2-D Clipping

2-D Clipping

- Clipping: The procedure that identifies the portions of a picture that are either inside or outside of a specified region of space is referred to as clipping.
- clipping window : The region against which an object is to be clipped is called a clip window or clipping window.
- It is rectangular in shape.

- The clipping algorithm determines which points, lines or portions of lines lie within the clipping window.
- These points, lines or portions of lines are retained for display.
- All others are discarded.
- Clipping algorithms can be applied in world coordinates, so that only the content of the window interior are mapped to device coordinates

Type of Clipping

- Clipping algorithm have the following primitive types
  - Point Clipping
  - Line Clipping
  - Polygon Clipping

Point Clipping :

- The points are said to be interior to the clipping window if

$$x_{w\ min} \leq x \leq x_{w\ max} \quad \text{and} \quad y_{w\ min} \leq y \leq y_{w\ max}$$

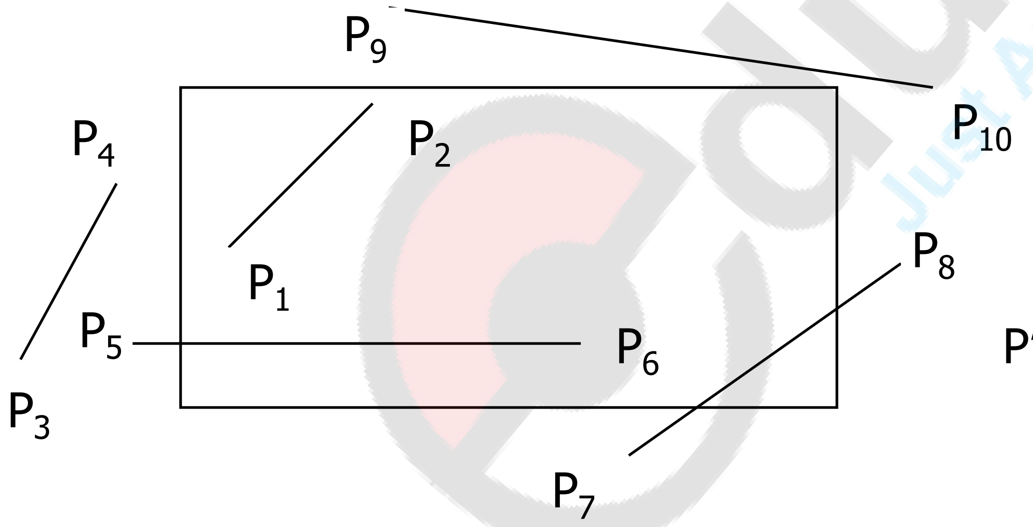
The equal sign indicates that points on the window boundary are included with in the window.

Line Clipping

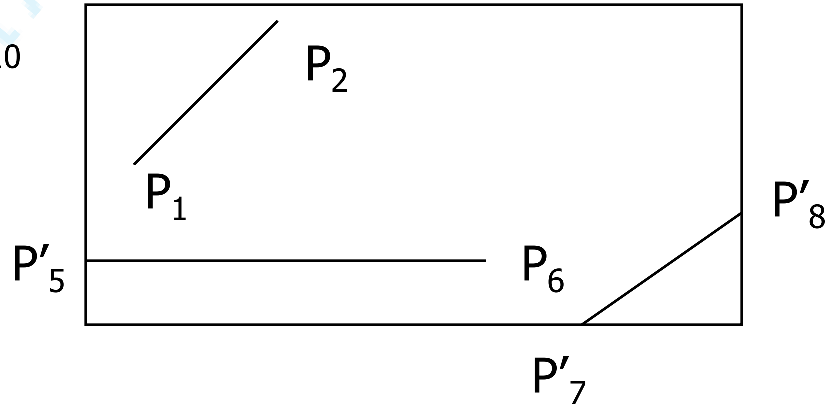
- First of all ,we have to test the given line segment.
- Lines that do not intersect the clipping window are either completely inside the window or completely outside the window.
- A lines is said to be interior to the clipping window and hence visible if both end points are interior to the window,
- If both end points of a line are completely to the right or, completely to the left or, completely above, or completely below the window, then the line is completely exterior to the window and hence invisible.

- If a line is not identified as a line completely inside or completely outside the window, we have to perform intersection calculations with one or more clipping boundary.

- To minimize the intersection calculations and to increase the efficiency of the clipping algorithm, initially, completely visible and invisible lines are identified and then intersection points are calculated for remaining lines.



Before clipping



After clipping

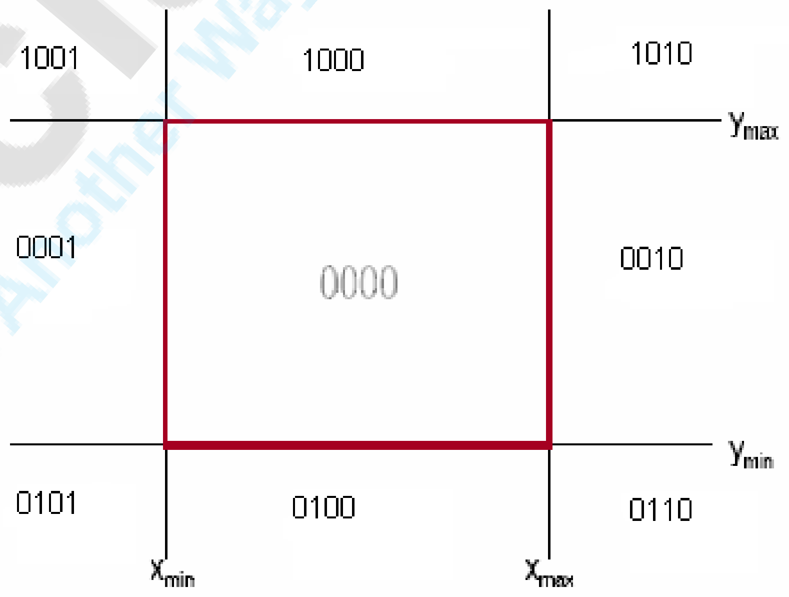
Cohen-Sutherland Line Clipping

- It is Developed by Dan Cohen and Ivan Sutherland.
- To speed up the processing this algorithm performs initial tests that reduce the number of intersections that must be calculated.
- In this, the clipping process is divided into two phases:-
  - Identify those lines which intersect the clipping window & need to be clipped.
  - Perform the clipping.
- All lines fall into one of the following clipping categories:
  - Visible- Both endpoints of the line lie within the window.
  - Not visible- The line definitely lies outside the window.
  - Clipping candidate- The line is in neither category 1 nor 2.

Cohen-Sutherland Line Clipping

- Assign a 4-bit code to each endpoint of the line. The four bit codes are called **region codes** or **out codes**. These codes identify the location of a point relative to the boundaries of the clipping window.

The rightmost bit is the first bit & the bits are set to 1 based on the following scheme:-



Cohen-Sutherland Line Clipping

Set Bit 1 – if the end point is to the left of the window (0001)

Set Bit 2 – if the end point is to the right of the window (0010)

Set Bit 3 – if the end point is to the below of the window (0100)

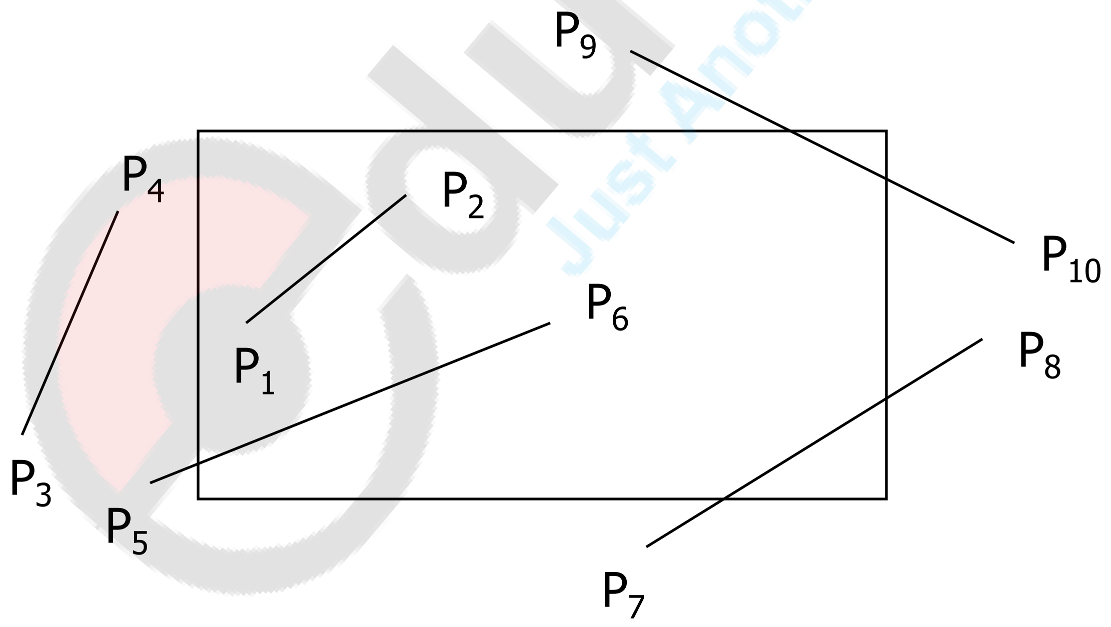
Set Bit 4 – if the end point is to the above of the window (1000)

Otherwise, the bit is set to zero.

• Line is visible:	If both region codes are 0000,
• Line not visible:	If the bitwise logical AND of the codes is not 0000
• Line for clipping :	If the bitwise logical AND of the region codes is 0000.

Example

- Consider the clipping window and the lines shown in the figure . Find the region codes for each point and identify whether the line is completely visible , partially visible or completely invisible.



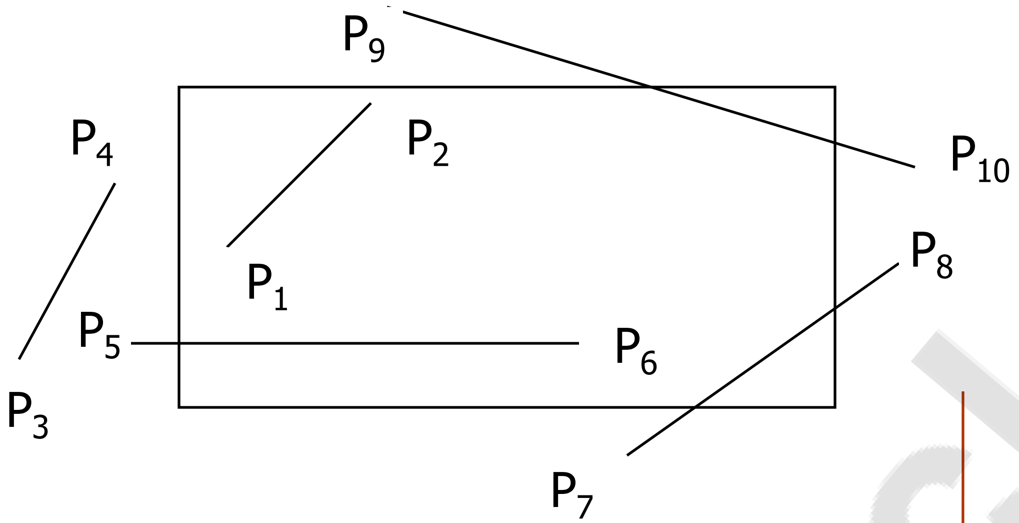


Fig .
Clipping window & lines

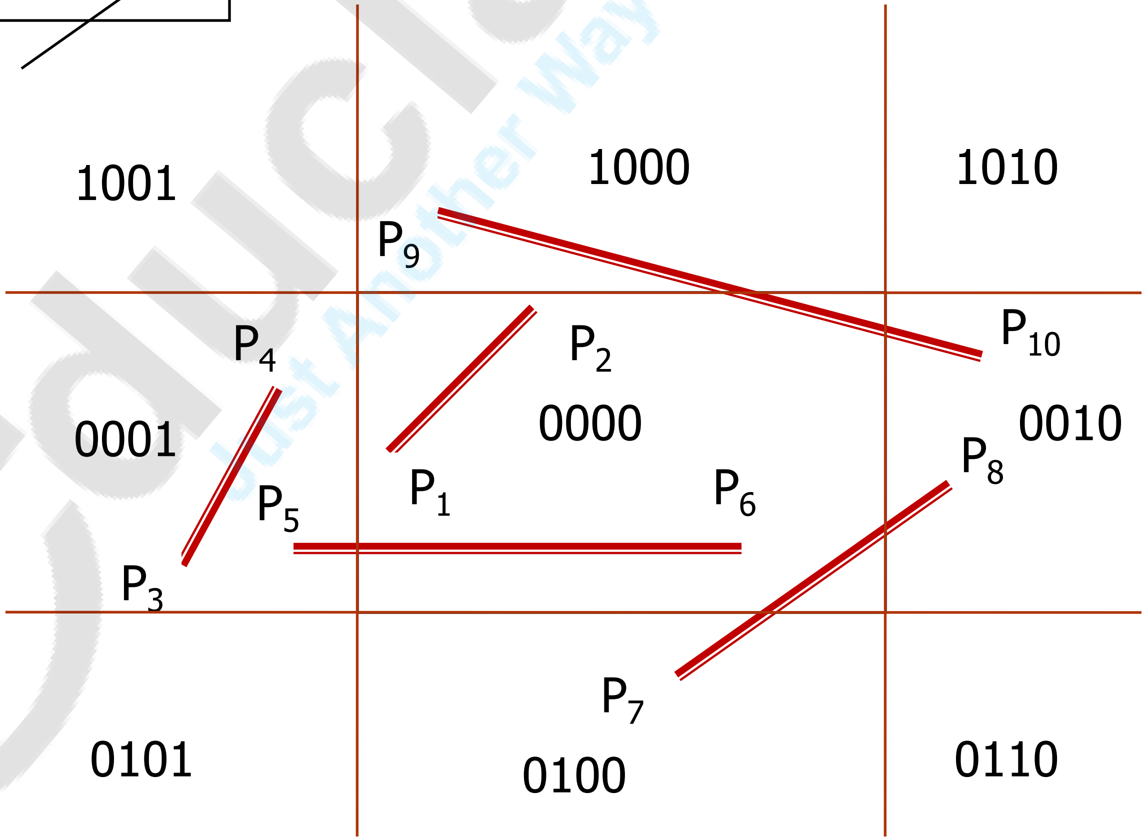


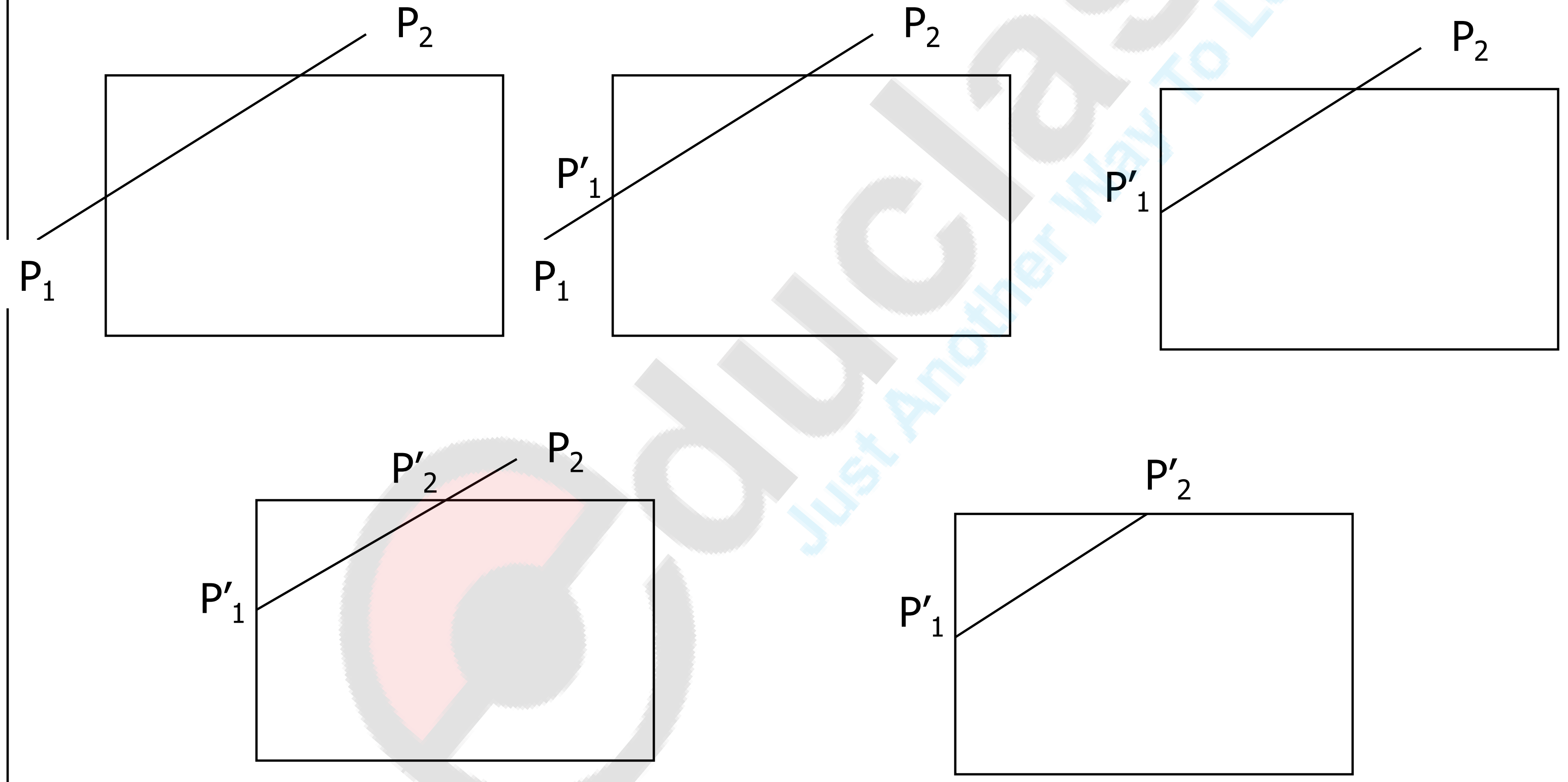
Fig .
Clipping window & lines with region code

line	End Point code		Logical ANDing	Result
P1 P2	0000	0000	0000	Completely visible
P3 P4	0001	0001	0001	Completely Invisible
P5 P6	0001	0000	0000	Partially visible
P7 P8	0100	0010	0000	Partially visible
P9 P10	1000	0010	0000	Partially visible

Cohen-Sutherland Line Clipping

- For a line in category 3 we proceed to find the intersection point of the line with one of the boundaries of the clipping window
- The Sutherland – Cohen algorithm begins the clipping process for a partially visible line by comparing an outside endpoint to a clipping boundary to determine how much of the line can be discarded.
- Then the remaining part of the line is checked against the other boundaries, and the process is continued until either the line is totally discarded or a section is found inside the window.

Cohen-Sutherland Line Clipping



Cohen-Sutherland Line Clipping

- The intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation.
- The equation of the line passing through points P_1 (x_1, y_1) and P_2 (x_2, y_2) is

$$y = m(x - x_1) + y_1 \quad \text{or}$$

$$y = m(x - x_2) + y_2$$

where

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

- Therefore, the intersections with clipping boundaries of the window are given as:

Left:	$x_L, y = m(x_L - x_1) + y_1;$	$m \neq \infty$
Right:	$x_R, y = m(x_R - x_1) + y_1;$	$m \neq \infty$
Top:	$y_T, x = x_1 + (1/m)(y_T - y_1);$	$m \neq 0$
Bottom:	$y_B, x = x1 + (1/m)(y_B - y_1);$	$m \neq 0$

Cohen-Sutherland Line Clipping Algorithm

1. Read two end points of the line say $P_1(x_1 , y_1)$ $P_2(x_2 , y_2)$
2. Read two corner(left top , right , bottom) of the window (say Wx_1 , Wy_1 & Wx_2 , Wy_2)
3. Assign region code for two end points P_1 & P_2 using following steps.

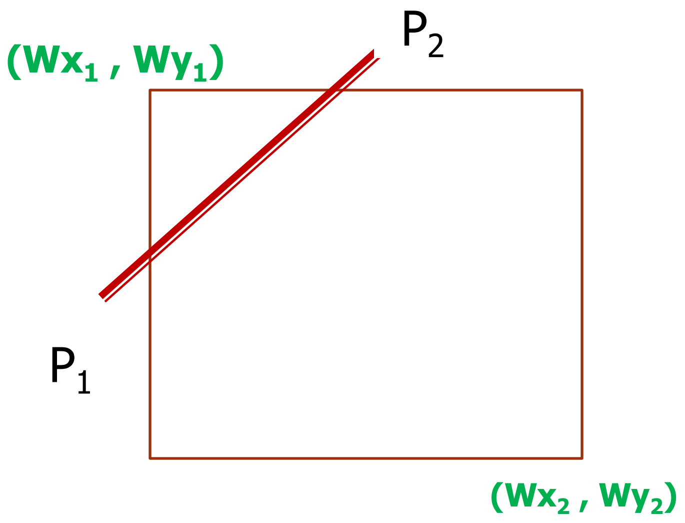
   Initialize code with bit 0000

   set bit 1 – if $(x < Wx_1)$

   set bit 2 – if $(x > Wx_2)$

   set bit 3 – if $(y < Wy_2)$

   set bit 4 – if $(y > Wy_1)$



4. Check for visibility of line P_1 P_2

a) If region code for both end points P_1 & P_2 are zero then the line is completely visible . Hence draw the line & go to step 9

b) If region code for both end points are not zero & logical ANDing of them is also nonzero then the line is completely invisible so reject the line & go to step 9

c) It region code for two end points do not satisfy the condition in 4 a) and 4 b) the line is partially visible

5. Determine the intersecting edge of clipping window by inspecting region code of two endpoints
    a) If the region code for both end points are nonzero find intersection point P1, & P2, with boundary edges of clipping window with respect to point P1 or P2 respectively
    b) If the region code for any one end points is nonzero then find intersection point P1, & P2, with boundary edges of clipping window with respect to it
6. Divide the line segment considering intersection point
7. Reject the lines segment if any one end point of it appears outside the clipping window
8. Draw the remaining line segment
9. stop

EXAMPLES

Q Use Sutherland algo to clip to lines P1(40,15)-P2(75,45) , p3(70,20)-p4(100,10)against window A(50,10)B(80,10),C(80,40),D(50,40) 10m

Q. Identify below given lines which are visible , rejected , clip when applied cohen-sutherland algo for line clipping find new coordinates of clipped lines if any.

Lower left of window is (10 ,10)& upper right is (20 ,20)

- Line AB (12 , 18) (16 ,12)
- Line CD (13 ,15) (25 ,15)
- Line EF (12 ,4) (25 , 8)
- Line GH (8,13)(13,24)
- Line IJ(5,25),(25,25) 10m

Q. Discuss cohen-sutherland line clipping algo & its implementation issues

10m

EXAMPLES

- Explain sutherland-cohen line clipping algo ? Given clipping window A(20,20),B(60,20),C(60,40),D(20,40)Using sutherland-cohen algo find visible portion of line segment joining the points

  P1(40,80),P2 (120,30) 10m

- Determine region code & clip the following line using cohen-sutherland algo windows boundary are
  - windows left corner =(100,100),
  - upper right corner=(200,200),
  - end points of the line are
    1) A= (250,150),B=(260 ,180)
    2) C=(150,50) D=(150,300)
    3) E=(150,90) F=(300 ,110)
    4) G=(70,170),H=(130,180) 10m

References:

Chap 6 : computer graphics by Donald

Chap 5: Windowing and clipping from computer graphics by A.P.Godse

Chap 5 : computer graphics by zhigang xiang - schaum’s

Mid point subdivision algorithm:

- It repetitively subdividing line at its mid point.
- Initially line is tested for visibility. if line is completely visible it is drawn & if it is completely invisible then it is rejected.
- If the line is partially visible then it is subdivided in two equal parts . Then visibility test is applied to each half
- This subdivision process is repeated until we get completely visible and completely invisible line segment

Mid point subdivision algorithm:

1. Read two end points of the line say $P_1(x_1, y_1)$ $P_2(x_2, y_2)$
2. Read two corner(left top , right bottom) of the window

   (say Wx_1, Wy_1 & Wx_2, Wy_2)
3. Assign region code for two end points P_1 & P_2 using following steps.

   Initialize code with bit 0000

   set bit 1 – if ($x < Wx_1$)

   set bit 2 – if ($x > Wx_2$)

   set bit 3 – if ($y < Wy_2$)

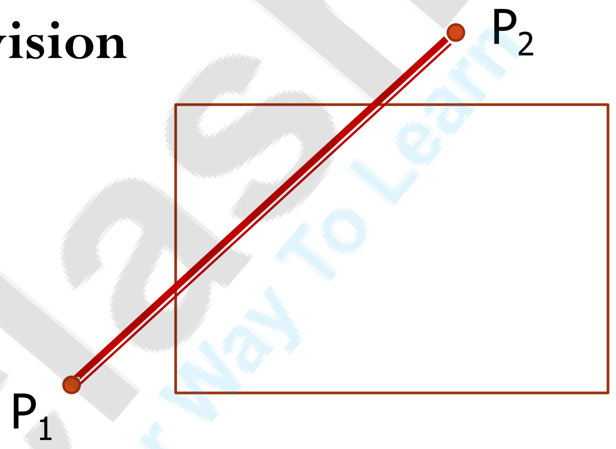
   set bit 4 – if ($y > Wy_1$)

4. Check for visibility of line P1 P2
   - If region code for both end points P1 & P2 are zero then the line is completely visible . Hence draw the line & go to step 6
   - If region code for both end points are not zero & logical ANDing of them is also nonzero then the line is completely invisible so reject the line & go to step 6
   - It region code for two end points do not satisfy the condition in 4 a) and 4 b) the line is partially visible
5. Divide the partially visible line segment in equal parts & repeat step 3 through 5 for both subdivided line segment until you get completely visible and completely invisible line segment.
6. Stop

- Example :

  Line clipping with midpoint subdivision

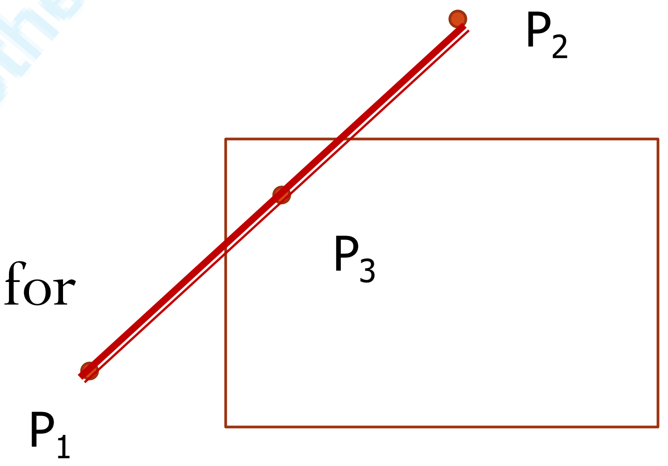
- Line $P_1 P_2$ is partially visible



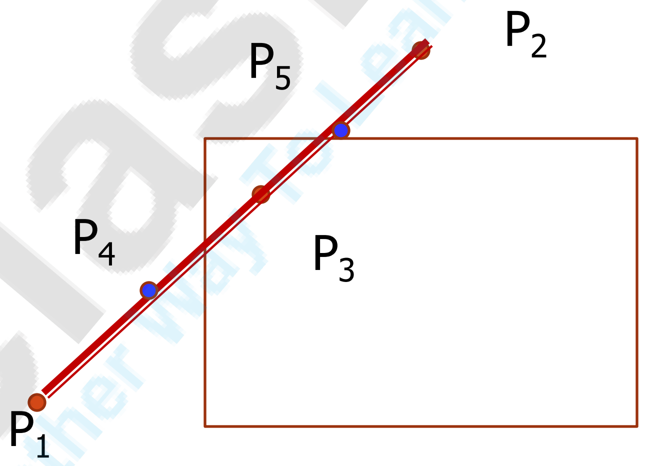
- It is sub divided into 2 equal parts

  $P_1 P_3$ & $P_3 P_2$ .both line segment tested for visibility & found to

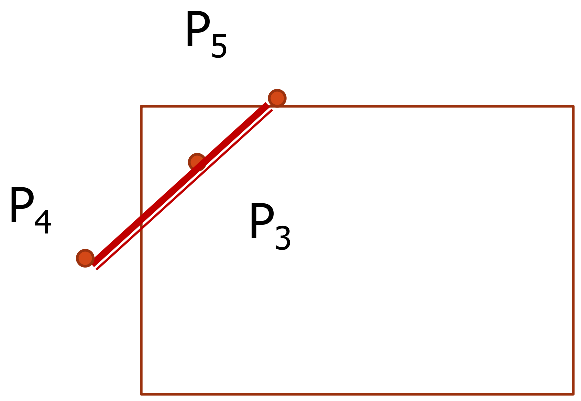
  be partially visible .



- so both line segments are subdivided into two equal parts to get midpoint P_4 & P_5

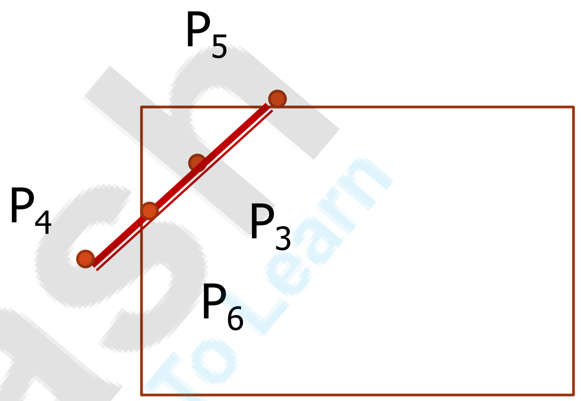


- Line segment $P_1 P_4$ and $P_5 P_2$ are completely invisible & hence rejected

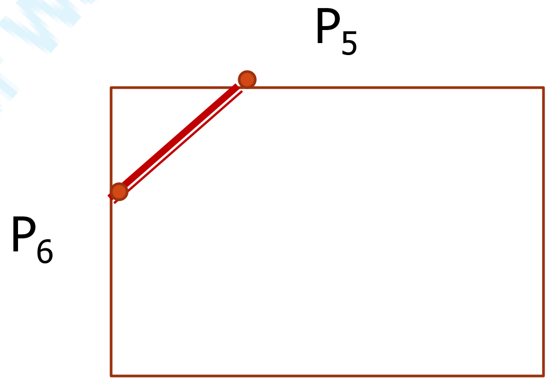


- Line segment P_3 P5 is completely visible hence draw

- Line P_4 P3 is partially visible it is subdivided to get midpoint P_6



- Line segment P_6 P_3 is completely visible & hence it is draw



Q Explain mid point subdivision algorithm. Prove that it work successfully with lines that are partially inside and partially outside the viewing window . 10m

Cyrus Beck algorithm:

- This algo applicable for arbitrary convex region
- It uses parametric equation of line segment to find intersection point of a line with the clipping edges.
- Parametric equation of line segment

from $P_1 P_2$ is,

$$P(t) = P_1 + (P_2 - P_1)\,t \quad 0 <= t <= 1$$

Where t is parameter t=0 at P1 and t=1 at P_2

Consider a convex clipping region R , f is a boundary point of convex region R & n is inner normal for one of its boundaries

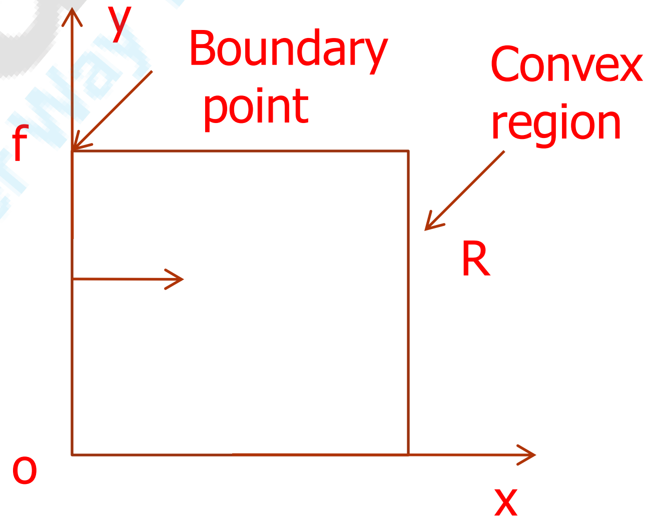


Fig : convex region , boundary point inner normal

- Using value of dot product n.[P(t) – f] as shown in figure.

- If . Product is negative n.[P(t) – f] < 0

  Then vector P(t)-f is pointed away from interior of R

- If . Product is zero n.[P(t) – f] = 0

  Then vector P(t)-f is pointed parallel to plane containing f

  & perpendicular to the normal

- If . Product is positive n.[P(t) – f] > 0

  Then vector P(t)-f is pointed towards the interior of R

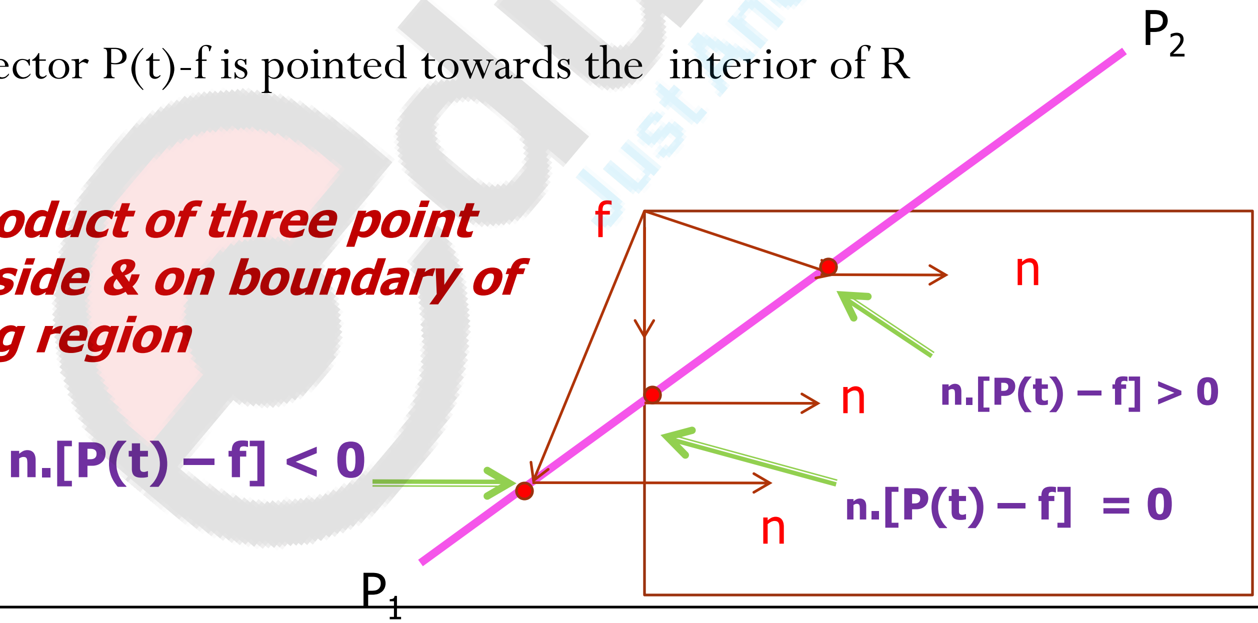


Fig: Dot product of three point inside outside & on boundary of the clipping region

Cyrus Beck algorithm:

1. Read two end points of line say P1 & P2
2. Read vertex coordinate of the clipping window
3. Calculate D= P2 – P 1
4. Assign boundary point (f) with particular b edge
5. Find inner normal vector for corresponding edge
6. Calculate D.n and W= P1 – f
7. If D.n > 0

```
        $t_L = \frac{W\ .\ n}{D\ .\ n}$
    Else
        $t_u = -\frac{W\ .\ n}{D\ .\ n}$
    end if
```

Cyrus Beck algorithm:

8. Repeat steps 4 through 7 for each edge of clipping window
9. Find maximum lower limit & minimum upper limit
10. If maximum lower limit & minimum upper limit do not satisfy condition

    $0 <= t <= 1$ then ignore the line

8. Calculate intersecting point by substituting values of maximum lower limit & minimum upper limit in parametric equation of line P1P2
9. Draw the line segment P (t_L) to P (t_U)
10. stop

Liang Barsky Line Clipping Algorithm

- It uses parametric equation

  $x = x_1 + t\,\Delta x$

  $y = y_1 + t\,\Delta y \qquad 0 <= t <= 1$

  where $\Delta x = x_2 - x_1$

  $\Delta y = y_2 - y_1$

- Point clipping condition in parametric form can be given as

  $x_{wmin} <= x_1 + t\,\Delta x <= x_{wmax}$ &

  $y_{wmin} <= y_1 + t\,\Delta y <= y_{wmax}$

- Liang Barsky express these 4 inequalities with two parameter p & q as

$$t_{P_i} <= q_i \qquad i=1,2,3\ 4$$

- Where parameter p & q define as

$$p_1 = -\Delta x \quad q_1 = x_1 - x_{wmin}$$

$$p_2 = \Delta x \quad q_2 = x_{wmax} - x_1$$

$$p_3 = -\Delta y \quad q_3 = y_1 - y_{wmin}$$

$$p_4 = \Delta y \quad q_4 = y_{wmax} - y_1$$

- If $P1=0$: Line is parallel to left clipping boundary
- If $P2=0$: Line is parallel to right clipping boundary
- If $P3=0$: Line is parallel to bottom clipping boundary
- If $P4=0$: Line is parallel to top clipping boundary
- If $Pi=0$: Line is parallel to one of the clipping boundary corresponding to value of i
- If $q_i<0$: Line is completely outside the boundary & can be eliminated
- If $q_i>=0$: Line is inside the clipping boundary
- If $p_i<0$: Line proceed from outside to inside of the clipping boundary
- If $p_i>0$: Line proceed from inside to outside of the clipping boundary

- For non zero values of Pi line cross the clipping boundary & we have to find parameter t.

$$t = \frac{q_i}{p_i} \quad i = 1 , 2 \ 3 \ 4$$

- This algo calculate two values of parameter t : t1 & t2 that define part of line that lies within the clip rectangle
- Value of t1 is taken by checking the rectangle edge for which line proceeds from outside to the inside ($p < 0$) t1 is taken as largest value among various intersection values with all edges.

- Value of t2 is taken by checking the rectangle edge for which line proceeds from inside to the outside ($p > 0$) t2 is taken as minimum value
- If $t1 > t2$ the line is completely outside the clipping window & it can rejected
- Otherwise values of t1 & t2 are substituted in parametric equation to get end point of clipped line

Liang Barsky Line Clipping Algorithm

1. Read two end points of line P1 (X1 ,Y1) & P2 (X2 ,Y2)
2. Read two corner(left top , right bottom) of the window

   (say Xwmin ,Ywmax , Xwmax ,Ywmin)
3. Calculate value of parameter p_i & q_i for i = 1 ,2 3 4

$$p_1 = -\Delta x \quad q_1 = x_1 - x_{wmin}$$

$$p_2 = \Delta x \quad q_2 = x_{wmax} - x_1$$

$$p_3 = -\Delta y \quad q_3 = y_1 - y_{wmin}$$

$$p_4 = \Delta y \quad q_4 = y_{wmax} - y_1$$

```
4.  If $p_i = 0$ then
    {
        line is parallel to ith boundary
        Now if $q_i < 0$ then
        {
            Line is completely outside the boundary
        hence discard the line segment & go to stop
        }
Else
    {
        Check whether the line is horizontal or vertical & accordingly check
    line end point with corresponding boundaries If line end point lie within the
    bounded area   then used them to draw line otherwise use boundary
    coordinates to draw line. Go to stop.

    }
}
```

```
5.  Initialize values for t1 & t2 as  $t1 = 0$  &  $t2 = 1$
6.  Calculate values for $q_i / p_i$  for  $i = 1,2,3,4$
7.  Select values for $q_i / p_i$ where $p_i < 0$ & assign maximum out of them
    as t1
8.  Select values for $q_i / p_i$ where $p_i > 0$ & assign minimum out of them
    as t2
9.  If( $t1 < t2$ )
    {
       calculate the endpoints of the clipped line as follows
        $xx_1 = x_1 + t1\,\Delta x$          $xx_2 = x_1 + t2\,\Delta x$
        $yy_1 = y_1 + t1\,\Delta y$          $yy_2 = y_1 + t2\,\Delta y$
        draw line ($xx_1, yy_1, xx_2, yy_2$)
    }
10. stop
```

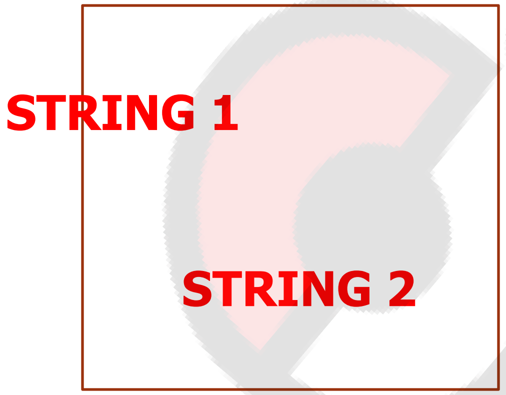
Text Clipping

Q . Write a note on text clipping 5m/ 8m

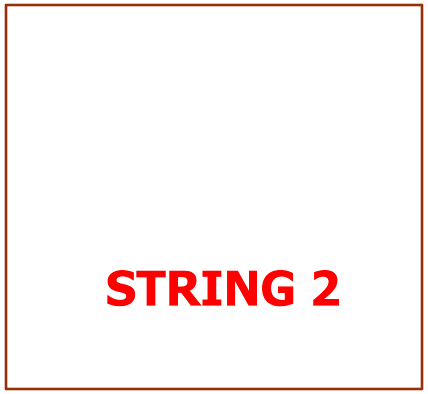
- There are three different method used for character or text clipping.
  1. All or none string clipping
  2. All or none character clipping
  3. Text clipping

All or none string clipping:

- If all the string is inside the clipped window we keep it otherwise the string is discarded . This procedure is implemented by considering a bounding rectangle around the text pattern the boundary position of the rectangle are then compared to the window boundary & the string is rejected if there is any overlapped this method gives fastest text clipping.



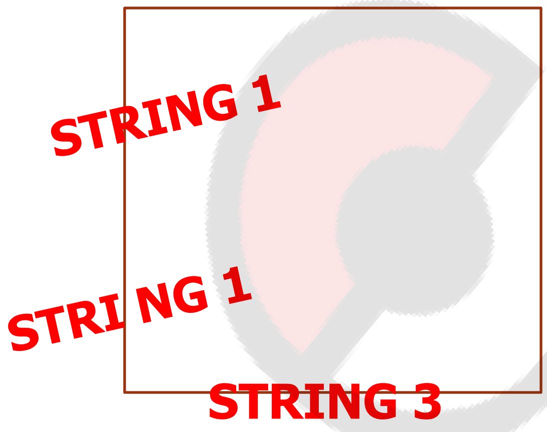
Before clipping.



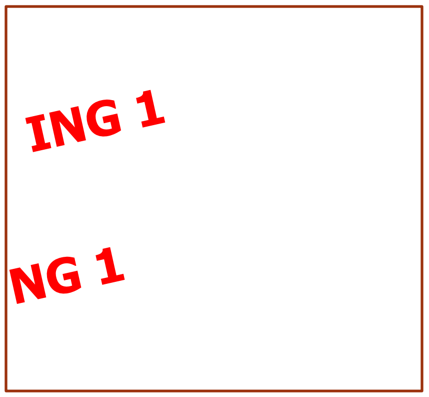
After clipping.

All or none character clipping:

- In this method we discard only those character that are not completely inside the window Boundary limit of individual character is compared to the window & character which either overlap or are outside a window boundary are clipped



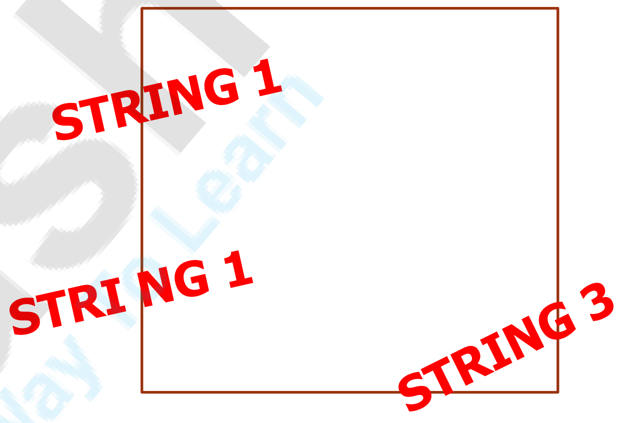
Before clipping.



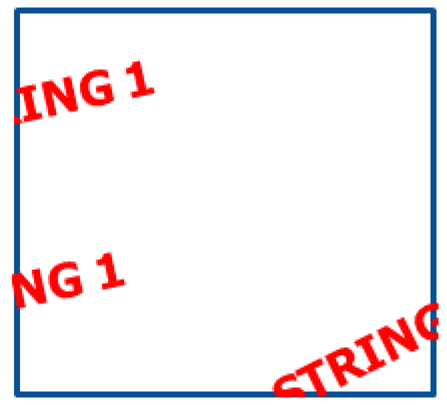
After clipping.

Text clipping:

- In this method either a part of character that is not inside a window boundary or a character outside a window boundary are clipped.
- We can treat character same as line. individual line which forms the character are process by the line clipping algorithm to clipped outside part of overlapping character with clip window boundary
- When character are define by bit maps they are clip by comparing relative position of individual pixel in character grid pattern to th clipping boundaries



Before clipping.



After clipping.