COMPUTER GRAPHICS
RCS-603
UNIT-V

Presented By :

Dr. Vinod Jain (Associate Professor, GLBITM)

GL BAJAJ
Institute of Technology & Management

[Approved by AICTE, Govt. of India & Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow, U.P. India]

Department Of Computer Science & Engineering

Unit- V – 2 sections

1. **V Hidden Lines and Surfaces:** Back Face Detection algorithm, Depth buffer method, A- buffer method, Scan line method
2. **Basic illumination models–** Ambient light, Diffuse reflection, Specular reflection and Phong model, Combined approach, Warn model, Intensity Attenuation, Color consideration, Transparency and Shadows.(, A- buffer method, Scan line method

V Hidden Lines and Surfaces:

1. Back Face Detection algorithm
2. Depth buffer method
3. A- buffer method
4. Scan line method

Hidden-surface elimination
Or Visible surface detection

- When we view a picture containing non-transparent objects and surfaces, then **we cannot see those objects from view which are behind other objects.**
- We must remove hidden surfaces to get a realistic screen image.
- The identification and removal of these surfaces is called **Hidden-surface elimination** or We need to identify those parts of a screen that are visible from a chosen viewing position. **(Visible surface detection)**

Hidden-surface elimination
Or Visible surface detection

There are two methods of it

1. Object-space Methods
2. Image-space Methods

Hidden-surface elimination
Or Visible surface detection

Object-space Methods

An object-space **method compare objects and parts of objects** to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.

Line display algorithms generally use object space methods to identify visible lines in wireframe displays.

Image-space Methods

- In an image space algorithm, **visibility is decided point by point at each pixel position** on the projection plane.

Back Face Detection algorithm (Object Space Method)

- A fast and simple **object-space method** for identifying the back faces of a polyhedron is based on the "inside-outside" tests.
- **A point (x, y, z) is "inside" a polygon surface with plane parameters A, B, C, and D if**
  - $Ax + By + Cz + D < 0$
- Where coefficients A, B, C, and *D* are constants describing the spatial properties of the plane.

Back Face Detection algorithm

- We can simplify this test by considering the normal vector **N** to a polygon surface, which has Cartesian components (A, B, C).
- In general, if V is a vector in the viewing direction from the eye (or "camera") position, then this polygon is a back face

if **V.N > 0**

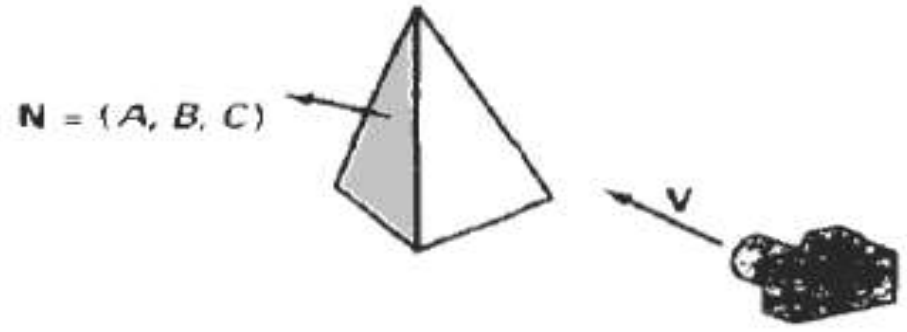
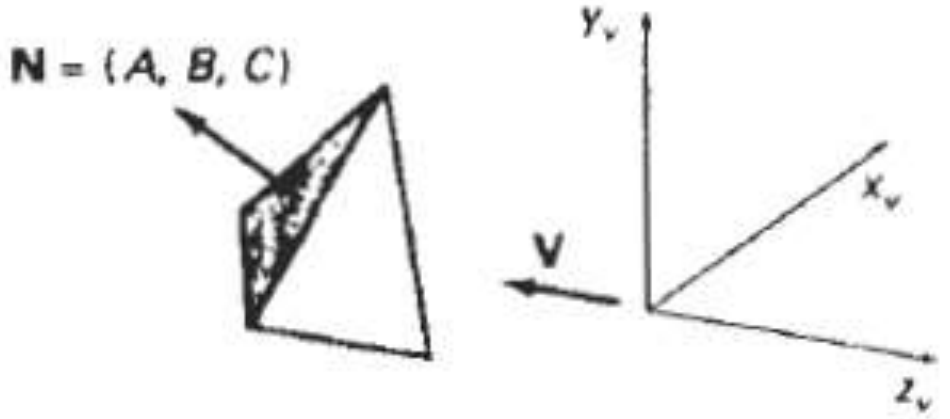


Figure 13-1
Vector **V** in the viewing direction and a back-face normal vector **N** of a polyhedron

Back Face Detection algorithm

- Furthermore, if object descriptions are converted to projection coordinates and your viewing direction is parallel to the viewing z-axis, then −
- **$V = (0, 0, V_z)$ and $V.N = V_zC$**
- So that we only need to consider the sign of C the component of the normal vector **N**.



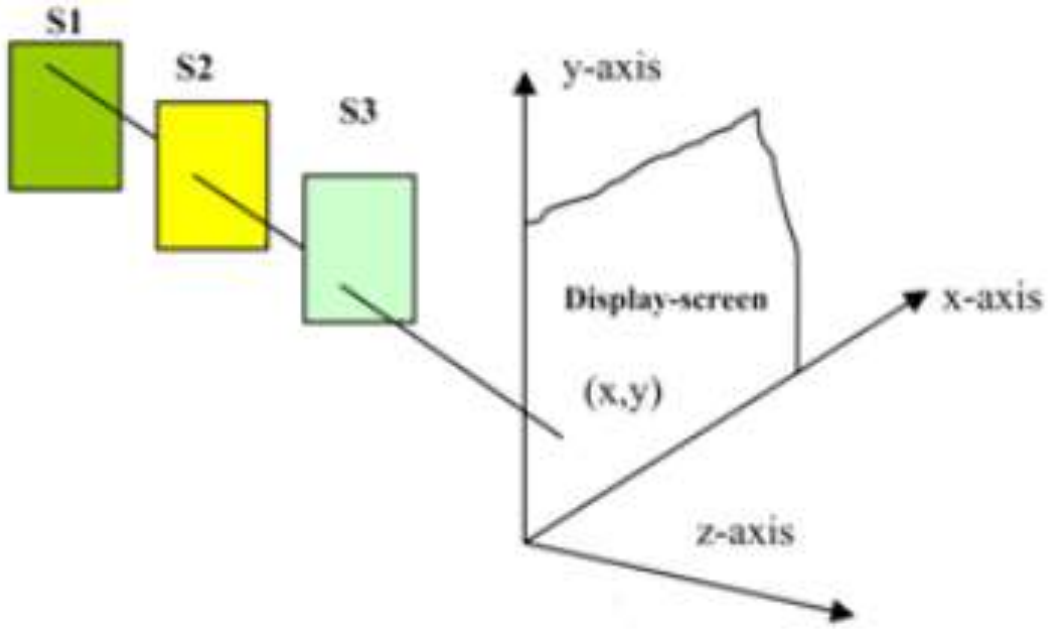
Back Face Detection algorithm

- In a right-handed viewing system with viewing direction along the negative Z_V axis, the polygon is a back face if $C < 0$.
- Thus, in general, we can label any polygon as a back face if its normal vector has a z component value −

$$C <= 0$$

Depth buffer method (Image space method)

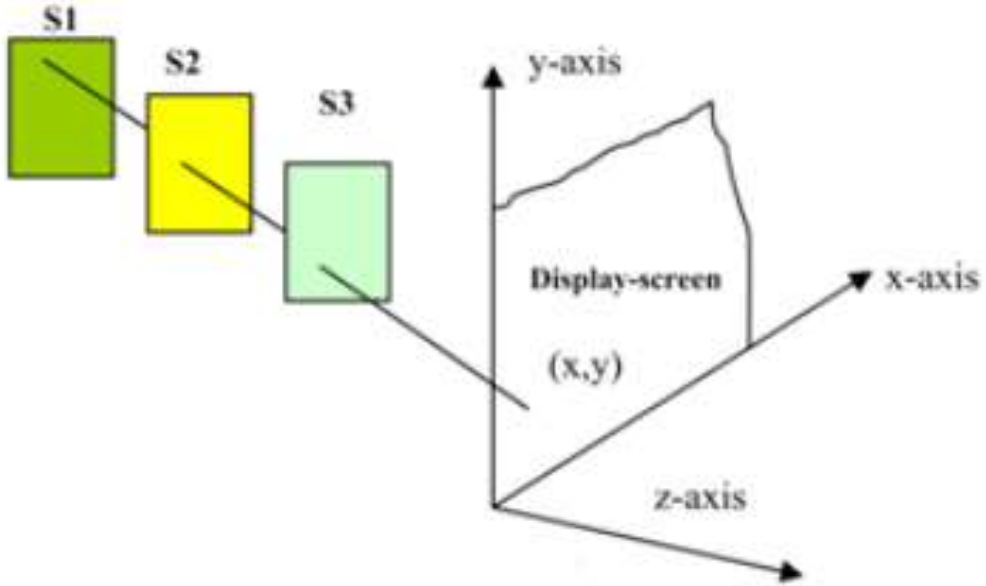
- It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface.
- **The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.**



Depth buffer method

- Two buffers named **frame buffer** and **depth buffer,** are used.
- The z-coordinates are usually normalized to the range [0, 1].
- **Depth buffer** is used to store depth values for (x, y) position, as surfaces are processed (0 ≤ depth ≤ 1).
- The **frame buffer** is used to store the intensity value of color value at each position (x, y).

Depth buffer Algorithm



Step-1 – Set the buffer values –

Depthbuffer (x, y) = 0

Framebuffer (x, y) = background color

Step-2 – Process each polygon (One at a tıme)

For each projected (x, y) pixel position of a polygon, calculate depth z.

If z > depthbuffer (x, y) then Compute surface color,

and set depthbuffer (x, y) = z,

framebuffer (x, y) = surfacecolor (x, y)

Depth buffer Algorithm

Using planar equation of a surface

$Ax + By + Cz + D = 0$

Depth values for a surface position (x, y) are calculated from the plane equation for each surface:

$$z = \frac{-Ax - By - D}{C} \qquad (13\text{-}4)$$

Depth buffer method

Advantages

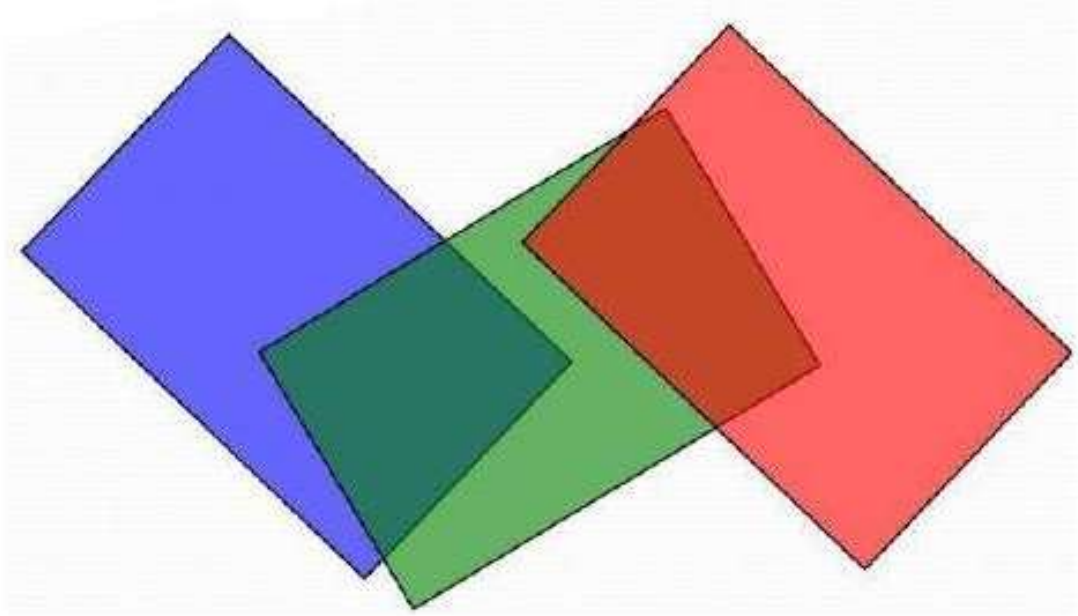
1. It is easy to implement.
2. It reduces the speed problem if implemented in hardware.
3. It processes one object at a time.

Disadvantages

1. It requires large memory.
2. It is time consuming process.

A- buffer method

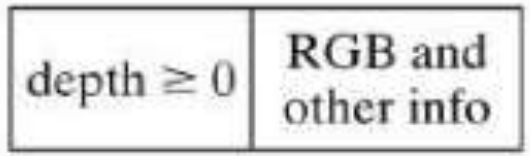
- The A-buffer expands on the depth buffer method to allow transparencies.
- The key data structure in the A-buffer is the accumulation buffer.



A- buffer method

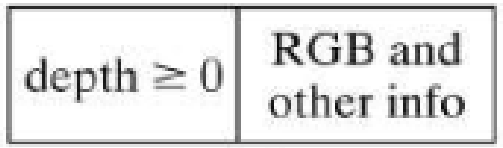
Each position in the A-buffer has two fields –

- **Depth field** – It stores a positive or negative real number
- **Intensity field** – It stores surface-intensity information or a pointer value

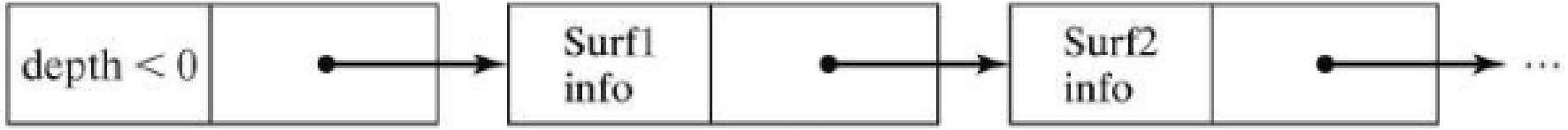


(a)

A- buffer method



(a)



(b)

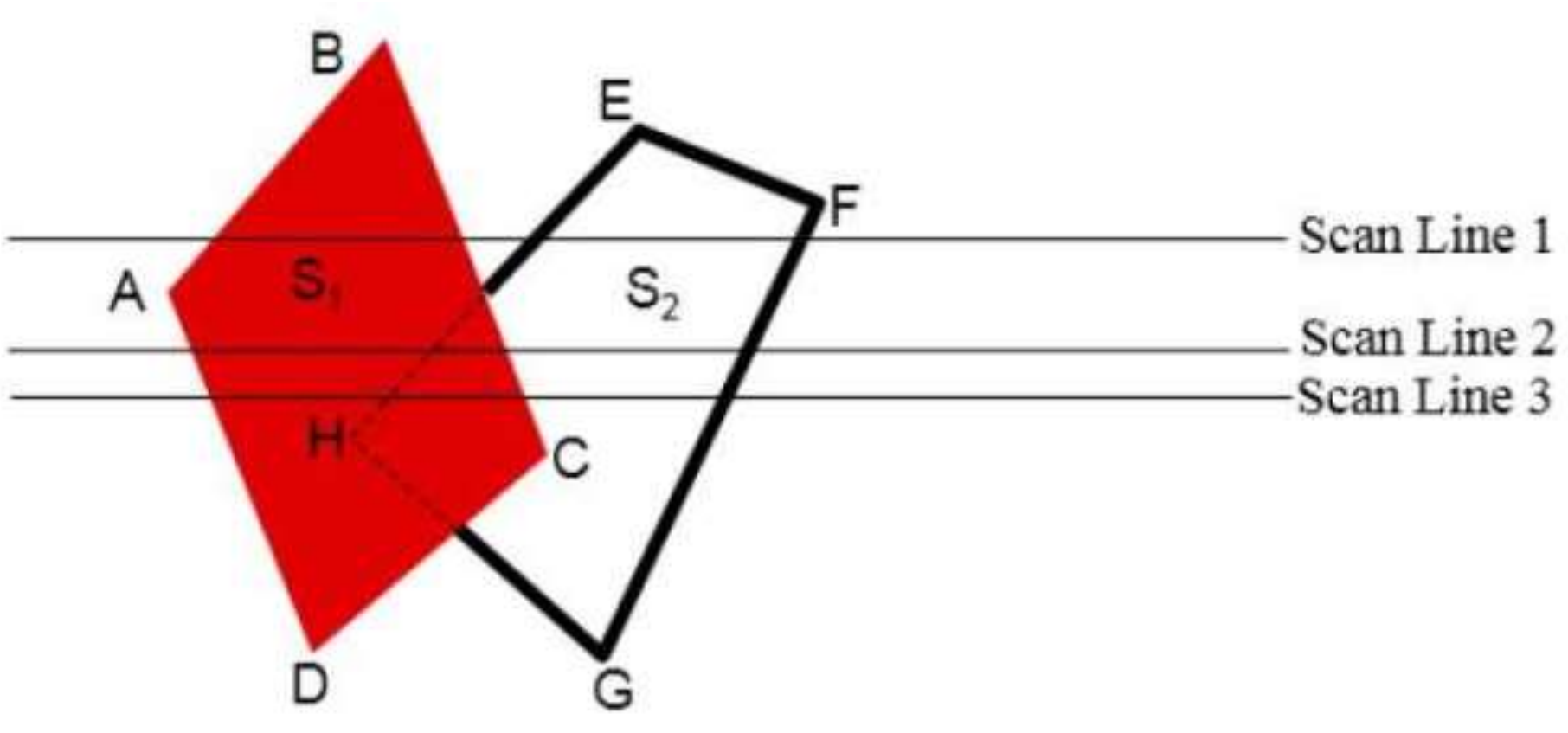
If depth >= 0, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

If depth < 0, it indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data. The surface buffer in the A-buffer includes –

A- buffer method

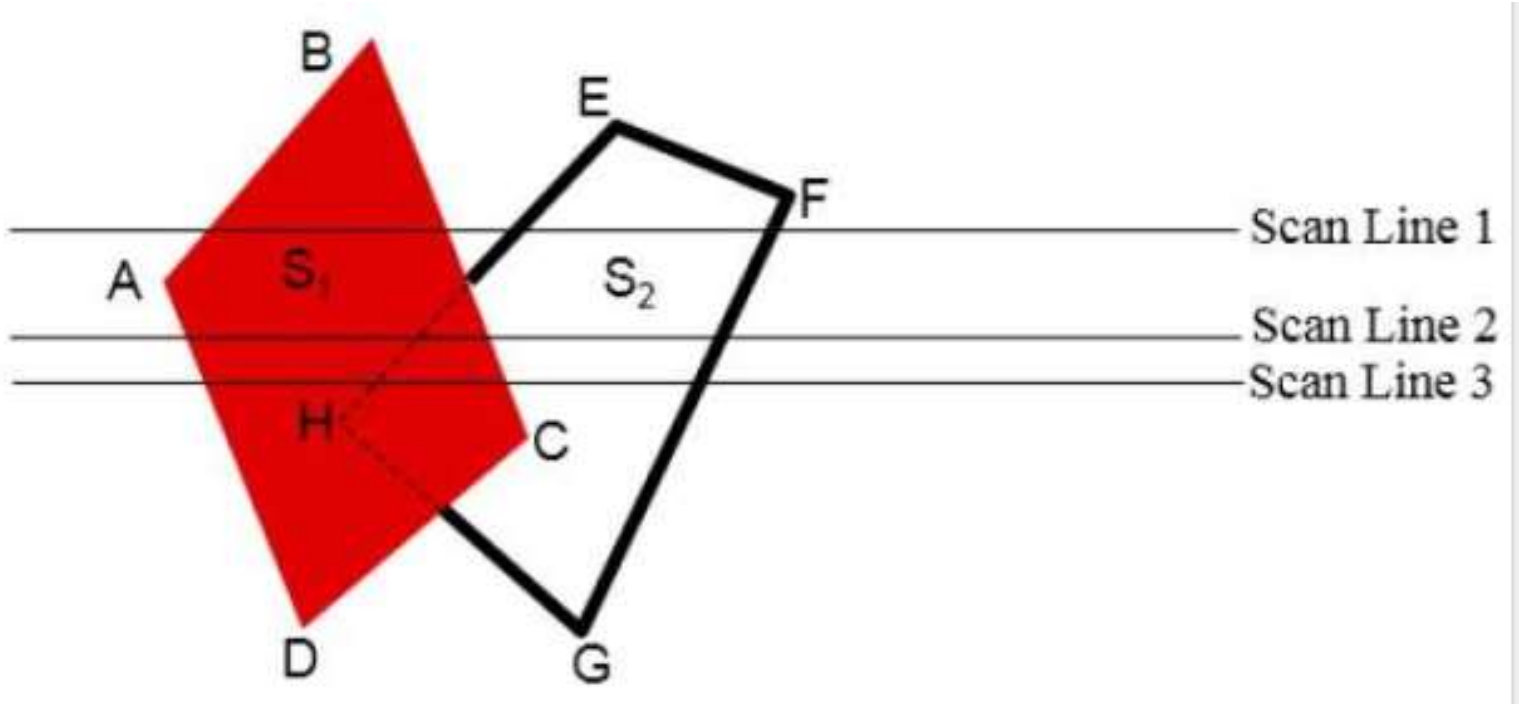
- The surface buffer in the A-buffer includes −
  1. RGB intensity components
  2. Opacity Parameter
  3. Depth
  4. Percent of area coverage
  5. Surface identifier
- The algorithm proceeds just like the depth buffer algorithm.
- The depth and opacity values are used to determine the final color of a pixel.

Scan line method



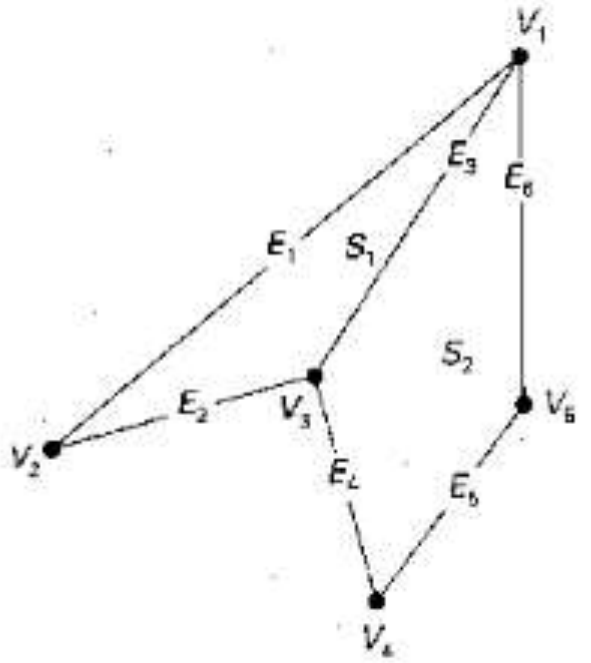
Scan line method

- It is an image-space method to identify visible surface.
- This method has a depth information for only single scan-line.



Scan line method

- Two important tables, **edge table** and **polygon table,** are maintained for this.

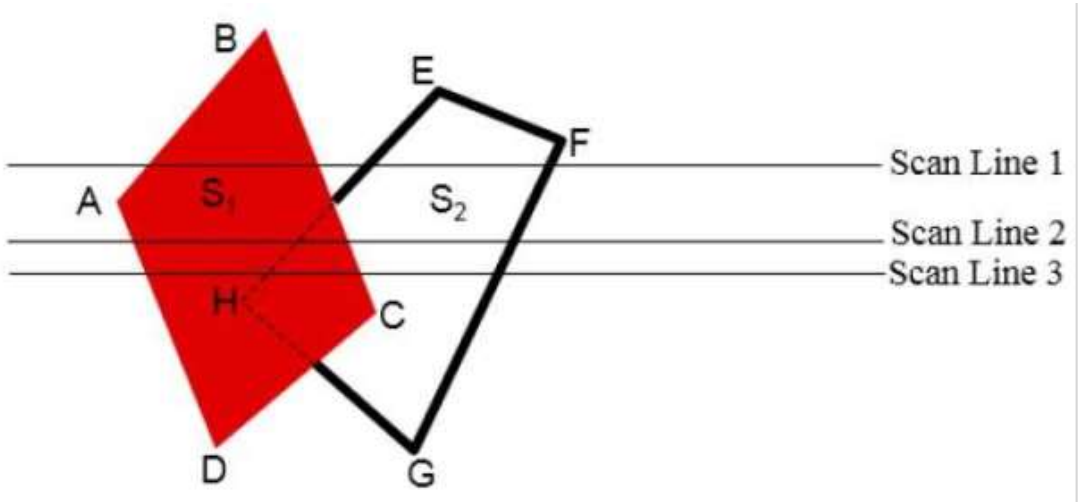


VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

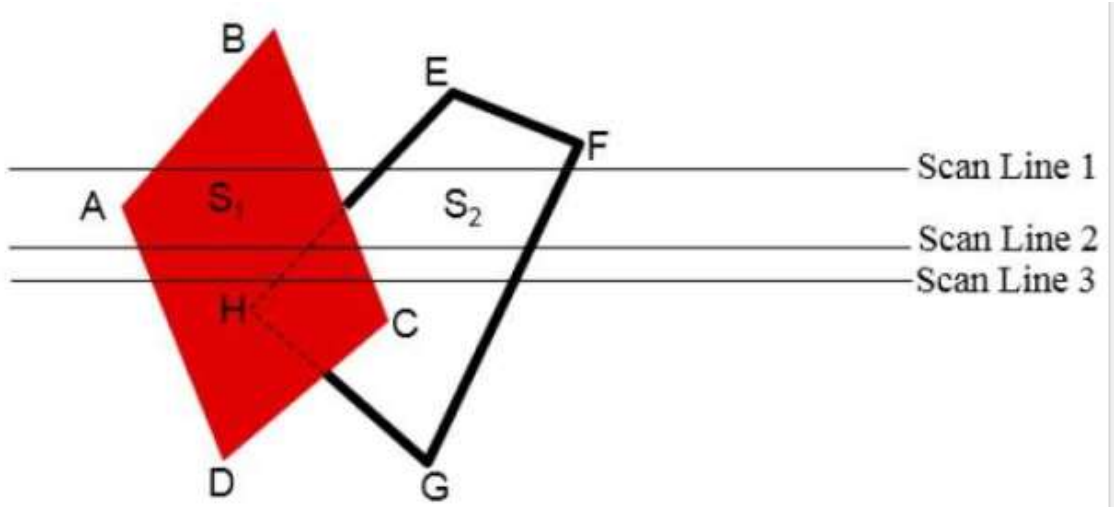
POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Scan line method



- To facilitate the search for surfaces crossing a given scan-line, an **active list of edges is formed.**
- The **active list stores only those edges that cross the scan-line** in order of increasing x.
- Also **a flag is set** for each surface to indicate whether a position along a scan-line is either inside or outside the surface.

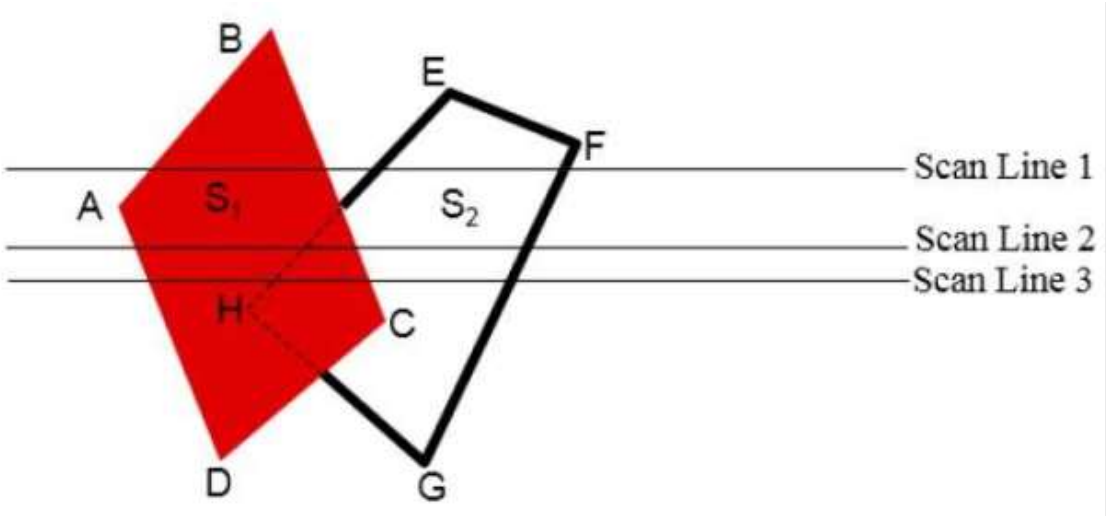
Scan line method



- Pixel positions across each scan-line are processed from left to right.
- At the left intersection with a surface, the surface flag is turned on and at the right, the flag is turned off.
- **You only need to perform depth calculations when multiple surfaces have their flags turned on** at a certain scan-line position.

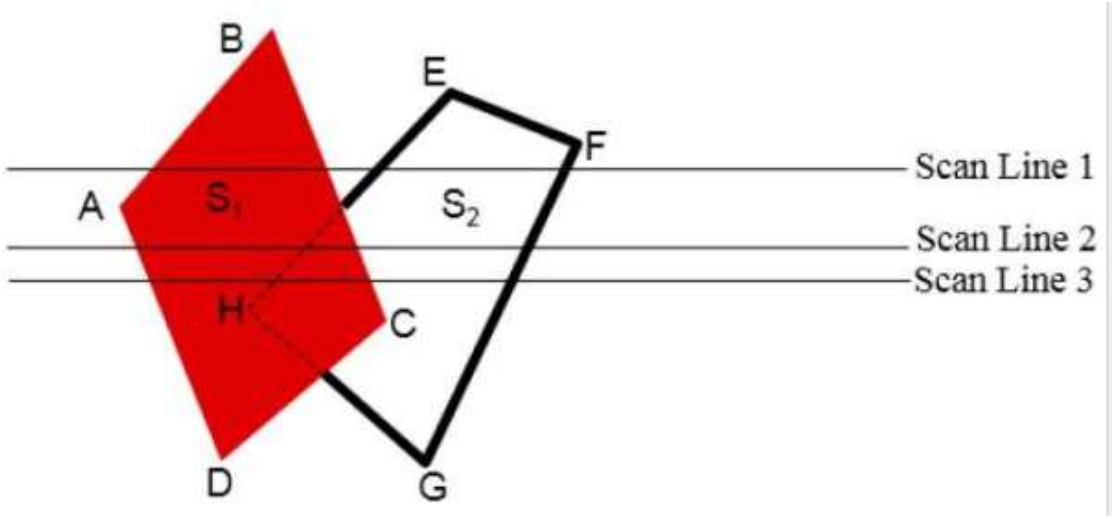
Scan line method

- The active list for scan line 1 contains information from the edge table for edges ***AB,*** BC, *EH,* and *FG.* For positions along this scan line between edges ***AB*** and BC, only the flag for surface **S1** is on.
- Therefore no depth calculations are necessary, and intensity information for surface S1 is entered from the polygon table into the refresh buffer.



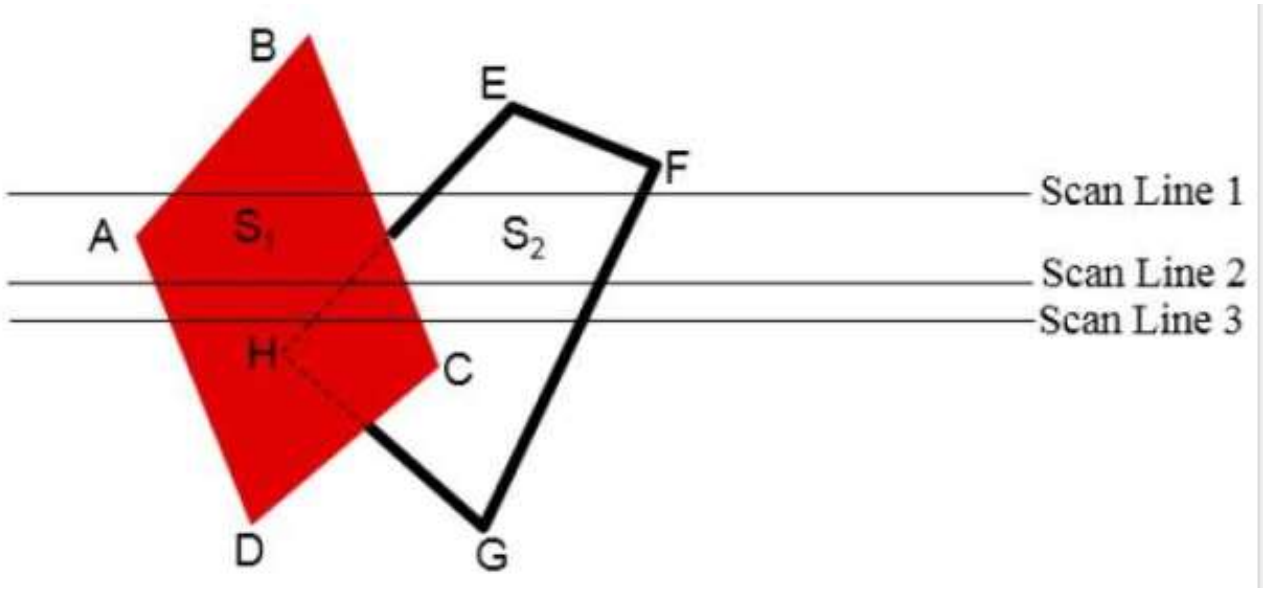
Scan line method

- Similarly, between edges EH and FG, only the flag for surface S2 is on.
- NO other positions along scan line **1** intersect surfaces, so the intensity values in the other areas are set to the background intensity.



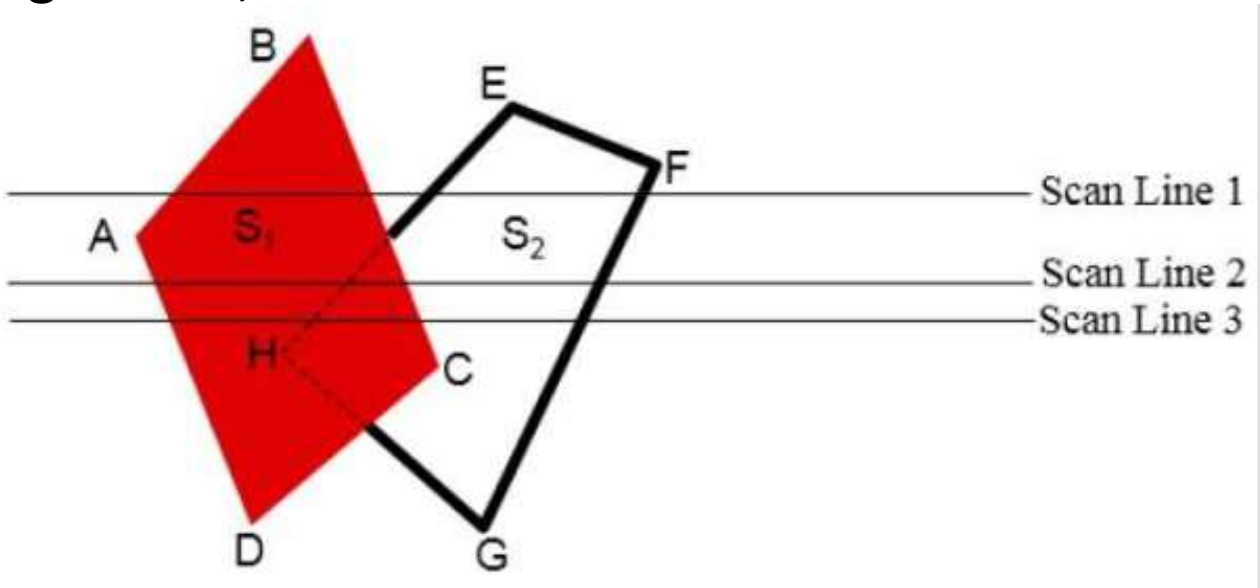
Scan line method

- For scan lines 2 and 3 in Fig. 13-10, the active edge list contains edges ***AD,*** *EH,* ***BC,*** and *FG.*
- Along scan line 2 from edge ***AD*** to edge *EH,* only the flag for surface S1 is on.
- But between edges EH and ***BC,*** the flags for both surfaces are on.



Scan line method

- Depth calculations must be made using the plane coefficients for the two surfaces.
- For this example, the depth of surface **S1** is assumed to be less than that of S2 so intensities for surface S1 are loaded into the refresh buffer until boundary BC is encountered.
- Then the flag for surface **S1** goes off, and intensities for surface ***S2*** are stored until edge FG is passed.



Basic Illumination Models

1. Ambient light
2. Diffuse reflection
3. Specular reflection and Phong model
4. Combined approach
5. Warn model
6. Intensity Attenuation
7. Color consideration
8. Transparency and Shadows

Basic Illumination Model

- An illumination model, also called a lighting model and sometimes referred to as a shading model, is used to calculate the intensity of light that we should see at a given point on the surface of an object.
- Illumination models in computer graphics are often loosely derived from the physical laws that describe surface light intensities.

Basic Illumination Model

- **Illumination Model -** the model for calculating light intensity at a single surface point is called *illumination model* or a *lighting model*
- **Surface Rendering** is a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.

LIGHT SOURCES

- When we view an opaque non uniform object, we see reflected light from the surfaces of the object.
- A surface that is not directly exposed to a light source may still be visible if nearby objects are illuminated.

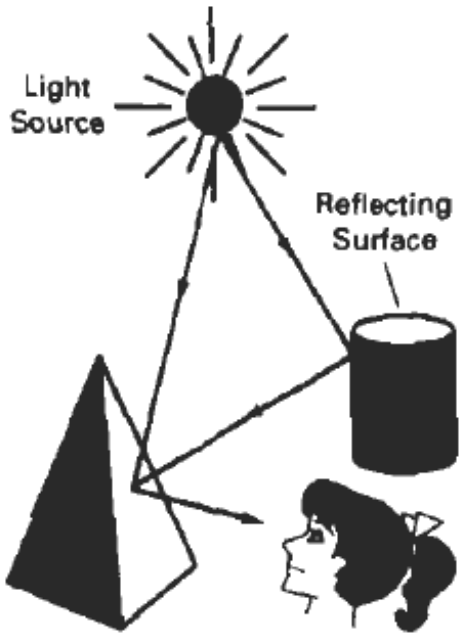


Figure 14-1
Light viewed from an opaque nonluminous surface is in general a combination of reflected light from a light source and reflections of light reflections from other surfaces

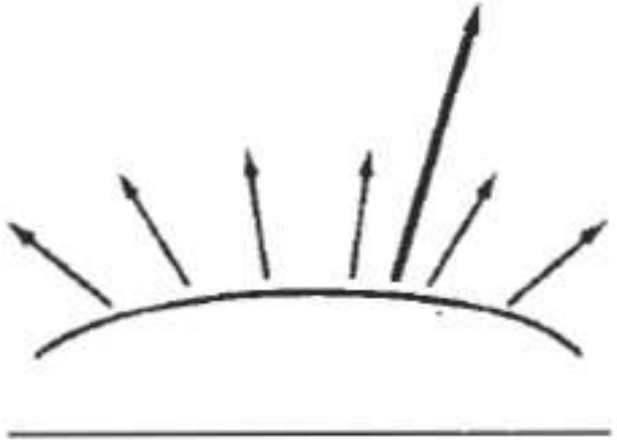
Diffuse reflection.

- Surfaces that are rough, or grainy, tend to scatter the reflected light in all directions.
- This scattered light is called diffuse reflection.

Figure 14-4
Diffuse reflections from a surface.

Specular Reflection

- In addition to diffuse reflection, light sources create highlights, or bright spots, called specular reflection.
- This highlighting effect is more pronounced on shiny surfaces than on dull surfaces.



Specular Reflection and Diffuse reflection

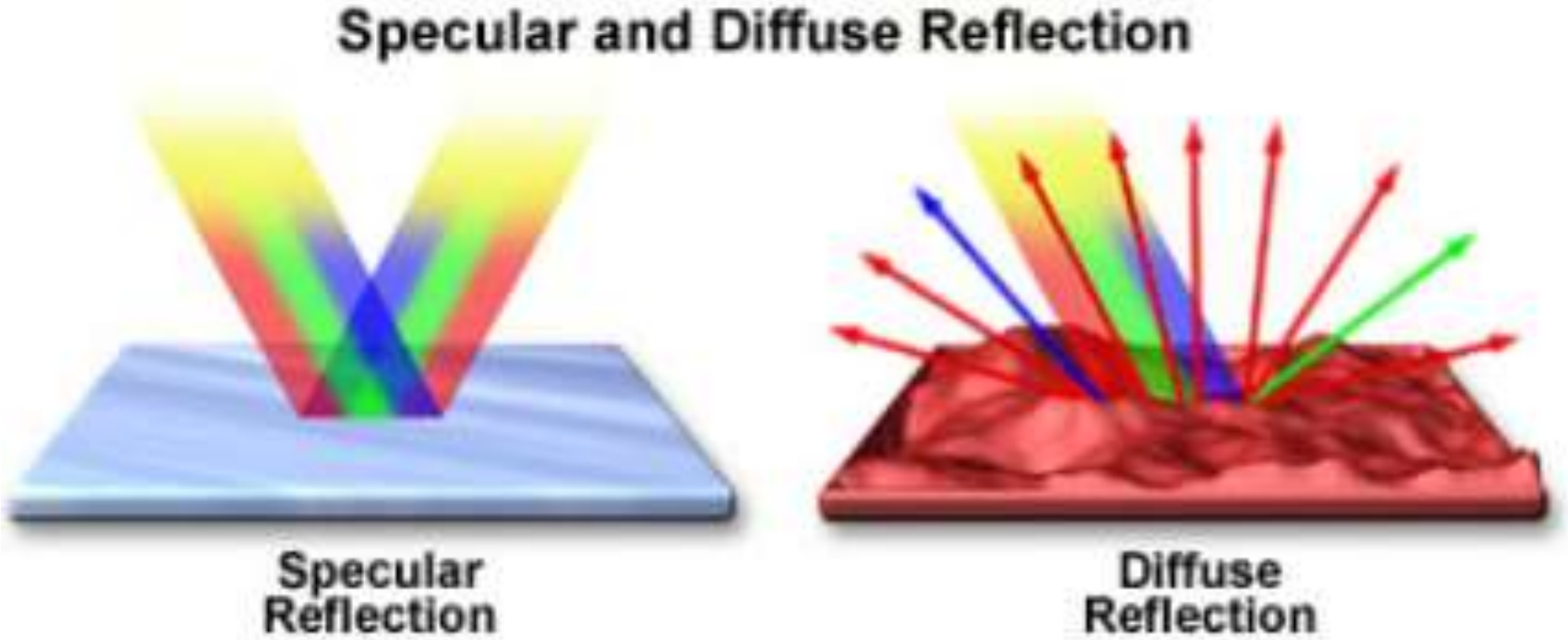


Figure 1

Ambient light

- This is a simple way to model the combination of light reflections from various surfaces to produce a uniform illumination called the ambient light, or background light.

Ambient light (I_a)

- Ambient light has no spatial or directional characteristics.
- The amount of ambient light incident on each object is a constant for all surfaces and over all directions.
- The intensity of the reflected light for each surface depends on the optical properties of the surface; that is, how much of the incident energy is to be reflected and how much absorbed.
- We can set the level for the ambient light in a scene with parameter I_a and each surface is then illuminated with this constant value.

Diffuse reflection (k_d)

- Diffuse reflections are constant over each surface in a scene, independent of the viewing direction.
- The fractional mount of the incident light that is diffusely reflected can be set for each surface with parameter ***kd,*** the diffuse-reflection coefficient, or diffuse reflectivity.
- Parameter ***kd*** is assigned a constant value in the interval **0** to **1,** according to the reflecting properties we want the surface to have.

Diffuse reflection

- If a surface is exposed only to ambient light, we can express the intensity of the diffuse reflection at any point on the surface as

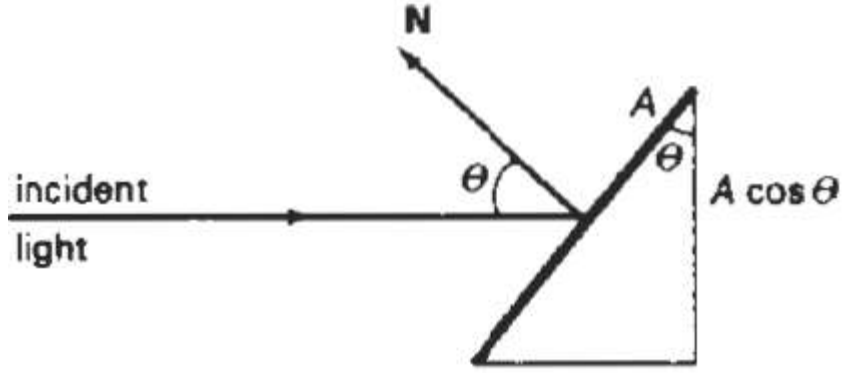
$$I_{ambdiff} = k_d I_a$$

- (where Kd is the diffuse-reflection coefficient)

Diffuse reflection

- If we denote the angle of incidence between the incoming light direction and the surface normal as θ , then the projected area of a surface patch perpendicular to the light direction is proportional to $\cos\theta$.
- If I_L is the intensity of the point light source, then the diffuse reflection equation for a point on the surface can be written as

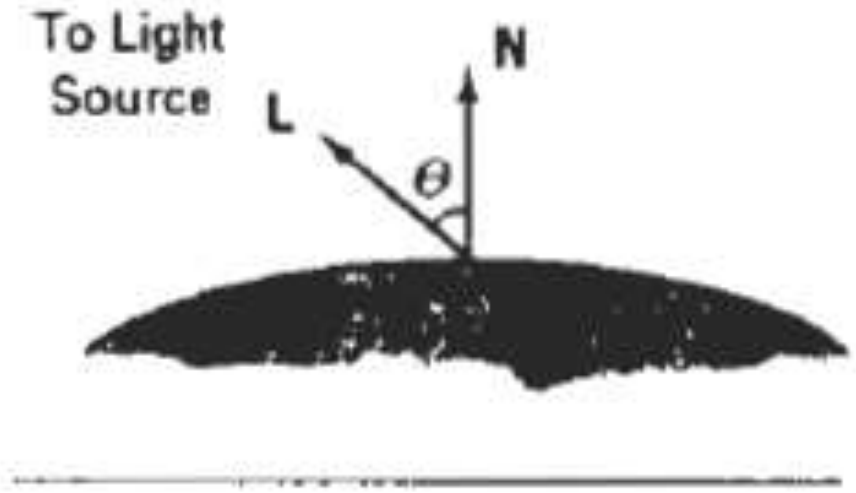
$$I_{l,diff} = K_d I_l \cos\theta$$



Diffuse reflection

- If N is the unit normal vector to a surface and L is the unit direction vector to the point light source from a position on the surface
- (Fig. 14-9), then $\cos\theta = N.\mathbf{L}$ and the diffuse reflection equation for single point-source illumination is

$$I_{l,diff} = K_d I_l (N.L)$$



Diffuse reflection

- We can **combine the ambient and point source intensity c**alculations to obtain an expression for the total diffuse reflection.
- In addition, many graphics packages introduce **an ambient-reflection coefficient *ka* to modify the ambient light intensity *Ia***, for each surface.

$$I_{diff} = k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L})$$

Diffuse reflection Ka=0 and kd=[0,1]

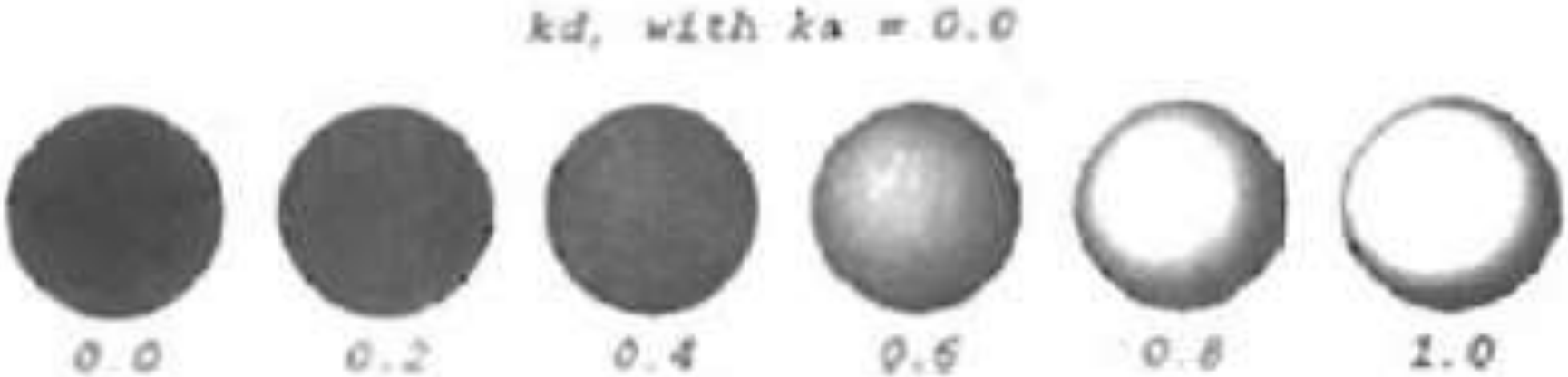


Figure 14-10
Diffuse reflections from a spherical surface illuminated by a point light source for values of the diffuse reflectivity coefficient in the interval $0 \leq k_d \leq 1$.

Diffuse reflection Ka and kd in [0,1]

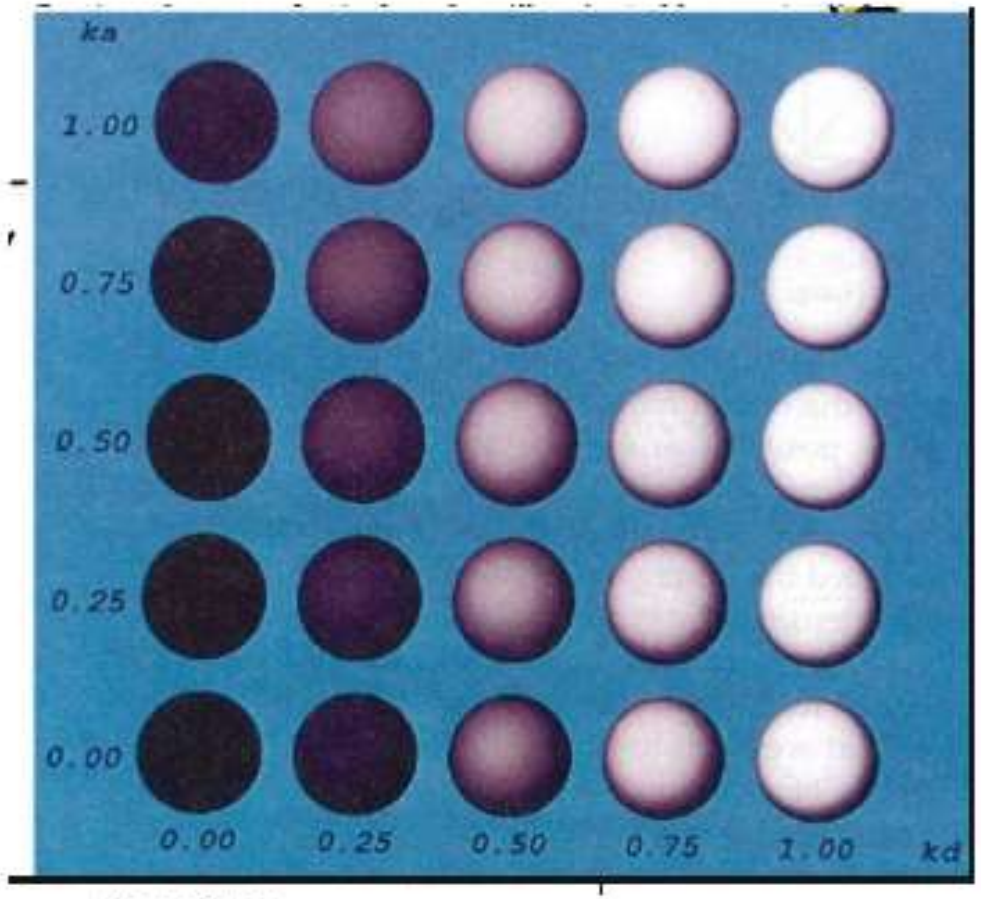
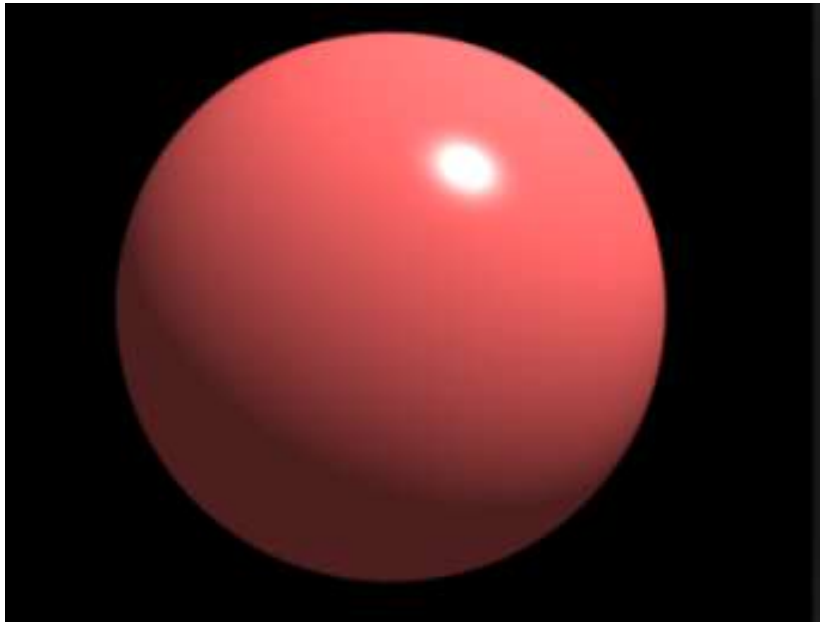


Figure 14-11
Diffuse reflections from a spherical surface illuminated with ambient light and a single point source for values of k_a and k_d in the interval (0, 1).

Specular reflection and Phong model

- **Specular reflection**
- When we look at an illuminated shiny surface, **such as polished metal**, an apple, or a person's forehead, we see a **highlight, or bright spot**, at certain viewing directions.
- This phenomenon, called *specular reflection,* is the result of total, or near total, reflection of the incident light in a concentrated region around the specular reflection angle.



Specular reflection and Phong model

- In this figure, we use
- **N** unit normal surface vector
- R to represent the unit vector in the direction of **ideal specular reflection**;
- L to represent the unit vector directed toward the **point light source**;
- **V** as the unit vector pointing to the **viewer** from the surface position.

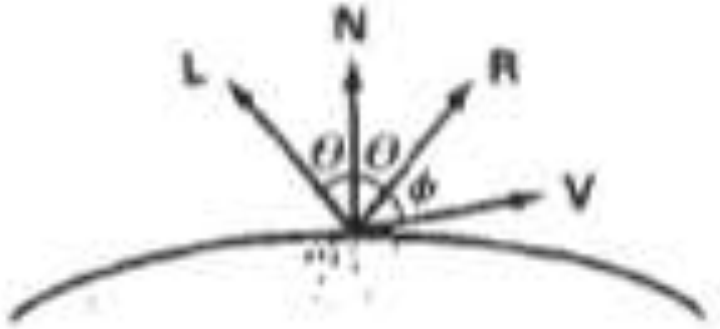


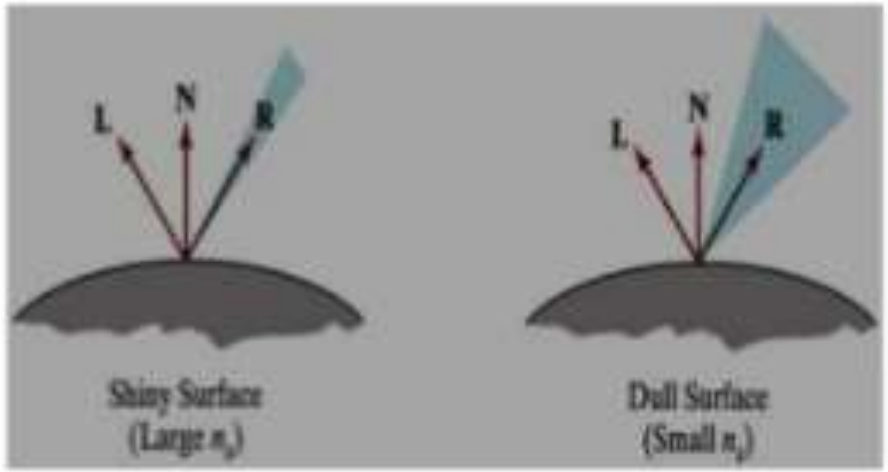
Figure 14-12
Specular-reflection angle equals angle of incidence θ.

Specular reflection and Phong model

- **Phong specular-reflection model**
- Shiny surfaces have a narrow specular-reflection range, and dull surfaces have a wider reflection range.
- An empirical model for calculating the specular-reflection range, developed by **Phong Bui Tuong** and called the Phong specular-reflection model.

Specular reflection and Phong model

Phong specular-reflection model, or **simply the Phong model,** sets the intensity of specular reflection proportional to $\cos^{n_s}\phi$. Angle ϕ can be assigned values in the range 0° to 90°, so that $\cos\phi$ varies from 0 to 1. The value assigned to *specular-reflection parameter* n_s is determined by the type of surface that we want to display. A very shiny surface is modeled with a large value for n_s (say, 100 or more), and smaller values (down to 1) are used for duller surfaces.



Specular reflection and Phong model

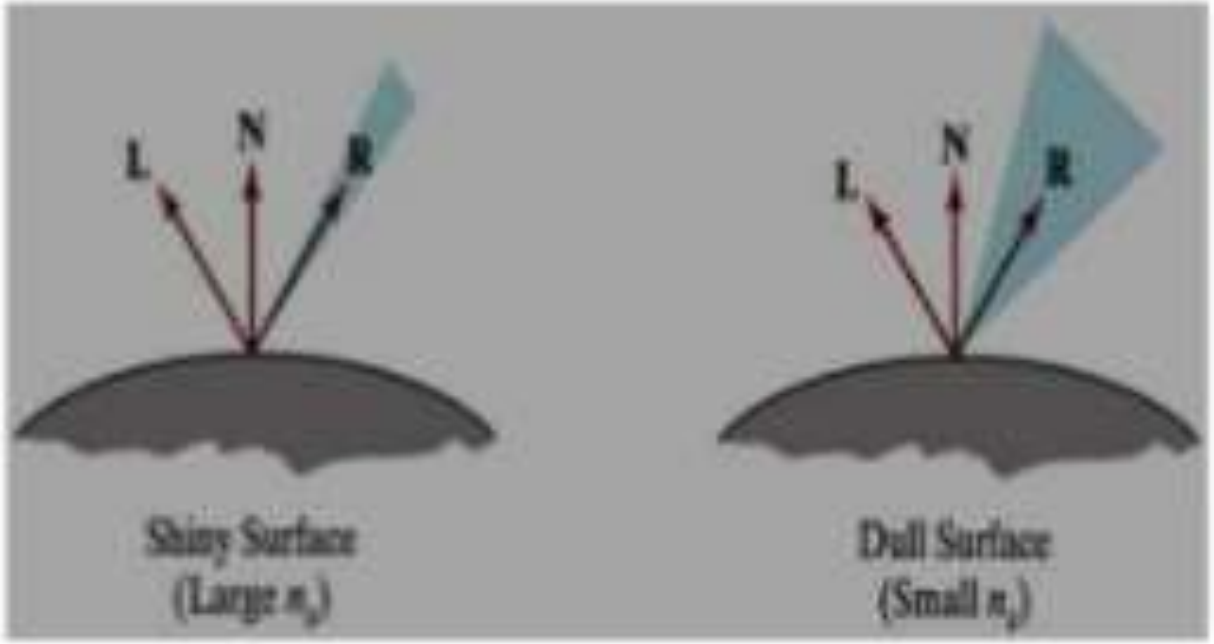
- The intensity of specular reflection depends on the **material properties of the surface and the angle of incidence**, as well as other factors such as the polarization and color of the incident light.
- We can approximately model monochromatic specular intensity variations using a **specular-reflection coefficient, W(θ),** for each surface.

$$I_{spec} = W(\theta)\, I_l \cos^{ns} \phi$$

$$I_{spec} = W(\theta) I_l \cos^{ns} \phi$$

Specular reflection and Phong model

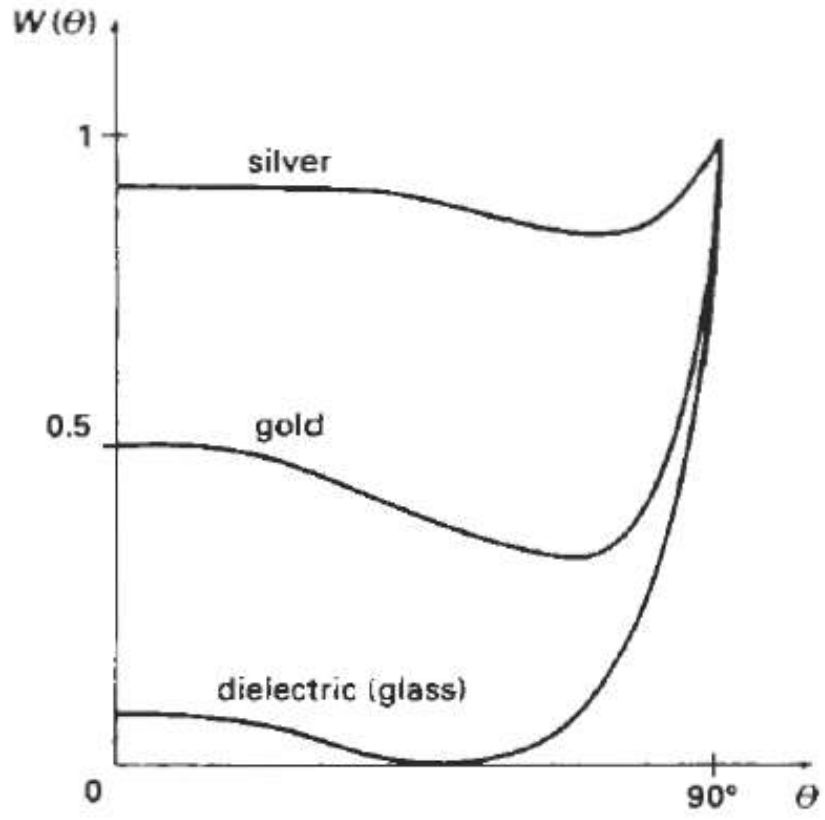
$$I_{spec} = W(\theta)\, I_l \cos^{ns} \phi$$



Specular reflection and Phong model

$I_{spec} = W(\theta)\, I_l \cos^{ns}\phi$

specular-reflection coefficient, W(θ) for different materials

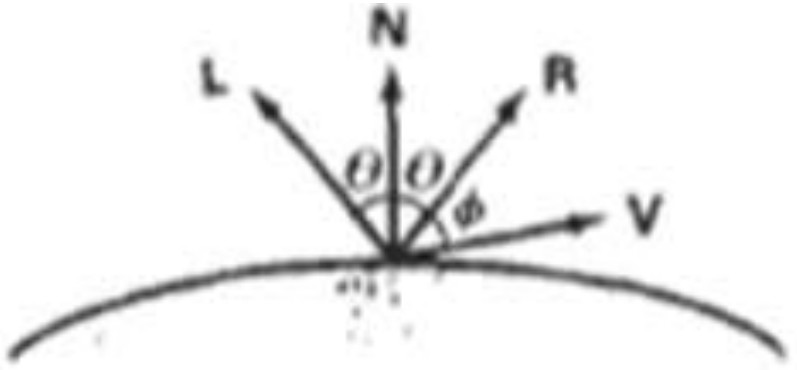


Specular reflection and Phong model

- We can reasonably model the reflected light effects by replacing ***W(θ)*** with a constant specular-reflection coefficient ***ks*** We then simply set ***ks*** equal to some value in the range 0 to 1 for each surface.
- Since V and R are unit vectors in the viewing and specular-reflection directions, we can calculate the value of cosϕ with the dot product V R.

$$I_{spec} = W(\theta)\, I_l\, \mathbf{COS}^{ns}\, \phi$$

$$I_{spec} = K_s I_l\, (V.R)^{ns}$$



Specular reflection and Phong model

Vector **R** in this expression can be calculated in terms of vectors **L** and **N**. As seen in Fig. 14-16, the projection of **L** onto the direction of the normal vector is obtained with the dot product $\mathbf{N} \cdot \mathbf{L}$. Therefore, from the diagram, we have

$$\mathbf{R} + \mathbf{L} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N}$$

and the specular-reflection vector is obtained as

$$\mathbf{R} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$$

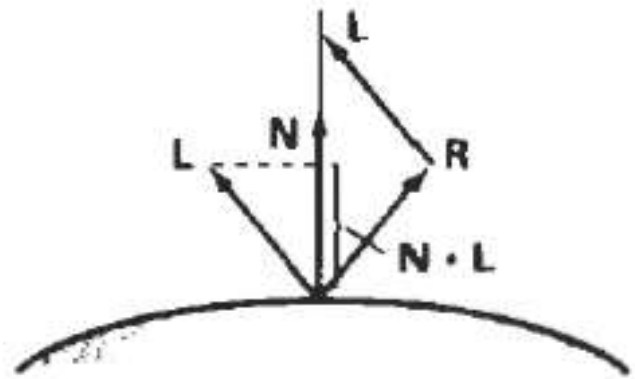


Figure 14-16
Calculation of vector **R** by considering projections onto the direction of the normal vector **N**.

Simplified Specular reflection and Phong model

- **A** somewhat simplified Phong model is obtained by using the halfway vector H between Land V to calculate the range of specular reflections.
- If we replace **V.**R in the Phong model with the dot product N.H
- (N.H requires less calculation as compared to V.R)

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$$

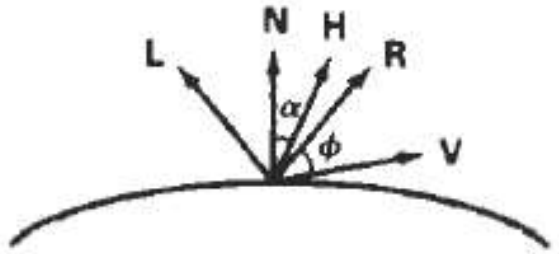


Figure 14-18
Halfway vector **H** along the bisector of the angle between **L** and **V**.

Combined Diffuse and Specular Reflections with Multiple Light Sources

- For a single point light source, we can model the combined diffuse and specula reflections from a point on an illuminated surface as

- $I = I_{diff} + I_{spec}$
- $I = K_a I_a + K_d I_l (N.L) + K_s I_l (N.H)^{ns}$

Combined Diffuse and Specular Reflections with Multiple Light Sources

- A single point light source

$$I = I_{diff} + I_{spec}$$

$$= k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}) + k_s I_l (\mathbf{N} \cdot \mathbf{H})^{n_s}$$

- Multiple light sources

$$I = I_{ambdiff} + \sum_{l=1}^{n} [I_{l,diff} + I_{l,spec}]$$

$$= k_a I_a + \sum_{l=1}^{n} I_l [k_d (\mathbf{N} \cdot \mathbf{L}_l) + k_s (\mathbf{N} \cdot \mathbf{H}_l)^{n_s}]$$

Warn model

- The Warn model provides a method for **simulating studio lighting effects by controlling light intensity in different directions**.
- Intensity in different directions is controlled by selecting values for the Phong exponent In addition, light controls, such as "barn doors" and spotlighting, used by studio photographers can be simulated in the Warn model.
- It is implemented in PHIGS+

Warn model

Figure 14-20
Studio lighting effects produced with the Warn model, using five light sources to illuminate a Chevrolet Camaro. (*Courtesy of David R. Warn, General Motors Research Laboratories.*)

Intensity Attenuation

- As radiant energy from a point light source travels through space, its amplitude is attenuated by the factor $1/d^2$, where d is the distance that the light has traveled.
- a general inverse quadratic attenuation function can be set up as

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

- A user can then fiddle with the coefficients a0, a1, and a2, to obtain a variety of lighting effects for a scene.

Intensity Attenuation

With a given set of attenuation coefficients, we can limit the magnitude of the attenuation function to 1 with the calculation

$$f(d) = \min\left(1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right) \qquad (14\text{-}12)$$

Using this function, we can then write our basic illumination model as

$$I = k_a I_a + \sum_{i=1}^{n} f(d_i) I_{li}[k_d(\mathbf{N} \cdot \mathbf{L}_i) + k_s(\mathbf{N} \cdot \mathbf{H}_i)^{n_s}] \qquad (14\text{-}13)$$

where d_i is the distance light has traveled from light source i.

Color consideration

- To incorporate color, we need to write the intensity equation as a function of the color properties of the light **sources** and object surfaces.
- For an RGB description, each color in a scene is expressed in terms of red, green, and blue components.

Color consideration

- The diffuse reflation coefficient vector, for example, would then have RGBcomponents *(kdR,* kdG, kdB).
- If we want an object to have a blue surface, we select a nonzero value in the range from 0 to 1 for the blue reflectivity component, kdB
- The red and green reflectivity components are set to zero
  (kdR = kdG = 0).
- The intensity calculation for this example reduces to the single expression

$$I_B = k_{aB} I_{aB} + \sum_{i=1}^{n} f_i(d) I_{lBi} [k_{dB}(\mathbf{N} \cdot \mathbf{L}_i) + k_{sB}(\mathbf{N} \cdot \mathbf{H}_i)^{n_s}]$$

Transparency

- Realistic transparency effects are modeled by considering light refraction.
- When light is incident upon a transparent surface, part of it is reflected and part is **refracted.**

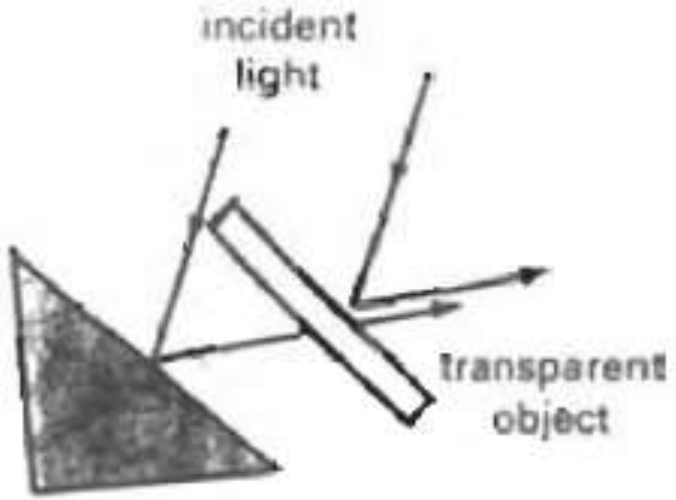


Figure 14-25
Light emission from a transparent surface is in general a combination of reflected and transmitted light.

Transparency

- ***Snell's Law***
- Angle of refraction **θr,** is calculated from the angle of incidence **θi,** the index of refraction **ni** of the "incident" material (usually air), and the index of refraction **nr** of the refracting material according to

$$\sin \theta_r = \frac{\eta_i}{\eta_r} \sin \theta_i$$

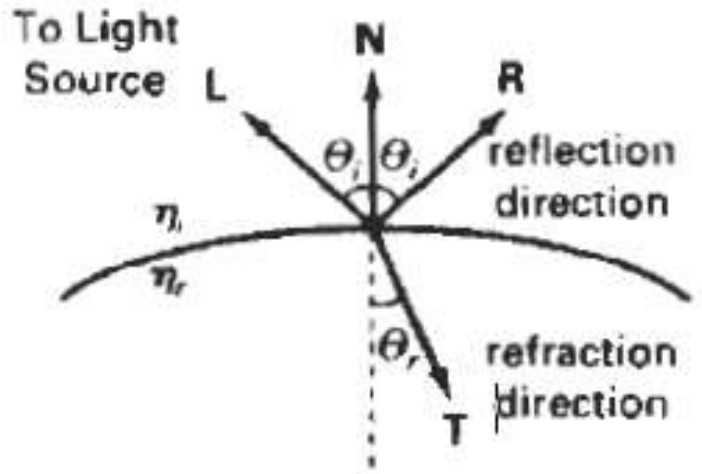


Figure 14-27
Reflection direction **R** and refraction direction **T** for a ray of light incident upon a surface with index of refraction η_r.

Transparency

From Snell's law and the diagram in Fig. 14-27, we can obtain the unit transmission vector **T** in the refraction direction θ_r as

$$\mathbf{T} = \left(\frac{\eta_i}{\eta_r} \cos \theta_i - \cos \theta_r \right) \mathbf{N} - \frac{\eta_i}{\eta_r} \mathbf{L} \qquad (14\text{-}18)$$

Transparency

- **A** simpler procedure for modeling transparent objects is to ignore the path shifts altogether.
- In effect, this approach assumes there is no change in the index of refraction from one material to another, so that the angle of refraction is always the same as the angle of incidence.

Transparency

- We can combine the transmitted intensity $\mathbf{I_{trans}}$ through a surface from a background object with the reflected intensity $\mathbf{I_{refl}}$ from the transparent surface (Fig. 14-29) using a **transparency coefficient kt.**
- We assign parameter kt, a value between 0 and 1 to specify how much of the background light is to be transmitted.
- Total surface intensity is then calculated as

$$I = (1 - k_t)I_{refl} + k_tI_{trans}$$

- The term **(1** - k_t) is the opacity factor.

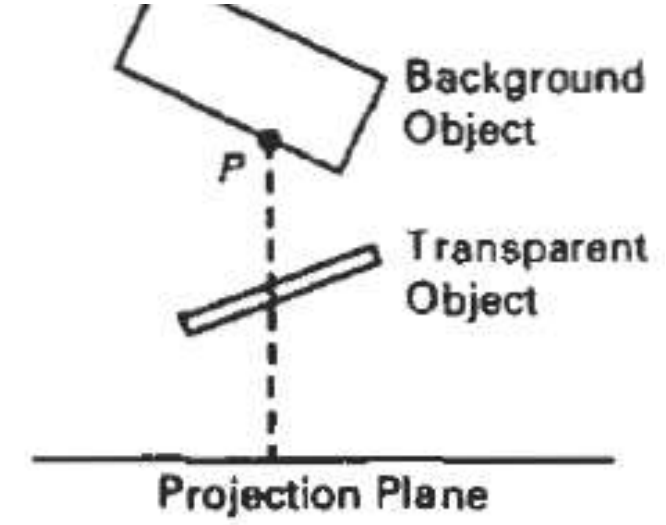
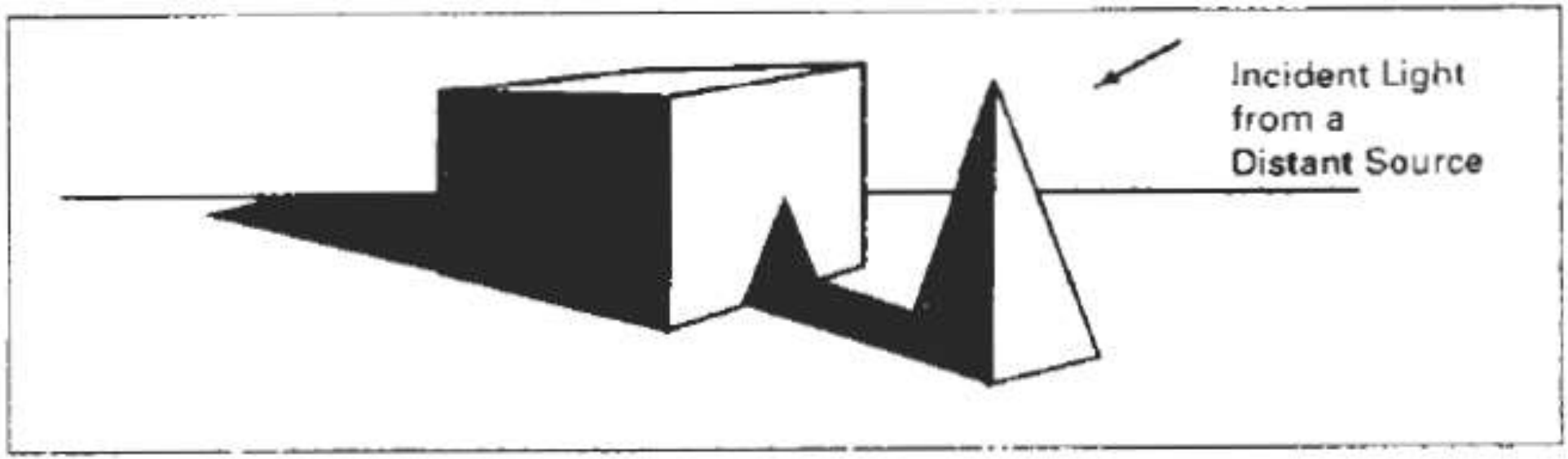


Figure 14-29

Shadows



Shadows

- Hidden-surface methods can be used to locate areils where light sources produce shadows.
- By applying a hidden-surface method with a light source at the view
- position, we can determine which surface sections cannot be "seen" from the light source.
- These are the shadow areas.

Shadows

Figure 14-26

Shadows

- Once we have determined the shadow areas for all light sources, the shadows could be treated as surface patterns and stored in pattern arrays.
- The scene in Fig. 14-26 shows shadow effects produced by multiple light sources.
- We can display shadow areas with ambient-light intensity only, or we can combine the ambient light with specified surface textures.

Figure 14-26

References

- https://www.tutorialspoint.com/computer_graphics/visible_surface_detection.htm