Window port and Viewport in Computer Graphics

Capturing images from the real world and displaying them on the screen is an astonishing process, only if we do not know the underlying process. Here, we will be studying how the images are captured. This process is held by the **Window port** and **Viewport** in Computer Graphics.

Window Port

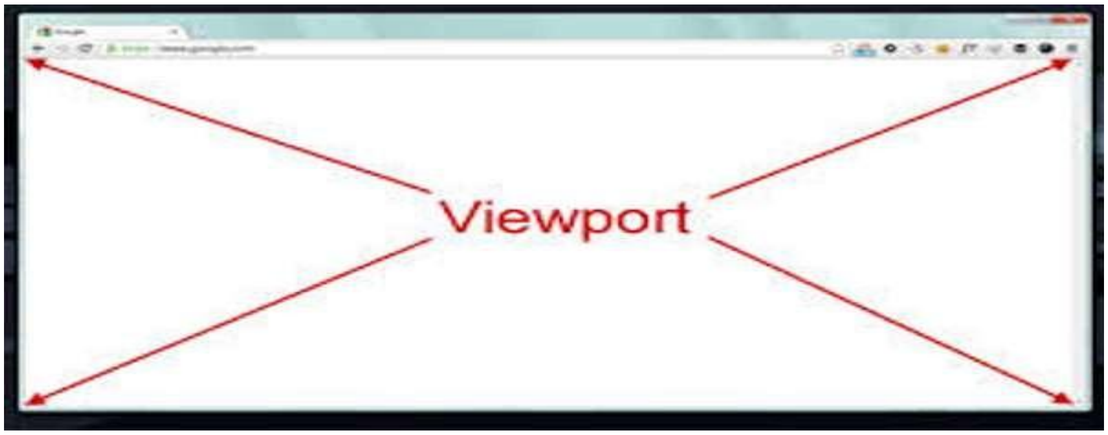
The window port can be confused with the computer window but it isn't the same. The window port is the area chosen from the real world for display. This window port decides what portion of the real world should be captured and be displayed on the screen. ***So, the Window defines what is to be viewed.*** The widow port can thus be defined as,

- ***"A world-coordinate area selected for display is called a window. A window defines a rectangular area in the world coordinates."***
- ***"The Capability to show some part of an object in a window is known as windowing."***
- ***"The rectangular area describes in the world coordinate system is called the window."***

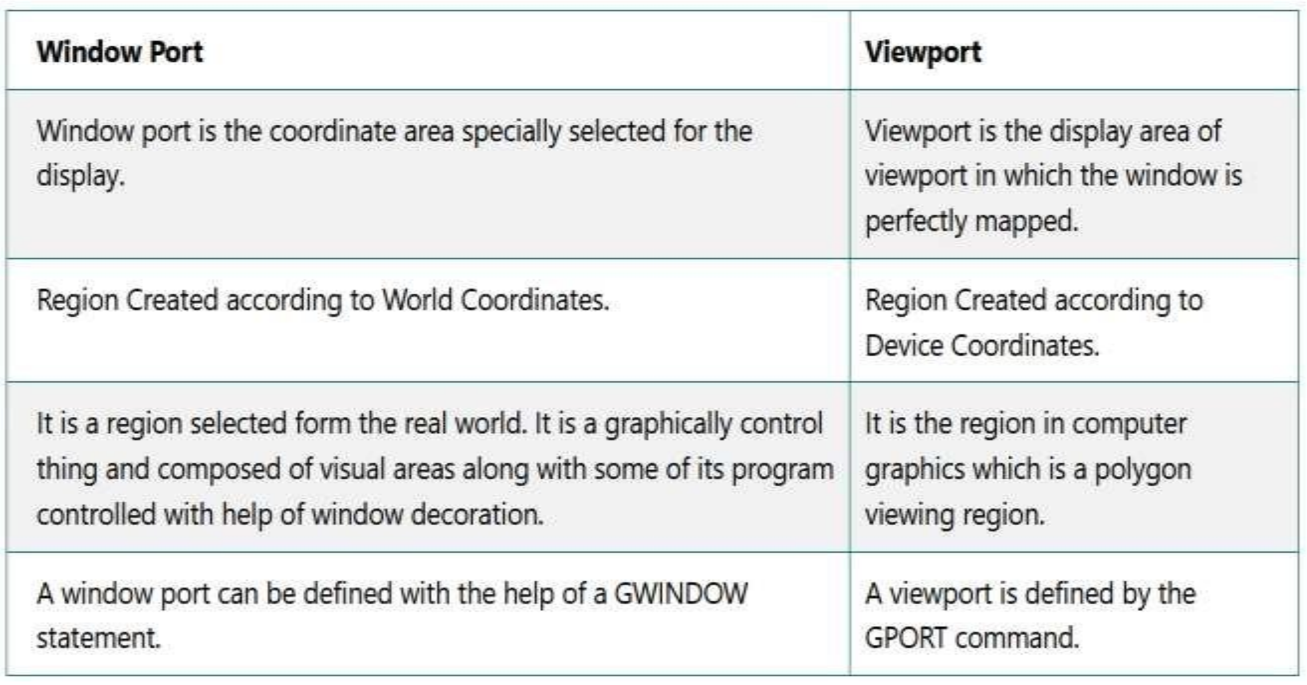
Viewport

Now, the Viewport is the area on a display device to which a window is mapped. **"The viewport can be defined as an area on the screen which is used to display the object"**. ***So, Viewport defines where is to be viewed.*** Thus, the viewport is nothing else but our device's screen. The viewport can thus be defined as follows:

"A viewport is a polygon viewing region in computer graphics. The viewport is an area expressed in rendering-device-specific coordinates, e.g. pixels for screen coordinates, in which the objects of interest are going to be rendered."



<u>Difference between Window Port and Viewport:</u>

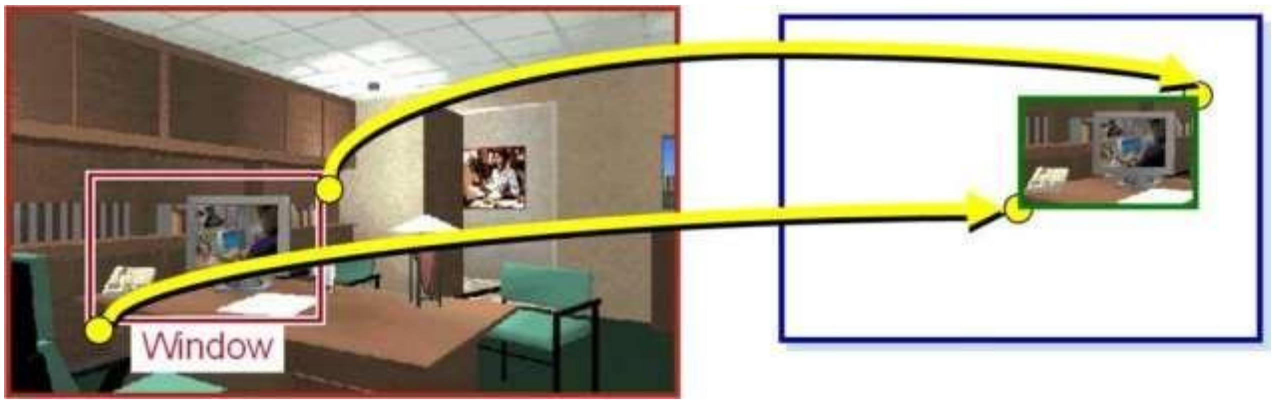


Window Port	Viewport
Window port is the coordinate area specially selected for the display.	Viewport is the display area of viewport in which the window is perfectly mapped.
Region Created according to World Coordinates.	Region Created according to Device Coordinates.
It is a region selected form the real world. It is a graphically control thing and composed of visual areas along with some of its program controlled with help of window decoration.	It is the region in computer graphics which is a polygon viewing region.
A window port can be defined with the help of a GWINDOW statement.	A viewport is defined by the GPORT command.

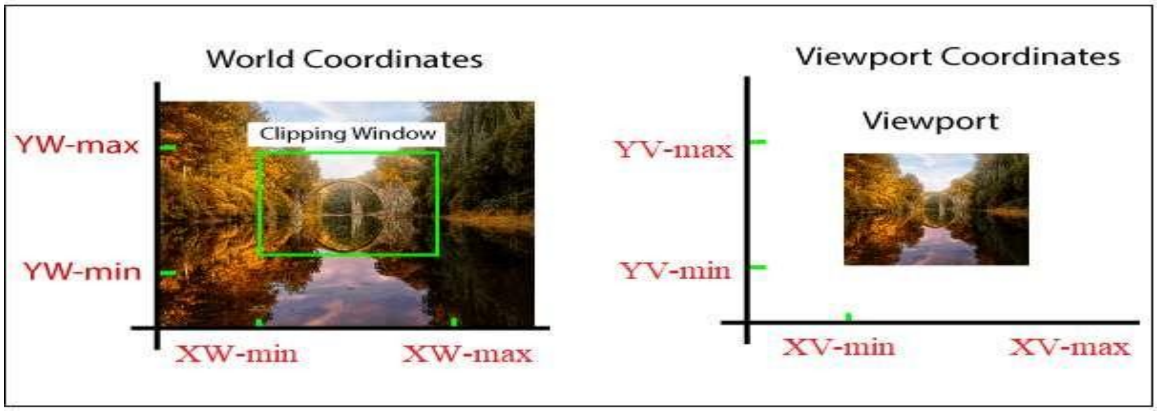
So, to display the image on the computer screen, we must map our window port to the viewport. The capture ratio of the window might not always be similar to or easily adjustable to the viewport. Thus, some necessary transformations adjustments like clipping and cropping are performed on the window.

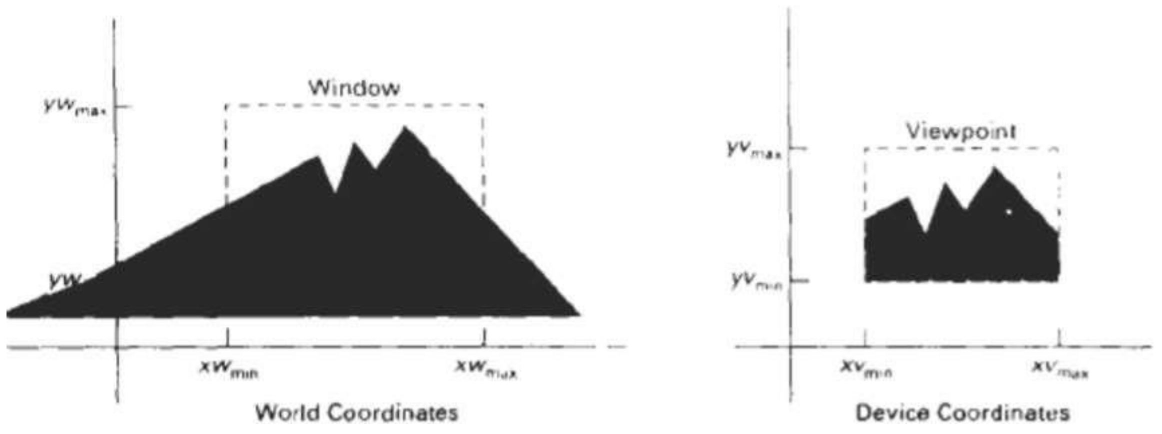
Window to Viewport Transformation:

Now there are two coordinate systems. One is the World Coordinate system. While other is the Device Coordinate system. Therefore, the mapping of a world coordinate scene to device coordinate is Window to Viewport Transformation or Windowing transformation. It is also known as "Viewing Transformation" or "Windowing Transformation". It is defined as follows:



1. Window to Viewport Transformation is the process of transforming a 2D world-coordinate object (Window Port) to device coordinates (Viewport).
2. In particular, objects inside the world or clipping window are mapped to the viewport which is the area on the screen where world coordinates are mapped to be displayed."
3. In other words, the clipping window is used to select the part of the scene that is to be displayed. The viewport then positions the scene on the output device.
4. **Example:**





Thus, we know the basic object description has been changed to the viewing reference frame, so we choose the window, and extend it in viewing coordinates and select or choose the viewport limits in normalized coordinate points.

Many other processes are performed while doing so like maintaining the aspect ratio of the clipping window and the viewport, clipping the excess parts, etc. This all comprises within the Window to the Viewport transformation process. There are various methods and algorithms to do so.

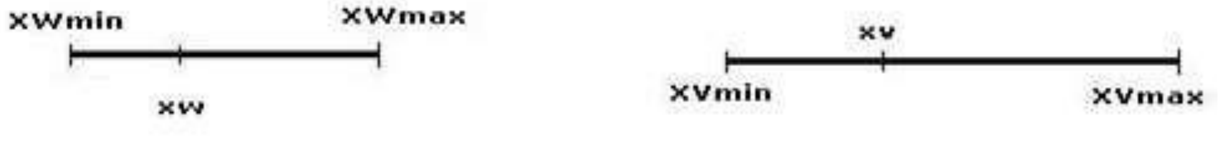
Mathematical Derivation

- This transformation involves developing formulas that start with a point in the world window, say (x_w, y_w).

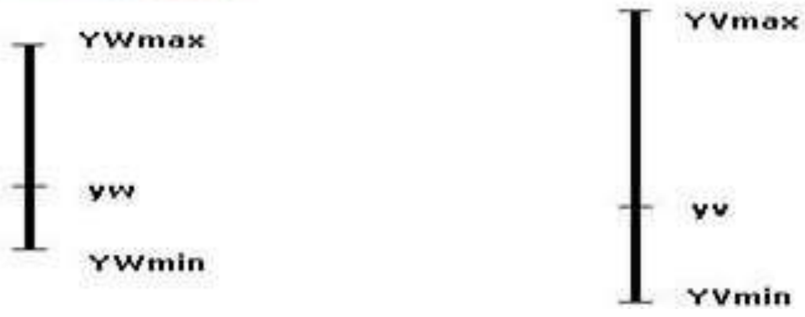
- The formula is used to produce a corresponding point in viewport coordinates, say (x_v, y_v). We would like for this mapping to be "proportional" in the sense that if x_w is 30% of the way from the left edge of the world window, then x_v is 30% of the way from the left edge of the viewport.

- Similarly, if y_w is 30% of the way from the bottom edge of the world window, then y_v is 30% of the way from the bottom edge of the viewport. The picture below shows this proportionality.

For proportionality in x:



For proportionality in y:



Using this proportionality, the following ratios must be equal.

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

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

By solving these equations for the unknown viewport position (xv, yv), the following becomes true:

$x_v = xv_{min} + (x_w - xw_{min})s_x$

$y_v = yv_{min} + (y_w - yw_{min})s_y$

The scale factors (S_x, S_y) would be:

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

And the translation factors (Tx, Ty) would be:

$$t_x = \frac{xw_{max}xv_{min} - xw_{min}xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max}yv_{min} - yw_{min}yv_{max}}{yw_{max} - yw_{min}}$$

Note:

1. The position of the viewport can be changed allowing objects to be viewed at different positions on the Interface Window.
2. Multiple viewports can also be used to display different sections of a scene at different screen positions. Also, by changing the dimensions of the viewport, the size and proportions of the objects being displayed can be manipulated.
3. Thus, a zooming affect can be achieved by successively mapping different dimensioned clipping windows on a fixed sized viewport.
4. If the aspect ratio of the world window and the viewport are different, then the image may look distorted.

Example 01

Let's assume, for window, Xwmin = 20, Xwmax = 80, Ywmin = 40, Ywmax = 80.

for viewport, Xvmin = 30, Xvmax = 60, Yvmin = 40, Yvmax = 60.

Now a point (Xw, Yw) be (30, 80) on the window. We have to calculate that point on the viewport i.e (Xv, Yv).

First of all, calculate the scaling factor of x coordinate Sx and the scaling factor of y coordinate Sy using the above-mentioned formula.

Sx = (60 - 30) / (80 - 20) = 30 / 60

Sy = (60 - 40) / (80 - 40) = 20 / 40

So, now calculate the point on the viewport (Xv, Yv).

Xv = 30 + (30 - 20) * (30 / 60) = 35

Yv = 40 + (80 - 40) * (20 / 40) = 60

So, the point on window (Xw, Yw) = (30, 80) will be (Xv, Yv) = (35, 60) on viewport.

Example 02

Let

$$s_x = \frac{vx_{\max} - vx_{\min}}{wx_{\max} - wx_{\min}} \qquad \text{and} \qquad s_y = \frac{vy_{\max} - vy_{\min}}{wy_{\max} - wy_{\min}}$$

Express window-to-viewport mapping in the form of a composite transformation matrix.

SOLUTION

$$N = \begin{pmatrix} 1 & 0 & vx_{\min} \\ 0 & 1 & vy_{\min} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -wx_{\min} \\ 0 & 1 & -wy_{\min} \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} s_x & 0 & -s_x wx_{\min} + vx_{\min} \\ 0 & s_y & -s_y wy_{\min} + vy_{\min} \\ 0 & 0 & 1 \end{pmatrix}$$

Example 03

Find the normalization transformation that maps a window whose lower left corner is at (1, 1) and upper right corner is at (3, 5) onto (*a*) a viewport that is the entire normalized device screen and (*b*) a viewport that has lower left corner at (0, 0) and upper right corner $(\frac{1}{2}, \frac{1}{2})$.

SOLUTION

(*a*) The window parameters are $wx_{\min} = 1$, $wx_{\max} = 3$, $wy_{\min} = 1$, and $wy_{\max} = 5$. The viewport parameters are $vx_{\min} = 0$, $vx_{\max} = 1$, $vy_{\min} = 0$, and $vy_{\max} = 1$. Then $s_x = \frac{1}{2}$, $s_y = \frac{1}{4}$, and

$$N = \begin{pmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & \frac{1}{4} & -\frac{1}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

(*b*) The window parameters are the same as in (*a*). The viewport parameters are now $vx_{\min} = 0$, $vx_{\max} = \frac{1}{2}$, $vy_{\min} = 0$, $vy_{\max} = \frac{1}{2}$. Then $s_x = \frac{1}{4}$, $s_y = \frac{1}{8}$, and

$$N = \begin{pmatrix} \frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & \frac{1}{8} & -\frac{1}{8} \\ 0 & 0 & 1 \end{pmatrix}$$

Example 04

Find the complete viewing transformation that maps a window in world coordinates with x extent 1 to 10 and y extent 1 to 10 onto a viewport with x extent $\frac{1}{4}$ to $\frac{3}{4}$ and y extent 0 to $\frac{1}{2}$ in normalized device space, and then maps a workstation window with x extent $\frac{1}{4}$ to $\frac{1}{2}$ and y extent $\frac{1}{4}$ to $\frac{1}{2}$ in the normalized device space into a workstation viewport with x extent 1 to 10 and y extent 1 to 10 on the physical display device.

SOLUTION

From Prob. 5.1, the parameters for the normalization transformation are $wx_{\min} = 1$, $wx_{\max} = 10$, $wy_{\min} = 1$, $wy_{\max} = 10$, and $vx_{\min} = \frac{1}{4}$, $vx_{\max} = \frac{3}{4}$, $vy_{\min} = 0$, and $vy_{\max} = \frac{1}{2}$. Then

$$s_x = \frac{1/2}{9} = \frac{1}{18} \qquad s_y = \frac{1/2}{9} = \frac{1}{18}$$

and

$$N = \begin{pmatrix} \frac{1}{18} & 0 & \frac{7}{36} \\ 0 & \frac{1}{18} & -\frac{1}{18} \\ 0 & 0 & 1 \end{pmatrix}$$

The parameters for the workstation transformation are $wx_{\min} = \frac{1}{4}$, $wx_{\max} = \frac{1}{2}$, $wy_{\min} = \frac{1}{4}$, $wy_{\max} = \frac{1}{2}$, and $vx_{\min} = 1$, $vx_{\max} = 10$, $vy_{\min} = 1$, and $vy_{\max} = 10$. Then

$$s_x = \frac{9}{1/4} = 36 \qquad s_y = \frac{9}{1/4} = 36$$

and

$$W = \begin{pmatrix} 36 & 0 & -8 \\ 0 & 36 & -8 \\ 0 & 0 & 1 \end{pmatrix}$$

The complete viewing transformation V is

$$V = W \cdot N = \begin{pmatrix} 36 & 0 & -8 \\ 0 & 36 & -8 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{18} & 0 & \frac{7}{36} \\ 0 & \frac{1}{18} & -\frac{1}{18} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -1 \\ 0 & 2 & -10 \\ 0 & 0 & 1 \end{pmatrix}$$

Example 05

Find normalization transformation that maps a window whose lower-left corner is at (1,1) and upper right corner is at (3,5) onto: a) Viewport with lower-left corner (0,0) and upper right corner (1,1) b) Viewport with lower left corner (0,0) and upper right corner (1/2,1/2).

Solution:

$wx_{min}=1$, $wy_{min}=1$, $wx_{max}=3$, $wy_{max}=5$

a)

$vx_{min}=0$, $vy_{min}=0$, $vx_{max}=1$, $vy_{max}=1$

$Sx = (vx_{max} - vx_{min}) / (wx_{max} - wx_{min})$, $Sy = (vy_{max} - vy_{min}) / (wy_{max} - wy_{min})$

While, Putting the values:

$Sx = 1-0/3-1 = 1/2$,

$Sy=1-0/5-1 = 1/4$

The normalized form:

$$N=\begin{bmatrix}1 & 0 & vx_{min}\\0 & 1 & vy_{min}\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}sx & 0 & 0\\0 & sy & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -wx_{min}\\0 & 1 & -wy_{min}\\0 & 0 & 1\end{bmatrix}$$

$$N=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1/2 & 0 & 0\\0 & 1/4 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -1\\0 & 1 & -1\\0 & 0 & 1\end{bmatrix}$$

$$N=\begin{bmatrix}1/2 & 0 & -1/2\\0 & 1/4 & -1/4\\0 & 0 & 1\end{bmatrix}$$

b) $vx_{min}=0$, $vy_{min}=0$, $vx_{max}=1/2$, $vy_{max}=1/2$

$Sx = (vx_{max} - vx_{min}) / (wx_{max} - wx_{min})$, $Sy = (vy_{max} - vy_{min}) / (wy_{max} - wy_{min})$

While, Putting the values:

$Sx = (1/2-0)/3-1 = (1/2)/2 = 1/4$

$Sy= (1/2-0)/5-1 = (1/2)/4 = 1/8$

The normalized form:

$$N = \begin{bmatrix} 1 & 0 & vx_{\min} \\ 0 & 1 & vy_{\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -wx_{\min} \\ 0 & 1 & -wy_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/4 & 0 & 0 \\ 0 & 1/8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$N = \begin{bmatrix} 1/4 & 0 & -1/4 \\ 0 & 1/8 & -1/8 \\ 0 & 0 & 1 \end{bmatrix}$$

Clipping in Computer Graphics

The objective of clipping is to determine which portion of a scene is visible within the window. For viewing transformation, only these portions are retained for display and everything outside the window are to be clipped.

Applications of clipping

The clipping process finds its wide use in computer graphics. As we have already seen its very important and crucial role in the window to view port transformation, the following are some the areas in this process where the clipping finds its application:

1. To extract the part of image that we desire.
2. To identify the invisible and visible area in the 3D object plane.
3. For enabling zooming feature in our screens.
4. For creating objects using solid modeling.
5. For drawing operations.
6. To perform the operations related to the pointing of an object.
7. To delete, copy, or move part of an object.

Types of lines

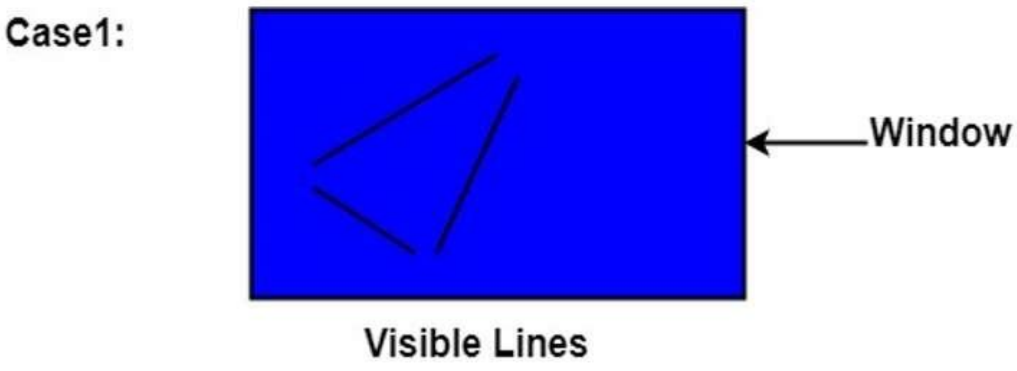
The lines are divided into three types.

- **Visible Line:** The line lying inside the view pane, is a visible line.
- **Invisible Line:** The line lying outside the view pane, is an invisible line.
- **Clipped Line:** **"**A line that lies inside or outside the window is called clipped line.**"** A point where the line cut the view pane is known as the Intersection point of the line

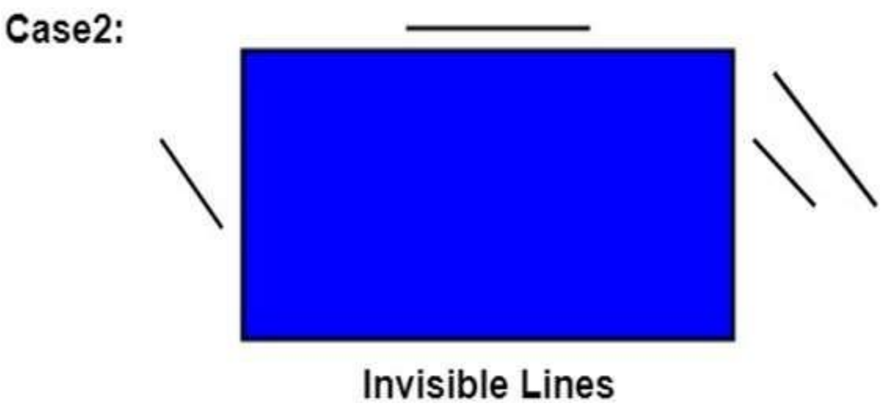
Conditions of Lines while performing window to view port transformation

There are three conditions that may occur while performing window to view port transformation:

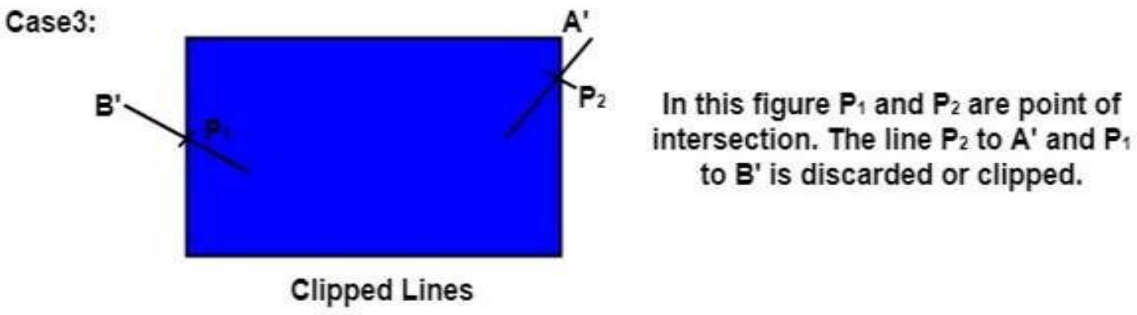
Case 1: A simple line or lines entirely lie inside the window. Such lines are considered as visible lines.



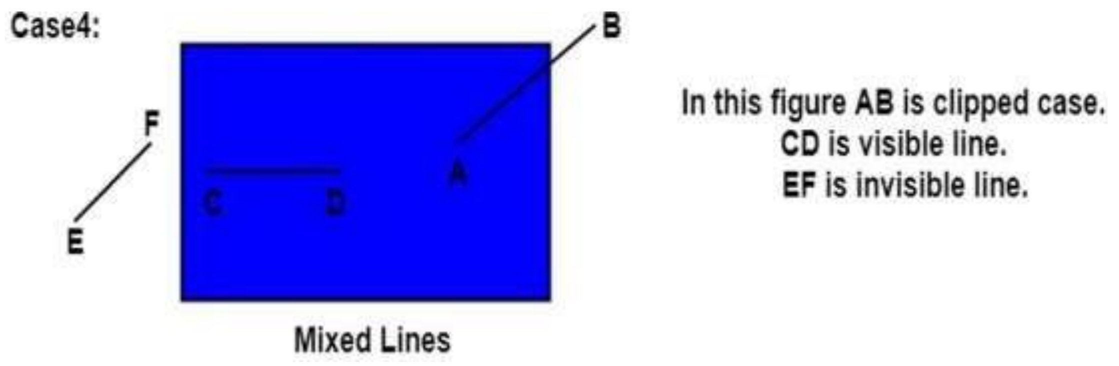
Case 2: A simple line lies entirely outside the window. Such lines are considered as invisible lines.



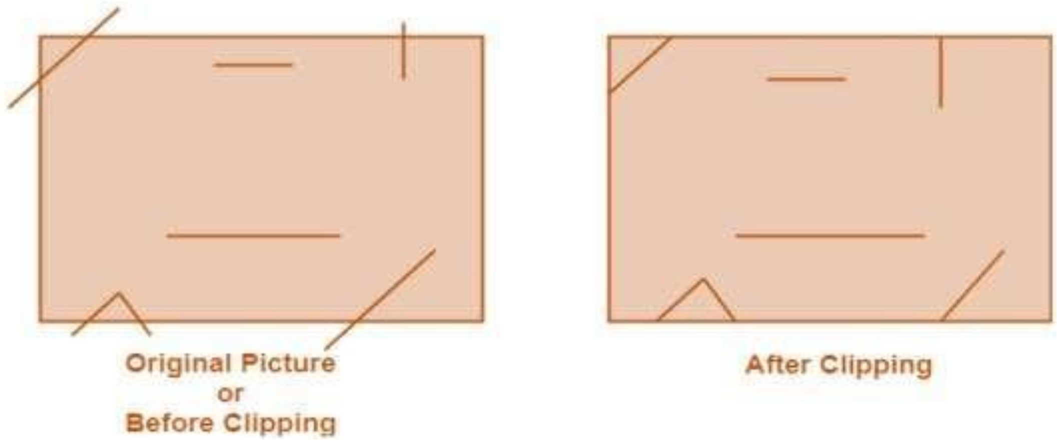
Case 3: A simple line partially lies inside the window and partially outside. In such conditions, we need to perform clipping. Thus, the clipping point of the intersection of a line with the window is determined.


Case3:
A'
B'
P1
P2
In this figure P1 and P2 are point of intersection. The line P2 to A' and P1 to B' is discarded or clipped.
Clipped Lines

So, for deciding the visible and invisible portion from the picture, clipping is performed on it. Clipping determines each element into the visible and invisible portion by omitting the extra part. So, the visible portion is selected. And an invisible portion is discarded.



The clipping can be performed either by hardware devices or by software devices. However, the clipping through hardware devices was used in earlier times. Nowadays, we use software clipping. The basic idea of clipping can be understood with the following diagram:



Types of Clipping

The types of clipping are determined by the shape that is clipped. The following are the types of clipping.

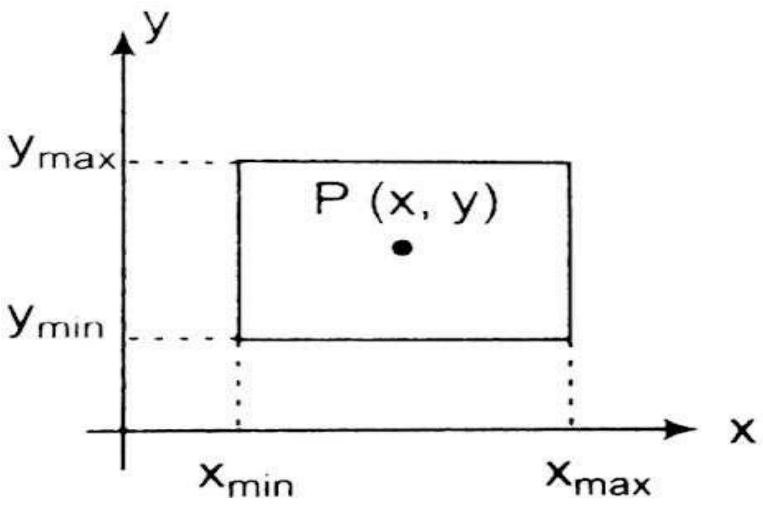
1. Point Clipping: Performed on a point
2. Line Clipping: Performed on a line
3. Area Clipping (Polygon): Performed on various types of polygons
4. Curve Clipping: Performed on curved lines and shapes

5. Text Clipping: Performed on texts.
6. Exterior Clipping: Performed without knowing about the content of the image.

Point Clipping

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

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



Line Clipping

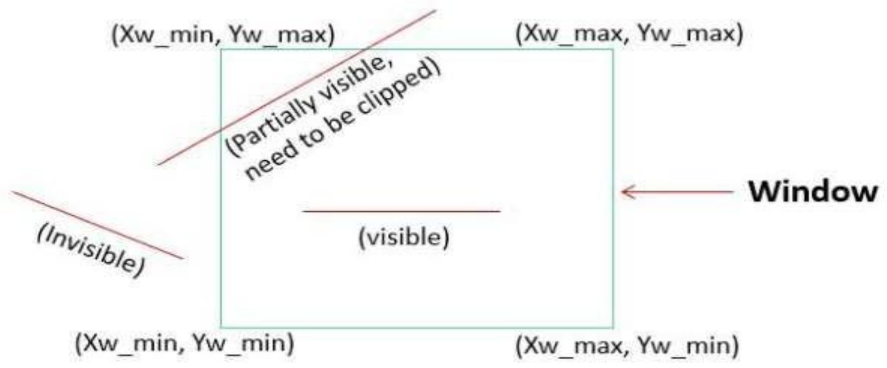
We can define the line clipping algorithm to set the portion of the line to be seen into the define viewport area. But due to the presence of two points (i.e., end points) it is different from point clipping. To decide whether the line can be viewed or not or whether some portion of the line can be viewed on the viewport, the following steps are followed:

If both the endpoints of a line are outside the viewing area, it can’t be displayed.

And if both the endpoints of a line are inside the viewing area, the complete line will be visible.

If some portions of the line or one or two endpoints is outside the viewing area, the portion of the line is to be displayed.

This can be illustrated by the following figure.



Line Algorithm

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

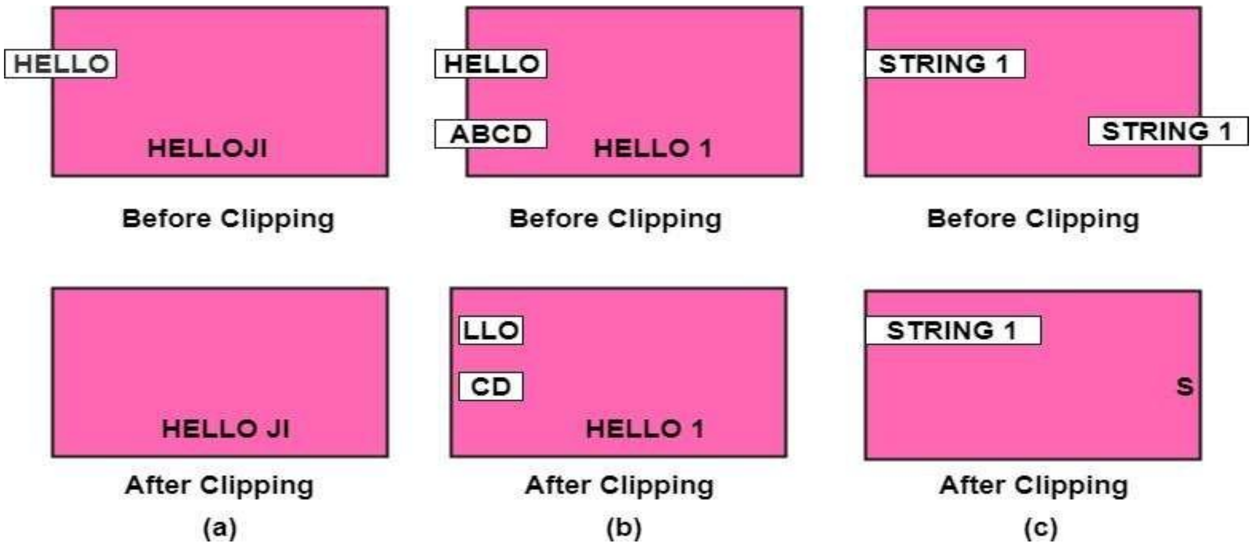
1. Cohen Sutherland Line Clipping Algorithm
2. Midpoint Subdivision Line Clipping Algorithm
3. Liang-Barsky Line Clipping Algorithm

Text Clipping

Several methods are available for clipping of text. Clipping method is dependent on the method of generation used for characters. A simple method is completely considered, or nothing considers method. This method is also called as all or none. If all characters of the string are inside window, then we will keep the string, if a string character is outside then whole string will be discarded in fig (a).

Another method is discarded those characters not completely inside the window. If a character overlap boundary of window. Those will be discarded in fig (b).

In fig (c) individual character is treated. Character lies on boundary is discarded as which it is outside the window.



Polygon Clipping

Polygon clipping is applied to the polygons. The term polygon is used to define objects having outline of solid. These objects should maintain property and shape of polygon after clipping.

"A Polygon can be described as the enclosed collection or group of the lines."

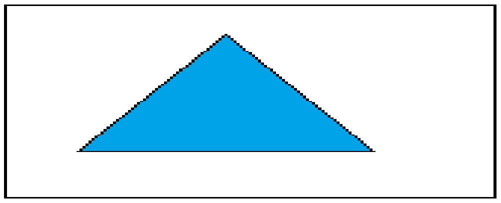
In a polygon, all lines are connected. Lines can be a combination of edges and vertices, which together form a polygon. A polygon refers to a two-dimensional architecture made up of a number of straight lines.

Some Examples of the polygon:

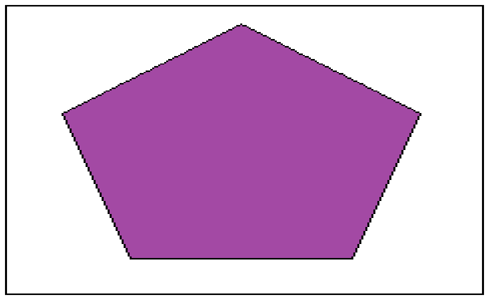
- Triangles
- Pentagons
- Hexagons
- Quadrilaterals

The polygon's name defines how many sides the architecture contains.

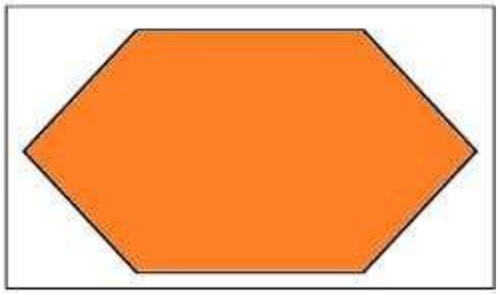
Triangle: It has three sides.



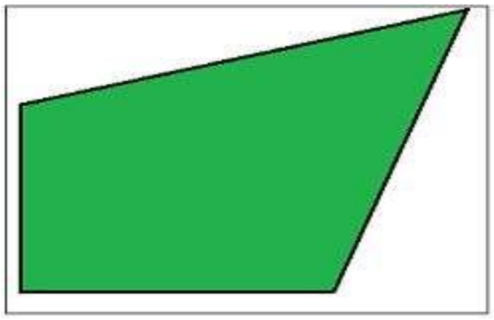
Pentagon: A pentagon has five sides.



Hexagon: It contains six sides.



Quadrilaterals: It contains four sides.

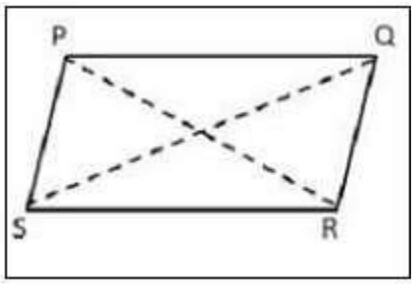


Types of Polygon

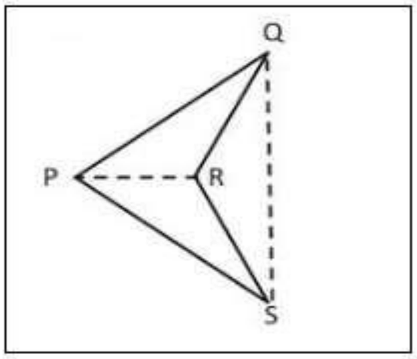
There are two basic types of polygon-

- Concave Polygon
- Convex Polygon

Concave Polygon: The concave polygon does not have any part of its diagonals in its exterior. In a concave polygon, at least one angle should be greater than 180° (angle >180°).



Convex Polygon: The convex polygon has at least one part of diagonal in its exterior. In a convex polygon, all the angles should be less than 180° (angle<180°).



Polygon Clipping

Polygon clipping is a process in which we only consider the part which is inside the view pane or window. We will remove or clip the part that is outside the window. We will use the following algorithms for polygon clipping–

1. Sutherland-Hodgeman polygon clipping algorithm
2. Weiler-Atherton polygon clipping algorithm

Curve Clipping

Curve Clipping involves complex procedures as compared to line clipping. Curve clipping requires more processing than for object with linear boundaries. Consider window which is rectangular in shape. The circle is to consider against rectangle window. If circle is completely inside boundary of the window, it is considered visible. So, save the circle. If a circle is in outside window, discard it. If circle cut the boundary then consider it to be clipping case.

Exterior Clipping

It is opposite to previous clipping. Here picture which is outside the window is considered. The picture inside the rectangle window is discarded. So, part of the picture outside the window is saved.

Uses of Exterior Clipping:

- It is used for displaying properly the pictures which overlap each other.
- It is used in the concept of overlapping windows.
- It is used for designing various patterns of pictures.
- It is used for advertising purposes.
- It is suitable for publishing.
- For designing and displaying of the number of maps and charts, it is also used.

Cohen Sutherland Line Clipping

Cohen Sutherland Line Clipping algorithm is quite interesting. The clipping problem is simplified by dividing the area surrounding the window region into four segments Up, Down, Left, Right (U, D, L, R) and assignment of number 1 and 0 to respective segments helps in positioning the region surrounding the window. How this positioning of regions is performed can be well understood by considering *Figure 1*.

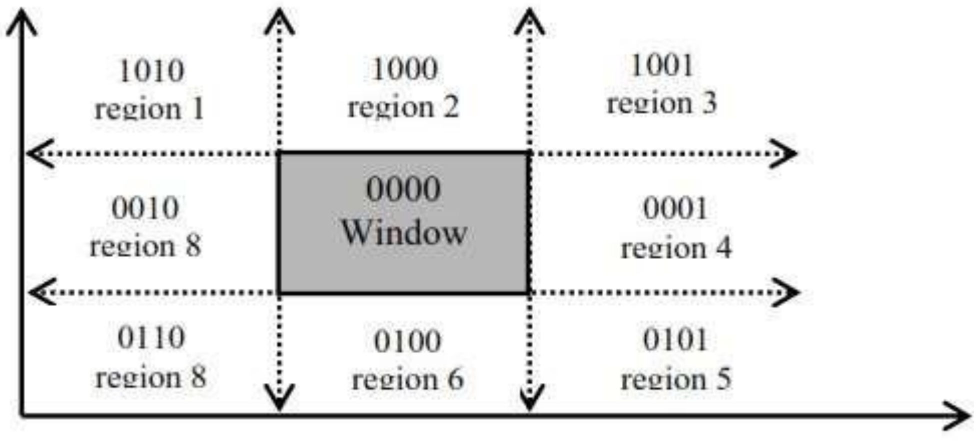


Figure – 01

In *Figure 1* you might have noticed, that all coding of regions U, D, L, R is done with respect to window region. As window is neither Up nor Down, neither Left nor Right, so, the respective bits UDLR are 0000; now see region 1 of *Figure 1*. The positioning code UDLR is 1010, i.e., the region 1 lying on the position which is upper left side of the window. Thus, region 1 has UDLR code 1010 (Up so U=1, not Down so D=0, Left so L=1, not Right so R=0).

The meaning of the UDLR code to specify the location of region with respect to window is: 1st bit ⇒ Up (U); 2nd bit ⇒ Down (D); 3rd bit ⇒ Left (L); 4th bit ⇒ Right (R)

Now, to perform Line clipping for various line segment which may reside inside the window region fully or partially, or may not even lie in the widow region; we use the tool of logical AND-ing between the UDLR codes of the points lying on the line.

Logical ANDing (^) operation between respective bits implies

1 ^ 1 = 1; 1 ^ 0 = 0; 0 ^ 1 = 0; 0 ^ 0 = 0

Note:

UDLR code of window is 0000 always and w.r.t. this will create bit codes of other regions.

A line segment is visible if both the UDLR codes of the end points of the line segment equal to 0000 i.e. UDLR code of window region. If the resulting code is not 0000 then, that line segment or section of line segment may or may not be visible.

Now, let us study how this clipping algorithm works. For the sake of simplicity, we will tackle all the cases with the help of example lines l_1 to l_5 shown in *Figure 2*.
Each line segment represents a case.

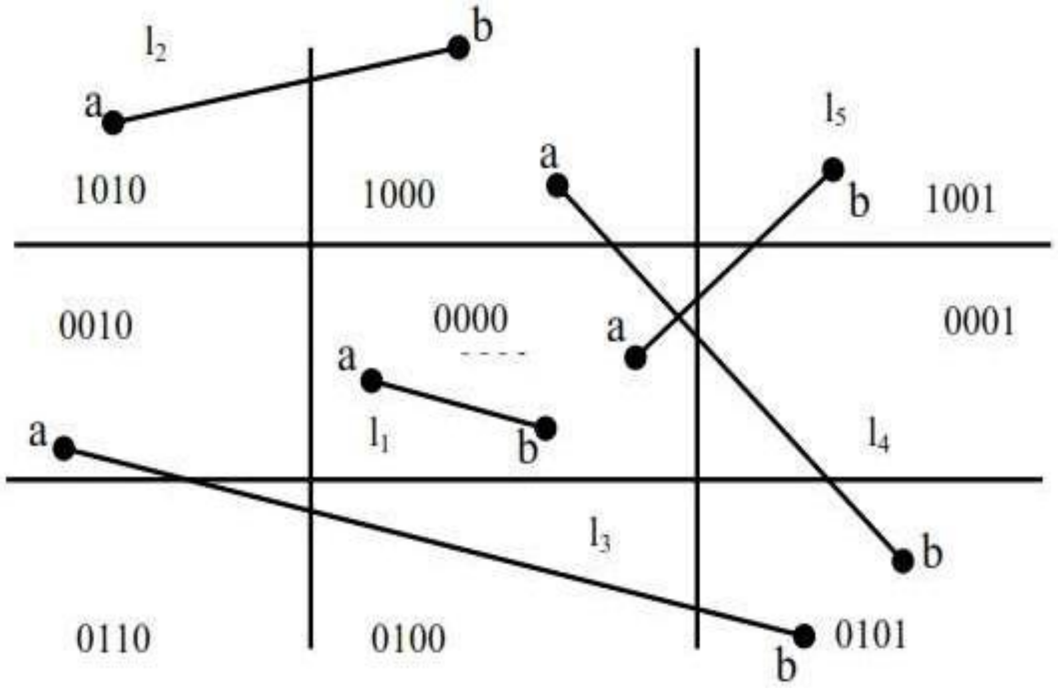


Figure – 02

Note, that in *Figure 2*, line l_1 is completely visible, l_2 and l_3 are completely invisible; l_4 and l_5 are partially visible. We will discuss these out comings as three separate cases.

Case 1: $l_1 \rightarrow$ completely visible, i.e., Trivial acceptance (both points lie inside the window)

Case 2: l_2 and $l_3 \rightarrow$ Invisible, i.e., Trivial acceptance rejection

Case 3: l_4 and $l_5 \rightarrow$ partially visible (partially inside the window)

Now, let us examine these three cases with the help of this algorithm:

Case 1 (Visible): (Trivial acceptance case) When both points (starting and ending) of the line are entirely situated inside the window. *If the UDLR bit codes of the end points P, Q of a given line is 0000 then line is completely visible.* Here this is the case as the end points a and b of line l_1 are: a (0000), b (0000). If this trivial acceptance test is failed then, the line segment PQ is passed onto Case 2.

Case 2 (Invisible): (Trivial Rejection Case) When both points (Starting and ending) of the line are completely situated outside the window. *If the logical intersection (AND) of the bit codes of the end points P, Q of the line segment is not 0000 then line segment is not visible or is rejected.*

Note that, in *Figure 2*, line 2 is completely on the top of the window but line 3 is neither on top nor at the in bottom plus, either on the LHS nor on the RHS of the window. We use the standard formula of logical ANDing to test the non-visibility of the line segment.

So, to test the visibility of line 2 and 3 we need to calculate the logical intersection of end points for line 2 and line 3.

line l_2: bit code of end points is 1010 and 1000 logical intersection of end points = (1010) ^ (1000) = 1000 as logical intersection i.e., not 0000. So, line 2 will be invisible.

line l_3: end points have bit codes 0010 and 0101. Now logical intersection = 0000, i.e., 0010 ^ 0101 = 0000 from the *Figure 2*, the line is invisible. Similarly, in line 4 one end point is on top and the other on the bottom so, logical intersection is 0000 but then it is partially visible, same is true with line 5. These are special cases and we will discuss them in case 3.

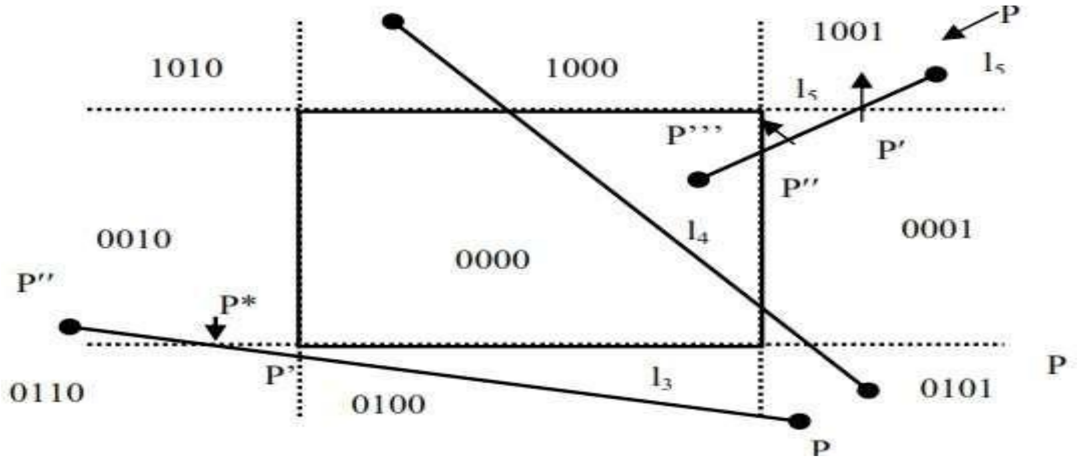


Figure-03

Case 3 (clipping Candidate): The line is in neither case 1 nor 2. *If the logical intersection (AND) of the bit codes of the end points P, Q of the line segment is 0000 then line segment is clipping candidate.*

Suppose for the line segment PQ, both the trivial acceptance and rejection tests failed (i.e., Case 1 and Case 2 conditions do not hold, this is the case for l_3, l_4 and l_5 line segments) shown above in *Figure 3*.

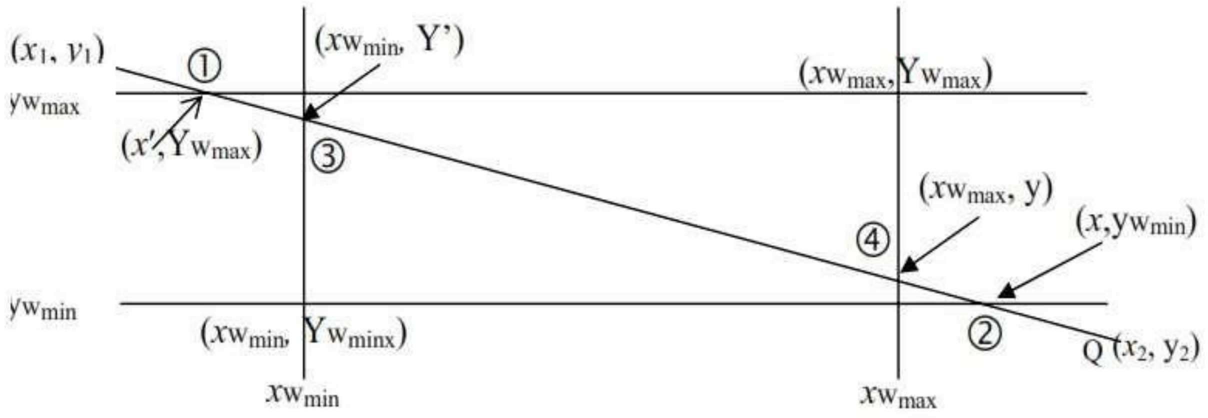
For such non-trivial Cases the Cohen Sutherland line clipping algorithm is processed, as follows:

Since, both the bit codes for the end points P, Q of the line segment cannot be equal to 0000. Let us assume that the starting point of the line segment is P whose bit code is not equal to 0000. For example, for the line segment l we choose P to be the bit codes 1001. Now, scan the bit code of P from the first bit to the fourth bit and find the position of the bit at which the bit value 1 appears at the first time. For the line segment l_5 it appears at the very first position.

If the bit value 1 occurs at the first position then proceed to intersect the line segment with the UP edge of the window and assign the first bit value to its point of intersection as 0.

Similarly, if the bit value 1 occurs at the second position while scanning the bit values at the first time then intersect the line segment PQ with the Down edge of the window and so on.

This point of intersection may be labeled as P’. Clearly the line segment PP’ is outside the window and therefore rejected and the new line segment considered for dipping will be P’Q. The coordinates of P’ and its remaining new bit values are computed. Now, by taking P as P’, again we have the new line segment PQ which will again be to Case 1 for clipping.



Geometrical study of the above type of clipping (it helps to find point of intersection of line PQ with any edge).

Let (x_1, y_1) and (x_2, y_2) be the coordinates of P and Q respectively.

- **Top case /above case:**

  If $y_1 > yw_{max}$ then 1st bit of bit code = 1 (signifying above) else bit code = 0
- **Bottom case / below case:**

  If $y_1 < yw_{min}$ then 2nd bit = 1 (i.e. below) else bit = 0
- **Left case:**

  If $x_1 < xw_{min}$ then 3rd bit = 1 (i.e. left) else bit = 0
- **Right case:**

  If $x_1 > xw_{max}$ then 4th bit = 1 (i.e. right) else bit = 0

(Or) we can calculate another way

Bit 1 = sign $(y-y_{max})$ Bit 2 = sign $(y_{min}-y)$

Bit 3 = sign $(x-x_{max})$ Bit 4 = sign $(x_{min}-x)$

Here

$$\left\{ \begin{array}{lll} \text{sign (a)} = & 1 & \text{if a is positive} \\ & 0 & \text{otherwise} \end{array} \right\}$$

Limitations of Cohen-Sutherland clipping algorithm

1) Clipping window region can be rectangular in shape only and no other polygonal shaped window is allowed.

2) Edges of rectangular shaped clipping window has to be parallel to the x-axis and y-axis.

3) If end points of line segment lies in the extreme limits i.e., one at R.H.S other at L.H.S., and on one the at top and other at the bottom (diagonally) then, even if the line doesn't pass through the clipping region it will have logical intersection of 0000 implying that line segment will be clipped but infect it is not so.

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 4 is "1" line intersects with left boundary of rectangle window i.e., if the line passes through the left region,

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

where $x = x_{wmin}$

where x_{wmin} is the minimum value of x co-ordinate of window

(b) If bit 3 is "1" line intersect with right boundary i.e., if the line passes through the right region,

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

where $x = x_{wmax}$

where x more is maximum value of x co-ordinate of the window

(c) If bit 2 is "1" line intersects with bottom/down boundary i.e., if the line passes through the bottom,

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

where $y = y_{wmin}$

y_{wmin} is the minimum value of Y co-ordinate of the window

(d) If bit 1 is "1" line intersects with the top/up boundary i.e., if the line passes through the top,

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

where $y = y_{wmax}$

y_{wmax} is the maximum value of Y co-ordinate of the window

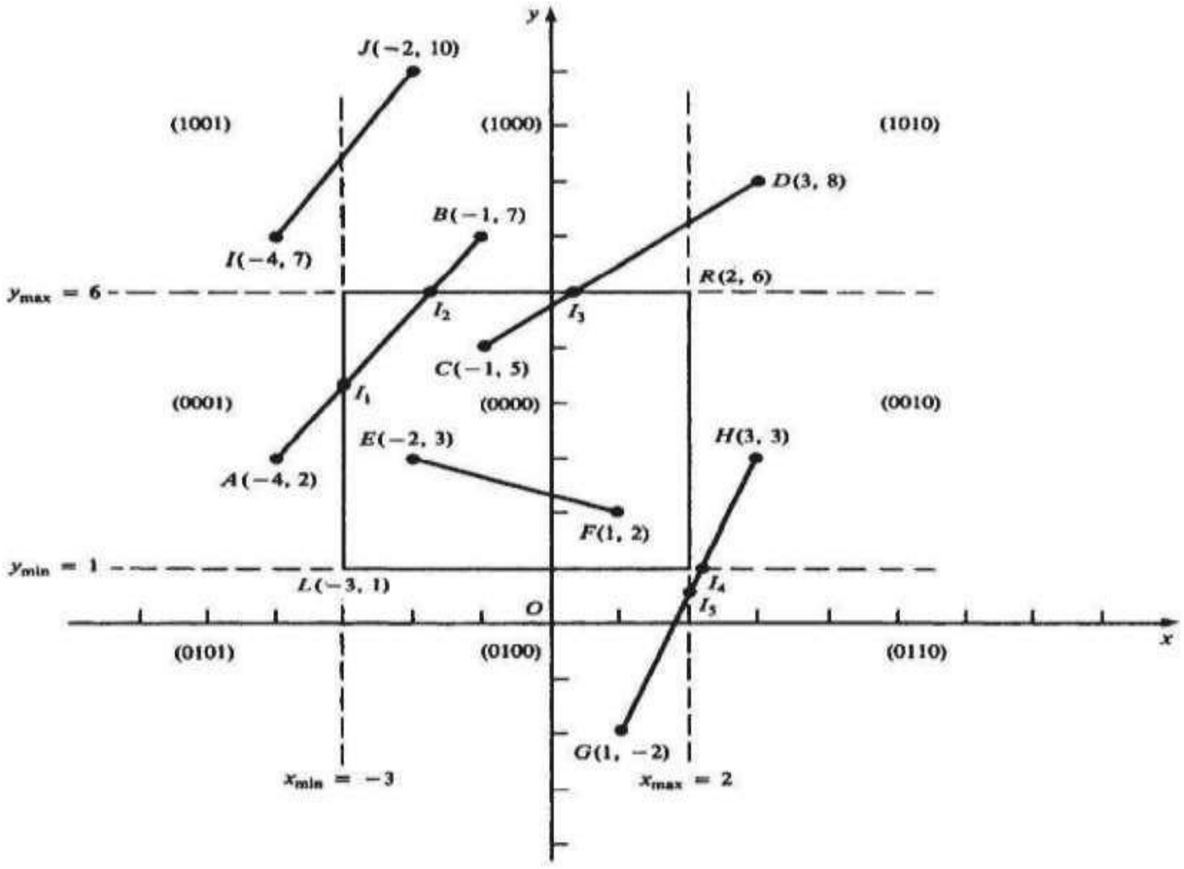
Example -01

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).

(a) Find the region codes for the endpoints in following figure.

(b) Find the clipping categories for the line segments.

(c) Use the Cohen-Sutherland algorithm to clip the line segments.



(a) Solution:

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

$$\text{Bit } 1 = \text{sign}(y - y_{\max}) = \text{sign}(y - 6) \qquad \text{Bit } 3 = \text{sign}(x - x_{\max}) = \text{sign}(x - 2)$$
$$\text{Bit } 2 = \text{sign}(y_{\min} - y) = \text{sign}(1 - y) \qquad \text{Bit } 4 = \text{sign}(x_{\min} - x) = \text{sign}(-3 - x)$$

Here

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

So

$$\begin{array}{ll} A(-4, 2) \rightarrow 0001 & F(1, 2) \rightarrow 0000 \\ B(-1, 7) \rightarrow 1000 & G(1, -2) \rightarrow 0100 \\ C(-1, 5) \rightarrow 0000 & H(3, 3) \rightarrow 0010 \\ D(3, 8) \rightarrow 1010 & I(-4, 7) \rightarrow 1001 \\ E(-2, 3) \rightarrow 0000 & J(-2, 10) \rightarrow 1000 \end{array}$$

(b) Solution:

We place the line segments in their appropriate categories by testing the region codes:

Category 1 (visible): $\overline{EF}$ since the region code for both endpoints is 0000.

Category 2 (not visible): $\overline{IJ}$ since (1001) AND (1000) = 1000 (which is not 0000).

Category 3 (candidates for clipping): $\overline{AB}$ since (0001) AND (1000) = 0000, $\overline{CD}$ since (0000) AND (1010) = 0000, and $\overline{GH}$ since (0100) AND (0010) = 0000.

(c) Solution:

The clipping candidates are $\overline{AB}$, $\overline{CD}$, and $\overline{GH}$.

1. In clipping $\overline{AB}$, 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 $I_1(-3, 3\frac{2}{3})$.

The coordinate of I_1,

Here, $x = x_{\min} = -3$

$m = \frac{7-2}{-1+4} = \frac{5}{3}$ [where A(-4, 2) and B(-1, 7)]

So, $y = y_1 + m(x - x_1) = y_1 + m(x_{\min} - x_1) = 2 + \frac{5}{3}(-3+4) = 2 + \frac{5}{3} = \frac{11}{3} = 3\frac{2}{3}$

Hence $I_1 = (-3, 3\frac{2}{3})$

We clip (do not display) $\overline{AI_1}$ and work on $\overline{I_1B}$. The code for I_1 is 0000. The clipping category for $\overline{I_1B}$ 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 $I_2(-1\frac{3}{5}, 6)$.

The coordinate of I_2,

Here, $y = y_{max} = 6$

$m = \frac{7-2}{-1+4} = \frac{5}{3}$ [where A(-4, 2) and B(-1, 7) and the slope is same]

So, $y = x_1 + (y-y_1)/m = x_1 + (y_{max} - y_1)/m = -4 + (6-2)/\frac{5}{3} = -4 + \frac{12}{5} = \frac{11}{3} = 1\frac{3}{5}$

Hence $I_2 = (-1\frac{3}{5}, 6)$.

Thus, $\overline{I_2B}$ is clipped. The code for I_2 is 0000. The remaining segment $\overline{I_1I_2}$ is displayed since both endpoints lie in the window (i.e., their codes are 0000).

2. For clipping $\overline{CD}$, 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 $(\frac{1}{3}, 6)$ and its code is 0000. Thus $\overline{I_3D}$ is clipped and the remaining segment $\overline{CI_3}$ has both endpoints coded 0000 and so it is displayed
3. For clipping $\overline{GH}$, 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 $\overline{GI_4}$ and work on $\overline{I_4H}$. Segment $\overline{I_4H}$ is not displayed since (0010) AND (0010) = 0010.

Example -02

Clip a line A (-1,5) and B (3,8) using the Cohen Sutherland algorithm with window coordinates (-3,1) and (2,6).

Solution:

Here $x_{min} = -3$ and $y_{min} = 1$ with $x_{max} = 1$ and $y_{max} = 6$ with $x_1 = -1$, $y_1 = 5$ and $x_2 = 3$, $y_2 = 8$.

Step 1: Find the region code of the line point $x_1 = -1$, $y_1 = 5$ as follows

Bit 1 = Sign of ($y-y_{max}$) = Sign of (5-6) = Negative, so bit 1 would be 0.

Bit 2 = Sign of ($y_{min}-y$) = Sign of (1-6) = Negative, so bit 2 would be 0.

Bit 3 = Sign of ($x-x_{max}$) = Sign of (-1-1) = Negative, so bit 3 would be 0.

Bit 4 = Sign of ($x_{min}-x$) = Sign of (-3-(-1)) = Negative, so bit 4 would be 0.

Find the Region Code of the Line Point $x_2=3$, $y_2=8$ as follows.

Bit 1 = Sign of ($y-y_{max}$) = Sign of (8-6) = Positive, so bit 1 would be 1.

Bit 2 = Sign of ($y_{min}-y$) = Sign of (1-8) = Negative, so bit 2 would be 0.

Bit 3 = Sign of ($x-x_{max}$) = Sign of (3-1) = Positive, so bit 3 would be 1.

Bit 4 = Sign of ($x_{min}-x$) = Sign of (-3-3) = Negative, so bit 4 would be 0.

The (x_1, y_1) has the code (0000) and (x_2, y_2) has the code (1010). ***Remember the codes are written from left to right. Left is bit-1.***

Step 2: Which category this line belongs to: Find, does it require clipping?

– Logical AND of both the region codes are (0000) ^ (1010) = (0^1, 0^0 ,0^1, 0^0) = (0000). Since the logical AND of codes is zero, so the line belongs to third category- Clipping category =>Computer would clip it.

Step 3: The line "Needs Clipping" based on the calculation above.

Step 4: Since Bit 1 is one therefore intersection point is

$y=y_{max}=6$

and $x=x_1+(y_1-y_{max})/m$.

x= -1+(6-5)/(3/4) = -1+4/3 = 1/3.

So, the intersection point would be (x=1/3, y=6).

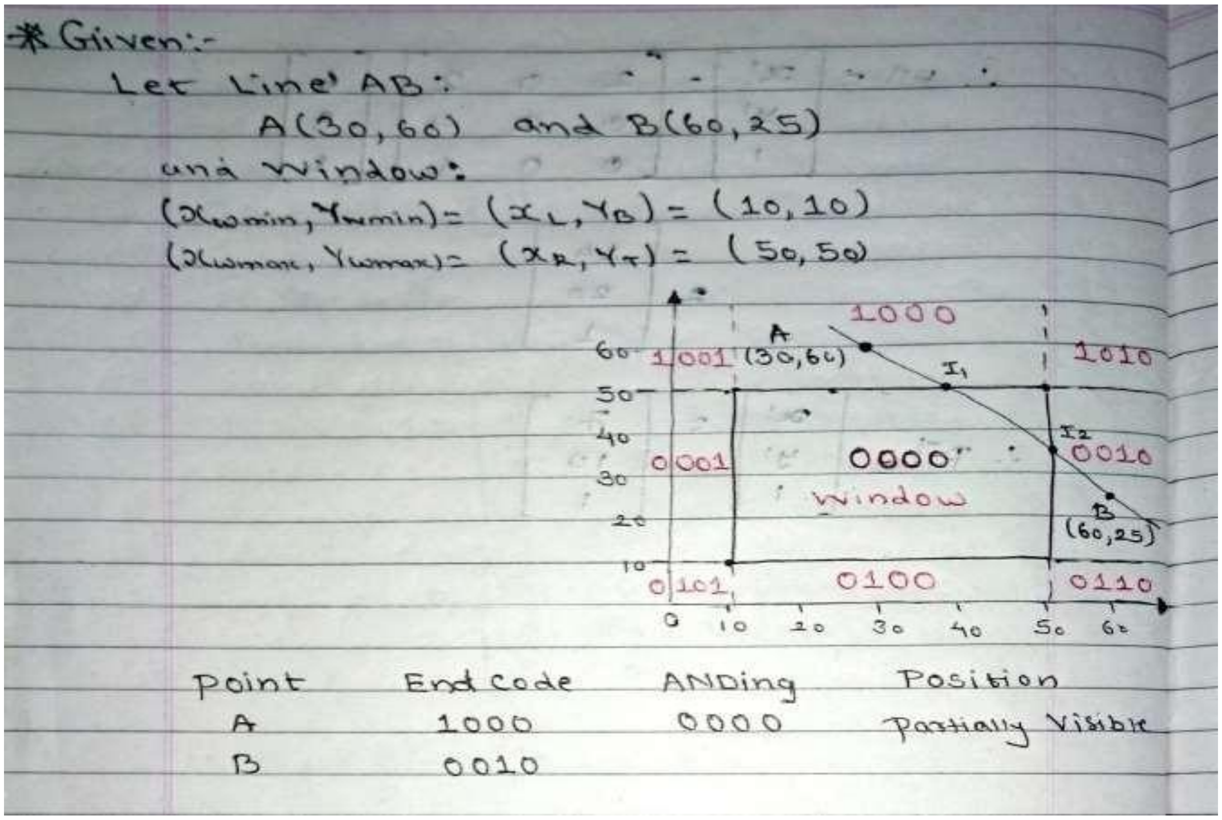
Step 5: Find the region code of newfound point, C (x=1/3 and y=6) = (0000)

Since the starting point, A (-1,5) has (0000) and new Point C is also (0000) that means both endpoints are visible. The line does not need more clipping.

The algorithm would stop here.

Example -03

Apply the Cohen Sutherland line clipping algorithm to clip the line segment with coordinates (30,60) and (60,25) against the window with (Xwmin,Ywmin)= (10,10) and (Xwmax,Ywmax)= (50,50).



* Given:-

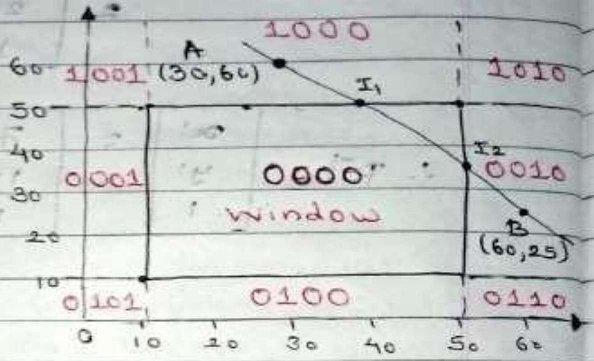
Let Line AB:

A(30,60) and B(60,25)

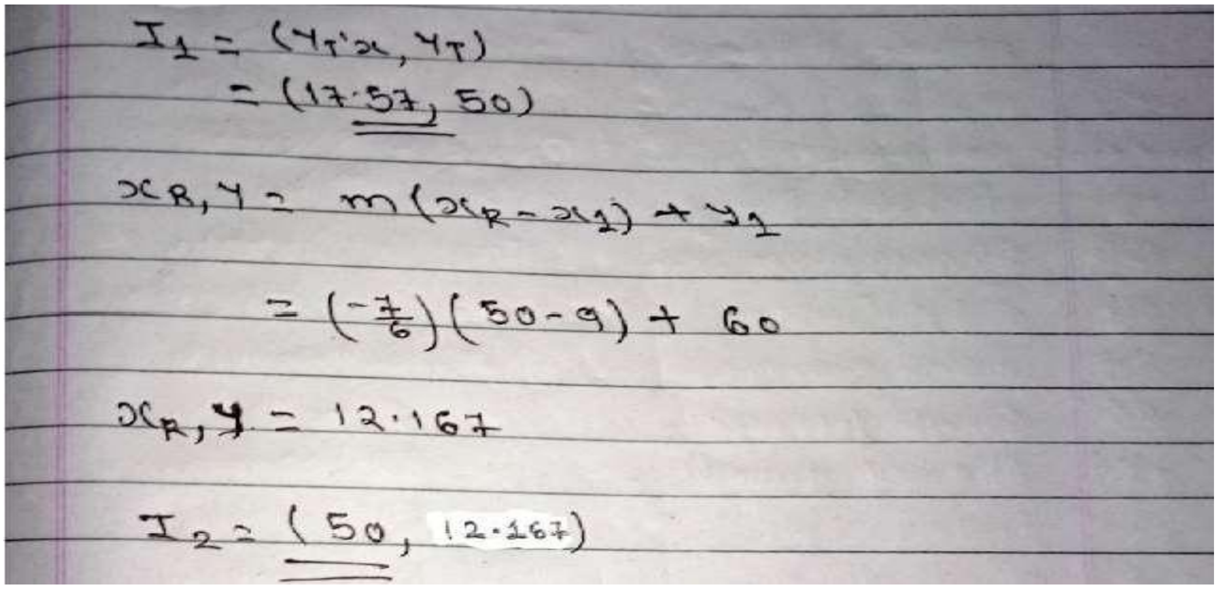
and Window:

$(x_{wmin}, y_{wmin}) = (x_L, y_B) = (10,10)$

$(x_{wmax}, y_{wmax}) = (x_R, y_T) = (50,50)$



Point	End code	ANDing	Position
A	1000	0000	Partially Visible
B	0010		



$I_1 = (y_T{\cdot}x, y_T)$

$= (\underline{17.57}, 50)$

$x_R, y = m(x_R - x_1) + y_1$

$= \left(-\frac{7}{6}\right)(50-9) + 60$

$x_R, y = 12.167$

$I_2 = (\underline{50}, 12.167)$

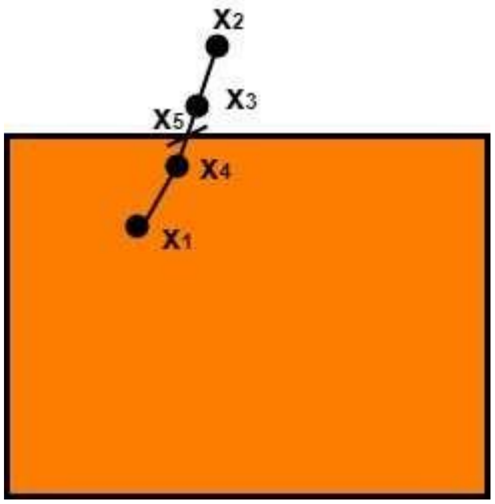
Midpoint Subdivision Algorithm

This algorithm is mainly used to compute visible areas of lines that are present in the view port are of the sector or the image. It follows the principle of the bisection method and works by bisecting the line in to equal halves but bisects the line numerous times.

Like other algorithm, initially the line is tested for visibility. If line is completely visible it is drawn and if it is completely invisible it is rejected. If line is partially visible then it is subdivided in two equal parts. The visibility tests are then applied to each half. This subdivision process is repeated until we get completely visible and completely invisible line segments.

Let (x_i,y_i) are midpoint

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



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 $x_5 = \frac{x_4+x_3}{2}$

X_5 lie on point of intersection of boundary of window.

<u>Difference between cohen sutherland and midpoint subdivision algorithm for line clipping:</u>

<u>The Cohen - Sutherland algorithm:</u>

- The **Cohen - Sutherland algorithm divides 2D space into 9 bits**, using an infinite extension of the four linear limits of the frame.
- This seems to 'calculate intersections' along a 'line path', and each calculation involves both a 'division and a multiplication'.

<u>The Midpoint subdivision algorithm:</u>

- The **'Midpoint subdivision algorithm'** is mainly used to calculate the visible area of the line that is 'present in the view port' of the 'sector or the image'.
- It takes the principle of the bisection technique by bisecting the line into equivalent division. **'Midpoint Subdivision Algorithm' bisects the line many times.**

<u>Algorithm of midpoint subdivision Line Clipping:</u>

```
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.