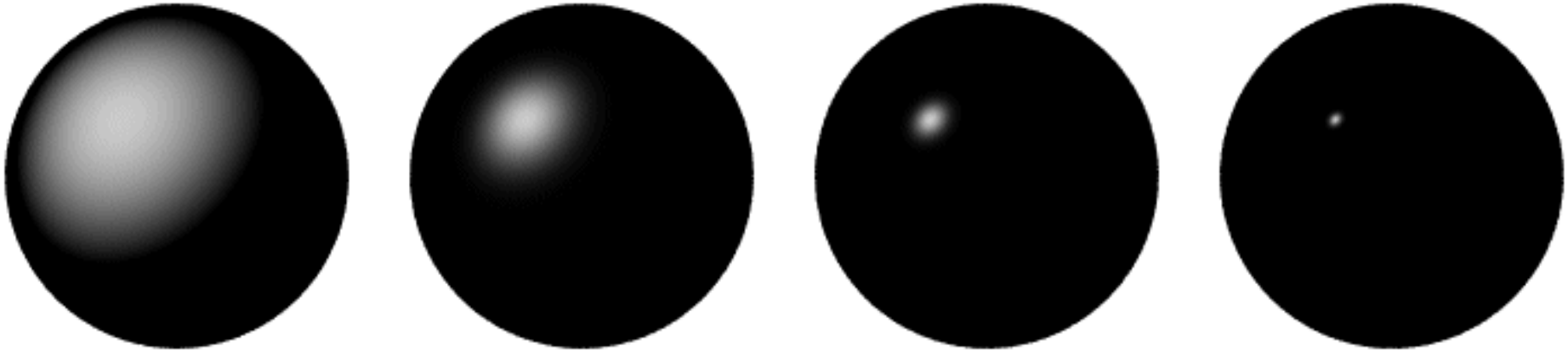
- **How the shape of the highlight changes with varying n_s**

KAIST

Examples of Phong



varying light directions



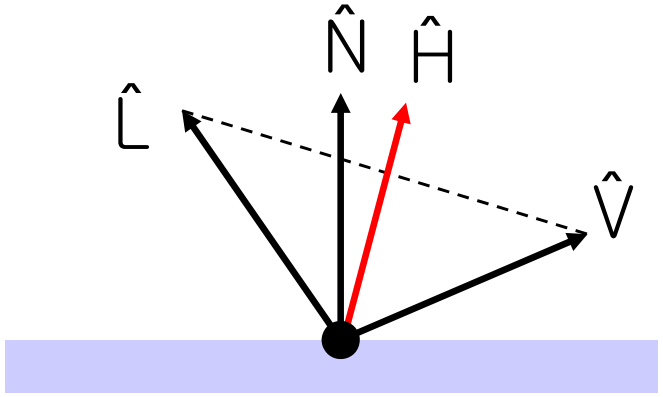
varying specular exponents

Blinn & Torrance Variation

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

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

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



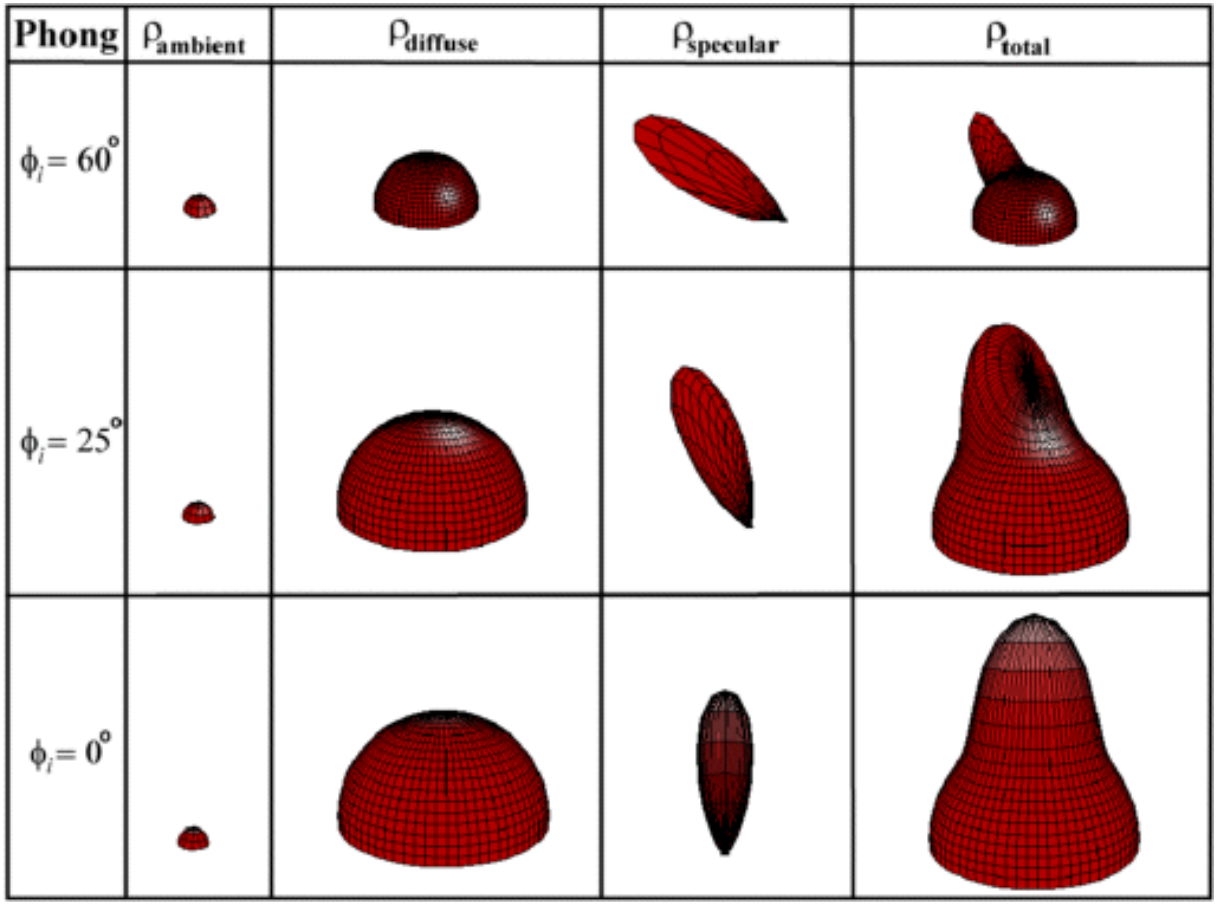
- $\hat{H}$ **is the half-way vector that bisects the light and viewer directions**

KAIST

Putting it All Together

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

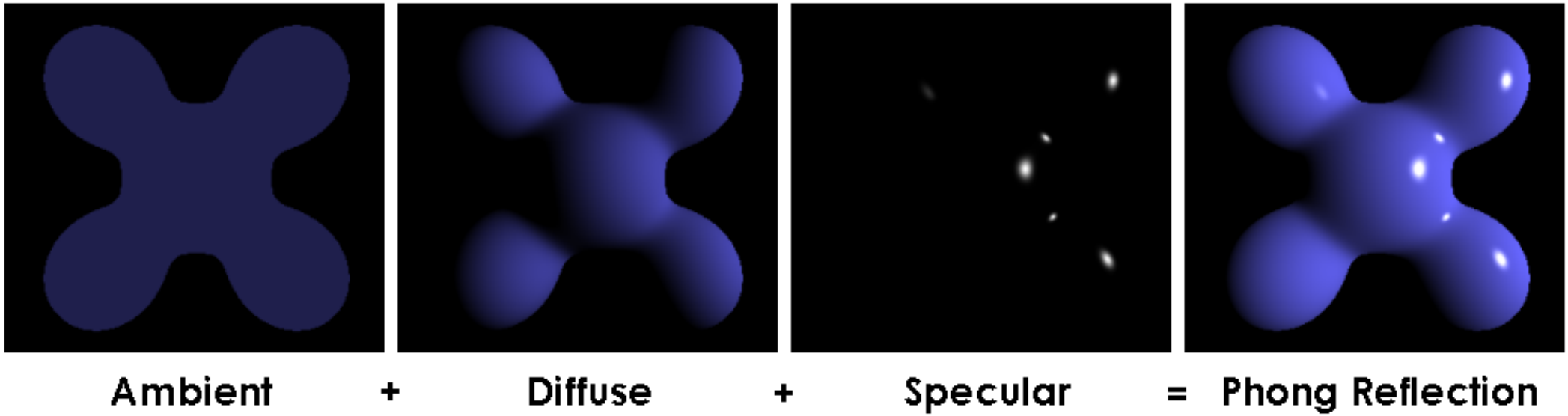
Light angle



KAIST

Putting it All Together, aka, Phong Illumination

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



From Wikipedia

KAIST

OpenGL's Illumination Model

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

- **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?**

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

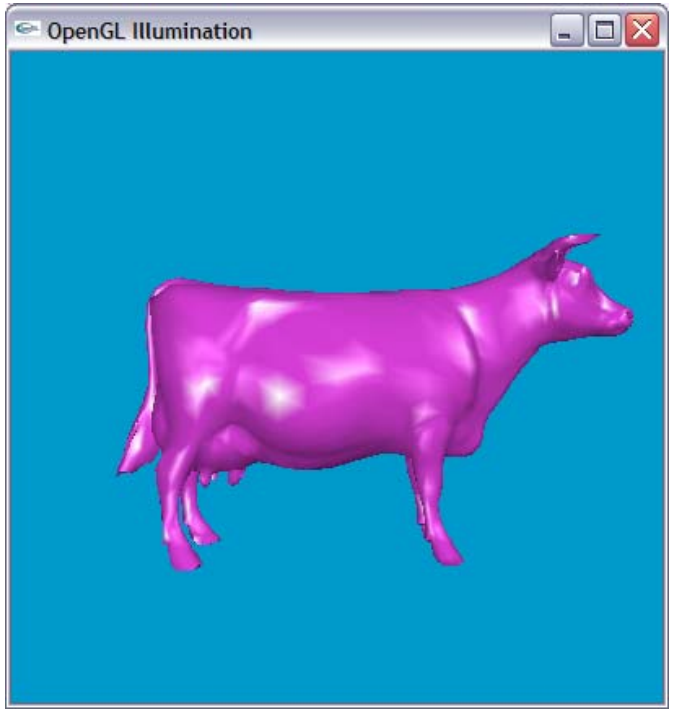
```
# define a directional light
lightDirection = [1, 1, 1, 0]
glLightfv(GL_LIGHT0, GL_POSITION, lightDirection)
glEnable(GL_LIGHT0)

# 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)
```



KAIST

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

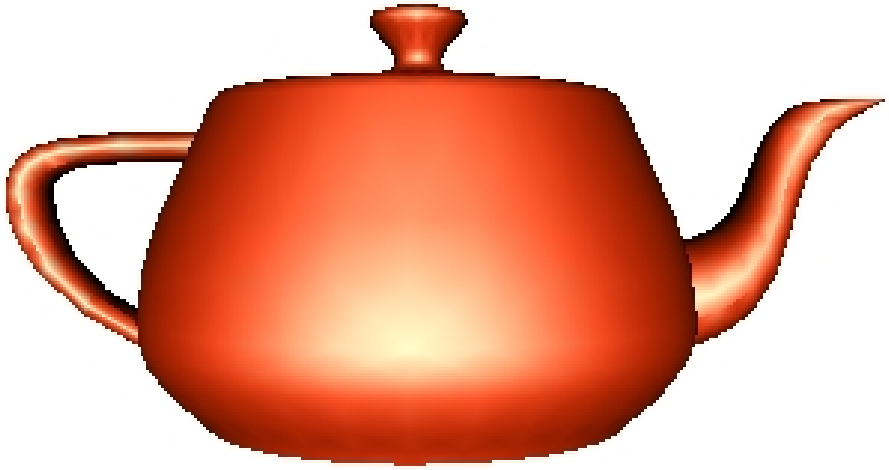
- **The simplest shading method**
  - **Applies only one illumination calculation per face**
- **Illumination usually computed at the centroid of the face:**

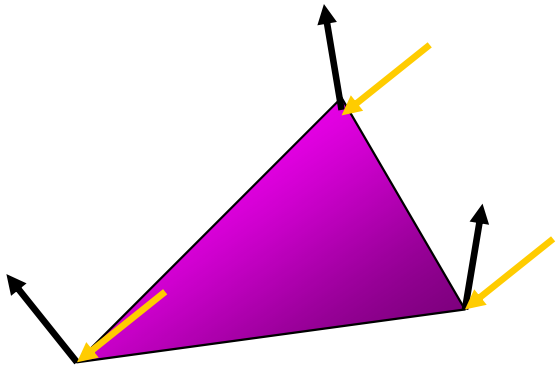
$$centroid = \frac{1}{n}\sum_{i=1}^{n} \dot{p}_i$$

- **Issues?**

Gouraud Shading

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



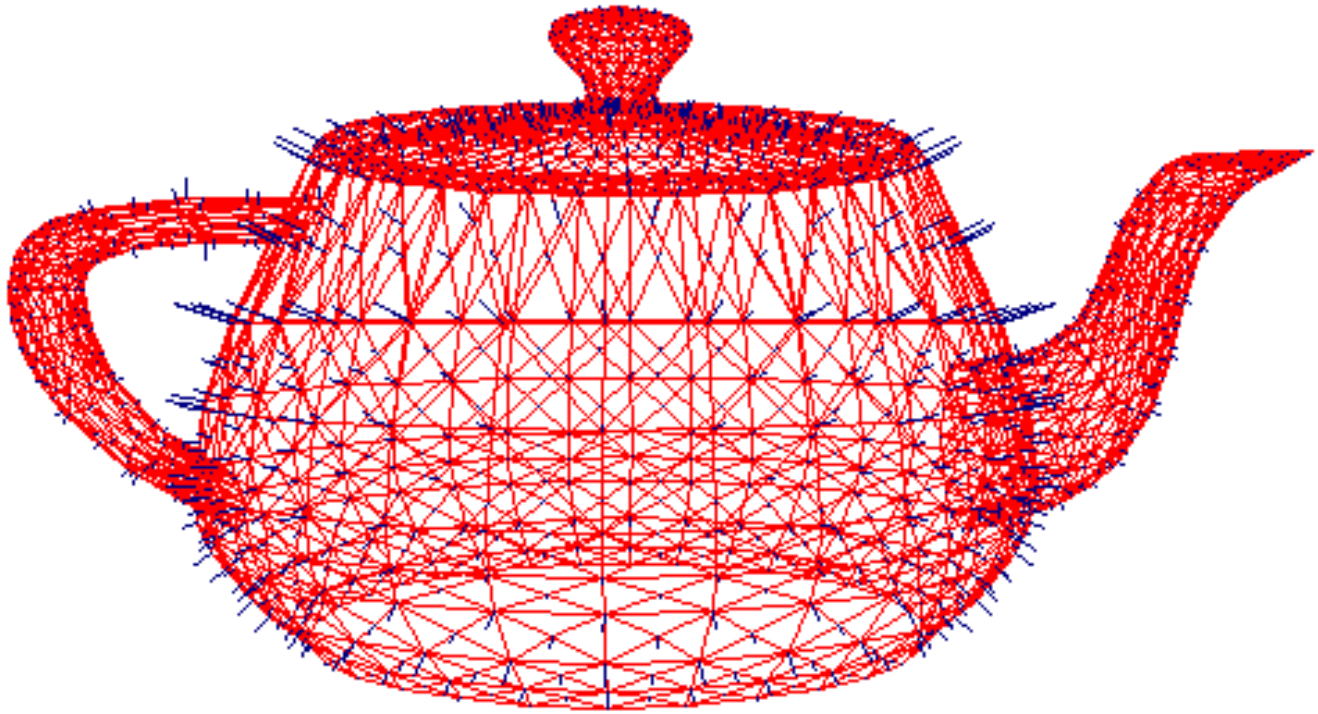


Notice that facet artifacts are still visible

Vertex Normals

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

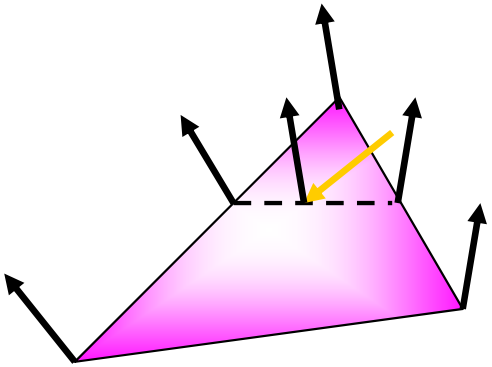
$$\vec{n}_v = \sum_{i=1}^{k} \vec{n}_{face,i}$$



KAIST

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**
- **Commonly adopted in 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**

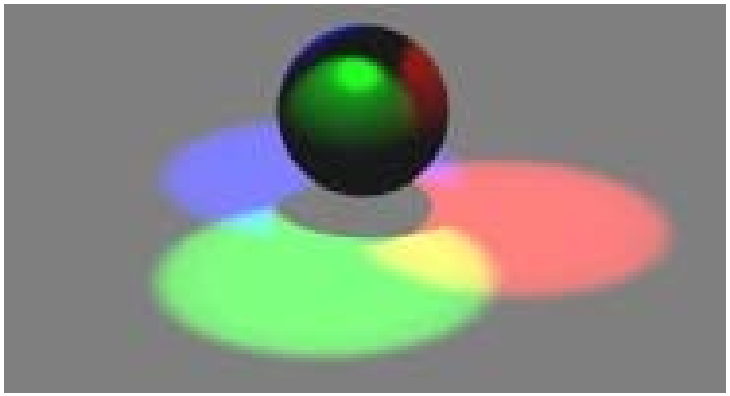
Various Effects using Physically-based Models

From slides of Pat Hanrahan

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

Course Objectives were:

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



Homework

- **Go over the next lecture slides before the class**
- **Watch 2 SIGGRAPH videos and submit your summaries before every Mon. class**
  - **Just one paragraph for each summary**
- **Submit questions two times during the whole semester**