Game Programming

Robin Bing-Yu Chen
National Taiwan University
Game Lighting

- Rendering Pipeline
- Illumination Model
- Shading Models
Rendering Pipeline

1. Polynomial Evaluator
2. Display List
3. Per Vertex Operations & Primitive Assembly
4. Rasterization
5. Texture Memory
6. Per Fragment Operations
7. Frame buffer

CPU

Pixel Operations
The Phong Illumination Model

- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To source
  - To viewer
  - Normal
  - Perfect reflector
Basics of Local Shading

- **Diffuse reflection**
  - light goes everywhere; colored by object color

- **Specular reflection**
  - happens only near mirror configuration; usually white

- **Ambient reflection**
  - constant accounted for other source of illumination

```
ambient = diffuse + specular
```
Ambient Shading

- add constant color to account for disregarded illumination and fill in black shadows; a cheap hack.
Diffuse Shading

- Assume light reflects equally in all directions
  - Therefore surface looks same color from all views; “view independent”
Illumination Models

- **Ambient Light**: \( I = I_a k_a \)
  - \( I_a \): intensity of the ambient light
  - \( k_a \): ambient-reflection coefficient: \( 0 \sim 1 \)

- **Diffuse Reflection**: \( I = I_p k_d \cos \theta \)
  - \( I_p \): point light source’s intensity
  - \( k_d \): diffuse-reflection coefficient: \( 0 \sim 1 \)
  - \( \theta \): angle: \( 0^\circ \sim 90^\circ \)
Diffuse Reflection

\[ I = I_p k_d (\mathbf{N} \cdot \mathbf{L}) \]

- \( \mathbf{L} \) is the direction to the light source.
- \( \mathbf{N} \) is the surface normal.
- \( \theta \) is the angle between \( \mathbf{N} \) and \( \mathbf{L} \).
Examples

diffuse-reflection model with different $k_d$

ambient and diffuse-reflection model with different $k_a$

and $I_a = I_p = 1.0$, $k_d = 0.4$
Light-Source Attenuation

\[ I = I_a k_a + f_{att} I_p k_d (\vec{N} \cdot \vec{L}) \]

- \( f_{att} \): light-source attenuation factor
- if the light is a point source
  \[ f_{att} = \frac{1}{d_L^2} \]
- where \( d_L \) is the distance the light travels from the point source to the surface
  \[ f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right) \]

\[ \text{Light-Source Attenuation} \]

\[ I = I_a k_a + f_{att} I_p k_d (\vec{N} \cdot \vec{L}) \]

- \( f_{att} \): light-source attenuation factor
- if the light is a point source
  \[ f_{att} = \frac{1}{d_L^2} \]
- where \( d_L \) is the distance the light travels from the point source to the surface
  \[ f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right) \]
Examples

\[ \frac{1}{d_L^2} \]

\[ \frac{1}{d_L} \]
Colored Lights and Surfaces

- If an object’s **diffuse color** is 
  \[ O_d = (O_{dR}, O_{dG}, O_{dB}) \]
  then 
  \[ I = (I_R, I_G, I_B) \]
  where for the red component
  \[ I_R = I_{aR} k_a O_{dR} + f_{att} I_{pR} k_d O_{dR} (\vec{N} \cdot \vec{L}) \]
  however, it should be
  \[ I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} k_d O_{d\lambda} (\vec{N} \cdot \vec{L}) \]
  where \( \lambda \) is the **wavelength**
Specular Shading

- Some surfaces have highlights, mirror like reflection; view direction dependent; especially for smooth shiny surfaces
Specular Surfaces

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

- \( \vec{L} \)
- \( \vec{N} \)
- \( \vec{R} \) direction of reflection
- \( \vec{V} \) direction to the viewpoint

\[ \theta \theta \alpha \]
The Phong Illumination Model

\[ I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \int_{att} I_{p\lambda} [k_d O_{d\lambda} \cos \theta + W(\theta) \cos^n \alpha] \]

- \[ W(\theta) = k_s \]: specular-reflection coefficient: \(0 \sim 1\)

so, the Eq. can be rewritten as

\[ I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \int_{att} I_{p\lambda} [k_d O_{d\lambda} (\hat{N} \cdot \hat{L}) + k_s (\hat{R} \cdot \hat{V})^n] \]

consider the object’s \textbf{specular color}

\[ I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \int_{att} I_{p\lambda} [k_d O_{d\lambda} (\hat{N} \cdot \hat{L}) + k_s O_{s\lambda} (\hat{R} \cdot \hat{V})^n] \]

- \[ O_{s\lambda} \]: specular color
The Phong Illumination Model

\[ a \cos^2 \alpha \]

\[ a \cos^4 \alpha \]

\[ a \cos^8 \alpha \]

\[ a \cos^{64} \alpha \]
Examples

$k_s$

$0.1$

$n = 3.0$
$n = 5.0$
$n = 10.0$
$n = 27.0$
$n = 200.0$

$0.25$

$0.5$
Specular Shading

diffuse

diffuse + specular
Multiple Light Sources

If there are \( m \) light sources, then

\[
I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \leq i \leq m} f_{att_i} I_{p_{\lambda_i}} [k_{d\lambda} O_{d\lambda} (\vec{N} \cdot \vec{L}_i) + k_{s\lambda} O_{s\lambda} \cos^n \alpha_i]
\]

\[
\approx I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \leq i \leq m} f_{att_i} I_{p_{\lambda_i}} [k_{d\lambda} O_{d\lambda} (\vec{N} \cdot \vec{L}_i) + k_{s\lambda} O_{s\lambda} (\vec{R}_i \cdot \vec{V})^n]
\]

\[
\approx I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \leq i \leq m} f_{att_i} I_{p_{\lambda_i}} [k_{d\lambda} O_{d\lambda} (\vec{N} \cdot \vec{L}_i) + k_{s\lambda} O_{s\lambda} (\vec{N} \cdot \vec{H}_i)^n]
\]
Computing Lighting at Each Pixel

- Most accurate approach: Compute component illumination at each pixel with individual positions, light directions, and viewing directions
- But this could be expensive...

\[ I_1 \]
\[ I_2 \]
\[ I_3 \]
\[ I_a \]
\[ I_b \]
\[ I_p \]
\[ y \]
\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ y_s \]

Scan line
Shading Models for Polygons

- Flat Shading
  - Faceted Shading
  - Constant Shading
- Gouraud Shading
  - Intensity Interpolation Shading
  - Color Interpolation Shading
- Phong Shading
  - Normal-Vector Interpolation Shading
Flat Shading

Assumptions

- The light source is at infinity
- The viewer is at infinity
- The polygon represents the actual surface being modeled and is not an approximation to a curved surface
Flat Shading

- Compute constant shading function, over each polygon
- Same normal and light vector across whole polygon
- Constant shading for polygon

$$I_p = I$$
Intensity Interpolation (Gouraud)

\[ I_a = I_1 \frac{y_s - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2} \]

\[ I_b = I_1 \frac{y_s - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_s}{y_1 - y_3} \]

\[ I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a} \]
Normal Interpolation (Phong)

\[
N_a = N_1 \frac{y_s - y_2}{y_1 - y_2} + N_2 \frac{y_1 - y_s}{y_1 - y_2}
\]

\[
N_b = N_1 \frac{y_s - y_3}{y_1 - y_3} + N_3 \frac{y_1 - y_s}{y_1 - y_3}
\]
Gouraud v.s. Phong Shading

Gouraud  Phong  Gouraud  Phong
Shadows

\[ I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \leq i \leq m} S_i f_{att} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \cdot \vec{L}_i) + k_s O_{s\lambda} (\vec{R}_i \cdot \vec{V})^n] \]

\[ S_i = \begin{cases} 
0, & \text{if light } i \text{ is blocked at this point} \\
1, & \text{if light } i \text{ is not blocked at this point} 
\end{cases} \]