Unit 5

Syllabus:- Hidden Lines and Surfaces: Back Face Detection algorithm, Depth(Z buffer) buffer method, A buffer method ,area subdivision, Scan line method, basic illumination models– Ambient light, Diffuse reflection, Specular reflection and Phong model, Combined approach, Warn model, Intensity Attenuation, Color consideration, Transparency and Shadows.

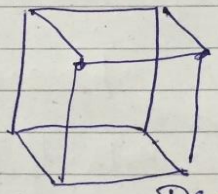
Unit 5

Hidden and Visible Surfaces

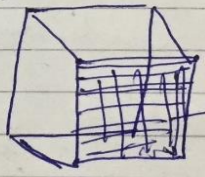
Properties
Advantages
Drawbacks
No. Derivation
No. Numerical
Diagrams imp

1. What is Hidden and Visible surfaces ?

Ans. The surfaces that are blocked or hidden from view must be "removed" in order to construct a realistic view of 3D scene. The identification and removal of these surfaces is called Hidden surface problem.



1. Draw a cube 2 cm.



(1) it takes extra time to rendering of picture

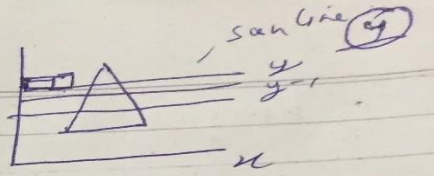
(2) But we have to see only front side.

(3) So it is required to identify & remove hidden surface ; saves processing time.

Hidden lines and surfaces :- The problem of hidden-surface removal or equivalently visible surface determination is solved by applying different hidden-surface algorithms.

We have two approaches (algorithm classified)

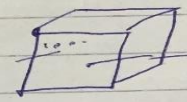
I Object space Method	II. Image space method.
1. Deal with Physical coordinates of object or World-coordinate system.	1. Deal with viewport/ projection port coordinate or device coordinate (Raster scan).
2. Compare object parts (which one is visible with other part) — Visibility.	2. Visiblity is decided point to point at each pixel level.
3 Line drawing algorithm	3). Line / surfaces algorithm
4) Geometrical relationship among objects.	4) Final Image configuration concentrate
5). Geometrical calculation - Precise, accurate float data ← Processing	5) Pixel 400 × 400 (calculation not above that)
6.) Processing time increase	6) Less processing time.
7) one pixel might draw many times.	7) Pixel draw once but surface draw many times.



6) Scan line Coherence (Span)

Span (a group of adjacent pixel of a scan line)

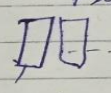
7) Depth Coherence :- (Z



adjacent Z in same face is slightly - smooth change (determined by simple difference equation)

8) Frame Coherence : (30 frames per second - animated video)

adjacent frame have some similar (little change) properties.



Algorithm used for hidden line surface detection

Back-face detection

1. Back-face Removal (object space method)
2. Depth-buffer (Z buffer (Image space method)
3. A-buffer (extension of Z-buffer) (Image space method)
4. Scan-line method (Image space method)
5. Area Sub division (Warnock's) (")

Techniques for efficient visible-surface algorithms.

1. **Coherence** (similarity or similar properties - reuse, save time (minimize processing time)

Types of Coherence

1. Object coherence :- Separate object (Non-overlapping

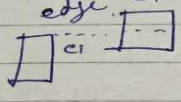
2. Face Coherence (surface / polygon / lines) :-

face.

two adjacent pixel belong to same face

2. same color, visibility

3) Edge coherence :-

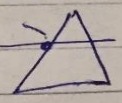


Stops visible when some other object crosses.

4. Area Coherence :- (window, Position

Area two adjacent pixel have same properties (same color, Intensity)

5. Implied Edge Coherence.



Intersection pt. can be calculated by finding endpts.

1 Back face detection (Back face removal)

(i) $Ax + By + Cz + D < 0$ — inside surface.

(if pt. is inside — check it is in back face).

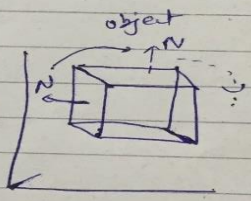
(ii) $V \cdot N > 0$ back face hidden.

$V \cdot N = |V||N| \cos\theta$

$0 \le \cos\theta \le 1$
$0 \le \theta \le \frac{\pi}{2}$ (90°)] Back face

[When an inside pt. is along the line of sight to the surface, the polygon must be a back face.

[where N is normal vector to object, V is a vector in viewing direction from (camera) position]



$\overleftarrow{N}\ \overleftarrow{V} = 0°$

$\overleftarrow{N}\ \overleftarrow{V} = 180°$ (more than 90°) (then point is visible) visibility start)

(iii) $V \cdot N = V_z \cdot C$ [value of C decides — back face visible face)

(negative) (∵ we cannot see any face C=0]

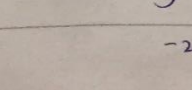
$\Rightarrow (c) \le 0$ (Back face)

$N(Ai + Bj + Ck)$

(i, j, k unit vector)

if we are viewing along −Z direction (Right handed)

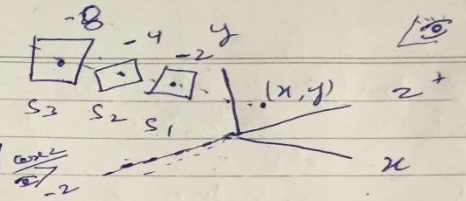
$V(0, 0, V_z)$ — Z direction



Advantage :- Fast and simple object space method for identifying back face of a polyhedron.

Disadvantage :- Can not be used for partially hidden or completely hidden objects. We need more test for finding these overlapping objects.

Z-Buffer Algorithm
Depth Buffer Algorithm



Case 1

(1) It is an image-space approach to eliminate hidden surfaces or detecting visible surface.

(2) It compares surface Depth at each pixel position.

Depth means Z value (how far the surface is from view plane).

3) Z buffer is extension of the frame buffer idea.

2 buffer.

(D) Z-buffer | holds Z-value (Depth)

(R) Refresh buffer (frame buffer) | holds Intensity.

Both have same size

write in paper

Case 1 :- Sitting +z looking at origin or -z axis

(mostly books covered this)

Larger Z value = closer, smaller Z value farther.

(1) $D(x,y) = 0$ (zero)
$R(x,y) = I_{background\ color}$ (Intensity)

2) $Z_{New} > D(x,y)$
$R(x,y) = I_{new}$ (Intensity at Z_{new})

Case 2 : ~~sitting~~ sitting at -Z ; looking towards +z axis

(just for clarity.)

Smaller Z value : closer
Larger Z value : far

(1) $D(x,y) = Z_{max}$ (∞)
$R(x,y) = I_{Background}$

(2) $Z_{new} < D(x,y)$
$D(x,y) = Z_{new}$
$R(x,y) = I_{new}$

The z buffer algorithm can be stated as:

Step 1 : Initialize frame buffer (Refresh buffer) to background color.

Step 2. Initialize z or D buffer to minimum z value (0).

Step 3: Scan convert each polygon in arbitrary order.

Step 4: For each (x,y) pixel, calculate depth 'z' of that pixel z(x,y) or D(x,y).

step 5. Compare calculated new depth z(x,y) with value previously stored in z buffer at that location. z(x,y).

step 6: if $z(x,y)_{new} > z(x,y)_{old}$, then write the new depth value to z-buffer and update the frame buffer.

Step 7. otherwise, no action is taken.

The Depth calculation.

equation of plane of current face being scan-converted.

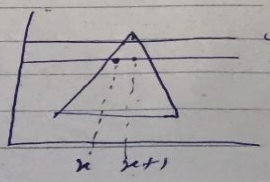
$$Ax + By + Cz + D = 0$$

then $$z = -\frac{(Ax + By + D)}{C}$$ (z value at x)

For same line for (x+1)

$$z' = \frac{-A(x+1) - By - D}{C}$$ (one scan line same y value.

So $z' = z - \frac{A}{C}$ and $-\frac{A}{C}$ is constant



(by coherence property we can reuse previous values)

For Next scan line

$$z' = \frac{B + Ax}{C}$$ (change $y+1$ in y).

Advantages of Z buffer or Depth Buffer.

1. It is essey to implement
2. It can be implemented in hardware to overcome the speed problem.
3. Since the algorithm processes object one at a time, the total number of polygons in a picture can be arbitrarily large (overlapping polygons).

Disadvantages

1. It requires an additional buffer and hence the large memory.
2. It is time consuming process.
3. It only works for Opaque surfaces.

A - Buffer Algorithm

(Extension of Z buffer) also called or represents [antialiased, area - averaged, accumulation - buffer] method for implementation in the surface - rendering system called.

RAYES (an acronym for "Renders Everything You Ever Saw").

[Z - Buffer - drawback - only works for opaque surfaces. means it cannot accumulate intensity values for more than one surface, as is necessary if transparent surfaces are to be displayed.]

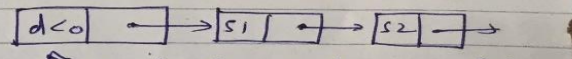
A - Buffer

Each position in A buffer has 2 fields.

2 field maintain

depth field
(stores a positive or negative real numbers)
→ Real no. +ve.
→ Real no -ve.

Intensity field
(stores surface - intensity information or a pointer value)
→ Pixel Intensity
(surface S1, surface S2)



single - surface overlap
(a)

multiple surface overlap.
(b) (Pixel is overlap by multiple surfaces)

(When a pixel is overlap by)

single surface

depth field | Intensity field

① if depth field is +ve single surface overlap
② if depth field is -ve multiple surface contributes to pixel Intensity.

↓ it maintains pointer to linked list (depth wise surface).

Data for each surface in the linked list includes.

(1) RGB Intensity components
(2) Opacity parameter (percent of transparency)
(3) Depth
(4) percent of area coverage
(5) Surface identifier
(6) Other surface-rendering parameters
(7) pointer to next surface.

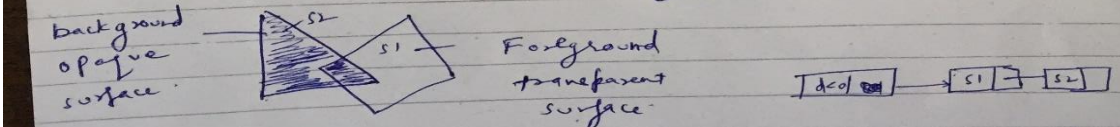
[final color of a pixel is computed by summing up of all of its subpixels. Due to this accumulation taking place at sub-pixel level; so. A-buffer method gets the name accumulation buffer.

Advantages of A-Buffer:

(1) A buffer method provides anti-aliasing
(2) A buffer method can be used for transparent objects. (It resolves visibility among an arbitrary opaque, transparent, intersecting objects)

Disadvantage:

(1) A buffer method requires more memory than Z buffer
(2) It is complex (exact image can be computed in $O(\log_2 n)$ time in the worst case $O(n^2)$ work in total.



Viewing an opaque surface through a transparent surface requires multiple surface intensity contributions for pixel position.

4 Scan line Algorithm for Hidden surfaces:

This image-space method for removing hidden surfaces is an extension of the scan-line algorithm for filling polygon interior. Instead of filling just one surface, we deal with multiple surfaces.

It maintain 3 tables.

- Edge Table (ET)
- Active edge table (AET)
- Polygon Table (PT)

(ET)

1	2	3	4	5
x	y_{max}	Δx	ID	→

Edge table

for edge which intersect with scan line or two edge intersects take value of x where value of y is smaller.

(1) x : y smaller
(2) y_{max} : max. y coordinate (end point)
(3) $\Delta x = \frac{1}{m}$ (slope)
(4) I_D = I_D of edge
(5) Pointers to polygon table to Identify the surface bounded by each line.

(3) We need slope for every intersection.
(4) ID of edge.

- Active edge table (AET)

1	2	3
scan line	edge	surface

(1) current scan line
(2) Edges that cross the current scan line (sorted in order of Increasing x)
3. Flag for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface.

(1) (2)

PT | ID | Polygon coefficients | shading/color/intensity | In-out | →

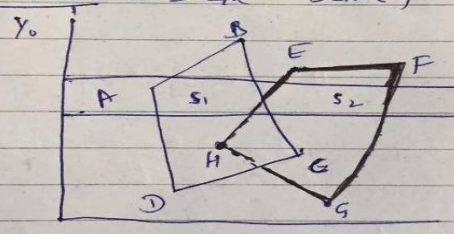
(1) ID of polygon (surface)

(2) Polygon coefficient - $Ax + By + Cz + D = 0$
(A, B, C, D)

(3) Intensity information for the surface

(4) In-out (default (false - out)

How it works (scan line)



(SL1)
scan line 1 · [y value]
scan line 2 [y-1 value]
(SL2)

SL1 - AB, BC - S_1
EH, FG - S_2
SL2 - BC, EH, S_1, S_2 → Depth compare
SL2 - AD - S_1
FG - S_2
S_2 depth Pixel position

AET (Table) active edge

SL1 — AB, BC, EH, FG.
scan line

(1) for position along this scan line between edges AB, BC only the flag for surface S_1 is on.

(2) ∴ No depth calculations are necessary, Intensity information for surface S_1 is entered from polygon table into refresh buffer.

3. Similarly, between edges EH, FG, only flag for surface S_2 is on.

4. No other positions ~~set to~~ intersect scan line 1, so intensity values in other areas are set to background Intensity.

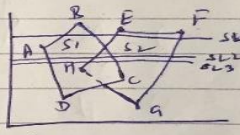
For scan line 2, 3.

(1) the active edge list contains AD, EH, BC and FG.

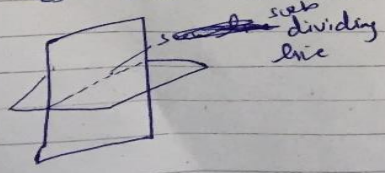
(2) Along scan line 2. from edge AD to edge EH, only flag for surface S_1 is on.

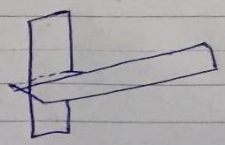
(3) But between edges EH and BC, flag for both the surfaces are on. In this interval, depth calculations must be made using plane coefficients for the two surfaces

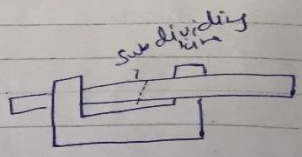
(4) For ex:- The depth of surface S_1 is assumed to be less than that of S_2, so intensities for surface S_1 are loaded into the refresh buffer until boundary BC is encountered. Then the flag for surface S_1 goes off and intensities for surface S_2 are stored until edge FG is passed.



5) We can take advantages of coherence along scan lines as we pass from one scan line to the next, ex:- scan line 3 has same active list of edges as scan line 2.







Advantages of scan line

1) It takes advantages of coherence resulting in fast algorithm.
2) It does require as much storage as depth buffer.
3) It only draws visible pixels.
4) This algorithm is common in software.

Disadvantages :-

(1) It is a more complex algorithm
(2) It requires all polygon sort to renderer before drawing

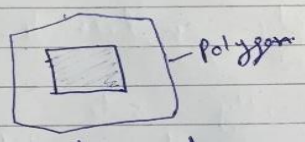
5 Area sub-division Algorithm (Warnock's)

(1) Image space method
(2) Uses Area coherence (relationship area, polygon)
(3) Divide and conquer
(4) It takes advantage by locating those views areas that represent part of a single surface
(5) Divide total viewing area into smaller & smaller rectangles until each small area is the projection of part of - single visible surface or no surface at all (single pixel.)

There are four possible relationships that a surface can have with a specified area boundary.

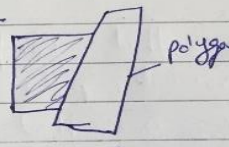
1. • Surrounding : One that completely encloses the area.
   Area color → surrounding polygon colour Fill.
2. • Overlapping surface : One that is partly inside and partly outside the area. (contained or overlapping
   Area color. (1) Background - (2) polygon color
3. • Inside surface. One that is completely inside the area. (contained or overlapping
   Area color → Background (2) polygon color.
4. • Outside (disjoint) surface. One that is completely outside the area.
   (Area color → background color.

5. 1.



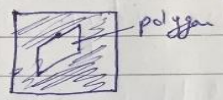
surrounding surface

2



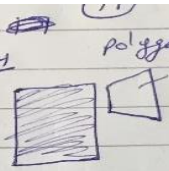
overlapping surface

3



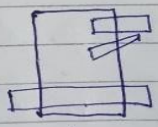
Inside surface

4



outside surface

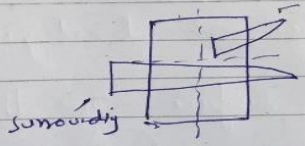
Special case. Area → surrounding + overlapping + contained polygon.



surrounding polygon

suppose surrounding polygon rear es compose depth with area.

So surrounding polygon colour



further area subdivision. 4 parts

↓

smallest unit Reach. (pixel position)

Illumination Model

Realistic displays of a scene are obtained by generating perspective projections of objects and by applying natural lighting effects to the visible surfaces.

An illumination also called lighting (brightness) model and sometimes called shading model, is used to calculate the intensity of light that we should see at a given point on the surface of an object.

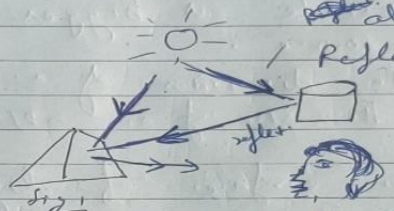
shading — Dark \ light

Illumination — Brightness (lighting)

Intensity — Quantity of illumination

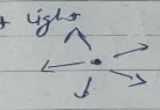
topic 1

Light source :-



Light viewed from an opaque nonluminous surface is in general a combination of reflected light from a light source and reflection of light reflections from other surfaces.

Types of light sources



(a)

Diverging ray paths from a point light source

(b)

An object illuminated with a distributed light source

(distributed light source)

Light source, we means an object that is emitting radiant energy such as light bulb or the sun.

Light sources that illuminate an object are of two basic types

(1) light - emitting source (eg: sun, bulb) ✓
(2) light - reflection source (eg walls of a room) ✓

~~Simplest light~~

(1) Simplest model for a light emitter is a point source when the surface of the object which we want to illuminate is bigger than the surface of light emitting source.

Thus the dimensions of a light source are smaller than the size of an object of Sun.

(2) But when the surface of light emit

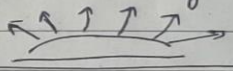
Light emitters. (two types.)

Point source (eg. sun) (bulb)	Distributed light source (neon tube light)
1. When the surface of light emitting source is smaller than the surface of object which we want to illuminate.	1. When the surface of light emitting source is greater than the surface of object.
2. Light source is far from object	2. Light source is near to object.

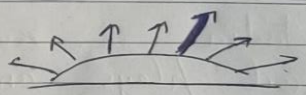
When light is incident on opaque surface, part of it is (1) is reflected.
(2) is absorbed

The amount of reflection depends on type of material.
(1) shiny object reflect more of incident light
(2) Rough, grainy, absorbs more of incident light

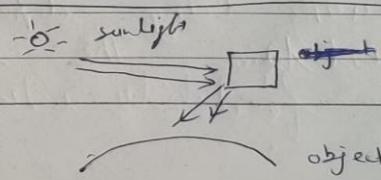
Diffuse Reflection :- surfaces that are rough, grainy, tend to scatter reflected light in all directions, and surface appears appears equally bright from all viewing direction.



Specular Reflection : In addition to diffuse reflection, light source create highlights, or bright spots. Highlighting effect is more on shiny than dull surfaces



Ambient light
(diffuse illumination)



object illuminate due to diffusing light from other surface.

object.

Q1: What is Ambient light (diffuse illumination eg. reflected light from walls)

Ans:- A surface that is not exposed directly to light source still will be visible if nearby objects are illuminated. In our basic illumination model, we can set a general level of brightness for a scene. This is a simple way to model the combination of light reflections from various surfaces to produce a UNIFORM ILLUMINATION. called AMBIENT Light or BACKGROUND Light.

Ambient light has no spatial or directional characteristics. The amount of ambient light incident on each object is constant for all surfaces and over all directions

We can set this level for Ambient light by I_a parameter.

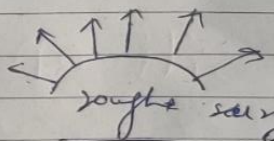
diagram from page 20.

Reflection two types → diffuse - (rough)
specular (shiny)

Q2 Diffuse Reflection :-

is the reflection of light from surface such that when the reflection are constant over each surface of object and they are independent of the viewing direction.

When object is illuminated
some part is - absorbed
" " " reflected.



The ratio of light reflected from the surface to the total incoming light to the surface is called coefficient of reflection or the reflectivity (R)

$R = \frac{I_R}{I_S}$ (The value of R - from 0-1.

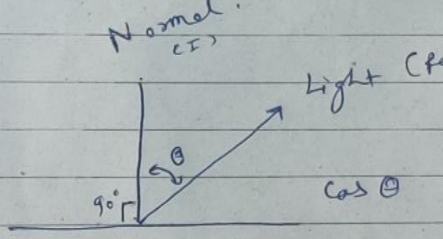
closer to 1 for white surface.
closer to 0 for black surface.

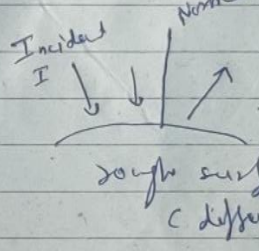
∴ white surfaces reflects all incident light.
and Black surface absorbs most of Incident light.

to How an object has colour

A blue object illuminated by a white light source.
- Reflect blue component of white light and absorb all other components.

LAMBERT'S COSINE Law [diffuse reflector or Lambertss reflectors]





(1) $I_{amb\,diff} = K_a I_a$

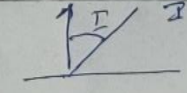
If a surface is exposed to ambient light.
[Intensity of diffuse reflection at any point on the surface.

[K_a - Coefficient or ambient reflection coefficient.
I_a = ambient light Intensity.

(2) Point source (+ light also from other point source).

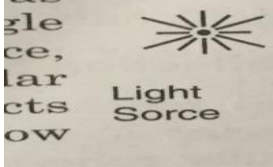
$I_{l,diff} = K_d I_l \cos\theta$

Diffuse reflection equation for a point on surface]



as
gle
ce,
lar
cts
ow

ing



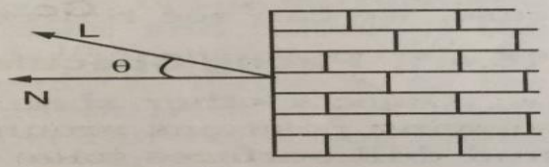
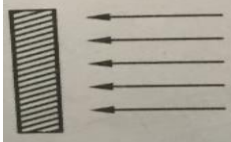
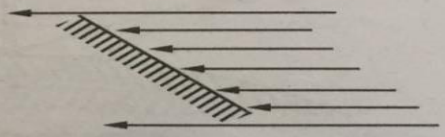


Fig. 15.5.



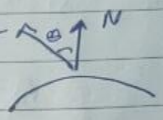


(a) More illuminated (b) Less illuminated

Fig. 15.6.

If N is the normal vector to a ~~light~~ surface and L is the unit of direction vector to the point light source from position on the ~~surface~~ surface.

then $\cos\theta = \underline{N \cdot L}$

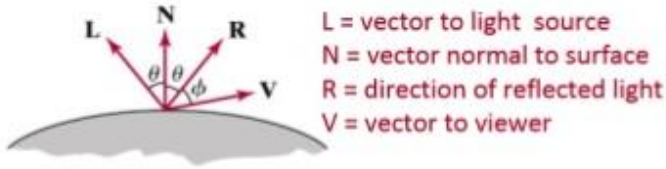


~~Diff~~ $I_{l,diff} = k_d I_l (N \cdot L)$.

So we can combine the ambient and point source intensity calculations to obtain an expression for total diffuse reflection.

$$I_{diff} = \underbrace{k_a I_a}_{\text{(Ambient-reflection coefficient to modify Ambient light intensity } I_a)} + k_d I_l (N \cdot L)$$

(value of both k_a, k_d - range from '0-1' and depend on surface material properties.



When the light source and the viewer are both at infinity, then the use of $N \cdot H$ offers a computational advantage, since H is constant for all surface points. Substituting $N \cdot H$ in place of $V \cdot R$ the intensity for specular reflection is given as

$$I_{spec} = k_s \, I_l \, (N \cdot H)^{n_s}$$

For given light-source and viewer positions, vector H gives the orientation direction for the surface that would produce maximum specular reflection in the viewing direction. Thus, H is also referred to as the surface orientation direction for maximum highlights.

15.4.2. Combined Diffuse and Specular Reflections

For a single point light source, the combined diffuse and specular reflections from any point on the illuminated surface is given as

$$I = I_{diff} + I_{spec} = k_a \, I_a + k_d \, I_l \, (N \cdot L) + k_s \, I_l \, (N \cdot H)^{n_s}$$

For a multiple point light source the above equation can be modified as

$$I = k_a \, I_a + \sum_{i=1}^{M} I_{li} \, [k_d \, (N \cdot L_i) + k_s \, (N \cdot H_i)^{n_s}]$$

Therefore, in case of multiple point light sources the light reflected at any surface point is given by summing the contributions from the individual sources.

Warn Model

So far we have considered only point light sources. The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.

Light sources are modelled as points on a reflecting surface, using the Phong model for the surface points. Then selecting values for the Phong exponent controls the intensity in different directions. In addition, light controls, such as "barn doors" and spotlighting, used by studio photographers can be simulated in the Warn model. *Flaps* are used to control the amount of light emitted by a source in various directions. Two flaps are provided for each of the x, y, and z directions. *Spotlights* are used to control the amount of light emitted within a cone with apex at a point-source position. The Warn model is implemented in **PHIGS+.**

Intensity Attenuation

As radiant energy from a point light source travels through space, its amplitude is attenuated by the factor $\mathbf{1/d^2}$, where d is the distance that the light has travelled. This means that a surface close to the light source (small $\boldsymbol{d}$) receives higher incident intensity from the source than a distant surface (large $\boldsymbol{d}$). Therefore, to produce realistic lighting effects, our illumination model should take this intensity attenuation into account. Otherwise, we are illuminating all surfaces with the same intensity, no matter how far they might be from the light source.

Our simple point-source illumination model, however, does not always produce realistic pictures, if we use the factor $\mathbf{1/d^2}$ to attenuate intensities. The factor $\mathbf{1/d^2}$ produces too much intensity variations when $\boldsymbol{d}$ is small, and it produces very little variation when $\boldsymbol{d}$ is large. This is because real scenes are usually not illuminated with point light sources, and our illumination model is too simple to accurately describe real lighting effects.

Graphics packages have remunerated for these problems by using *inverse linear or quadratic functions* of d to attenuate intensities. For example, a general inverse quadratic attenuation function can be set up as

$$f(d) = \frac{1}{\ldots}$$

A user can then fraud with the coefficients a_0, a_1, and a_2, to obtain a variety of lighting effects for a scene. The value of the constant term a_0, can be adjusted to prevent $f(d)$ from becoming too large when d is very small. Also, the values for the coefficients in the attenuation function, and the optical surface parameters for a scene, can be adjusted to prevent calculations of reflected intensities from exceeding the maximum allowable value. This is an effective method for limiting intensity values when a single light source is used to illuminate a scene.

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

$$I = k_a I + \sum_{i=1}^{n} f(d_i) I_{li} \left[k_d (N.L_i) + k_s (N.H_i)^{R_s} \right]$$

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

Color Considerations

Most graphics displays of realistic scenes are in color. But the illumination model we have described so far considers only monochromatic lighting effects. 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. We then specify the RGB components of light source intensities and surface colors, and the illumination model calculates the RGB components of the reflected light.

One way to set surface colors is by specifying the reflectivity coefficients as three-element vectors. The *diffuse reflection coefficient vector*, for example, would then have RGB components (k_{dR}, k_{dG}, k_{dB}). If we want an object to have a red surface, we select a nonzero value in the range from 0 to 1 for the red reflectivity component, k_{dR} while the green and blue reflectivity components are set to zero ($k_{dG} = k_{dB} = 0$).

Surfaces typically are illuminated with white light sources, and in general we can set surface color so that the reflected light has nonzero values for all three RGB components. Calculated intensity levels for each color component can be used to adjust the corresponding electron gun in an RGB monitor.

In his original *specular-reflection model*, Phong set parameter k_s to a constant value independent of the surface color. This produces specular reflections that are the same color as the incident light (usually white), which gives the surface a plastic appearance. For a non plastic material, the color of the specular reflection is a function of the surface properties and may be different from both the color of the incident light and the color of the diffuse reflections. We can approximate specular effects on such surfaces by making the specular-reflection coefficient color dependent.

Another method for setting surface color is to specify the components of diffuse and specular color vector for each surface, while retaining the reflectivity coefficients as single-valued constants. For an RGB color representation, for instance, the components of these two surface color vectors can be denoted as (S_{dR}, S_{dG}, S_{dB}) and (S_{sR}, S_{sG}, S_{sB}).

Other color representations besides RGB can be used to describe colors in a scene. And sometimes it is convenient to use a color model with more than three components for a color specification.

(UPTU 2009-10)

15.5. POLYGON-RENDERING METHODS

In this section, we consider the application of an illumination model to the rendering of standard graphics objects. Each polygon can be rendered with a single intensity or the intensity can be obtained at each point of the surface.

15.5.1. Constant-Intensity Shading

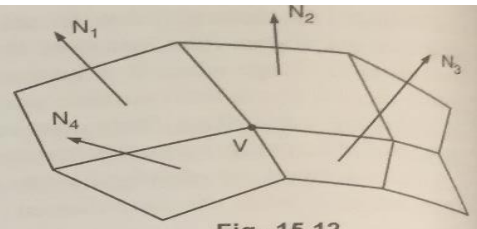


Fig. 15.12.

The fast and simplest method for shading polygon is *constant shading*, also known as **faceted shading** or **flat shading**. In this method, illumination model is applied only once for each polygon to determine single intensity value. The entire polygon is then displayed with the single intensity value.

This method is valid for the following assumptions:

1. The light source is at infinity, so $N \cdot L$ is constant across the polygon face.
2. The viewer is at infinity, so $V \cdot R$ is constant over the surface.
3. The polygon represents the actual surface being modeled, and is not an approximation to a curved surface.

If either of the first two assumptions are not true still we can use constant intensity shading approach.; however, we require some method to determine a single value for each of L and V vectors.

15.5.2. Gouraud Shading

This intensity-interpolation scheme, developed by *Henri Gouraud* and generally referred to as Gouraud shading. The polygon surface is displayed by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges. Thus, eliminates the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud shading by performing following calculations :

1. Determine the average unit normal vector at each polygon vertex.
2. Apply an illumination model to each polygon vertex to determine the vertex intensity.
3. Linearly interpolate the vertex intensities over the surface of the polygon.

We can obtain a normal vector at each polygon vertex by averaging the surface normals of all polygons sharing that vertex.

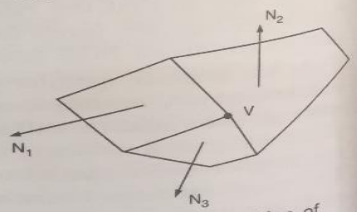


Fig. 15.13 Calculation of normal vector at polygon vertex V

As shown in the Fig. 15.13 there are three surface normals N_1, N_2 and N_3 of polygon sharing vertex V. Therefore, normal vector at vertex V is given as

$$N_v = \frac{N_1 + N_2 + N_3}{|N_1 + N_2 + N_3|}$$

In general, for any vertex position V, we can obtain the unit vertex normal by equation.

$$N_v = \frac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$$

Where n is the number of surface normals of polygons sharing that vertex.

The next step in Gouraud shading is to find vertex intensities. Once we have the vertex normals, their vertex intensities can be determined by applying illumination model to each polygon vertex. Finally, each polygon is shaded by linear interpolating of vertex intensities along each edge and then between edges along each scan line. This is illustrated in Fig. 15.14.

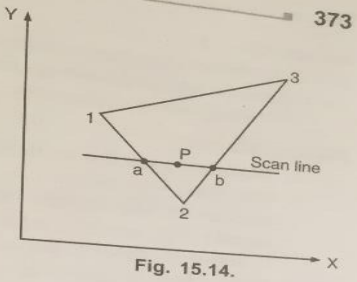


Fig. 15.14.

For each scan line, the intensity at the intersection of the scan line with a polygon edge is linearly interpolated from the intensities at the edge endpoints. *For example*, in Fig. 15.14, the polygon edge with endpoint vertices 1 and 2 is intersected by the scan line at point 'a'. The intensity at point 'a' can be interpolated from intensities I_1 and I_2 as

$$I_a = \frac{y_a - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_a}{y_1 - y_2} I_2$$

Similarly, we can interpolate the intensity value for right intersection (point b) from intensity values I_2, and I_3 as

$$I_b = \frac{y_a - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_b}{y_3 - y_2} I_2$$

Once the intensities of intersection points a and b are calculated for a scan line, the intensity of an interior point (such as P) can be determined as

$$I_p = \frac{x_b - x_p}{x_b - x_a} I_a + \frac{x_p - x_a}{x_b - x_a} I_b$$

Incremental calculations are used to obtain successive edge intensity values between scan lines and to obtain successive intensity along a scan line, this eliminates the repeatative calculations.

As shown in the figure 15.15, if the intensity at edge position (x, y) is interpolated as

$$I = \frac{(y - y_2)}{(y_1 - y_2)} I_1 + \frac{(y_1 - y)}{(y_1 - y_2)} I_2$$

then we can obtain the intensity along this edge for the next scan line, $y - 1$, as

$$I' = I + \frac{I_2 - I_1}{y - y_2}$$

Similarly, we can obtain intensities at successive horizontal pixel positions along each scan line as

$$I' = I + \frac{I_b - I_a}{x_b - x_a}$$

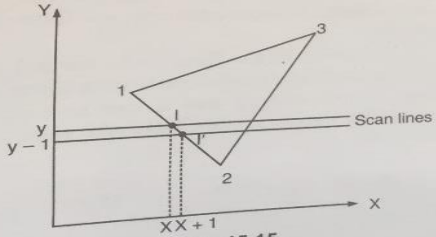


Fig. 15.15 .

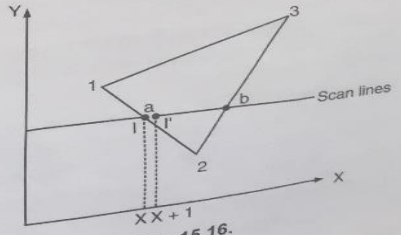


Fig. 15.16.

Advantages

1. Gouraud shading can be combined with a hidden surface algorithm to fill in the visible polygons along each scan line.
2. It removes the intensity discontinuities exists in constant shading model.

Disadvantages

1. Highlights on the surface are sometimes displayed with anomalous shapes.
2. The linear intensity interpolation can result bright or dark intensity streaks to appear on the surface. These bright or dark intensity streaks, are called **mach bands**. The mach band effect can be reduced by breaking the surface into a greater number of smaller polygons.
3. Sharp drop of intensity values on the polygon surface can not be displayed.

15.5.3. Phong Shading

A more accurate method for rendering a polygon surface is to interpolate normal vectors and then apply the illumination model to each surface point. This method, developed by Phong Bui Tuong, is called **Phong shading or normal-vector interpolation shading**, interpolates the surface normal vector N, instead of the intensity.

By performing following steps, we can display polygon surface using Phong shading.

1. Determine the average unit normal vector at each polygon vertex.
2. Linearly interpolate the vertex normals over the surface of the polygon.
3. Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

The first steps in the Phong shading is same as first step in the Gouraud shading. In the second step the vertex normals are linearly interpolated over the surface of the surface of the polygon.

As shown in the figure, the normal vector N for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Like, Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line. Once the surface normals are evaluated the surface intensity at the point is determined by applying the illumination model.

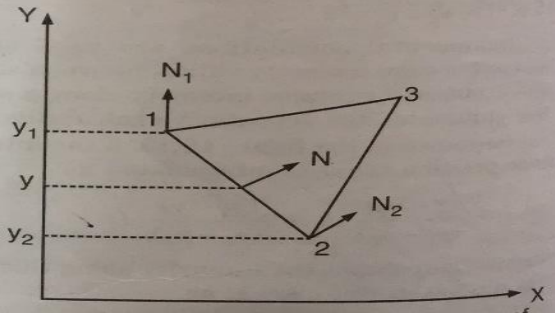


Fig. 15.17. Calculation of interpolation of surface normals along a polygon edge

Advantages

1. It displays more realistic highlights on a surface.
2. It greatly reduces the Mach-band effect.
3. It gives more accurate results.

Disadvantages

1. It requires more calculations and greatly increases the cost of shading steeply.

15.6. TRANSPARENCY

Illumination models have so far been used to determine intensity of light that is only reflected from surface of opaque objects. We did not consider transparent objects that allow some part of the incident light energy to transmit through it apart from reflecting a part from the surface.

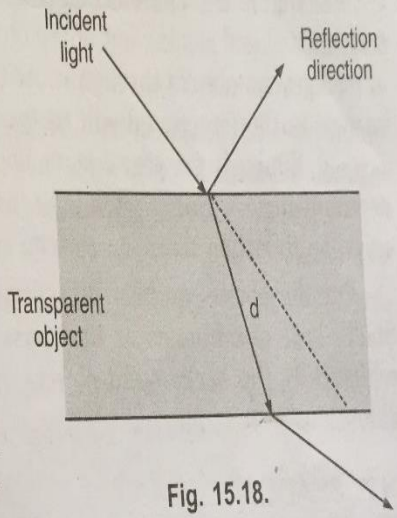


Fig. 15.18.

A transparent surface, in general, produces both reflected and transmitted light. It has a transparency coefficient T as well as values for reflectivity and specular reflection. The coefficient of transparency depends on the thickness of the object because the transmission of light depends exponentially on the distance which the light ray must travel within the object. The expression for coefficient of transparency is given as

$$T = t\, e^{-ad}$$

Where t is the coefficient of property of material which determines how much of the light is transmitted at the surface instead of reflected, a is the coefficient of property of material which tells how quickly the material absorbs or attenuates the light, d is the distance the light must travel in the object.

When light crosses the boundary between two media it changes the direction as shown in the Fig. 15.18 This effect is called **refraction**. The effect of refraction is observed because the speed of light is different in different materials resulting different path for refracted light

from that of incident light. The direction of the refracted light is specified by the **angle of refraction** (θ_r). It is the function of the property materials called the **index of refraction** (η). The angle of refraction θ_r is calculated from the angle of incidence θ_i, the index of refraction η_i of the incident material (usually air) and the index of refraction η_r of the refracting material according to **Snell's Law**.

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

Actually, the index of refraction a material is a function of the wave length of the incident light, so that the different colour components of a light ray refracts at different angles.

For modeling of transparent surface we have to consider contribution from the light reflected from the surface and the light coming from behind the object.

In the simplest form intensity of light (I) effectively reflected by

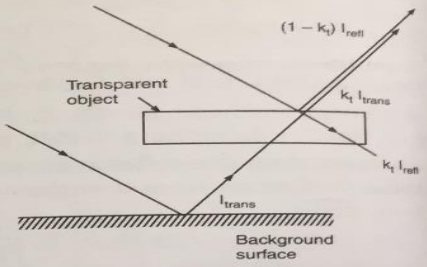


Fig. 15.19.

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

where I_{refl} is the intensity of light that is supposed to be reflected from the surface of the transparent object (due to ambient, diffuse and specular reflection) had it been an opaque object.

I_{trans} is intensity of light transmitted from a background source of object.

k_t is the ***transparency coefficient*** of the transparent object giving a measure of its transparency; $0 \le k_t \le 1$, $(1 - k_t)$ is the ***opacity factor*** carrying the opposite sense of k_t.

For highly transparent object $k_t \approx 1$, which implies almost all background light will be transmitted and coming out of the object's surface as $k_t\ I_{trans} \approx I_{trans}$ resulting clear visibility of background objects through it. At the same time most of the light incident on the object's surface in the foreground will be transmitted through instead of being reflected as $(1 - k_t)$ $I_{\text{refl}} \approx 0$. Similarly for near opaque objects $k_t \approx 0$ and transmits no light neither from behind nor from surface. So reflected light intensity is then just the resultant of ambient, diffuse & specular reflection from the object's surface.

For the above equation it is however assumed that light rays are not bent following Snell's laws of refraction as they pass through the transparent object. Otherwise and more realistically the background objects viewed through the transparent object would appear dimmer as well as blurred.

15.7. SHADOW

A shadowed object is one where light is blocked from reaching *i.e.*, the object which is hidden with respect to the light source. However, shadows are seen only when observer's position is not coincides with the light source. The shadowed areas are not visible from the light source position but are visible from observer's position. Such areas can be identified by applying the hidden surface detection methods with viewpoint assumed at the light source position.

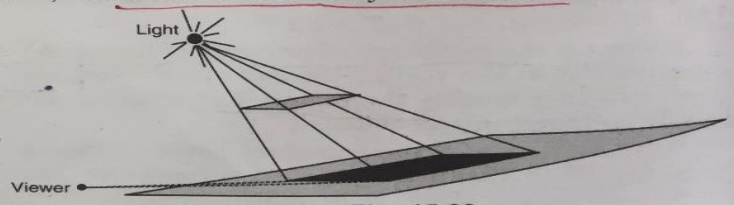


Fig. 15.20.

A shadowed object is one which is hidden from the light source. It is possible to use hidden surface algorithms to locate the areas where light sources produce shadows. In order to achieve this we have to repeat the hidden-surface calculation using light source as the viewpoint. This calculation divides the surfaces into shadowed and unshadowed groups. The surfaces that are visible from the light source are not in shadow; those that are not visible from the light source are in shadow. Surfaces which are visible and which are also visible from the light source are shown with both the background illumination and the light-source illumination. Surfaces which are visible but which are hidden from the light source are displayed with only the background illumination, as shown in the Fig. 15.20.

There are two types of shadows: ***shelf shadow*** and ***projected shadow***. Shelf-shadows are created on some of the object planes when the object itself occludes light from illuminating those planes. A projected shadow results on a surface when an intervening object blocks light from reaching the surface.

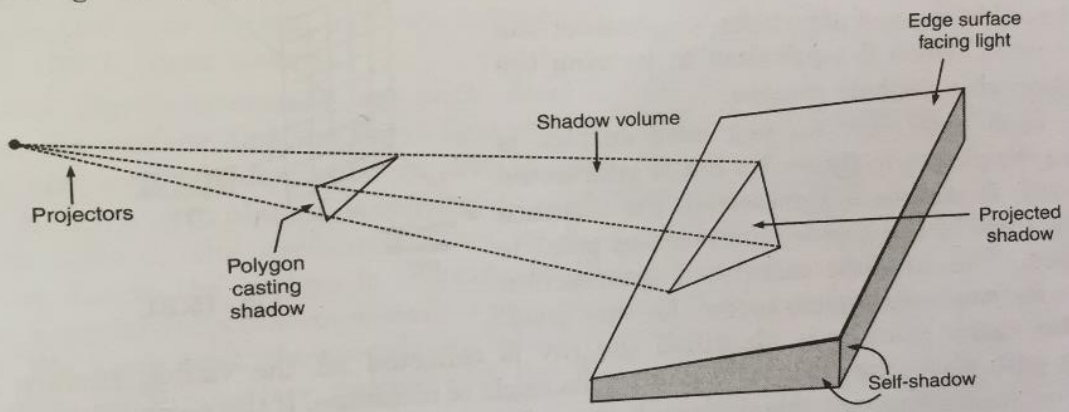


Fig. 15.21.

Another way to located shadow areas is the use of ***shadow volumes***. A shadow volume is defined by the light source and an object and is bounded by a set of invisible shadow polygons. This volume is also known as polygon's shadow volume. By comparing visible polygon with the volume, we can identify the portions which lie inside of the volume and which are outside of the volume. The portions which lie inside of the volume are shadowed, and their intensity calculations do not include a term from the light source, the polygons or portions of polygons which lie outside the shadow volume are not shaded by this polygon, but might be shaded by some other polygon. So they must be checked against the other shadow volume.