## UNIT II

# Clipping

When we have to display a large portion of the picture, then not only scaling & translation is necessary, the visible part of picture is also identified. This process is not easy. Certain parts of the image are inside, while others are partially inside. The lines or elements which are partially visible will be omitted.

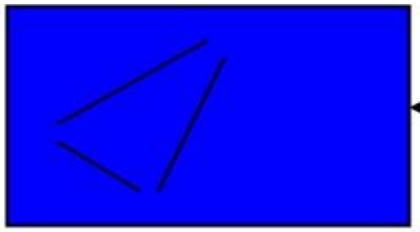
For deciding the visible and invisible portion, a particular process called clipping is used. Clipping determines each element into the visible and invisible portion. Visible portion is selected. An invisible portion is discarded.

## Types of Lines:

Lines are of three types:

1. **Visible:** A line or lines entirely inside the window is considered visible
2. **Invisible:** A line entirely outside the window is considered invisible
3. **Clipped:** A line partially inside the window and partially outside is clipped. For clipping point of intersection of a line with the window is determined.

Case1:

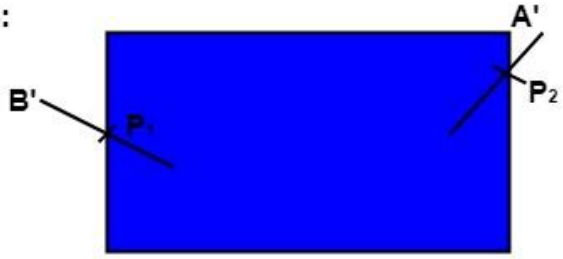



Visible Lines

Case2:

Invisible Lines

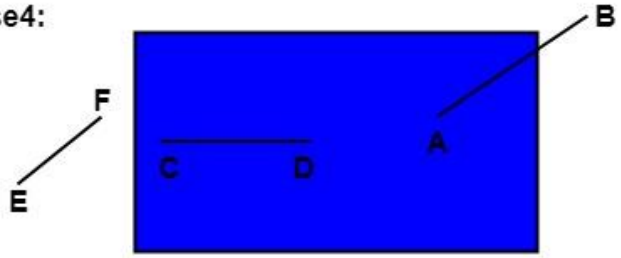
Case3:




In this figure $P_1$ and $P_2$ are point of intersection. The line $P_2$ to A' and $P_1$ to B' is discarded or clipped.

Clipped Lines

Case4:

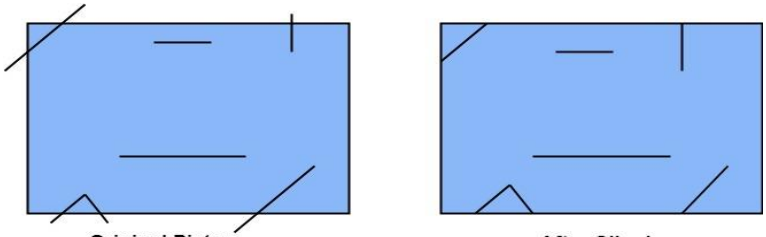



In this figure AB is clipped case.
CD is visible line.
EF is invisible line.

Mixed Lines

Clipping can be applied through hardware as well as software. In some computers, hardware devices automatically do work of clipping. In a system where hardware clipping is not available software clipping applied.

Following figure show before and after clipping



Original Picture
or
Before Clipping

After Clipping

The window against which object is clipped called a clip window. It can be curved or rectangle in shape.

## Applications of clipping:

1. It will extract part we desire.
2. For identifying the visible and invisible area in the 3D object.
3. For creating objects using solid modeling.
4. For drawing operations.
5. Operations related to the pointing of an object.
6. For deleting, copying, moving part of an object.

Clipping can be applied to world co-ordinates. The contents inside the window will be mapped to device co-ordinates. Another alternative is a complete world co-ordinates picture is assigned to device co-ordinates, and then clipping of viewport boundaries is done.

## Types of Clipping:

1. Point Clipping
2. Line Clipping
3. Area Clipping (Polygon)
4. Curve Clipping
5. Text Clipping
6. Exterior Clipping

# Point Clipping:

Point Clipping is used to determining, whether the point is inside the window or not. For this following conditions are checked.

1. $x \leq x_{max}$
2. $x \geq x_{min}$
3. $y \leq y_{max}$
4. $y \geq y_{min}$

(x, y) is coordinate of the point. If anyone from the above inequalities is false, then the point will fall outside the window and will not be considered to be visible.

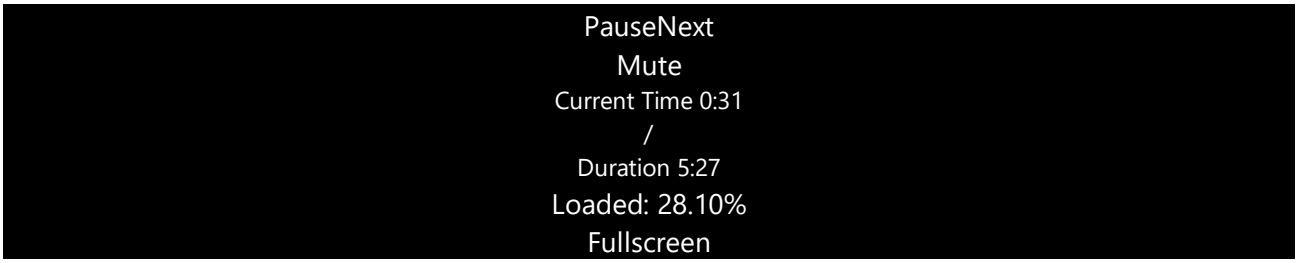
# Line Clipping:

It is performed by using the line clipping algorithm. The line clipping algorithms are:

1. Cohen Sutherland Line Clipping Algorithm

2. Midpoint Subdivision Line Clipping Algorithm
3. Liang-Barsky Line Clipping Algorithm

# Cohen Sutherland Line Clipping Algorithm:




In the algorithm, first of all, it is detected whether line lies inside the screen or it is outside the screen. All lines come under any one of the following categories:

1. Visible
2. Not Visible
3. Clipping Case

**1. Visible:** If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.

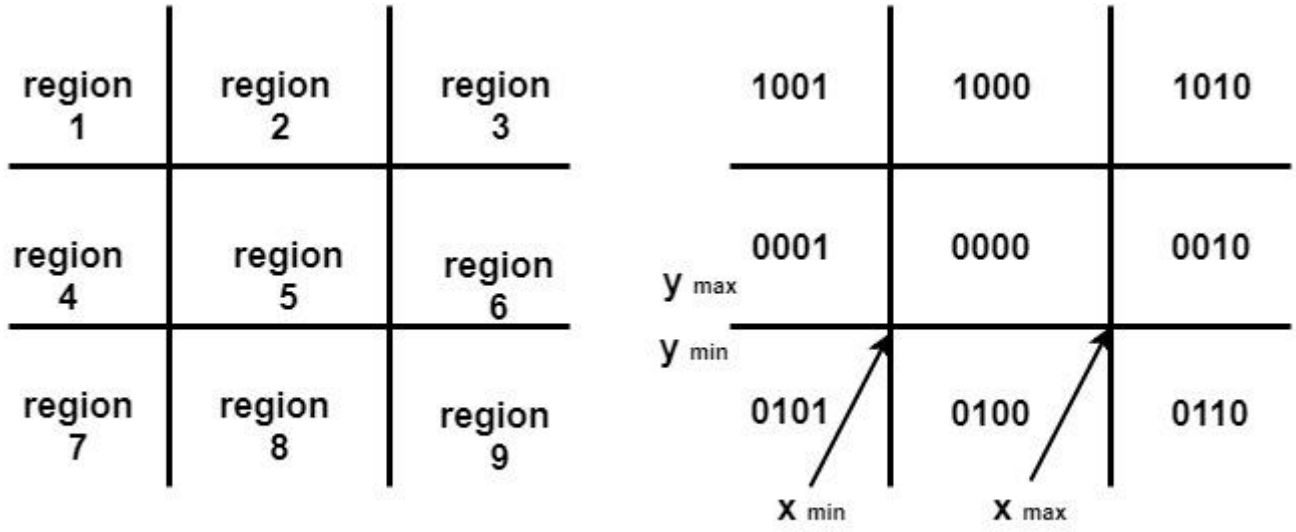
**2. Not Visible:** If a line lies outside the window it will be invisible and rejected. Such lines will not display. If any one of the following inequalities is satisfied, then the line is considered invisible. Let A $(x_1,y_2)$ and B $(x_2,y_2)$ are endpoints of line.

$x_{min}$,$x_{max}$ are coordinates of the window.

$y_{min}$,$y_{max}$ are also coordinates of the window.

$x_1 > x_{max}$
$x_2 > x_{max}$
$y_1 > y_{max}$
$y_2 > y_{max}$
$x_1 < x_{min}$
$x_2 < x_{min}$
$y_1 < y_{min}$
$y_2 < y_{min}$

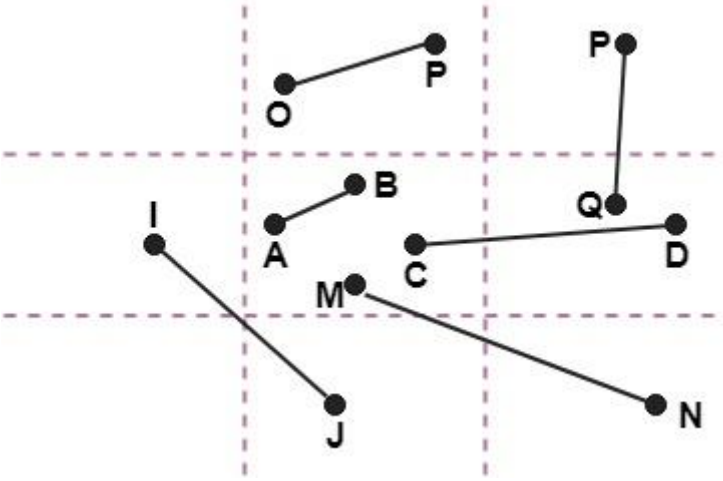
**3. Clipping Case:** If the line is neither visible case nor invisible case. It is considered to be clipped case. First of all, the category of a line is found based on nine regions given below. All nine regions are assigned codes. Each code is of 4 bits. If both endpoints of the line have end bits zero, then the line is considered to be visible.




**9 region** **bits assigned to 9 regions**

The center area is having the code, 0000, i.e., region 5 is considered a rectangle window.

**Following figure show lines of various types**




Line AB is the visible case
Line OP is an invisible case
Line PQ is an invisible line
Line IJ are clipping candidates
Line MN are clipping candidate
Line CD are clipping candidate

# Advantage of Cohen Sutherland Line Clipping:

1. It calculates end-points very quickly and rejects and accepts lines quickly.
2. It can clip pictures much large than screen size.

# Algorithm of Cohen Sutherland Line Clipping:

**Step1:**Calculate positions of both endpoints of the line

**Step2:**Perform OR operation on both of these end-points

**Step3:**If the OR operation gives 0000
Then
line is considered to be visible
else
Perform AND operation on both endpoints
If And ≠ 0000
then the line is invisible
else
And=0000
Line is considered the clipped case.

**Step4:**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 intersects with left boundary of rectangle window
$y_3=y_1+m(x-X_1)$
where $X = X_{wmin}$
where $X_{wmin}$is the minimum value of X co-ordinate of window

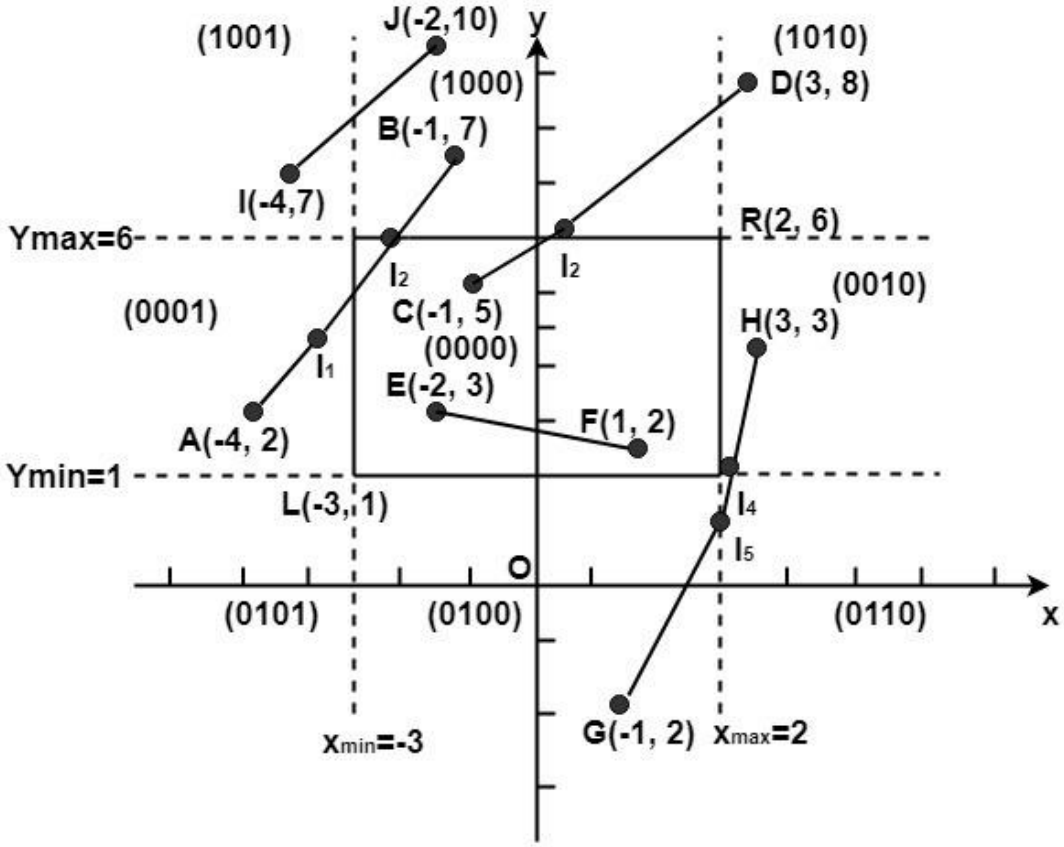
**(b)** If bit 2 is "1" line intersect with right boundary
$y_3=y_1+m(X-X_1)$
where $X = X_{wmax}$
where X more is maximum value of X co-ordinate of the window

**(c)** If bit 3 is "1" line intersects with bottom boundary
$X_3=X_1+(y-y_1)/m$
where $y = y_{wmin}$
$y_{wmin}$ is the minimum value of Y co-ordinate of the window

**(d)** If bit 4 is "1" line intersects with the top boundary
$X_{3=X}1+(y-y_1)/m$
where $y = y_{wmax}$
$y_{wmax}$ is the maximum value of Y co-ordinate of the window

# Example of Cohen-Sutherland Line Clipping Algorithm:

Let R be the rectangular window whose lower left-hand corner is at L (-3, 1) and upper right-hand corner is at R (2, 6). Find the region codes for the endpoints in fig:




The region code for point (x, y) is set according to the scheme

Bit 1 = sign $(y-y_{max})$=sign (y-6) Bit 3 = sign $(x-x_{max})$= sign (x-2)

Bit 2 = sign $(y_{min}-y)$=sign(1-y) Bit 4 = sign $(x_{min}-x)$=sign(-3-x)

Here

$$\text{sign}(a) = \begin{cases} 1 & \text{if a is positive} \\ 0 & \text{otherwise} \end{cases}$$

So

| | |
|---|---|
| A (-4, 2)→ 0001 | F (1, 2)→ 0000 |
| B (-1, 7) → 1000 | G (1, -2) →0100 |
| C (-1, 5)→ 0000 | H (3, 3) → 0100 |
| D (3, 8) → 1010 | I (-4, 7) → 1001 |
| E (-2, 3) → 0000 | J (-2, 10) → 1000 |

We place the line segments in their appropriate categories by testing the region codes found in the problem.

**Category1 (visible):** EF since the region code for both endpoints is 0000.

**Category2 (not visible):** IJ since (1001) AND (1000) =1000 (which is not 0000).

**Category 3 (candidate for clipping):** AB since (0001) AND (1000) = 0000, CD since (0000) AND (1010) =0000, and GH. since (0100) AND (0010) =0000.

The candidates for clipping are AB, CD, and GH.

In clipping AB, the code for A is 0001. To push the 1 to 0, we clip against the boundary line $x_{min}$=-3. The resulting intersection point is $I_1$ (-3,3$\frac{2}{3}$). We clip (do not display) A$I_1$ and $I_1$ B. The code for $I_1$is 1001. The clipping category for $I_1$ B is 3 since (0000) AND (1000) is (0000). Now B is outside the window (i.e., its code is 1000), so we push the 1 to a 0 by clipping against the line $y_{max}$=6. The resulting intersection is $I_2$ (-1$\frac{3}{5}$,6). Thus $I_2$ B is clipped. The code for $I_2$ is 0000. The remaining segment $I_1$ $I_2$ is displayed since both endpoints lie in the window (i.e., their codes are 0000).

For clipping CD, we start with D since it is outside the window. Its code is 1010. We push the first 1 to a 0 by clipping against the line $y_{max}$=6. The resulting intersection $I_3$ is ($\frac{3}{5}$,6),and its code is 0000. Thus $I_3$ D is clipped and the remaining segment C$I_3$ has both endpoints coded 0000 and so it is displayed.

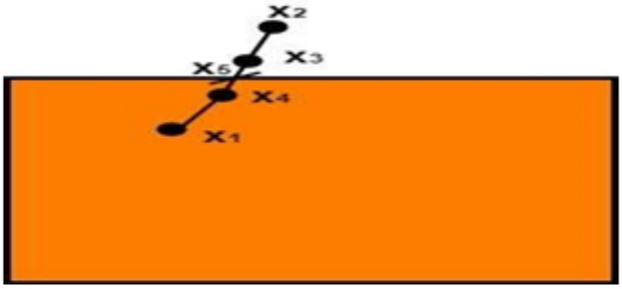
For clipping GH, we can start with either G or H since both are outside the window. The code for G is 0100, and we push the 1 to a 0 by clipping against the line $y_{min}$=1.The resulting intersection point is $I_4$ (2$\frac{1}{5}$,1) and its code is 0010. We clip G$I_4$ and work on $I_4$ H. Segment $I_4$ H is not displaying since (0010) AND (0010) =0010.

# Mid Point Subdivision Line Clipping Algorithm

It is used for clipping line. The line is divided in two parts. Mid points of line is obtained by dividing it in two short segments. Again division is done, by finding midpoint. This process is continued until line of visible and invisible category is obtained. Let ($x_i$,$y_i$) are midpoint.

$$x_m = \frac{x_1+x_2}{2} \quad y_m = \frac{y_1+y_2}{2}$$

$x_5$lie on pointof intersection of boundary of window.




**Step1:** Find $\frac{x_2+x_1}{2}$ i.e. $x_3 = \frac{x_2+x_1}{2}$

**Step2:** Find $x_4 = \frac{x_3+x_1}{2}$

**Step3:** Find $x_5 = \frac{x_4+x_3}{2}$

**Advantage of midpoint subdivision Line Clipping:**

It is suitable for machines in which multiplication and division operation is not possible. Because it can be performed by introducing clipping divides in hardware.

---

**Algorithm of midpoint subdivision Line Clipping:**

**Step1:** Calculate the position of both endpoints of the line

**Step2:** Perform OR operation on both of these endpoints

**Step3:** If the OR operation gives 0000

then

Line is guaranteed to be visible

else

Perform AND operation on both endpoints.

If AND ≠ 0000

then the line is invisible

else

AND=6000

then the line is clipped case.

**Step4:** For the line to be clipped. Find midpoint

$X_m=(x_1+x_2)/2$

$Y_m=(y_1+y_2)/2$

$X_m$ is midpoint of X coordinate. $Y_m$ is midpoint of Y coordinate.

**Step5:** Check each midpoint, whether it nearest to the boundary of a window or not.

**Step6:** If the line is totally visible or totally rejected not found then repeat step 1 to 5.

**Step7:** Stop algorithm.

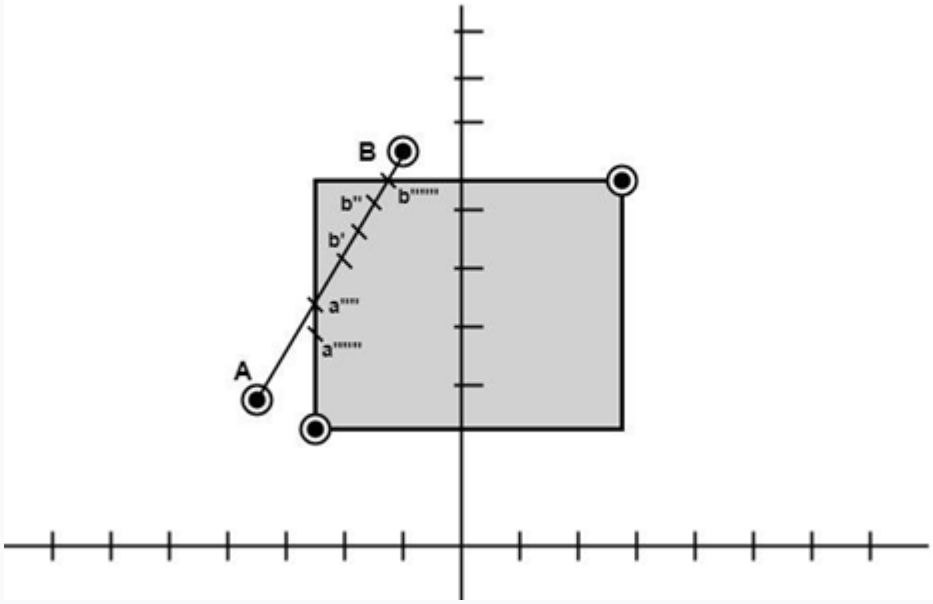
---

**Example:** Window size is (-3, 1) to (2, 6). A line AB is given having co-ordinates of A (-4, 2)and B (-1, 7). Does this line visible. Find the visible portion of the line using midpoint subdivision?

**Solution**

**Step1: Fix point A (-4, 2)**

$$b = \left(\frac{-4+(-1)}{2}, \frac{2+7}{2}\right) = \frac{-5}{2}, \frac{9}{2} = (-2,4)$$




**Step2:** Find b''=mid of b' and b

$$b'' = \left(\frac{-2+(-1)}{2}, \frac{4+7}{2}\right)$$

b''= (-1, 5)

So (-1, 5) is betterthan (2, 4)

Find b''&bb''(-1, 5) b (-1, 7)

$$b'''' = \left(\frac{-1+(-1)}{2}, \frac{5+7}{2}\right)$$

$$b = (-1,6)$$

So B''''to B length of line will be clipped from upper side Now considered left-hand side portion

A and B''''are now endpoints

Find mid of A and B''''

A(-4, 2) B''''(-1, 6)

$$a' = \left(\frac{-4+(-1)}{2}, \frac{2+6}{2}\right)$$

$$a' = (-2.5,4)$$

$$a' = (-2,4)$$

Now good a (-4, 2) and $a'(-2,4)$

$$a'' = \left(\frac{-4+(-2)}{2}, \frac{2+4}{2}\right)$$

$$a'' = (-3,3)$$

Now find mid of $a''$ and a

$a'''''$=a (-4, 2) and $a''(-3, 3)$

$$= \left(\frac{-4+(-3)}{2}, \frac{2+3}{2}\right)$$

$$= \left(\frac{-7}{2}, \frac{5}{2}\right)$$

$$= (-3.5,2.5)$$

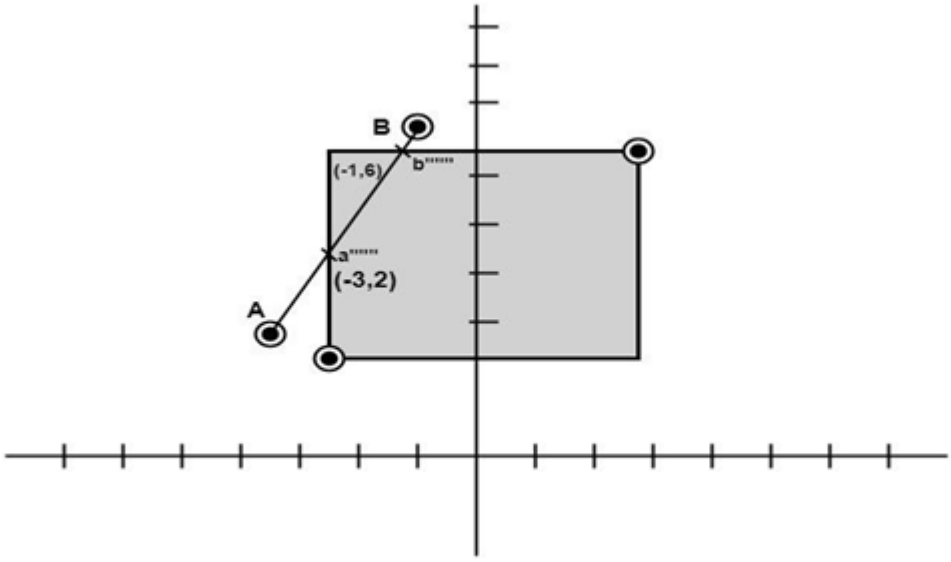
$a'''''$= (-3, 2)

So line from A to $a'''''$ will clipped

Line after clipping from both sides will be $a''''$ to $b''''$

$a'''''$= (-1, 6)

$b'''''$= (-3, 2)



# Polygon Clipping | Sutherland–Hodgman Algorithm

A convex polygon and a convex clipping area are given. The task is to clip polygon edges using the Sutherland–Hodgman Algorithm. Input is in the form of vertices of the polygon in **clockwise order**. **Examples:**

```
Input : Polygon : (100,150), (200,250), (300,200)
        Clipping Area : (150,150), (150,200), (200,200),
                        (200,150) i.e. a Square
Output : (150, 162) (150, 200) (200, 200) (200, 174)
Example 2
Input : Polygon : (100,150), (200,250), (300,200)
        Clipping Area : (100,300), (300,300), (200,100)
Output : (242, 185) (166, 166) (150, 200) (200, 250) (260, 220)
```

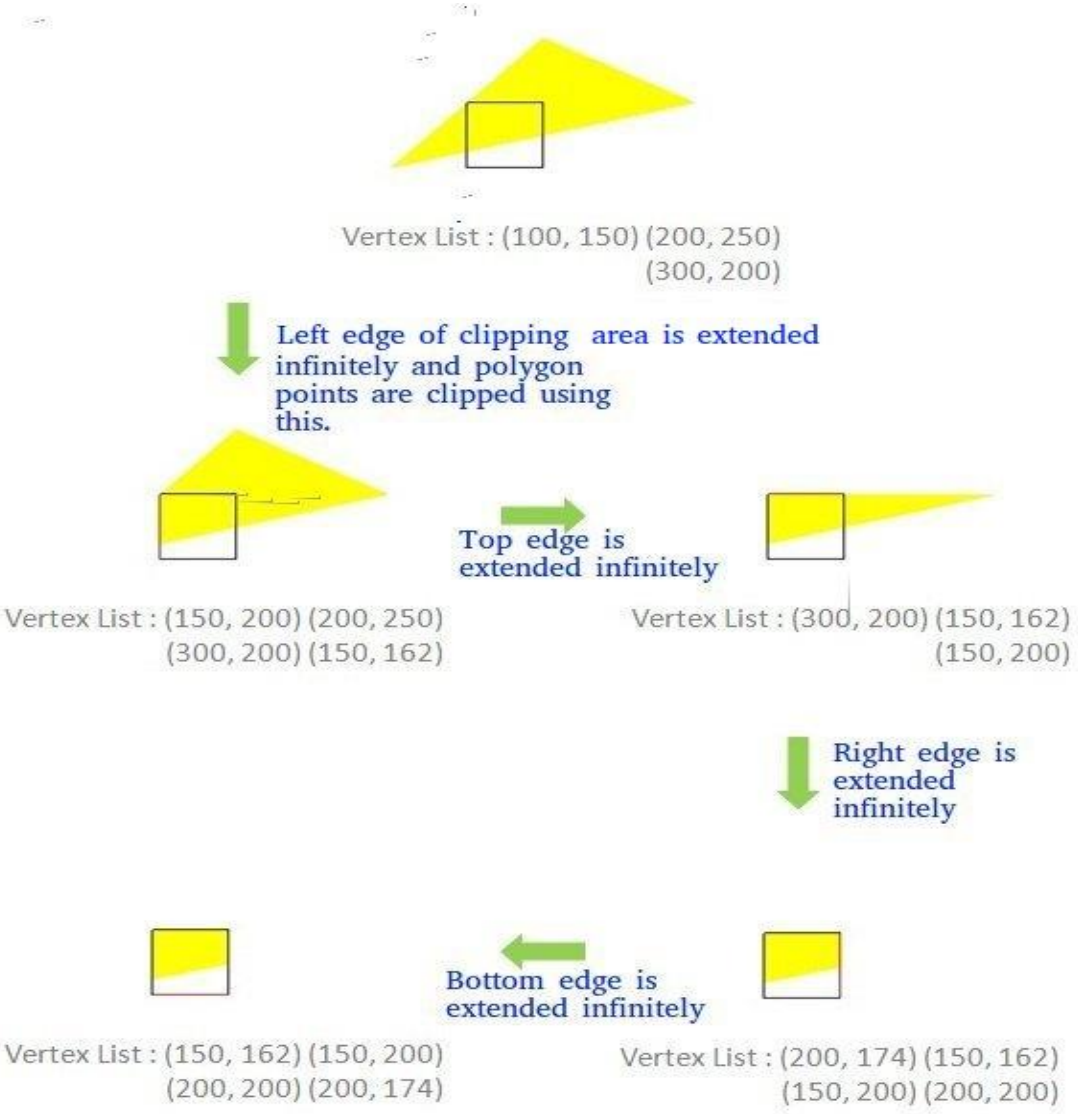
**Overview of the algorithm:**

```
Consider each edge e of clipping Area  and do following:
   a) Clip given polygon against e.
```

**How to clip against an edge of clipping area?** The edge (of clipping area) is extended infinitely to create a boundary and all the vertices are clipped using this boundary. The new list of vertices generated is passed to the next edge of the clip polygon in clockwise fashion until all the edges have been used.

There are four possible cases for any given edge of given polygon against current clipping edge e.

1. **Both vertices are inside :** Only the second vertex is added to the output list
2. **First vertex is outside while second one is inside :** Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
3. **First vertex is inside while second one is outside :** Only the point of intersection of the edge with the clip boundary is added to the output list
4. **Both vertices are outside :** No vertices are added to the output list




There are two sub-problems that need to be discussed before implementing the algorithm:- **To decide if a point is inside or outside the clipper polygon** If the vertices of the clipper polygon are given in clockwise order then all the points lying on the **right side** of the clipper edges are inside that polygon. This can be calculated using :

Given that the line starts from $(x_1, y_1)$ and ends at $(x_2, y_2)$

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

if P<0, the point is on the right side of the line

P=0, the point is on the line

P>0, the point is on the left side of the line

**To find the point of intersection of an edge with the clip boundary** If two points of each line(1,2 & 3,4) are known, then their point of intersection can be calculated using the formula :-

$$(P_x, P_y) = \left( \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

**Sutherland-Hodgeman Polygon Clipping Algorithm :**

- Read coordinates of all vertices of the polygon.
- Read coordinates of the clipping window
- Consider the left edge of the window
- Compare the vertices of each edge of the polygon, individually with the clipping plane
- Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary discussed earlier.
- Repeat the steps 4 and 5 for remaining edges of the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.
- Stop.