Unit-1

PAGE NO.:

Computer Graphics -: Computer graphics is the study of techniques to improve comm^n b/w human and m/c.

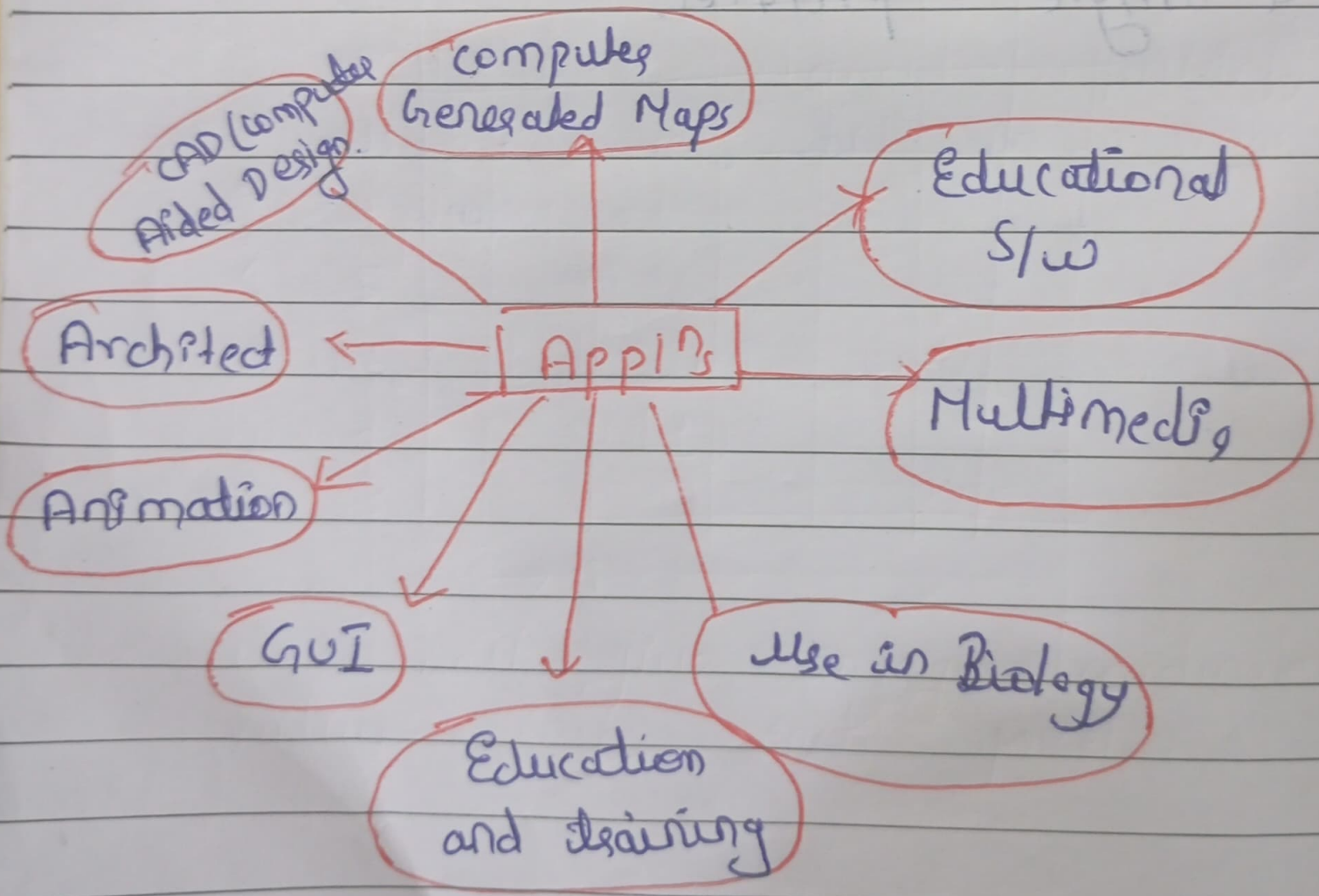
The word "Computer Graphics" means pictures, graph's or scene is drawn with the help of computer.

OR

It is an art of drawing pictures, lines, chart etc. on computer screen by using programming language.

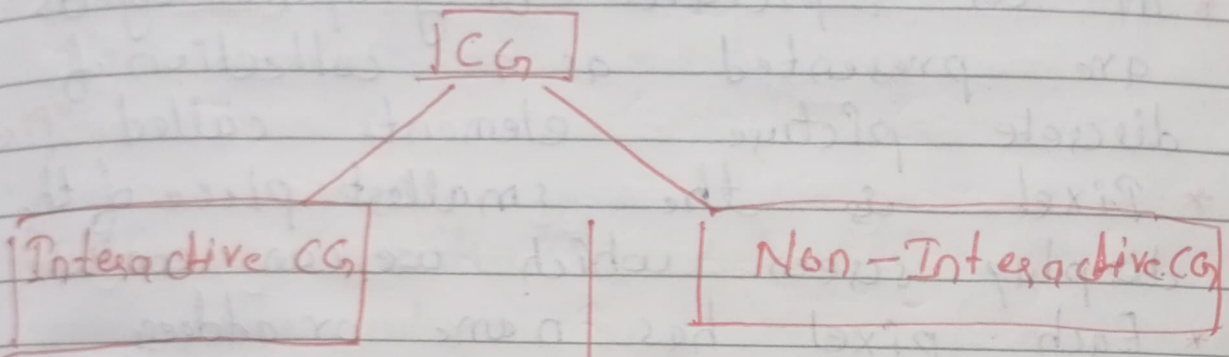
Thousand words can be replaced by a single pictures.

Appln of Computer Graphics :-



PAGE NO.:

Types of Computer Graphics:



It is two-way comm^n. b/w the computer and the user.

User have some control over the image.

A user can make any changes by sending his command with an input device.

Ex: Vedio Game.

It is one-way comm^n b/w the computer and the user

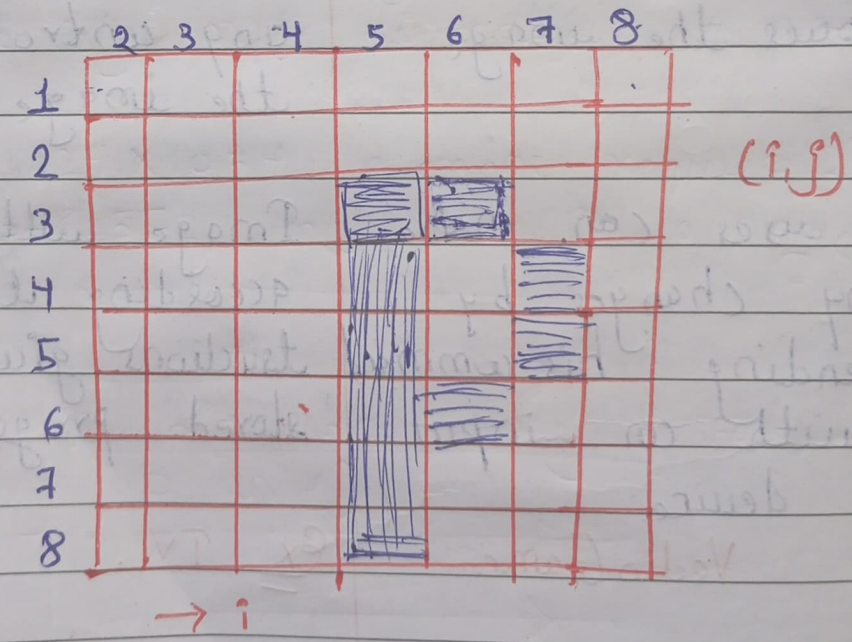
User does not have any controlled over the image.

Image will work according to the instructions given in the stored program.

Ex: TV.

Pixels ⇒ * The pixel is the smallest addressable screen element.

* In CG, pictures or graphics objects are presented as a collection of discrete picture elements called "Pixel"
* Pixel is the smallest piece of the display screen which we can control.
* Each pixel has name or address.
* Pixels are normally arranged in a rectangular 2D grid array. and are often represented using dots or square

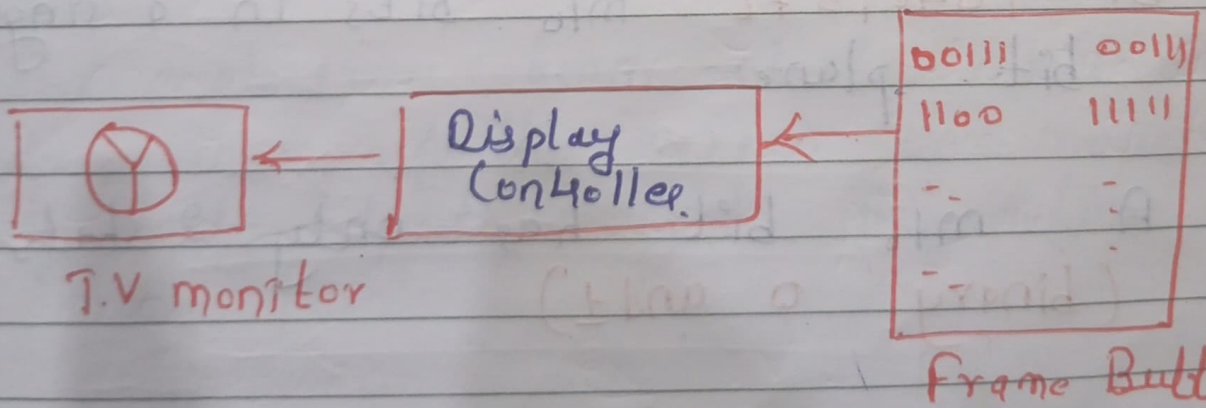


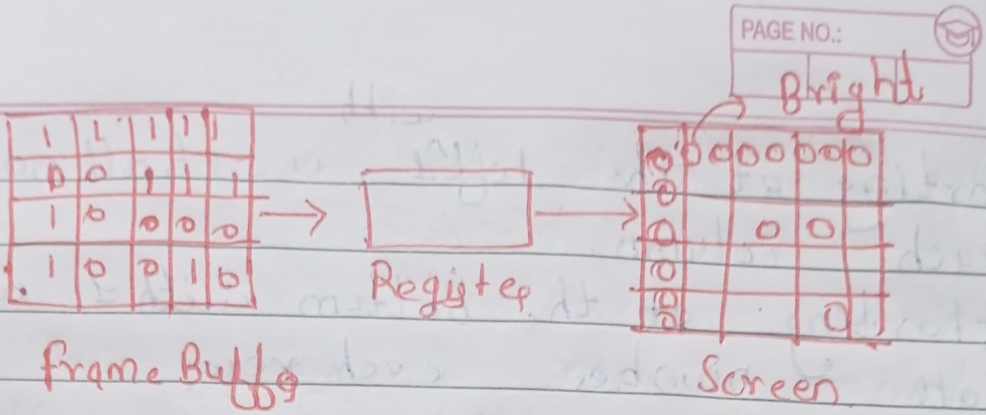
* We shall give integer cordinate values to each pixel.

starting at the left with 1, we shall no each column.
starting at the bottom with 1, we shall numbes each row.

The coordinate (i, j) will then give the column and row of a pixel.

Frame Buffer ⇒ A frame buffer is large, contiguous piece of computer m/o used to hold or map the image displayed on compiter screen.





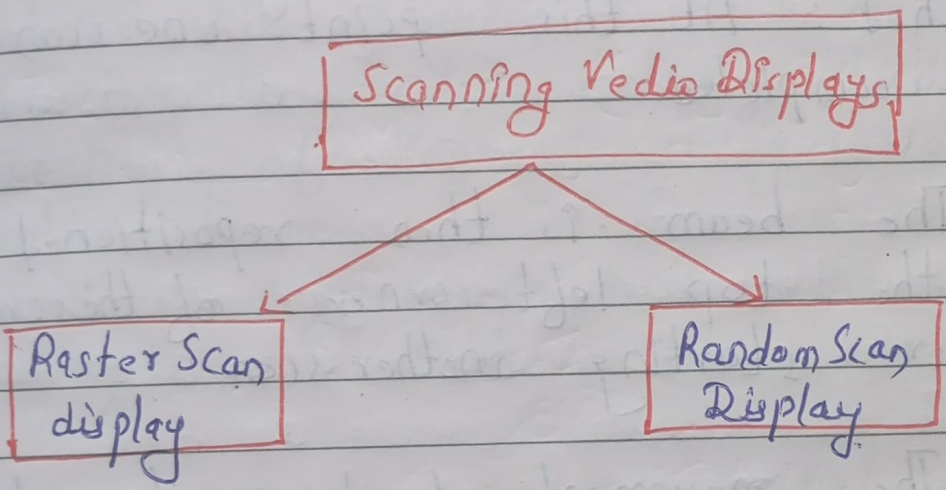
* In the frame buffer current image details is stored in the form of bits.
* It is a part of main m/o for modern system
* At a minimum, 1 memory bit for each pixel in a raster. This amount of m/o is called bit plane.
* A 1024 x 1024 element requires 2^{20} or 1,048,576 m/o bits in a single bit plane.

A m/o bit has only 2 states (binary 0 and 1).

PAGE NO.:

Raster and Random Scan display ⇒

There are two different ways of scanning video displays.



Raster Scan Displays ⇒ * In raster scan the electron beams follows a fixed path.

* The electron beam starts at top left corner of the screen and moves horizontally to the right. This defines a scan line.

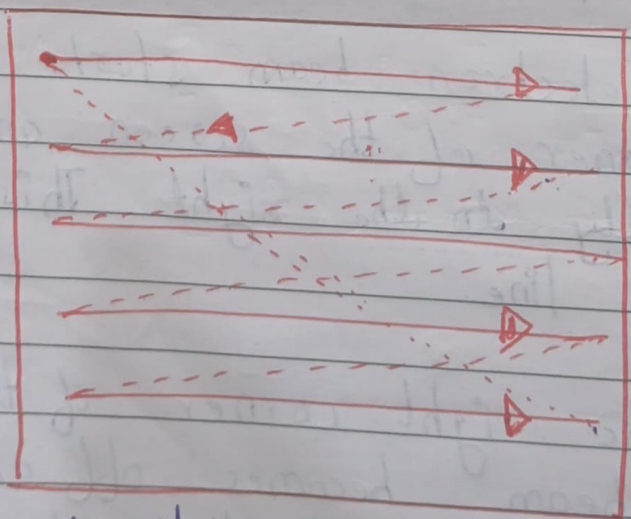
* At the right corner of the screen, the beam becomes off and moved back to the left edge of the screen at the starting point on next line

This is shown as dotted line, which is called Horizontal Retrace.

* This way of scanning is continued till the bottom right corner is reached. At this point one scan is completed.

* The beam is then repositioned at the top left corner of the screen for starting another scan.

* The movement of beam from bottom right corner to top left corner is called as Vertical Retrace.



Raster graphics can be used in animation

PAGE NO.:

Random Scan Display / Vector Graphics =>

Display device with random scan operate by directing the electron beam to only those part of the screen where the picture is to be drawn. This is also called vector graphics. Creation of diagram using vector graphics become easier, so can be used in engineering and scientific drawing.

In this pen plotters and DVST (Direct view storage Tube) devices are used.

Advantages:
1. Easy animation, just draw at different
2. Require less memory [positions]

Disadvantages
1. Can't draw a complex image.
2. Limited colour capability

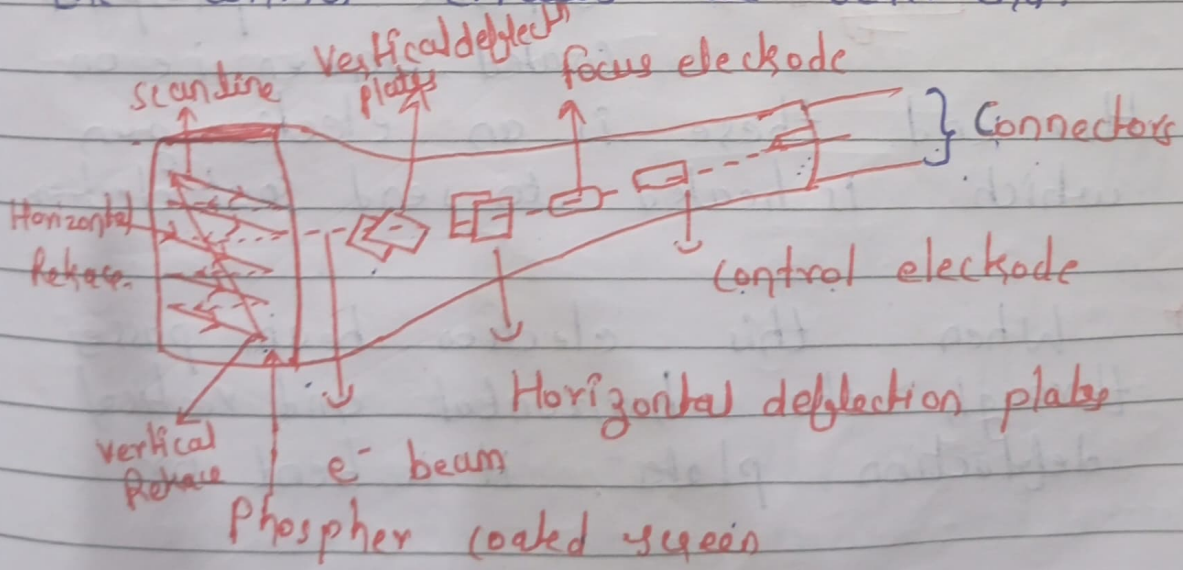
Display devices ⇒ There are several h/w devices which may be used to display images.

1. Display or Vedio monitor ⇒

There are two types of video monitor :-

1. Monochrome display monitor / CRT ⇒

It mainly consists a CRT (cathode Ray Tube) along with related circuits. The CRT is a vaccum glass tube with the display screen at one end and connectors to the control circuits at other end.



PAGE NO.:

* Phosphor is coated on the inside of the display screen which emits light for a period of time when hit by a beam of electron.

* The light given off by the phosphor during exposure to the e⁻ beam is known as fluorescence.

* The continuing glow given off after the beam is removed is known as phosphorescence and duration of phosphorescence is known as the phosphors persistence (kitne time tak e⁻ display hote hai screen par)

* electron gun ⇒ opposite to phosphor coated screen i.e on other end there is an electron gun which is heated to send out electron

* When this electron beam passes through the horizontal and vertical deflection plates, it is bent or

deflected by the electric field b/w the plates.

* Horizontal plates ⇒ It control the beam to scan from left to right & release from right to left.

* Vertical plates ⇒ It control the beam to go from the first scan line at the top to the last scan line at the bottom and release from the bottom back to the top.

Color Display Monitor ⇒ There are two basic technologies for color display.

(i) Beam penetration method ⇒ Inside of CRT coated with two phosphor layers.

* Color depends on how far the e⁻ beam penetrates into the phosphor layer.
→ fast e⁻ penetrates more and excites inner green layer.
→ slow e⁻ excites outer red layer.

→ Intermediate beam speed produce combination of red and green lights which emit additional two colors orange & yellow.

Beam acceleration voltage control the speed of the e^- and hence color of pixel

Advantage :-

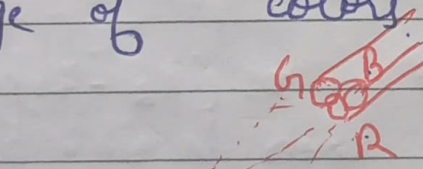
Disadvantage :- 1) Only 4 colors are possible

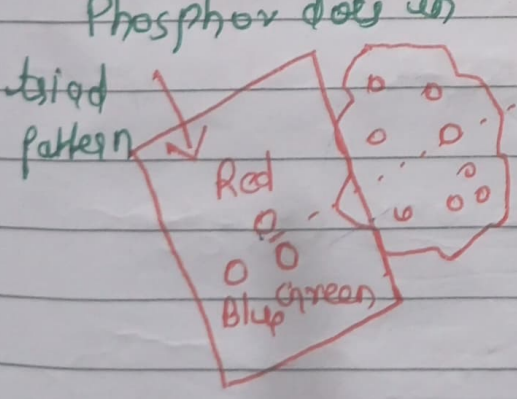
2) Quality of picture is not a good as with other method.

Shadow mask CRT ⇒ It commonly used

In this method there are 3 electron gun instead of one with one e^- gun for each primary color. The phosphor coating on the inside of the display screen consists

of dot pattern of 3 different types of phosphors. These phosphors are capable of emitting Red, Green and Blue light. A thin metal screen called a shadow mask is placed b/w the coating and e⁻ guns. The tiny holes on the shadow masks constrain each e⁻ beam do hit its corresponding phoshpor dots. When viewed from a certain distance, light emitted by the 3 types of phosphors blends together to give us a broad range of colors.





PAGE NO.:

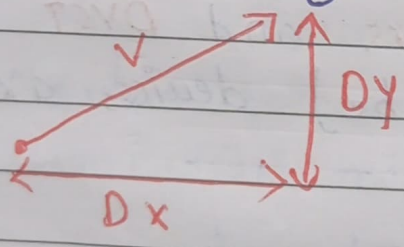
Vector and character Generation =>

Vector ⇒ Vector has a single direction and length. A vector may be denoted as [Dx, Dy], where

Dx ⇒ indicates how far to move in x direction

Dy ⇒ indicates how far to move in y-direction.

Vector tell us how far and in what direction to move, but they don't tell us from where to start.



$$V_1 + V_2 = [Dx_1 + Dy_1] + [Dx_2 + Dy_2]$$

$$= [Dx_1 + Dx_2 + Dy_1 + Dy_2]$$

Character Generation method ⇒ Usually characters are generated by h/w But we can prepare characters by s/w also.

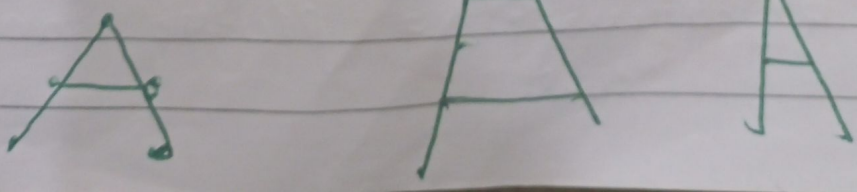
There are 3 primary methods for character generation

1. Stroke / vector character Generation Method ⇒ This method creates characters by using a set of line segments.

To produce a character we will give a sequence of commands that define the start point and end points of the straight lines

* Scale ⇒ We can change the scale of the characters. We can make a character twice as large as its original size.

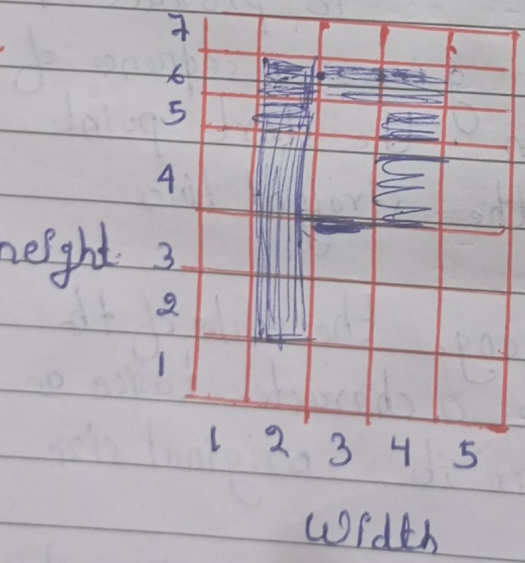
* Scan ⇒ We can make a character slanted also. By using this method we can change the style of characters also.



2. Dot-matrix / Bit map method =>

* In this method, characters are represented by an array of dots. The size of array may vary. An array of 5 dots wide and 7 dots high is generally used. but 7x9 are also used.

* This array is like a small buffer just big enough to hold a single character



5x7 = array.

3. starburst method ⇒ In this method a fix pattern of line segment are used to generate characters.

* There are 24 line segments out of 24 line segments required to display for particular characters are highlighted.

* The pattern for particular character are stored in the form of 24 bit code

* Each bit representing 1 line segment

* The bit is set to 1 to Highlight the line segment. Otherwise it is set to 0.

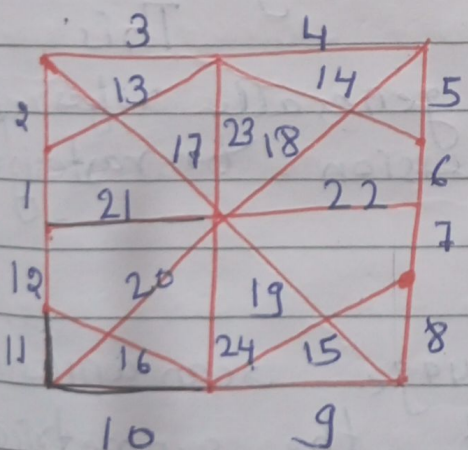


fig: starBurst Pattern of 24 line segments

2.1 = 1 11 = 1 9 = 1 3 = 1 1 = 1

12 = 1 10 = 1 84 = 1 2 = 1

PAGE NO.:

Line Drawing Algorithm ⇒

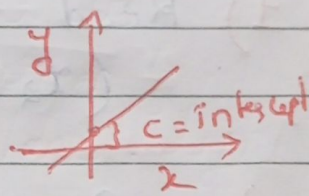
1 DDA (Digital Differential Analyzer) ⇒

It is based on incremental method

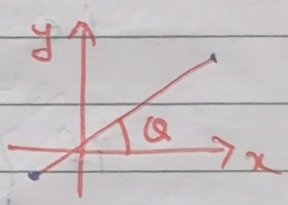
Line eq^n ⇒

$$y = mx + c$$

c = intercept
y, x = co-ordinate
m = slope of line



[slope ⇒ जब मैं x की value change करती जा रही हूँ तो किस rate से y की value change हो रही है!]



[The angle made b/w x and y axis (tangent) are also called slope]

(x_1, y_1) (x_2, y_2)

$$y = mx + c$$
$$y_2 = mx_2 + c$$
$$y_1 = mx_1 + c$$

$$y_2 - y_1 = m(x_2 - x_1)$$

y_1 y_2 x_1 x_2 → start & end point of line

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

PAGE NO.:

Slope $\boxed{m = \frac{\Delta y}{\Delta x}}$ Δ = change in y value / change in x value

DDA Algorithm =>

1) Read the end point (x_1, y_1) and (x_2, y_2)

2) Approximate the length of the line i.e

if (abs $(x_2 - x_1) \geq$ abs $(y_2 - y_1)$)
then

length = abs $(x_2 - x_1)$

else

length = abs $(y_2 - y_1)$

3) $\Delta x = \frac{(x_2 - x_1)}{length}$

$\Delta y = \frac{(y_2 - y_1)}{length}$

4) Round the value. Make use of SIGN funct^n so as to enable line drawing in any quadrant.

$$x = x_1 + 0.5 * sign(\Delta x)$$
$$y = y_1 + 0.5 * sign(\Delta y)$$

5) Now plot the point {start point}

```
i = 1

while (i <= length)
{
    plot (integer(x), integer(y))

    x = x + Δx
    y = y + Δy

    i = i + 1
}
```

6) stop

PAGE NO.:

Numericals :-

Draw a line from (0,0) to (6,6) using DDA algorithm?

$x_1 = 0 \quad , \quad y_1 = 0$

$x_2 = 6 \quad , \quad y_2 = 6$

if $(abs(6-0) \geq abs(6-0))$

then

length = $abs(6-0)$

$= abs(6)$

$= 6$

else

length = $abs(6-0)$

$= abs(6)$

$= 6$

$\Delta x = \frac{6}{6} = 1$

$\Delta y = \frac{6}{6} = 1$

$x = x_1 + 0.5 * Sign(\Delta x)$

$= 0 + 0.5 * Sign(1)$

$\boxed{x = 0.5}$

$y = y_1 + 0.5 * sign(\Delta y)$

$= 0 + 0.5 * sign(1)$

$\boxed{y = 0.5}$

Now plot the point (start point) (0,0)

```
i=1
while (i<=6)
  {
  plot (int(x), int(y))
  plot (0.5, 0.5) i.e = (0,0)
```

$x = x + \Delta x$

$= 0.5 + 1$

$= 1.5$

$y = y + \Delta y$

$= 0.5 + 1$

$y = 1.5$

```
i = i+1
i = 1+1
```

$\boxed{i = 2}$

```
while (2<=6)
  {
  plot (int(x), int(y))
  plot (1.5, 1.5) i.e = (1,1)
  x = x+Δx
```

$= 1.5 + 1 = 2.5$

PAGE NO.:

$x = 2.5$

$y = y + \Delta y$

$= 1.5 + 1$

$y = 2.5$

$i = 2 + 1$

$i = 3$

while $(3 \leq 6)$

{

plot $(2.5, 2.5) = (3, 3)$

$x = x + \Delta x$

$x = 2.5 + 1$

$x = 3.5$

$y = y + \Delta y$

$y = 2.5 + 1$

$y = 3.5$

$i = 4$

while $(4 \leq 6)$

{

PAGE NO.:

plot (3.5, 3.5) = (4, 4)

x = 4.5

y = 4.5

r = 5

while (5 <= 6)

{

plot (4.5, 4.5) = (5, 5)

x = 5.5 , y = 5.5

r = 6

while (6 <= 6)

{

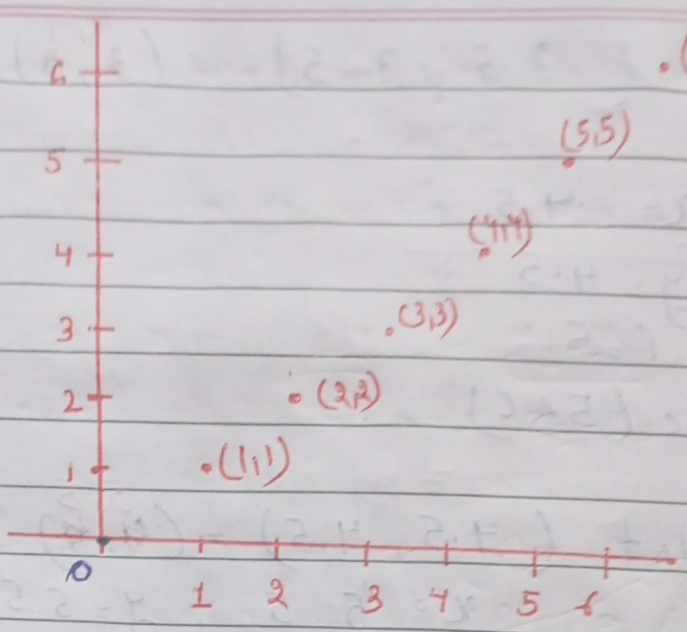
plot (5.5, 5.5) = (6, 6)

x = 6.5

y = 6.5

r = 7 → false

stop.



Numerical - 2 ⇒ Plot a line b/w the points (5, 4) and (12, 7) using DDA.

Sol

Plot (5, 4)

i=1	plot (6, 4)
i=2	plot (7, 5)
i=3	plot (8, 5)
i=4	plot (9, 6)
i=5	plot (10, 6)
i=6	plot (11, 6)
i=7	(12, 7)

Numerical 3 :- Plot a line b/w (0,0) & (4,6) using DDA.

Sol

Plot (0,0)

r=1	plot (1,1)
i=2	plot (1,2)
r=3	plot (2,3)
r=4	plot (3,4)
r=5	plot (3,5) ⇒
r=6	plot (5,6)

Disadvantages :-

1. Floating point add^n.
2. Round off fun$^{ct^n}$

PAGE NO.:

Bresenham's Line drawing algorithm => This is one more algorithm to generate line.

1. Slope $(m) = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

2. Decision parameter $(P) = 2\Delta y - \Delta x$
   $(P) = 2\Delta x - \Delta y$

3.) If slope $(m) < 1$

(i) $p < 0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i$

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

(ii) $p \geq 0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i + 1$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

If $m \geq 1$

$p < 0$

$x_{i+1} = x_i$

$y_{i+1} = y_i + 1$

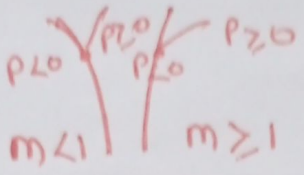
$P_{k+1} = P_k + 2\Delta x$

$p \geq 0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i + 1$

$P_{k+1} = P_k + 2\Delta x - 2\Delta y$



PAGE NO.:

Numericals :- Plot a line by using Bresenham's line generation algorithm from (1,1) to (5,3)

$x_1 = 1$, $y_1 = 1$ $x_2 = 5$, $y_2 = 3$

1 slope $(m) = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-1}{5-1} = \frac{2}{4}$

$= \frac{1}{2} = 0.5$

3) Then we have to check the slope value is greater than 1 or less than 1.

2) Then after we have to find decision parameter (P).

In this the value of m is less than 1.

$0.5 < 1$ $(m < 1)$

3) $P = 2\Delta y - \Delta x$

$= 2 \times 2 - 4$

$= 4 - 4$

$P = 0$

Then check $(P \geq 0) = (0 \geq 0)$

$x_{i+1} = x_i + 1 = 1 + 1 = 2$

$y_{i+1} = y_i + 1 = 1 + 1 = 2$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x = 0 + 2 \times 2 - 2 \times 4$

$0 + 4 - 8$

$P = 4 - 8 = -4$

↑ new value ↑ Previous value

Now check value of P is less than or greater than 0

$P = -4$

$(P < 0) = (-4 < 0)$

$x_{i+1} = x_i + 1 = 2 + 1 = 3$

$y_{i+1} = y_i = 2$

$P_{K+1} = P_K + 2\Delta y = -4 + 2 \times 2$

$= -4 + 4$

$P = 0$

check

$P = 0$

$(P \geq 0) = (0 \geq 0)$

$x_{i+1} = x_i + 1 = 3 + 1 = 4$

$y_{i+1} = y_i + 1 = 2 + 1 = 3$

$P_{K+1} = P_K + 2\Delta y - 2\Delta x = 0 + 2 \times 2 - 2 \times 4$

$= 0 + 4 - 8$

$P_{K+1} = -4$

$P = -4$

check

$(P < 0) = (-4 < 0)$

$x_{i+1} = x_i + 1 = 4 + 1 = 5$

$y_{i+1} = y_i = 3$

$P_{k+1} = P_k + 2\Delta y = -4 + 2 \times 2 = -4 + 4$

$= 0$

P	x_i	y_i	x_{i+1}	y_{i+1}
0	1	1	2	2
-4	2	2	3	3
0	3	2	4	3
-4	4	3	5	3

d = decision Parameter.

IDO = Initial decision Parameter

PAGE NO.:

Circle Drawing Algorithm ⇒ A circle is a set of all points that lie at an equal distance from a fixed point called centre

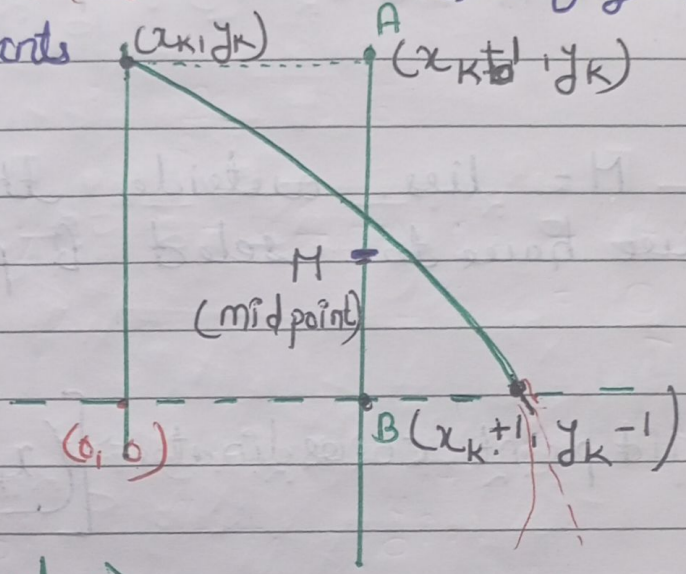
equal distance ⇒ r

1) Midpoint Circle Algo ⇒ [Circle is a symmetric figure]

(i) 4 way → 4 Quadrant

(ii) 8 way → Octants

[A से select करेंगे B
तो we have to take mid point]



Eq^n of circle ⇒

$$\Rightarrow x^2 + y^2 = r^2$$

$$\Rightarrow x^2 + y^2 - r^2 = 0$$

$$\Rightarrow (x_m)^2 + (y_m)^2 - r^2 = 0$$ → [M = mid point also have coordinate i.e x_m, y_m]

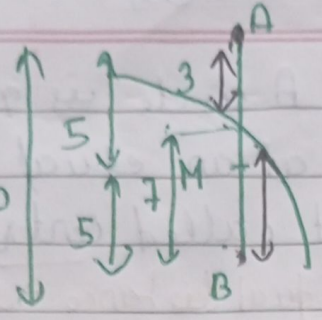
(varied point)

⇒ → 0 : point lies on circle → [Mid point कहाँ पर होगा वो देखना है]

→ less. < 0 : point lies on inside → A

→ > 0 : point lies on outside → B

[Circle से Boundary से distance ज्ञात करना)



M = lies inside the circle
[यदि दिया point से circle की minimum distance होगा उसे select करना है]

we have to select A point ($m < 0$)

M = lies outside the circle
we have to select B point ($m > 0$)

Mid point co-ordinates $\Rightarrow \left[\frac{(x_{k+1} + x_{k+1})}{2}, \frac{(y_k + y_{k-1})}{2}\right]$

$\Rightarrow \left[\frac{2x_k + 2}{2}, \frac{2y_k - 1}{2}\right]$

$\Rightarrow \left[(x_k + 1), (y_k - \frac{1}{2})\right]$

$$d_k = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \quad \text{--- (i)}$$

d_k = decision parameter

$$d_{k+1} = (x_{k+1}+1)^2 + (y_{k+1}-\tfrac{1}{2})^2 - r^2$$

- (ii)

Subkad eq? (ii) - (i)

$\Rightarrow d_{k+1} - d_k$

$\Rightarrow$ Substitute $\Rightarrow x_{k+1} = x_k + 1$

$$\Rightarrow d_{k+1} - d_k = \left[(x_k+1+1)^2 + (y_{k+1}-\tfrac{1}{2})^2 - r^2\right]$$

$$-\left[(x_k+1)^2 + (y_k-\tfrac{1}{2})^2 - r^2\right]$$

$$\Rightarrow d_{k+1} - d_k = \left[(x_k+2)^2 + (y_{k+1}-\tfrac{1}{2})^2 - \cancel{r^2}\right.$$

$$\left. - (x_k+1)^2 - (y_k-\tfrac{1}{2})^2 + \cancel{r^2}\right]$$

$$\Rightarrow d_{k+1} - d_k = \cancel{x_k^2} + 4 + 4x_k + (y_{k+1}-\tfrac{1}{2})^2$$

$$- \cancel{x_k^2} - 2x_k - 1 - (y_k-\tfrac{1}{2})^2$$

$$d_{k+1} - d_k = 2x_k + 3 + \left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2$$

$$d_{k+1} = d_k + 2x_k + 3 + y^2_{k+1} + \frac{1}{4} - 2 \times \frac{1}{2} \times y_{k+1}$$

$$- \left[y^2_k + \frac{1}{4} - 2 \times \frac{1}{2} \times y_k \right]$$

$$d_{k+1} = d_k + 2x_k + 3 + y^2_{k+1} + \cancel{\frac{1}{4}} - y_{k+1}$$

$$- y^2_k - \cancel{\frac{1}{4}} - y_k$$

$$\boxed{d_{k+1} = d_k + 2x_k + 3 + y^2_{k+1} - y_{k+1} - y^2_k + y_k}$$

If $d_k < 0$ then $y_{k+1} = y_k$

$$d_{k+1} = d_k + 2x_k + 3 + y^2_k - y_k - y^2_k + y_k$$

$$d_{k+1} = d_k + 2x_k + 3$$

If $d_k \geq 0$ then $y_{k+1} = y_k - 1$

$$d_{k+1} = d_k + 2x_k + 3 + (y_k - 1)^2 - (y_k - 1)$$

$$= -y^2_k + y_k$$

$$d_{k+1} \Rightarrow d_k + 2x_k + 3 + y^2_k + 1 - 2y_k - y_k + 1$$

$$- y^2_k + y_k$$

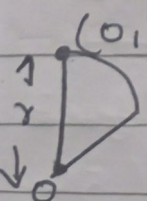
$$d_{k+1} \Rightarrow d_k + 2x_k + 5 - 2y_k.$$

$$d_{k+1} = d_k + 2x_k - 2y_k + 5$$

Now find Initial decision parameter

$x_k = 0$, $y_k = r$

$d_0 =$



PAGE NO.:

Substitute in eqn (i)

$$d_0 = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$= 1 + r^2 + \frac{1}{4} - r - r^2$$

$$= \frac{5}{4} - r$$

$$\boxed{d_0 = 1 - r}$$

step by step

Algorithm of Mid-Point ⇒ If r is the radius of the circle to be drawn and origin is its centre, then to plot first octant of circle, do the following

1. plot the initial point (x_k, y_k) such that

$$x_k = 0$$
$$y_k = r$$

2. find initial decision parameter

$$d_0 = 1 - r = \frac{5}{4} - r$$ floating point value

3. If

$d_0 < 0$ then

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

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

$$d_{k+1} = d_k + 2x_k + 3$$

4. If $d_k > 0$ then

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

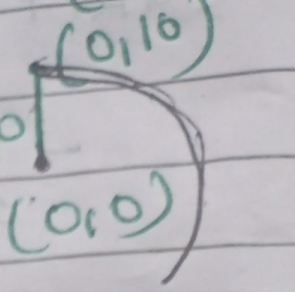
$$y_{k+1} = y_k - 1$$

$$d_{k+1} = d_k + 2(x_k - y_k) + 5$$

5. Repeat step 3, 4. until x becomes greater than or equal to y

6. To plot the complete circle, reflect each point of the first octant into 7 symmetry other octant making use of 8-way symmetry

Numerical ⇒ $(0, 0)$
$r = 10$



Sol^n (i) $x_k = 0 \quad y_k = 10$
$r = 10$

(ii) $d_0 = 1 - r$
$= 1 - 10$
$= -9$

(iii) decision parameter is less than 0.

$d_0 < 0$

$\Rightarrow x_{k+1} = x_k + 1 = 0 + 1 = 1$

$x_1 = 1$

$y_{k+1} = y_k = 10$

$y_1 = 10$

$d_{k+1} = d_k + 2x_k + 3 = -6$

$= -9 + 2 \times 0 + 3$

$= -9 + 3$

$d_{k+1} = d_1 = -6$

$\Rightarrow d_1 < 0 \Rightarrow -6 < 0$

$x_2 = x_1 + 1 = 1 + 1 = 2 \quad y_2 = 10$

$d_{k+1} = d_2 \Rightarrow d_k + 2x_k + 3 = -6 + 2 \times 1 + 3$

$\Rightarrow -6 + 2 + 3 \Rightarrow -6 + 5 \Rightarrow -1$

$d_2 = -1$

$x_2 = 2 \quad y_2 = 10$

$\Rightarrow d_2 < 0 \Rightarrow -1 < 0$

$x_3 = 2 + 1 = 3 \quad x_3 = 3$

$y_3 = 10$

$d_3 = d_2 + 2x_2 + 3 = -1 + 2 \times 2 + 3 = -1 + 4 + 3$

$= 6$

$d_3 = 6$

$d_3 > 0 = 6 > 0$

$x_4 = 3+1 = 4$, $x_4 = 4$

$y_4 = 10-1 =$ $y_4 = 9$

$d_4 = d_3 + 2(x_3 - y_3) + 5$

$= 6 + 2(3-10) + 5$

$= 6 + 2 \times -7 + 5$

$= 6 - 14 + 5$

$= -3$ 6 - 9

$d_4 = -3$

$d_4 < 0 = -3 < 0$

$x_5 = 4+1 = 5$ $x_5 = 5$

$y_5 = 9$

$d_5 = d_4 + 2x_4 + 3$

$= -3 + 2 \times 4 + 3$

$= -3 + 8 + 3$

$d_5 = 8$

$d_5 > 0$, $8 > 0$

$x_6 = 6$ $y_6 = 8$

$d_6 = d_5 + 2(x_5 - y_5) + 5$

$= 8 + (5-9) + 5$

$= 8 - 4 + 5$

$d_6 = 9$

PAGE NO.:

Numerical

$d_6 > 0 \Rightarrow$ 9:20

$x_{k+1} = x_7 = 6+1 = 7$

$y_{k+1} = y_7 = 8-1 = 7$

5 $x \geq y$

$7 \geq 7 \rightarrow$ stop

6 plot the complete circle

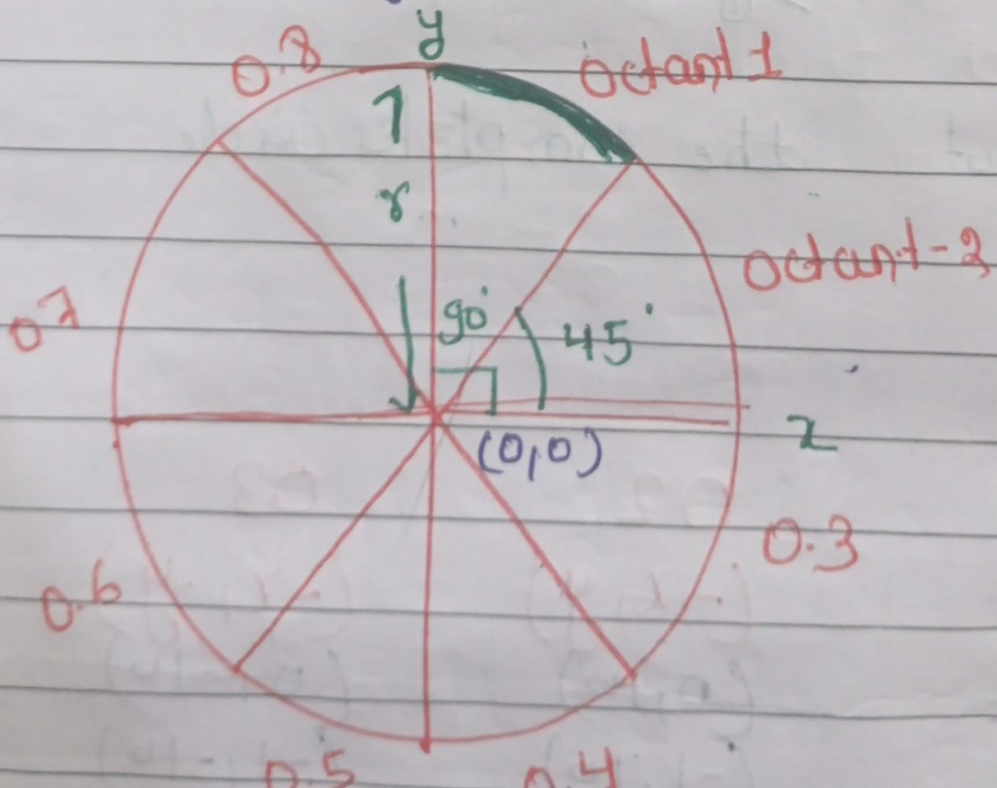
01	02	03	04
(x, y)	(-x, y)	(-x, -y)	(x, -y)
(0, 10)	(0, 10)	(0, -10)	(0, -10)
(1, 10)	(-1, 10)	(-1, -10)	(1, -10)
(2, 10)	(-2, 10)	(-2, -10)	(2, -10)
(3, 10)	(-3, 10)	(-3, -10)	(3, -10)
(4, 9)	(-4, 9)	(-4, -9)	(4, -9)
(5, 9)	(-5, 9)	(-5, -9)	(5, -9)
(6, 8)	(-6, 8)	(-6, -8)	(6, -8)
(8, 6)	(-8, 6)	(-8, -6)	(8, -6)
(9, 5)	(-9, 5)	(-9, -5)	(9, -5)
(9, 4)	(-9, 4)	(-9, -4)	(9, -4)
(10, 3)	(-10, 3)	(-10, -3)	(10, -3)
(10, 2)	(-10, 2)	(-10, -2)	(10, -2)
(10, 1)	(-10, 1)	(-10, -1)	(10, -1)
(10, 0)		(-10, 0)	(10, 0)

PAGE NO.:

Bresehan's Circle Drawing Algorithm ⇒

This algorithm does only integer arithmetic which makes it faster than floating point.

We are creating only one octant of the circle i.e from 90° to 45°.



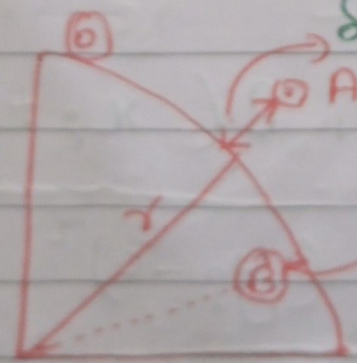
}

Features :-

1. Utilize 8-way symmetry of circle.
2. Plots 1/8th part of circle from 90° to 45°
3. From 90° to 45° x moves in (+)ve direct? and y moves in (-)ve direct?
4. In every ~~si~~ iteration, we have to chose b/w 2 points.

x_i+1, y_i $\quad$ x_i+1, y_i-1

where i: It is the previous iteration number



$\delta dA = \sqrt{(x_i+1-0)^2 + (y_i-0)^2}$

dA = distance from centre (0,0) to point

$$\delta B = \sqrt{(x_k+1-0)^2 + (y_k-1-0)^2}$$

$dA = \delta A + r \rightarrow (+)ve$

$\delta A = dA + r$

$\delta B = dB - r$

$dB = \delta B - r \rightarrow (-)ve$

$r = \delta B + dB$

$dB = r - \delta B$ — 1

$\delta A \Rightarrow$ Distance from boundary to point A

$\delta B \Rightarrow$ Distance from point B to circle boundary

$dA \Rightarrow$ Distance from point A to centre (0,0)

$dB \Rightarrow$ Distance from point B to centre

$\delta A' < \delta B'$

$dA = r + \delta A$

$dB + \delta B$

Select pixel A to plot the next pixel

$\delta A' > \delta B'$ (+)ve

Choose pixel B to approximate the circle

A (x_i+1, y_i) , B (x_i+1, y_i-1)

$$dA = \sqrt{(x_i+1-0)^2 + (y_i-0)^2}$$

$$= \sqrt{(x_i+1)^2 + y_i^2}$$

$$dB = \sqrt{(x_i+1-0)^2 + (y_i-1-0)^2}$$

$$= \sqrt{(x_i+1)^2 + (y_i-1)^2}$$

$$\delta A = dA - r \qquad \delta B = dB - r$$

$$\delta A^1 = dA^2 - r^2 \qquad \delta B^1 = dB^2 - r^2$$

$$DP \text{ (Decision parameter)} = \delta A^1 + \delta B^1$$

$$P_k \Rightarrow (dA^2 - r^2) + (dB^2 - r^2)$$

$$\Rightarrow [(x_k+1)^2 - r^2 + y_i] + [(x_i+1)^2 + (y_i+1)^2 - r^2]$$

$$P_k \Rightarrow 2(x_i+1)^2 - 2r^2 + (y_i-1)^2 + y_i^2 \quad \text{--- (1)}$$

$$P_{k+1} = 2(x_{i+1}+1)^2 - 2r^2 + (y_{i+1}-1)^3 + y_{i+1}^3 \quad \text{--- (2)}$$

$$D_{K+1} - D_K = 2(x_{i+1}+1)^2 - 2(x_i+1)^2 + (y_{i+1}-1)^2 + (y_i-1)^2 + y_{i+1}^2 - y_i^2 - 2x_i^2 + 2x_i^2$$

putting $x_{i+1} = x_i + 1$

$$D_{K+1} - D_K = 2(x_i+2)^2 - 2(x_i+1)^2 + (y_{i+1}-1)^2 + (y_i-1)^2 + y_{i+1}^2 - y_i^2$$

$$D_{K+1} - D_K = 2[x_i^2 + 4 + 4x_i] - 2[x_i^2 + 1 + 2x_i] + [y_{i+1}^2 + 1 - 2y_{i+1}] - [y_i^2 + 1 - 2y_i] + y_{i+1}^2 - y_i^2$$

$$D_{K+1} = D_K + 4x_i + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) + 6$$

If $D_K < 0 \Rightarrow$ Point A is chosen

$\Rightarrow y_{i+1} = y_i$

$D_{K+1} = D_K + 4x_i + 2(y_i^2 - y_i^2) - 2(y_i - y_i) + 6$

$$\boxed{D_{K+1} = D_K + 4x_i + 6}$$

If $D_K \geq 0 \Rightarrow$ Point B is chosen

$\Rightarrow y_{i+1} = y_i - 1$

$D_{K+1} = D_K + 4x_{i+1} + 2((y_i - 1)^2 - y_i^2) - 2((y_i - 1 - y_i) + 6$

$= D_K + 4x_{i+1} + 2(y_i^2 + 1 - 2y_i - y_i^2) - 2(-1) + 6$

$\Rightarrow D_K + 4x_i + 2(1 - 2y_i) + 2 + 6$

PAGE NO.:

$$D_{k+1} \Rightarrow D_k + 4x_i - 4y_i + 10$$

from eqn (1)

Initial Decision Parameter from eqn (1)

$$D_0 = 2(x_0+1)^2 + y_0^2 + (y_0-1)^2 - 2y_0^2$$

$$x_0 = 0$$
$$y_0 = r$$

$$D_0 = 2(0+1)^2 + (r)^2 + (r-1)^2 - 2r^2$$

$$= 2 + r^2 + r^2 + 1 - 2r - 2r^2$$

$$D_0 = 3 - 2r$$ - Integral value

features: Algorithm of Bresenham's circle Draw

1. Determine the radius 'r' of the circle from the given information

2. Ensure the centre of the circle lies at ORIGIN i.e (0,0)

3. Plot the first pixel of the

first octant as $(0, r)$ $[x=0, y=r]$

4. Calculate initial decision parameter as $\boxed{d = 3-2r}$

5. Repeat till $x \leq y$:

(i) If $d_k < 0$

↳ $d_{k+1} = d_k + 4x_k + 6$

↳ $x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

6. Else if $d_k \geq 0$ then

↳ $d_{k+1} = d_k + 4(x_k - y_k) + 10$

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

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

7 Plot (x_{k+1}, y_{k+1}) octant as well.

8. Determine and plot symmetric point in other

Numericals:- Plot the circle whose radius is 8 and whose center co-ordinate $(0,0)$

Soln (i) start point $x_0 = 0$

$y_0 = r = 8$

(ii) Decision parameter (P) $P = 3 - 2r$

$P = 3 - 2 \times 8$

$= 3 - 16$

$\boxed{P = -13}$

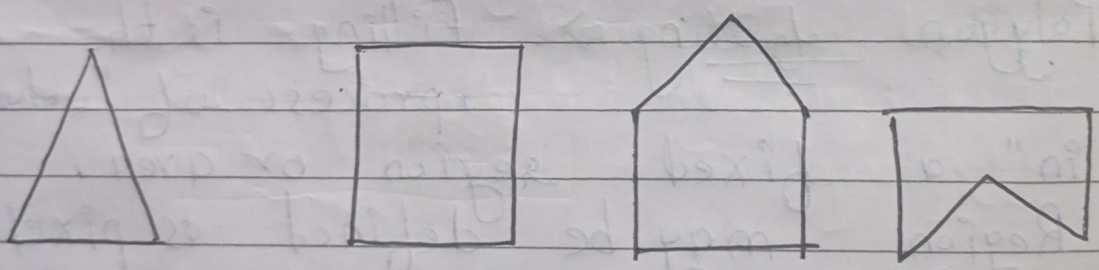
PAGE NO.:

Polygon ⇒ Polygon is a figure having many sides. It may be represented as a no. of line segment connected end to end to form a closed figure.

The line segment which form the boundary of polygon are called edges or side of polygon.

The end points of the sides are called the polygon vertices.

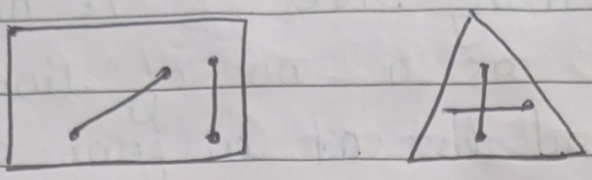
Polygon must be a close figure always.



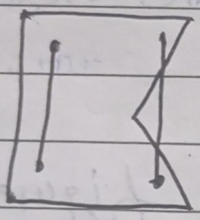
We can divide polygon into 2 types
1. Convex
2. Concave

1. Convex ⇒ Convex polygon is a polygon in which if you take any two points which are surely

inside the polygon.



2. Concave ⇒ The polygon which are not convex are called as concave polygon



Polygon filling ⇒ filling is the process of "coloring in" a fixed region or area. Region may be defined as pixel level or geometric level.

When the region are defined at pixel level (There are 2 basic approaches to area filling on raster systems) we are having 2 diff algorithm

1. Boundary Fill
2. Flood fill

Before we going to actual algorithm we will see 4 connected and 8 connected pixel concept.

1. 4 connected ⇒ In 4 connected method the pixels may have up to 4 neighboring pixels. In this method the pixels may have right, above, left and below of the current pixel.

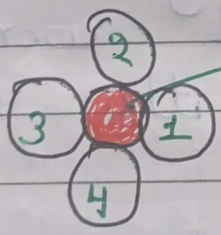


fig-: 4 connected

2. 8 connected method ⇒ In 8 connected method the pixels may have up to 8 neighboring pixels. Here from one pixel location we are finding the neighboring 8 pixel position.

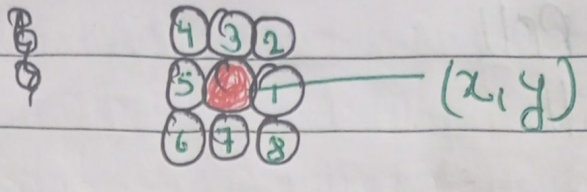
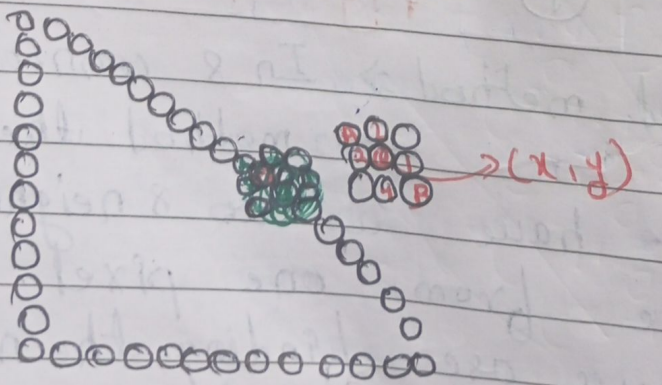


fig:- 8 connected.

By using any one of these technique we can fill the interior of the polygon. But there are some case where the use of 4-connected method is not efficient.

Suppose we want to fill a Δ, so we are using 4-connected method. It will work fine. but at boundary this method is not efficient



suppose we have filled point (x,y) Now from that point we have to select either point A or B which is not possible by using 4 connected method. because in 4 connected method we can go from (x,y) to either 1, 2, 3 or 4 and not A and B So in this case we have to use 8. 8 connected method where we can choose any one of the neighboring pixel.

1 Boundary fill Algorithm ⇒ This algorithm is very simple. It needs one point which is surely inside the polygon. This point is called "Seed Point". which is nothing but a point from which we are starting the filling process. This is a recursive method. The algorithm check to see if the seed pixel has a boundary pixel color or not.

If the answer is no, then fill that pixel with color of boundary and make recursive call to itself using each of its neighboring pixels as new seed. If the pixel color is same as boundary color then return to its caller.

Algorithm →

```
boundary (x, y, f, b)        [ b = boundary color
{                              f = fill color
  If (getpixel (x,y) != b
        &&
      getpixel (x,y) != f)
  {
    putpixel (x,y,f);
```

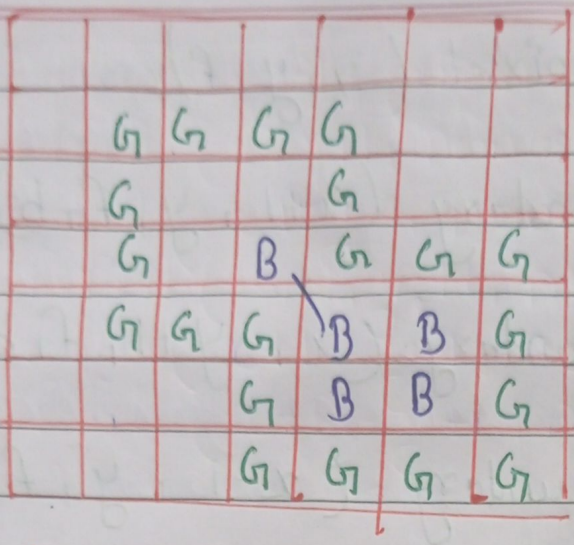
PAGE NO.:

```
putpixel (x, y, f);
boundary (x+1, y, f, b);
boundary (x, y+1, f, b);
boundary (x-1, y, f, b);
boundary (x, y-1, f, b);
boundary (x-1, y-1, f, b);
boundary (x-1, y+1, f, b);
boundary (x+1, y-1, f, b);
boundary (x+1, y+1, f, b);
}
}
```

4 connected

8 connected

Boundary



x-1, y+1	x, y+1	x+1, y+1
x-1, y	x, y	x+1, y
x-1, y-1	x, y-1	x+1, y-1

Limitations :-

1. It uses high no. of recursive calls, it take more time and m/o.

2. Flood fill Algorithm | → Seed fill Algorithm | Forest Fire fill Algorithm

The limitation of boundary fill algorithm are overcame

in flood fill algorithm.

This algo also begins with speed point which must surely inside the polygon. Now instead of checking the boundary color this algorithm checks whether the pixel is having the polygon original color i.e previous or odd color. If yes, then fill that pixel with new color. and uses each of the pixels neighbouring pixel as a new seed in a recursive call.

If the answer is no. ie the color of pixel is already changed then return to its caller.

Algorithm →

```
f-fill (x, y, newcolor)
{
    current = getpixel (x, y);
    if (current != newcolor)
    {
        putpixel (x, y, newcolor);
        f-fill (x-1, y, newcolor);
        f-fill (x+1, y, newcolor);
```

```
f-fill (x, y-1, newcolor);
f-fill (x, y+1, newcolor);
f-fill (x-1, y-1, newcolor);
f-fill (x-1, y+1, newcolor);
f-fill (x+1, y-1, newcolor);
f-fill (x+1, y+1, newcolor);
}
}
```

Other way =>

```
flood (x, y, n, o)        { n = newcolor
{                           o = oldcolor
  If (getpixel (x, y) == o)   { n = B }
  {                             o = W
    putpixel (x, y, n);
      flood (x+1, y, n, o)
      flood (x, y+1, n, o)
      flood (x-1, y, n, o)
      flood (x, y-1, n, o)
```