Module - II

Output Primitives

~~process of~~

=> **scan conversion:**

'process of converting basic low-level obj into their corresponding pixel map representa^n

-> **scan conversion of line:**

I **DDA line drawing (al):**

DDA (digital diffrncial analysr (Al)) is a line drawing (al) which uses cartitions slope intercept eq of straight line

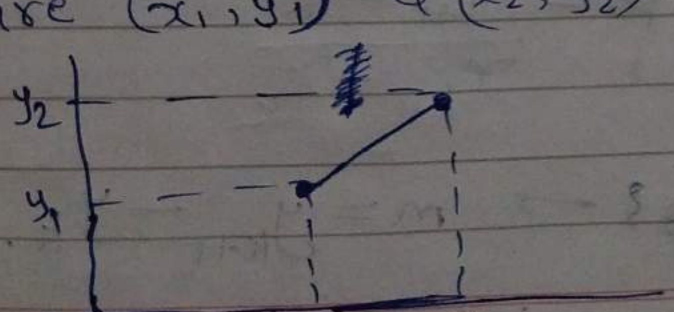
Line drawing is accomblished by calculating intrmediate point coordinates along the line path b/w 2 gvn 'n' points.

The cartition slope intrcept eq for a straight line is -

$$\boxed{y = mx + b}$$

where $m \to$ slope ; $b \to$ intrcept

The 2 end points of the line are gvn which are (x_1, y_1) & (x_2, y_2).

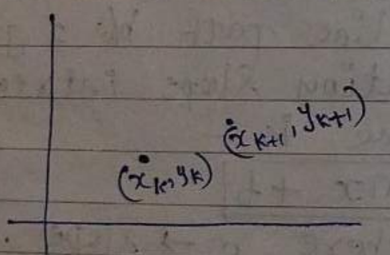


To load the specified color into the frame buffer at a particular position, we will assume we hve available low-level procedure of the form putpixel(x, y).
Similarily, for retrive the crnt frame buffr intesity, we assume to hve procedure getpixel(x,y).
we can determine values for the slope 'm' by eq – $\boxed{m = \frac{y_2 - y_1}{x_2 - x_1}}$

There are 3 cases based on value of m –

<u>case 1</u>: $(m<1)$ → eg = m = 0.28 (i.e) m<1.

x changes with unit intrval



$(x_1, y_1) \to (x_k, y_k)$
$(x_2, y_2) \to (x_{k+1}, y_{k+1})$

<u>coordinates</u>
$(x_{k+1}, y_k + m)$

1) m? $x_{k+1} = x_k + 1$

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$

2) for finding m? → $m = y_{k+1} - y_k$.
whn y_1, y_2 gvn

for finding y_2 ⟹ $\underline{y_{k+1} = m + y_k}$

(1) whn x_1 & x_2 gvn →

<u>case 2</u>: $(m>1)$ $(x_{k+1}/m, y_{k+1})$

y changes with unit intrval.

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

① $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$

② $m = \frac{1}{x_{k+1} - x_k}$

③ $x_{k+1} = \frac{x_k + 1}{m}$

④ $x_{k+1}, y_{k+1} = \text{round}(x_k + 1/m), y_k + 1$

<u>case 3</u>: $(m=1)$ (x_{k+1}, y_{k+1})

x changes with unit intrval

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

y changes with unit intrval

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

$\boxed{x_{k+1}, y_{k+1} = (x_k + 1, y_k + 1)}$.

if $(x_k, y_k) \to 2, 3$
then $(x_{k+1}, y_{k+1}) = 3, 4$.

<u>egs</u> –

1) $(10, 10)$ & $(15, 16)$
A) x_1 y_1 x_2 y_2.

?
(10,10) (15,16)

$\therefore m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{16-10}{15-10} = \frac{6}{5}$

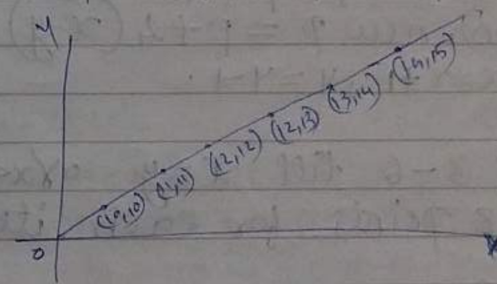
$= \underline{\underline{1.2}}$

$\therefore m > 1$

	x	y	xplot	yplot	coor	calculation
1)	10	10	10	10	(10,10)	$(x_{k+1}/m, y_{k+1})$ $\Rightarrow x_k = 10$ $x_{k+1} = 10+1 = 11$ $\Rightarrow x_{k+1}/m = {}^{10+1}/_{1.2}$ $= 10.83$ $\Rightarrow y_k = 10$ $\therefore y_{k+1} = 10+1 = 11$ $\Rightarrow (10.83, 11)$
2)	10.83	11	11 (round for plotting)	11	(11,11)	$(x_{k+1}/m, y_{k+1})$ $x_{k+1}/m =$ $= 10.83 + 0.83$ $= 11.66$ $y_{k+1} = 12$
3)	11.66	12	12	12	(12,12)	$x_{k+1}/m = 11.66 + 0.83$ $= 12.49$ $y_{k+1} = 13$
4)	12.49	13	12 ↑	13	(12,13)	$x_{k+1}/m = 12.49 + 0.83$ $= 13.32$ $y_{k+1} = 14$
5)	13.32	14	13	14	(13,14)	$x_{k+1}/m = 13.32 + 0.83$ $= 14.15$ $y_{k+1} = 15$
6)	14.15	15	14	15	(14,15)	$x_{k+1}/m = 14.15 + 0.83$ $= 14.98$ $y_{k+1} = 16.$

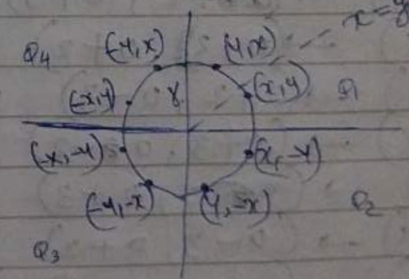
$12.4 \to (<5) \to 12$

$$\left[\; x_{k+1}/m \to x_k + 1/m = 10.83 + 1/1.2 = 10.83 + 0.83 = \underline{11.66} \;\right]$$



II Bresenham's Circle generating (al):

(8 octat)



$Q_1 \to (+,+)$

$Q_2 \to (+,-)$

$Q_3 \to (-,-)$

$Q_4 \to (-,+)$

* If the coor is $(0,0) \rightarrow (x, y)$
but we take $\rightarrow x = 0, y = r$.
$(r \rightarrow \theta)$

8 octat (or [illegible]) $x = y$ [illegible]

Steps :-

1) Input X_c, Y_c, r
2) Set $X = 0, Y = r$, then $P = 3 - 2r$
3) Plot 8 points (X_c, Y_c, X, Y).
4) check $P < 0$, new $P = P + 4x + 6$, then $x = x+1, Y = Y$.
5) otherwise $(P > 0)$, new $P = P + 4(x - y) + 10$. [illegible]
then, $x = x+1, Y = Y-1$.
6) $X+1$
7) repeat Steps 3-6 till $x = y$ or $x <= Y$.

* plot this 8 points for each iteration -

plot 8 points $(X_c, Y_c, X, Y) \rightarrow$ plot 8 points $(0, 0, 2, 3)$

coords (8)

1) $(X_c + X, Y_c + Y) \rightarrow (0 + 2, 0 + 3) = (2, 3)$
2) $(X_c + Y, Y_c + X) \rightarrow (0 + 3, 0 + 2) = (3, 2)$
3) $(X_c + X, Y_c - Y) \rightarrow (0 + 2, 0 - 3) = (2, -3)$
4) $(X_c + Y, Y_c - X) \rightarrow (0 + 3, 0 - 2) = (3, -2)$
5) $(X_c - X, Y_c + Y) \rightarrow (0 - 2, 0 + 3) = (-2, 3)$
6) $(X_c - Y, Y_c + X) \rightarrow (0 - 3, 0 + 2) = (-3, 2)$
7) $(X_c - X, Y_c - Y) \rightarrow (0 - 2, 0 - 3) = (-2, -3)$
8) $(X_c - Y, Y_c - X) \rightarrow (0 - 3, 0 - 2) = (-3, -2)$

eg → Q) draw a ○ radius = 3
origin (0, 0).

A) $X_c = 0, Y_c = 0, X = 0, Y = 3$.
Assign $x = 0, Y = r = 3$
∴ Initial value of P,
$P = 3 - 2r = 3 - 2 \times 3 = -3$.
∴ $P < 0$, then
$P = P + 4x + 6$
$= -3 + 4 \times 0 + 6$
$P = 3$
then $x = x + 1 = 0 + 1 = 1$
$Y = 3$.
$\Rightarrow (x, y) = (1, 3)$

$X_c = 0$
$Y_c = 0$
$x = 1$
$Y = 3$

8 coords,

1) $(X_c + X, Y_c + Y) = (0 + 1, 0 + 3) = (1, 3)$
2) $(X_c + Y, Y_c + X) = (0 + 3, 0 + 1) = (3, 1)$
3) $(X_c + X, Y_c - Y) = (0 + 1, 0 - 3) = (1, -3)$
4) $(X_c + Y, Y_c - X) = (0 + 3, 0 - 1) = (3, -1)$
5) $(X_c - X, Y_c + Y) = (0 - 1, 0 + 3) = (-1, 3)$
6) $(X_c - Y, Y_c + X) = (0 - 3, 0 + 1) = (-3, 1)$
7) $(X_c - X, Y_c - Y) = (0 - 1, 0 - 3) = (-1, -3)$
8) $(X_c - Y, Y_c - X) = (0 - 3, 0 - 1) = (-3, -1)$

$P \geq 0 \Rightarrow P = P + 4(x - y) + 10$
$= 3 + 4(1 - 3) + 10$
$= 3 + 4 \times -2 + 10$ $= 3 - 12 + 10$
$= 3 - 8 + 10$ $= 5$
$= 5$

thn, $x = x+1 = 1+1 = 2$.
$y = y-1 = 3-1 = 2$

∴ plot 8 points $(x_c, y_c, x, y) \rightarrow (0, 0, 2, 2)$

1) $(x_c+x, y_c+y) = (0+2, 0+2) = (2,2)$
2) $(x_c+y, y_c+x) = (0+2, 0+2) = (2,2)$
3) $(x_c+x, y_c-y) = (0+2, 0-2) = (2,-2)$
4) $(x_c+y, y_c-x) = (0+2, 0-2) = (2,-2)$
5) $(x_c-x, y_c+y) = (0-2, 0+2) = (-2,2)$
6) $(x_c-y, y_c+x) = (0-2, 0+2) = (-2,2)$
7) $(x_c-x, y_c-y) = (0-2, 0-2) = (-2,-2)$
8) $(x_c-y, y_c-x) = (0-2, 0-2) = (-2,-2)$

→ Symmetric proprty of ○ :

The most imp thing in drawing a ○ is learning how the ○ is drawn using 8-way symmetric.
Based on mirror reflection, if we see right hand in the mirror we will see left hand, similarily if we see pixel (x,y) in mirror we will see (y,x).
so point P_1 (x,y) will become P_2 (y,x) after reflection; P_3 (-y,x) will become (-x,y).

let us understand how this -ve signs are takn remembr 2 things—

1) If we mirror reflect (x,y) with respect to x-axis than 'y' will change, so it will become (x,-y) according to x axis. Similarly if we reflect mirror according to y axis, x will change, so the point will become (-x,y).
2) read point (x,y) such as x is on x position & y is on y position, so when we reflect with respect to x axis then y position becomes -ve & with respect to y axis.

Take this logic & reflection of points P_2 (y,x) with respect to y axis will make the x position -ve & not where x is written, so it will become P_3 (-y,x).

Reflection of point P_2 (x,y) with respect to x axis will make y position -ve & not where y is written, so it will become P_1 (x,y).

III Polygon filling (al):

Givn vertices of (poly) & a color, our aim is to fill the (poly) with the particular color.

Here we are using 2 methods –

i) Scan line method (2) seed method

I Scan line (M):

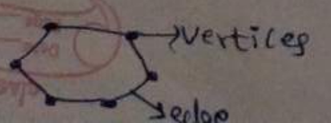
* intersect (poly) with horizontal lines & fills the (poly) b/w pairs of intrsection.
* eg → scan line (poly) filling (al).

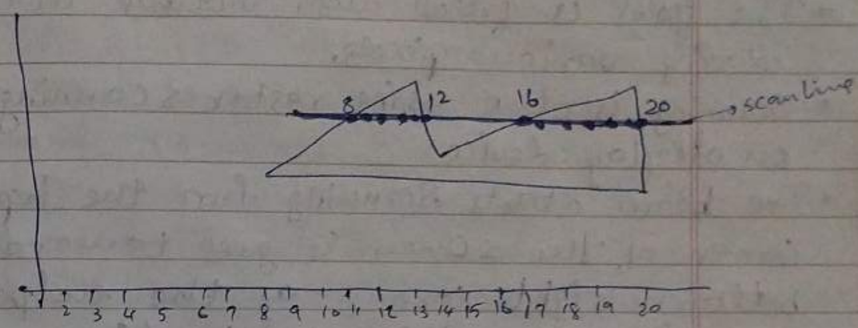
II Seed (M):

* Start from an interior position & paint until the boundary condi^n is reached.
* eg → Boundary filling (al) & flood filling (al).

a) Scan line (poly) filling (al):

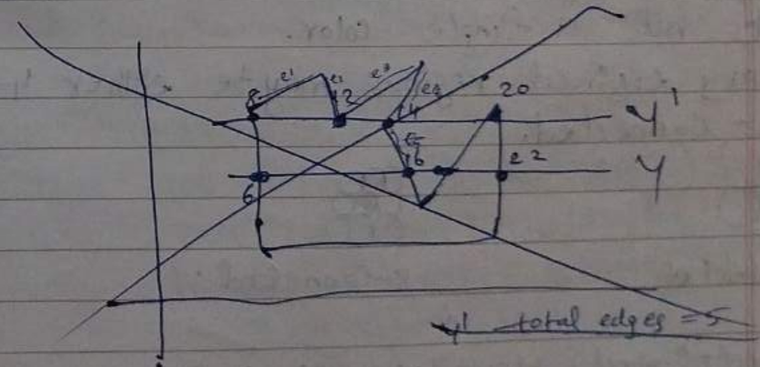
* find the intrsection of scan line with all edges of the (poly).
* Sort the intrsections by using X coords (i.e) from Left to Right.
* Make pairs of intrsection & fill in color within all the pixels.

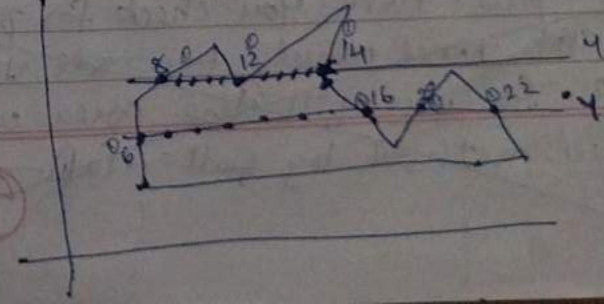




① (8, 12, 16, 20) ② already sorted.
③ make pair → (8,12) (16,20) രണ്ട് pairs-ന്റെ ഇടയിൽ
ഉള്ള space color ചെയ്യൽ.

① list of intrsection point
② Sort it off.
③ make pairs of intrsection – (8,12) (16,20)
fill in all pixel with the gvn color inside the pixel.





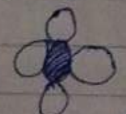
y' → total vertices = 5

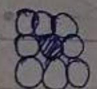
y' = (8,12) (12,14)

y = (6,16) (20,22)

* The (poly) is filled with various colors by coloring various pixels.
* Scanning is done using raster scanning concept on display device.
* The beam starts scanning from the top left corner of the screen & goes toward the bottom right corner as the endpoint.
* The (al) finds points of intersection of the line with (poly) while moving from L to R & T to btm.

b) Boundary Filling (al):
* here, the basic concept is filling the color in closed area by starting at a point inside a region & paint the interior outward towards the boundary.
* 1 requirement for B.F.(al) is that the boundary has to hue a single color.
* Boundary defined regions may be either 4-connected or 8-connected.


4-Connected


8-Connected.

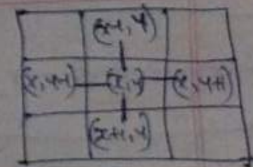
~~4-connected pixel: Here, you check for pixels adjacent to the crnt pixel namely towards the left, R, T & B. So you fill the area with 4 connected method by follo. (al) —~~

* This (al) starts at a pixel inside the (poly) to be filled & paints the interior proceeding outwards towards the boundary.
* This (al) works only if the color with which the region has to be filled & the color of the boundary of the region are diffrnt.
* It can be implemnted by 4-connected pixels / 8-connected pixels.

a) 4-connected pixels:
* After painting a pixel, the () is called for 4 neighbouring points.
* These are the pixel positions that are R, L, above & below the crnt pixel.
* Areas filled by this method → 4-connected
* (al) —

```
void boundaryFill4 (int x, int y, int boundry,
                    int default) {
    if (getpixel(x,y) != default &&
        getpixel (x,y) != boundary) {
        putpixel (x, y, boundary);
        boundaryFill4 (x+1, y, boundary, default);
              "       (x , y+1,    "        "   );
              "       (x-1, y ,    "        "   );
              "       (x , y-1,    "        "   );
    }
}
```


4-Connected



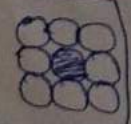
b) 8-Connected pixels:

- More complx figures are filled using this approach. The pixels to be tested are the 8 neighbouring pixels, the pixel on the R, L, above & below & 4 diagonal pixels.
- Areas are filled by this method → 8-connected.
- (al) —

```
void boundaryFill8( ~, ~ , ~ , ~ ) {
    if( ~~  && ~~ ) {
        putpixel (x, y, boundary);
        boundaryFill8 (x+1, y, boundary, default);
          "           (x, y+1,      "         );
          "           (x-1, y,      "         );
          "           (x, y-1,      "         );
          "           (x-1, y-1,    "         );
          "           (x-1, y+1,    "         );
          "           (x+1, y-1,    "         );
          "           (x+1, y+1,    "         );
    }
}
```

(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y-1)	(x, y)	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

8-connected →



- get.pixel () gives the color of specified pixel & put.pixel () draws the pixel with this specified color.

c) Flood-fill (al):

- Sometimes it is required to fill in an area that is not specified within a single color boundary.
- In such cases, we can fill areas by replacing a specified interior color instead of searching for a boundary color. This approach → A flood fill (al).
  Like Boundary fill (al), here we start with some seed & examine the neighbouring pixel.
- Here pixels are check for the specified interior color instead of boundary color & they are replaced by a new color. Using either 4-connected / 8-connected approach we can step through pixel position until all interior point hue been filled.
- (al) →

```
FloodFill4( x, y , new color, old color) {
    if get.pixel (x, y) = old color) {
        put.pixel (x, y, new color);
```

```
FloodFill4(x+1, y, newcolor, old color);
    "     (x, y+1, new color, old color);
    "     (x-1, y,       "              );
    "     (x, y-1,       "              );
  }
}
```