**Aim:** To Implement line clipping algorithms

**Theory:**

1. **Explain the viewing transformation pipeline:**

* A point at position (xw,yw) in a designed window is mapped to view port coordinate (xv,yv) so that relative position in the two areas are the same

So the mapping of a world co-ordinate to device co-ordinate is referred to as viewing transformation. This transformation is from window to view port.

* The viewing transformation is formed by the following transformation:
* The normalization transformation N that maps world co-ordinate to normalized device Co-ordinates.
* The workstation transformation W, that maps normalized device coordinate to device coordinates
* Two dimensional viewing transformation pipeline

Mo= Moduling coordinate

WCS=World coordinate

Dc=Device coordinate.

Construct Wcs using modeling coordinate

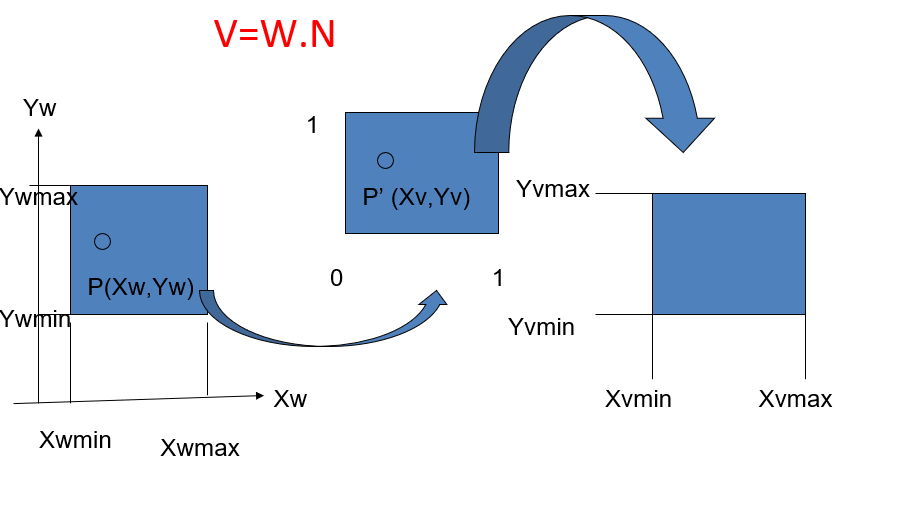
MO

Convert WCS to Viewing Co-ordinate

Map view coordinate to normalized view co-ordinate

Map normalized view port to Device coordinate

DC

****

**Consider a point P(Xw, Yw) in WC’s mapped to P’ (Xv,Yv) in the device coordinate system.**

Then

XV- Xvmin = XW- Xwmin ……….[1]

Xvmax-Xvmin Xwmax-Xwmin

YV- Yvmin = YW- Ywmin ………..[2]

Yvmax-Yvmin Ywmax-Ywmin

From equation [1] & [2]

Xv =Xvmin+ (Xw- Xwmin) \* SX

Yv =Yvmin+ (Yw- Ywmin) \* SY

Where

Xvmax-Xvmin ViewportXextent

SX = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Xwmax-Xwmin WindowXextent

Yvmax-Yvmin ViewportYextent

SY = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ywmax-Ywmin WindowYextent

* This conversation is performed with the following sequence of transformation.
* Translate the object so that the lower left corner of the window is moved to origin.
* Scale the object so that the window has the dimension of the view port.
* Translate the object so that the scaled window area is positioned to view port.

Relative proportions are maintained if the scaling factors are the same

**2. Explain Cohen-Sutherland line clipping algorithm with an example:**

1.Accept the endpoints of the line segment and window boundaries x1, y1, x2, y2, Xwmin,Xwmax,Ywmin,Ywmax

2.Assign a 4 bit code to each end point of the line segment i.e. B1,B2,B3,B4

Where,

If (pt.x < Xwmin.x) then code=code | Left.

If (pt.x > Xwmax.x) then code=code| Right.

If (pt.y < Ywmin.x) then code=code | Bottom.

If (pt.y > Ywmax.x) then code=code | Top.

Such that sign(A)=1 if A>0

=0 otherwise

3.If both the endpoints codes are 0000,the line is visible. Display the line segment STOP. If the logical AND of the endpoint codes is not 0000,then line is not visible. Discard the line segment and STOP.

If the logical AND Of the endpoints codes is 0000,the line segment is clipping candidate.

4.Determine the intersecting boundary:

If Bit B1 is 1,interect with x=Xwmin

If Bit B2 is 1,interect with x=Xwmax

If Bit B3 is 1,interect with y=Ywmin

If Bit B4 is 1,interect with y=Ywmax

5.Determine the intersecting point co-ordinate (x’, y’). The equation of a line passing through

(x1,y1) & (x2,y2) & (x’, y’) will be

X’-x1 / x2-x1 = y’-y1 / y2-y1

X’-x1=(x2-x1/y2-y1)\*(y’-y1)

X’=x1+1/m\*(y’-y1) …………………….(1)

Y’=Ywmin or y’=Ywmax

Similarly

y’=y1+m(x’-x1) then

X’=Xwmin or Xwmax

Where m =y2-y1/x2-x1

6. GOTO to step II

**Example:** Find the clipping coordinate of line joining A(x1,y1)=(-1,5) & B(x2,y2)=(3,8)  
Xwmin=-3,ywmin=1,xwmax=2,ywmax=6

**Solution:**

Step 1: A(x1,y1)=(-1,5) & B(x2,y2)=(3,8)

Xwmin=-3,ywmin=1,xwmax=2,ywmax=6

Step 2: For A(x1,y1)=(-1,5) i.e B4,B3,B2,B1

B1=sign(xwmin-x)=sign(-3+1)=sign(-2)=0

B2=sign(x-xwmax)=sign(-1-2)=sign(-3)=0

B3=sign(ywmin-y)=sign(1-5)=sign(-4)=0

B4=sign(y-ywmax)=sign(5-6)=sign(-1)=0

The code of A(x1,y1)=0000

Point A(-1,5) is visible.

1. **Derive Liang-Barsky line clipping algorithm:**

Faster line clipper have been developed that are based on analysis of the parametric equation a

line segment, which can be written in the form

x=x1+ µ Δx

y=y1+ µ Δy 0≤ µ ≤1 ……..[1]

Where,Δx=x2-x1 and Δy=y2-y1.

Using parametric equation, Cyrus and Beck developed an algorithms that is generally more

efficient than the Cohen-Sutherland algorithms.

Later, Liang-Barsky independently developed faster parametric line clipping algorithms.

Following the Liang-Barsky approach. We first write the point-clipping conditions in parametric form.

Consider the line segment with endpoints (x1,y1) and (x2,y2).

The equation of line is determined by,

x-x1/x2-x1 = y-y1/y2-y1= µ .………….. [1]

x-x1= µ(x2-x1) and y-y1= µ(y2-y1)

X=x1+ µ(x2-x1)=x1+ µ(Δx) …………… [2]

Y=y1+ µ(y2-y1)=y1+ µ(Δy) …………… [3]

Assuming that the clip window is a rectangle in standard position, any point (x,y) is visible if the following inequalities are satisfied.

Xwmin ≤ x1 + μ Δx ≤ xwmax

Ywmin ≤ y1 + μ Δy ≤ ywmax

Where (xwmin xwmax ywmin ywmax) are window boundaries.

Each of these four inequalities can be expressed as:

Μpk ≤ qk …….. ( k=1,2,3,4)

Where K=1,2,3 and 4 corresponds to the left, right

Top, bottom boundaries

Where k=1, left 🡨 boundary

X ≥ xwmin

X1+μ∆x ≥xwmin

X1-xwmin ≥ -μ∆x

Q1 ≥ μp1

Hence p1= - ∆x and q1=x1-xwmin

Where k=2, Right 🡪 boundary

X ≤ xwmax

X1+μ∆x ≤xwmax

Μ∆x ≤ xwmax-x1

Μp2 ≤q2

Hence p2= ∆x q2= xwmax-x1

Where k=3, Bottom |boundary

y ≥ ywmin

y1+μ∆y ≥ ywmin

y1-ywmin ≥ -μ∆y

q3 ≥ μp3

Hence p3= - ∆y and q3=y1-ywmin

Where k=4, Top |boundary

y ≤ ywmax

y1+μ∆y ≤ywmax

μ∆y ≤ ywmax-y1

μp4 ≤q4

Hence p4= ∆y q4= ywmax-y1

1. **Liang-Barsky Line Clipping Pseudocode:**

function Liang\_Barsky\_Line\_Clip(x1, y1, x2, y2, xmin, ymin, xmax, ymax)

dx = x2 - x1

dy = y2 - y1

p1 = -dx

p2 = dx

p3 = -dy

p4 = dy

q1 = x1 - xmin

q2 = xmax - x1

q3 = y1 - ymin

q4 = ymax - y1

u1 = -float('inf')

u2 = float('inf')

if p1 = 0 and q1 < 0:

return "Line is outside the window."

if p3 = 0 and q3 < 0:

return "Line is outside the window."

u1 = max(u1, q1 / p1)

u2 = min(u2, q2 / p2)

u1 = max(u1, q3 / p3)

u2 = min(u2, q4 / p4)

if u1 <= u2:

x1\_clip = x1 + u1 \* dx

y1\_clip = y1 + u1 \* dy

x2\_clip = x1 + u2 \* dx

y2\_clip = y1 + u2 \* dy

return "Clipped line: (" + x1\_clip + ", " + y1\_clip + ") to (" + x2\_clip + ", " + y2\_clip + ")"

else:

return "Line is outside the window."

**Example:**

Find the clipping coordinate of a line AB with endpoints (7,5) and (9,7).

Solution:

**Step1:**  Let x1=7 , x2=9 , y1=5 , y2=7

Xwmin=4 , Ywmin=4 , XWmax =10 , YWmax=9

**Step2:**

p1= -∆x=-2 q1=x1-xwmin =3

p2= ∆x =2 q2=xwmax-x1 =3

p3=-∆y=-2 q3=y1-ywmin =1

p4= ∆y =2 q4= ywmax-y1=4

**Step3:**

Pk != 0

**Step4:**

P1<0 r1=q1/p1=-3/2

P2>0 r2=q2/p2=3/2

P3<0 r3=q3/p3=-1/2

P4>0 r4=q4/p4=4/2

**Step5:**

P1<0 r1=q1/p1=-3/2

P3<0 r3=q3/p3=-1/2

µ1={-3/2 , -1/2 , 0 } max

= 0

**Step 6:**

P2>0 r2=q2/p2=3/2

P4>0 r4=q4/p4=4/2

μ2={3/2 , 4/2 , 1} min

= 1

**Step 7:**

μ1!= μ2 i.e. 0 !=1

**Step 8:**

* X’=x1+ μ1∆x =7+0=7
* y’=y1+ µ1∆y=5+0=5

I1=(7,5)

* X’’=x1+μ2 ∆x =7+1 x (2) = 9
* Y’’=y1+µ2∆y=5+1 x (2) = 7

I2=(9,7)

I1I2 is visible.

**Source code:**

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

int x1,y1,x2,y2,xmax,xmin,ymax,ymin,xx1,yy1,xx2,yy2,dx,dy,i;

int p[4],q[4];

float t1,t2,t[4];

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

printf("Enter the lower co-ordinates of window");

scanf("%d%d",&xmin,&ymin);

printf("Enter the upper co-ordinates of window");

scanf("%d%d",&xmax,&ymax);

setcolor(RED);

rectangle(xmin,ymin,xmax,ymax);

printf("Enter x1 y1 x2 y2");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

line(x1,y1,x2,y2);

dx=x2-x1;

dy=y2-y1;

p[0]=-dx;

p[1]=dx;

p[2]=-dy;

p[3]=dy;

q[0]=x1-xmin;

q[1]=xmax-x1;

q[2]=y1-ymin;

q[3]=ymax-y1;

for(i=0;i < 4;i++){

if(p[i]!=0){

t[i]=(float)q[i]/p[i];

}

else

if(p[i]==0 && q[i] < 0)

printf("line completely outside the window");

else

if(p[i]==0 && q[i] >= 0)

printf("line completely inside the window");

}

if (t[0] > t[2]){

t1=t[0];

}

else{

t1=t[2];

}

if (t[1] < t[3]){

t2=t[1];

}

else{

t2=t[3];

}

if (t1 < t2){

xx1=x1+t1\*dx;

xx2=x1+t2\*dx;

yy1=y1+t1\*dy;

yy2=y1+t2\*dy;

printf("line after clipping:");

setcolor(WHITE);

line(xx1,yy1,xx2,yy2);

}

else{

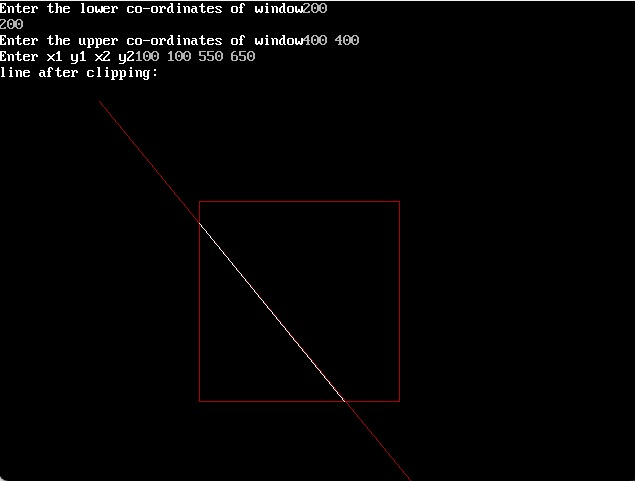
printf("line lies out of the window");

}

getch();

}

**Output:**



**Conclusion:** I have understood how to implement line clipping algorithms in computer graphics.