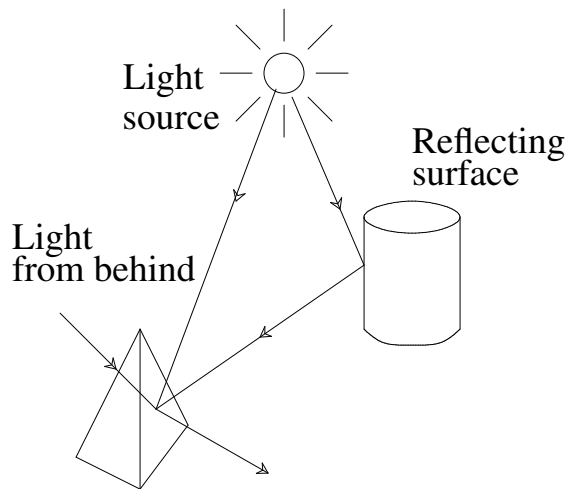


9. Illumination and Shading

Approaches for visual realism:

1. Remove hidden surfaces
2. Shade the visible surfaces and reproduce shadows
3. Reproduce surface properties:
 - texture
 - degree of transparency,
 - roughness, ... etc

9.1 How to shade a visible surface? depends on



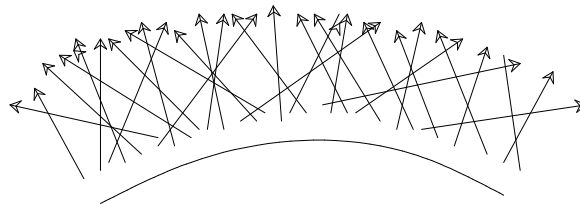
position
orientation
characteristics of the sur-
faces
light sources

Light sources:

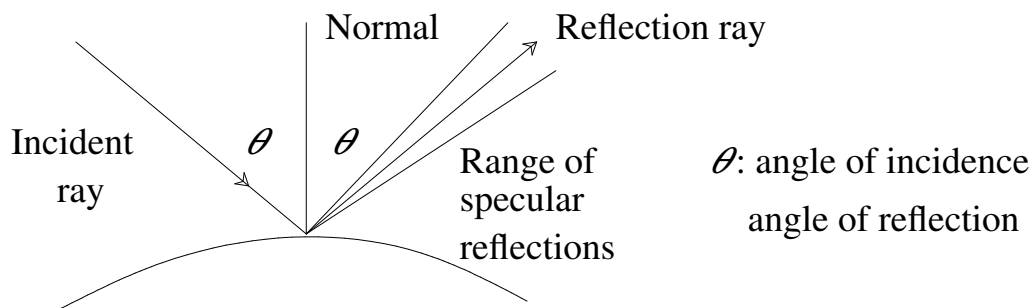
- Light-emitting sources -- light bulb, sun, ...
- Light-reflecting sources -- illuminated surfaces of other objects

Reflection Type:

- **diffuse reflection:** reflections scattered in all directions



- **specular reflection:** reflections point in a range of direction only



Ambient light (background light)

- results of multiple reflections from nearby objects
- can be considered to be of uniform intensity I_a in all directions
- use a general level of brightness to model it to produce a uniform illumination
- reflected in all directions

Shade at a point of a surface

= intensity of light reflection from that point of the surface

= diffuse reflection of ambient light

+ diffuse and specular reflection of one or more point light sources

+ diffuse and specular reflection of transparent effect

How to estimate the intensities of these?

Lambert's (cosine) Law:

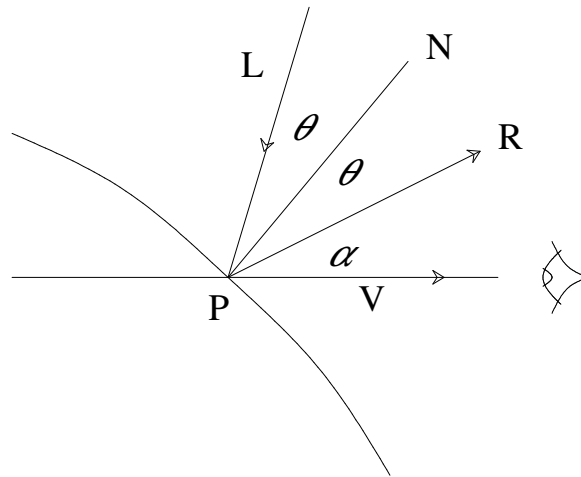
$$\text{Diffuse reflection} = I_p \cdot k_d \cdot \cos \theta$$

I_p : intensity of light source

k_d : coefficient of reflection

$k_d \rightarrow 0$ surface is highly absorptive

$k_d \rightarrow 1$ surface is highly reflective



Phong Model:

$$\text{Specular reflection} = I_p \cdot W(\theta) \cdot \cos^n \alpha$$

$W(\theta)$: specular reflection coefficient

depends on the material

Specular reflection models must produce the highest intensity in the direction of R , with the intensity decreasing rapidly as the viewing angle α increases.

Guidelines:

Shining surface ($n \geq 200$)

Rough surface ($1 \leq n \leq 200$)

For glass

$$W(\theta) \rightarrow 1 \quad \text{if } \theta \rightarrow 90^\circ$$

$$W(\theta) \rightarrow 0 \quad \text{if } \theta \rightarrow 0^\circ$$

Other materials have constant $W(\theta)$

Effect of Transparency:

$$\text{Transparency effect} = I_{pt} \cdot k_{pt}$$

I_{pt} : intensity arriving from behind

(= I_a if no light source behind object)

k_{pt} : transmission coefficient

depends on the material

Hence, shade at point P (viewpoint at infinity)

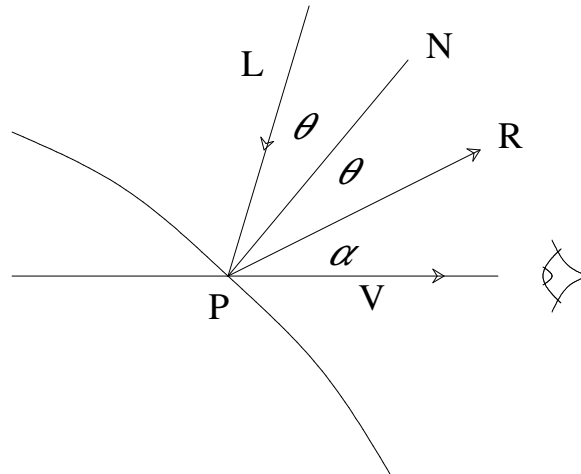
$$I = I_a k_d + I_p [k_d \cos \theta + W(\theta) \cos^n \alpha] + I_{pt} k_{pt}$$

or (viewpoint not at infinity)

$$I = I_a k_d + I_p [k_d \cos \theta + W(\theta) \cos^n \alpha] / (r + r_0) + I_{pt} k_{pt}$$

r : distance from viewpoint to P ; r_0 : constant

How to compute R ?



In the above formula, if L , N , R , and V are normalized, we have

$$\cos \theta = L \cdot N$$

$$\cos \alpha = R \cdot V$$

But $R = ?$

$$R = 2(N \cdot L)N - L$$

9.2 Polygon-Mesh Rendering Methods

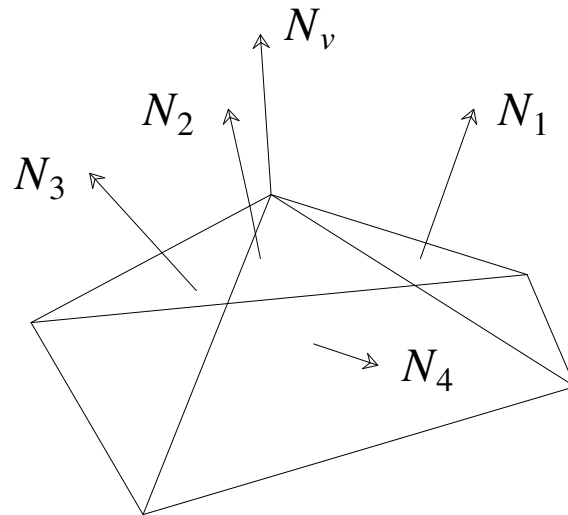
Rendering of graphics objects approximated by polygonal meshes

Constant-Intensity Shading (Flat Shading)

- a single intensity is computed for each polygon
- assumption: light source and viewer are both sufficiently far from the object
- works only if the polygons are small

Intensity Interpolation Shading (Gouraud Shading)

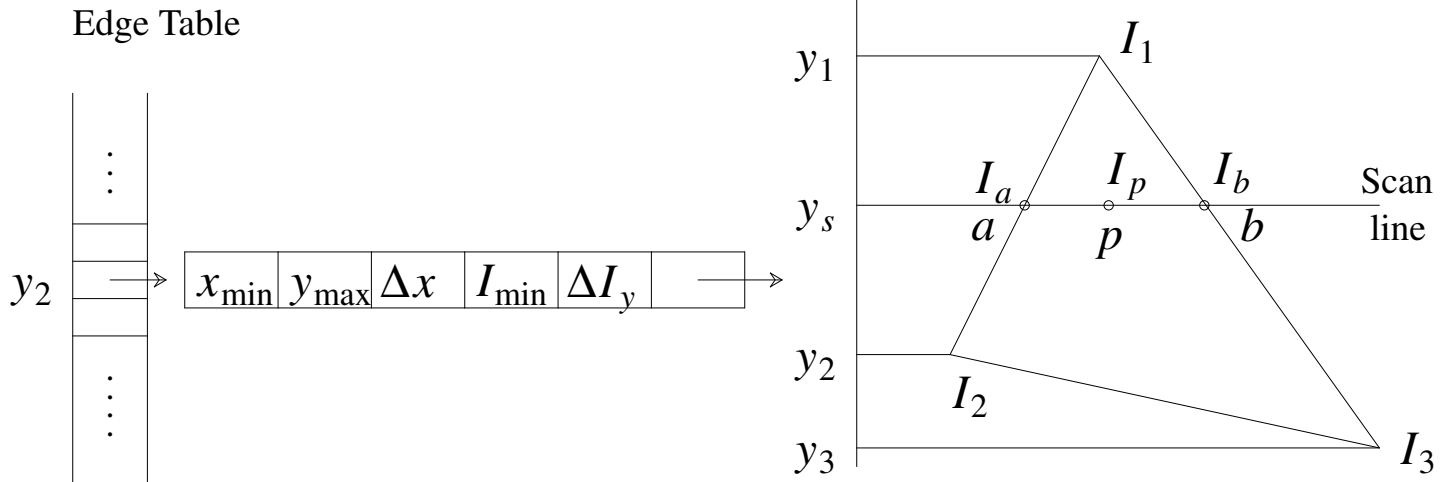
- Eliminate intensity discontinuities
- Basic idea:
 1. compute polygon (unit) normals
 2. compute the average unit normal vector at each polygon vertex



$$N_v = (N_1 + N_2 + N_3 + N_4)/4$$

3. compute vertex intensities
 4. shade polygon by linear interpolation of vertex intensities along each edge and then between edges along each scan line
- May cause Mach band effect
(bright or dark intensity streaks, occurs when the light intensity on a surface changes abruptly)

Note: interpolation along edges can be integrated with the scan-line hidden surface elimination algorithm

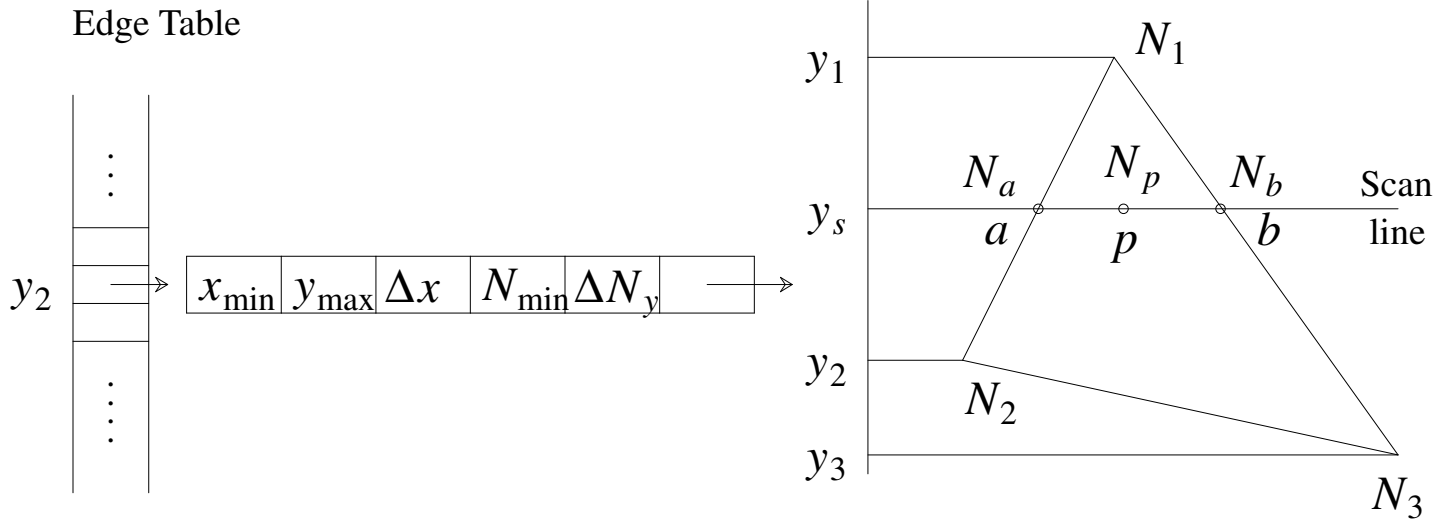


- I_{\min} has to be updated for each new scan line
- If the span (a, b) is visible, it is filled by interpolating the intensity values of I_a and I_b
- Need to compute ΔI_x
- The intensity I_p at p is $I_{p-1} + \Delta I_x$

Normal-Vector Interpolation Shading (Phong Shading)

- Interpolates the normal vector N rather than intensity
- Basic idea:
 1. compute polygon normals
 2. compute vertex normals
 3. compute normals of start and end points of a visible span using linear interpolation
 4. compute normal at each pixel along a scan line, then compute intensity using a desired illumination model

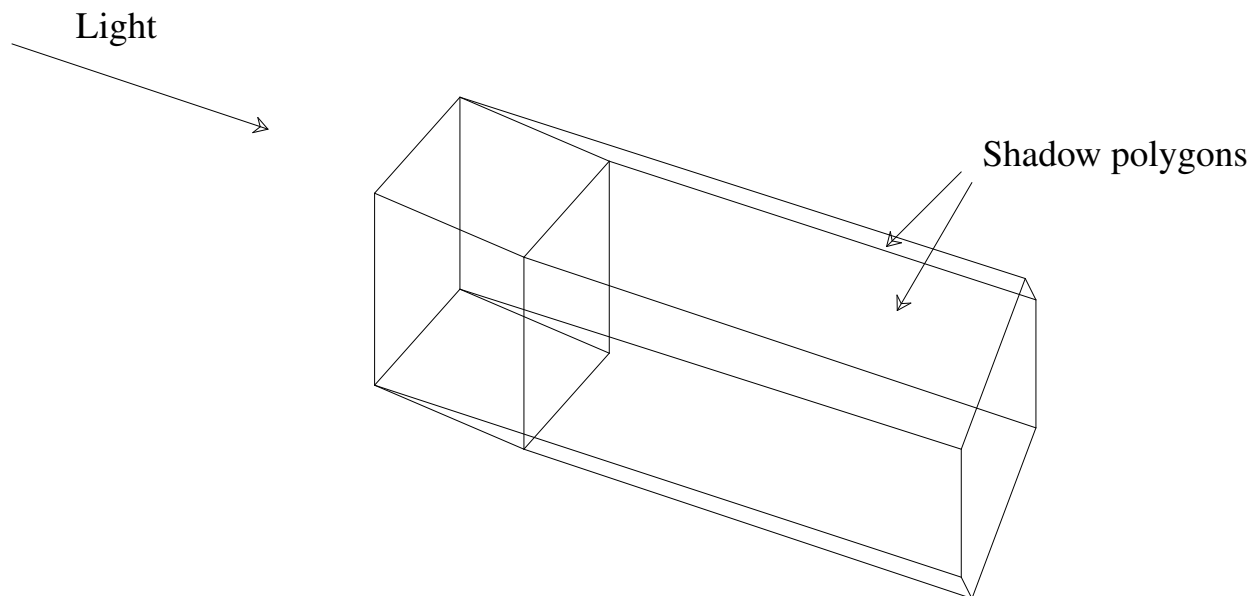
- Can also be integrated with scan-line method



- Highlights are more faithfully reproduced
- Reduces Mach band problems

9.3 Shadows

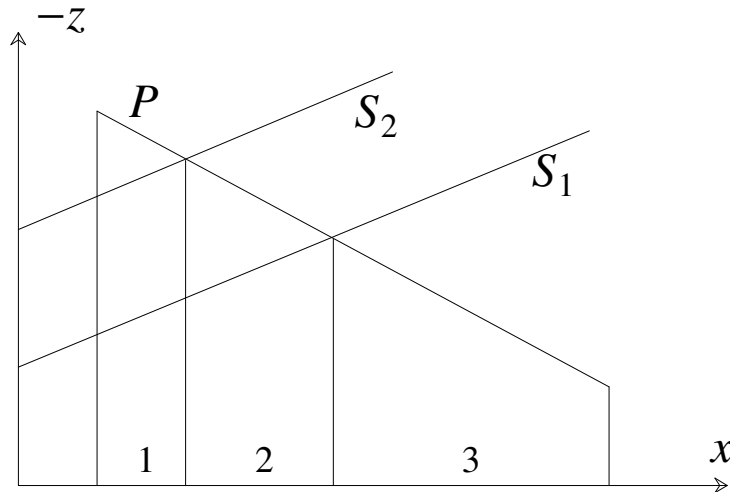
- Determines which faces or parts of faces are visible when viewed from the light sources
- Surfaces visible from both the viewpoint and the light source are not in shadow
- Surfaces visible from the viewpoint but not from the light source are in shadow
- Basic idea:
 1. Each edge of an object that is an outline of the object when viewed from the light source is extended to form a shadow polygon



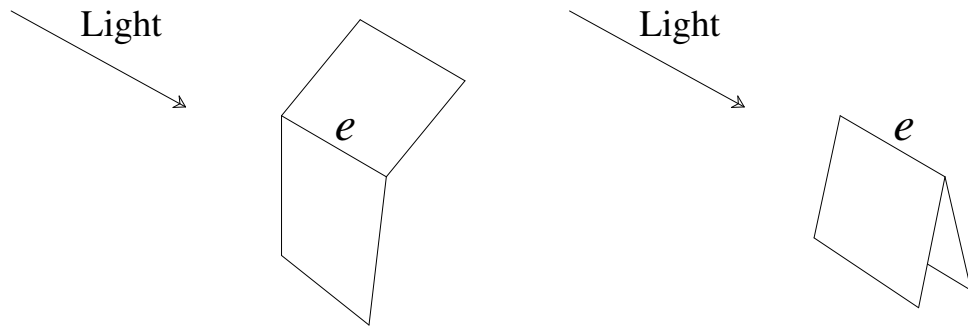
If several light sources are being simulated then shadow polygons generated by different light sources will be tagged differently

Shadow polygons are passed to the scan conversion process along with polygons in the scene

2. As the scan conversion process proceeds, if an even number of shadow polygons with the same tags is encountered, the face is not shadowed by the object and the light source given in the tags. A set of tags encountered an odd times indicates that the face is shadowed by the light source in the tags.



How to determine if an edge is an outline when viewed from the light source?

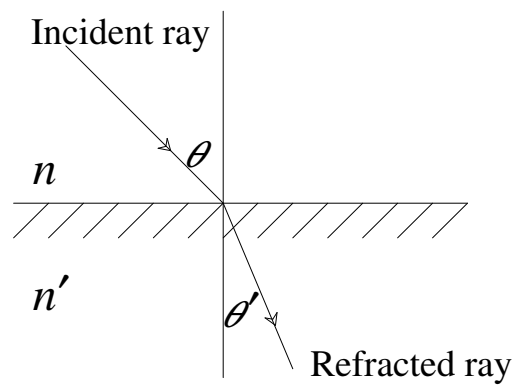
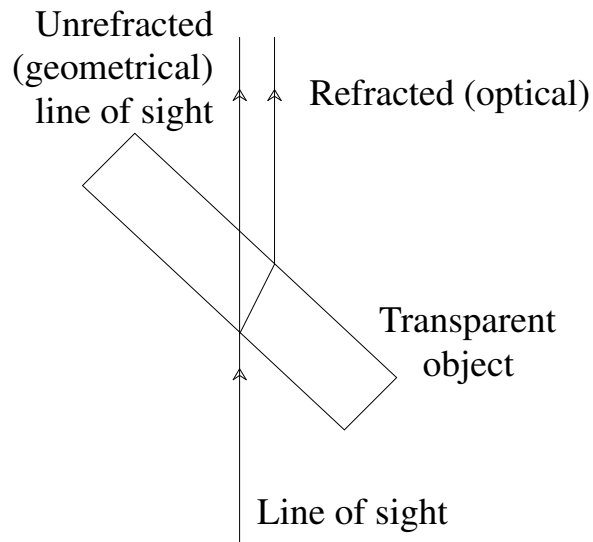


Draw a plane through e that is parallel to the DOP

If all the other vertices of the object are on the same side of this plane then e is an edge of the outline

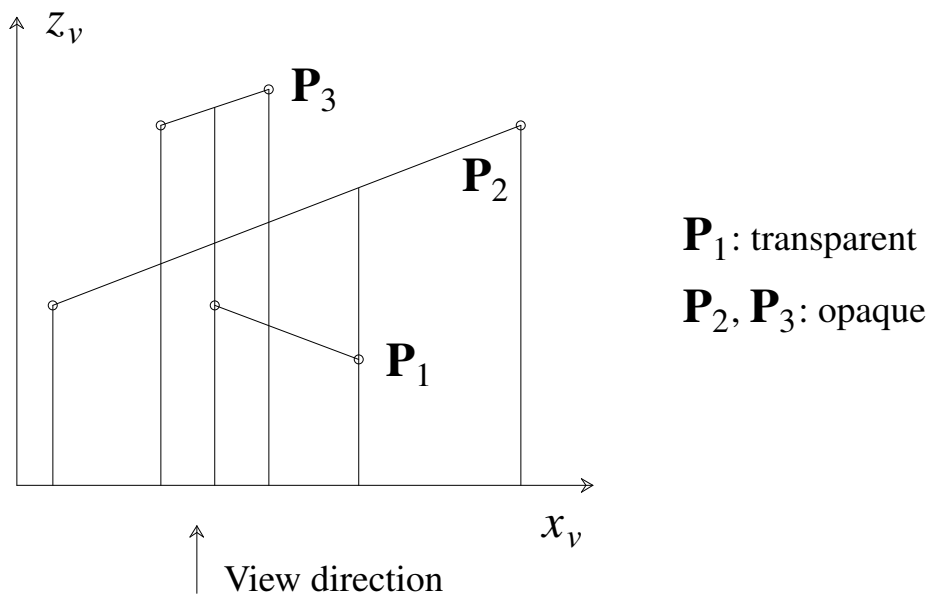
9.4 Light-Transmitting Surfaces

- specular transmittance: refracted & unrefracted
- diffuse transmittance: not easy to model



Law of refraction
(n, n' : indices of refraction)

- Using scan-line algorithm (ignoring refraction)
 1. Within a **span**, sort the segments by **depth**
 2. Find the **opaque** polygon closest to the observer and determine its **shade**
 3. This **shade** is then modified by any **transparent segments** that lie between the observer and the **opaque surface**



$$I = k \cdot I_1 + (1 - k) \cdot I_2$$

$0 \leq k \leq 1$: degree of transparency of \mathbf{P}_1