DigiPen
INSTITUTE OF TECHNOLOGY

CS300
Lighting

Broken Age (2014)

The Wolf Among Us (2014)

Forza Horizon 3 (2016)

Jotun (2016)

What remains of Edith Finch? (2017)

Lighting Techniques

Lighting Techniques

I. Light source types
II. Surface color
III. Lighting and material model
IV. Attenuation
V. Spotlight effect
VI. Lighting equation
VII. Lighting in OpenGL
VIII. Lighting in GLSL

I. Light Source Types

a) Point Light

b) Directional Light

c) Spot Light

a) Point Light

- When the light emanates from a position in space uniformly in all directions, the light type is called point light.
- A point light can be used to represent a light bulb.
- Since the point light has a position, the point light intensity is based on the distance.
- The attenuation is inversely proportional to the distance from the light source to the object being lit

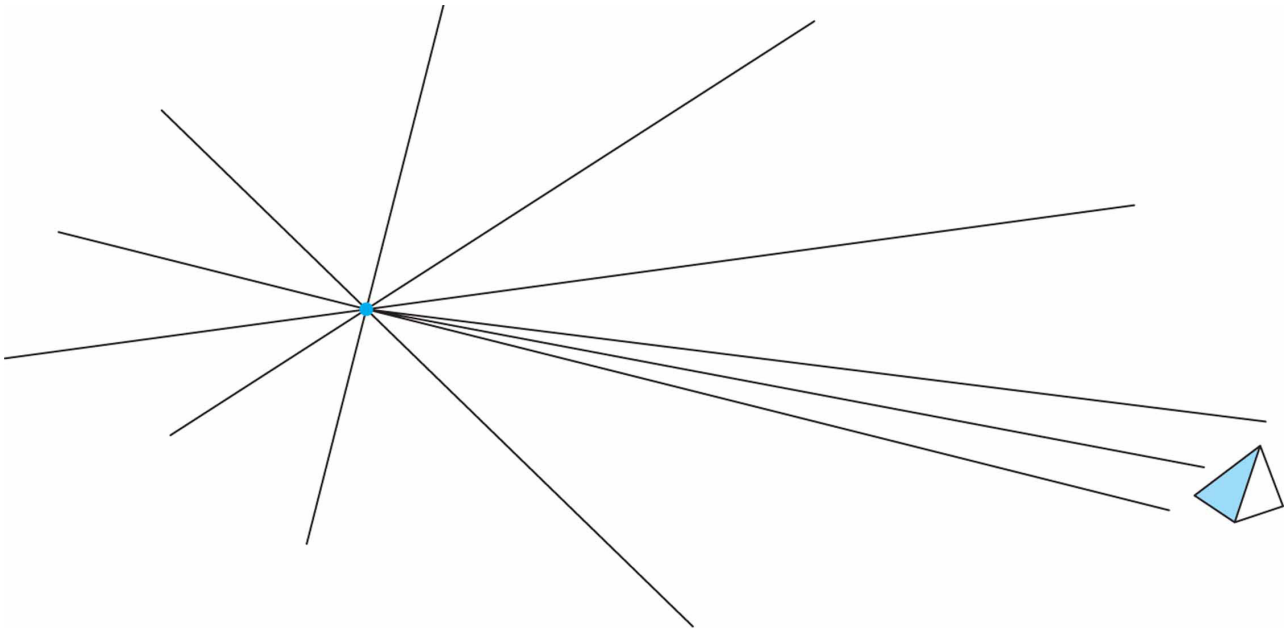
Figure 17-1 Diverging ray paths from a point light source.

Chapter 17
Computer
Graphics with
OpenGL
(Hearn & Baker)

b) Directional Light

- When the light source is infinitely distant from the object, we say that the light source represents a directional light.
- When the rays emanating from a directional light source reach an object, the rays would all be parallel.
- The sun is an example of a directional light.

Figure 17-2 Light rays from an infinitely distant light source illuminate an object along nearly parallel light paths.



b) Directional Light *(cont'd)*

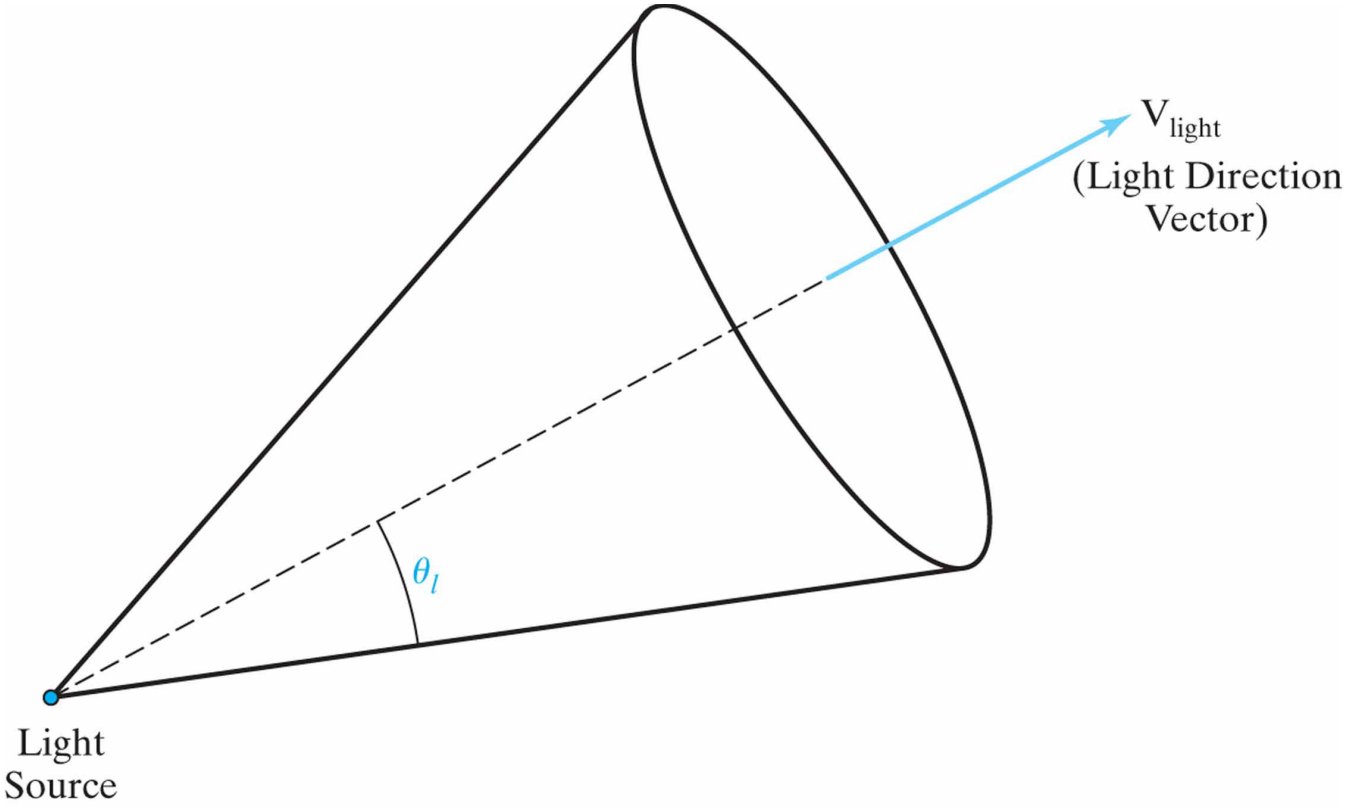
- Note that attenuation is not considered when dealing with directional light.

- To represent a directional light, we need only the direction and color of the light since a directional light has no position

c) Spot Light

- When the light rays emanate from a single point following a direction with a cut off angle, the emitted light will have the shape of a cone.
- This type of light is called a spot light.
- The spot light has a source position, a vector representing the light direction, and the inner and outer angles of the cone.

Figure 17-3 A directional point light source. The unit light-direction vector defines the axis of a light cone, and angle θ_l defines the angular extent of the circular cone.

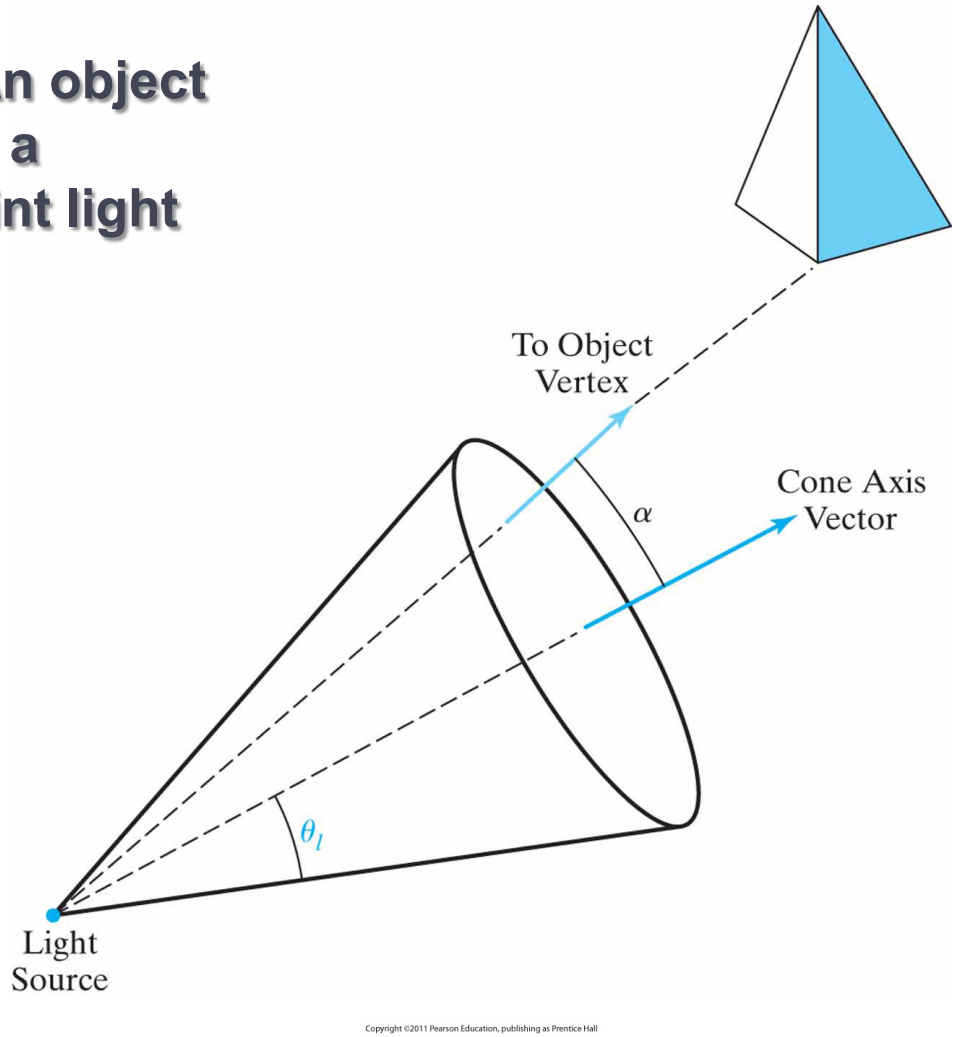


c) Spot Light *(cont'd)*

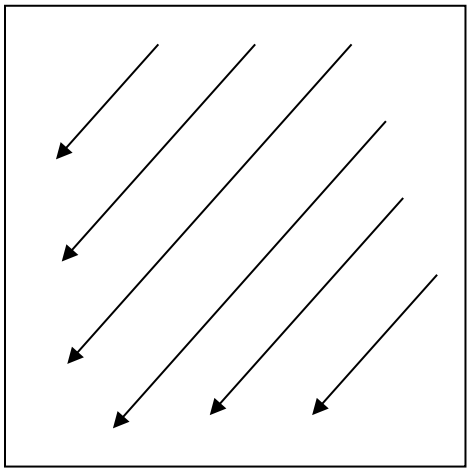
- Light intensity within the inner angle of the cone is constant (usually, its maximum intensity). Outside the outer angle, its intensity is zero. Intensity between the inner and outer angle varies (interpolated between the inner and outer angle intensity).

- Since the spot light has a position in space, attenuation of the light's intensity is inversely proportional to the distance from the spot light to the object being lit.

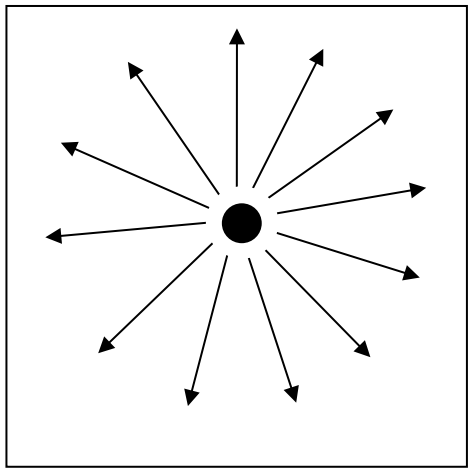
Figure 17-4 An object illuminated by a directional point light source.



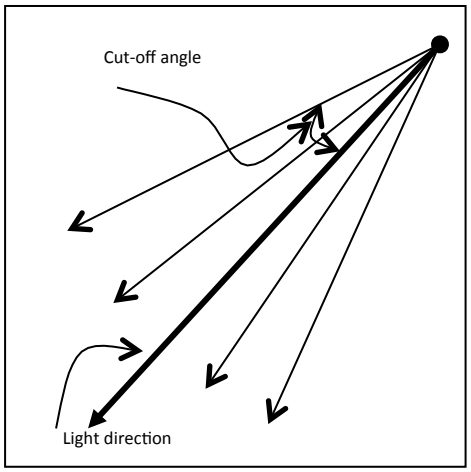
c) Spot Light *(cont'd)*



Directional Light



Point Light



Spot Light

II. Surface Color

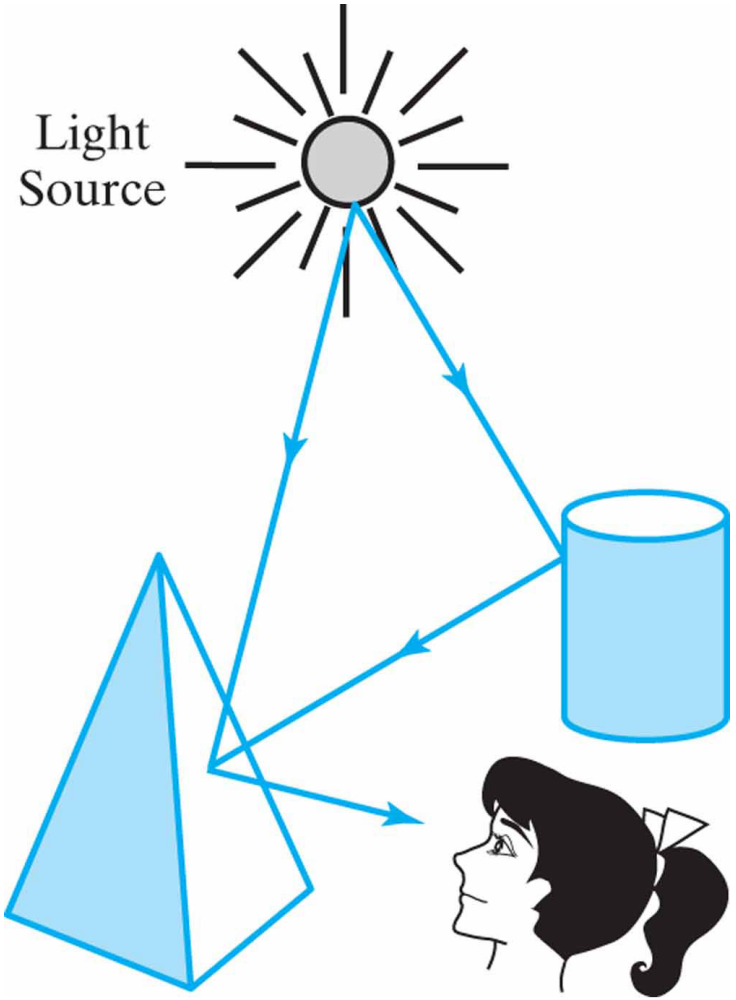
- What is it that determines the color of the surface?

II. Surface Color

The surface color is determined by:

- The color of the light source that shines on that surface.
- The color that is absorbed, reflected, emitted, and transmitted by the surface.

Figure 17-8 Surface lighting effects are produced by a combination of illumination from light sources and reflections from other surfaces.

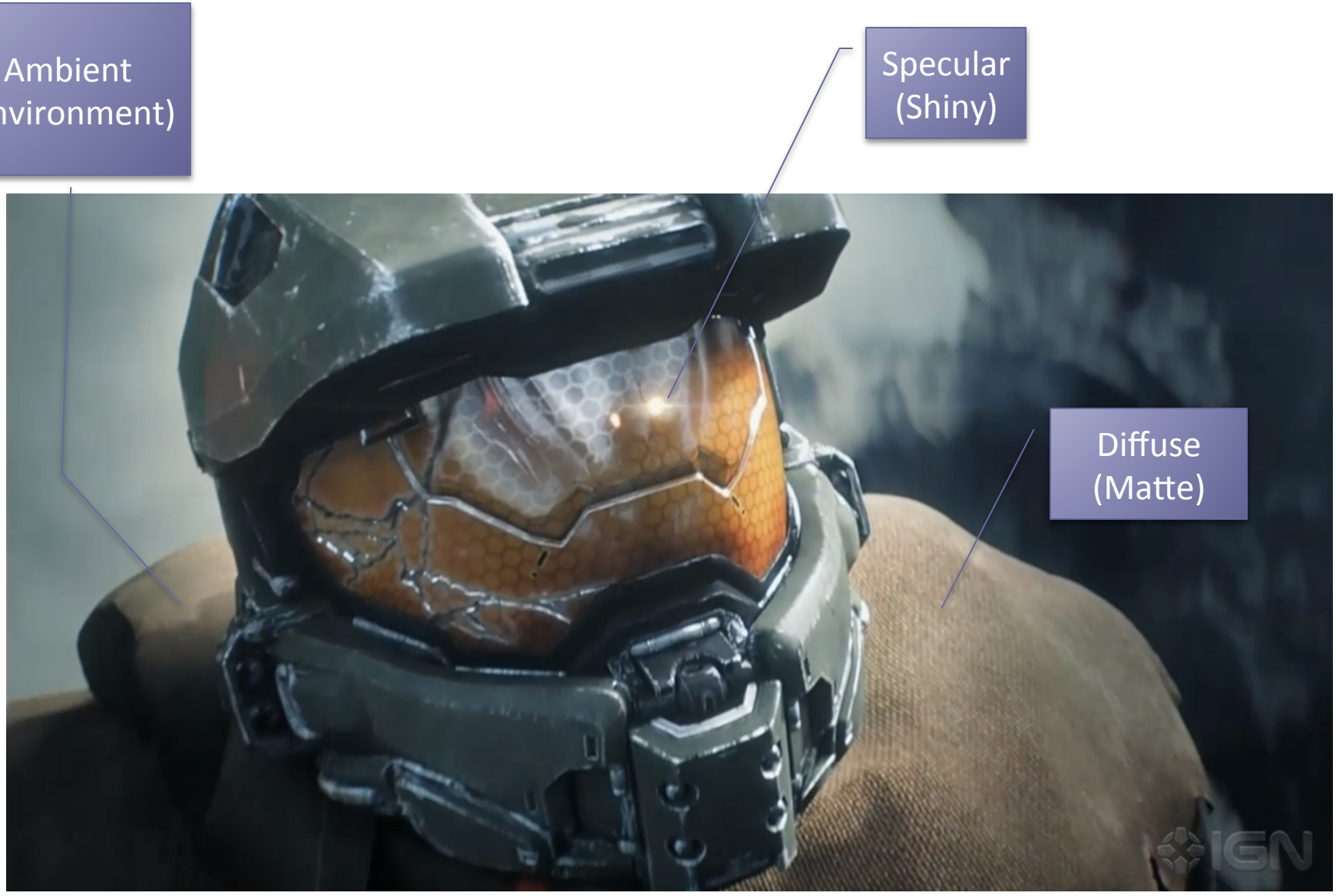


It is IMPOSSIBLE to imitate/ reproduce the natural behavior in light-object interaction.
In computer graphics, we will approximate this interaction with the use of "lighting models"

A MODEL is a mathematical formulation of the physical phenomena

III. Lighting and Material Model

- In computer graphics, the lighting and material (surface) model is divided into four independents components.
- Each component is calculated independently and added together at the end.
- They are:
  a) Ambient
  b) Diffuse
  c) Specular
  d) Emissive



a) Ambient

- Ambient light has been scattered so much by the environment that its direction is impossible to determine.
- In a real environment a light emanating from a light source might bounce off multiple objects before reach the surface.
- The object will then be illuminated by this indirect light source(s).
- In our illumination model we try to simulate this indirect lighting with the ambient component.

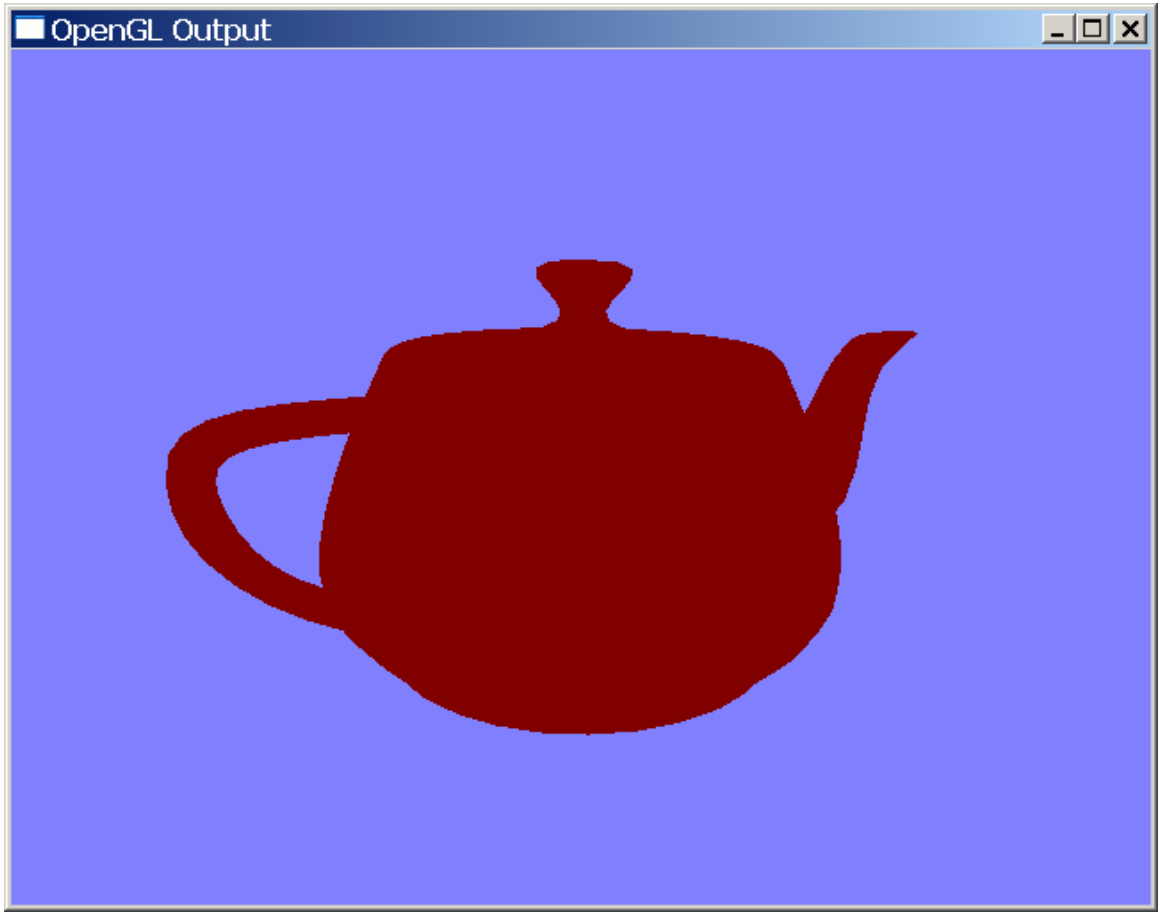
a) Ambient *(cont'd)*

- The ambient component can be modeled by

$$I_{ambient} = I_a K_a$$

- Where:
  - I_a is the intensity of the ambient light, assumed to be a constant for all objects.
  - K_a is a coefficient determining the amount of ambient light reflected from the object's surface.

Ambient Term



b) Diffuse

- Diffuse light comes from a direction and it is scattered equally as it hits the surface.
- Based on Lambert's law, this component of lighting model is the closest to real physics.
- Lambert's law states that totally matte surfaces reflect the light according to the cosine of the light vector and the surface normal.

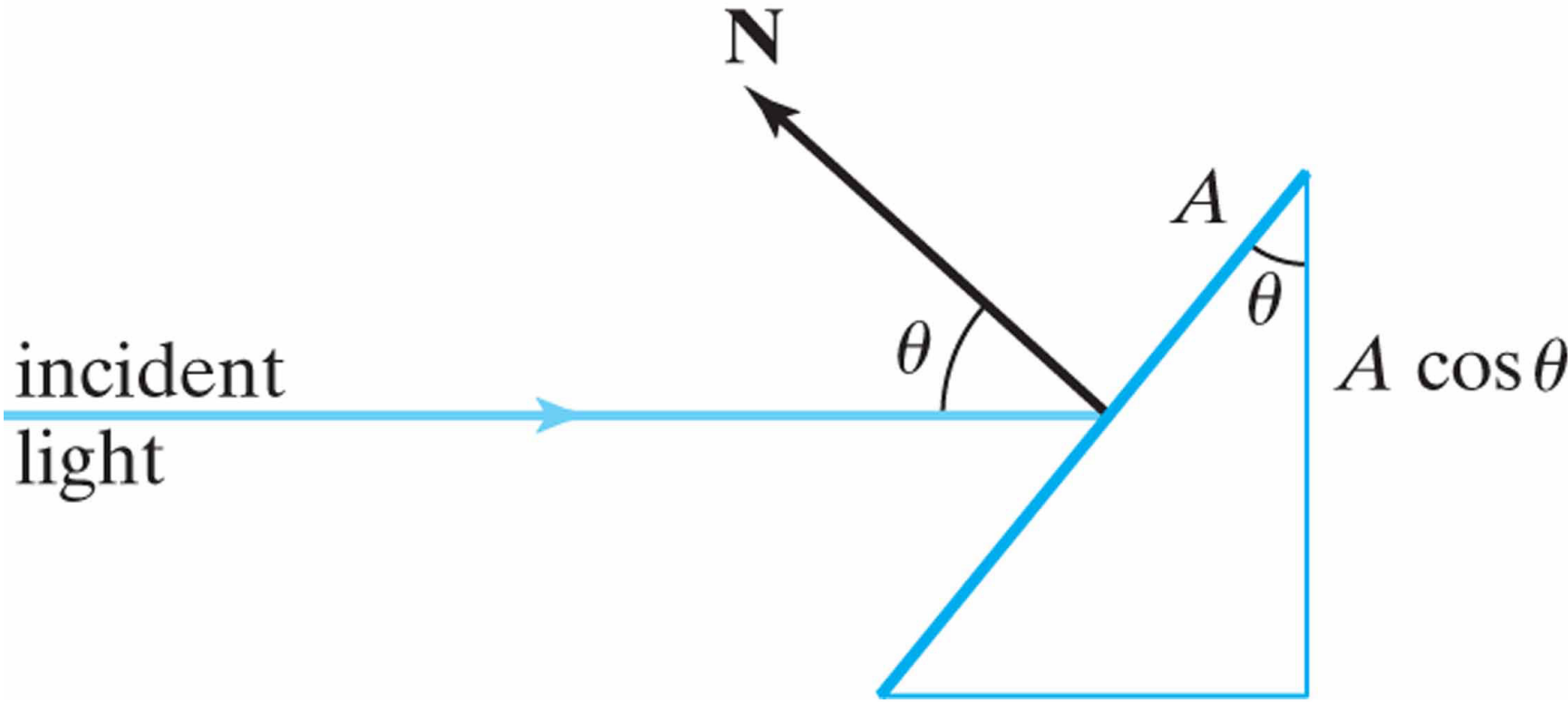
b) Diffuse *(cont'd)*

- The diffuse component can be modeled by:

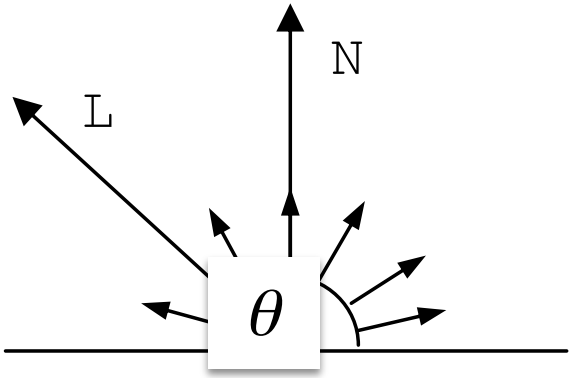
$$I_{diffuse} = I_d K_d \cos\theta = I_d K_d (N.L)$$

- Where:
  - I_d is the light source diffuse intensity.
  - K_d is the material's diffuse-reflection coefficient.
  - L is the vector to the light source.
  - N is the surface normal.
  - θ is the angle between the vector L and N.

Figure 17-11 An illuminated area *A* projected perpendicular to the path of incoming light rays. This perpendicular projection has an area equal to *A* cos *θ*.



b) Diffuse *(cont'd)*



- L and N are **unit vectors** representing the light vector and the normal vector to the surface respectively.

- Note that the light vector L is pointing towards the light (not away from it).

b) Diffuse *(cont'd)*

- When the angle between N and L becomes greater than 90 degree, then the diffuse component becomes zero.

- This is reflected by the equation below:

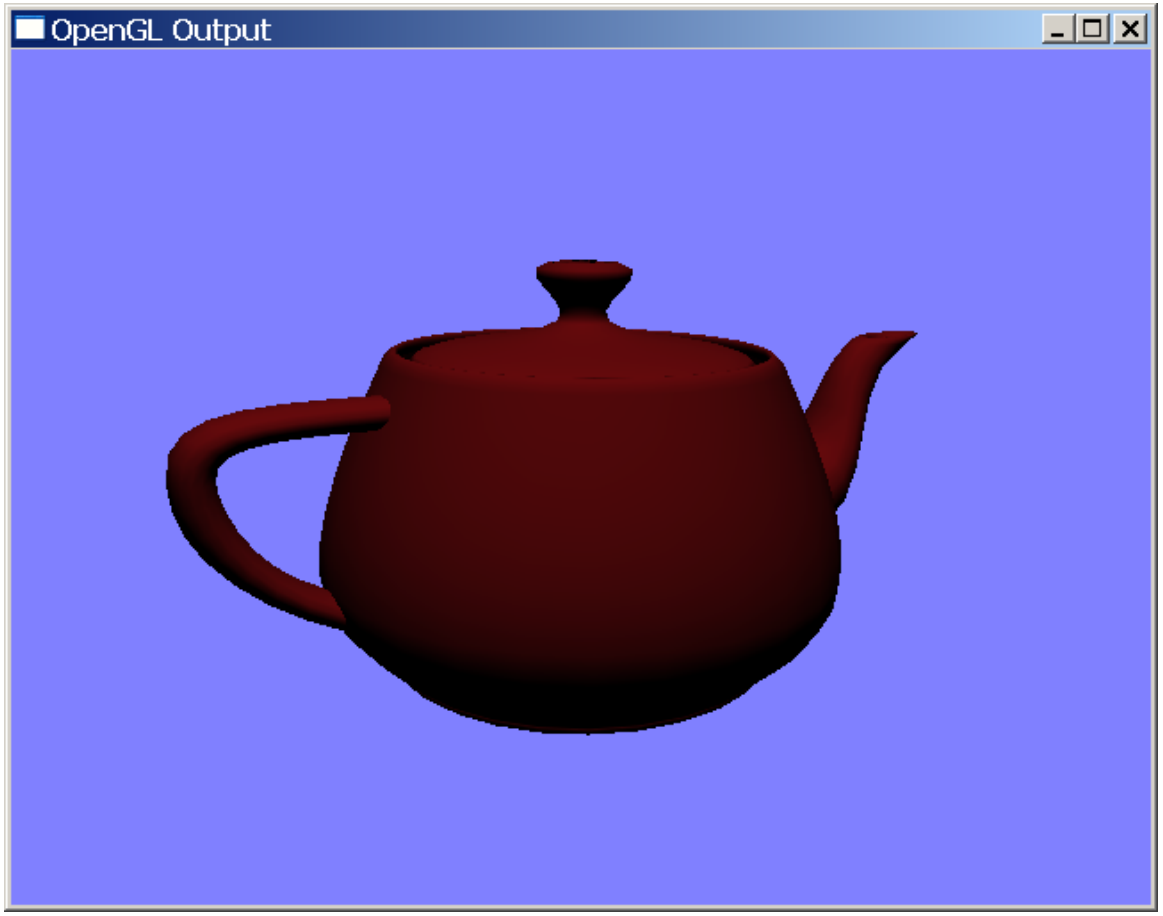
$$I_{diffuse} = I_d K_d \max(N.L, 0)$$

b) Diffuse *(cont'd)*

- Notice that since diffuse light get scattered equally in all direction when it hits the surface, its brightness will not depend on where the eye/camera is located.

- As such, the camera view vector is not present in the diffuse equation. The diffuse component is independent from the camera's orientation and position

Diffuse Term



c) Specular

- The specular component, like diffuse, comes from a particular direction. But, unlike diffuse, the specular component bounces off in a preferred direction.
- It is responsible for the highlight on a shiny surface.
- Since it bounces off in a preferred direction, its brightness depends on the eye/camera position.

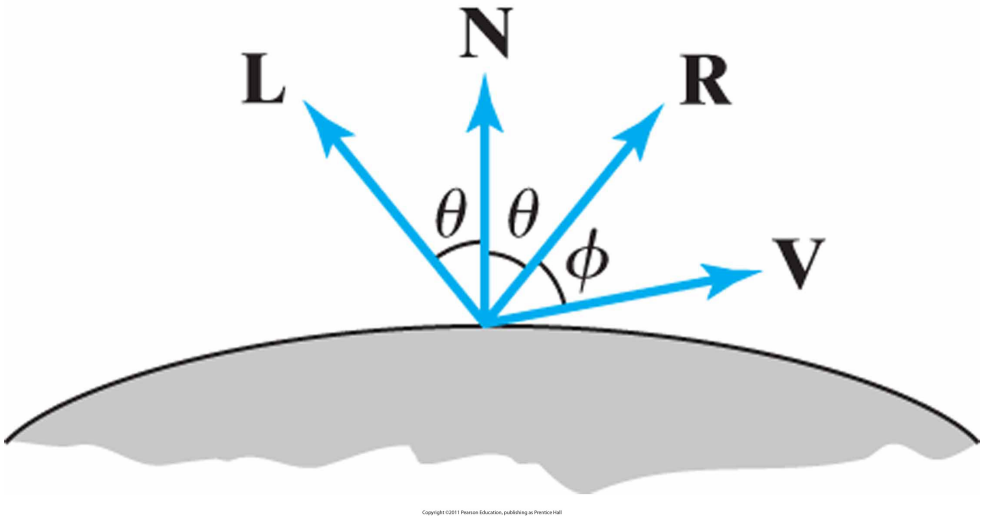
c) Specular *(cont'd)*

- It can be modeled by:

$$I_{specular} = I_s K_s (R.V)^{ns}$$

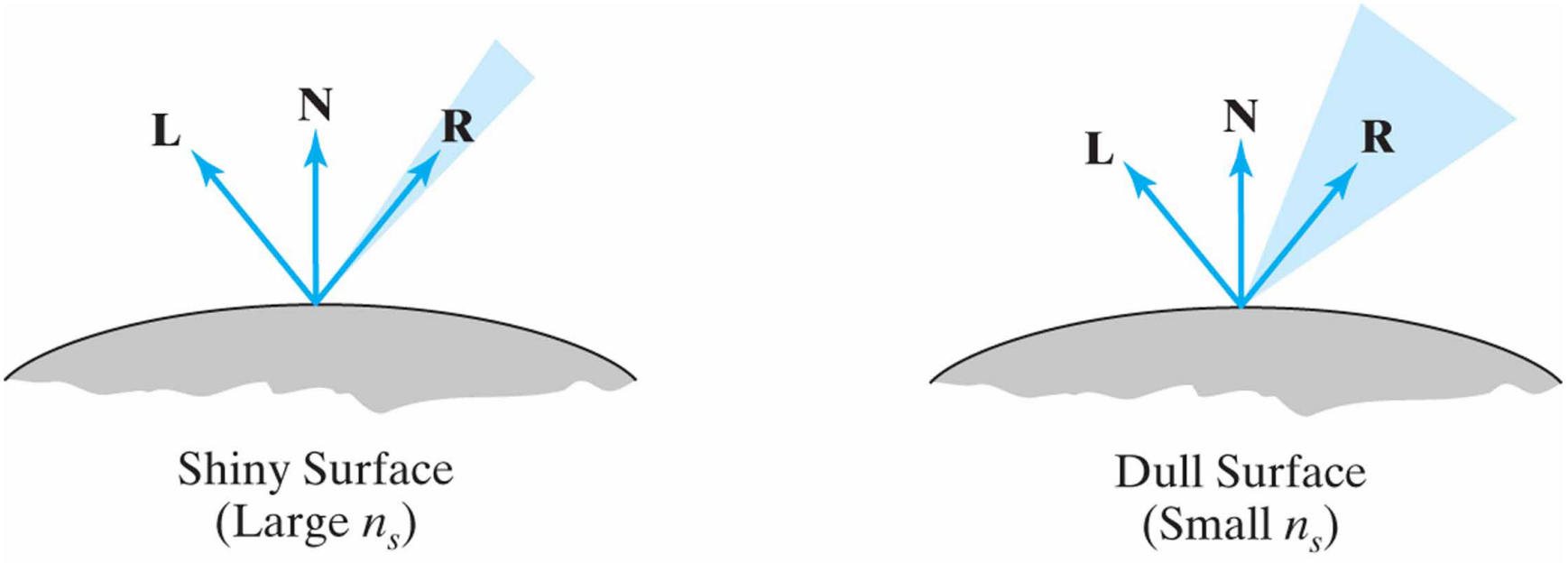
- Where:
  - I_s is the incident light source specular intensity.
  - K_s is the specular reflection coefficient.
  - R is the unit reflection vector.
  - V is the unit view direction vector.
  - ns is the index that controls the shininess or the tightness of the specular highlight, determined by the surface properties.

c) Specular *(cont'd)*



- The specular highlight varies as the reflection and view vector change; the more the reflection and view vector are aligned the higher is the specular component.

Effect of ns – the "material shininess"



Shiny Surface
(Large n_s)

Dull Surface
(Small n_s)

c) Specular *(cont'd)*

- Similar to the diffuse component, if the angle between the reflection vector R and the view vector V is greater than 90 degrees, then the specular component is zero.

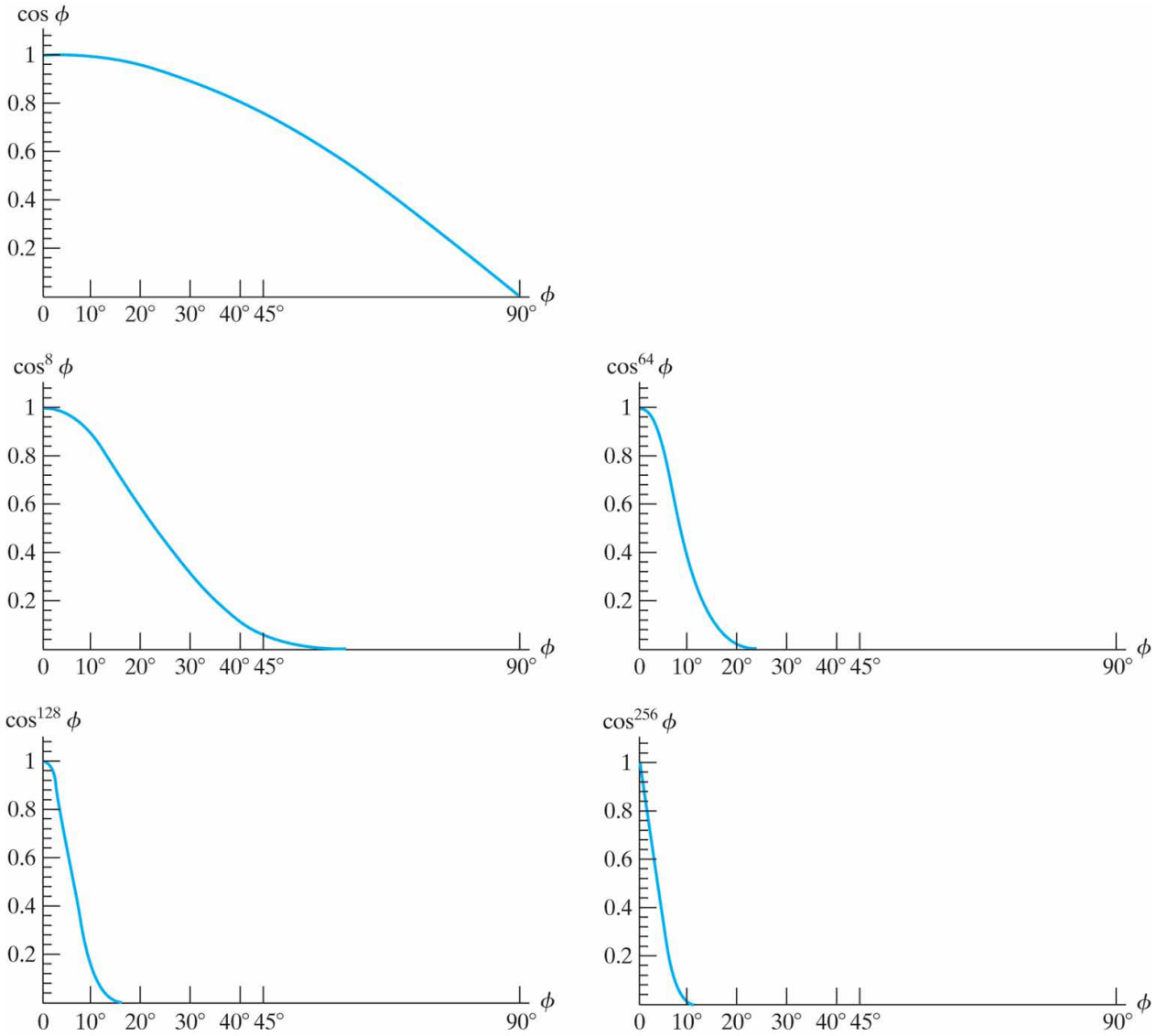
- This is reflected by the equation below

$$I_{specular} = I_s K_s \max(R.V, 0)^{ns}$$

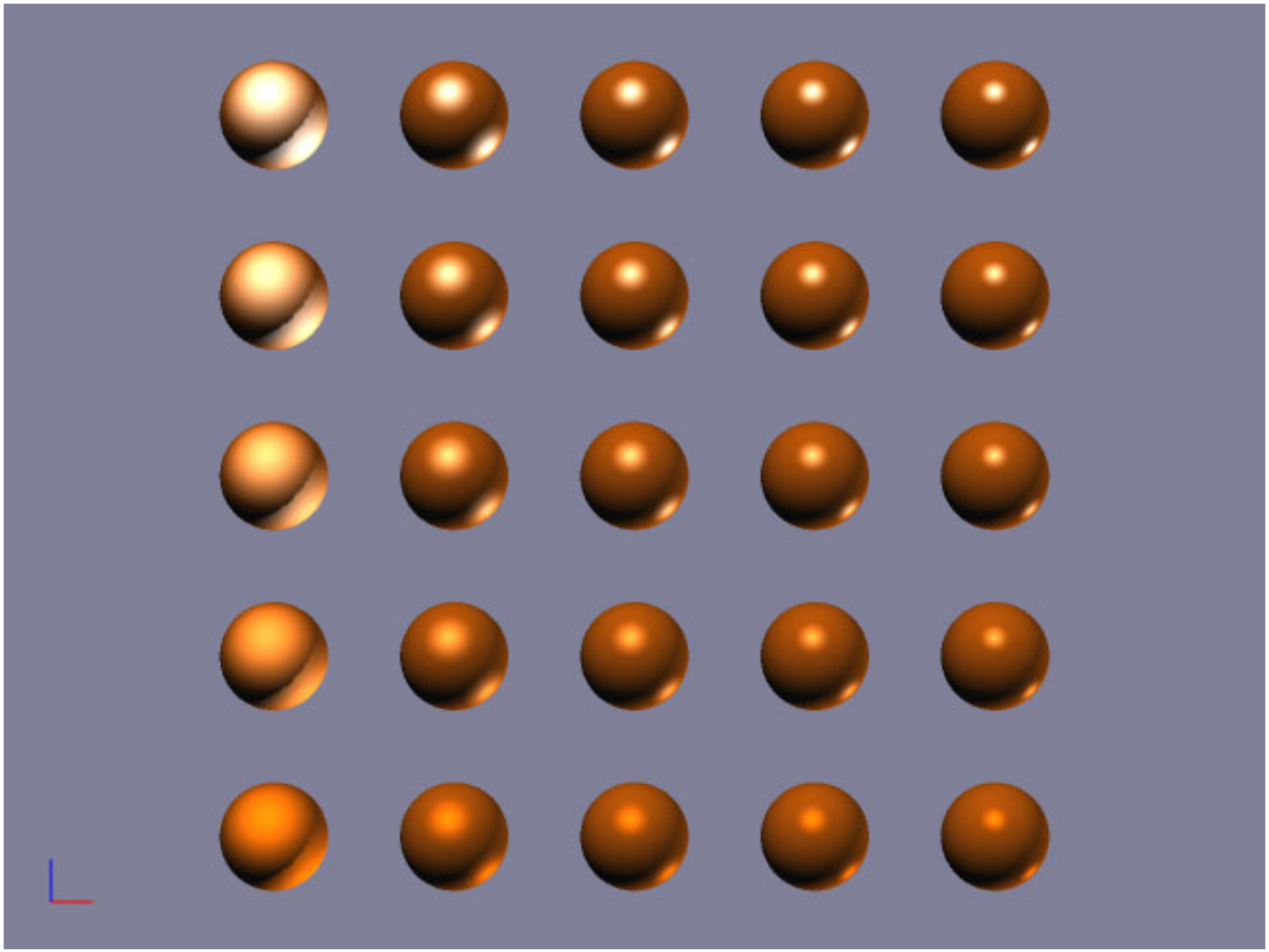
- Note that when the diffuse component is zero, the specular component will be zero too.

Figure 17-15 Plots of $\cos^{ns} \phi$ using five different values for the specular exponent n_s .

ns = 1
ns = 8, 64
ns = 128, 256



Specular = White



Specular = Diffuse (gold)

ns	5	25	45	65	85

Figure 17-17 The projection of either L or R onto the direction of the normal vector N has a magnitude equal to N · L.

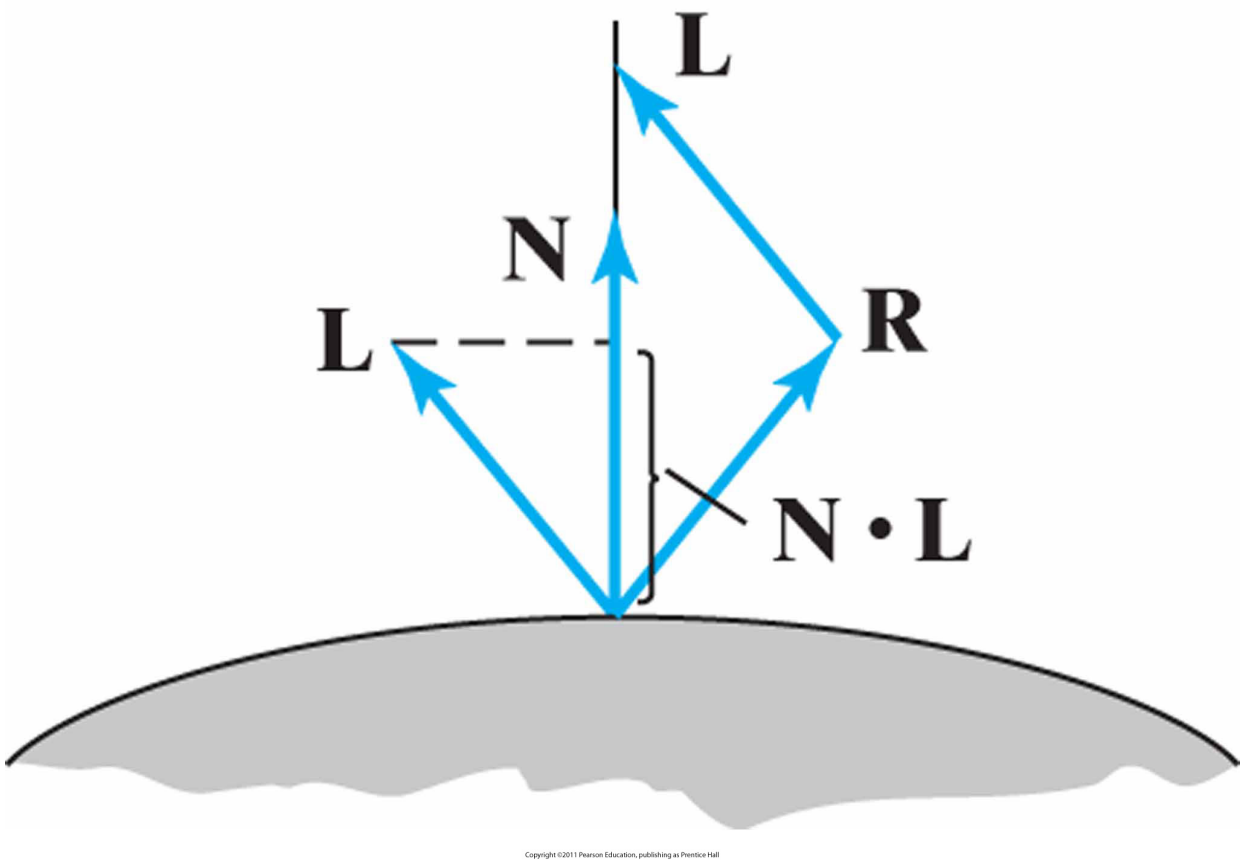
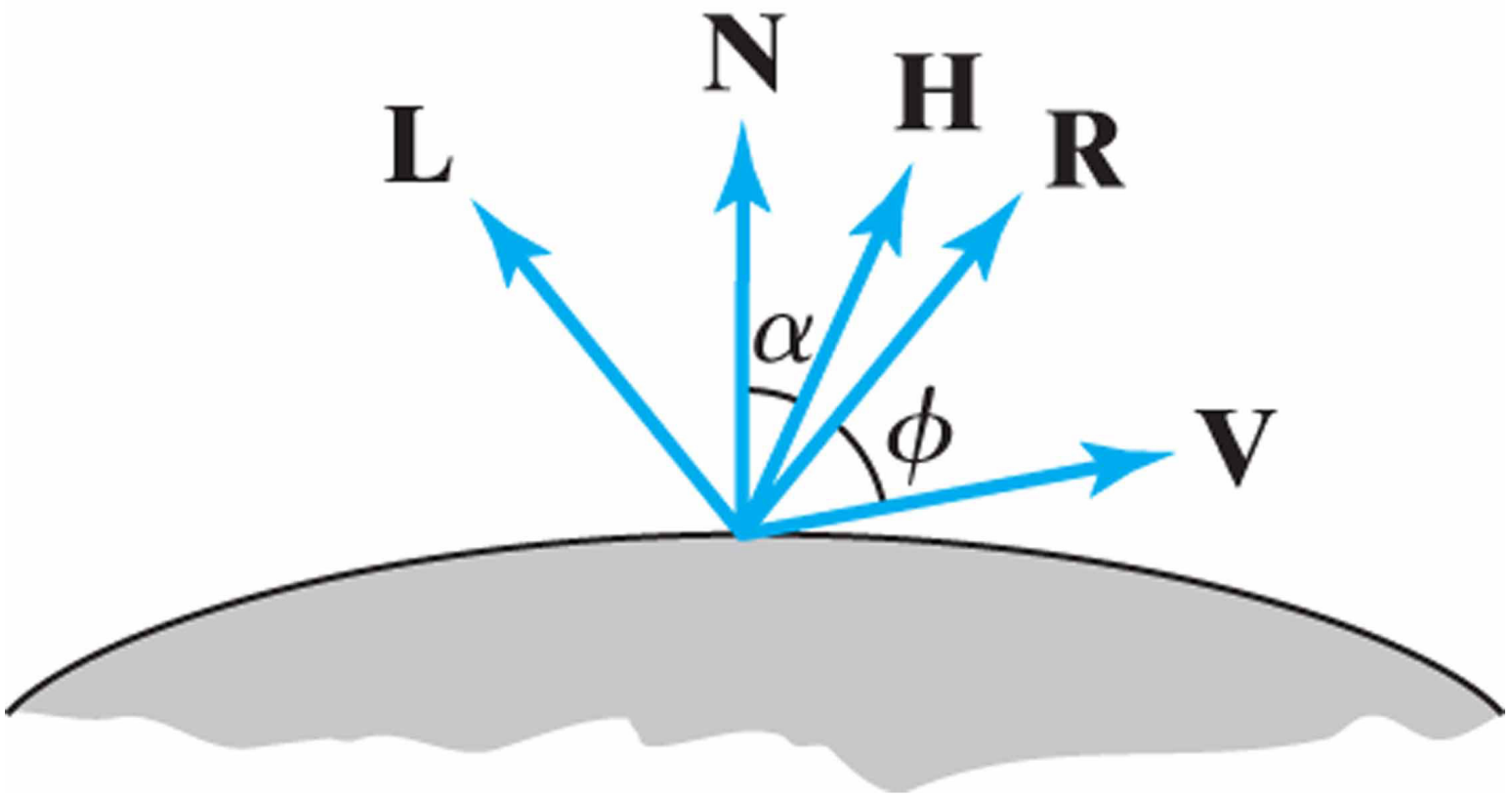


Figure 17-18 Halfway vector H along the bisector of the angle between L and V.

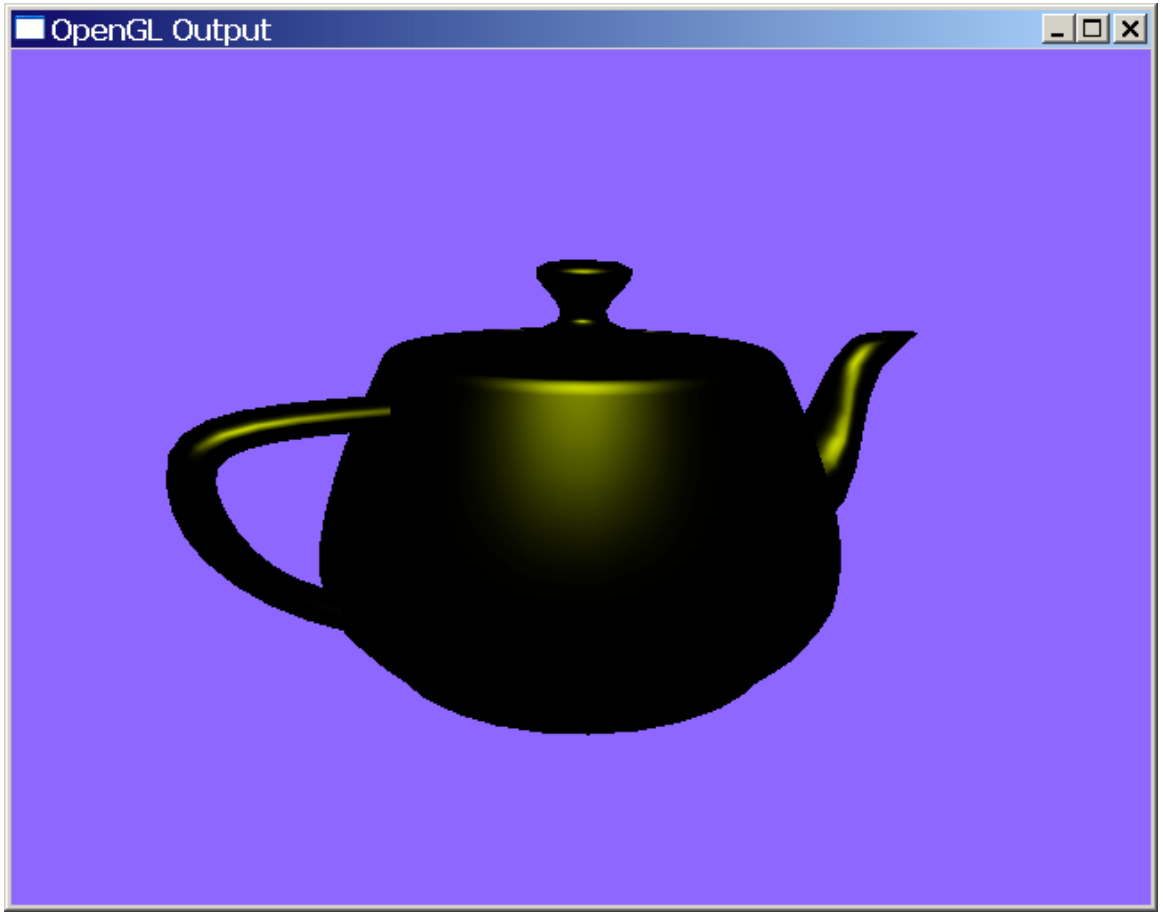


Modified Phong Model

- Use the Half-vector instead of the surface normal
- More efficient computation for largely planar surfaces
  - The normal does not vary and R need not be calculated per fragment
  - For lights and viewer sufficiently far from the object, V and L are constants

$$I_{specular} = I_s K_s \max(N.H, 0)^{ns}$$

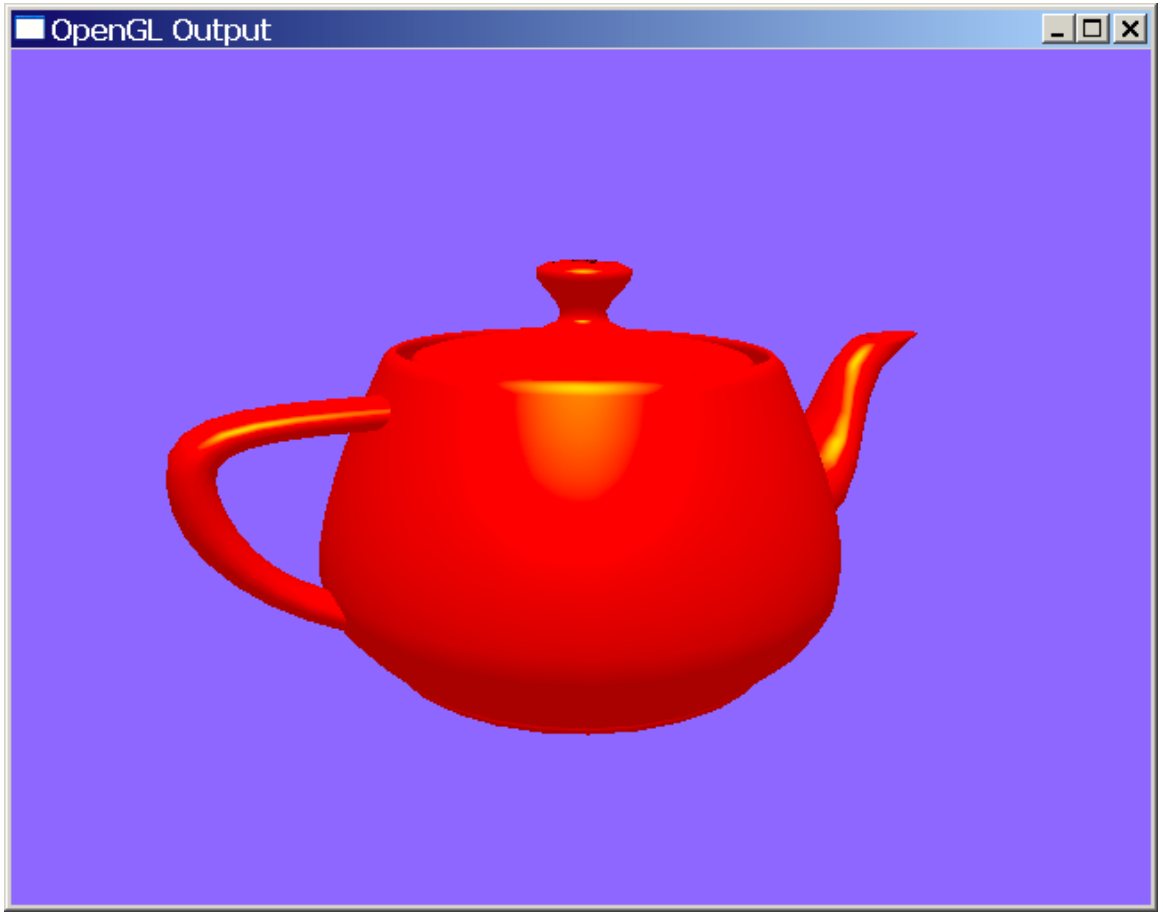
Specular Term

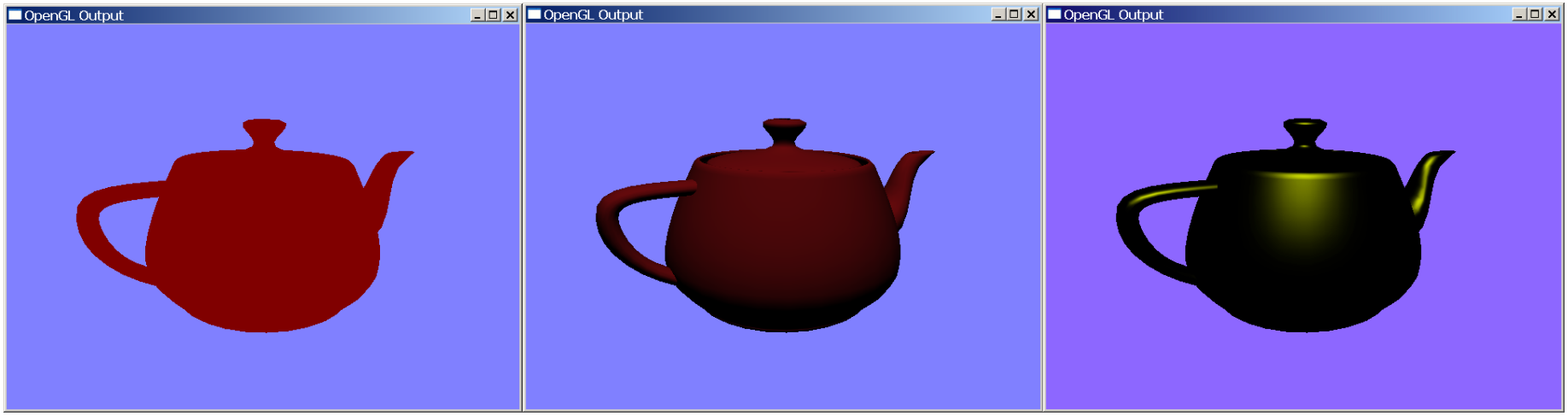


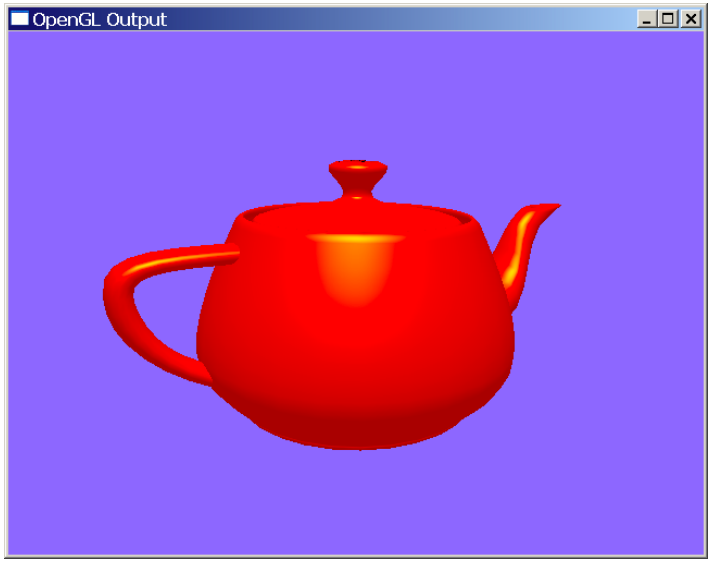
d) Emissive

- Emissive lighting originates from the object itself. It is not affected by other light sources.
- It is property of the material. Not the light sources.
- It is expressed as $I_{emissive}$.
- Examples on emissive lighting include the sun, fluorescent materials etc…

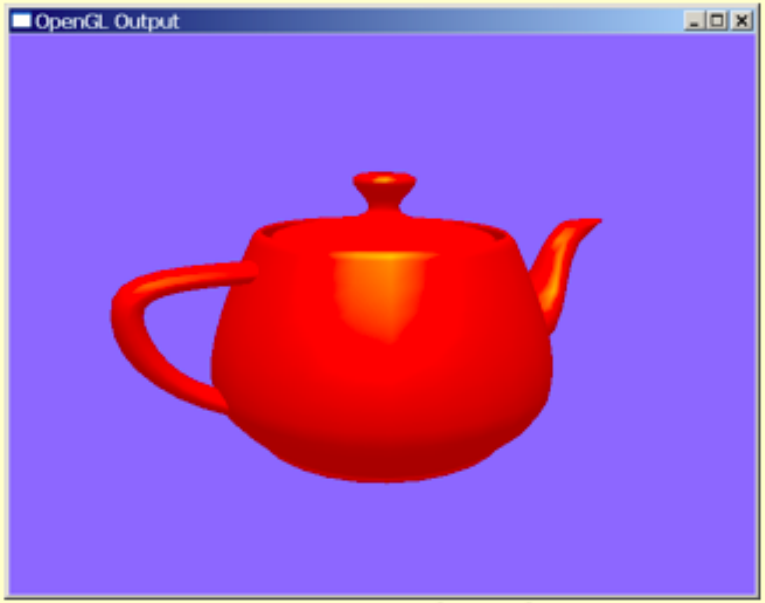
Combine all three terms …



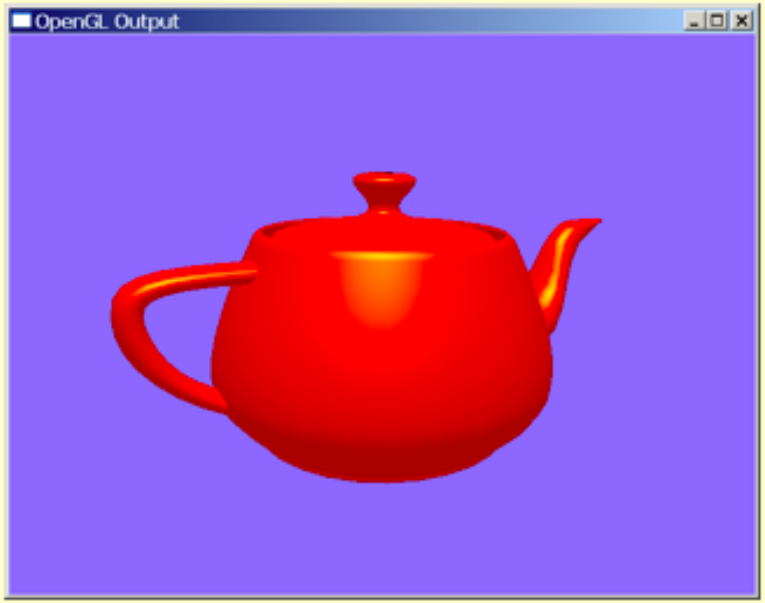




Fixed Function Vs Shader

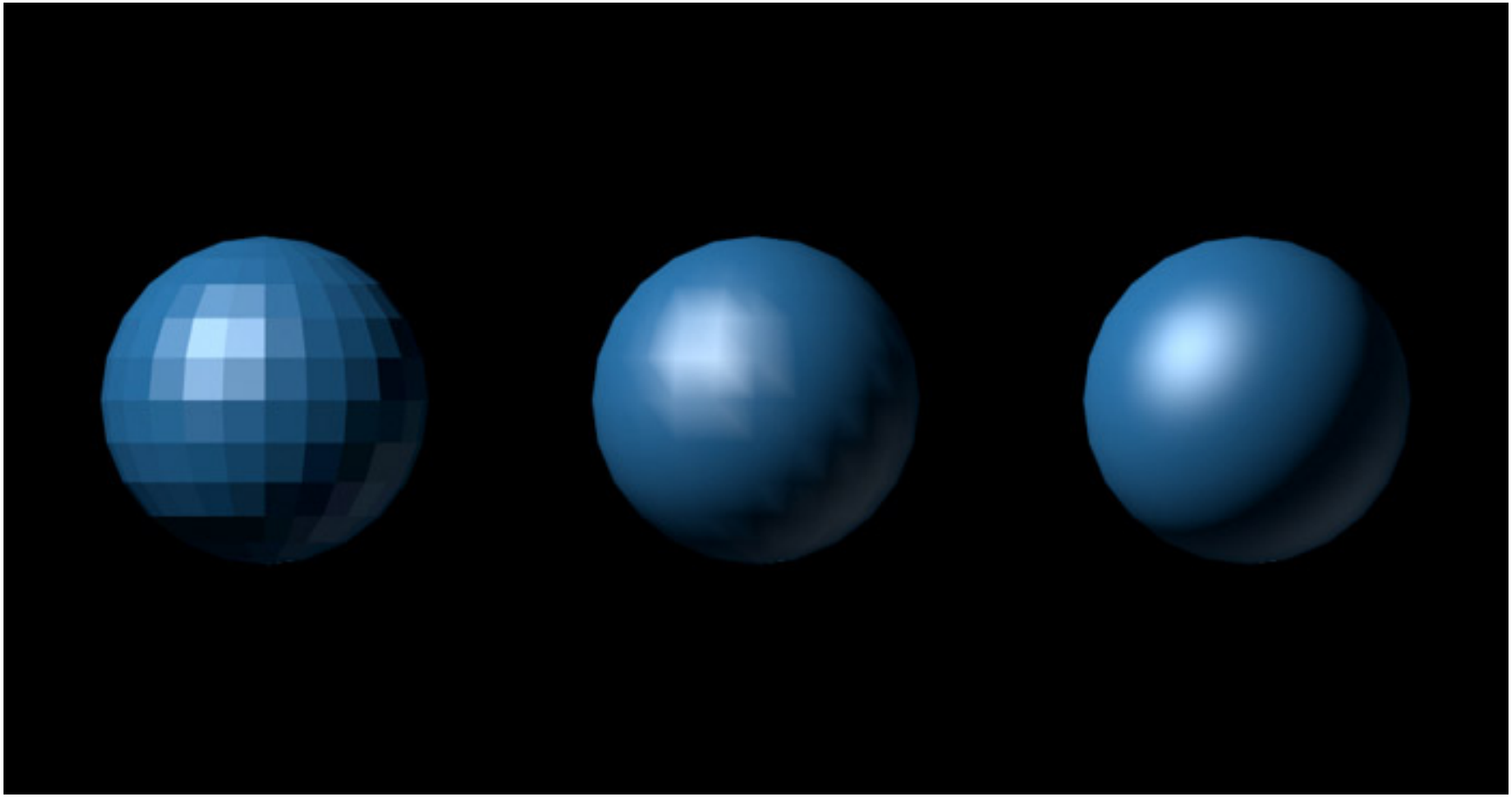


Per Vertex Lighting using fixed function pipeline (without GLSL)



Per Fragment Lighting using phong shading model (phong fragment shader)

Flat Vs Goroud Vs Phong Shading



IV. Attenuation

a) Light source (dim effect)

b) Atmospheric (blend effect)

a) Light source attenuation

- In reality, two parallel surfaces with identical properties will show different light intensities if their distances to the same light source are different.
- This is due to the fact that light energy is absorbed by particles in the air when it travels through a distance.
- To simulate this phenomenon, a light source attenuation factor ***Att*** is added to the light intensity equation.

a) Light source attenuation *(cont'd)*

- Light attenuation factor equation:

$$Att = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 {d_L}^2}, 1.0\right)$$

- Where:
  - c_1, c_2 and c_3 are user-defined constants associated with the light source.
  - c_1 is the constant attenuation.
  - c_2 is the linear attenuation.
  - c_3 is the quadratic attenuation.
  - d_L is the distance between the light source and the point under consideration.

b) Atmospheric attenuation

- Distant objects are usually perceived with lower intensities than closer objects and tend to be blended into the intervening atmosphere.

- This can be modeled by introducing the atmospheric attenuation to provide the 'depth-cueing'.

b) Atmospheric attenuation *(cont'd)*

- The object color intensity becomes:

$$I_{final} = SI_{local} + (1 - S)I_{fog}$$

- Where:
  - I_{local} is the local color intensity computed as usual.
  - I_{fog} is the depth-cued value representing atmospheric color.
  - S is derived as a function of the object's z value.

b) Atmospheric attenuation *(cont'd)*

- An example of a simple linear equation for computing S would be:

$$S = \frac{(z_{far} - |v|)}{z_{far} - z_{near}}$$

- Where:
  - 'z_{far}' and 'z_{near}' is the distance to the far and near clip plane
  - '|v|' is the distance to the camera
    - length of the view vector

V. Spotlight Effect

- A spotlight is made up from a bright inner cone and a dimmer outer cone where the light intensity diminishes as it goes from the inner cone to the outer cone.
- The inner cone is controlled by **Theta**, the outer cone is controlled by **Phi** and the diminishing of the light from the inner to the outer cone is controlled by a **falloff** value.
- The angle is calculated with respect to the light direction vector.

V. Spotlight Effect *(cont'd)*

- If a vertex is found within the inner cone, then the vertex color is computed as usual.
- If it is outside the outer cone, it is not affected by the light, i.e. the spotlight effect value will be zero.
- If the vertex is found to be outside the inner cone but inside the outer cone, then the intensity with the falloff factor is calculated.

V. Spotlight Effect *(cont'd)*

- Let L be the spot light unit vector direction, and D the unit vector from the light source to the vertex.
- If (L.D)<cos(Phi), where Phi is the outer cone angle, then the vertex lies outside the outer cone and the vertex color is not affected by the spot light.
- If (L.D)>cos(Theta), where Theta the inner cone angle, then the vertex lies inside the inner cone and the vertex color receives the maximum amount of light.

V. Spotlight Effect *(cont'd)*

- Finally, if the vertex position is between the inner and outer cone angle then the spotlight effect is:

$$SpotlightEffect = \left(\frac{\cos(Alpha) - \cos(Phi)}{\cos(Theta) - \cos(Phi)} \right)^{P}$$

- Where:
  - Alpha is the angle between L (spotlight direction vector) and D (vector from light position to vertex).
  - Theta is the inner cone angle
  - Phi is the outer cone angle
  - P is the falloff value

VI. Lighting equation

- Ambient component ($I_{ambient} = I_a K_a$)
- Diffuse component ($I_{diffuse} = I_d K_d (N.L)$)
- Specular component ($I_{specular} = I_s K_s (R.V)^{ns}$)
- Emissive component ($I_{emissive}$)
- Total light intensity

$$I_{tot} = I_{emissive} + I_{globalAmbient} K_a + Att * I_{ambient} + Att * SpotlightEffect * (I_{diffuse} + I_{specular})$$

VI. Lighting equation *(cont'd)*

- Total intensity for multiple lights:

$$I_{local} = I_{emissive} + I_{globalAmbient} K_a +$$
$$\sum_{i=0}^{n-1} Att_i * (I_{ambient})_i + Att_i * SpotlightEffect_i * (I_{diffuse} + I_{specular})_i$$

$$I_{final} = SI_{local} + (1 - S) I_{fog}$$

VII. Lighting in OpenGL *(cont'd)*

- Note that when you load the light position or direction data, the data will be transformed by the current matrix in the ModelView stack.
- The light position and/or direction are stored in the view space.
- So, if you define your light in world space, multiply it by the camera matrix before you use the light information.

VII. Lighting in OpenGL *(cont'd)*

- When changing light position per frame (orbiting light)
  - Make sure you set the light position to the new values **every frame**
  - Specified light position will be multiplied by the **current** model-view matrix
  - Light follows the correct camera transformation!

Resources

- OpenGL Lights FAQ (very helpful)
  - http://www.opengl.org/resources/faq/technical/lights.htm
  - Tips and tricks
  - Common pitfalls in code
    - Why are all my objects dark/uniformly lit/not visible?
    - How to render lit and unlit geometry?
    - How to implement a movable light source?
    - And more …
  - http://www.falloutsoftware.com/tutorials/gl/gl8.htm
  - http://www.glprogramming.com/red/

Resources

- OpenGL Shader Designer, OS X
  - IDE for developing shaders (limited capability)
- gDebugger/ GLSLDevil / nVidia Nsight
  - Debug, profile and step through shader code

http://www.opengl.org/sdk/tools/