COMPUTER GRAPHICS (NOV 2018)

Q.P.Code: 60317

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

a) Compare Raster and Random Scan Techniques. **(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.

b) What are the disadvantages of DDA algorithm? **(5 M)**

Ans:

- Floating point arithmetic in DDA algorithm is still time consuming.
- The algorithm is orientation dependent. Hence end point accuracy is poor.
- Although DDA is fast, the accumulation of round-off error in successive additions of floating point increment, however can cause the calculation pixel position to drift away from the true line path for long line segment.
- Rounding-off in DDA is time consuming.

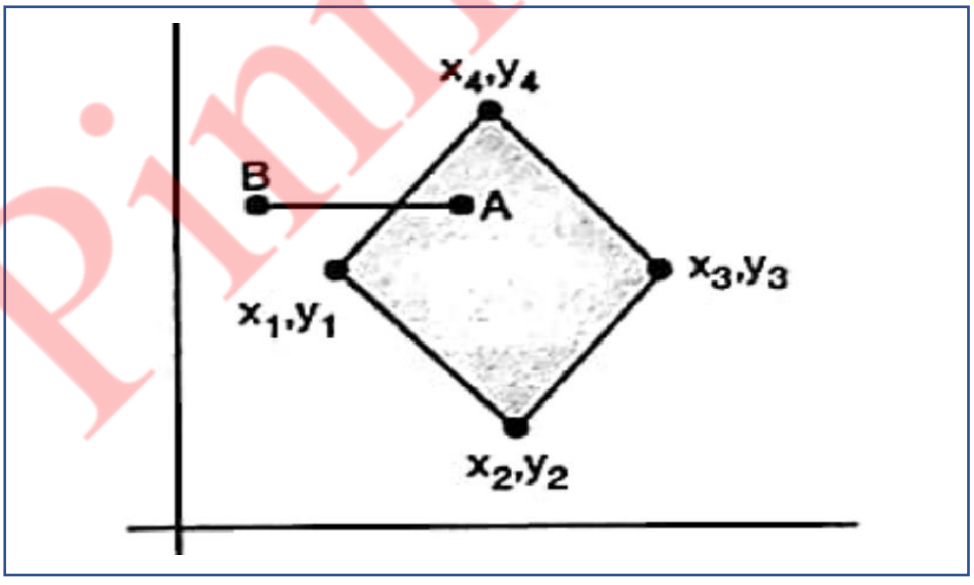
- Because of round off, errors are introduced and causes the calculated pixel position to drift away from the true line path.
- Because of floating point operation the algorithm is time consuming.

---

c) Explain inside outside test used in filling algorithm. (5 M)

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', id reached. If there are an odd number of intersections then the point is outside the polygon.

---

d) What are aliasing and antialiasing? 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.

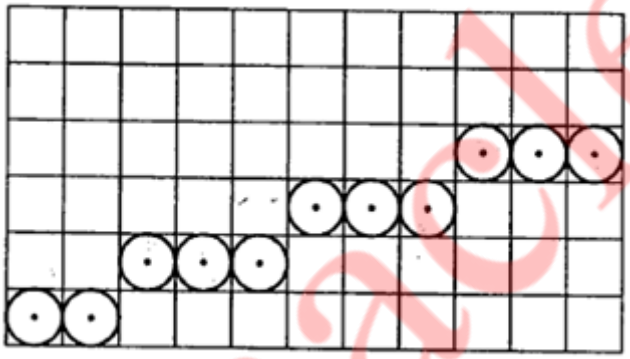


Fig. Aliasing effect

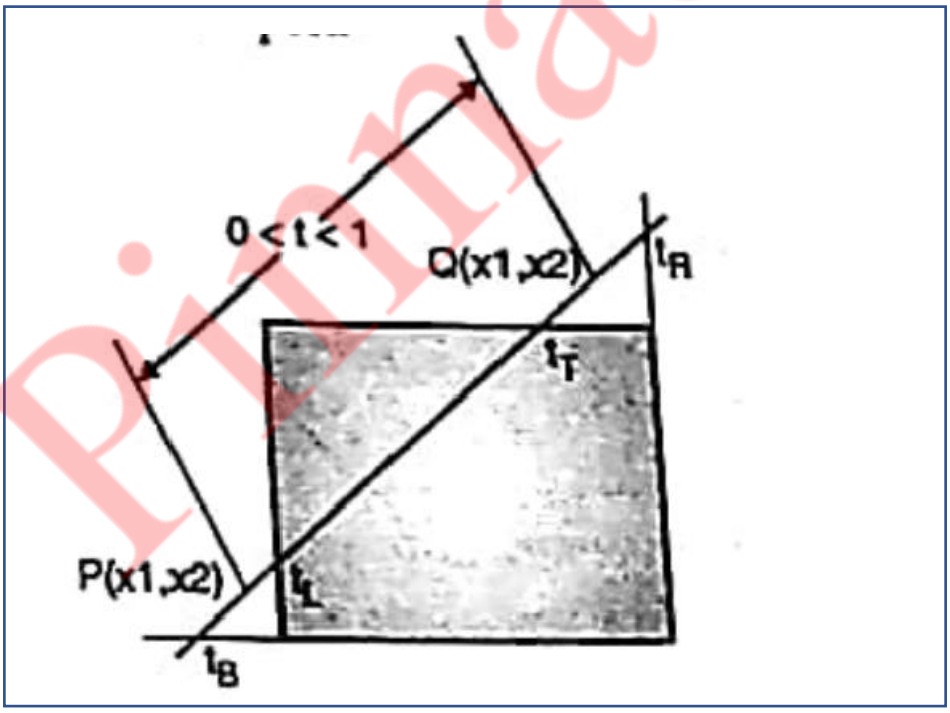
- The aliasing effect can be reduced by adjusting intensities of the pixels along the line. The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called antialiasing.
- **Antialiasing** is a term for techniques that are designed to mitigate the effects 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.
- 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.

Q.2)

a) Explain Liang Barsky line clipping algorithm. Apply the algorithm to clip the line with coordinates(35, 60) and (80, 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

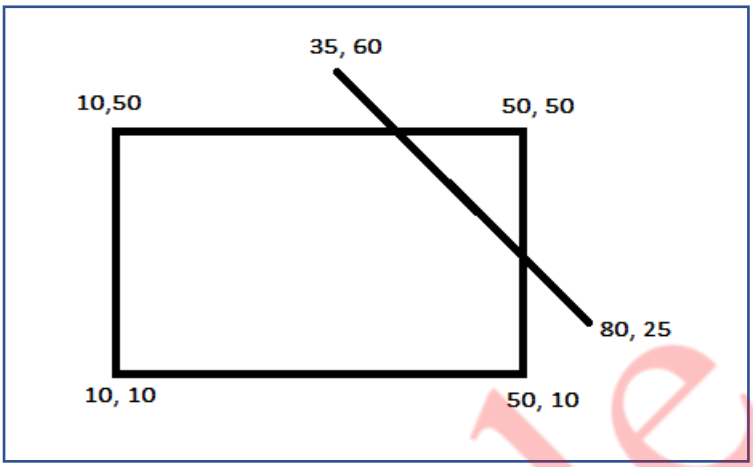
  $X = X1 + (X2 – X1)*t = Y1 + dY*t$

  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 = 35, Y1 = 60, X2 = 80, Y2 = 25

Now we have to find Dx and Dy as

Dx = X2 – X1 = 80 – 35 = 45

Dy = Y2 – Y1 = 25 – 60 = -35

Lets calculate the values of parameters P and Q as

P1 = -Dx = -45

P2 = Dx = 45

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

P4 = Dy = -35

Now,

Q1 = X1 – XL = 35 -10 = 25

Q2 = XR – X2 = 50 - 80 = -30

Q3 = Y1 – YB = 60 -10 = 50

Q4 = YT – Y2 = 30 - 25 = 5

Now lets find P,

P1 = Q1/P1 = 25/(-45) = (-5/9)

P2 = Q2/P2 = -30/(45) = (-2/3)

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

P4 = Q4/P4 = 5/(-35) = (-1/7)

Now,

t1 = Max(-5/9, 10/7, 0) = 10/7

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

Now,

X1' = X1 + Dx*t1

= 35+45*(10/7)

= 99.29

Y1' = Y1 + Dy*t1

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

= 10

And with t2 it will be

X2' = X1 + Dx*t2

= 35+45*(-2/3)

= 5

Y2' = Y1 + Dy*t2

= 60+(-35)*(-2/3)

= 83.33

From this we will come to know that a point (5, 83.33) is an intersection point with respect to the edge of the window boundary. So we need to discard the line from point (35, 60) to (5, 83.33) and consider line (5, 83.33) to (80, 25).

---

b) Derive the matrix for 2D rotation about an 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.

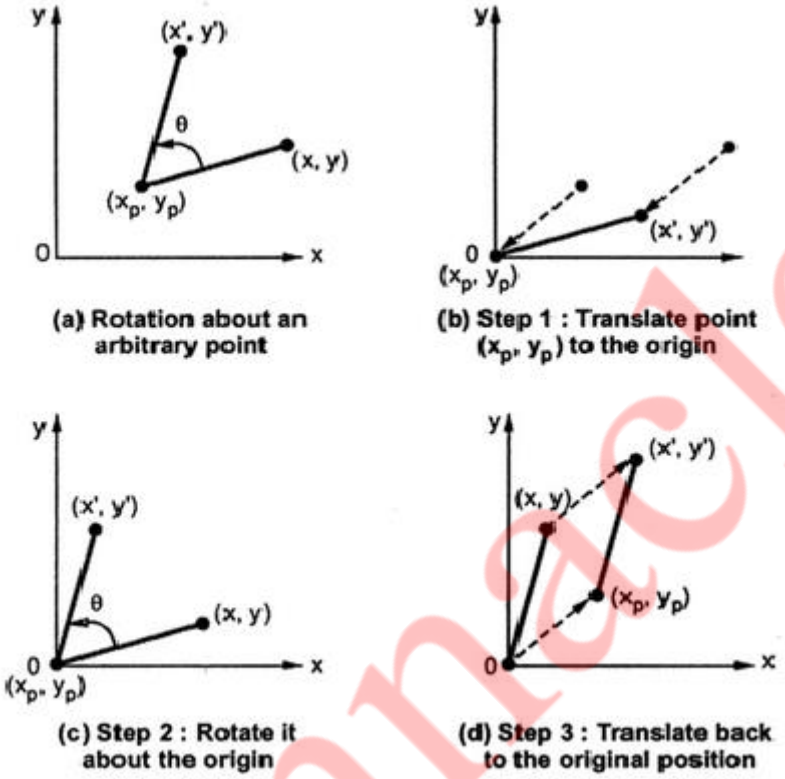


Fig. 1

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,

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

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

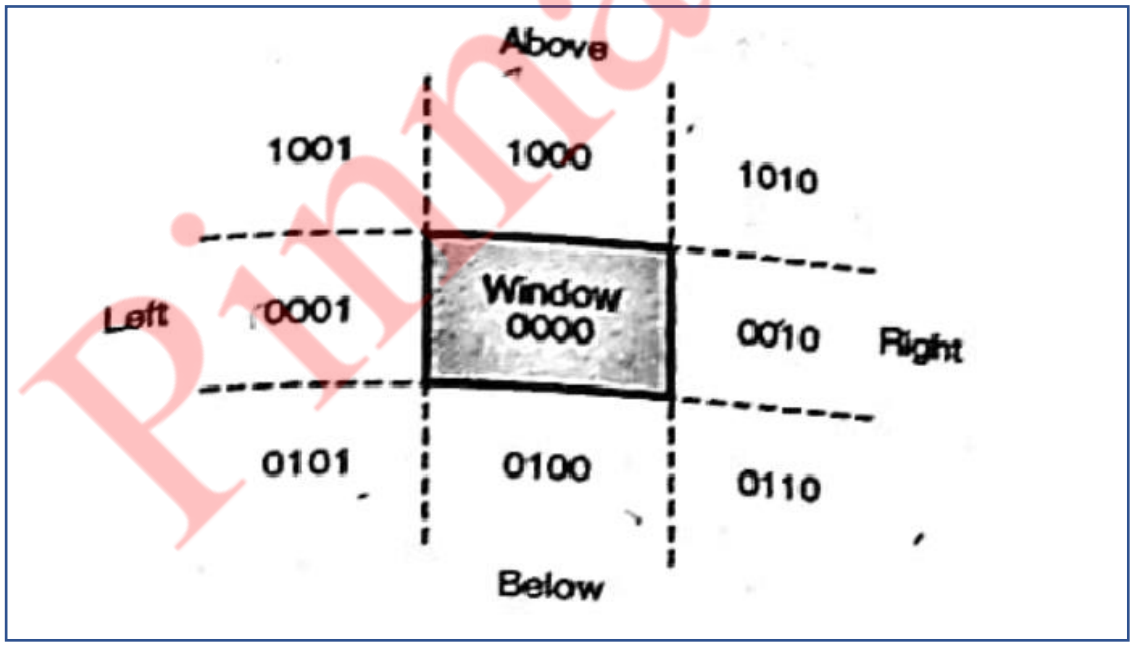
$$= \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ -xpcos\Theta + ypsin\Theta + xp & -xpsin\Theta - ypcos\Theta + yp & 1 \end{bmatrix}$$

Q.3)

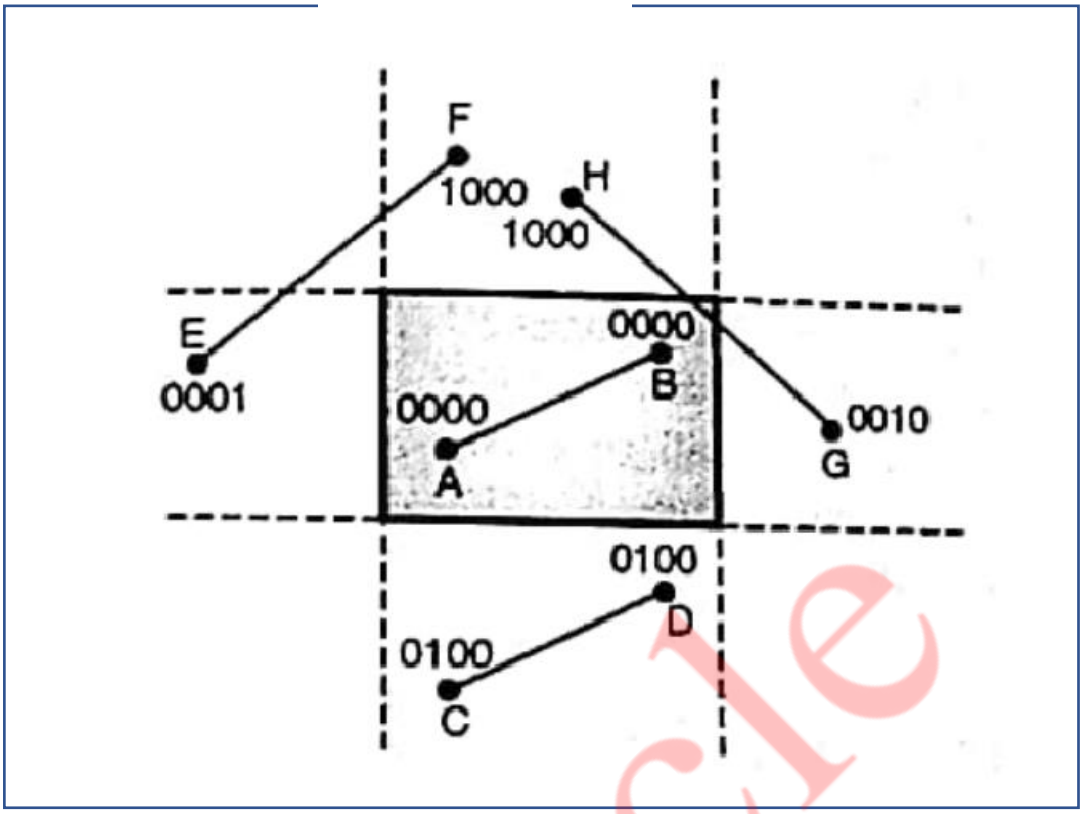
a) Explain the Cohen-Sutherland line clipping algorithm with suitable example. (10 M)

Ans:

- It is one of the popular line clipping algorithm. This algorithm immediately removes the lines which are lying totally outside the window.



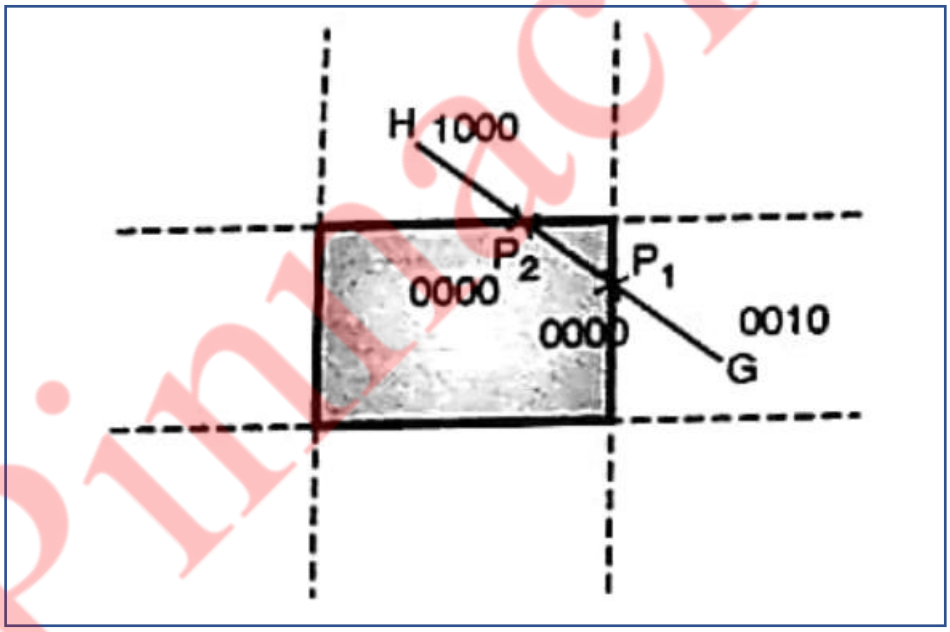
- End point of each line is assigned a four bit binary code which is called ass out code. We can give abbreviated name to this four bit code as, “ABRL” where A = above, B = below, R = right, L = left.



- The cohen Sutherland algorithm tells that if the line is totally on one side of the window then immediately we have to discard that line. Similarly if the line is totally inside the window directly display that line without clipping.
- **Case 1:**
  - Consider a line AB. Here both end points of AB are surely inside the window. Therefore the out code for end point A will be out code(A) = ABRL = 0000, out code(B) = ABRL = 0000
  - The algorithm says that if the out codes of both endpoints of a line are 0000. Then the line is fully visible. So we should not clip the line.
- **Case 2:**
  - Consider a line CD. Here endpoints of CD are surely outside the window. The out code for endpoint C will be, out code(C) = ABRL = 0100, outcode(D) = ABRL = 0100
  - Now here for line CD, the outcodes of both endpoints of a line are not zero, so we have to perform logical AND operation of both outcodes. Outcode(C) = 0100 AND Outcode(D) = 0100 = 0100.
  - If the result of AND operation is nonzero, then the line is surely outside the window. So clip that line. Here line CD is clipped as its AND result is nonzero.
- **Case 3:**
  - But if the logical AND result is zero, then we cannot make form statement about a line, whether it should be visible or partially visible. Consider line EF and GH. For line EF outcode of E will be 0001 and outcode of F will be 1000. As both outcodes are not zero, so we have to perform logical AND.

Outcode(E) = 0001 AND outcode(F) = 1000 = 0000

- The AND result is zero. It means the line EF is not lying completely on any side of the window.
  Similarly for line GH,
  Outcode(G) = 0010 AND outcode(H) = 1000 = 0000
- If we observe both lines GH and EF, the AND result of both line is zero. But line GH is partially visible and line EF is fully invisible. So, far such line we have to edge of the window the line is intersecting and then we have to find the intersection point. To find out this we can make use of outcode.
- Consider line EF. Outcode of E is 0001 and F is 1000. Here we have to compare the first bit of both outcodes..
- First bit of E is 1 and first bit of F is 0. That means point E of line EF is lying outside the left boundary of window. So we have to find out intersection point of line EF with left boundary of window. Call that intersection as P_1. Now we are having two lines as EP_1 and P_1F.



- As point E is surely outside the left boundary of window so we are clipping the line EP_1. Now we will concentrate an P_1F. now we have to find outcode of P_1, which will be 1000. Here both outsides are nonzero i.e. P_1= 1000 and F = 1000 so, logical AND will be nonzero i.e. 1000, it means P_1F is lying completely on one side of window i.e. above the window, so discard the line. In this we are eliminating whole line.
- The same method is used for partially visible lines also such as GH.

b) What is meant by parallel and perspective projection? Derive matrix for matrix 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.

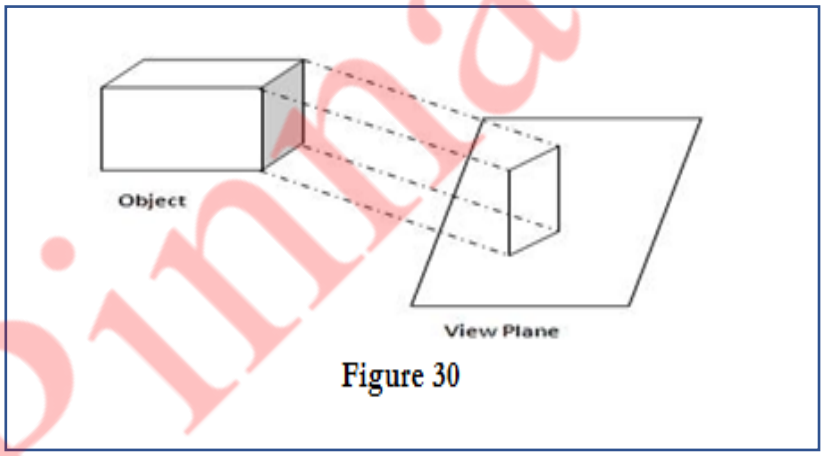


Figure 30

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.

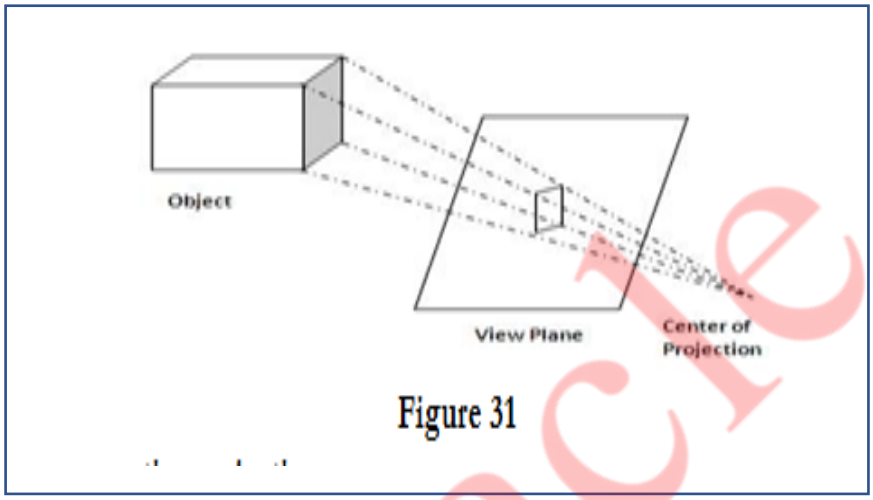
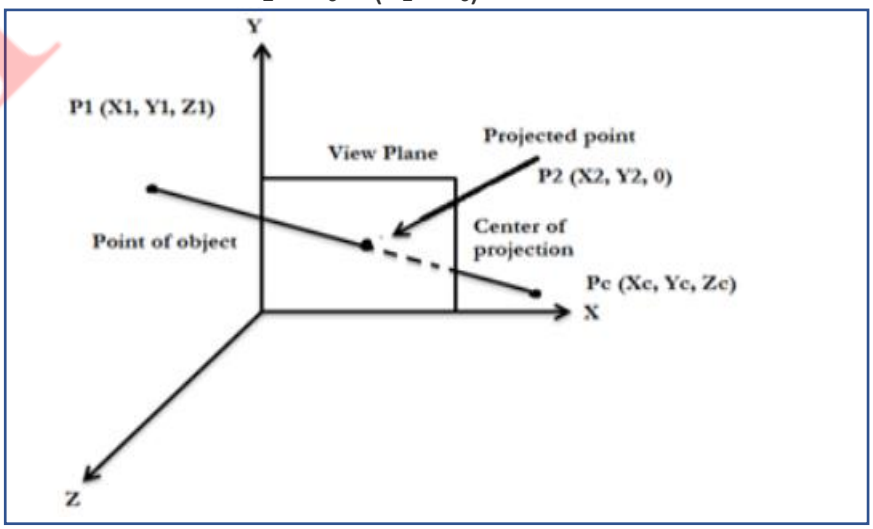


Figure 31

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

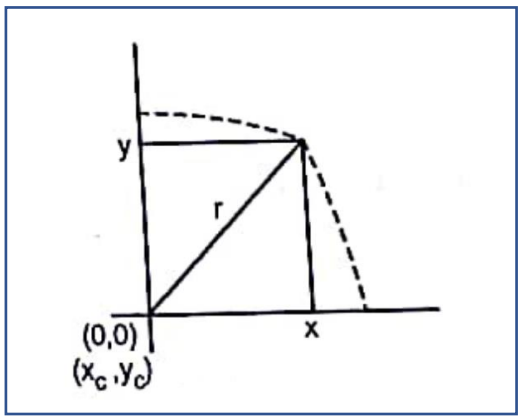
$Z = -Z_C$

Q.4)

a) Specify midpoint circle algorithm. Using the same, plot the circle whose radius is 8 units and center is at (10, 10). (10 M)

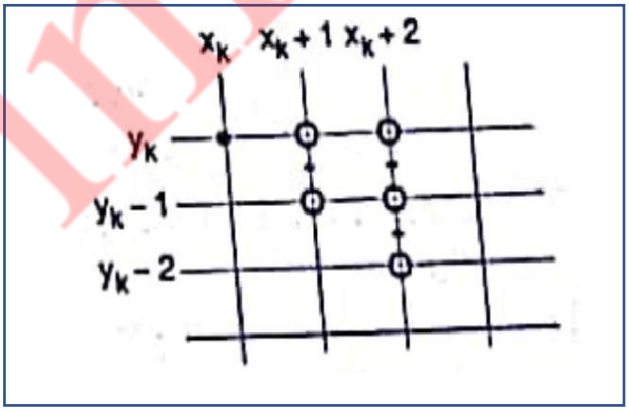
Ans:

- Here we have to determine the closest pixel position to the specified circles path at each step. For this we have to accept radius 'r' and center point x_c and y_c.
- Then each calculated point is to be moved to proper position by adding x_c and y_c to corresponding x and y-value. In first quadrant we are moving in x-direction with starting point as x=0 to ending point x=y.
- To apply a midpoint method, we define a circle function.



$F_{circle}(x,y) = x^2+y^2-r^2$ which is same as $r^2=x^2+y^2$ which is derived from equation of circle, $(x-x_c)^2+(y-y_c)^2=r^2$. But here we are considering x_c, y_c as (0, 0)
Hence, $x^2+y^2 = r^2$.

- Any point (x, y) on the boundary of the circle with radius r satisfies the equation $F_{circle}(x,y) = 0$. If point is inside the circle function is negative and if point is outside circle then circle function is positive.
  i.e. if
  $f_{circle}(x,y) < 0$, then x, y is inside the circle boundary.
  $= 0$, then x, y is on circle boundary.
  > 0, then x, y is outside the circle boundary.
- Thus circle function is the decision parameter in this algorithm. Assuming that we have just plotted x_k, y_k point.
- Now we have to determine whether next point $(x_k + 1, y_k)$ is closer or $(x_k + 1, y_k - 1)$ is closer to actual circle.



- Our decision parameter (P_k) is circle function.
  $P_k = f_{circle}(x_k + 1, y_k)$ or
  $P_k = f_{circle}(x_k + 1, y_k - 1)$
- Therefore we will take midpoint of these two point as,
  $P_k = f_{circle}(x_k + 1, y_k – 1/2)$
  $= f_{circle}(x_k + 1)^2 +(y_k – 1/2)^2 - r^2$ from circle equation.
- If $P_k < 0$, then it means this point $(x_k + 1, y_k – 1/2)$ is inside circle. So we have to plot pixel $(x_k + 1, y_k)$.

After this we have by 1.

i.e. $P_{knew} = f_{circle}(x_k + 1, y_k - 1/2)$

$= (x_k + 2)^2 + (y_k - 1/2)^2 - r^2$

If we solve this we will get

$P_{knew} = P_k + (2\,x_k + 1)$

i.e. when, earlier we have not changed row.

- If $P_k > 0$, then we will select $y_k - 1$ for displaying. Like this we will select the point to be displayed.

  $P_{knew} = f_{circle}(x_k + 2, y_k - 3/2)$

  $= (x_k + 2)^2 + (y_k - 3/2)^2 - r^2$

  If we solve this we will get

  $P_{knew} = P_k + (2\,x_k - 2\,y_k + 1)$

  i.e. when, earlier we have not changed row.
- Similarly the initial decision parameter is obtained by evaluating the circle function at start position $(x_0, y_0) = (0, r)$

  Now calculate the decision parameter.

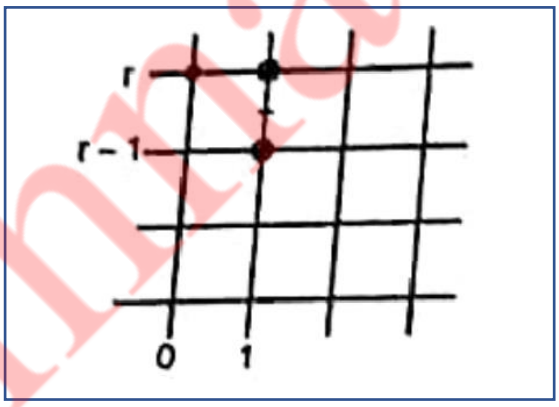
  $P_0 = f_{circle}(x_k + 1, y_k - 1/2)$

  But $x_k = 0$ and $y_k = r$

  $x_k + 1 = 1$ and $y_k - ½ = r - ½$

  $P_0 = f_{circle}(1, r - ½) = 1^2 + (r - ½)^2 - r^2$

  $P_0 = 5/4 - r$



If radius r is integer we will round it ass $P_0 = 1 - r$.

- Given: $r = 8$, $x_c = 10$, $y_c = 10$

  As stated in algorithm,
  - We first calculate the points assuming the centre co-ordinates (0, 0)
  - At the end we translate the circle.

  Now,

  $P_0 = 1 - r$

  $= 1 - 8 = -7$

  $P_0 < 0$

$X_{k+1} = x_{k+1} = x_k + 1 = 0 + 1 = 1$

$Y_{k+1} = y_k = 8$

$P_{k+1} = P_k + 2* x_{k+1} + 1 = -7+2*1+1 = -4$

Continue this process till (x=y) i.e. till angle 45, where x and y becomes equal.

X	Y
10	18
11	18
12	17
13	16
14	15
14	15
16	14

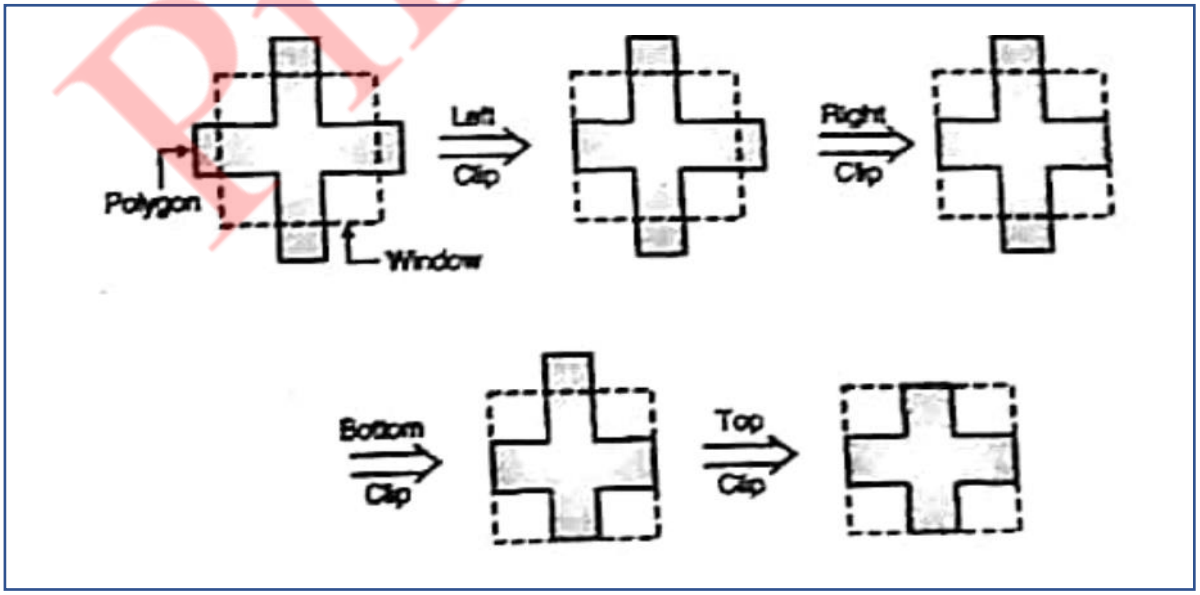
---

b) Explain any one polygon clipping algorithm. (10 M)

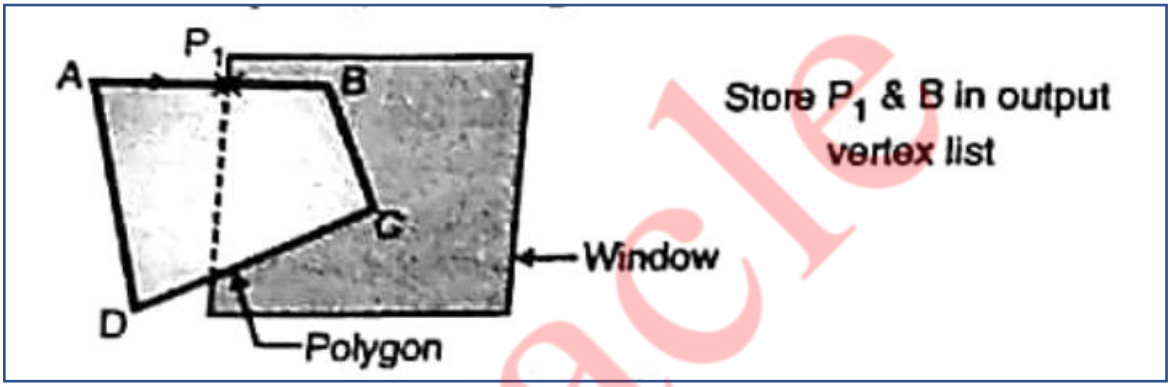
Ans:

Sutherland Hodgman polygon clipping algorithm:

- 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.

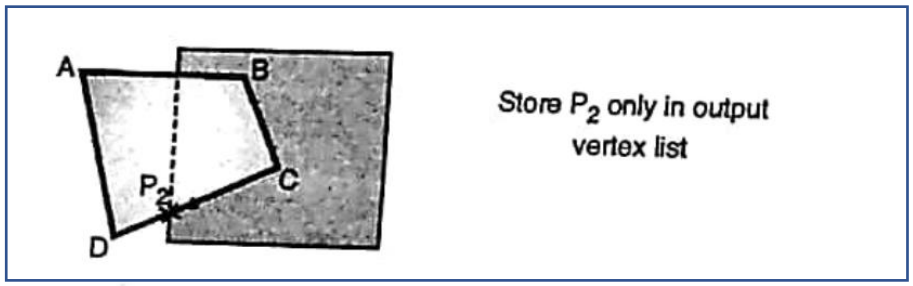


- At the end of every a new set of vertices is generated and this new set or modified polygon is passed to the next clipping stage.
- After clipping a polygon with respect to all four boundaries we will get final clipped polygon.
- When we clip a polygon w.r.t any particular edge of the window four different 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.

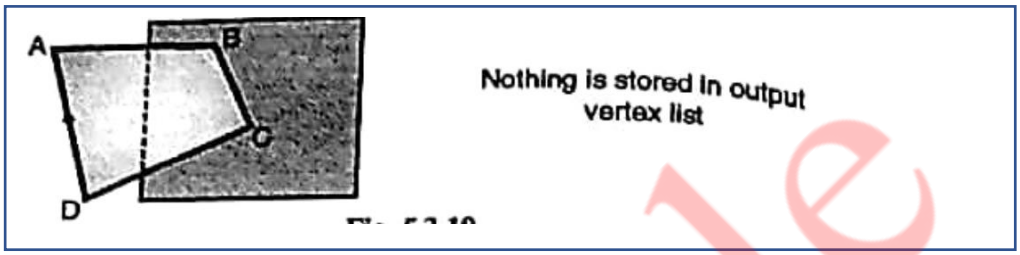


- For edge AB instead of storing vertex A and B are storing P_1 and B in output vertex list.
- **Case 2:** If both, first and second vertices of a polygon are lying inside the window, then we have to store the second vertex only in output vertex list.

- **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.



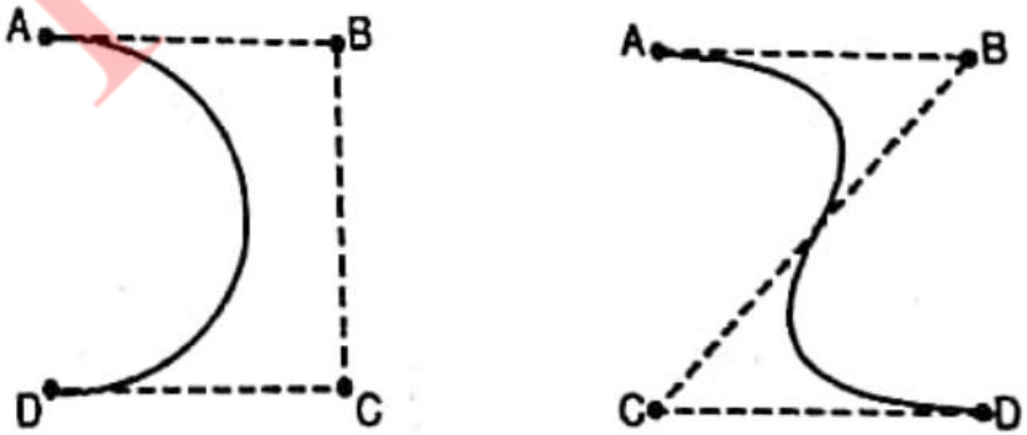
- Once all vertices have been considered for one clip window boundary, the output list of vertices is clipped against the next window boundary.

Q.5)

a) Explain Bezier curve with its properties and construct. (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.

---

b) Explain Gourard and Phong Shading along with their advantages and disadvantages. (10 M)

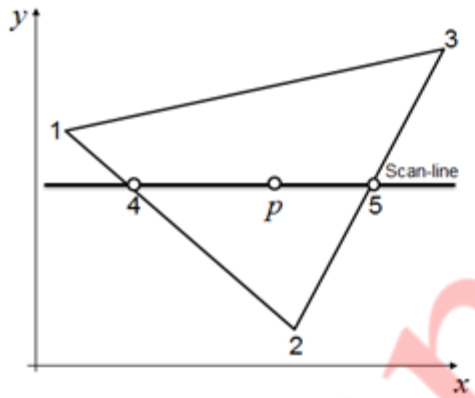
Ans:

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.



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

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

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

Figure 41

Advantages: Removal of discontinuities associated with the constant shading model.

Disadvantages: Highlighted surfaces are sometimes displayed with anomalous shape and the linear intensity interpolation can use bright or dark intensity interpolation can use bright or dark intensity strips. This effect can be reduced by sung phong shading method.

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

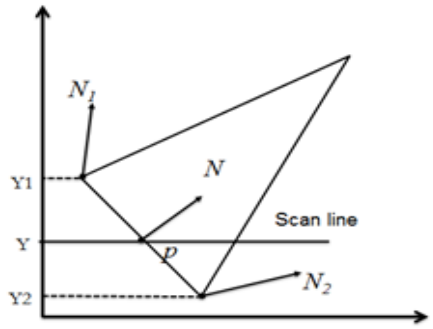


Figure 42

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

Advantages: It is very effective in dealing with specular highlights.

Disadvantages: This technique is relatively time-consuming since the illumination model is evaluated at every point using interpolated normal vectors.

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

a) Depth Buffer method **(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.
- **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.

---

b) Halftone and Dithering techniques.

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

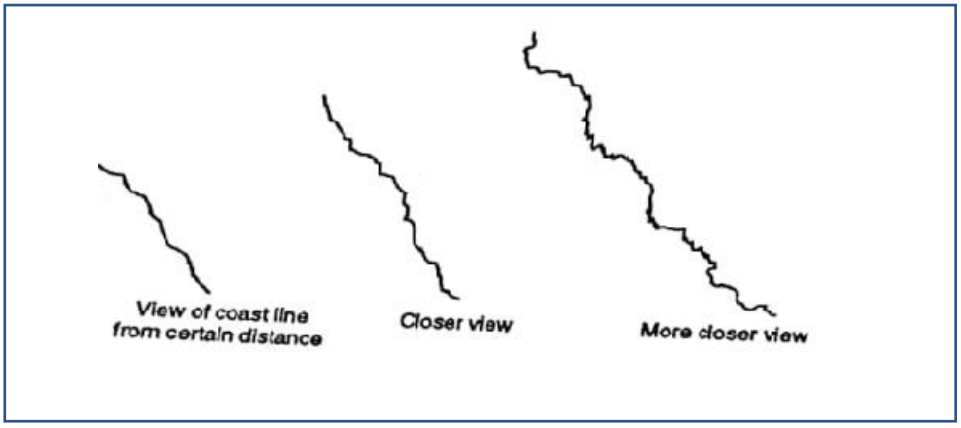
- Dithering refers to techniques for approximating halftones without reducing resolution, as pixel grid patterns do.
- The term dithering is also applied to halftone approximation method using pixel grid, and something it is used to refer to color halftone approximations only.
- Random values added to 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.

---

c) Fractals

Ans:

- A fractal is defined as a rough or fragmented geometric shape that can be split into parts, each of which is approximately a reduced-size reproduction of the complete shape based on the property known as self- similarity. It was derived from the Latin word fractus which means broken or fractured. Natural objects can be realistically described using fractal geometry methods. Example.- cloud, mountains, trees, stone etc.
- Fractal methods use procedures rather than equations to model objects. so it uses procedural modelling. The major characteristic of any procedural model is that the model is not based on data ,but rather on the implementation of the procedure following a particular set of rules.
- A fractal combines the following characteristic:
- Its parts have the same form or structure as a whole, except that they are at a different scale and may be slightly deformed.
- Its form is extremely irregular or fragmented, and remains so, whatever the scale of examination.
- It is formed by iteration i.e. the procedure is used repeatedly(recursively)
- Example: if P0 = (X0, Y0, Z0) is a selected initial position, the successive levels P1 = F (P0), P2 = F (P1), …… Pn = F (Pn-1) are generated by a transformation function F.
- Fractional dimensions.



- Imagine that one object is made up of clay. If we break that objects in terms of line or line segments, then we will give the dimensions D_1= 1. If the object is broken into a plane, then we will give the dimension D_t = 2. And if the object is broken into 3D objects like cube, sphere etc. then we will give the transformation as D_t = 6. Here the variables D_t is called topological dimensions.

---

d) 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).

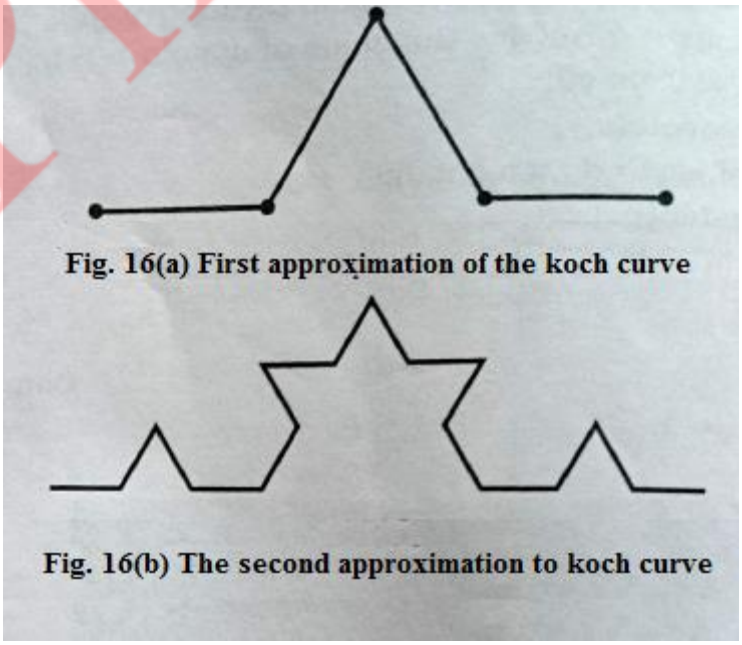


Fig. 16(a) First approximation of the koch curve

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

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

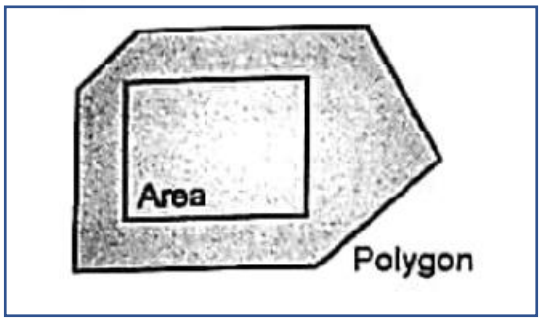
- Therefore for koch curve topological dimension is 1 but fractal dimension is 1.2618.
- From the above discussion we can say that point sets, curves and surfaces which give a fractal dimension greater than the topological dimension are called fractals The Hilbert's curve and koch curves are fractals,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

---

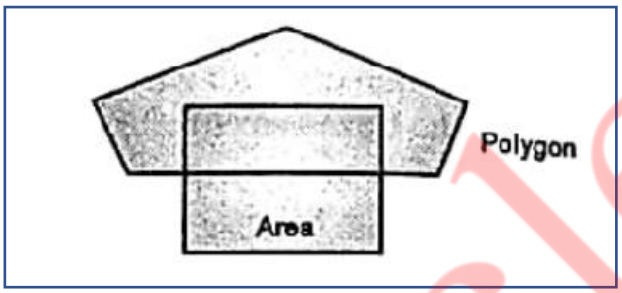
e) Area subdivision method

Ans:

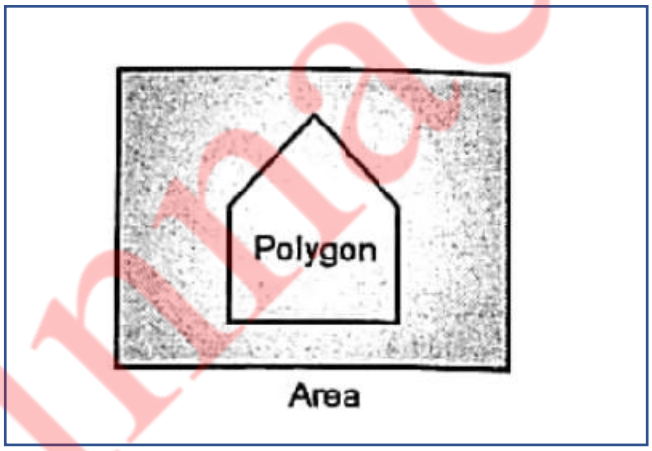
- The area-subdivision method takes advantage by locating those view areas that represent part of a single surface.
- Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.
- Continue this process until the subdivisions are easily analysed as belonging to a single surface or until they are reduced to the size of a single pixel.
- The algorithm is a recursive procedure based on a 2-step strategy.
    1. Decide which polygons overlap the given area on the screen.
    2. Which polygons are visible in that area.
- Categories of polygons are:
- **Surrounding polygon:** In this case a polygon completely covers the given screen area



- **Intersection Polygon:** In this case a polygon is getting intersected with the screen area i.e. the polygon is partially inside the screen area.



- **Contained Polygon:** In this case a polygon is lying completely inside the given screen area.



- **Disjoint polygon:** Whenever the polygon is lying completely outside the given screen area, it is called as disjoint polygon.