Computer Graphics

Clipping

Many graphics application programs give the user the impression of locking through a window at a very large picture. The program makes use of the scaling and translation techniques to generate a variety of different views of a single representation of a plan.

To display an enlarged portion of a picture, we must not only apply the appropriate scaling and translation but also identify the visible parts of the picture for inclusion in the displayed image.

Certain lines may lie partly inside the visible portion of the picture and partly outside.

The correct way to select visible information for display is to use clipping, a process which divides each element of the picture into its visible and invisible portions, allowing the invisible portion to be discarded.

1 Point Clipping

Is to determine if a point (X,Y) is visible or not by a simple pair of inequalities:

$$X_{left} \leq X \leq X_{right}$$

$$Y_{bottom} \leq Y \leq Y_{top}$$

Where X left , X right , Y bottom , Y top are the position of the edge of the screen.

These inequalities provides us with a very simple method of clipping pictures on a point-by-point basis.

There are 3 possibilities for the line:

1. Line can be completely inside the window (This line should be accepted).

2. Line can be completely outside of the window (This line will be completely removed from the region).

3. Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).

Algorithm

Step 1: Assign a region code for each endpoints.

Step 2: If both endpoints have a region code **0000** then accept this line.

Step 3: Else, perform the logical **AND** operation for both region codes.

Step 3.1: If the result is not **0000**, then reject the line.

Step 3.2: Else you need clipping.

Step 3.2.1: Choose an endpoint of the line that is outside the window.

Step 3.2.2: Find the intersection point at the window boundary (base on region code).

Step 3.2.3: Replace endpoint with the intersection point and update the region code.

Step 3.2.4: Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

Step-4: Repeat step 1 for other lines.

tutorialspoint
SIMPLYEASYLEARNING

2 Line clipping

It would be quit inappropriate to clip pictures by converting all picture elements into points and using point clipping, the clipping process would take far too long. We must instead attempt to clip large elements of the picture.

Figure (1) shows a number of different lines with respect to the screen :

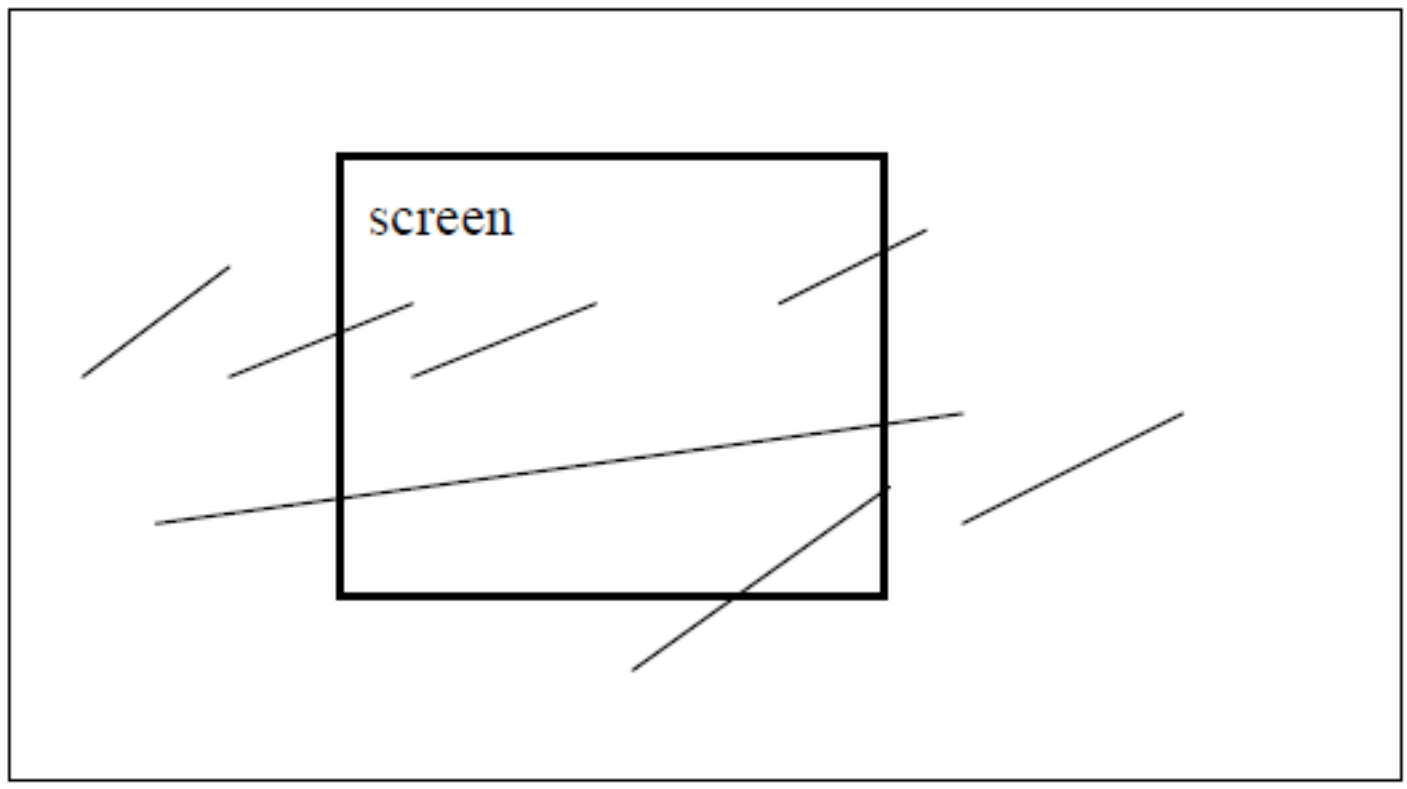


Figure – 1 –

Notice that those lines which are partly invisible are divided by the screen boundary into one or more invisible portions but into only one visible segment.

This means that the visible segment of a straight line can be determined simply by computing its two end points.

We divide the line clipping process into two phases :

1- Identify those lines which intersect the window and so need to be clipped
2- Perform the clipping

All line segments fall into one of the following clipping categories :

1- Visible : both end points of the line segment lie within the window
2- Not visible : the line segment definitely lies out side the window.

   This will occur if the line segment from (X1, Y1) to (X2 , Y2) satisfies any one of the following four inequalities :

   X1 , X2 > X max Y1 , Y2 > Y max

   X1 , X2 < X min Y1 , Y2 < Y min

3- Clipping candidate :the line is in neither category 1 nor category 2.

Consider figure - 2 –

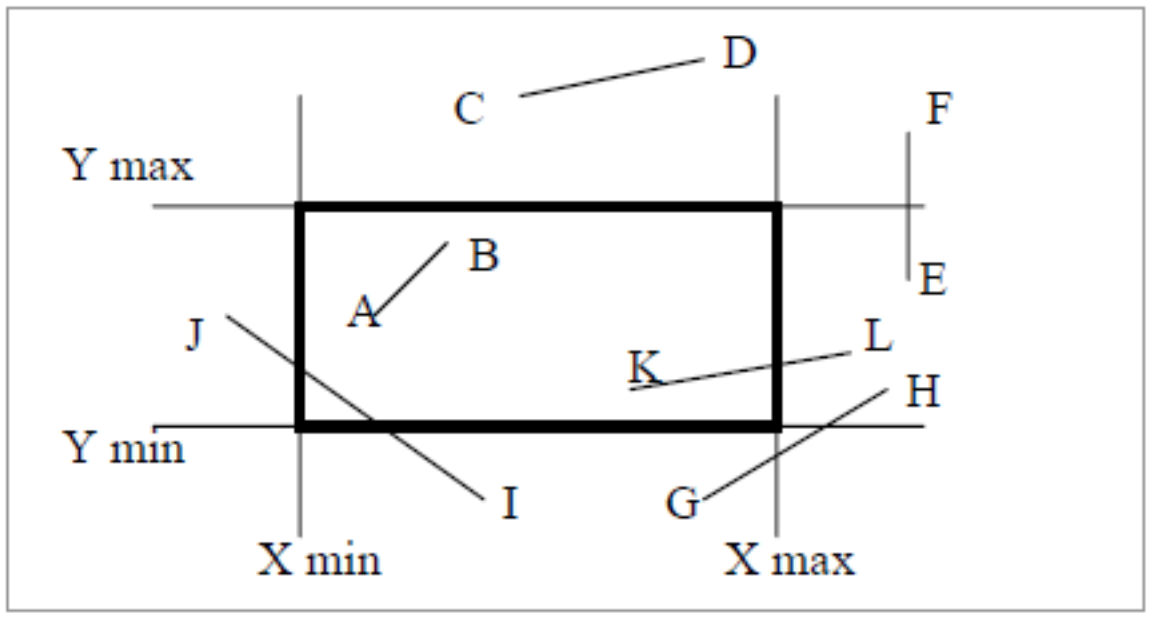


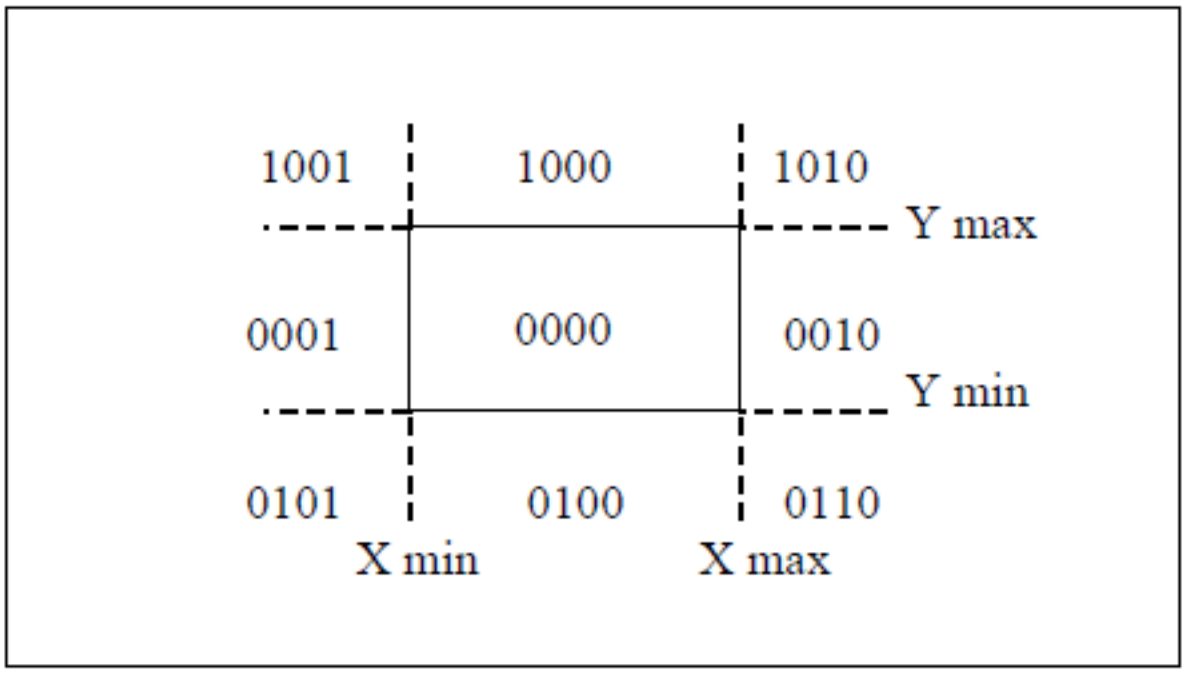
Figure - 2 -

Line segment AB is in category 1 (visible).
Line segments CD and EF are in category 2 (invisible)
Line segments GH , IJ , KL are in category 3 (clipping candidate)

Cohen – Sutherland algorithm

This algorithm is to find the category of the line segment . The algorithm proceeds in two steps :

1- Assign a 4-bit code to each endpoint of the line segments . The code is determined according to which of the following nine regions the endpoints lies in :



Starting from the leftmost bit, each bit of the code is set to true(1) of false(0) according to the schema:

Bit 1 ≡ endpoint is above the window = sign (Y – Ymax)
Bit 2 ≡ endpoint is below the window = sign (Ymin -Y)
Bit 3 ≡ endpoint is to the right of the window = sign (X - Xmax)
Bit 4 ≡ endpoint is to the left of the window = sign (Xmin - X)

Note : sign (a)=1 if a is positive , =0 otherwise

2- The line segment is visible if both endpoint codes are 0000

The line segment is not visible if the logical AND of the codes is not 0000

The line segment is candidate for clipping if the logical AND of the endpoint codes is 0000

We now decide whether the line candidate for clipping either intersect the window and so is to be clipped or don't intersect the window and so is not displayed.

The point of intersection of the line segment with an extended window edge, each of the four directions is tested in the order : left , right , top , bottom. The part of the line that is clearly outside is discarded.

To calculate the point of intersection of the line segment with the window border, the point-slope formula is used. If M is the slope of a line segment between (X1,Y1) and (X2,Y2) , then if $X1 \neq X2$:-

$$M= (Y2-Y1) / (X2-X1)$$

for any other point (X,Y) on the line :-

$$M=(Y-Y1) / (X-X1)$$

If we are testing against a left or right direction the X value is known (left or right) edge value. This X value is substituted into the equation :

$$Y=M * (X-X1) + Y1$$

If we are testing against a top or bottom, the Y value is known and substituted into :

$$X= (1/M) * (Y-Y1) + X1$$

The Cohen Algorithm

1-Determine the endpoint (X1 , Y1) code1

If X1<Xmin Then
If Y1>Ymax Then code1=1001
If (Y1<Ymax) And (Y1>Ymin) Then code1=0001
If Y1<Ymin Then code1=0101
End IF

If X1>Xmax Then
If Y1>Ymax Then code1=1010
If (Y1<Ymax) And (Y1>Ymin) Then code1=0010
If Y1<Ymin Then code1=0110
End IF

If (X1 ≤ Xmax) And (X1 ≥ Xmin) Then
If Y1>Ymax Then code1=1000
If (Y1≤Ymax) And (Y1≥Ymin) Then code1=0000
If Y1<Ymin Then code1=0100
End IF

2- Determine the endpoint (X2,Y2) code2 [as code1]

3-Determine the visibility of the line

If code1= code2 = 0000 then the line is visible; draw the line

If code1 And code2 <> 0000 then the line is not visible

If code1 And code2 = 0000 then the line is candidate for clipping

4- Calculate the intersection points of the candidate lines with the window :

$$M = \Delta Y / \Delta X$$

Left : (XL,YS) : $YS = M(XL - X1) + Y1$; $M \neq \infty$
Right : (XR,YS) : $YS = M(XR - X1) + Y1$; $M \neq \infty$
Top : (XS,YT) : $XS = 1/M(YT - Y1) + X1$; $M \neq 0$
Bottom : (XS,YB) : $XS = 1/M(YB - Y1) + X1$; $M \neq 0$

If the candidates is out side the boundary of the window then it is Rejected.

Notes:

1- If M=∞ we don't calculate the intersection points with the left edge and right edge because the line is parallel to them, we only calculate the intersection points with the top edge and bottom edge.

2- If M=0 we don't calculate the intersection points with the top edge and bottom edge because the line is parallel to them, we only calculate the intersection points with the left and right edge.

Example 1 : If the clipping window is XL=-4 ; XR=4 ; YT=4 ; YB=-4

Clip the lines KL=(4,5)-(6,5) and AB=(1,1)-(1,3)

Solution 1 :

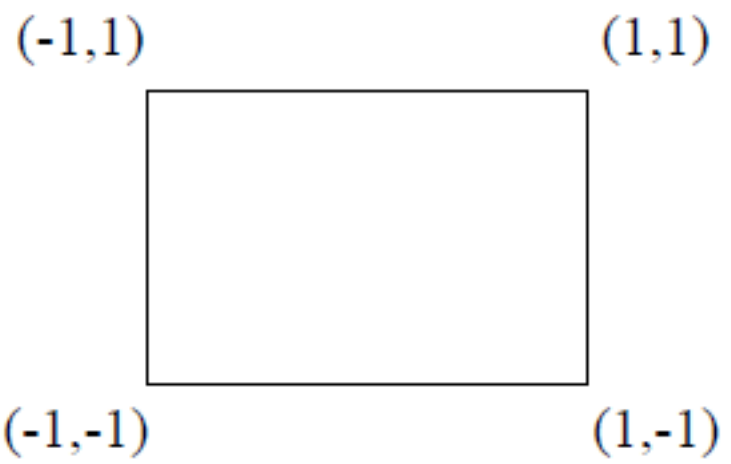
For the line KL : code1 of (4,5) is 1000
code2 of (6,5) is 1010

code1 And code2 : 1000 And 1010 = 1000 (not zero)
then the line KL is not visible

For the line AB : code1 of (1,1) is 0000
code2 of (1,3) is 0000

code1 And code2 : 0000 And 0000 = 0000 (zero)
then the line AB is visible

Example 2: Consider the clipping window



clip the line (- 3/2 , 1/6) – (1/2 , 3/2)

Solution 2 :

Code1 of (- 3/2 , 1/6) is 0001
Code2 of (1/2 , 3/2) is 1000

Code1 And Code2 : 0001 And 1000=0000
Then the line is candidate for clipping

We must calculate the intersection points:

M= 1.34 / 2 = 0.6

Left intersection : XL=-1
YL=0.6 (-1-(-3/2))+1/6=1/2
Then (-1, 0.5) is the first point of intersection

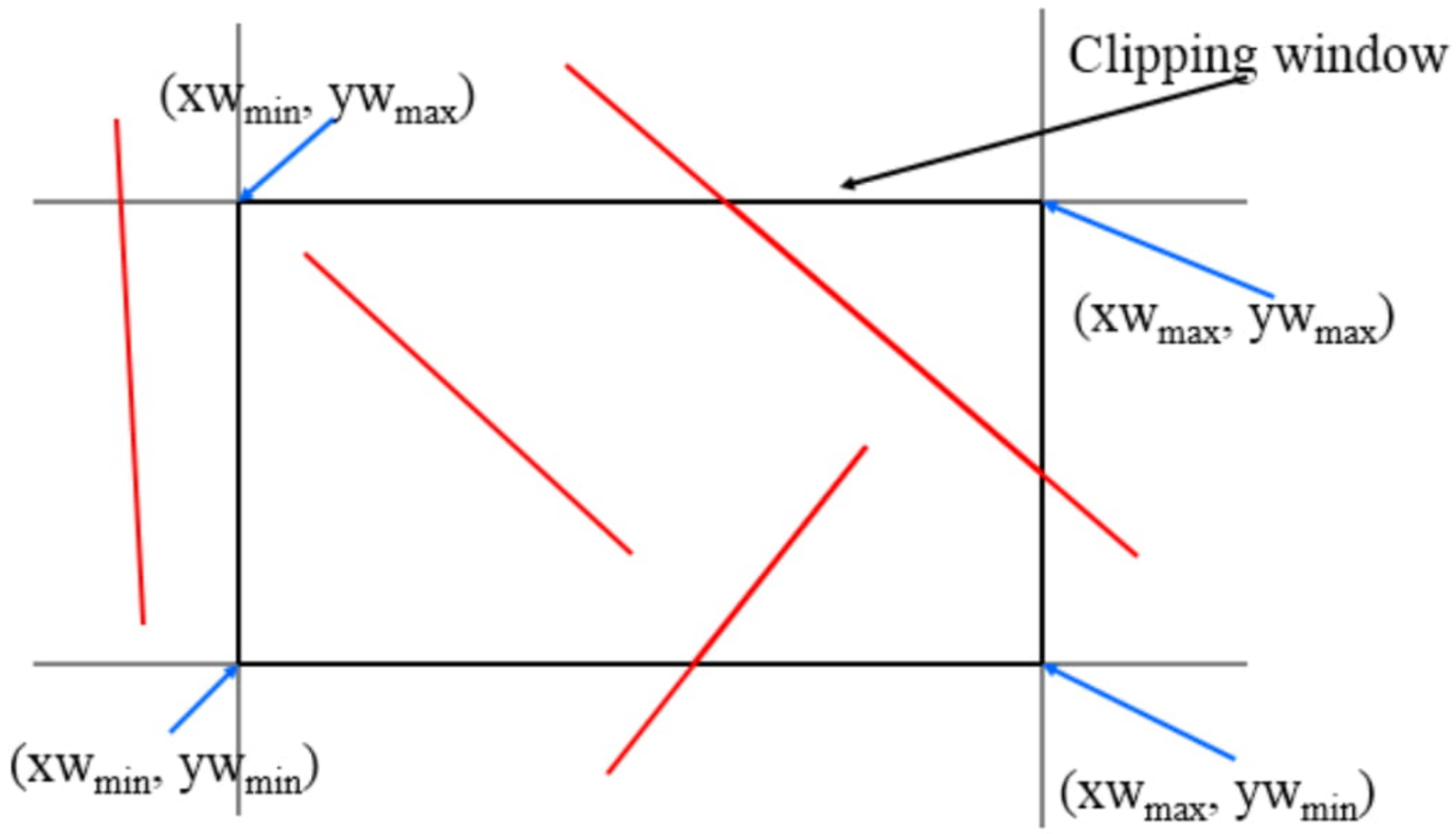
Right intersection : XR=1
YR=0.6 (1-(-3/2))+1/6=1.8
Then (1, 1.8) is rejected

Top intersection : YT=1
XT =1.6 (1- 1/6)+(- 3/2) = - 1/4
Then (-1/4,1) is the second point of intersection

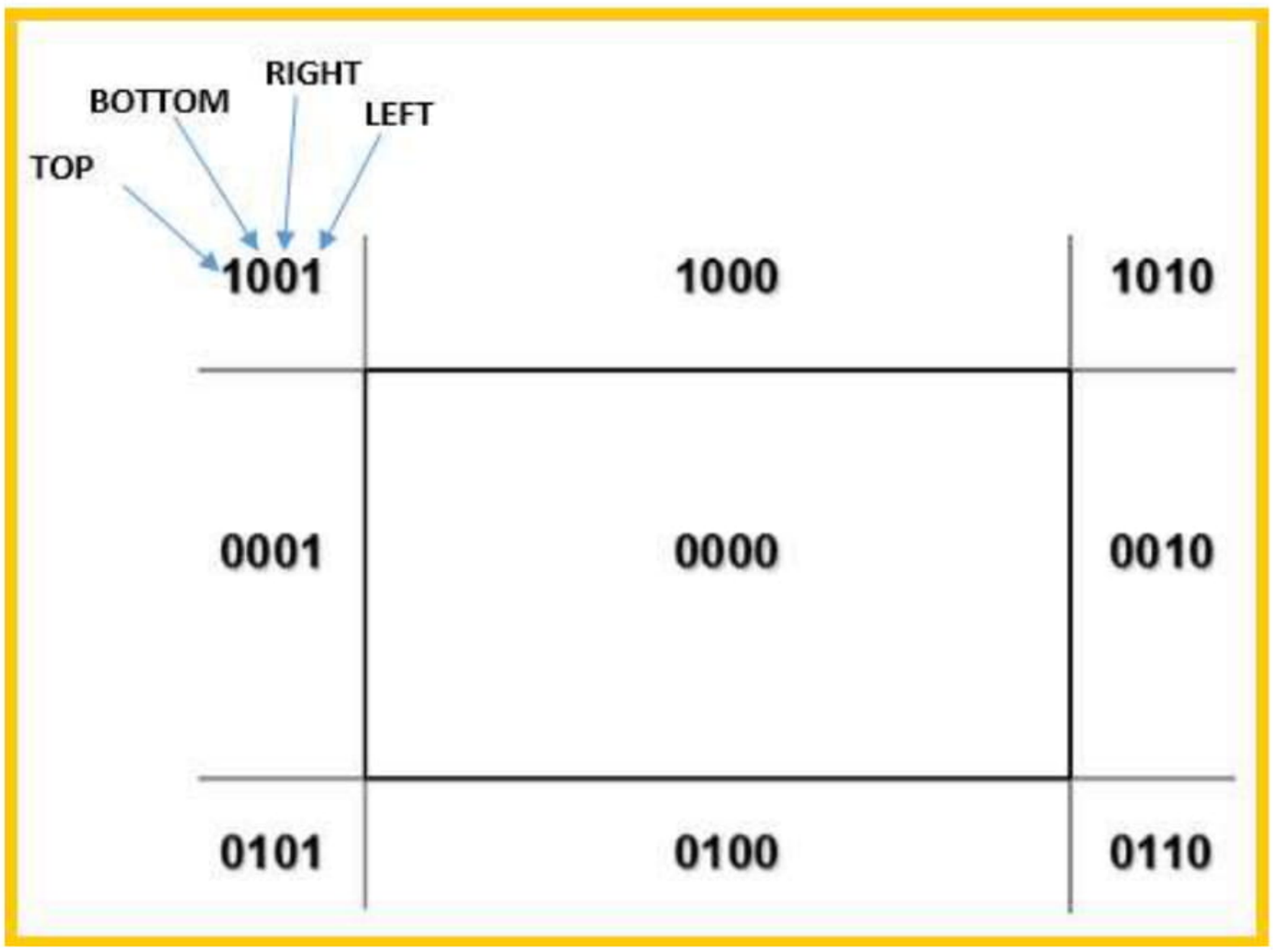
Bottom intersection : YB=-1
XB =1.6 (-1 - 1/6)+(- 3/2) = - 3.25
Then (-3.25 ,-1) is rejected

Cohen-Sutherland Line Clippings

This algorithm uses the clipping window as shown in the following figure. The minimum coordinate for the clipping region is (XW_{min}, YW_{min}) and the maximum coordinate for the clipping region is (XW_{max}, YW_{max}).



We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.



tutorialspoint
SIMPLYEASYLEARNING