Application

→ At this stage user interacts with the scene and alters it as per his requirements.

- In this stage 3D objects are created using modeling tools ~~etc~~.

3D objects can be moved by transformation
models created needs to be positioned and oriented correctly in 3D space.

Geometry Stage

The geometry step which is responsible for the majority of the operation with polygon and their vertices.

Sub-stages are

a) Model Transformation
b) View Transformation → Perspective
Isometric

c) Lighting
d) Projection
e) Clipping
f) Screen Mapping

Date.
Page No.

Basic computer graphics pipeline

1) Modeling → The first step to create a 3D model of object that you want to render.

2) Transformation → Once the 3D model is created it need to be positioned and oriented correctly in 3D space.

3) ~~Ut~~ Mapping →

Computer graphics pipeline or rendering pipeline defines the primitive operation required for this mapping

Graphics pipeline primarily works in 3 stages.

(i) Application
(ii) Geometry
(iii) Rasterization

Date.
Page No.

e) Clipping

3D model is clipped so that only parts within the view are left.

f) Screen Mapping

Object which survive clipping are passed to screen mapping stage.

Rasterization Stage

In this stage all primitives are converted into pixels in the window.

Model Transform

Transform the vertices and normals of 3D objects from model space to world space.

View transform

The camera and all the 3D objects are transformed here.

Lighting

A lighting equation used to calculate color at each vertex of 3D object that is to be affected by lighting

Projection

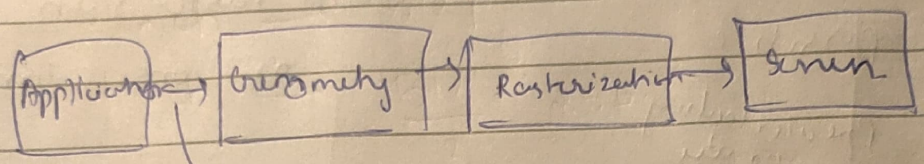
The purpose of projection is to simplify clipping

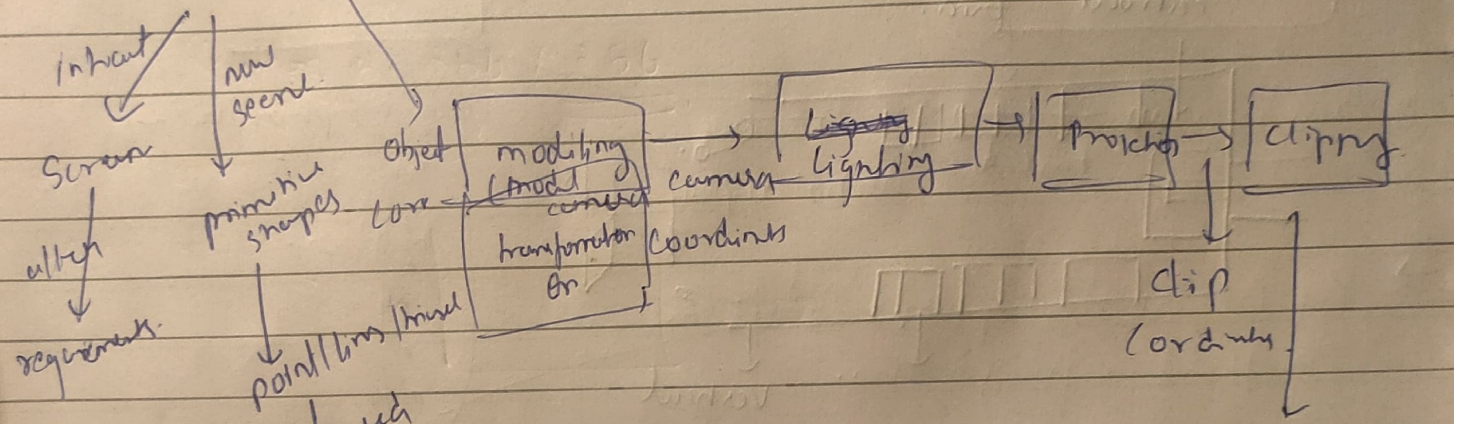
Done by orthographic projection

Graphics pipeline / rendering pipeline

defines primitive operation required for mapping.

- Application
- Geometry
- Rasterization





Application (perform collision detection, morphing animation)

↳ helps in reducing memory requirements

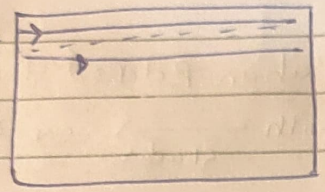
Raster Scan System.

→ Refresh rate. (60, 80) this [independent picture complexity]

→ It is scanning technique in which e⁻ beam move top along with screen

→ It moves top → bottom, left → right covering one scan line at time.

→ line

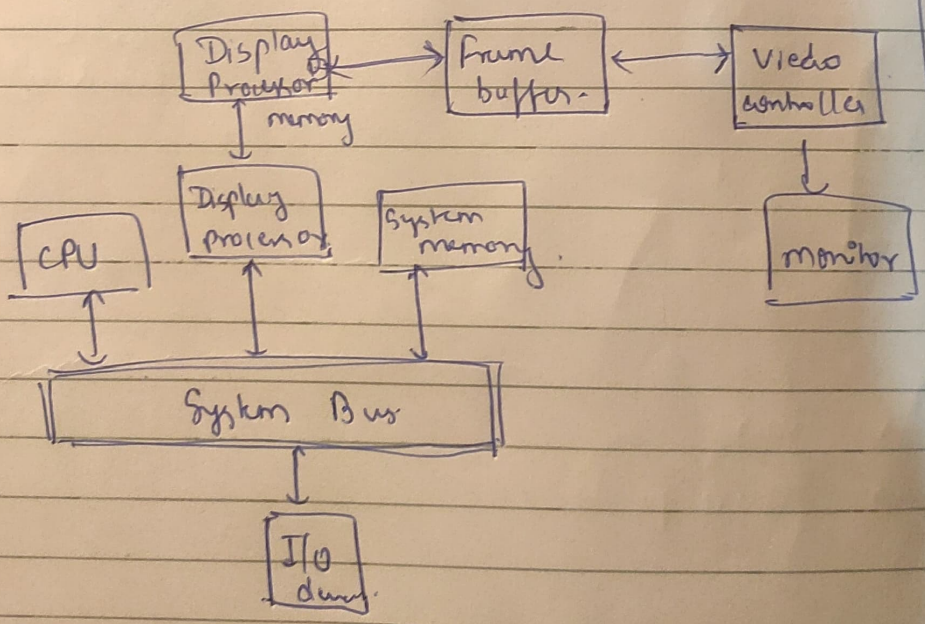


→ raster scan ——based——→ pixel intensity. Control display rectangles box.

→ Picture description ——stored——→ frame buffer/ refresh paths.

→ Frame buffer ——also know——→ bit

Eg Television



→ Special memory area ——allocated——→ graphics only.

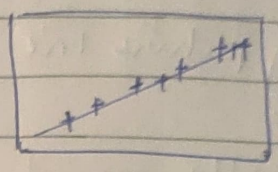
→ Frame buffer store intensity values

Random Scan System.

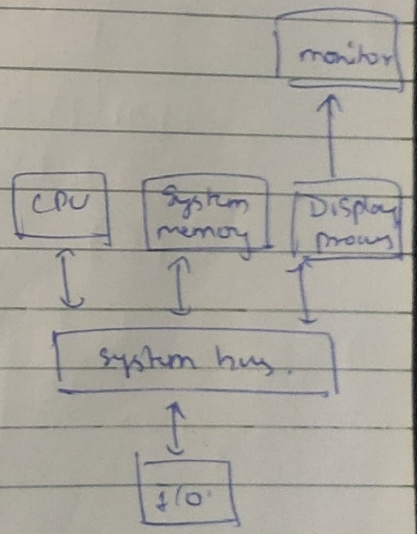
→ Refresh (30, 60) hz [depends number of line]

→ It is scanning technique e⁻ beam directly directed on areas where where picture to be drawn

→ Vector.



→ In random we does require any frame buffer.



Eg Pen plotter

Advantages

1. Real image
2. many color produced
3. Dark scene

Disadvantages

1. ~~less resolution~~
2. Display picture line by line
3. More costly. invert of time

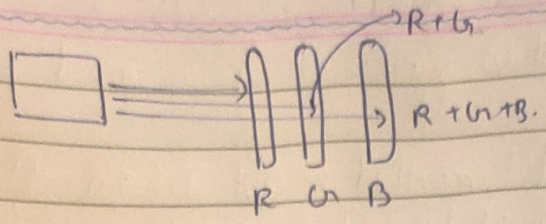
Advantages

1. High resolution
2. Draw smooth line
3. less required as frame buffer not used.

Disadvantages

1. ~~Less resolution~~ Refresh static images with shades corrupt the drawn
2. Color limitations.

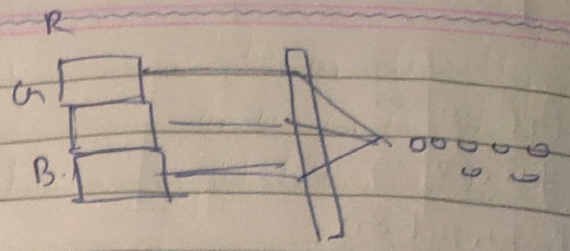
Beam Penetration method.



Used. Random scan →

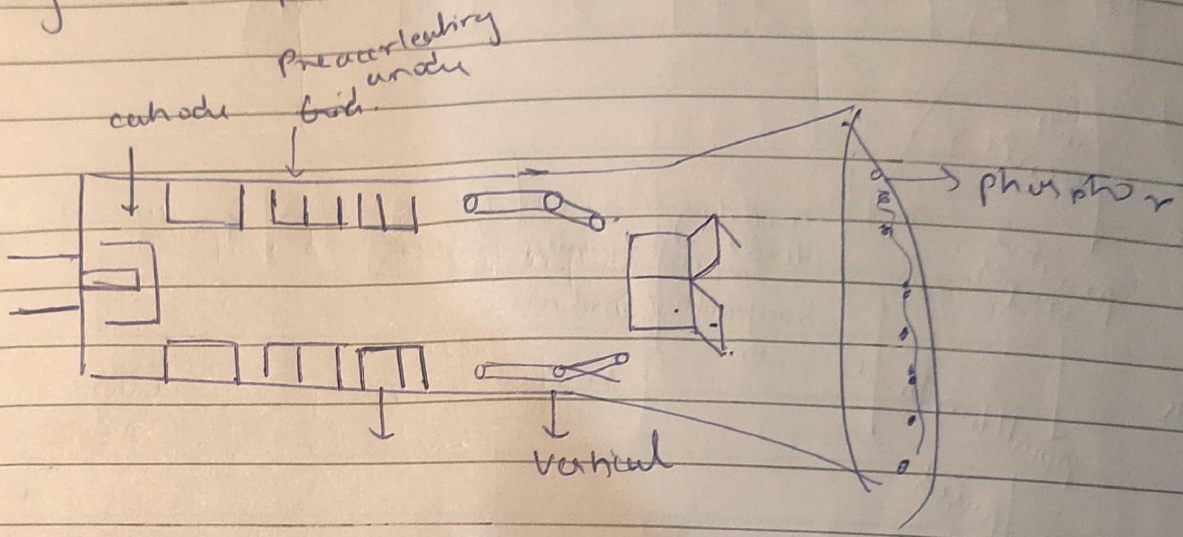
only 4 color possible.

Shadow mask



Used in raster scan.

millions color

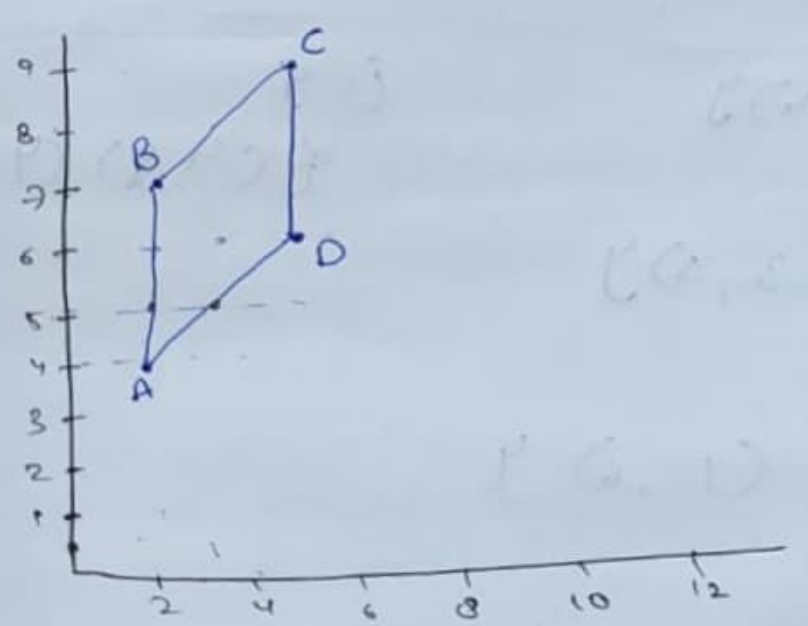


Inside / outside Test

→ If the scan line pass through a vertices both the edges that are connected to the vertex are on same side of the scanline. Then we need to count a vertex twice.

→ If the scanline pass through a vertex both the edges that are connected to the vertex are on the opposite side of the scanline. Then we need to count a single intersection point.

Q Explain the data structure used by scanline polygon filling alg. Determine the content of active edge table to fill the polygon with the vertex A(2,4) B(2,7) C(4,9) D(4,6)



A(2,4) D(4,6) B(2,7) C(4,9)

x_{k+1}	y_{k+1}	$dx+1$
1	6	72
2	6	144
3	6	216
4	5	288
5	5	360
6	4	432
7	3	504

Ques $r_x = 8$ $r_y = 6$

Start (0, 6)

x_k	y_k	p_{1_0}	dy	dx
0	6	-332	768	0
1	6	-224	768	72
2	6	-44	768	144
3	6	208	640	216
4	5	-108	640	288
5	5	288	512	360
6	4	244	384	432

→ find $P_{20} = x_0^2 y\left(\frac{x}{x_0} - 1\frac{1}{2}\right)^2 + x_0^2(y_0 - 1)^2 - x_0^2 x_0^2$

Repeat till ($y > 0$)

plot (x, y)

if $P_2 > 0$

{

$x = x \quad y = y - 1$

update $dy \to dy - 2x^2x$

$p2 = p^2 - dy + x^2x$

}

else

{

$x = x + 1$

$y = y - 1$

$dy = dy - 2x_x^2$

$dx = dx + 2x^2y$

$P2 = p2 + dx - dy + x_x^2$

}

computer graphics

→ DDA (Digital Differential Analysis)
scan conversion, integrated approach

straight line equation

$y = mx + c$

$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1}$$

case I

$m > 1$

$x_n = x_1 + 1/m$

$y_n = y_1 + m$

case II

$m < 1$

$x_n = x_1 + 1$

$y_n = y_1 + m$

case III

$m = 1$

$x_n = x_1 + 1$

$y_n = y_1 + 1$

Algorithm

1. calculate the slope m
2. check the cases
3. fetch the new coordinates
4. Based on new coordinate calculate fetch coordinate
5. calculate the final wordinate by abs(New wordinate)
6. End

A(0,0) & B(8,4) $m = \frac{4-0}{8-0} = 0.5$ | case II

$x_n = 0+1 = 1$

$y_n = 0+0.5$ or

x_i	y_i	x_n	y_n	final
0	0	1	0.5	(1,1)
1	0.5	2	1	(2,1)
2	1	3	1.5	(3,2)
3	1.5	4	2	(4,2)
4	2	5	2.5	(5,3)
5	2.5	6	3	(6,3)
6	3	7	3.5	(7,4)
7	3.5	8	4	(8,4)

$x_n = x_i + 1/m = 5 + \frac{1}{2} = \frac{10+1}{10} = \frac{11}{10}$

$y_n = y_i + m = 10+2 = 12$

$x_n = 20+1$

$y_n = 20 + 0.8$

$x_n = 21+1$

$30.8 + 0.8$

$\frac{18-10}{30-20} = \frac{8}{10}$

NOTE

on the boundary b/w two region / o starts slope of curve is -1

Algorithm

→ Read radius r_x & r_y

→ Initialise start point of region 1 as $x=0, y=r_y$

→ Calculate $P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$

→ calculate $dx = 2r_y^2 x$, $dy = 2r_x^2 y$

→ Repeat while $dx < dy$

Plot (x, y)

if $P1 < 0$

{

$x = x+1$

update $dx \rightarrow dx + 2r_y^2$

$P1 = P1 + 2r_y^2 x + r_y^2$

}

else

{

$x = x+1 \quad y = y-1$

update $dx \Rightarrow 2r_y^2 * dx$

update $dy \rightarrow dy - 2r_x^2$

$P_1 \rightarrow P1 + dx - dy + r_y^2$

}

when $dx \geq dy$ plot P_2 as

Bresenham's Algorithm

It determines the points of n-dimensional raster that should be selected in order to form a close approximation to a straight line b/w two points

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

P_k = Decision Parameter

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

$\Delta y = y_2 - y_1$, $\Delta x = x_2 - x_1$

Case I

$P_k > 0$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$x_n = x_i + 1$

$y_n = y_i + 1$

Case II

$P_k < 0$

$P_{k+1} = P_k + 2\Delta y$

$x_n = x_i + 1$

$y_n = y_i$

Ex A(20, 10) B(30, 18)

$\Delta x = 10$

$\Delta y = 8$.

$P_k = 2 \cdot 8 - 10$

$= 16 - 10$

$= 6$

$P_{k+1} = 2 + 16 - 20$

$= 2 - 4$

$= -2$

Case I

$x_n = 20 + 1$

$y_n = 10 + 1$

$P_{k+1} = 6 + 16 - 20$

$= 22 - 4$

$= 18$

$x_n = 21 + 1$

$y_n = 11 + 1$

$-2 - 4$

-6

$-6 - 4$

10

$6 - 4$

2

$14 + 16 - 20$

$30 - 20$

10

$10 - 4$

6

Advantages

1. fastest method than method of using direct use of line equation
2. This method does not use multipl
3. This allows us to ~~dea~~ted to changes in the value of x & y value so plotting of same
4. It gives overflow indication when a point a reposition
5. It is easy method because easy step involves just two addition

Dis

1. It involves floating point addition rounding of ~~error~~ accumulation of round of error cause accumulation of error
2. rounding of ~~operation~~ floating point operation consume ~~of~~ a lot of time

~~It~~ is more suitable for generating line using the software but it is less suited hardware implementation

Circle Generation Algorithm

- Mid Point
- Bresham's circle

given $y = 10$

$y = RR$ = radius

given $(x_c, y_c) = (2, 2)$

Decision Parameter

$P_k = 1 - R$

Case I

$P_k < 0$

$P_{k+1} = P_k + 2x + 1$

$x_n = x + 1$

$y_n = y$

Case II

$P_k > 0$

$P_{k+1} = P_k + 2x + 1 - 2y$

$x_n = x + 1$

$y_n = y - 1$

Note when $x = y$ stop "x is always less than y"

$P = 3 - 2R$

Case I

$P_k < 0$

$P_{k+1} = P_k + 4x + 6$

$x_n = x + 1$

$y_n = y$

Case II

$P_k > 0$

$P_k = P_k + 4(x - y) + 10$

$x_n = x + 1$

$y_n = y - 1$

Algo

① Input of circle radius r & (x_c, y_c)

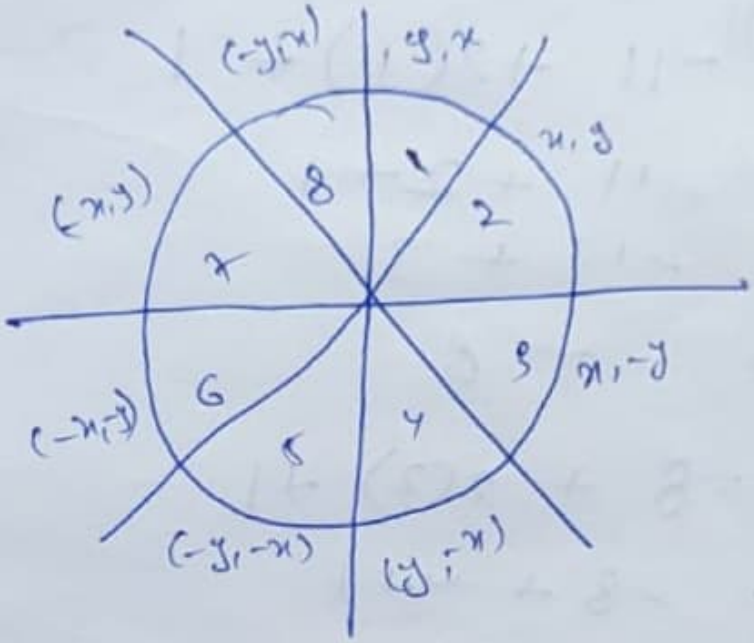
② Initial value $x = 0$ $y = r$

③ Plot pixel $(x + x_c, y + y_c)$

④ DP $P_x = 1 - r$ → $1 - 10 = -9$

⑤ Start of loop

$-9 + 2 + 1$
$-9 + 3$
-6
$-6 + 3$

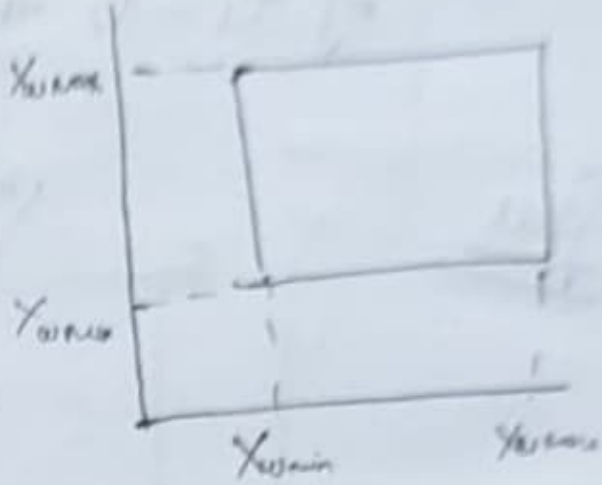


P_{k+1}	x_n	y_n	$2(x_n + x_c)$
-9	0	10	2
-6	1	10	3
-1	2	10	4
6	3	10	5
-3	4	9	6
8	5	9	
5	6	8	

$-1 + 6 + 1$
$-1 + 7$
-3

2	12
3	12
4	12
5	12
6	11

Liang Barsky Algo.



① $x = x_1 + t\Delta x$
$y = y_1 + t\Delta y$

② $x_{wmin} \leq x \leq x_{wmax}$
$y_{wmin} \leq y \leq y_{wmax}$

③ $p \not\$ q$
$tp_k \leq q_k$

$p_1 = -\Delta x \quad q_1 = x_1 - x_{wmin}$
$p_2 = \Delta x \quad q_2 = x_{wmax} - x_1$
$p_3 = -\Delta y \quad q_3 = y_1 - y_{wmin}$
$p_4 = \Delta y \quad q_4 = y_{wmax} - y_1$

$x_{wmin} \leq x_1 + t\Delta x$
$x_{wmax} \geq x_1 + t\Delta x$

$y_{wmin} \leq y_1 + t\Delta y$
$y_{wmax} \geq y_1 + t\Delta y$

$t\Delta y \geq y_{wmin} - y_1$
$t\Delta y \leq y_{wmax} - y_1$

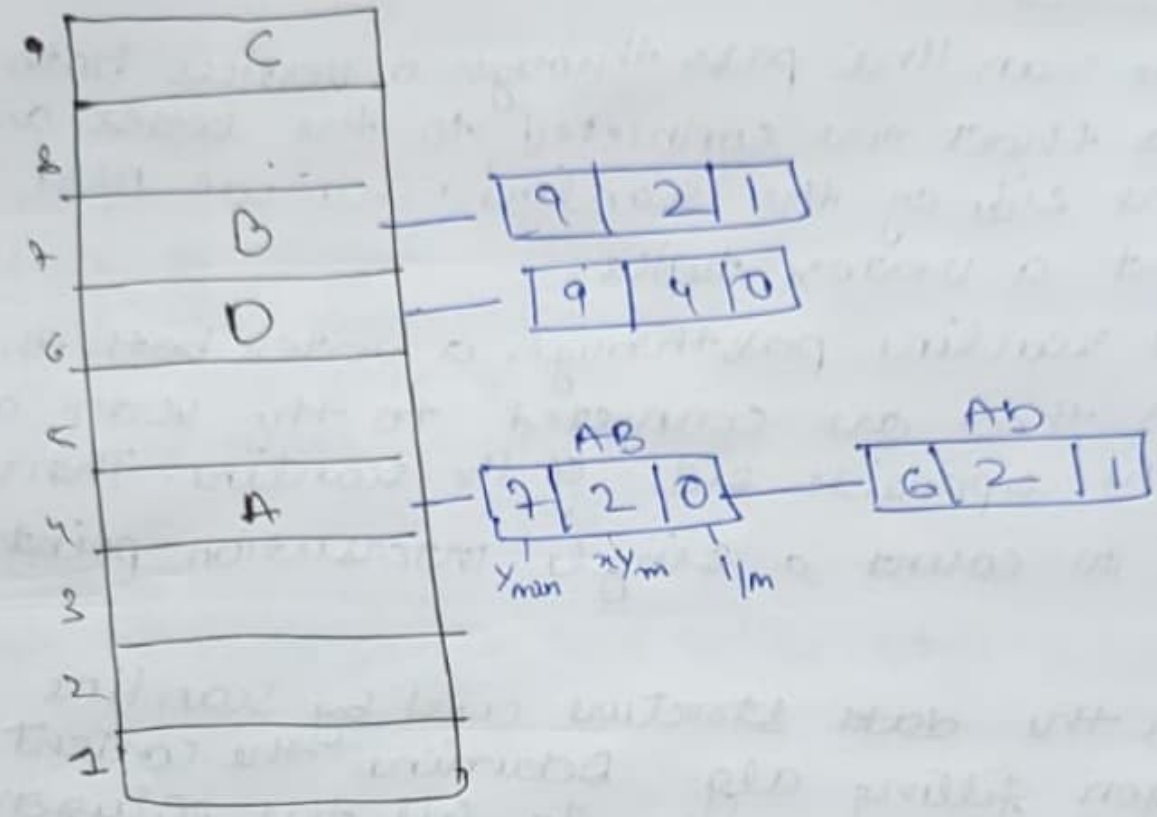
④ If $p_k = 0$
then line is parallel to window

If $p_k < 0$
then line is completely outside

If $p_k < 0$
$t_1 = \max\left(0, \frac{q_k}{p_k}\right)$

If $p_k > 0$
$t_2 = \min\left(1, \frac{q_k}{p_k}\right)$

GET (Global Edge Table)



AET (Active edge Table)

i = 4

{(2, 4) (2, 4)}

i = 5

{(2, 5) (3, 5)}

i = 6

{(2, 6) (4, 6)}

i = 7

{(2, 7) (4, 7)}

i = 8

{(3, 8) (4, 8)}

i = 9

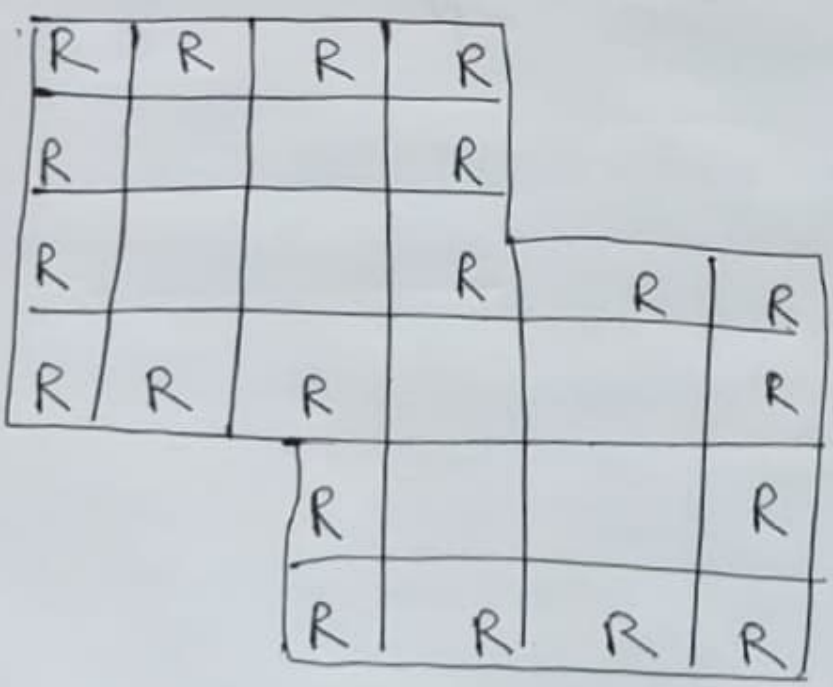
{(4, 9) (4, 9)}

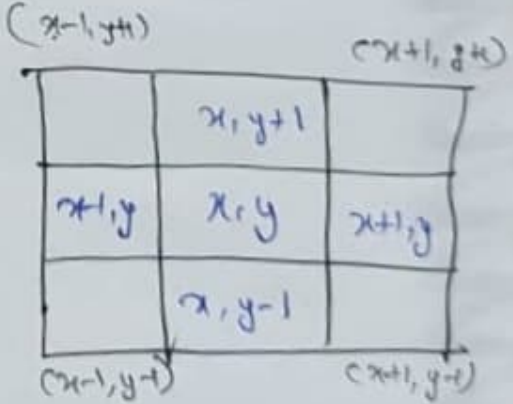
Polygon filling Alg.

- Scan line (large polygon)
- Seed fill (small polygon)
  - Boundary fill (BG color ≠ inside polygon)
  - Flood fill

* In case of boundary filling edges of polygon are first drawn then starting with the seed any pixel of polygon, we examine neighbouring pixel to check wheater boundary pixel is reached

* Boundary drawn here is a boundary pixel (R) and highlight each pixel till it is reached





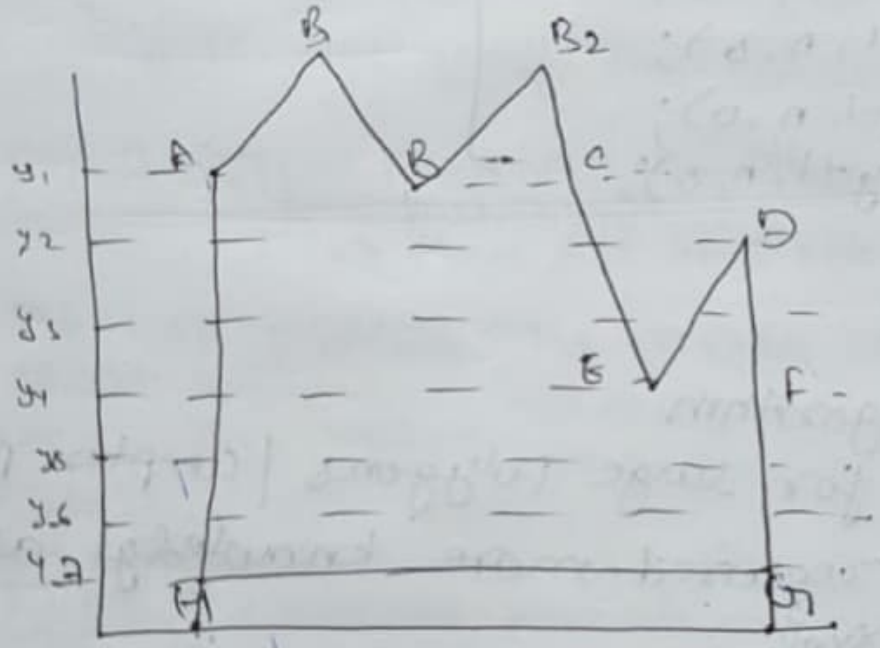
Inside | outside Test (Odd & even parity rule)

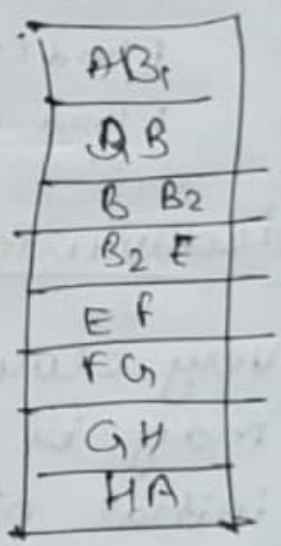
Algorithm Scan line :-

→ Find the intersecting point of the scan lines with all edges of polygon
→ Sort the intersection of the point by incr x coordinat
→ make the pair of the intersection
→ All the areas coming within these all ~~points~~ pairs

Note

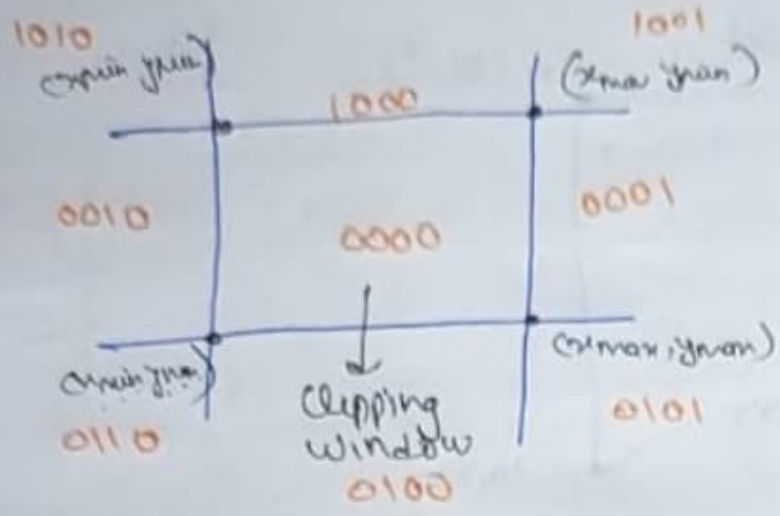
We need to creat and active edge table



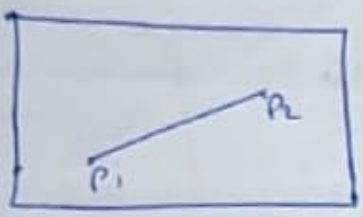


Line clipping (Cohen Sutherland)

T B R L
↓ ↓
Top Bottom



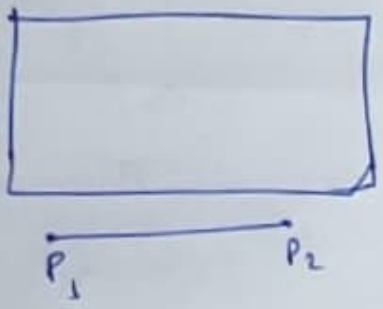
Case I :-



$P_1 = 0000$
$P_2 = 0000$

No clipping required

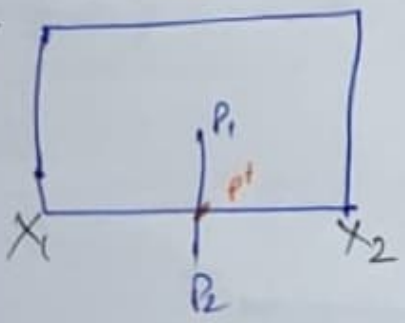
Case II :-



$P_1 = 0100$
$P_2 = 0100$
0100 AND

Line is removed

Case III :-



$P_1 = 0000$
$P_2 = 0100$
0000

Line clipping is required

P' = is end point

(P_2 is removed to P')

Problem with ~~boundary~~ 4 connected boundary fill algorithm

— In this approach for some particular category of polygons the diagonals pixels are left uncoloured

→ It may not filled Regions some time correctly when ~~that~~ some interior pixel ~~is not~~ is already filled with colo. The algorithm will checked of boundary pixel for filling & will found already filled so recursive process is terminate. So there is need to check all pixel colour before applying the colours

Imp

Flood fill → Flood fill required use amount of memory.
→ Flood fill is simple and efficient

→ It can process the image containing more than boundary colours
→ Flood fill algo. compartively slower than boundary fill algo.
→ In flood fill the random colour can be used to paint the interior portion than the old one is replaced with the new one

Boundary fill

→ It can only process the image containing single boundary colours
→ It is faster than flood fill Algo.
→ Boundary fill interior points are painted by continously searching for boundary colour
→ Memory consumption is relatively low
→ boundary fill have high complexity

Algorithm Flood fill

n = new colour
o = old colour

```
Flood (x, y, n, o)
{
  if get pixel (x,y) == o
  {
    put pixel (x, y, n)
    {
      Flood (x+1, y, n, o);
      Flood (x, y+1, n, o);
      Flood (x-1, y, n, o);
      Flood (x, y-1, n, o);
      Flood (x-1, y-1, n, o);
      Flood (x-1, y+1, n, o);
      Flood (x+1, y-1, n, o);
      Flood (x+1, y+1, n, o);
    }
  }
}
```

4 connector

8 connector

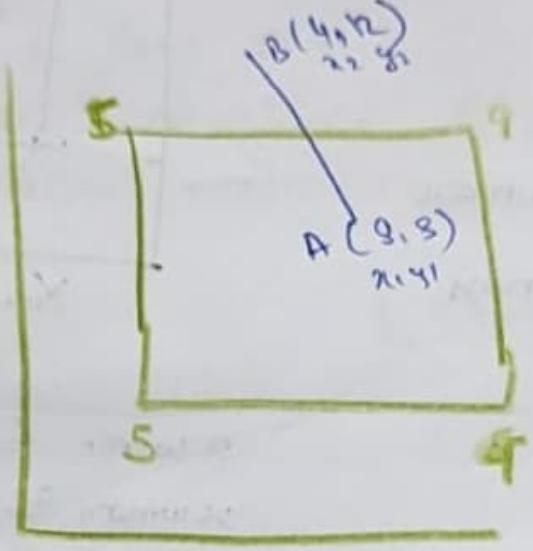
Disadvantages

→ very slow algorithm.
→ may be fail for large polygons / complex ploy
→ initial pixel required more knowledge abo
surrounding pixel.

If $t_1 > t_2$ (x)

If $t_1 < t_2$ then move to step 1

Ques



$X_{wmin} = 5$ $X_{wmin} = 5$

$X_{wmax} = 9$ $X_{wmax} = 9$

$p_1 = -4$ $q_1 = -1$

$p_2 = 4$ $q_2 = 5$

$p_3 = -4$ $q_3 = 7$

$p_4 = 4$ $q_4 = 1$

$t_1 = \max\left(0, \frac{q_k}{p_k}\right)$ $\max(0, 0.25)$

$t_2 = \min\left(1, \frac{q_k}{p_k}\right)$ $\min(1, 0.25)$

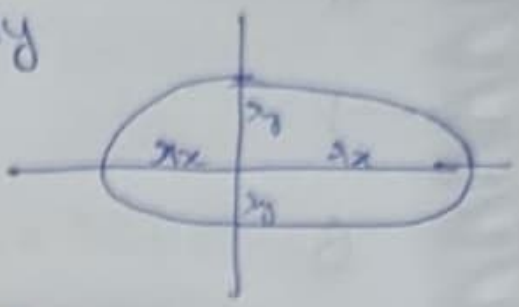
Algorithm

```
boundary (x, y, f, b)
{
    if (get pixel (x,y) != b
        &&
        get pixel (x,y) != f )
    {
      Put pixel (x, y, f )
      boundary (x+1, y, f, b);
      boundary ( x, y+1, f, b);
      boundary (x-1, y, f, b);
      boundary (x, y-1, f, b);

      boundary (x-1, y-1, f, b);
      boundary (x-1, y+1, f, b);
      boundary (x+1, y-1, f, b);
      boundary (x+1, y+1, f, b);
```

Ellipse Generation Algorithm (mid-point)

→ elongated circle
→ 2 radius r_x and r_y
→ 2 axis



- Major (Horizontal) $2r_x$
- minor (Vertical) $2r_y$

circle	ellipse
→ 8 ways symmetry	4 ways symmetry
→ 1 octant → 7 other octant	2 region → 1 Quadrant

Intersecting tangent slope $= -1$

eq. of ellipse $= x^2 r_y^2 + y^2 r_x^2 - r_x^2 r_y^2 = 0$

where
$a = r_x$
$b = r_y$

slope $= \frac{dy}{dx}$ (eq^n of ellipse)

$= \frac{-2r_y^2 x}{2r_x^2 y}$

P_k	x_i	y_i	x_n	y_n
6	20	10	21	11
-2	21	11	22	12
+14	22	12	23	12
+10	23	12	24	13
6	24	13	25	14
2	25	14	26	15
-2	26	15	27	16
+14	27	16	28	16
10	28	16	29	17
	29	17	30	18

A(-20, -10) B(30, 18)

Advantages

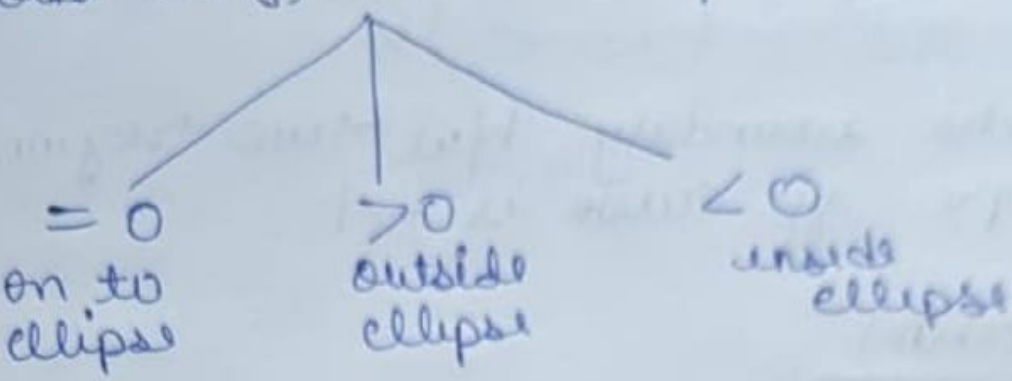
→ It involves only integer arithmatic so it is simple

→ It avoids the generation of duplicates points

→ It can be implemented using hardware because it does not use multiplication & division

→ It is faster as compare to DDA because it does not involve floating point calculation like DDA algorithm

If a point (x, y) enters in eq then



→ ellipse is an elongated circle with two radius ~~x~~ rx and ry and two axis major axis and minor axis

→ The difference b/w circle and ellipse are

* In circle we need to plot only 1 octant of any ~~coord~~ quadrant but in ellipse we need to plot 2 region that is (i.e) one complete ~~coordi~~ quad. entire ellipse

⟹ Quadrant 1 - Region 1
slope < -1
Start point (0, ry)
eg (x+1)

Take unit step in +ve x direction till boundary b/w region is reached

⟹ Quadrant 2 :- Region 2
slope > -1
Take unit step in negative direction till end of quadrant
eg (y-1)

Disadvantages

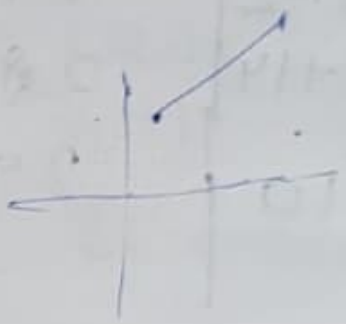
→ This algorithm is meant for basic line drawing only initially is not a part of Bresenham's line algorithm so to draw smooth line you should work to look for another algorithm.

Ques (1,4) & (8,12)

$\Delta x = 8-1$ $\quad \Delta y = 12-4$

$= 7$ $\quad \Delta y = 8$

$m = \frac{8}{7}$



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

$= 2(8) - 7$

$= 16 - 7$

$= 9$

case I

$P_k > 0$

$x_n = x_{(+1)} = 1+1 = 2$

$y_n = y_{1+1} = 4+1 = 5$

$P_{k+1} = P_1 + 2$

$P_{k1} = 9 + 2(8) - 2(7)$

$= 9 + 16 - 14$

$= 9 + 2$

11

3 possibilities for the line

1) line can be completely inside the window [which should be accepted]
2) Line can be completely outside the window [this line will be completely removed from the region
3) Line can be partially inside and partially outsi then we will find the intersection point of the line with the edge of the clipping window

Algorithm

1) Assign a region code for each end point.
2) If both end point have the region code 0000 the ~~exp~~ ~~accepted~~ accept the line.
3) Else perform the logical AND operations for both the region code

* If the result is not 0000 then reject the line.
* If the result is 0000 then you need line clipping

How to perform clipping

① Choose an end point of the line that is outsi the window
② Find the intersection point at the window boundary (based on region code)
③ Replace end point with the intersection point and update the region code

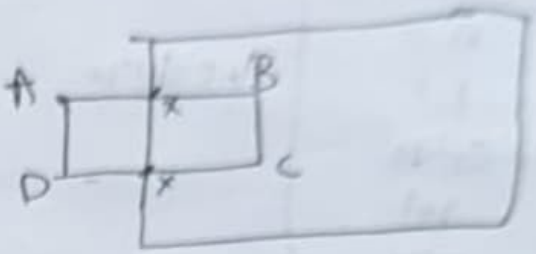
Satherland Hodgeman Polygon Clipping

case 1 : O → I
(XB) save

case 2 :- I → I
(C) save

case 3 : I → O
(Y) save

case 4 : O → O
Ignore



Left

CasI : O → I
(A'B) save

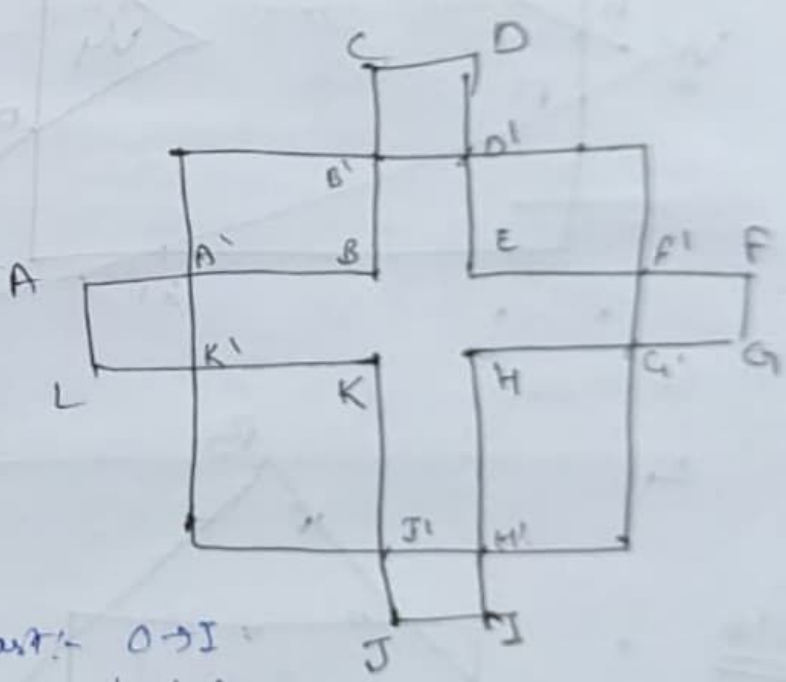
Cas 2 :- I → O
(B') save

Cas 3 :- O → O
Ignore

Cas 4 : O → I
(D'E)

Cas 5 :- I → O
(F') save

Cas 6 :- O → O
Ignore



Cas 7 :- O → I
(G'H)

Cas 8 : I → O
(H')

Cas 9 :- O → O
Ign

Cas 10 :- O → I
(J'H)

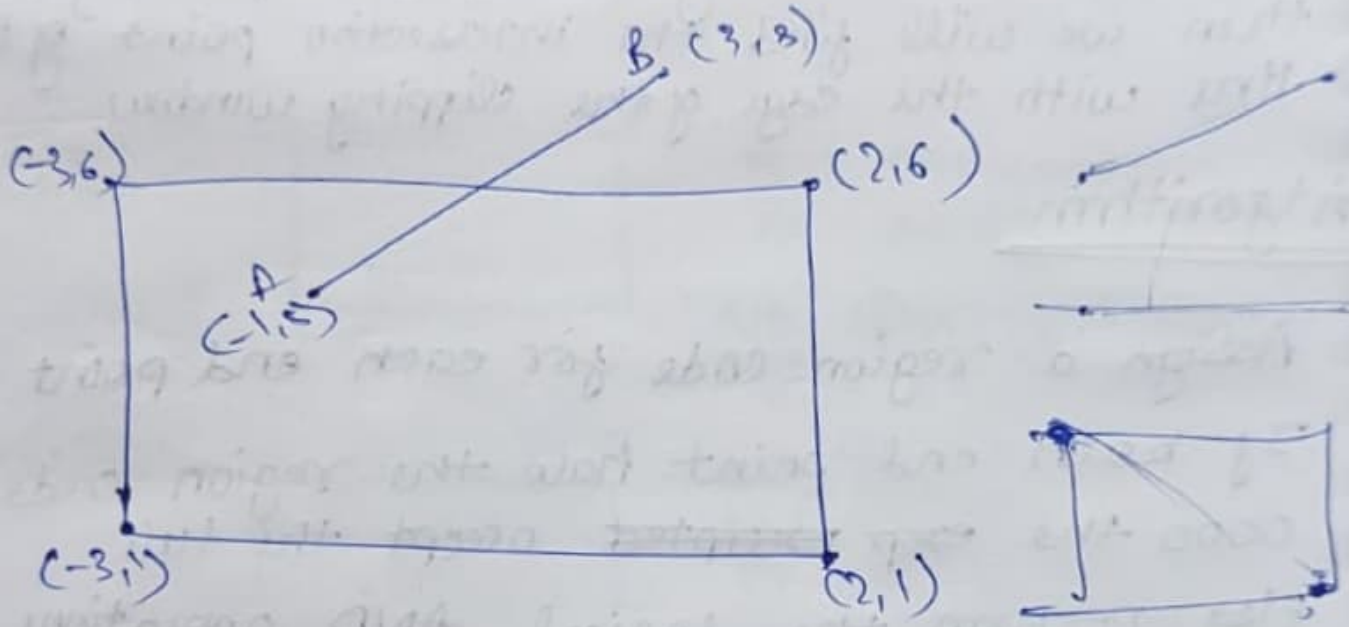
Cas 11 :- I → O
K'

Cas 12 :- O → O
Ignore

④ Repeat step 2 until we find the clipped line either trivially accepted or trivially rejected

⑤ Repeat step 1 for all the lines

Ques Clip a line A(-1,5) and B(3,8) using cohen Sutherland Algorithm with window coordinate (-3,1) and (2,6)

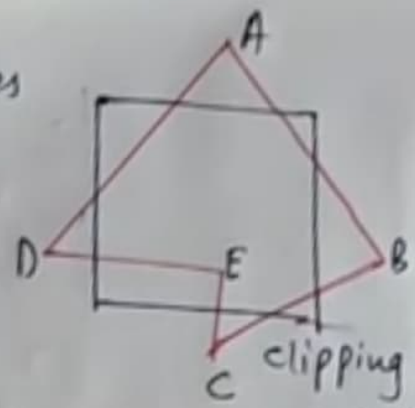


A = 0000

B = 1000

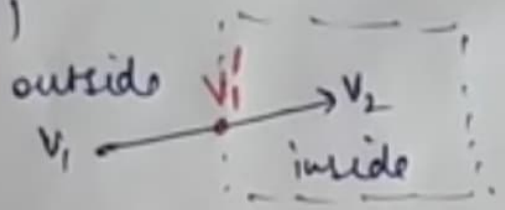
SUTHERLAND HODGMAN POLYGON CLIPPING ALGO

* It clips the region of the polygon lying outside the window.
* clip against each edge of window & obtain new set of vertices



for finding new sequence of vertices, there are 4 cases:

(i)



out → in

o/p → intersect point + destination point

→ $V_1' V_2$

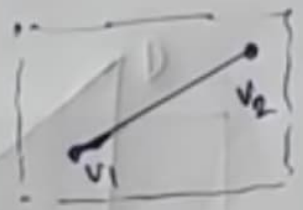
(ii)

movement → (in –

o/p → intersection po

V_1'

(iii)



movement → in to in

o/p → destination vertex

(iv)

movement → out to ou

output → Nothing / Null

Steps of Sutherland Hodgeman Polygon Clipping Algorithm:

1. Read the coordinates of all vertices of the polygon.
2. Read the coordinates of the clipping window.
3. Consider the left edge of the window.
4. Compare the vertices of the edge of the polygon, individually with the clipping plane.
5. Save the resulting intersections and the vertices according to the rule
6. Repeat step (4), (5) for the remaining edges of the clipping window
7. Each time pass the resultant list of vertices to the next edge of the clipping window.
8. STOP

Algorithm:

(1) Input four clippers as $C_l = x_{min}$, $C_r = x_{max}$, $C_t = y_{max}$, $C_b = y_{min}$ corresponding to the left, right, top, and bottom window boundaries. The polygon is specified in term of its vertex list $v_{in} = \{v_1, v_2, \ldots v_n\}$ where the vertices are named anticlockwise

(2) For each clipper in the order C_l, C_r, C_t, C_b do

(3) Set output vertex list $V_{out} = NULL$, $i = 1$, $j = 1$

(4) repeat

(5) Consider the vertex pair v_i & v_j in v_{in}

(6) if v_i is inside and v_j outside then
ADD the intersection of the clipper with the edge (to

(7) else
inside the clipper then

(8) ADD v_j to V_{out}

(9) else if v_i is outside & v_j inside of the clipper then

(10) ADD the intersection point of the clipper with the edge (v_i, v_j) and v_j to V_{out}

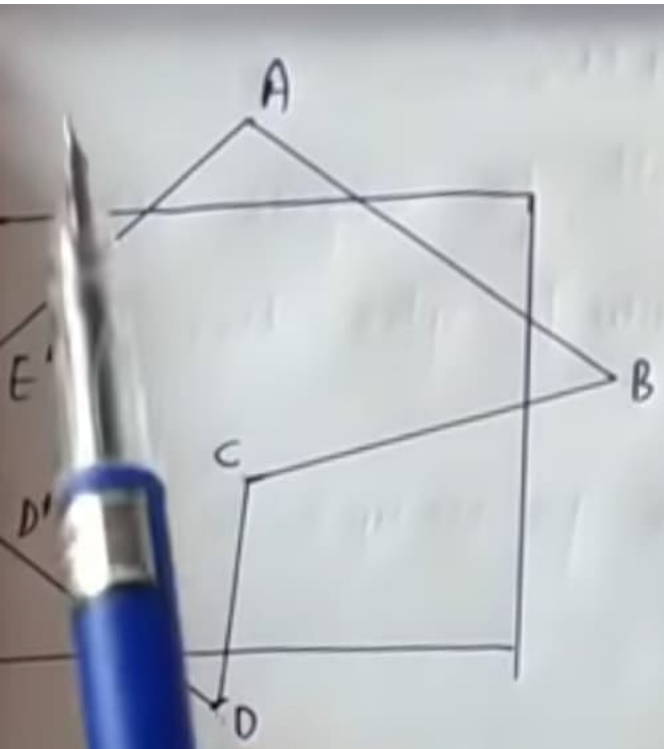
(11) else
ADD NULL to V_{out}

(12) end if

(13) Until all edge (ie consecutive vertex pairs) in v_{in} are checked.

(14) Set $v_{in} = V_{out}$

Vout



① clipping against left edge of the window

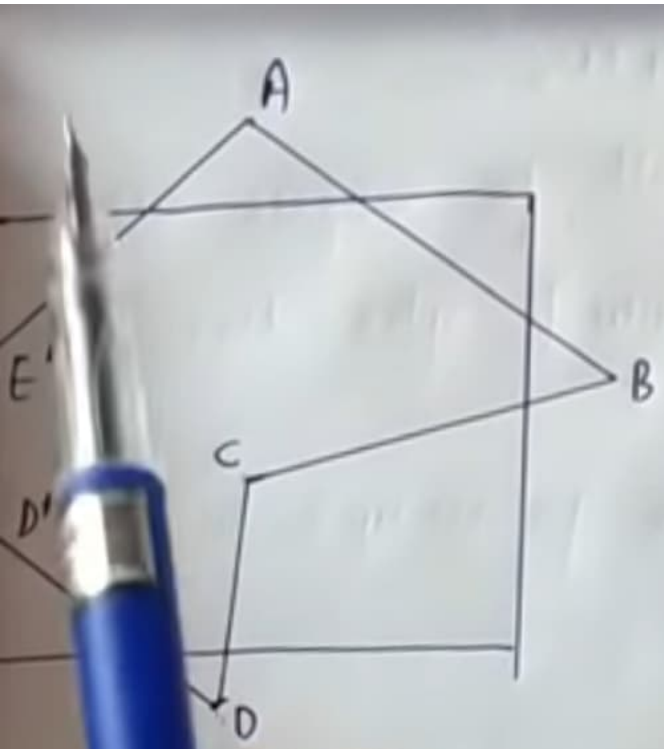
vertex	case	output	
AB	in→in	B	
Bc	in→in	C	
CD	in→in	D	
DE	in→out	D'	} new vertices
EA	out→in	E'A	

② clipping against right edge of window

vertex	case	OIP	
AB	in→out	A'	} new vertex
Bc	out→in	B'c	
cD	in→in	D	
DD'	in→in	D'	
D'E'	in→in	E'	
E'A	in→in	A	

OIP	
A''A'	→ new vertex
B'	
c	
'	

B'c
cc'
c'



① clipping against left edge of the window

vertex	case	output	
AB	in→in	B	
Bc	in→in	C	
CD	in→in	D	
DE	in→out	D'	} new vertices
EA	out→in	E'A	

② clipping against right edge of window

vertex	case	OIP	
AB	in→out	A'	} new vertex
Bc	out→in	B'c	
cD	in→in	D	
DD'	in→in	D'	
D'E'	in→in	E'	
E'A	in→in	A	

OIP	
A''A'	→ new vertex
B'	
c	
'	

B'c
cc'
c'

2) Connectivity / Adjacency

Two pixels that are neighbours and have the same gray level are adjacent.

a) 4-adjacency

Binary image

0	1	0	1
0	0	1	0
0	0	1	0
1	0	0	0

$V = \{1\}$

b) 8-adjacency

Gray-scale image

54	10	100	8
81	150	2	34
201	200	3	45
7	70	147	56

$V = \{1, 2, 3, \ldots 10\}$

c) $N_8(p)$ 8-neighbours : union of 4-neighbours and diagonal neighbours

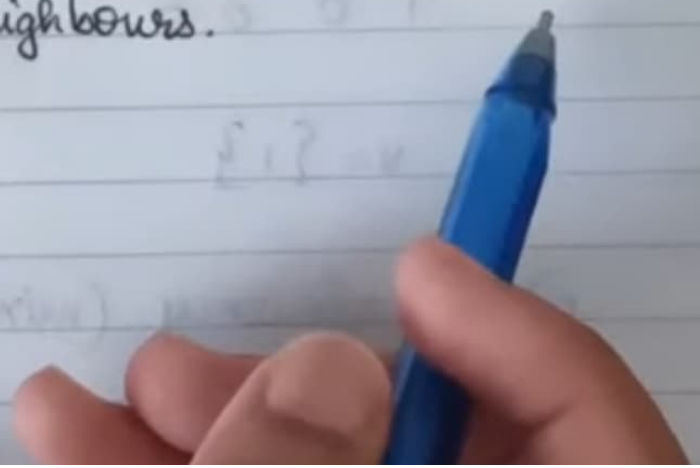
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$

a) $N_4(p)$ 4-neighbours : the set of horizontal and vertical neighbours

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

b) $N_D(p)$ diagonal neighbours : the set of 4 diagonal neighbours.

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



SR. NO.	BASIS	BEAM PENETRATION	SHADOW MASK
1.	Colors produced	In this method, there is the production of only four colors i.e., red, green, yellow, orange.	In this method, there is the production of millions of colors.
2.	Color dependency	As in this method only four colors are produced it is because of the speed of the electron gun.	As in this method millions of colors are produced because it depends upon the intensity value of the three available guns.
3.	Number of electron guns used.	In this method, only one electron gun is used.	In this methods, three electron guns are used; i.e red, green and blue.
4.	Picture quality	As we know in this different colors and shades are not possible. So, it's picture quality is poor.	As we know in this different colors and shades are possible. So, it's picture quality is quite good.
5.	Realistic view	This method is not suitable for providing the realistic view.	This method is suitable for providing the realistic view.
6.	Resolution	This method provides high resolution.	Whereas, this method does not able to provide high resolution.
7.	Cost	It is cheaper than shadow mask method.	It is an expensive method.
8.	Application	It is used in random scan system to display color.	It is used in raster scan system to display color.

Color CRT Monitor:

- Color CRT Monitors:
- The CRT Monitor display by using a combination of phosphors. The phosphors are different colors.
- This was one the earlier CRTs to produce color displays. Coating phosphors of different compounds can produce different colored pictures. But the basic problem of graphics is not to produce a picture of a predetermined color, but to produce color pictures, with the color characteristics chosen at run time.
- The basic principle behind colored displays is that combining the 3 basic colors – Red, Blue and Green, can produce every color. By choosing different ratios of these three colors we can produce different colors – millions of them in-fact. We also have basic phosphors, which can produce these basic colors. So, one should have a technology to combine them in different combinations.
- There are two popular approaches for producing color displays with a CRT are:
- Beam Penetration Method
- Shadow-Mask Method

Random Scan System

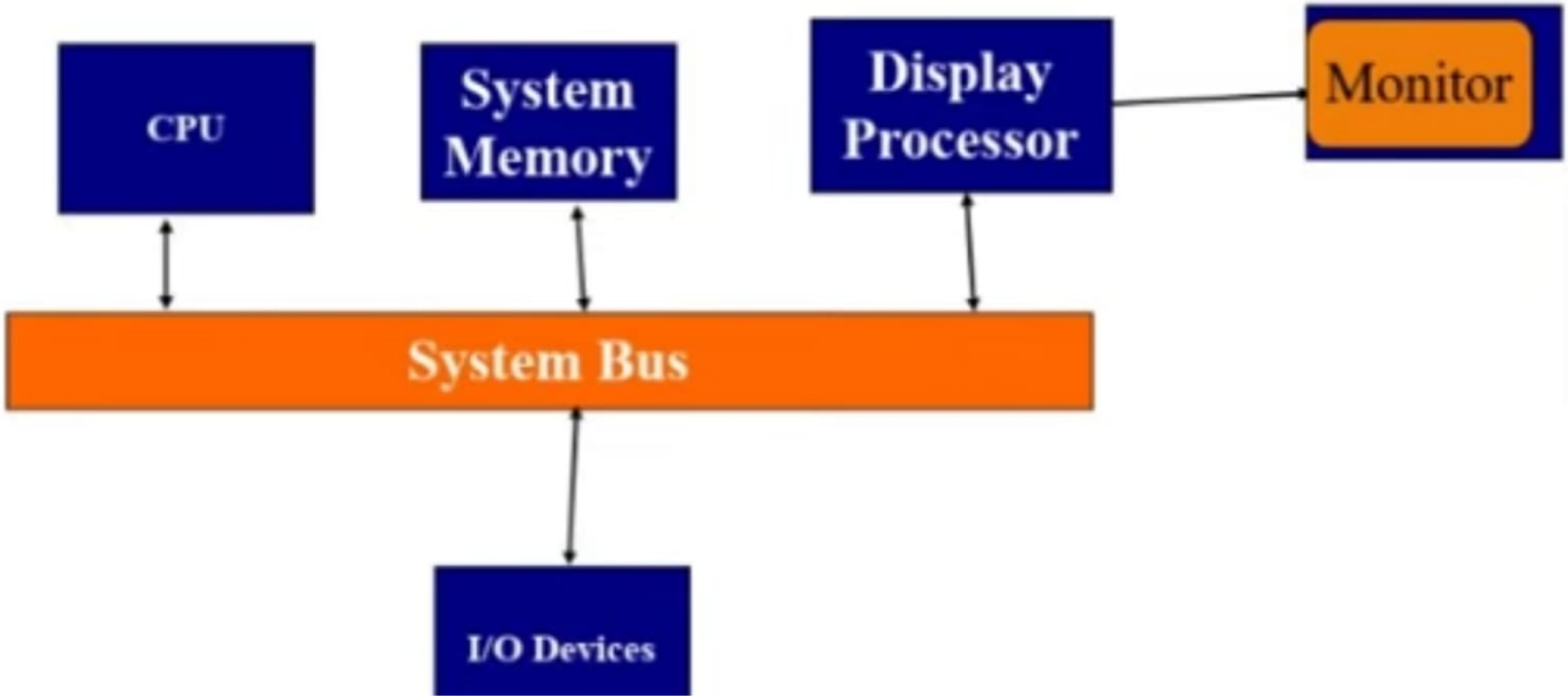
- **Advantages:**

- High Resolution
- Draw smooth line Drawing
- Less Memory Required

- **Disadvantages:**

- Realistic images with different shades cannot be drawn.
- Colour limitations.

- Figure: Graphics System



Raster Scan System

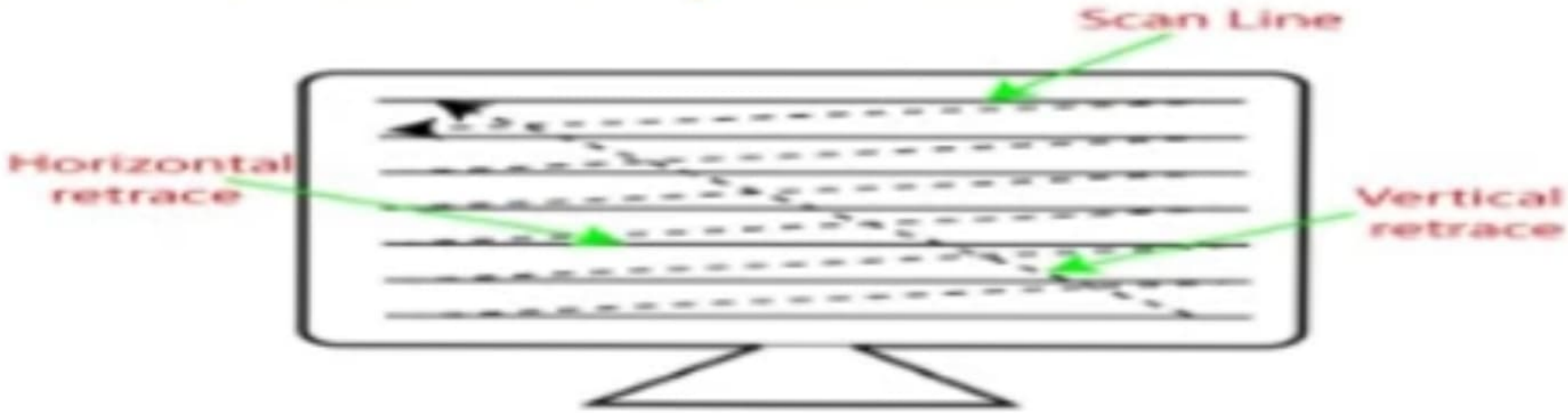
- **Advantages:**

- Real image
- Many colors to be produced
- Dark scenes can be pictured

- **Disadvantages:**

- Less resolution
- Display picture line by line
- More costly

Raster Scan System



Here, the beam is swept back and forth from the left to the right across the screen.
When the beam is moved from left to the right, it is ON.
The beam is OFF, when it is moved from the right to the left as shown by dotted line in figure.
When the beam reaches the bottom of the screen, it is made OFF and rapidly retracted back to the top left to start again.
A display produced in this way is called as raster scan.
In raster scan display a special area of memory is dedicated to graphics only. This memory area is called frame buffer.
This frame buffer stores the intensity values for all the screen points. Each screen point is called a pixel (picture element).

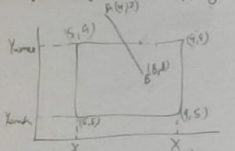
Raster Scan System

- **Raster Scan:** It is a scanning technique in which the electron beam moves along the screen. It moves from top to bottom, covering one line at a time.
- A raster scan is based on pixel intensity control display as a rectangular box on the screen called a **raster**.
- Picture description is stored in the memory area called as **Refresh buffer, or Frame Buffer.**
- Frame buffer is also known as **Raster or Bitmap.** Raster scan provides the refresh rate of 60 to 80 frames per second.

- **For Example: Television**

- The beam refreshing has two types:
- 1.Horizontal Retracing
- 2.Vertical Retracing

Liang Barsky Line Clipping



① $x = x_1 + t\Delta x$
$y = y_1 + t\Delta y$

② $X_{wmin} \le x \le X_{wmax}$ — ①
$Y_{wmin} \le y \le Y_{wmax}$ — ②

③ $tp_k \le q_k$

$p_1 = -\Delta x$ $q_1 = x_1 - X_{wmin}$
$p_2 = \Delta x$ $q_2 = X_{wmax} - x_1$
$p_3 = -\Delta y$ $q_3 = y_1 - y_{wmin}$
$p_4 = \Delta y$ $q_4 = Y_{wmax} - y_1$

4 - 5
9 - 4
12 - 5
9 - 12

B' (x, y)

④ If $p_k = 0$
then line is parallel to window
If $q_k < 0$
then line is completely outside
If $p_k < 0$ $t_1 = \max\left(0, \frac{q_k}{p_k}\right)$
If $p_k > 0$ $t_2 = \min\left(0, \frac{q_k}{p_k}\right)$
If $t_1 > t_2$ then terminate

$y \ge 2$
$-y \le -2$

$X_{wmin} \le x \Rightarrow X_{wmin} \le x_1 + t\Delta x \Rightarrow -(t\Delta x \ge X_{wmin} - x_1) \Rightarrow -t\Delta x \le x_1 - X_{wmin}$

$X_{wmax} \ge x \Rightarrow X_{wmax} \ge x_1 + t\Delta x \Rightarrow t\Delta x \le X_{wmax} - x_1$

$Y_{wmin} \le y$

$Y_{wmax} \ge y$

$tp_k \le q_k$

$p_1 = -4$	$p_2 = 4$	$p_3 = 4$	$p_4 = -4$
$q_1 = -1$	$q_2 = 5$	$q_3 = 2$	$q_4 = 3$

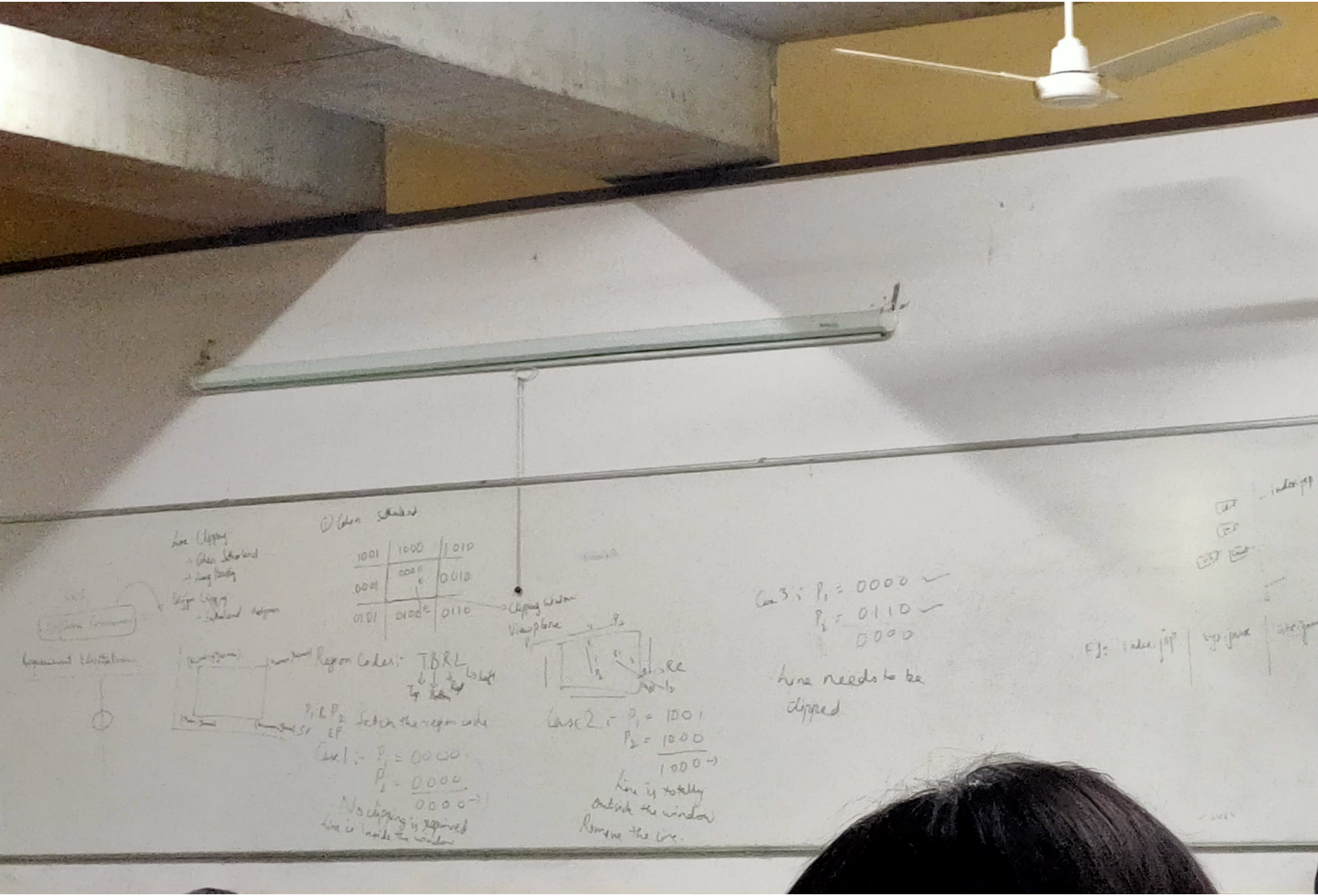
$p_1 = -\Delta x$
$q_1 = x_1 - X_{wmin}$
$p_2 = \Delta x$
$q_2 = X_{wmax} - x_1$

If $t_1 < t_2$ then move to step 5

$t_1 = \max\left(0, \frac{2}{-8}\right) - 0$

$t_2 =$

$\frac{-4+4+4-4}{4}$ $t_1 = 1.7$
$- 0$ $t_2 =$


1001
1000
1010
0001
0000
0010
0101
0100
0110
Region Codes: TBRL
Case1 :- P1 = 0000
P2 = 0000
0000
No clipping is required
Line is inside the window
Case2 :- P1 = 1001
P2 = 1000
1000
Line is totally outside the window
Remove the line.
Case3 :- P1 = 0000
P2 = 0110
0000
Line needs to be clipped