1. Basics of Computer Graphics

1. Give applications of computer graphics.

- **DTP (Desktop Publishing)**
  Used for common paper and book publishing are sometimes used to create graphics for point of sale displays, presentations, infographics, brochures, business cards, promotional items, trade show exhibits, retail package designs and outdoor signs.
- **Graphical User Interface (GUI)**
  The use of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion.
- **Computer-Aided Design**
  Designing of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specifications.
- **Computer-Aided Learning (Cal)**
  Computer Aided Learning (CAL) is the application of computers as an integral part of the learning system for learning and teaching process.
- **Animations**
  Used for creating motion pictures, music video, television shows, **cartoon animation** films.

2. Define Bitmap Graphics

A **bitmap** is an image or shape of any kind-a picture, a text character, a photo-that's composed of a collection of tiny individual dots. A wild landscape on your screen is a **bitmapped** graphic, or simply a bitmap. It is a pixel based image, not scalable and size of image is high.

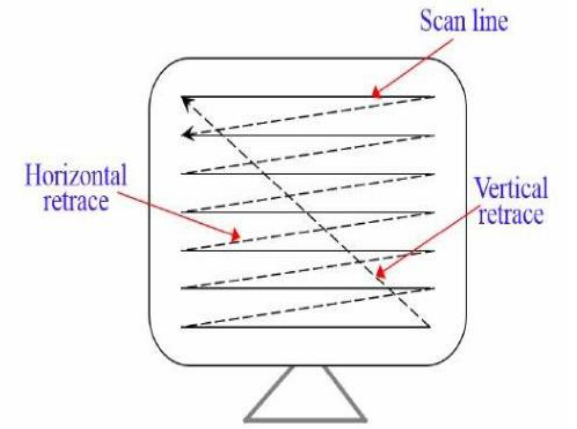
3. Write short note on Augmented Reality.

- Augmented reality (AR) is made up of the word “augment” which means to make something great by adding something to it. **A***ugmented* **R***eality* is a type of virtual reality that aims to duplicate the world's environment in a computer.
- Augmented reality is a method by which we can alter our real world by adding some digital elements to it.
- This is done by superimposing a digital image on the person’s current view thus it enhances the experience of reality.
- For Augmented reality you only need a modern smartphone then you can easily download an AR app like Google’s “**just a line**” and try this technology.
- One of the most popular ways AR has infiltrated everyday life is through mobile games. In 2016, the AR game "Pokémon Go" became a sensation worldwide, with over 100 million estimated users at its peak, according to CNET.

4. Explain with diagram the techniques of Raster Scan Display.

- The most common type of graphics monitor employing a CRT is the Raster-scan displays, based on television technology
- JPG images are raster based. Light occurs when an electron beam stimulates a phosphor.
- In Raster scan, the electron beam from electron gun is swept horizontally across the phosphor one row at time from top to bottom.

- The electron beam sweeps back and forth from left to right across the screen. The beam is on, while it moves from left to right. The beam is off, when it moves back from right to left. This phenomenon is called the horizontal retrace.
- As soon as the beam reaches the bottom of the screen, it is turned off and is rapidly retraced back to the top to start again. This is called the vertical retrace.
- Raster scan displays maintain the steady image on the screen by repeating scanning of the same image. This process is known as refreshing of screen.



5. Define aspect ratio. Give one example of an aspect ratio.

Aspect ratio is the ratio between width of an image and the height of an image.

Example: The term is also used to describe the dimensions of a display resolution. For example, a resolution of 800x600, 1027x768, 1600x1200 has an aspect ratio of 4:3.
Resolution 1280x1024 has an aspect ratio 5:4
Resolution 2160x1440, 2560x1700 has an aspect ratio 3:2

6. Define virtual reality. List any two advantages of virtual reality.

Virtual reality (VR) means experiencing things through our computers that don't really exist.

Advantages:

- Virtual reality creates a realistic world
- Through Virtual Reality user can experiment with an artificial environment.

7. Compare vector scan display and raster scan display

Raster	Vector
Raster graphics are composed of pixels.	Vector graphics are composed of paths.
Raster graphics are resolution dependent.	Vector graphics are resolution independent
More expensive	Less expensive.
Graphics primitives are specified in terms of their endpoints and must be scan converted into their corresponding points in the frame buffer.	Scan conversion is not required
They occupies more space which depends on image quality.	They occupies less space
File extensions are: .bmp, .gif, .jpg, .tif	File extensions are: .pdf, .ai, .svg, .eps, .dxf

8. Define: (i)Pixel (ii)Frame Buffer

Pixel or Pel is defined as "the smallest addressable screen element".

The frame buffer is the video memory (RAM) that is used to hold or map the image displayed on the screen.

9. Compare Bitmap Graphics and Vector based graphics.

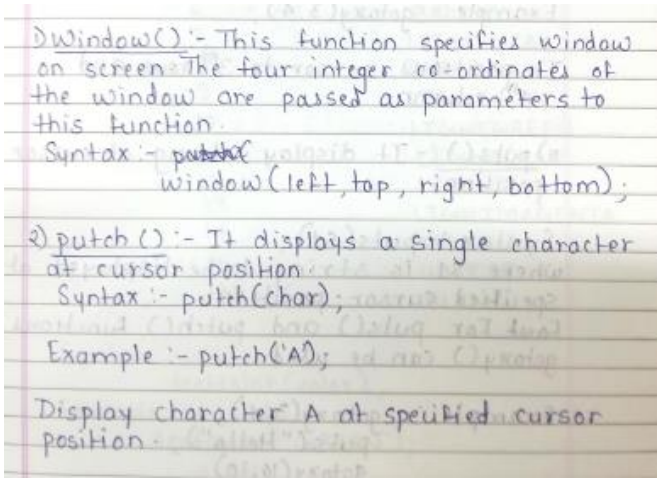
Bitmap Graphics	Vector Based Graphic
It is pixel based image	It is Mathematical based image

Images are resolution dependent.	Images are formula based / dependent.
These images are not easily scalable.	Easily scalable with the help of formula.
Size of image is high.	Size of image is low.

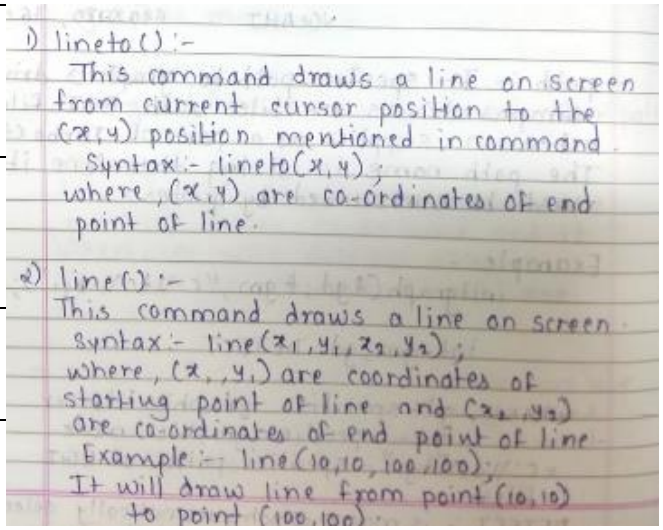
10. List the Graphics Standards.

1)CORE(Core Of a Graphics system.)
2)GKS(Graphics Kernel system)
3)IGES(Initial Graphics Exchange System)
4)PHIGS(Programmer's Hierarchical Interactive Graphics System.)
5)CGM(Computer Graphics Metafile)
6)CGI(Computer Graphics Interface)

11. Explain two text mode graphics function.



12. Explain two graphics mode graphics function.



2. Line , Circle and Polygon

1. List any two line drawing algorithms. Also, list two merits of any line drawing algorithm.

Line drawing algorithms:

- Digital Differential Analyzer (DDA) algorithm
- Bresenham's algorithm

Merits of DDA algorithms:

- It is the simplest algorithm and it does not require special skills for implementation.
- It is a faster method for calculating pixel positions than the direct use of equation y = mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or v direction to find the pixel positions along the line path
- Floating point Addition is still needed.

Merits of Bresenham's Algorithm:

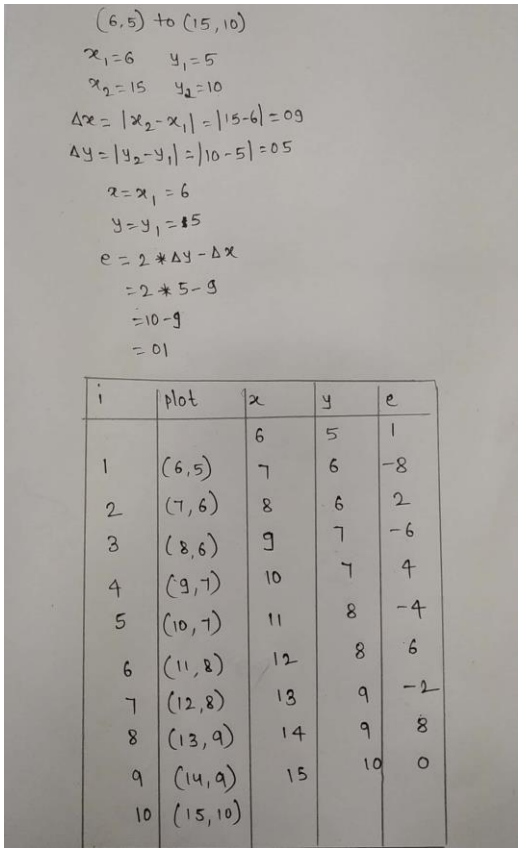
- Bresenhams algorithm is faster than DDA algorithm
- Bresenhams algorithm is more efficient and much accurate than DDA algorithm.

2. Define polygon clipping.

A set of connected lines are considered as polygon; polygons are clipped based on the window and the

portion which is inside the window is kept as it is and the outside portions are clipped.

3. Use Bresenham's line drawing algorithm to rasterize line from (6,5) to (15,10).



(6,5) to (15,10)

$x_1 = 6 \quad y_1 = 5$

$x_2 = 15 \quad y_2 = 10$

$\Delta x = |x_2 - x_1| = |15-6| = 09$

$\Delta y = |y_2 - y_1| = |10-5| = 05$

$x = x_1 = 6$

$y = y_1 = 5$

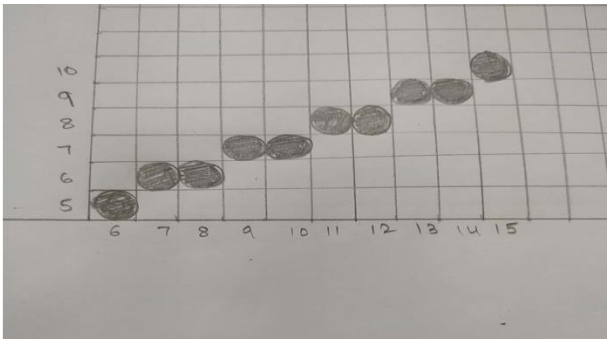
$e = 2 * \Delta y - \Delta x$

$= 2 * 5 - 9$

$= 10 - 9$

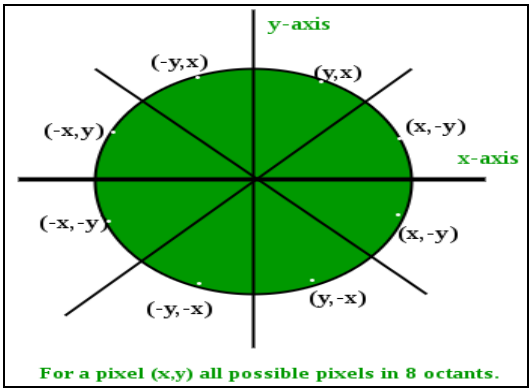
$= 01$

i	plot	x	y	e
		6	5	1
1	(6,5)	7	6	-8
2	(7,6)	8	6	2
3	(8,6)	9	7	-6
4	(9,7)	10	7	4
5	(10,7)	11	8	-4
6	(11,8)	12	8	6
7	(12,8)	13	9	-2
8	(13,9)	14	9	8
9	(14,9)	15	10	0
10	(15,10)			



4. Rephrase the Bresenham's algorithm to plot 1/8th of the circle and write the algorithm required to plot the same.

The key feature of circle that it is highly symmetric. So, for whole 360 degree of circle we will divide it in 8-parts each octant of 45 degree. In order to that we will use Bresenham's Circle Algorithm for calculation of the locations of the pixels in the first octant of 45 degrees. It assumes that the circle is centered on the origin. So for every pixel (x, y) it calculates, we draw a pixel in each of the 8 octants of the circle as shown below:



Algorithm:

Step 1: Read the radius of circle (r).

Step 2: Set decision parameter d = 3 – 2r.

Step 3: x=0 and y=r.

Step 4: do

```
{
    Plot (x,y)
    If(d<0) then
 {
        d = d + 4x + 6
            }
          Else
            {
d=d + 4(x-y) + 10
            y=y-1
            }
          X=x-1
            }
        While(x<y)
```

Step 5: stop

Plotting 8 points, each point in one octant

Call Putpixel (X + h, Y + k).

Call Putpixel (-X + h, Y + k).

Call Putpixel (X + h, -Y + k).

Call Putpixel (-X + h, -Y + k).
Call Putpixel (Y + h, X + k).
Call Putpixel (-Y + h, X + k).
Call Putpixel (Y + h, -X - k).
Call Putpixel (-Y + h,-X + k).

5. Consider the line from (0,0) to (4,6). Use the simple DDA algorithm to rasterize this line

1.

Evaluating steps 1 to 5 in the DDA algorithm we have,

$X_1 = 0, Y_1 = 0$

$X_2 = 4, Y_2 = 6$

Length = $|Y_2 - Y_1| = 6$

$\Delta X = |X_2 - X_1| / \text{Length} = 4/6$

$\Delta Y = |Y_2 - Y_1| / \text{Length} = 6/6 = 1$

Initial value for,

$X = 0 + 0.5 \times (4/6) = 0.5$

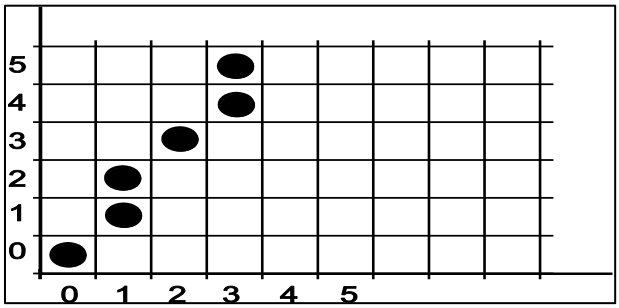
$Y = 0 + 0.5 \times (1) = 0.5$

Plot integer now:

1. Plot (0,0), x=x+ΔX=0.5+4/6=1.167, y=y+ ΔY=0.5+1=1.5
2. Plot (1,1), x=x+ΔX=1.167+4/6=1.833 y=y+ ΔY=1.5+1=2.5
3. Plot (1,2), x=x+ΔX=1.833+4/6=2.5 y=y+ ΔY=2.5+1=3.5
4. Plot (2,3), x=x+ΔX=2.5+4/6=3.167 y=y+ ΔY=3.5+1=4.5
5. Plot (3,4), x=x+ΔX=3.167+4/6=3.833 y=y+ ΔY=4.5+1=5.5
6. Plot (3,5), x=x+ΔX=3.833+4/6=4.5 y=y+ ΔY=5.5+1=6.5

Tabulating the results of each iteration in the step 7 we get,

i	Plot	x	y
		0.5	0.5
1	(0,0)	1.167	1.5
2	(1,1)	1.833	2.5
3	(1,2)	2.5	3.5
4	(2,3)	3.167	4.5
5	(3,4)	3.833	5.5
6	(3,5)	4.5	6.5



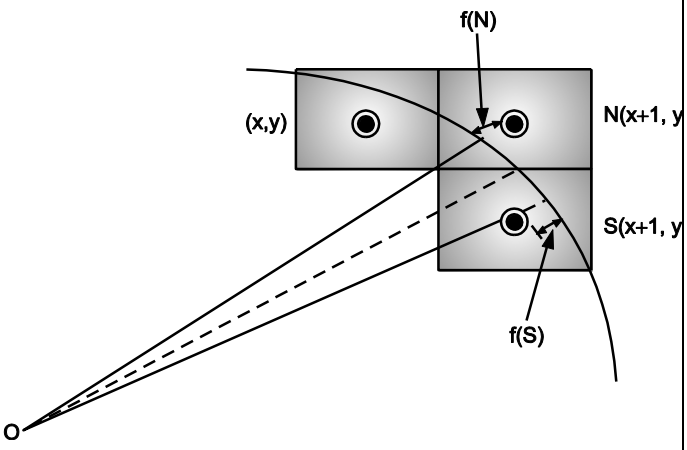
6. Write a Program in 'C' for DDA Circle drawing algorithm

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
 int
gdriver=DETECT,gmode,errorcode,tmp,i=1,rds;
 float st_x,st_y,x1,x2,y1,y2,ep;
 initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
      printf("Enter Radius:");
 scanf("%d",&rds);
 while(rds>pow(2,i))
  i++;
   ep=1/pow(2,i);
 x1=rds;  y1=0;
 st_x=rds; st_y=0;
 do
 { x2=x1+(y1*ep);
  y2=y1-(x2*ep);
  putpixel(x2+200,y2+200,10);
  x1=x2;
  y1=y2;
 }while((y1-st_y)<ep || (st_x-x1)>ep);
```

```
    getch();
}
```

7. Derive the expression for decision parameter used in Bresenham's circle drawing algorithm.

- We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.
- From the following fig, you can see that we have put the pixel at (X, Y) location and now need to decide where to put the next pixel – at N (X + 1, Y) or at S (X + 1, Y – 1).



This can be decided by the decision parameter d.

- If d <= 0, then N(X + 1, Y) is to be chosen as next pixel.
- If d > 0, then S(X + 1, Y – 1) is to be chosen as the next pixel.

8. Write procedure to fill polygon with flood fill.

```
flood_fill(x,y,old_color,new_color)
{
 if(getpixel(x,y) = old_color)
{
 putpixel(x,y,new_color);
flood_fill(x+1,y,old_color, new_color);
flood_fill(x-1,y,old_color, new_color);
flood_fill(x,y+1,old_color, new_color);
flood_fill(x,y-1,old_color, new_color);
flood_fill(x+1,y+1,old_color, new_color);
flood_fill(x-1,y-1,old_color, new_color);
flood_fill(x+1,y-1,old_color, new_color);
flood_fill(x-1,y+1,old_color, new_color);
}
}
```

9. Explain inside and outside test for polygon.

This method is also known as counting number method. While filling an object, we often need to identify whether particular point is inside the object or outside it.

There are two methods by which we can identify whether particular point is inside an object or outside namely, Odd-Even Rule, and Non-zero winding number rule.

1. Odd-Even Rule:

In this technique, we count the edge crossing along the line from any point (x, y) to infinity. If the number of interactions is odd then the point (x, y) is an interior point. If the number of interactions is even then point (x, y) is an exterior point.

Here is the example to give you the clear idea,

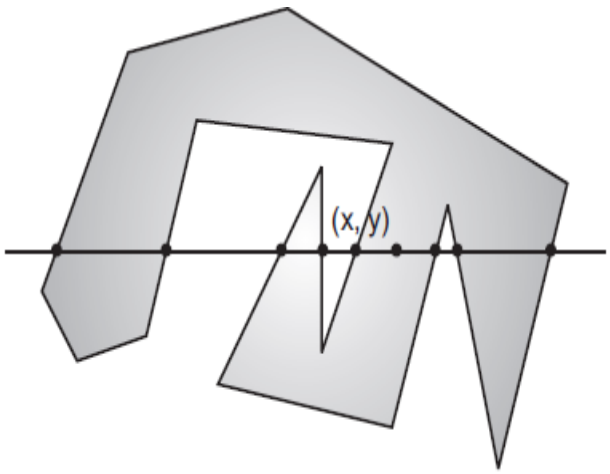


Fig a: Odd-Even Rule

From the Fig., we can see that from the point (x, y), the number of interactions point on the left side

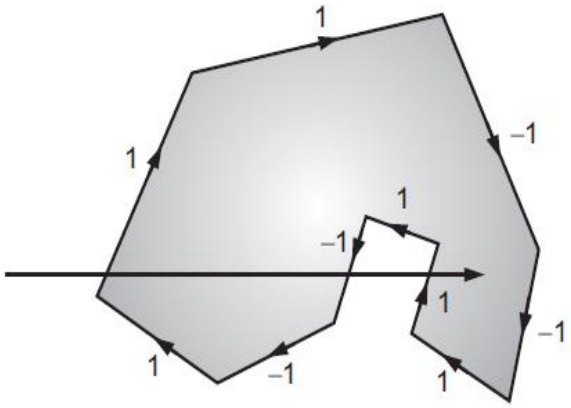
is 5 and on the right side is 3. So the total number of interaction point is 8, which is odd. Hence, the point is considered within the object.

2. Non-zero Winding Number Rule:

This method is also used with the simple polygons to test the given point is interior or not. It can be simply understood with the help of a pin and a rubber band.

Fix up the pin on one of the edge of the polygon and tie-up the rubber band in it and then stretch the rubber band along the edges of the polygon.

When all the edges of the polygon are covered by the rubber band, check out the pin which has been fixed up at the point to be test. If we find at least one wind at the point consider it within the polygon, else we can say that the point is not inside the polygon.



In another alternative method, give directions to all the edges of the polygon. Draw a scan line from the point to be test towards the left most of X direction.

Given the value 1 to all the edges which are going to upward direction and all other – 1 as direction values.

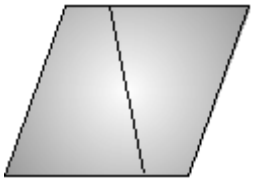
Check the edge direction values from which the scan line is passing and sum up them.

If the total sum of this direction value is non-zero, then this point to be tested is an interior point, otherwise it is an exterior point.

In the above figure, we sum up the direction values from which the scan line is passing then the total is 1 – 1 + 1 = 1; which is non-zero. So the point is said to be an interior point.

10.Define convex and concave polygons.

Convex Polygon: It is a polygon in which if you take any two positions of polygon then all the points on the line segment joining these two points fall within the polygon itself.



Concave Polygon: It is a polygon in which if you take any two positions of polygon then all the points on the line segment joining these two points does not fall entirely within the polygon.

11. State the different character generation methods. Describe any one with diagram.

Character Generator Methods:

1) Stroke Method

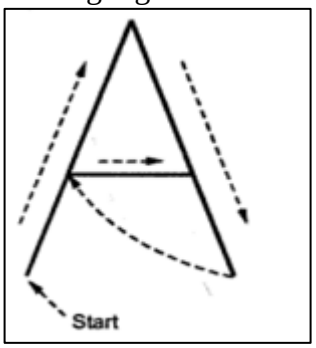
2) Bitmap Method

3) Starburst Method

1) STROKE METHOD

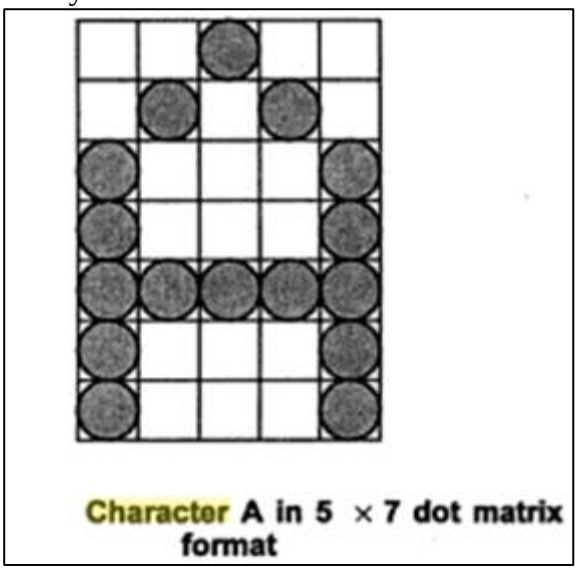
- Stroke method is based on natural method of text written by human being. In this method graph is drawing in the form of line by line.
- This method uses small line segments to generate a character. The small series of line segments are drawn like a stroke of pen to form a character.
- We can build our own stroke method character generator by calls to the line drawing

algorithm. Here it is necessary to decide which line segments are needed for each character and then drawing these segments using line drawing algorithm.



2)BITMAP METHOD

- Bitmap method is a called dot-matrix method as the name suggests this method use array of bits for generating a character. These dots are the points for array whose size is fixed.
- In bit matrix method when the dots is stored in the form of array the value 1 in array represent the characters i.e. where the dots appear we represent that position with numerical value 1 and the value where dots are not present is represented by 0 in array.
- It is also called dot matrix because in this method characters are represented by an array of dots in the matrix form.



3) Starbust method:

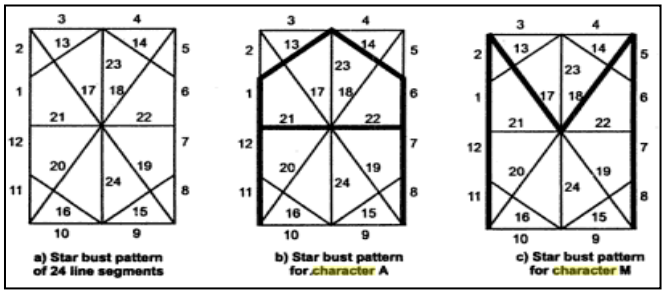
In this method a fix pattern of line segments are used to generate characters. Out of these 24 line segments, segments required to display for particular character are highlighted. This method of character generation is called starbust method because of its characteristic appearance

This method of character generation has some disadvantages. They are

1. The 24-bits are required to represent a character. Hence more memory is required.

2. Requires code conversion software to display character from its 24-bit code.

3. Character quality is poor. It is worst for curve shaped characters.



Character A : 0011 0000 0011 1100 1110 0001

Character M:0000 0011 0000 1100 1111 0011

3. Overview of Transformation

1.What is homogeneous co-ordinate? Why is it required?

We have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra dummy coordinate W. In this way, we can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system.

Homogeneous coordinates are used extensively in computer vision and graphics because they allow

common operations such as translation, rotation, scaling and perspective projection to be implemented as matrix operations

2.Write the transformation matrix for y-shear.

The Y-Shear can be represented in matrix from as:

$$Y_{sh}\begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_X \cdot Y$$

$$Y' = Y$$

3.Translate the polygon with co-ordinates A (3, 6), B (8, 11), & C (11, 3) by 2 units in X direction and 3 units in Y direction.

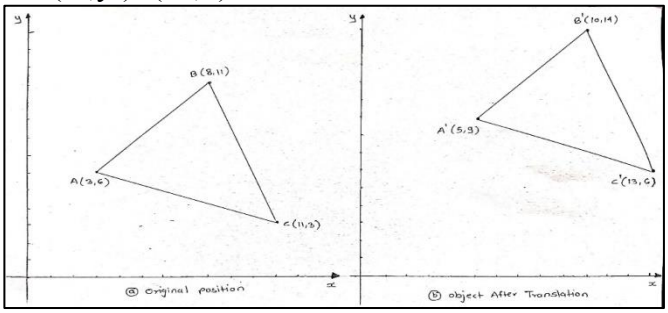
X'=x+tx
Y'=y+ty
tx=2
ty=3

for point A(3,6)
x'=3+2=5
y'=6+3=9

for point B(8,11)
x'=8+2=10
y'=11+3=14

for point C(11,3)
x'=11+2=13
y'=3+3=6

A'=(x',y')=(5,9)
B'=(x',y')=(10,14)
C'=(x',y')=(13,6)



4.Obtain a transformation matrix for rotating an object about a specified pivot point.

To do rotation of an object about any selected arbitrary point P1(x1 ,y1), following sequence of operations shall be performed.

1. Translate: Translate an object so that arbitrary point P1 is moved to coordinate origin.

2. Rotate: Rotate object about origin.

3. Translate: Translate object so that arbitrary point P1 is moved back to the its original position.

Rotate about point P1(x1,y1).

1) Translate P1 to origin.

2) Rotate

3) Translate back to P1.

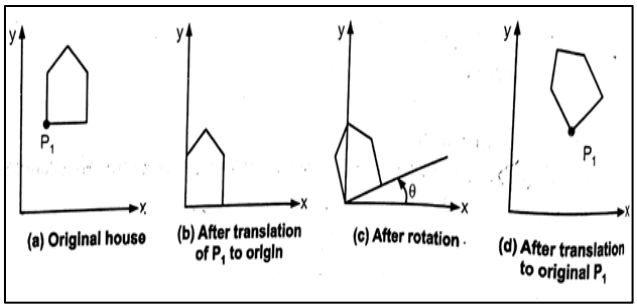
Equation for this composite transformation matrix form is as follows:

$$P' = T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1)$$

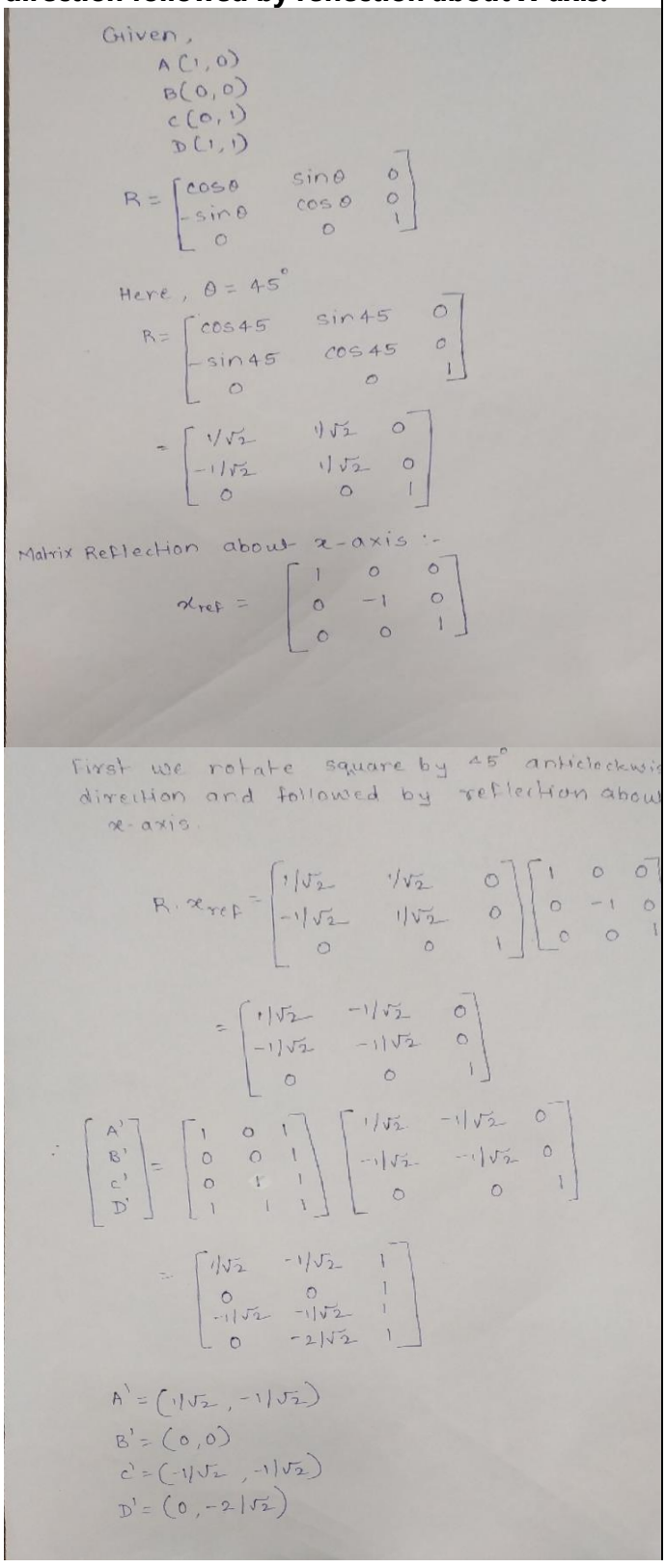
$$P' = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta) + y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta) - x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

It is demonstrated in following figure:



5.Consider a square A(1,0), B(0,0), C(0,1), D(1,1). Rotate the square by 45 anti-clockwise direction followed by reflection about X-axis.



6. Write 2D and 3D scaling matrix.

2D Matrix: $\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$

3D Matrix:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.Explain 2D transformations with its types.

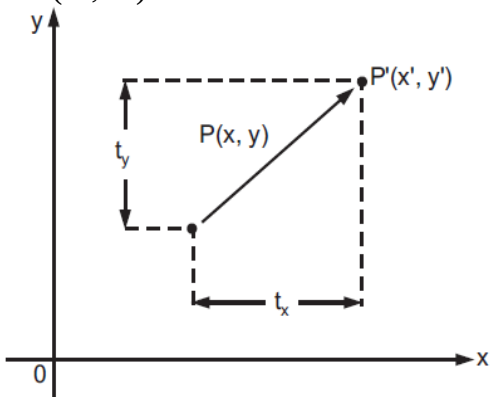
Basic Transformations:

1)Translation
2)Scaling
3)Rotation

1)Translation:

- A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another.

- Translation can be defined as "the process of repositioning an object along a straight line path from one co-ordinate location to new co-ordinate location."
- A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X', Y')



From the above Fig. you can write that:

X' = X + tx

Y' = Y + ty

The pair (tx, ty) is called the translation vector or shift vector.

Rotation

- Rotation as the name suggests is to rotate a point about an axis. The axis can be any of the co-ordinates or simply any other specified line also.
- In rotation, we rotate the object at particular angle θ (theta) from its origin. From the following figure, we can see that the point P(X, Y) is located at angle φ from the horizontal X coordinate with distance r from the origin.
- Let us, suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P' (X', Y').

Using standard trigonometric the original coordinate of point P(X, Y) can be represented as:

$X = r \cos \phi$ (1)

$Y = r \sin \phi$ (2)

Same way we can represent the point P' (X', Y') as:

$x' = r \cos(\phi + \theta) = r \cos \phi \cos \theta - r \sin \phi \sin \theta$ (3)

$y' = r \sin(\phi + \theta) = r \cos \phi \sin \theta + r \sin \phi \cos \theta$ (4)

Substituting equation (1) and (2) in (3) and (4) respectively, we will get

$x' = x \cos \theta - y \sin \theta$

$y' = x \sin \theta + y \cos \theta$

Representing the above equation in matrix form,

$$[X'\ Y'] = [X'\ Y'] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

OR

$$P' = P \cdot R$$

Where, R is the rotation matrix

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

The rotation angle can be positive and negative.

Scaling:

Scaling means to change the size of object. This change can either be positive or negative.

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object.

Scaling can be achieved by multiplying the original co-ordinates of the object with the scaling factor to get the desired result.

Let us assume that the original co-ordinates are (X, Y), the scaling factors are (SX, SY), and the produced co-ordinates are (X', Y'). This can be mathematically represented as shown below:

X' = X .SX and Y' = Y .SY

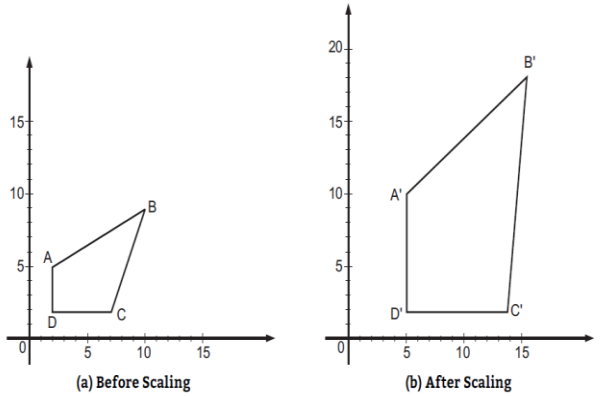
The scaling factor SX, SY scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

OR

$$P' = P \cdot S$$

Where, S is the scaling matrix.



(a) Before Scaling (b) After Scaling

If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

8.Apply shearing transformation to square with A(0,0), B(1,0), C(1,1) and D(0,1) as shear parameter value of 0.5 relative to the line Y_{ref} = -1 and X_{ref}=-1.

a) Here $Sh_x = 0.5$ and $y_{ref} = -1$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ -Sh_x \cdot y_{ref} & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0 & 1 \end{bmatrix}$$

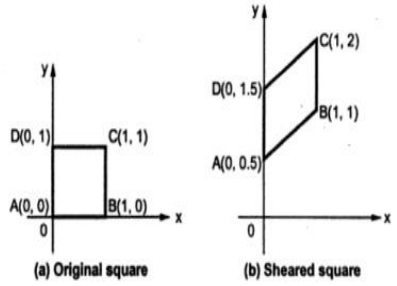
$$= \begin{bmatrix} 0.5 & 0 & 1 \\ 1.5 & 0 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

b) Here $Sh_y = 0.5$ and $x_{ref} = -1$

$$\therefore \begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & -Sh_y \cdot x_{ref} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1.5 & 1 \end{bmatrix}$$

It is important to note that shearing operations can be expressed as sequence of basic transformations. The sequence of basic transformations involve series of rotation and scaling transformations.



9.Consider the square A (1, 0), B (0, 0), C (O, 1), D (1, I). Rotate the square ABCD by 45° anticlockwise about point A (1. 0).

$$\begin{bmatrix} \cos\theta & \sin\theta\theta \\ -\sin\theta\theta & \cos\theta \\ -Xp\cos\theta + Yp\sin\theta + Xp & -Xp\sin\theta - Yp\cos\theta + Yp \end{bmatrix}$$

Here, $\theta = 45°$, $X_p = 1$ $Y_p = 0$

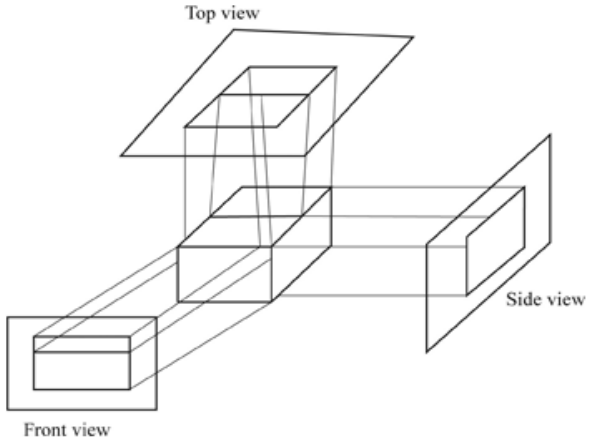
$$[T_1.R.T_2] = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} + 1 & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ \frac{-1}{\sqrt{2}}+1 & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 \\ \frac{-1}{\sqrt{2}}+1 & -\frac{1}{\sqrt{2}} & 1 \\ 1-\sqrt{2} & 0 & 1 \\ 1-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 \end{bmatrix}$$

10.Explain types of Parallel Projection with example.

- Orthographic projection – the projection direction is a normal one to the plane and it is categorized as
  - Top projection
  - Front projection
  - Side projection



- Oblique projection – the projection direction is not a normal one to the plane; it gives a better view and it is categorized as

○ Cavalier projection

○ Cabinet projection

4. Windowing and Clipping

1.Write the midpoint subdivision algorithm for line clipping.

Step 1: Scan two end points for the line $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.

Step 2: Scan corners for the window as $(\omega x_1, \omega y_1)$ and $(\omega x_2, \omega y_2)$.

Step 3: Assign the region codes for endpoints P_1 and P_2 by initializing code with 0000.

Bit 1 - if $(x < \omega x_1)$

Bit 2 - if $(x > \omega x_2)$

Bit 3 - if $(y < \omega y_2)$

Bit 4 - if $(y > \omega y_1)$

Step 4: Check for visibility of line P_1, P_2.

- If region codes for both end points are zero then the line is visible, draw it and jump to step 6.
- If region codes for end points are not zero and the logical Anding operation of them is also not zero then the line is invisible, reject it and jump to step 6.
- If region codes for end points does not satisfies the condition in 4 (i) and 4 (ii) then line is partly visible.

Step5: Find midpoint of line and divide it into two equal line segments and repeat steps 3 through 5 for both subdivided line segments until you get completely visible and completely invisible line segments.

Step 6: Exit.

2.Describe Sutherland-Hodgeman algorithm for polygon clipping.

- In Sutherland-Hodgeman, a polygon is clipped by processing the polygon boundary as a whole against each window edge. Clipping window must be convex.
- This could be accomplished by processing all polygon vertices against each clip rectangle boundary in turn beginning with the original set of polygon vertices, first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.
- The new set of vertices could then be successively passed to a right boundary clipper, a top boundary clipper and a bottom boundary clipper.
- At each step a new set of polygon vertices us generated and passed to the next window boundary clipper. This is the logic used in Sutherland-Hodgeman algorithm.

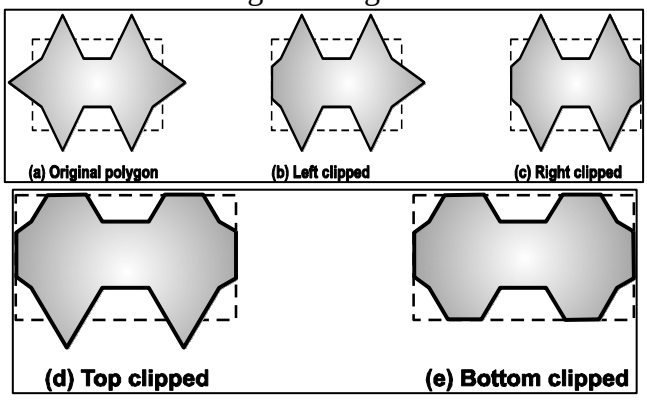


Fig. Clipping polygon against successive window boundaries

Algorithm for Sutherland-Hodgeman Polygon Clipping:

Step 1: Read co-ordinates of all vertices of the polygon.

Step 2: Read co-ordinates of the clipping window.

Step 3: Consider the left edge of window.

Step 4: Compare vertices of each of polygon, individually with the clipping plane.

Step 5: Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary.

Step 6: Repeat the steps 4 and 5 for remaining edges of clipping window. Each time

resultant list of vertices is successively passed to process next edge of clipping window.

Step 7: Stop.

3.Apply the Liang-Barsky algorithm to the line with co-ordinate (30,60) & (60,25) against the window:
(Xmin, Ymin) = (10.10) & (Xmax,Ymax) = (50,50)
Given:

$(X_{min},Y_{min})=(10,10)$ and $(X_{max},Y_{max})=(50,50)$
P1 (30, 60) and P2 = (60, 25)

Solution:

Set Umin = 0 and Umax = 1

ULeft= q1 / p1
= X1 – Xmin / - Δ X

= 30 – 10 / - (60 - 30)

= 20 / - 30

= -0.67

URight= q2 / p2
= Xmax – X1/ ΔX

= 50 – 30 / (60 - 30)

= 20 / 30

= 0.67

UBottom= q3 / p3
= Y1 – Ymin / - ΔY

= 60 – 10 / - (25 – 60)

= 50 / 35

= 1.43

UTop= q4 / p4
= Ymax – Y1 / ΔY

= 50 – 60 / (25 - 60)

= -10 / - 35

= 0.29

Since ULeft=−0.57which is less than Umin. Therefore we ignore it.

Similarly UBottom=1.43 which is greater than Umax. So we ignore it.

URight=Umin = 0.67 (Entering)
UTop=Umax = 0.29 (Exiting)
We have UTop= 0.29 and URight= 0.67
Q – P = (ΔX, ΔY) = (30, -35)

Since Umin>Umax, there is no line segment to draw.

4. Explain Text Clipping.

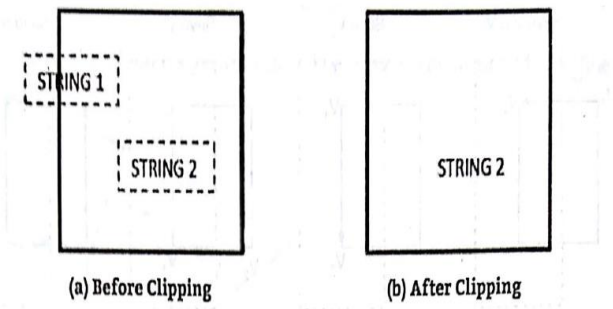
Many techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application. There are three methods for text clipping which are listed below –

1) All or none string clipping

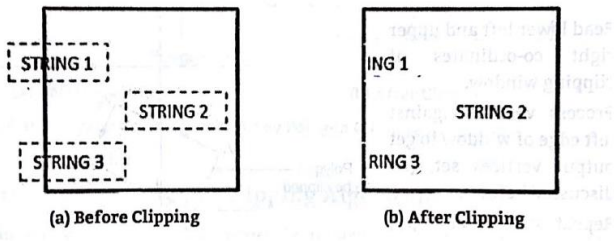
2) All or none character clipping

3) Text clipping

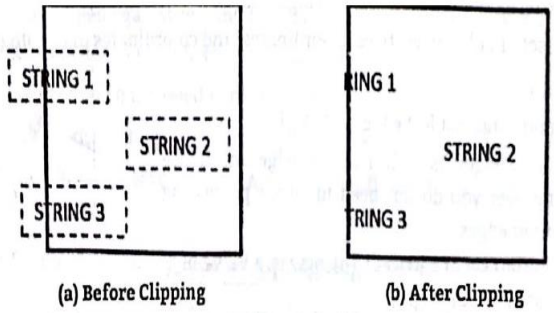
The following figure shows all or none string clipping –



In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figure, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.
The following figure shows all or none character clipping –



This clipping method is based on characters rather than entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then −
You reject only the portion of the string being outside If the character is on the boundary of the clipping window, then we discard that entire character and keep the rest string.
The following figure shows text clipping –



This clipping method is based on characters rather than the entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then you reject only the portion of string being outside. If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window.

5.Define polygon clipping.
A set of connected lines are considered as polygon; polygons are clipped based on the window and the portion which is inside the window is kept as it is and the outside portions are clipped.

6.Write down Cohen-Sutherland Line clipping algorithm.

Step 1: Scan end points for the line P1(x1, y1) and P2(x2, y2)

Step 2: Scan corners for the window as (Wx1, Wy1) and (Wx2, Wy2)

Step 3: Assign the region codes for endpoints P1 and P2 by

Bit 1 - if (x < Wx1)

Bit 2 - if (x < Wx2)

Bit 3 - if (x < Wy2)

Bit 4 - if (x < Wy1)

Step 4: Check for visibility of line P1, P2

- If region codes for both end points are zero then the line is visible, draw it and jump to step 9.
- If region codes for end points are not zero and the logical and operation of them is also not zero then the line is invisible, reject it and jump to step 9.
- If region codes for end points does not satisfies the condition in 4(i) and 4(ii) then line is partly visible.

Step 5: Determine the intersecting edge of the clipping window by inspecting the region codes for endpoints.

- If region codes for both the end points are non-zero, find intersection points P1 and P2 with boundary edges of clipping window with respect to point P1 and P2.
- If region code for any one end point is non zero then find intersection point P1 or P2 with the boundary edge of the clipping window with respect to it.

Step 6: Divide the line segments by considering intersection points.

Step 7: Reject the line segment if any of the end point of it appear outside the window.

Step 8: Draw the remaining line.

Step 9: Exit

7.Use Cohen-Sutherland algorithm to clip two lines PI (40, 15) -- P2 (75. 45) and P3 (70, 20) — P4 (100, 10) against a window A (50, 10), B (80, 10). C(80, 40) & D(50,40)

Line 1 : P1 (40, 15) - P2 (75, 45) Wxi = 50 Wy2 = 40 Wx2 = 80 Wy2 = 10

Point	Endeode	ANDing
P1	0001 (Partially visible)	0000
P2	0000	

$y_1 = m(x_L - x) + y = \frac{6}{7}(50-40)+15$

$m = \frac{45-15}{75-40}$

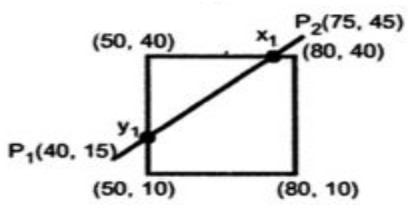
$= 23.57$

$x_1 = \frac{1}{m}(y_T - y) + x = \frac{7}{6}(40-50)+40 = 69.16$

$y_2 = m(x_R - x) + y = \frac{6}{7}(80-40)+15 = 49.28$

$x_2 = \frac{1}{m}(y_B - y) + x = \frac{7}{6}(10-15)+40 = 34.16$

Hence:



Line 2 : P3 (70,20) – P4 (100,10) Wxi = 50 Wy2 = 40 Wx2 = 80 Wy2 = 10

Point	Endeode	ANDing
P3	0000 (Partially visible)	0000
P4	0010	

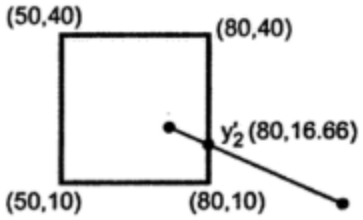
Slope $m` = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$

$y`_1 = m(x_L - x) + y = \frac{-1}{3}(50-70)+20 = 26.66$

$x`_1 = \frac{1}{m}(y_T - y) + x = -3(40-20)+70 = 10$

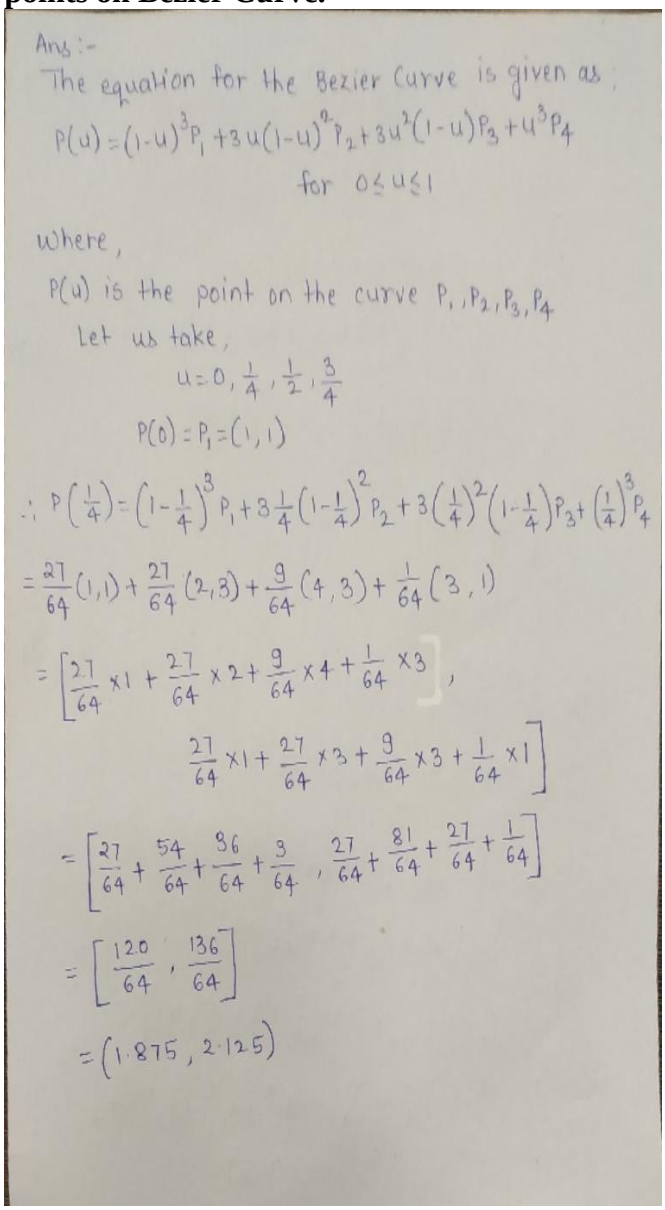
$y`_2 = m(x_R - x) + y = \frac{-1}{3}(80-70)+20 = 16.66$

$x`_2 = \frac{1}{m}(y_B - y) + x = -3(10-20)+70 = 100$



5.Introduction to Curve

1)Given the vertices of Bezier Polygon as $P_0(1, 1)$, $P_1(2,3)$, $P_2(4,3)$, $P_3(3,1)$, determine five points on Bezier Curve.



Ans :-

The equation for the Bezier Curve is given as :

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u)P_3 + u^3 P_4$$

for $0 \le u \le 1$

where,

P(u) is the point on the curve P_1, P_2, P_3, P_4

Let us take,

$u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

$P(0) = P_1 = (1, 1)$

$$\therefore P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 P_1 + 3\frac{1}{4}\left(1-\frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right)P_3 + \left(\frac{1}{4}\right)^3 P_4$$

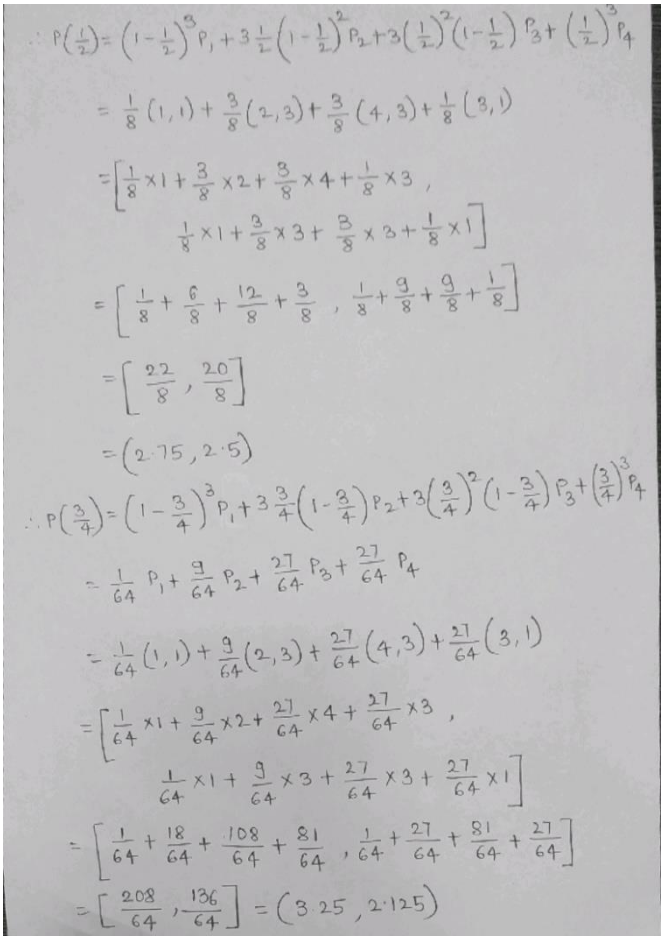
$$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(3,1)$$

$$= \left[\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 3 \;,\; \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 1\right]$$

$$= \left[\frac{27}{64} + \frac{54}{64} + \frac{36}{64} + \frac{3}{64} \;,\; \frac{27}{64} + \frac{81}{64} + \frac{27}{64} + \frac{1}{64}\right]$$

$$= \left[\frac{120}{64}, \frac{136}{64}\right]$$

$$= (1.875, 2.125)$$



$$\therefore P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 P_1 + 3\frac{1}{2}\left(1-\frac{1}{2}\right)^2 P_2 + 3\left(\frac{1}{2}\right)^2\left(1-\frac{1}{2}\right)P_3 + \left(\frac{1}{2}\right)^3 P_4$$

$$= \frac{1}{8}(1,1) + \frac{3}{8}(2,3) + \frac{3}{8}(4,3) + \frac{1}{8}(3,1)$$

$$= \left[\frac{1}{8}\times 1 + \frac{3}{8}\times 2 + \frac{3}{8}\times 4 + \frac{1}{8}\times 3 \;,\; \frac{1}{8}\times 1 + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}\times 1\right]$$

$$= \left[\frac{1}{8} + \frac{6}{8} + \frac{12}{8} + \frac{3}{8} \;,\; \frac{1}{8} + \frac{3}{8} + \frac{9}{8} + \frac{1}{8}\right]$$

$$= \left[\frac{22}{8}, \frac{20}{8}\right]$$

$$= (2.75, 2.5)$$

$$\therefore P\left(\frac{3}{4}\right) = \left(1-\frac{3}{4}\right)^3 P_1 + 3\frac{3}{4}\left(1-\frac{3}{4}\right)P_2 + 3\left(\frac{3}{4}\right)^2\left(1-\frac{3}{4}\right)P_3 + \left(\frac{3}{4}\right)^3 P_4$$

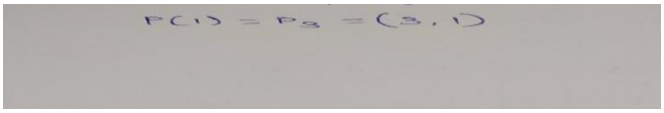
$$= \frac{1}{64}P_1 + \frac{9}{64}P_2 + \frac{27}{64}P_3 + \frac{27}{64}P_4$$

$$= \frac{1}{64}(1,1) + \frac{9}{64}(2,3) + \frac{27}{64}(4,3) + \frac{27}{64}(3,1)$$

$$= \left[\frac{1}{64}\times 1 + \frac{9}{64}\times 2 + \frac{27}{64}\times 4 + \frac{27}{64}\times 3 \;,\; \frac{1}{64}\times 1 + \frac{9}{64}\times 3 + \frac{27}{64}\times 3 + \frac{27}{64}\times 1\right]$$

$$= \left[\frac{1}{64} + \frac{18}{64} + \frac{108}{64} + \frac{81}{64} \;,\; \frac{1}{64} + \frac{27}{64} + \frac{81}{64} + \frac{27}{64}\right]$$

$$= \left[\frac{208}{64}, \frac{136}{64}\right] = (3.25, 2.125)$$

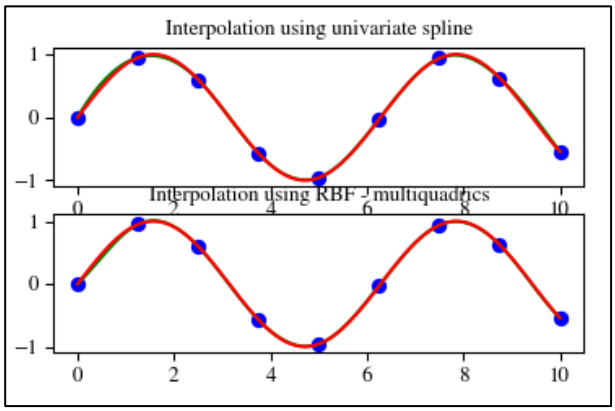


$P(1) = P_4 = (3, 1)$

2)What is interpolation? Describe the Lagrangian Interpolation method.

Specify a spline curve by giving a set of coordinate positions, called control points, which indicates the general shape of the curve These, control points are then fitted with piecewise continuous parametric polynomial functions in one of two ways. When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said to interpolate the set of control points. On the other hand, when the polynomials are fitted to the general control-point path without necessarily passing through any control point, the resulting curve is said to approximate the set

of control points interpolation curves are commonly used to digitize drawings or to specify animation paths. Approximation curves are primarily used as design tools to structure object surfaces an approximation spline surface credited for a design application. Straight lines connect the control-point positions above the surface



Lagrangian Interpolation Method:

Suppose we want a polynomial curve that will pass through n sample points - $x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)$, the function can be constructed as the sum of terms, one term for each sample point.

a. Blending Function :

$$fx(u) = \sum_{i=1}^{n} x_i B_i(u)$$

$$fy(u) = \sum_{i=1}^{n} y_i B_i(u)$$

$$fz(u) = \sum_{i=1}^{n} z_i B_i(u)$$

The function $B_i(u)$ is called as a blending function. For each value of u, the blending function determines which i^{th} sample point affects the position of the curve.

The function $B_i(u)$ tells how hard the i^{th} sample point is pulling it for some value of u, $B_i(u) = 1$ and for each $j \neq i$, $B_j(u) = 0$, then i^{th} sample point has complete control of the curve. The curve will pass through i^{th} sample point. Create a blending function for which the sample points (x_1, y_1, z_1) has complete control when $u = -1$, the third when $u = 1$ and so on. Therefore, we require a blending function.

$B_1(u) = 1$ at $u = -1$

and $B_1(u) = 0$ at $u = 0, 1, 2, 3, \ldots, n-2$

An expression is 0 at $u (u-1)(u-2) \ldots [u-(n-2)]$

At $u = -1$, it is $(-1)(-2)(-3) \ldots (1-n)$

So dividing by above constant, it gives 1 at $u = -1$

Therefore

$$B_1(u) = \frac{u(u-1)(u-2)\ldots[(u-(n-2)]}{(-1)(-2)(-3)\ldots(1-n)}$$

The i^{th} blending function can be constructed in the same way to be 1 at $u = i$ – 2 and 0 at other integers.

$$\therefore B_i(u) = \frac{(u+1)(u)(u-1)\ldots[u-(i-3)][u-(i-1)]\ldots[u-(i-2)]}{(i-1)(i-2)(i-3)\ldots(1)(-1)\ldots(i-n)}$$

The curve which is approximated using above equation is called **Lagrange Interpolation.**

3)Write a program in 'C' to generate Hilbert's curve.

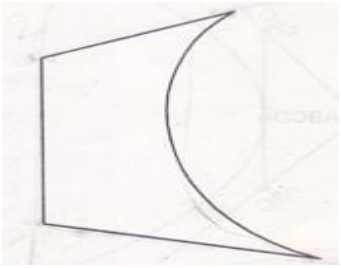
```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>
void move(int j,int h,int &x,int &y)
{
      if(j==1)
      y-=h;
      else if(j==2)
      x+=h;
      else if(j==3)
      y+=h;
      else if(j==4)
      x-=h;
      lineto(x,y);
}

void hilbert(int r,int d,int l,int u,int i,int h,int &x,int &y)
{
      if(i>0)
      {
            i--;
            hilbert(d,r,u,l,i,h,x,y);
            move(r,h,x,y);
            hilbert(r,d,l,u,i,h,x,y);
            move(d,h,x,y);
            hilbert(r,d,l,u,i,h,x,y);
            move(l,h,x,y);
            hilbert(u,l,d,r,i,h,x,y);
      }
}

int main()
{
      int n,x1,y1;
      int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;

      printf)"\nGive the value of n: ");
      scanf("%d",&n);
      x=x0;y=y0;
      int gm,gd=DETECT;
      initgraph(&gd,&gm,NULL);
      moveto(x,y);
      hilbert(r,d,l,u,n,h,x,y);
      delay(10000);
      closegraph();
      return 0;
}
```

4)Draw Cubic Bezier Curve.



5)Explain Koch curve with diagram.

Koch Curve: - In Koch curve, begin at a line segment. Divide it into third and replace the center by the two adjacent sides of an equilateral triangle as shown below. This will give the curve which starts and ends at same place as the original segment but is built of 4 equal length segments, with each 1/3rd of the original length. So the new curve has 4/3 the length of original segments. Repeat same process for each of the 4 segment which will give curve more wiggles and its length become 16/9 times the original. Suppose repeating the replacements indefinitely, since each repetition increases the length by a factor of 4/3, the length of the curve will be infinite but it is folded in lots of tiny