1

Scan Converting Lines :-

↳ Scan the pixels and convert the lines from the frame buffer on to the Screen.

There are four different Attributes :-

↳ Lines, Straight lines.
↳ Arcs, Curves, Ellipses, Circle.
↳ Polygon Drawing and Filling.
↳ Drawing text on the screen.

Drawing Lines :-

↳ Digitized spaces (Start point and end points).

Specification of Straight Line :-

↳ collection of all the addressable points.
↳ which is closely/approximates this lines.

Goals :-

↳ Not all of them are achievable with discrete space of a raster devices.

↳ Straight lines should appear straight.

↳ Lines should start and end accurately matching end points and connecting lines.

↳ Lines should have constant brightness.

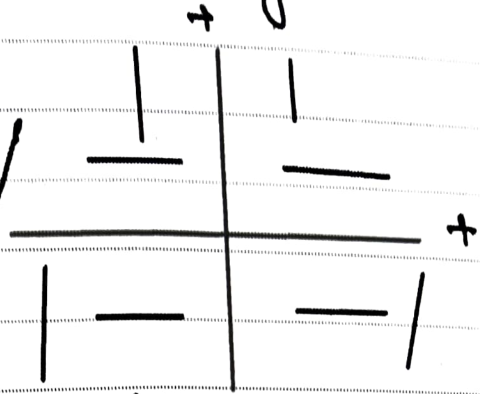
↳ Lines should be drawn as rapidly as possible.

Problems :-

↳ Determine which pixels to illuminates to satisfy the above goals.

↳ In raster display straight lines appears straight, accurate and uniform.

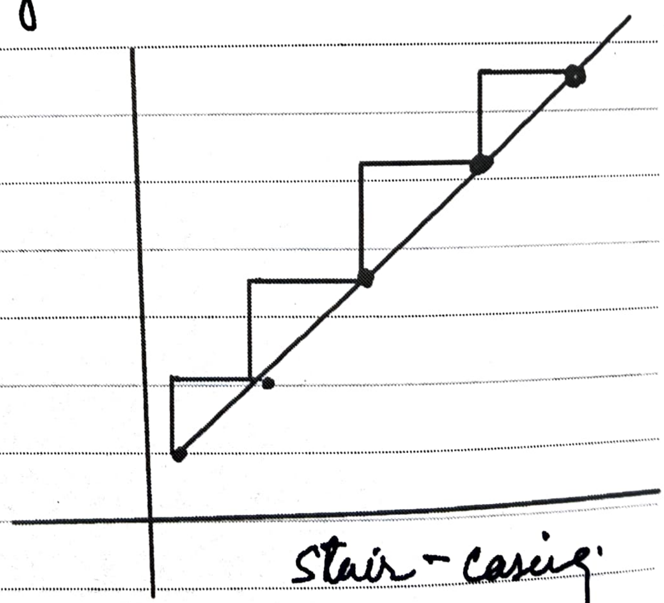
↳ Vertical and Horizontal lines with slopes = +/- 1 are easy to draw.



↳ stair-casing / jaggies / aliasing effect.

→ distortion and error.

↳ Quality of the lines draw depends on the location of pixels and their brightness.



stair-casing

Solution :-

Direct Solution :-

1. We use the straight line formula to generate the lines for more Accuracy.

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

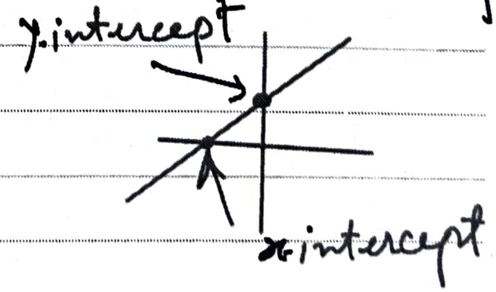
Here,

— 'y' and 'x' = Keep thing as variable.

m = slope of the lines.

c = 'x' or 'y'– intercept of the lines. where (0,6) is the y- intercept.

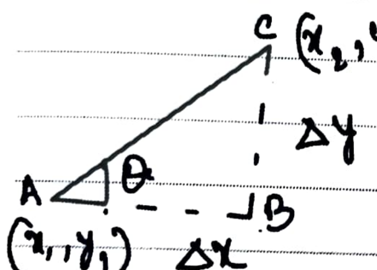
and (6,0) is the x- intercept.



So from x_0 to x_1:

Calculate round(y) from the eq^n :-

$$\text{Slope}(m) = \frac{\Delta y}{\Delta x} = \frac{\text{difference in 'y'}}{\text{difference in 'x'}} \quad \text{— (1)}$$



$$\tan\theta = \frac{\Delta y}{\Delta x} \quad \text{— (2)}$$

from (1) and (2)

$$\boxed{m = \tan\theta}$$

Example:-

Let's $b = 1$ starting point $(0,1)$ and $m = 3/5$.

$y = mx + c$

$\Rightarrow y = 3/5 \times 1 + 1$

$= 8/5$

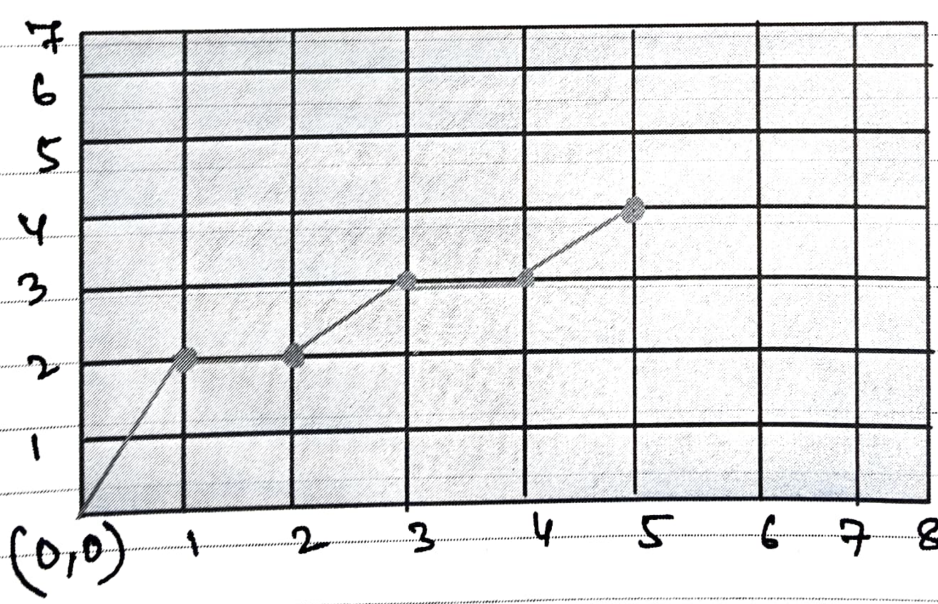
$= 1.6 \cong 2$

then :-

Pixel generation on raster screen as integer values.

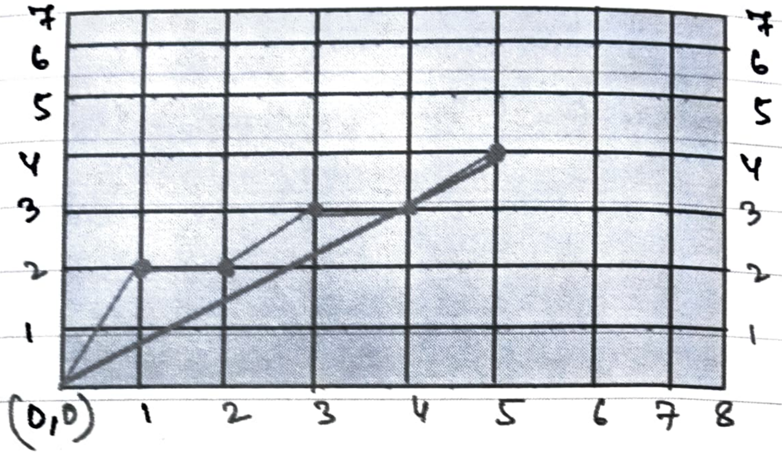
$x = 1$ $y = 2$ $=$ round $(8/5)$
$x = 2$ $y = 2$ $=$ round $(11/5)$
$x = 3$ $y = 3$ $=$ round $(14/5)$
$x = 4$ $y = 3$ $=$ round $(17/5)$
$x = 5$ $y = 4$ $=$ round $(20/5)$

Idel case



↳ we have to find the most closest points.

↳ Nearest Addresable value.



Why this method is undesirable:-

- ↳ '*' and '/' are expensive.
- ↳ Round () function needed.
- ↳ can get gaps in the lines (if slope > 1)

ex1- $y = mx + b$

Let's take slope is high = $m = 10$

$b = 2$

$x = 1, \quad y = 12$ ✓

$x = 2, \quad y = 22$ ✓

Raster Based Algorithm:-

Algorithm:-

DDA → Digital Differential Analyzer.

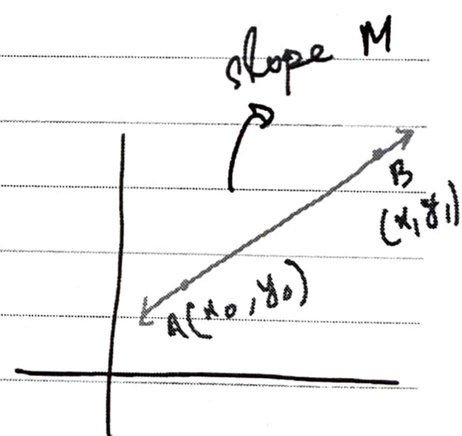
→ It is an incremental Algorithm.

→ formula :-

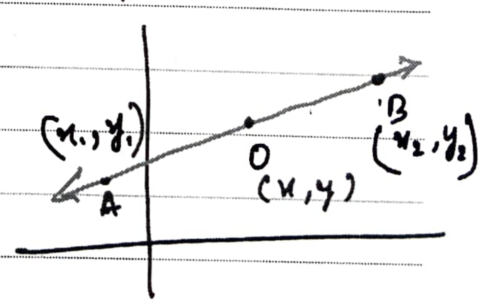
$$\boxed{y = mx + c}$$

Slope - Point form

$$\therefore m = \frac{x_1 - x_0}{y_1 - y_0}$$



Two - Point Form



$$\boxed{y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)}$$

Intercept Formula

$$\boxed{\frac{x}{a} + \frac{y}{b} = 1}$$

$a =$ x- intercept.
$b =$ y- intercept.

· Algorithm is based on :-

$$y = \frac{(y_1 - y_0)}{(x_1 - x_0)} x + c$$

Assume that $x_1 > x_0$ and

$$|dx| > |dy|$$

slope $m = \frac{dy}{dx}$, then value of slope is less than 1.

Initalization Condition :-

$$\left.\begin{aligned} dx &= x_1 - x_0; \\ dy &= y_1 - y_0; \end{aligned}\right\}$$

float point variable $\leftarrow$ $m = \frac{dy}{dx}$;

$y = y_0$;

Integer values but rational of two integer is not always integers.

for ($x = x_0$ to x_1)

draw_point (x , round(y));

$\boxed{y = y + m;}$ $\rightarrow$ float point Add.

Problem with DDA :-

↳ still uses floating point and round() inside the loop.

↳ How can be get rid of this problem.

Solve the DDA

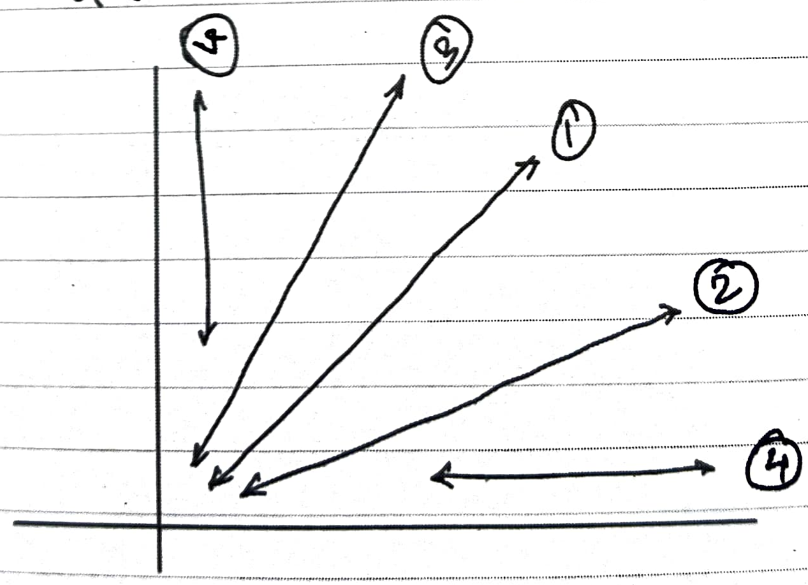
There are four condition :-

1. $\frac{dy}{dx} = 1$ { At 45° }

2. $\frac{dy}{dx} < 1$ { less than 45° }

3. $\frac{dy}{dx} > 1$ { greater than 45° }

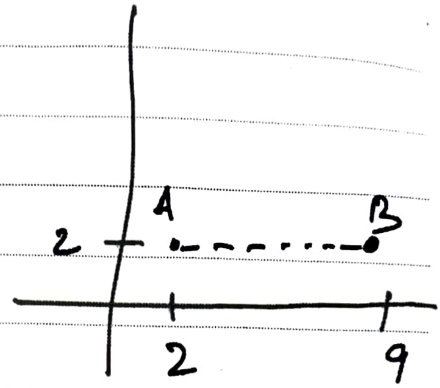
4. $\frac{dy}{dx} = 0$ { ||el to 'x' or 'y' axis }



check the conditions

Q. $A(x_1, y_1) = (2, 2)$

$B(x_2, y_2) = (9, 2)$



Sol^n :- step 1 :- Calculate Δx and Δy.

$\Delta x = 9 - 2 = 7$ (difference in x)

$\Delta y = 2 - 2 = 0$ (difference in y)

step 2 :- slope calculation

$$m = \frac{\Delta y}{\Delta x} = \frac{0}{7} = 0$$

step 3 :- Pixels generation

steps = { greater value in between Δx and Δy }

Here we consider No of step required for pixel generation is $= \underline{\underline{7}}$

because $\boxed{\Delta x = 7 \quad \text{and} \quad \Delta y = 0}$

steps is also consider to be the no. of points required to reach from point A to point B.

step 4:- Increments from one pixel to another.

$$x_{inc} = \frac{\Delta x}{steps} \quad \text{and} \quad y_{inc} = \frac{\Delta y}{steps}$$

$$x_{inc} = \frac{7}{7} = 1 \quad \text{and} \quad y_{inc} = \frac{0}{7} = 0$$

{1 unit increment} {No increment}

step 5:- Generates the points to draw.

A	x	y
0	2	2
1	3	2
2	4	2
3	5	2
4	6	2

A	x	y
5	7	2
6	8	2
7	9	2

In conclusion, we generate the line based on the given value.

Ex 2:- $A(x_1, y_1) = (2,5)$
$B(x_2, y_2) = (2,12)$

Solⁿ:- **Step 1:-**

$\Delta x = x_2 - x_1 = 2-2 = 0$
$\Delta y = y_2 - y_1 = 12-5 = 7$

Step 2:- slope

$$m = \frac{\Delta y}{\Delta x} = \frac{7}{0} = \infty$$

Step 3:- Pixel steps :-

steps = 7

Step 4:- Increment

$$x_{inc} = \frac{\Delta x}{step} = \frac{0}{7} = 0.$$

$$y_{inc} = \frac{\Delta y}{step} = 7/7 = 1.$$

Steps.

A	x	y
0	2	5
1	2	6
2	2	7
3	2	8
4	2	9
5	2	10
6	2	11
7	2	12

---

condition 3 $\{m < 1\}$

$A(x_1, y_1) = (5, 4)$

$B(x_2, y_2) = (12, 7)$

solⁿ :- step 1 :-

$\Delta x = x_2 - x_1 = 12 - 5 = 7$

$\Delta y = y_2 - y_1 = 7 - 4 = 3$

step 2 :- slope

$$m = \frac{\Delta y}{\Delta x} = \frac{3}{7} = 0.42 \simeq 1$$

step 3:- Steps for Pixels.

Steps = 7

step 4:- Increment

$$x_{inc} = \frac{\Delta x}{Steps} = \frac{7}{7} = 1$$

$$y_{inc} = \frac{\Delta y}{Steps} = \frac{3}{7} = 0.4$$

step 5:-

A	x	y	round(y)
0	5	4 →	4
1	6	4.4 →	4
2	7	4.8 →	5
3	8	5.2 →	5
4	9	5.6 →	6
5	10	6.0 →	6
6	11	6.4 →	6
7	12	6.8 →	7

Remarks :- Optimization of Algorithm

In the case of $\{m < 1\}$

we direct use the formula.

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

$$\boxed{y_{k+1} = y_k + m}$$

Now step 1 and step 2 is common

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

$x_1 = x_0 + 1$
$= 5 + 1 = 6$ //

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

$y_1 = y_0 + m$
$= 4 + 0.4$
$= 4.4$ //

we generate all the points.

Condition 4 :— $\{m > 1\}$

Q

$A(x_1, y_1) = (5, 7)$

$B(x_2, y_2) = (10, 15)$

Sol^n :— **step 1 :—**

$\Delta x = x_2 - x_1 = (10-5) = 5$

$\Delta y = y_2 - y_1 = (15-7) = 8$

step 2 :— slope

$$m = \frac{\Delta y}{\Delta x} = \frac{8}{5} = 1.6 \simeq 2$$

$\{> 1\}$

step 3: **steps**

steps = 8 $\quad \Delta y$ is greater.

step 4: we use the optimization

$$\boxed{x_{k+1} = x_k + \frac{1}{m}}$$

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

$$x_1 = x_0 + \frac{1}{m} \Rightarrow 5 + \frac{5}{8} = 5.6$$

$$y_1 = y_0 + m \Rightarrow 7 + 1 = 8$$

A	x	y
0	5	7
1	5.6	8
2	6.2	9
3	6.8	10
4	7.4	11
5	8	12
6	8.6	13
7	9.2	14
8	9.8	15

condition 5 :- $\{m = 1\}$ slope

Q A $(x_1, y_1) = (12, 9)$
B $(x_2, y_2) = (17, 14)$

sol^n :- **step 1 :-**

$\Delta x = x_2 - x_1 = 17 - 12 = 5$
$\Delta y = y_2 - y_1 = 14 - 9 = 5$

Step 2:- Slope

$$m = \frac{\Delta y}{\Delta x} = \frac{5}{5} = 1.$$

Step 3:- Steps :-

steps = 5

Step 4:- Increment

$$\boxed{\begin{aligned} x_{k+1} &= x_k + m \\ y_{k+1} &= y_k + m \end{aligned}}$$

$x_1 = x_0 + 1$
$= 12 + 1 \Rightarrow 13$

$y_1 = y_0 + 1$
$= 9 + 1 \Rightarrow 10$

A	x	y
0	12	9
1	13	10
2	14	11
3	15	12
4	16	13
5	17	14

Remarks :- For First co-ordinate is larger then second one.

~~Q~~ A $(x_1, y_1) = (17, 14)$
B $(x_2, y_2) = (12, 9)$

step 1 :- $\Delta y = y_2 - y_1 = 9 - 14 = -5$
$\Delta x = x_2 - x_1 = 12 - 17 = -5$

step 2 :- slope $(m) = \frac{\Delta y}{\Delta x} = \frac{\cancel{-5}}{\cancel{-5}} = 1$

step 3 :- steps $= 5$

step 4 :- $x_{inc} = \frac{\Delta x}{steps} = \frac{-\cancel{5}}{\cancel{5}} = -1$

$y_{inc} = \frac{\Delta y}{steps} = \frac{-5}{5} = -1$

Now it will generate the (-ive) graph.

Algo :-

```
DDA algo (x1, y1, x2, y2)
{
    dx = x2 - x1 ;
    dy = y2 - y1 ;
  if abs (dx) > ads(dy)
        step = abs(dx)
  else
        step = abs (dy)

  xinc = dx/step ;

  yinc = dy/step ;

  for (i=1 ; i<=step ; i++)
  { putpixel (x1, y1)
        x1 = x0 + xinc ;
        y1 = y0 + yinc ;
  }
}
```