




## Illumination and Shading

- ◆ In order to produce realistic images, we must simulate the appearance of surfaces under various lighting conditions.
- ◆ *Illumination models*: given the illumination incident at a point on a surface, what is reflected?
- ◆ *Shading algorithms*: determine when and how to apply the illumination model, in order to provide a color for every visible surface point.

1

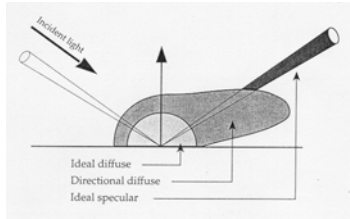
## Light Source Models

- ◆ General (real life) light sources often have complex geometry and emission characteristics.
- ◆ In computer graphics, the following simplified models are commonly used:
  - ◆ Directional light source: all light rays are parallel to a particular direction. 
  - ◆ Point light source: all light rays originate at a particular point in the scene, in all directions. 
  - ◆ Spotlight: like a point like source, but ray directions are limited to a cone. 

2

## The BRDF

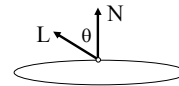
- ◆ Bidirectional Reflection Distribution Function - describes the ratio of light intensity leaving a point in some outgoing direction to the differential irradiance from some incoming direction:



3

## Diffuse Reflection

- ◆ Diffuse (Lambertian) surfaces appear equally bright from all directions.



- ◆ The intensity of light reflected by a point  $x$  on a diffuse reflector is

$$I_r = I_p k_d \cos \theta = I_p k_d (N \cdot L)$$

4

## The Ambient Term

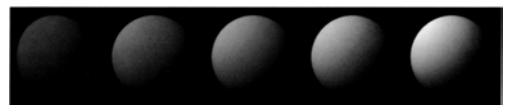
- ◆ In the real world, objects receive light both from light sources and from other surfaces in the scene.
- ◆ This indirect illumination is very coarsely modeled by adding an ambient term into the shading model:

$$I_r = I_a k_a + I_p k_d (N \cdot L)$$

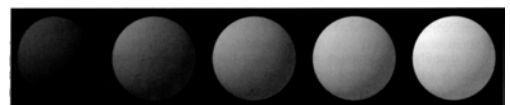
5

## Examples: Lambertian model

- ◆ Diffuse term only:



- ◆ Diffuse + Ambient:



6

## Light-source attenuation

- ◆ The irradiance due to physical light source falls off proportionally to the square of the distance.
- ◆ This is accounted for by introducing a light-source attenuation factor into the shading equation:

$$I_r = I_a k_a + f_{att} I_p k_d (N \cdot L)$$

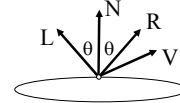
- ◆ Typically,

$$f_{att} = \min\left(\frac{1}{a + bd_L + cd_L^2}, 1\right)$$

7

## Specular Reflection

- ◆ Phong Bui-Tuong introduced a term for simulating specular highlights on non-ideal (glossy) reflectors:

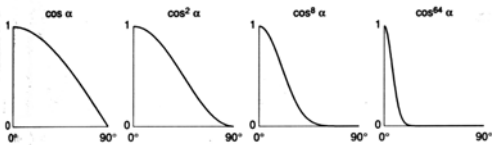


$$I_r = I_a k_a + f_{att} I_p [k_d (N \cdot L) + k_s (R \cdot V)^n]$$

8

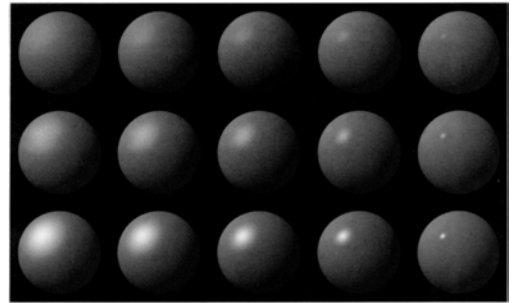
## The Specular Exponent

- ◆ The Phong exponent  $n$  determines how concentrated the specular peak is:



9

## Example: Phong model

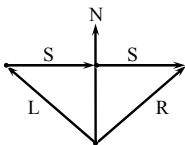


10

## The Reflected Vector

- ◆ The Phong model makes use of the reflected vector  $R$ , which is computed as follows:

$$R = 2N(N \cdot L) - L$$



11

## Multiple light sources

- ◆ The shading model easily extends to the case of multiple light sources:

$$I_r = I_a k_a + \sum_{i=1}^{\ell} f_{att_i} I_{p_i} [k_d (N \cdot L_i) + k_s (R_i \cdot V)^n]$$

12

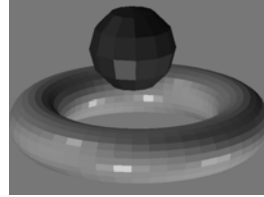
## Polygon Shading

- ◆ Flat (constant) shading
  - ◆ Evaluate the shading model once per polygon, use resulting color for all of its pixels.
- ◆ Gouraud shading
  - ◆ Evaluate the shading model at each vertex, and linearly interpolate resulting values inside the polygon
- ◆ Phong shading
  - ◆ Evaluate the normal at each vertex, and linearly interpolate it inside the polygon. Having the interpolated normal at each point inside the polygon, we can use it to calculate the shading model in each point.

13

## Gouraud shading

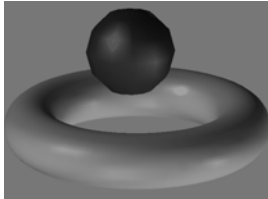
- ◆ Smooth surfaces are commonly represented as a collection of polygonal facets for the purposes of interactive display.
- ◆ If each facet is shaded individually, it is easy to see the shading discontinuities, which result in faceted appearance



14

## Gouraud shading

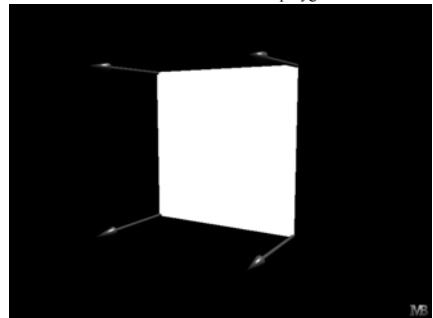
- ◆ To eliminate faceted appearance we can use Gouraud shading (linearly interpolated shading). For the resulting shading to be continuous we need each vertex in the polygon mesh to have the same normal for all faces incident on it.



15

## Gouraud shading

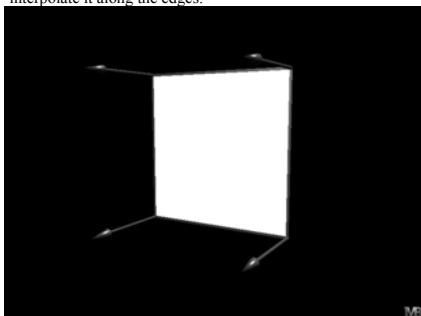
Calculate normals on each vertex of the polygon



16

## Gouraud shading

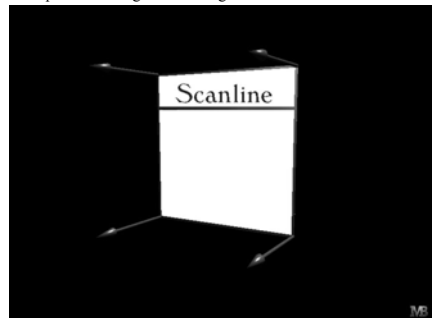
Calculate shading model on each vertex and interpolate it along the edges.



17

## Gouraud shading

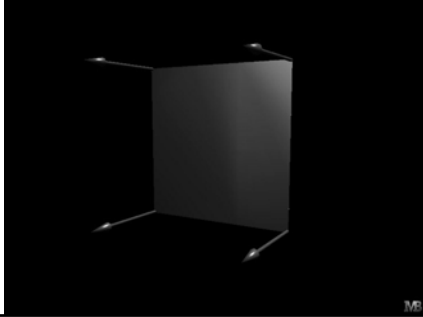
Interpolate shading model along the scanlines



18

## Gouraud shading

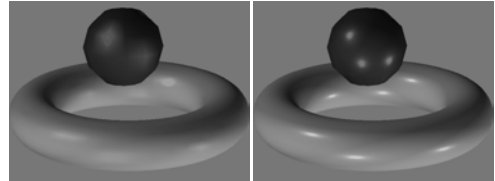
Result, smooth shading with slow (linear) specular effect



19

## Phong Shading

- ◆ Better results can be obtained by linearly interpolating the normals between the vertices, and recomputing the shading at every pixel:



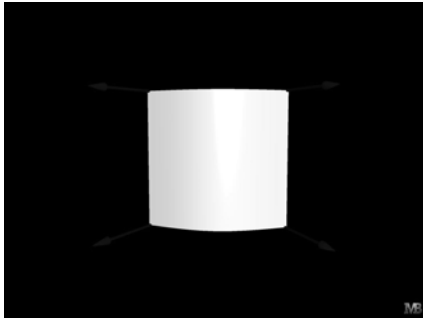
Gouraud shading

Phong shading

20

## Phong Shading

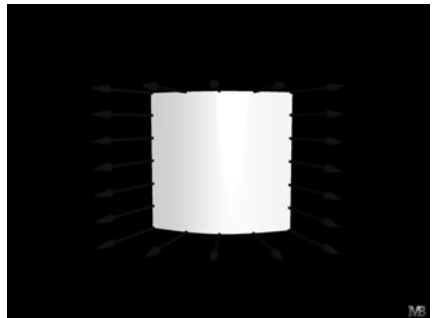
Evaluate the normals at each vertex



21

## Phong Shading

Interpolate normals on the polygon's edges



22

## Phong Shading

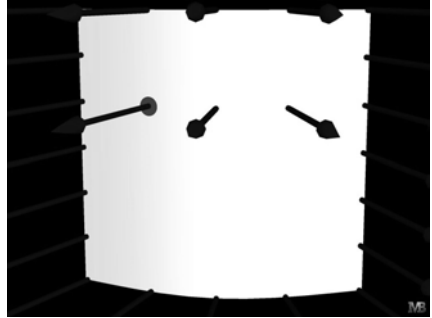
Interpolate normals on each pixel on the scanline



23

## Phong Shading

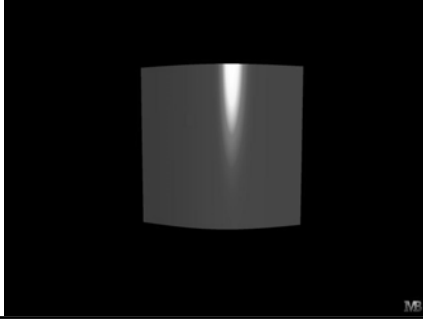
Calculate shading model on each pixel of the scanline



24

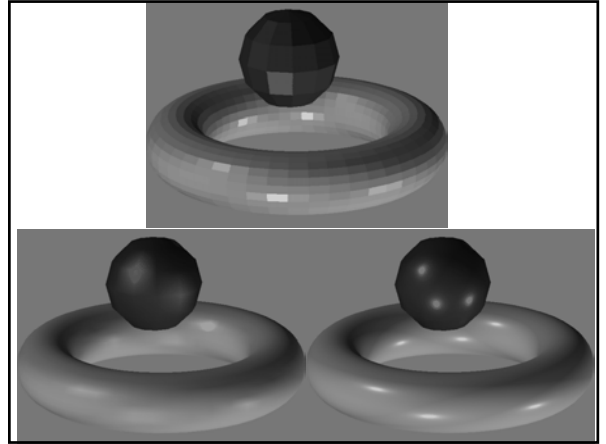
## Phong Shading

Result, shows specular highlight clearly.



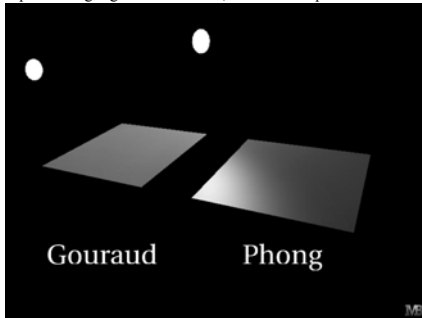
MB

25



## Comparisons

Specular highlights are different, linear and exponential



Gouraud

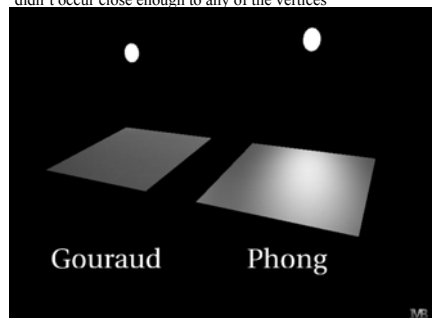
Phong

MB

27

## Comparisons

Specular highlight completely missing since it didn't occur close enough to any of the vertices



Gouraud

Phong

MB

28

## Comparisons

Normals estimation, should not be based only on the polygon's plane. Neighboring polygons should be taken in to account.



MB

29