

# Computer Graphics 2016

## 12. Illumination and shading

Hongxin Zhang

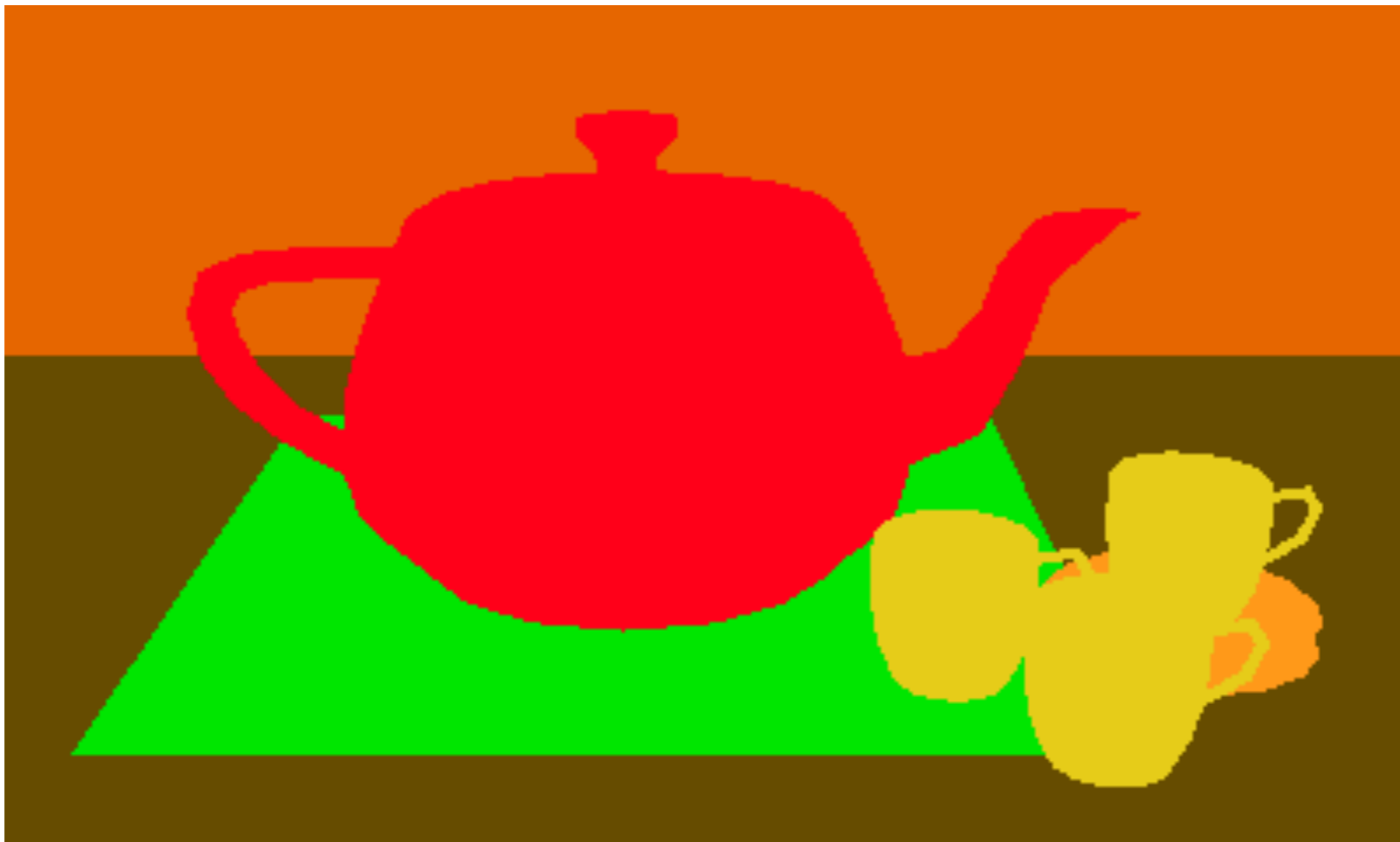
State Key Lab of CAD&CG, Zhejiang University

2016-12-19

# Lighting & Shading

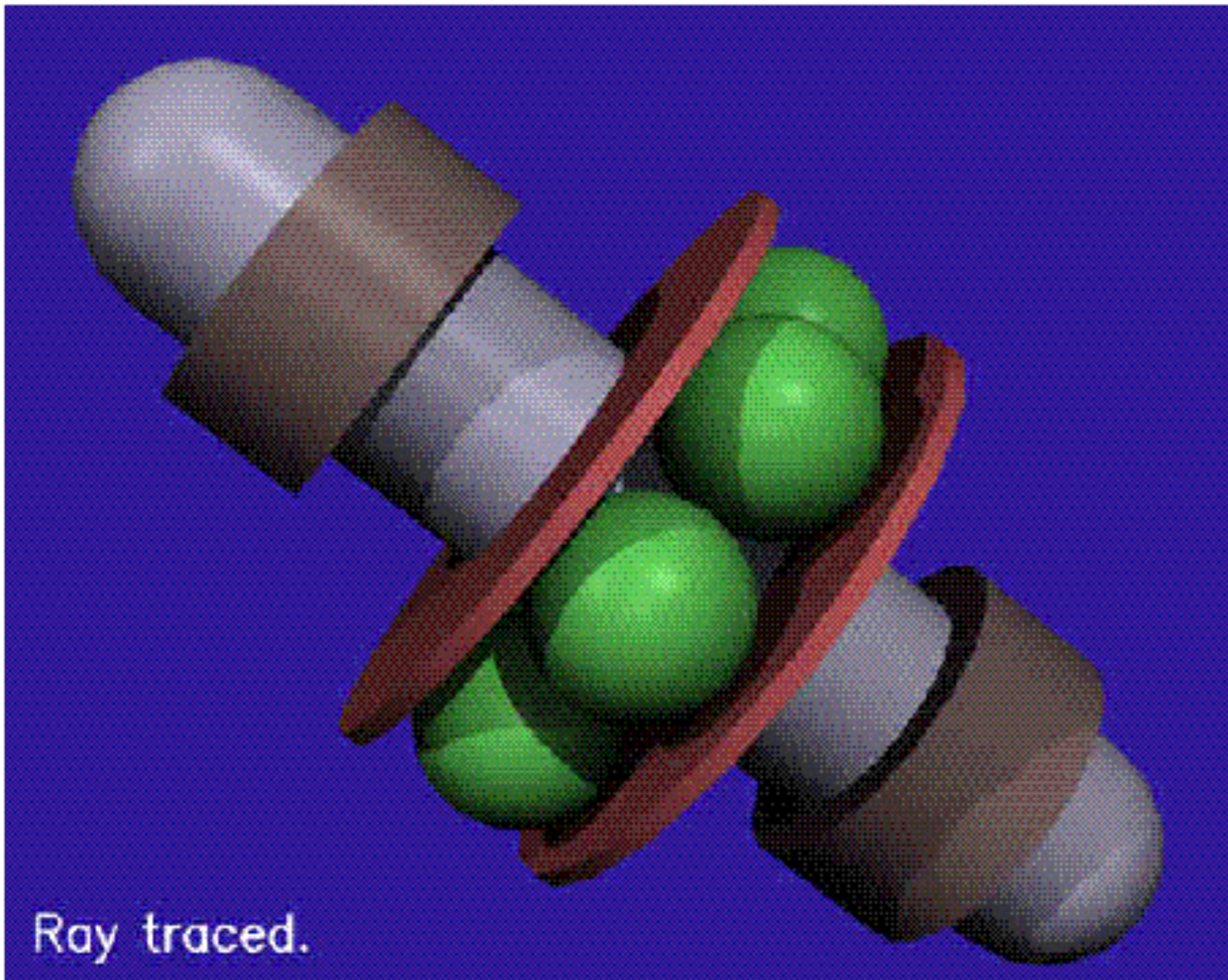
---

Without light ... we don't see much of our scene!



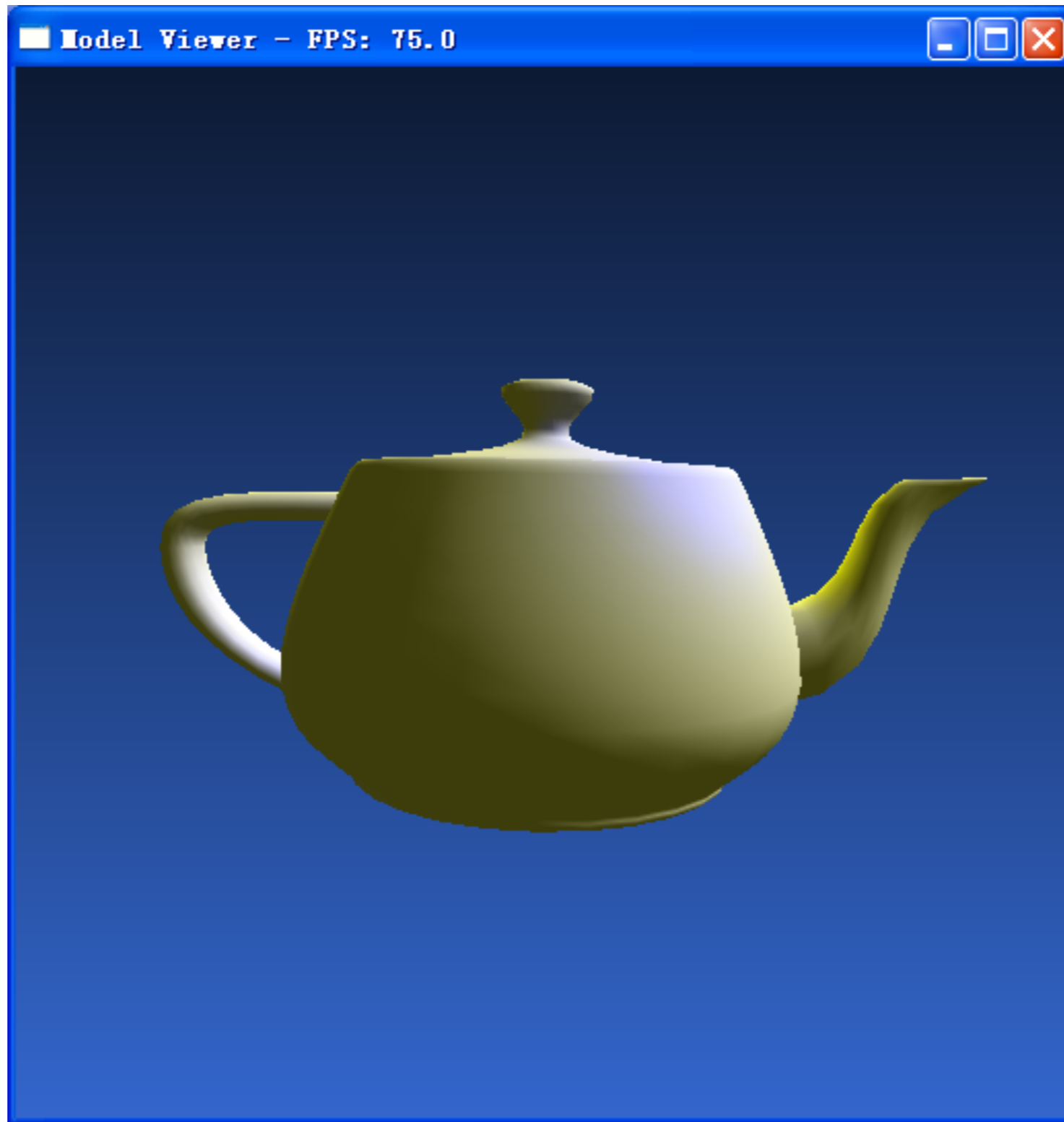
Without shading, objects do not look three dimensional!

# Lighting & Shading



# Demo – Multi-light

---



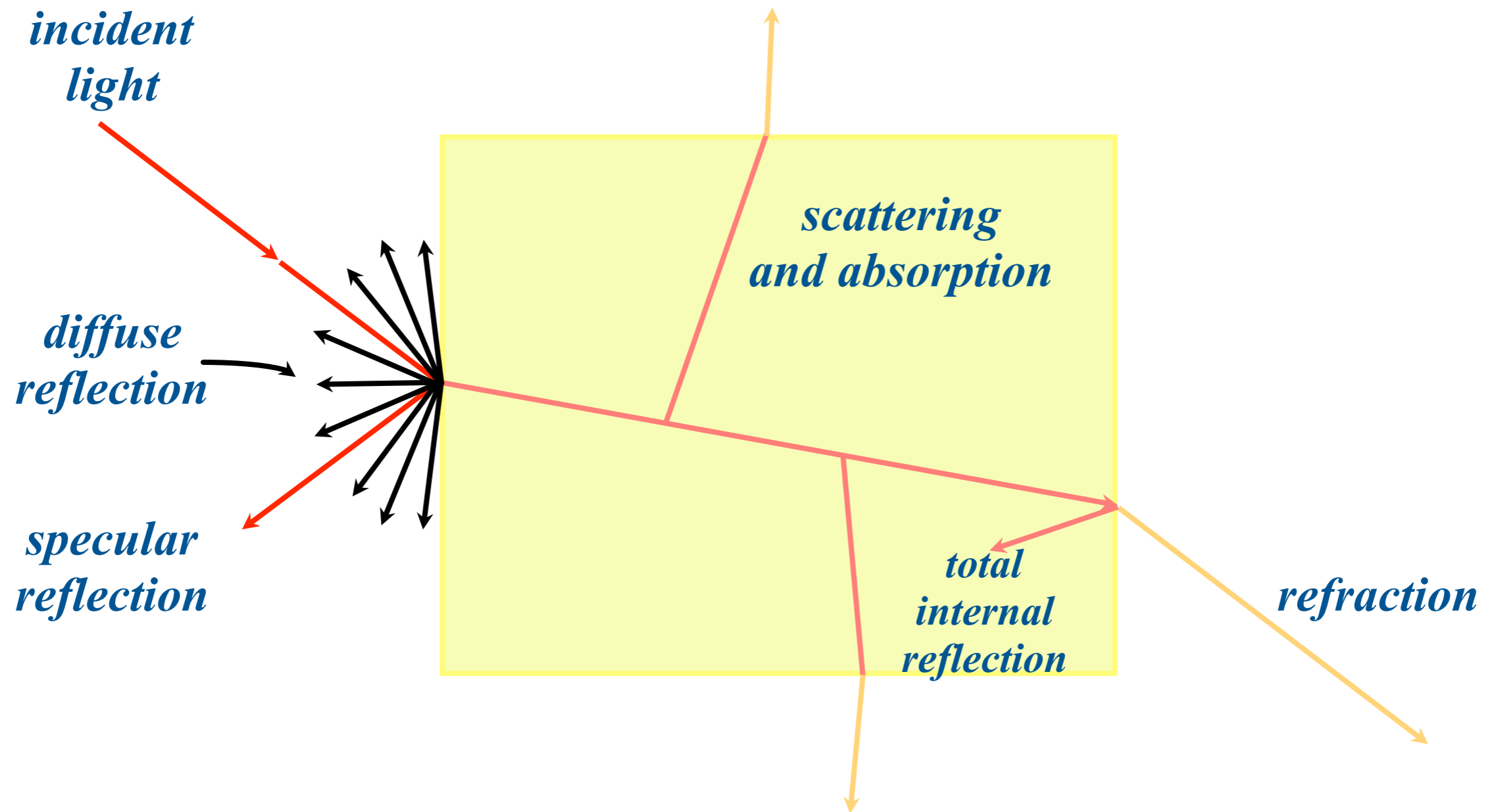
# Illumination

---

- ***Illumination*** is the complete description of all the light striking a particular point on a particular surface
- **Color** at a point on an object is decided by the properties of the light leaving that point
- Knowing the ***illumination*** and the ***surface physics*** at a point on a surface, we can determine the properties of the light leaving that point
- In order to generate realistic images we need to understand how light interacts with the surface of objects

# Interaction of light with a Solid

---





piper



# Interaction of Light

---

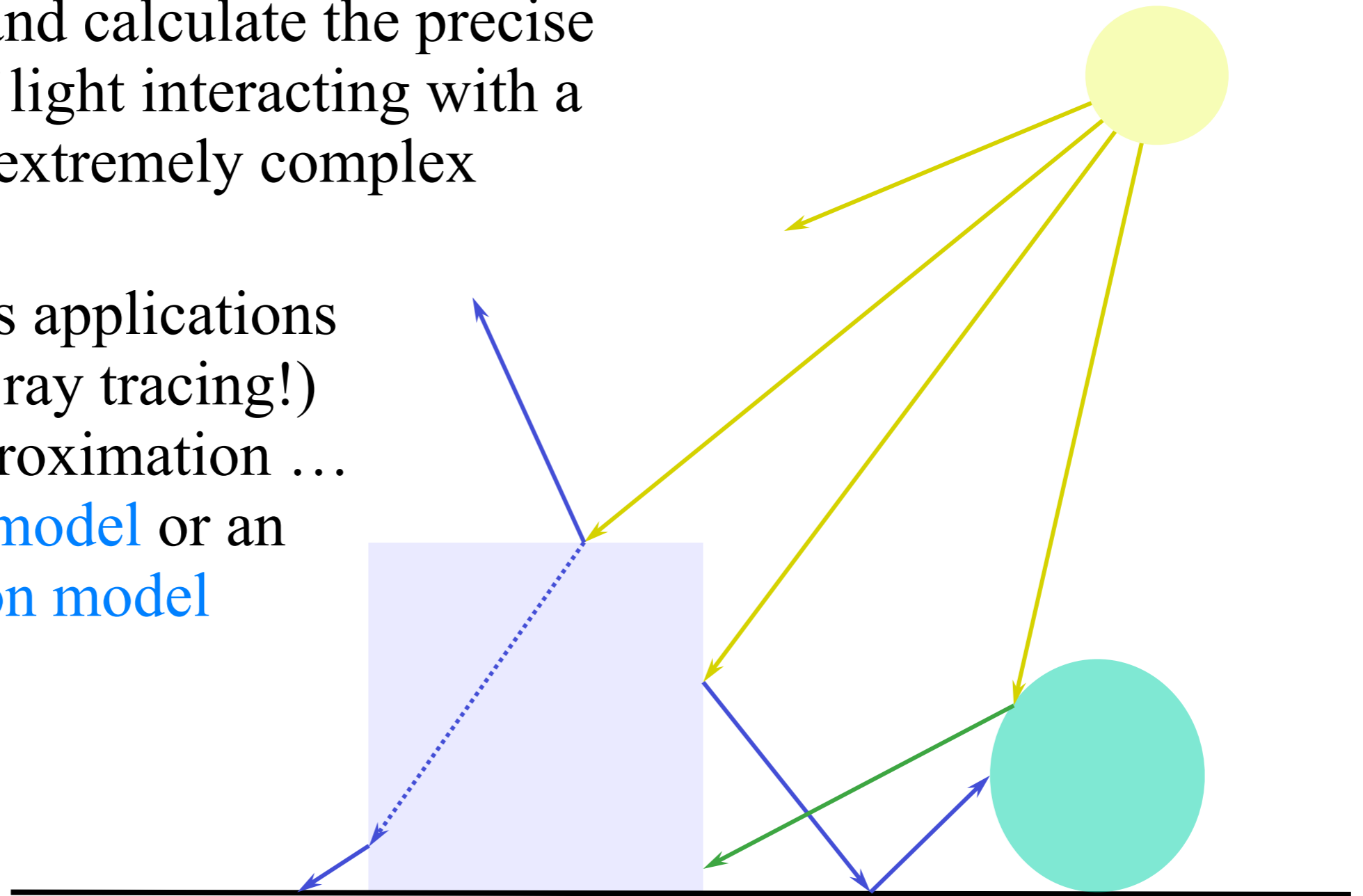
- There are two illumination phenomena of importance
  - **interaction of light** with the boundaries between materials
  - **scattering and absorption of light** as it passes through the material
- Boundaries between materials are surfaces which make up the environment
- Light striking a boundary is either reflected or transmitted. For opaque materials light is absorbed on passing through the boundary



# Light interaction in a Scene

To simulate and calculate the precise physics of light interacting with a surface is extremely complex

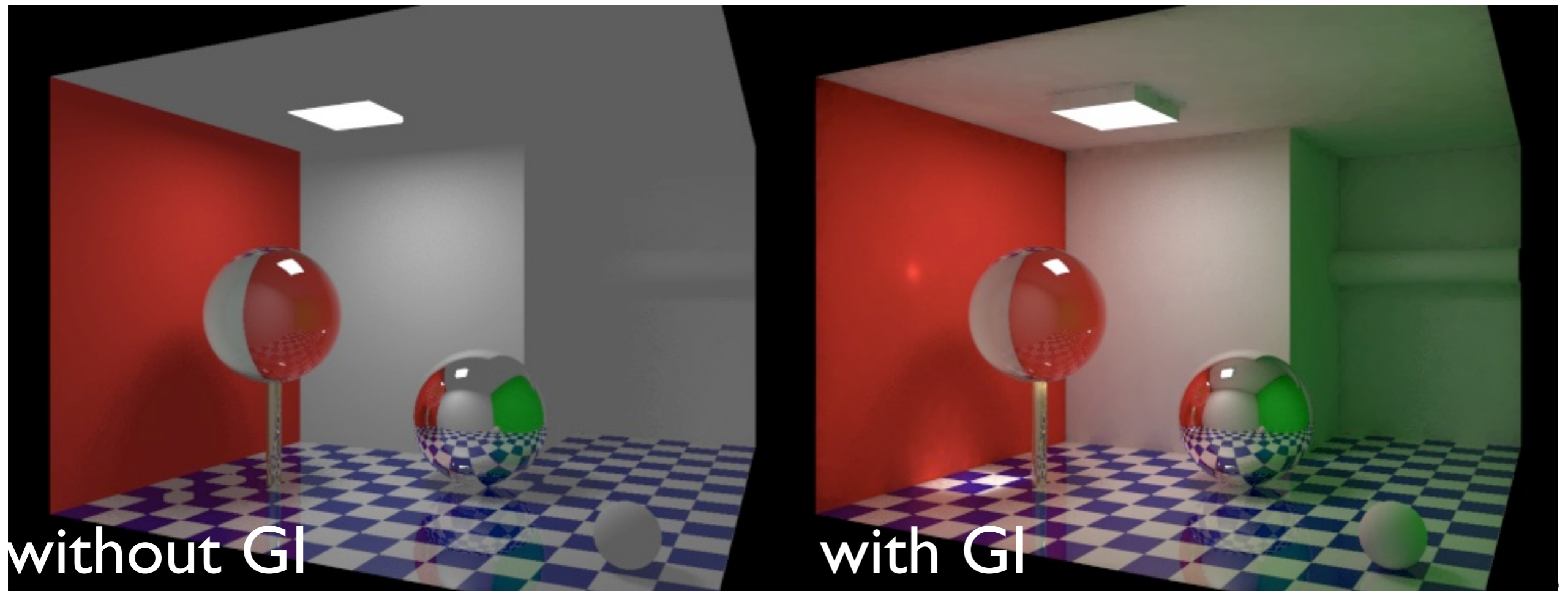
Most graphics applications (including ray tracing!) use an approximation ... a **lighting model** or an **illumination model**



# Illumination models

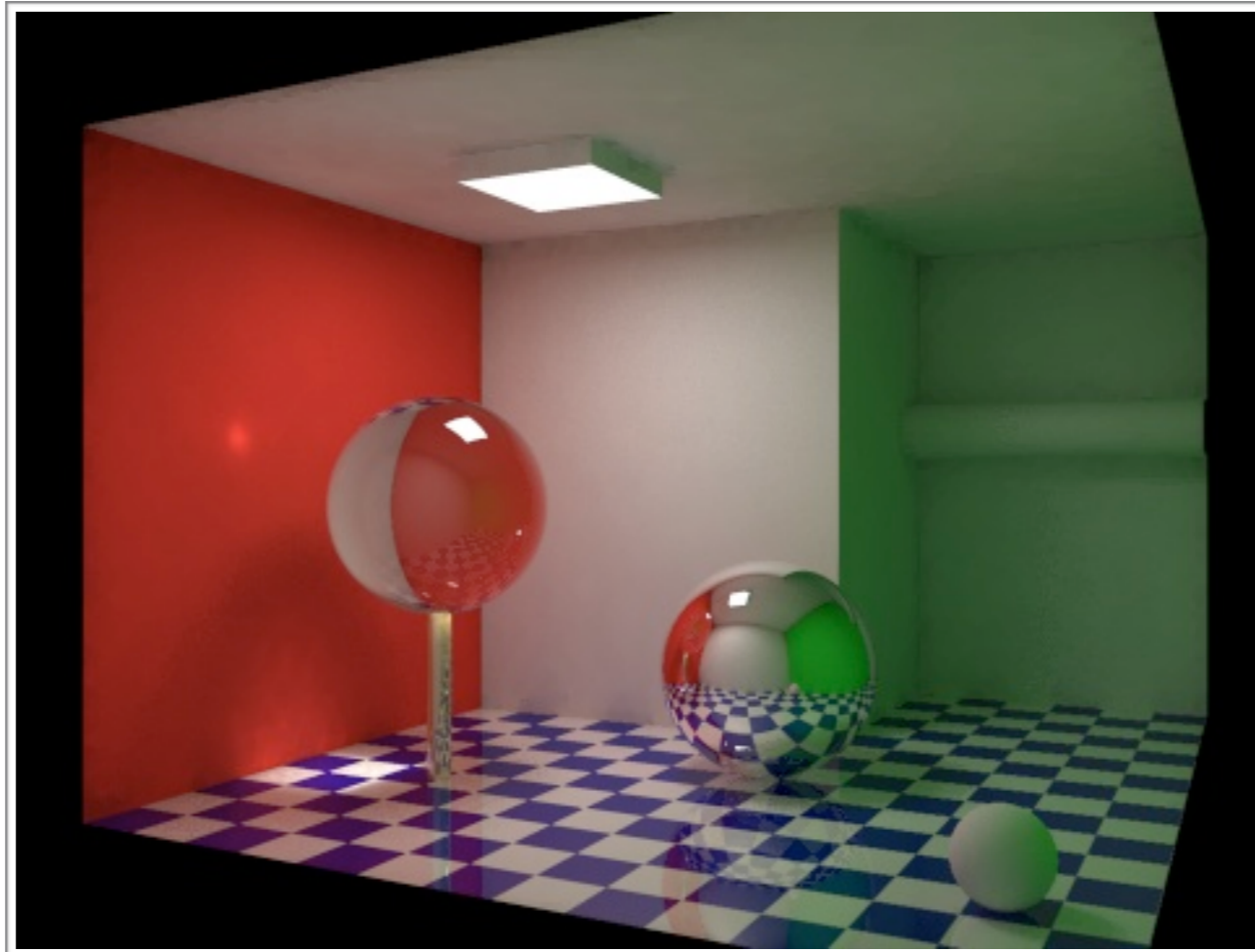
---

- A surface point could be illuminated by
  - *local illumination*, light directly emitted by light sources
  - *global illumination*, light reflected from and transmitted through its own and other surfaces



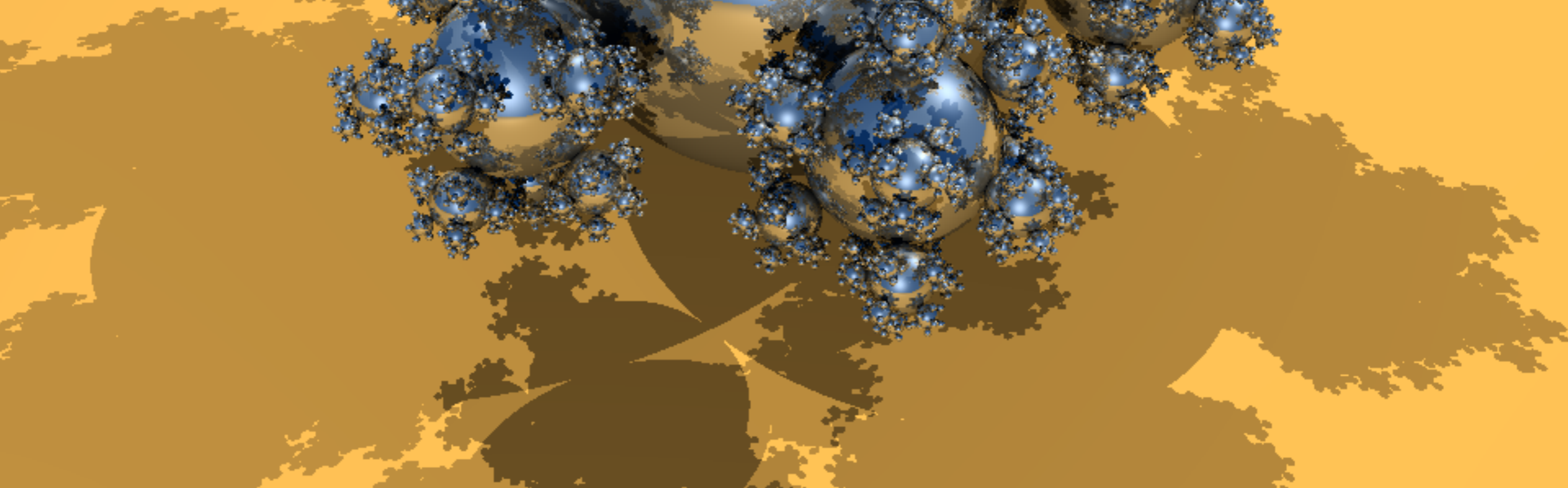
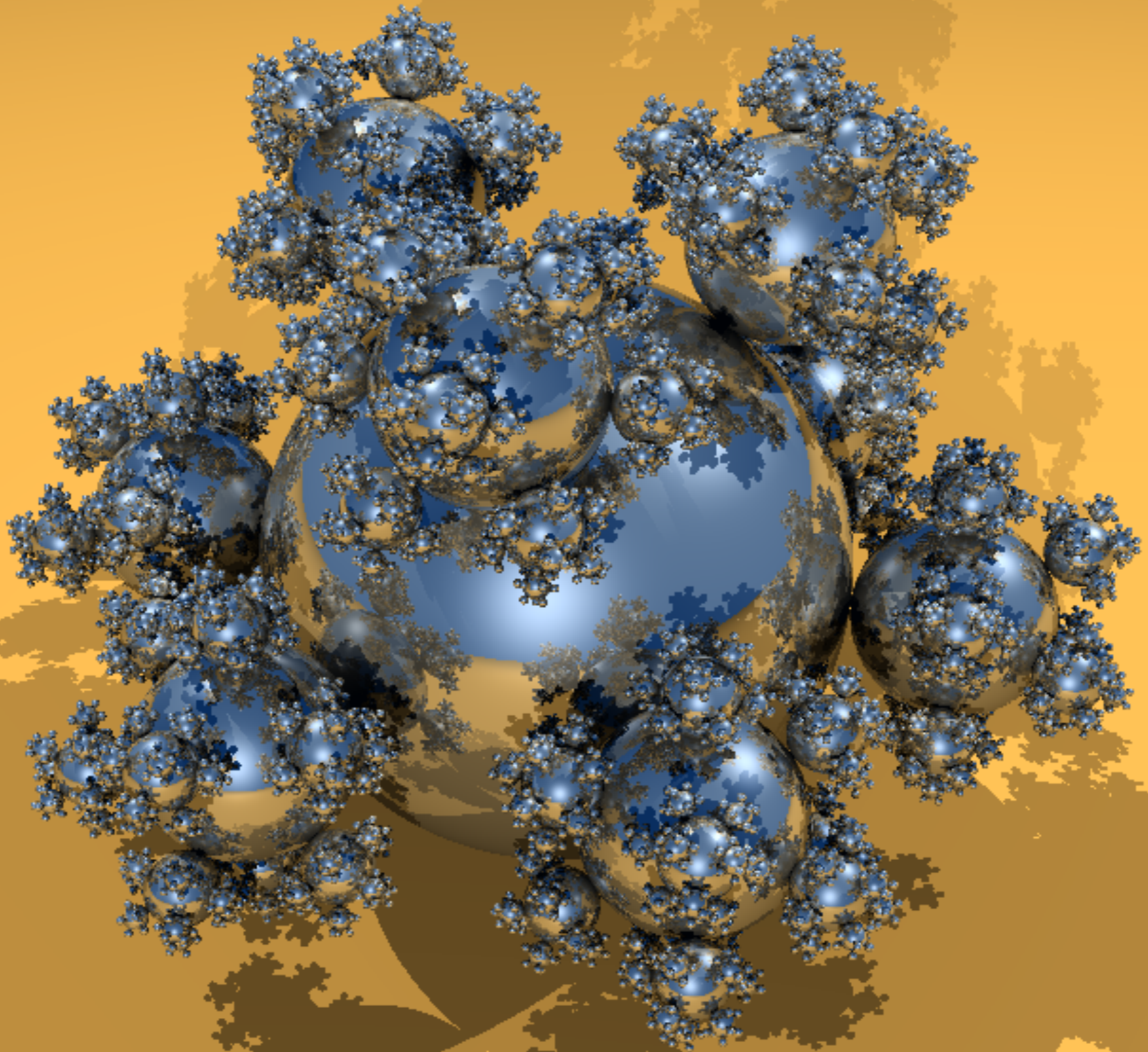
# Illumination models

---



- *Illumination models*

- express the factors which determine the surface color at a given point on a surface
- compute the color at a point in terms of both local and global illumination



# Reflection Models

---

- *Illumination models* used for graphics initially were often approximate models
- Main goal was to model the interaction of light with matter in a way that appears *realistic* and is *fast*
- *Reflection Models* are the simplest of illumination models
- *Reflection Models* assume
  - *local illumination* only. No global illumination
  - light is only *reflected* from the surface. There is no transmission through the object
  - there is *no propagation* media. Surfaces are in vacuum

# Ambient Light

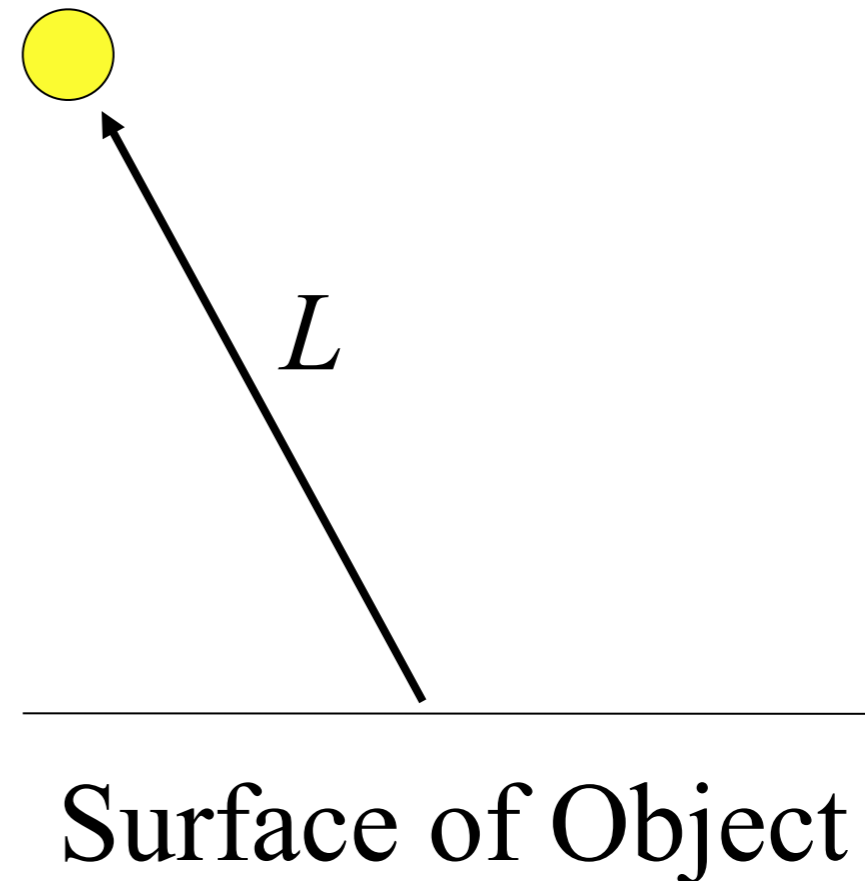
- In reality, parts of the object not directly illuminated by the light source are not completely dark
- These parts are lit by global illumination i.e. light reflected by the surrounding environment; light that is reflected so many times that doesn't seem to come from any where
- This light is approximated by adding a constant light called ambient light
- The ambient light is  $I_a k_a$ , where  $I_a$  is the intensity of ambient light and  $k_a$  is the materials *ambient reflection coefficient* ranging from 0 to 1



# Light Vectors

---

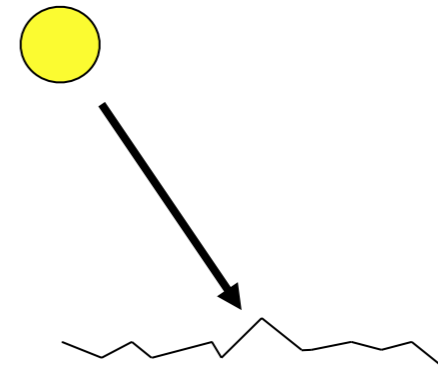
- When considering light, we look at the vector between the light and the object
- Take the vector **from the object to the light!**



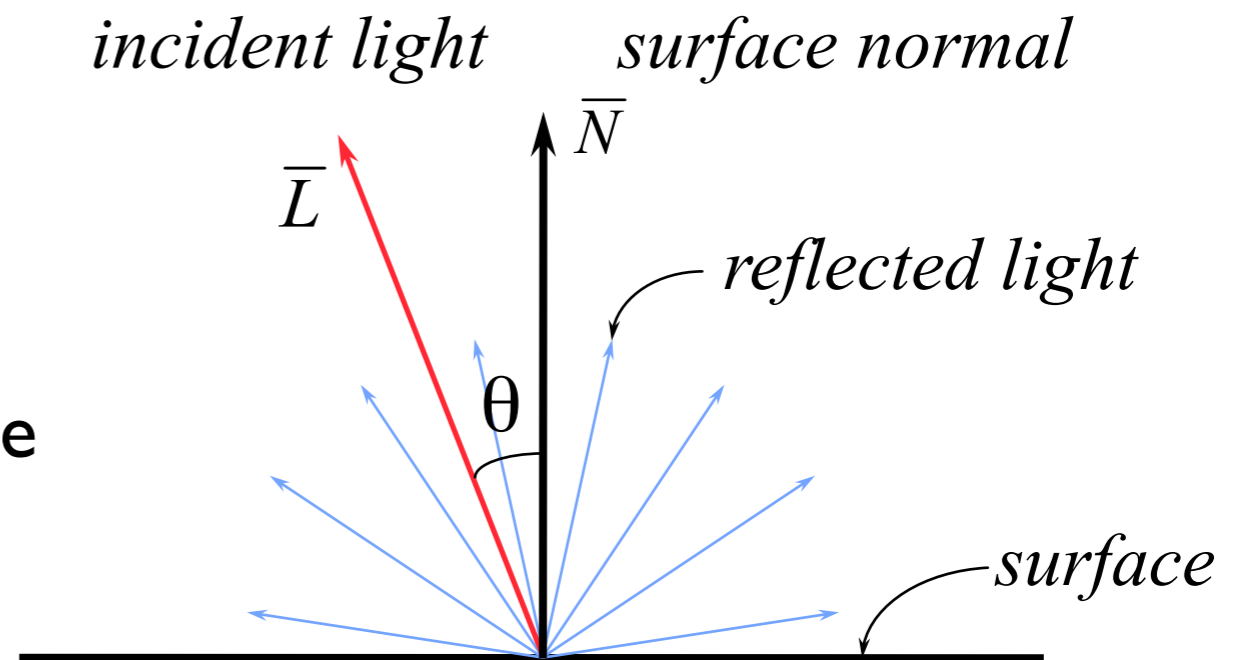
# Diffuse Reflection

- **Diffuse** reflection also called **Lambertian** reflection is a characteristic of dull, matte surfaces like chalk
- Amount of light reflected from a point on the surface is equal in all directions
- The brightness depends on the angle between the direction to the light source and the surface normal
- The brightness is independent of the viewing direction

$$I_d = I_p k_d \cos(\theta)$$



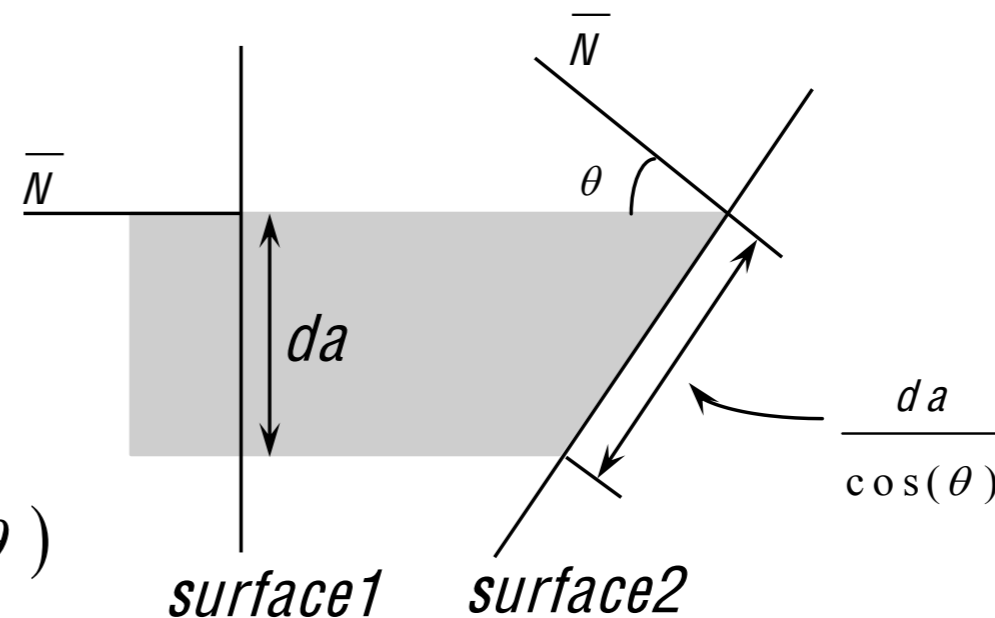
Microscopic view of a rough (matte) surface





# Diffuse Reflection

- If the beam of light has a cross-sectional area  $da$  and is inclined to the surface at an angle  $\theta$  then the beam intercepts the area  $da / \cos(\theta)$



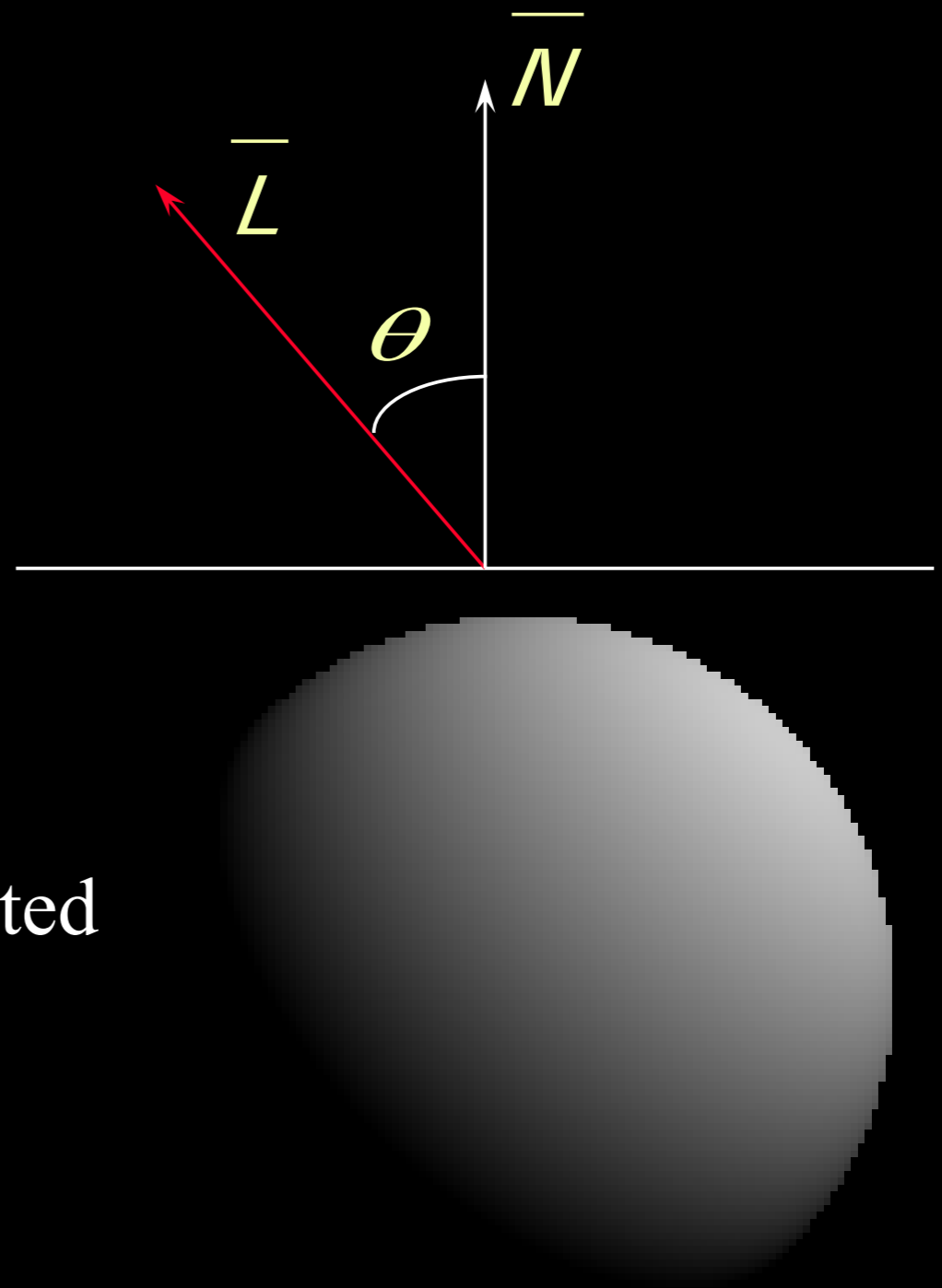
- The amount of light energy that falls on a surface is proportional to  $\cos(\theta)$
- The diffuse illumination equation is  $I_d = I_p k_d \cos(\theta)$  where,  $I_p$  is the point light source intensity,  $k_d$  is the materials diffuse reflection coefficient ranging from 0 to 1. and  $\theta$  must be between  $0^\circ$  and  $90^\circ$  for the point to be lit

# Diffuse Reflection

- With  $\bar{N}$  and  $\bar{L}$  normalized the diffuse illumination equation could also be written as

$$I_d = I_p k_d (\bar{N} \cdot \bar{L})$$

- With the diffuse reflection the part of the surface not illuminated by the light is dark



# Illumination equation

- By adding the ambient term to the diffuse reflection the illumination equation is

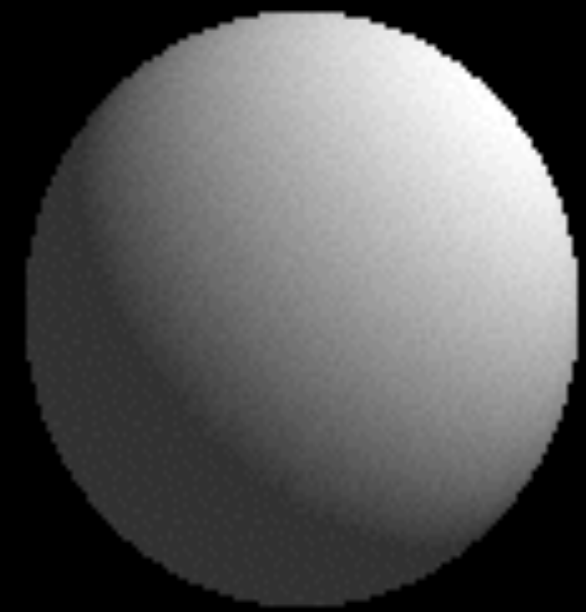
$$I = I_a k_a + I_p k_d (\bar{N} \cdot \bar{L})$$



$$k_a = 0.2, \quad k_d = 0.4$$



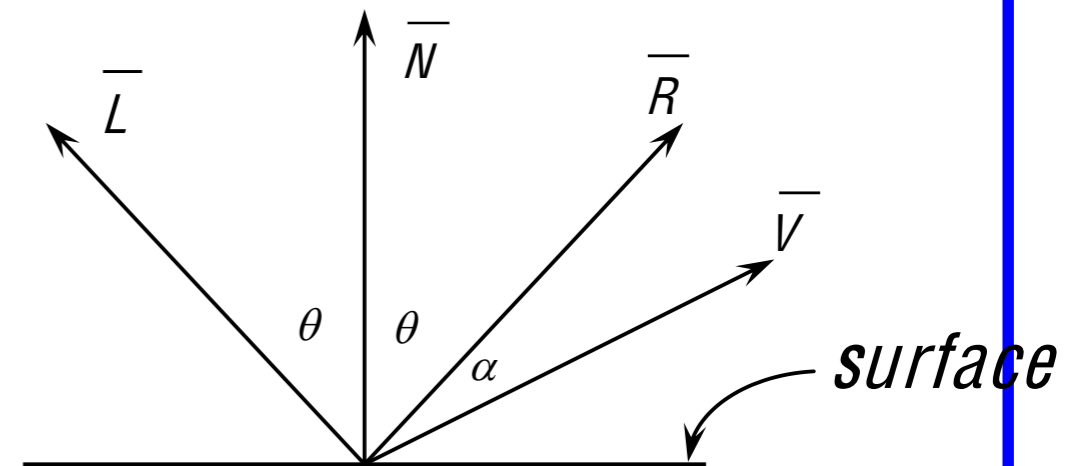
$$k_a = 0.2, \quad k_d = 0.6$$



$$k_a = 0.2, \quad k_d = 0.8$$

# Specular Reflection

- *Specular reflection* is exhibited by shiny surfaces like plastics
- In case of a perfect mirror light is reflected only in the reflection direction  $\vec{R}$ . For the viewer to see the reflected light, his direction  $\vec{V}$  should be same as the reflected direction i.e. angle  $\alpha = 0$
- Natural shiny surfaces reflect light unequally in different directions
- This is captured by the *Phong illumination model*



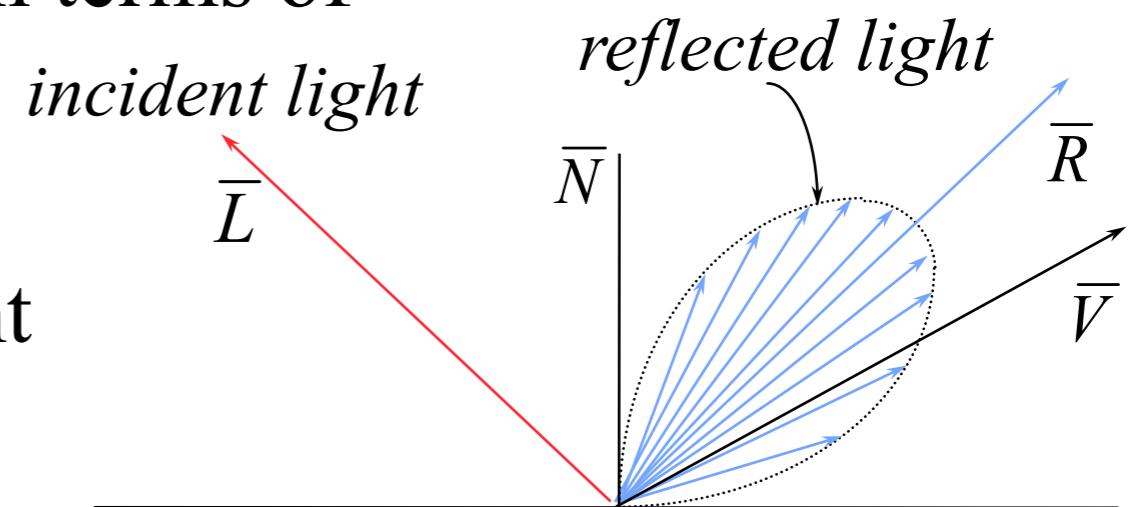
# Phong Illumination Model

---

- These two types of reflection together make up the phong illumination model
  - Diffuse reflectance
  - Specular reflectance
  
- So, lighting depends on the properties of the surface as well as the light itself
  - Light Properties
  - Material Properties

# Phong Illumination Model

- Developed by Phong Bui-Tuong (1975) is a popular model for non-perfect reflectors
- Specular reflection of shiny objects is considered. It assumes that maximum specular reflection occurs at  $\alpha = 0$
- The light calculation depends on the viewing direction
- Reflected intensity is modeled in terms of
  - Ambient component
  - Diffuse reflection component
  - Specular reflection component



# Phong Illumination model

- The illumination equation in its simplest form is given as

$$I = I_a k_a + I_p k_d (\overline{N} \cdot \overline{L}) + I_p k_s (\overline{R} \cdot \overline{V})^n$$

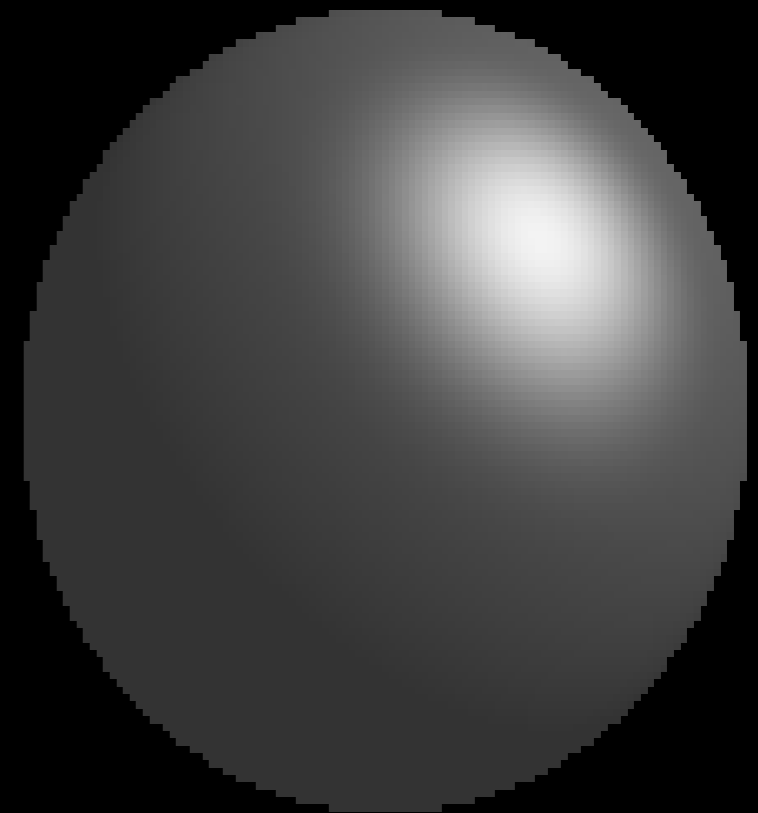
where,  $k_s$  is the materials  
*specular reflection coefficient*

ranging between 0 and 1,  $n$  is  
the *specular reflection exponent*

$k_a$  is the material property the object,

$\overline{R}$  is the *reflected ray vector*

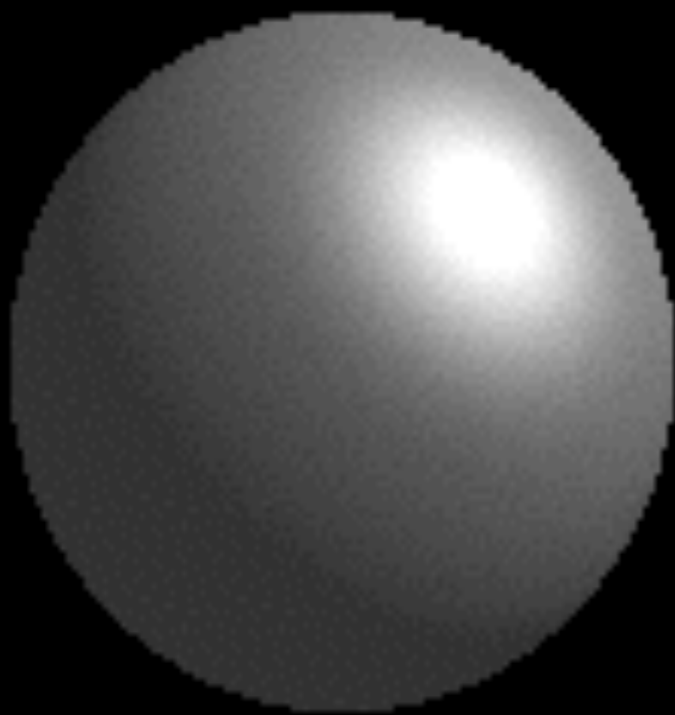
and  $\overline{V}$  is the *view vector*



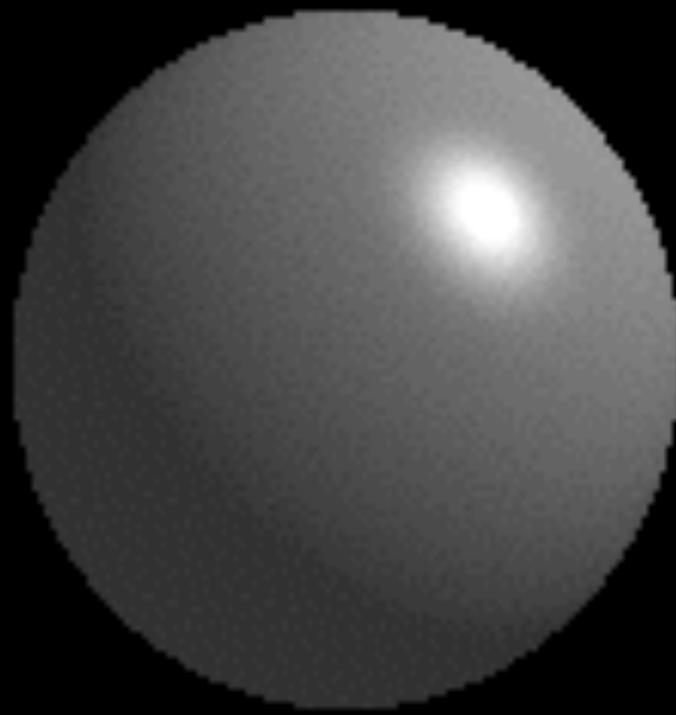
$$k_a = 0.2, \quad k_d = 0.2, \quad k_s = 0.6, \quad n = 4$$

# Phong Illumination model

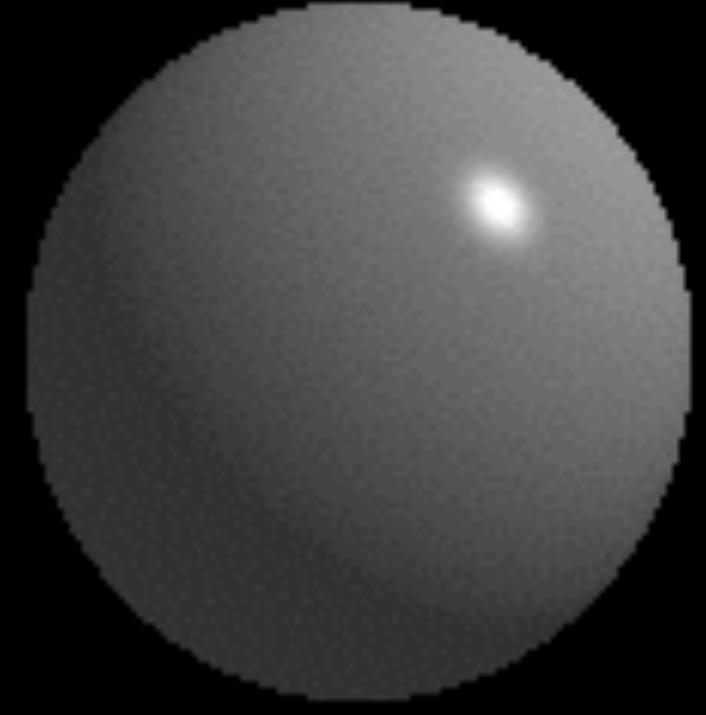
- The *specular reflection exponent*  $n$  controls the shine/gloss of the surface. The surface is more glossy as  $n$  increases and the shine becomes sharper



$n = 4$



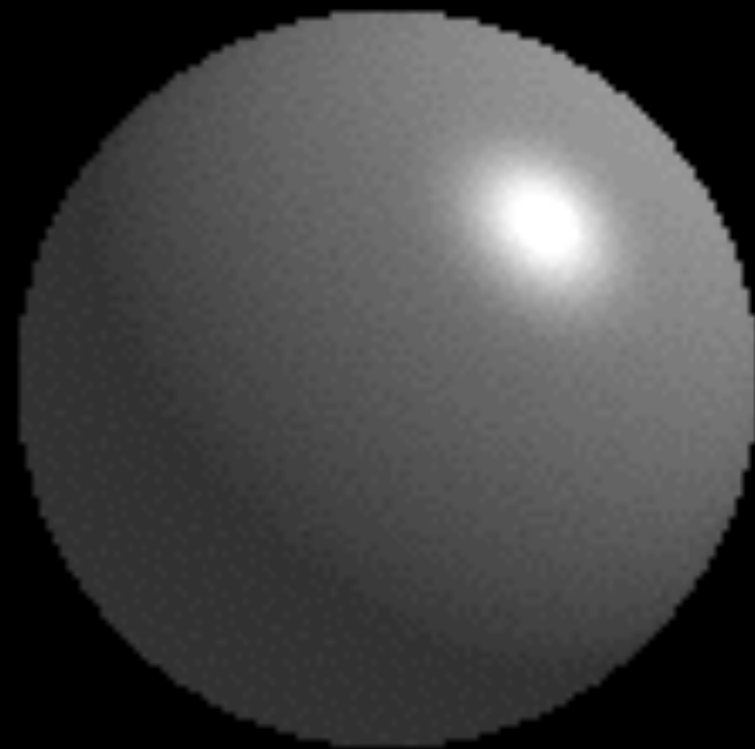
$n = 16$

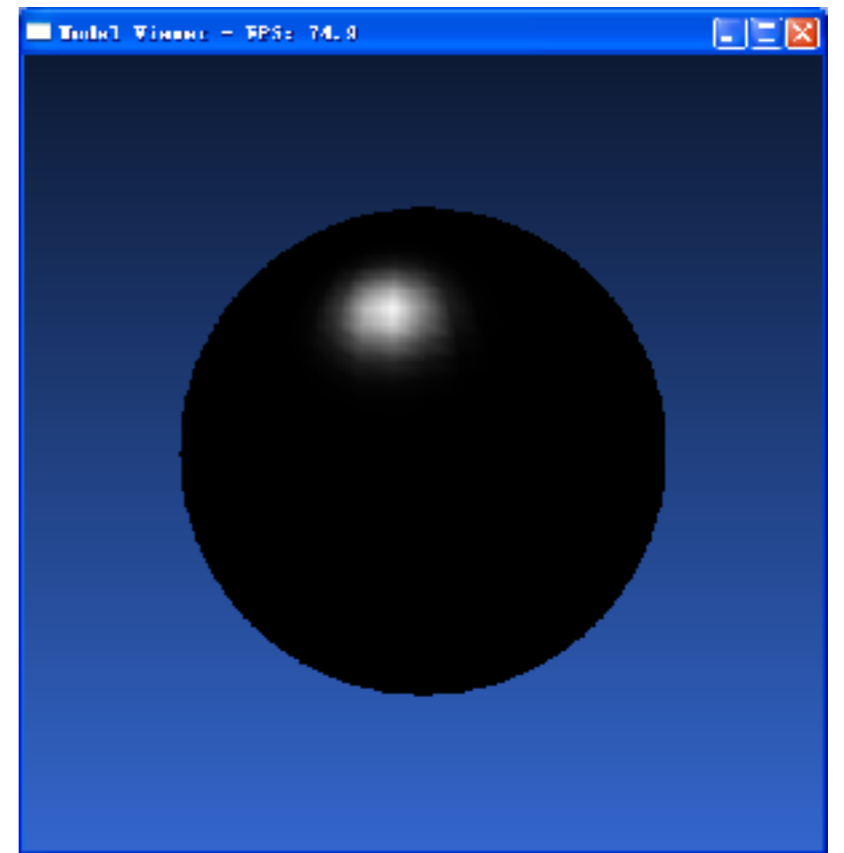
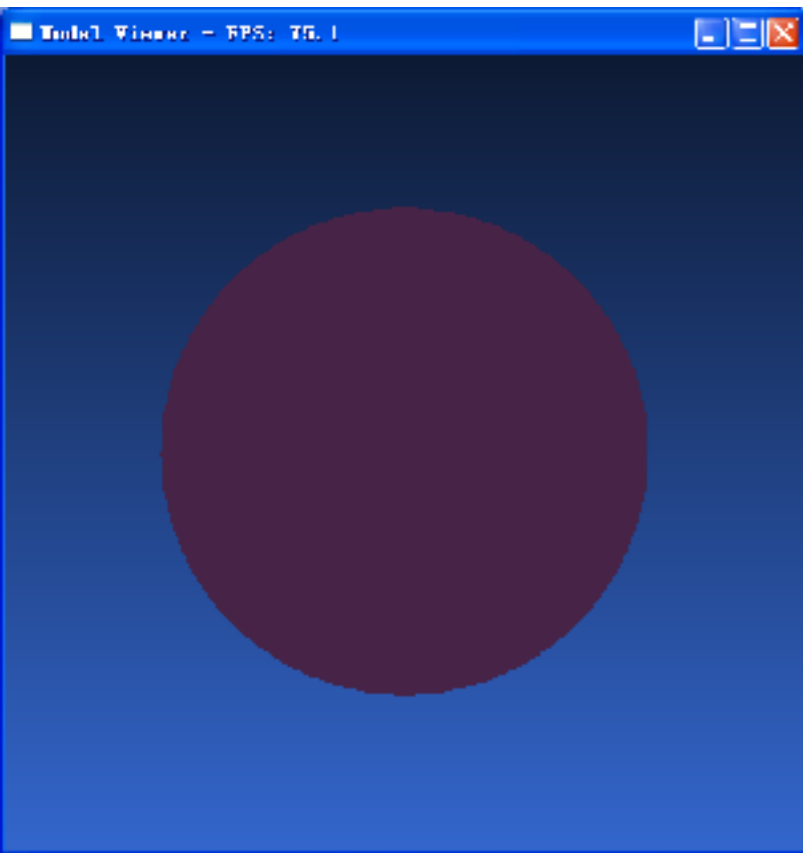
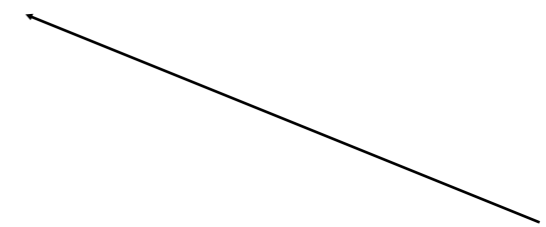
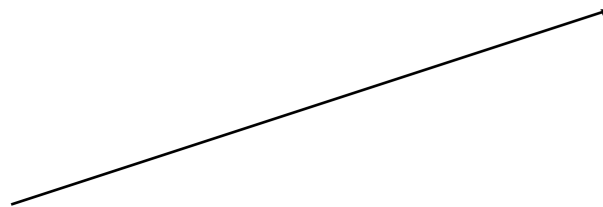
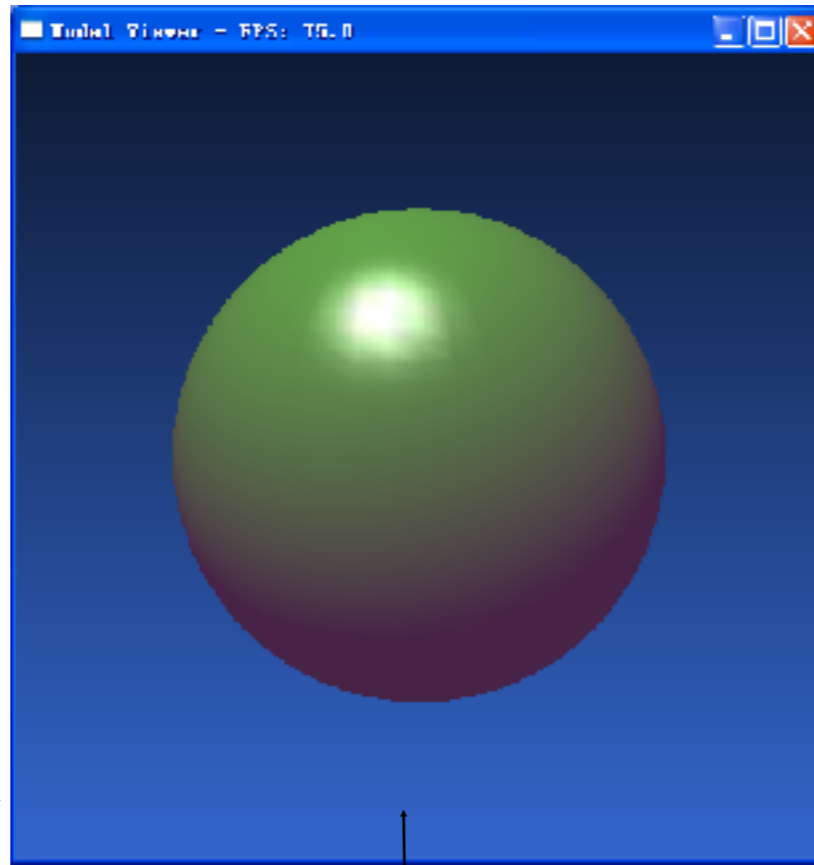


$n = 64$



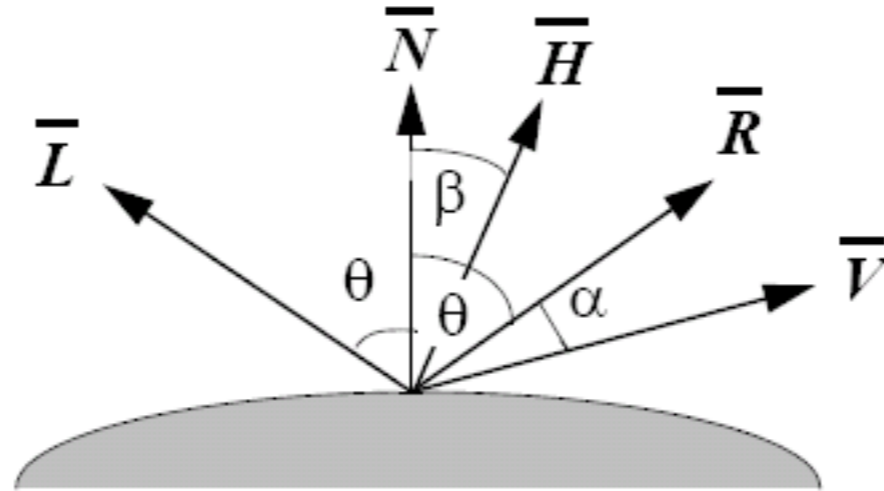
# Difference between illumination models





# Halfway vector

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{\|\mathbf{L} + \mathbf{V}\|}$$



- H is used to simplify the computation of  $\text{dot}(\mathbf{V}, \mathbf{R}) \Rightarrow \text{dot}(\mathbf{H}, \mathbf{N})$

# Multiple Light Sources

- In case of multiple light sources, the terms for each light source are summed. So for  $m$  light sources the illumination equation is

$$I = I_a k_a + \sum_{1 \leq i \leq m} I_{pi} [k_d (\bar{N} \cdot \bar{L}_i) + k_s (\bar{R}_i \cdot \bar{V})^n]$$

- With summation it is possible that the value of  $I$  might exceed the maximum displayable pixel value
  - it could be avoided with proper selection of material
  - the resulting value of  $I$  could be clamped to the maximum value
  - divide each pixel value by the peak value of  $I$

# Colored objects

- The color of objects is set by appropriate setting of the ambient and the diffused reflection coefficients
- Specular coefficient is not decided by the color
- There are now three intensity equations

$$I_r = I_a k_{ar} + I_p [k_{dr} (\bar{N} \bullet \bar{L}) + k_s (\bar{R} \bullet \bar{V})^n]$$

$$I_g = I_a k_{ag} + I_p [k_{dg} (\bar{N} \bullet \bar{L}) + k_s (\bar{R} \bullet \bar{V})^n]$$

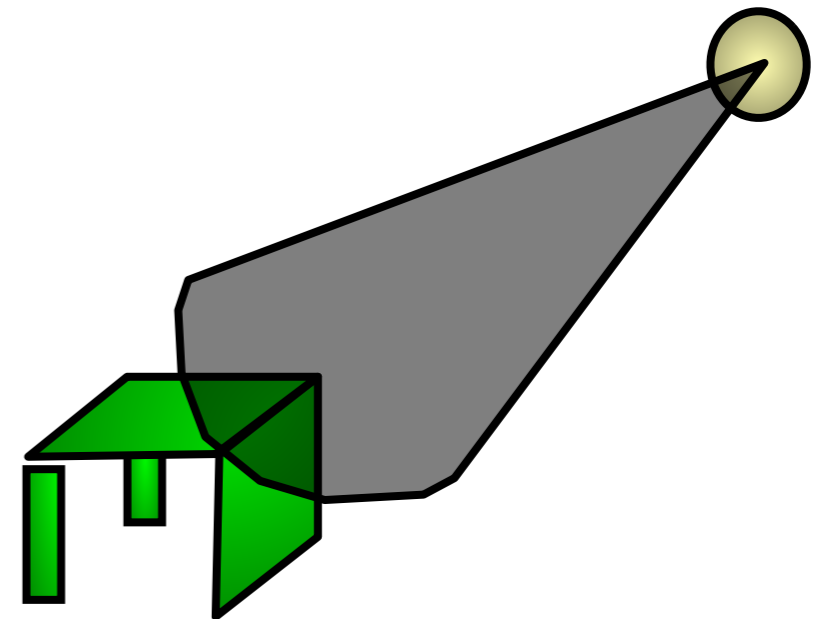
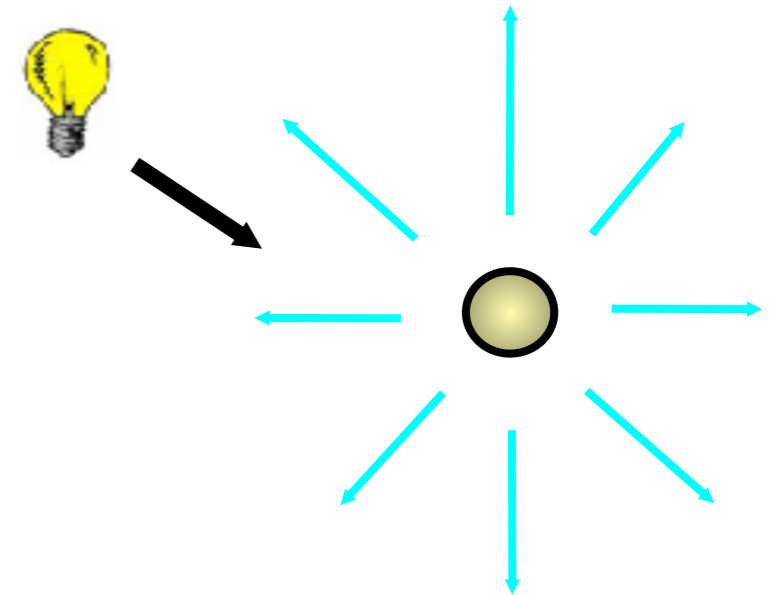
$$I_b = I_a k_{ab} + I_p [k_{db} (\bar{N} \bullet \bar{L}) + k_s (\bar{R} \bullet \bar{V})^n]$$

- Summarizing these three equations as single expression

$$I(r, g, b) = I_a k_a(r, g, b) + I_p [k_d(r, g, b) (\bar{N} \bullet \bar{L}) + k_s (\bar{R} \bullet \bar{V})^n]$$

# OpenGL Lighting Model

- The OpenGL lighting model is simple.
- Types of lights
  - **Ambient light** is light that has been reflected so much that it doesn't seem to come from anywhere and illuminates from all directions equally
  - **Point lights** – rays emanate in all directions. Small compared to objects in the scene
  - **Spot lights** – rays emanate in a narrow range of angles



# Lights in OpenGL

---

- Most implementations of OpenGL can have up to 8 lights in the scene
  - Each light can have a diffuse and a specular component
  - Each light can also have an ambient component (light that is reflected off of so many surfaces, we can't tell where it comes from)
  - Lights are referred to by the macros `GL_LIGHT0`, `GL_LIGHT1`, ..., `GL_LIGHT7` (???)
  - We set the properties of lights with calls to the function “`glLightfv`” (v stands for vector)

# Lights in OpenGL

---

- `glLightfv(Light #, Property, Array of Vals)`
  - Light # = One of `GL_LIGHTi`
  - Property, one of
    - `GL_AMBIENT`
    - `GL_DIFFUSE`
    - `GL_SPECULAR`
    - `GL_POSITION` = where is the light?
  - Array of values = value for the property



# Shading

---

- Shading is the process of determining the colors of all the pixels covered by a surface using an illumination model
- Simplest method is to
  - determine surface visible at each pixel
  - compute normal of the surface
  - evaluate light intensity and color using an illumination model
- This is quite expensive. The shading methods could be made efficient by customizing for specific surface representation

# Shading Models

---

- Shading Models give a technique to determine the colors of all the pixels covered by a surface using appropriate illumination model
- Polygonal meshes are commonly used for representing complex surfaces
- The geometric information is available only at the vertices of a polygon
- Interpolative shading models could be used to increase the efficiency substantially

# Constant Shading

---

- It is the simplest of the shading models and is also called as *faceted shading* or *flat shading*
- One polygon receives only one intensity value
- Illumination model is applied only once for each polygon
- Makes the following assumptions
  - light source is at infinity, so  $\bar{N} \cdot \bar{L}$  is constant across a polygon face
  - viewer is at infinity, so  $\bar{R} \cdot \bar{V}$  is constant across the polygon face
  - polygon represents the actual surface being modeled

# Constant Shading

- It is a fast technique for shading as it involves very less calculations
- If the polygons are very small (say one pixel large) when projected on the screen then the result is as good as any interpolative technique
- Usually used for coarse preview of scenes



OpenGL uses the normal of the **first vertex** of a single polygon to determine the color.

```
glShadeModel(GL_FLAT);
```

# Gouraud Shading

---

- It is an interpolative shading method, also called *intensity interpolation shading* or *color interpolation shading*
- Involves the following steps
  - Normals are computed at the vertex as the average of the normals of all the faces meeting at that vertex
  - Intensity at each vertex is calculated using the normal and an illumination model
  - For each polygon the intensity values for the interior pixels are calculated by linear interpolation of the intensities at the vertices

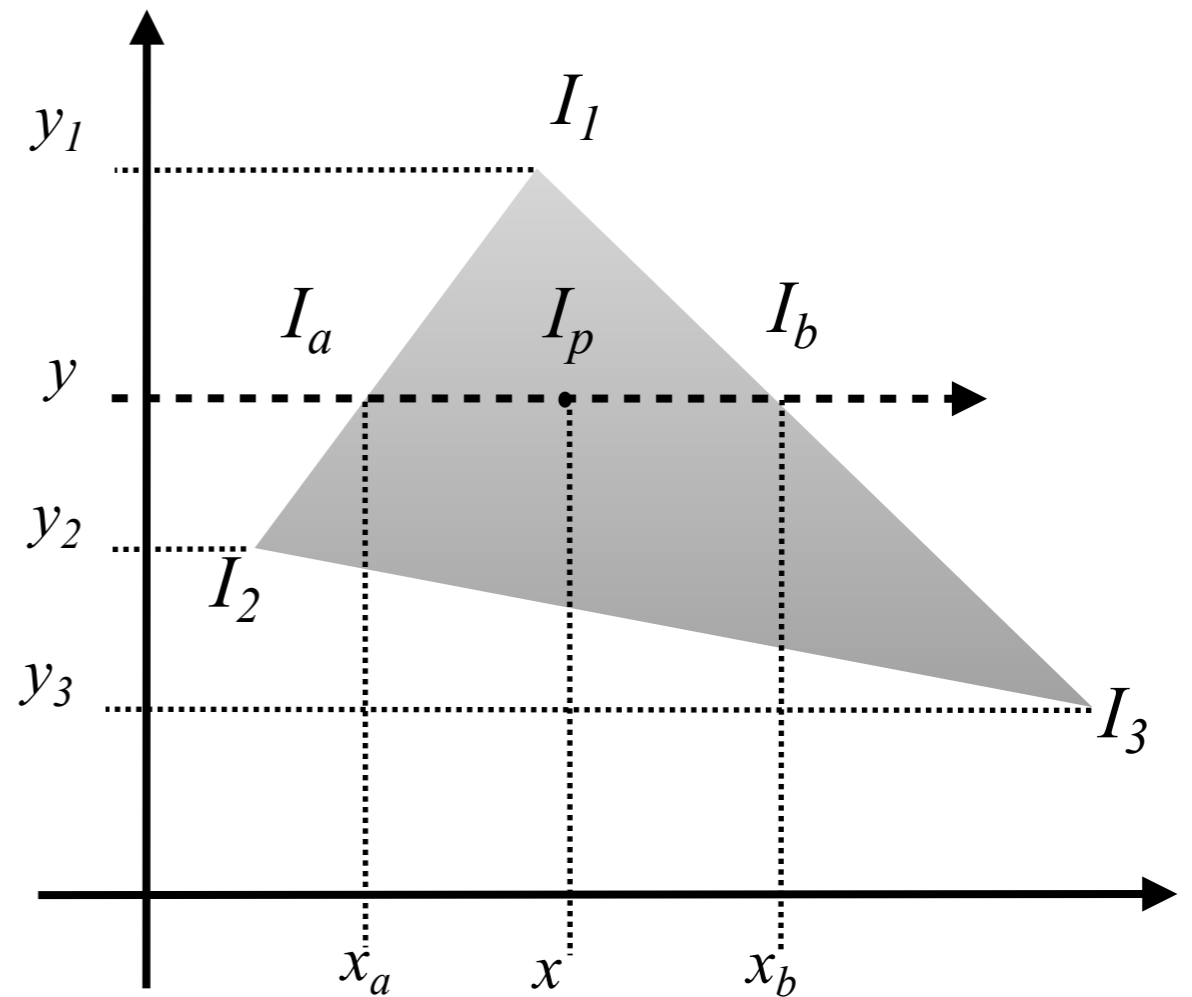
# Gouraud Shading

- Intensity  $I_p$  at a point is calculated as

$$I_a = I_1 + (I_2 - I_1) \frac{y - y_1}{y_2 - y_1}$$

$$I_b = I_1 + (I_3 - I_1) \frac{y - y_1}{y_3 - y_1}$$

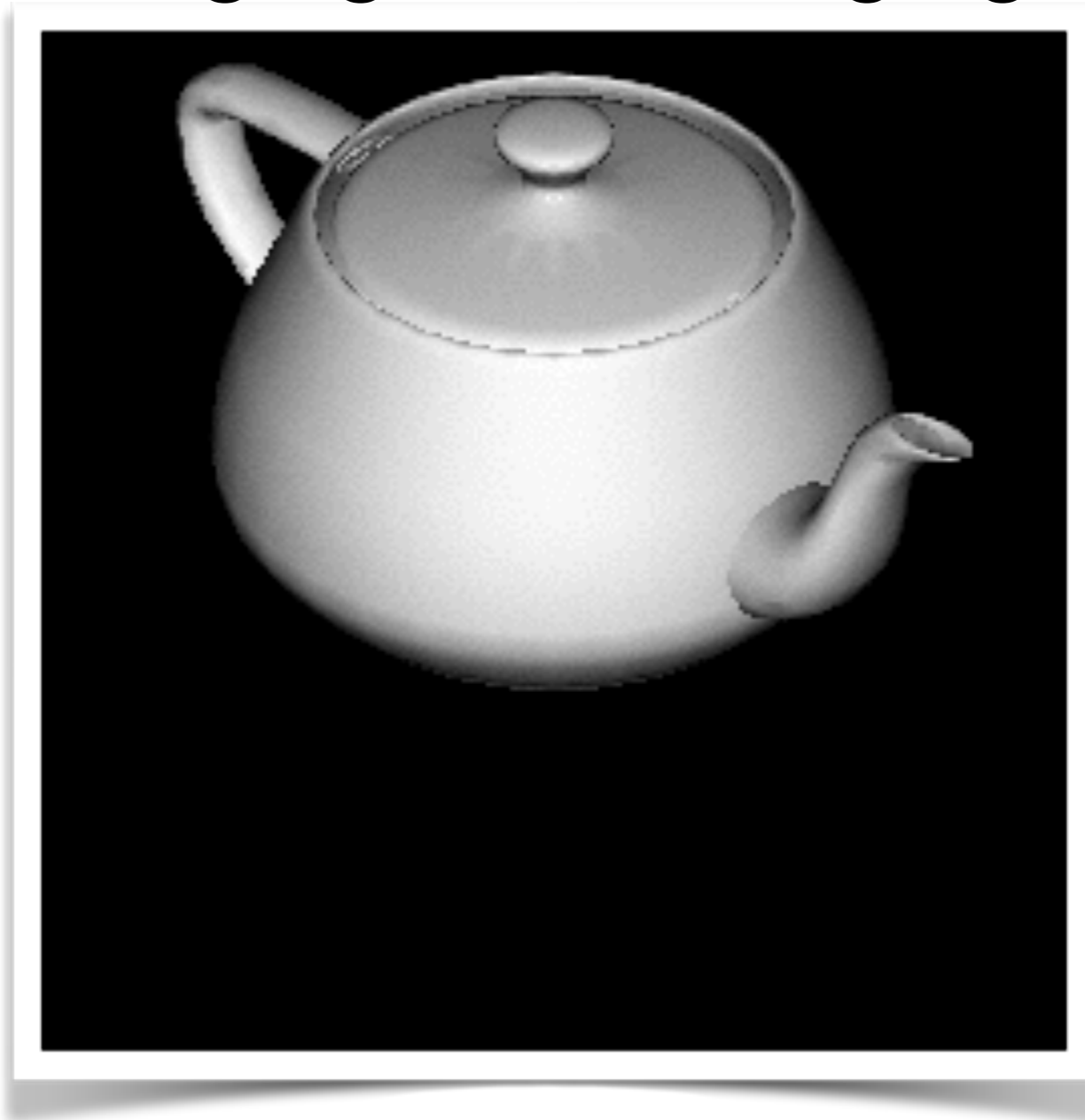
$$I_p = I_a + (I_b - I_a) \frac{x - x_a}{x_b - x_a}$$



# Gouraud Shading

---

Might miss specular highlights, if the highlight doesn't fall at the vertex



```
glShadeModel (GL_SMOOTH) ;
```

# Phong Shading

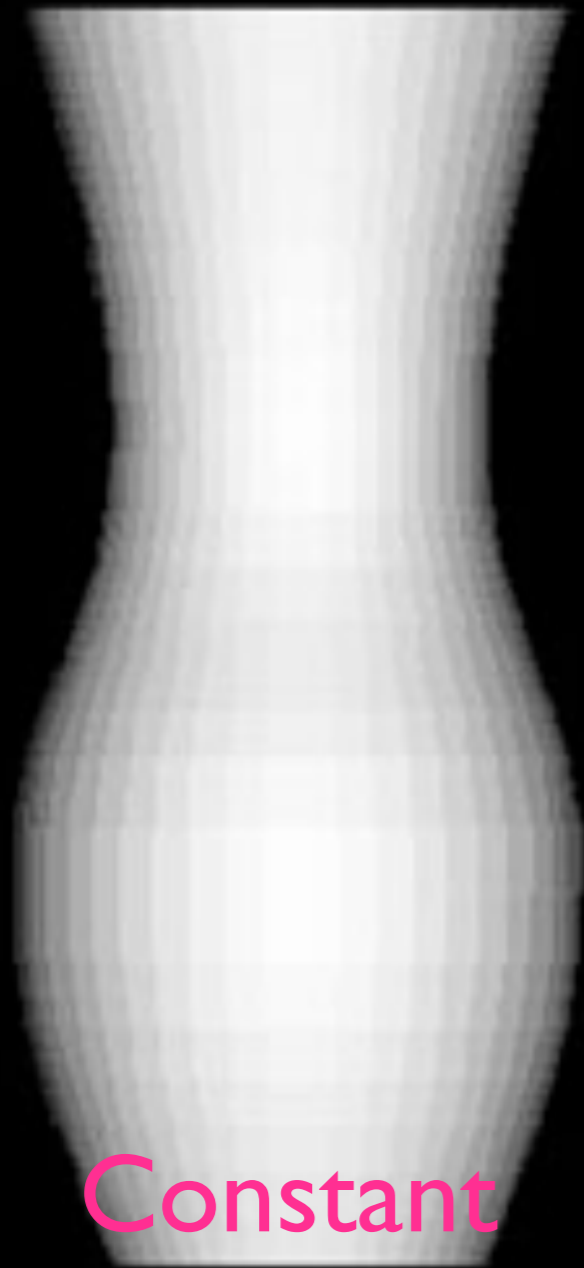
---

- It is an interpolative shading method, also called:
  - **normal-vector interpolation shading**
- Involves the following steps
  1. **Normals** are computed at the vertex as the **average** of the normals of **all the faces** meeting at that vertex
  2. For each polygon the value of the **normal** for the surface occupied by each interior pixel is calculated by **linear interpolation** of the normals at the vertices
- Specular reflections are also incorporated
- Interpolation of normals is done exactly like intensity interpolation in Gouraud shading



# Phong Shading

---



Constant  
Shading



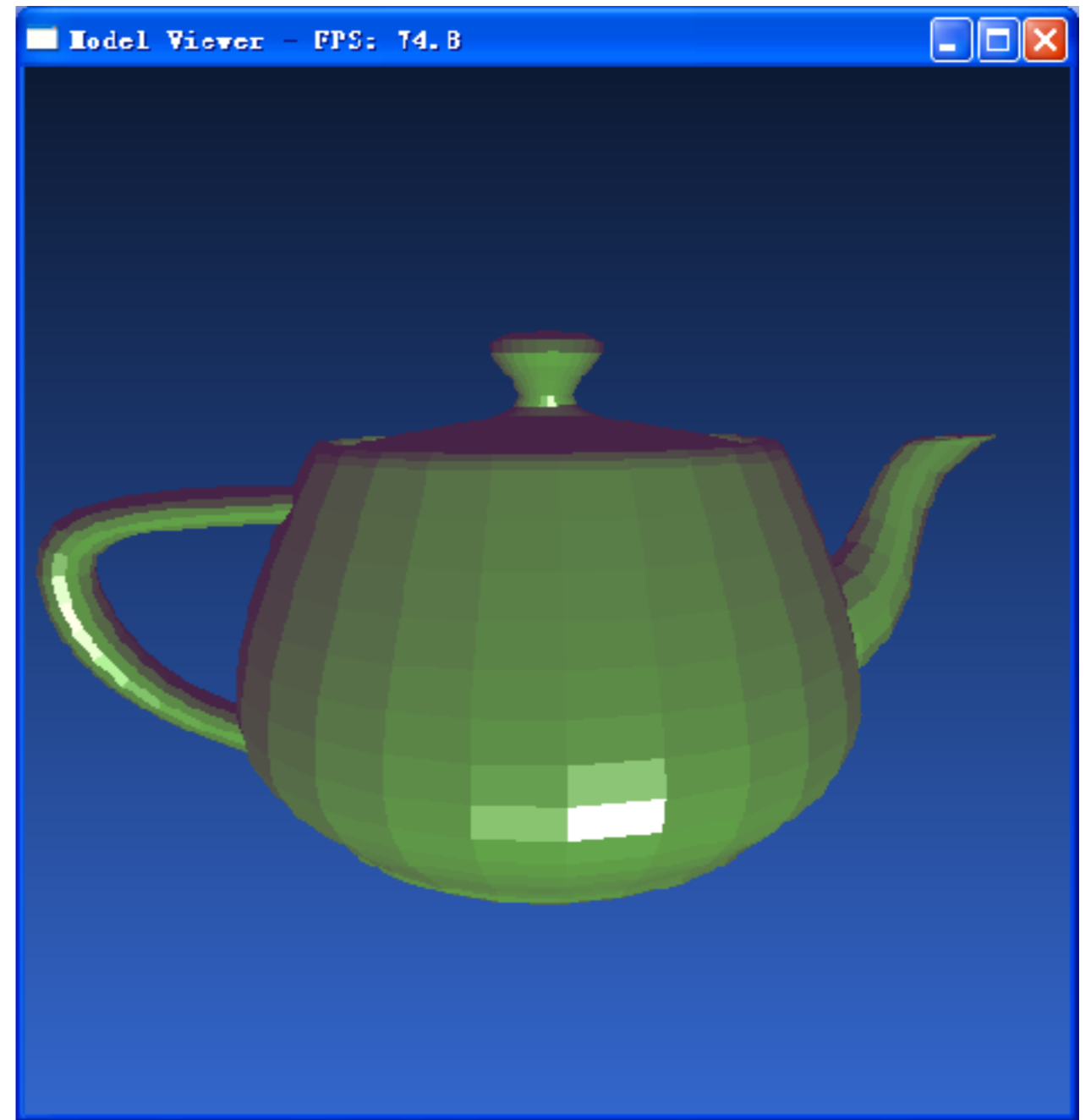
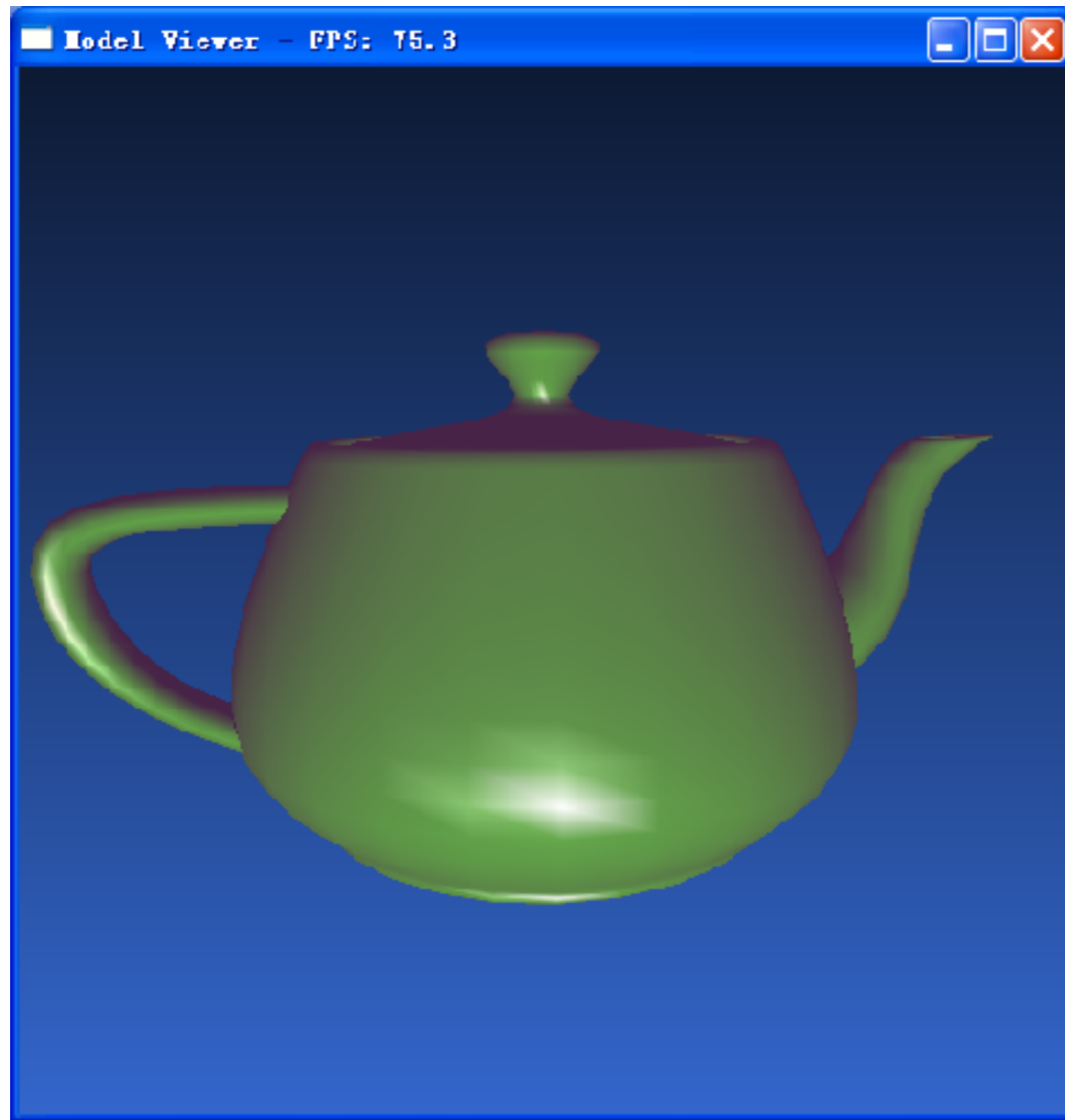
Gouraud  
Shading



Phong  
Shading

# Demo – OpenGL Shade Model

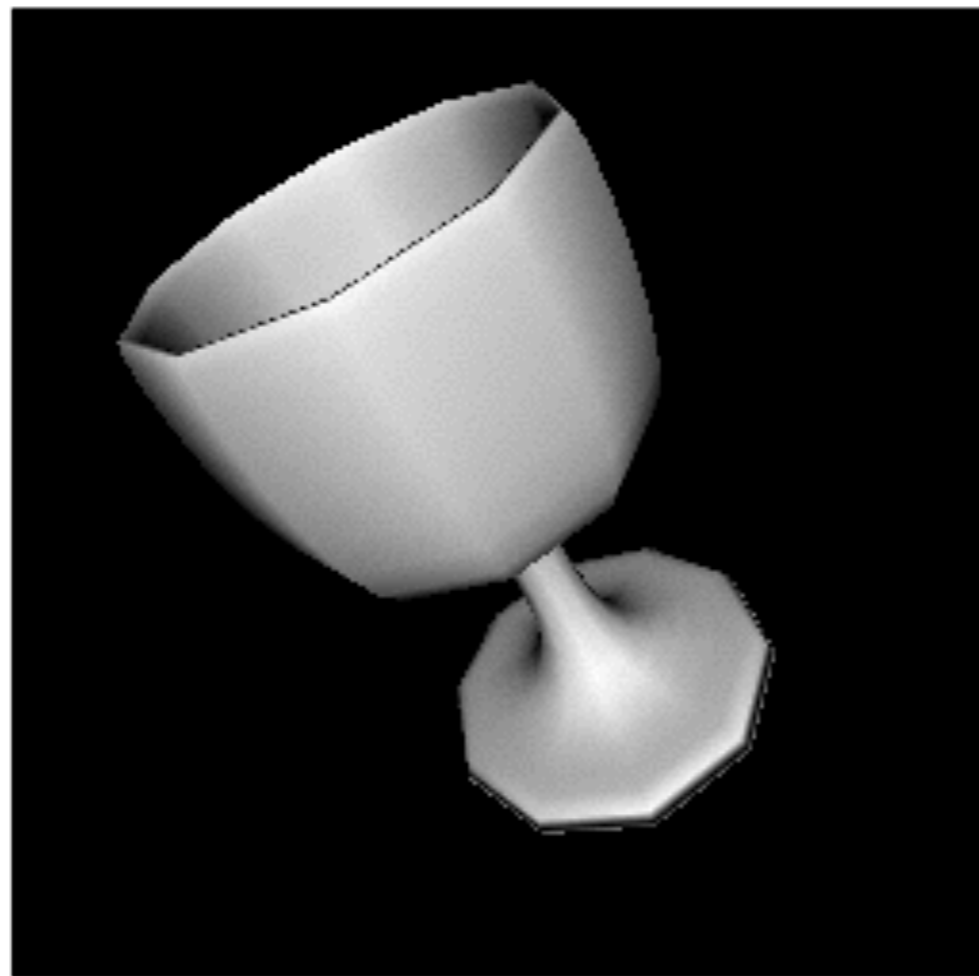
---



# Problems with Interpolated Shading

---

- Polygon silhouette :
  - The silhouette edge of the mesh is always a polygon
- Solution :
  - finer subdivision for the entire surface
  - finer subdivision only along silhouette (view dependent)

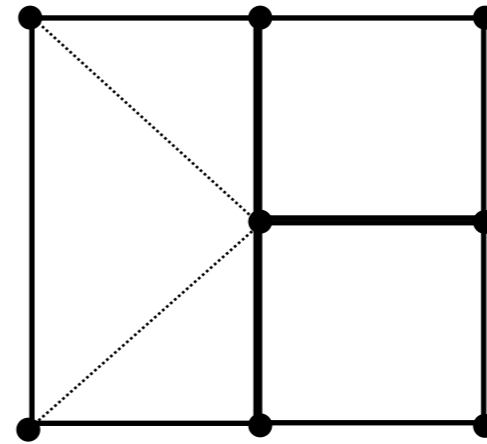


# Problems with Interpolated Shading

---

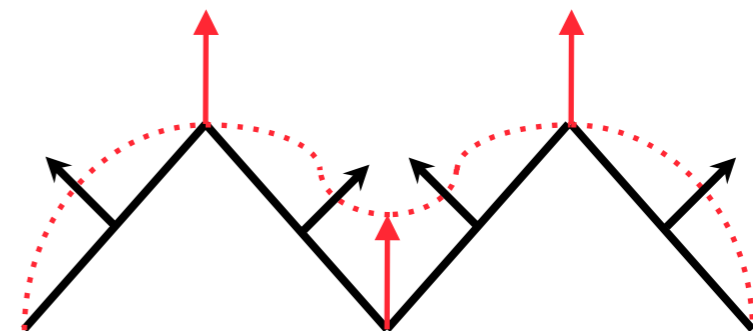
- Problems at shared vertices

- avoid such cases



- Unrepresentative vertex normals

- subdividing the polygons before calculating normals

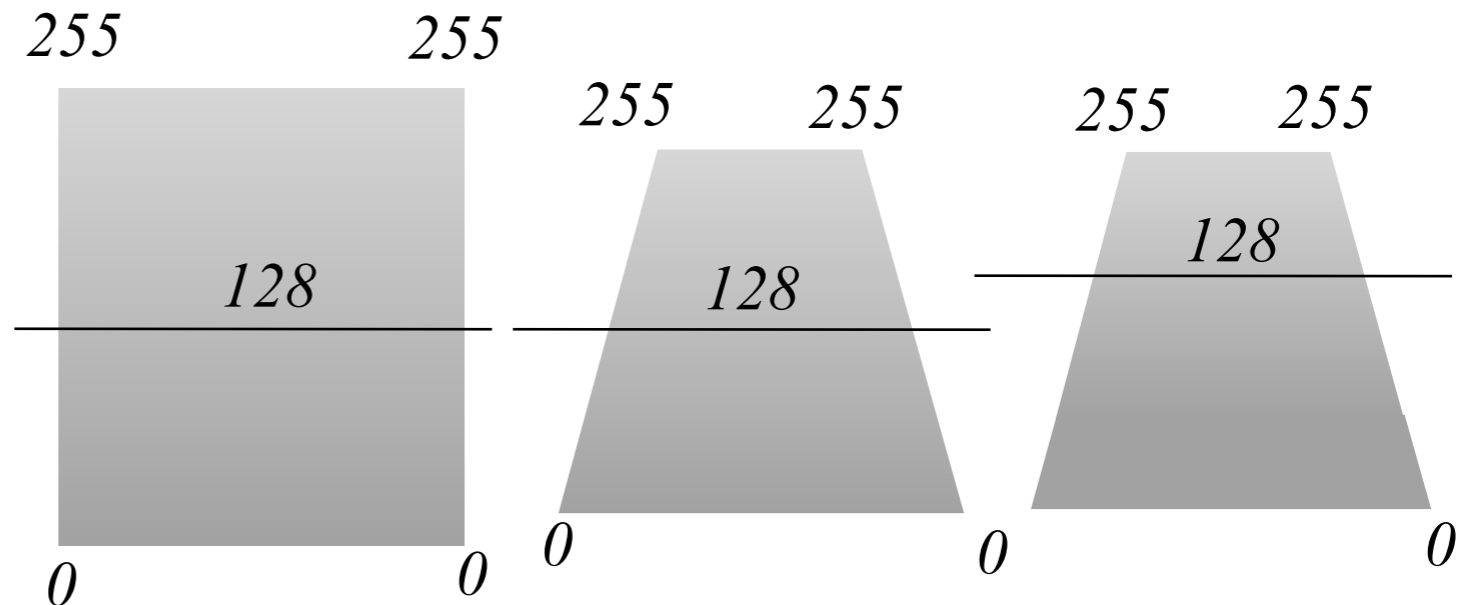


# Problems with Interpolated Shading

- Perspective distortion

Solution :

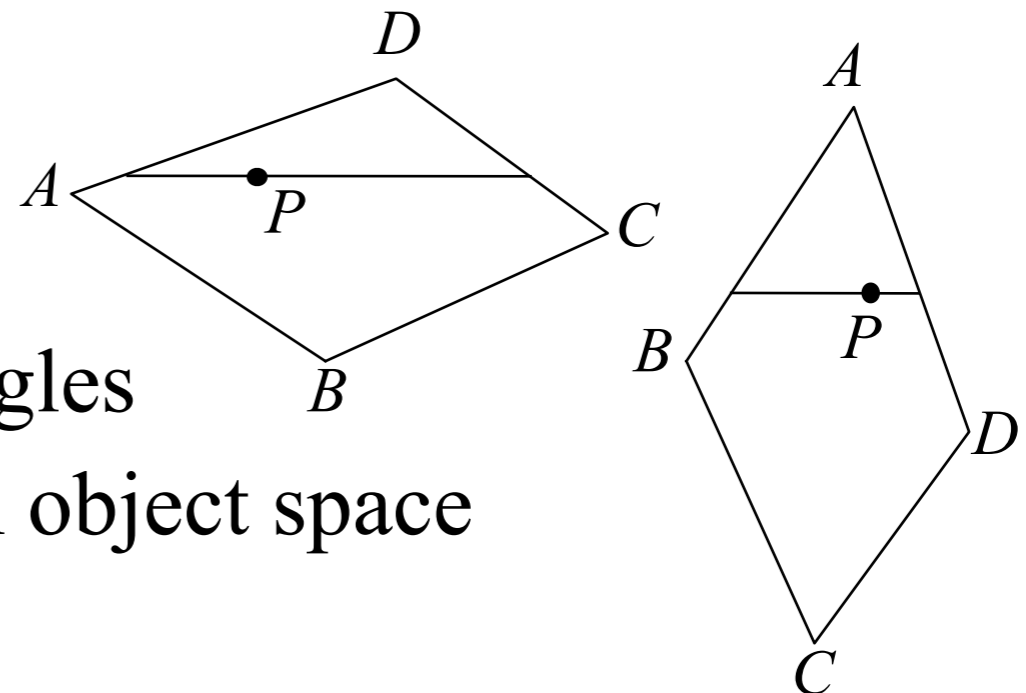
- smaller polygons by finer subdivision
- interpolate in world space



- Orientation dependence

Solution :

- decomposing polygons into triangles
- rotation invariant interpolation in object space



# Lighting in OpenGL

---

Steps of specifying lighting in OpenGL

1. Define **normal vectors** for each vertex of every object.
2. Create, position, and enable one or more **light sources**.
3. Select a **lighting model**.
4. Define **material properties** for the objects in the scene.

# Specifying Materials in OpenGL

---

## General form:

```
glMaterialf(face, parameter,value);  
glMaterialfv(face, parameter,*array);
```

face **is** GL\_FRONT, GL\_BACK, GL\_FRONT\_AND\_BACK

parameter **is:**

**GL\_AMBIENT** four values that specify the ambient RGBA reflectance of the material. (0.2,0.2,0.2,1.0)

**GL\_DIFFUSE** four values that specify the diffuse RGBA reflectance of the material. (0.8,0.8,0.8,1.0)

**GL\_SPECULAR** four values that specify the ambient RGBA reflectance of the material. (0.0,0.0,0.0,1.0)

**GL\_SHININESS** specifies the specular reflectance exponent of the material. 0.0

# Demo - Materials

