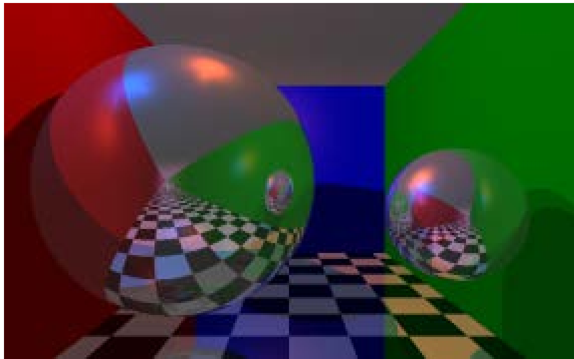
CSI 4105 Computer Graphics

Spring 2017

Lecture 7: Illumination (Part I)

Seon Joo Kim

Yonsei University



Objectives

- To introduce the standard illumination model
  - (Phong illumination)
  - Ambient + Diffuse + Specular
- Discuss shading models
  - Flat, Gourard, Phong
- Discuss relevant OpenGL commands
  - Lighting in OpenGL

Illumination Model
(Part 1)

Photo realism in CG

<u>Two key elements are important to produce realistic CG</u>

(1) Accurate graphical representations of the object
(ie, the 3D model must be good)

(2) Good physical descriptions of the lighting effects in the scene
(ie, lighting must look natural and realistic)

Accurate illumination is often very hard. Think about the real world and how light and objects interact with one another -- very, very complex.

In computer graphics, we often greatly simplify light interaction into models that are loosely derived from physical laws that describe light interactions with surface materials.

Terms

- **Illumination model**

  refers to the mathematic model for calculating light intensity at a single surface point

- **Shading**

  the procedure for applying a lighting model to obtain pixel intensities for surfaces

- **Light Source**

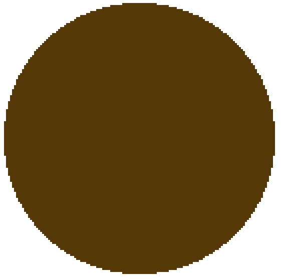
  describes how light is introduced into the scene

- **Surface Materials**

  parameters assigned to a surface to determine how it will respond to the light

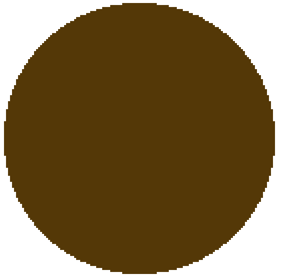
Why we need shading

- Suppose we build a model of a sphere using many polygons and color it with **`glColor`**. We get something like -- not very impressive.

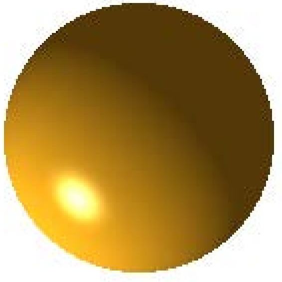


Why we need shading

- Suppose we build a model of a sphere using many polygons and color it with `glColor`. We get something like -- not very impressive.

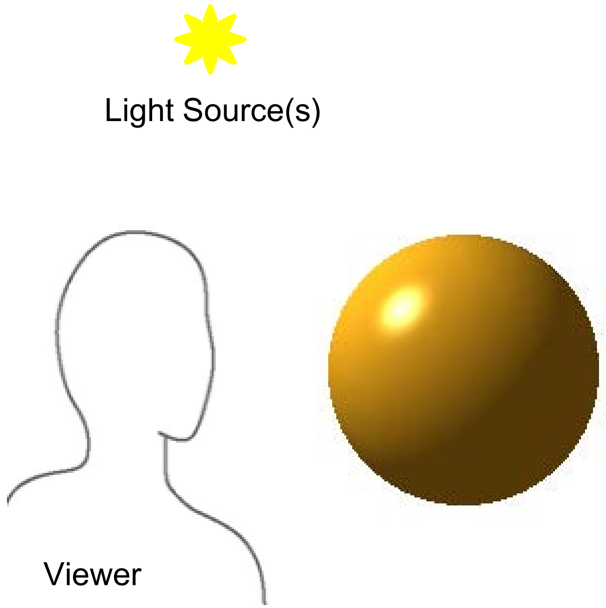


- How can we get something like this, instead?

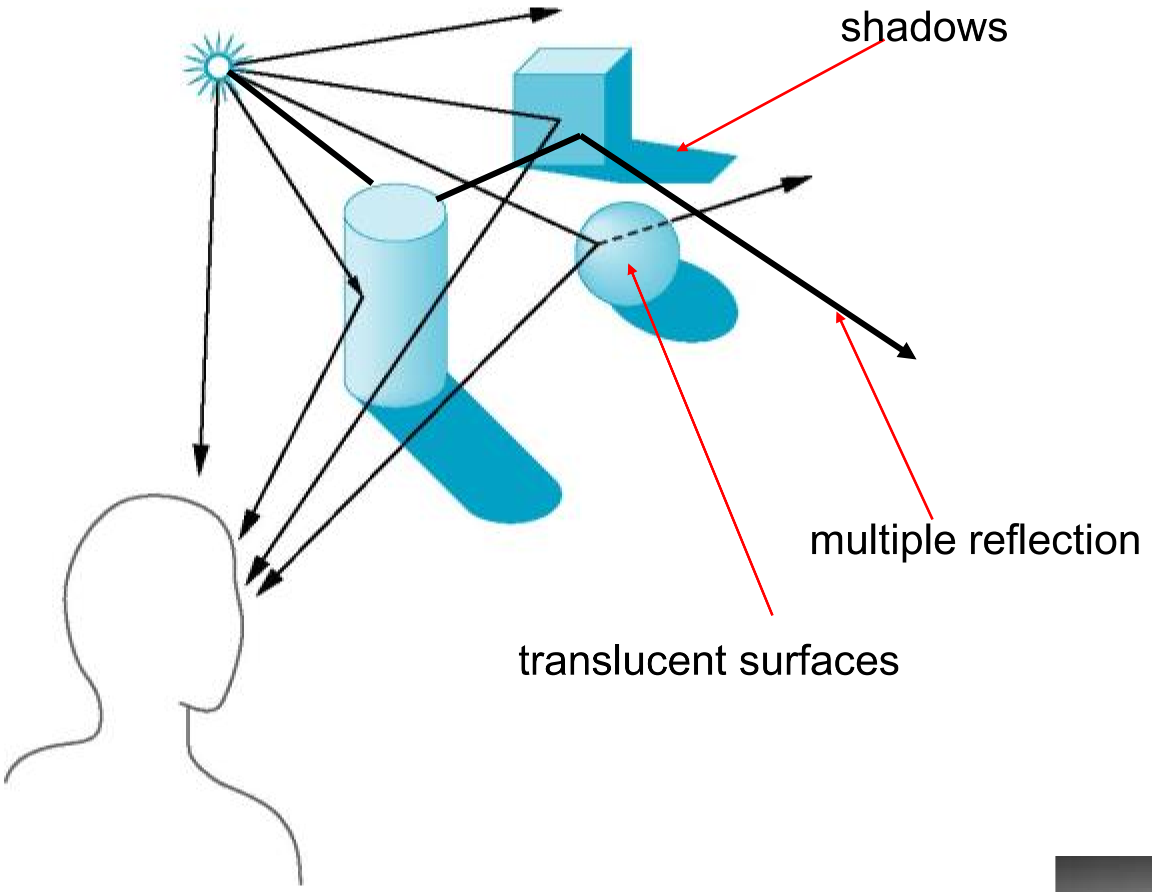


Shading Considerations

- Light-material interactions cause each point to have a different color or shade
- We will need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation



Real Lighting has “Global Effects”



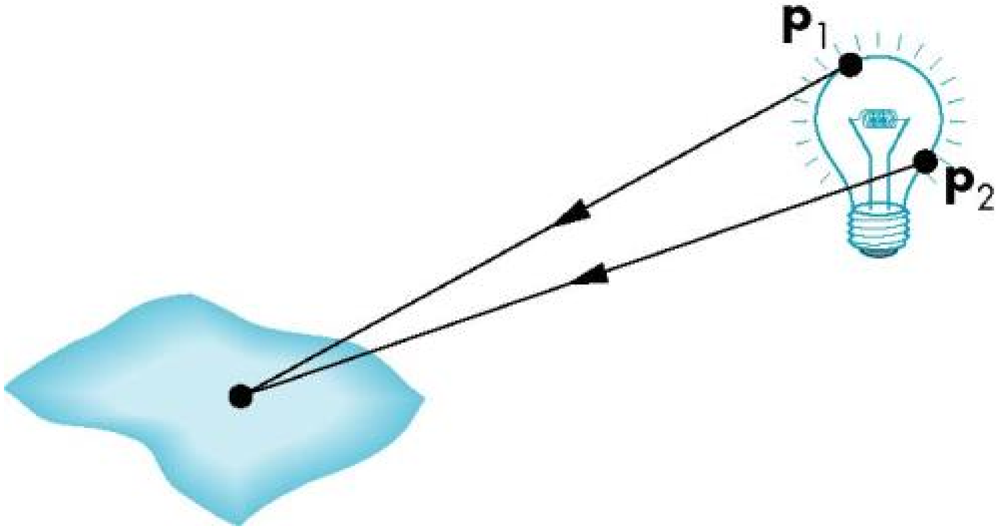
Light source illuminates objects. Light from the objects also “bounces” around the scene and illuminates objects. We have reflections, shadows, translucent materials (i.e. light passes through the materials), and so on. “Real Lighting” is very complex.

Local vs Global Rendering

- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with the rendering pipeline model (used by graphics cards) which shades each polygon independently (local rendering)
  - Also, global shading is slow to compute (currently requires ray-tracing)
- We will have to use an “approximation” of the global lighting
- However, in computer graphics, especially real time graphics, we are happy if things “look right”

Light Sources

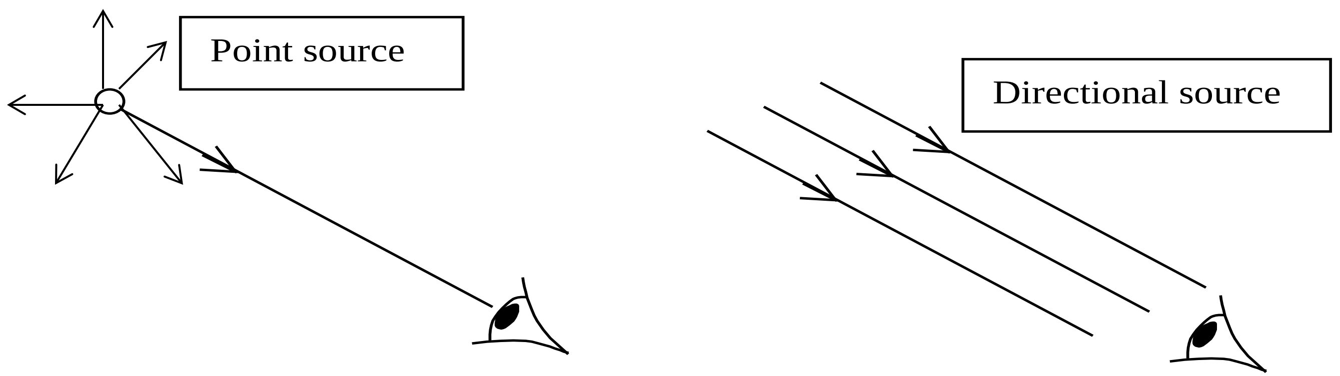
General light sources are difficult to work with because we must integrate light coming from all points on the source



Ideal Light Sources (Simplified model)

Light Sources:

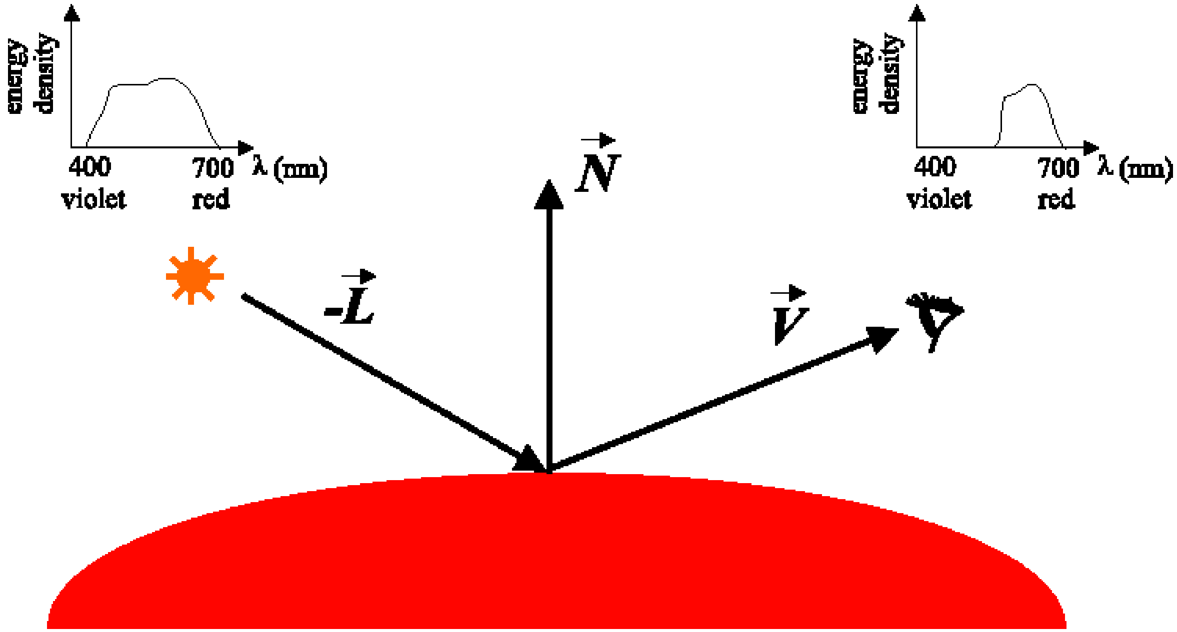
- In computer graphics, two types of light sources are commonly used
  - **<u>point source</u>** The light source is a zero-volume point
  - **<u>directional source</u>** The point source that are infinite far away

- 

- Both types of light sources are *ideal* light sources (i.e. not realistic)
- But they are easy for computation.

Surface Reflection from Light

- Although we modeled different light source in computer graphics, we seldom draw light sources.
- What we draw is usually the surfaces that reflect light.



Basic Illumination Models

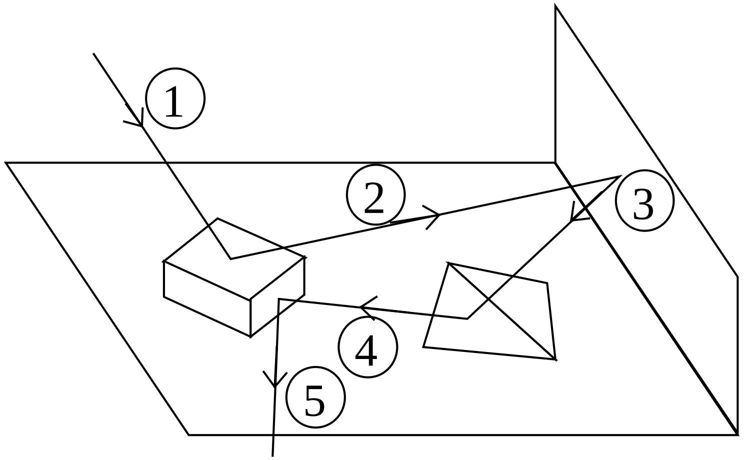
- **Ambient Light**
  - This is *non-directional* light that is “magically” present in the scene
  - It is unrealistic light, but is used to give some lighting effect to objects that are not directly illuminated by a light source

- **Diffuse Reflection**
  - Reflected light will depend on surfaces position to the light source
  - A reasonable approximation to how light interacts with an object

- **Specular Reflection**
  - Reflected light will depend on surface, light position, and viewing position
  - Sometimes we call these the “highlights”
    - Shiny objects (like plastic) have lots of specular reflections
  - When combined with diffuse reflections adds realism to an object

Ambient (Indirect Illumination)

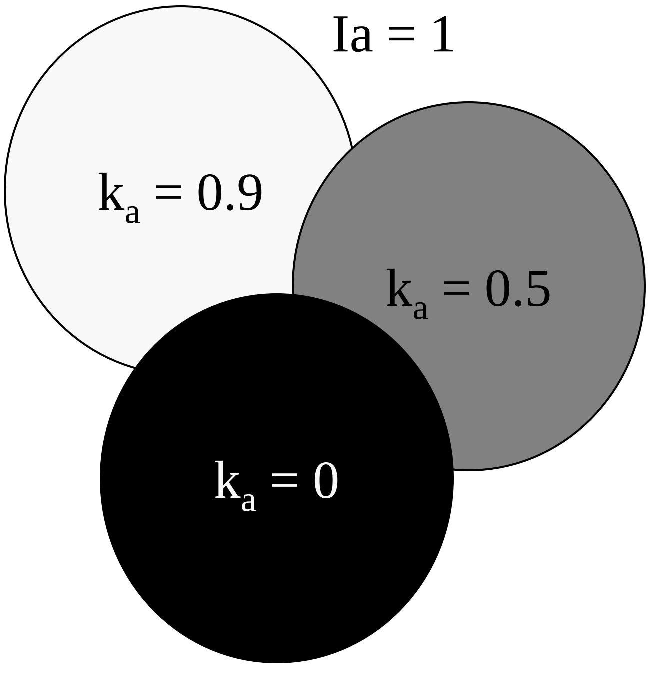
- Some surfaces are illuminated even it is in shadow. Why?
- There is indirect lighting (background lighting) reflected from other surfaces



- Each surface illuminated by all point and extended sources becomes itself, a source of light for illumination off all other line-of-sight surface of the scene.
- Each of these surfaces, in turn, reflect light to other surfaces, including the original one, thus achieving an “infinite regression” of reflections and illumination.
- Heuristic: Simply assume the indirect lighting is constant (same to all objects in the scene) in most graphics systems.

Modeling Ambient Light

Example



I_a = ambient light

Material property:
k_a = [0-1]

k_a determines how much of the ambient light the surface will reflect.

Illumination model

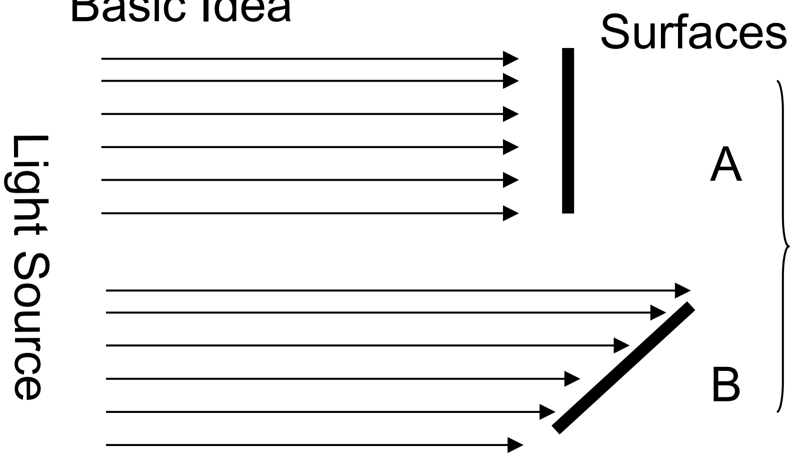
$$I = k_a I_a$$

In the above example. Ambient light is Ia = 1
[assume grayscale color now,where 0 is black and 1 is white]

Each object's material has a different k_a

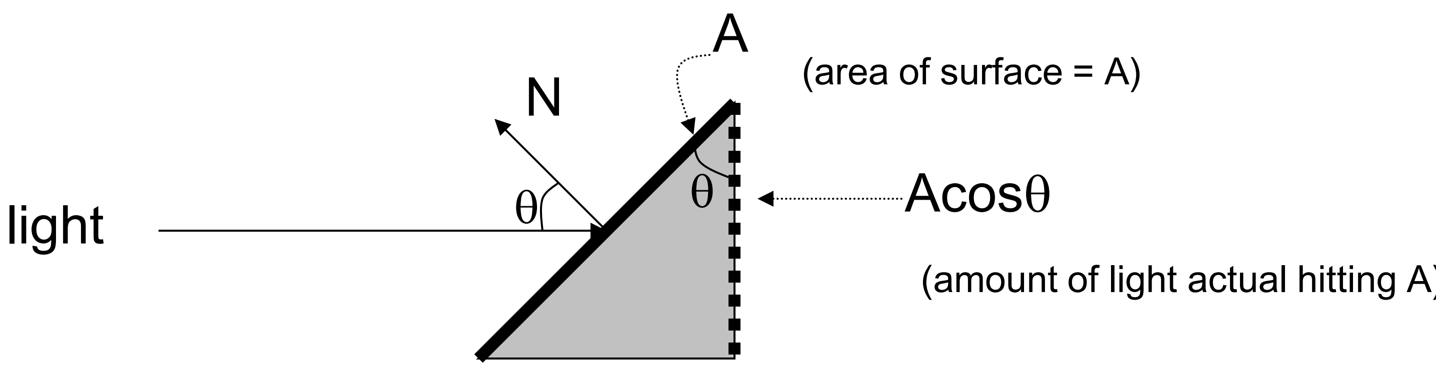
Diffuse Reflection

Basic Idea



Surfaces are A & B are equal size.

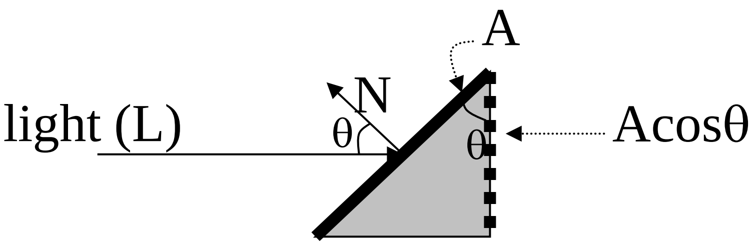
A surface perpendicular to the direction of the light (A) is more
illuminated than an equal-sized surface at an oblique angle (B), to the incoming light direction.



θ angle of incidence between light and surface

Diffuse Reflection Model

Note, we draw the light vector from the light coming to the surface, however, when we compute the dot-product, we use a vector whose direction is from the surface to the light-source. This keeps the value positive.



θ angle of incidence between light and surface

$$I_{l,diff} = k_d I_l \cos\theta$$

$$I_{l,diff} = k_d I_l (N \cdot L)$$

I_l = is a directional light source

encodes two values

(1) some illumination power

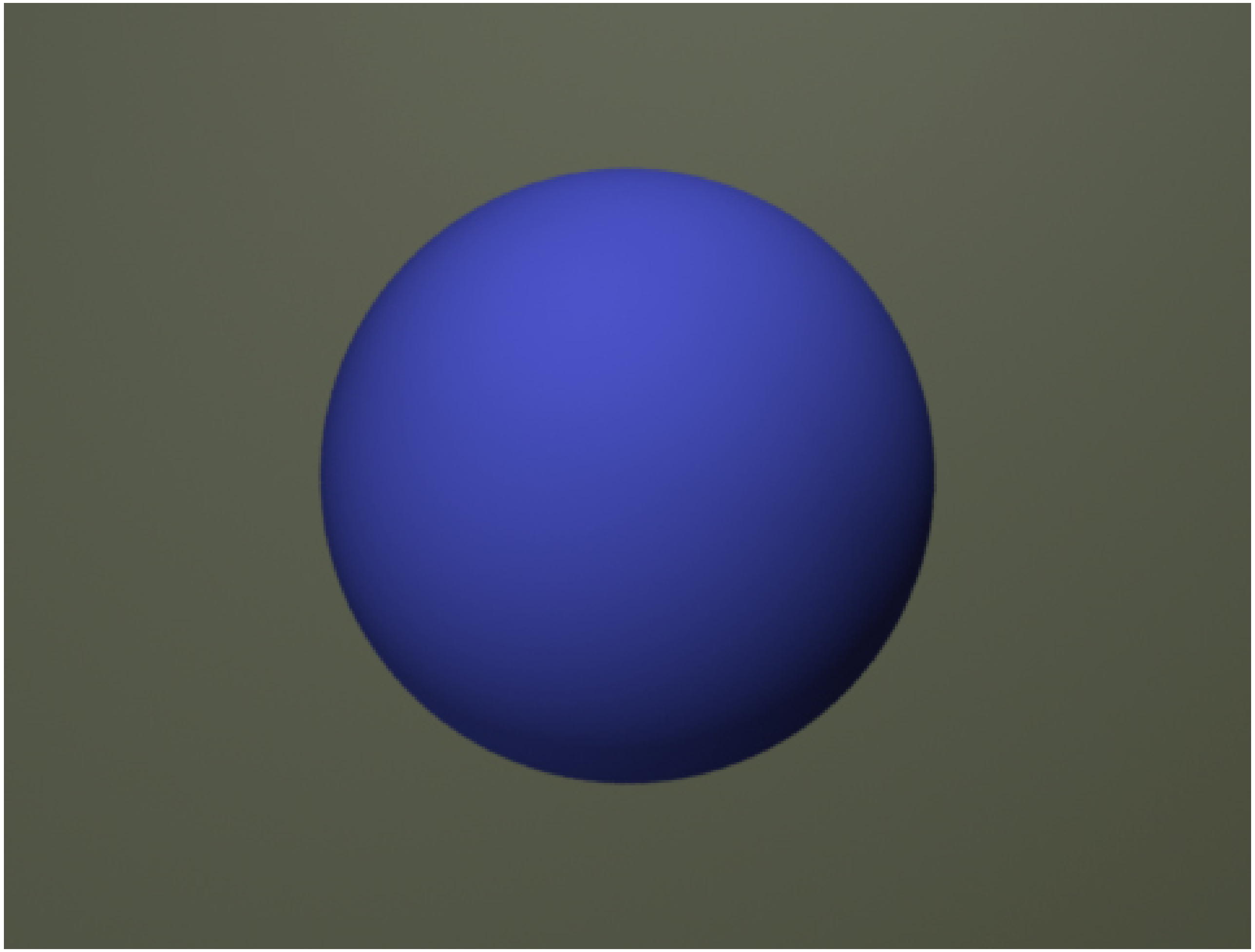
(2) direction of the light L [unit vector]

Material property:

k_d = [0-1]

k_d determines how much of the diffuse light the surface will reflect.

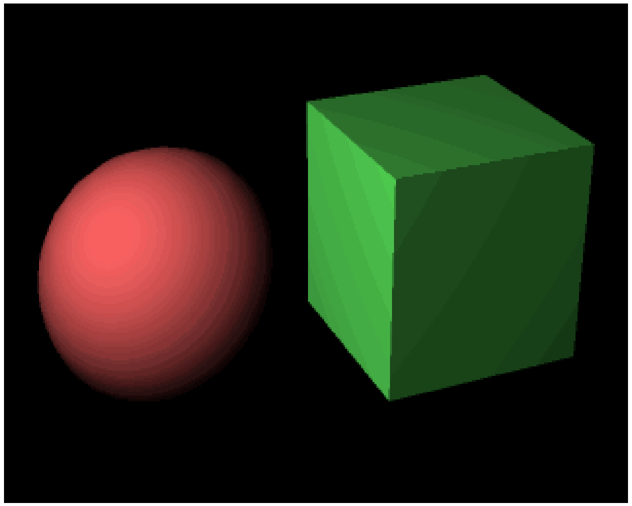
Diffuse Reflection Example



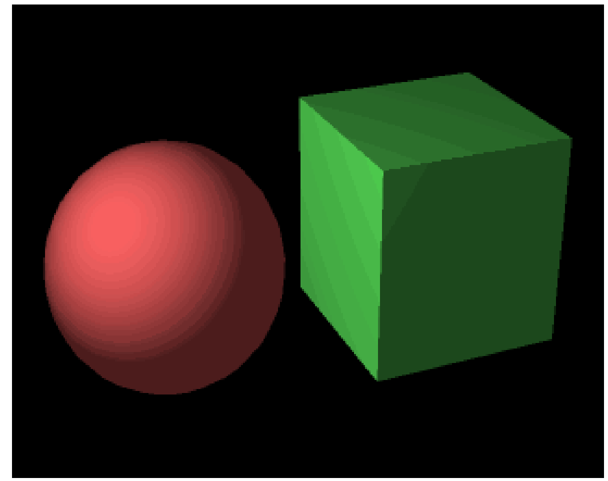
$$I_{l,diff} = k_d I_l (N \cdot L)$$

Requires light source (point or direction) and surface normals of the geometry.
This is independent of the viewers location.

Combining Ambient and Diffuse



Diffuse Alone
(note how part of the sphere is completely black. Why?)



Diffuse + Ambient

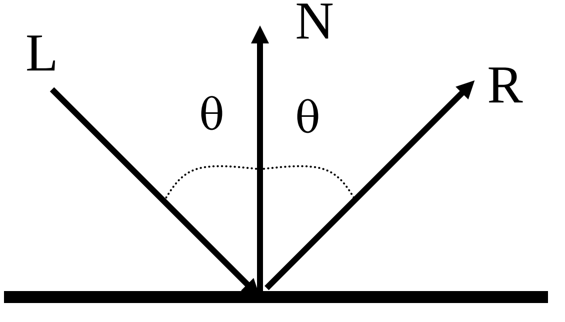
$$I = k_a I_a + k_d I_l (N \cdot L)$$

Specular Reflection

Specular

- When you look at a shiny surface, such as polished metal, an apple, or even a person's forehead, you see a "specular" hightlight, or bright spot, at a certain viewing direction
- Specular reflection models the glossy appearance of shiny object.
- Where this bright spot appears on the surface is a function of where it the surface is viewed from (ie, it is very dependent)

- What is happening? The light is being reflected at a certain direction



Pure reflector
angle in = angle out

[with pure reflection we could only see the light if our viewing direction is along the reflected ray R]

Bui Tuong-Phong

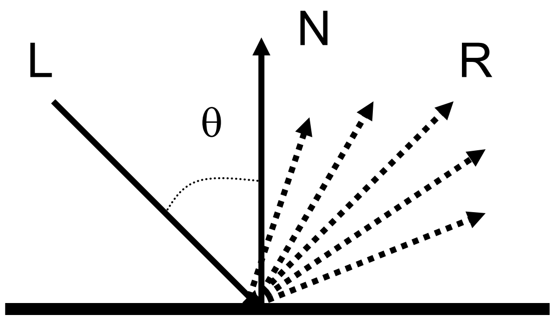
- What you are about to see is called the “Phong Illumination Model”
  - It is named after Bui Tuong Phong
- Bui Tuong Phong was a Vietnamese PhD student at University of Utah in 1973
  - Remember Ivan Sutherland (father of interactive CG)
  - Ivan went to Utah and founded their computer science department in 1965
- Phong introduced this model in 1972-73 (it was published in 1975)
  - It has been used every since
- Sadly, Phong had terminal leukemia and died 1975
  - He knew he had terminal as a PhD student
  - He had joined Stanford as a professor shortly before his passed away

http://en.wikipedia.org/wiki/Phong_shading

Specular Reflection

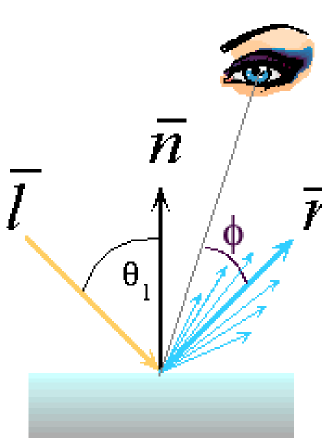
Specular reflection in reality

- Angle in ≠ Angle out exactly



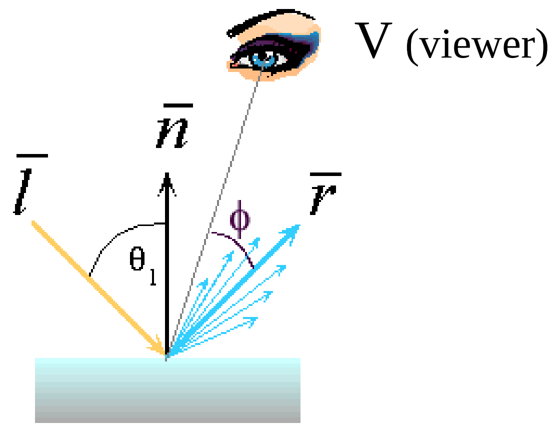
Most of the light travels in the reflect direction, but some is scattered around this direction

We approximate this with what we call the Phong Illumination Model:



$$I_{spec} = k_s I_l \cos^{n_{shinny}} \phi$$

Phong Specular Model

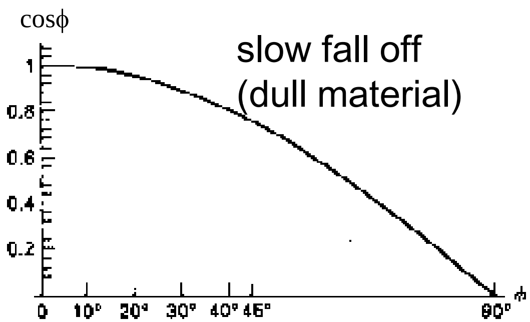


$$I_{spec} = k_s I_l \cos^{n_{shinny}} \phi$$

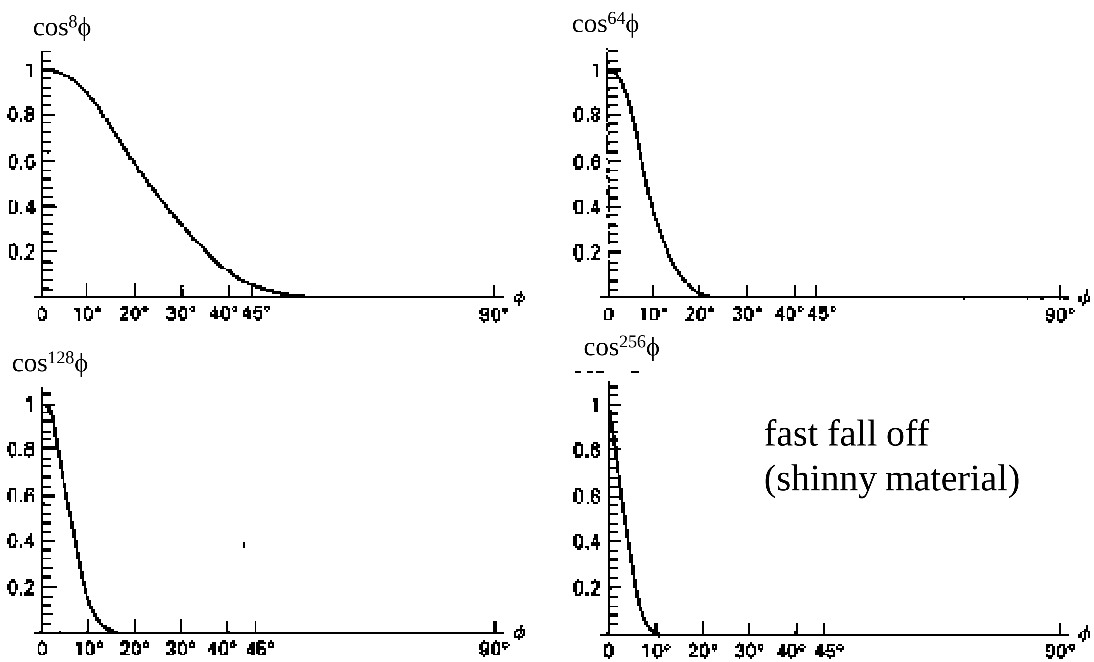
$$I_{spec} = k_s I_l (V \cdot R)^{n_s}$$

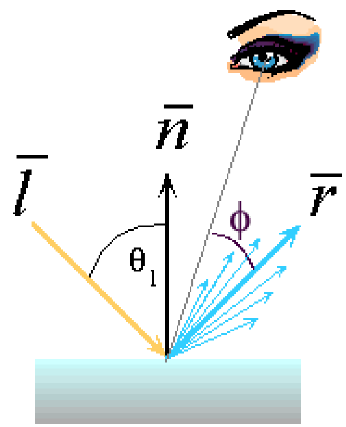
n_s controls the how "shiny" the surface appears.
Large n_s (Shiny Surface)
Small n_s (Dull Surface)

Shinny Exponent n_s

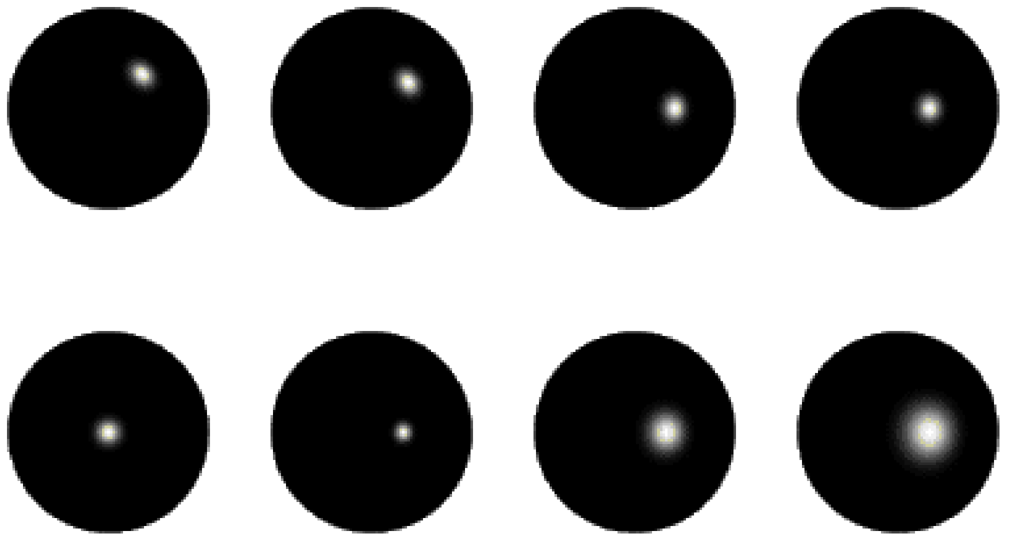


$\cos^n\phi$ controls the fall-off of the specular reflection as the viewing ray V moves away from the reflected ray R.





Phong Specular Model Example



Various lighting directions with different shinny exponents

Combining it all together

$$I = k_a I_a + k_d I_l (N \cdot L) + k_s I_l (V \cdot R)^{n_s}$$

ambient diffuse specular

What about color? Have a parameter for each R, G, B channel.

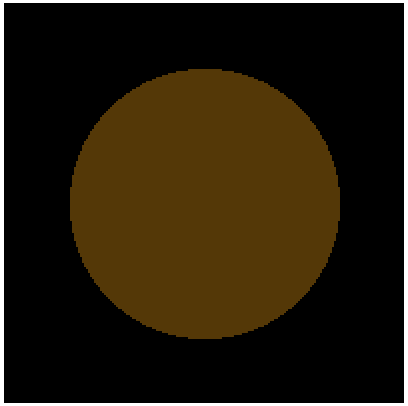
$$I_r = k_{a_r} I_{a_r} + k_{dr} I_{lr} (N \cdot L) + k_{s_r} I_{l_r} (V \cdot R)^{n_s}$$

$$I_g = k_{a_g} I_{a_g} + k_{d_g} I_{l_g} (N \cdot L) + k_{s_g} I_{l_g} (V \cdot R)^{n_s}$$

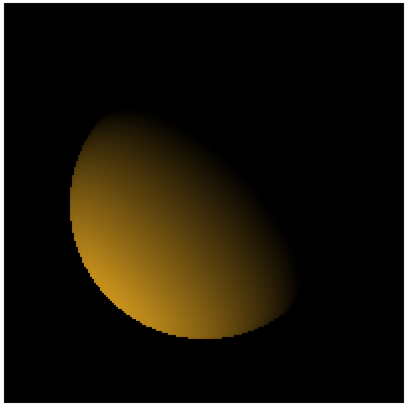
$$I_b = k_{a_b} I_{a_b} + k_{d_b} I_{l_b} (N \cdot L) + k_{s_b} I_{l_b} (V \cdot R)^{n_s}$$

[note light direction L, N, and viewing direction V are the same]

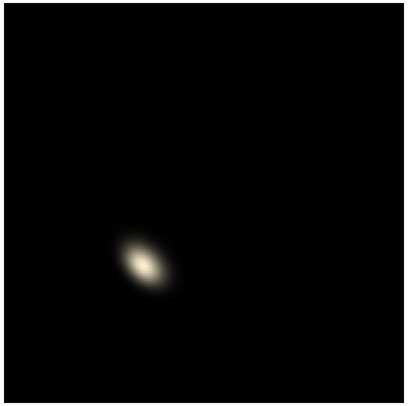
Combining it all together



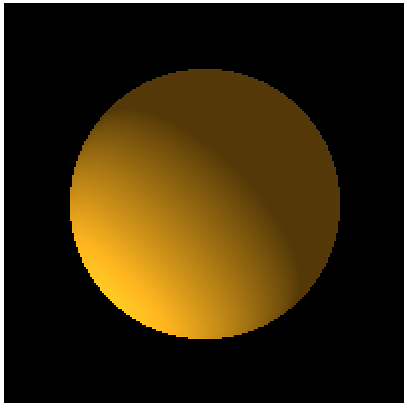
Ambient



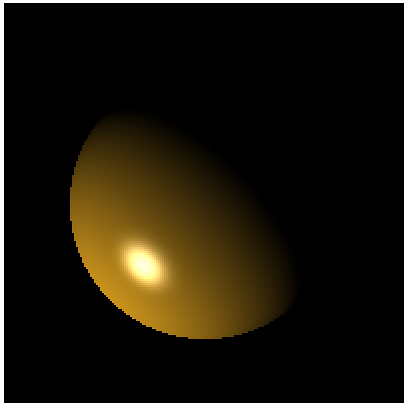
Diffuse



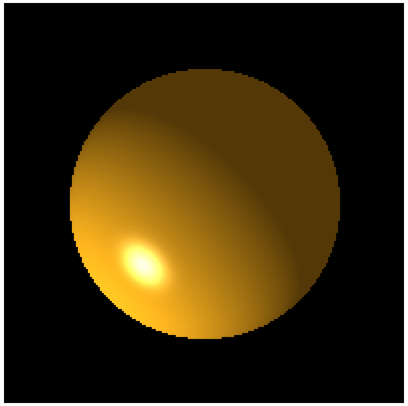
Specular



Ambient+Diffuse



Diffuse+Specular



Ambient+Diffuse+Specular

see
http://www.codeproject.com/KB/graphics/Basic_Illumination_Model.aspx

Multiple Light Sources

$$I = k_a I_a + \sum_{i=1}^{N} k_d I_i (N \cdot L_i) + k_s I_i (V \cdot R_i)^{n_s}$$

Simply sum up the contributions from all the light sources.

Note, we typically only have one ambient light source.

Must check to make sure you don't over flow max intensity value.

OpenGL Lighting

OpenGL supports ambient, diffuse, and specular.

They also introduce another surface property term called:
Emission. This means the surface emits light [its equivalent to ambient]

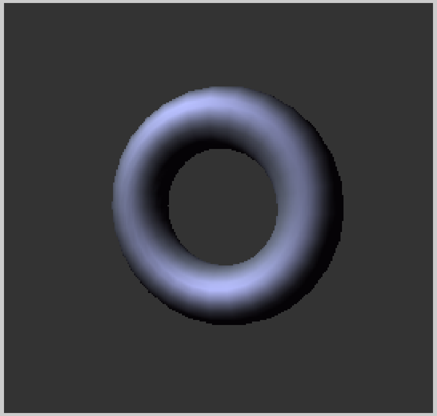
$$I = \boxed{I_e} + k_a I_a + k_d I_l (N \cdot L) + k_s I_l (V \cdot R)^{n_s}$$

Emission (just some constant)

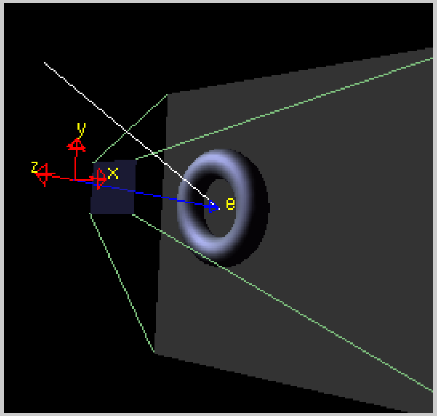
OpenGL allows you to set the parameters for each R, G, B, A values.

OpenGL Lighting

Screen-space view



World-space view



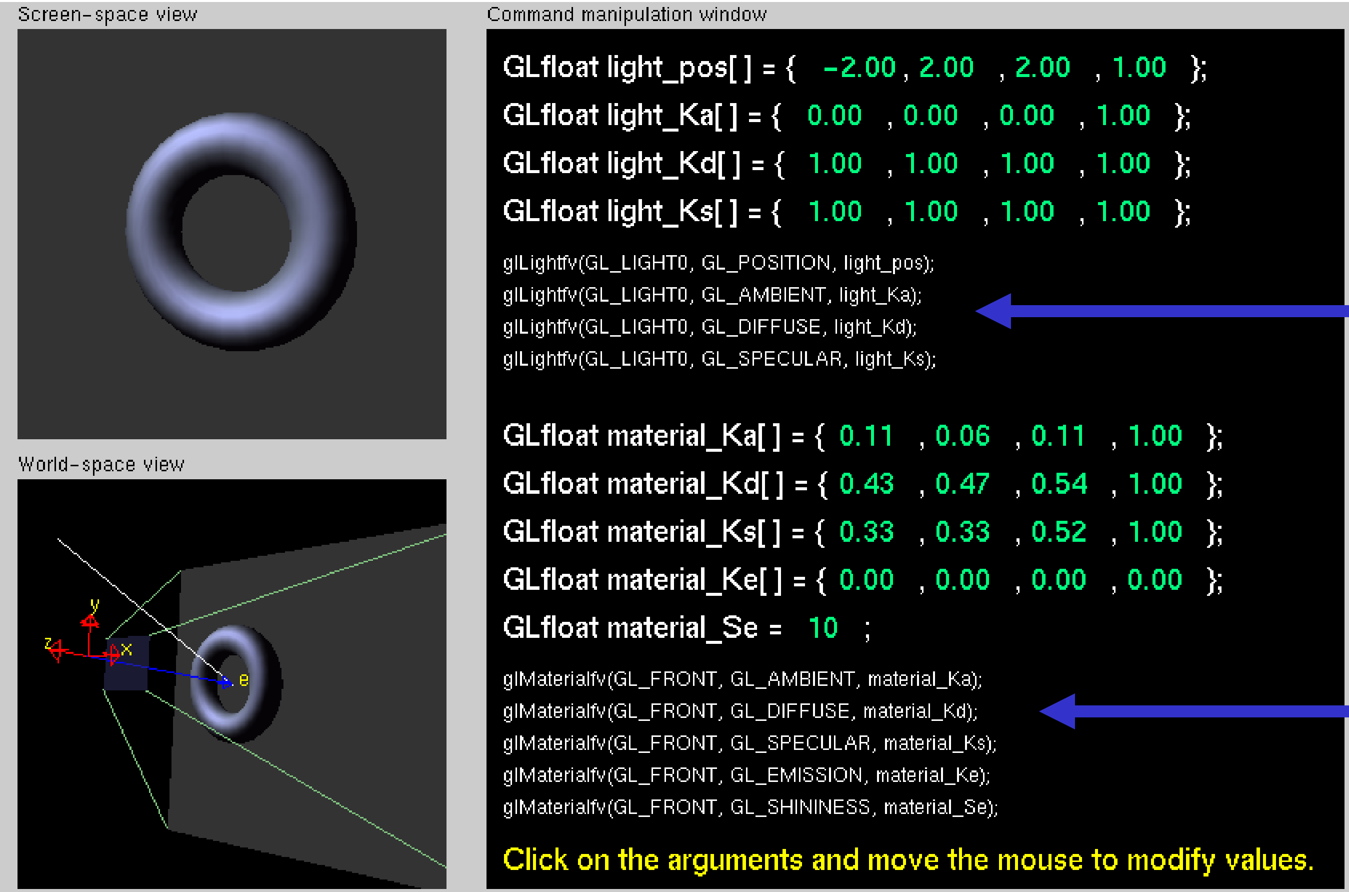
Command manipulation window

```
GLfloat light_pos[ ] = {  -2.00 , 2.00  , 2.00  , 1.00  };
GLfloat light_Ka[ ] = {  0.00  , 0.00  , 0.00  , 1.00  };
GLfloat light_Kd[ ] = {  1.00  , 1.00  , 1.00  , 1.00  };
GLfloat light_Ks[ ] = {  1.00  , 1.00  , 1.00  , 1.00  };

glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_Ka);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_Kd);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_Ks);

GLfloat material_Ka[ ] = { 0.11  , 0.06  , 0.11  , 1.00  };
GLfloat material_Kd[ ] = { 0.43  , 0.47  , 0.54  , 1.00  };
GLfloat material_Ks[ ] = { 0.33  , 0.33  , 0.52  , 1.00  };
GLfloat material_Ke[ ] = { 0.00  , 0.00  , 0.00  , 0.00  };
GLfloat material_Se =  10  ;

glMaterialfv(GL_FRONT, GL_AMBIENT, material_Ka);
glMaterialfv(GL_FRONT, GL_DIFFUSE, material_Kd);
glMaterialfv(GL_FRONT, GL_SPECULAR, material_Ks);
glMaterialfv(GL_FRONT, GL_EMISSION, material_Ke);
glMaterialfv(GL_FRONT, GL_SHININESS, material_Se);
```

Click on the arguments and move the mouse to modify values.

- Light Source position
- Ambient light s. values RGBA
- Diffuse light source values
- Specular light source values
- Materials ambient Ka (RGBA)
- Materials Diffuse Kd
- Materials Specular Ks
- Emission (constant color)
- Shinny coefficient ns

OpenGL Lighting



Define up to 8 lights sources (GL_LIGHT0-8)

Setting glMaterialfv() Will affect all polygons drawn after this command. So, draw some polygons, then change material properties, draw some more, etc . .

Note, there are more things you have to do than listed above to get lighting to work in OpenGL
You must enable lighting and materials glEnable(. . .)
See OpenGL RedBook, Chapter 5: lighting

Summary Illumination (Part 1)

<u>Phong Illumination Model – 3 components</u>

- **Ambient Light**
  - Non-directional light
- **Diffuse Reflection**
  - Reflected light will depend on surfaces position to the light source
- **Specular Reflection**
  - Reflected light will depend on surface, light position, and viewing position

Combine this models to calculate the intensity for each polygon.

$$I = k_a I_a + \sum_{i=1}^{N} k_d I_i (N \cdot L_i) + k_s I_i (V \cdot R_i)^{n_s}$$