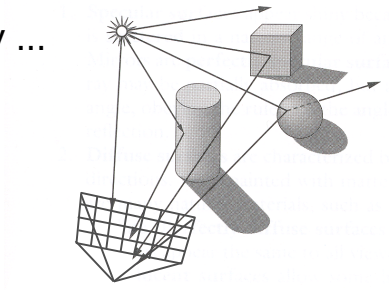
Illumination and Shading

Thomas Funkhouser
Princeton University
C0S 426, Fall 1999

Overview

- What we did last time ...
  - Emission at light sources
  - Reflectance at surfaces
- What we will cover today ...
  - Camera models
  - Illumination models
  - Shading algorithms



Angel Figure 6.2

Overview

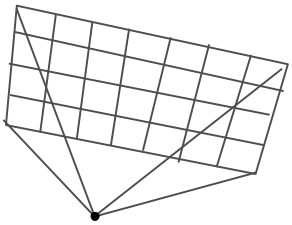
- Camera models
- Illumination models
- Shading algorithms

Overview

- **Camera models**
- Illumination models
- Shading algorithms

Camera Models

- Most common model is pin-hole camera
  - All captured light rays arrive along paths toward focal point without lens distortion (everything is in focus)
  - Sensor response proportional to radiance



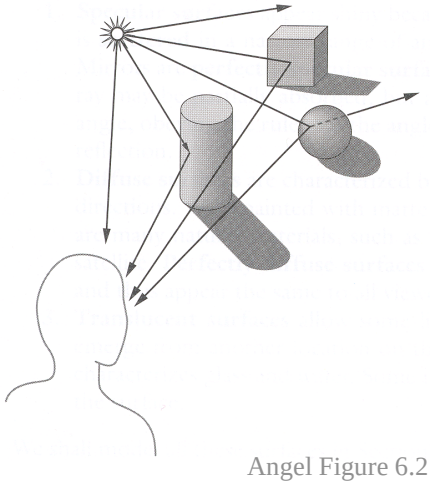
- Other models consider ...
  - Depth of field
  - Motion blur
  - Lens distortion

Overview

- Camera models
- **Illumination models**
- Shading algorithms

Illumination Models

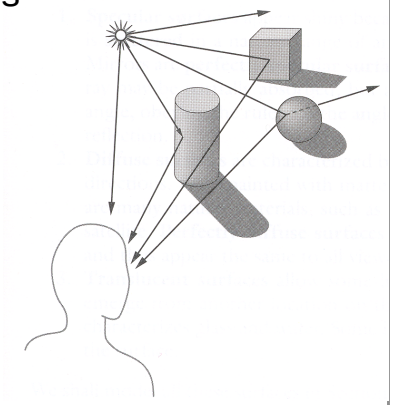
- How do we compute radiance for a sample ray?
  - Direct illumination
  - Global illumination



Angel Figure 6.2

Direct Illumination

- Ignore inter-object reflections
- Illumination is sum of ...
  - Emissions
  - Ambient reflections
  - Diffuse reflections
  - Specular reflections
  - Transmissions



$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T$$

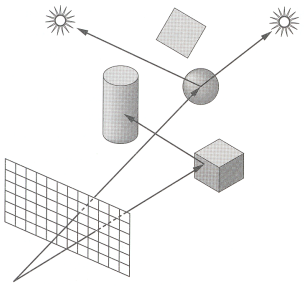
Angel Figure 6.2

Shadows and Transparency

Greg Larson

Shadows

- Shadow terms tell which light sources are blocked
  - Cast ray towards each light source L_i
  - $S_i = 0$ if ray is blocked, $S_i = 1$ otherwise



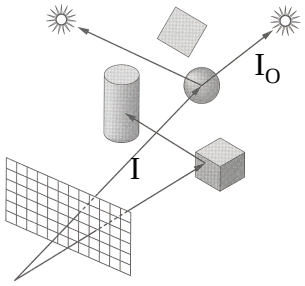
Shadow Term

$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T$$

Angel Figure 6.44

Transparency

- Opacity coefficient tells how much light is blocked
  - $K_O = 0$ if object is translusent, $K_O = 1$ if object is opaque
  - $0 < K_O < 1$if object is semi-translusent



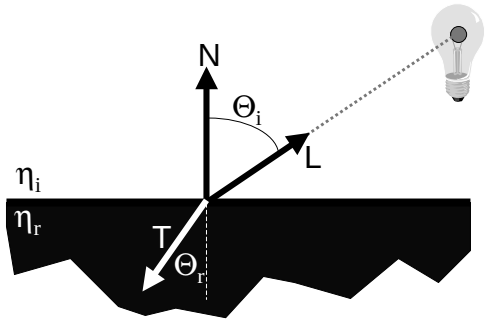
Opacity Coefficient

$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + (1 - K_O) I_O$$

Angel Figure 6.44

Transparency

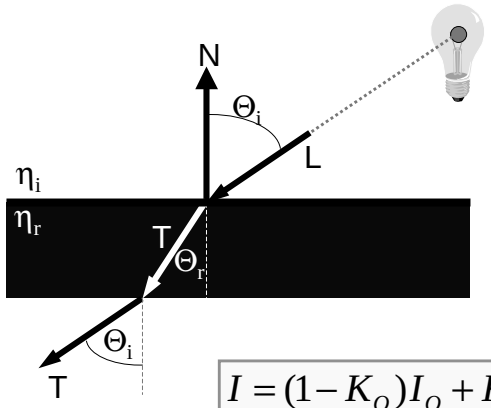
- Snell's law: $\eta_r \sin \Theta_r = \eta_i \sin \Theta_i$



$$T = (\frac{\eta_i}{\eta_r} \cos \Theta_i - \cos \Theta_r) N - \frac{\eta_i}{\eta_r} L$$

Transparency

- Usually ignore refraction
  - Assume light travels straight through surface



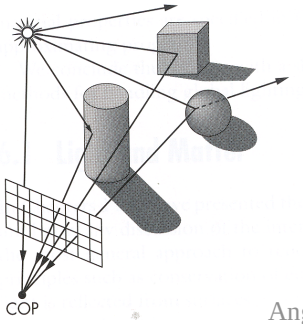
$$I = (1 - K_O) I_O + K_O I_R$$

Overview

- Camera models
- Illumination models
- **Shading algorithms**

Shading Algorithms

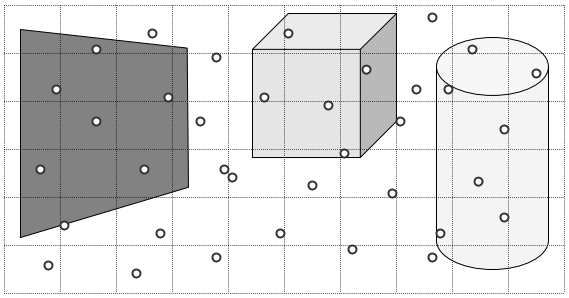
- How do we use illumination to make an image?
  - Each illumination calculation for a ray from the eyepoint through the view plane provides a radiance sample
    - » How do we choose where to place samples?
    - » How do we filter samples to reconstruct image?



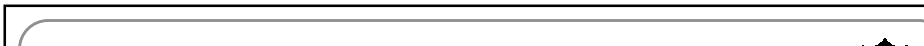
Angel Figure 6.34

Shading Algorithms

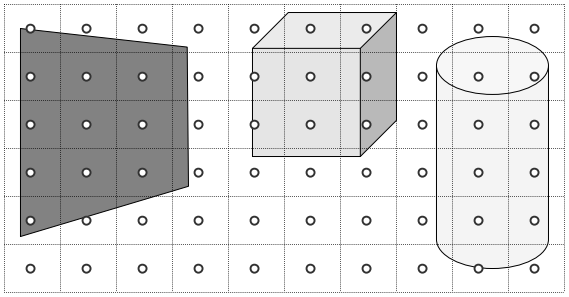
- Shading is a sampling and reconstruction problem



Ray Casting

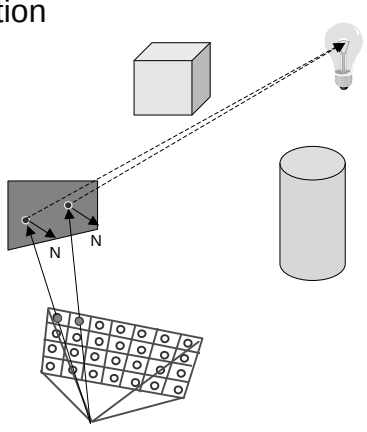


- Simplest shading approach is to perform independent lighting calculation for every pixel



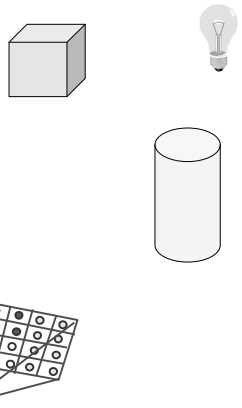
Ray Casting

- For each pixel …
  - Find first intersected by ray through pixel
  - Perform illumination calculation



Ray Casting

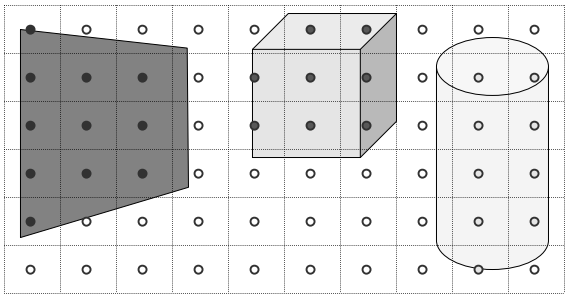
- For each pixel …
  - Find first intersected by ray through pixel
  - Perform illumination calculation



What is bad about this method?

Polygon Shading

- Can take advantage of spatial coherence



Polygon Shading Algorithms

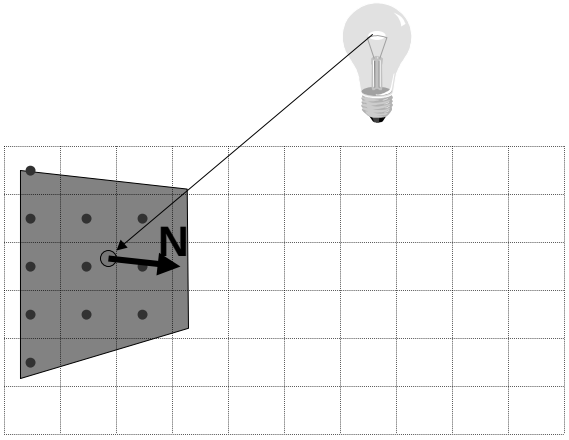
- Flat Shading
- Gouraud Shading
- Phong Shading

Polygon Shading Algorithms

- **Flat Shading**
- Gouraud Shading
- Phong Shading

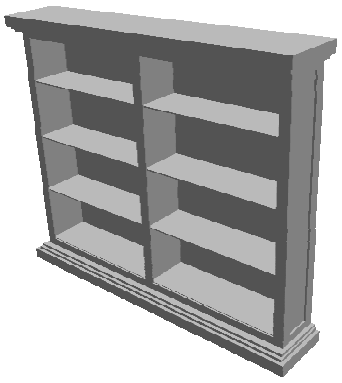
Flat Shading

- One illumination calculation per polygon
  - Assign all pixels inside each polygon the same color



Flat Shading

- This is a good approximation for faceted objects illuminated by directional light sources viewed from infinitely far away

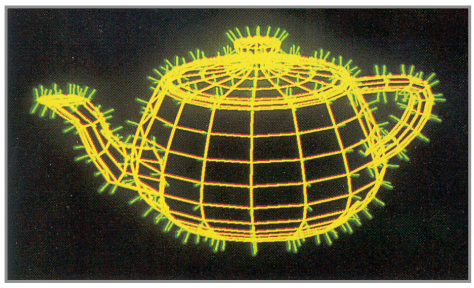


Polygon Shading Algorithms

- Flat Shading
- **Gouraud Shading**
- Phong Shading

Gouraud Shading

- Smooth shading over adjacent polygons
  - Curved surfaces
  - Illumination highlights
  - Soft shadows

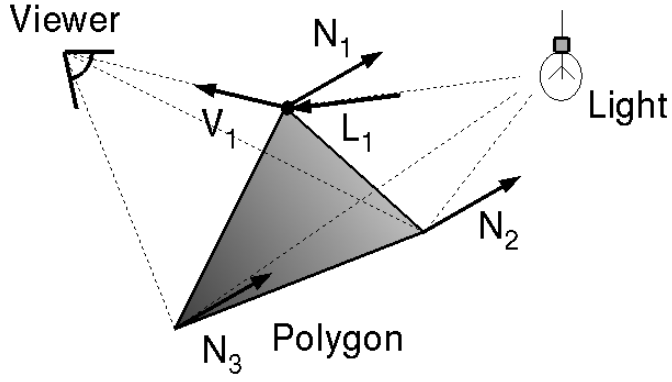


Mesh with shared normals at vertices

Watt Plate 7

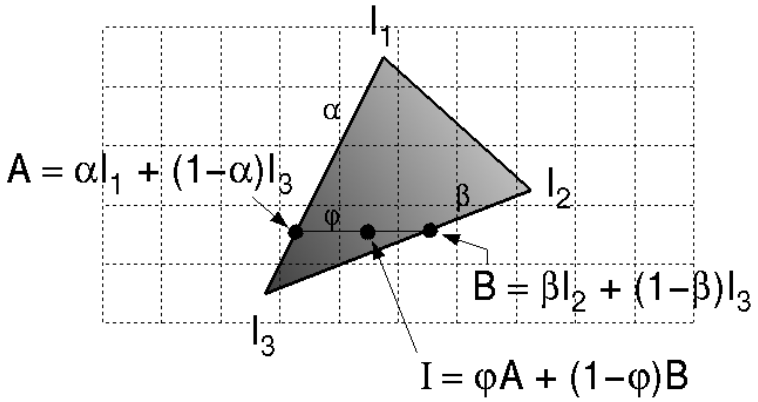
Gouraud Shading

- One lighting calculation per vertex
  - Assign pixels inside polygon by interpolating colors computed at vertices



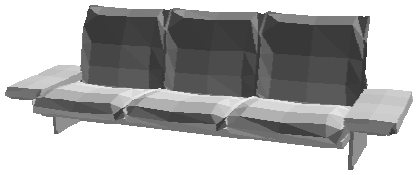
Gouraud Shading

- Bilinearly interpolate colors at vertices down and across scan lines

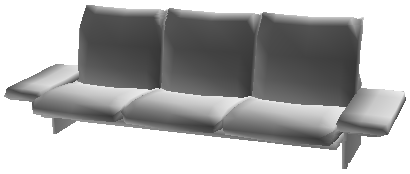


Gouraud Shading

- Produces smoothly shaded polygonal mesh
  - Need a fine mesh to capture subtle lighting effects



Flat Shading



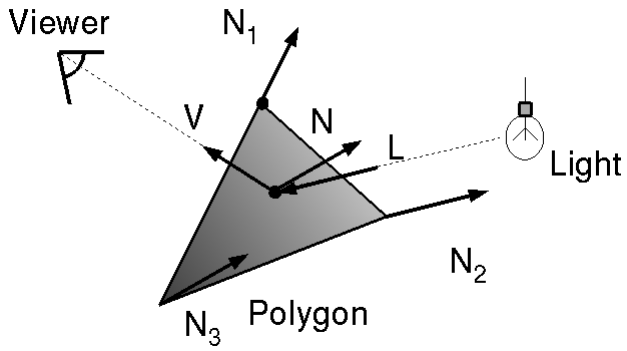
Gouraud Shading

Polygon Shading Algorithms

- Flat Shading
- Gouraud Shading
- **Phong Shading**

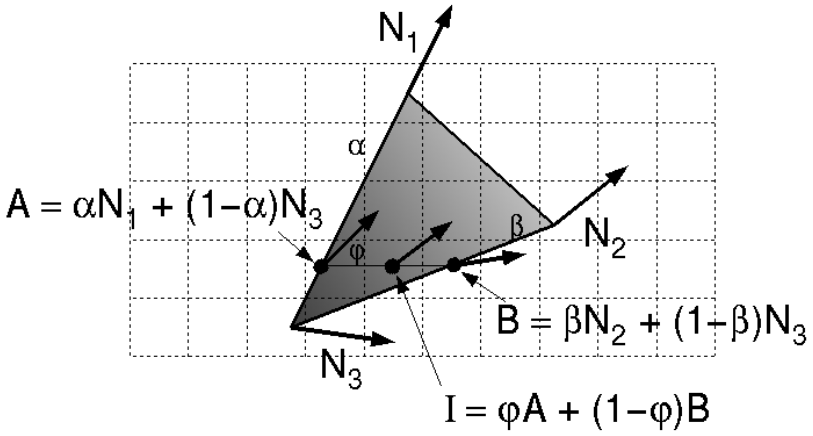
Phong Shading

- One lighting calculation per pixel
  - Approximate surface normals for points inside polygons by bilinear interpolation of normals from vertices

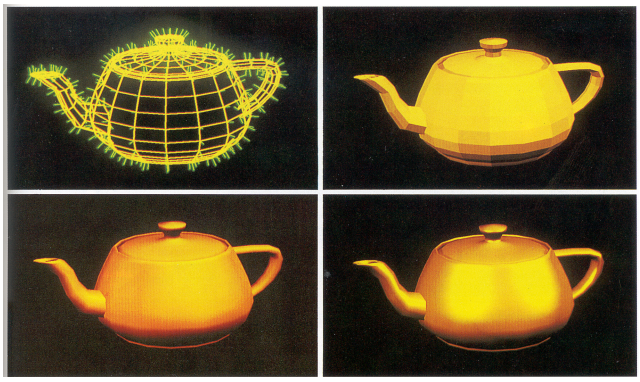


Phong Shading

- Bilinearly interpolate surface normals at vertices down and across scan lines

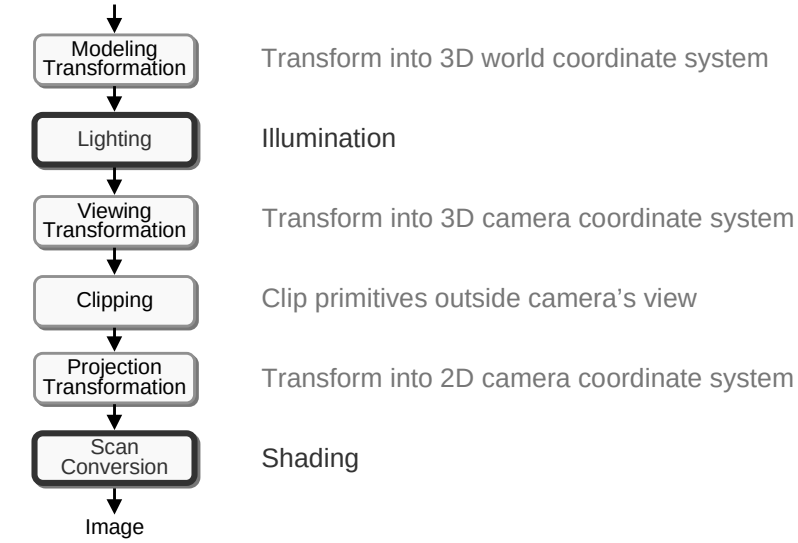


Polygon Shading Algorithms



Watt Plate 7

3D Rendering Pipeline



Shading Issues

- Problems with interpolated shading:
  - Polygonal silhouettes
  - Perspective distortion
  - Orientation dependence
  - Problems at T-vertices
  - Problems computing shared vertex normals

Summary

- Shading
  - Computing illumination for each pixel
- Polygon Shading Algorithms
  - Flat
  - Gouraud
  - Phong
  - Ray casting

Less expensive

More accurate

- Key ideas:
  - Sampling and reconstruction
  - Coherence