Illumination Model

Wireframe rendering
simple, ambiguous

Color filling
flat without any 3D information

Requires modeling interaction of light with the object/surface to have a different color (shade) in 3D
Illumination Model

The reflected light is scattered depending upon the surface properties and incident light.

**Ambient** light comes from all directions, is scattered in all directions.

**Diffuse** light comes from one direction and is scattered in all directions.

**Specular** light comes from one direction and is scattered in preferred direction.
Illumination Model

Phong Illumination Model

\[ I_{\text{total}} = \text{ambient reflection} + \text{diffuse reflection} + \text{specular reflection} \]
\[ = k_a I_a + k_d l_i \cos \theta + k_s l_i \cos^n \alpha \]
\[ = k_a I_a + k_d l_i (L \cdot N) + k_s l_i (R \cdot V)^n \]
\[ = k_a I_a + \sum_{i=1}^{m} k_d l_i (L_i \cdot N) + k_s l_i (R_i \cdot V)^n \]
Illumination Model

Phong Illumination Model

Local computation for obtaining color (intensity) at a point of the surface

Basic inputs are light(s), material properties
Polygon Shading

Shading
Process of applying illumination model to surface points

Polygon (approximates the 3D shape/surface)

Approaches

• Flat Shading
• Gouraud Shading
• Phong Shading
Polygon Shading

Flat Shading

One intensity for the whole polygon

constant shading

For each face/polygon

1. Select a point P on the face
2. Find normal to the face $n_p$
3. Find intensity I at P
4. Fill the polygon with I

Not smooth
Polygon Shading

Flat Shading

Example
Polygon Shading

Flat Shading

- Computationally fast
- Not smooth
- Mach Band effect
Polygons Shading

Gouraud Shading

Smooth shading

• Compute intensity at vertices of a polygon
  ⇒ Needs vertex normal

• Fill the interior with shade (intensity) using interpolation
Gouraud Shading

Vertex Normal

Normal at the vertex is average of normals of the faces incident at the vertex

\[ N_v = \frac{1}{\sum_{i=1}^{P} ||n_i||} \sum_{i=1}^{P} n_i \]
Rendering

Polygon Shading

Gouraud Shading

Vertex Normal
Polygon Shading

Gouraud Shading

Interpolation

Scan line

Scan conversion!
Gouraud Shading

Interpolation

Scan line

\[(x_1, y_1) \quad I_1 \quad (x_a, y_s) \quad (x_s, y_s) \quad (x_b, y_s) \quad (x_4, y_4) \]

\[I_a = \frac{1}{y_1 - y_2} [l_1(y_s - y_2) + l_2(y_1 - y_s)]\]

\[I_b = \frac{1}{y_1 - y_4} [l_1(y_s - y_4) + l_4(y_1 - y_s)]\]

\[I_s = \frac{1}{x_b - x_a} [l_a(x_b - x_s) + l_b(x_s - x_a)]\]
Polygon Shading

Gouraud Shading

Example
Polygon Shading

Gouraud Shading

Handling Specular Reflections- Highlights

Not Right
Polygon Shading

Phong Shading

*Interpolate normals and then compute intensity*

Not to confuse with Phong Illumination Model
Phong Shading

Polygon Shading

Scan line

Illumination using $N_s$

$(x_1, y_1)$

$N_1$

$(x_2, y_2)$

$N_2$

$(x_3, y_3)$

$N_3$

$N_a$

$N_b$

$N_4$

$(x_4, y_4)$

$N_s$ $\rightarrow$ $I_s$
Polygon Shading

Phong Shading

- More accurate specular component
- Reduced Mach band effect
- Better shape approximation

Original surface

Computationally more intensive
Rendering

Polygon Shading

Phong Shading

Example
Polygon Shading

Phong Shading

Example
Polygon Shading

Problems

Interpolated shading

Polygon Silhouette
Problems

Interpolated shading

Interpolation Inaccuracy (screen space vs world space)

Methods:
- Linear Interpolation
- Perspective Interpolation
Animation

Polygon Shading

Problems

Interpolated shading

Rotate
Rendering

Polygon Shading

Problems

Interpolated shading

Vertex Normal

Vertex Normals

Face Normals
Polygon Shading

Graphics Pipeline Order

Illumination computation is done *early* after modeling transformation

Shading is done *towards the end* with rasterization (scan conversion)
Polygon Shading

Transparency

Simple Model

Non refractive

\[ I = kl_{\text{reflected}} + (1 - k)l_{\text{transmitted}} \]

\( k \) is the opacity factor (coefficient)