Illumination and Shading

Computer Graphics – Lecture 4

Taku Komura
What are Lighting and Shading?

- **Lighting**
  - How to compute the color of objects according to the position of the light, normal vector and camera position
  - Phong illumination model

- **Shading**
  - Different methods to compute the color of the entire surface
The procedure of producing images

1. For every vertex of the object, prepare its attributes (normal vectors, colors, etc)
2. Project the vertices onto the screen
3. Interpolate the attributes to determine the color of the pixel (rasterization)
Overview

- Lighting
  - Phong Illumination model
    - diffuse, specular and ambient lighting

- Shading
  - Flat shading
  - Gouraud shading
  - Phong shading
The eye works like a camera
Lots of photo sensors at the back of the eye
Sensing the amount of light coming from different directions
Similar to CMOS and CCDs
What Affects the Light that Comes into the Eye

- position of the point
- position of the light
- color and intensity of the light
- camera vector
- normal vector of the surface at the vertex
- physical characteristics of the object (reflectance model, color)
Phong Illumination Model

● Simple 3 parameter model
  ● The sum of 3 illumination terms:
    ● **Diffuse**: non-shiny illumination and shadows
    ● **Specular**: bright, shiny reflections
    ● **Ambient**: 'background' illumination

\[
\text{Diffuse} + \text{Specular} + \text{Ambient} = R_c
\]
Diffuse Reflection (Lambertian Reflection)

• When light hits an object
  – If the object has a rough surface, it is reflected to various directions

• Result: Light reflected to all directions

• The smaller the angle between the incident vector and the normal vector is, the higher the chance that the light is reflected back

• When the angle is larger, the reflection light gets weaker because the chance the light is shadowed / masked increases
Diffuse Reflection

\[ I = I_p k_d \cos \theta \]

- \( I_p \): Light Intensity
- \( \theta \): the angle between the normal vector direction towards the light and the light source
- \( k_d \): diffuse reflectivity

Example: sphere (lit from left)

No dependence on camera angle!
Specular Reflection

- Direct reflections of light source off shiny object
  - The object has a very smooth surface
  - Specular highlight on object
Specular Reflection

- Direct reflections of light source off shiny object
  - specular intensity $n = \text{shiny reflectance of object}$
  - Result: specular highlight on object

$$I = I_p k_s (\cos \alpha)^n$$

No dependence on object color.
• Specular light with different $n$ values
Combining Diffuse and Specular Reflections
What is Missing?

- Only the side that is lit by the light appears brighter
- The other side of the object appears very dark, as if it is in the space
Multiple Light Sources

- If there are multiple light sources, we need to do the lighting computation for each light source and sum them altogether.

\[ I_\lambda = I_a k_a + \sum_{p=1}^{\text{lights}} I_p [k_d \cos \theta + k_s \cos^n \alpha] \]
Ambient Lighting

- Light from the environment
- Light reflected or scattered from other objects
- Coming uniformly from all directions and then reflected equally to all directions
- A precise simulation of such effects requires a lot of computation
Ambient Lighting

Simple approximation to complex 'real-world' process

Result: globally uniform color for object

$I = \text{resulting intensity}$

$I_a = \text{light intensity}$

$k_a = \text{reflectance}$

$I = k_a I_a$

Example: sphere
Combined Lighting Models

• Summing it altogether: Phong Illumination Model

\[ I_\lambda = I_a k_a + I_p \left[ k_d \cos \theta + k_s \cos^n \alpha \right] \]

Ambient (color) + Diffuse (directional) + Specular (highlights) = \( R_c \)
Using the Dot Products

Use dot product of the vectors instead of calculating the angles $\theta, \alpha$

\[ I_\lambda = I_a k_a + \sum_{p=1}^{\text{lights}} I_p [k_d (\overrightarrow{N} \cdot \overrightarrow{L}) + k_s (\overrightarrow{V} \cdot \overrightarrow{R})^n] \]

$\overrightarrow{V}$: Vector from the surface to the viewer

$\overrightarrow{N}$: Normal vector at the colored point

$\overrightarrow{R}$: Normalized reflection vector

$\overrightarrow{L}$: Normalized vector from the colored point towards the light source
Demo applets

Color

\[ I^R_\lambda = I_a k_a^R + \sum_{p=1}^{\text{lights}} I_p^R (k_d^R (\vec{N} \cdot \vec{L}) + k_s^R (\vec{V} \cdot \vec{R})^n) \]

\[ I^G_\lambda = I_a k_a^G + \sum_{p=1}^{\text{lights}} I_p^G (k_d^G (\vec{N} \cdot \vec{L}) + k_s^G (\vec{V} \cdot \vec{R})^n) \]

\[ I^B_\lambda = I_a k_a^B + \sum_{p=1}^{\text{lights}} I_p^B (k_d^B (\vec{N} \cdot \vec{L}) + k_s^B (\vec{V} \cdot \vec{R})^n) \]

- Finally color the pixel by the RGB color
Exercise 1

Light position
(0,1, \sqrt{2}, 1, \sqrt{2})

Camera
(0,0,1)

N(0,0,1)

P(0,0,0)

Ip = 1, kd = 1, ks = 1, n = 2

What is the diffuse colour?
What is the specular colour?
What is the specular colour if the camera position
(0, \sqrt{3}/2, 1/2)?
Attenuation

- Haven’t considered light attenuation – the light gets weaker when the object is far away.
- Use $1/(s+k)$ where $s$ relates to eye-object distance and $k$ is some constant for scene.

$$I_\lambda = I_a k_a + \sum_{p=1}^{\text{lights}} \frac{I_p}{(s_o + k)} (k_d (\overrightarrow{N} \cdot \overrightarrow{L}) + k_s (\overrightarrow{V} \cdot \overrightarrow{R})^n) + I_t k_t$$
Local Illumination Model

- Considers light sources and surface properties only.
  - Not considering the light reflected back onto other surfaces
- Fast real-time interactive rendering.
- Cost increases with respect to the number of light sources
- Most real-time graphics (games, virtual environments) are based on local illumination models
- Implementation - OpenGL, Direct3D
What Cannot be Rendered by The Empirical Reflectance Model

- Brushed Metal
- Marble surface
Overview

- Lighting
  - Phong Illumination model
    - diffuse, specular and ambient lighting
- Shading
  - Flat shading
  - Gouraud shading
  - Phong shading
How do we color the whole surface?

- The illumination model computes the color of sample points
- How do we color the entire object?

→ This is done at the rasterization level

The procedure to color the entire surface is called **shading**
Shading Models

- Flat Shading (lighting computation is done once per polygon)
  → less computation needed
- Gouraud shading (once per vertex)
- Phong Shading (once per pixel)
  → heavy computation needed
Flat Shading

- Compute the color at the middle of the polygon
- All pixels in the same polygon are colored by the same color
- Works well for objects really made of flat faces.
Flat Shading

- Suffers from Mach band effect
- Humans are very sensitive to the sudden change of the brightness
- The artefact remains although the polygon number is increased
Mach Band (by Ernst Mach)

- An optical illusion
Gouraud Shading
Gouraud Shading (by Henri Gouraud)

- Computing the color per vertex by local illumination model
- Then, interpolating the colors within the polygons

We can interpolate the color by barycentric coordinates
Computing the Vertex Normals

Find vertex normals by averaging face normals

\[ \overline{N}_V = \frac{\sum_{1 \leq i \leq n} \overline{N}_i}{\sqrt{\sum_{1 \leq i \leq n} \overline{N}_i^2}} \]

Use vertex normals with desired shading model,

Interpolate vertex intensities along edges.

Interpolate edge values across a scanline.
\[
\begin{align*}
\text{vertex 1} & \quad \text{triangle 1, triangle 2} \\
\text{vertex 2} & \quad \text{triangle 6, triangle 7} \\
\text{vertex 3} & \quad \text{triangle 9, triangle 10}
\end{align*}
\]
Problems with Gouraud Shading.

- For specular reflection, highlight falls off with $\cos^n \alpha$

- Gouraud shading linear interpolates – makes highlight too big.

- Gouraud shading may well miss a highlight that occurs in the middle of the face.
Gouraud shading is not good when the polygon count is low
Phong Shading (by Bui Tuong Phong)
Phong Shading (by Bui Tuong Phong)

- Doing the lighting computation at every pixel during rasterization
- Interpolating the normal vectors at the vertices (again using barycentric coordinates)

![Phong shading model]
Phong Shading

- For specular reflection, highlight falls off with $\cos^n \alpha$.
- Can well produce a highlight that occurs in the middle of the face.
Phong example
Problems with interpolation shading.

Problems with computing vertex normals.

Face surface normals and averaged vertex normals shown.

Unrepresentative so add small polygon strips along edges or test angle and threshold to defeat averaging.
Problems with interpolation shading.

Problems with computing vertex normals.

A, B are shared by all polygons, but C is not shared by the polygon on the left.

Shading information calculated to the right of C will be different from that to the left.

Shading discontinuity.

Solution 1: subdivide into triangles that share all the vertices
Solution 2: introduce a ‘ghost’ vertex that shares C’s values
Recommended Reading

- Foley et al. Chapter 16, sections 16.1.6 up to and including section 16.3.4.
- Introductory text Chapter 14, sections 14.1.6 up to and including section 14.2.6.