# 6 Clipping

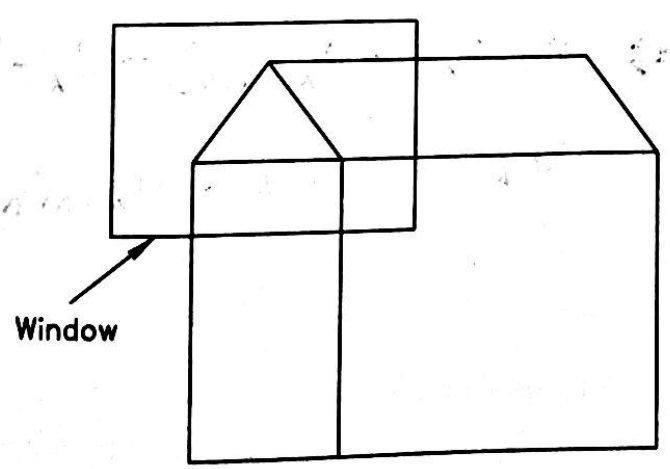


## 6.1 WINDOWPORT AND VIEWPORT

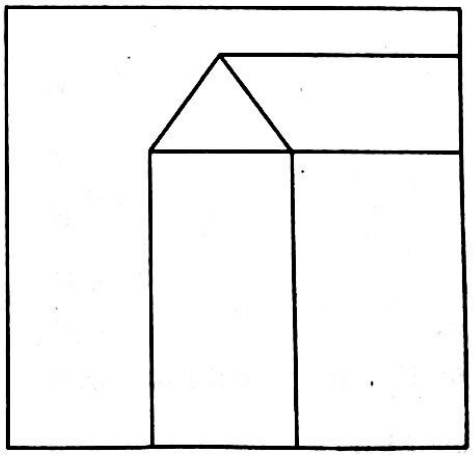
**1. Windowport.** This is a rectangle surrounding the object as a part of it that we wish do draw on the screen. The sides of the window (Windowport) are parallel to the $x$ and $y$ axes respectively.

Window is associated with the object rather than with the image and it is the viewport which brought the window content to the screen.

**2. Viewport.** This is a rectangular region of the screen which is selected for displaying the object or a part of it described in a window. Viewport is part of the computer screen.




(*a*)
A window



(*b*)
The part of the object inside the window is displayed in the viewport of the computer screen.

**Fig. 6.1**

### Windowport to Viewport Transformation

Let $(W_{x\,min}, W_{y\,min})$ and $(W_{x\,max}, W_{y\,max})$ are the bottom left and top right corners of the Windowport. Also, $(V_{x\,min}, V_{y\,min})$ and $(V_{x\,max}, V_{y\,max})$ be the bottom-left and top-right corners of the viewport.

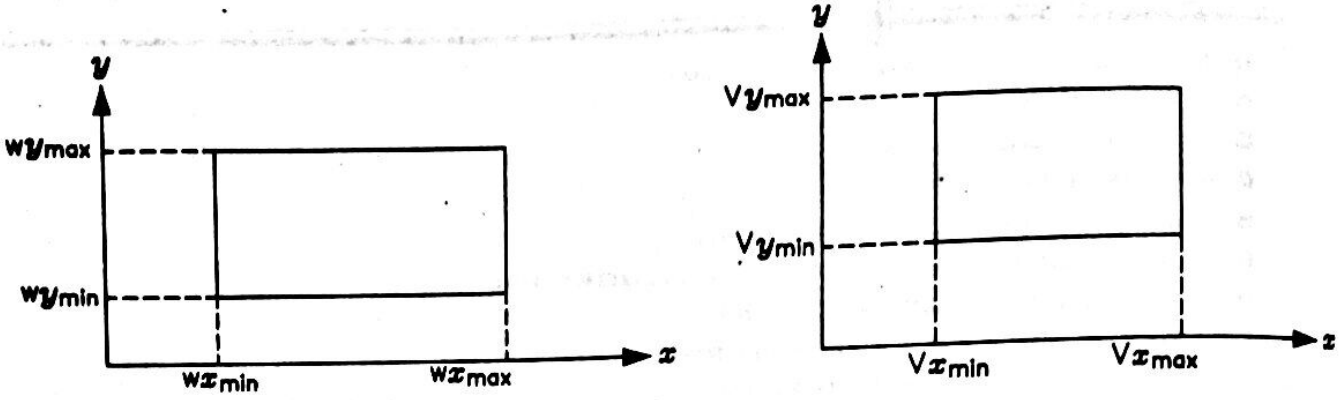



Fig. 6.2

Then, Windowport to viewport transformation is given by

$$f_x = \frac{V_{x\,max} - V_{x\,min}}{W_{x\,max} - W_{x\,min}} \text{ along the } x\text{-axis}$$

and $$f_y = \frac{V_{y\,max} - V_{y\,min}}{W_{y\,max} - W_{y\,min}} \text{ along the } y\text{-axis.}$$

Also, the transformation factor $f$ is given by

if $(f_x < f_y)$

Then $f = f_x$.

else $f = f_y$.

Also, Let $V_x = V_{x\,max}$.

$W_x = W_{x\,max}$

$V_y = V_{y\,max}$

$W_y = W_{y\,max}$

Then, $$f = \frac{V_x - V_{x\,min}}{W_x - W_{x\,min}} \Rightarrow f(W_x - W_{x\,min}) = V_x - V_{x\,min}$$

$$V_x = f(W_x - W_{x\,min}) + V_{x\,min}$$

and $$f = \frac{V_y - V_{y\,min}}{W_y - W_{y\,min}}$$

$\Rightarrow$ $$V_x = V_{x\,min} + f(W_x - W_{x\,min})$$

and $$V_y = V_{y\,min} + f(W_y - W_{y\,min})$$

So, $V_x$ and $V_y$ are the required transformed viewport co-ordinates.

## 6.2 CLIPPING

The process of cutting at the lines which are outside the window is called clipping. In other words, "Clipping" can be defined as a procedure using which we can identify whether the portion of any graphics object is within or outside of a specified region of space.

The region (or space) which is used to see the object is called window and the region on which the object (or a part of it) is shown is called viewport.

The process of clipping is required for the portion of the object(s) which does not contribute anything to the final image.

So, there are three cases :

**Case 1 :** The object may be completely outside the viewing area defined by the camera (or the viewport). It means that the object is not going to contribute anything in the final image.

**Case 2 :** The object may be seen partially, *i.e.*, some part of it lies in the viewing area and the other lies outside the viewing area.

**Case 3 :** The object may be completely seen from within the viewing area, *i.e.*, no part of the object lies outside the viewport area.

**Note :** It is clear that there is portion of the object(s) in case 1 and case 2 which does not contribute anything to final image. So, clipping is required in case 1 and case 2 while no clipping is required in case 3 (because no part of the object **in case 3** lies outside the viewing area).

## 6.3 POINT CLIPPING

In point clipping, a point is clipped (not shown in the viewport) if it is outside the viewing area (Windowport) otherwise it is not clipped.

Mathematically, a point $P$ having co-ordinates $(x, y)$ can be seen if the following two conditions are satisfied :

(*i*) $W_{x\,min} \le x \le W_{x\,max}$.

(*ii*) $W_{y\,min} \le y \le W_{y\,max}$.

Where $W_{x\,min}$ and $W_{x\,max}$ are the points showing the range of the Windowport in $x$-direction and $W_{y\,min}$ and $W_{y\,max}$ are the points show in the range of Windowport in $y$-direction.

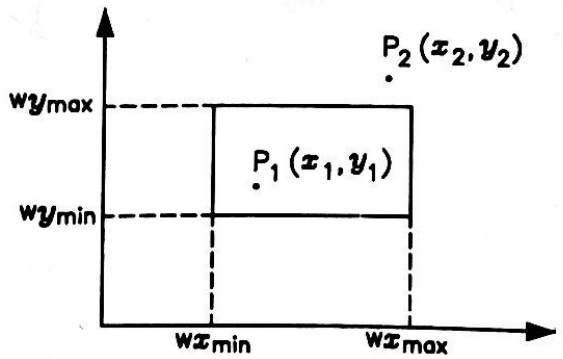



Fig. 6.3.

Considers two points $P_1$, $(x_1, y_1)$ and $P_2$ $(x_2, y_2)$ as shown in the figure.

Clearly, $W_{x\,min} \le x_1 \le W_{x\,max}$ and $W_{y\,min}$ $y_1$ $W_{y\,max}$ Both are satisfied.

So, no clipping required for point $P_1$ $(x_1, y_1)$.

But in case of point $P_2$ $(x_2, y_2)$.

$W_{x\,min} x_2 \le W_{x\,max}$ is satisfied and not $W_{y\,min} \le y_2 \le W_{y\,max}$ ($\because y_2 > W_{y\,max}$). So, point $P_2(x_2, y_2)$ less outside the viewing area (*i.e.*, Windowport) and so requires clipping. Thus, the point $P_2(x_2, y_2)$ will not be shown in the viewport.

## 6.4 LINE CLIPPING

In line clipping, A line or a part of the line is clipped if it is outside the Windowport.

In line clipping, there are three cases :

**Case 1 :** The line is totally outside the Windowport and so is directly clipped.

**Case 2 :** The line is partially clipped if a part of the line lies outside the Windowport.

**Case 3 :** The line is not clipped at all, *i.e.* It does not require clipping when the whole line is in the Windowport (*i.e.* no part of it lies outside the Windowport).

It will be more clear from fig. 6.4.

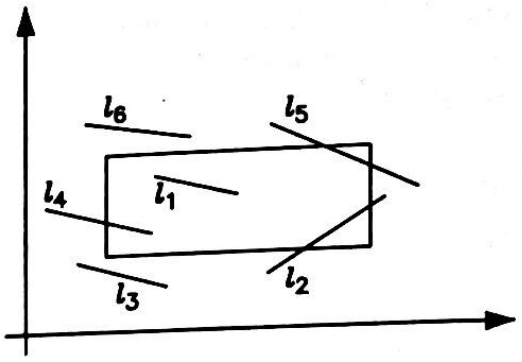



Fig. 6.4

From the figure, it is clear that

(*i*) The lines $l_3$ and $l_6$ require total clipping as they do not contribute anything in the final image.

(*ii*) The lines $l_2$, $l_4$ and $l_5$ requires partial clipping as a part of these lines lies outside the windowport.

(*iii*) The line $l_1$ does not require any clipping as the whole line $l_1$ lies inside the Windowport.

## 6.5 END POINT LINE CLIPPING

In end point line clipping algorithm, we check for the end-points of a line and if they lies inside the Windowport, then the line does not require clipping otherwise it require clipping.

Let $PQ$ be a line with end point coordinates as $P(x_s, y_s)$ and $Q(x_e, y_e)$.

Then, the line does not require clipping if the following four conditions are satisfied :

$$W_{x\,min} \le x_s \le W_{x\,max}$$

$$W_{y\,min} \le y_s \le W_{y\,max}$$

$$W_{x\,min} \le x_e \le W_{x\,max}$$

$$W_{y\,min} \le y_e \le W_{y\,max}$$

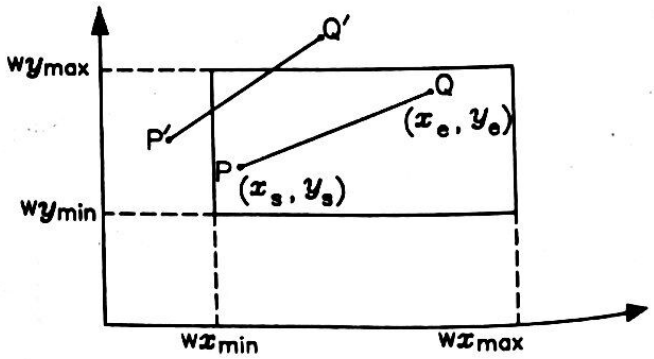



Fig. 6.5

Now, consider another line $P'Q'$ which cuts the Windowport in two points say $A$ and $B$. Then the line requires partial clipping and we need to compute the points $A$ and $B$.

For this, we have another method called mid point line clipping algorithm.

## 6.6 MID-POINT LINE CLIPPING ALGORITHM

First, we check whether the line intersects the Windowport or not. If it does not intersect the Windowport, then it is totally discarded if it lies outside the Windowport, and is not clipped at all (*i.e.* accepted) if it lies within the Windowport.

Now, if the line intersects the Windowport in two points say $A$ and $B$. Then we will show only the part $AB$ (of the whole line) in the viewport. For this, we need to compute the coordinates of the points $A$ and $B$ and we apply the mid-point line clipping algorithm in the following way :

(*i*) First, we divide the entire line into two (equal) parts from the mid-point of the line.

(*ii*) Then, we check the two segments (separately), and repeat the whole process again (*i.e.*, again divide them into two parts from their mid-points) until all the parts are divided into two types of line, one which is completely inside the Windowport and other which is completely outside the Windowport.

Consider the line $PQ$ which cuts the Windowport at two points say $A$ and $B$. Since, both the points $P$ and $Q$ are outside the Windowport. So, let us divide the line in two parts $PM$ and $MQ$.

Now, consider the segment $PM$. Its mid point is $M'$ and divide it in two parts $PM'$ and $M'M$ but $M' = A$ which is on the boundary of the Windowport. So, no need for further division.

Now, consider the segment $MQ$. Let us divides it into two parts, from its mid-point $M''$, in $MM''$ and $M''Q$ but $M'' = B$ which lies on the boundary of the Windowport. So need for further division.

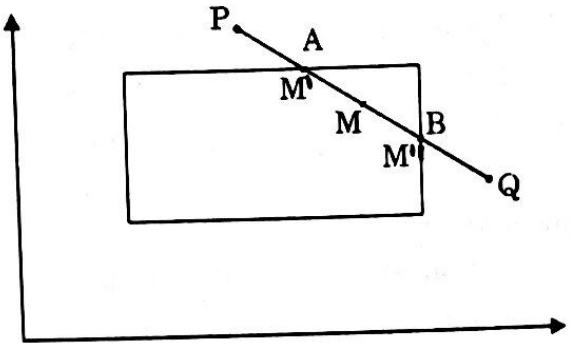



**Fig. 6.6**

Now, we have two points $M'$ and $M''$ (equal to $A$ and $B$) which lies on the boundary of the Windowport. So, we will display the line segment $M'M''$ (or $AB$) in the viewport and the parts $PM'$ and $M''Q$ will be discarded.

**Problem 1.1.** *Find the visible segment to be displayed in the viewport for the line from P (60, 140) to Q (220, 40) in the Windowport having range* $(W_{x\,min}, W_{x\,max}) = (40, 200)$ *and* $(W_{y\,min}, W_{y\,max}) = (20, 120)$.

**Solution.** Let $P_1 = P$ and $Q_1 = Q$

The line with start point $P_1$ (60, 140) and end point $Q_1$ (220, 40) has the mid-point.

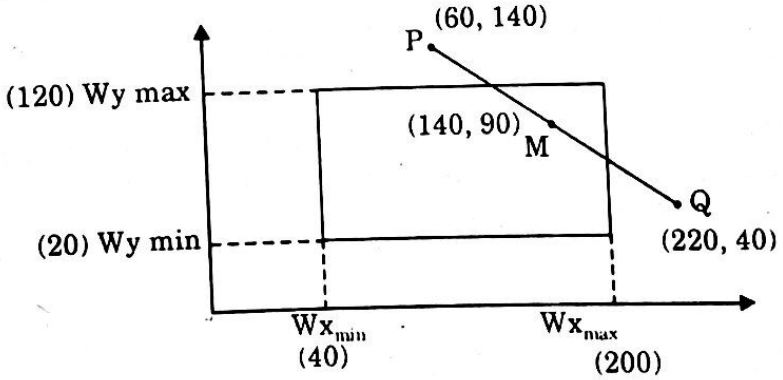



**Fig. 6.7**

$$M_1\left(\frac{60+220}{2}, \frac{140+40}{2}\right)$$

*i.e.*
$$M_1\left(\frac{280}{2}, \frac{180}{2}\right)$$

*i.e.*
$$M_1\,(140, 90)$$

$\therefore$ The line $PQ$ has been divided in two parts $P_1M_1$ and $M_1Q_1$.

Now again, we divides the line segments $P_1M_1$ and $M_1Q_1$ until we find two points lying on the boundary of the Windowport.

Further working is shown in the following table :

| *Start point* $P(x_s, y_s)$ | *End point* $Q(x_e, y_e)$ | *Mid point* $M(x_s - x_e)/_2, (y_s - y_e)/2)$ | *Location of Mid-point* |
|---|---|---|---|
| **Segment PQ** | | | |
| $P_1$ (60, 140) | $Q_1$ (220, 40) | $M_1$ (140, 90) | Inside |
| **Segment $P_1M_1$** | | | |
| $P_2$ (60, 140) | $Q_2$ (140, 90) | $M_2$ (100, 115) | Inside |
| $P_3$ (60, 140) | $Q_3$ (100, 115) | $M_3$ (80, 123) | Outside |
| $P_4$ (80, 123) | $Q_4$ (100, 115) | $M_4$ (90, 119) | Inside |
| $P_5$ (80, 123) | $Q_5$ (90, 119) | $M_5$ (85, 121) | Outside |
| $P_6$ (85, 121) | $Q_6$ (90, 119) | $M_6$ (88, 120) | On the top edge. |
| **Segment $M_1Q_1$** | | | |
| $P_2'$ (140, 90) | $Q_2'$ (220, 40) | $M_2'$ (180, 65) | Inside |
| $P_3'$ (180, 65) | $Q_3'$ (220, 40) | $M_3'$ (200, 53) | On the right edge. |

The visible segment (that can be displayed on the viewport) is therefore from $M_6$ (88, 120) to $M_3'$ (200, 53) *i.e.*

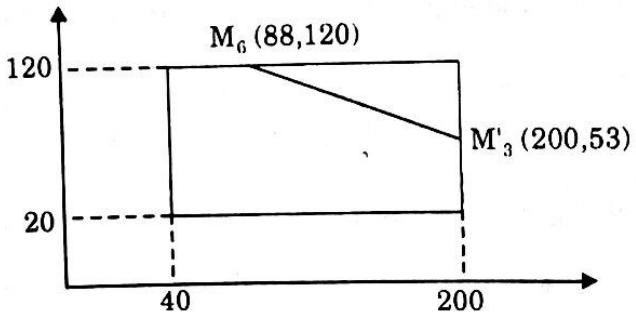



**Fig. 6.8**

**Problem 1.2.** *Windowport is given by (100, 100, 300, 300) and viewport is given by (50, 50, 150, 150). Convert the Windowport co-ordinates (200, 200) to the viewport co-ordinates ?*

**Solution.** We know that Windowport is given by ($W_{x\,min}$, $W_{y\,min}$, $W_{x\,max}$, $W_{y\,max}$) and viewport is given by ($V_{x\,min}$, $V_{y\,min}$, $V_{x\,max}$, $V_{y\,max}$).

So, we have,

$$W_{x\,min} = 100 \qquad V_{x\,min} = 50$$
$$W_{y\,min} = 100 \qquad V_{y\,min} = 50$$
$$W_{x\,max} = 300 \qquad V_{x\,max} = 150$$
$$W_{y\,max} = 300 \qquad V_{y\,max} = 150$$

Now,
$$f_x = \frac{V_{x\,max} - V_{x\,min}}{W_{x\,max} - W_{x\,min}}$$
$$= \frac{150 - 50}{300 - 100} = \frac{100}{200} = \frac{1}{2}$$

$$f_y = \frac{V_{y\,max} - V_{y\,min}}{W_{y\,max} - W_{y\,min}}$$

$$= \frac{150 - 50}{300 - 100} = \frac{100}{200} = \frac{1}{2}$$

Since $f_x \lhd f_y$

$\therefore$ $f = f_y = \frac{1}{2}$

Also, $W_x = 200$

$W_y = 200$

So, the transformed viewport co-ordinates $(V_x, V_y)$ are given by

$$V_x = V_{x\,min} + f(W_x - W_{x\,min})$$

and $$V_y = V_{y\,min} + f(W_y - W_{y\,min})$$

$\therefore$ $$V_x = 50 + \frac{1}{2}(200 - 100)$$

$$= 50 + \frac{1}{2}(100) = 50 + 50 = 100$$

and $$V_y = 50 + \frac{1}{2}(200 - 100)$$

$$= 50 + \frac{1}{2}(100) = 50 + 50 = 100$$

So, the transformed viewport co-ordinates are (100, 100).

## 6.7 COHEN-SUTHERLAND LINE CLIPPING ALGORITHM

The cohen-Sutherland line clipping algorithm uses a four bit binary code for checking the line that it lies in which region of the plane. In this, the plane is divided into nine parts and each sub-part is assigned a unique binary code and these binary codes are used to determine the position of the line that in which region of the plane it lies.

Codes are assigned in the following way :

If both the end points have binary code 0000, then the entire line is within the Windowport.

Else, we test both the end points and assign code as follows :

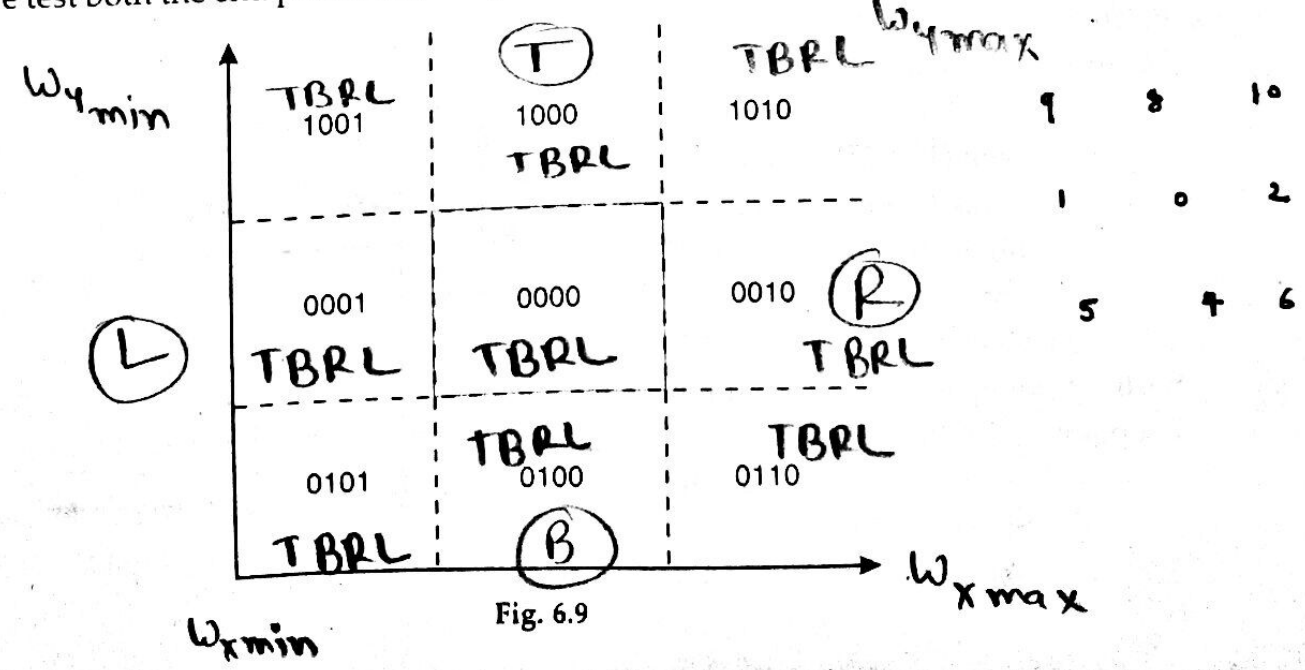



Fig. 6.9

(*i*) If the point is on the top (above the window), then top bit is set to 1.
(*ii*) If the point is on the bottom (below the window), then bottom bit is set to 1.
(*iii*) If the point is on the right of the window, then right bit is set to 1.
(*iv*) If the point is on the left of the window, then left bit is set to 1.

| | | *T* | *B* | *R* | *L* | |
|---|---|---|---|---|---|---|
| | (*i*) | 0 | 0 | 0 | 0 | (point is inside the window) |
| *T* | (*ii*) | 1 | 0 | 0 | 0 | (point is on the top of the window) |
| *B* | (*iii*) | 0 | 1 | 0 | 0 | (point is on the bottom of the window) |
| *R* | (*iv*) | 0 | 0 | 1 | 0 | (point is on the right of the window) |
| *L* | (*v*) | 0 | 0 | 0 | 1 | (point is on the left of the window) |
| *T–R* | (*vi*) | 1 | 0 | 1 | 0 | (point is on the right-top of the window) |
| *T–L* | (*vii*) | 1 | 0 | 0 | 1 | (point is on the left-top of the window) |
| *B–R* | (*viii*) | 0 | 1 | 1 | 0 | (point is on the bottom-right of the window) |
| *B–L* | (*ix*) | 0 | 1 | 0 | 1 | (point is on the bottom-left of the window). |

Let $(W_{x\,min}, W_{y\,min})$ and $(W_{x\,max}, W_{y\,max})$ be the bottom-left and top-right corners of the Windowport.
For Assigning the binary code to a point $P(x, y)$, following procedure is used :

(1) Compute

T Sign $(y - y_{max})$ for top $(\because y > y_{max})$
B Sign $(y_{min} - y)$ for bottom $(\because y_{min} > y)$
R Sign $(x - x_{max})$ for right $(\because x > x_{max})$
L Sign $(x_{min} - x)$ for left $(\because x_{min} > x)$

(2)

| Sign | Value do be assigned |
|---|---|
| + ve | 1 |
| – ve | 0 |

**Problem 1.3.** *Consider the Windowport given by* $(W_{x\,min}, W_{y\,min}, W_{x\,max}, W_{y\,max}) = (5, 15, 5, 25)$
*Give the binary codes for the points.*
$P_1(10, 10), P_2(20, 10), P_3(10, 30), P_4(20, 30)$.

**Solution.** For point $P_1(10, 10)$

$x = 10$ and $y = 10$
$\therefore$ Sign $(10 - 25) =$ – ive $\Rightarrow T = 0$
Sign $(5 - 10) =$ – ive $\Rightarrow B = 0$
Sign $(10 - 15) =$ – ive $\Rightarrow R = 0$
Sign $(5 - 10) =$ – ive $\Rightarrow L = 0$
$\therefore$ Binary code for $P_1$ is 0000 (TBRL).
So, the point $P_1$ lies inside the Windowport.
For point $P_2(20, 10)$

$x = 20, y = 10$
$\therefore$ Sign $(10 - 25) =$ – ive $\Rightarrow T = 0$
Sign $(5 - 10) =$ – ive $\Rightarrow B = 0$

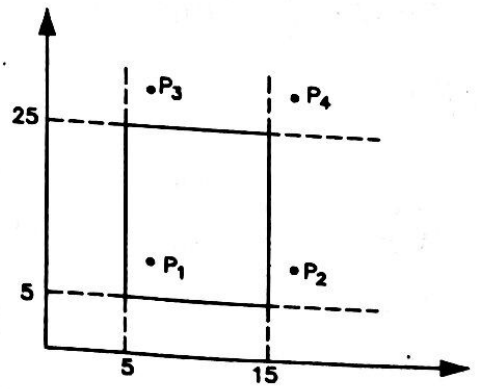



Fig. 6.10

Sign $(20 - 15) = +\text{ive} \Rightarrow R = 1$

and Sign $(5 - 20) = -\text{ive} \Rightarrow L = 0$

$\therefore$ Binary code for $P_2$ is 0010.

So, the point $P_2$ lies on the right of the Windowport.

For point $P_3$ (10, 30)

$x = 10, y = 30$

$\therefore$ Sign $(30 - 25) = +\text{ive} \Rightarrow T = 1$

Sign $(5 - 30) = -\text{ive} \Rightarrow B = 0$

Sign $(10 - 15) = -\text{ive} \Rightarrow R = 0$

and Sign $(5 - 10) = -\text{ive} \Rightarrow L = 0$

$\therefore$ Binary code for $P_3$ is 1000.

and so, the point $P_3$ lies above the Windowport.

For point $P_4$ (20, 30)

$x = 20, y = 30$

$\therefore$ Sign $(30 - 25) = +\text{ive} \Rightarrow T = 1$

Sign $(5 - 30) = -\text{ive} \Rightarrow B = 0$

Sign $(20 - 15) = +\text{ive} \Rightarrow R = 1$

and Sign $(5 - 20) = -\text{ive} \Rightarrow L = 0$

$\therefore$ Binary code for $P_4$ is 1010.

and so, the point $P_4$ lies on the top-right of the Windowport.

### 6.7.1 To Find the Intersection Point

Let a line $AB$ intersects the Windowport at a point $P$. Let coordinates of $A$ and $B$ are $(x_1, y_1)$ and $(x_2, y_2)$ respectively and that of $P$ are $(x, y)$.

Now, the equation of the line $AB$, from $(x_1, y_1)$ to $(x_2, y_2)$ can be written as

$$(x - x_1)(y_2 - y_1) = (y - y_1)(x_2 - y_1) \quad ...(i)$$

Where $(x, y)$ are the co-ordinates of the point $P$.

Also, the co-ordinates of the point $P$ can be found using equation $(i)$.

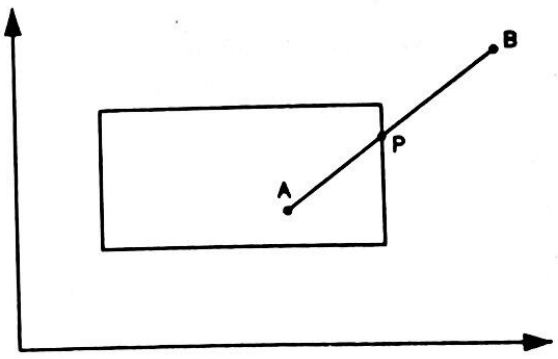



Fig. 6.11

It is illustrated in the following example :

**Problem 1.4.** *Let the Windowport be (5, 25, 10, 30). Clip a quadrilateral ABCD with corner coordinates A (10, 22), B (18, 22), C (30, 32) and D (10, 42).*

**Solution.** For point $A$,

Sign $(y - y_{max}) = (22 - 30) = -\text{ive} \therefore T = 0$

Sign $(y_{min} - y) = (10 - 22) = -\text{ive} \therefore B = 0$

Sign $(x - x_{max}) = 10 - 25 = -\text{ive} \therefore R = 0$

Sign $(x_{min} - x) = 5 - 10 = -\text{ive} \therefore L = 0$

$\therefore$ Binary code for A = 0000

$\therefore$ Point $A$ lies within the Windowport.

Similarly, $B$ also lies within the Windowport. Also, it is very clear from the figure that points C and $D$ lies outside the Windowport. Let lines $BC$ and $AD$ intersect the Windowport at points $P$ and $P'$ respectively.

To find, points $P$ and $P'$.

Clearly, the points of $P'$ are (10, 30). [from the figure] put in new line now, to find co-ordinates of point $P$.

Explain of line $AB$ is

$$(x - x_1)(y_2 - y_1) = (y - y_1)(x_2 - x_1)$$

$$(x - 18)(32 - 22) = (y - 22)(30 - 18)$$

$$(x - 18)(10) = (y - 22)(12)$$

$$\Rightarrow \quad 10x - 12y + 84 = 0 \qquad \ldots(1)$$

Now, point $P$ lies on this line.

Clearly, the $x$-coordinate of point $P$ is $x = 25$.

So, putting $x = 25$, in equation (1),

We have

$$10 \times 25 - 12y + 84 = 0$$

$$\Rightarrow \quad 250 + 84 = 12y$$

$$\Rightarrow \quad y = \frac{334}{12} = 27.8$$

$\therefore$ Co-ordinates of $P$ are (25, 27.8).

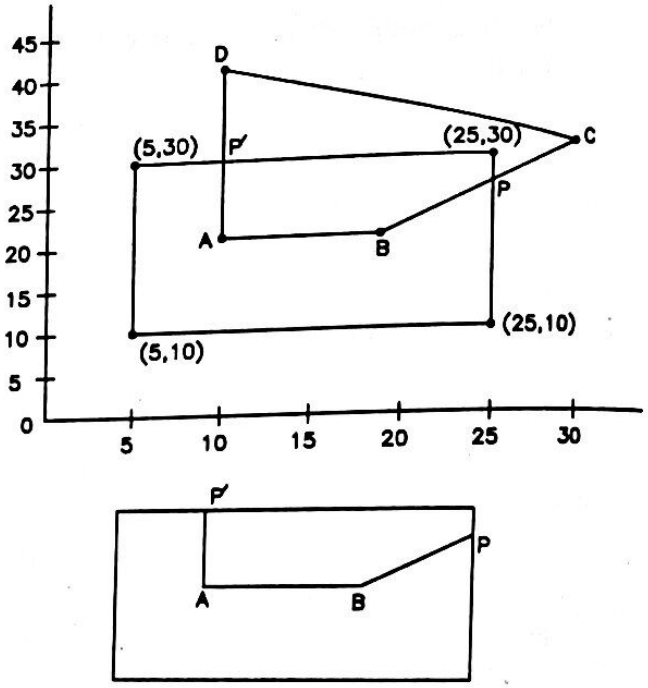



Fig. 6.12

So, the part of the quadrilateral to be shown in the viewport is $ABPP'$; where $A$ (10, 22), B (18, 22), $P$ (25, 27.8) and $P'$ (10, 30).

**Problem 1.5.** *Let R be the rectangular window whose lower right hand corner is at B (+ 2, 2) and upper left hand corner is at D (– 2, 5). Find out the end point codes for the points in the following figure.*

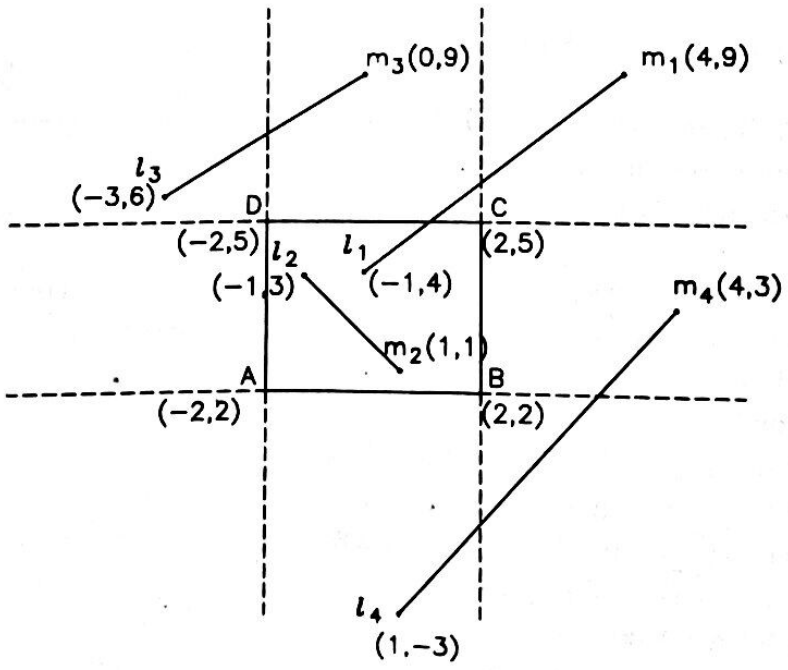



Fig. 6.13

**Solution.** The end-points codes for point $(x, y)$ are set according to the scheme as follows :

$$\text{Bit 1} = \text{Sign}\,(y - y_{max}) = \text{Sign}\,(y - 5)$$
$$\text{Bit 2} = \text{Sign}\,(y_{min} - y) = \text{Sign}\,(2 - y)$$
$$\text{Bit 3} = \text{Sign}\,(x - x_{max}) = \text{Sign}\,(x - 2)$$
$$\text{Bit 4} = \text{Sign}\,(x_{min} - x) = \text{Sign}\,(-2 - x)$$

Here $$\text{Sign } a = \begin{cases} 1 & \text{if } a \text{ is + ve} \\ 0 & \text{otherwise} \end{cases}$$

So,
$$l_1(-1, 4) = 0000$$
$$m_1(4, 9) = 1010$$
$$l_2(-1, 3) = 0000$$
$$m_2(1, 1) = 0000$$
$$l_3(-3, 6) = 1001$$
$$m_3(0, 9) = 1000$$
$$l_4(1, -3) = 0100$$
$$m_4(4, 3) = 0010$$

**Problem 1.6.** *Clipping against rectangular windows whose sides are aligned with the x and y axes involves computing intersections with vertical and horizontal lines. Find the intersection of a line segment $P_1 P_2$ having co-ordinates $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ with a.*

**Solution.** We know that parametric equation for any line (let $P_1 P_2$) is as follows

$$x = x_1 + u(x_2 - x_1)\ 0 \le u \le 1 \quad \ldots(1)$$
$$y = y_1 + u(y_2 - y_1) \quad \ldots(2)$$

(*i*) Here $x = a$ Substitute it in above equation and find $u = \dfrac{a - x_1}{x_2 - x_1}$ and Substitute this value into eq. (2) we find that the intersection point $x_i = a$ and

$$y_i = y_1 + \left(\frac{a - x_1}{x_2 - x_1}\right)(y_2 - y_1)$$

(*ii*) Here $y = b$ again calculating as in (1) $y = b$ and

$$x_i = x_1 + \left(\frac{b - y_1}{y_2 - y_1}\right)(x_2 - x_1)$$

**Problem 1.7.** *Use Cohen-Sutherland algorithm to find the visible portion of the line P (40, 80) Q (120, 30) inside the window, the window is defined as ABCD : A (20, 20) B (60, 20), C (60, 40) and D (20, 40).*

**Solution.** All the four bit of the outcodes can be calculated as follows :

$$\text{Bit 1} = \text{Sign}\,(y - y_{max}) = \text{Sign}\,(y - 40)$$
$$\text{Bit 2} = \text{Sign}\,(y_{min} - y) = \text{Sign}\,(20 - y)$$
$$\text{Bit 3} = \text{Sign}\,(x - x_{max}) = \text{Sign}\,(x - 60)$$
$$\text{Bit 4} = \text{Sign}\,(x_{min} - x) = \text{Sign}\,(20 - x)$$

$$\text{Sign}\,(a) = \begin{cases} 1 & \text{if + ve} \\ 0 & \text{otherwise} \end{cases}$$

Out codes for the point $P$ (40, 80) → 1000

$Q$ (120, 30) → 0010

(1000) AND (0010) = 0000 which is zero.

Equation of $PQ = 5x + 8y = 840$

Solving this equation with all four equations of window boundary

$$x = 20 \Rightarrow y = 92.5$$
$$x = 60 \Rightarrow y = 67.5$$
$$y = 20 \Rightarrow x = 136$$
$$y = 40 \Rightarrow x = 104$$

Since both the values of $y$ are greater than $y_{max}$ and both are also greater than $x_{max}$. Therefore, the line segment $PQ$ lies completely outside the window.

### 6.7.2 Cohen-Sutherland Line Clipping Algorithm

In the Cohen-Sutherland clipping algorithm, we calculate a binary code for both of the end points of a line. And if the binary code for both the end points is zero, then the whole line is visible and is in the Windowport and can be shown in the viewport. No clipping is required in this case. If the binary code is non-zero, then the point lies outside the Windowport. In case of non-zero binary code, there are three cases, which are discussed in the following table :

Let $PQ$ be the line.

| Case | Outcode for end-point $P$ | Outcode for end-point $Q$ | Result | Conclusion for line segment PQ |
|---|---|---|---|---|
| 1. | 0000 | 0000 | Not required | Line is fully visible. Retain it. No clipping required. |
| 2. (*a*) | 0000 | Non-zero | Not required | Line $PQ$ is partially visible. It crosses from Windowport to the surrounding region corresponding to the bits 1. It requires clipping. |
| (*b*) | Non-zero | 0000 | Not required | |
| 3. | Non-zero | Non-zero | Non-zero | Line is completely outside the window port in the region with outcode bit 1. Fully invisible, reject it. |
| 4. | Non-zero | Non-zero | zero | Undecidable. Requires more investigation by segments. |

### 1.7.3 Algorithm for Cohen-Sutherland Clipping

**Step 1.** Compute the outcodes for each of the vertices of the given polygon.

**Step 2.** Start a loop on the sides of the polygon from $k = 1$ to $n$.

**Step 2.1.** Compute the outcode for the starting and ending vertices of the $k$th sides.

**Step 2.2.** If both the outcodes comes out to be zero, then the sides $k$ is within the windowpart and is fully visible. So retain it and go to step 3.

**Step 2.3.** If both the outcodes are non-zero, compute the value of 'AND' operation between them.

**Step 2.3.1.** If the value comes out to be non-zero, the side $k$ is completely outside the windowport, so is fully invisible and because will be rejected go to step 3.

**Step 2.3.2.** If the value comes out to be zero, the side may need clipping go to step 2.5.

**Step 2.4.** If one of the two outcodes is zero, the side requires to be clipped, go to step 2.5.

**Step 2.5.** Compute the equation of the side using the intersection formula

$$(x - x_1)(y_2 - y_1) = (y - y_1)(x_2 - x_1)$$

**Step 2.6.** Find intersections of the side $K$ with each of the edges of the Windowport, pushing the segments towards the edge of the windowpart, and get the new fresh outcodes. Go to step 2.2.

**Step 3.** End the loop on $i$.

## 6.8 PRINCIPLES OF SUTHERLAND-HODGMAN METHOD

Let $(x_1, y_1)$ and $(x_2, y_2)$ are the co-ordinates of the start and end-points of the four edges of the Windowport. (edges take one by one).

Now, we would like to find out the location of a point $P(x, y)$ on the polygon in correspondence with the Windowport as follows

$$r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

This expression is evaluated for all four edges of the Windowport.

The following decisions are drawn depending on the value of $r$.

**Case 1 :** The point $P$ lies within the Windowport if the value of $r$ is positive for all the four edges considered of the Windowport.

**Case 2 :** The point $P$ lies outside the Windowport if the value of $r$ is negative for one or more edges considered.

**Case 3 :** The point $P$ lies on a side/edge of the Windowport if the value of $r$ is zero for the corresponding edge considered.

**Case 4 :** The point P lies on the intersection point of two of edges of the Windowport if the value of $r$ is zero for both the corresponding edges considered.

Consider the Windowport shown in Fig. 6.14.

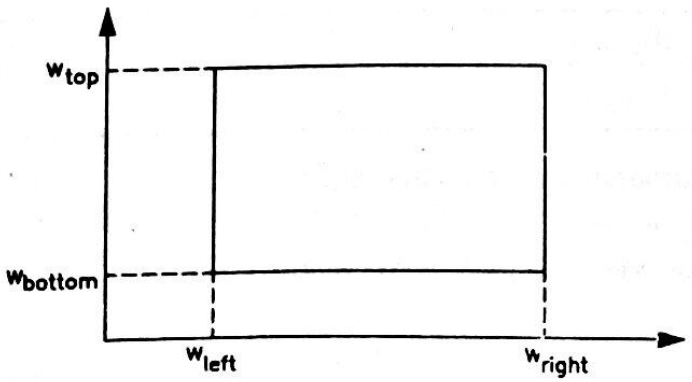



Fig. 6.14

Now, consider the bottom window edge.

$$(x_1, y_1) = (W_{left}, W_{bottom})$$
$$(x_2, y_2) = (W_{right}, W_{bottom})$$
$$\therefore \quad r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$
$$= (W_{right} - W_{left})(y - W_{bottom}) - (W_{bottom} - W_{bottom})(x - W_{left})$$
$$= L(y - W_{bottom}) \text{ where } L = (W_{right} - W_{left})$$

Now, consider the right edge,

$$(x_1, y_1) = (W_{right}, W_{bottom})$$
$$(x_2, y_2) = (W_{right}, W_{top})$$
$$\therefore \quad r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$
$$= (W_{right} - W_{right})(y - W_{bottom}) - (W_{top} - W_{bottom})(x - W_{right})$$
$$= -D(x - W_{right}) \text{ where } D = W_{top} - W_{bottom}.$$

Now, consider the top edge of the window

$$(x_1, y_1) = (W_{right}, W_{top})$$
$$(x_2, y_2) = (W_{left}, W_{top})$$

So,

$$r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$
$$= (W_{left} - W_{right})(y - W_{top}) - (W_{top} - W_{top})(x - W_{right})$$
$$= -L(y - W_{top}) \text{ where } L = (W_{right} - W_{left})$$

Now, consider the left edge of the Windowport

$$(x_1, y_1) = (W_{left}, W_{top})$$

and

$$(x_2, y_2) = (W_{left}, W_{bottom})$$
$$\therefore \quad r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$
$$= (W_{left} - W_{left})(y - W_{top}) - (W_{bottom} - W_{top})(x - W_{left})$$
$$= D(x - W_{left}) \text{ where } D = W_{top} - W_{bottom}$$

The above four cases, can be described in the following table :

| Window Edge | $(x_1, y_1)$ | $(x_2, y_2)$ | Expression $r$ $(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$ |
|---|---|---|---|
| Bottom | $(W_{left}, W_{bottom})$ | $(W_{right}, W_{bottom})$ | $L(y - W_{bottom})$ |
| Right | $(W_{right}, W_{bottom})$ | $(W_{right}, W_{top})$ | $-D(x - W_{right})$ |
| Top | $(W_{right}, W_{top})$ | $(W_{left}, W_{top})$ | $-L(y - W_{top})$ |
| Left | $(W_{left}, W_{top})$ | $(W_{left}, W_{bottom})$ | $D(x - W_{left})$ |

### 6.8.1 Algorithm for Sutherland-Hodgman Clipping

**Step 1.** Set the vertex count $m = n$, the total no. of vertices for the given polygon.

**Step 2.** Start a loop on the edges of the Windowport from $i = 1$ to 4 (Bottom, Right, Top, Left).

**Step 3.** Initialize vertex counter $i = 0$.

**Step 4.** Calculate $r_1$ for the 1st vertex using the equation.

$$r = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

If $r_1$ is zero or + ive, retain the vertex 1 in the list of visible vertices and set $K = 1$.

**Step 5.** Start loop for the vertices of the polygon from $i = 1$ to $m$.

**Step 5.1.** If $j = 1$, set $r_j = r_1$
else set $r_j = r_{j+1}$

**Step 5.2.** If $j = m$ set $r_{j+1} = r_1$ and go to step 5.3.
else compute $r_{j+1}$ for vertex $(j + 1)$ from the same equation as above.

**Step 5.3.** If the $r_j$ value is zero or + ive, pass j to the list of vertices, and set $k = k + 1$.

**Step 5.4.** If $r_j$ and $r_{j+1}$ are of opposite signs, then the polygon side $j$ intersects the window edge $i$, say at $P$. Compute the equation of the polygon side $j$ from vertex $j$ to the next vertex using the same equation.

**Step 5.5.** Depending on whether $i = 1, 2, 3$ or 4, Substitute $y = W_t$, $x = W_r$, $y = W_b$ or $x = W_l$, in the equation to side $j$ and compute the other co-ordinate of the intersection point $P$.

**Step 5.6.** Store the co-ordinates of the intersection point $P$, pass $P$ to the list of vertices, and set $k = k + 1$. If the sign change of $r$ is from – ive to + ive, interchange vertices $J$ and $P$ in the vertex list.

**Step 6.** End the loop on $j$ for polygon vertices, set $m = k$.

**Step 7.** End the loop on $i$ for the edge of the Windowport.

**Step 8.** Join all the vertices in the vertex output list to get the clipped polygon.

## 6.9 SUTHERLAND-HODGMAN POLYGON CLIPPING METHOD

In 1974, Sutherland and Hodgman gave a technique/method which was used for testing that whether the portions of a polygon are visible within a window or not, and the portions which are outside the window are clipped.

### 6.9.1 Cohen-Sutherland Vs Sutherland-Hodgman

In Cohen-Sutherland line clipping, we test for each of the edges of the polygon for visibility against the windowport and in Sutherland-Hodgman polygon clipping, we test each of the edges of the Windowport for visibility against the polygon and the part/portion of the polygon which lies outside the Windowport is clipped. It is illustrated in the following figure (Fig. 6.14).

### 6.9.2 Basic Idea

— Consider each edge of the viewport individually.

— Clip the polygon against the edge equation.

— After doing all planes, the polygon is fully clipped.

**e.g.** Consider the following figure (Fig. 6.15 on next page).

### 6.9.2.1 Basic Idea Behind Sutherland-Hodgman Clipping Algorithm

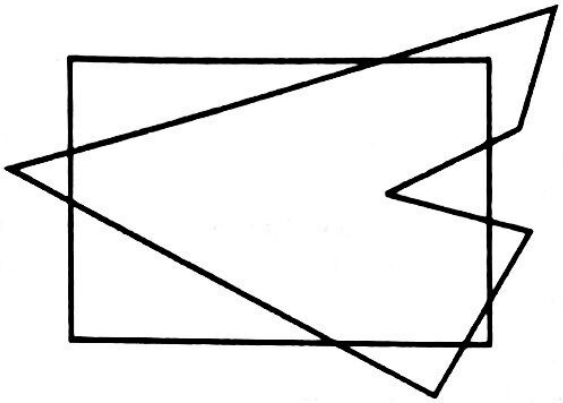
→ Consider each edge of the windowpart individually

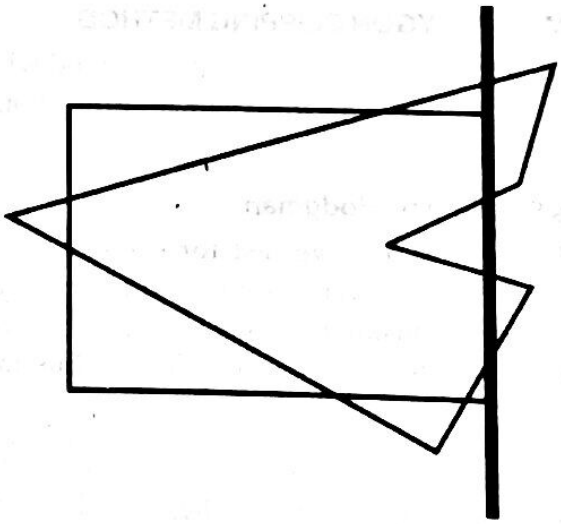
→ Clip the polygon against the edge equation

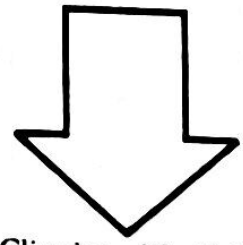
→ After doing clipping about all the planes, the polygon is fully clipped.

- **Input/output for algorithm :**
  - Input : list of polygon vertices in order
  - Output : list of clipped polygon vertices consisting of old vertices (may be) and new vertices (may be)
- **Note : This is exactly what we expect from the clipping operation against each edge.**

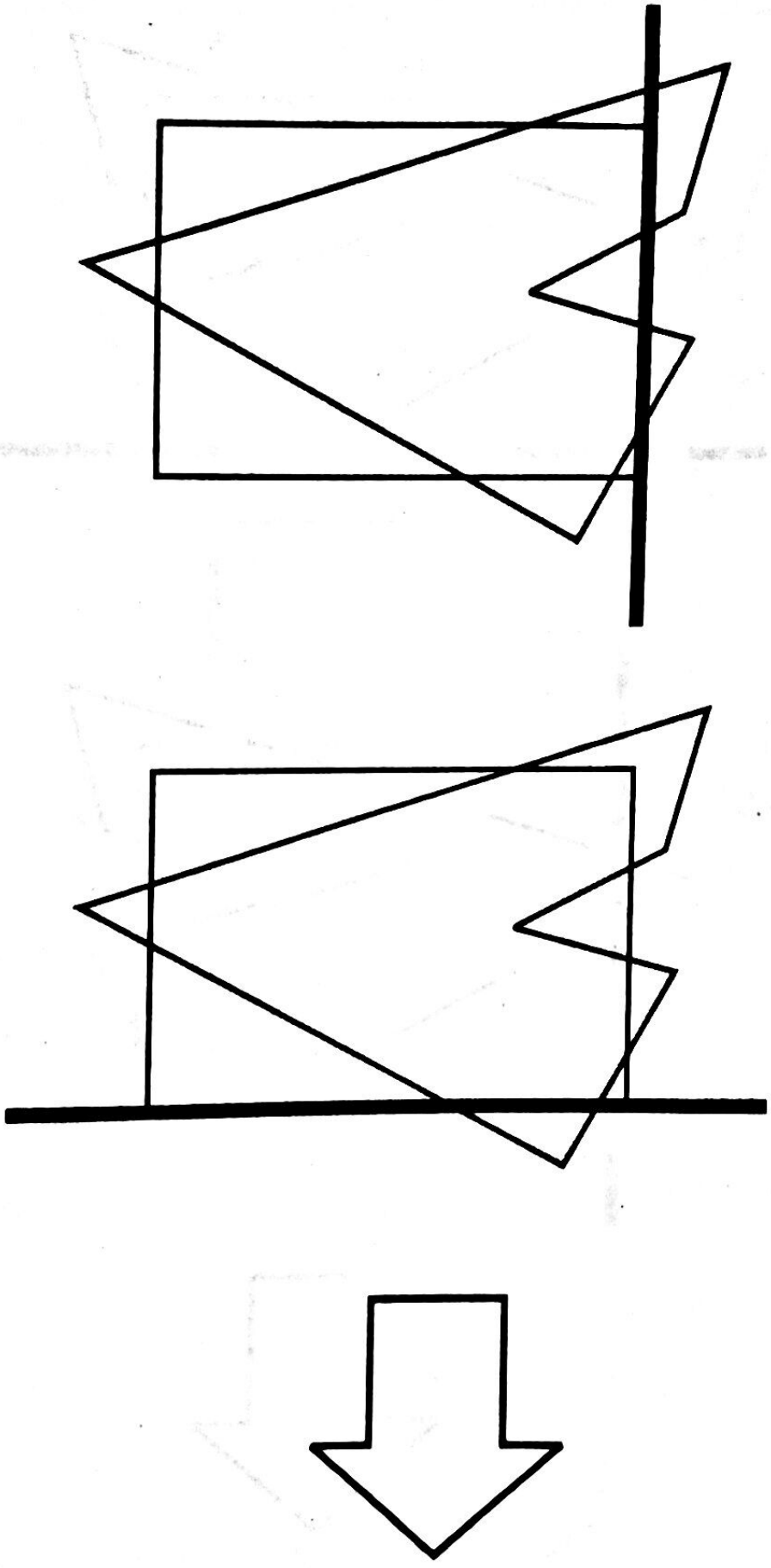
*e.g.* Consider the following Polygon in the Windowpart, some part of polygon is lying outside the windowport. Our task is to clip the portion of the polygon which is outside the windowport.



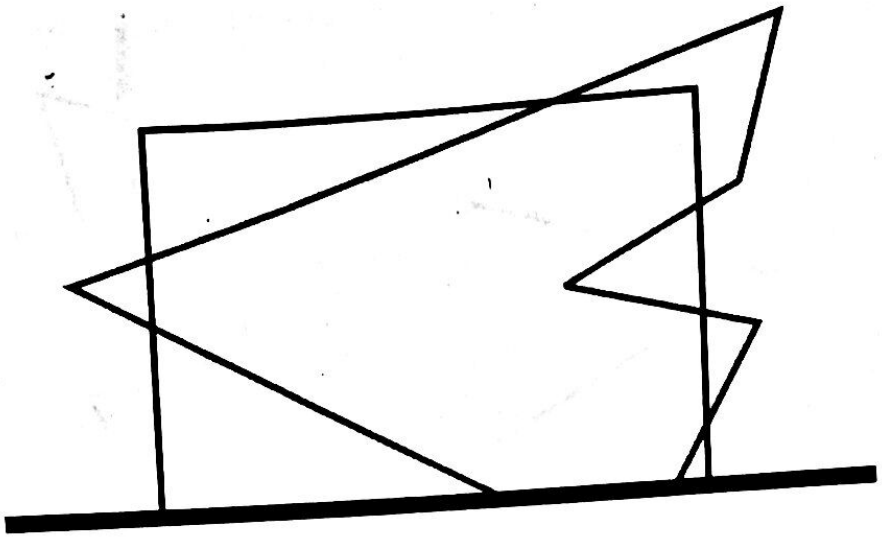


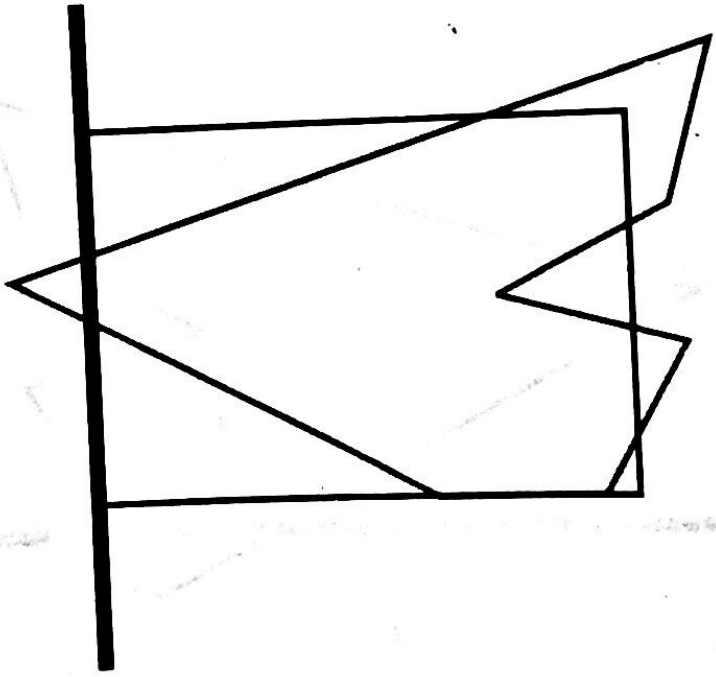


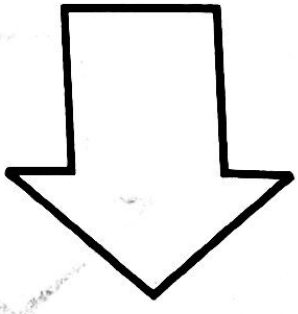
After Right Clipping of the Polygon.



After Bottom Clipping of the Polygon.







After Left Clipping of the Polygon.

After Top Clipping of the Polygon.

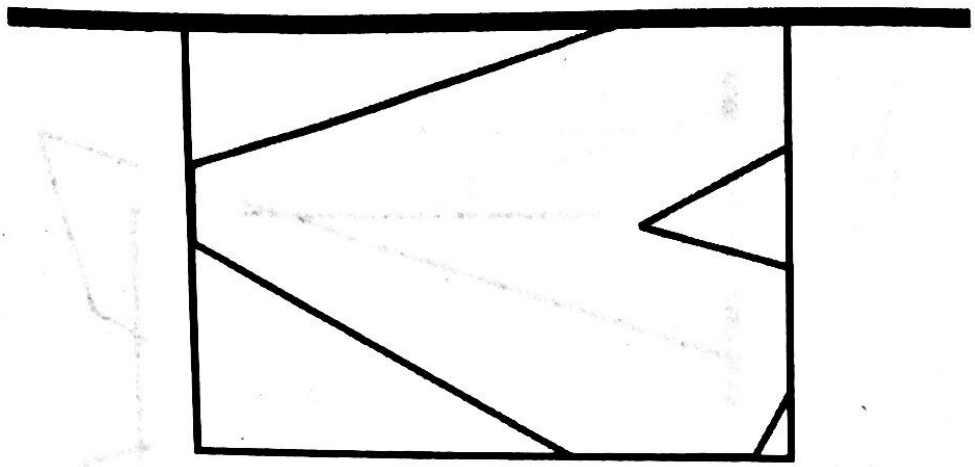


Fig. 6.15.

So after doing clipping for all the edges of the windowport, The required polygon after clipping is shown below in the viewport in Fig. 6.16.

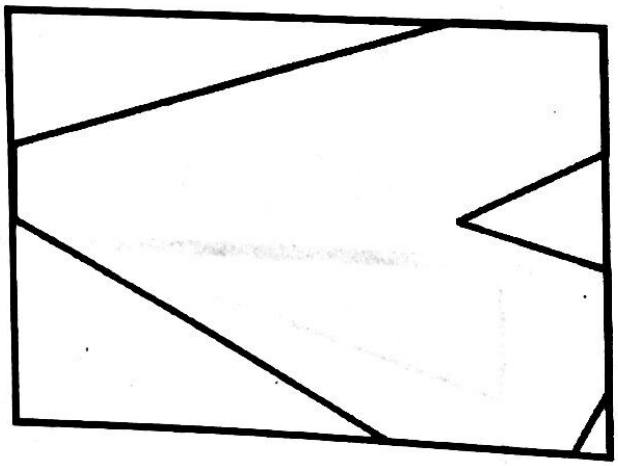


Fig. 6.16.

## FLASH BACK

- **Windowport.** This is a rectangle surrounding the object as a part of it that we wish do draw on the screen. The sides of the window (Windowport) are parallel to the $x$ and $y$ axes respectively.
- **Viewport.** This is a rectangular region of the screen which is selected for displaying the object or a part of it described in a window. Viewport is part of the computer screen.
- In end point line clipping algorithm, we check for the end-points of a line and if they lies inside the Windowport, then the line does not require clipping otherwise it require clipping.
- The cohen-Sutherland line clipping algorithm uses a