CSE 167 (FA21) Computer Graphics: Lighting

Albert Chern

(slides courtesy Jürgen Schulze)
Normal shading

- Coloring based on surface normal
  - X component maps to Red.
  - Y component maps to Green.
  - Z component maps to Blue.

- Need to map the range \([-1,1]\) of the components of the normal vector to the range \([0,1]\) for color.

\[\text{Color} = 0.5 \times \mathbf{n} + (0.5,0.5,0.5)\]
Realistic shading

- Appearance = Material definition + light sources.
- Compute interaction of light with surfaces
- Requires simulation of physics
- “Global illumination”
  - Multiple bounces of light
  - Computationally expensive, minutes per image
  - Used in movies, architectural designs, etc.
Interactive applications

- No physics-based simulation
- Simplified models
- Reproduce perceptually most important effects
- “Local illumination”
  - Only one bounce of light between light source and viewer
Local illumination

- Light can be reflected by
  - Mirror
  - White wall
  - Glossy material
  - etc

- Gives material its color
Local illumination

• Model reflection of light at surfaces
  ▸ Assumption: no subsurface scattering

• Bidirectional reflectance distribution function (BRDF)
  ▸ Given light direction, viewing direction, how much light is reflected towards the viewer (per unit range of direction)
  ▸ For any pair of light/viewing direction!

= 0.2
= 0.05
= 0.8
Local illumination

- Simplified model
  - Sum of 3 components
  - Covers a large class of real surfaces
Local illumination

• Simplified model
  ▶ Sum of 3 components
  ▶ Covers a large class of real surfaces
Diffuse reflection

- Ideal diffuse material reflects light equally in all directions
- View-independent
- Matte, non-shiny materials
  - Paper
  - Unpolished wood, stone
- Provides visual cues
  - Surface curvature
  - Depth variation

portion of the light that contributes to a unit area of the surface
Diffuse reflection

- Beam of parallel rays shining on a surface
  - Amount of rays intercepted by the surface per unit area varies with the angle between the beam and the normal.
- Lambert’s cosine law (1760)
  - Incident light per unit area is proportional to the cosine of the angle between the light direction and the normal.
  - Object darkens as normal turns away from light.
  - Diffuse surfaces are also called Lambertian surfaces.
Diffuse reflection

• Given
  ▶ Unit (normalized) surface normal $\mathbf{n}$
  ▶ Unit (normalized) light direction $\mathbf{l}$
  ▶ Material diffuse reflectance (material color) $C_{\text{diffuse}}$
  ▶ Light color (intensity) $L$

• The reflected diffuse color (intensity) is

$$R_{\text{diffuse}} = C_{\text{diffuse}} L \max(\mathbf{n} \cdot \mathbf{l}, 0)$$

zero out negative cosines

product of colors?!
Note on multiplication of colors

- There are two types of color vectors
  - **Light** $L$ (values are positive)
  - **Transmittance/reflectance** $C$ (values usually range $[0,1]$)
- Both have RGB channels
- Product of a light and a transmittance is a light
  \[ R = C L \]
  with the multiplication carried out componentwise
  \[ R_i = C_i L_i, \quad i = r, g, b \]
- Product of transmittances is still a transmittance. There is no product between lights.
  \[ R = C_1 C_2 C_3 L \]
Note on multiplication of colors

- The color finally shown on the pixel is of the type of **light** (rather than **transmittance**)
  - The **value of light** can be greater than 1.
  - Over exposure can happen.
  - 8-bit integer \{0, \ldots, 255\} can represent \([0,1]\) interval, but not values greater than 1.
  - To keep the real value of light, one can use float to store the color. This is called **high dynamic range image (HDRI)**
  - Or, one can map between \([0,1]\) and \([0,a]\) intervals using \(y = ax^\gamma\). This is called **tone mapping**. The use of the power law is called **Gamma color correction**.
Local illumination

• Simplified model
  ▶ Sum of 3 components
  ▶ Covers a large class of real surfaces
Specular reflection

- Shiny surfaces
  - Polished metal
  - Glossy car finish
  - Plastics

- Specular highlight
  - Blurred reflection of the light source
  - Position of highlight depends on viewing direction
Specular reflection

• Ideal specular reflection
  ▶ Perfectly smooth surface
  ▶ Incoming light is bounced in single direction
  ▶ Angle of incidence equals angle of reflection

same angle, coplanar
Specular reflection

• Reflection direction
  ▶ Given unit surface normal \( \mathbf{n} \) and light direction \( \mathbf{l} \)
  ▶ The projection of \( \mathbf{l} \) on \( \mathbf{n} \) is \((\mathbf{n} \cdot \mathbf{l})\mathbf{n}\)
  ▶ The unit reflection direction \( \mathbf{r} \) satisfies
    \[
    \frac{\mathbf{r} + \mathbf{l}}{2} = (\mathbf{n} \cdot \mathbf{l})\mathbf{n}
    \]
  ▶ Therefore, \( \mathbf{r} = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l} \)
Specular reflection (glossy)

- Many materials are not perfect mirrors
  - Glossy material
- Microscopic variation/noise of normals
- Smooth surface has sharp highlight
- Rough surface has blurred highlight
Specular reflection (glossy)

- Most light still follow the mirror reflection direction.
- Due to microscopic variation/noise of normals, some light is reflected off the ideal reflection direction.
  - Brightest when view vector $\mathbf{v}$ aligns with reflection $\mathbf{r}$.
  - Decreases as the angle between $\mathbf{v}$ and $\mathbf{r}$ increases.
Phong reflection model

- Developed by Bui Tuong Phong (1973)
- Let $C_{\text{specular}}$ be the specular reflectance coefficient.
- Let $p$ be the Phong exponent (bigger $p$ gives sharper the highlight)

$$R_{\text{Phong}} = C_{\text{specular}} L \left[ \max (\mathbf{v} \cdot \mathbf{r}, 0) \right]^p$$
Blinn–Phong reflection model

• Modified by Jim Blinn (1977)

• Compute the half-way vector

\[ h = \frac{v + l}{|v + l|} \]

• Replace \((v \cdot r)\) by \((n \cdot h)\)

\[ R_{\text{BlinnPhong}} = C_{\text{specular}} L \left[ \max (n \cdot h, 0) \right]^\sigma \]

• For distant light and camera, \(h\) is constant. This can speed up the rendering.
Blinn–Phong reflection model

\[ \sigma = 4p \]
Local illumination

• Simplified model
  ▸ Sum of 3 components
  ▸ Covers a large class of real surfaces
Ambient light

- In real world, light is bounced all around the scene
- Areas with no direct illumination are not completely dark
- Could use global illumination techniques to simulate
- Simple approximation

\[ \mathbf{R}_{\text{ambient}} = \mathbf{C}_{\text{ambient}} \mathbf{L} \]
Complete Phong shading model

- Phong model supports multiple light sources

\[ R = E + \sum_j L_j (C_{\text{ambient}} + C_{\text{diffuse}} \max(n \cdot l_j, 0) + C_{\text{specular}} [\max(n \cdot h_j, 0)]^\sigma) \]

self-emission
Complete Phong shading model

- Phong model supports multiple light sources

\[ R = E + \sum \frac{L_j}{f_j(d_j)} (C_{\text{ambient}} + C_{\text{diffuse}} \max(n \cdot l_j, 0) + C_{\text{specular}} [\max(n \cdot h_j, 0)]^\sigma) \]

- \( d_j \): distance to the j-th light
- \( f(x) = a_0 + a_1 x + a_2 x^2 \)
More recent shading model

- Ambient occlusion (2010’s)
  - At every point on the surface, we ask how much of the sky is occluded by other geometries
  - The computed result can be stored
  - Shade the pixels according to occlusion in real time
Types of shading

- Per triangle
- Per vertex
- Per pixel
Per-triangle shading

- Also known as *flat shading*
- Evaluate shading once per triangle, based on normal vector
- Advantage
  - Fast
- Disadvantage
  - Faceted appearance
Per-vertex shading

• Also known as **Gouraud shading** (Henri Gouraud 1971)

• Compute color per vertex, then interpolate the result across triangles

• Advantage
  ▶ Fast
  ▶ Smoother surface appearance than flat shading

• Disadvantage
  ▶ Problem with highlights
Per-pixel shading

- Also known as **Phong interpolation** (not to be confused with Phong’s illumination model)
  - Let the rasterizer interpolate normals (instead of colors) across triangle
  - Illumination evaluated at each fragment
  - Simulates shading with normals of a curved surface
- Advantage
  - High rendering quality
- Disadvantage
  - Slightly slower (not really)
- We always use per-pixel shading in CSE167
Per-pixel shading

- Flat Shading
- Gouraud Shading
- Phong Shading
# HW3

\[ R = E + \sum_j L_j (C_{\text{ambient}} + C_{\text{diffuse}} \max(n \cdot l_j, 0) + C_{\text{specular}} [\max(n \cdot h_j, 0)]^\sigma) \]

- Make sure all positions & vectors are respecting a common coordinate system (world or camera frame)
- Are normal vectors transformed correctly?
- Light positions might be at infinity (w coord = 0)
- Useful way to debug is to set pixel color by variables you want to visualize.

```glsl
in vec4 position; // raw position in the model coord
in vec3 normal; // raw normal in the model coord

uniform mat4 modelview; // from model coord to eye coord
uniform mat4 view; // from world coord to eye coord

// Material parameters
uniform vec4 ambient;
uniform vec4 diffuse;
uniform vec4 specular;
uniform vec4 emission;
uniform float shininess;

// Light source parameters
const int maximal_allowed_lights = 10;
uniform bool enablelighting;
uniform int nlights;
uniform vec4 lightpositions[ maximal_allowed_lights ];
uniform vec4 lightcolors[ maximal_allowed_lights ];

// Output the frag color
out vec4 fragColor;

void main (void) {
    // HW3: You will compute the lighting here.
}
```