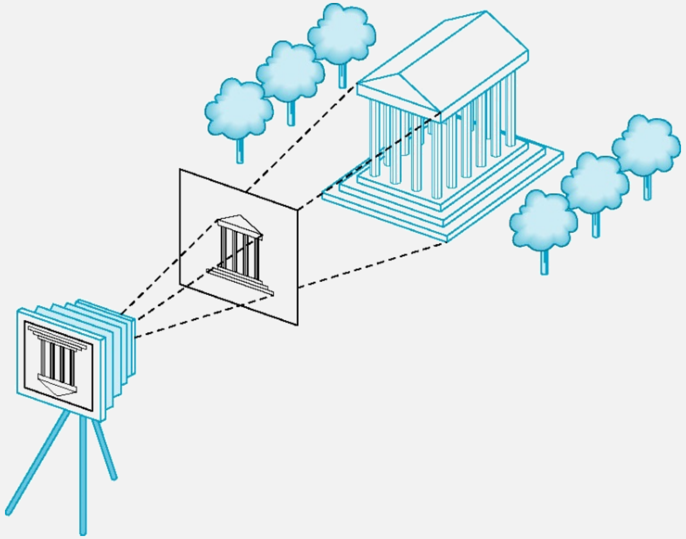
Shading

김준호
Visual Computing Lab.
국민대학교 소프트웨어학부

Elements of Image Formation

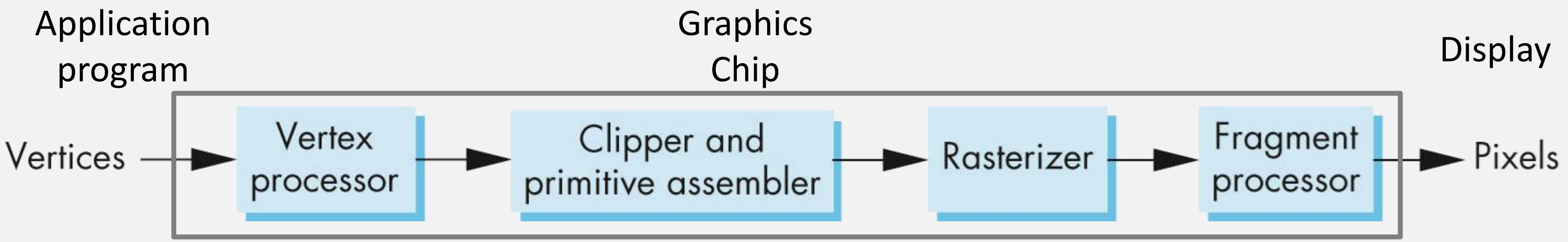
- Viewer (or camera)
  - Synthetic camera
- Objects
  - Synthetic objects
- Light source(s)
  - Synthetic lights
- Attributes
  - Material, surface normal
    for reflection model
    (i.e., light-material interaction)



Synthetic image formation
in Computer Graphics

Overview of Rendering Pipeline

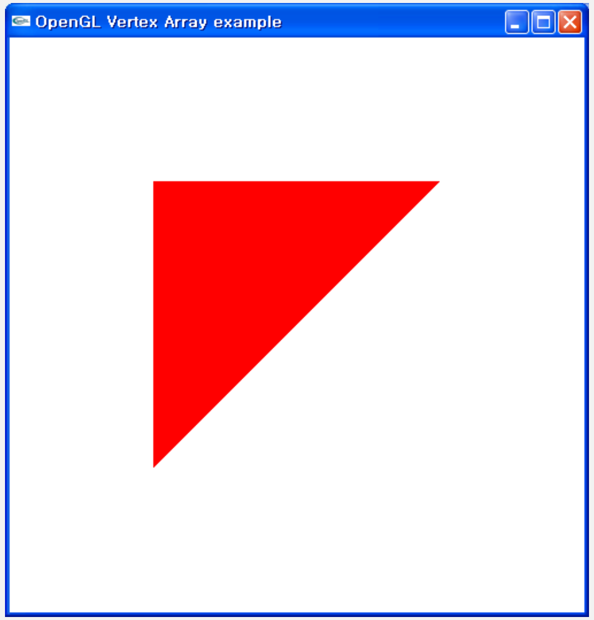
- Pipeline architecture
  - This is everything for interactive computer graphics!
    - First, we focus on the *fixed rendering pipeline*
  - Mechanism: a *state* machine
    - All information for image formations should be specified

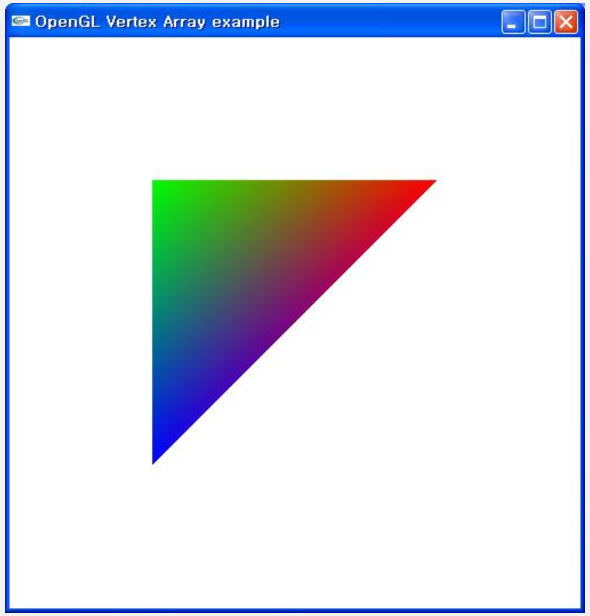


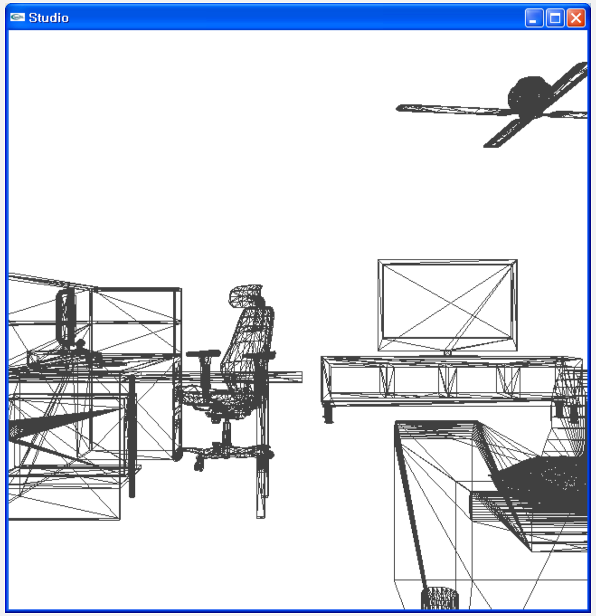
Our goal

What's going on?

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

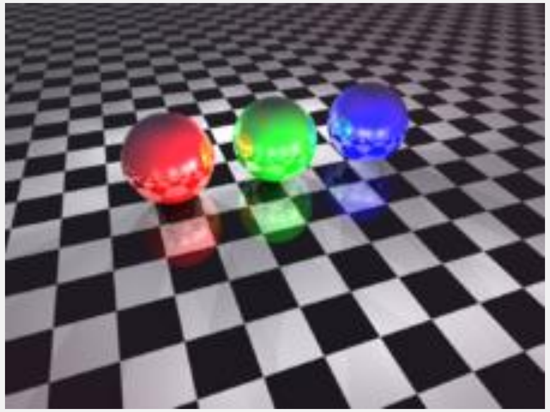


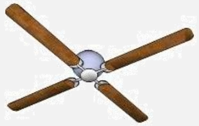




Let There Be Light





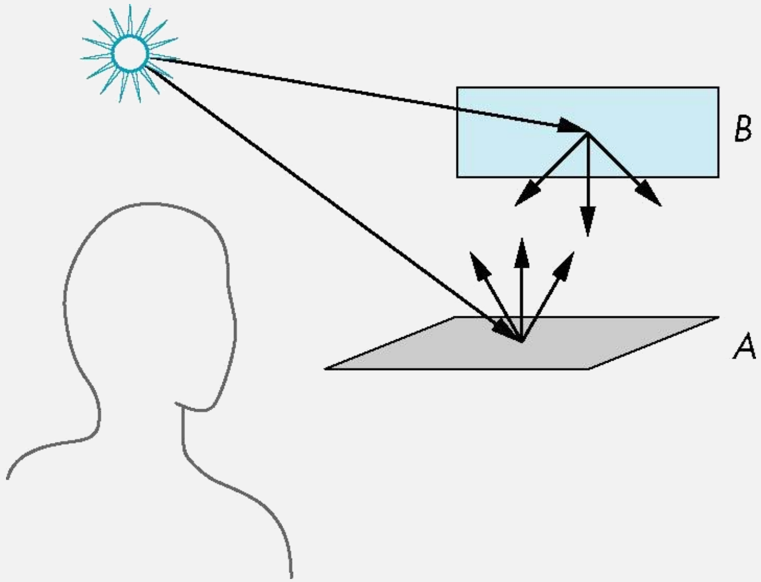



Objectives

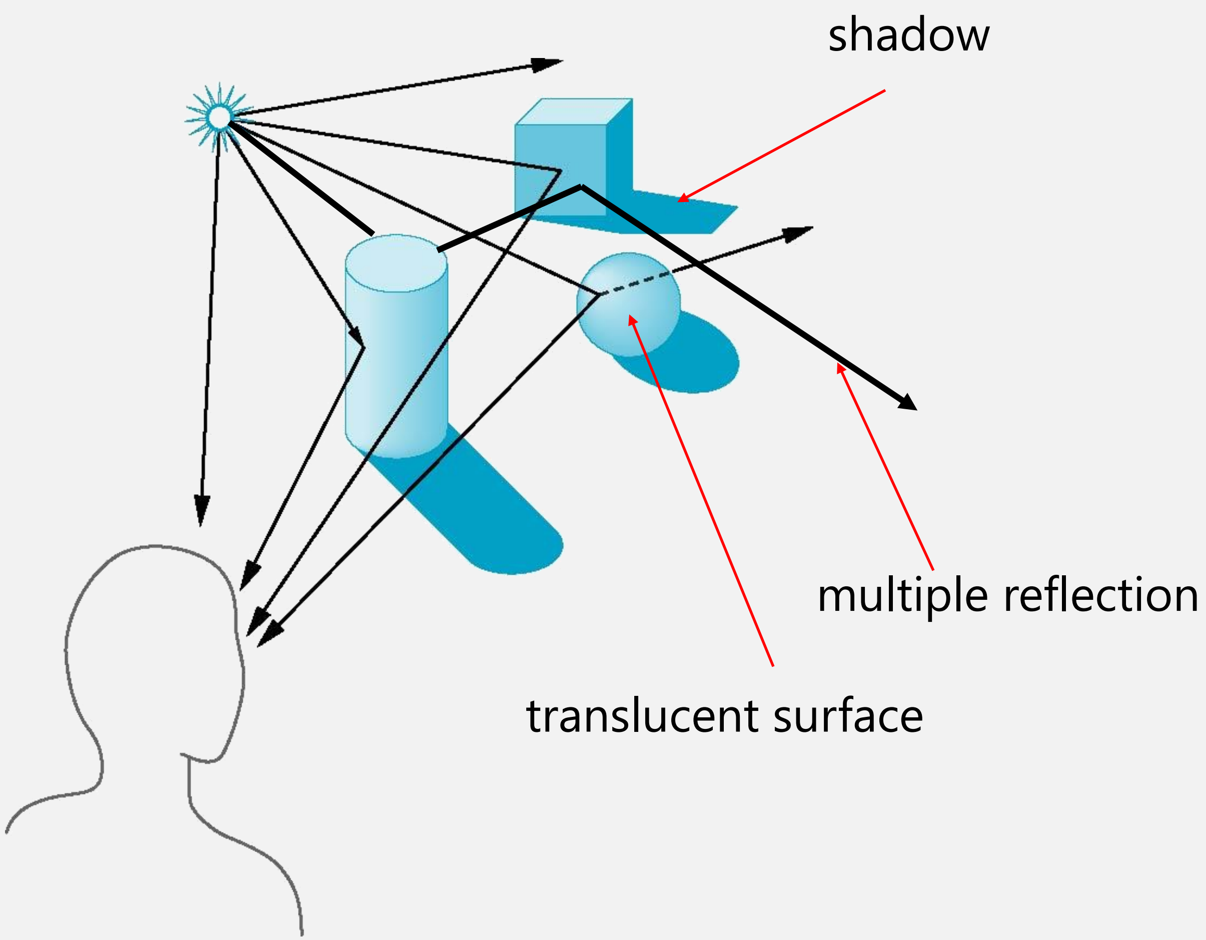
- To understand the following concepts
  - The fundamentals on Light and Material
  - Global illumination v.s. local illumination
  - Basic concepts about BRDF and Rendering Equation
  - Phong lighting model

Light

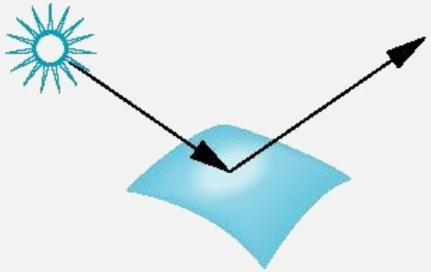
- When light strike a surface
  - Some scattered
  - Some absorbed
- Notice, the scattered light strike again other surfaces



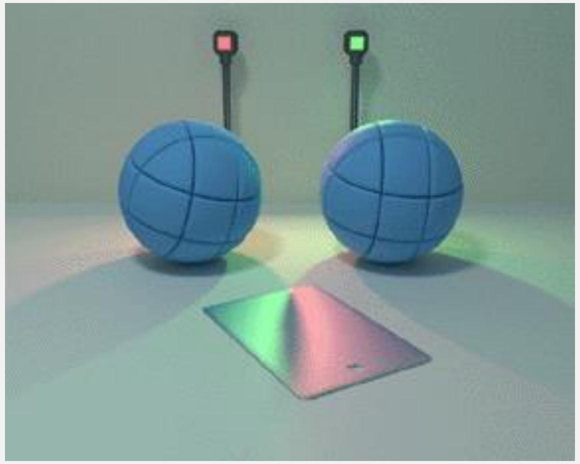
Global Effects

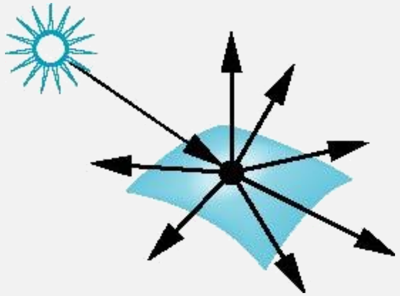


Light – Material Interaction

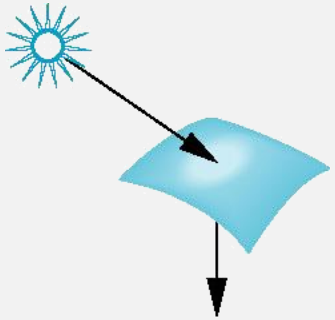


Specular surface





Diffuse surface



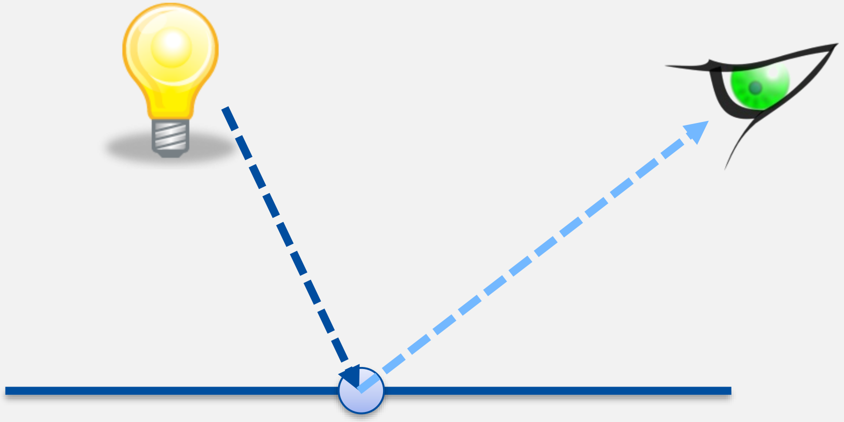
Translucent surface

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

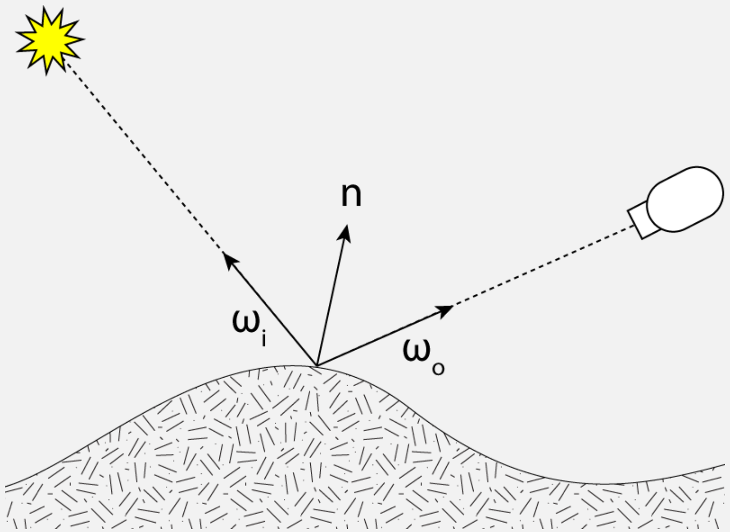
Color

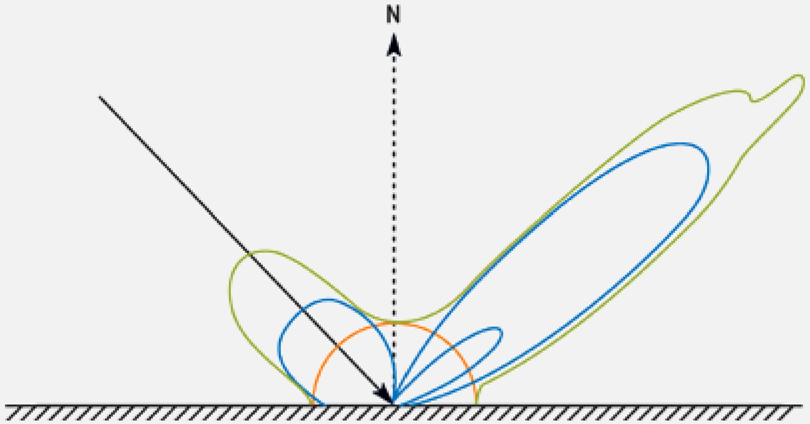
- Where does color come from?
  - Color of light sources
  - Reflection ratio of a surface point
  - Angle of light sources and eyes



Bidirectional Reflectance Distribution Function (BRDF)

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

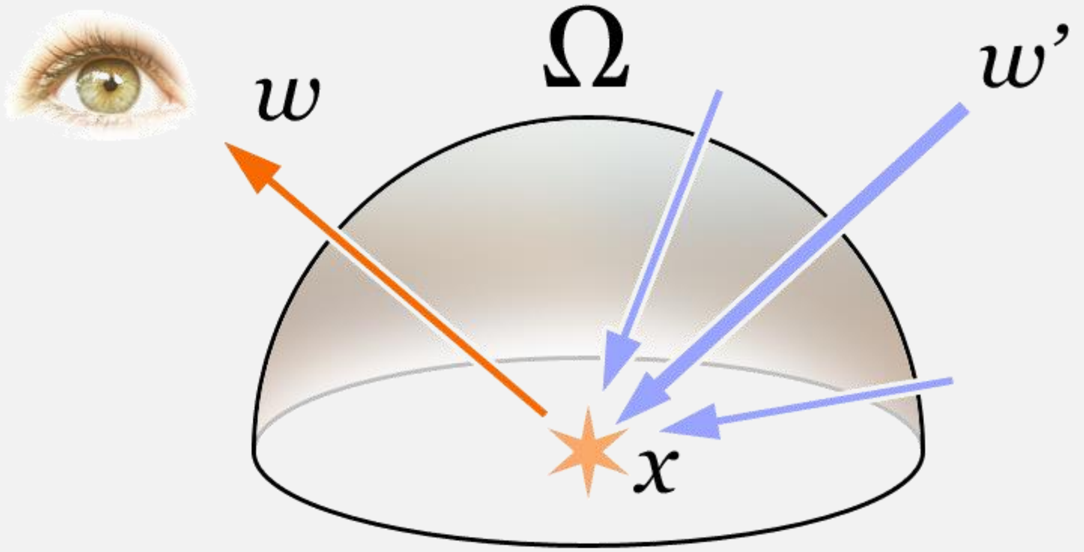




Rendering Equation

- 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

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

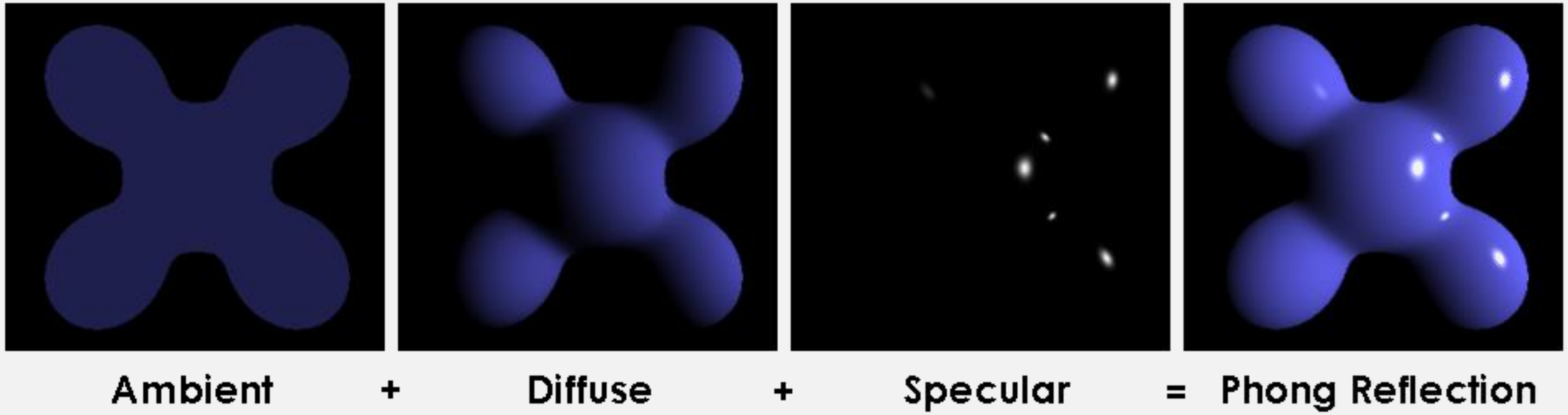
- Global lighting model
  - Radiosity, Ray tracing
    - Movie & animation use this approach
  - Non-interactive approach (not in real-time)

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 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]

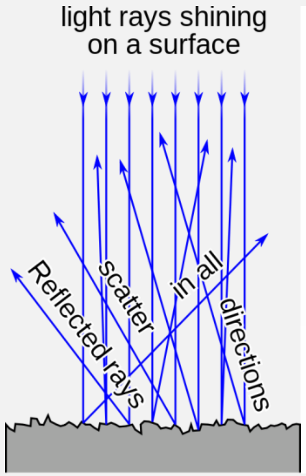
Ambient Reflection (주변광)

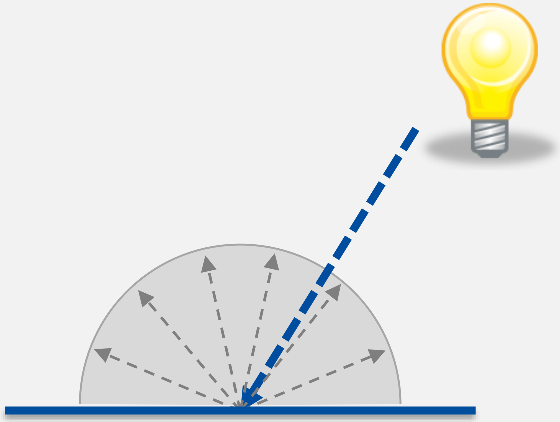
- Ambient term
  - Same at every point on the surface
  - Usually, it should be set as small as possible

$$\underbrace{I_a}_{\text{intensity of ambient light}} = \underbrace{\kappa_a}_{\text{ambient reflection coefficient}} \underbrace{L_a}_{\text{illumination of ambient light}}$$

Diffuse Reflection (난반사광)

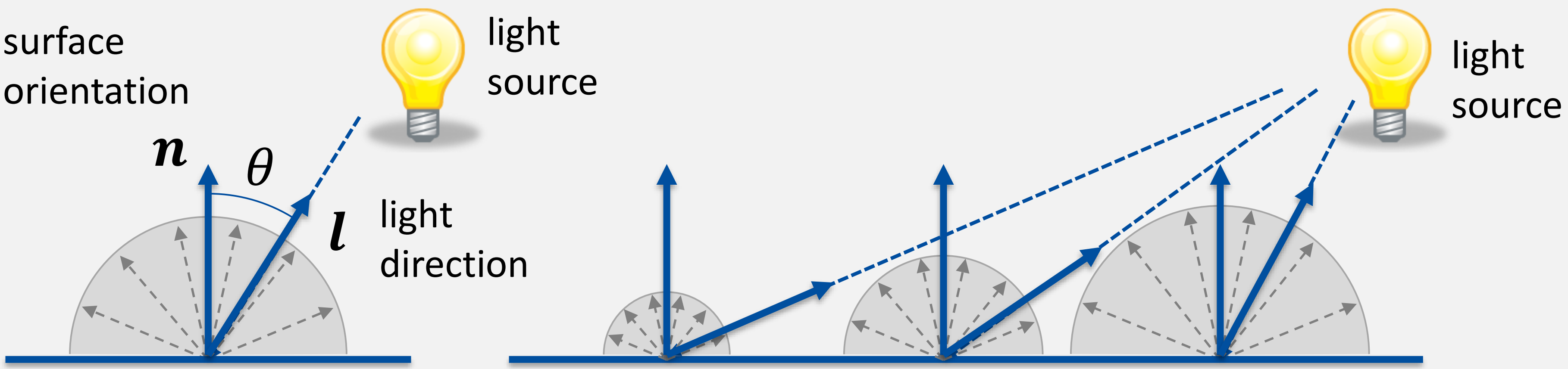
- In physics, an incident ray is reflected at many angles
- In graphics, we consider diffuse reflection as lambertian reflectance
  - The apparent brightness of a lambertian surface to an observer is the same regardless of the viewer's angle of view





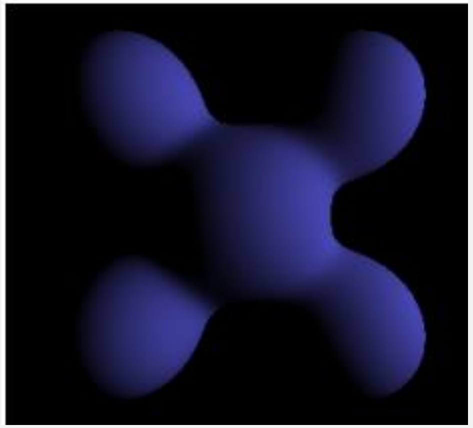
Diffuse Reflection (난반사광)

- The brightness of diffuse reflection is related to the angle θ 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



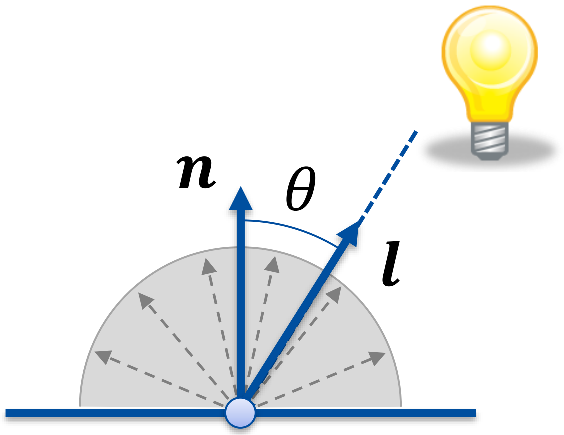
$$\underbrace{I_d}_{\text{intensity of diffuse light}} = \underbrace{\kappa_d}_{\text{diffuse reflection coefficient}} \max\Big(\underbrace{(\boldsymbol{l} \cdot \boldsymbol{n})}_{\text{angle}}, 0\Big) \underbrace{L_d}_{\text{illumination of diffuse light}}$$

~~Diffuise Reflection – OpenGL 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(…);
```



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

intensity of diffuse light

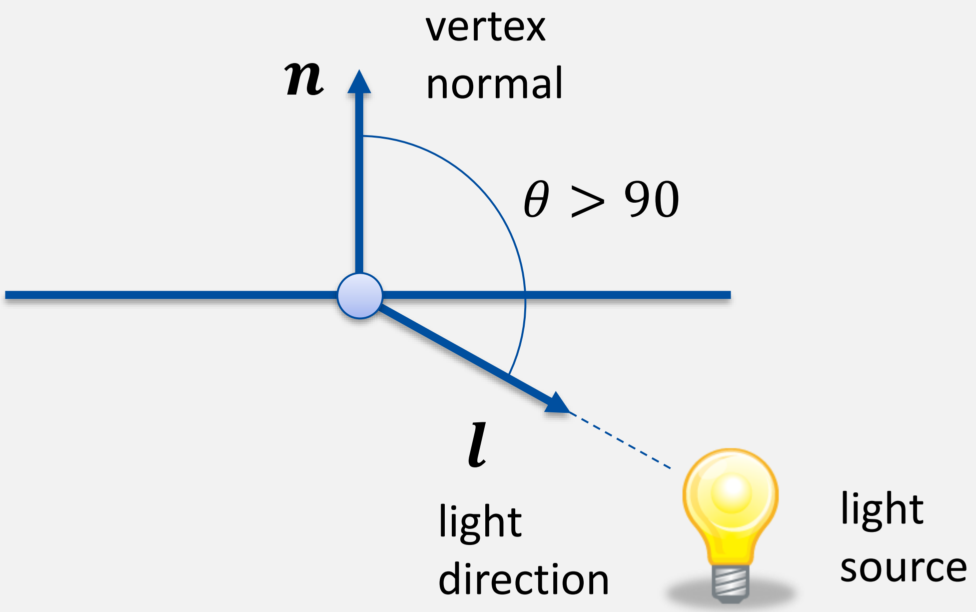
diffuse reflection coefficient

angle

illumination of diffuse light

Why using max(·,·)?

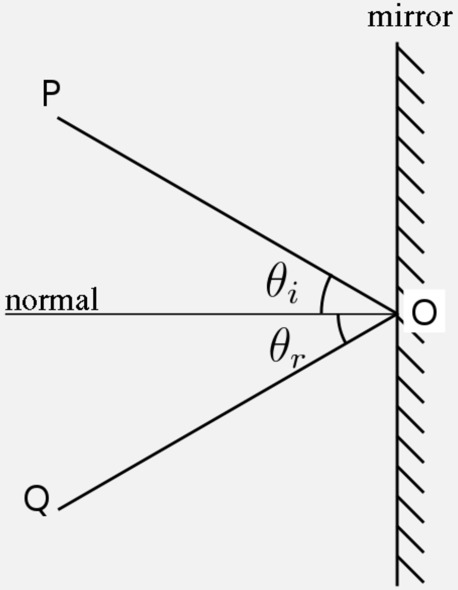
- What is the meaning of $\max\big((\boldsymbol{l} \cdot \boldsymbol{n}), 0\big)$?
  - If $\boldsymbol{l} \cdot \boldsymbol{n} \geq 0$, then light hits the surface on its **front** face
  - If $\boldsymbol{l} \cdot \boldsymbol{n} < 0$, then light hits the surface on its **back** face
  - It means, we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



Specular Reflection (정반사광)

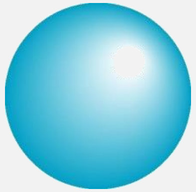
- Specular reflection is the mirror-like reflection of light from a surface, in which light from a single incoming direction is reflected into single outgoing direction

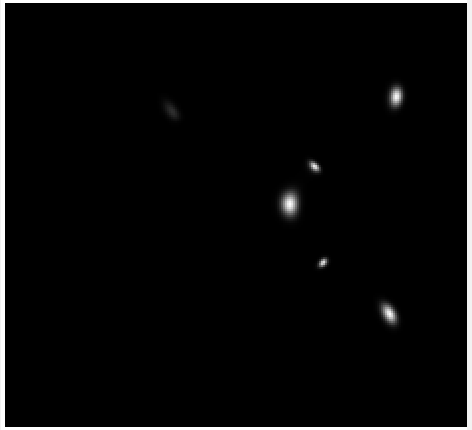
[Image from Top Gear UK]



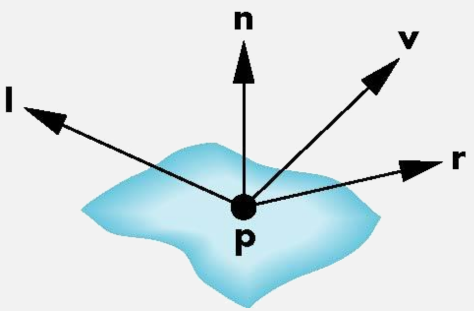
Specular Reflection (정반사광)

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



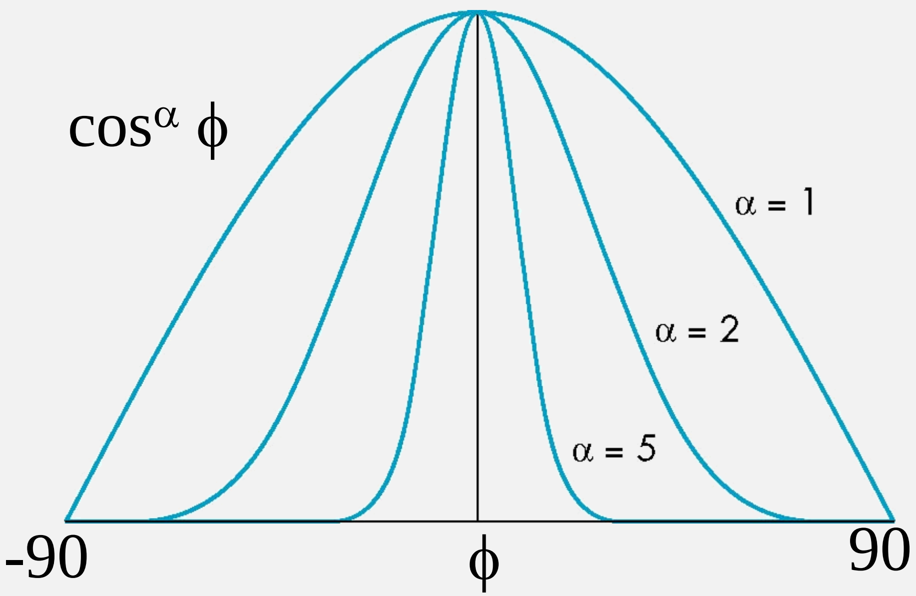


$$\underbrace{I_s}_{\text{intensity of specular light}} = \underbrace{\kappa_s}_{\text{specular reflection coefficient}} \max\Big(\underbrace{(\boldsymbol{r}\cdot\boldsymbol{v})}_{\text{angle}}^{\alpha}, 0\Big) \underbrace{L_s}_{\text{illumination of specular light}}$$

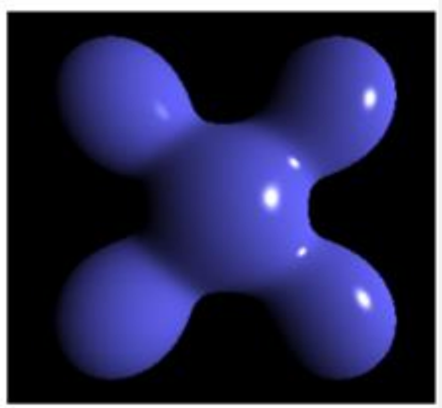


Shininess Coefficient in Specular Reflection

- Values of α between 100 and 200 correspond to metals
- Values of α 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

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

Constants for light attenutations can be set by ~~glLight~~();

- a: GL_CONSTANT_ATTENUATION
- b: GL_LINEAR_ATTENUATION
- c: GL_QUADRATIC_ATTENUATION

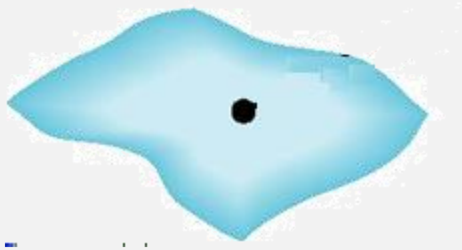
Phong Reflection Model

- 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

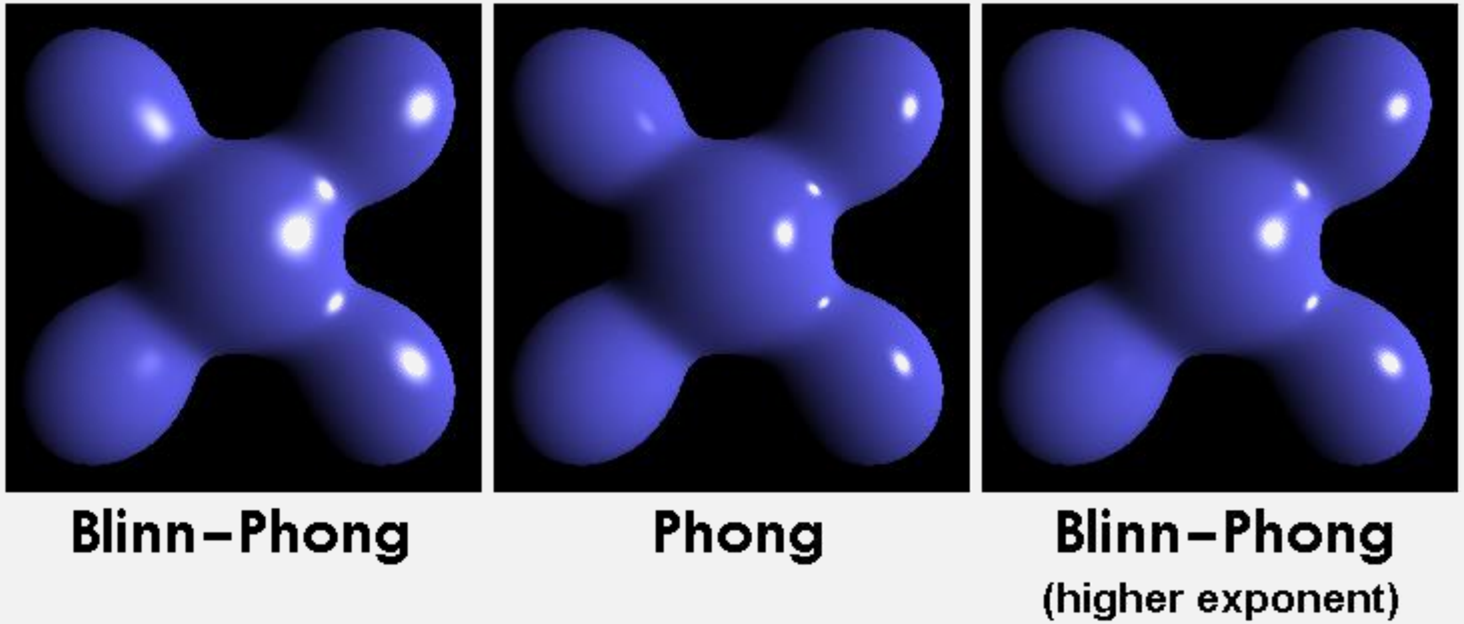


$$I = \sum_i \left(I_{i,a} + I_{i,d} + I_{i,s}\right) + I_{global,a}$$

- 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

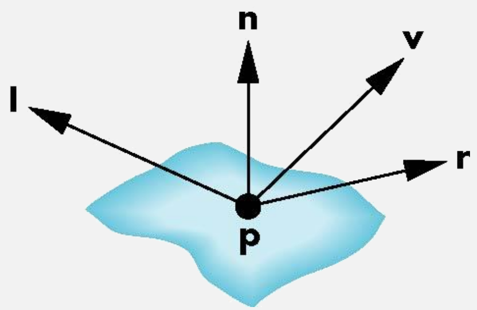
- Phong-Blinn reflection model
  - A variation of Phong reflection model
  - In graphics HW, Phong-Blinn reflection model is implemented, instead of Phong reflection model



Phong-Blinn Reflection Model

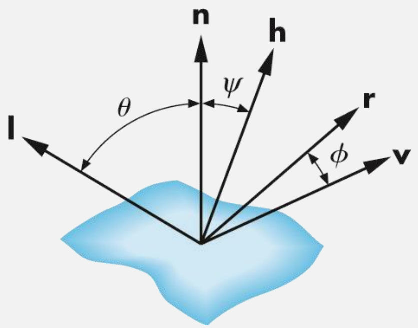
- Main purpose of Phong-Blinn reflection
  - To reduce computation time, by eliminating per-vertex computations
    - Especially, when we consider the viewer and light are treated to be at infinity

Phong reflection model



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

Phong-Blinn reflection model

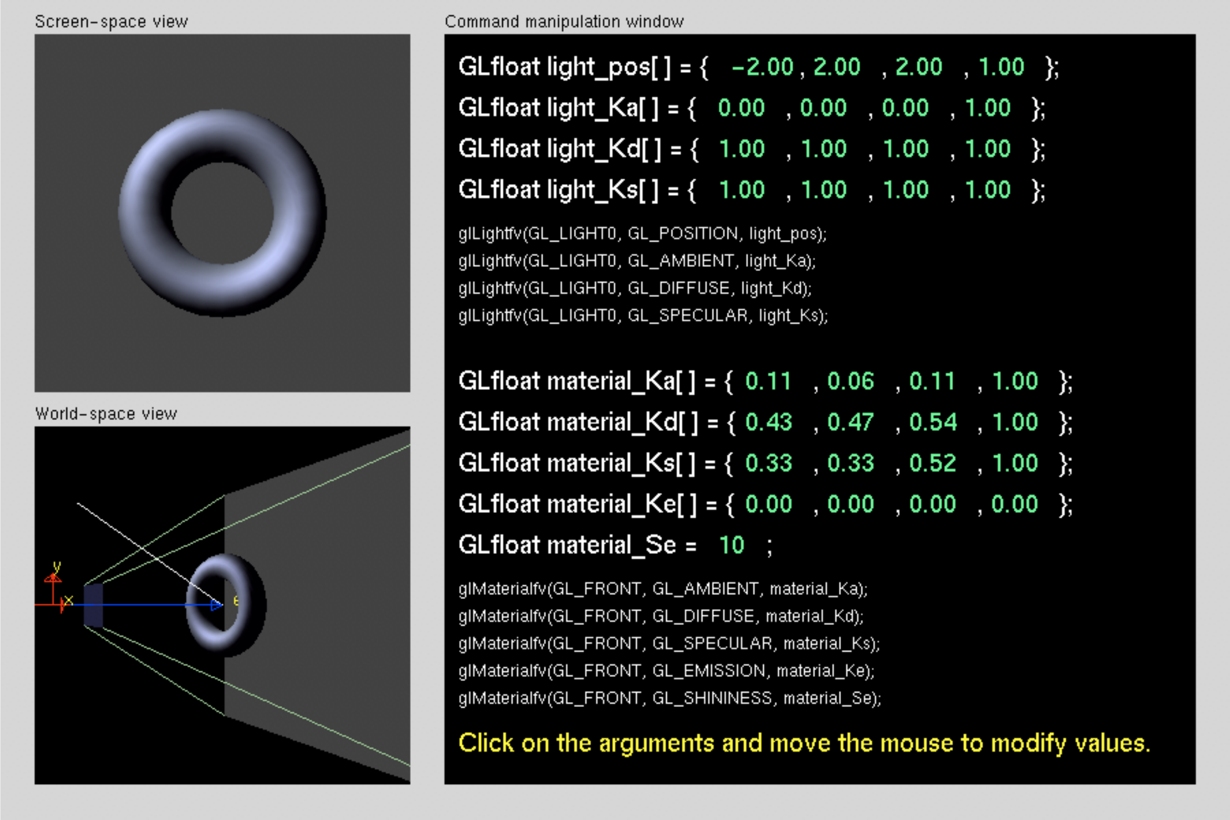


$$I_s = \kappa_s \max\left((\boldsymbol{n} \cdot \boldsymbol{h})^{\alpha}, 0\right) L_s$$

where the half vector $\boldsymbol{h} = \dfrac{\boldsymbol{l} + \boldsymbol{v}}{|\boldsymbol{l} + \boldsymbol{v}|}$

Demo – Phong-Blinn Reflection Model

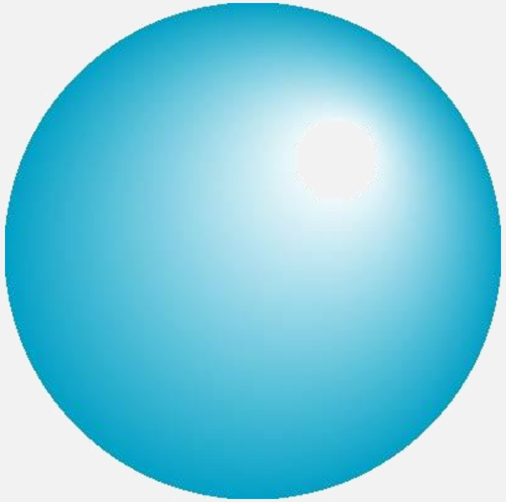
- Tutorial from Nate Robins
  - http://user.xmission.com/~nate/tutors.html



Computing Per-vertex Normals

Computing Per-Vertex Normal w/ Algebraic Surfaces

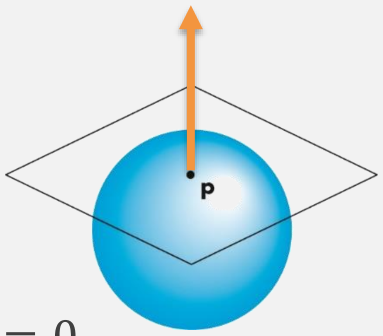
- Algebraic surfaces
  - We can describe a set of points on the surface in an algebraic form
    - Implicit equation
      - Ex) $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
    - Parametric equation
      - Ex) $x(u, v) = \cos u \sin v$
        $y(u, v) = \cos u \cos v$
        $z(u, v) = \sin u$



Computing Per-Vertex Normal w/ Algebraic Surfaces

- Normal w/ Implicit Equation

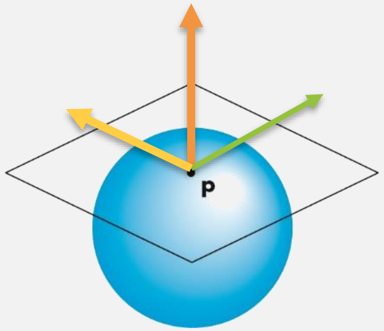
$$\boldsymbol{n}_p = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \partial f / \partial z \end{bmatrix} = \nabla f$$



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

- Normal w/ Parameteric Equation

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



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

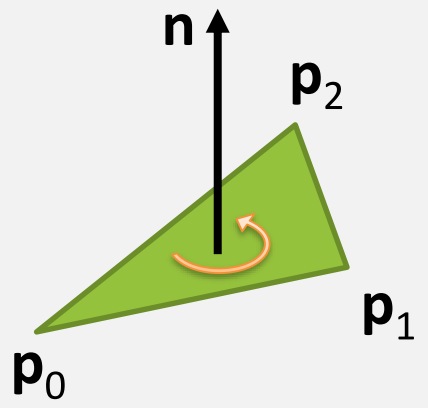
Computing Per-Vertex Normal w/ Polygonal Model

- Compute each face normal

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

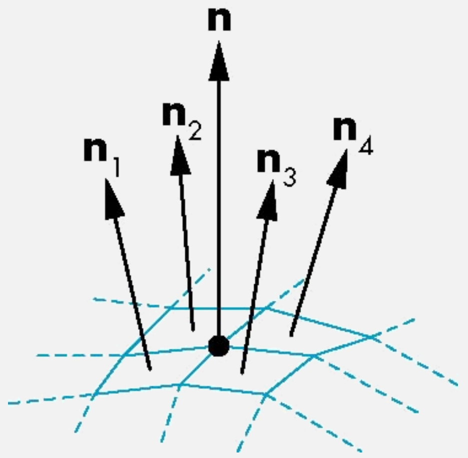
Note:

$$\boldsymbol{u} \times \boldsymbol{v} = \begin{bmatrix} u_2v_3 - v_2u_3 \\ u_3v_1 - v_3u_1 \\ u_1v_2 - v_1u_2 \end{bmatrix}, \text{ where } \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \text{ and } \boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$



- Averaging incident face normals

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

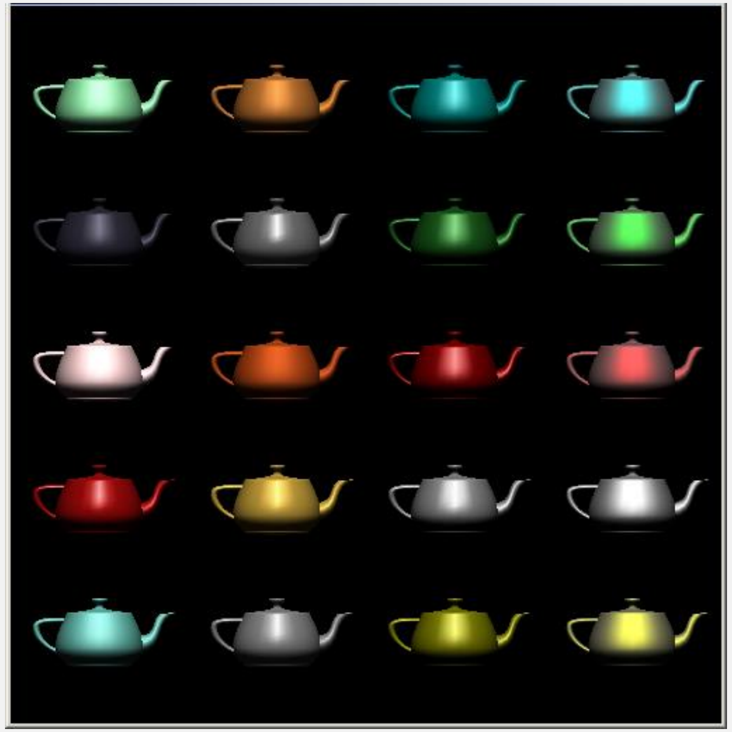


Phong Reflection Model

Light-Material Interaction in OpenGL

Light-Material Interactions in OpenGL

- 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

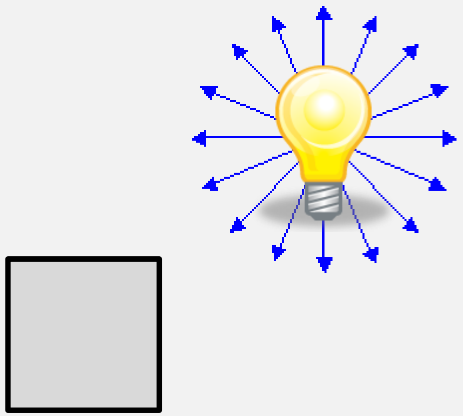
- The current material contains the coefficients for each reflection model
  - Coeff. for ambient reflection: $\kappa_a = (\kappa_r, \kappa_g, \kappa_b, \kappa_a)_a$
  - Coeff. for diffuse reflection: $\kappa_d = (\kappa_r, \kappa_g, \kappa_b, \kappa_a)_d$
  - Coeff. for specular reflection: $\kappa_s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

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

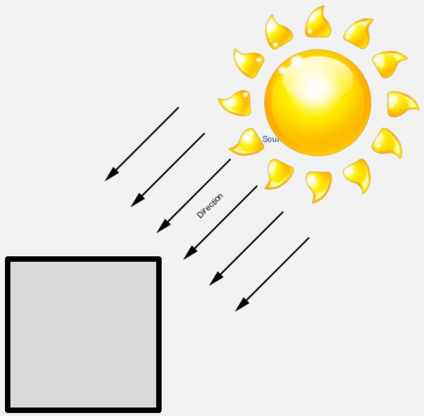
- See more details in ~~glMaterial~~()

Types of Light Source

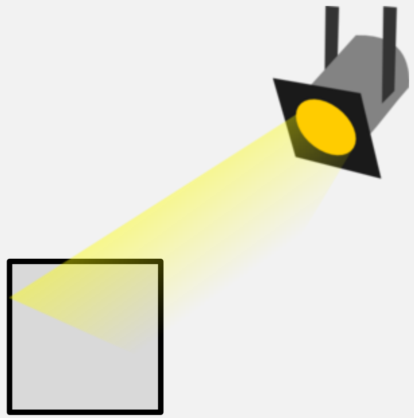
- Point light
  - Emitting light in all directions from one single point
- Directional light
  - Emitting light rays in a parallel direction
- Spot light
  - Emitting light in focus direction of the cone



Point light



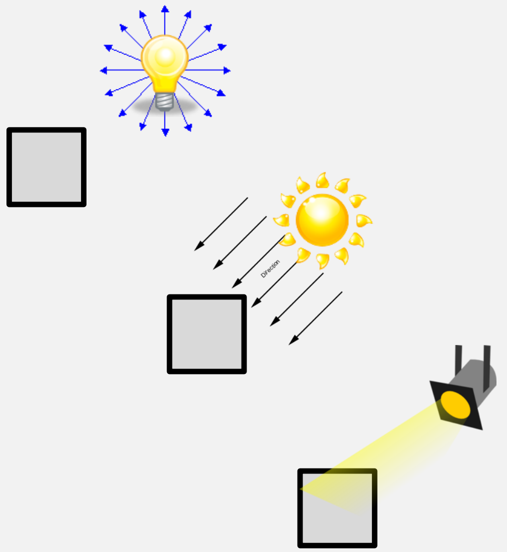
Direct light

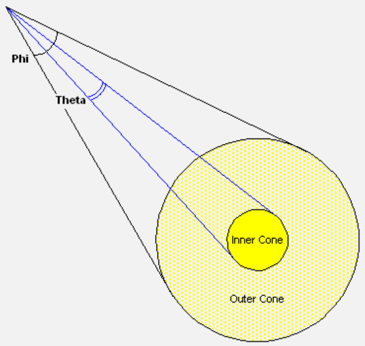


Spot light

Lights in OpenGL

- 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





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

- See more details in ~~glLight~~(…)

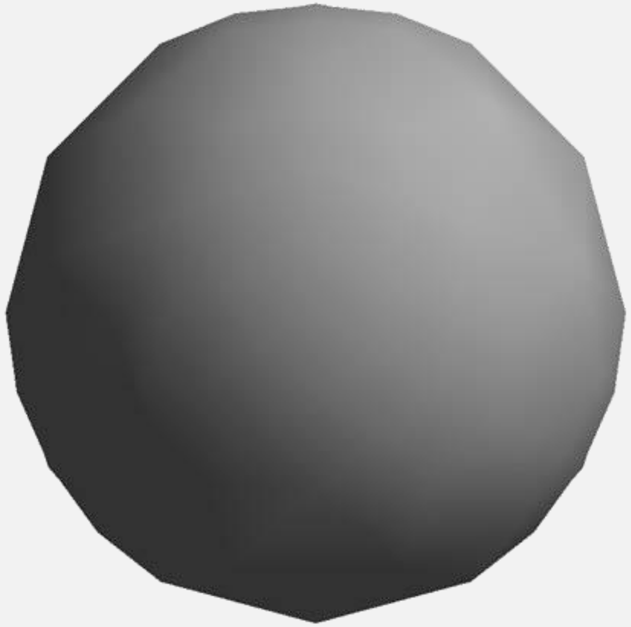
~~Polygonal Shading in OpenGL 1.x~~

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



Per-vertex Lighting v.s. Per-pixel Lighting

- With programmable rendering pipeline, you can get more elegant smooth shading results
  - ~~Per-vertex lighting with fixed-rendering pipeline: Gouraud shading~~
  - Per-fragment lighting with programmable rendering pipeline: Phong shading

Per-vertex lighting

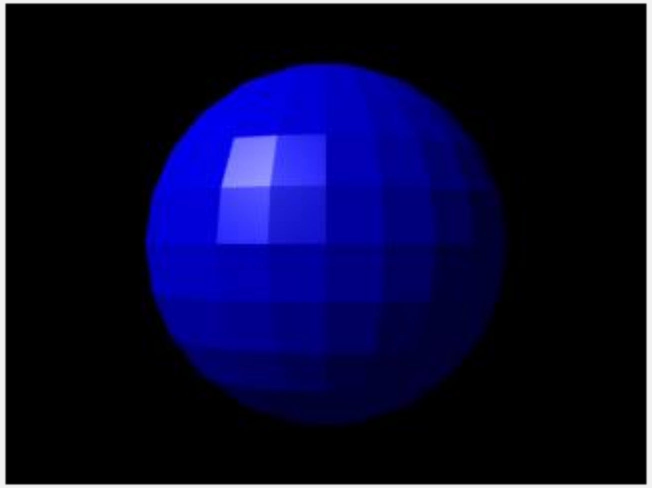
(Gouraud shading)

per-fragment lighting

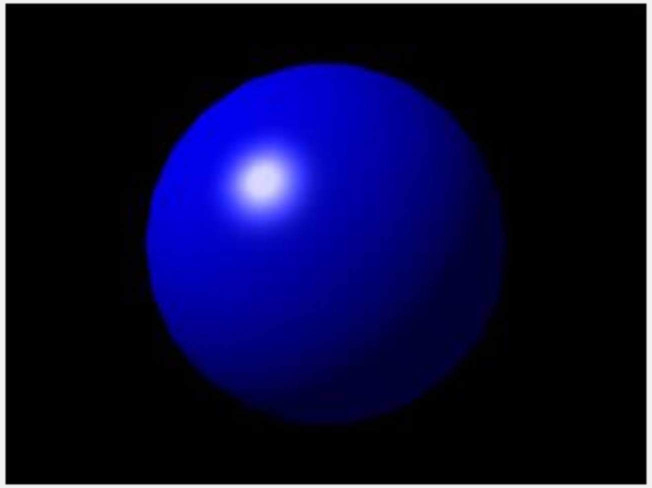
(Phong shading)

Per-vertex Lighting v.s. Per-pixel Lighting

- With programmable rendering pipeline, you can get more elegant smooth shading results
  - ~~Per-vertex lighting with fixed-rendering pipeline: Gouraud shading~~
  - Per-fragment lighting with programmable rendering pipeline: 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/