Shading

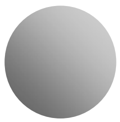
Objectives

- Learn to shade objects so their images appear three-dimensional
- Introduce the types of light-material interactions
- Build a simple reflection model---the Phong model--- that can be used with real time graphics hardware

Why we need shading

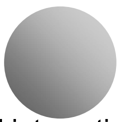
- Suppose we build a model of a sphere using many polygons and color it with `glColor`. We get something like

- But we want



Shading

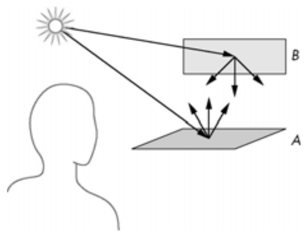
- Why does the image of a real sphere look like



- 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

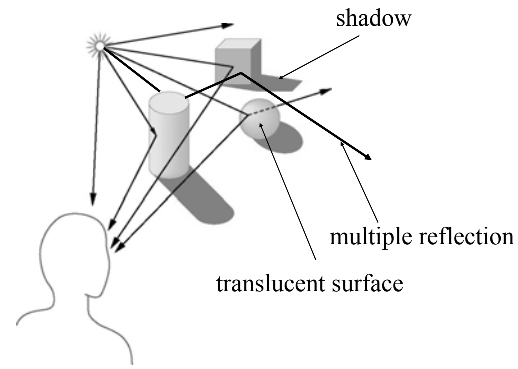
- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A and so on



Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
  - Cannot be solved in general
  - Ray tracing is a special case for perfectly reflecting surfaces
- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object

Global Effects



Local vs Global Rendering

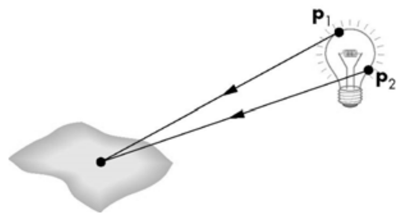
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things "look right"
  - There exist many techniques for approximating global effects

Light-Material Interaction

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

Light Sources

General light sources are difficult to work with because we must integrate light coming from all points on the source

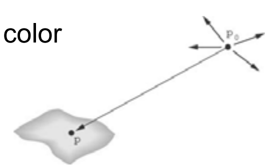


Simple Light Sources

- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

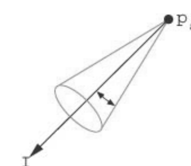
Simple Light Sources

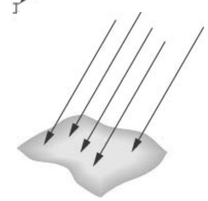
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces
- Point source
  - Model with position and color



Simple Light Sources

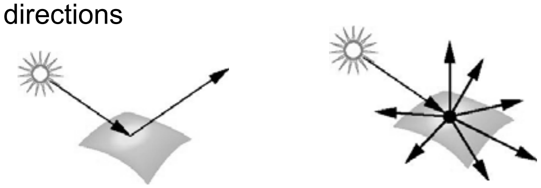
- Spotlight
  - Restrict light from ideal point source to range of angles
- Distant source
  - Treat as infinite distance away
  - rays are all parallel





Surface Types

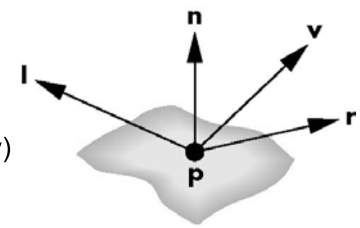
- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflect the light
- A very rough surface scatters light in all directions



smooth surface　　　rough surface

Phong Model

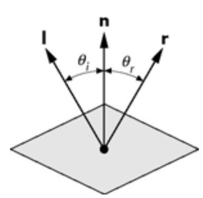
- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To light source (l)
  - To viewer or COP (v)
  - Normal (n)
  - Perfect reflector (r)



Ideal Reflector

- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar
- Thus (see 6.4.2)

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$

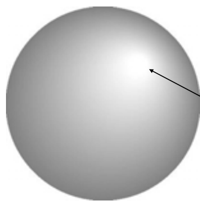


Lambertian Surface

- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light
  - reflected light ~$\cos \theta_i$
  - $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized
  - There are also three coefficients, k_r, k_b, k_g that show how much of each color component is reflected

Specular Surfaces

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



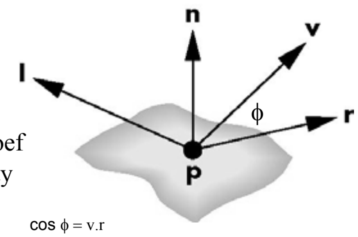
specular highlight

Modeling Specular Relections

- Phong proposed using a term that dropped off as the angle ϕ between the viewer and the ideal reflection increased

$\mathbf{I}_r \sim k_s \mathbf{I} \cos^{\alpha}\phi$

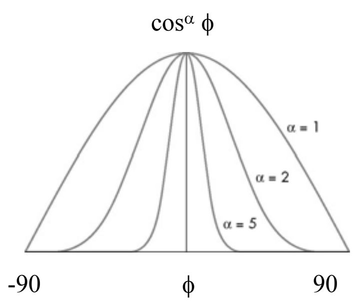
reflected intensity; absorption coefficient; incoming intensity; shininess coef



$\cos\phi = \mathbf{v}.\mathbf{r}$

The Shininess Coefficient

- Light concentrated more in narrow region centered on perfect reflector as α increases
- Values of α between 100 and 200 correspond to metals
- Values between 5 and 10 give surface that look like plastic



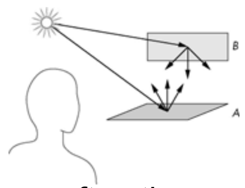
Ambient Light

- Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment
- Amount and color depend on both the color of the light(s) and the material properties of the object
- Add $k_a I_a$ to diffuse and specular terms

reflection coef; intensity of ambient light

Distance Terms

- The light from a point source that reaches a surface is inversely proportional to the square of the distance between them
- We can add a factor of the form $1/(a + bd + cd^2)$ to the diffuse and specular terms



- The constant and linear terms soften the effect of the point source

Light Sources

- In the Phong Model, we add the results from each light source
- Each light source has separate diffuse, specular, and ambient terms to allow for maximum flexibility even though this form does not have a physical justification
- Separate red, green and blue components
- Hence, 9 coefficients for each point source
  - I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}

Material Properties

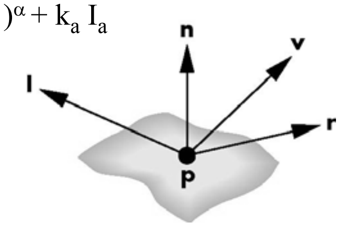
- Material properties match light source properties
  - Nine absorbtion coefficients
    - k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - Shininess coefficient α

Adding up the Components

For each light source and each color component, the Phong model can be written (without the distance terms) as

$$I = k_d I_d\ \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

For each color component we add contributions from all sources



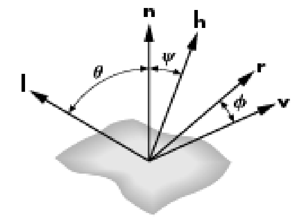
Modified Phong Model

- The specular term in the Phong model is problematic because it requires the calculation of a new reflection vector and view vector for each point on the surface
- Blinn suggested an approximation using the halfway vector that is more efficient

The Halfway Vector

- $\mathbf{h}$ is normalized vector halfway between $\mathbf{l}$ and $\mathbf{v}$

$$\mathbf{h} = (\mathbf{l} + \mathbf{v}) / |\mathbf{l} + \mathbf{v}|$$

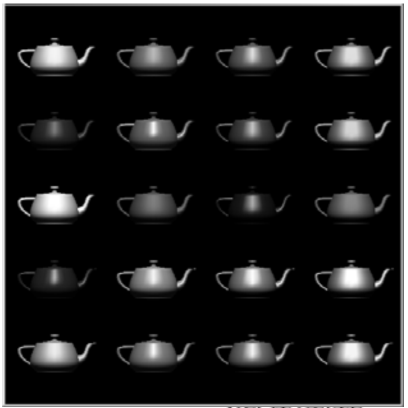


Using the halfway angle

- Replace $(\mathbf{v} \cdot \mathbf{r})^{\alpha}$ by $(\mathbf{n} \cdot \mathbf{h})^{\beta}$
- β is chosen to match shineness
- Note that halfway angle is half of angle between $\mathbf{r}$ and $\mathbf{v}$ if vectors are coplanar
- Resulting model is known as the modified Phong or Blinn-Phong or Blinn lighting model
  - Specified in OpenGL standard

Example

Only differences in these teapots are the parameters in the modified Phong model



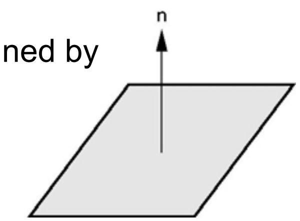
Computation of Vectors

- **l** and **v** are specified by the application
- Can compute **r** from **l** and **n**
- Problem is determining **n**
- For simple surfaces it can be determined but how we determine **n** differs depending on underlying representation of surface
- OpenGL leaves determination of normal to application
  - Exception for GLU quadrics and Bezier surfaces (Chapter 11)

Plane Normals

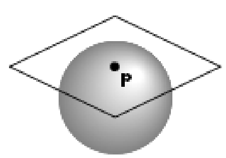
- Equation of plane: $ax+by+cz+d = 0$
- From Chapter 4 we know that plane is determined by three points p_0, p_2, p_3 or normal **n** and p_0
- Normal can be obtained by

$$\mathbf{n} = (p_2 - p_0) \times (p_1 - p_0)$$



Normal to Sphere

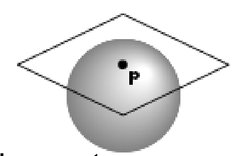
- Implicit function $f(x,y.z) = x^2 + y^2 + z^2 - 1 = 0$
- or $f(\mathbf{p}) = \mathbf{p} \cdot \mathbf{p} - 1$
- Normal given by gradient
- n = $[\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T = [2x, 2y, 2z]^T = \mathbf{2p}$
- **Unit normal n = p**



Parametric Form

- For sphere

x=x(u,v)=cos u sin v
y=y(u,v)=cos u cos v
z= z(u,v)=sin u



- Tangent plane determined by vectors

$\partial\mathbf{p}/\partial u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]T$
$\partial\mathbf{p}/\partial v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]T$

- Normal given by cross product

$$\mathbf{n} = \partial\mathbf{p}/\partial u \times \partial\mathbf{p}/\partial v$$

General Case

- We can compute parametric normals for other simple cases
  - Quadrics
  - Parameteric polynomial surfaces
    - Bezier surface patches (Chapter 11)