**CMSC 427: Chapter 5**

**Illumination and Shading**

**Reading:** Chapt 10 in Shirley.

**Overview:**
- Light and Illumination Models
- Phong Model: Ambient, Diffuse and Specular Reflection
- Shading: Flat, Gouraud, Phong
- Lighting in OpenGL

---

**Elements of Realistic Rendering**

**Realism in Rendering:**
- Perspective projection
- Illumination and shading
- Texture mapping and surface detail
- Hidden surface removal
- Color
- Reflection and other effects
Overview

- Light and Illumination Models.
- Phong Model: Ambient, Diffuse and Specular Reflection.
- Shading: Flat, Gouraud, Phong.
- Lighting in OpenGL.

Illumination

Illumination Models:
- Light is a very complex physical phenomenon.
- Most illumination models in graphics are based on simple geometric optics, as opposed to more complex (but realistic) wave optics.

Local Illumination Models:
- Point light sources and direct interactions with light.

Global Illumination Models:
- Point-, area-, or volumetric-light sources.
- Models object interactions with and other objects, indirect reflections.

Image generated by Luxology
Local and Global Illumination

Indirect reflected light from walls and floor

Color bleeding from walls

Caustic effects from light refracted through ball

Chapter 5, Slide 5
Copyright © D. M. Mount and A. Varshney

Shading

Shading:
- Determining the intensity of illumination (and its color) incident at a surface point.
- Can be computationally intensive, so it is common to compute accurately at a few points (e.g., vertices) and interpolate in between.

Interpolation techniques:
- Incremental: Updating intensity from pixel to pixel.
- Order of interpolation: How accurate? (zero, linear, quadratic)
- What is interpolated: light intensity? surface normals?
- Surface type: Curved or flat.

Chapter 5, Slide 6
Copyright © D. M. Mount and A. Varshney
Local Illumination

Local Illumination Models:
- Easy to implement and very efficient.
- Provide an acceptable level of realism.
- Often augmented to enhance realism (e.g., draw your own shadows).

Elements of Local Illumination Models:
- Point light sources.
- Local surface geometry.
- No shadows.
- No inter-object reflections or refractions.
- No diffraction or polarization.

Light/Surface Interaction

Light/Surface Interaction:
- Light contacting a surface interacts with it in various ways.
- Light leaving the object (e.g., by reflection) is what we see.

Illumination Model:
- The job of an illumination model is to describe these interactions.
- Tradeoff: Greater realism ⇒ greater computational complexity.

Basic Interactions:
Emission: The surface generates its own light, e.g. phosphorescence.
Reflection: Light strikes the surface and bounces off.
Absorption: Light strikes the surface and dissipates (as heat).
Transmission: Light strikes the surface and passes through.
Light/Surface Interaction

**Reflection:**
- **Ambient:** A background glow that illuminates all objects, irrespective of light source location. (A common cheat used in light local models.)
- **Diffuse:** A uniform scattering of light, characterized by matte (non-shiny) objects, like cloth or foam rubber.
- **Specular:** Shiny (metallic-like) reflection, like a polished wood table.
- **Reflective (Pure):** No light scattering, like a mirror.

**Transmission:**
- **Transparent:** No scattering, like smooth colored glass.
- **Translucent:** Some scattering, like tissue paper.

Perspective View

Image courtesy, Foley, van Dam, Feiner, Hughes
Colored Visible Wireframe

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 13
Copyright © D. M. Mount and A. Varshney

Ambient Illumination

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 14
Copyright © D. M. Mount and A. Varshney
Overview

- Light and Illumination Models.
- Phong Model: Ambient, Diffuse and Specular Reflection.
- Shading: Flat, Gouraud, Phong.
- Lighting in OpenGL.

Ambient Illumination

Ambient Illumination:
- Background illumination.
- Uniform in all directions, independent of light source locations.
- Compensates for the lack of indirect lighting in local models.

Ambient Intensity: (one of many possible models)
- \( I_a \) is the intensity of ambient light source (RGB).
- \( C \) is object's color (RGB).
- \( k_a \) is the coefficient of ambient reflection. The fraction of reflected ambient light.
- \( I_a \) is the intensity of ambient reflection (RGB).

\[
I_a \leftarrow k_a \cdot L_a \cdot C.
\]

- This is a component-wise product applied to RGB components:

\[
I[R]_a \leftarrow k_a \cdot L[R]_a \cdot C[R]_a \\
I[G]_a \leftarrow k_a \cdot L[G]_a \cdot C[G]_a \\
I[B]_a \leftarrow k_a \cdot L[B]_a \cdot C[B]_a
\]
Ambient Illumination

**Ambient Intensity:** (OpenGL model) OpenGL combines $C$ with $k_a$, allowing the user to specify an ambient object color.
- $L_a$ is the intensity of ambient light source (RGB).
- $C_a$ is object’s ambient color (RGB).
- $I_a$ is the intensity of ambient reflection (RGB).

$$I_a \leftarrow L_a \cdot C_a.$$

**Example:** $L_a = (1, 0, 1)$, $C_a = (0.7, 1, 0)$, then $I_a = (0.7, 0, 0)$.

We will present the **OpenGL model** throughout.

---

Diffuse Illumination

Image courtesy, Foley, van Dam, Feiner, Hughes

---

Chapter 5, Slide 17
Copyright © D. M. Mount and A. Varshney

Chapter 5, Slide 18
Copyright © D. M. Mount and A. Varshney
Diffuse Illumination

Diffuse Illumination:
- Also known as Lambertian Illumination.
- Simulates non-shiny surface reflection.
- Based on surface normal and light direction.
- Independent of viewer direction.
- Light is scattered uniformly in all directions.

Lambert’s Cosine Law:
- Reflected luminous intensity varies as the cosine of the angle between the direction of incidence and the local surface normal.
- For a differential area \(dA\), the amount of incident radiation is \(dA \cos \theta\).

Diffuse Intensity: Repeated for each light source.
- \(L_d\) is the intensity of diffuse light emission (RGB).
- \(C_d\) is object’s diffuse color (RGB).
- \(n\) is the normalized (unit length) surface normal vector.
- \(l\) (letter "ell") is the normalized vector pointing to the light source. \((n \cdot l)\) is the cosine of the light incidence angle.
- \(I_d\) is the intensity of diffuse reflection (RGB).

\[
I_d \leftarrow \max(0, n \cdot l) \cdot L_d \cdot C_d.
\]

What does it mean if \((n \cdot l)\) is negative?
Specular Illumination

Specular Illumination:
- Also known as Phong Illumination.
- Used to simulate metallic, shiny surfaces.
- Needs to take into account relative positions of light and viewer.

Specularity brightest when viewer is aligned with reflection vector.

Let $h$ be the vector halfway between the view and light vectors.

Specularity is high if the normal vector $n$ and $h$ are close.
Specular Illumination

Specular Illumination: Repeated for each light source.
- \(L_s\) is the intensity of specular light emission (RGB).
- \(C_s\) is object’s specular color (RGB).
  - **Note:** This should usually be set to \((1, 1, 1)\), since specularity is a reflection of the light’s color, not the object’s color.
- Let \(v\) be the normalized view vector from the surface to viewer.
- Let \(h \leftarrow \text{normalize} \left(\frac{l + v}{2}\right)\) be the halfway vector.
- Let \(\alpha\) be a parameter that controls specular spreading:
  - Low \(\alpha\) produces a diffuse specular reflection.
  - High \(\alpha\) produces a concentrated specular reflection.
- \(I_s\) is the intensity of specular reflection (RGB).
  \[
  I_s \leftarrow \max(0, n \cdot h)^\alpha \cdot L_s \cdot C_s.
  \]

Ambient, Diffuse, and Specular Combined

Examples of parameter settings:
- AmbientStrength = \(C_a\)
- DiffuseStrength = \(C_d\)
- SpecularStrength = \(C_s\)
- SpecularExponent = \(\alpha\).

Image source: Øyvind Andreassen, Univ. of Oslo
Attenuation

**Attenuation**:  
- From physics we know that light intensity decreases as the inverse square of the distance to the light source.  
- In practice, light sources are not pure points, and so this does not produce the most realistic results.  
- OpenGL allows for attenuation to include constant, linear, and quadratic terms.

**Attenuation Factor**:

Let $d$ be the distance from the surface point to the light source.  
Let $a$, $b$, $c$ be parameters set by the user.

**Attenuation Factor**: (multiplied by diffuse and specular terms)

$$ \frac{1}{a + bd + cd^2} $$

Final Illumination Equation

**Final Illumination Equation**:  
- Let $C_c$ denote the color of emitted light from surface (RGB).  
- Assume that there are $m$ light sources.  
- Let $I_f$ denote the final intensity for a surface point.

$$ I_f = C_c + L_c C_a + \sum_{j=1}^{m} \frac{1}{a + bd_j + cd_j^2} \left( \max(0, n \cdot l_j) \cdot L_{sd} C_d + \max(0, n \cdot h_j) \cdot L_{s} C_s \right) $$

**Remember**: This is really 3 equations, for each color component: red, green and blue.
Overview

- Light and Illumination Models.
- Phong Model: Ambient, Diffuse and Specular Reflection.
- Shading: Flat, Gouraud, Phong.
- Lighting in OpenGL.

Illumination vs. Shading

**Illumination model**: How light and surface properties affect illumination.

**Shading**: How pixels are colored (usually a function of lighting model).

- Flat shading: One shade for entire polygon.
- Gouraud shading: Interpolate colors.
- Phong shading: Interpolate surface normals.

Increased realism and computational cost
**Flat Shading**

*Flat Shading*: Compute the color of a polygon, and use that same color on every pixel of the polygon.

- Fast and simple.
- *One normal per polygon*, constant color per polygon.
- Best for flat surfaces to give a *faceted appearance*.

In **OpenGL**:

```
glShadeModel(GL_FLAT)
```

![Images of flat, gouraud, and phong shading]

**Gouraud Shading**

*Gouraud Shading*: (after Henri Gouraud, 1971)

- Compute the color of each vertex.
- Use linear interpolation to color each interior pixel.
- Produces *tolerably good* results for curved surfaces, provided that mesh elements are small.

In **OpenGL**:

```
glShadeModel(GL_SMOOTH)
```

![Images of flat, gouraud, and phong shading]

**Observations:**

- Works quite well for *diffuse shading* where shades vary slowly and if polygon mesh elements are small.
- Problems with *specular reflection*: If no vertex hits the specular region, we see no specular contribution when we interpolate.
Diffuse Illumination & Flat Shading

Note variation in appearance of facets between flat shading and Gouraud.

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 31
Copyright © D. M. Mount and A. Varshney

Diffuse Illumination and Gouraud Shading

Note variation in appearance of facets between flat shading and Gouraud.

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 32
Copyright © D. M. Mount and A. Varshney
Gouraud Shading: Mach Bands

Named after Austrian-Czech physicist and philosopher **Ernst Mach** (1838-1916).

Mach bands:
- Optical illusion resulting from the human visual system's tendency to **enhance** variations in shade.
- Acts like a **high-pass filter**, accentuating discontinuities in the first derivative.
- Caused by **interaction** of neighboring **retinal neurons**.
- A well-known artifact with both **flat** and **Gouraud shading**. Linear interpolation causes **first derivative discontinuities** at polygon edges, and our visual system enhances them.

Images courtesy Univ. of North Carolina

---

Gouraud Shading: More Issues

**Interpolation with Perspective:**
- Linear interpolation among pixels ignores the fact that more **distant pixels** cover **greater depth range**.
- Can fix using **perspective-sensitive** nonlinear interpolation.

**Solutions?**
- **Finer mesh size** minimizes the problem.
  - But suffers from **higher performance cost**.
- Gouraud suggested **higher-order interpolation**:
  - But suffers from **higher performance cost**.
  - Probably **not worth it**.
- **Use a better shading model**
  - **Phong shading** (we'll present next).
  - **Perform shading computations on a per-pixel basis** (modern GPUs).
Phong Shading

Phong Shading:
- Compute the normal for each vertex. (Either supplied by the user or computed automatically from normals of incident faces.)
- For each pixel, interpolate normal vector from the normals at the vertices. (Note: Need to normalize vectors to unit length, because linear interpolation of vectors does not preserve lengths.)
- Compute the color of the pixel (e.g., by Phong illumination model).

Interpolation Methods:

**Linear Interpolation (Lerp):** Given two vectors \( v_1 \) and \( v_2 \),
\[
\text{Lerp}(v_1, v_2, t) = (1-t)v_1 + tv_2 \quad \text{for} \quad 0 \leq t \leq 1.
\]

**Spherical Linear Interpolation (Slerp):** Interpolates cyclically between two vectors \( v_1 \) and \( v_2 \), where \( \theta \) is the angle between them:
\[
\text{Slerp}(v_1, v_2, t) = \frac{\sin(1-t)\theta}{\sin\theta}v_1 + \frac{\sin(t\cdot\theta)}{\sin\theta}v_2 \quad \text{for} \quad 0 \leq t \leq 1.
\]

Performance Issues:
- Computationally more intensive than Gouraud shading.
- Produces much better results, especially for curved surfaces when specular reflection is involved.
- Reduces Mach banding effects, which are evident in Gouraud.

Note:
- Do not confuse Phong illumination with Phong shading.
Specular Illumination and Gouraud Shading

Note variation in specular variations between Gouraud and Phong.

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 37
Copyright © D. M. Mount and A. Varshney

Specular Illumination and Phong Shading

Note variation in specular variations between Gouraud and Phong.

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 38
Copyright © D. M. Mount and A. Varshney
Phong + Specular + Curved Surfaces

Note smoothness of object boundaries.

Overview

- Light and Illumination Models.
- Phong Model: Ambient, Diffuse and Specular Reflection.
- Shading: Flat, Gouraud, Phong.
- Lighting in OpenGL.
Lighting in OpenGL

- Ambient, diffuse, specular illuminations are supported.
- Attenuation is supported.
- Users define light sources: position, type, color.
- The concept of color (for unlit scenes) is replaced with object material for each object drawn.
- Front and back sides of polygons may be given different colors.

OpenGL support for Phong Shading?
- No! OpenGL does not do Phong shading.
- Most graphics APIs (D3D, OGL) employ some variation of Phong illumination and Gouraud shading.

There is a bewildering number of options and parameters that need to be set up in OpenGL in order to use lighting:

- **Lighting/Shading model**: Global parameters that affect how illumination and shading are computed.
- **Light properties**: Options that define the location, colors, and intensities of the lights.
- **Object material properties**: The color of the object and degree of ambient, diffuse, specular reflection.
- **Enabling**: Lighting can be enabled or disabled.

```plaintext
glEnable ( GL_LIGHTING );
glDisable ( GL_LIGHTING );
```
Lights and Lighting:
- OpenGL supports at least 8 light sources: GL_LIGHT0, ..., GL_LIGHT7.
- `glLight*( )`: used to define individual light properties.
- `glLightModel*( )`: used to define global lighting properties.
- To determine the maximum number of lights supported in your implementation use:
  ```
  glGetIntegerv ( GL_MAX_LIGHTS, GLint* num_lights )
  ```
- You need to enable (turn on) each light that you plan to use.
  ```
  glEnable ( GL_LIGHT0 );
  glEnable ( GL_LIGHT1 ); ...
  ```

`glLight*( )`

For **scalar-valued** parameters:
```
glLight{if} ( GLenum  light , GLenum  pname , (TYPE  param  ) )
```
For **vector-valued** parameters:
```
glLight{if}v ( GLenum (light), GLenum (pname), (TYPE)* (param) )
```

where:
- (light) can be: GL_LIGHT0, ..., GL_LIGHT7.
- (pname) can be:
  - **GL_POSITION:**
  - Light position.
  - **GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR:**
  - Light colors. (RGBA vector)
  - **GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF:**
  - Spotlight parameters.
  - **GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION:**
  - Parameters for attenuation. (scalar)
Light Position

Lights at infinity:
- The light position is given as a homogenous [x, y, z, w] vector. When w = 0, this defines a light source at infinity.
- Provides additional efficiency, since light vector is the same for all points in the scene.

When to set light position:
- The [x, y, z, w] vector is given in world coordinates. OpenGL needs to convert these into view-frame coordinates. This is done by multiplying this vector times the Modelview transformation.
- But, with each redraw cycle, the viewer typically moves and we alter the Modelview transformation (by calling gluLookAt()).
- Light positions should be set after issuing the gluLookAt() call.
- Other lighting settings need be done only once.

Sample Lighting Setup

```c
void setUpMyLighting () {
    // light intensity and location
    GLfloat ambientIntensity [4] = { 0.9, 0.0, 0.0, 1.0 };  // red
    GLfloat diffSpecIntensity [4] = { 1.2, 1.2, 1.2, 1.0 };  // white
    GLfloat position [4] = { 2.0, 4.0, 5.0, 1.0 };  
    // global lighting options
    glShadeModel ( GL_SMOOTH );  // (or GL_FLAT)
    glEnable ( GL_LIGHTING );  // enable lighting
    glEnable ( GL_LIGHT0 );  // enable light 0
    // set up light 0 properties
    glLightfv ( GL_LIGHT0,   GL_AMBIENT,   ambientIntensity );
    glLightfv ( GL_LIGHT0,   GL_DIFFUSE,    diffSpecIntensity );
    glLightfv ( GL_LIGHT0,   GL_SPECULAR,  diffSpecIntensity );
    glLightfv ( GL_LIGHT0,   GL_POSITION,  position );
}
```
Drawing Objects with Lighting

\textbf{glMaterial}(\texttt{f})\hspace{.5em}: \text{with lighting use \texttt{glMaterial}(\texttt{f}) to specify colors:}

- Object colors under illumination are computed as a \textit{component-wise multiplication} of the light colors and material colors.
- Material properties can be specified differently for \textit{ambient}, \textit{diffuse}, and \textit{specular} reflection.
- In addition to this \textit{emission} (glowing) can be defined:
  - Lights do not influence emission.
  - Emissive objects do not illuminate other objects.

\textbf{glNormal}(\texttt{f})\hspace{.5em}: 

- Used to specify vertex normals for Gouraud/flat shading.

\begin{itemize}
  \item \textbf{glMaterial}(\texttt{f})\hspace{.5em}: \\
  \hspace{1.5em}\textit{For scalar-valued parameters:} \\
  \quad \texttt{glMaterial}(\texttt{f})\hspace{.5em}\texttt{( GLenum (face), GLenum (pname), (TYPE) (param) )} \\
  \textit{For vector-valued parameters:} \\
  \quad \texttt{glMaterial}(\texttt{f})\hspace{.5em}\texttt{v ( GLenum (face), GLenum (pname), (TYPE)* (param) )} \\
\end{itemize}

\texttt{where:}

- \texttt{(face)} can be:
  \hspace{1.5em}\texttt{GL_FRONT, GL_BACK, GL_FRONT_AND_BACK:}
  \hspace{2em}Indicates which \textit{side} of the polygon is being colored. (Recall that front face is the side from which vertices are listed in CCW order.)

- \texttt{(pname)} can be:
  \hspace{1.5em}\texttt{GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_EMISSION:}
  \hspace{2em}Material colors (RGBA vectors). (You can set both ambient and diffuse at once using \texttt{GL_AMBIENT_AND_DIFFUSE}.)
  \hspace{1.5em}\texttt{GL_SHININESS:}
  \hspace{2em}Specular illumination exponent (called $\alpha$ above).
Sample Drawing with Lighting

```c
void doMyDrawing ( ) {
  GLfloat red[4] = {1.0, 0.0, 0.0, 1.0};  // RGBA object color (red)
  // set material color
  glMaterialfv ( GL_FRONT_AND_BACK,
                  GL_AMBIENT_AND_DIFFUSE, red );

  glBegin ( GL_POLYGON );  // draw polygon
    glNormal3f ( ... );  glVertex3f ( ... );
    glNormal3f ( ... );  glVertex3f ( ... );
    glNormal3f ( ... );  glVertex3f ( ... );
  glEnd ( );
}
```

You can assign different colors to different vertices. OpenGL blends.

In flat shading only one normal is given.

---

**glColorMaterial*()**

**glColorMaterial**: Allows a **simpler** but more limited way to specify material properties using `glColor*( )`.

To use `glColorMaterial`, it must be enabled (and can be disabled):

```c
glEnable ( GL_COLOR_MATERIAL );
...
```

```c
glDisable ( GL_COLOR_MATERIAL );
```

**Example:**

```c
  glEnable ( GL_COLOR_MATERIAL );
  glColorMaterial ( GL_FRONT, GL_DIFFUSE );
  glColor3f ( 0.2, 0.5, 0.8 );  // changes the diffuse material color
                             (Draw objects here)
  glColorMaterial ( GL_FRONT, GL_SPECULAR );
  glColor3f ( 0.9, 0.0, 0.2 );  // changes the specular material color
                             (Draw objects here)
  glDisable ( GL_COLOR_MATERIAL );
```
More and Better Lights

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 51
Copyright © D. M. Mount and A. Varshney

Image Textures

Image courtesy, Foley, van Dam, Feiner, Hughes

Chapter 5, Slide 52
Copyright © D. M. Mount and A. Varshney
Summary

Summary:
- Light and Illumination Models.
- Phong Model: Ambient, Diffuse and Specular Reflection.
- Shading: Flat, Gouraud, Phong.
- Lighting in OpenGL.

What’s Next?
- Texture mapping and surface detail.