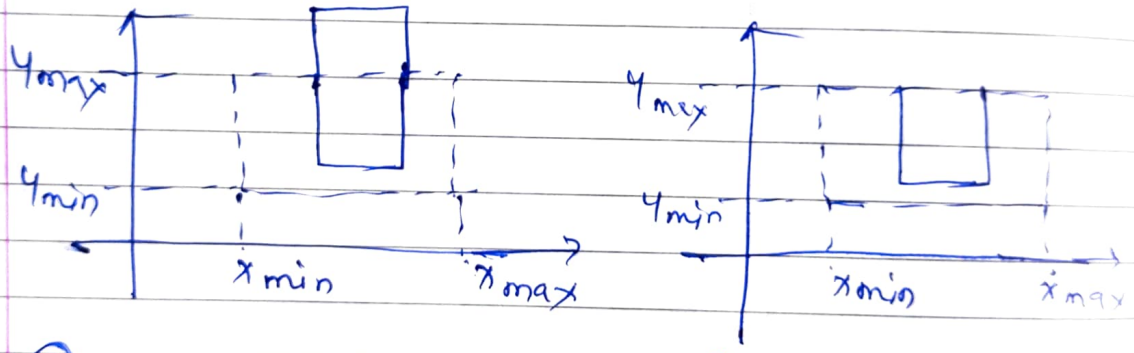
Chap 4: 2D viewing & clipping

* 2D viewing

- Graphics packages allow the user to specify which part of the scene is to displayed

- process of selecting the part of the real-world scene to display it on devices called windowing.

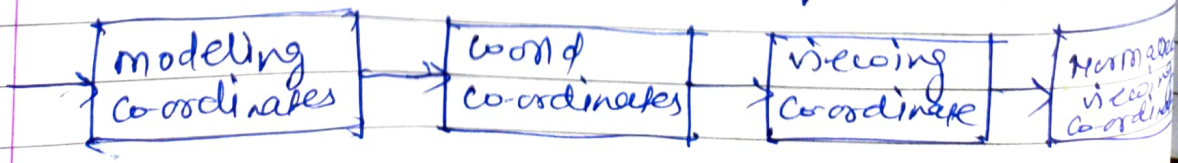
- clipping is the process of deciding & removing the portion of the obj which is outside the clipping window.



(a) word co-ordinates (b) viewport co-ordinates

The process of mapping the part of world co-ordinate scene to device co-ordinate is called viewing transformation or windowing transformation

* window to viewport Transformation.



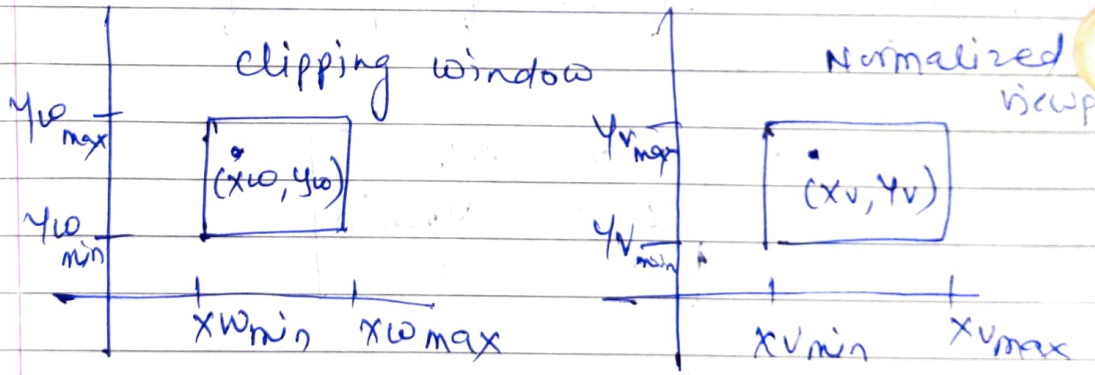
* window to viewport co-ordinate transformation

- window to the viewport transformation is necessary because the size of the window & viewport may not be the same all the time.

So actual picture selected by window needs to be rescaled to fit it in the viewport.

Let (xw_{min}, yw_{min}) & (xw_{max}, yw_{max}) point of clipping window.

- (xv_{min}, yv_{min}) & (xv_{max}, yw_{max}) point of viewport.



$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$xv - xv_{min} = (xv_{max} - xv_{min}) \cdot \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$xv = xv_{min} + (xw - xw_{min}) \cdot sx$$

Similary for sy.

where the scaling factors are -

$$S_x = \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}$$

$$S_y = \frac{YV_{max} - YV_{min}}{YV_{max} - YW_{min}}$$

Matrix

If the lower left corner of the window is not at the origin, we translate it to the origin before we map the point in window to viewport.

(1) Translate lower-left point of window to the origin.

$$T = \begin{bmatrix} 1 & 0 & -XW_{min} \\ 0 & 1 & -YW_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

(2) Apply scaling to viewport.

$$S = \begin{bmatrix} \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}} & 0 & 0 \\ 0 & \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(3) Inverse translation in the viewport.

$$T^{-1} = \begin{bmatrix} 1 & 0 & x_{Vmin} \\ 0 & 1 & y_{Vmin} \\ 0 & 0 & 1 \end{bmatrix}$$

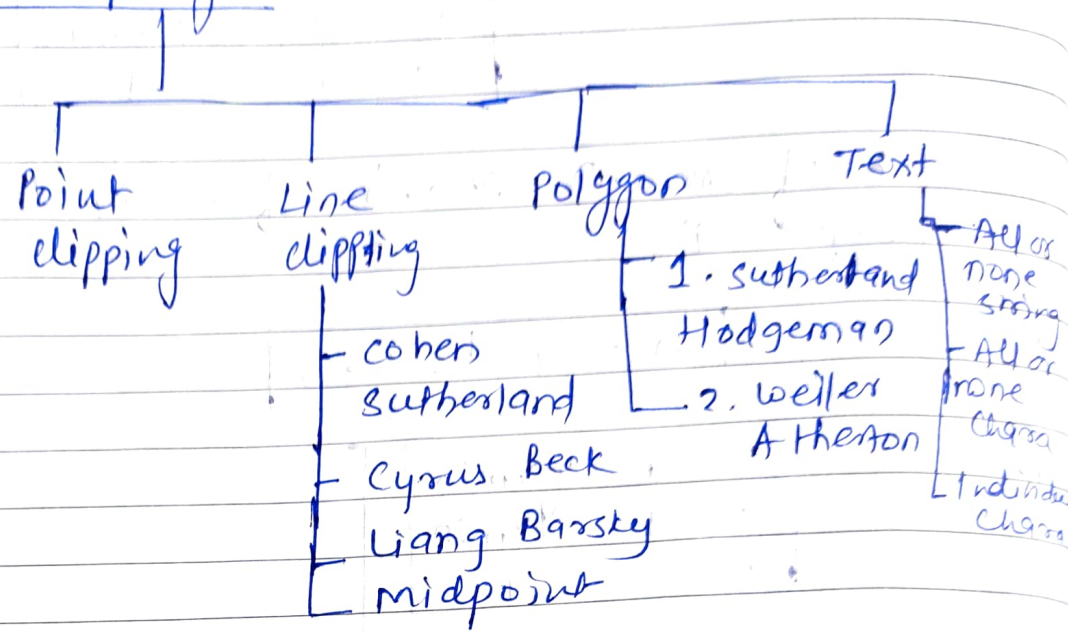
Transformation Matrix for window to viewport is -

$$\boxed{M = T^{-1} \cdot S \cdot T}$$

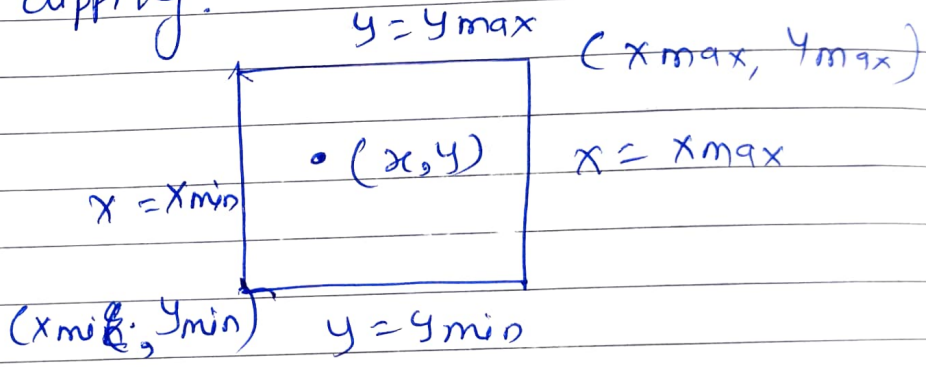
$$= \begin{bmatrix} 1 & 0 & x_{Vmin} \\ 0 & 1 & y_{Vmin} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

* 2D clipping

-



① Point clipping.



- The point can be either fully inside or fully outside the clipping region.

- ~~The point can be either fully inside or fully outside the clipping region.~~

The point on the clipping window boundary is considered inside.

Let (x_{min}, y_{min}) & (x_{max}, y_{max}) represents the lower-left & top-right corner of the clipping window.

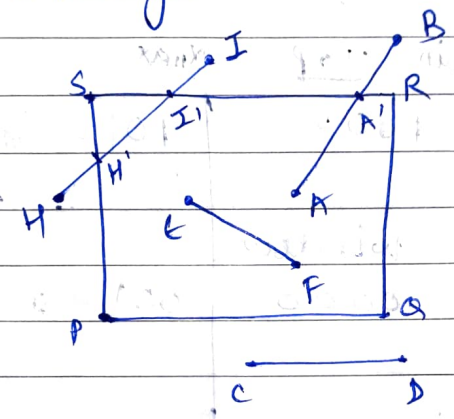
Point (x, y) is inside the region only if the following four inequalities are true,

$$x_{min} \leq x \leq x_{max}$$

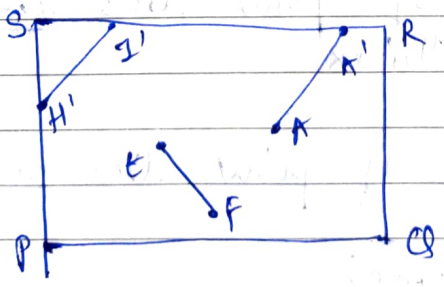
$$y_{min} \leq y \leq y_{max}$$

If any of the four inequalities does not hold, the point is outside the clipping rectangle.

(2) Line clipping

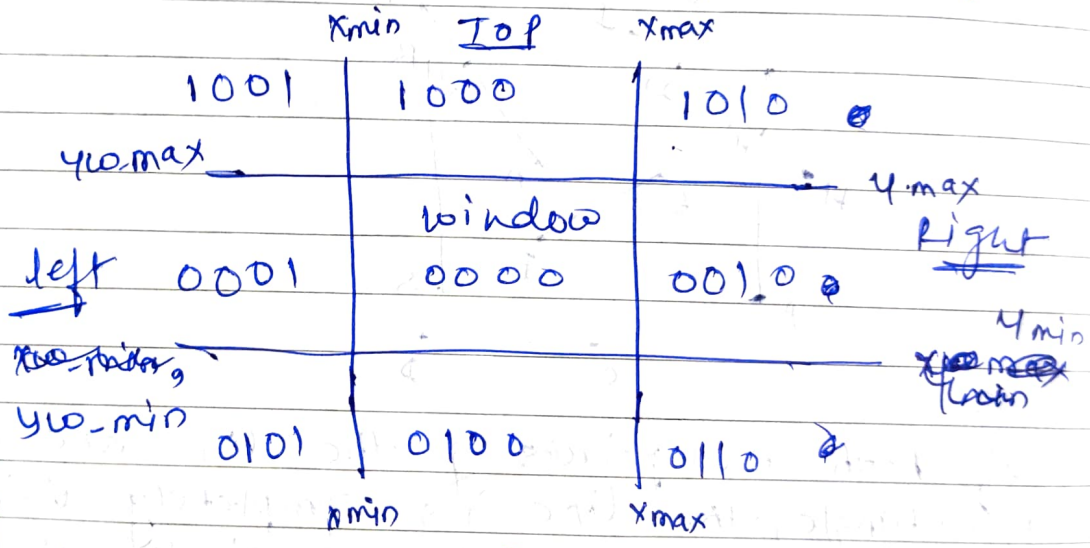


- If both end points of the line are within a rectangle, the line is completely visible. otherwise line may be partially visible or completely outside the window.



Cohen-Sutherland Line clipping Algorithm

- In this algo, we are given 9 regions on the screen out of which one region is of the window & the rest 8 regions are around it given by 4 digit binary
- The division of regions are based on (x_max, y_max) & (x_min, y_min)
- The central part is the viewing region or window. All the lines which lie within this region are completely visible



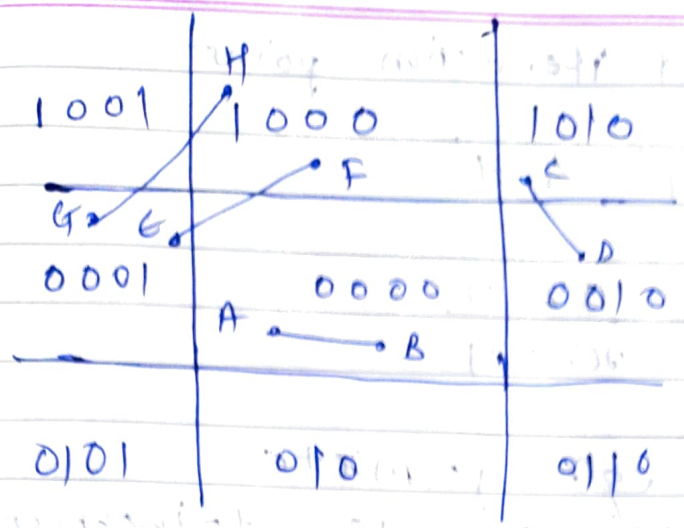
① Set first bit if point lies above window i.e. $y > y_{max}$

② Set second bit if point lies below $y < y_{min}$

③ Right $x > x_{max}$

④ Left $x < x_{min}$

TBRL code.



① If both endpoints of a line inside window line is visible. i.e. 0000 logical OR-AND conditions are evaluated.

② If logical OR of the region code is not 0000 there are two possibilities, the line may be partially visible or it may be completely outside.

if $x < x_{min}$ → left of window

$x > x_{max}$ → Right

$y < y_{min}$ - Bottom

$y > y_{max}$ - Top.

Ex:- Line GH

(0001) . (1000)

G H

0 0 0 1

1 0 0 0

0 0 0 0

Perform logical AND

If result ≠ 0000 line is outside

line partially invisible

- Find intersection point

G' & H'

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

line eqn $y = mx + c$
we can compute y-intercept of line as $c = y - mx$

- y-coordinate of an intersection with a vertical window boundary can be cal as
$y = mx + c$ where x is either x_{min} or x_{max} depends on for which vertical line we are cal. y.

$$y = m \cdot x_{min} + c$$
or. $$y = m \cdot x_{max} + c$$

x coordinate of an intersection with a horizontal window boundary is computed as,

$$x = (y - c)/m$$

where y is either y_{min} or y_{max} depends on for which horizontal line we are cal x.

$$x = \frac{y_{min} - c}{m}$$

$$\text{or } x = \frac{y_{max} - c}{m}$$

Algo

1. Read (x_{min}, y_{min}) & (x_{max}, y_{max})
2. Read A (x_1, y_1) & B (x_2, y_2) end pts of line
3. Compute outcode of A & B

if $y_1 > y_{max}$ then outcode A1 = 1 else 0

if $y_1 < y_{min}$ then outcode A2 = 1 else 0

if $x_1 > x_{max}$ then outcode A3 = 1 else 0

if $x_1 < x_{min}$ then outcode A4 = 1 else 0

if $y_2 > y_{max}$ — B1 = 1 else 0

$y_2 < y_{min}$ — B2 = —

$x_2 > x_{max}$ — B3 = —

$x_2 < x_{min}$ — B4 = —

4. If outcode_A OR outcode_B == 0000 then disp entire line & goto Step 5

else

if outcode_A AND outcode_B ≠ 0000 then Reject the entire line

else

compute the intersection point with window boundaries

Repeat step 4

5. Stop

Cohen Sutherland Algo.

1. Cal position of both endpoints of the line.
2. Perform OR operation on both these end points.
3. If OR gives 0000
   Then
   line is visible
   else
   Perform AND operation
   If AND ≠ 0000
   then line is invisible
   else
   AND = 0000
   Line is considered the clipped case
4. If a line is clipped case, find an intersection with boundaries of the window.

$$m = (y_2 - y_1) / (x_2 - x_1)$$

(a) If bit 1 is "1" line intersect with left boundary of rectangle window

$$y_3 = y_1 + m(x - x_1)$$

where $x = x_{wmin}$

(b) If bit 2 is "1" line intersect with Right boundary

$$y_3 = y_1 + m(x - x_1)$$

where $x = x_{wmax}$

(c) If bit 3 is "1" line intersects with bottom boundary

$$x_3 = x_1 + (y - y_1) / m$$

where

$$y = y_{wmin}$$

(d) If bit 4 is "1" line intersects with top boundary

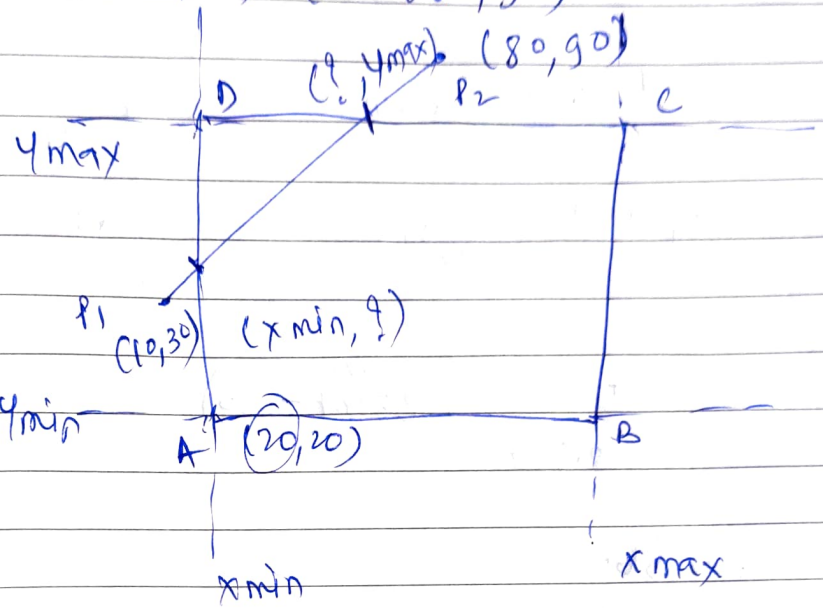
$$x_4 = x_1 + (y - y_1) / m$$

where

$$y = y_{wmax}$$

freedom

Q. Let ABCD be rectangle window with A(20,20) B(90,20) C(90,70) & D(20,70) find region code for the endpoints & use cohen sutherland line clip algo to clip the line P_1P_2 with $P_1(10,30)$ & $P_2(80,90)$



$P_1 = 0001$ TBRL

$P_2 = 1000$

AND 0000

OR 1001

} partially visible.

Slop, $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90-30}{80-10} = \frac{6}{7} = 0.857$

$P_1(10,30)$ $P_2(80,90)$

(x_1, y_1) (x_2, y_2)

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

$= 0.857(20-10) + 30$

$y = 38.57$

1st intersection point

$i_1 = (20, 38.57) \cong (20, 39)$

$(x_{min}, 4)$

2nd intersection.

$x = x_1 + \frac{1}{m}(y_{max} - y_1)$

$= 10 + \frac{1}{0.857}(70 - 30)$

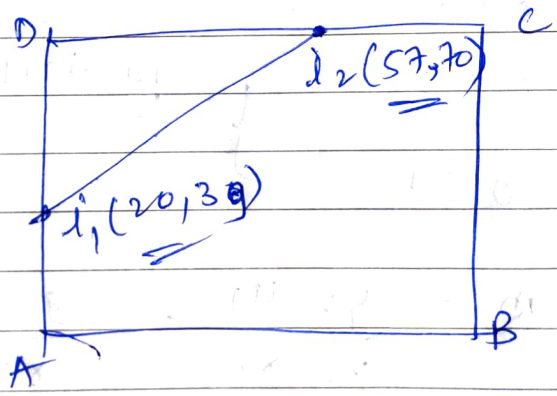
$= 56.67$

$i_2 (56.67, 70) \cong (57, 70)$

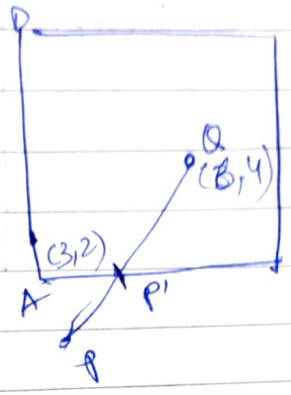
$y = y_1 + m(x_{min} - x_1$

else

$x = x_1 + \frac{1}{m}(y_{max} - y_1$



Q. clip the line PQ, P(4,1) Q(6,4) against the clip window A(3,2) B(7,2) C(7,6) & D(3,6) using cohen sutherland

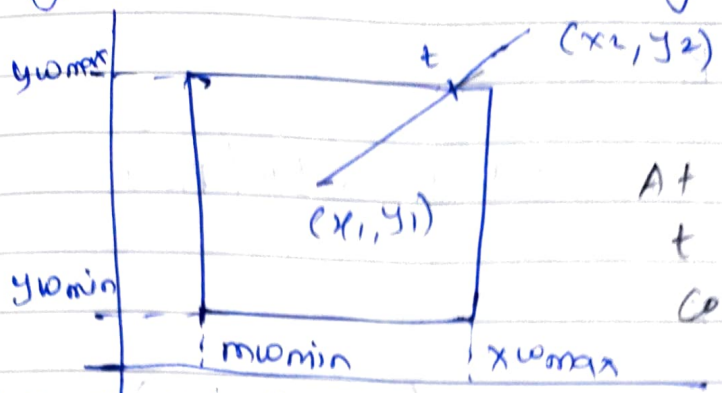


$y = (2, y_{min})$

$x = (4.67, 2)$

↑ new

Liang Barsky Line clipping Algo



At time interval t what are the co-ordinates?

(1) $x = x_1 + t\,\Delta x$

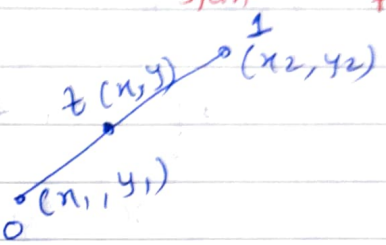
$y = y_1 + t\,\Delta y$

$0 \le t \le 1$

start — final

$x = x_1 + t(x_2 - x_1)$

$y = y_1 + t(y_2 - y_1)$



(2) $xw_{min} <= x_1 + t(x_2 - x_1) <= xw_{max}$

$yw_{min} <= y_1 + t\underbrace{(y_2 - y_1)}_{\Delta y} <= yw_{max}$

$t\Delta x \le xw_{max} - x_1$ $\qquad$ $t\Delta x \ge xw_{min} - x_1$

$t\Delta y \ge yw_{min} - y_1$ $\qquad$ $t\Delta y \le yw_{max} - y_1$

(3) $tp_k <= q_k$ $\qquad$ $\{k = 1, 2, 3, 4\}$

$p_1 = -\Delta x$, $q_1 = x_1 - xw_{min}$ (Left Boundary)

$p_2 = \Delta x$, $q_2 = xw_{max} - x_1$ (Right B)

$p_3 = -\Delta y$, $q_3 = y_1 - yw_{min}$ (Bottom B)

$p_4 = \Delta y$, $q_4 = yw_{max} - y_1$ (Top B)

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

if $q_k < 0$ — Line completely outside.

if $p_k < 0$

$$t_1 = \max(0, -q_k/p_k)$$

if $p_k \geq 0$

$$t_2 = \min(1, q_k/p_k)$$

if $t_1 > t_2$ → Line completely outside & it can be rejected.

if $t_1 < t_2$ then

$$x = x_1 + t\Delta x \qquad x_{wmin}, x_{wmax}$$
$$y = y_1 + t\Delta y \qquad y_{wmin}, y_{wmax}$$

} 4 times

---

condition

1. $p_k = 0$ — parallel to clipping boundary
2. $p_k = 0$ & $q_k < 0$ — completely outside
3. $p_k = 0$ & $q_k >= 0$ — inside the parallel clipping
4. $p_k < 0$ — line proceeds from outside to
5. $p_k > 0$ — line proceeds from inside to outside.

$y = y_1 + t\Delta y$

* Algorithm

1. Read 2 endpoints $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$

2. Read 2 corners (left top & right Bottom) of the clipping window as -

   $(x_{wmin}, y_{wmin}, x_{wmax}, y_{wmax})$

3. cal values of parameter p_i & q_i for $i = 1, 2, 3, 4$

- $p_1 = -dx$ $\quad q_1 = x_1 - x_{wmin}$
- $p_2 = dx$ $\quad q_2 = x_{wmax} - x_1$
- $p_3 = -dy$ $\quad q_3 = y_1 - y_{wmin}$

$p_4 = dy$, $q_4 = y_{max} - y_1$

(4) If $p_i = 0$ line is parallel

If $q_i < 0$ line is completely outside so discard

else

check line is horizontal or vertical

(5) Initialize t_1 & t_2 as $t_1 = 0$ $t_2 = 1$

Conditions.

(6) Parameters t_1 & t_2 can be calculated that define the part of line that lies within the clip rectangle.

when

1. $p_k < 0$, $\max(0, q_k/p_k)$

2. $p_k > 0$, $\min(1, q_k/p_k)$

If $(t_1 > t_2)$ — line completely outside

otherwise the endpoints of the clipped line are calculated from the two values of parameter t.

(7) If $(t_1 < t_2)$

{

$xx_1 = x_1 + t_1 dx$

$xx_2 = x_1 + t_2 dx$

$yy_1 = y_1 + t_1 dy$

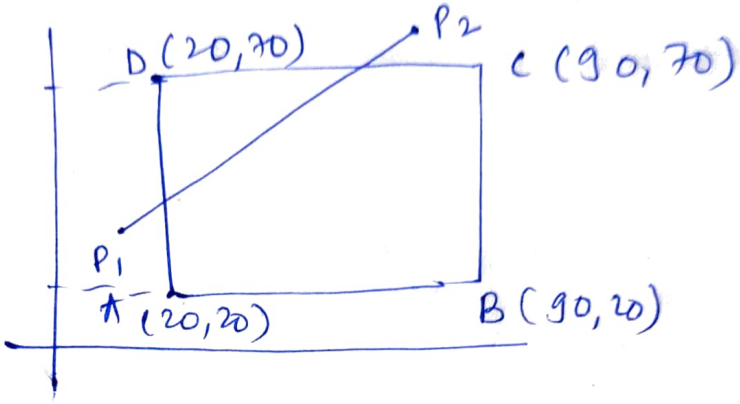
$yy_2 = y_1 + t_2 dy$

line (xx1, yy1, xx2, yy2)

}

(1) window A (20,20) B (90,20), C (90,70
D (20,70) Line P_1 (10,30) P_2 (80,80)

→ $x_{wmin} = 20$ $x_{wmax} = 90$
$y_{wmin} = 20$ $y_{wmax} = 70$



$p_1 = -\Delta x$
$= -(80-10)$
$= -70$

$q_1 = x_1 - x_{wmin}$
$= 10 - 20$
$= -10$

q_1/p_1
$= -70/-70$
$= 1$

$p_2 = \Delta x$
$= 70$

$q_2 = x_{wmax} - x_1$
$= 90 - 10$
$= 80$

q_2/p_2
$= 80$
70

$p_3 = -\Delta y$
$= -60$
$=$

$q_3 = y_1 - y_{min}$
$= 30 - 20$
$= 10$

$p_4 = \Delta y$
$= 60$

$q_4 = y_{wmax} - y_1$
$= 70 - 30$
$= 40$

$(P_K < 0)$

$$t_1 = \max\left(0, \frac{q_1}{p_1}, \frac{q_3}{p_3}\right)$$

$$= \max\left(0, \frac{-10}{-70}, \frac{10}{-60}\right)$$

$$t_1 = 1/7$$

$(P_K > 0)$

$$t_2 = \min\left(1, \frac{q_2}{p_2}, \frac{q_4}{p_4}\right)$$

$$= \left(1, \frac{80}{70}, \frac{40}{60}\right)$$

$$t_2 = 2/3$$

$(t_1 < t_2)$ then find (x, y) for t_1 & t_2 ⇒

$$x = x_1 + t_1 \Delta x$$

$$y = y_1 + t_1 \Delta y$$

$$x = 10 + 1/7 \times 70 = 20$$

$$y = 30 + 1/7\ 60 = 38.57$$

for t_1

A $(20, 38.57)$

$$y = y_1 + t_2 \Delta y$$

$$= 30 + 2/3 \times 60$$

$$= 70$$

$$x = x_1 + t_2 \Delta x$$

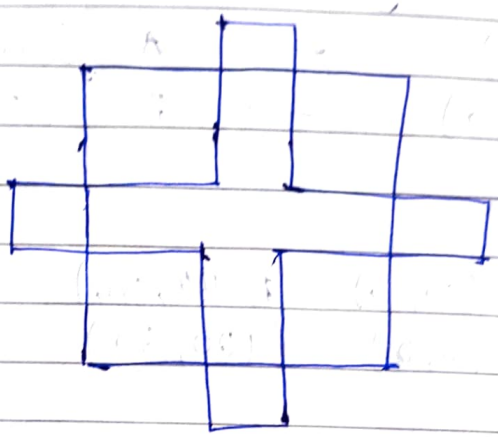
$$= 10 + 2/3 \times 70$$

$$= 56.67$$

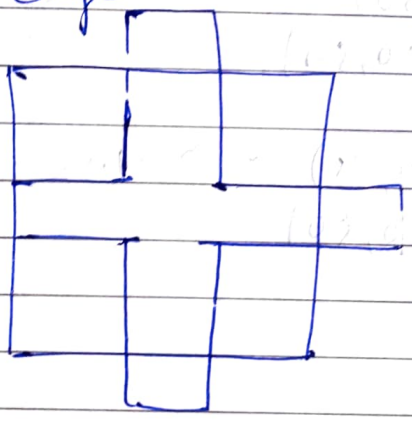
$B = (56.67, 70)$ ⇒

* Polygon clipping

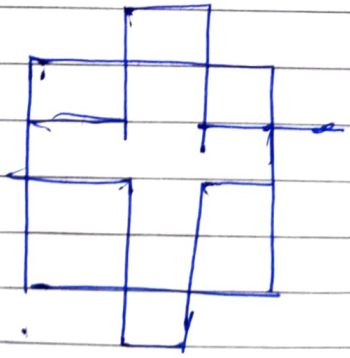
① Sutherland - Hodgeman Polygon clipping



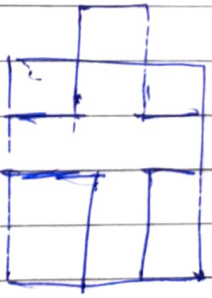
① clip Lef edge



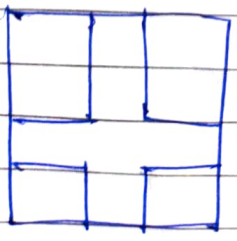
② Right



③ Bottom



④ Top

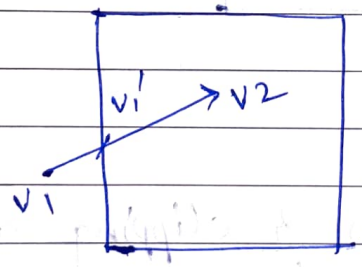


The sutherland Hodgman algo performs a clipping of a polygon against each window edge in turn

(1) clipping against the left side of the clip window
(2) clipping against ~~Right~~ top side
(3) clipping against right side
(4) clipping against bottom side.

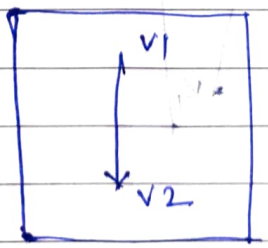
It consist of Four cases:-

1. If the first vertex of the edge is outside of window & second vertex is inside then the intersection point of the polygon edge with the window boundary & the second vertex are added to the O/P vertex list.



save { v_1' & v_2 }

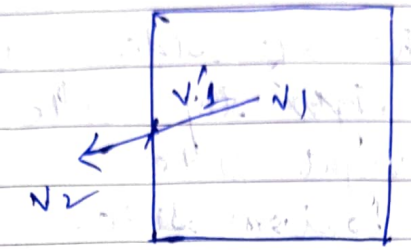
2. If both vertices of the edge are inside the window. Then only second vertex is added to the o/p vertex list.



save { v_2 }

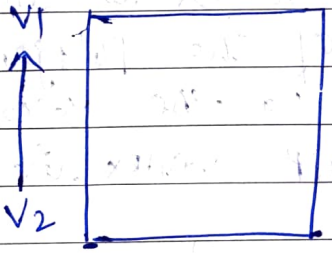
3. If the first vertex of the edge is inside the window & second vertex is outside

then only the intersection point of the polygon edge with the window boundary is added to the o/p vertex list.



save {V1'}

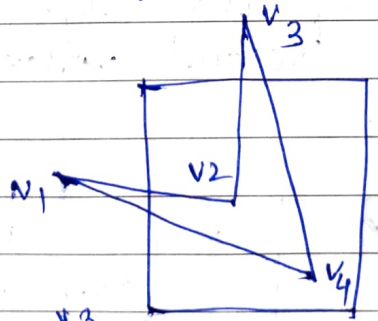
4. If both vertices of the edge are outside the window. Then nothing is added to the o/p vertex list.



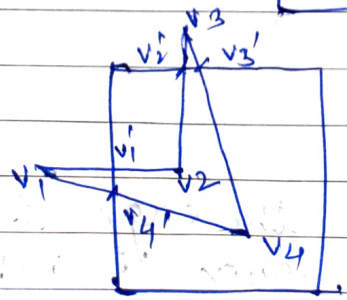
save {ϕ}

Ex:-

Q1. For a polygon & clipping window give the list of vertices after boundary clipping.

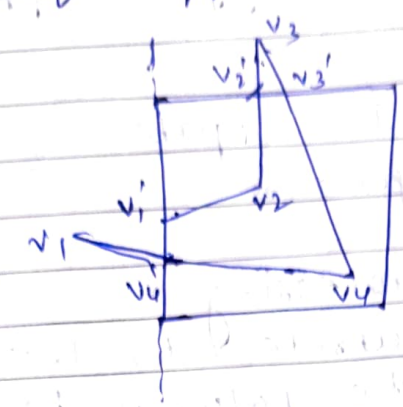


⇒



v_1v_2	out-in	v_1'
$v_1'v_2$	in-in	v_2
v_2v_3	in-out	v_2'

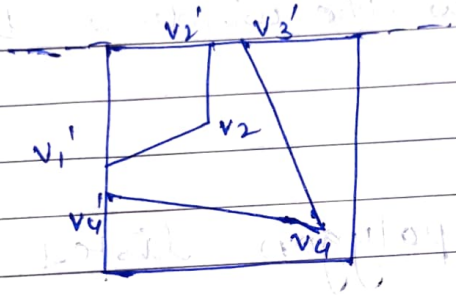
1. Left clipped:



∴ O/P vertex =

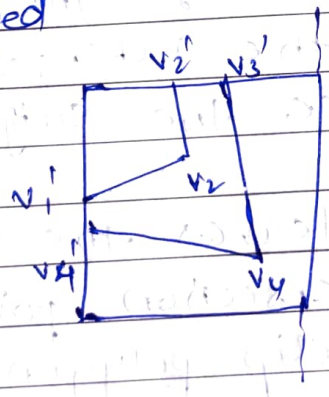
$\{v_1', v_2, v_2', v_3, v_3', v_4, v_4'\}$

3. Top clipped



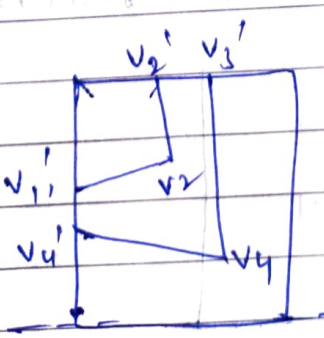
O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

2. Right clipped



O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

4. Bottom clipped



O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

Finally clipped polygon: $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

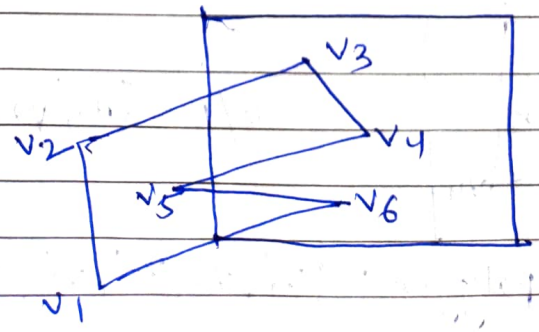
* Weiler - Atherton Polygon clipping Algo

- This Algorithm is capable of clipping of a concave polygon.
- For clockwise processing :
  - For an outside - to - Inside pair of vertices follow the polygon boundary.
  - For an Inside - to - outside pair of vertices, follow the window boundary.

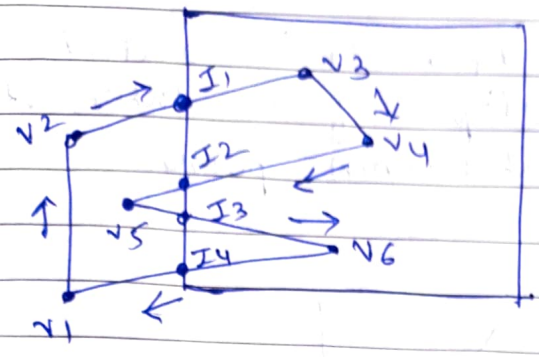
Algorithm :-

- Assume the polygon listed in clockwise order.
- If the edge enter the clip ~~window~~ polygon, record the intersection points & continue to trace the subject polygon.
- If the edge leaves the clip polygon, record the intersection point & make a right follow the clip polygon in same manner.
- Continue until vertex reach visited vertex

Ex :-



⇒ 1. clockwise notation in subject polygon.



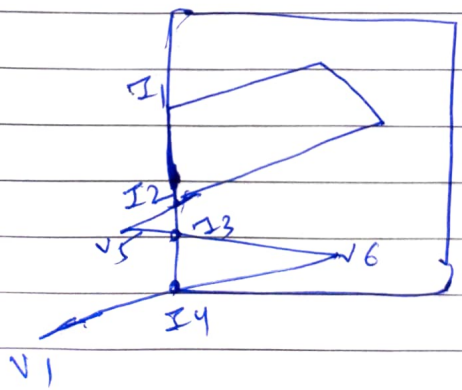
Here Intersection points are I_1, I_2, I_3 & I_4

2. Start from V_1 vertex to V_2 vertex in clockwise direction, both outside vertices then leave.

3. From V_2 to V_3 in clkwise, Here V_2 is outside & V_3 in inside, so record Intersection point I_1 & continue to V_3 vertex.

4. From V_3 to V_4 both are inside, so continue to subject polygon to V_4 vertex

5. From V_4 vertex to V_5 both V_4 in inside point & V_5 is outside so continue to clip polygon to intersection point from I_2 to I_1



6. From V_5 to V_6 : follow V_6

7. From V_6 to V_1 . V_1 is outside so continue to clip from intersection point I_4 to I_3 intersection point