Practical No 12

Cohen Sutherland Line Clipping Algorithm

Line Clipping

- The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.

There are 3 possibilities for the line –

- Visible
- Not Visible
- Clipping Case

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

$X_{3=X}1+(y-y_1)/m$

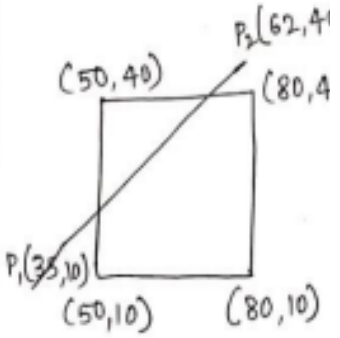
where $y = y_{wmax}$

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

Use the Cohen Sutherland algorithm to clip two lines P1(35,10)- P2(65,40) and P3(65,20)-P4(95,10) against a window A(50,10), B(80,10), C(80,40) and D(50,40).

$P_1 = (35, 10)$ $\quad wx_1 = 50$ $\quad wy_1 = 40$

$P_2 = (62, 40)$ $\quad wx_2 = 80$ $\quad wy_2 = 10$



$P_1 = 0001$

$P_2 = 1000$

ANDing 0000

Line is partially visible.

$$m = \frac{40-10}{62-35} = \frac{30}{27}$$

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

$= \frac{30}{27}(50-35)+10$

$= \frac{30}{27}(15)+10 = 26.66$

$x_1 = \frac{1}{m}(y_T - y) + x$

$= \frac{27}{30}(40-10)+35$

$= \frac{27}{30}(30)+35 = 62$

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

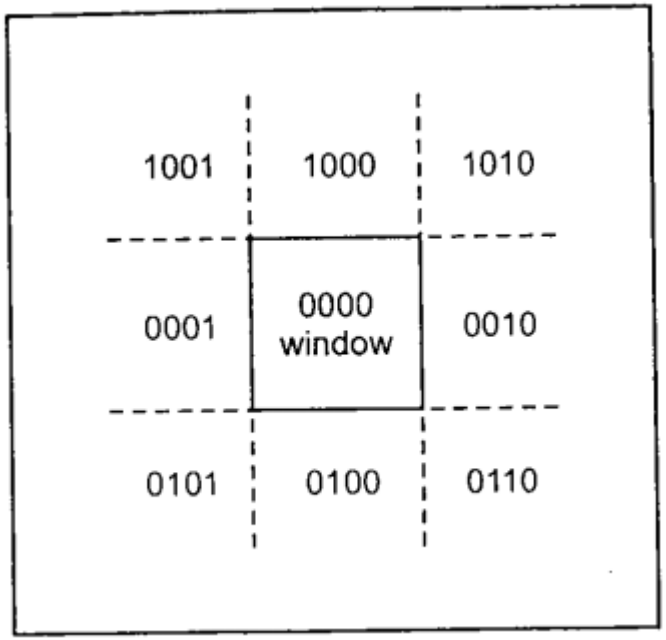
$= \frac{30}{27}(80-35)+10$

$= \frac{30}{27}(45)+10 = 60$

$x_2 = \frac{1}{m}(y_B - y) + x$

$= \frac{27}{30}(10-10)+35$

$= 35$



Four-bit codes for nine regions

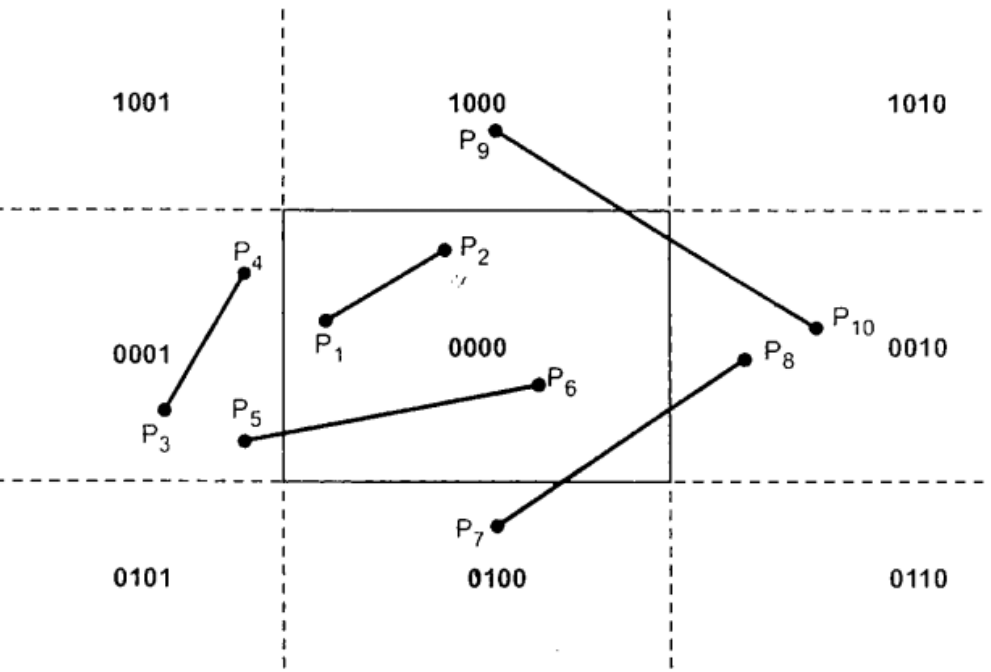
Set Bit 1 – if the end point is to the **left** of the window

Set Bit 2 – if the end point is to the **right** of the window

Set Bit 3 – if the end point is **below** the window

Set Bit 4 – if the end point is **above** the window

Otherwise, the bit is set to zero.



$$\text{Left} : x_L, y = m(x_L - x_1) + y_1 \quad ; \quad m \neq \infty$$

$$\text{Right} : x_R, y = m(x_R - x_1) + y_1 \quad ; \quad m \neq \infty$$

$$\text{Top} : y_T, x = x_1 + \left(\frac{1}{m}\right)(y_T - y_1) \quad ; \quad m \neq 0$$

$$\text{Bottom} : y_B, x = x_1 + \left(\frac{1}{m}\right)(y_B - y_1) \quad ; \quad m \neq 0$$

Line	End Point Codes		Logical ANDing	Result
$P_1\ P_2$	0000	0000	0000	Completely visible
$P_3\ P_4$	0001	0001	0001	Completely invisible
$P_5\ P_6$	0001	0000	0000	Partially visible
$P_7\ P_8$	0100	0010	0000	Partially visible
$P_9\ P_{10}$	1000	0010	0000	Partially visible

Cohen-Sutherland Line clipping algorithm

Step 1: Scan end points for the line P1(x1, y1) and P2(x2, y2)

Step 2: Scan corners for the window as (Wx1, Wy1) and (Wx2, Wy2)

Step 3: Assign the region codes for endpoints P1 and P2 by

Bit 1 - if (x < Wx1)

Bit 2 - if (x < Wx2)

Bit 3 - if (x < Wy2)

Bit 4 - if (x < Wy1)

Step 4: Check for visibility of line P1, P2

- If region codes for both end points are zero then the line is visible, draw it and jump to step 9.
- If region codes for end points are not zero and the logical and operation of them is also not zero then the line is invisible, reject it and jump to step 9.
- If region codes for end points does not satisfies the condition in 4(i) and 4(ii) then line is partly visible.

Step 5: Determine the intersecting edge of the clipping window by inspecting the region codes for endpoints.

- If region codes for both the end points are non-zero, find intersection points P1 and P2 with boundary edges of clipping window with respect to point P1 and P2.
- If region code for any one end point is non zero then find intersection point P1 or P2 with the boundary edge of the clipping window with respect to it.

Step 6: Divide the line segments by considering intersection points.

Step 7: Reject the line segment if any of the end point of it appear outside the window.

Step 8: Draw the remaining line.

Step 9: Exit

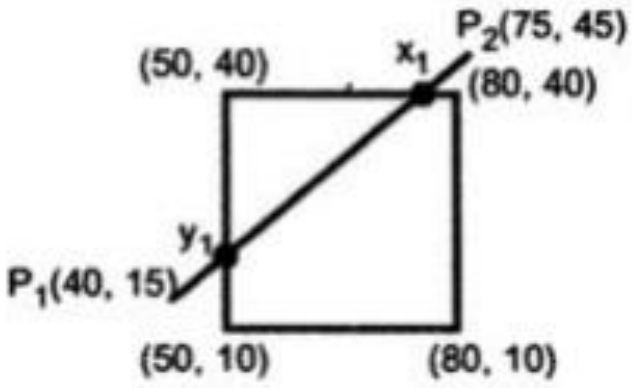
Use Cohen-Sutherland algorithm to clip two lines PI (40, 15) - - P2 (75, 45) and P3 (70, 20) — P4 (100, 10) against a window A (50, 10), B (80, 10). C(80, 40) & D(50,40)

Line 1 : P1 (40, 15) - P2 (75, 45) $Wx_1 = 50$ $Wy_1 = 40$ $Wx_2 = 80$ $Wy_2 = 10$

Point	Endeode	ANDing		
P1	0001	0000	(Partially visible)	$m = \frac{45-15}{75-40}$
P2	0000			

$$y_1 = m(x_L - x) + y = \frac{6}{7}(50\text{-}40)+15 = 23.57$$

$$y_2 = m(x_R - x) + y = \frac{6}{7}(80\text{-}40)+15 = 49.28$$



$$x_1 = \frac{1}{m}(y_T - y) + x = \frac{7}{6}(40\text{-}50)+40 = 69.16$$

$$x_2 = \frac{1}{m}(y_B - y) + x = \frac{7}{6}(10\text{-}15)+40 = 34.16$$

P3 (70,20) – P4 (100,10) Wx1 = 50 Wy1 = 40 Wx2 = 80 Wy2 = 10

Point	Endeode	ANDing	
P3	0000	0000	(Partially visible)
P4	0010		

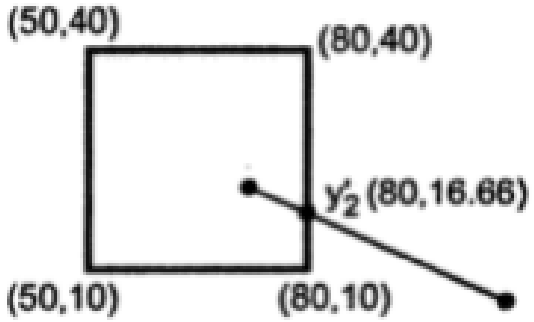
$$\text{Slope } m` = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$$

$$y`_1 = m(x_L - x) + y = \frac{-1}{3}(50-70)+20 = 26.66$$

$$x`_1 = \frac{1}{m}(y_T - y) + x = -3\,(40-20)+70 = 10$$

$$y`_2 = m(x_R - x) + y = \frac{-1}{3}(80-70)+20 = 16.66$$

$$x`_2 = \frac{1}{m}(y_B - y) + x = -3\,(10-20)+70 = 100$$



```
// Cohen Sutherland Line Clipping
#include<stdio.h>
#include<graphics.h>
void main()
{
int gd=DETECT, gm;
float i,xmax,ymax,xmin,ymin,x11,y11,x22,y22,m;
float a[4],b[4],c[4],x1,y1;
clrscr();
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
printf("\nEnter the top-left coordinate of viewport: ");
scanf("%f %f",&xmin,&ymin);
printf("\nEnter the bottom-right coordinate of viewport: ");
scanf("%f %f",&xmax,&ymax);
rectangle(xmin,ymin,xmax,ymax);
printf("\nEnter the coordinates of 1st end point of line: ");
scanf("%f %f",&x11,&y11);
printf("\nEnter the coordinates of 2nd endpoint of line: ");
scanf("%f %f",&x22,&y22);
line(x11,y11,x22,y22);
for(i=0;i<4;i++)
{
a[i]=0;
b[i]=0;
}
m=(y22-y11)/(x22-x11);
if(x11<xmin) a[3]=1;
if(x11>xmax) a[2]=1;
if(y11<ymin) a[1]=1;
if(y11>ymax) a[0]=1;
if(x22<xmin) b[3]=1;
if(x22>xmax) b[2]=1;
if(y22<ymin) b[1]=1;
if(y22>ymax) b[0]=1;
printf("\nRegion code of 1st pt ");
for(i=0;i<4;i++)
{
        printf("%f",a[i]);
}
printf("\nRegion code of 2nd pt ");
for(i=0;i<4;i++)
{
        printf("%f",b[i]);
}
printf("\nAnding : ");
for(i=0;i<4;i++)
{
        c[i]=a[i]&&b[i];
}
for(i=0;i<4;i++)
        printf("%f",c[i]);
getch();
if((c[0]==0)&&(c[1]==0)&&(c[2]==0)&&(c[3]==0))
{
if((a[0]==0)&&(a[1]==0)&&(a[2]==0)&&(a[3]==0)&&(b[0]==0)&&
(b[1]==0)&&(b[2]==0)&&(b[3]==0))
{
clrscr();
clearviewport();
printf("\nThe line is totally visible\nand not a clipping
candidate");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
}
else
{
clrscr();
clearviewport();
printf("\nLine is partially visible");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
if((a[0]==0)&&(a[1]==1))
{
        x1=x11+(ymin-y11)/m;
        x11=x1;
        y11=ymin;
}
else if((b[0]==0)&&(b[1]==1))
{
        x1=x22+(ymin-y22)/m;
        x22=x1;
        y22=ymin;
}

if((a[0]==1)&&(a[1]==0))
{
        x1=x11+(ymax-y11)/m;
        x11=x1;
        y11=ymax;
}
else if((b[0]==1)&&(b[1]==0))
{
        x1=x22+(ymax-y22)/m;
        x22=x1;
        y22=ymax;
}
if((a[2]==0)&&(a[3]==1))
{
        y1=y11+m*(xmin-x11);
        y11=y1;
        x11=xmin;
}
else if((b[2]==0)&&(b[3]==1))
{
        y1=y22+m*(xmin-x22);
        y22=y1;
        x22=xmin;
}
if((a[2]==1)&&(a[3]==0))
{
        y1=y11+m*(xmax-x11);
        y11=y1;
        x11=xmax;
}
else if((b[2]==1)&&(b[3]==0))
{
        y1=y22+m*(xmax-x22);
        y22=y1;
        x22=xmax;
}
clrscr();
clearviewport();
printf("\nAfter clippling:");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
}
}
else
{
clrscr();
clearviewport();
printf("\nLine is invisible");
rectangle(xmin,ymin,xmax,ymax);
getch();
}
closegraph();
getch();
}
```