Computer Graphics (CS 543)
Lecture 8 (Part 1): Lighting, Shading and Materials (Part 1)

Prof Emmanuel Agu

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Lighting?

- **Problem**: Model light-surface interaction at vertices to determine vertex color and brightness
- Need to calculate angle that surface makes with light
- Per vertex calculation? Usually done in vertex shader
Shading?

• After triangle is rasterized (drawn)
  • Triangle converted to a group of pixels
  • Per-vertex lighting calculation means we know color of pixels coinciding with vertices (red dots)
• Shading problem: figure out color of interior pixels
• How? Assume linear change => interpolate
Lighting (or Illumination) Model?

- Governing principles for computing illumination
- An illumination model usually considers:

1. **Light attributes:** intensity, color, position, direction, shape
2. **Surface attributes**
   - color, reflectivity, transparency, etc
3. **Interaction**
   - between lights and objects
Why do we need Lighting & shading?

- Sphere without lighting & shading

- We want (sphere with shading):
  - Has **visual cues** for humans (shape, light position, viewer position, surface orientation, material properties, etc)
Light Bounces at Surfaces

- Light strikes A
  - Some reflected
  - Some absorbed
- Some reflected light from A strikes B
  - Some reflected
  - Some absorbed
- Some of this reflected light strikes A and so on
- The infinite reflection, scattering and absorption of light is described by the *rendering equation*
Rendered Equation

- Introduced by James Kajiya in 86 Siggraph paper.
- Mathematical basis for all global illumination algorithms

\[ L_o = L_e(x, \vec{\omega}) + \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') (\vec{\omega}' \cdot \vec{n}) d\vec{\omega}' \]

- \( L_o \) is outgoing radiance
- \( L_i \) incident radiance
- \( L_e \) emitted radiance,
- \( f_r \) is bidirectional reflectance distribution function (BRDF)
  - Describes how a surface reflects light energy
  - Fraction of incident light reflected
Rendering Equation

\[ L_0 = L_e(x, \omega) + \int_{\Omega} f r(x, \omega', \omega) L_i(x, \omega')(\omega' \cdot \hat{n}) \, d\omega' \]

- **Rendering equation** cannot be solved in general
- Rendering algorithms solve approximately. E.g. by sampling discretely
- Ray tracing solves special case for perfectly reflecting surfaces
- Rendering equation includes many effects
  - Reflection
  - Shadows
  - Multiple scattering from object to object
Global Illumination (Lighting) Model

- **Global illumination**: model interaction of light from all surfaces in scene (track multiple bounces)

  - shadow
  - multiple reflection
  - translucent surface
Local Illumination (Lighting) Model

- One bounce!
  - Doesn’t track inter-reflections, transmissions

- Simple! Only considers
  - Light
  - Viewer position
  - Surface Material properties
Local vs Global Rendering

- Global Illumination is accurate, looks real
  - But raster graphics pipeline (like OpenGL) renders each polygon independently (local rendering)
  - OpenGL cannot render full global illumination
- However, we can use techniques exist for approximating (faking) global effects
Light-Material Interaction

- Light strikes object, some absorbed, some reflected
- Fraction reflected determines object color and brightness
  - Example: A surface looks red under white light because red component of light is reflected, other wavelengths absorbed
- Reflected light depends on surface smoothness and orientation
Light Sources

- General light sources are difficult to model because we must integrate light coming from all points on light source
Basic Light Sources

- We generally use simpler light sources
- Abstractions that are easier to model

- **Point light**
- **Directional light**
- **Spot light**
- **Area light**

Light intensity can be **independent** or **dependent** of the distance between object and the light source.
Phong Model

- Simple lighting model that can be computed quickly
- 3 components
  - Diffuse
  - Specular
  - Ambient
- Compute each component separately
- Vertex Illumination = ambient + diffuse + specular
- Materials reflect each component differently
  - Material reflection coefficients control reflection
Phong Model

- Compute lighting (components) at each vertex (P)
- Uses 4 vectors, from vertex
  - To light source (l)
  - To viewer (v)
  - Normal (n)
  - Mirror direction (r)
Mirror Direction?

- Angle of reflection = angle of incidence
- Normal is determined by surface orientation
- The three vectors must be coplanar

\[ r = 2 \left( l \cdot n \right) n - l \]
Surface Roughness

- **Smooth surfaces:** more reflected light concentrated in mirror direction
- **Rough surfaces:** reflects light in all directions
Diffuse Lighting Example
Diffuse Light Reflected

- Illumination surface receives from a light source and reflects equally in all directions

Eye position does not matter
Diffuse Light Calculation

- How much light received from light source?
- Based on Lambert’s Law

Receive more light

Receive less light
Diffuse Light Calculation

- Lambert’s law: radiant energy $D$ a small surface patch receives from a light source is:

$$D = I \times k_D \cos(\theta)$$

- $I$: light intensity
- $\theta$: angle between light vector and surface normal
- $k_D$: Diffuse reflection coefficient.

Controls how much diffuse light surface reflects
Specular light example
Specular light contribution

- Incoming light reflected out in small surface area
- Specular bright in **mirror direction**
- Drops off away from mirror direction
- Depends on viewer position relative to mirror direction

Mirror direction: Sees lots of specular

Away from mirror direction: Sees a little specular
Specular light calculation

- Perfect reflection surface: all specular seen in mirror direction
- Non-perfect (real) surface: some specular still seen away from mirror direction
- $\phi$ is deviation of view angle from mirror direction
- Small $\phi = \text{more specular}$
Modeling Specular Reflections

\[ I_s = k_s \cdot I \cdot \cos^\alpha \phi \]

- \( I_s \): reflected intensity
- \( k_s \): shininess coefficient
- \( I \): incoming intensity
- \( \cos^\alpha \phi \): piano scale
- \( \alpha \): absorption coefficient
The Shininess Coefficient, $\alpha$

- $\alpha$ controls falloff sharpness
- High $\alpha$ = sharper falloff = small, bright highlight
- Low $\alpha$ = slow falloff = large, dull highlight
  - $\alpha$ between 100 and 200 = metals
  - $\alpha$ between 5 and 10 = plastic look
Specular light: Effect of ‘α’

\[ I_s = k_s I \cos^\alpha \phi \]

\( \alpha = 10 \)

\( \alpha = 90 \)

\( \alpha = 30 \)

\( \alpha = 270 \)
Ambient Light Contribution

- Very simple approximation of global illumination
- Assume to be a constant
- No direction!
  - Independent of light position, object orientation, observer’s position or orientation

\[ \text{Ambient} = I_a \times K_a \]
Ambient Light Example

**Ambient:** background light, scattered by environment
Light Attenuation with Distance

- Light reaching a surface inversely proportional to square of distance
- We can multiply by factor of form $1/(ad + bd + cd^2)$ to diffuse and specular terms
Adding up the Components

- Adding all components (no attenuation term), the Phong model for each light source can be written as:

\[ I = k_d I_d \cos \theta + k_s I_s \cos \phi^\alpha + k_a I_a \]

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

- Note:
  - \( \cos \theta = \mathbf{l} \cdot \mathbf{n} \)
  - \( \cos \phi = \mathbf{v} \cdot \mathbf{r} \)
Separate RGB Components

- We can separate red, green and blue components
- Instead of 3 light components $I_d, I_s, I_a$,
  - E.g. $I_d = I_{dr}, I_{dg}, I_{db}$
  - 9 coefficients for each point source
    - $I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}$
- Instead of 3 material components $k_d, k_s, k_a$,
  - E.g. $k_d = k_{dr}, k_{dg}, k_{db}$
  - 9 material absorption coefficients
    - $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$
Put it all together

- Instead of:
  \[ I = k_d I_d (l \cdot n) + k_s I_s (v \cdot r)^\alpha + k_a I_a \]

- We computing lighting for RGB colors separately
  \[
  I_r = k_{dr} I_{dr} l \cdot n + k_{sr} I_{sr} (v \cdot r)^\alpha + k_{ar} I_{ar} \\
  I_g = k_{dg} I_{dg} l \cdot n + k_{sg} I_{sg} (v \cdot r)^\alpha + k_{ag} I_{ag} \\
  I_b = k_{db} I_{db} l \cdot n + k_{sb} I_{sb} (v \cdot r)^\alpha + k_{ab} I_{ab}
  \]

- Above equation is just for one light source!!

- For N lights, repeat calculation for each light

Total illumination for a point \( P = \sum \) (Lighting for all lights)
## Coefficients for Real Materials

<table>
<thead>
<tr>
<th>Material</th>
<th>Ambient Kar, Kag, kab</th>
<th>Diffuse Kdr, Kdg, kdb</th>
<th>Specular Ksr, Ksg, ksb</th>
<th>Exponent, $\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black plastic</td>
<td>0.0</td>
<td>0.01</td>
<td>0.5</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.01</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.01</td>
<td>0.5</td>
<td></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></td>
<td>0.223529</td>
<td>0.568627</td>
<td>0.941176</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.027451</td>
<td>0.113725</td>
<td>0.807843</td>
<td></td>
</tr>
<tr>
<td>Polished Silver</td>
<td>0.23125</td>
<td>0.2775</td>
<td>0.773911</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>0.23125</td>
<td>0.2775</td>
<td>0.773911</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.23125</td>
<td>0.2775</td>
<td>0.773911</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8.17, Hill, courtesy of McReynolds and Blythe
References