Questions about Last Week?

- Information about ACIS will be available tomorrow ... Check the newsgroup

- You should have started to work on the assignment

- We have fudged the data to make things fit better
  - The universe is mostly empty, after all

- Tips about the assignment
  - Ellipsis equation ... check out your favorite math book
  - Ellipses can be generate by non-uniform scaling of circles
  - Caution: Non-uniform scaling of spheres generate ellipsoid
  - Make sure you think about the order of transformations
Overview of Week 4

Lighting Models
- General lighting concepts
- Basic illumination models
- Principles of polygon shading
- Surface properties
- Advanced illumination models

OpenGL
- Rendering primitives
- OpenGL lighting and shading

Lighting

So far we only considered the effects of the viewpoint on the geometry
Even a minimum of realism requires to take into account interactions of light with the scene.

We need models for
- Lights
- Materials and surface properties
- Interaction of lights and material

We will develop several lighting models of increasing complexity and visual accuracy
Shading vs. Lighting

Lighting models define how to compute the color of a surface point, given a number of parameters:
- lighting conditions, e.g. number and positions of light sources
- viewpoint
- surface properties, e.g. surface orientation or material
- atmospheric properties, e.g. fog or smoke

Shading models define when and where a lighting model is evaluated
- at every vertex (Gouraud shading)
- at every pixel (Phong shading)

Lighting Models

Different lighting models have been developed to account for various illumination effects

Physically based Lighting Models
- Model one or several physical lighting effects
- Assumptions are stated explicitly

Empiric Lighting Models
- Many lighting models are not "strictly" based on physics
- Many popular ad-hoc methods deliver attractive results
- Reasons are computational efficiency

We will discuss various such lighting models
Modeling Materials

- The real problem in computing illumination effect stems from the interaction of light with matter
- Materials are characterized by ...
  - color
  - glossiness or dullness
  - reflectivity
  - polarization of reflected light
  - transparency
  - index of refraction
- These properties result from various physical effects
  - We will not try to explain these effects here
  - However, we will try to model / simulate some of them

Bidirectional Reflection Function (BDRF)

- Amount and direction of reflected light for a given direction of incident light.
  - Bidirectional: $\alpha_i$ and $\alpha_R$
  - Many other ways to describe this relationship
- BDRFs for real materials are complex ...
Reflectance Functions

- Reflectance function describe the amount of reflected light for given angles of incidence and observation.
  - Each curve describes the amount of light reflected into various directions
  - Different curves for different angles of incidence
  - Example: For light coming in at 30°, approx. 50% is reflected into a direction of 45°.

Reflectance Function for White Paper

From A. Glassner, Principles of Image Synthesis
**Reflectance Function for the Moon**

From A. Glassner, *Principles of Image Synthesis*

---

**Lighting Concepts (1)**

- At every surface point we consider several vectors:
  - Surface normal $\mathbf{N}$
  - Light source vector $\mathbf{L}$
  - Vector to the eye $\mathbf{E}$
  - Reflected vector $\mathbf{R}$

- Also we will take into account various surface properties of the object
Lighting Concepts (2)

▶ Intensity
- "Amount" of light along a given direction, i.e. luminous intensity
- Usually a vector (multiple colors or wavelengths)
- Can be a scalar (monochromatic light, grey values)
- We will often use the terms "color" and "intensity" as synonyms

▶ Coefficients \( k_x \)
- Dimensionless scale factors affecting the intensity
- For example: absorption, reflectivity, ...
- If \( k_x \) is wavelength dependent it describes a color
  - Example: If only red light is reflected, the object appears red

▶ Wavelength Dependence
- Typically lighting is evaluated for RGB components
- Although, a rough approximation, often produces acceptable results

Lighting Concepts (3)

▶ Direct Illumination
- Light received directly from a light source
- All lighting models account for direct illumination

▶ Indirect Illumination
- Light received via reflecting/refracting intermediate objects
- E.g. mirrors, lenses, fog, ...
- Many lighting models simulate indirect illumination crudely or not all
- Global illumination models perform more accurate estimation of indirect illumination effects
**Directional vs. Point Light Sources**

- **Directional light source**
  - Light source is at infinity
  - All light rays are parallel

- **Point light source**
  - Light source at finite position
  - Light rays radiate in all directions
  - Intersect surface at different angles with normal

- **Similar for eye position**
  - Infinite Viewer
  - Finite Viewer

**Empirical Lighting Models**

- Emission
- Ambient Light
- Diffuse Reflection
- Specular Reflection
- Light Attenuation
- Atmospheric Effects
Emitting Light

- The simplest lighting model is NO lighting model
  - Assigns a constant color to the entire object
  - Ignores any lights in the environment
  - No discernable surface features
  - Useful to model luminous (i.e. light-emitting) objects

\[ I = k_i \]

Ambient Light

- Light with equal intensity in the entire scene: \( I_A \)
  - No preferred direction to light rays
  - Independent of object and viewer position
  - Approximates background light in a scene as light bounces off multiple objects before it reaches a given surface point
  - Background light is a crude simplification of indirect illumination
  - Softens boundaries between lit and unlit regions

- Objects reflect certain percentage of ambient light: \( k_A \)

\[ I = I_A \cdot k_A \]
**Diffuse Reflection**

- **Lambertian Reflection**
  - Diffuse surfaces appear equally bright from all viewing directions.
  - Brightness depends on the angle between surface normal and light.
  
  \[ I = I_L \cdot k_D \cdot \cos \alpha = I_L \cdot k_D \cdot \mathbf{N} \cdot \mathbf{L} \]

**Specular Reflection (1)**

- "Shiny" surfaces reflect light distributed around the direction of ideal reflection.
  - Intensity maximal along direction of ideal reflection.
  
  \[ I = I_L \cdot k_S \cdot \cos^n \beta = I_L \cdot k_S \cdot (\mathbf{R} \cdot \mathbf{E})^n \]
Specular Reflection (2)

- Intensity depends on position of viewer and light source

- Color of specular reflection depends on material
  - Plastics reflect the color of the light source, $k_\lambda$ same for a $\lambda$
  - Metals reflect the material color, $k_\lambda$ different for different $\lambda$
  - Other materials, combination of light and material color

Specular Reflection (3)

- Glossiness coefficient $n$ determines extent of the highlights

- Exponentiation of Cosine roll-off is purely empirical
  - Result look attractive
  - Due to Bui-Tuong Phong
Specular Reflection (4)

- Computing the reflected vector
  - Calculate projection of incident ray \( L \) onto surface normal \( N \)
  - Compute vector from incident ray to projection: \( S \)
  - Reflected ray is \( L + 2S \)

- Reflected vector must recomputed if \( N \) (or \( L \)) changes

\[
R = L + 2S
\]
\[
= L + 2(\mathbf{N} \cdot \mathbf{N} - \mathbf{L})
\]
\[
= 2\mathbf{N} \cdot \mathbf{N} - \mathbf{L}
\]
\[
= 2\cos\alpha \cdot \mathbf{N} - \mathbf{L}
\]

Specular Reflection (5)

- Avoid computing \( R \)
  - Simplification due to Blinn
  - Halfway vector \( H \) bisects angle between \( L \) and \( E \)
  - Maximum intensity if \( H = N \)
  - For directional light and infinite viewer, \( H \) is constant and does not depend on \( N \)

\[
I = I_L \cdot k_s \cdot \cos^n \beta = I_L \cdot k_s \cdot (N \cdot H)^n
\]
Specular Reflection (6)

- Both Phong and Blinn model are strictly empirical
  - Results do not match any real materials
  - Simulate plastic fairly well

- Phong and Blinn model are not identical
  - Similar results
  - Both are approximations anyway

Multiple Light Sources

- Light from different light sources can be superimposed
  - Ignores interference effects
  - Assume incoherent light

\[ I = \sum_{i} I(i) \]
Putting it all together: Phong Lighting

- Useful empirical lighting model
  - Straight-forward to compute
  - Acceptable results
  - Used in many APIs and some hardware implementations

\[
I = k_A \cdot I_A + k_D \cdot \sum_{i=0}^{l} I_i \cdot (N \cdot L_i) + k_S \cdot \sum_{i=0}^{l} I_i (R \cdot E)^n
\]

Demo

Also: check out glaze from E+S
http://www.es.com/Products/Workstation/download.html
Computing Surface Normals (1)

- **Surface normal**
  - Unit length (same goes for all other vectors !!)
  - Non-unit vectors will result in false lighting
  - Perpendicular to surface

- **Perpendicular vectors**
  are computed using the vector cross product

\[
V_3 = V_1 \times V_2 \\
|V_3| = |V_1| \times |V_2| \sin \alpha
\]

Computing Surface Normals (2)

- **For polygons:** Compute normal from 2 edges
  - Edges must not be collinear
  - Numerical problems for almost collinear edges
  - Only triangles are guaranteed to be planar

\[
N = \frac{(P_3 - P_1) \times (P_2 - P_1)}{|(P_3 - P_1) \times (P_2 - P_1)|}
\]
Computing Surface Normals (3)

- Polygon meshes approximate smooth surfaces
- Vertex normals are computed by averaging adjacent surface normals
  - Phong Shading: Interior normals are computed as interpolation of vertex normals

Computing Surface Normals (4)

- Averaging surface normals to obtain vertex normals is a linear approximation
  - Apparent curvature depends on size of participating polygons
  - Interior surface feature may be missed
Transforming Normal Vectors (1)

- Normals are often defined in the local coordinates together with the geometry.
- Normals must therefore also be transformed by modeling and viewing transformations.

Transforming Normal Vectors (2)

- Rigid body transformations work fine.
- Uniform Scaling: Vectors must be renormalized.
- General transformations: Normals do not remain perpendicular to the associated surface.
Transforming Normal Vectors (3)

- Scaling distorts the normal vector, thereby changing its orientation to the surface

- The scaling operation distorts surfaces and normals in opposite directions ...

- ... not surprising given that normals are rotated 90° with respect to the surface.

\[ N' = M \cdot N \]

Transforming Normal Vectors (4)

- Let's try to tackle this a bit more formally:

- Looking at two transformed vectors \( N' \) and \( V' \):
  - All vectors \( V' \) lie in a plane
  - \( N' \) perpendicular to this plane

- We assert that they are perpendicular after the transformation with the matrix \( M \).

\[
\begin{align*}
N' \cdot V' &= 0 \\
N' \cdot (M \cdot V) &= 0 \\
N'^T \cdot (M \cdot V) &= 0 \\
N'^T \cdot M^{T^T} \cdot V &= 0 \\
(M^T \cdot N')^T \cdot V &= 0
\end{align*}
\]
Transforming Normal Vectors (5)

We require that $N$ be perpendicular to any $V$ lying in the given plane.

Therefore: $N \cdot V = 0$

Comparing coefficients yields ...

Normals are transformed by the transpose of the inverse of the transformation matrix.

Transforming Normal Vectors (6)

Normals are transformed by the transpose of the inverse of the transformation matrix.

Special cases:
- Rotation: $M^{-1} = M^T \Rightarrow (M^{-1})^T = M$
  
  Vectors and points are rotated the same way.

- Scaling: $(M^{-1})^T = M^{-1}$
  
  Scaling is accomplished by transforming vectors with the inverse of the transformation matrix.

- Translations do not affect the transformation of vectors ($W = 0$). Therefore, we can transform vectors and points the same way.
Front and Back Faces (1)

➤ Light from a given direction affects only one side of a surface, i.e. the surface is self-occluding.

➤ This requires that normals are consistent across the surfaces, e.g. all normals point outward (or inward).
  ● By convention, for front-facing polygons the vertices are oriented counter-clockwise (mathematically positive).
  ● For front-facing polygons, the normal vector points toward the viewer.

Front and Back Faces (2)

➤ Different materials properties can be assigned to front and back faces to distinguish front and back faces.
  ● Different interior and exterior colors for solid objects
  ● Visual differentiation of top and bottom of a surface
Front and Back Faces (3)

- For solid models, back-facing polygons are always hidden if the viewpoint is outside of the solid.
- Then, no need to process back-facing polygons.

- Back-face culling
  - Only process front-facing polygons.
  - Discard back-facing polygons.

- After modeling and viewing transformation, check surface normal to identify back-facing polygons
- Saves unnecessary work for lighting, clipping, and rasterization

Light Attenuation

- The light intensity diminishes with increased distance from the light source.
- We model this with an attenuation factor: $f_{\text{att}}(d)$
  - For true point lights, the intensity falls off with square of the distance: $f_{\text{att}}(d) = 1 / d^2$
  - Inspite of physics, the results do not appear natural: objects close to the light appear too bright, distant objects appear too dark.
    - Apparent harshness in contrast because true point light sources are rare in reality
  - An empirical model accounts for these shortcomings, using user-defined coefficients for constant, linear and quadratic fall-off:

$$f_{\text{att}}(d) = \frac{1}{c_1 + c_2 d + c_3 d^2}$$
Athmospheric Effects (1)

- Athmospheric effects account for changes to the light as it travels from the object to the viewer: depth cueing.
- Typically involves a shift in the color value:
  - Haze (grey), fog (white), smoke (black), water (blue or green)
  - The farther the object, the less dominant the object color.
- The color shift is modeled by linearly blending between object color and depth-cue color:

\[ I' = s \cdot I + (1 - s) \cdot I_{DC} \]

Athmospheric Effects (2)

- Often, athmosperic attenuation of the objet color is limited to a certain depth range:
  - Distance must be measure in eye coordinates or normalized device coordinates.

\[
s = \begin{cases} 
  S_F & \text{for } z > z_F \\
  S_B + \left( z - z_B \right) \frac{S_F - S_B}{z_F - z_B} & \text{for } z_F > z > z_B \\
  S_B & \text{for } z_B > z 
\end{cases}
\]
Spotlights (1)

Spotlights do not emit light into all directions

Cone
- Cone specifies direction and angle of emitted light
- Light distribution across either uniform or fall-off towards perimeter

Flaps ("barn doors")
- Limit light to range for each coordinate axis
- Only points within those coordinate ranges receive light

Spotlights (2)

Warn lighting model
- Reflect a virtual light through a specular reflector into the scene
- The light distribution is that of a specular reflection with $k_c=1$

\[ I = I_{L'} \cdot \cos^p \gamma \]
Managing Maximum Intensity

None of the lighting models discussed attempted to limit the amount of light in the scene.

- Intensity may be oversaturated, i.e. > 100%
- Graphics hardware clamps color components to 100%, resulting in color shift towards white and loss of color differentiation.

Therefore, intensities must be adjusted.

- Scale back all color channels to maintain color value.
- Attempt to prescale light intensities so that intensity cannot saturate.
  - Consider color emission and maximum reflectance in all material properties.
- Store true intensities in off-screen area and rescale intensities.

Physically-based Lighting Models (1)

Empirical model discussed so far, are approximations of illumination effects that are not derived from physics.

Most shortcomings are related to the specular reflection.

- Empirical models are adequate for small angles of incidence.
- Real BDRFs show very different behavior for large angles of incidence.
- Reflectivity increases for larger angles of incidence.
- Highlight is shifted off the specular peak, i.e. off $R$.
- Color shifts due to interactions of light with the material are not modelled at all by Phong lighting.
Physically-based Lighting Models (2)

▶ Torrance-Sparrow Lighting Model
- Physics-based surface model or microscopic, ideal mirrors
- Mirrors are inclined towards the surface normal, forming V-shaped grooves across the surface
- Orientation of these micro-grooves is given by statistical distribution
- This distribution determines how much incident and reflected light is shadowed by the other mirrors

Lighting as Part of the Rendering Pipeline

▶ Lighting is often part of the geometry pipeline
- Lit vertices are sent to the rasterization stage
- Phong shading moves lighting into the pixel processing step

▶ Lighting occurs in world coordinates (or equiv.)
Shading (1)

- Shading specifies when and where the lighting model is evaluated.

- **Gouraud Shading**
  - Evaluate lighting model at vertices to compute vertex color
  - Interpolate color across polygon
  - Lighting calculations in world/eye coordinate

- **Phong Shading**
  - Interpolate surface normal across polygon
  - Evaluate lighting model at every pixel
  - Lighting calculations starting in screen coordinates

Shading (2)

- Why is Phong shading "better"
  - Interior highlights are missed by Gouraud shading
Shading (3)

Why is Phong shading expensive
- Interpolation of normal vectors generates non-unit vectors

Rendering Primitives (1)

- The basic rendering primitive in most raster graphics systems is the triangle
- Other primitives:
  - Triangle-strip
  - Quad
  - Quad-mesh
  - Polygon
  - Line
  - Poly-line
Rendering Primitives (2)

- All rendering primitives are based on vertices
  - E.g. polygons are defined as a list of vertices

- Primitives are processed by processing their vertices
  - Transformed primitives are the collection of transformed vertices
  - Exploits the property of affine transformations that lines are mapped into lines (or points)
  - Lighting is applied to the vertices (Gouraud shading)

Rendering Primitives (3)

- Triangles
  - Linear functions are naturally defined over triangles
    - Depth varies linearly, e.g. for color, texture, alpha, ...

- Triangle Strips and Fans
  - Connected triangles
  - Neighboring triangles share 2 vertices, don’t need to be redefined for every triangle
  - 3-deep queue of vertices
  - Strips: Use prev. 2 vertices
  - Fans: Reference first and previous vertices
  - Proper traversal to ensure consistent normals
Rendering Primitives (4)

- **Quadrilaterals (Quads)**
  - 4 vertices
  - Not necessarily planar, convex or simple

- **Quad-strips**
  - Connected quads
  - Neighboring quads share 2 vertices
  - 4-deep queue of vertices

Rendering Primitives (5)

- **General polygons**
  - Planar
  - Convex vs. Concave
  - Self-intersecting
  - Holes
Rendering Primitives (6)

- Higher-order primitives
  - Quadrics, e.g. spheres, cylinders, cones etc.
  - Freeform surfaces, e.g. splines, Bezier patches, NURBS

- Tessellation
  - Usually, higher order primitives are approximated with polygons
  - The finer the subdivision, the better the approximation

OpenGL

- OpenGL implements the Phong lighting model with light attenuation and depth cueing
OpenGL: Lights

- **glLight()**
  - At least 8 lights supported
  - Each parameter of the lighting model is set separately
  - Parameters:
    - Ambient, diffuse, specular intensity
    - Glossiness coefficient
    - Position or Direction
    - Spotlight parameters (direction, exponent, angle)
    - Attenuation (constant, linear and quadratic coefficients)

- **glEnable() / glDisable()**
  - GL_LIGHTING
    - Performs lighting calculations or use vertex colors
  - GL_LIGHTi, i = 0, 1, 2, ...

OpenGL: Lighting

- **glLightModel()** specifies
  - global ambient light
  - whether to support local or only infinite viewer
  - whether to apply lighting to front and back faces

- **glShadeModel()**
  - GL_FLAT: Single color for the entire polygon
    - Color specified by one of the vertices in the polygon (see Reference Manual)
  - GL_SMOOTH: Interpolate vertex colors across the polygon
OpenGL: Material Properties

- **glMaterial()**
  - Separate material for front face, back face or both
  - Each material parameter is set separately
  - Parameters:
    - Ambient, diffuse, specular reflectance
    - Glossiness coefficient
  - Material applies to all surfaces defined after call to glMaterial()

- **glColorMaterial()**
  - Material properties are tracking active color
  - More efficient than using glMaterial()
  - Activate parameter tracking w/ glEnable(GL_COLOR_MATERIAL)

OpenGL: Normal Vectors

- **Lighting calculations require surface normals**

- **glNormal()**
  - Specifies normal vector in active coordinate system

- **glEnable (GL_NORMALIZE)**
  - Vectors specified by glNormal() are scaled to unit length
  - Requires extra computation

- **glFrontFace()**
  - Specifies whether front faces are defined by clockwise or counter-clockwise orientation of vertices

- **glCullFace() and glEnable(GL_CULL_FACE)**
  - Discard front-facing (or back-facing) polygons from processing
OpenGL: Depth Cueing

- **glFog()**
  - Specifies depth range of depth cueing
  - Interpolation between object and fog color
    - Linear
    - Exponential (default)
    - Squared-exponential

OpenGL: Rendering Primitives

- Definition of rendering primitives bracketed by `glBegin()` and `glEnd()`

- Argument to `glBegin()` determines primitive type
  - More complex primitives like quadrics, concave polygons, teapot and NURBS are supported by Glu and Glut libraries
Summary

- Lighting model versus shading Model
- Lighting models for ambient, diffuse, specular illumination
  - Phong and Blinn lighting model
  - Light attenuation and atmospheric effects
  - Computation and transformation of normal vectors
- Rendering Primitives

Homework

- Foley et al., Chapter 3.11-13, 19.1
Next Week ...  

Clipping

Line:
\[ P_L = P_1 + t \cdot (P_2 - P_1) \]

Plane:
\[ 0 = ax + by + cz + d \]
\[ 0 = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = V \cdot P \]