14/3/22

[Computer Graphics]

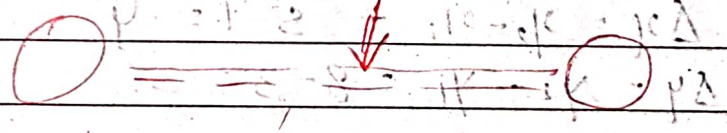
Date
Page No.

Computer graphics is used where a set of images needs to be manipulated or the creation of the image in the form of pixels and is drawn on the computer.

1) Creating object
2) Storing the object
3) Manipulation of still object

Graphics
- Interactive Graphics → where user can interact
  Ex- Game
- Passive Graphics → Here no user interaction
  Ex- cartoon.

memory is known as frame buffer



Info about object

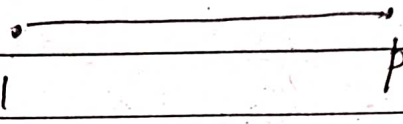
Applications →

1. Game playing
2. Marketing
3. Advertisement
4. Medical diagnosis
5. Business
6. Education
7. Reports
8. Air craft (learning purpose)
9. Architecture.

Date
Page No.

DDA (Digital Differential Analyzer) :-
Line generation Algorithm

→ An incremental Algorithm :-



→ Consider one point of line as $P_1(x_1, y_1)$ and the second point $P_2(x_2, y_2)$.

Calculate Δx, Δy = ?

$P_1(x_1, y_1) \rightarrow (1, 3)$ $\quad P_2(x_2, y_2) \rightarrow (5, 8)$

$\Delta x = x_2 - x_1 = 5 - 1 = 4$

$\Delta y = y_2 - y_1 = 8 - 3 = 5$

Slope $\Rightarrow m = \frac{y_2 - y_1}{x_2 - x_1} = m = \frac{\Delta y}{\Delta x} = 5/4$

→ whose value will be greater will be increase more as compare to the another one - just like the m will increase more

→ slope gives us an idea about which value increase more.

Step 1 → Calculate slope m.

If $|m| < 1$

$x = x + 1$

$y = y + m$

II nd condition

If $|m| > 1$

$y = y + 1$

$x = x + 1/m$

III rd condition

If $m = 1$

$x = x + 1$

$y = y + 1$

I st condition

Ex) Draw a line by using DDA from (2, 1) to (8, 5)
(x_1, y_1) (x_2, y_2)

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

$= \frac{5-1}{8-2} = 4/6 = 2/3 = 0.6667$

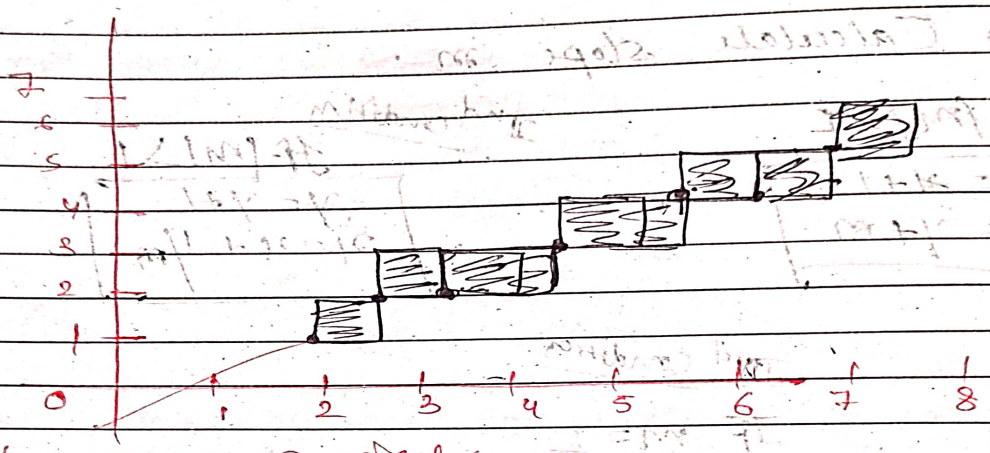
So $|m| < 1$ condition 1

$x = x + 1$

$y = y + m$

	x	y	R(y)
1	2	1	1
2	3	1.6667 →	2
3	4	2.3334 →	2
4	5	3.0001 →	3
5	6	3.6668 →	4
6	7	4.3334 →	4
7	8	5. →	5

Plotting :-



Lower corner of pixel :-

IInd condition (x1 (3,2) to (7,8))
x1 y1 x2 y2

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8-2}{7-3} = 6/4 = 3/2$$

1/m = 0.6667

m > 1

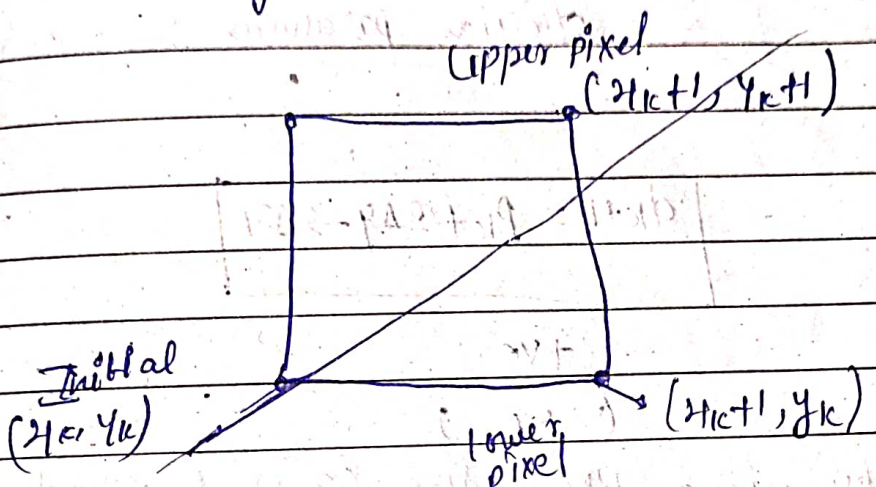
y2 y+1
x2 x+1/m

P(x)	x	y
3	3	2
4	3.6667	3
4	4.3334	4
5	5.0000	5
6	5.6668	6
6	6.3335	7
7	7.0002	8

15/03/23

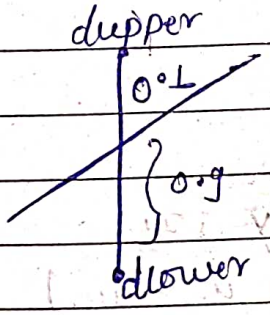
⁂ Bresenham's line generation Algorithm :-

→ It makes in 1st corner
→ x is always increase



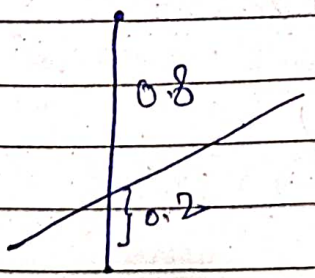
⇒ dlower - dupper = +ve (≥0) Select upper pixel
⁂ -ve (<0) Select lower pixel.

Eg:-



dlower - dupper = 0.9 - 0.1 = 0.8
= 0.8 > 0 (Select upper pixel)

Eg:)



= dlower-dupper = 0.2 - 0.8 = -0.6
= -ve (Select lower pixel)

Formula:- Initial Parameter $P_k = 2\Delta y - \Delta x$.
It provide 2 conditions,

(a) ≤ 0 than $x_k = x_{k+1}$

$y_k = y_{k+1}$

decision parameter

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

−ve (NC)

(b) ≥ 0 than $x_k = x_{k+1}$

$y_k = y_{k+1}$

decision parameter

$d_{k+1} = p_k + 2\Delta y - 2\Delta x$

+ve (+1)

Q) Drawing a line by using BLG Algo (2,1) to (8,5).

Sol) $\Delta x = 8-2 = 6$ $\Delta y = 5-1 = 4$.

$2\Delta x = 12$ $2\Delta y = 8$

$2\Delta y - 2\Delta x = 8-12 = -4$

	x	y	p_k initially $2\Delta y - \Delta x$
initially	2	1	8 − 6 = 2 (+ve) (y is increm)
	3	2	2 + (−4) = −2 (−ve) y(NC)
	4	2	−2 + 8 = 6 (+ve) (y+1)
	5	3	6 − 4 = 2 (+ve) (y+1)
	6	4	2 − 4 = −2 (−ve) (NC)
	7	4	−2 + 8 = 6 (+ve) (y+1)
	8.	5	

16/03/23

Derivation :-

$d_1 = y - y_k = m(x_k+1) + c - y_k$

$d_u = y_{k+1} - y = y_{k+1} - [m(x_k+1) + c]$

$y = mx + c$ — eq comman.

$y = m(x_k+1) + c$

$= d_1 - d_u$

$= m(x_k+1) + c - y_k - y_{k-1} + m(x_k+1) + c$

$= 2m(x_k+1) - 2y_k + 2c - 1$

$= 2\frac{\Delta y}{\Delta x}(x_k+1) - 2y_k + 2c - 1$

multipling Δx both side.

$\Delta x(d_1 - d_u) = 2\cdot\Delta x\cdot\frac{\Delta y}{\Delta x}(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$

$\Rightarrow \Delta x(d_1 - d_u) = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$

decision parameter $d_k = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$

$d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x c - \Delta x$ — (1)

$2\Delta y$: constant

we write the three constant as b.

$= 2\Delta y x_k - 2\Delta x y_k + b$, for all three constant

$\therefore d_k = 2\Delta y x_k - 2\Delta x y_k + b$ — (2)

$\therefore d_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + b$ — (3).

$\therefore d_{k+1} - d_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + b$
$- 2\Delta y x_k + 2\Delta x y_k - b$

$= 2\Delta y x_{k+1} - 2\Delta x y_{k+1} - 2\Delta y x_k + 2\Delta x y_k$

$\therefore d_{k+1} - d_k = 2\Delta y (x_{k+1} - x_k) - 2\Delta x (y_{k+1} - y_k)$

$= 2\Delta y - 2\Delta x (y_{k+1} - y_k)$

$(y_k - y_k)$ — for (2)

0.

$d_{k+1} - d_k = 2\Delta y$

$\boxed{d_{k+1} = d_k + 2\Delta y}$ ← Formula

$\therefore d_{k+1} - d_k = 2\Delta y - 2\Delta x (y_{k+1} - y_k)$

$\therefore d_{k+1} - d_k = 2\Delta y - 2\Delta x$.

$\boxed{d_{k+1} = 2\Delta y - 2\Delta x + d_k}$ ← Formula.

eq no. (1)

$\therefore d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x_k - \Delta x$
(replace this)

$\therefore 2\Delta y x_k + 2\Delta y - 2\Delta x y_k - 2\Delta x (y_k - m x_k) - \Delta x$

$\therefore 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta x \frac{\Delta y}{\Delta x} x_k - \Delta x$

$\boxed{d_k = 2\Delta y - \Delta x}$

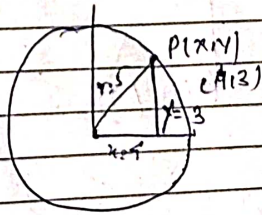
9.3 Circle generation in Computer Graphics

eq. of circle :—

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

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

any points on the circle



$= 4^2 + 3^2 - 5^2$

$= 16 + 9 - 25 = 0$ → If it comes '0' then the point lies on the circle, x and y both lies on the circle.

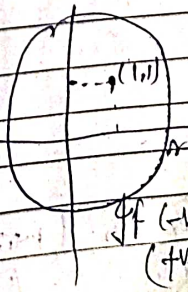
Eg: Point not on circle.

$= x^2 + y^2 - r^2$

$= 1^2 + 1^2 - 5^2 = 0$

$= 2 - 25 = 0$

$= -23 \neq 0$.

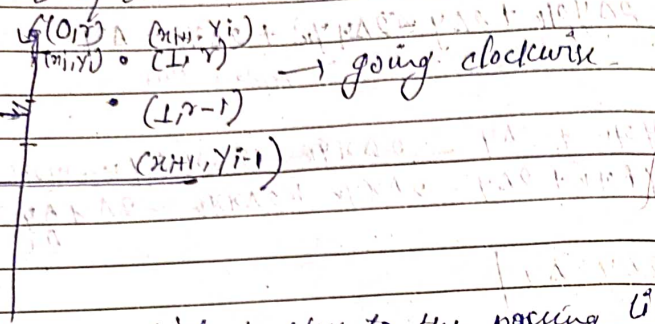


If this ≠ 0 then its not lies on the circle

If (-ve) Inside of circle

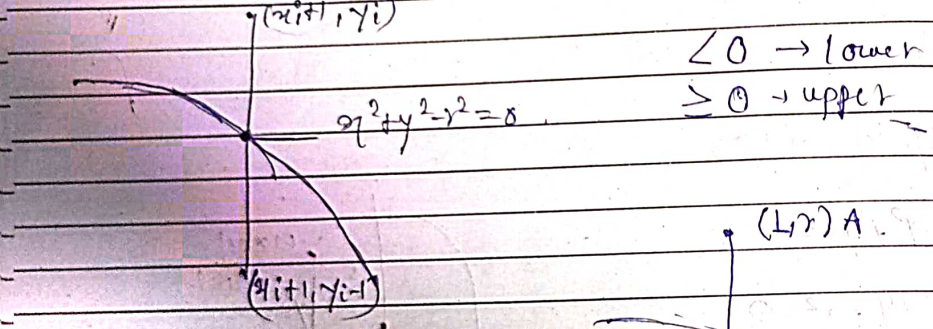
(+ve) Outside

= Generation → only we know radius r.

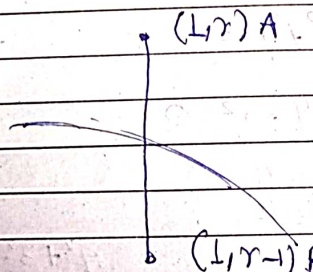


→ going clockwise

→ Choosing the one which is close to the passing line.



< 0 → lower
≥ 0 → upper



(A) A(1, r)

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

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

$= 1 \neq 0$

+ve > 0

(B) B(1, r-1)

$x^2+y^2-r^2$

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

$1 + r^2 + 1 - 2r - r^2 = 0 \quad (1-r \neq 0)$

$= 2 - 2r \neq 0$ (-ve)

Now adding both value

$1 - r + 1$

$= 2 - r$

Date
Page No.

21/03/23 Circle generation →

Bresenham's algorithm →

If any point inside circle.

$x^2 + y^2 - r^2 < 0$

(-ve)

If any point outside circle.

$x^2 + y^2 - r^2 > 0$

(+ve).

⇒ Suppose r = 10

(1) $d_i = 3 - 2r$ → will get value < 0, ≥ 0

(2) $d_i < 0$, y_i (lower pixel) (no change) (-ve) → $d_{i+1} = d_i + 4x_i + 6$

by bresenham parameter

x_{i+1}

(3) $d_i \geq 0$, $y_i - 1$ (upper pixel) (change) (+ve) → $d_{i+1} = d_i + 4(x_i - y_i) + 10$

x_{i+1}

r = 10

x_i	y_i	d_i	x_{i+1}	y_{i+1}
0	10	-17	1	10
1	10	-11	2	10
2	10	-1	3	10
3	10	13	4	9
4	9	-5	5	9
5	9	17	6	8
6	8	11	7	7

∴ initial.

$d_i = 3 - 2r$

$= 3 - 2 \times 10$

$= 3 - 20 = -17$

-ve $= d_i + 4x_i + 6$

$= -17 + 4(0) + 6$

$= -17 + 6 = -11$

-ve = di + 4xi + 6

= -1 + 4(1) + 6

= -1 + 4 + 6

= -1 + 10 = -1.

Stopping condition when $x_i = y_i$ or $x_i > y_i$

-ve = di + 4xi + 6

= -1 + 4(2) + 6

= -1 + 8 + 6

= -1 + 14 = 13

+ve = di + 4(xi - yi) + 10

= 13 + 4(3 - 10) + 10

= 13 + 4(-7) + 10

= 13 - 28 + 10

= 13 - 18

= -5

-ve = di + 4xi + 6

= -5 + 4(4) + 6

= -5 + 16 + 6

= 1 + 16

= 17.

+ve = di + 4(xi - yi) + 10

= 17 + 4(5 - 4) + 10

= 17 + 4 X (-4) + 10

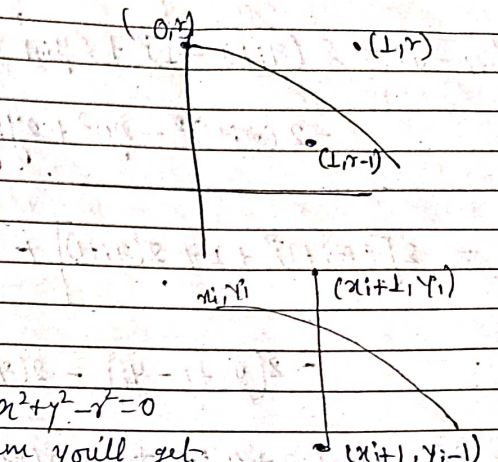
= 17 - 16 + 10

= 27 - 16

= 11

Derivation :-

1. 3 - 2r
2. di + 4xi + 6
3. di + 4(xi - yi) + 10



for 3 - 2r

put (1,r) and (1,r-1)

in eq. of circle $x^2 + y^2 - r^2 = 0$

and then add them you'll get

$\Rightarrow 3 - 2r$

~~for (xi+1, yi)~~

Putting in circle eq.

$(x_i+1)^2 + (y_i)^2 - r^2 = 0$

$= x_i^2 + 1 + 2x_i + y_i^2 - r^2 = 0$

~~x_i^2~~

$= (x_i+1)^2 + (y_i)^2 - r^2 + (x_i+1)^2 + (y_i-1)^2 - r^2$

$di = (x_i+1)^2 + y_i^2 - r^2 + (x_i+1)^2 + (y_i-1)^2 - r^2$

$= 2(x_i+1)^2 + y_i^2 + y_i^2 + 1 - 2y_i - 2r^2$

$di = 2(x_i+1)^2 + 2y_i^2 - 2y_i - 2r^2 + 1.$ (i की जगह i+1)

$di+1 = 2(x_{i+1}+1)^2 + 2y_{i+1}^2 - 2y_{i+1} - 2r^2 + 1$

$= d_{i+1} - d_i = 2(x_{i+1}+1)^2 + 2y_{i+1}^2 - 2y_{i+1} - 2x^2 + 2$

$-2(x_i+1)^2 - 2y_i^2 + 2y_i + 2x^2 - 1$

$= 2[(x_i+1)^2 + 1 + 2(x_i+1)] + 2(y_{i+1}^2 - y_i^2)$

$- 2(y_{i+1} - y_i) - 2(x_i+1)^2$

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

$- 2(y_{i+1} - y_i)$

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

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

y_{i+1}^2 y_i^2 y_{i+1} y_i

$\leq 0 \longrightarrow$

$= 4x_i + 6$

$\therefore \boxed{d_{i+1} = 4x_i + 6 + d_i}$

$y_i = y_{i-1}$

≥ 0

$= 4x_i + 6 + 2[(y_{i-1} - y_i)^2] - 2(y_{i-1} - y_i)$

$= 4x_i + 6 + 2[y_i^2 + 1 - 2y_i - y_i^2] + 2$

$= 4x_i + 6 + 2 - 4y_i + 2$

$\boxed{d_{i+1} = 4(x_i - y_i) + 10 + d_i}$

Mid-point circle gen :-

If mid point +ve choose below
if -ve choose above.

(0,r)

(1,r)

(1, r-1/2)

(1, r-1)

Skip the print value →

→ Derive the Initial DP for mid point circle generation

$\simeq 1 - r$

23/03/23

Mid-point circle generation:-

Let Given a circle radius $r = 10$. we demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant From $x=0$ to $x=y$. the initial value of the decision parameter is :-

$$p_0 = 1-r$$
$$= 1-10 = -9$$

(Only one time 1-r will be calculated)

$\Rightarrow d_i + 2x_{i+1} + 1 < 0$ (y_i)

$\Rightarrow d_i + 2x_{i+1} - 2y_{i+1} + 1 \geq 0$ ($y_i - 1$)

Successive decision parameter values and positions along the circle path are calculated using the midpoint method as :-

(0, 10)

d_i	x_{i+1}, y_{i+1}	$(2x_{i+1}, 2y_{i+1})$	
			$d_i + 2x_{i+1} + 1$
$-9 < 0$	(1, 10)	(2, 20)	$= -9 + 2 + 1$
			$= -9 + 3 = -6$
$-6 < 0$	(2, 10)	(4, 20)	$= -6 + 4 + 1$
$-1 < 0$	(3, 10)	(6, 20)	$= -6 + 5 = -1$
$6 \geq 0$	(4, 9)	(8, 18)	$= -1 + 6 + 1$
			$= 6$
$-3 < 0$	(5, 9)	(10, 18)	$d_i + 2x_{i+1} - 2y_{i+1} + 1$
$8 \geq 0$	(6, 8)	(12, 16)	$= 6 + 8 - 18 + 1$
$5 > 0$	(7, 7)	(14, 14)	$= 7 - 10 = -3$

Now will stop at diagonal

Mathematical expression: (10 marks)

Date
Page No.

* Derivation:

$x^2 + y^2 - r^2$

$(x_{i+1}, y_i - 1/2)$ — mid point coordination.

$d_i = (x_{i+1})^2 + (y_i - 1/2)^2 - r^2$ — (2)

(Square the i subscript)

(in d_i subscript in (i) form, in d_{i+1} they are in $i+1$ form.)

$d_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1/2)^2 - r^2$ — (1)

Now (1) - (2)

$d_{i+1} - d_i = (x_{i+1})^2 - (y_i - 1/2)^2 + x^2$

$(x_{i+1} + 1)^2 + (y_{i+1} - 1/2)^2 - x^2$

$= (\overbrace{x_{i+1}}^{a} + \overbrace{1}^{b})^2 + (y_{i+1} - 1/2)^2 - (x_{i+1})^2 - (y_i - 1/2)^2$

$= (x_{i+1})^2 + 1 + 2(x_{i+1}) + y_{i+1}^2 + 1/4 - 2y_{i+1}$

$- (x_{i+1})^2 - y_i^2 - 1/4 + 2y_i$

$= 1 + 2(x_{i+1}) + y_{i+1}^2 - y_i^2 - (y_{i+1} - y_i)$

= Now put less than condition.

$= < 0$ y_i

$y_{i+1} = y_i$

$= 2(x_i+1) + y_i^2 - y_i^2 - (y_i - y_i) + 1$

$d_{i+1} - d_i = 2(x_i+1) + 1$

$d_{i+1} = d_i + 2(x_i+1) + 1$

$d_{i+1} = d_i + 2(x_i+1) + 1$

$\geq 0 \quad y_i - 1$

$= 2(x_i+1) + (y_i-1)^2 - y_i^2 - [(y_i-1) - y_i)] + 1$

$= 2(x_i+1) + y_i^2 + 1 - 2y_i - y_i^2 + 1 + 1$

$= 2(x_i+1) - 2y_i + 2 + 1$

$= 2(x_i+1) - 2(y_i - 1) + 1$

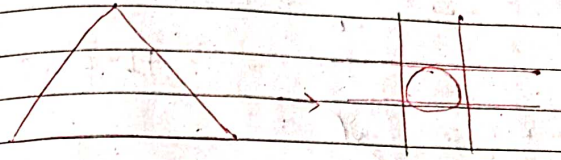
$= 2(x_i+1) - 2y_{i+1} + 1$

☆ Difference b/w Random / Raster :- (2 marks)

Random	Raster
☆ Better Quality	☆ Relatively less
☆ less space	more space.
☆ 256	16 million colour
☆ Unrealistic	Realistic
☆ Random point-	From pixel
-> Based on equation	-> Based on pixel point by point
☆ logo	.Jpg

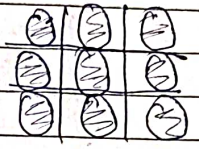
23/03/23

☆ Polygon filling Algorithm:

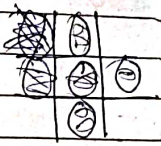


→ 8 connected or 4 connected

= 8 connected



4 connected



⇒ Seed pixel / Starting pixel:-

For finding this their are 2 rules.

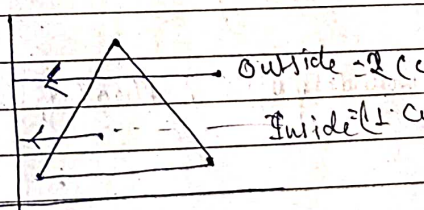
(i) Inside – Outside rule.

(ii) Non-zero winding rule.

(i) Inside - outside rule :- Select only one direction (left/right)

for even cutting points → outside.

odd cutting points → inside.



(ii) Non-zero winding rule :-

→ Choose one direction.

Slope :- (a) increasing

(b) decreasing

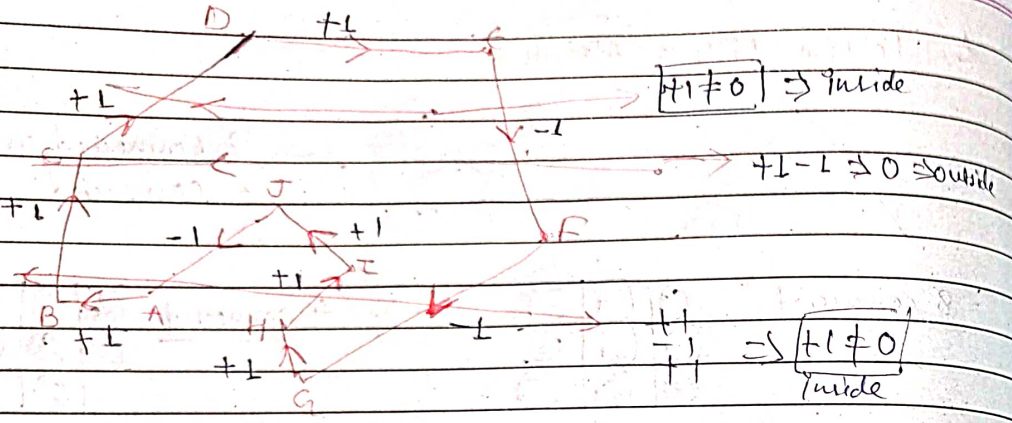
let we go upper direction

up - increase (+)

down - decrease (-)

Non-zero → inside

Zero = outside



☆ How to color :-

8 Connected :-

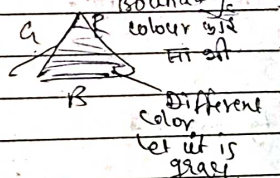
P(x-1, y+1)	P(x, y+1)	P(x+1, y+1)
P(x-1, y)	P(x, y)	P(x+1, y)
P(x-1, y-1)	P(x, y-1)	P(x+1, y-1)

= Process of Filling → Boundary algo, Flood filling algo.

☆ Boundary filling algo :- for 4 connected

```
Void BF (int x, int y, int fill, int bound)   → For boundary colour
{                                               → continue until boundary color ≠ bound
 If (get pixel (x,y) != bound
     && (get pixel (x,y) != fill)
 {
  Putpixel (x,y, fill)
  BF (x+1, y, fill, bound)
  BF (x-1, y, fill, bound)        } 4-connected
  BF (x, y+1, fill bound)
  BF (x, y-1, fill, bound)
 }
}
```

→ connected 8 ate... 4 connect extra

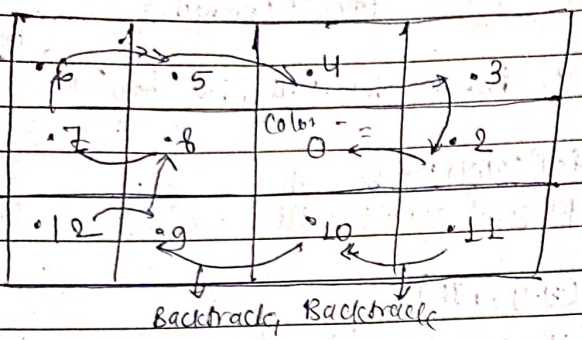


☆ Flood Filling :- Initially

→ old color it is always unique

= let old color → gray, new

```
Void FF (int x, int y, int fill, int old)
{ If (get pixel (x,y) == old &&
    { put pixel (x,y, fill)
      FF (x+1, y, fill, old)
      FF (x-1, y, fill, old)      } 4 connected
      FF (x, y+1, fill, old)
      FF (x, y-1, fill, old)
} }
```



4 connect

here Boundary filling is (recursive call)

(R) x+1, y

(L) x-1, y

(TOP) x, y+1

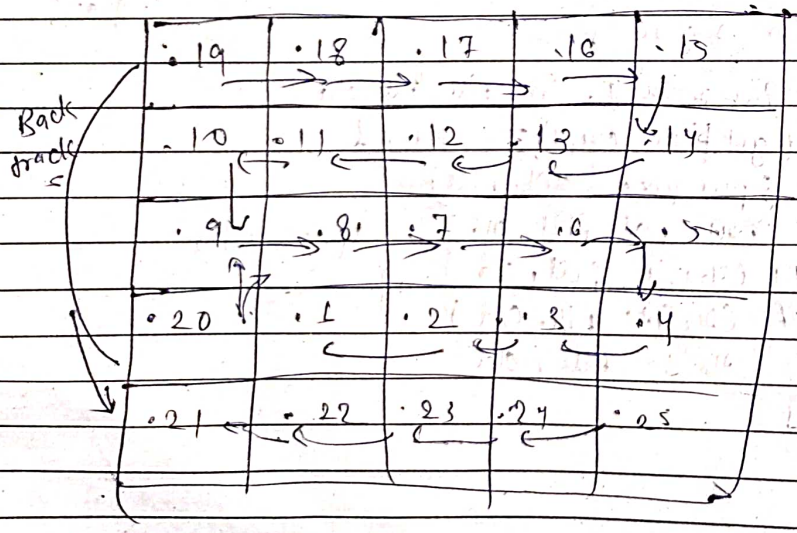
(Bottom) x, y-1

New pixel will do all process

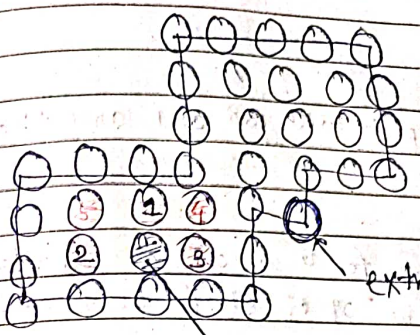
1st check → R - L - T - B

here 4 - R → L → T - B is complete so backtrack to 10

eg - 2 marks



30/03/23



→ Boundary is already colored. we have to colour the middle one.

→ using stack method

Seed pixel → from where we have to start colouring

↓

Can choose randomly.

4 connected method :

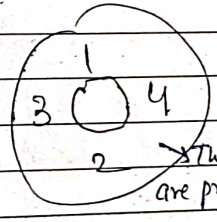
Order

Top - T

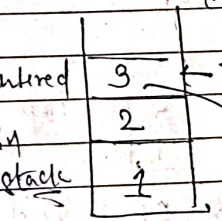
Bottom - B

left - L

Right - R



(Pop means coloured)



⇒ Only 4 times we have to check for each pixel.

⇒ Now we will check which pixel is at the top of stack.

⇒ First PUSH in stack

↳ we'll colour while popping

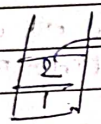
⇒ 3 will be coloured after seed pixel.

⇒ Now will check T, B, L, R for the new coloured pixel which is 3, and put the member in stack

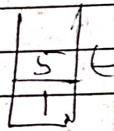
Pop ← 4 ← TOP
2
1

= Now since 4 is in top it will pop out and then coloured.

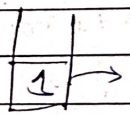
= After this will see T,B,L,R for 4 which are already occupied so will see the TOP of stack and colour it



=> Now 2 will be popped and coloured and will see it TBLR.

= 

. Now pop-5 and colour it and push the 4 connected pixel if not already in stack



= Now after popping 1 stack is basic now we can't process because stack is empty.

= 5 pixels are processed in 4 connected.

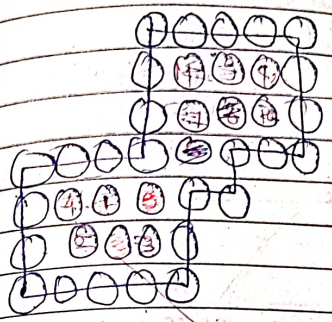
=> Drawback:-

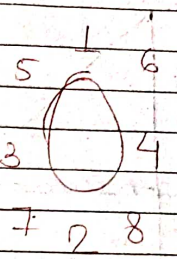
→ In diagonal pixel won't be coloured which will left other pixels uncoloured

→ It can't cross the diagonal pixel.

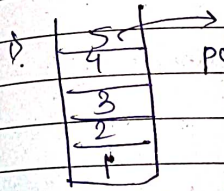
Sequence is important

= 8 connected:-

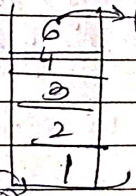




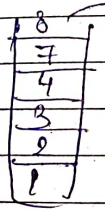
Seed pixel same process just we have to colour 8 pixels.



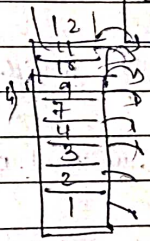
pop 5 and colour



Pop and colour & check 8 connected



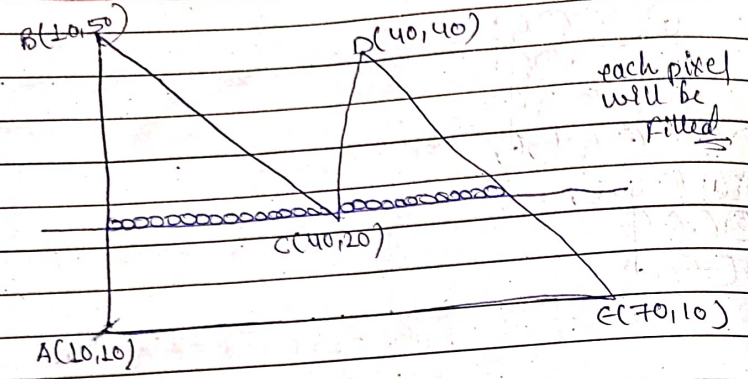
pop & colour



=> Now pop all and colour.

05/04/23

Scanline Polygon filling :-



- First find out y → minimum value y
  → will have to see y how its increasing and where intersected
  → will stop at the maximum value

Step (1) Write all edges which are not horizontal

(AB, BC, CD, DE) → vertical or slant edge

Step (2) Find (y_{min}) of each edge.

- AB = 10
  BC = 20
  CD = 20
  DE = 10

Their are different y value

Step-3) Now we will find out edge for different y values.

y_{max}	x_{max}	slope

AB

50	10	0

DE

40	70	-1

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

$= \frac{50-10}{10-10} = \frac{40}{0} = 0$

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

$= \frac{40-10}{40-70} = \frac{30}{-30} = -1$

Step-4) Draw/Fill colour pixel from 10 to 70.

Step-5) Increase y

- y = 10
- Now y = 11

AB

50	10	0

50	10	0

DE

40	69	-1

40	68	-1

y = 20

Note : y value is increased.
y value is decreased.

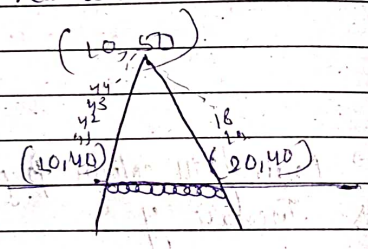
AB			BC			CD			DE		
50	10	0	50	40	-1	40	40	0	40	60	-1
				33						59	
				32						58	
				31							

= AB and CD will remain same because slope is 0 but BC and DE will decrease by -1 in each step because slope is -1.

= Now After $y = 40$.
Remove the edge whose max y is reached.

AB ✓ BC ✓ CD ✗ DE ✗

max = y



AB			BC		
50	10	0	50	20	-1

Algo → (1) Find all y_{min} of every edge.
(2) Initially Active edge list is empty.

(3) Repeat → for each y_{min}.
1) Store all cutting edges.
2) Sort on x_1 value.
3) Fill pixel from x_1 to x_2
4) If y max reached remove the edge

(4) $y = y+1$;
$x = x + 1/m$ → 10+0 = 10
10+(-1) = 9

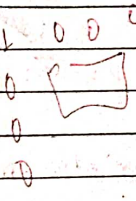
10/04/23

3D Transformation

(1) Translation
(2) Scaling
(3) Rotation
(4) Reflection
(5) Shearing

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

= 3D is same just z cordinate extra.

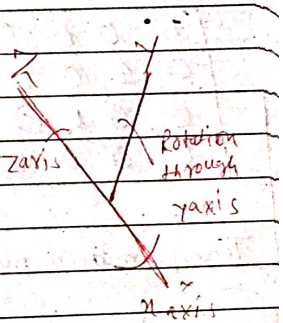


(1) $$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$x' = x + t_x$
$y' = y + t_y$
$z' = z + t_z$

(2) Scaling → $$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$x' = x \cdot S_x$
$y' = y \cdot S_y$
$z' = z \cdot S_z$



(3) Rotation → z axis

$$\begin{array}{c} \\ x \\ y \\ z \end{array} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

x axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$