14 Jan, 19.

Line Drawing Algorithms

→ We can't plot a floating pt. pixel value thus we round off value in DDA algo.

y_0

error { (e_0) $y_1 = y_0 + m$
$(2e_0)$ $y_2 = y_1 + m$
⋮
⋮

} error keeps on increasing and adding up

Thus actual line slightly gets drifted from original position.

Drawback of DDA

To get rid of these algorithms, we must have an algorithm that takes integer values. Thus, we have another line drawing algorithm

↓

Bresenham's Line Drawing algorith
(or mid pt. Line drawing algo.)

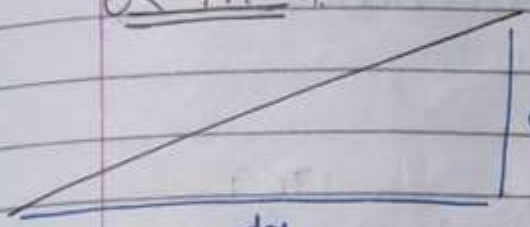
- use of mid pt. concept.
- In DDA, we take 2 cases $|m| > 1$ or $|m| < 1$
- For Bresenhams, we have 4 cases

↓

CLASSTIME | Page No.
Date / /

+ve slope, gradual
-ve slope, gradual

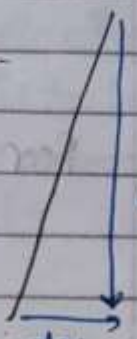
$0 < m \leq 1$



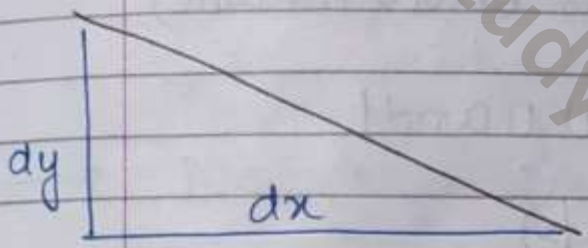
$dx \geq dy$ (+ve slope)

+ve slope, steep
-ve slope, steep.

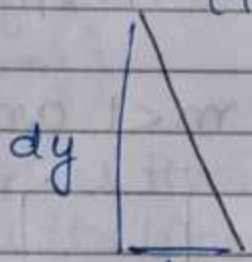
$m > 1$



$dx < dy$
(+ve slope)



$(dx > dy)$
$m \leq 1$ on -ve side



$dx < dy$
$m > 1$ on -ve side
(b/w 0 and -1)

* Derivation ↓

For a pt. (x, y) lying on the line we have

$$F(x, y) = ax + by + c = 0 \quad \text{---(1)}$$

and $y = mx + B$

Now $m = \frac{dy}{dx}$

$$y = \frac{dy}{dx} \cdot x + B$$

$$y\,dx = x\,dy + B\,dx.$$

$$x\,dy - y\,dx + B\,dx = 0 \quad \text{---(2)}$$

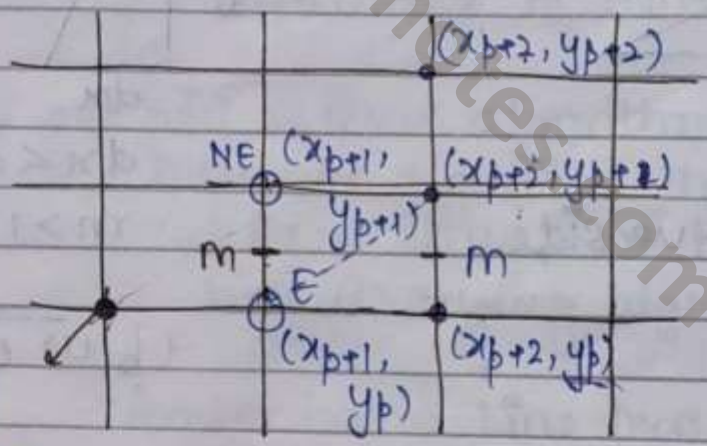
comparing (1) and (2)

$$a = dy, \quad b = -dx, \quad c = B\,dx$$

→ for any pt. (x, y) ↓

$F(x, y)$ can have values
$F(x, y) = 0$ (on the line)
$F(x, y) < 0$ (pts. above the line)
$F(x, y) > 0$ (pts. below the line)

Case I :- $m < 1$ and m is b/w 0 and 1



If m is $0 < m < 1$
i.e line is gradual
we increment x and keeps finding values of y

This is because $dx > dy$ ∴ for every value of x we get a y as x contains more no. of pts.
If we increment y, then there are some

pts. for which no x is there.

$(x_p, y_p) \rightarrow (x_{p+1}, y_p)$ or (x_{p+1}, y_{p+1})

↓

Now how to make this choice whether pt. E or pt. NE is to be chosen on basis of mid pt.

mid pt. of (x_{p+1}, y_p) and (x_{p+1}, y_{p+1})

↓

$(x_{p+1}, y_p + \frac{1}{2})$

→ Now our line either passes through this mid pt., or below the line or above the line.

→ On basis of pt. M we decide whether the line passes above the value of M, then NE has to be plotted.
If line lies below M, then E has to be plotted
If line lies on M, we can choose either pt. E or NE.

(1) Find $F(x_{p+1}, y_p + \frac{1}{2})$ ↓ 3 cases $< 0, > 0$ or $= 0$.

Case I - value of mid pt. is > 0.

deciding factor { $F(x_{p+1}, y_p + \frac{1}{2}) > 0$ mid pt. lies

below the line
∴ choose pt. NE

Now we choose amongst
(x_{p+2}, y_{p+1}) and (x_{p+2}, y_{p+2})

↓

Midpt. → $(x_{p+2}, y_p + 3/2)$

deciding factor → $F(x_{p+2}, y_p + \frac{3}{2})$
(d_{new})

$= a(x_{p+2}) + b(y_p + \frac{3}{2}) + c$ — (3)

(2) Write d_{new} in terms of d_{old}

↓

(3) can be rewritten as.

$d_n = a(x_p + 1 + 1) + b(y_p + 1 + \frac{1}{2}) + c$

$= a((x_p + 1) + 1) + b(y_p + \frac{1}{2} + 1) + c$

$= \boxed{a(x_p + 1) + b(y_p + \frac{1}{2}) + c} + a + b$

↓ d_{old}

$\{ d_{new} = d_{old} + \underbrace{a + b}_{\Delta NE} \}$

If we choose pt. NE, then we add an increment of ΔNE to obtain next pt. Iteration

CLASSTIME | Page No.
Date / /

Case II - Value is < 0.

Initial $d_{value} = F(x_p + 1, y_p + \frac{1}{2}) < 0$

Choose E $(x_p + 1, y_p)$

↓

Now we have choice of 2 pts.

$(x_p + 2, y_p)$ and $(x_p + 2, y_p + 1)$

midpt → $(x_p + 2, y_p + 1/2)$

d_{new}

$d_{new} = F(x_p + 2, y_p + 1/2)$

$= a(x_p + 2) + b(y_p + \frac{1}{2}) + c$

$= a(x_p + 1) + b(y_p + \frac{1}{2}) + c + a$

$d_{new} = d_{old} + (a) \rightarrow \underline{\Delta E}$

NOTE:- First d value is calculated by 1st pt. on line given by user.

↓

$d_0 = F(x_0 + 1, y_0 + \frac{1}{2})$

$= a(x_0 + 1) + b(y_0 + \frac{1}{2}) + (c)$

$= \underbrace{a x_0 + b y_0 + c} + a + b/2$

$F(x_0, y_0) = 0$ ← pt. lies on line.

Since it is not known, we need to remove it

$\therefore d_0 = a + \frac{b}{2}$

On this value, decide whether pt. E or NE is chosen

If NE is chosen

$d_0 = a + \frac{b}{2} + \Delta NE$

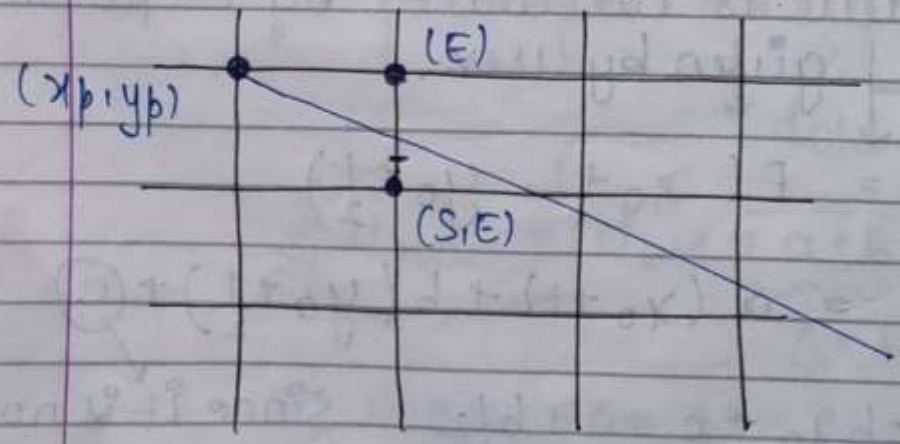
If E is chosen

$d_0 = a + \frac{b}{2} + \Delta E$

→ We follow these steps until (x_n, y_n) last pt. of line.

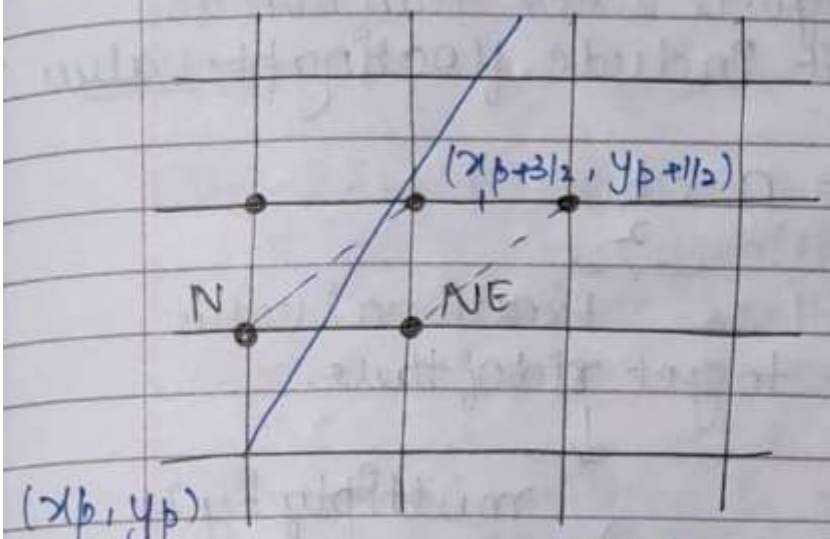
NOTE:- On screen, the lines / will be appearing as \ as origin is from top left corner.

- For -ve slope side ($m < -1$)



Initial value ↓ $(x_0 + 1)$ and $(y_0 - \frac{1}{2})$ instead of $y_0 + \frac{1}{2}$

→ Slope of line is $\underline{m > 1}$ (steep line) {case 3}



here we increment y and find values of x

coordinates changed.

Known (x_p, y_p)

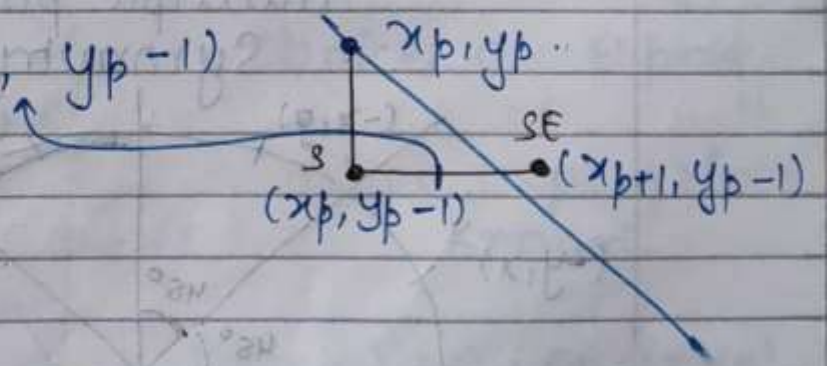
we have 2 choices N and NE

deciding factor → (x_p+1, y_p+1) {interchange $\underline{x \& y}$}

choices are either
North point (x_p, y_p+1)
or North East point (x_p+1, y_p+1)

mid pt. → $(x_p+1/2, y_p+1)$

→ $\underline{\text{Case 4}}$:- $m > -1$ b/w $0 \underline{\geq m \geq} -1$



CLASSTIME | Page No.
Date / /

NOTE: Line drawing algo. is better than DDA as it doesn't include floating pt. value

$$d_0 = a + \frac{b}{2}$$

Fractional value

to get rid of these.

↓

multiply by 2

$$(d_0 = 2a + b)$$

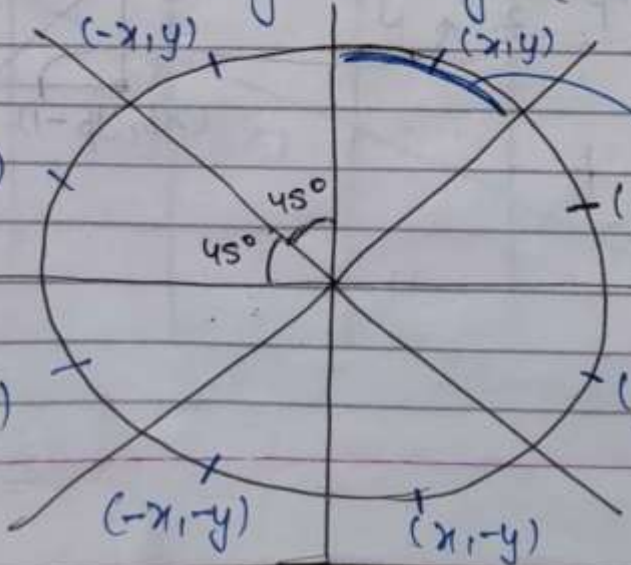
$F(x,y) = 0$
$2F(x,y) = 0$
⋮
$nF(x,y) = 0$ → no effect of multiplied constant.

$\Delta E \rightarrow a$ Now ΔE becomes $2a$
$\Delta NE \rightarrow a+b$ Now $\Delta NE \rightarrow 2(a+b)$

- Circle Drawing Algorithm-

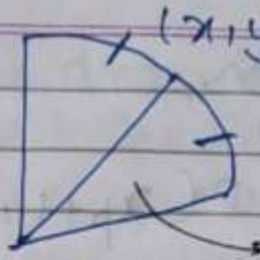
→ has curves & is symmetrical (advantage over line)

i.e If we know one pt., we can find multiple pts. using 8 way Symmetry (octants).



If we find this value of octant, then we can find other octants

{ xy was known }



Since it is at 45° this becomes mirror image of (x,y)

We can feed all the 8 octants in a function instead of plotting 1 pt. of octant

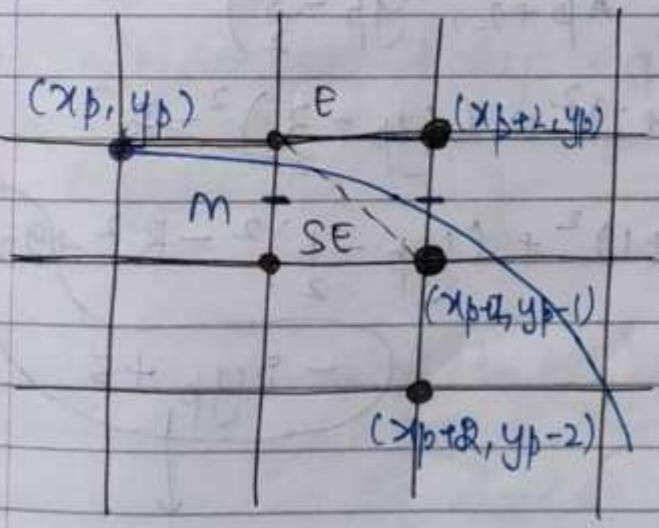
Circle algorithm is given by Bresenham's & it is known as. { uses same concept }

- Bresenham's Circle Drawing algo. / mid pt. circle drawing algo

→ eqn of circle → $x^2+y^2=R^2$
{ Assuming centre of circle is origin (0,0) }

→ If centre (h,k)

$$(x-h)^2+(y-k)^2=R^2.$$



choices are E and SE when (xp, yp) is known.

F(x,y) = 0 (on circle)
F(x,y) > 0 (above circle)
F(x,y) < 0 (below circle)

→ Deciding factor.

$$d_{old} = F(x_{p+1}, y_p - \frac{1}{2})$$

If < 0 ↓ choose pixel E

$E \rightarrow (x_{p+1}, y_p)$

$$d_{new} = F(x_{p+2}, y_p - \frac{1}{2})$$

↓

$$= (x_p+2)^2 + (y_p - \frac{1}{2})^2 - R^2$$

$$= (x_p+1)^2 + (y_p - \frac{1}{2})^2 - R^2 + 2x_p + 3$$

ΔE

→ If $d > 0$

↓ (outside circle)

choose SE. $(x_{p+1}, y_p - 1)$

as line is below mid pt. thus closer to SE

$$d_{new} = F(x_{p+2}, y_p - \frac{3}{2})$$

$$= (x_p+2)^2 + (y_p - \frac{3}{2})^2 - R^2$$

$$= (x_p+1)^2 + (y_p - \frac{1}{2})^2 - R^2 + 2x_p - 2y_p + 5$$

ΔSE

NOTE:- In case of line drawing algos, increments are constant i.e independent of x_p and y_p but in circle, every circle is dependent on values of x_p and y_p.
We need to calculate increments of circle at every pt.

initial pt. $\longrightarrow (0, r)$ → as radius is known.

∴ First initial d value

$$d_0 = f(x_0+1, y_0-\tfrac{1}{2})$$

$$= f(1, r-\tfrac{1}{2})$$

$$= (1)^2 + (r-\tfrac{1}{2})^2 - r^2$$

d_0 can be $=0$, <0, >0 ← $d_0 = \frac{5}{4} - r$

again a floating pt. value.

Soln :-

(1) either multiply eqn by 4.

(2) (Better soln) ↓

$$d_0 = \frac{5}{4} - r$$

$$h \leftarrow \left(d_0 - \frac{1}{4}\right) = 1 - r$$

{ subtracting $\frac{1}{4}$ from both sides }

If $d_0 < 0 \Longrightarrow h < 0$

$d_0 > 0 \Longrightarrow h > 0$

$d_0 - \frac{1}{4} < 0$

We are comparing d_0 by 0
∴ This new value can be compared by $\frac{1}{4}$

effective value of $\frac{1}{4}$ becomes 0 only

But value b/w 0 and $\frac{1}{4}$ can't come as we are using integral values

∴ We can use $(1-r)$ and safely compare d_0 with 0.

∴ $\boxed{d_0 = 1 - r}$ → now d_0 can't get any value of $\frac{1}{4}$ as 1 and r are both integers.

→ Now we will write a func. to plot 8 pixels.

```
func()
{
    // call putpixel 8 times
    with diff. coordinates.
}
```

→ On screen, we will be able to see only 1 quadrant as origin is top left corner

CLASSTIME | Page No.
Date / /

∴ We need to take radius and centre from user & shift origin to centre

↓

$(x-h)^2 + (y-k)^2 = r^2$

$\{ \because x = X+h \}$
$\{ \quad\quad y = Y+h \}$.

15 Jan 19

• Practical

→ How to do CG programs in DevC++ :-

1. Copy graphics.h and winbgim.h to C:\Program Files(X86)\Dev-CPP\MinGW64\include

2. Copy libbgi.a to C:\Program Files(X86)\Dev-Cpp\MinGW64\lib.

3. Copy 6-ConsoleAppGraphics.template and Console App-cpp-graph.txt to C:\Program Files(X86)\Dev-Cpp\Templates.

4. From compiler selection dropdown choose TDM.GCC X.X.X 32bit Release.

↓

5. Done.

Select 32bit Release.

File → New → Project → Graphics Console → OK → make a folder → Project → Project options → Parameters → Linker window

- lbgi - lgdi32 - luser32 (by default included)
- lbgi
- lgdi32
- lcomdlg32
- luuid
- loleaut32
- lole32