Picking

- Goal: To use the mouse (2D) to select 3D objects
- Analytical method
  - gluUnproject
  - expensive

What are we trying to find?

- The objects that lie on the line that projects to the mouse position

Screen corresponds to Canonical View Volume

- What sliver lies under the mouse?

Scale Sliver to Screen: gluPickMatrix

- After Viewing Transform
- Before Clipping

How to know what gets drawn?

- OpenGL Selection Modes (Picking and Feedback) (chapter 13)
- Add "names" to rendering stream

Illumination and Smooth Shading

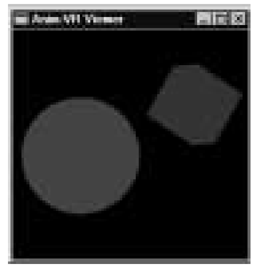
Local Reflection Models

- Function of Surface and Lights

Illumination and Smooth Shading

- Illumination Models
  - Ambient
  - Diffuse
  - Attenuation
  - Specular Reflection
- Interpolated Shading Models
  - Flat, Gouraud, Phong
  - Problems

Illumination Models: Ambient Light

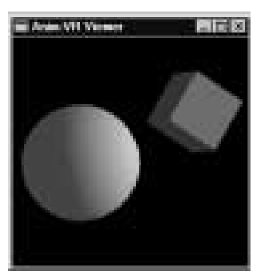


- Simple illumination model

  $I = k_i$
- Use nondirectional lights

  $I = I_a k_a$
- I_a = ambient light intensity
- k_a = ambient-reflection coefficient
- Uniform across surface

Diffuse Light



- Account for light position
  - Ignore viewer position
- Proportional to cosΘ between N and L

  $I = I_p k_d \cos\Theta = I_p k_d (N \cdot L)$
- Model:

  $I = I_a k_a + I_p k_d (N \cdot L)$

Attenuation: Distance

- f_{att} models distance from light
  $I = I_a k_a + f_{att} I_p k_d (N \cdot L)$
- Realistic
  $f_{att} = 1/(d_L^2)$
- Hard to control, so use
  $f_{att} = 1/(c_1 + c_2 d_L + c_3 d_L^2)$

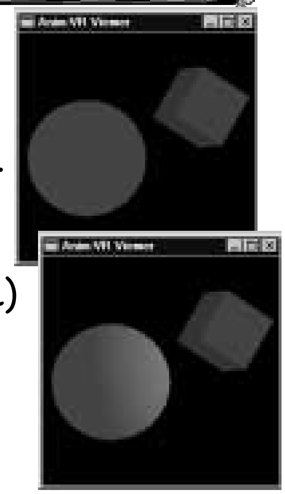
Attenuation: Atmospheric (fog, haze)

- z_f and z_b: near/far depth-cue plane
- s_f and s_b: scale factors
- I_{dc}: depth cue color
- Given $z_f > z_0 > z_b$ interpolate $s_f > s_0 > s_b$
- Adjust intensity
  $I' = s_0 I + (1 - s_0) I_{dc}$

Colored Lights

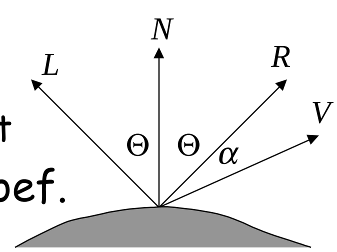
- O_d: diffuse color
  - (O_{dR}, O_{dG}, O_{dB})
- Compute for each component
- i.e.

  $I_R = I_{aR} k_a O_{dR} + f_{att} I_{pR} k_d O_{dR} (N \cdot L)$
- Note: single k, use O_d for ambient and diffuse



Specular Reflection: Phong Model

- Account for viewer position
  - Create highlights
- Based on $\cos^n \alpha = (R \cdot V)^n$
  - Larger n, smaller highlight
- k_s: specular reflection coef.



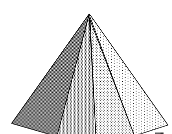
$$I = I_a k_a O_d + f_{att} I_p [\, k_d O_d (N \cdot L) + k_s (R \cdot V)^n]$$

Multiple Light Sources

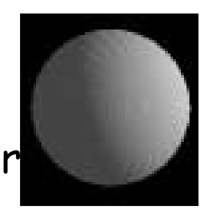
- Obvious summation over *m* lights:

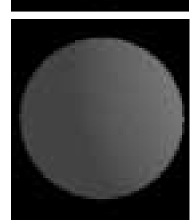
$$I = I_a k_a O_d + \sum_{1 \le 0 \le m} f_{att\,i} I_{p\,i} [\, k_d O_d (N \cdot L_i) + k_s (R_i \cdot V)^n]$$

Shading Models: Flat Shading



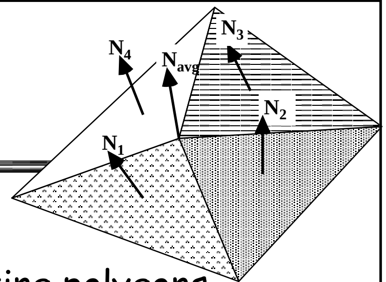
- Compute one color for polygon
  - Use polygon normal in lighting eqs.
- Every pixel is assigned same color



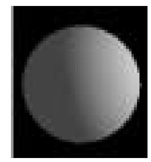


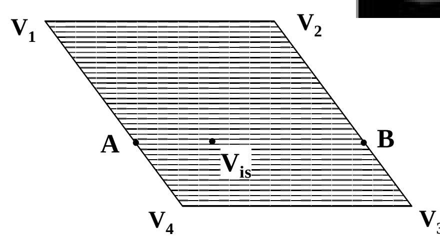
- Fast and simple
- Shade of polygons independent

Gouraud Shading



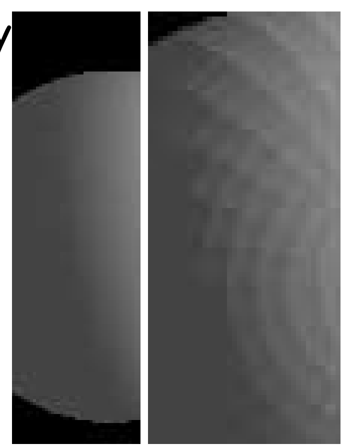
- Compute vertex normals
  - Average normals of abutting polygons
- Use vertex normal in lighting eqs.
- Linearly interpolate vertex intensities
  - Along edges
  - Along scan lines





Gouraud Shading

- Often appears dull, chalky
  - Lacks accurate specular component
    - If included, will be averaged over entire polygon
- Mach banding
  - Artifact at discontinuities in intensity or intensity slope

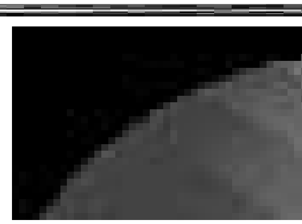


Phong Shading

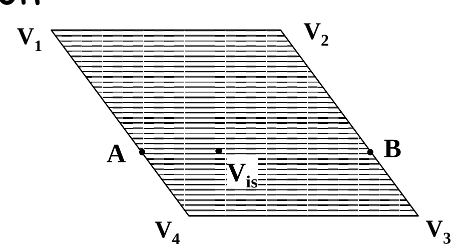
- Linearly interpolate the vertex normals
  - Compute lighting eqs. at each pixel
    - Normals must be backmapped to WC
- Can use specular component
- Approximate by recursive subdivision and Gouraud shading

Problems with Interpolated Shading

- Polygonal silhouette

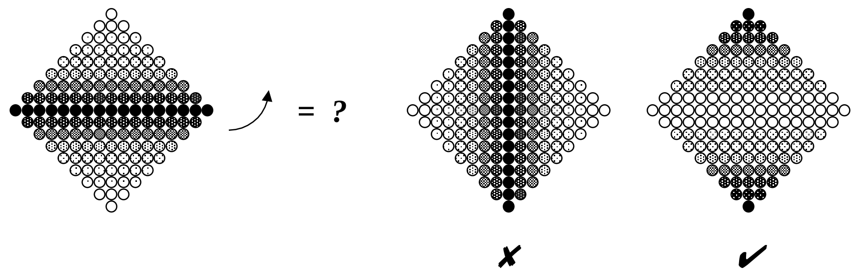


- Perspective distortion



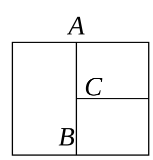
Problems with Interpolated Shading

- Scanline/orientation dependent
  - Creates temporal aliasing when used to render animation frames:

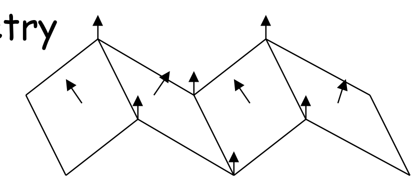


Problems with Interpolated Shading

- Shared vertices



- Unrepresentative vertex normals
  - Missed specular highlights
  - Missed geometry



Lighting, in practice

- Full lighting equation:

$$I = I_a k_a O_d + \sum_{1 \le 0 \le m} f_{att\,i} I_{p\,i} [\, k_d O_d (N \cdot L_i) + k_s (R_i \cdot V)^n]$$

- Lets ignore specular
- Each surface: O_d, k_a, k_d, $v_{i\ (i=0..n)}$, N
- Each light: $I_{a\ or\ d}$, f_{att} (c_1, c_2, c_3), P_L (position)

At a given point

- Start with ambient: $I_a k_a O_d$
  - R/G/B using I_{aR}, I_{aG}, I_{aB}, O_{dR}, O_{dG}, O_{dB}
- For each Light, compute: $f_{att} I_p\, k_d O_d (N \cdot L_i)$
  - Position (P_P), normal (N_P)
  - L vector
  - d_L
  - $f_{att} = 1/(c_1 + c_2 d_L + c_3 d_L^2)$
  - R/G/B using I_{pR}, I_{pG}, I_{pB}, O_{dR}, O_{dG}, O_{dB}

Light Intensity Values

- I_a, I_d
  - Represent intensity
  - Have R,G,B components
  - Do not need to fall in the 0..1 range!
    - Often need I_d>1
    - Final computed $I \leq 1$

Specular

- A light might have a diffuse and specular specification, say I_s
  - Allow slightly different colors, more control
    - Remember, it's a hack anyway!
- I_s would have RGB parts, as with I_a, I_d
- Illumination formula becomes

$$I = I_a k_a O_d + \sum_{1 \leq 0 \leq m} f_{att\,i} [I_{pd\,i} k_d O_d (N \cdot L_i) + I_{ps\,i} k_s (R_i \cdot V)^n]$$