Computer Graphics

It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify represent pictures. The end product of computer graphics is a picture it may be a business graph, drawing and engineering

Types :- 1) Passive or Non Interactive Computer Graphics

In passive computer graphics, the picture is produced on the monitor and user does not have any control over the image. It involves one way communication between the computer and the user.

Ex Titles shown on TV.

Interactive Computer Graphics.

In interactive computer graphics user have some over the picture. It requires two way communication between computer and user. A user can see an image and make changes by sending his command with an input device

Ex Games like ping-pong.

2 marks.

Applications :- 1) Education and training
2) Computer generated maps
3) Presentation Graphics like bar charts, pie chart
4) Medical Diagnosis
5) Business Education
6) Flight Simulators.

Line Generation Algorithm

Digital Differential Analyzer (DDA)

Difference between two points

It is an incremental algorithm (means progressive) In this method calculation is performed at each step but by using previous steps.

If $P_1(x_1, y_1) \rightarrow (1, 3)$

$P_2(x_2, y_2) \rightarrow (5, 8)$

$\therefore \Delta x = x_2 - x_1 \qquad \Delta y = y_2 - y_1$

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

Step 1 calculate slope m

Step 2 check condition

a) if $|m| < 1$ then $x = x + 1$ and $y = y + m$

b) if $|m| \geq 1$ then $x = x + \frac{1}{m}$ and $y = y + 1$

c) if $m = 1$ $x = x + 1$ and $y = y + 1$

Eg 1) Draw line by using DDA from (2,1) to (8,5)

$x_1 = 2, y_1 = 1, x_2 = 8, y_2 = 5$

$\Delta x = x_2 - x_1 = 8 - 2 = 6$

$\Delta y = y_2 - y_1 = 5 - 1 = 4$

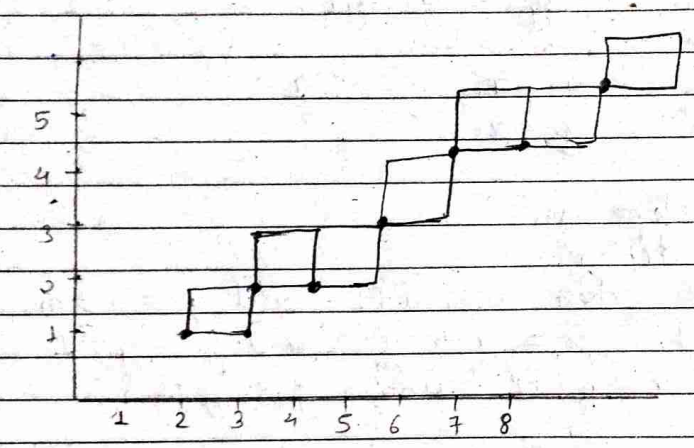
$$m = \frac{\Delta y}{\Delta x} = \frac{4}{6} = 0.6667$$

$m < 1$ ∴ using condition 1 $x = x+1$, $y = y+m$

x will increase until last point 8.

x	y	$R(y)$
2	1	1
3	1.6667	2
4	1.6667 + 0.6667 = 2.334	2
5	2.334 + 0.6667 = 3.001	3
6	3.001 + 0.6667 = 3.6668	4
7	3.6668 + 0.6667 = 4.3335	4
8	4.3335 + 0.6667 = 5.002	5

Here if .5 or >.5 upper value else take lower value



Eg 2 (3, 2) to (7, 8)

$x_1 = 3$, $x_2 = 7$, $y_1 = 2$, $y_2 = 8$

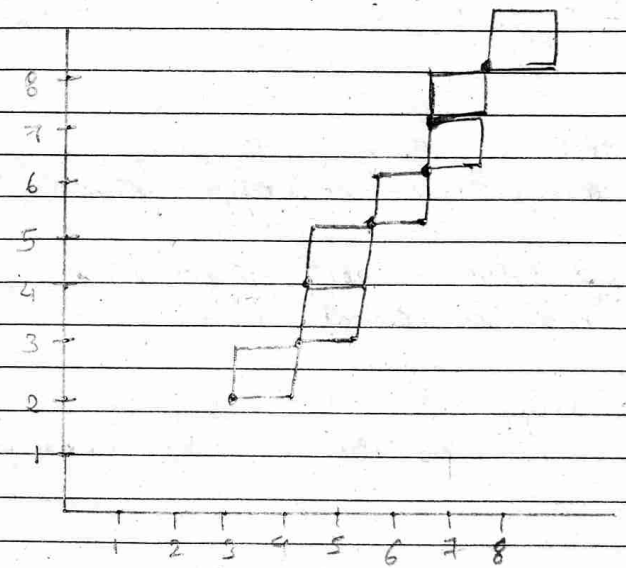
$\Delta x = x_2 - x_1 = 7 - 3 = 4$

$\Delta y = y_2 - y_1 = 8 - 2 = 6$

$m = \frac{\Delta y}{\Delta x} = \frac{6}{4} = 1.5$

$m > 1$ ∴ $y = y+1$, $x = x + \frac{1}{m}$, $\frac{1}{m} = \frac{1}{1.5} = 0.$

x	$R(x)$	y
3	3	2
3.6667	4	3
3.6667 + 0.6667	4	4
5.0001	5	5
5.6668	6	6
6.3335	6	7
7.0002	7	8



x	y
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19

Eg (12, 12) to (19, 19)

$\Delta x = 19 - 12 = 7$

$\Delta y = 19 - 12 = 7$

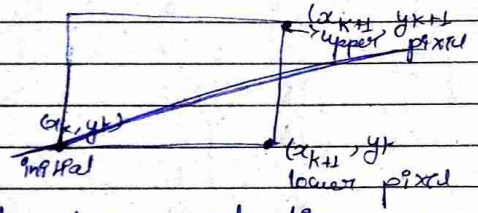
$m = \frac{\Delta y}{\Delta x} = \frac{7}{7} = 1$

$m = 1$ ∴ $x = x+1$, $y = y+1$

Bresenham's Line Algorithm

The algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, substraction and multiplication operations. In this method, next pixel selected is that one who has the least distance from true line.

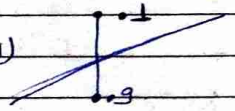
- The line is always in 1st quadrant



Rule for selecting among lower and upper pixel

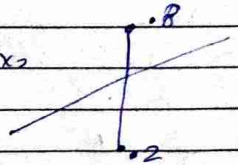
If $d_{lower} - d_{upper}$ is positive select upper pixel
If $d_{lower} - d_{upper}$ is negative select lower pixel

Ex1)



$d_{lower} - d_{upper} = 0.9 - 0.1 = 0.8$
positive ∴ select upper pixel

Ex2



$d_{lower} - d_{upper} = 0.2 - 0.8 = -0.6$
negative ∴ select lower pixel

Initial Decision parameter $= 2\Delta y - \Delta x$

If d_k is negative i.e. < 0
Then $x_k = x_k + 1$, $y_k = y_k$ (no change)

If P_k is positive i.e > 0
Then $x_k = x_{k+1}$, $y_k = y_k + 1$

∴ $d_{k+1} = d_k + 2\Delta y$ when negative

and $d_{k+1} = d_k + 2\Delta y - 2\Delta x$ when positive

Draw a line using BLA from (2,1) to (8,5)

$x_1 = 2$, $x_2 = 8$, $y_1 = 1$, $y_2 = 5$

$\Delta y = y_2 - y_1 = 5 - 1 = 4$

$\Delta x = x_2 - x_1 = 8 - 2 = 6$

$2\Delta y = 2 \times 4 = 8$, $2\Delta x = 2 \times 6 = 12$

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

d_0 (Initial parameter) $= 2\Delta y - \Delta x$
$= 8 - 6 = 2$ (positive)

∴ $d_1 = d_0 + 2\Delta y - 2\Delta x$
$= 2 + 8 - 12 = 2 - 4 = -2$ (negative)

∴ $d_2 = d_1 + 2\Delta y$
$= -2 + 8 = 6$ (positive)

∴ $d_3 = d_2 + 2\Delta y - 2\Delta x$
$= 6 - 4 = 2$ (positive)

∴ $d_4 = d_3 + 2\Delta y - 2\Delta x$
$= 2 - 4 = -2$ (negative)

∴ $d_5 = d_4 + 2\Delta y$
$= -2 + 8 = 6$ (positive)

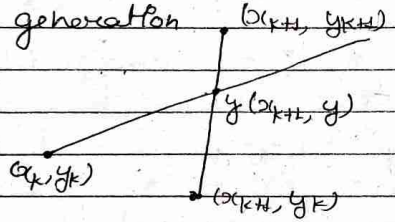
∴ $d_6 = d_5 + 2\Delta y - 2\Delta x$
$= 6 - 4 = 2$

→ 10 marks

x	y	d_k
2	1	initially $(2\Delta y - \Delta x)$ 8 − 6 = 2
3	2	2 + (−4) = −2
4	2	−2 + 8 = 6
5	3	6 + (−4) = 2
6	4	2 + (−4) = −2
7	4	−2 + 8 = 6
8	5	6 + (−4) = 2

10 marks

Derivation Bresenham's line generation



Eq^n of line $y = mx + c$

d_l (dlower) $= y - y_k$ — (I)

d_u (dupper) $= y_{k+1} - y$ — (II)

For eqn (I) $x = x_k + 1$

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

and $d_l = m(x_k + 1) + c - y_k$

For eqn (II)

$d_u = y_k + 1 - [m(x_k + 1) + c]$

$\therefore d_l - d_u = [m(x_k + 1) + c - y_k] - [y_k + 1 - [m(x_k + 1) + c]]$

$d_l - d_u = m(x_k + 1) + c - y_k - y_k - 1 + m(x_k + 1) + c$

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

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

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

here $\Delta x(d_l - d_u)$ = decision parameter d_k

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

let $2\Delta y + 2\Delta x c - \Delta x$ → constant (b)

$\therefore d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x c - \Delta x$

let $2\Delta y + 2\Delta x c - \Delta x$ → constant (b)

now $d_k = 2\Delta y x_k - 2\Delta x y_k + b$

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

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

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

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

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

Checking Condition

For lower pixel (−ve, < 0) Here $y_{k+1} = y_k$

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

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

For upper pixel (+ve, > 0) $y_{k+1} = y_k + 1$

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

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

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

For initial parameter

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

let $y_k = mx_k + c$

$$c = y_k - mx_k$$

putting value of c in eqn

$$d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x (y_k - mx_k) - \Delta x$$

$$d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta x m x_k - \Delta x$$

$$\because m = \frac{\Delta y}{\Delta x}$$

$$d_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x$$

$$d_k = 2\Delta y - \Delta x$$

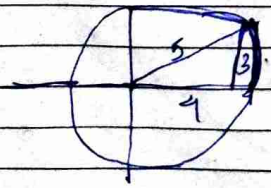
Circle Generation Algorithm

Eqn of circle $x^2 + y^2 = r^2$

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

Here x, y are any point on circle and r is radius

Suppose $x = 4$ $y = 3$ $r = 5$



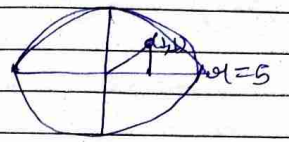
$4^2 + 3^2 - 5^2 = 0$

$16 + 9 - 25 = 0$

$0 = 0$

If $x^2 + y^2 - r^2 = 0$ then the point lies on the circle.

If a point does'nt lie on the circle.

Ex 

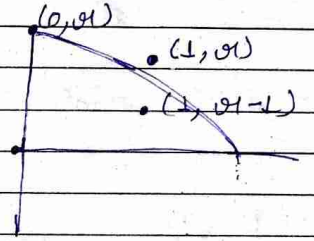
$x^2 + y^2 - r^2 = 0$

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

$-23 \neq 0$

$\therefore$ 1, 1 does not lie on circle.

Suppose we have radius $r = y$



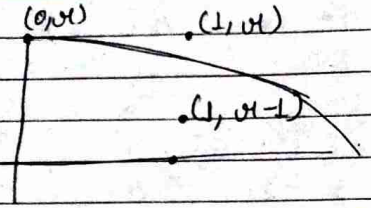
Note :- choose the point closest to the passing line

$(x_i + 1, y_i)$ → common subscript

Note :- If any point is outside then $(x^2 + y^2 - r^2 > 0)$ positive

If any point is inside then $(x^2 + y^2 - r^2)$ is negative

- Find out the eqn for upper and lower point and add the resulting equations of both point and check < 0 and > 0. (to find out decision parameter)



Eqn for upper point $x=1$, $y=r$,

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

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

$1 \neq 0$ — eqn (I)

For lower point $x=1$, $y=r-1$, radius $=r$

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

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

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

$2 - 2r \neq 0$ — eqn (II)

Adding eqn (I) and (II)

$1 + 2 - 2r \Rightarrow \boxed{3-2r}$ ⇒ initial parameter

Ex Suppose $r=10$

Initially d_i (decision parameter) $= 3-2r$)

if $d_i < 0$ (x will always increase)

then $y_i = y_i$ (no change) and $x_i = x_i + 1$

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

if $d_i > 0$

$y_i = y_i - 1$, $x_i + 1$

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

Initial parameter $d_i = 3-2r$ $x_i = 0$, $y_i = 10$

$d_i = 3-2r \Rightarrow 3 - 2(10) \Rightarrow 3-20 \Rightarrow -17$

$d_i < 0$

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

and $x_{i+1} = x_i + 1$ $y_{i+1} = y_i$

$\Rightarrow d_{i+1} = -17 + 0 + 6 = -11$, $x_{i+1} = 1$, $y_{i+1} = 10$

Now $x_i = 1$, $y_i = 10$, $d_i = -11$

Here $d_i < 0$ $\therefore d_{i+1} = d_i + 4x_i + 6$

$d_{i+1} = -11 + 4(1) + 6 = -1$

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

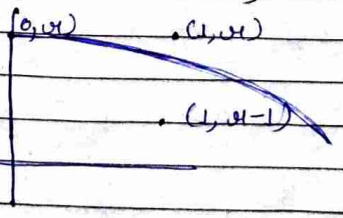
$y_{i+1} = y_i = 10$

Similarly calculate upto stoping condition

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

Derivation Bresenham's circle generation

i) $3-2r$ ii) $d_i + 4x_i + 6$ iii) $d_i + 4(x_i - y_i) + 10$



$\therefore$ eqn of circle is $x^2 + y^2 - r^2$

For point $(1,r)$

$\Rightarrow 1^2 + r^2 - r^2$

$\Rightarrow 1 + r^2 - r^2$

$\Rightarrow 1$ — eqn (I)

For point $(1, r-1)$

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

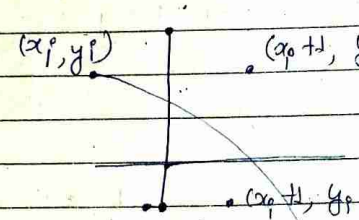
$1 + r^2 + 1 - 2r = r^2$

$2 - 2r = -$ eq^n ①

Adding both eq^n ① and ⑪

$1 + 2 - 2r = \boxed{3 - 2r}$

ii) $d_i + 4x_i + 6$



For point (x_i, y_i)

$(x_i+1)^2 + (y_i)^2 = r^2$ — ①

For point (x_i+1, y_i-1)

$(x_i+1)^2 + (y_i-1)^2 = r^2$ — ⑪

Adding both eq^n ① and ⑪

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

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

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

Now

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

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

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

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

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

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

If d_i is < 0 $\quad y_{i+1} = y_i$

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

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

If d_i is > 0 $\quad y_{i+1} = y_i - 1$

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

$d_{i+1} - d_i = 4x_i + 2[y_i^2 + 1 - 2y_i - y_i^2] + 2 + 6$

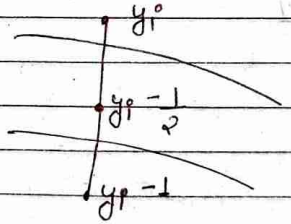
$d_{i+1} - d_i = 4x_i + 2 - 4y_i + 2 + 6$

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

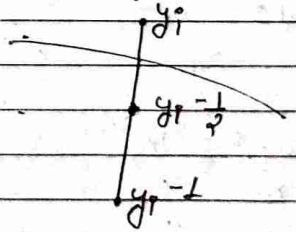
$d_{i+1} = d_i + 4(x_i - y_i) + 10$

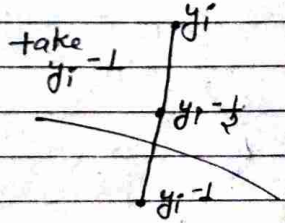
Mid point Circle generation.

- only one point will be used.



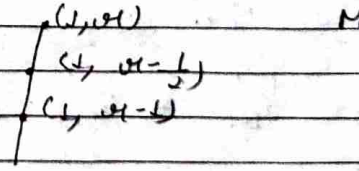
If negative take y_i if positive(+ve)





Initial decision parameter.

$(0, r)$



Mid point $(1, r - \frac{1}{2})$

x $\quad y$

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

$x^2 + (r - \frac{1}{2})^2 - r^2$

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

$1 - r + \left(\frac{1}{4}\right)$ → skip this value because we are working on unit value

Hal parameter = $1 - r$

< 0 $d_{i+1} = d_i + 2x_{i+1} + 1$

> 0 $d_{i+1} = d_i + 2x_{i+1} - 2y_{i+1} + 1$

Suppose $r = 10$ $x_i = 0$, $y_i = 10$

$\therefore x_{i+1} = 1$, $y_{i+1} = 10$ $2x_{i+1} = 2$, $2y_{i+1} = 20$

$d_i = 1 - r = 1 - 10 = -9$ (negative)

$d_{i+1} = d_i + 2x_{i+1} + 1$

$= -9 + 2(1) + 1 = -6$ (negative)

$x_{i+1} = 2$, $y_{i+1} = 10$, $d_i = -6$ (negative)

$\therefore d_{i+1} = d_i + 2x_{i+1} + 1$

$= -6 + 2(2) + 1 = -1$

$y_{i+1} = y_i = 10$, $x_{i+1} = x_i + 1 = 3$

$2x_{i+1} = 4$, $2y_{i+1} = 20$

again $x_{i+1} = 3$, $y_{i+1} = 10$, $2x_{i+1} = 6$, $2y_{i+1} = 20$

$d_i = -1$ (negative)

$\therefore d_{i+1} = d_i + 2x_{i+1} + 1$

$= -1 + 6 + 1 = 6$

Similarly solve further steps still stoping condition $x_{i+1} = y_{i+1}$ or $x_{i+1} > y_{i+1}$.

d_i	x_{i+1}	y_{i+1}	$2x_{i+1}$	$2y_{i+1}$
-9	1	10	2	20
-6	2	10	4	20
-1	3	10	6	20
6	4	9	8	18
-3	5	9	10	18
8	6	8	12	16
5	7	7	14	14

Derivation mid point circle generation.

• mid point co-ordinates are $(x_i + 1, y_i - \frac{1}{2})$

$d_i = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$

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

$d_{i+1} - d_i = (\overbrace{x_{i+1}}^{a} + \overbrace{1}^{b})^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$

here take $x_{i+1} = x_i + 1$ $(x_i + 1)^2 - (y_i - \frac{1}{2})^2 + r^2$

$d_{i+1} - d_i = (x_{i+1})^2 + 1 + 2(x_i + 1) + (y_{i+1}^2 - 2y_{i+1} + \frac{1}{4}) - (y_i^2 - y_i + \frac{1}{4}) - (x_i + 1)^2$

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

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

If d_i is < 0 $y_{i+1} = y_i$

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

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

If d_i is ≥ 0 $\quad y_{i+1} = y_i - 1$

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

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

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

$= 2(x_i + 1) - 2(y_i - 1) + 1 \quad \because x_{i+1} = x_i + 1,\ y_{i+1} = y_i - 1$

$= 2x_{i+1} - 2y_{i+1} + 1$

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

$d_{i+1} = d_i + 2x_{i+1} - 2y_{i+1} + 1$ —

- Random Scan and Raster Scan is left. →

Polygon Filling Algorithm

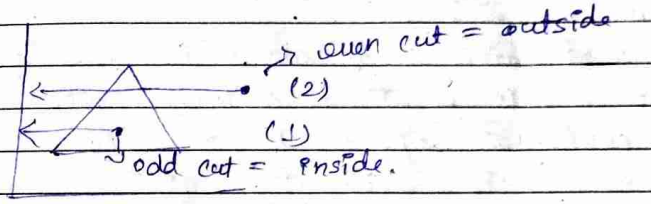
This algorithm lines interior points of a polygon on scan line and these points are done on or off according to requirement. The polygon is filled with various colors by colouring various pixels.

Seed pixel → A starting pixel from which coloring of polygon starts, is called seed pixel.

For finding seed pixel there are two rules

- Inside outside rule → A line is drawn parallel to X axis from left to right direction cutting the borders of polygon
- If the line cuts the border of polygon even number of times, then it is outside point or pixel

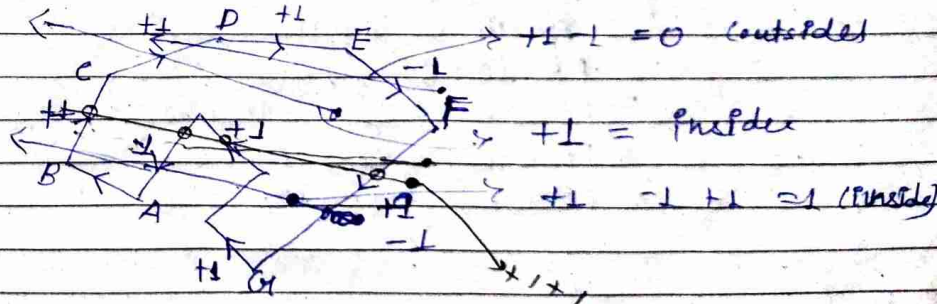
- If the line cuts border of polygon odd number of times then it is point/pixel is inside.



ii) Non-zero / winding number rule.

It is an algorithm to perform the test of finding whether the point is inside or outside of polygon. A winding number is calculated for given point with respect to polygon. If winding number is non-zero then point lies inside the polygon, Else it lies outside of polygon.

Assign +1 to the arrows going upwards
Assign -1 to the arrows going downwards.



How to colour.

There are two process which is used to colour in the polygon 1) Boundary 2) Flood fill

Flood fill algorithm.

In this method, a point or seed which is inside region is selected. this point is called a seed point. Then 4 connected approach or eight connected approach is used to fill with specified colour.

This method is more suitable for filling multiple colors boundary. When boundary is of many colors and interior is to be filled with one color we use this algorithm

Disadvantages- Very slow algorithm
May be fail for large polygon.

```
Void FF (int x, int y, int fill, int old)
{ if (getpixel (x,y) == old)
  { putpixel (x,y, fill)
    FF (x+1, y, fill, old)
    FF (x-1, y, fill, old)
    FF (x, y+1, fill, old)
    FF (x, y-1, fill, old)
  }
}
```

→ Similarly for 8 connected.

4 connected approach

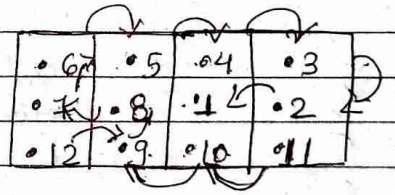
	P(x, y+1)	
P(x-1, y)	P(x, y)	P(x+1, y+1)
	P(x, y-1)	

8 connected approach

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)

→ 2 marks

Eg



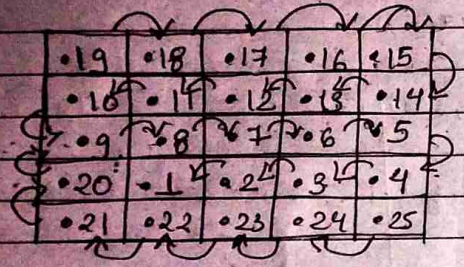
order is R-L-T-Bottom

→ 4 connected approach

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

Here at 11 check R, L, T, B if all four sides are filled and then also any pixel is left then start back tracking

Ex-2



ORDER R L T B

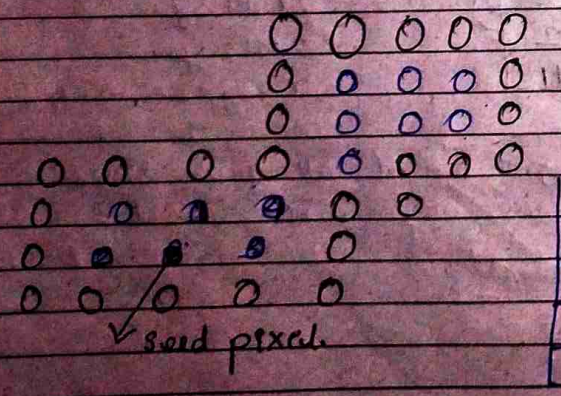
Boundary Filling Algorithm.

For boundary color

```
Void BF (int x, int y, int fill, int bound) →
  If (getpixel(x,y) != bound && getpixel(x,y) != fill)
    { putpixel (x, y, fill)
      BF (x+1, y, fill, bound)
      BF (x-1, y, fill, bound);
      BF (x, y+1, fill, bound)
      BF (x, y-1, fill, bound)
    }
}
```

Note → continue till boundary is found.

Stack Method. (3013123)



4 connected Method

order

T B L R

1

3 0 4

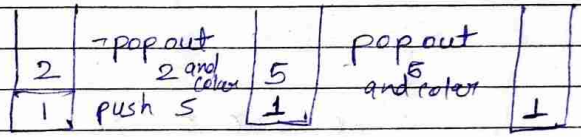
2

Insert the stack that get coloured according to their sequence.

Check top of the stack and pop out the pixel present at top and colour it. (this pixel will get colour after seed pixel).

- Now insert the pixel associated with 3 to prevent the loss of information

Again pop out 4 and check pixel associated with it and insert them in stack according to their sequence, and select only empty or uncoloured pixel.



When stack is blank Stop the method.

Disadvantage → The diagonal pixel are left uncolored as we cannot process the diagonal pixel.

8 connected Method

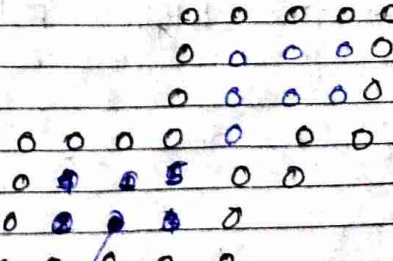
order

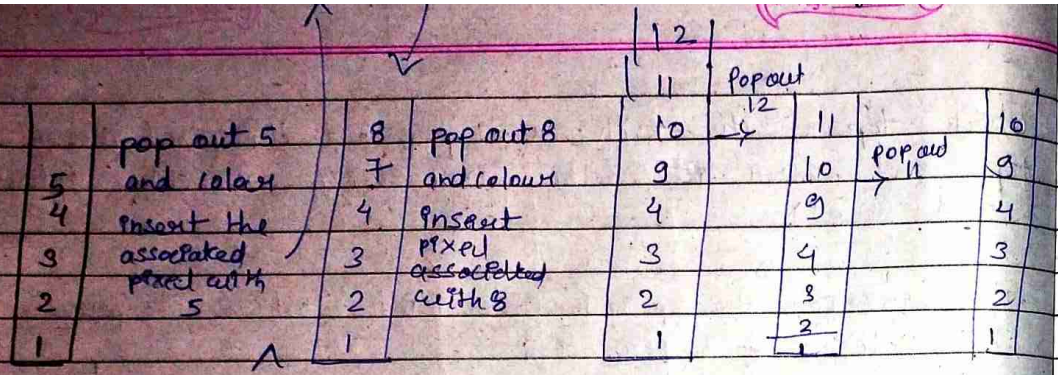
5 1 6

3 0 4

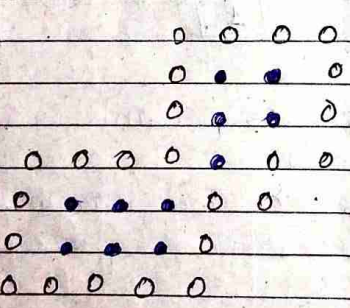
7 2 8

The process is same as 4 connected method for coloring





Polygon After colouring



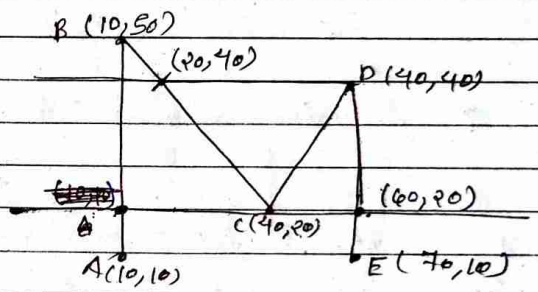
Scan line Polygon Filling.

This algorithm works by intersecting scanline with polygon edges and fills the polygon between pairs of intersections.

Algorithm

1. Find all y_{min} of every edge.
2. Initially Active edge list is empty
3. Repeat for each y_{min}
   - Store all cutting edges
   - Sort on x value
   - Fill pixel from x_1 to x_2
   - If y_{max} is reached remove the edge
4. $y = y+1$ and $x = x + \frac{1}{m}$

Example



Take edges which are not Horizontal, but vertical or slanted
Here (AB, BC, CD, DE)
Take y minimum of each edge
y_{min} of AB = 10, y_{min} of BC = 20, y_{min} of CD = 20, y_{min} of DE = 10

Find out edges for different y value

For 10 →

AB			DE		
y_{max} AB	x_{min} AB	Slope AB	y_{max} DE	x_{min} DE	Slope DE
50	10	0	40	70	-1

for 20 → AB, BC, CD, DE

$$\text{Slope}(m) = \frac{y_2 - y_1}{x_2 - x_1} = \frac{50 - 10}{10 - 0} = \frac{40}{0}$$

$$= \frac{1}{m} = \frac{0}{1} = 0$$

AB			DE		
y_{max}	x_{max}	Slope	y_{max}	x_{max}	Slope
50	10	0	40	69	-1
50	10	0	40	68	-1
50	10	0	40	67	-1
50	10	0	40	66	-1
50	10	0	40	65	-1
50	10	0	40	64	-1
50	10	0	40	63	-1
50	10	0	40	62	-1
50	10	0	40	61	-1
50	10	0	40	60	-1

The value of x_{max} is decreased according to value of slope.

The value of scan line which is height of y is increased by 1 unit, when one row is completely filled with colour.

Now $y = 20$, i.e the height of scan line is 20.

Edges intersected by scan line.

AB	BC	CD	DE
50 | 10 | 0	50 | 40 | -1	40 | 40 | 0	40 | 60 | -1
50 | 10 | 0	50 | 20 | -1	40 | 40 | 0	40 | 40 | -1

. Now Remove Edges whose max y is reached.
∴ CD and DE is removed.

(Now height scan line is 41) → The process after $y = 40$

Edges intersected by scanline.

AB	BC
50 | 10 | 0	50 | 40 | -1
50 | 10 | 0	50 | 10 | -1

→ Stopping condition as end the process

Transformation (6|4|23) → page 194

Geometric Transformation - The object itself is transformed relative to system or background. The mathematical statement of this viewpoint is defined by geometric transformation applied to each point of object.

$(x, y) \rightarrow (x', y')$

Ex Moving Automobile while keeping the bg fixed.

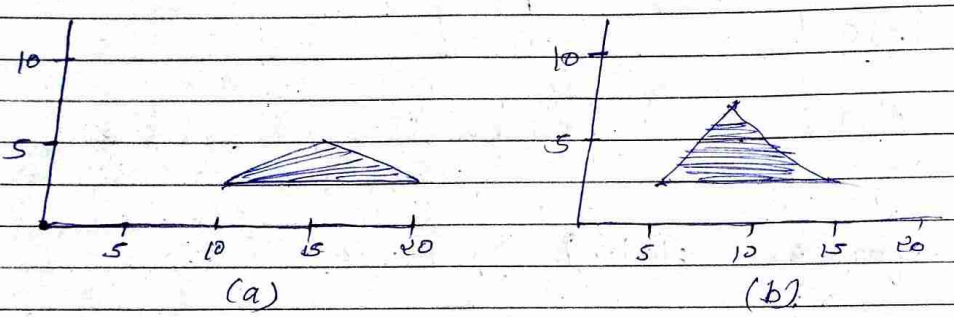
Types of Transformation

Translation → It is the straight line movement from one position to another, here the object is positioned from one co-ordinate to another. We translate a two-dimensional point by adding translation distance (t_x) and (t_y) to the original co-ordinate position (x, y) to move the point to a new position (x', y')

$$x' = x + t_x, \quad y' = y + t_y$$

The translation distance pair (t_x, t_y) is called a translation vector or shift vector.

Ex Moving a polygon from position (a) to position (b) with translation vector.



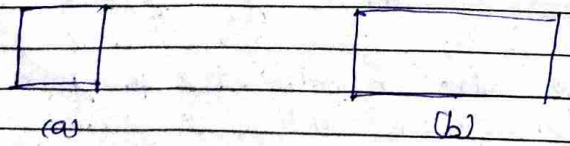
Translation Matrix.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling - A scaling transformation alters the size of an object. This operation can be carried out for polygon by multiplying the co-ordinates values (x, y) of each vertex by scaling factors s_x and s_y to produce transformed co-ordinates (x', y')

$x' = x \cdot s_x$, $y' = y \cdot s_y$

Ex. Turning a square (a) into rectangle (b) with scaling factors $s_x = 2$ and $s_y = 1$



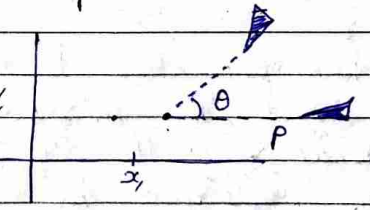
Scaling Matrix

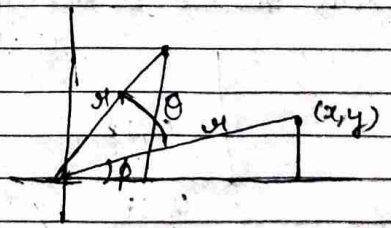
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation :- A two dimensional rotation is applied to an object by repositioning it along a circular path in xy plane. To generate a rotation we specify a rotation angle θ and the position (x, y) of the rotation point or (pivot point) about which the object is to be rotated.

Positive values for rotation angle define counter clockwise rotation pivot point and negative values rotate objects in clockwise direction.

Ex Rotation of an object through angle θ about the pivot point.





r is constant distance of the point from origin.

$\cos\phi = \frac{x}{r} \quad \therefore x = r \cdot \cos\phi$

$\sin\phi = \frac{y}{r} \quad \therefore y = r \cdot \sin\phi$

$r \cdot \cos(\theta+\phi) = r\cos\theta \cdot \cos\phi - r \sin\theta \sin\phi$

$\because x = r\cos\phi$ and $y = r \cdot \sin\phi$

$\therefore r \cdot \cos(\theta+\phi) = x\cos\theta - y \cdot \sin\theta$

$r \cdot \sin(\theta+\phi) = r \cdot \sin\theta \cos\phi + r\cos\theta \cdot \sin\phi$

$r \cdot \sin(\theta+\phi) = x\sin\theta + y\cos\theta$

$x' = r\cos\theta - y \cdot \sin\theta$

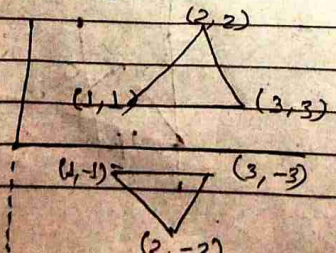
$y' = r\sin\theta + y\cos\theta$

Rotation Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection → It is a transformation that produces mirror image of an object. The mirror image for a two dimensional reflection is generated relative to an axis of reflection by rotating the object 180° about the reflection axis.

- Reflection of an object about x-axis.



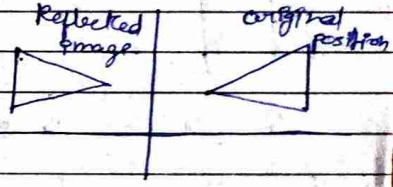
Matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This transformation keeps x values the same, but flips the y values of co-ordinates position.

Reflection about y axis → It flips x co-ordinates while keeping y coordinates same. The matrix for this transformation is

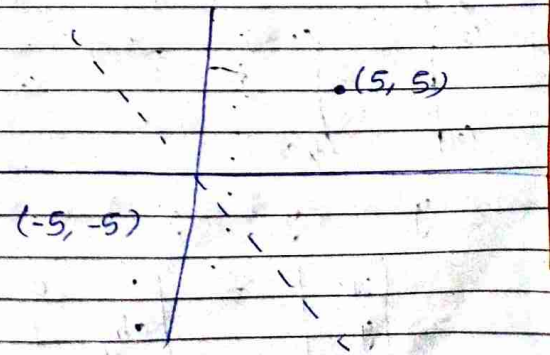
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Reflection along axis perpendicular to xy plane and passing through origin.

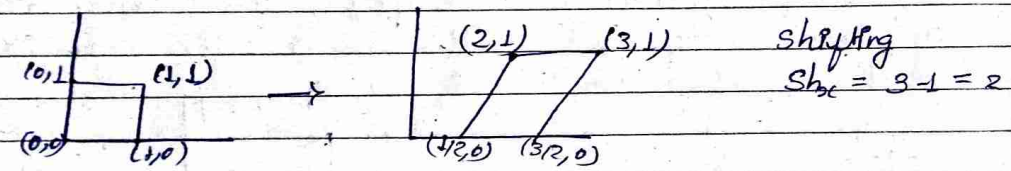
We flip both x and y co-ordinates of a point by reflecting to an axis that is perpendicular to the xy plane and passes through origin

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Shear → A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called shear.

x shear

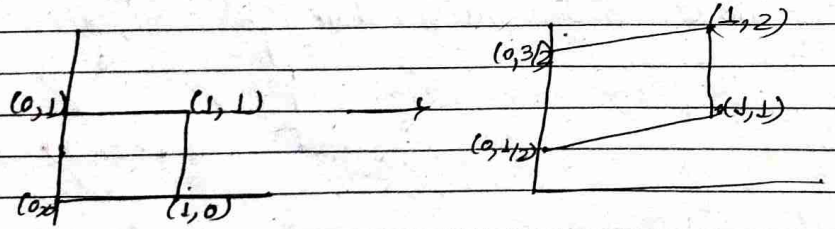


shifting
$Sh_x = 3-1 = 2$

$\therefore x' = x + sh_x * y \quad y' = y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

y shear



$y' = y + sh_y * x \quad x' = x$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3D Transformation

Translation :- $x' = x + t_x, \; y' = y + t_y, \; z' = z + t_z$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaling $\quad x' = x \cdot S_x, \; y' = y \cdot S_y, \; z' = z \cdot S_z$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

Rotation through z axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation through y x axis , x portion is same

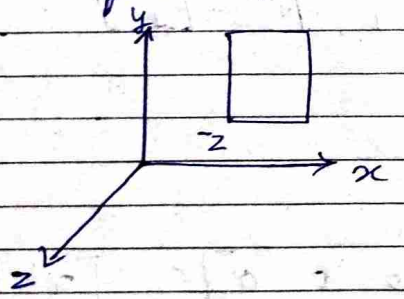
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation through y axis, y position as it is

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

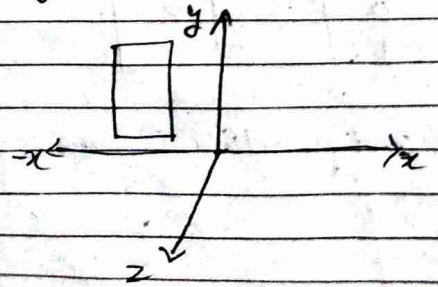
Reflection

xy reflection, x and y no change, $z' = -z$



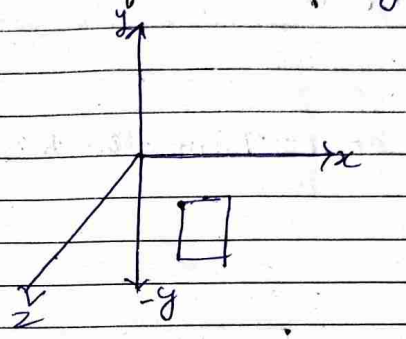
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

yz Reflection (x is negative)



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

zx reflection, y is negative



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Shearing

x shearing, x is fixed y and z is changed

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & shy & shz & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

y shearing y is fixed x and z is change

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ shx & 1 & shy & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

z shearing z is fixed x and y is changed

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ shx & shy & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Concatenation → Combining or applying more than 1 transformation.

In 2D Transformation

1) Y axis reflection 2) Rotation 90° 3) Translation $t_x=5, t_y=8$ point (1, 1)

- y axis reflection

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

- Rotation 90°.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

Translation $t_x=5$ $t_y=8$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1+5 \\ -1+8 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 1 \end{bmatrix}$$

Polygon Clipping Algorithm (Page no. 236).

This is one of the oldest and most popular line clipping procedures. Generally, the method speeds up the processing of line segments by performing initial tests that reduce the number of intersections that must be calculated. Every line end point in a picture is assigned a four digit binary code, called a region code, that identifies the location of the point relative to the boundaries of the clipping rectangle. A value of 1 in any bit position indicates that point is in that relative position; otherwise, the bit position is set to 0.

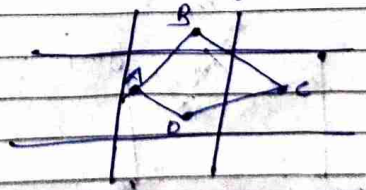
4 bit Region Code

T	B	R	L

cohen sutherland algorithm

1001	1000	1010
0001	0000	0010
0101	0100	0110

Step 1 Assign bit code to every edge (two points)



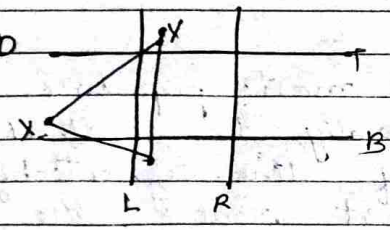
Bit code of A is 0000

Bit code of B is 0000

0000

Inside window

AND



Bit code of X 0010
Bit code of Y 1000
0000

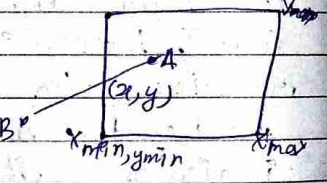
If the result is 0000, the line is completely outside of clipping region.

Formula

Left $X = x_{min}$

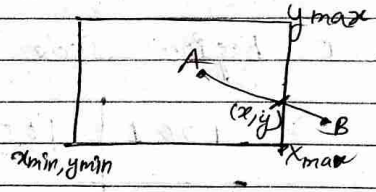
$m = \frac{y_2 - y_1}{x_2 - x_1}$ $\quad m = \frac{y - y_1}{x - x_1}$

$\therefore y = y_1 + m(x_{min} - x_1)$



Right $x = x_{max}$

$\therefore y = y_1 + m(x_{max} - x_1)$



Top $y = y_{max}$

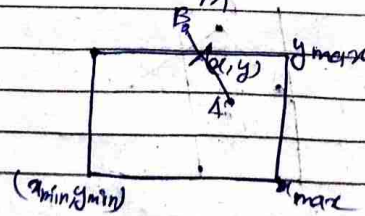
$m = \frac{y_2 - y_1}{x_2 - x_1}$

$m = \frac{y_{max} - y_1}{x - x_1}$

$\therefore (x - x_1)\, m = y_{max} - y_1$

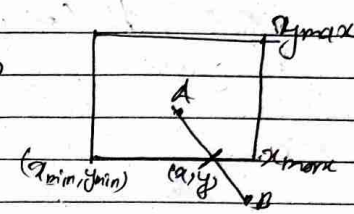
$x - x_1 = \frac{y_{max} - y_1}{m}$

$x = x_1 + \frac{y_{max} - y}{m}$

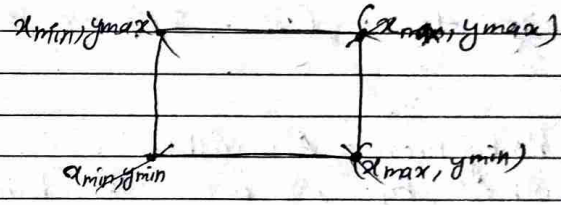


Bottom $y = y_{min}$

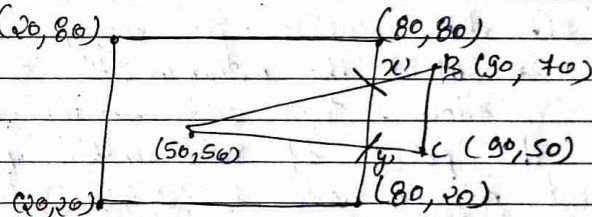
$\therefore x = x_1 + \frac{y_{min} - y_1}{m}$

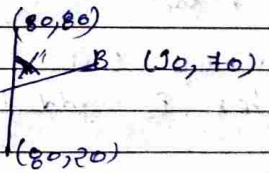


Corners



Example –





Taking line AB (50,50)
x' be point with (x,y)

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{70 - 50}{90 - 50} = \frac{20}{40} = \frac{1}{2}$

$x = 80$

$y = y_1 + m(x_{max} - x_1)$

$= 50 + \frac{1}{2}(80 - 50) = 50 + 15 = 65$

on line AC let 'y' be point with (x, y)

$x = 80 \quad , y = 1$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{50 - 50}{90 - 50} = 0$$

$$y_s = y_1 + m(x_s - x_1) \Rightarrow 50 + 0 = 50$$

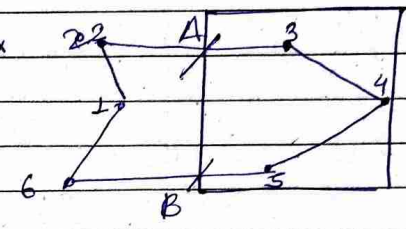
Cohen – Hodgeman Algorithm

It is performed by processing the boundary of polygon against each window corner or edge.

Four possible situation while processing

- If the vertex is outside the window, the second vertex is inside the window, then 2nd vertex is saved as well as the point of intersection of window boundary and polygon edge is saved.
- If both vertex are inside window boundary, then only 2nd vertex is added
- If the 1st vertex is inside and 2nd vertex is outside, the edge which intersects with window is added to output list.
- If both vertices are outside window, then nothing is added to output list.

Ex



A 3 4 5 B → is saved According to all the above 4 points.

Progection. (Java point)