

## Unit 2 RASTER SCAN GRAPHICS( marks 18)

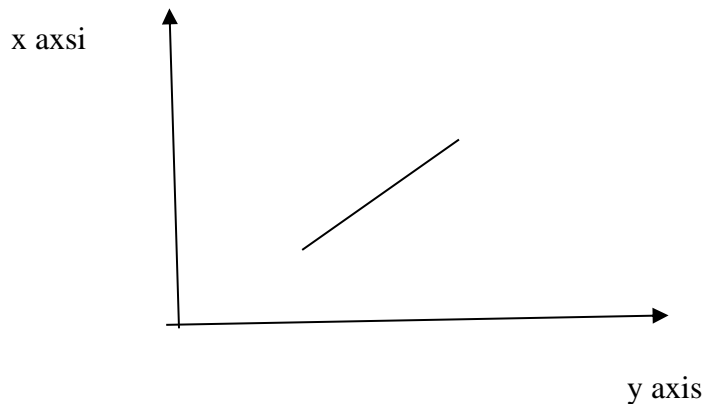
- 2.1 Basic concepts in line drawing line
  - 2.2 Line drawing algorithms
    - 2.2.1 DDA
    - 2.2.2 Bresenham's algorithm
  - 2.3 Bresenham's Circle generation algorithm
  - 2.4 polygon
  - 2.5 Polygon filling
    - 2.5.1 Scan converting polygon
    - 2.5.2 Edge fill
    - 2.5.3 Edge flag
    - 2.5.4 Seed fill
- Scan conversion , frame buffers

### 2.1 LINE DRAWING ALGORITHMS

Line is straight object with no curves no thickness and it extends in both directions without end if it doesn't ends it is called a line segment

A vector is a quantity having direction as well as magnitude specially as determining the position of one point in space relative to other

the process of turning on the pixel for a line segment is called vector generation or line generation and the algorithms or line drawing algorithm



- Slope –intercept line equation
- $y = mx + b$
- Given two end points  $(x_0, y_0), (x_1, y_1)$  how to compute  $m$
- $M = \frac{dy}{dx} = \frac{y_1 - y_0}{x_1 - x_0}$

#### Direct Use of the Line Equation

A line can be scan converted by following the steps given below:

1. Scan-convert the endpoints P1 and P2 to pixel coordinates  $(x_1, y_1)$  and  $(x_2, y_2)$  respectively
2. Set  $m = (y_2 - y_1) / (x_2 - x_1)$  and  $b = y_1 - m \cdot x_1$

3. If  $|m| \leq 1$ , calculate the corresponding value of  $y$  using the equation and scan-convert  $(x, y)$  for every integer value of  $x$  between and excluding  $x_1$  and  $x_2$
4. If  $|m| > 1$ , calculate corresponding value of  $x$  using the equation and scan-convert  $(x, y)$  for every integer value of  $y$  between and excluding  $y_1$  and  $y_2$

### in computer graphics popular algorithm used to generate line are

- line drawing algorithm:
- DDA
- Bresenham line drawing algorithm
- Mid point line drawing algorithm

#### 2.1.1 DDA Algorithm

- Digital differential analyzer (DDA)
- Line drawing algorithm in computer graphics
- DDA algorithm is the simplest line drawing algorithm
- Working

given the starting and ending coordinates of a line

DDA algorithm attempts to generate the points between the starting and ending coordinates.

- Procedure:

- step 01

given

Starting coordinates =  $(x_0, y_0)$

Ending coordinates =  $(x_n, y_n)$

the points generation using DDA algorithm involves the following steps:

- Step-02
- Calculate  $\Delta X(dx)$ ,  $\Delta Y(dy)$  and  $m$  from the given input

These parameters are calculated as

- $\Delta X = x_n - x_0$
- $\Delta Y = y_n - y_0$
- $M = \Delta y / \Delta x$

- Step-03

- Find the number of steps or pointers in between the starting and ending coordinates if  $(\text{absolute}(\Delta X) > \text{absolute}(\Delta Y))$

Steps =  $\text{absolute}(\Delta X)$

Else

steps =  $\text{absolute}(\Delta Y)$

- Step-04

- Suppose the current point is  $(x_n, y_n)$  And the next point is  $(x_{p+1}, y_{p+1})$

Find the next point following the below three case:

- case 01 : If  $m < 1$   
 $X_{n+1} = \text{round off}(1 + x_p)$   
 $Y_{n+1} = \text{round off}(m + y_p)$
- Case02 :if  $m = 1$   
 $X_{n+1} = \text{roundoff}(1 + x_n)$   
 $Y_{p+1} = \text{roundoff}(1 + y_n)$
  - Case-03  
 $X_{p+1} = \text{roundoff}(1/m + x_p)$   
 $Y_{p+1} = \text{roundoff}(1 + y_p)$
  - Step5  
 Keep repeating step 04 until the end point is reached or the number of generated new points (including the starting and ending points) equals to the steps count

**Advantage:** The DDA algorithm is faster than the direct use of the line equation. This is because DDA algorithms calculate points on the line without any floating-point multiplication.

It is simple algorithm

It is easy to implement

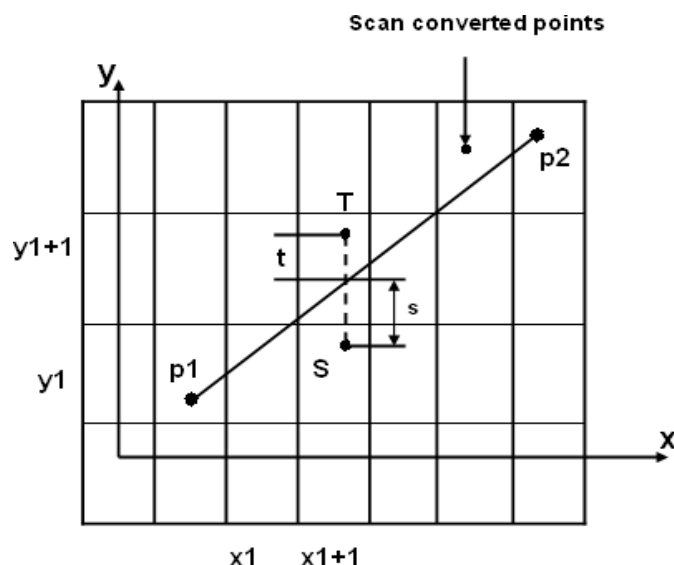
### 2.1.1 Bresenham's Line Algorithm

Bresenham's line algorithm is one of the highly efficient incremental methods for scan-converting lines.

It is an algorithm that determines which points in an n-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points. It is an accurate and efficient raster line-generating algorithm that uses only incremental integer calculations.

Bresenham's algorithm can be used to display lines, circles and other curves. The vertical axes show scan-line positions, and the horizontal axes identify pixel columns, that is, Bresenham's algorithm can be successfully used for drawing lines on a computer screen, subtracting, and bit shifting.

Bresenham's line algorithm works as follows. Assume that you want to scan-convert the line shown in figure, where  $0 < m < 1$ . For this, start with pixel  $p_1 (x_1, y_1)$ , then select the subsequent



pixels. After the pixels are chosen at any step, the next pixel is either the one to its right and down or one to its right and up due to the limit on m.

- given the starting and ending coordinates of a line
- Bresenham line drawing algorithm attempts to generate the point between the starting and ending coordinates
- The point generation using bresenham line drawing algorithm involves the following steps
- Step 01
- Given
- starting coordinates  $= (x_0, y_0)$
- Ending coordinates  $= (x_n, y_n)$
- Step-02
- Calculate  $\Delta X$  and  $\Delta y$  from the given input these parameters are calculated as
- $\Delta X = x_n - x_0$
- $\Delta y = y_n - y_0$
- step-03
- Calculate the decision parameter  $P_k$
- it is calculated as  $p_k$
- $p_k = 2 \Delta y - \Delta X$
- Step 04
- suppose the current point is  $(x_k, y_k)$
- 
- And the next point is  $(x_{k+1}, y_{k+1})$
- Find the next point depending on the value of decision parameter  $p_k$
- Follow the below two case:
- Case no1:  $p_k < 0$

$$P_{k+1} = p_k + 2 \Delta y$$

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

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

- Case -02:  $p_k \geq 0$

$$p_{k+1} = p_k + 2 \Delta y - 2 \Delta X$$

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

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

Step-05

Keep repeating step-4 until the end point is reached

### advantages

- It is easy implement
- It is fast and incremental
- It is faster than DDA algorithm
- It use fixed points only

## 2.2 Bresenham's Circle Generation Algorithm

A circle is a symmetrical figure. Any circle-generating algorithms may be used to plot eight points for each value that the algorithm calculates. As shown in the figure 4.3, only one octant of the circle has to be generated. A successive reflection obtains the other parts. If the first octant (0 to 45°) is generated the second octant can be obtained by reflection through the line  $y = x$  to yield the first quadrant. The results in the first quadrant are returned through line  $x = 0$ , to obtain those in the second quadrant.

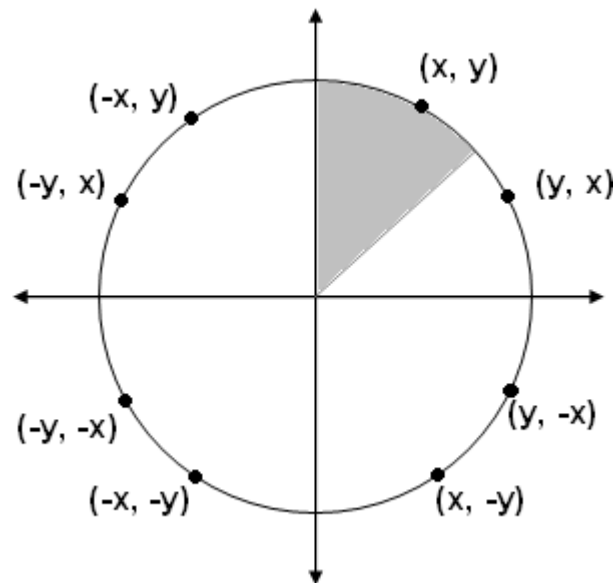


Figure 4.3: Eight-way Symmetry of a Circle

Equation of a circle with (0, 0) as its centre is given by:

$$F_{\text{circle}}(X, Y) = X^2 + Y^2 - R^2$$

Checking the sign of the circle function determines the relative position of any point (X, Y), that is,

$F_{\text{circle}}(X, Y) < 0$ , if (X, Y) is inside the circle boundary

$F_{\text{circle}}(X, Y) = 0$ , if (X, Y) is on circle boundary

$F_{\text{circle}}(X, Y) > 0$ , if (X, Y) is outside the circle boundary

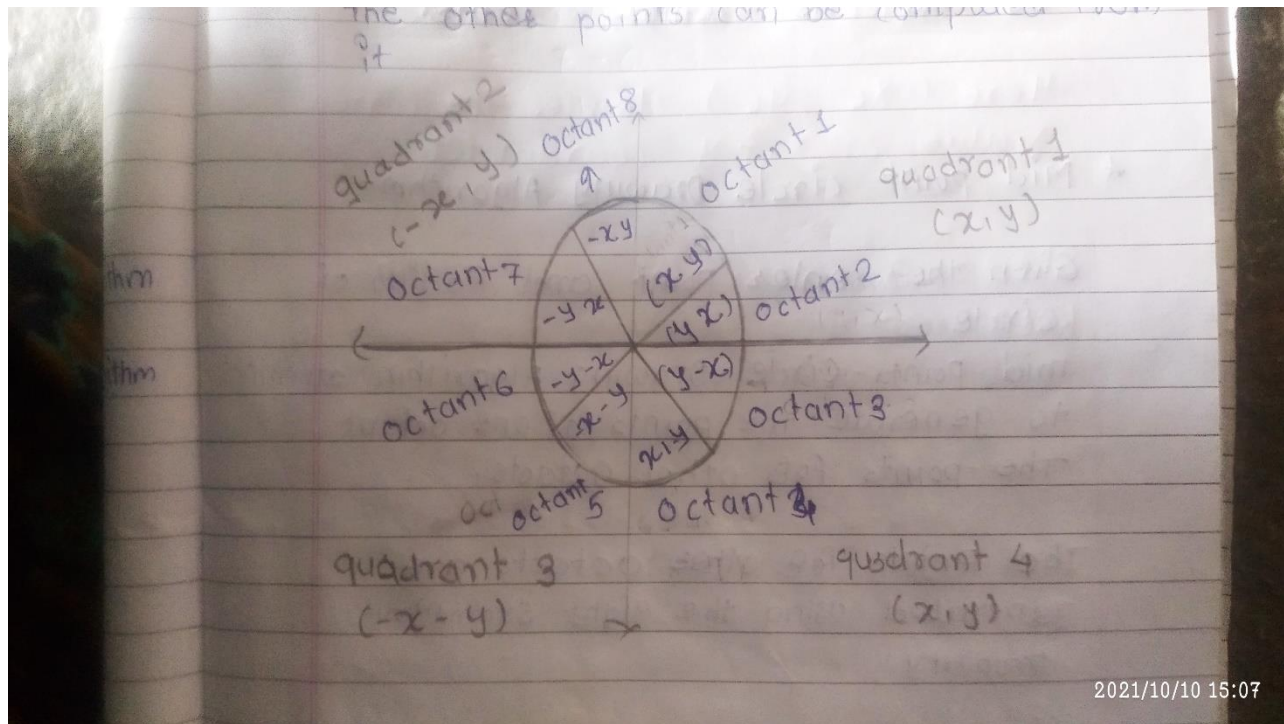
In computer graphic popular algorithm used to generate circle are:

Circle drawing algorithms:

1. Mid point circle drawing algorithm
2. bresenham's circle drawing algorithm
3. Symmetry of circle:

a circle is a geometric figure which is round and can be divided into 360 degrees a circle is a symmetrical figure which follows 8-way symmetry

- **8-way symmetry** : any circle follows 8-way symmetry this means that for every point  $(x,y)$  8 point can be plotted these  $(x,y), (y,x), (-y,x), (-x,y), (-x,-y), (-y,-x), (y,-x), (x,-y)$
- For any point  $(x+a,y+b)$  points  $((x+a)(x-a), (y+a)(y-a))$  and  $((y+b)(y-b), (x+b)(x-b))$  also lie on the same circle so it is sufficient to computer only 1/8 of a circle and all the other point can be computed from it



- drawing a circle – to draw a circle we need two things the coordinates of the center and the radius of the circle
- Radius: the radius of a circle is the length of the centre to any point on its edge
- Equation of the circle : for any point on the circle  $(x,y)$  and the center at point  $(x_c, y_c)$  the equation of the circle is
- $x^2 + y^2 = r^2$

Here  $r$  is radius of the circle if the circle has origin  $(0,0)$  as its center

Mid point circle drawing algorithm

- given the center point and radius of circle
- Mid point circle drawing algorithm attempts to generate the points of one

Octant the point for other octacts

The points for other octacts are generated using the eight symmetry property

Procedure:

Given

Center point of circle  $= (x_0, y_0)$

Radius of circle  $= r$

the point generation using mid point circle drawing algorithm involves the following steps:

- Step-01
- Assign the starting point coordinates  $(x_0, y_0)$  as

$X_0=0$

$Y_0=R$

Step-02

Calculate the value of initial decision parameter  $p_0$  as-

$P_0=1-R$

Step-03

Suppose the current point is  $(x_k, y_k)$  and the next point is  $(x_{k+1}, y_{k+1})$

- Find the next point of the first octant depending on the value of decision parameter  $P_k$

Follow the below two cases

Case-01

$P_k < 0$

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

$Y_{k+1} = y_k$

$P_{k+1} = p_k + 2x_{k+1} + 1$

Case 02

$P_k \geq 0$

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

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

$P_{k+1} = p_k - 2y_{k+1} + 2x_{k+1} + 1$

- Step-05 generates all the points for one octant

To find the points for other seven octants follow the eight symmetry property of circle

Advantage:

it is a powerful and efficient algorithm

It is easy to implement from the programmer's

Disadvantage:

accuracy of the generated points is an issue in this algorithm

The circle generated by this algorithm is not smooth

Bresenham circle drawing algorithm

- Given the center point & radius of circle

Bresenham circle drawing algorithm attempts to generate the points 0 and one octant

Procedure

given

Center point of circle  $= (x_0, y_0)$

Radius of circle  $= R$

the point generation using Bresenham circle drawing algorithm involves the following steps

Step-01

- Assign the starting point coordinates( $x_0, y_0$ ) as
- $X_0=0$
- $Y_0=R$

Step-02

calculate the value of initial decision parameter  $p_0$  as

$$P_0 = 3 - 2 \times R$$

Step-03

Suppose the current point is  $(x_k, y_k)$

And the next point is  $(x_{k+1}, y_{k+1})$

Find the next point of the first octant depending on the value of decision parameter  $P_k$

- Follow the below two case
- Case-01
- $P_k < 0$
- $X_{k+1} = x_k + 1$
- $Y_{k+1} = y_k$
- $P_{k+1} = p_k + 4x_{k+1} + 6$

Case -02

$$P_k \geq 0$$

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

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

$$P_{k+1} = p_k + 4x_{k+1} - 4y_{k+1} + 10$$

- Step-04
- If the given centre point  $(x_0, y_0)$  is not  $(0, 0)$  then do the following and plot the point
- $X_{plot} = x_c + x_0$
- $Y_{plot} = y_c + y_0$

here  $(x_c, y_c)$  denotes the current value of  $x$  and  $y$  coordinates

step-05

Keep repeating step -03 and step -04

Until  $x_{plot} \rightarrow y_{plot}$

- Step-06
- Step-05 generates all the points for one octant
- To find the point for other seven octants follow the eight symmetry property of circle



## 2.3 POLYGON FILLING

- Definition : a closed plane figure made up of several line segments that are joined together the sides do not cross each other exactly two sides meet at every vertex
- Every polygon usually follows:
- Features:
  - ☐ ordered counter clockwise
  - ☐ Two consecutive vertices defines an edge
  - ☐ Usually counter clock
  - ☐ Left side of edge is inside
  - ☐ Right side if edge is inside
  - ☐ Last vertex implicitly connected to first
  - ☐ In 3D vertices are coplanar

Polygons can be regular or irregular

They can be simple or complex convex or concave

The word “ polygon ” means “many\_ angled ” from Greek

To be a polygon a flat closed shape must use only line segment to create its sides so a circle or any shape that has a curve is not a polygon

The three identifying properties of any polygon are that polygon is:

1. **A two dimensional shape**
2. **Closing in a space**
3. **Made with straight sides**

**Type of polygons:**

**Basically there are two types of polygons as**

- ❖ **Convex**
- ❖ **Concave**
- ❖ **complex**

A polygon is a closed 2D object or figure that has three or more sides. Polygons are named based on the number of sides it has.

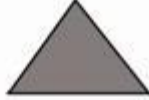









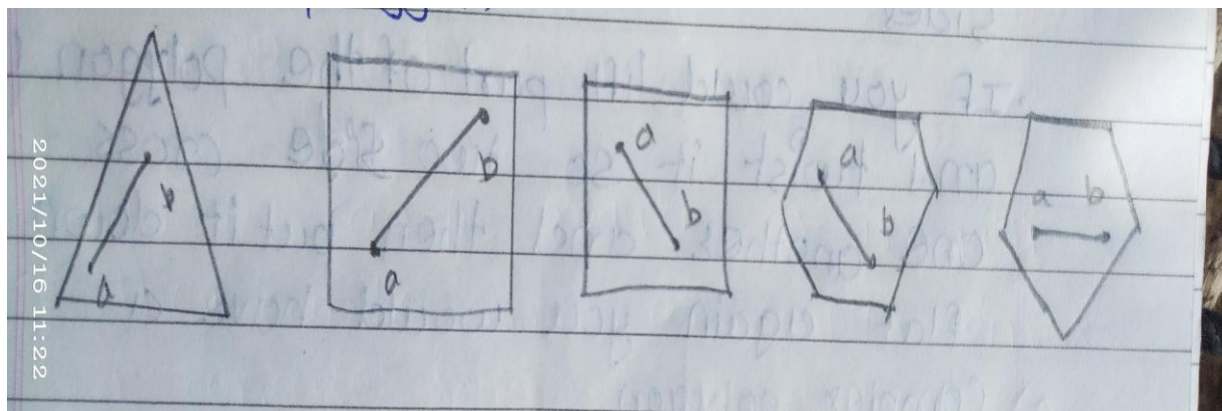
Triangle		
Quadrilateral		
Pentagon		
Hexagon		
Octagon		

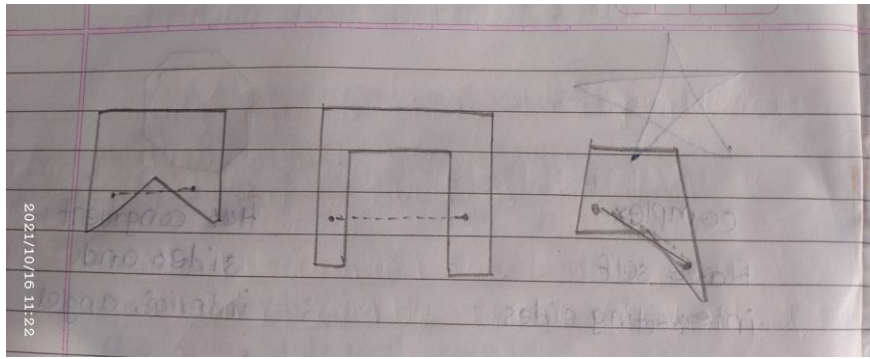
Fig. List of Regular and Irregular Polygons

**Convex polygon**

- Definition : a polygon in which the line segment joining any two points within polygon lies completely inside the polygon is called as convex polygon

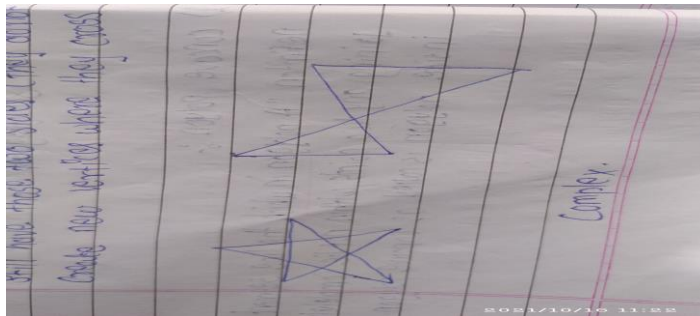
**concave polygon**

- definition: a polygon in which the line segment joining any two points within not lie completely inside polygon is called as concave polygon



### Complex polygon

- Complex polygon also called self –intersecting polygon have sides that cross over each other
- Complex polygons may be hard to imagine unless you think of them with elastic sides
- If you could lift part of the polygon up and twist it so two side cross one another and then put it down flat again you would have a complex polygon
- Because you twisted two side you still have those two sides (they do not create new vertices where they cross)



### Representaion of polygons

- when we are going to use the polygons in our graphics system first we have to decide how to represent the polygons some graphics devices give permission to directly image polygon shapes in that case a polygon acts as a whole unit
- Some graphics devices may provide trapezoid primitives trapezoid primitive means polygons are imaged or drawn in the form of trapezoid trapezoids are formed by using two scan line and two line segments
- Some graphics devices may not provide support to polygons at all in that case we have to break up the polygon into lines and points which can be imaged or drawn to from polygon

### Basic approaches to represent polygons

- Polygon drawing primitives
- Trapezoid primitives

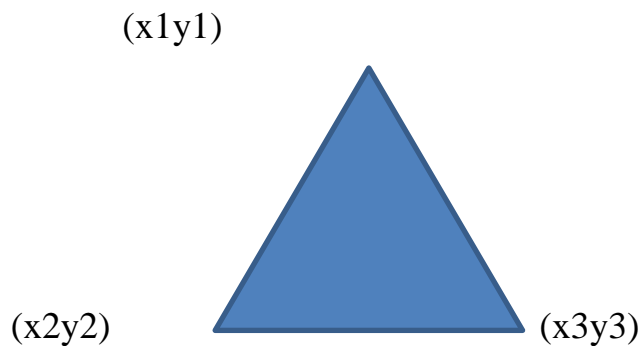
- Line and point

Above approaches depend upon the type of graphic device used

### 1. Polygon drawing primitive

Some graphics devices support polygon drawing approach where directly basic polygon shapes are drawn so here on polygon is saved as one unit

- Standard command for drawing triangle (polygon) could be  $\text{triangle}(x_1, y_1, x_2, y_2, x_3, y_3)$
- Where  $(x_1, y_1)$ ,  $(x_2, y_2)$  and  $(x_3, y_3)$  are co-ordinates of three vertices of triangle and thus supports direct polygon drawing commands

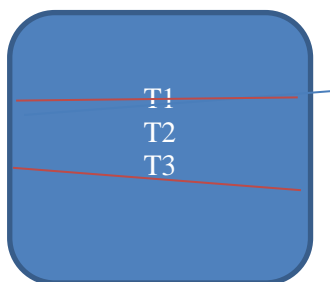


### 2. Trapezoid primitive

Some graphics devices support trapezoid primitive where a polygon is represented

As a set of trapezoids polygon as set three trapezoids  $(t_1, t_2, t_3)$

Trapezoids are formed from two scan lines and two line segments (edges of polygon) as shown in the fig



### Line and point

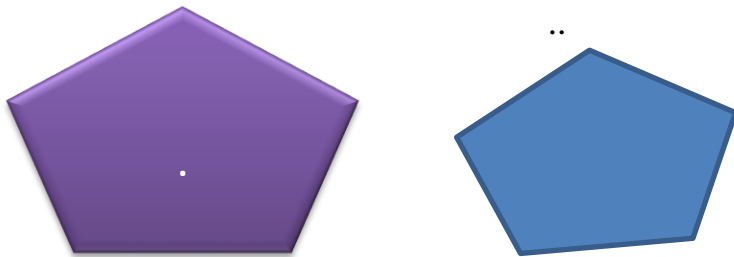
Here polygons are represented using series of only lines and points. This series of lines and points is represented as a single unit and is stored in display file. The display file stores data in following format as array

op x y

- Where op= opcode number which specifies command it is 2 which means line to command which means draw a line from current position to new (x,y) position
- X,y = co-ordinates of a point (vertex of polygon) upto which to draw a line
- Any number 'n' greater than 2 in op field indicates number of line segments in polygon and thus indicate next n number of commands to use to draw a single polygon

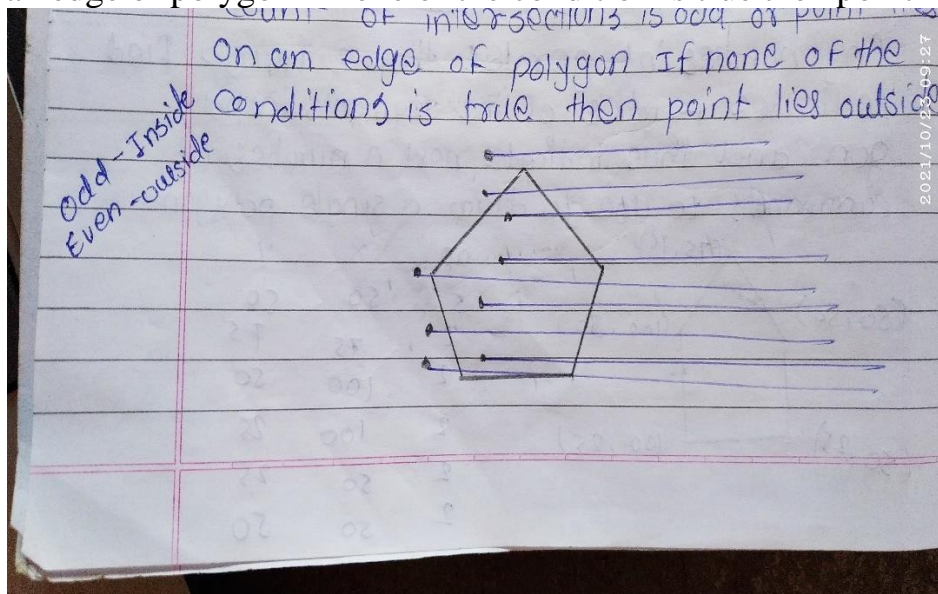
- Insider or outside test of a polygon

Given a polygon and a point 'p' find if 'p' lies inside the polygon or not the point lying on the border are consider inside



Following is a simple idea to check whether a point is inside or outside

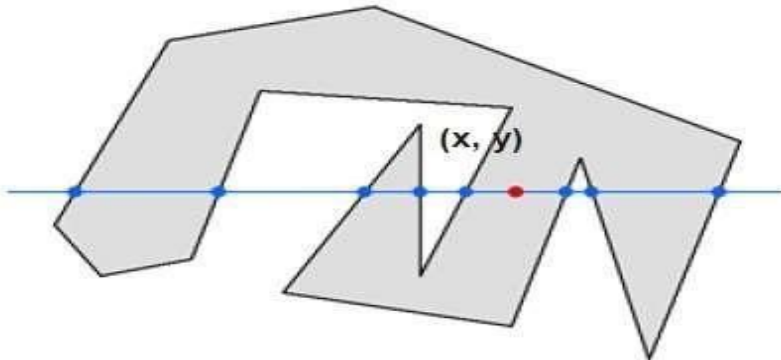
1. draw a horizontal line to the right of each point and extend it to infinity
2. Count the number of time the line intersects with polygon edges
3. A point is inside the polygon if either count of intersection is odd or point lies on an edge of polygon if none of the condition is true then point lies outside



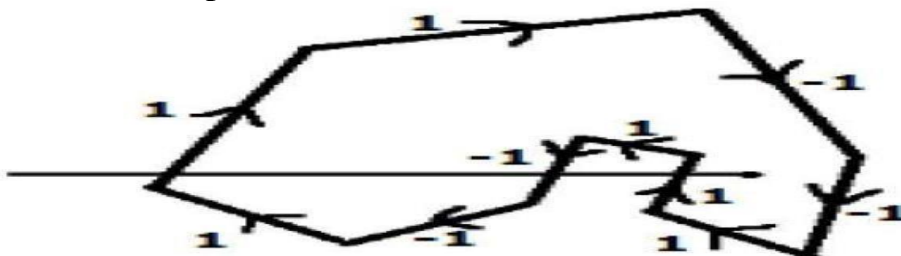
- This method is also known as **counting number method**. While filling an object, we often need to identify whether particular point is inside the object or outside it. There are two methods by which we can identify whether particular point is inside an object or outside.
- Odd-Even Rule
- Nonzero winding number rule

### Odd-Even Rule

- In this technique, we will 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; and if the number of interactions is even, then the point  $(x,y)$  is an exterior point. The following example depicts this concept.



- **Nonzero Winding Number Rule**
- This method is also used with the simple polygons to test the given point is interior or not. Give 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.





### Polygon filling algorithm

- Filling is the process of “coloring in ”a fixed area or region
- Region may be defined at pixel level or geometric level when the regions are defined at pixel level we are having different algorithms likes .

- 1) Boundary fill algorithm
- 2) Flood fill algorithm
- 3) Edge fill algorithm
- 4) Fence fill algorithm

- Geometric level

#### ❖ Scan line algorithm

boundary fill algorithm and flood fill algorithm is define at pixel level

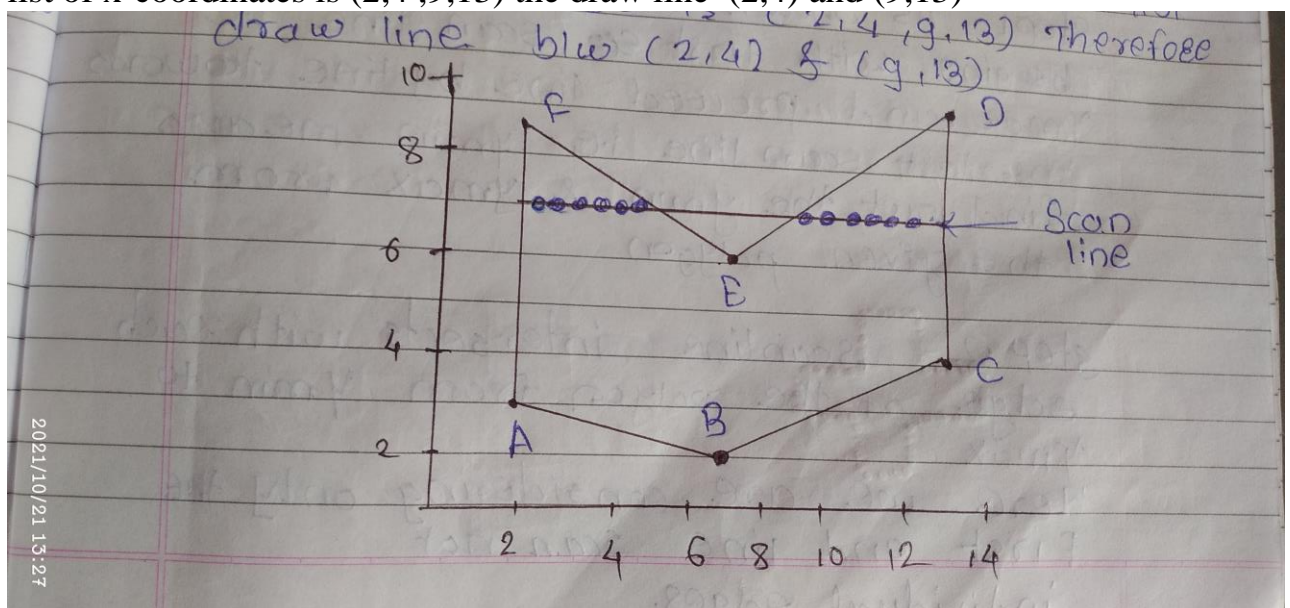
Scan line fill algorithm is defined at geometric level i.e coordinates edges vertices

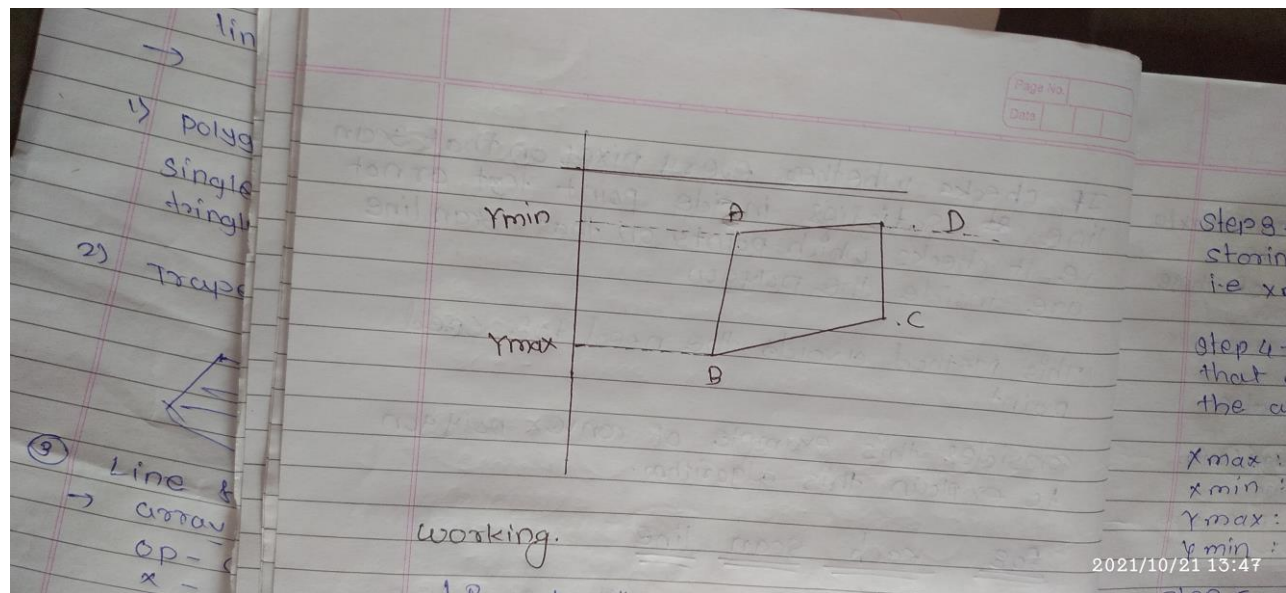
The algorithm start with first scan line and proceeds line by line to the last scan line

If checks whether every pixel on that scan line satisfies inside point test or not i.e it checks which points on that scan line are inside the polygon

This method avoids the need for seed point

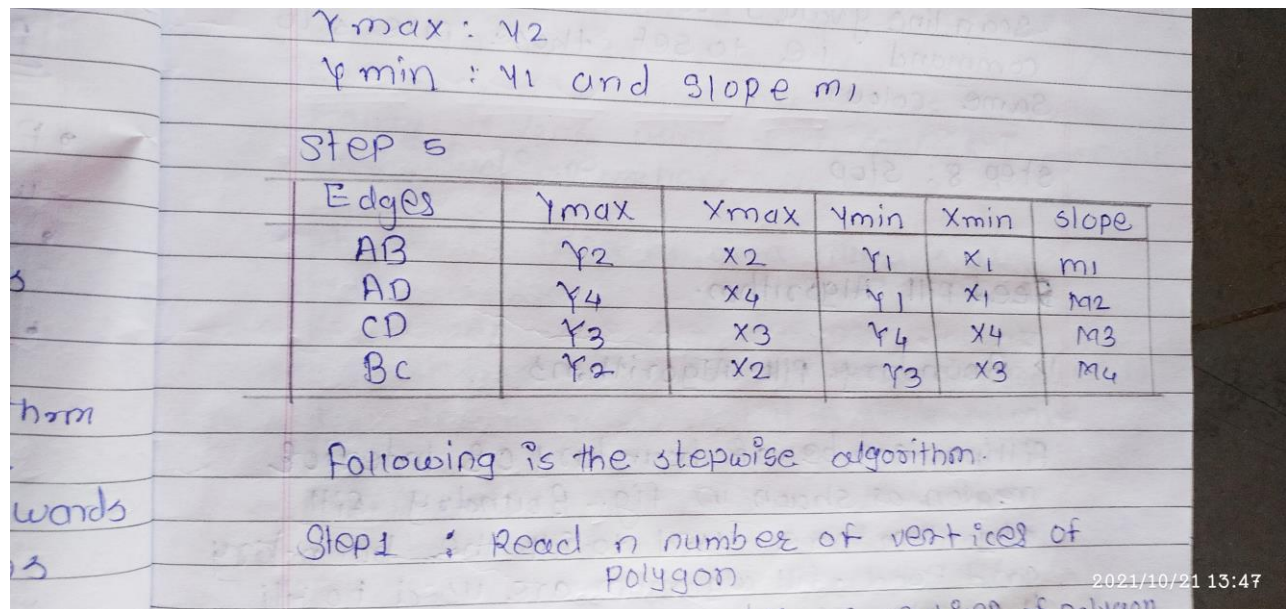
- Consider this example of convex polygon to explain this algorithm
- For each scan line
- Find the intrsctions of the scan line with all edges of the polygon
- Sort the intersection by increasing x coordinates
- Fill in all pixel between pairs of intersections for scan line number 8 the sorted list of x-coordinates is (2,4,9,13) the draw line (2,4) and (9,13)





- Working
- this algorithm works by intersecting scan line with polygon between pairs of intersections the following steps depict how this algorithm works
- Step1: as show in figure 12 algorithm begins with first scan line i.e y max and proceed line by line towards the last scan line i.e y min means find out the y min and y max from the given polygon
- Steps2: scan line intersects with each edge of the polygon from y min to y max Here we are considering only the first and line scan not individual edges
- step3 : for each edge of polygon we are storing 5 attributes along with the slope i.e x max ,xmin ,ymax ,ymin and slope
- Step4 : fill all those pair of coordinates that are inside polygons and ignore the are inside polygons and ignore the alternate pair for AB we are storing
- X max:x2
- Y min :x1
- Y max : y2
- Y min : y1 and slope m1





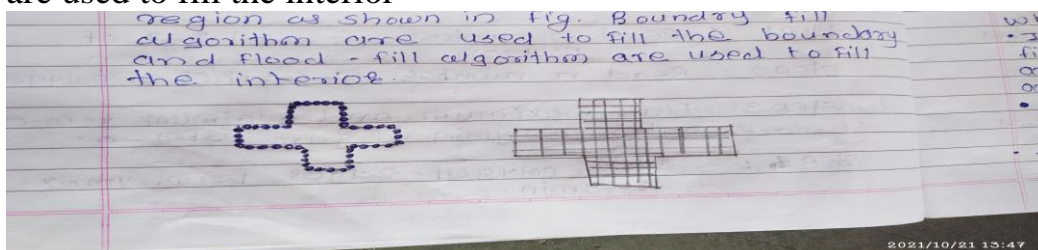
- Following is the stepwise algorithm
- Step1: read n number of vertices of polygon
- Step2: read n number of vertices of polygon
- Step3 : find maximum and minimum y- coordinates as y min and y max
- Step4: sort polygon edges from y max to y min
- step5: for each y value (from max to min ) determine which side can be intersected and find x values of these intersection points

- step-6:
- sort these x values pairs x values
- Step 7: pass these x value pairs (with scan line y value) to line drawing command i.e to set those pixels to same colour
- Step 8: stop

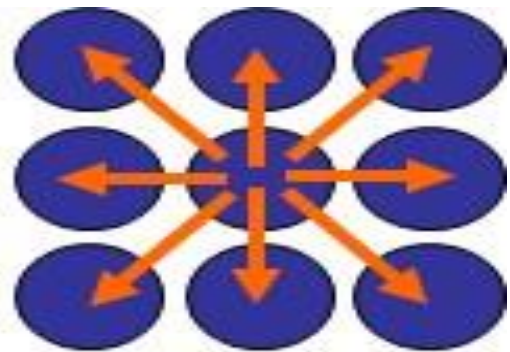
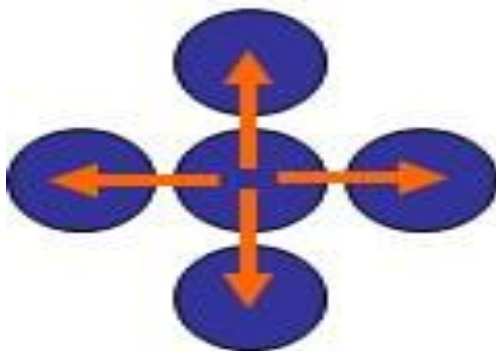
- Seed fill algorithm

#### Boundary fill algorithms

- filling can be of boundary or interior region as shown in fig
- Boundary fill algorithm are used to fill the boundary and flood –fill algorithm are used to fill the interior



- This algorithm use the recursive method first of all a starting pixel called as the seed is considered the algorithm checks boundary pixel or adjacent pixels are colored or not if the adjacent pixel is already filled or colored then leave it otherwise fill it
- boundary fill is the algorithm used frequently in computer graphics to fill a desired color inside a closed polygon having the same boundary color for all of its side the filling is done using four connected or eight connected approaches
- ❖ there are two method for proceeding to neighboring pixels from current test point
- ❖ 4 connected method
- ❖ 8 connected method



- Four connected approaches is more suitable than the eight connected approaches
- 4 connected method

in this approach left right above below pixel are tested

8 connected method

in this approach left right above below and four diagonals are selected

### Algorithm

- Step1 : initialize the value of seed point (seedx, seedy) fill color (f color) and default color
- Step2: define the boundary values of the polygon
- Step3: check if the current seed point is of default color, then repeat the steps 4 and 5 till the boundary pixels reached
- Step4: change the default color with the fill color at the seed point
- Step5 : change the default color with the fill color at the seed point
- Step 6: exit

## 2. flood fill algorithm

flood fill algorithm is also known as a seed fill algorithm

It determines the area which is connected to a given node in a multi- dimensional array

This algorithm works by filling or recolouring selected area containing different colours at the inside portion and therefore the boundary of the image it is often illustrated by a picture having a neighbourhood bordered by various distinct colour regions.

To paint such regions we will replace a specific interior colour instead of discovering a boundary colour value

- This is the rationale the approach is understood because of the flood fill algorithm in this method a point or seed which is inside region is selected this point is called a seed point
- Then four connected approaches or eight connected approaches is used to fill with specified color now there are two methods which will be used for creating endless boundary by connecting pixels
- 4 connected and 8 connected approach
- In the 4 connected method the pixel can have at maximum four neighbours that are positioned at the proper left above and below the present pixel.
- 8 – connected method it can have eight and the neighbouring positions are checked against the four diagonal pixels so any of the 2 methods are wont to repaint the inside points
- The flood fill algorithm has many characters similar to boundary fill but this method is more suitable for filling multiple colors boundary
- When boundary is of many colors and interior is to be filled with one color we use this algorithm
- In fill algorithm we start from a specified interior point (x,y) and reassign all pixel values are currently set to a given interior color with the desired color using either a 4 connected or 8 connected approaches we then step through pixel positions until all interior points have been repainted.

### algorithm

- Step1: start
- Step 2: read any seed pixels position(x,y)
- Step3: check to see if this (x,y) has old interior colour if old colour then set it to new fill colour
- Step4: repeat step3 for 4 neighbouring pixels of (x,y)
- Step5: in step3 and 4 if pixel (x,y) do not have old interior colour then jump to step 6.
- Step6: stop:

Disadvantage:

Very slow algorithm

May be fail for large polygons

Initial pixel required more knowledge about surrounding pixels

### 2.3.1 Scan converting polygon

Two important properties are required to scan convert polygons:

1. Spatial Coherence: This property states that the adjacent pixels of the polygons usually have the same characteristics.
2. Scan Line Coherence: This property states that scan line will have the same characteristics.

Many algorithms are used to scan convert a polygon. Parity scan conversion algorithm is one of the most popular algorithms. This algorithm is based on parity bit. Figure illustrates the parity scan conversion algorithm. Here, P1, P2, P3, P4, P5, P6, and P7 are the parity pairs.

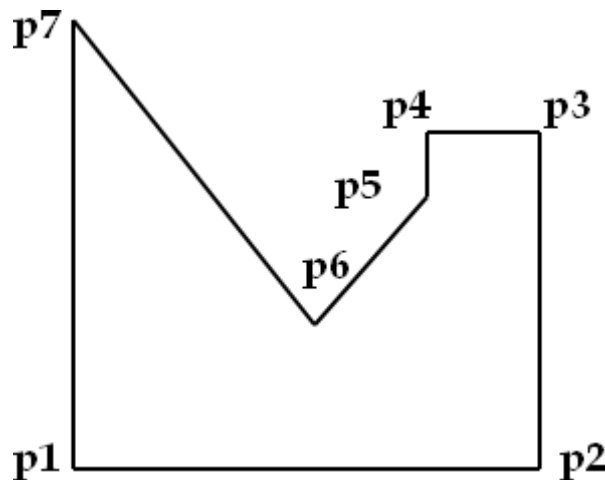


Fig. Parity Scan Conversion Algorithm

When the scan lines intersect, polygon pairs set the parity flag to 1. The parity bit is set to check whether the pixel is within the polygon. If it is within the polygon the parity flag is set to 1 else to 0. Thus, if the parity flag is 1, then the scan line is inside the polygon and if the parity flag is zero then the scan line is outside the polygon.

When the parity flag is 1, the pixels inside the polygon gets the respective polygon color, if the parity flag is set to zero the polygon pixels takes the background color as shown in figure.

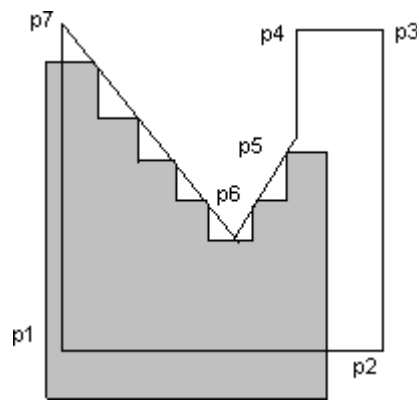


Fig. Parity Scan Conversion Algorithm

### 2.3.2 Edge fill

In scanline filling algorithm, we take the intersection of each scanline with the edges of the polygon.

#### Steps:

1. Read  $n$
2. Read  $(x_i, y_i)$  for all  $i=1, 2, 3, \dots, n$
3. Read edges and store it in the array  $E$  which will be sorted accordingly to  $y$  axes.
4.  $X_{min}=a$ ;  $x_{max}=b$ ;  $y_{min}=c$ ;  $y_{max}=d$
5. Take intersection
6. Take the scanline  $y=c$  and scan from  $x=a$  to  $x=b$
7. Find the intersecting edges of  $E$  with  $y=c$  by comparing the  $y$  coordinate of the end points with  $y=c$
8. Activate those edges
9. Scan through the line  $y=c$  and compute the next  $x$  position by applying the formulation  

$$X_{k+1} = x_k + 1/m$$
 Check whether the point  $(X_{k+1}, Y_k)$  is inside or outside the polygon, by inside outside procedure. If the point  $(X_{k+1}, Y_k)$  is inside, paint it.
10. Repeat the procedure from  $Y_c$  to  $Y_d$  i.e.  $y=c$  to  $y=d$ .

### 2.3.3 Edge flag

### 2.3.4 Seed fill

To fill a polygon we start with a seed and point the neighboring points till the boundary of the polygon is reached. If boundary pixels are not reaching pixels are illuminated one by one and the process is continuing until the boundary points are reached. Here, at every step we need check the boundary. Hence, this algorithm is called “boundary fill algorithm”.

To find the neighboring pixels we can use either a 4 connected or 8 connected region filling algorithm.

The algorithm is recursive one and can be given as follows:

Here, we specify the parameters, fore-color by F and back color by B 4 Neighbors are:

$N4 = \{ (X+1, Y), (X-1, Y), (X, Y+1), (X, Y-1) \}$

Seed pixel (x,y)

Seed\_Fill ( x, y, F, B)// 4 connected approach

```
{/
/Seed= getpixel (x,y);
If (getpixel(x,y)!= B && getpixel(x,y)!= F)
{
putpixel (x,y,F);
Seed_Fill (x-1, y, F, B);
Seed_Fill (x+1, y, F, B);
Seed_Fill (x, y-1, F, B);
Seed_Fill (x, y+1, F, B);
}}
```

getpixel (): is a procedure which gives the color of the specified pixel.

putpixel(): is a procedure which draws the pixel with the specified color.

B : is the boundary color.

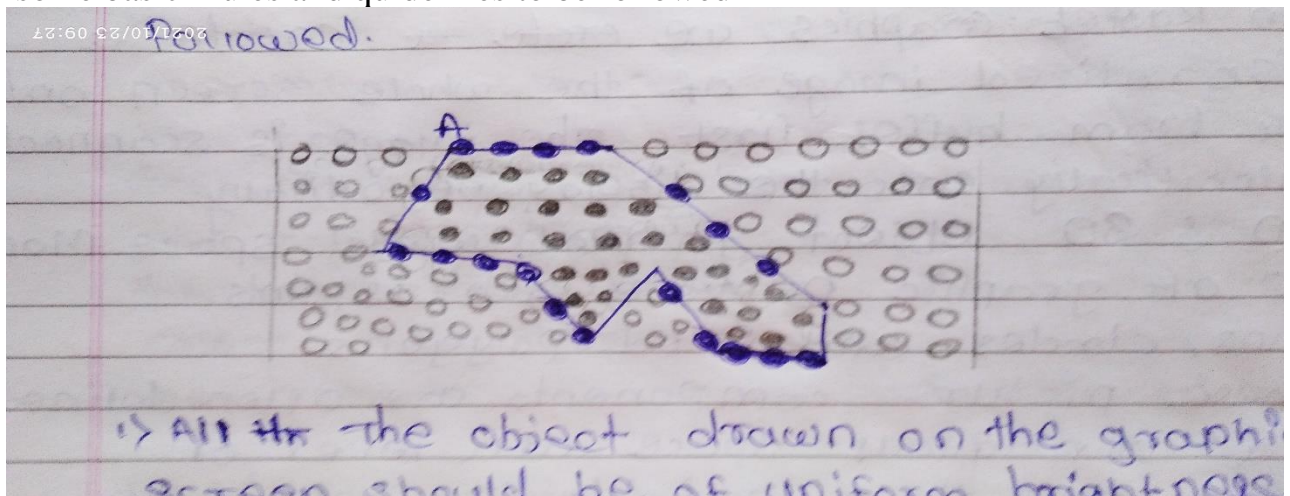
Drawbacks: in Seed Fill algorithm we have 2 drawbacks.

1. If some inside pixels are already painted with color F then the recursive branch terminates leaving further internal pixels unfilled.
2. The procedure required stacking of neighboring pixels. If the polygon is too large the stack space may become insufficient for all the neighboring pixels.

### Scan conversion

- in raster graphic we create a discretized image of the whole screen on to the frame buffer first the image is scanned automatically onto the display periodically

- 2D and 3D object in real world space made up of graphic primitive such as points, lines, circles, and filled polygons
- these picture components are often defined as abstraction rather than individual pixel in the discrete image space
- the process of representing continuous graphics object as a collection of discrete pixels is known as scan conversion
- this can be demonstrated by a line that is defined by its two endpoints and the equation of that line, also by a circle that is represented by its midpoint and the radius
- It is the responsibility of the graphics system or the application program to convert each primitive from its geometric definition into a set of pixels that make up the primitive in the image space
- the job of conversion of every primitive object represented on the graphics screen into a set
- of pixels that its basic form is termed as scan conversion or rasterization. Thus, the process of scan conversion requires
- some basic rules and guidelines to be followed



- 1. All the object drawn on the graphics screen should be of uniform brightness. The brightness content must be the same for every point so that the conversion is made easy.
- 2. The object drawn on the graphics screen should be free from any orientation and independent of length and size.
- pixels

A pixel is the smallest unit of a graphics image or referred to as a physical point in a raster.

- two methods

1. analog method: This conversion is done for large numbers of display cells and is appropriate for an analog video.

It may be performed also using a particular specialized scan converter vacuum

tube in this case all polar coordinates (angle and distance) data from a source such as a radar receiver so that it can be displayed on any raster scan display

2 Digital methods :- in this particular method any picture is stored in a line or frame buffer with  $n_1$  speed (data rate)

And is read with  $n_2$  speed several picture processing techniques are applicable when the picture is stored in buffer memory including kinds of interpolation from simple to smart high

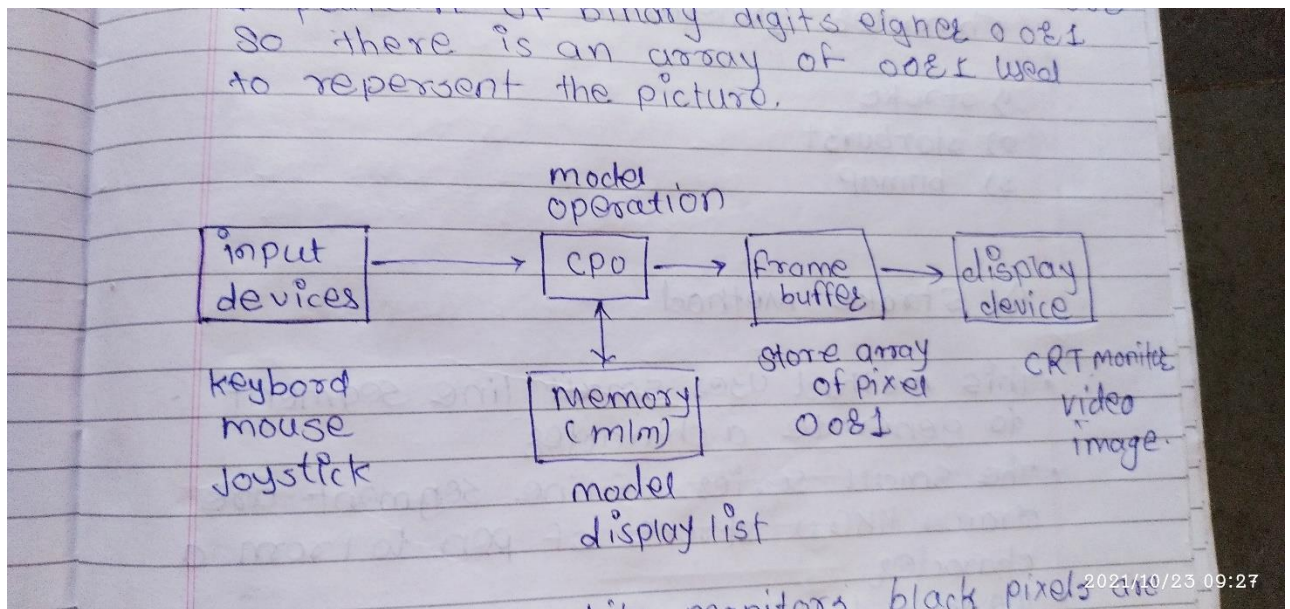
Order come motion detection and to improve the picture quality and prevent the conversion complications

- Need of scan conversion algorithm this helps in accelerating the process of scan conversion and can generate graphic objects at a faster rate comparatively using these algorithms memory is also used efficiently and the production quality and prevent the conversion complications
- need of scan conversion algorithms this helps in accelerating the process of scan conversion and scan generate graphic object at a faster rate comparatively using these algorithm memory is also used
- efficiently and the production quality of the object on the screen is also improved the scan converting rate is a technique that is used in video processing in this technique
- We can change the horizontal and vertical frequencies for different purposes
- we can apply the scan conversion algorithm for various object that are
- Point
- Line
- Polygon example rectangle triangle and many more
- Circle
- Arc
- Sector any filled region

### **frame buffer**

- frame buffer an area of memory used to store information related to the pixels of a display
- A frame buffer for a color display has set of planes each defining one bit of the color information of each point
- each screen pixel corresponds to a particular entry in a 2D array residing in memory each pixel has some intensity value which is represented in memory of computer called a frame buffer frame buffer is also called a refresh buffer
- The memory is a storage area for storing pixels values using which picture are displayed it is also called as digital memory
- Inside the buffer image is stored as a pattern of binary digits 0 or 1 so there is an array of 0 or 1 used to represent the picture





- In black and white monitors black pixels are represented using 1s and white pixels are represented using 0s
- in case of system having one bit per pixel frame buffer is called a bitmap
- the number of rows in the frame buffer equals to the number of raster lines on the display screen the number of columns in this array equals to the number of pixels on each raster line
- a frame buffer or sometimes frame store is a portion of ram containing a bitmap that drives a video display

### character generation methods

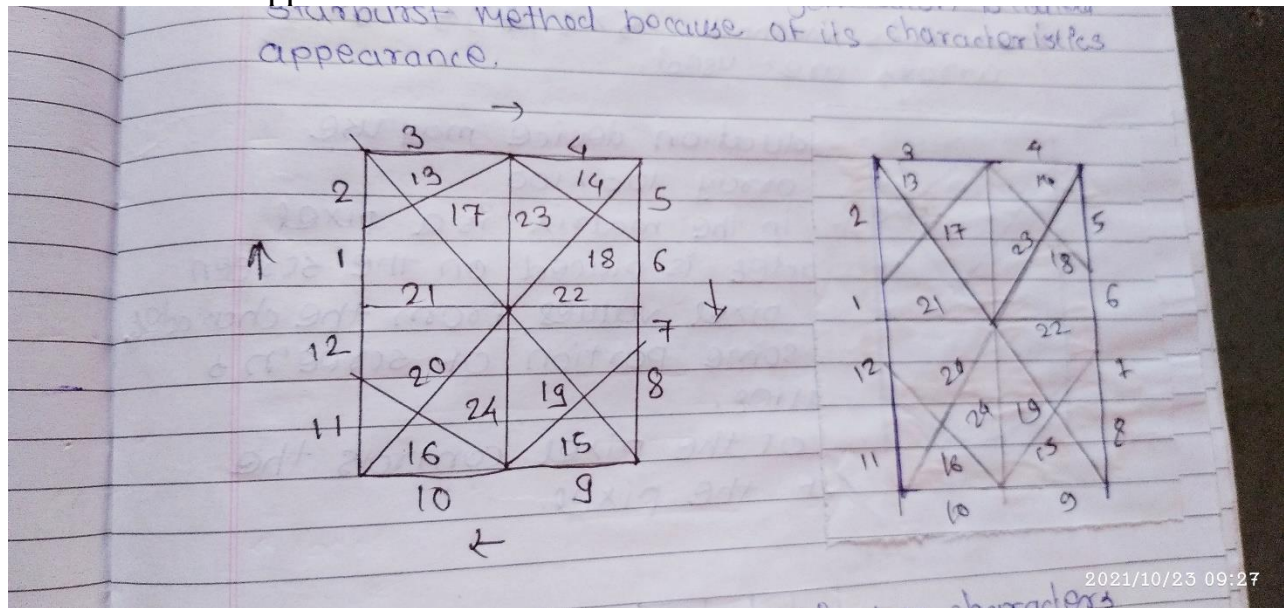
- Stroke
- Starburst
- Bitmap

#### 1 stroke method

- this method uses small line segment to generate a character
- The small series of line segment 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
- Starburst 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 starnurst method because of its character appearance



- the patterns for particular characters are stored in the form of 24 bits code
- Each bit representing one line segment
- the bit is set to one to highlight the line segment otherwise it is set to zero

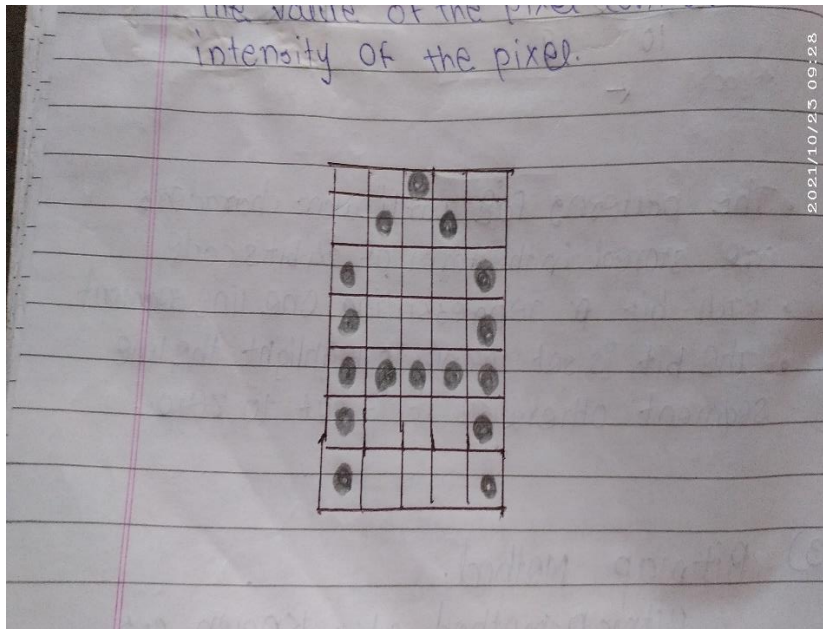
### 3 Bit map method

bitmap method also known as dot matrix

because in this method characters are represented by an array of dots in the matrix from

its two dimensional array having columns and rows  
in this method 5x7, 7x9 and 9x13 arrays are used

- Higher resolution device may use character array 100x100
- each dot in the matrix is a pixel
- the character is placed on the screen by copying pixel values from the character array into some portion of screen's frame buffer
- the value of the pixel controls the intensity of the pixel



CLASS: S. Y.

SUBJECT: COMPUTER GRAPHICS