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 :

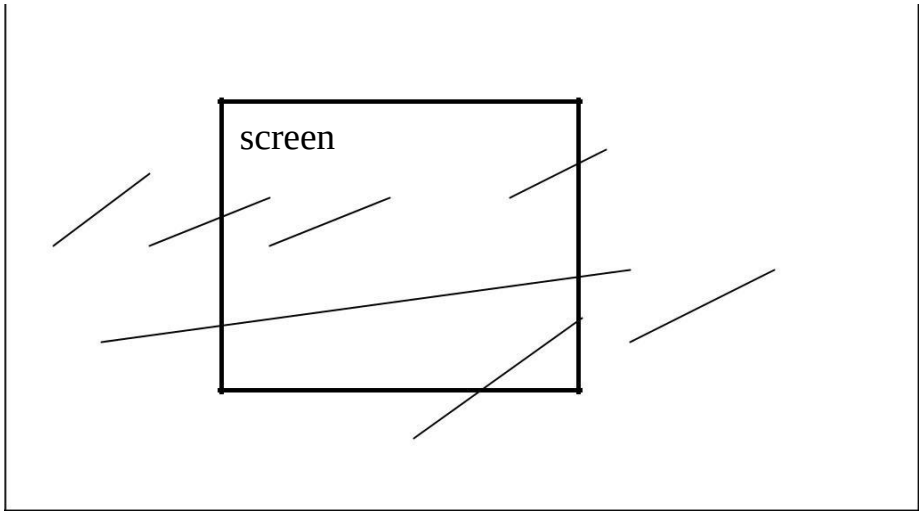


Figure – 1 –

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 –

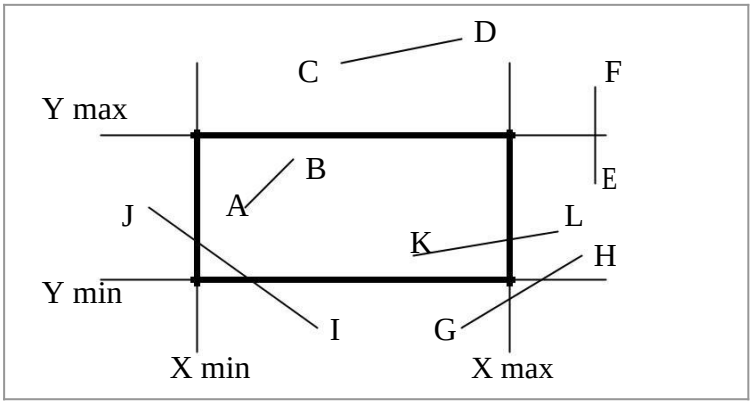


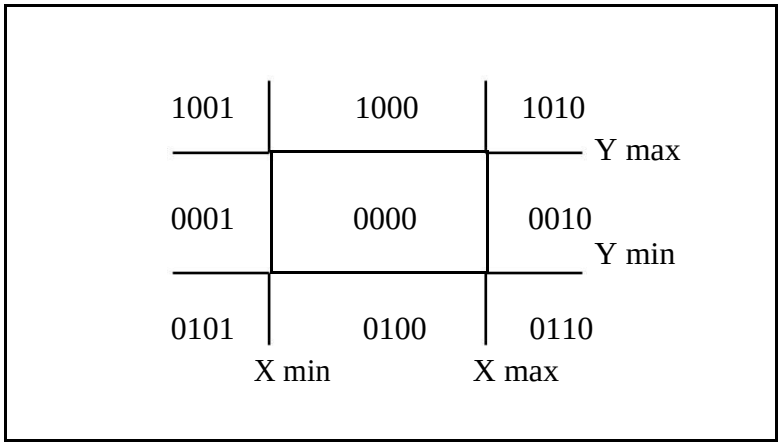
Figure - 2 -

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 :

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