COMPUTER GRAPHICS (MAY 2018)

Q.P.Code: 21848

Q.1) Attempt any five from the following:- (20 M)

a) Explain CSG method for solid modelling. (5 M)

Ans:

- Another technique for solid modelling is to combine the volumes occupied by overlapping three-dimensional objects using Boolean set operations.
- This modelling technique is called Constructive Solid Geometry (CSG). It creates a new volume by applying Boolean operators such as union. intersection, or difference to two specified objects.
- The Fig. (10), Fig. (11), Fig. (12) show the example for forming new shapes using Boolean set operations The Fig. 10 (a) shows that two rectangular blocks are placed adjacent to each other. We can obtain the combined object with the union operation as shown in Fig. 10 (b).

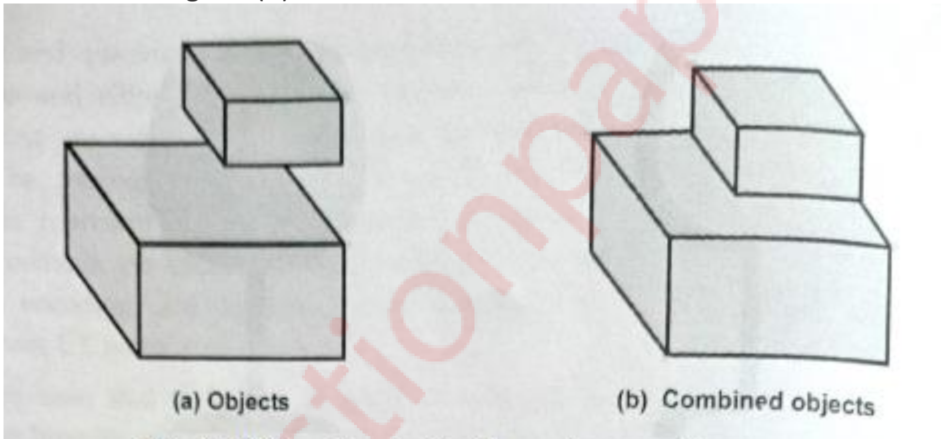


Fig. (10) Combined object by union operator.

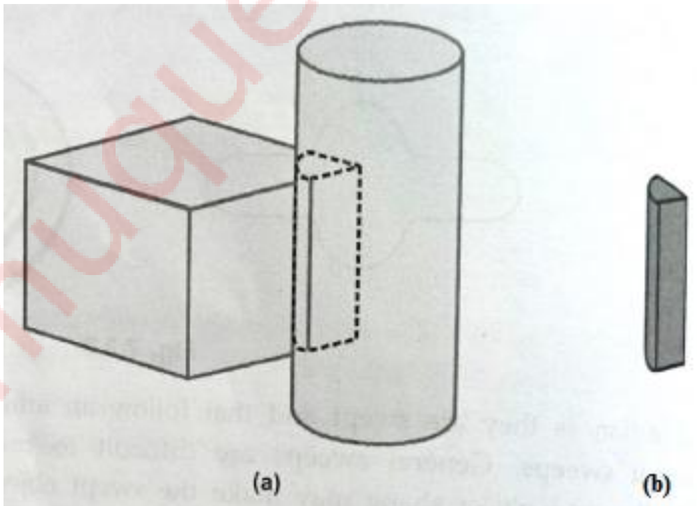


Fig. (11)

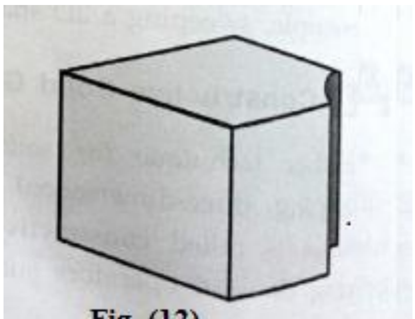


Fig. (12)

- The Fig.(11) shows the result of intersection operation obtained by overlapping cylinder and cube.
- With the difference operation, we can obtain the resulting solid as shown in Fig. (12).
- The CSG method Uses three dimensional objects such as blocks, pyramids, cylinders, cones, spheres, and closed spline surfaces to generate other solid objects In this method, an object is stored as a tree with operators at the internal nodes and simple primitives at the leaves.
- Some nodes represent Boolean operators, whereas others represent operations such as translation, rotation, and scaling. It is important to note that Boolean operations are not, in general, communicative Therefore the edges of the trees must be in proper order.

---

b) What is aliasing and Explain any one antialiasing method. (5 M)

Ans:

- In computer graphics, the process by which smooth curves and other lines become jagged because the resolution of the graphics device or file is not high enough to represent a smooth curve.
- In the line drawing algorithms, we have seen that all rasterized locations do not match with the true line and we have to select the optimum raster locations to represent a straight line. This problem is severe in low resolution screens. In such screens line appears like a stair-step, as shown in the figure below. This effect is known as aliasing. It is dominant for lines having gentle and sharp slopes.

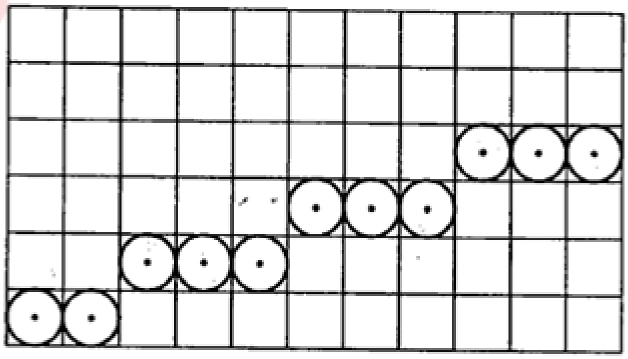


Fig. Aliasing effect

- The aliasing effect can be reduced by adjusting intensities of the pixels along the line. The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called antialiasing.
- The technique of anti-aliasing is:
- **Prefiltering:** This is technique that determines pixel intensity based on the amount of that particular pixel coverage by the object in the scene i.e. It computes pixel colours depending on the objects coverage.
- It means how much part or fraction of that pixel is covered by the object and depending on that, it sets the value of the pixel. It requires large number of calculations and approximations. Prefiltering generates more accurate antialiasing effect. But due to its high complexity of calculations it is not used.

---

c) Compare Raster Scan and Random Scan displays. (5 M)

Ans:

Random Scan	Raster Scan
1. It has high resolution.	1. Its resolution is low
2. It is more expensive.	2. It is less expensive.
3. Any modification if needed is easy	3. Modification is tough.
4. solid pattern is tough to fill.	4. Solid pattern is easy to fill.
5. Refresh rate depends on resolution.	5. Refresh rate does not depend on the picture.
6. Only screen with view on an area is displayed.	6. Whole screen is scanned.
7. Beam penetration technology comes under it.	7. Shadow mark technology comes under this.
8. It does not use interlacing method.	8. It uses interlacing.

---

d) Prove that two successive rotations are additive i.e. R1(Θ_1)*R2(Θ_2) = R(Θ_1+Θ_2). (5 M)

Ans:

Lets assume that $\Theta_1 = \Theta_2 = 90^0$

It means that first we will perform rotation of Θ_1 i.e. 90^0 in anticlockwise direction and then on that resultant rotation matrix we will perform again anticlockwise rotation by angle Θ_2

By performing two times of rotation by Θ_1 and Θ_2 we will get results as if roation by (Θ_{1+} Θ_2).

R(Θ_1) = $\begin{vmatrix} \cos 90 & \sin 90 \\ -\sin 90 & \cos 90 \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix}$

R(Θ_2) = $\begin{vmatrix} \cos 90 & \sin 90 \\ -\sin 90 & \cos 90 \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix}$

Now if R(Θ_1)* R(Θ_2)

then, $\begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} * \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} = \begin{vmatrix} -1 & 0 \\ 0 & -1 \end{vmatrix}$

now R($\Theta_1 + \Theta_2$) i.e. R($90^0 + 90^0$) = R(180^0)

hence, R(180^0) = $\begin{vmatrix} \cos 180 & \sin 180 \\ -\sin 180 & \cos 180 \end{vmatrix} = \begin{vmatrix} -1 & 0 \\ 0 & -1 \end{vmatrix}$

Hence, R1(Θ_1)*R2(Θ_2) = R(Θ_1+Θ_2).

Q.2)

a) Explain Bresenham line drawing algorithm with proper mathematical analysis and identify the pixel positions along a line between A(10,10) and B(18,16) using it. (10 M)

Ans:

- There is one more algorithm to generate lines which is called as Bresenham's line algorithm. The distance between the actual line and the nearest grid location is called error and it is denoted by G.
- The beauty of Bresenham's algorithm it is implemented entirely with integer numbers and the integer numbers are much more faster than floating-point arithmetic. But in previous section we have seen that we are checking slope with 0.5 i.e. floating point variable.
- Consider P as height of pixel or error. Our condition is whether P>0.5 or not,
  If $P > 0.5$ ……(1)

- But here 0.5 is floating point so we have to convert floating point to integer. For that we will multiply both sides by 2.
  $2P-1 > 0$ …….(2)
- In this equation we have removed fractional part 0.5. But it still it may contain fractional part from the denominator of slope. Because every time we are updating P by,
  $P = P + slope$ or $P = P+ slope-1$

  i.e. here we are adding slope to P.

  But slope is nothing but Dy/Dx.

- Now,
  $2PDx – Dx > 0$
- Now we will define $G = 2PDx – Dx$ …….(3)
- As $G = 2PDx – Dx$

  $G + Dx = 2PDx$

  $G + Dx / 2Dx = P$

- Now if we consider $P = P + slope$, then
  We will get $G + Dx / 2Dx = G + Dx / 2Dx + Dy/Dx$
  $G = G + 2Dy$ …….(4)
- Similarly if we have $P = P + slope – 1$
  We will get $G = G + 2Dy – 2Dx$ …….(5)
- So, we can use test $G > 0$ to determine when a row boundary Is crossed by a line instead of $P > 0.5$
- Put initial value of P as Dy/Dx to find initial value of G.
  $G = 2Dx (Dy / Dx) – Dx$
  $G = 2Dy – Dx$
  - For each column we check G. If it is we move to the next row and add (2 Dy – 2 Dx) to G, because it is equivalent to P = P + slope -1, otherwise we will keep the same row and add (2 Dy) to G.

  - Given A(10, 10) B(18, 16)

    A(x1, y1) = (10, 10)
    B(x2, y2) = (18, 16)
    Dx = x2-x1 = 18 – 10 = 8
    Dy = y2-y1 = 16 – 10 = 6
    Now, lets find decision factor G,
    G = 2Dy – Dx = 2(6) – 8 = 4
    First plot point (18, 16)
    Now increase x by 1

Hence, x = 19
Here G = 4 i.e. positive so increase y by 1 and update G
As, G = G+2(Dy - Dx)
= 4+2(6 - 8) = 0
Now plot point (19, 17)
Similarly finding different values,

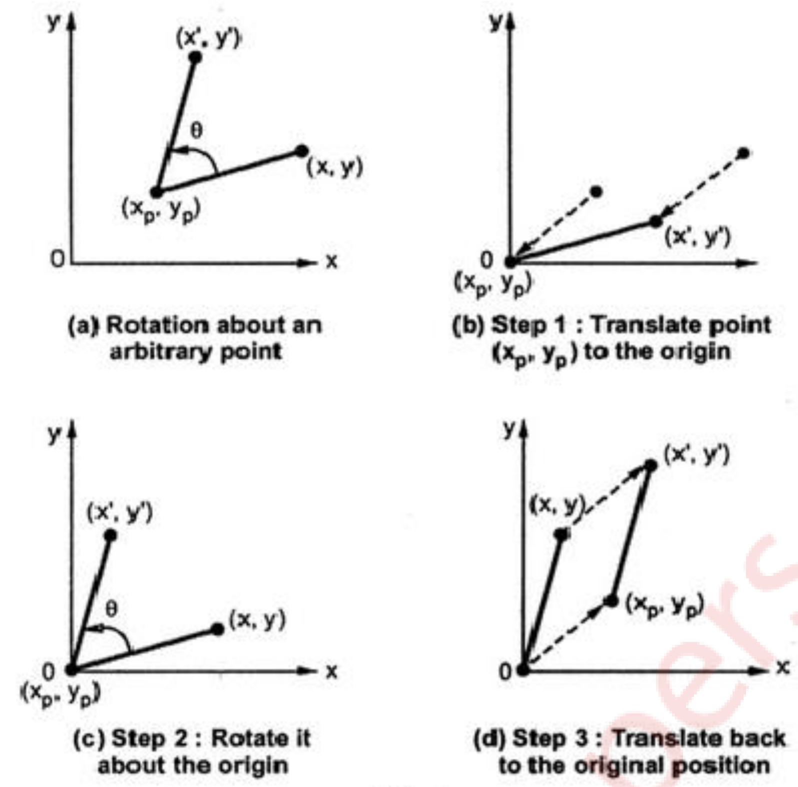
A	B
18	16
19	17
20	18
21	18
22	19
23	20
24	21
25	21
26	22
27	23
28	24

---

b) Explain the steps for 2D rotation about arbitrary point and provide a composite transformation for the same. (10M)

Ans:

- **Rotation about an Arbitrary Point:-**
- To rotate an object about an arbitrary point, (Xp ,Yp) we have to carry out three steps:
    - Translate point (Xp, Yp) to the origin.
    - Rotate it about the origin and,
    - Finally, translate the centre of rotation back where it belongs (See figure 1.).
- we have already seen that matrix multiplication is not commutative, i.e. multiplying matrix A by matrix B will not always yield the same result as multiplying matrix B by matrix A. Therefore, in obtaining composite transformation matrix, we must be careful to order the matrices so that they correspond to the order of the transformations on the object. Let us find the transformation matrices to carry out individual steps.



y
(x', y')
θ
(x, y)
(x_p, y_p)
0
x
(a) Rotation about an arbitrary point
y
0
(x_p, y_p)
(x', y')
(b) Step 1 : Translate point (x_p, y_p) to the origin
y
(x', y')
θ
(x, y)
0
(x_p, y_p)
x
(c) Step 2 : Rotate it about the origin
y
(x', y')
(x, y)
(x_p, y_p)
0
x
(d) Step 3 : Translate back to the original position

Fig. 1

The translation matrix to move point (x_p, y_p) to the origin is given as, x_p

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\mathrm{xp} & -yp & 1 \end{bmatrix}$$

The rotation matrix for counterclockwise rotation of point about the origin is given as,

$$R = \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The translation matrix to move the center point back to its original position is given as,

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xp & yp & 1 \end{bmatrix}$$

Therefore the overall transformation matrix for a counterclockwise roatation by an angle Θ about the point (xp, yp) is given as,

$$T_1{*}R{*}T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xp & -yp & 1 \end{bmatrix} * \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xp & yp & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ -xpcos\Theta + \mathrm{ypsin}\Theta & -xpsin\Theta - \mathrm{ypcos}\Theta & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xp & yp & 1 \end{bmatrix}$$

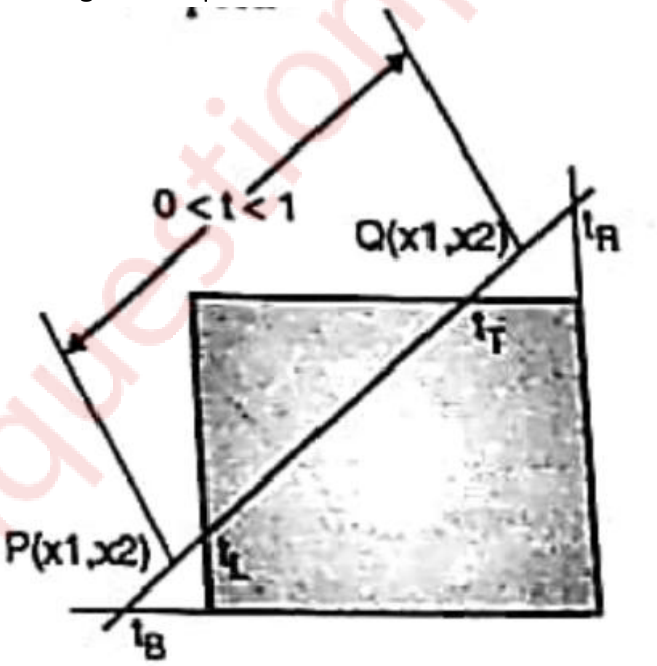
$$= \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ -xpcos\Theta + \text{ypsin}\Theta + \text{xp} & -xpsin\Theta - \text{ypcos}\Theta + \text{yp} & 1 \end{bmatrix}$$

Q.3)

a) Explain Liang Barsky line clipping algorithm. Apply the algorithm to clip the line with coordinates(30, 60) and (60, 20) against window (xmin, ymin) = (10, 10) and (xmax, ymax) = (50, 50). (10 M)

Ans:

- The Liang-Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping box to determine the intersections between the line and clipping box. With these intersections it knows which portion of the line should be drawn. This algorithm is more efficient than Cohen-Sutherland. The ideas for clipping line of Liang-Barsky and cyrus-Beck are the same. The only difference is Liang-Barsky algorithm has been optimized for an upright rectangular clip window. So we will study only the idea of Liang-Barsky.
- Liang and Barsky have created an algorithm that uses floating-point arithmetic but finds the appropriate ends points with at most four computations. This algorithm uses the parametric equations for a line and solves for inequalities to find the range of the parameter for which the line is in the viewport.



- Let P(X1, Y1), Q(X2, Y2) be the line which we want to study. The parametric equation of line segment from gives x-values and y-values for every point in terms of a parameter that ranges from 0 to 1. The equations are

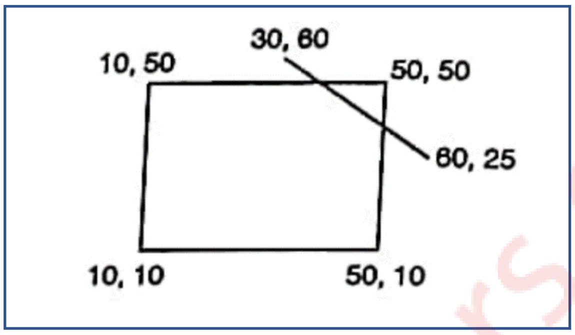
  X = X1 + (X2 – X1)*t = Y1 + dY*t

  And

Y = Y1 + (Y2 – Y1)*t = X1 + dX*t

We can see that when t=0, the point computed is P(x1, y1) and when t =1, the point computed is Q(x2, y2).

- Lets first draw the window and line as shown in fig.



Given things are XL = 10, YB = 10, XR = 50, YT = 50

Lets call a line AB with its coordinates as X1 = 30, Y1 = 60, X2 = 60, Y2 = 25

Now we have to find Dx and Dy as

Dx = X2 – X1 = 60 – 30 = 30

Dy = Y2 – Y1 = 25 – 60 = -35

Lets calculate the values of parameters P and Q as

P1 = -Dx = -30

P2 = Dx = 30

P3 = -Dy = -(-35) = 35

P4 = Dy = -35

Now,

Q1 = X1 – XL = 30 -10 = 20

Q2 = XR – X2 = 50 -30 = 20

Q1 = Y1 – YB = 30 -10 = 50

Q1 = YT – Y1 = 30 -10 = -10

Now lets find P,

P1 = Q1/P1 = 20/(-30) = (-2/3)

P2 = Q2/P2 = 20/(30) = (2/3)

P3 = Q3/P3 = 50/(35) = (20/7)

P1 = Q4/P4 = -10/(-35) = (2/7)

Now,

t1 = Max(-2/3, 10/7, 0) = 10/7

t2 = Min(2/3, 2/7, 1) = 2/7

Now,

X1' = X1 + Dx*t1

= 30+30*(10/7)

= 72.71

Y1' = Y1 + Dy*t1

= 60+(-35)*(10/7)

= 10

And with t2 it will be

X2' = X1 + Dx*t2

= 30+30*(2/7)

= 38.57

Y2' = Y1 + Dy*t2

= 60+(-35)*(2/7)

= 50

From this we will come to know that a point (38.57, 50) is an intersection point with respect to the edge of the window boundary. So we need to discard the line from point (30, 60) to (38.57, 50) and consider line (38.57, 50) to (60, 25).

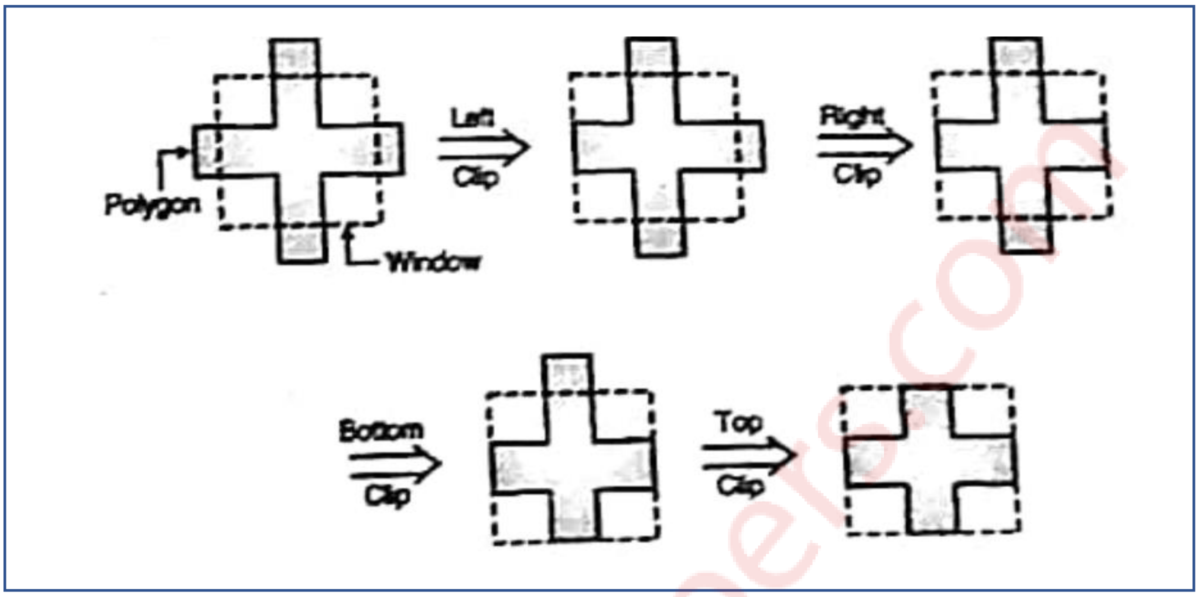
---

b) Explain Sutherland Hodgman polygon clipping algorithm with suitable example and comment on its shortcoming. (10 M)

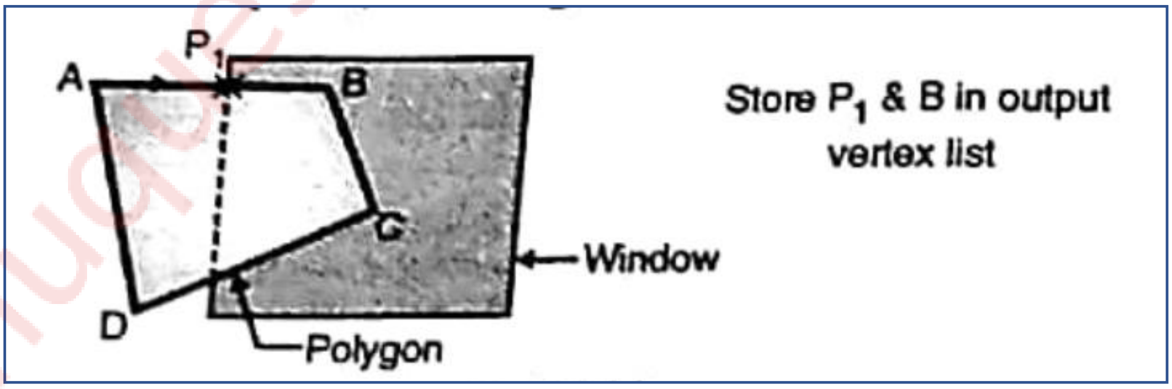
Ans:

- We can clip a polygon by clipping whole polygon against each boundary edge of the window. We know that to represent a polygon we need a set of vertices.
- This left clip procedure generates new set of vertices which indicates left clipped polygon. Against this new set of vertices is passed to the right boundary clipper

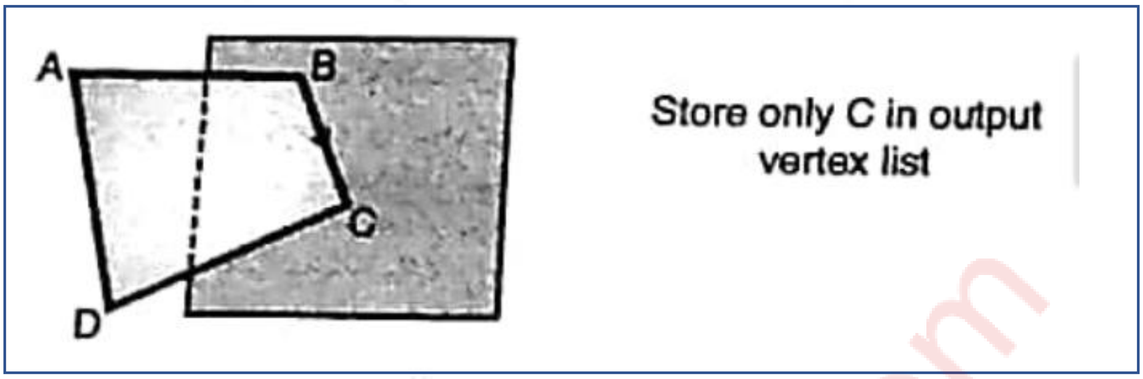
procedure. Again we will get new set of vertices. Then we will pass this new set of vertices to bottom boundary clipper and lastly to top boundary clipper procedure.



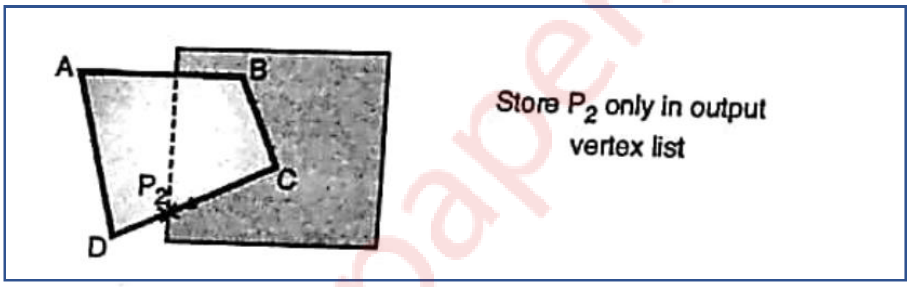
- At the end of every a new set of vertices is generated and this new set or modified polygon is passed to the next clipping stage.
- After clipping a polygon with respect to all four boundaries we will get final clipped polygon.
- When we clip a polygon w.r.t any particular edge of the window four different cases, as each pair of adjacent polygon vertices is passed to a window boundary clipper.
- **Case 1:** If the first vertex is outside the window boundary and the second vertex is inside the window, then the intersection point of polygon with boundary edge of window and the vertex which is inside the window is stored in a output vertex list.



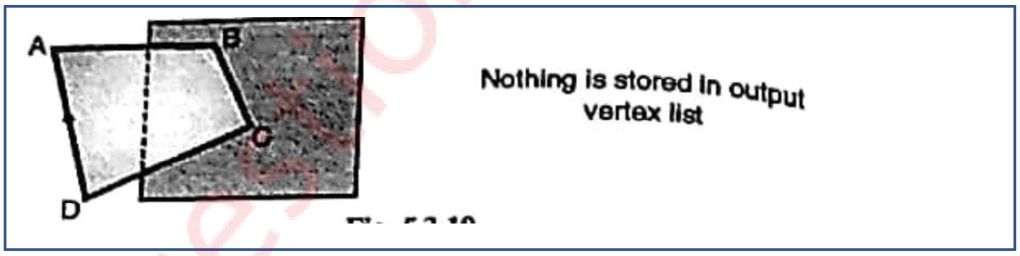
- For edge AB instead of storing vertex A and B are storing P_1 and B in output vertex list.
- **Case 2:** If both, first and second vertices of a polygon are lying inside the window, then we have to store the second vertex only in output vertex list.



- **Case 3:** If the first vertex is inside the window and second vertex is outside the window i.e. opposite to case 1, then we have to store only intersection point of that edge of polygon with window in output vertex list.



- **Case 4:** If both first and second vertices of a polygon are lying outside the window then no vertex is stored in vertex list.



- Once all vertices have been considered for one clip window boundary, the output list of vertices is clipped against the next window boundary.

Q.4)

a) **What is window and viewport? Derive the window to viewport transformation and also identify the geometric transformation involved. (10 M)**

Ans:

Window:

1. A world-coordinate area selected for display is called a window.
2. In computer graphics, a window is a graphical control element.
3. It consists of a visual area containing some of the graphical user interface of the program it belongs to and is framed by a window decoration.
4. A window defines a rectangular area in world coordinates. You define a window with a GWINDOW statement. You can define the window to be larger than, the same size as, or smaller than the actual range of data values, depending on whether you want to show all of the data or only part of the data.

Viewport:

1. An area on a display device to which a window is mapped is called a viewport.
2. A viewport is a polygon viewing region in computer graphics. The viewport is an area expressed in rendering-device-specific coordinates, e.g. pixels for screen coordinates, in which the objects of interest are going to be rendered.
3. A viewport defines in normalized coordinates a rectangular area on the display device where the image of the data appears. You define a viewport with the GPORT command. You can have your graph take up the entire display device or show it in only a portion, say the upper-right part.

Window to viewport transformation:

1. Window-to-Viewport transformation is the process of transforming a two-dimensional, world-coordinate scene to device coordinates.
2. In particular, objects inside the world or clipping window are mapped to the viewport. The viewport is displayed in the interface window on the screen.
3. In other words, the clipping window is used to select the part of the scene that is to be displayed. The viewport then positions the scene on the output device.

1. Using this proportionality, the following ratios must be equal.

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

By solving these equations for the unknown viewport position (xv, yv), the following becomes true:

$$xv = S_x xw + t_x$$

$$yv = S_y yw + t_y$$

The scale factors (Sx, Sy) would be:

$$S_x = \frac{xv_{min} - xv_{min}}{xw_{max} - xw_{min}}$$

$$S_y = \frac{yv_{min} - yv_{min}}{yw_{max} - yw_{min}}$$

And the translation factors (Tx, Ty) would be:

$$t_x = \frac{xw_{max}xv_{min} - xw_{min}xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max}yv_{min} - yw_{min}yv_{max}}{yw_{max} - yw_{min}}$$

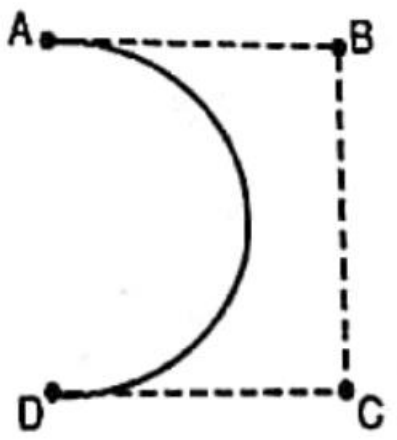
1. The position of the viewport can be changed allowing objects to be viewed at different positions on the Interface Window.
2. Multiple viewports can also be used to display different sections of a scene at different screen positions. Also, by changing the dimensions of the viewport, the size and proportions of the objects being displayed can be manipulated.
3. Thus, a zooming affect can be achieved by successively mapping different dimensioned clipping windows on a fixed sized viewport.
4. If the aspect ratio of the world window and the viewport are different, then the image may look distorted.

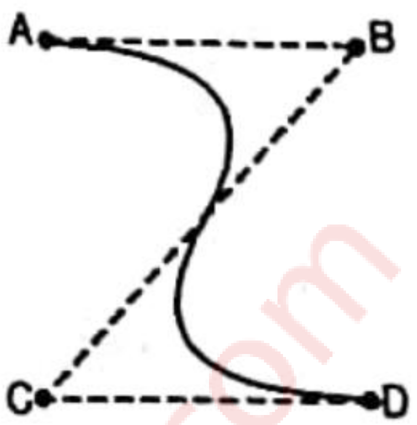
---

b) Explain what is meant by Bezier curve? State the various properties of Bezier curve. (10 M)

Ans:

- It is a different way of specifying a curve, rather same shapes can be represented by B-spline and Bezier curves. The cubic Bezier curve require four sample points, these points completely specify the curve.





- The curve begins at the first sample point and ends at fourth point. If we need another Bezier curve then we need another four sample points. But if we need two Bezier curves connected to each other, then with six sample points we can achieve it. For this, the third and fourth point of first curve should be made same as first and second point of curve.
- The equation for the Bezier curve are as follows:

$$X = X_4a^3 + 3X_3a^2(1-a) + 3X_2a(1-a)^2 + X_1(1-a)^3$$

$$Y = Y_4a^3 + 3Y_3a^2(1-a) + 3Y_2a(1-a)^2 + Y_1(1-a)^3$$

$$Z = Z_4a^3 + 3Z_3a^2(1-a) + 3Z_2a(1-a)^2 + Z_1(1-a)^3$$

- Here as the value of 'a' moves from 0 to 1, the curve travels from the first to fourth sample point. But we can construct a Bezier curve without referencing to the above expression. It is constructed by simply taking midpoints.
- Properties of Bezier curve are as follows:
  1. The basic function are real in nature.
  2. Bezier curve always passes through the first and last control points i.e. curve has same end points as the guiding polygon.
  3. The degree of polynomial defining the curve segment is one less than the number of defining polygon point.
  4. The curve generally follows the shape of the defining polygon.
  5. The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segments.
  6. The curve lies entirely within the convex hull formed by four control points.
  7. The curve exhibits the variation diminishing property. This means that the curve does not oscillate about any straight line more than the defining polygon.
  8. The curve is invariant under an affine transformation.

Q.5)

a) What is meant by parallel and perspective projection? Derive matrix for oblique projection. (10 M)

Ans:

Parallel Projection:

i. In parallel projection, Z coordinate is discarded and parallel lines from each vertex on the object are extended until they intersect the view plane.

ii. The point of intersection is the projection of the vertex.

iii. We connect the projected vertices by line segments which correspond to connections on the original object.

iv. A parallel projection preserves relative proportions of objects.

v. Accurate views of the various sides of an object are obtained with a parallel projection. But not a realistic representation.

vi. Parallel projection is shown below in figure 30.

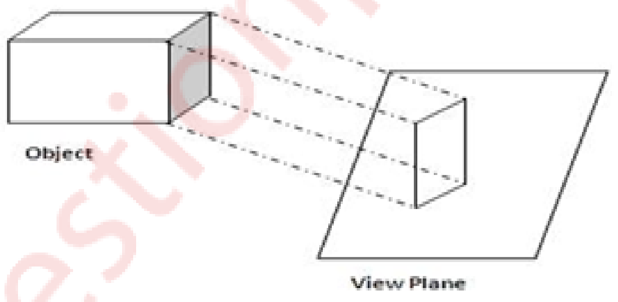


Figure 30

Perspective Projection:

i. In perspective projection, the lines of projection are not parallel.

ii. Perspective Projection transforms object positions to the view plane while converging to a center point of projection.

iii. In this all the projections are converge at a single point called the "center of projection" or "projection reference point".

iv. Perspective projection produces realistic views but does not preserve relative proportions.

v. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.

vi. Perspective projection is shown below in figure 31.

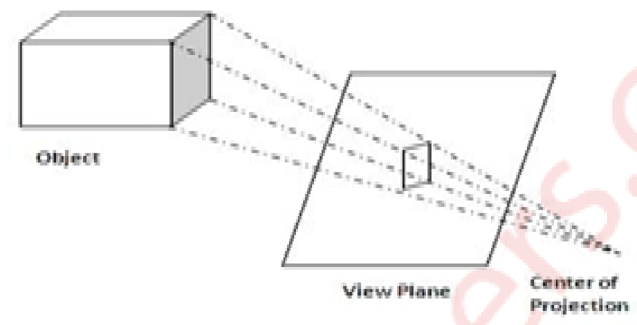
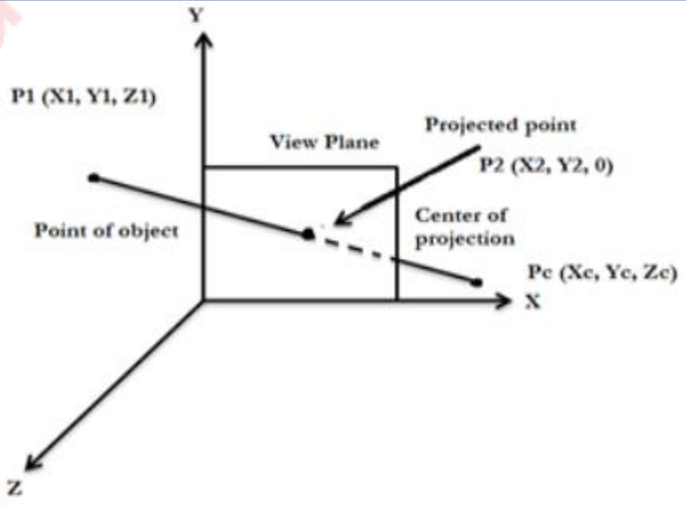


Figure 31

Matrix for perspective projection:

Let us consider the center of projection is at Pc(Xc,Yc,Zc)Pc(Xc,Yc,Zc) and the point on object is P1(X1,Y1,Z1)P1(X1,Y1,Z1), then the parametric equation for the line containing these points can be given as

$$X_2 = X_c + (X_1 - X_c)U$$
$$Y_2 = Y_c + (Y_1 - Y_c)U$$
$$Z_2 = Z_c + (Z_1 - Z_c)U$$



For projected point Z2 is 0, therefore the third equation can be written as

$0 = Z_C + (Z_1 - Z_C)U$

$U = -Z_C/Z_1 – Z_2$

Substituting the value of U in first two equations we get,

$X_2 = (X_C - Z_C) * (X_1 - X_C) / (Z_1 - Z_C)$

$= X_CZ_1 – X_CZ_C – X_1Z_C + X_CZ_C/Z_1 – Z_C$

$= X_CZ_1 – X_1Z_C / Z_1 – Z_C$

$Y_2 = (Y_C - Z_C) * (Y_1 - Y_C) / (Z_1 - Z_C)$

$= Y_CZ_1 – Y_CZ_C – Y_1Z_C + Y_CZ_C / Z_1 - Z_C$

The above equation can be represeented in the homogeneous matrix form as given below,

$$[X_2\ Y_2\ Z_2\ 1] = [X_1\ Y_1\ Z_1\ 1]\begin{bmatrix} -ZC & 0 & 0 & 0 \\ 0 & -ZC & 0 & 0 \\ XC & YC & 0 & 1 \\ 0 & 0 & 0 & -ZC \end{bmatrix}\begin{bmatrix} -ZC & 0 & 0 & 0 \\ 0 & -ZC & 0 & 0 \\ XC & YC & 0 & 1 \\ 0 & 0 & 0 & -ZC \end{bmatrix}$$

Here, we have taken the center of projection as PC(XC, YC, ZC), if we take the center of projection on the negative Z axis such that

X = 0

Y = 0

$Z = -Z_C$

---

b) Explain Z buffer algorithm for hidden surface removal. (10 M)

Ans:

- Another way to handle hidden lines and surfaces is z buffers. It is also called as depth buffer algorithm. Here we are sorting the polygons according to their position in space. And then in frame buffer itself. We are storing polygons which are closer to viewer. We know that frame buffer is used to store the images which we want to display on monitor. Here for visibility test we are making use of z buffer along with frame buffer.
- The z buffer is a large array to hold all the pixels off display. Z buffer is somewhat similar to frame buffer. In frame buffer we are having arrays to store x and y co-ordinates of an image. Similarly z buffer contains z co-ordinates of pixels which we want to display. It helps to sort the polygons by comparing their z position. In

simple terms it keeps track of nearest z coordinate of those points which are seen from the pixel (x, y).

- When there is nothing to display on monitor i.e. frame buffer is empty, at that time we have to initialize z buffer elements to a v. A large negative value on z axis represents a point beyond which there is nothing i.e. setting background color.
  $Z_{buffer}(x, y) = z_{initial}$
- Polygons will be entered one by one into frame buffer by the display file interpreter. During this process, for each pixel (x, y) that lies inside the polygon we have to calculate z (x, y) for the pixel (x, y). Then we have to compare the z position of the polygon point with the z_{buffer} (x, y) value and decide whether the new surface is in front of or behind the current contents of frame buffer.
- If the new surface has z value greater than z_{buffer} value, then it lies in front. So we have to modify the contents of z_{buffer} (x, y) by new z value and set the pixel value at (x, y) to the color of the polygon at (x, y).

```
i.e. if($z(x,y)>z_{buffer}(x,y)$)
{
        $z_{buffer}(x,y) = z(x,y)$
        put pixel (x,y,polygon-colour)
}
```

- If the z value of new surface is smaller than z_{buffer} (x, y) then it lies behind some polygon which was previously entered. So the new surface should be hidden and should not be displayed. The frame buffer and z_{buffer} should not be modified here. Here the comparison should be carried out by using pixel by pixel method.

Advantages:

- It is easy to implement.
- As z buffer algorithm processes one object at a time, total number of objects can be large.
- It does not require any additional technique.

Disadvantages:

- It requires lots of memory as we are storing each pixel's z value.
- It is a time consuming process as we are comparing each and every pixel.

Q.6) Write short notes on **(20 M)**

a) Koch Curve: **(5 M)**

Ans:

- The Koch curve can be drawn by dividing line into 4 equal segments with scaling factor 1/3 and middle two segments are so adjusted that they form adjacent sides of an equilateral triangle as shown in the Fig. 16(a) .This is the first approximation to the koch curve.
- To apply the second approximation to the Koch curve we have to repeat the above process for each of the four segments. The resultant curve is shown in Fig. 16(b).

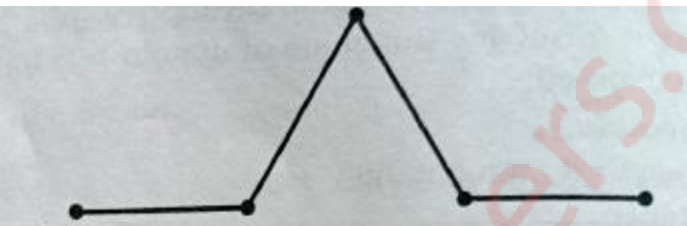


Fig. 16(a) First approximation of the koch curve

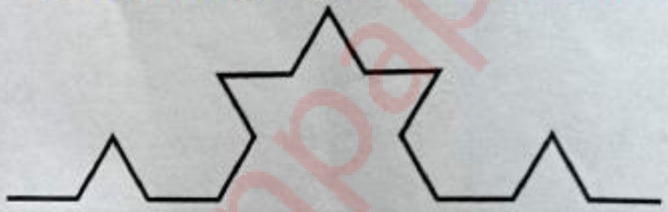


Fig. 16(b) The second approximation to koch curve

- The resultant curve has more wiggles and its length is 16/9 times the original length.
- From the above figures we can easily note following points about the koch curve :
  - Each repetition increases the length of the curve by factor 4/3.
  - Length of curve is infinite.
  - Unlike Hibert's curve, it doesn't fill an area.
  - It doesn't deviate much from its original shape.
  - If we reduce the scale of the curve by 3 we find the curve that looks just like the original one; but we must assemble 4 such curves to make the originals, so we have

$$4 = 3^D$$

Solving for D we get

$$D = \log_3 4 = \log 4 / \log 3 = 1.2618$$

- Therefore for koch curve topological dimension is 1 but fractal dimension is 1.2618.
- From the above discussion we can say that point sets, curves and surfaces which give a fractal dimension greater than the topological dimension are called fractals The Hilbert's curve and koch curves are fractals, because their fractal dimensions (respectively, 2 and 1.2618) are greater than their topological dimension which is 1.

b) Sweep representation and Octree representation (5 M)

Ans:

- Sweep representations are used to construct three dimensional objects from two dimensional shape .There are two ways to achieve sweep: Translational sweep and Rotational sweep. In translational sweeps, the 2D shape is swept along a linear path normal to the plane of the area to construct three dimensional object. To obtain the wireframe representation we have to replicate the 2D shape and draw a set of connecting lines in the direction of shape, as shown in the figure (8)

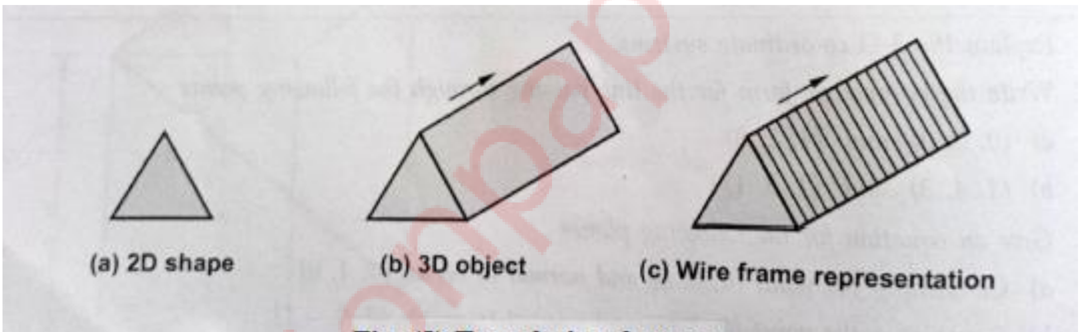


Fig. (8) Translational sweep

- In general we can specify sweep constructions using any path. For translation we can vary the shape or size of the original 2D shape along the sweep path. For rotational sweeps, we can move along a circular path through any angular distance from 0° to 360°.
- These sweeps whose generating area or volume changes in size, shape or orientation as they are swept and that follow an arbitrary curved trajectory are called general sweeps .General sweeps are difficult to model efficiently for example, the trajectory and object shape may make the swept object intersect itself, making volume calculations complicated.
- **Octree Representation:** Octrees are the hierarchical structure used to represent solid objects in some graphics systems. The tree structure of octrees is organized so that each node corresponds to a region of three dimensional space.
- Octrees are in turn derived from quad-trees, an encoding scheme used for representing two dimensional images. The fundamental idea behind both the quad-tree and octree is the divide and conquer power of binary subdivision. A quad-tree is derived successively dividing a 2D plane into quadrants. After first

subdivision, each node in the quad-tree has four data elements, one for each of the quadrants in the region. These quadrants can be of two types homogeneous and heterogeneous.

- The quadrant is said to be homogeneous quadrant the colour information for that quadrant is stored in the corresponding data element of the node. In addition, a flag is set in the data element to indicate that the quadrant is homogeneous.
- The heterogeneous quadrant contains pixels of different colours. If some of them are heterogeneous then they are again subdivided and the process is repeated until all quadrants are homogeneous or until a predetermined cut-off depth is reached.

---

c) Gouraud and phong shading (5 M)

Ans:

Gouraud Shading:

1. Gouraud surface shading was developed in the 1970s by Henri Gouraud.
2. It is the interpolation technique.
3. Intensity levels are calculated at each vertex and interpolated across the surface.
4. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges.
5. This eliminates the intensity discontinuities that can occur in flat shading.
6. To render a polygon, Gouraud surface rendering proceeds as follows:
    - Determine the average unit normal vector at each vertex of the polygon.
    - Apply an illumination model at each polygon vertex to obtain the light intensity at that position.
    - Linearly interpolate the vertex intensities over the projected area of the polygon

Illumination values are linearly interpolated across each scan-line as shown in figure 41.

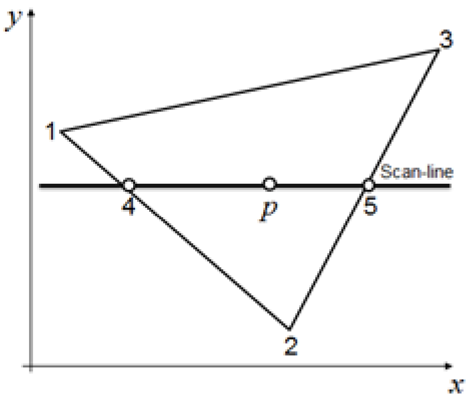


Figure 41

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Phong Shading:

1. A more accurate interpolation based approach for rendering a polygon was developed by Phong Bui Tuong.
2. Basically the Phong surface rendering model is also called as normal-vector interpolation rendering.
3. It interpolates normal vectors instead of intensity values.
4. To render a polygon, Phong surface rendering proceeds as follows:
5. Determine the average unit normal vector at each vertex of the polygon.
6. Linearly interpolate the vertex normal over the projected area of the polygon.
7. Apply an illumination model at positions along scan lines to calculate pixel intensities using the interpolated normal vectors as shown in figure 42

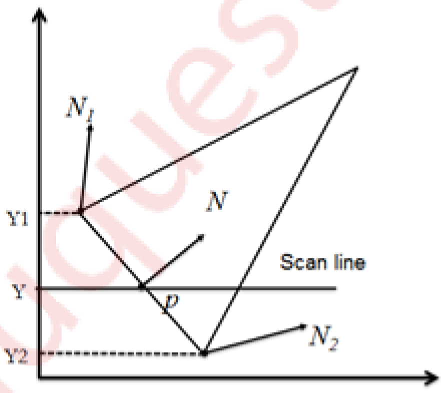


Figure 42

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

d) Halftoning and Dithering. (5 M)

Ans:

Half toning

1. Many displays and hardcopy devices are bi-level
2. They can only produce two intensity levels.
3. In such displays or hardcopy devices we can create an apparent increase in the number of available intensity value.
4. When we view a very small area from a sufficient large viewing distance, our eyes average fine details within the small area and record only the overall intensity of the area.
5. The phenomenon of apparent increase in the number of available intensities by considering combine intensity of multiple pixels is known as half toning.
6. The half toning is commonly used in printing black and white photographs in newspapers, magazines and books.
7. The pictures produced by half toning process are called halftones.
8. In computer graphics, halftone reproductions are approximated using rectangular pixel regions, say 2*2 pixels or 3*3 pixels.
9. These regions are called halftone patterns or pixel patterns.

Dithering Techniques

- Dithering refers to techniques for approximating halftones without reducing resolution, as pixel grid patterns do.
- The term dithering is also applied to halftone approximation method using pixel grid, and something it is used to refer to color halftone approximations only.
- Random values added to pixel intensities to break up contours are often referred as dither noise.
- Number of methods is used to generate intensity variations.
- Ordered dither methods generate intensity variations with a one-to-one mapping of points in a scene to the display pixel.
- To obtain n2n2 intensity levels, it is necessary to set up an n*n dither matrix DnDn whose elements are distinct positive integers in the range of 0 to n2−1n2−1.