

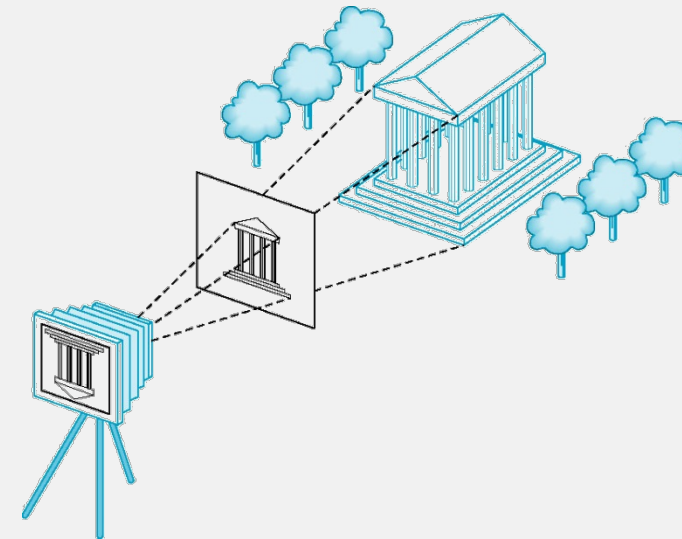
# Shading

Junho Kim

Kookmin University

# Elements of Image Formation

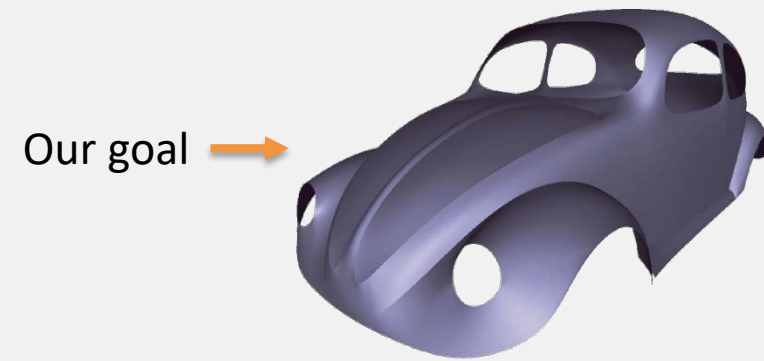
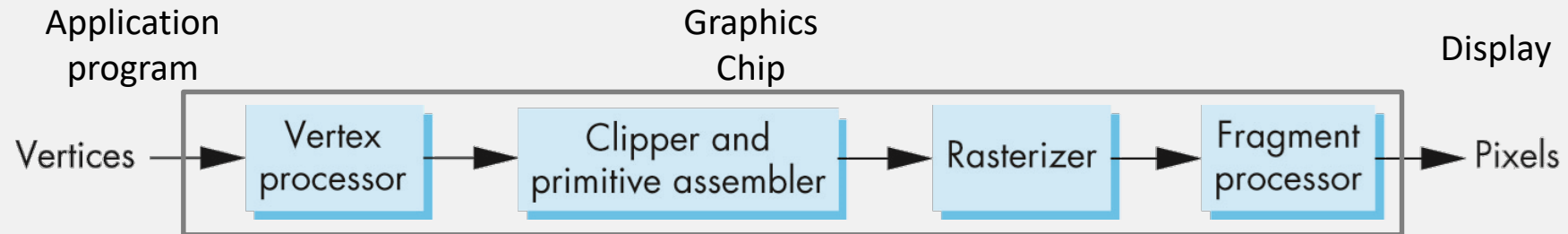
- Viewer (or camera)
  - Synthetic camera
- Objects
  - Synthetic objects
- Light source(s)
  - **Synthetic lights**
- Attributes
  - **Material, surface normal**  
for reflection model  
(i.e., light-material interaction)



**Synthetic image formation**  
in Computer Graphics

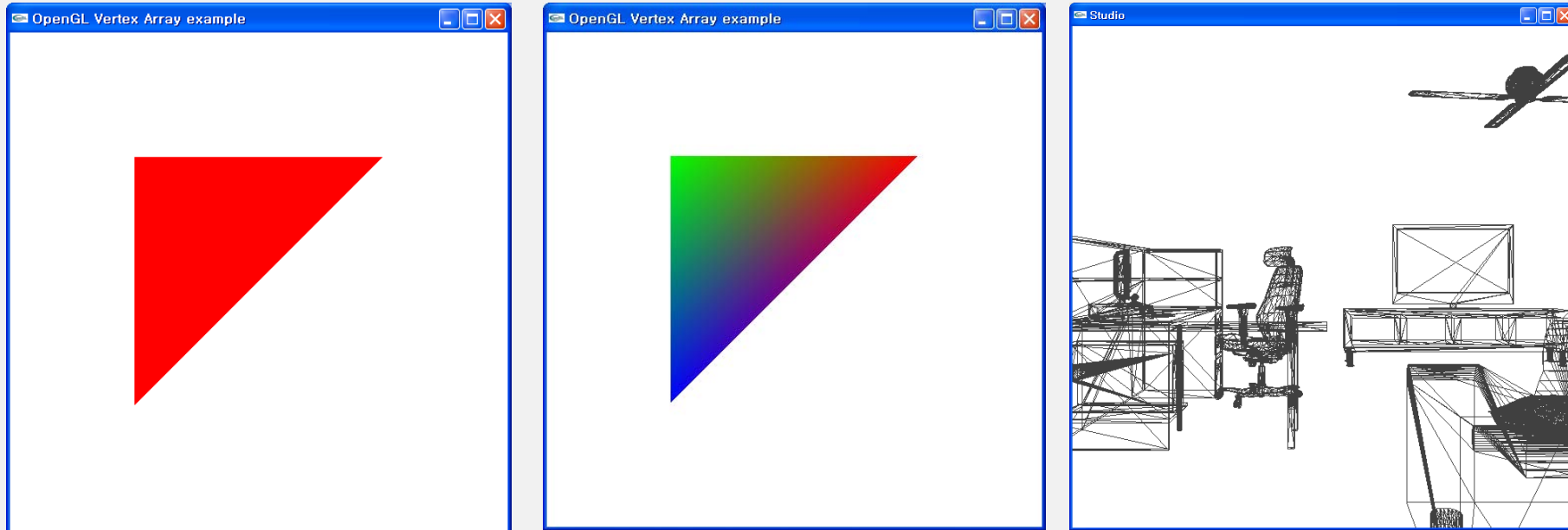
# Overview of Rendering Pipeline

- Pipeline architecture
  - This is everything for interactive computer graphics!
    - First, we focus on the *fixed rendering pipeline*
  - Mechanism: a *state* machine
    - All information for image formations should be specified

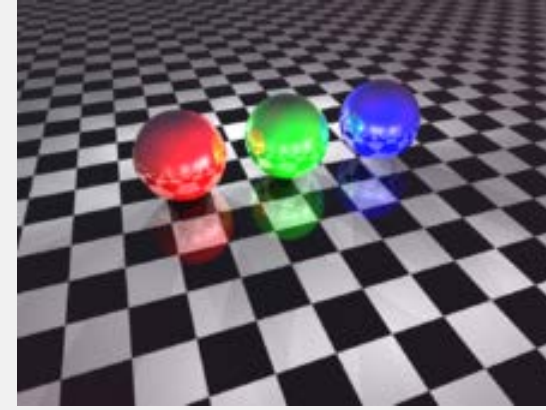


# What's going on?

- This is NOT what I expect on the graphics!!!



# Let There Be Light

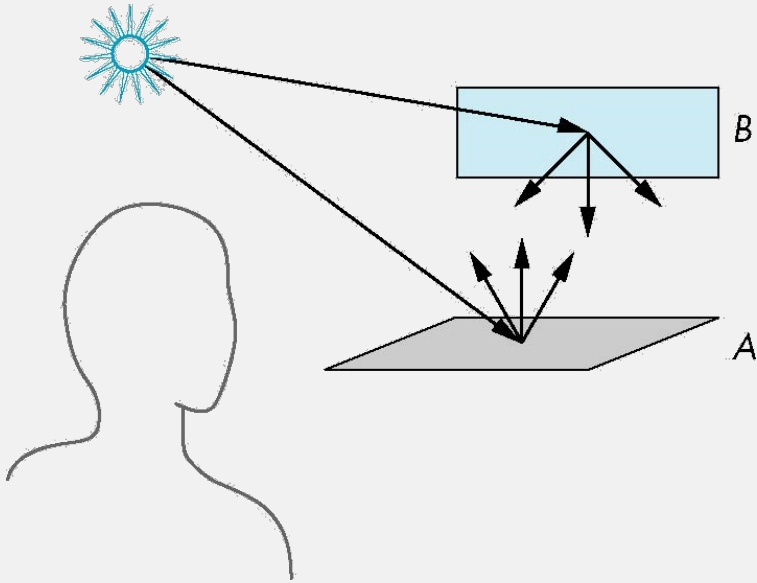


# Objectives

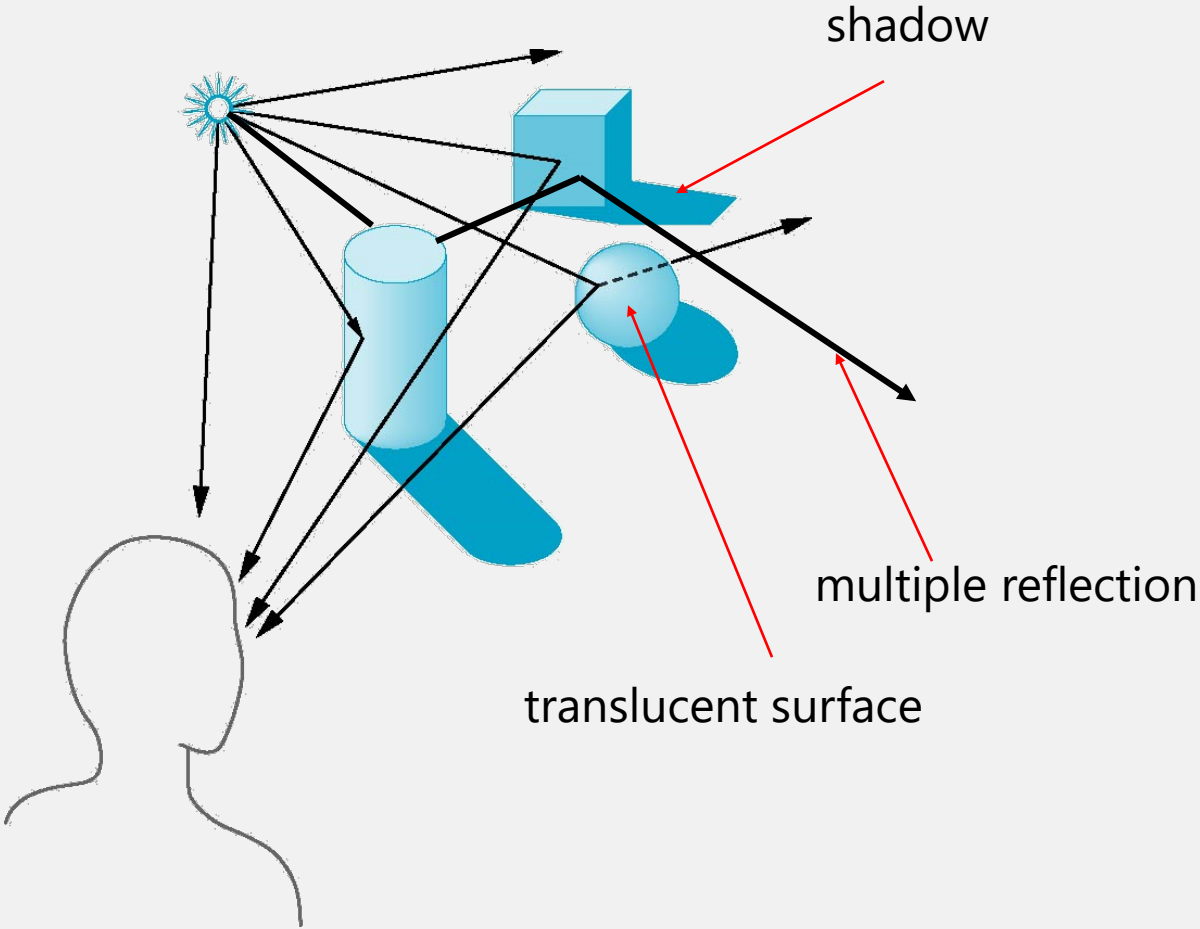
- To understand the following concepts
  - The fundamentals on Light and Material
  - Global illumination v.s. local illumination
  - Basic concepts about BRDF and Rendering Equation
  - Phong lighting model

# Light

- When light strike a surface
  - Some scattered
  - Some absorbed
- Notice, the scattered light strike again other surfaces

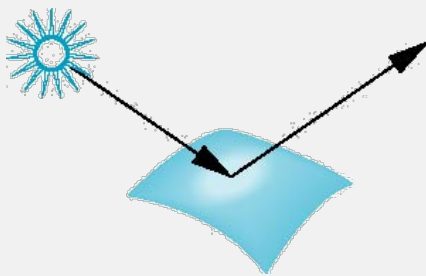
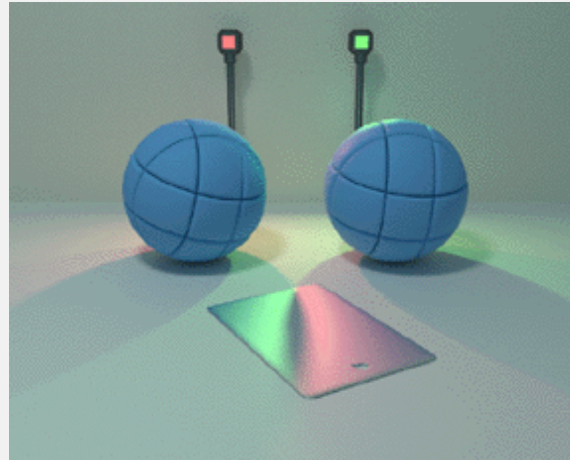


# Global Effects

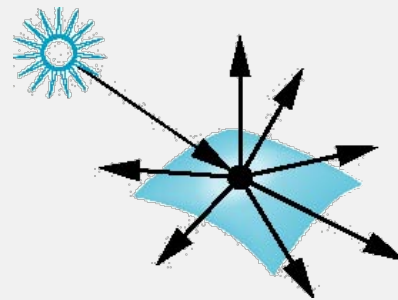




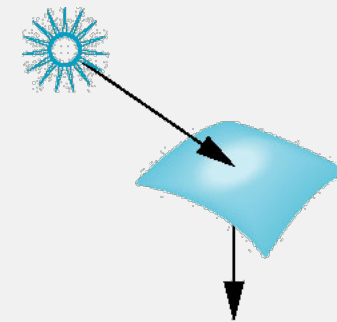
# Light – Material Interaction



Specular surface



Diffuse surface



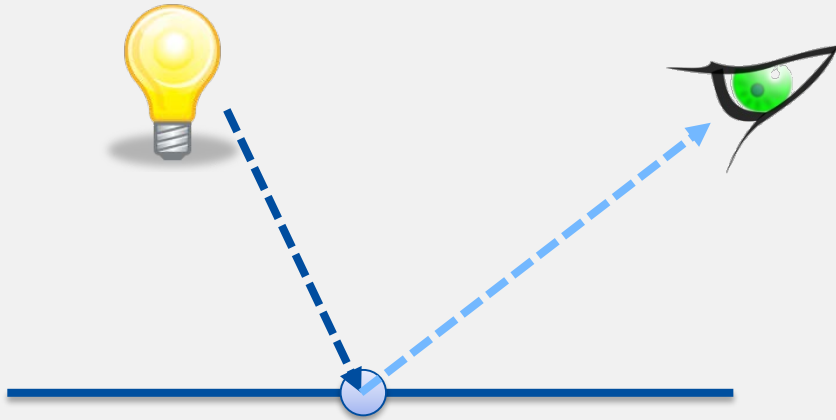
Translucent surface

# Light – Material Interaction

- Actually, very complicated
- It depends on
  - The portion of absorbed/scattered light
  - Color of the object
    - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
  - Smoothness and orientation of the surface

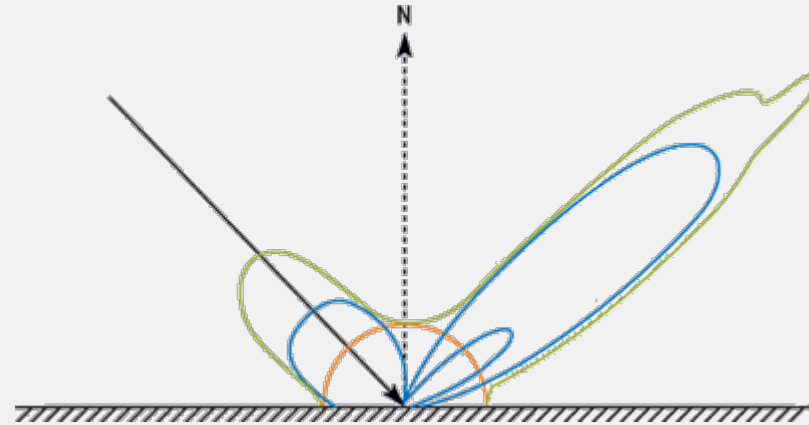
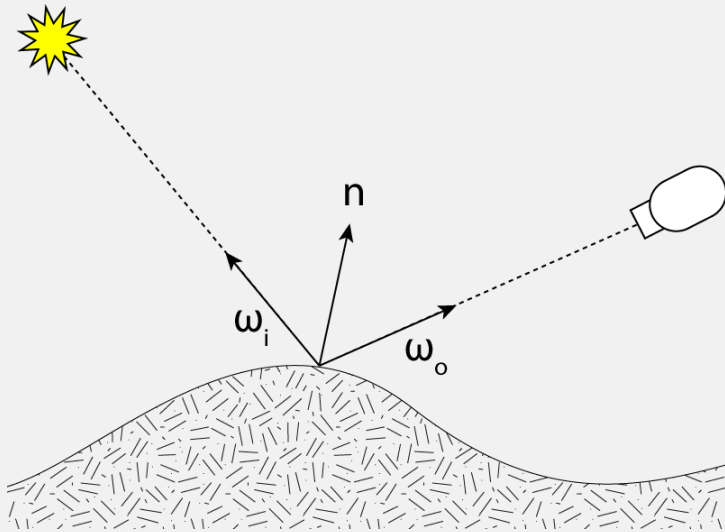
# Color

- Where does color come from?
  - Color of light sources
  - Reflection ratio of a surface point
  - Angle of light sources and eyes



# Bidirectional Reflection Distribution Function (BRDF)

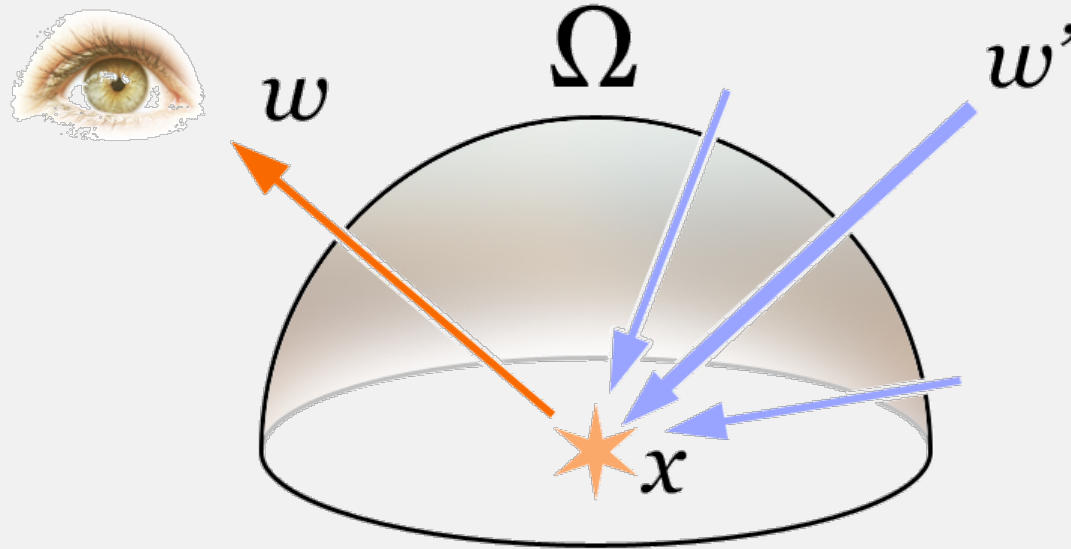
- Physically correct model
  - light direction
  - viewer direction
  - frequency of the light



# Rendering Equation

- Global Effects with BRDF

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



# Rendering Equation

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

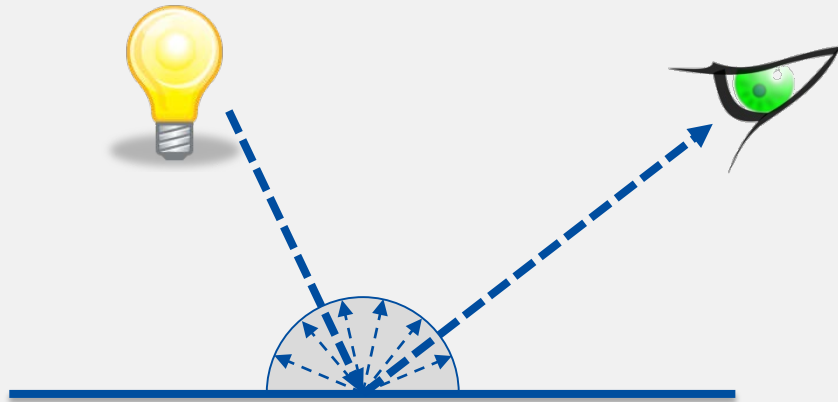
- Global lighting model
  - Radiosity, Ray tracing
    - Movie & animation use this approach
  - Non-interactive approach (not in real-time)





# Color

- Where does color come from?



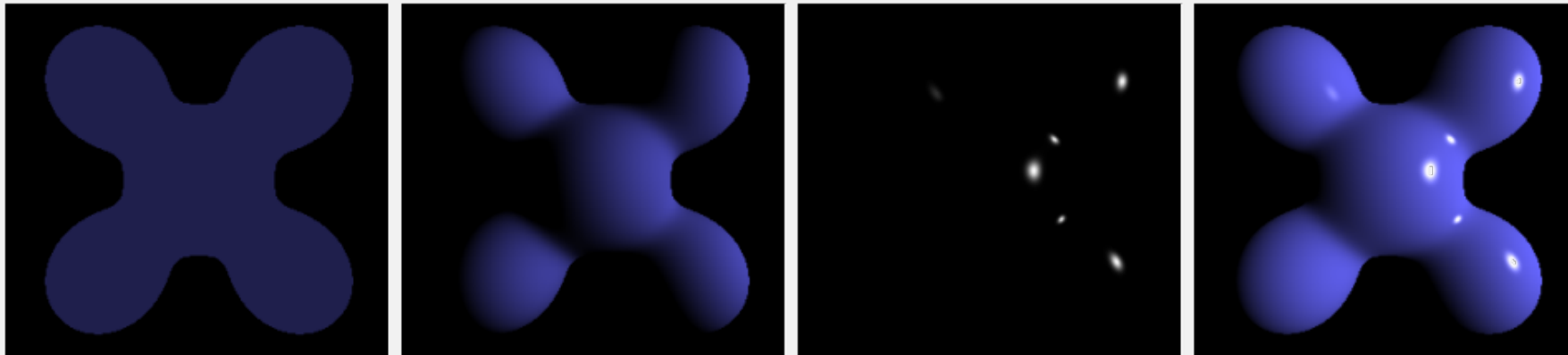
# Phong Reflection Model



# Phong Reflection Model

- Local lighting model
  - Physically, somewhat strange
  - But, close enough approximation to physical reality

$$I_a + I_d + I_s = I$$



Ambient + Diffuse + Specular = Phong Reflection

# Ambient Reflection (주변광)

- Ambient reflection accounts for the small amount of light that is scattered about the entire scene



[<http://forums.steves-digicams.com/photo-critiques/196118-interior-arches-ambient-light.html>]

# Ambient Reflection (주변광)

- Ambient term
  - Same at every point on the surface
  - Usually, it should be set as small as possible

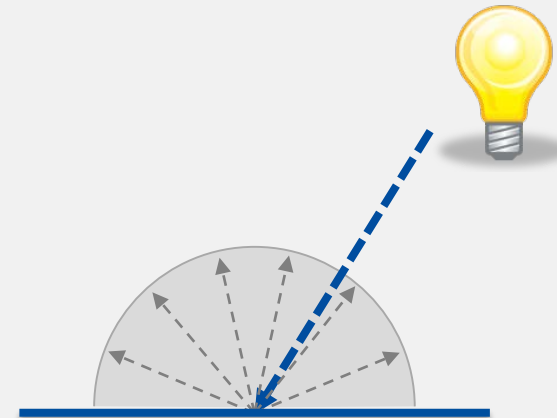
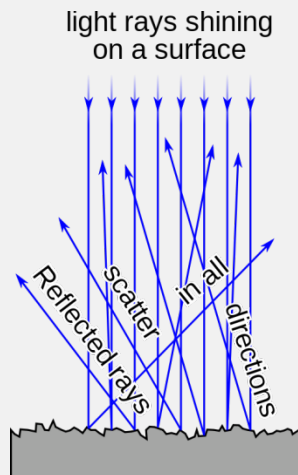


$$I_a = \kappa_a L_a$$

intensity of ambient light      ambient reflection coefficient      illumination of ambient light

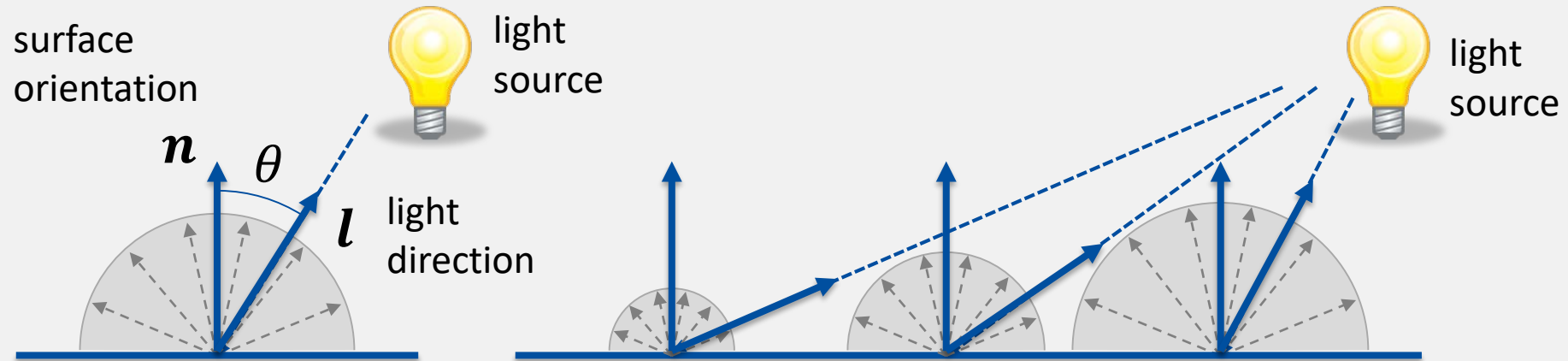
# Diffuse Reflection (난반사광)

- In physics, an incident ray is reflected at many angles
- In graphics, we consider diffuse reflection as lambertian reflectance
  - The apparent brightness of a lambertian surface to an observer is the same regardless of the viewer's angle of view



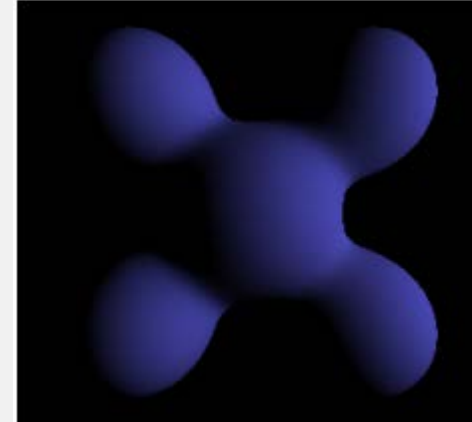
# Diffuse Reflection (난반사광)

- The brightness of diffuse reflection is related to the angle  $\theta$  between the light direction  $\boldsymbol{l}$  and the surface orientation  $\boldsymbol{n}$



# Diffuse Reflection (난반사광)

- Diffuse term
  - It reflect equally in all direction,
  - But the intensity depends on
    - Angle between light dir. & surface normal



$$\underline{I_d} = \underline{\kappa_d} \max(\underline{(l \cdot n)}, 0) \underline{L_d}$$

intensity of  
diffuse light

diffuse reflection  
coefficient

angle

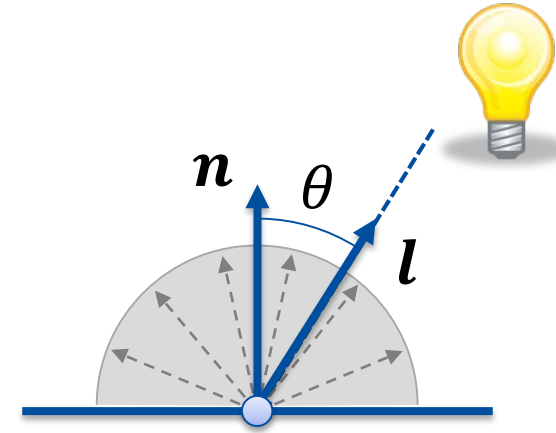
illumination of  
diffuse light

# Diffuse Reflection – OpenGL ES codes

```
// Parameters on the current Material and a Light
float mat_Kd[4] = {0.1f, 0.1f, 0.1f, 1.0f};
float light_Ld[4] = {1.0f, 1.0f, 1.0f, 1.0f};
float light_pos[4] = {-2.0f, 2.0f, 2.0f, 1.0f};

// Setting the current material and a light GL_LIGHT0
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_Kd);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_Ld);
glLightfv(GL_LIGHT0, GL_POSITION, light_pos);

// Setting an object info.
glVertexPointer(...);
glNormalPointer(...);
```



$$I_d = \kappa_d \max((l \cdot n), 0) L_d$$

intensity of  
diffuse light

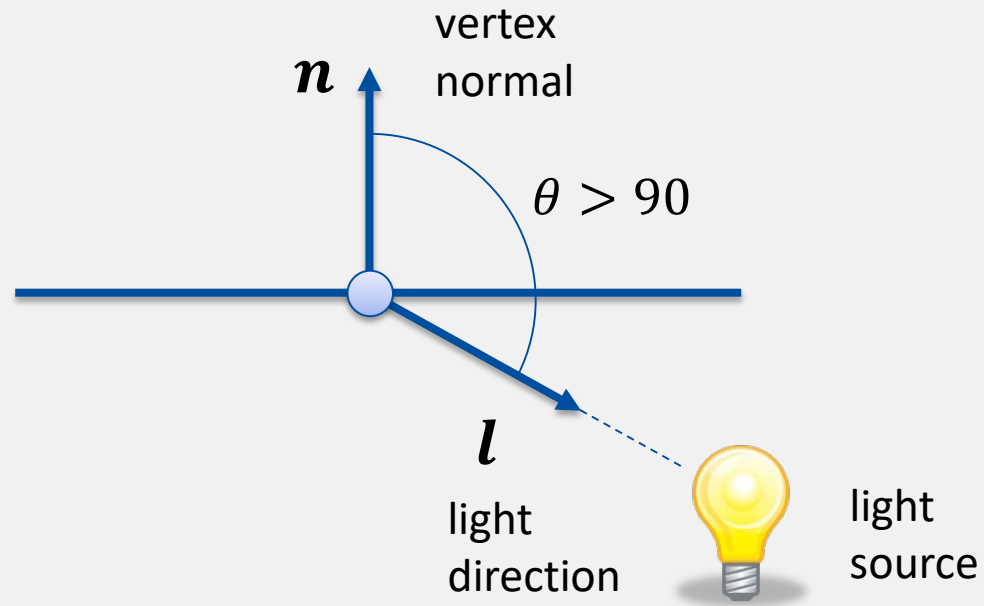
diffuse reflection  
coefficient

angle

illumination of  
diffuse light

# Why using $\max(\cdot, \cdot)$ ?

- What is the meaning of  $\max((\mathbf{l} \cdot \mathbf{n}), 0)$ ?
  - If  $\mathbf{l} \cdot \mathbf{n} \geq 0$ , then light hits the surface on its **front** face
  - If  $\mathbf{l} \cdot \mathbf{n} < 0$ , then light hits the surface on its **back** face
  - It means, we just consider lights that hit the surface of its front face only



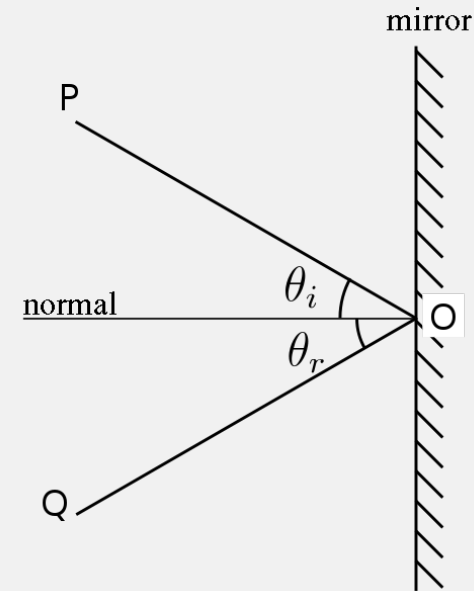


# Specular Reflection (정반사광)

- Specular reflection is the mirror-like reflection of light from a surface, in which light from a single incoming direction is reflected into single outgoing direction

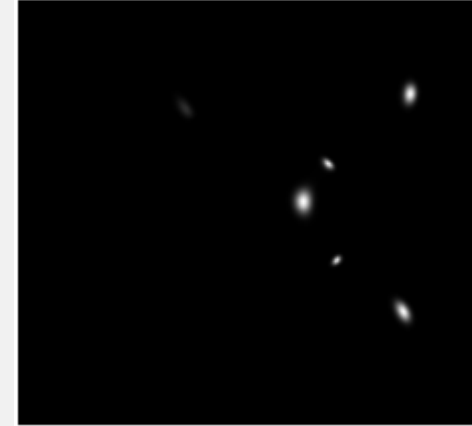
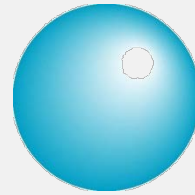


[Image from Top Gear UK]



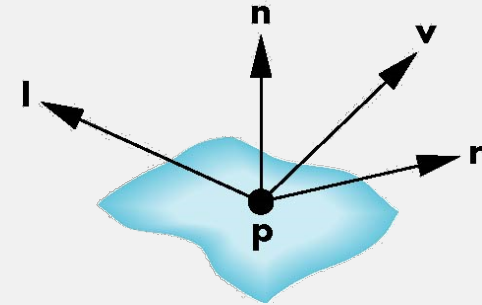
# Specular Reflection (정반사광)

- Specular term
  - It represent the highlights
  - The intensity depends on
    - Angle among reflection dir. & viewer dir.
    - shininess of the material



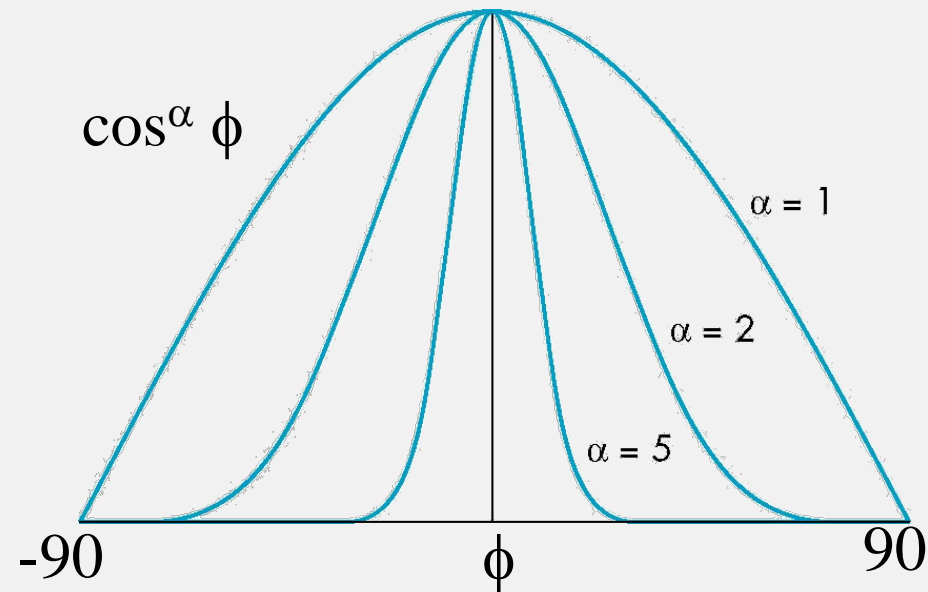
$$I_s = \kappa_s \max((\mathbf{r} \cdot \mathbf{v})^\alpha, 0) L_s$$

intensity of specular light    specular reflection coefficient    angle    illumination of specular light

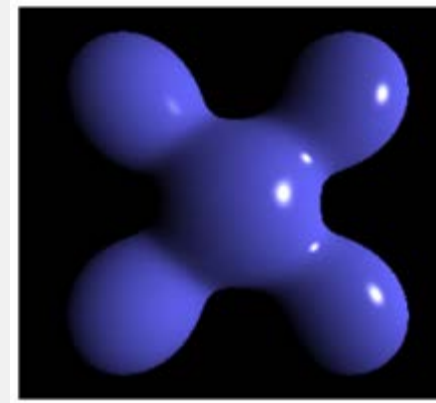


# Shininess Coefficient in Specular Reflection

- Values of  $\alpha$  between 100 and 200 correspond to metals
- Values of  $\alpha$  between 5 and 10 give surface that look like plastic



# Phong Reflection Model



- All together
  - ambient reflection
  - diffuse reflection
  - specular reflection

} Effective light intensity may be attenuated by the distance from the light

$$I = \frac{1}{a + bd + cd^2} (I_d + I_s) + I_a$$
$$= \frac{1}{a + bd + cd^2} (\kappa_d \max((\mathbf{l} \cdot \mathbf{n}), 0) L_d + \kappa_s \max((\mathbf{r} \cdot \mathbf{v})^\alpha, 0) L_s) + \kappa_a L_a$$

Constants for light attenuations can be set by [glLight\(\)](#);

- $a$ : GL\_CONSTANT\_ATTENUATION
- $b$ : GL\_LINEAR\_ATTENUATION
- $c$ : GL\_QUADRATIC\_ATTENUATION

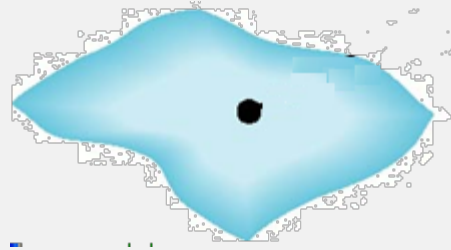
# Phong Reflection Model

- A light source have three types of lighting
  - Ambient lighting
  - Diffuse lighting
  - Specular lighting

$$L = \begin{bmatrix} L_a \\ L_d \\ L_s \end{bmatrix} = \begin{bmatrix} L_{r,a} & L_{g,a} & L_{b,a} \\ L_{r,d} & L_{g,d} & L_{b,d} \\ L_{r,s} & L_{g,s} & L_{b,s} \end{bmatrix}$$

# Phong Reflection Model

- A color of a point on the surface

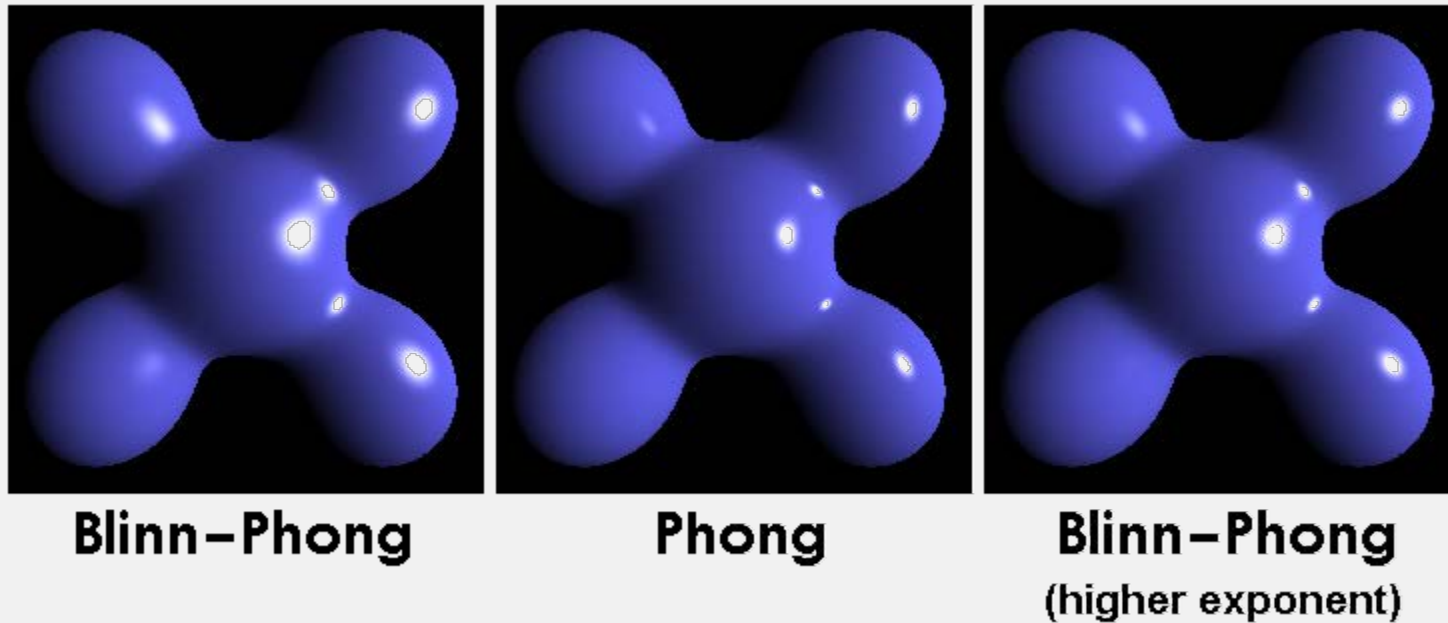


$$I = \sum_i (I_{i,a} + I_{i,d} + I_{i,s}) + I_{global,a}$$

- What does it mean?
  - Your vertex color is computed by using
    - Information of several lights
    - Information of the current material
    - Surface normal of a vertex

# Phong-Blinn Reflection Model

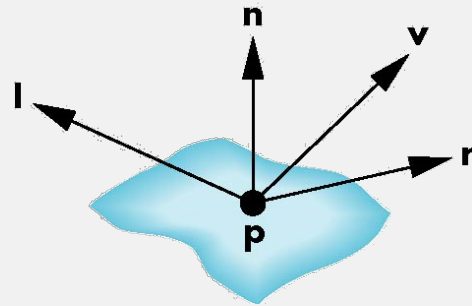
- [Phong-Blinn reflection model](#)
  - A variation of Phong reflection model
  - In graphics HW, Phong-Blinn reflection model is implemented, instead of Phong reflection model



# Phong-Blinn Reflection Model

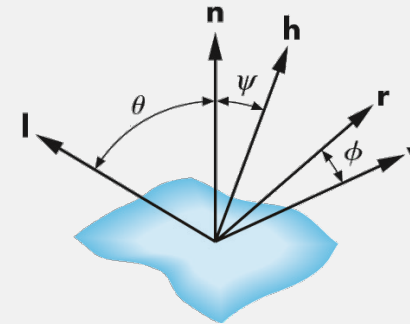
- Main purpose of Phong-Blinn reflection
  - To reduce computation time, by eliminating per-vertex computations
    - Especially, when we consider the viewer and light are treated to be at infinity

Phong reflection model



$$I_s = \kappa_s \max((\mathbf{r} \cdot \mathbf{v})^\alpha, 0) L_s$$

Phong-Blinn reflection model



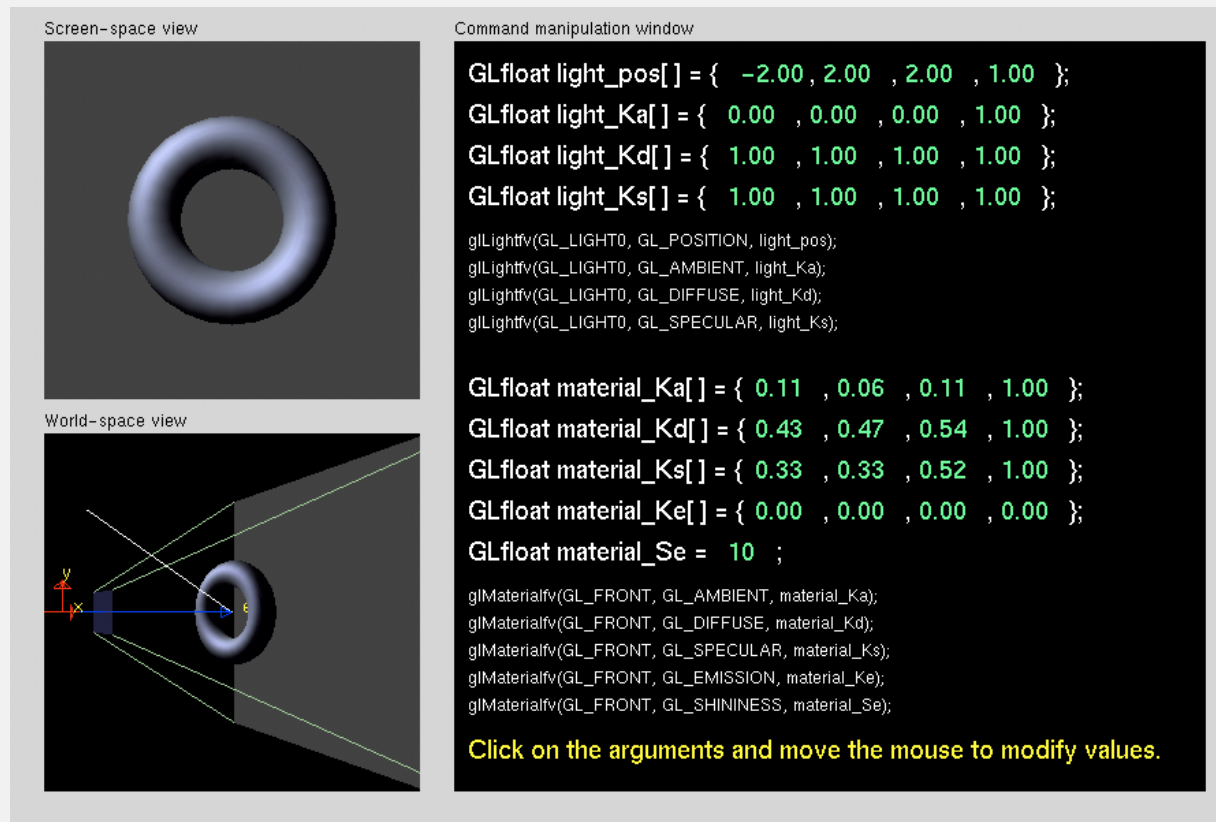
$$I_s = \kappa_s \max((\mathbf{n} \cdot \mathbf{h})^\alpha, 0) L_s$$

where the half vector  $\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}$



# Demo – Phong-Blinn Reflection Model

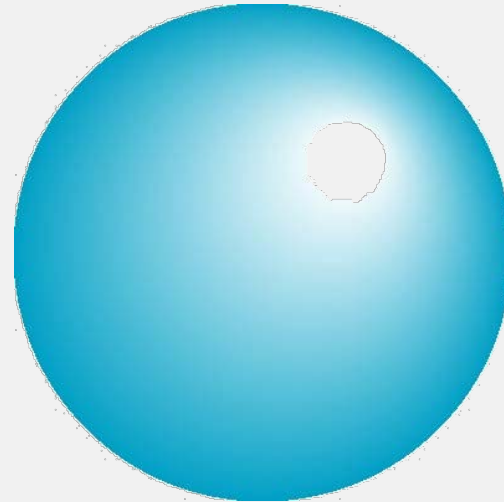
- Tutorial from Nate Robins
  - <http://user.xmission.com/~nate/tutors.html>



# Computing Per-vertex Normals

# Computing Per-Vertex Normal w/ Algebraic Surfaces

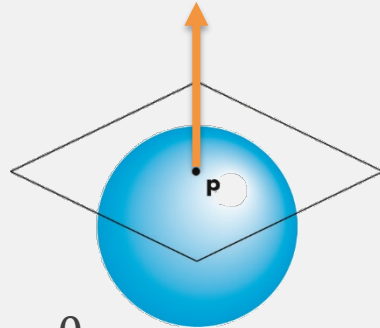
- Algebraic surfaces
  - We can describe a set of points on the surface in an algebraic form
    - Implicit equation
      - Ex)  $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
    - Parametric equation
      - Ex)  $x(u, v) = \cos u \sin v$   
 $y(u, v) = \cos u \cos v$   
 $z(u, v) = \sin u$



# Computing Per-Vertex Normal w/ Algebraic Surfaces

- Normal w/ Implicit Equation

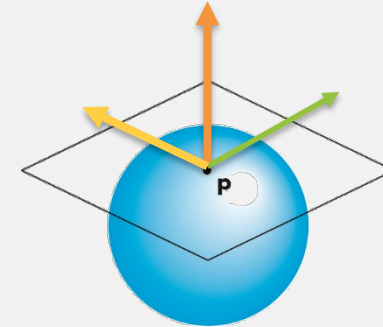
$$\mathbf{n}_p = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \partial f / \partial z \end{bmatrix} = \nabla f$$



- Ex)  $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$

- Normal w/ Parameteric Equation

$$\mathbf{n}_p = f_u(\mathbf{p}) \times f_v(\mathbf{p})$$



- Ex) 
$$\begin{aligned} x(u, v) &= \cos u \sin v \\ y(u, v) &= \cos u \cos v \\ z(u, v) &= \sin u \end{aligned}$$

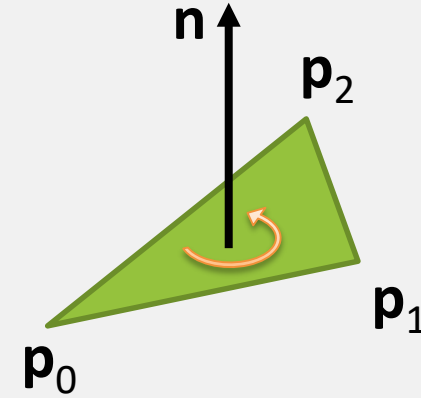
# Computing Per-Vertex Normal w/ Polygonal Model

- Compute each face normal

$$\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)$$

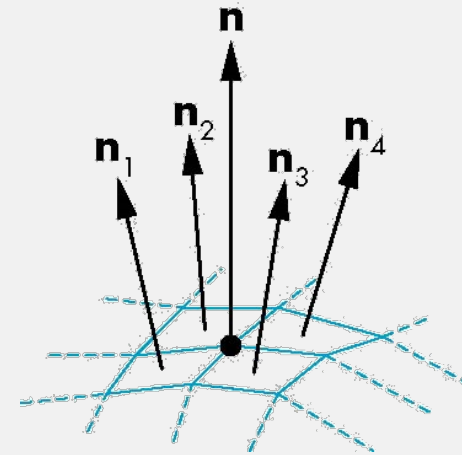
Note:

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_2 v_3 - v_2 u_3 \\ u_3 v_1 - v_3 u_1 \\ u_1 v_2 - v_1 u_2 \end{bmatrix}, \text{ where } \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \text{ and } \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$



- Averaging incident face normals

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$

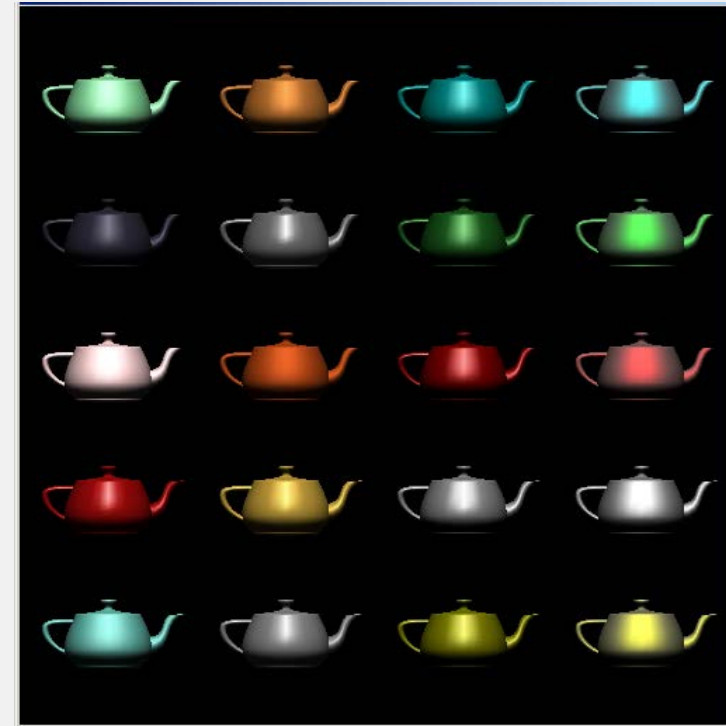


# Phong Reflection Model

## Light-Material Interaction in OpenGL ES

# Light-Material Interactions in OpenGL ES

- OpenGL uses Phong-Blinn model
  - Local lighting approach
  - No shadows, no multiple reflections
- Light
  - # of light: MAX eight lights in the scene
  - # of types: three types of lights
- Material
  - You can set the current material
- Normal
  - You can set a normal for each vertex data



# Materials in OpenGL ES

- The current material contains the coefficients for each reflection model
  - Coeff. for ambient reflection:  $\kappa_a = (\kappa_r, \kappa_g, \kappa_b, \kappa_a)_a$
  - Coeff. for diffuse reflection:  $\kappa_d = (\kappa_r, \kappa_g, \kappa_b, \kappa_a)_d$
  - Coeff. for specular reflection:  $\kappa_s = (\kappa_r, \kappa_g, \kappa_b, \kappa_a)_s$ 
    - Shininess for specular reflection:  $\alpha$

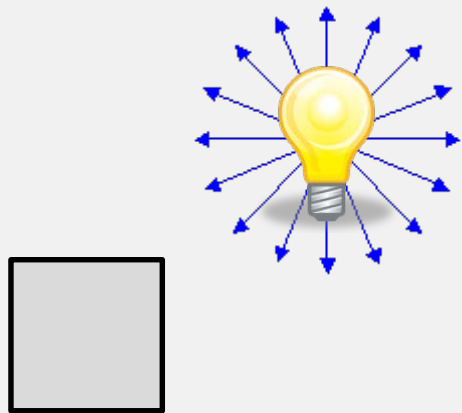
$$I = \frac{1}{a + bd + cd^2} (\kappa_d \max((\mathbf{l} \cdot \mathbf{n}), 0) L_d + \kappa_s \max((\mathbf{r} \cdot \mathbf{v})^\alpha, 0) L_s) + \kappa_a L_a$$

- See more details in [glMaterial\(\)](#)

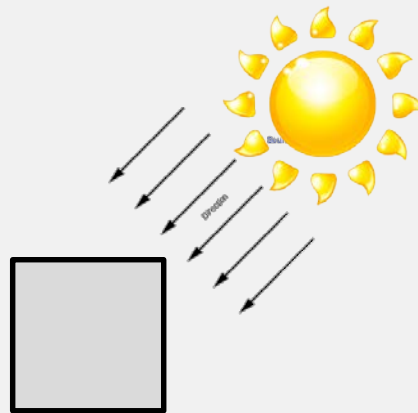


# Types of Light Source

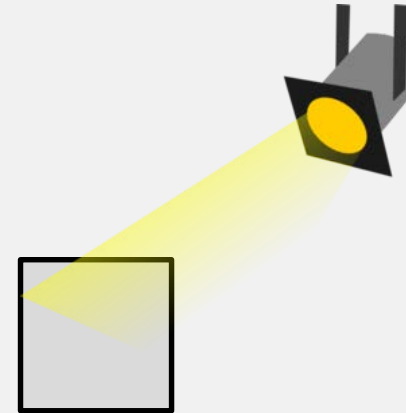
- Point light
  - Emitting light in all directions from one single point
- Directional light
  - Emitting light rays in a parallel direction
- Spot light
  - Emitting light in focus direction of the cone



Point light



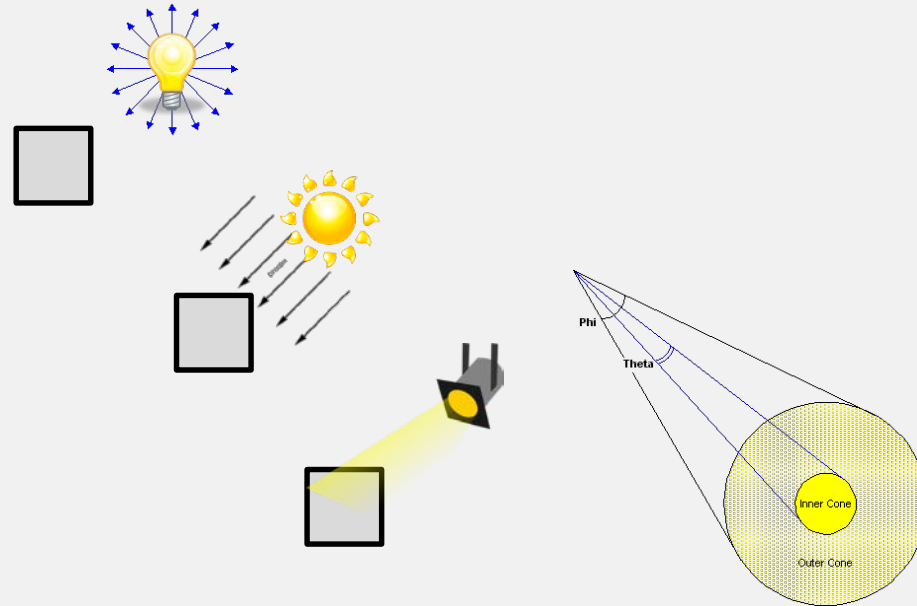
Direct light



Spot light

# Lights in OpenGL ES

- Depending on your light type, you need to specify some properties
  - Point light
    - position, attenuation
  - Directional light
    - direction, attenuation
  - Spot light
    - direction, exponent, cutoff, attenuation



$$I = \frac{1}{a + bd + cd^2} (\kappa_d \max((\mathbf{l} \cdot \mathbf{n}), 0) L_d + \kappa_s \max((\mathbf{r} \cdot \mathbf{v})^\alpha, 0) L_s) + \kappa_a L_a$$

- See more details in [glLight\(...\)](#)

# Polygonal Shading in OpenGL ES

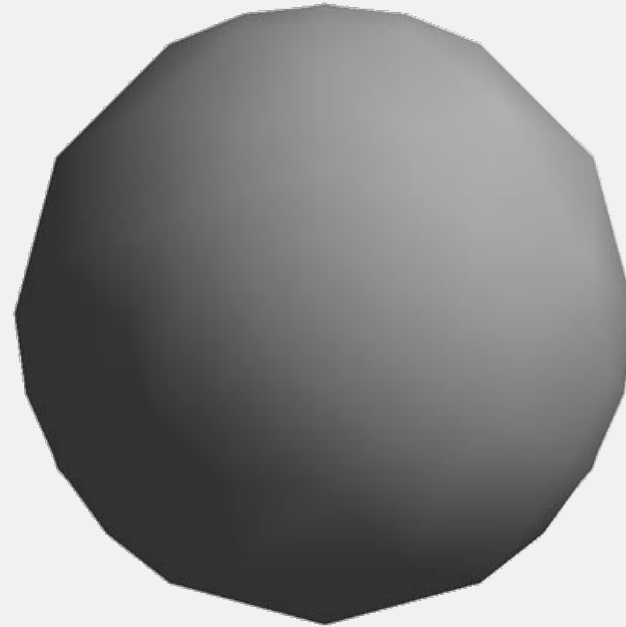
## Flat Shading

- `glShadeModel(GL_FLAT);`
- In each face, only the first normal is considered



## Smooth (Gouraud) Shading

- `glShadeModel(GL_SMOOTH);`
- In each face, per-vertex normal is considered

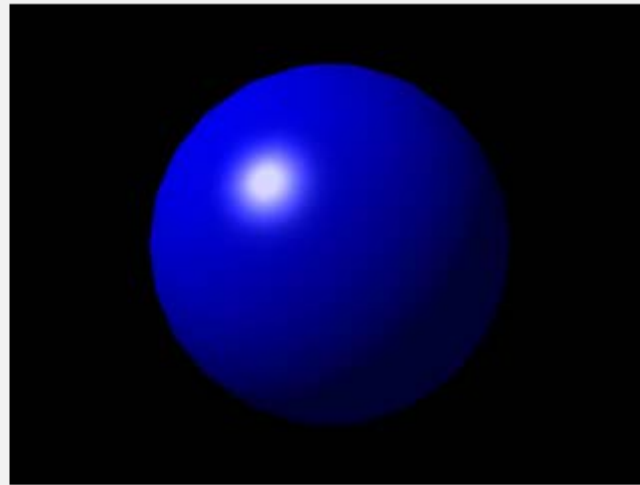


# Phong Shading

- With programmable rendering pipeline, you can use more elegant shading method, called Phong shading



FLAT SHADING



PHONG SHADING

# References

- OpenGL lighting tutorial
  - <http://www.falloutsoftware.com/tutorials/gl/gl8.htm>
- Are you interested in global illumination?
  - POV-Ray: <http://www.povray.org/>
  - PBRT: <http://www.pbrt.org/>
  - Renderman: <https://renderman.pixar.com/>

