COMPUTER GRAPHICS (MAY 2019)

Q.P.Code: 21849

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

a) What is aliasing antialiasing (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.

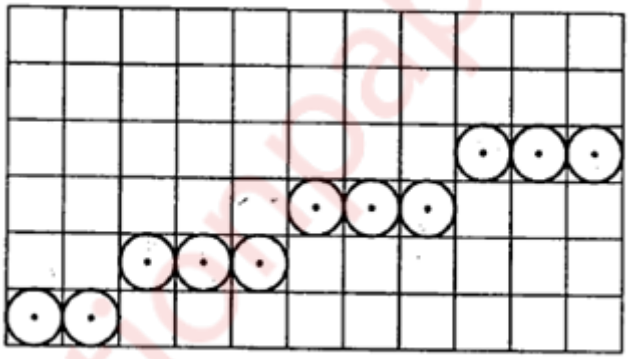


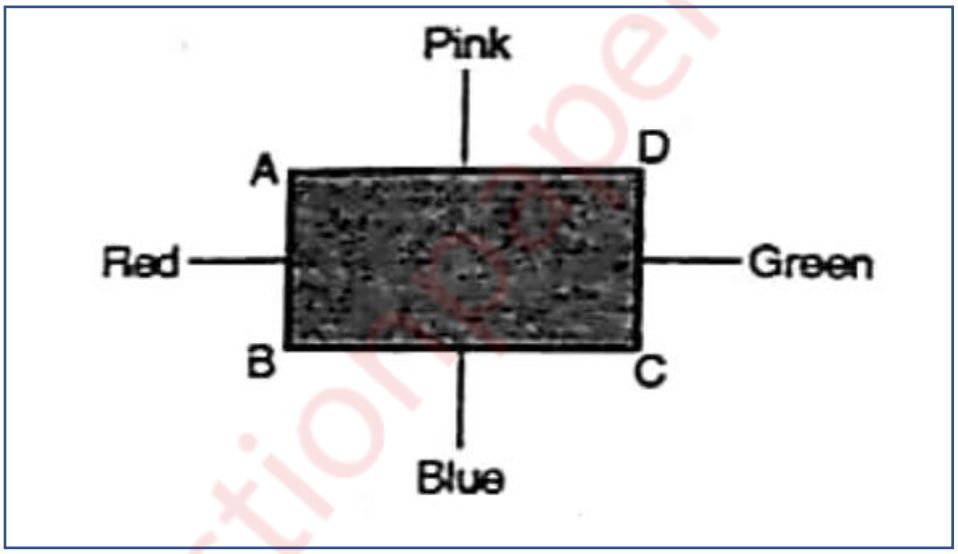
Fig. Aliasing effect

- 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.
- **Antialiasing** is a term for techniques that are designed to mitigate the effects of aliasing.
- The idea is that when a pixel is only partially covered by a shape, the colour of the pixel should be a mixture of the colour of the shape and the colour of the background.
- When drawing a black line on a white background, the colour of a partially covered pixel would be grey, with the shade of grey depending on the fraction of the pixel that is covered by the line.

b) Write the flood fill approach for 8 connected method. (5 M)

Ans:

- The limitations of boundary fill algorithm are overcome in flood fill algorithm. Like boundary fill algorithm this algorithm also begins with seed point which must be surely inside the polygon.
- Now instead of checking the boundary colour this algorithm checks whether the pixel is having the polygons original colour i.e. previous or old colour.
- If yes, then fill that pixel with new colour and uses each of the pixels neighbouring pixel as a new seed in a recursive call. If the answer is no i.e. the colour of pixel is already changed then return to its caller.
- Sometimes we want to fill an area that is not defines within a single colour boundary. Here edge AB, BC, CD, DA are having red, blue, green and pink colour, respectively.



- We can paint such areas by replacing a specified interior colour instead of searching for a boundary colour value. Here we are setting empty pixel with new colour till we get any coloured pixel.
- Flood fill and boundary fill algorithms are somewhat similar. A flood fill algorithm is particularly useful when the region or polygon has no uninformed coloured boundaries.
- The flood fill algorithm is sometimes also called as seed fill algorithm or forest fill algorithm. Because it spreads from a single point i.e. seed point in all direction.
- The following procedure illustrates a recursive method for flood fill by sing 8-connected method.

```
f-fill (X, Y, newcolour)
{
    Current = getpixel(X, Y);
    If(Current != newcolor)
    {
```

```
                Putpixel(X, Y, newcolour);
                f-fill(X-1, Y, newcolour);
                f-fill(X+1, Y, newcolour);
                f-fill(X, Y-1, newcolour);
                f-fill(X, Y+1, newcolour);
                f-fill(X+1, Y+1, newcolour);
                f-fill(X-1, Y+1, newcolour);
                f-fill(X-1, Y-1, newcolour);
                f-fill(X+1, Y-1, newcolour);
        }
}
```

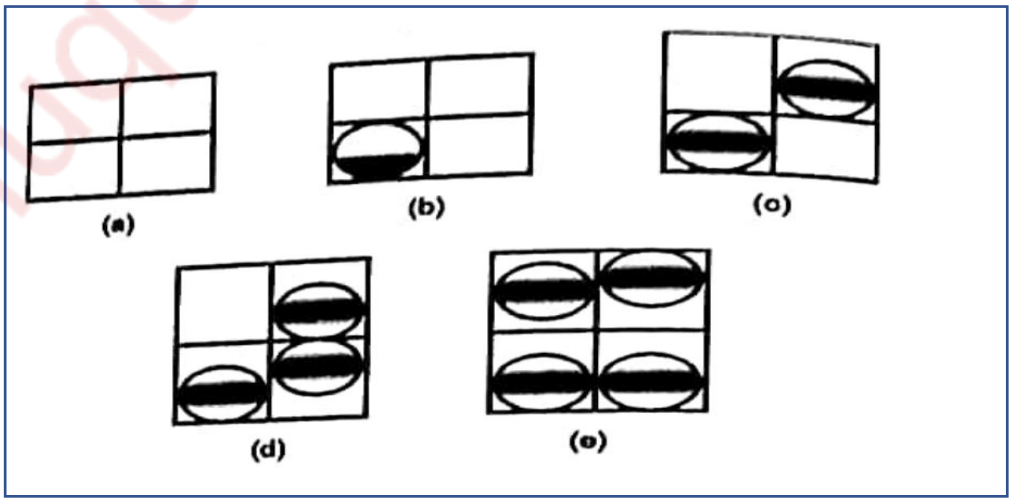
---

c) Explain the concept of halftoning with example. (5 M)

Ans:

Half toning

- Many displays and hardcopy devices are bi-level
- They can only produce two intensity levels.
- In such displays or hardcopy devices we can create an apparent increase in the number of available intensity value.
- 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.
- The phenomenon of apparent increase in the number of available intensities by considering combine intensity of multiple pixels is known as half toning.
- The half toning is commonly used in printing black and white photographs in newspapers, magazines and books.
- The pictures produced by half toning process are called halftones.
- In computer graphics, halftone reproductions are approximated using rectangular pixel regions, say 2*2 pixels or 3*3 pixels.



- These regions are called halftone patterns or pixel patterns.
- For bi-level systems one can represent shading intensities with a halftoning method that converts the intensity of each point on a surface into a regular pixel grid that can display a number of intensity level. The number of intensity level with this method depends on hoe many pixels we include on the grid.
- A graphics package employing a halftone technique displays a scene by replacing each position in the original scene an (n * n) grid pixels that are turned on. Such a technique of turns on two level systems into one with the five possible intensities.

---

d) Prove that two successive rotations are additive. (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 $(\Theta_{1+} \Theta_2)$.

$$R(\Theta_1) = \begin{vmatrix} \cos 90 & \sin 90 \\ -\sin 90 & \cos 90 \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix}$$

$$R(\Theta_2) = \begin{vmatrix} \cos 90 & \sin 90 \\ -\sin 90 & \cos 90 \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix}$$

Now if $R(\Theta_1) * R(\Theta_2)$

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, $R1(\Theta_1) * R2(\Theta_2) = R(\Theta_1 + \Theta_2)$.

Q.2)

a) Plot the points for midpoint ellipse with rx = 3 and ry =5 for region 1. (10 M)

Ans:

Given: rx = 3 and ry =5 for region 1

(xc, yc) =(0, 0)

Plot first point as (x, y) where x = 0 & y= ry, hence y =5

Initial decision parameter is calculated by

dp -1 = $(rx)^2 – (ry)^2 (-ry + 1/4)$

= $5^2 -3^2 (-5 + 1/4) = 271/4$

Now, we have to check whether we have crossed region – 1 or not

$2 (ry)^2 x < 2 (rx)^2 y$

$2 (3)^2*0 < 2 (5)^2*5$

$0 < 250$

As $2 (ry)^2 x < 2 (rx)^2 y$ it means point lies in region – 1

Here dp – 1 = 271/4

Hence, increase x by 1 and decrease by 1

(x, y) = (1, 4)

Every time we have to check whether we have crossed the region or not. If yes, then find out the new initial value of region-2 and proceed along y-axis till y becomes zero.

At the end i.e. when y becomes 0 we will get following table of x any y co-ordinates.

At last replicate these points to rest of the quadrants to get full ellipse.

X	Y
0	5
1	4
2	3
3	2
3	1
3	0

b) Explain the steps for 2D rotation about arbitrary point. (10 M)

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.

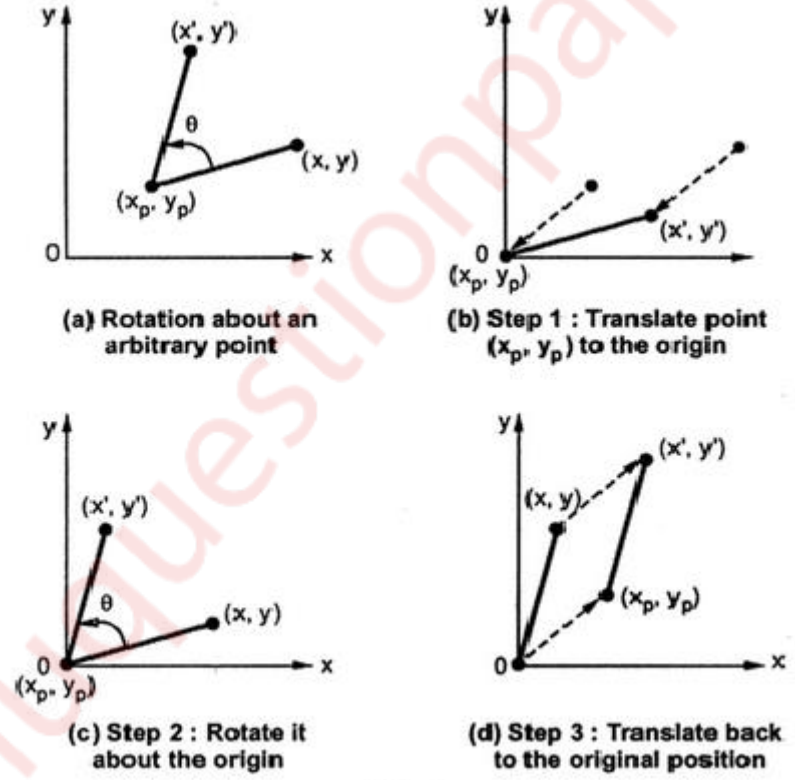


Fig. 1

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

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

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,

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

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

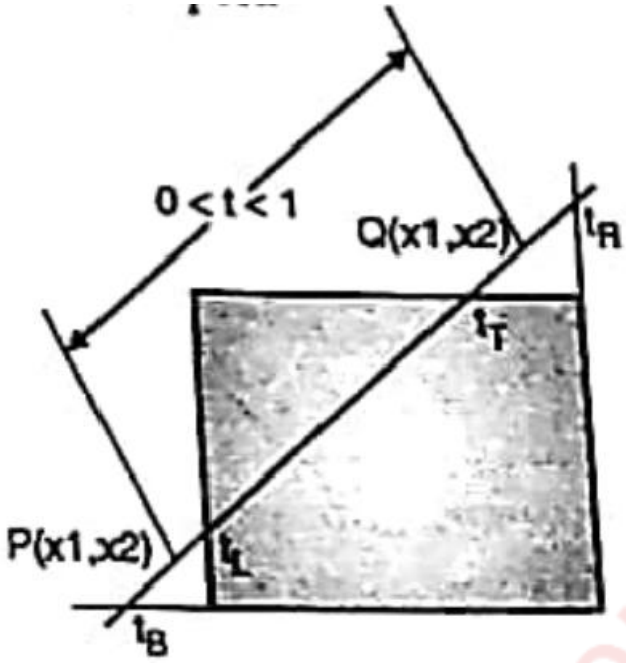
$$= \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ -xpcos\Theta + ypsin\Theta + xp & -xpsin\Theta - ypcos\Theta + yp & 1 \end{bmatrix}$$

Q.3)

a) Explain Liang Barsky line clipping algorithm. Apply the algorithm to clip the line with coordinates(30, 60) and (60, 25) 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

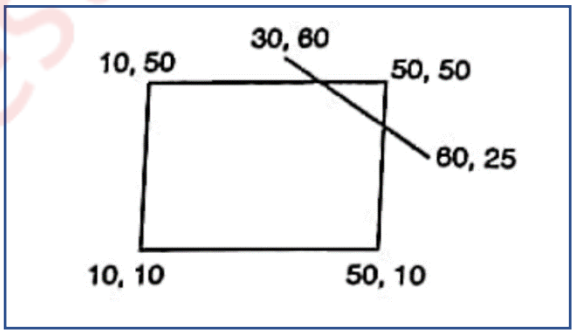
  X = X1 + (X2 – X1)*t = Y1 + dY*t

  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 – X1 = 50 -30 = 20

Q3 = Y1 – YB = 60 -10 = 50

Q4 = YT – Y1 = 50 -60 = -10

Now lets find P,

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

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

P3 = Q3/P3 = 50/(35) = (10/7)

P4 = 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

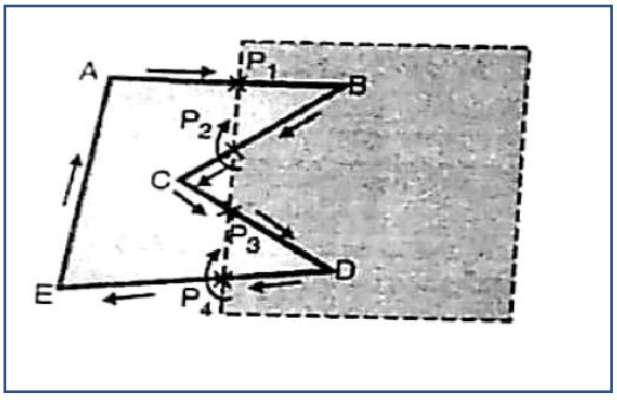
From this we will come to know that a point (38.57, 50) is an intersection point with respect to the edge of the window boundary. So we need to discard the line from point (30, 60) to (38.57, 50) and consider line (38.57, 50) to (60, 25).

---

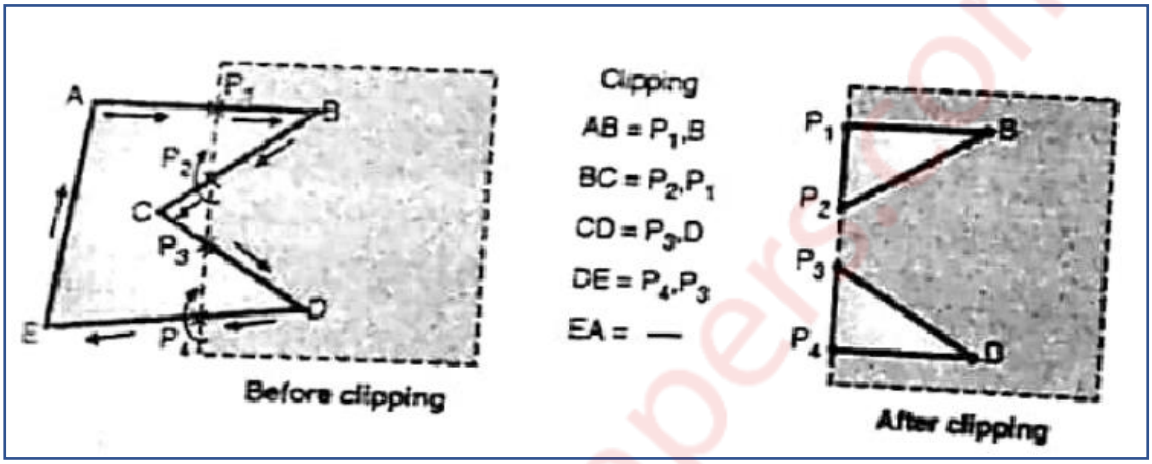
b) Explain Weiler Artherton polygon clipping algorithm with suitable example. (10 M)

Ans:

- This algorithm is one of the generalized polygon clipping algorithm. In Sutherland-Hodgeman algorithm, we always follow the path of polygon. But in this case sometime we are selecting polygon path and sometime window path. When to follow which path, that will depend on polygon processing direction, and whether a pair of polygon processing direction, and whether a pair of polygon vertices currently being processed represents an outside to inside pair or an inside to outside pair.
- We can follow polygon processing path either clockwise or anticlockwise. When we are following a path of polygon in clockwise direction at that time we have to use following rules.
- We have to follow the polygon boundary, same as that of Sutherland-hodgeman algorithm, if the vertex pair is outside to inside.
- We have to follow the window boundary in clockwise direction, if the vertex pair is inside to outside.
- Consider the concave polygon as shown in fig.
- Let us process the polygon in clockwise path. Here our vertex list will be the set of all vertices i.e. {A, B, C, D}.



- For edge Ab, as we are moving from outside to inside so store intersection point P1 and second point i.e. B. as we are following polygon in clockwise direction, so we have to consider next edge as BC. Here we are moving from inside to outside. So we are storing only intersection point P2. so we have to store that. It means we are storing P1 and P2. For next edge i.e. CD, again we are storing intersection point P3 and next point D. for edge DE we are storing intersection point P4 and following boundary in clockwise direction i.e. storing P3. For edge EA, as both points are outside, no vertex is stored.



- Here in Weiler-Atherton algorithm we are maintaining as such two different vertex list in single list. Here after clipping, the vertex list will be {P1, B, P2, P1, P3, D, P4. P3}. We have to draw edges by joining vertices as P1 to B, B to P2 to P1. Now there is no need to form an edge between P1 to P3. We have to continue with edge P3 to D, D to P4 and P4 to P3. We may use some logic here such as, when any vertex is stored twice in vertex list then don't form edge from that vertex to next vertex. Now we will not have edge P1P3 and edge P3P1. That's why we are sating, we are maintaining two sub array in single array of vertex.

Q.4)

a) What is window and viewport? Derive the matrix for viewport transformation. (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:

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

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

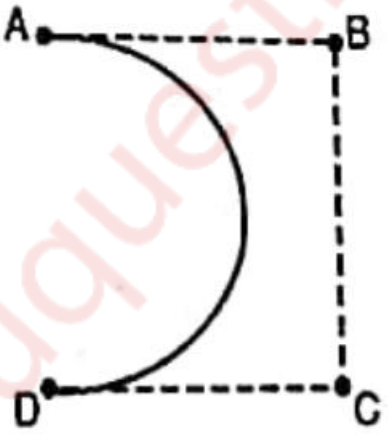
1. The position of the viewport can be changed allowing objects to be viewed at different positions on the Interface Window.
2. Multiple viewports can also be used to display different sections of a scene at different screen positions. Also, by changing the dimensions of the viewport, the size and proportions 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.

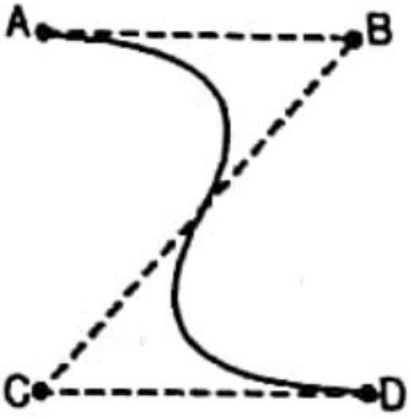
---

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 perspective 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.

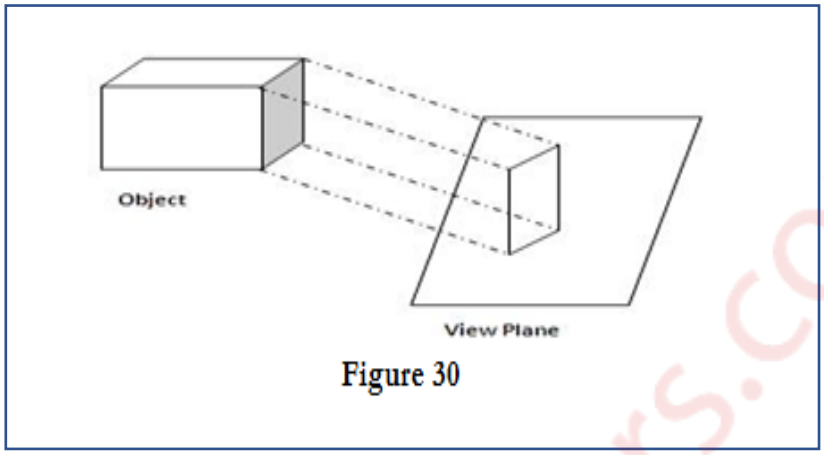


Figure 30

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.

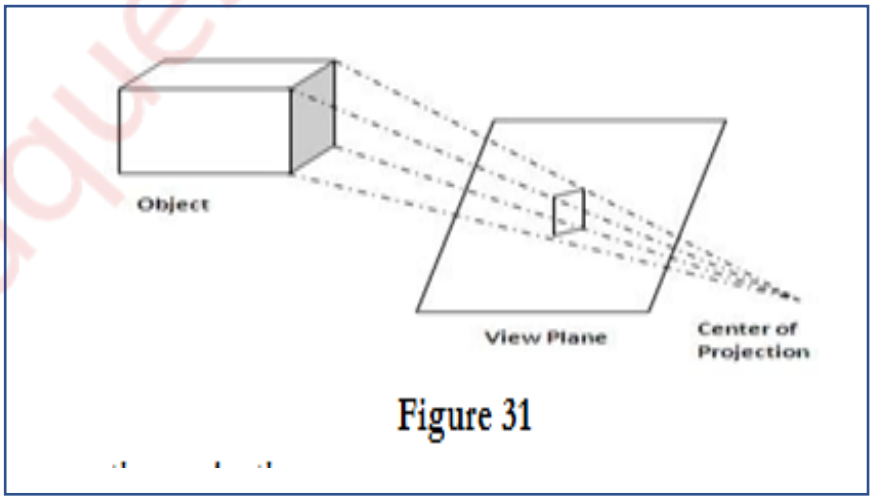
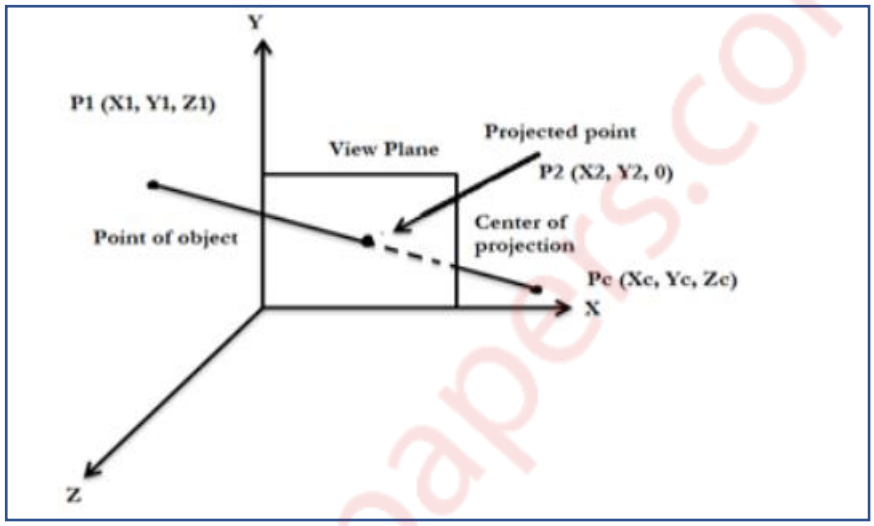


Figure 31

Matrix for perspective projection:

Let us consider the center of projection is at $P_c(X_c,Y_c,Z_c)$Pc(Xc,Yc,Zc) and the point on object is $P_1(X_1,Y_1,Z_1)$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 liens 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 sorting polygons which are closer to viewer. We know that frame buffer is used to store 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 of 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.
- When there is nothing to display on monitor i.e. frame buffer is empty, at that time we have t initialize z buffer elements to a very large negative values. A large negative values on z axis represents a point beyond which there is nothing i.e. setting background colour. $z_{buffer}(x, y) = z_{initialvalue,}$
- If the new surfaces has z value greater than z_{buffer} then it lies in front. So we have to modify the contents of $z_{buffer}(x, y)$ by new z values and set the pixel value at (x, y) to the colour 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 new value of new surface is smaller than $z_{buffer}(x, y)$ then it lies behind some polygon which was previously entered. So 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.
- For example, in figure (e) shown below, among three surfaces, surface S1 has the smallest depth at view position (x, y) and hence highest z value. So it is visible at that position.

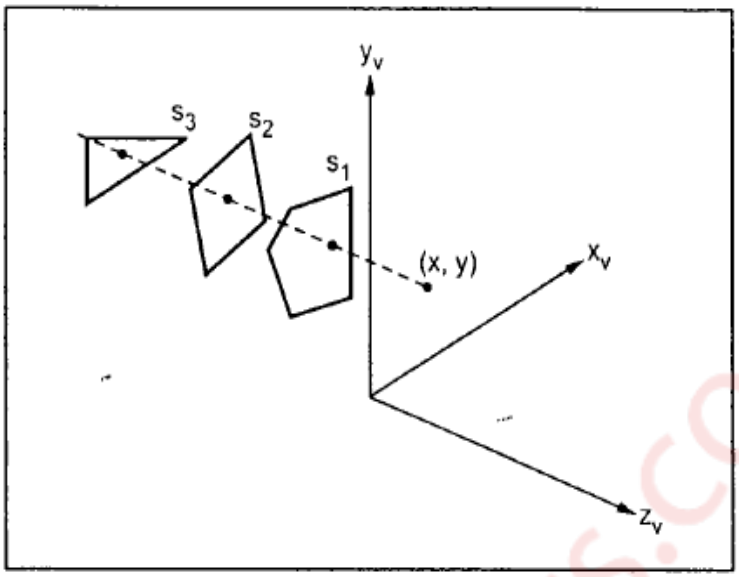


Fig. (e)

- To calculate z-values, the plane equation

  Ax + By + Cz + D =0

  is used where (x, y, z) is any point on the Plane, and the coefficient A, B, C and D are constants describing the spatial properties of the Plane.

- Therefore, we can write

  Z = (-AX -BY – D) / C

  Note, if at (x, y) the above equation evaluates to z1, then at (x + Δx, y) the value of z. is,

  Z1 = A / C (Δx)

  Only one subtraction is needed to calculate z(x + 1. y), given z(x, y). Since the quotient A/C is constant and Δx = 1. A similar incremental calculation can be performed to determine the first value of z on the next scan line, decrementing by B/C for each Δy.

  - **Advantages:**
    - It is easy to implement.
    - As z buffer algorithm processes one object at a time, total number of objects can be large.
  - **Dis-advantages:**
    - It requires lots of memory as we are storing each pixels 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

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).

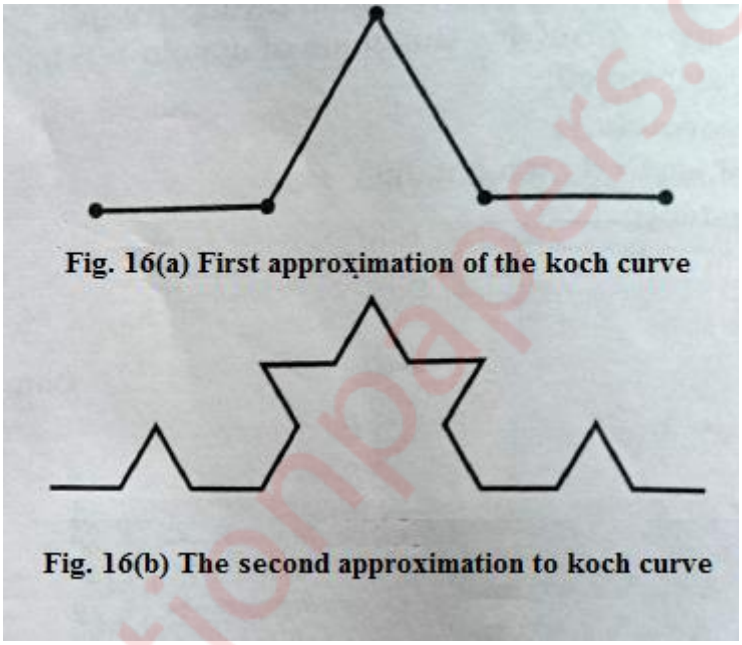


Fig. 16(a) First approximation of the koch curve

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

Ans:

- Sweep representations are used to construct three dimensional objects from two dimensional shape .
- There are two ways to achieve sweep:
  - Translational sweep
  - 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)

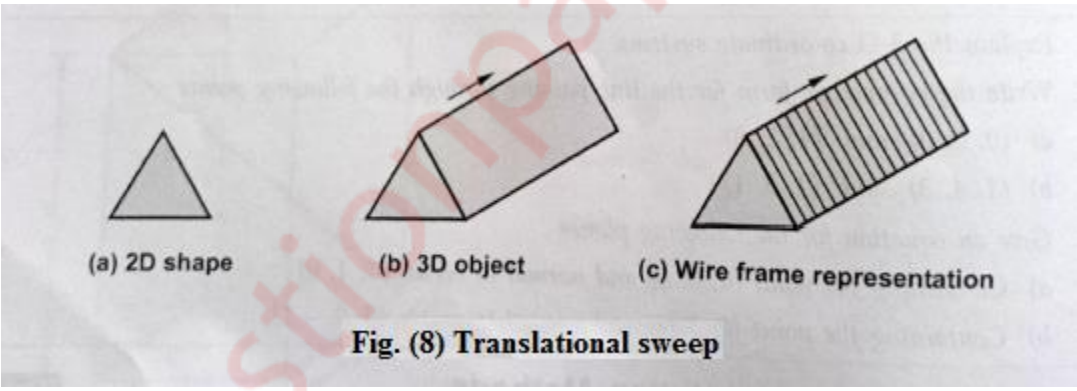
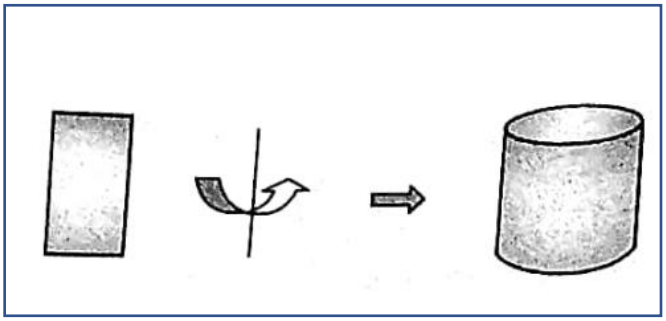


Fig. (8) Translational sweep

- 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.
- In rotational sweep, the 2D is rotated about an axis of rotation specified in the plane of 2D shape to produce three dimensional objects.



- Sweep representation are widely used in compute vision. However, the generation of arbitrary objects rather difficult using this technique.

---

c) Gouraud and Phong shading

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.

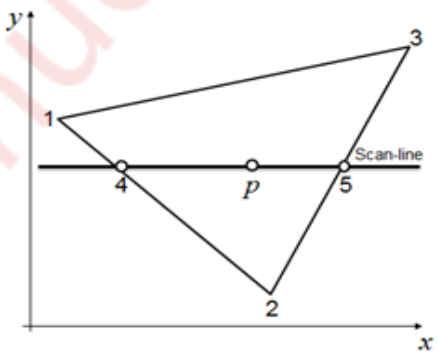


Figure 41

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

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

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

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

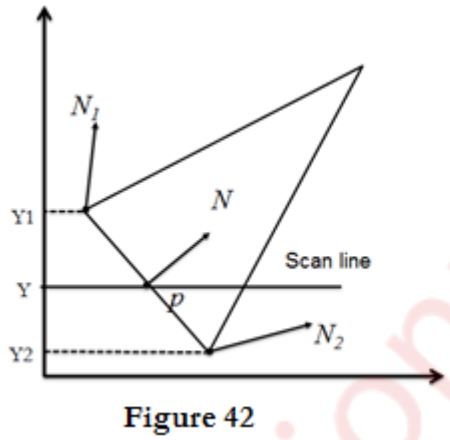


Figure 42

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

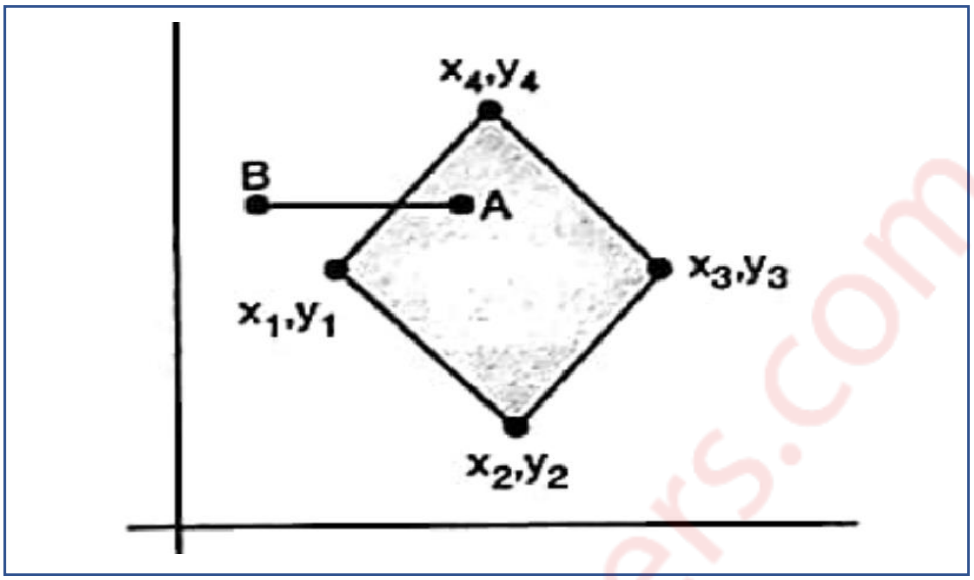
---

d) Inside Outside test.

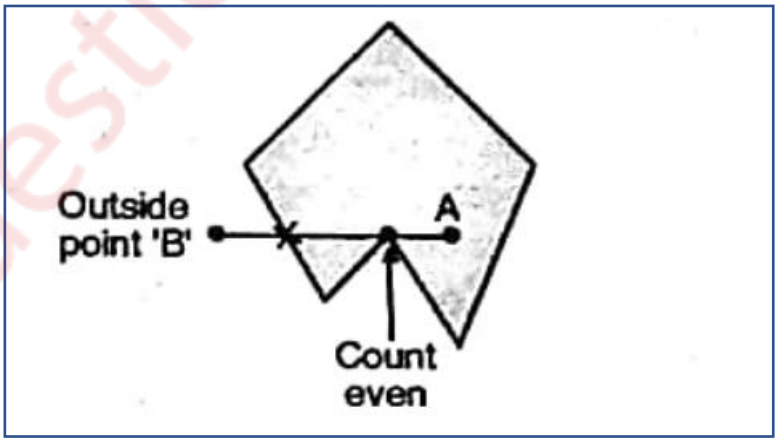
Ans:

- One method of doing this is to construct a line segment between a point in question i.e. point to test whether inside or outside, and a point which is surely outside the polygon. But ow we are going to find out a point which is surely outside the polygon? It is very easy to find out that point.
- For example, pick a point with an x co-ordinate smaller than the smallest co-ordinate of the polygons vertices and the y co-ordinate will be any y value, or for simplicity we will take y value same as the y value of point in question.
- In this case point A is one, which we want to check i.e. whether point A is inside polygon or not.
- As we are using arrays to store vertices of polygon we can easily come to know the vertex which is having lowest value of x and that is x_1. So we have to select a point smaller than x_1 and generally we select same value of y as that of point A,

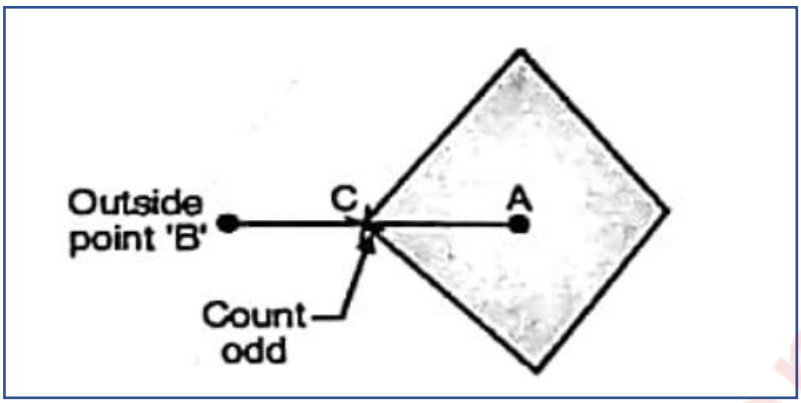
in order to draw straight line. But even if you select any y value for outside point, it will not make any difference.



- Then count how many intersections are occurring with polygon boundary by this line till the point in question i.e. 'A', is reached. If there are an odd number of intersections then the point is outside the polygon.
- This is called even-odd method to determine interior points of polygon.
- But this even odd test fails for one case i.e. when the intersection point is vertex. To handle this case we have to make few modifications we must look at other end points of two segments of a polygon which meet at this vertex.
- If these points lie on the same side of the constructed line AB, then the intersection point counts as an even number of intersection.



- But if they lie on opposite sides of the constructed line AB, then the intersection point is counted as a single intersection.



- In this case the constructed line AB intersects polygon in a vertex ‘C’. here the other points of both the line segments which meets at vertex ‘C’ are lieing in opposite sides of this constructed line Ab. So the intersection at vertex ‘C’ is counted as odd count i.e. 1. The total number of C intersection made by this line with polygon is 1 i.e. odd and as it is odd therefore the point ‘A’ is inside.
