MATERIALS AND LIGHTING

Illumination Models

- Function or algorithm used to describe the reflective characteristics of a given surface
- Think about how that different surfaces reflect light in different ways
  - Glass vs. concrete vs. velvet
  - The way the light behaves/interacts will be based on viewpoint
- Sometimes known as shading

Illumination Models

- Lambertian
  - Perfectly diffuse surfaces
  - Reflection is constant in all directions (kd)
  - Independent of viewer direction
- Phong
  - Viewer direction becomes important
- Cook-Torrance
  - Based on physics of a surface – geometry (*microfacet*), reflectance, roughness
- Ward
  - *Anisotropic* reflection varies not only with angle of incidence, but also with the angle of the incident light w.r.t some viewing angle.
- Many others!

Vector Review

- For a vector *w*:
  - Magnitude:
    $$|w| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$$
  - Normalized unit vector: $\hat{w} = \frac{w}{|w|} \quad |\hat{w}| = 1$
  - Dot product of vectors *A* and *B*:
    $$A \cdot B = a_x b_x + a_y b_y + a_z b_z = |A||B| \cos\theta$$

Vector Review

- Cross Product

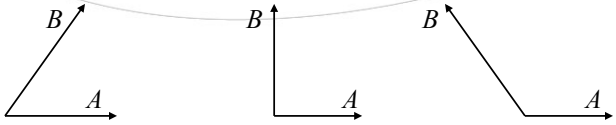
$$\mathbf{u}\times\mathbf{v} \quad \hat{\mathbf{x}}(u_y v_z - u_z v_y) + \hat{\mathbf{y}}(u_z v_x - u_x v_z) + \hat{\mathbf{z}}(u_x v_y - u_y v_x),$$

$$|A \times B| = |A||B| \sin\Theta$$

Perpendicular to both A and B

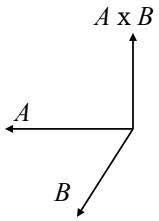
Vector Review

- The sign of the dot product:



$A \cdot B > 0$ $A \cdot B = 0$ $A \cdot B < 0$

- The sign of the cross product uses the right hand rule
  - Fingers align with A, palm faces direction of B
  - Positive non-zero values point in direction of thumb.



The Phong Model

- *Ambient*
  - Light from no direction (due to scattering)
  - Surfaces illuminated by ambient light reflect in all directions
- *Diffuse*
  - Light from a certain direction
  - Reflected equally in all directions from the surface (causing the surface to look equally bright)
- *Specular*
  - Directional light which reflects off a surface in a particular direction
  - Causes the surface to have a shiny highlight
- *Emissive*
  - Light originating within an object
  - Does not affect other objects in the scene
  - We won't cover this in any detail

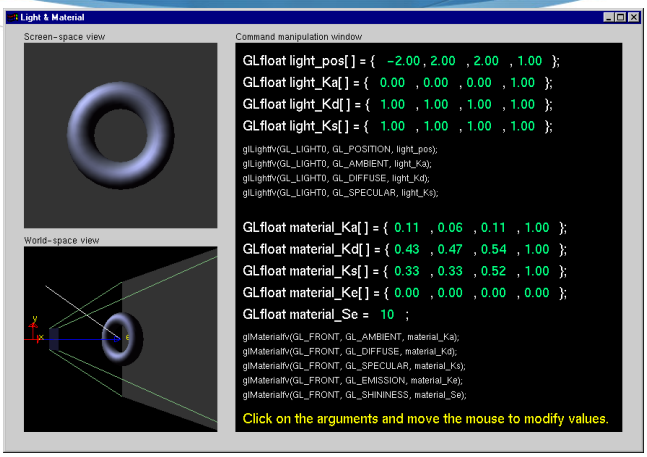
Ambient Light

- No apparent source – simply present
- Nondirectional, uniform illumination
- Illumination equation:

$$A = I_a k_a$$

- I_a is the ambient illumination
- k_a is the *ambient reflection coefficient*
  - The amount of ambient light reflected by an object
  - Property of the object's material
  - Ranges between 0 and 1

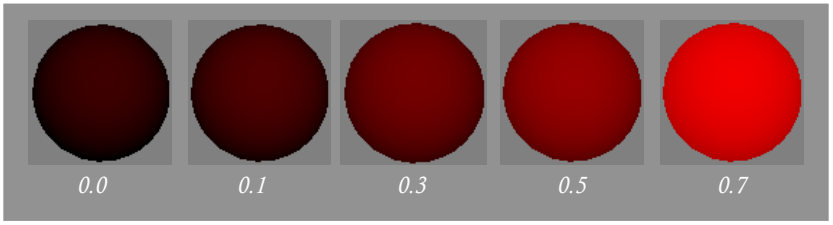
Light & Material Tutorial



Questions

Ambient Light Examples

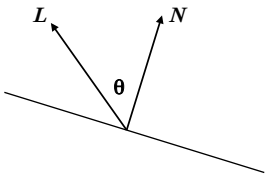
- A **`gluSphere`** illuminated with red light
  - Generated by the MATLIGHT application from the *OpenGL Super Bible*, second edition
  - Ambient red coefficient varied from 0.0 to 0.7
  - Diffuse red coefficient and ambient & diffuse intensities at 0.5



Diffuse Reflection

- Reflection from dull, matte surfaces (e.g., chalk)
  - Also called *Lambertian* reflection
  - Light comes from a point source
  - Light is reflected with equal intensity in all directions
- For a given surface, brightness depends only on the angle between the direction to the light source and the surface normal

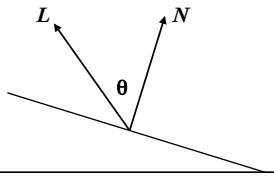
$$D = I_d k_d \cos\Theta$$



- k_d is the *diffuse reflection coefficient*, $0 <= k_d <= 1$.

Diffuse Reflection

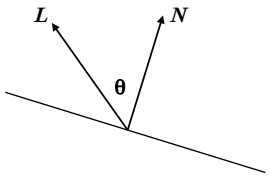
- The angle θ must be between 0° and 90° for the light source to have any effect on the object
  - This means that a light source behind an object doesn't illuminate it
  - We say this object is *self-occluding*
- Technically, should be a *max(cos θ, 0)* term in the equation
- It's simpler to just assume the angle is "legal"



Diffuse Reflection

- If the vectors have been normalized, we can replace *cos θ* with their dot product:

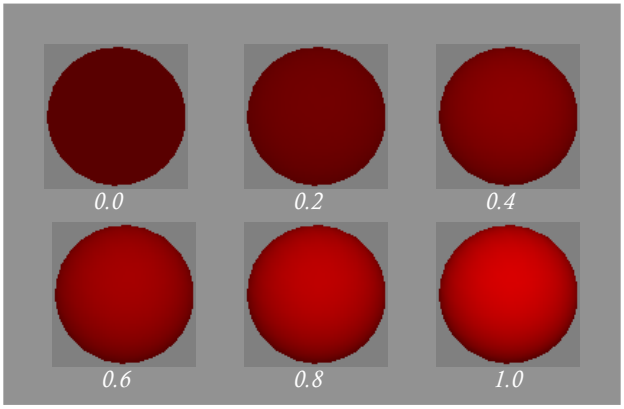
$$D = I_d k_d (L \cdot N)$$



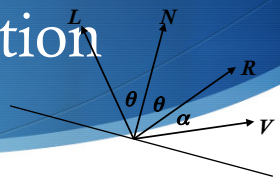
- If the point source is far enough away, it makes essentially the same angle with all surfaces sharing the same surface normal

Diffuse Reflection Examples

- Another `gluSphere` illuminated by red light
  - Ambient red coefficient and ambient & diffuse light intensities at 0.5



Specular Reflection



- Observable on any shiny surface
  - "Highlights" are caused by *specular reflection*
- On a perfectly reflective surface, light is reflected only in one direction
  - Angle of reflection = angle of incidence
- The angle between the reflection and the viewpoint, α, determines the amount of reflection seen
  - Also affected by the *specular reflection exponent* of the material
  - For a perfect mirror, can only see reflection when α is zero

Specular Reflection

- Maximum reflection occurs when α is zero
- Falloff is approximated by $cos^n \alpha$
  - ***n*** is the specular reflection exponent
  - Low values of ***n*** give gentle falloff; high values give sharp, focused highlights
- Note that the color of the reflected light is determined mostly by the color of the light.

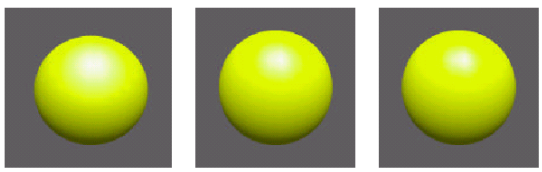


Figure 9.3: specular highlights with specular coefficients 20, 50, and 80 (left, center, and right), respectively

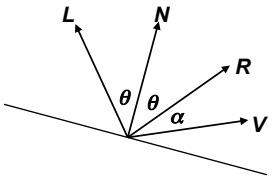
Phong Illumination Model

- Note: don't confuse with the Phong shading model
  - Same person, different algorithm
- Introduced the first *specular* (mirror-like) reflections
  - Viewer direction becomes important
- Model natively supported in OpenGL
- Three components
  - Ambient: background light (k_a)
  - Diffuse: Lambertian reflection (k_d)
  - Specular: mirror-like reflection(k_s)

Phong Illumination Model

ambient diffuse specular

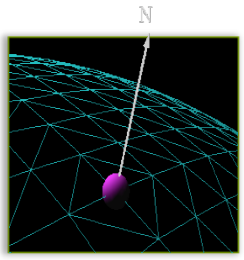
$$I_\lambda = I_{a\lambda} k_a O_{a\lambda} + f_{att} I_{d\lambda} \left[k_d O_{d\lambda} (L \cdot N) + k_s O_{s\lambda} (R \cdot V)^n \right]$$



break

Surface Normals

- Normals define how a surface reflects light
  - Application usually provides normals as a vertex atttribute
  - Use *unit* normals for proper lighting
    - scaling affects a normal's length
  - Cross product of polygon edges



Calculating Surface Normals

```
Begin Function CalculateSurfaceNormal (Input Triangle)
Returns Vector

Set Vector U to (Triangle.p2 minus Triangle.p1)
Set Vector V to (Triangle.p3 minus Triangle.p1)

Set Normal.x to (multiply U.y by V.z) minus (multiply U.z by V.y)
Set Normal.y to (multiply U.z by V.x) minus (multiply U.x by V.z)
Set Normal.z to (multiply U.x by V.y) minus (multiply U.y by V.x)
Returning Normal

End Function
```

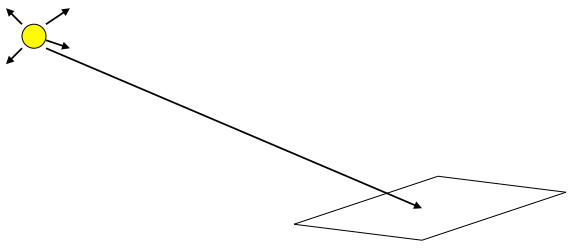
Opengl.org

Material Properties - Phong

Property	Description
Diffuse	Base object color
Specular	Highlight color
Ambient	Low-light color
Emission	Glow color
Shininess	Surface smoothness

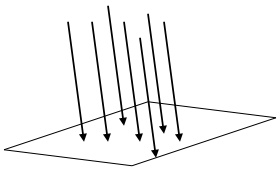
Light - Simple Point Light Source

- Light distributed equally in all directions



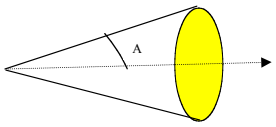
Light - Directional Light

- Light distributed equally from a given direction
- Point light source at infinity is an estimation for sunlight



Light - Spotlight

- Basic CG Spotlight
  - Point light source with a limited beam



Light -- Spotlight

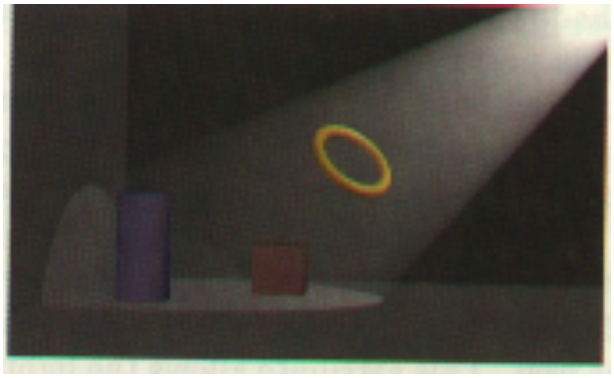
- Spotlight controls
  - Beam shape
  - Beam Falloff
    - How intensity is effected as one moves across the beam
  - Intensity Falloff
    - How intensity is effected by the distance from light source to shading point.

Light - Spotlight

- Intensity falloff
  - Intensity of light will decrease proportional to the inverse of the distance squared.

Light -- Spotlight

- Intensity Falloff



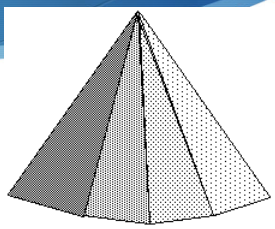
Shading

- Computing the color at a point
- ***Shading point*** - point under investigation
- ***Illumination model*** - function or algorithm used to describe the reflective characteristics of a given surface.
- ***Shading model*** – algorithm for using an illumination model to determine the color of a point on a surface.

Shading Models

- Direct Shading Models
  - Only consider direct lighting from light sources
  - Flat (constant) shading
  - Gouraud Shading
  - Phong Shading
- Global Illumination Algorithms
  - Considers indirect lighting from inter-reflections from other objects
  - Radiosity
  - Ray Tracing

Shading Models



- Flat Shading
  - Illumination model applied once per polygon.
  - Constant color for entire polygon
  - Assumes normal vector is constant across the entire polygon

Shading Models

- Flat shading

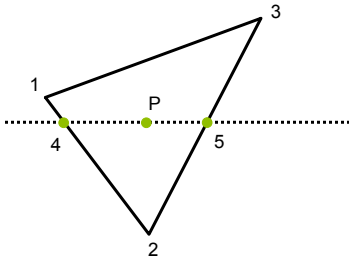
Hwang

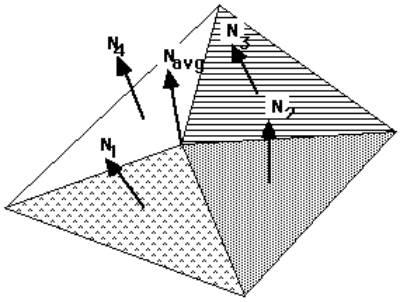
Shading Models

- Gouraud Shading
  - Illumination is interpolated across each polygon
  - Normals required at each polygon vertex, calculated as an average of the normals of each face that shares that vertex
  - Illumination is calculated for each polygon vertex
  - Interior points interpolated from endpoint illumination intensities

Shading Models

- Gouraud Shading – interpolating normals





Shading Model

- Gouraud Shading

Hwang

Shading Models

- Phong Shading
  - Normal vectors are interpolated across each polygon
  - Averaged normal (per Gouraud required at each polygon vertex)
  - Illumination is calculated for each polygon interior point by applying illumination model directly using interpolated normal

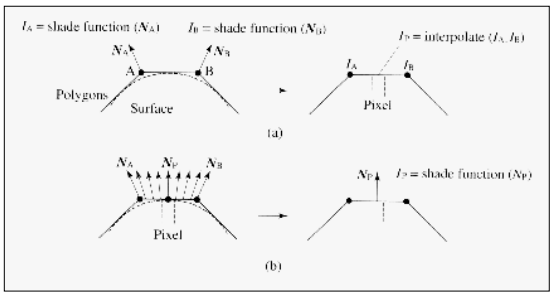
Shading Models

- Phong Shading

Hwang

Shading models

- Phong vs. Gouraud



Shading Models



Flat Gouraud Phong

Summary

- Lights
- Illumination Model
- Shading

- All applied via shader