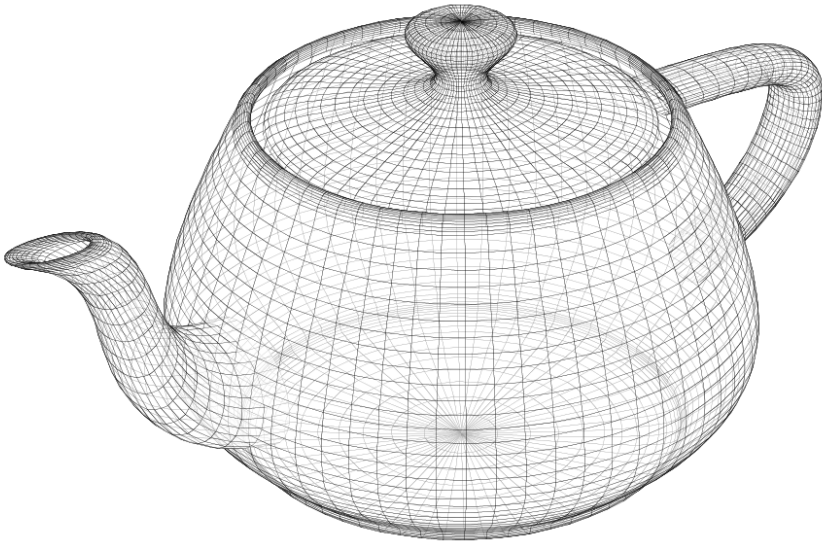


# Shading

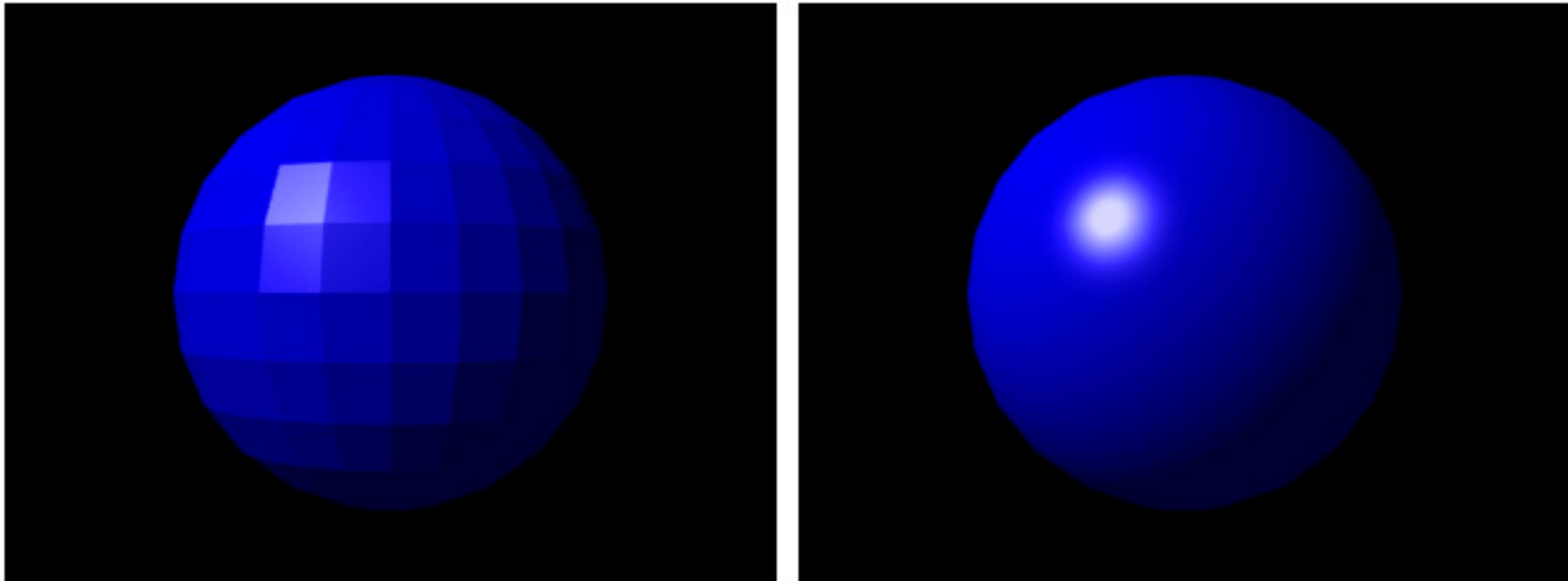
## Phong Reflection Model



Interactive Computer Graphics  
Eric Shaffer

# Shading

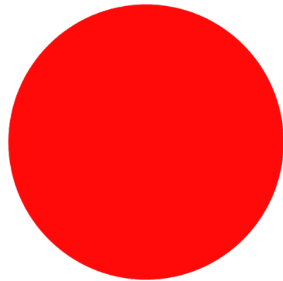
**Shading** refers to the process of determining the color for a pixel (or vertex...or polygon) during the rendering process



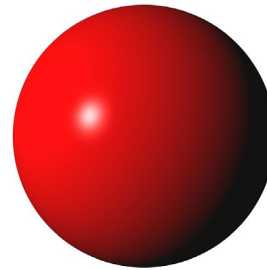
What is the difference between the two images?

# Why we need shading

Shading is one of the key elements of 3D photorealistic rendering



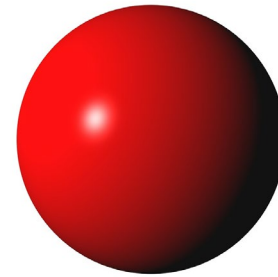
Flat shading  
...not realistic



Shading with Phong reflection model  
...more realistic

# Shading

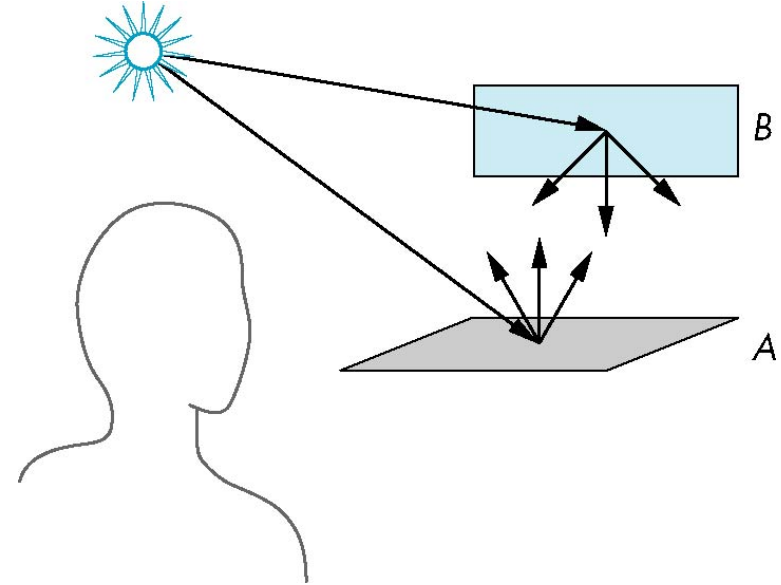
- Why does the image of a real sphere look like



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

# Scattering

- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A and so on



# Super-simple Model of Light-Material Interaction

- Light that strikes an object is
  - partially absorbed and
  - partially scattered (reflected)
- The amount reflected determines
  - color of the object
  - brightness of the object
  - A surface appears red under white light because the red component of the light is reflected
- How the reflected light is scattered depends on
  - the smoothness and orientation of the surface

Mathematically we can model this as  
component-wise multiplication of RGB values

Light  $i = (r_i, g_i, b_i)$

Material  $k = (r_k, g_k, b_k)$

Light reflecting off surface  $= (r_i r_k, g_i g_k, b_i b_k)$

# Shading

We will create a *simple* mathematical model for shading

It will be function that takes in:

- Light sources
- Material properties
- Location of viewer
- Surface orientation

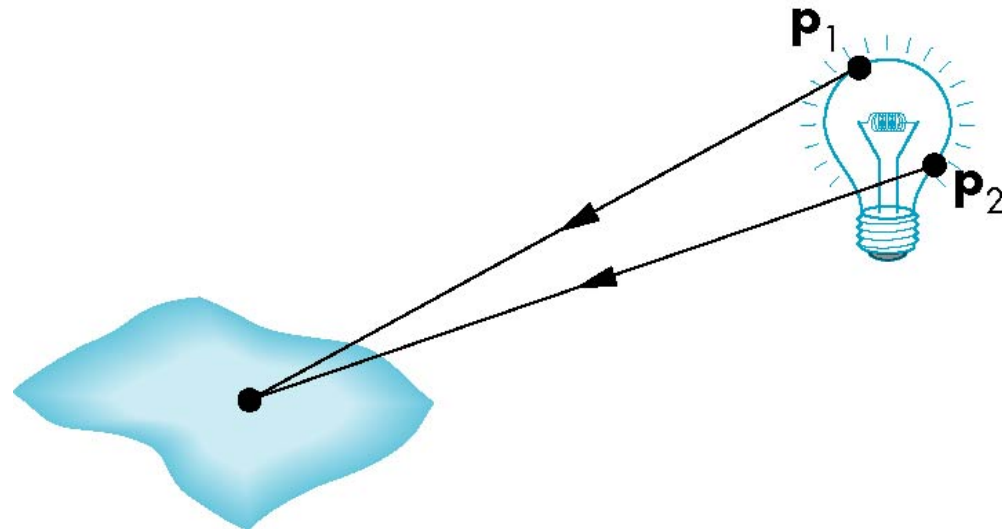
Well...there's a lot of symbols but the ideas and math are simple

And returns a color....

# Light Sources

General light sources are complex to model

Would need to integrate light coming from all points on the source



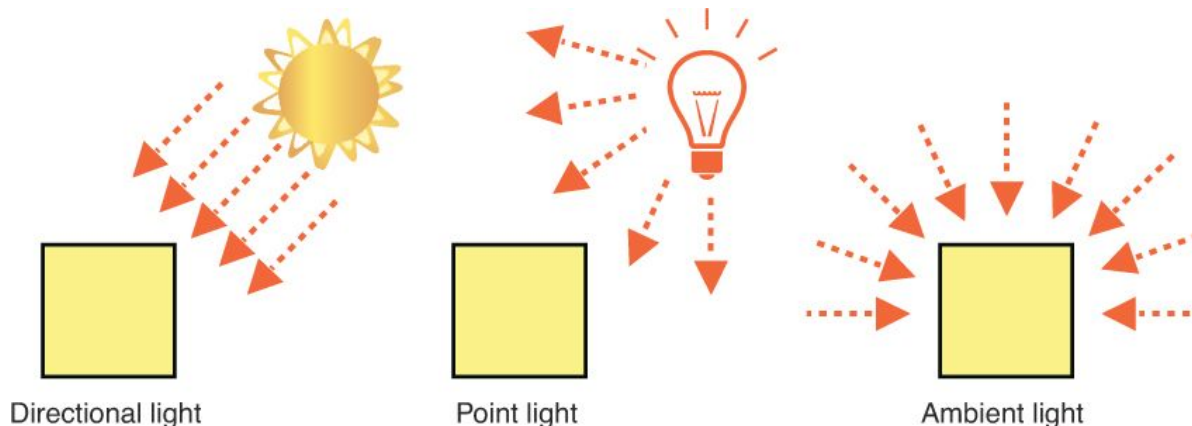


# Simple Light Source Models

- Point source
  - Model with position and color
- Directional source
  - Distant source = infinite distance away (parallel)
- Ambient light
  - Same amount of light everywhere in scene
  - Models indirect light from reflecting surfaces

Creating a point light in JS code.....

```
var diffuse0 = glmatrix.vec4.fromValues(1.0, 0.0, 0.0, 1.0);  
var specular0 = glmatrix.vec4.fromValues (1.0, 1.0, 1.0, 1.0);  
var light0_pos = glmatrix.vec4.fromValues (1.0, 2.0, 3.0, 1.0);
```



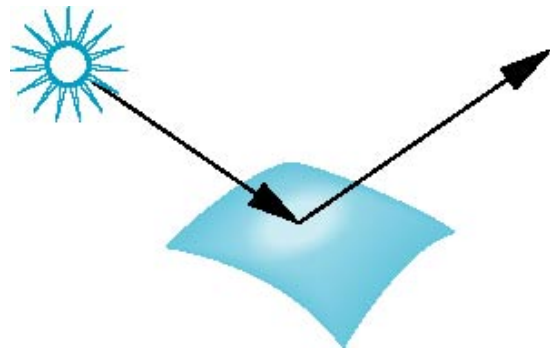
# Moving Light Sources

- Point light sources are geometric objects
- Positions and directions can be affected a model-view matrix
- **Should you apply the view transformation to a light?**
  - **Yes (usually)**
  - **Exception is when the light should move with the viewpoint**

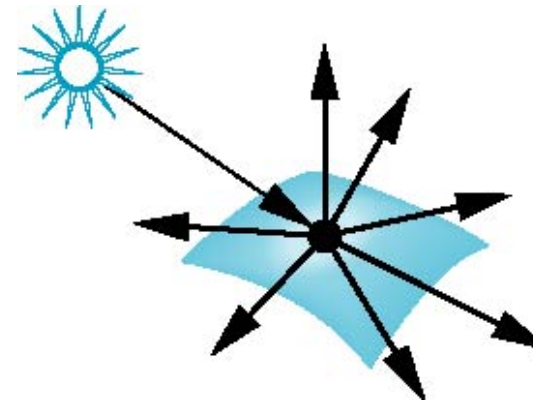


# Surface Types

- Consider light traveling along a specific ray
- The smoother a surface, the more reflected light is concentrated in a single direction
  - Perfect mirror reflects perfectly in a single direction
- A very rough surface scatters light in all directions



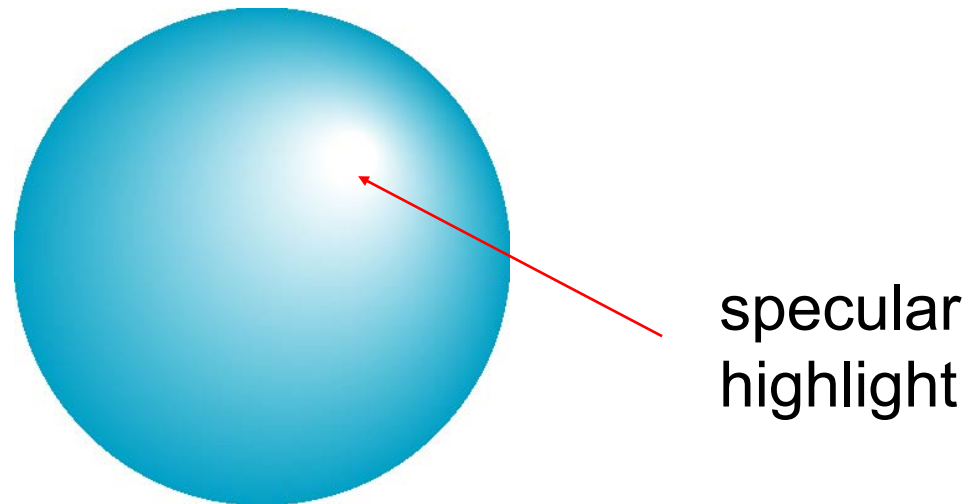
smooth surface



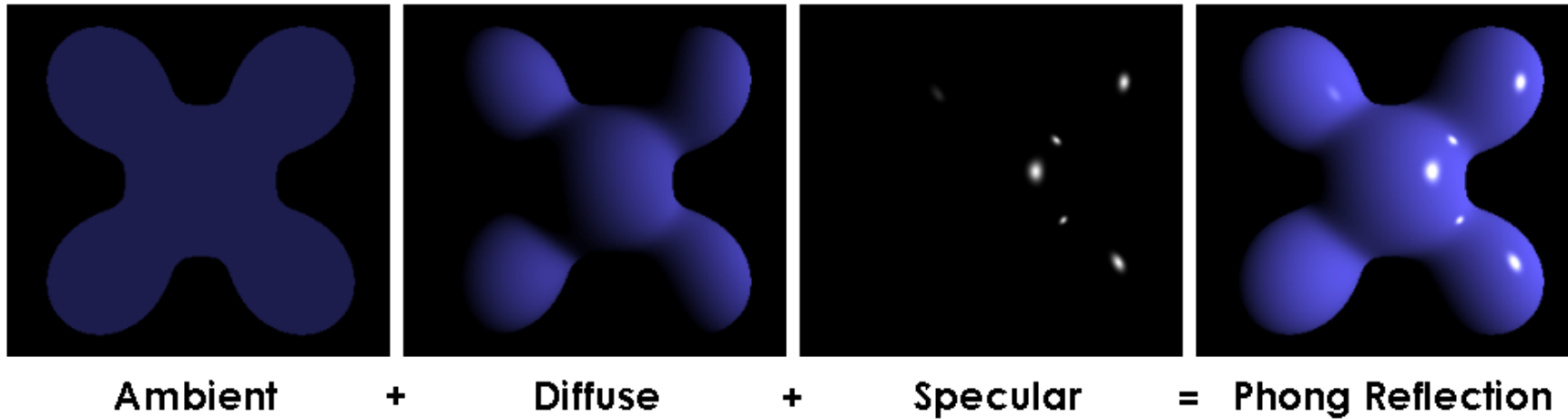
rough surface

# Specular and Diffuse Surfaces

- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- Smooth surfaces show specular highlights
  - incoming light is reflected in directions concentrated close to the direction of a perfect reflection



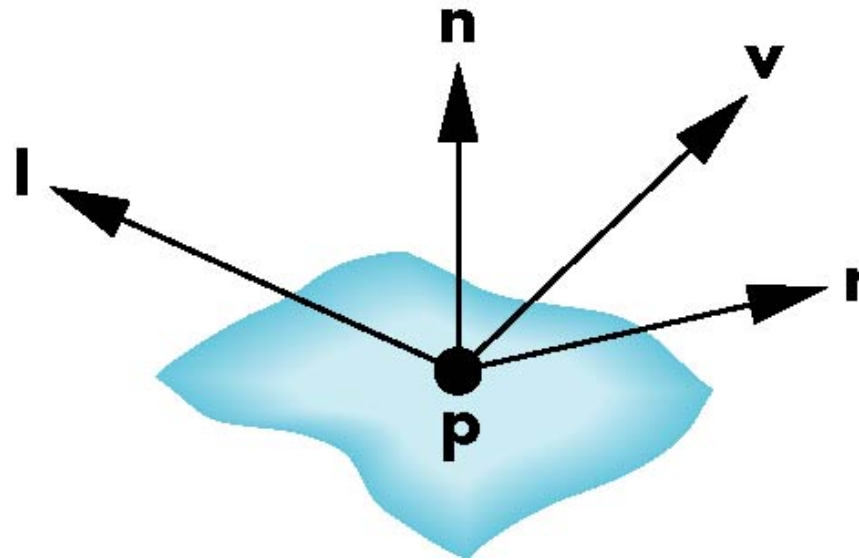
# Phong Reflection Model



$$I_p = k_a i_a + \sum_{m \in \text{lights}} (k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^\alpha i_{m,s})$$

# The Phong Reflection Model

- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To source
  - To viewer
  - Normal
  - Perfect reflector

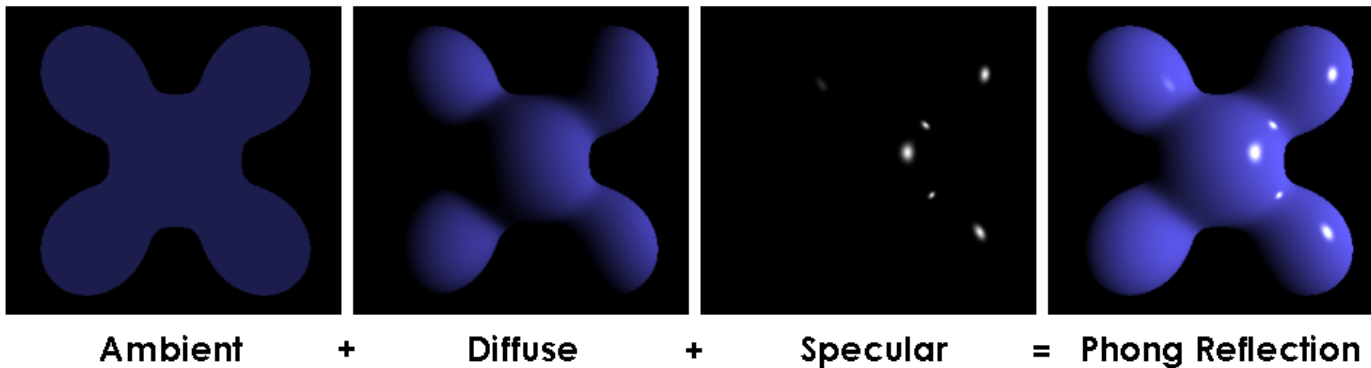


# Ambient Light

- Result of multiple interactions between light sources and surfaces
- Amount and color depend on the color of the light(s) and the material properties
- Add  $k_a I_a$  to diffuse and specular terms

reflection ← intensity of ambient light

Remember that  $k_i$  multiplications are component-wise multiplications of rgb values  
 $(k_r, k_g, k_b)(i_r, i_g, i_b) = (k_r i_r, k_g i_g, k_b i_b)$

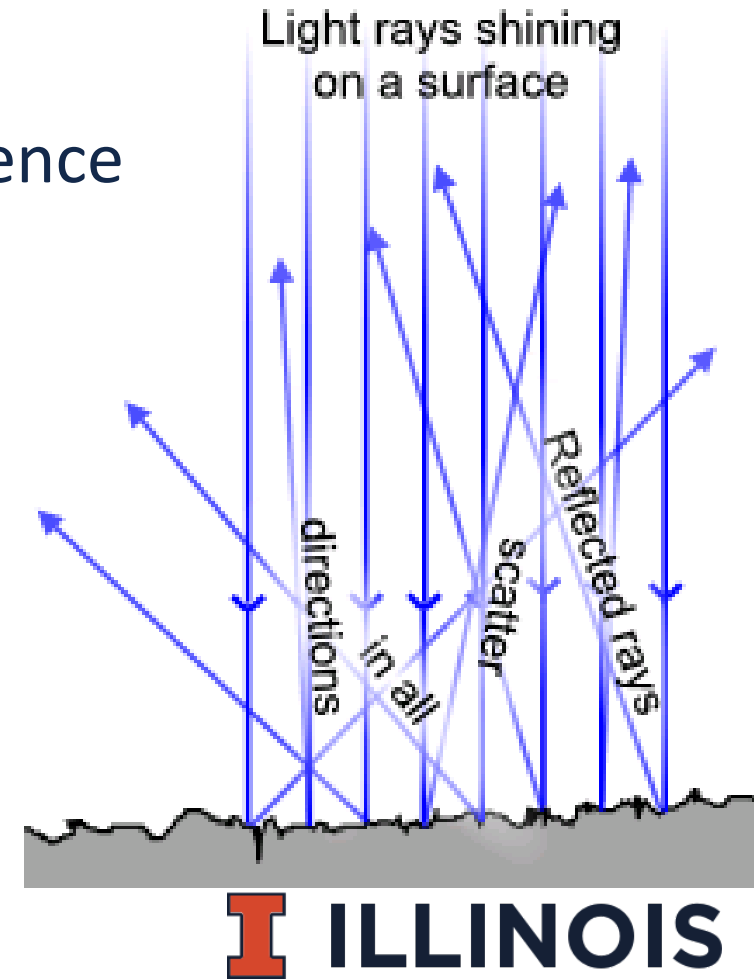


# Modeling a Lambertian Surface – Diffuse Reflection

- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is affected by the angle of incidence
  - reflected light proportional to **cosine of angle between  $l$  and  $n$**
  - if vectors normalized

$$\cos(\theta) = n \cdot l$$

- Amount of reflected light also affected by  $k_d$  and  $i_d$ 
  - Each is an rgb value with each channel in  $[0,1]$

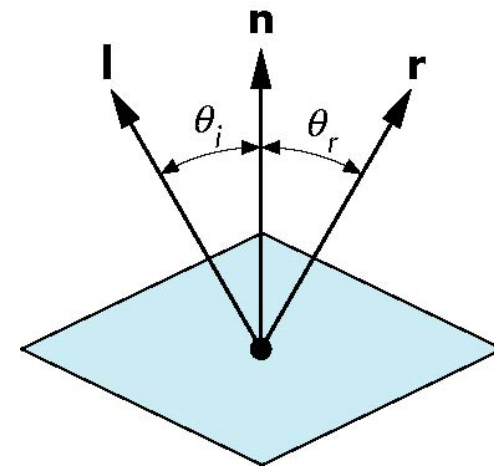




# Modeling a Ideal Reflector – Specular Reflection

- Incoming light ray is reflected in a single
- Normal is determined by local orientation
- Given the direction of incoming light...we can find  $r$ 
  - $l$  and  $n$  are unit vectors
  - Angle of incidence = angle of reflection
  - The three vectors will be coplanar
- What is another name for an ideal reflector?
- $r$  is computed as

$$r = 2(l \cdot n)n - l$$



# Deriving the Reflection Vector

$R_i$  is the incoming light vector

$R_1$  is the reversed light vector used in shading

The reflection vector  $R_r$  can be seen as a vector sum:

$$R_r = N(R_1 \cdot N) + a$$

Similarly

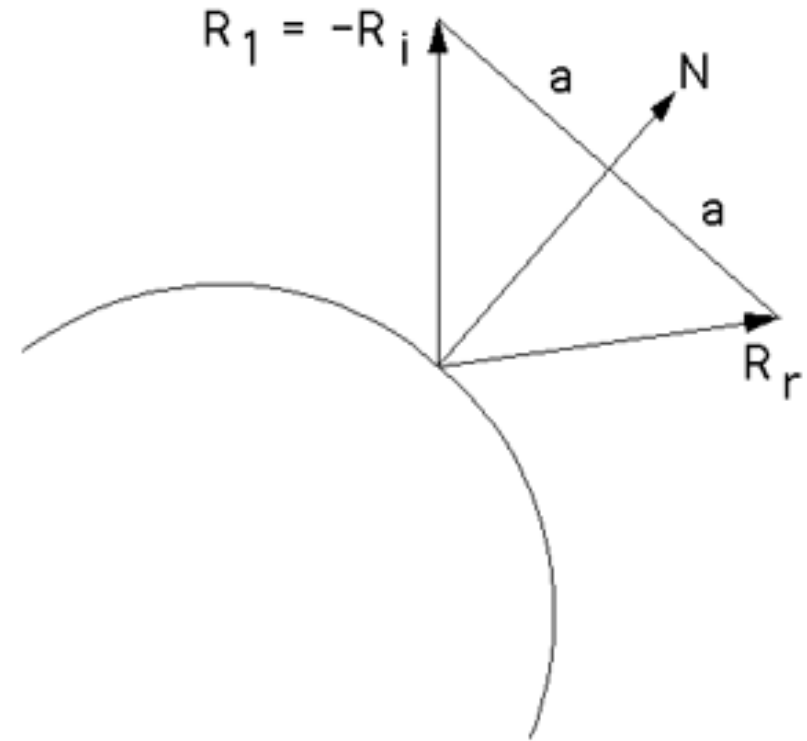
$$R_1 + a = N(R_1 \cdot N)$$

so

$$a = N(R_1 \cdot N) - R_1$$

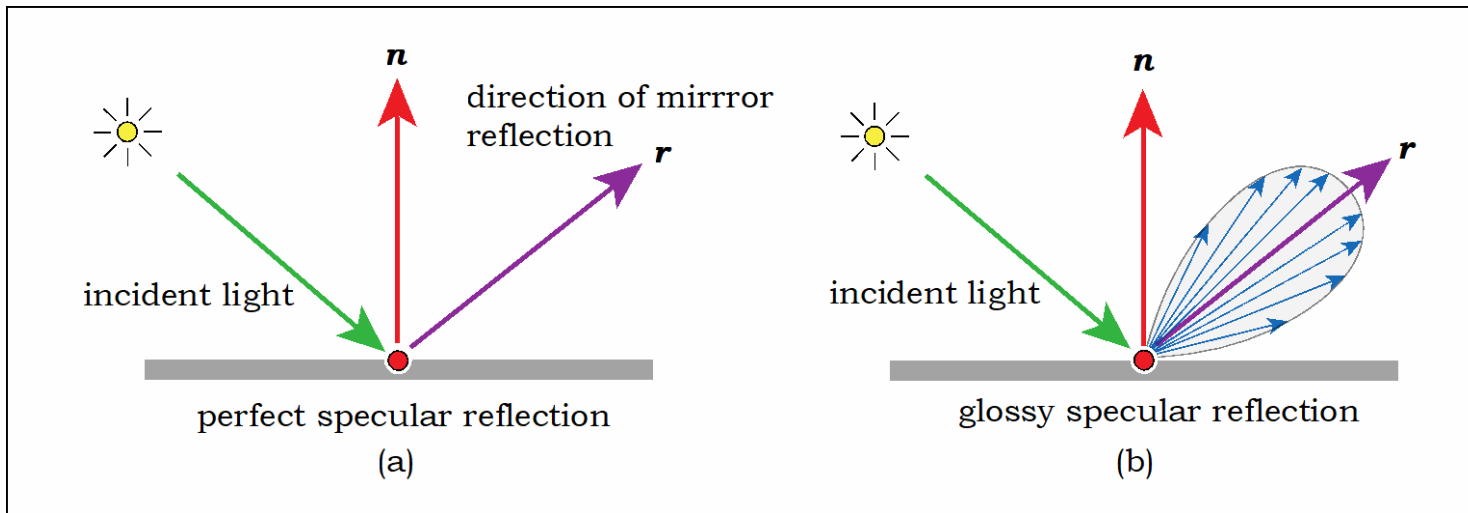
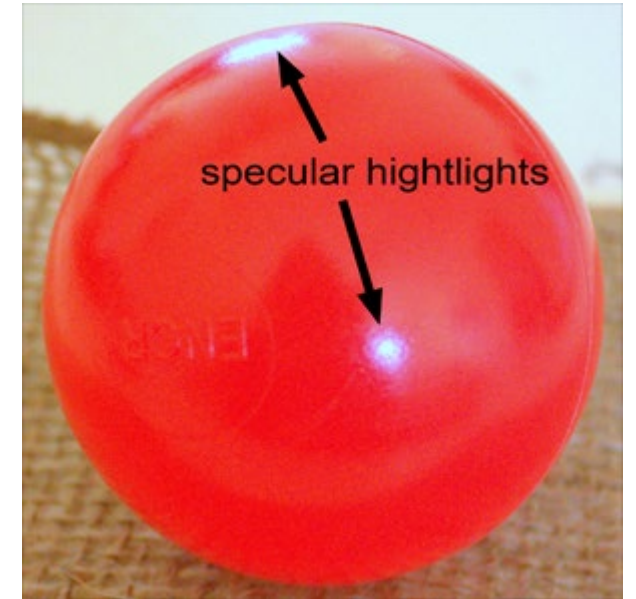
And substituting that expression for back into our first equation we have

$$R_r = 2N(R_1 \cdot N) - R_1$$



# Specular Reflection

- Perfect specular reflection
  - Light is reflected in the single direction  $r$
  - ...the mirror reflection direction
- Glossy specular reflection
  - Scattering clustered around mirror reflection direction



# Specular Reflection

- Reflectance determined by
  - Alignment of view vector with mirror reflection vector
  - Shininess coefficient
- High coefficient means smoother look
  - Maybe 100 for metal
  - Maybe 10 for plastic

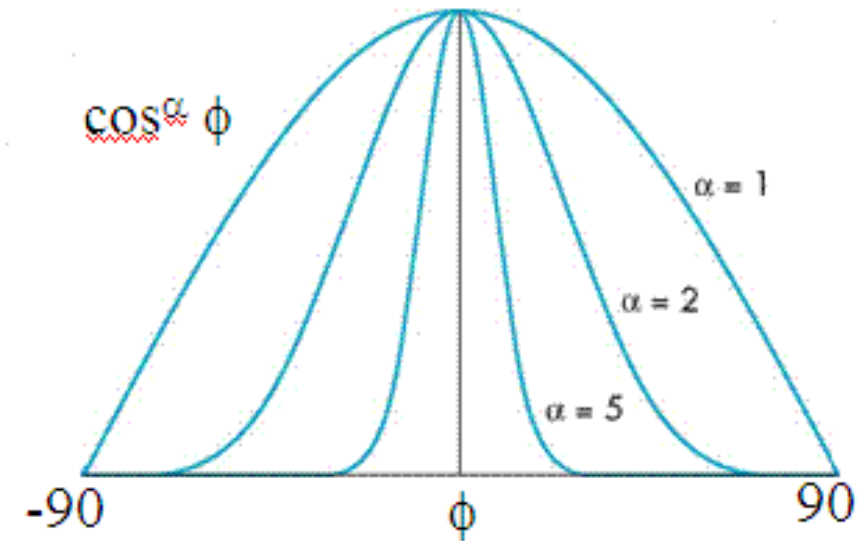
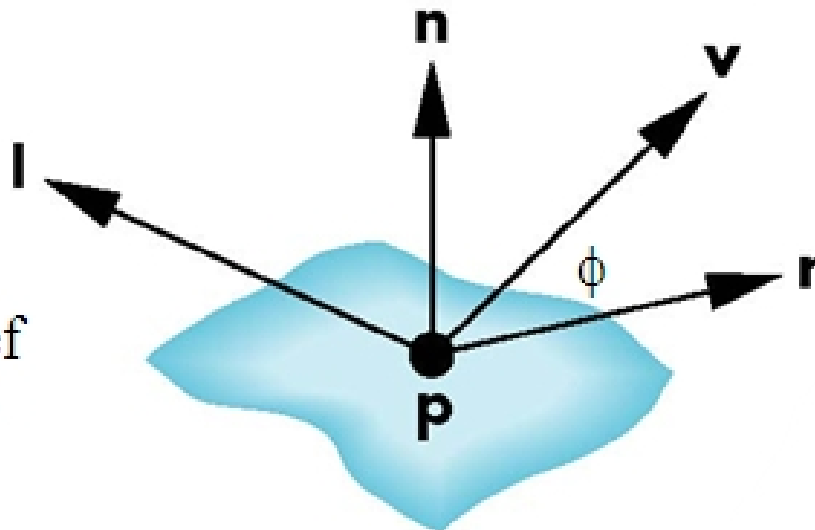
$$I_r \sim k_s I \cos^\alpha \phi$$

reflected intensity

incoming intensity

absorption coef

shininess coef



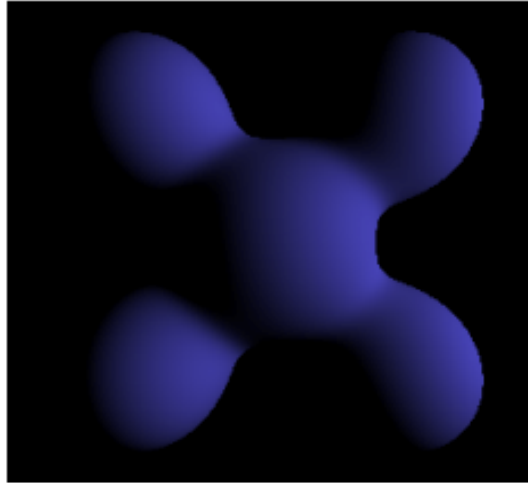
# Phong Reflectance Model

Summing over all the light sources, the Phong model can be written as



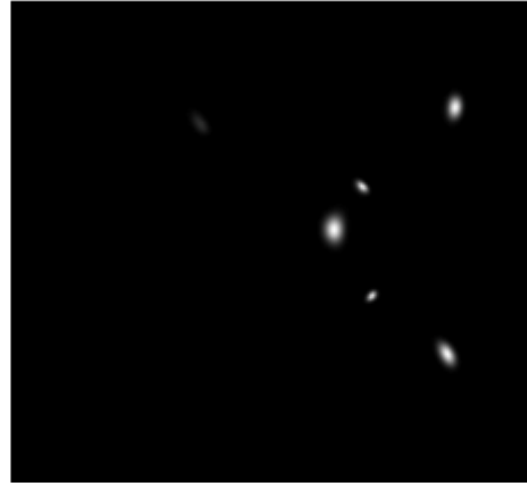
Ambient

+



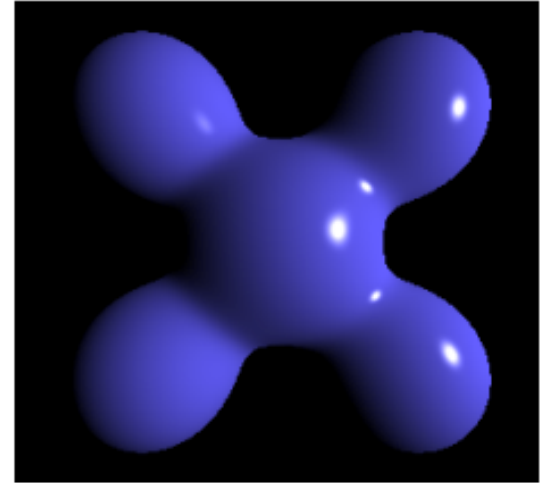
Diffuse

+



Specular

=



Phong Reflection

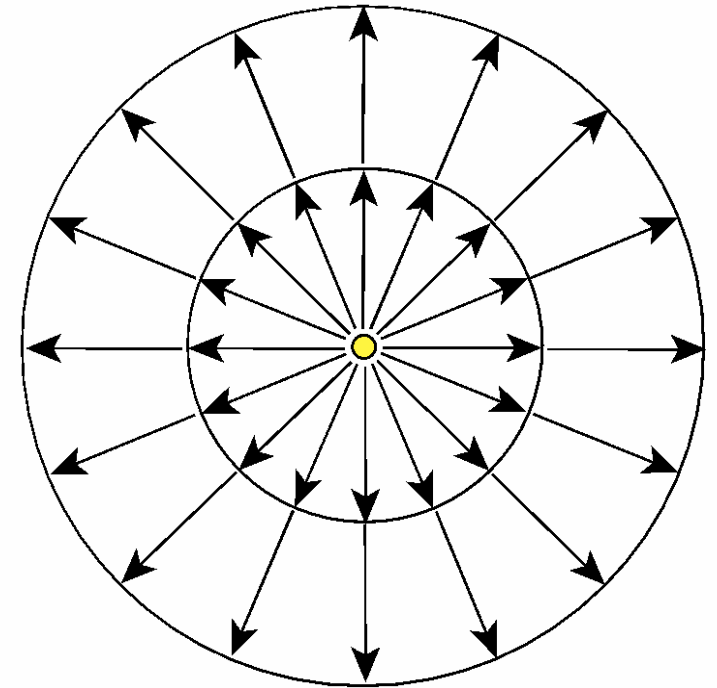
$$I_p = k_a i_a + \sum_{m \in \text{lights}} (k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^\alpha i_{m,s})$$

# Distance Terms

- The light from a point source that reaches a surface is ***attenuated***
  - Intensity falls off with the square of the distance
- We can add a factor of to the diffuse and specular terms

$$\frac{1}{ad^2+bd+c}$$

- **d** is the distance from the light to surface
- **a,b,c** are constants you choose to get different effects



# Blinn-Phong Reflectance Model

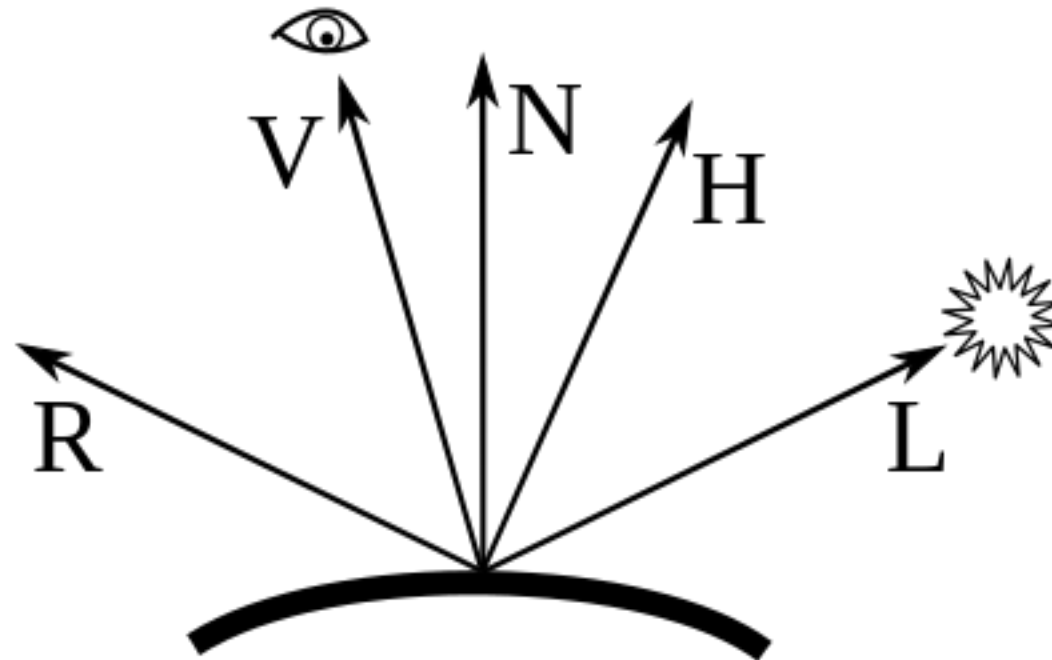
- Jim Blinn suggested an approximating changing specular term
- Replace  $(\mathbf{V} \cdot \mathbf{R})^a$  by  $(\mathbf{N} \cdot \mathbf{H})^b$  where
  - “Halfway vector”
- More efficient in terms of the operations used
- Closer to physically correct lighting
- Pick exponent  $b$  to match what you want
  - Using higher  $b > a$  will make output similar to Phong with  $a$

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

# The Halfway Vector

**H** is normalized vector halfway between **L** and **V**

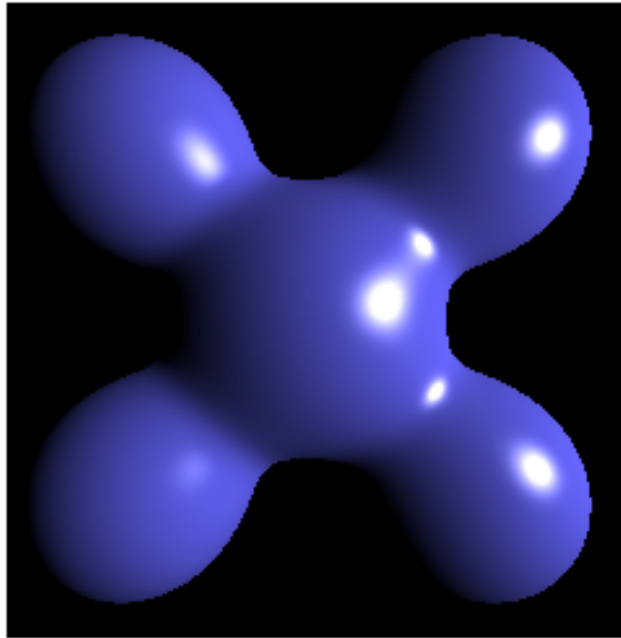
$$H = \frac{L + V}{\|L + V\|}$$



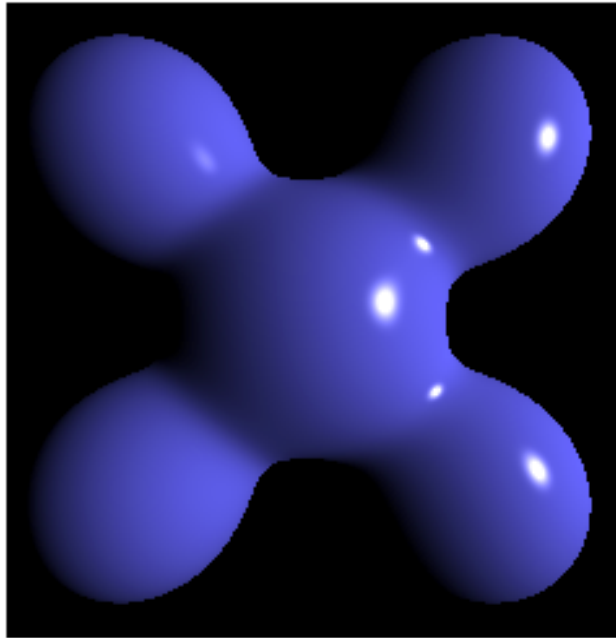
$$r = 2(l \cdot n)n - l$$



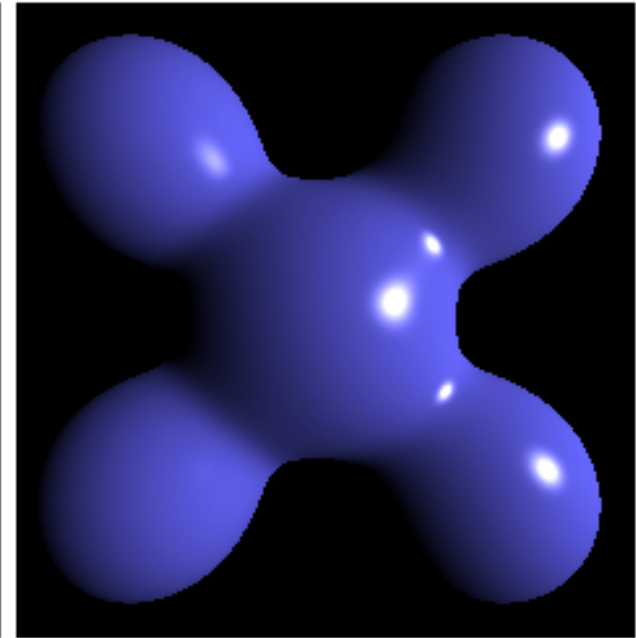
# Phong versus Blinn-Phong



**Blinn-Phong**



**Phong**



**Blinn-Phong**  
(higher exponent)

# Smooth Shading on a Sphere Mesh

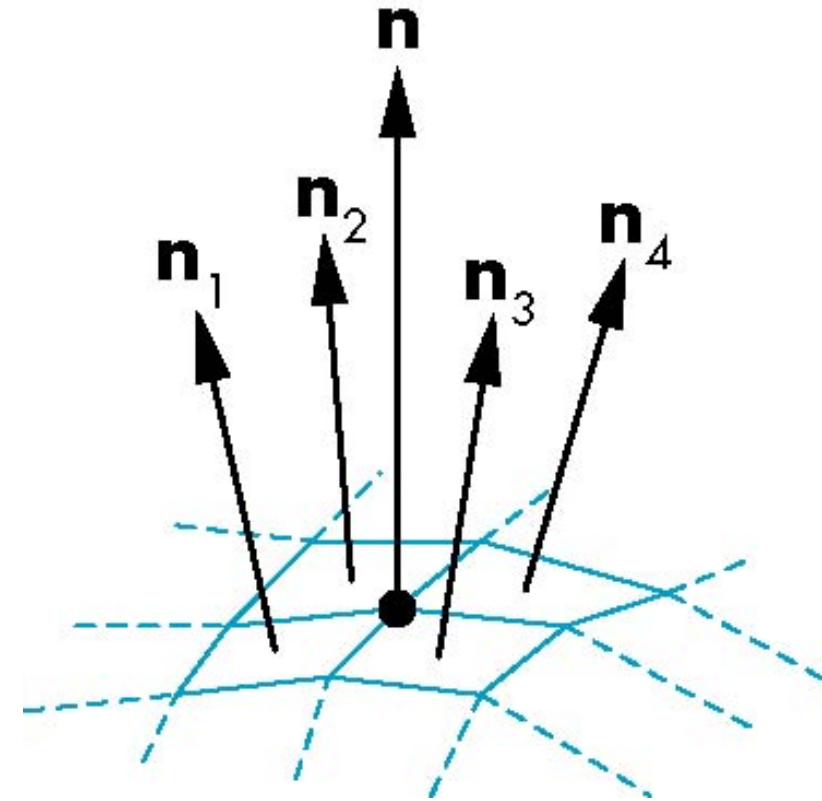
- I know...silhouette edge is bumpy for a sphere...
  - How do you make it less bumpy?
  - Why does the middle look smooth?
- We set a new normal at each vertex
- Per-vertex normal easy for a sphere model
  - If centered at origin  $\mathbf{n} = \mathbf{p}$

What is the normal at the vertex (1,0,0)?



# Mesh Shading

- The previous example is not general
  - We knew the normal at each vertex analytically
- For polygonal meshes, Henri Gouraud proposed we
  - use the average of the normals around a vertex



$$\mathbf{n} = (\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4) / |\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|$$

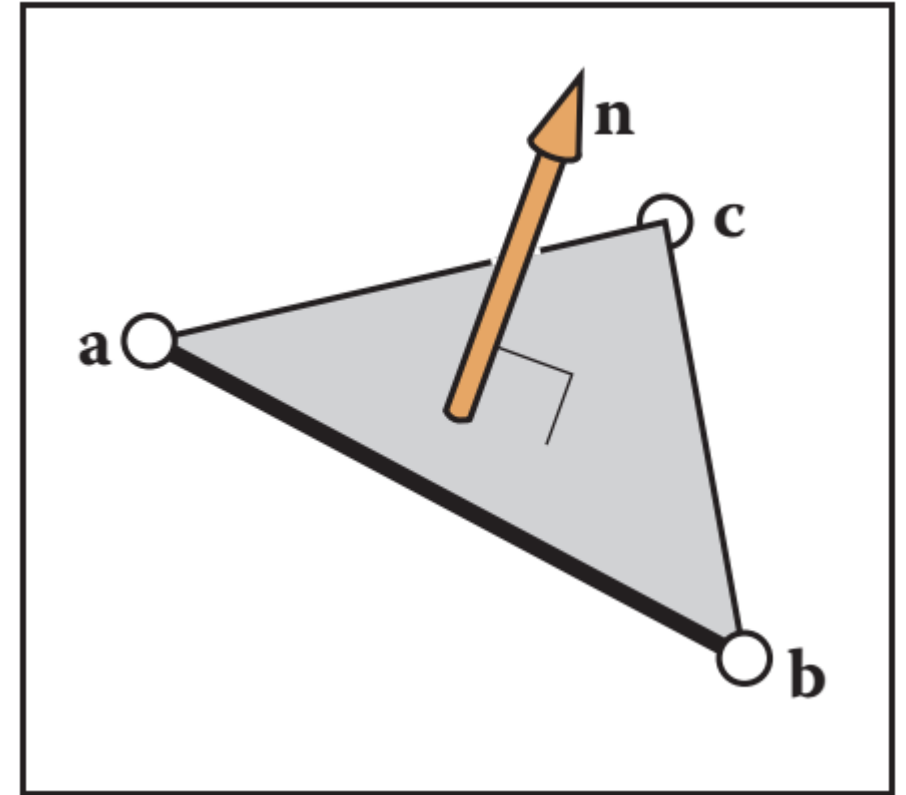
# Computing a Normal for a Triangle

$$\vec{n} = (b - a) \times (c - a)$$

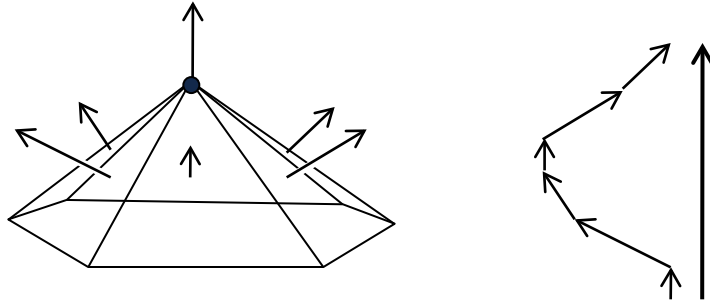
To make unit length:

$$\vec{n}_{unit} = \frac{\vec{n}}{\|\vec{n}\|}$$

You can use the glmatrix library to compute normals for vertices  
Remember to make them unit-length for shading



# Computing per-Vertex Normals



To compute per-vertex normal on a mesh with  $M$  vertices

- Initialize an array `NArray` containing  $M$  normals
  - Each normal starts as  $[0,0,0]$
- Iterate over all triangles  $T=[v1,v2,v3]$  with vertices in CCW order
  - Compute normal  $N$  for  $T$  using  $N = (v2-v1) \times (v3-v1)$ 
    - `NArray[v1]+=(NArray[v1]+N)`
    - `NArray[v2]+=(NArray[v2]+N)`
    - `NArray[v3]+=(NArray[v3]+N)`
- Normalize each normal in `NArray` to unit length

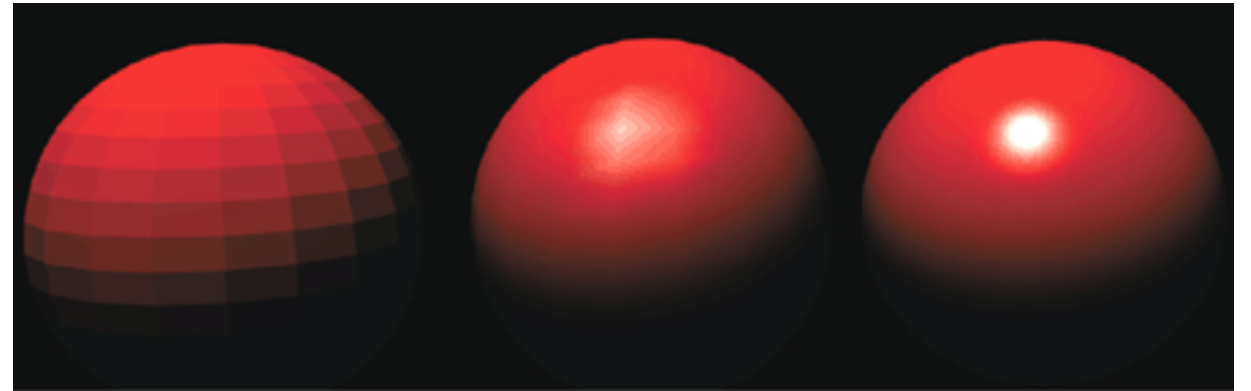
# Gouraud and Phong Shading

## Gouraud Shading

- Find average normal at each vertex
- Apply Blinn-Phong model at each vertex
- Interpolate vertex shades across each polygon

## Phong Shading

- Find average normal at each vertex
- Interpolate vertex normals across edges
- Interpolate edge normals across polygon
- Apply Blinn-Phong model at each fragment



Flat

Gouraud

Phong

# Bui Tuong Phong

Graphics and  
Image Processing

W. Newman  
Editor

## Illumination for Computer Generated Pictures

Bui Tuong Phong  
University of Utah

- December 14, 1942 – July 1975
- Born in Hanoi
- Earned his PhD in 2 years at the University of Utah (1973)
  - Worked with Professor Ivan Sutherland
  - Dissertation work was the Phong reflectance model
  - Also produced model and realistic image of a VW bug

Fig. 9. Improved shading, applied to the example of Figure 2.

