Illumination and Shading

- Problem: Model light/surface point interactions to determine final color and brightness
- Apply the lighting model at a set of points across the entire surface
Illumination Model

- The governing principles for computing the illumination

- An illumination model usually considers
  - Light attributes (intensity, color, position, direction, shape)
  - Object surface attributes (color, reflectivity, transparency, etc.)
  - Interaction among lights and objects

Basic Light Sources

- Light intensity can be independent or dependent of the distance between object and the light source

- Point light
- Spot light
- Directional light
- Area light
Local Illumination

- Only consider the light, the observer position, and the object material properties
- OpenGL does this

Global Illumination

- Take into account the interaction of light from all the surfaces in the scene
- Example:
  - Ray Tracing
  - Model light rays bouncing around
Global Illumination (cont.)

- Example:
  - Radiosity
    - Model energy moving from emitters (e.g., lights) into the scene
    - View independent

Simple Local Illumination

- The model used by OpenGL
- Reduce the complex workings of light to three components
  - Ambient
  - Diffuse
  - Specular
- Final illumination at a point (vertex) = ambient + diffuse + specular
- Materials reflect each component differently
  - Use different material reflection coefficients
    - $K_a$, $K_d$, $K_s$
Ambient Light Contribution

- Ambient light = background light
- Light that is scattered by the environment
  - It’s just there
- Frequently assumed to be constant
- Very simple approximation of global illumination
- No direction: independent of light position, object orientation, observer’s position/orientation

\[ \text{Ambient} = I \times K_a \]

Diffuse Light Contribution

- Diffuse light: The illumination that a surface receives from a light source that reflects equally in all direction
  - Eye point does not matter
Diffuse Light Calculation

- Need to decide how much light the object point receives from the light source
  - Based on Lambert’s Law

Diffuse Light Calculation (cont.)

- Lambert’s law: the radiant energy \( D \) that a small surface patch receives from a light source is:
  \[
  \text{Diffuse} = K_d \times I \times \cos(\theta)
  \]
  - \( K_d \): diffuse reflection coefficient
  - \( I \): light intensity
  - \( \theta \): angle between the light vector and the surface normal

\( \theta \): angle between the light vector and the surface normal

\( N \): surface normal
Diffuse Light Examples

$I = 1.0$

$K_d = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$

Specular Light Contribution

- The bright spot on the object
- The result of total reflection of the incident light in a concentrate region

Sees no specular
Sees lots of specular
Specular Light Calculation

- How much reflection you can see depends on where you are
  - But for non-perfect surface you will still see specular highlight when you move a little bit away from the ideal reflection direction
  - $\Phi$ is deviation of view angle from mirror direction
  - When $\phi$ is small, you see more specular highlight

Specular Light Calculation (cont.)

- Phong lighting model
  - Not Phong shading model
    - Specular = $K_s \times I \times \cos(f)$

- The effect of 'f' in the Phong model

$f = 10$  \hspace{1cm} $f = 90$  \hspace{1cm} $f = 30$  \hspace{1cm} $f = 270$
Specular Light Examples

- $K_s = 0.25$
- $K_s = 0.5$
- $K_s = 0.75$

Putting It All Together

- Illumination from a light
  - $\text{Illum} = \text{ambient} + \text{diffuse} + \text{specular}$
  - $= K_a \times I + K_d \times I \times \cos(\theta) + K_s \times I \times \cos^f(\phi)$

- If there are $N$ lights
  - Total illumination for a point $P = \Sigma (\text{Illum})$

- Some more terms to be added (in OpenGL)
  - Self emission
  - Global ambient
  - Light distance attenuation and spot light effect
Putting It All Together (cont.)

\[
\text{Illum} = \text{ambient} + \text{diffuse} + \text{specular}
\]

- color and ambient
- diffuse
- specularity

Ambient Lighting Example
Diffuse Lighting Example

Specular Lighting Example
Adding Color

- Sometimes light or surfaces are colored
- Treat R, G, and B components separately
  - i.e., can specify different RGB values for either light or material
- Illumination equation goes from
  \[ \text{Illum} = \text{ambient} + \text{diffuse} + \text{specular} \]
  \[ = K_a \times I + K_d \times I \times \cos(\theta) + K_s \times I \times \cos^f(\phi) \]

To:

\[
\text{Illum}_r = K_{ar} \times I \times \cos(\theta) + K_{dr} \times I \times \cos(\theta) + K_{sr} \times I \times \cos^f(\phi) \\
\text{Illum}_g = K_{ag} \times I \times \cos(\theta) + K_{dg} \times I \times \cos(\theta) + K_{sg} \times I \times \cos^f(\phi) \\
\text{Illum}_b = K_{ab} \times I \times \cos(\theta) + K_{db} \times I \times \cos(\theta) + K_{sb} \times I \times \cos^f(\phi)
\]

R.W. Lindeman - WPI Dept. of Computer Science

Adding Color (cont.)

<table>
<thead>
<tr>
<th>Material</th>
<th>Ambient</th>
<th>Diffuse</th>
<th>Specular</th>
<th>Exponent f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{ar}$</td>
<td>$K_{dr}$</td>
<td>$K_{sr}$</td>
<td>$K_{ar}$</td>
</tr>
<tr>
<td>Black plastic</td>
<td>0.0</td>
<td>0.01</td>
<td>0.5</td>
<td>32</td>
</tr>
<tr>
<td>Brass</td>
<td>0.329412</td>
<td>0.780392</td>
<td>0.992157</td>
<td>27.8974</td>
</tr>
<tr>
<td>Polished Silver</td>
<td>0.23125</td>
<td>0.2775</td>
<td>0.773911</td>
<td>89.6</td>
</tr>
</tbody>
</table>

(excerpt from Hill, Fig. 8.17)
Lighting in OpenGL

- Adopt Phong lighting model
  - Ambient + Specular + Diffuse lights
  - Lighting is computed at vertices
    - Interpolate across surface (Gouraud/smooth shading)

- Setting up OpenGL lighting
  - Light properties
  - Enable/disable lighting
  - Surface material properties
  - Provide correct surface normals
  - Set light model properties

---

Light Properties

- Properties
  - Colors / Position and type / attenuation

\texttt{glLightfv( light, property, value )}

1. Constant: specify which \textit{light} you want to set the property for \texttt{e.g., GL\_LIGHT0, GL\_LIGHT1, GL\_LIGHT2 \ldots}
   - you can create multiple lights (OpenGL allows at least 8 lights)
2. Constant: specify which light \textit{property} you want to set \texttt{e.g., GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION}
   - (check the red book, or the Web, for more)
3. The value you want to set to the property
Property Example

- Define colors and position a light

```c
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };  // Colors
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };  // Position
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };  // What if I set
GLfloat light_position[] = { 0.0, 0.0, 1.0, 1.0 };  // Position to

glLightfv( GL_LIGHT0, GL_AMBIENT, light_ambient );
glLightfv( GL_LIGHT0, GL_DIFFUSE, light_diffuse );
glLightfv( GL_LIGHT0, GL_SPECULAR, light_specular );
glLightfv( GL_LIGHT0, GL_POSITION, light_position );
```

Types of Lights

- OpenGL supports two types of lights
  - Local light (point light)
  - Infinite light (directional light)

- Determined by the light positions you provide
  - w = 0: Infinite light source
  - w != 0: Point light
    - Position = (x/w, y/w, z/w)

```c
GLfloat light_position[] = { x, y, z, w };  // Position
glLightfv( GL_LIGHT0, GL_POSITION, light_position );
```
Turning on the Lights

- Turn on/off the power (for all the lights)
  - `glEnable(GL_LIGHTING);`
  - `glDisable(GL_LIGHTING);`

- Turn on each light switch
  - `glEnable(GL_LIGHTn)(n = 0, 1, 2, ...)`

Controlling Light Position

- Modelview matrix affects a light’s position
- Two options
  - Option a:
    - Treat light like vertex
    - Do `pushMatrix, translate, rotate, ...` `glLightfv POSITION, popMatrix`
    - Then call `gluLookAt`
    - Light moves independently of camera
  - Option b:
    - Load identity matrix in modelview matrix
    - Call `glLightfv` then call `gluLookAt`
    - Light appears at the eye (like a miner’s lamp)
Material Properties

- The color and surface properties of a material (dull, shiny, etc.)
- How much the surface reflects the incident lights
  - ambient/diffuse/specular reflection coefficients

```c
glMaterialfv( face, property, value );
```

- **face**: Material property for which side of the polygon (e.g., `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`)
- **property**: What material property you want to set (e.g., `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`, `GL_SHININESS`, `GL_EMISSION`, etc.)
- **value**: The value to assign to the property

Material Example

- Define ambient/diffuse/specular reflection and shininess

```c
GLfloat mat_amb_diff[] = { 1.0, 0.5, 0.8, 1.0 }; // Refl. coeff.
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };    // Spec. coeff.
GLfloat shininess[] = { 5.0 }; // Range: dull 0 – very shiny 128

glMaterialfv( GL_FRONT_AND_BACK,
              GL_AMBIENT_AND_DIFFUSE, mat_amb_diff );
glMaterialfv( GL_FRONT, GL_SPECULAR, mat_specular );
glMaterialfv( GL_FRONT, GL_SHININESS, shininess );
```

R.W. Lindeman - WPI Dept. of Computer Science
Surface Normals

- Correct normals are essential for correct lighting
- Associate a normal with each vertex

```gl
glBegin(...);
glNormal3f( u, v, n );
glVertex3f( x, y, z );
...
glEnd( );
```

- All normals must be specified in unit length
  - More on why in the next slide set!

Colored Wireframe
Colored Hidden-Line Removal

Ambient Term Only
Flat Shading

Diffuse Shading + Interp.
Normals
Gouraud Shading

Ambient + Diffuse + Specular
Ambient + Diffuse + Specular + Interpolated Normals

Radiosity
Texture Mapping

Texture Mapping + Ray Tracing