Computer Graphics Lab Experiment No: - 09
Aim: Implement Line Clipping Algorithm: Liang Barsky.

Clipping

When we have to display a large portion of the picture, then not only scaling & translation is necessary, the visible part of picture is also identified. This process is not easy. Certain parts of the image are inside, while others are partially inside. The lines or elements which are partially visible will be omitted.

For deciding the visible and invisible portion, a particular process called clipping is used. Clipping determines each element into the visible and invisible portion. Visible portion is selected. An invisible portion is discarded.

Clipping can be applied through hardware as well as software. In some computers, hardware devices automatically do work of clipping. In a system where hardware clipping is not available software clipping applied.

Liang-Barsky Line Clipping Algorithm:

The algorithm was introduced by “You-Dong Liang” and “Brian A. Barsky.” It is used for line clipping. It is a more powerful algorithm than the Cohen-Sutherland algorithm.

We can use the parametric equation of line and inequalities.

• These are used to describe the range of windows to find out the intersection points between the line and the clipping window.

In this algorithm, we have to find the intersection point based on a time interval.

Time interval (t) can be defined as travelling time between initial position (0) to final position (1). Then we have,

$0 < t < 1$ (Here, t lies between 0 and 1)

We have the formula to find x and y points of the line-

x= x1 + t. Δx (For point x)

y= y1 + t. Δy (For point y)

To check that the point lies between the window or outside the equation is-

Xwmin <= x1 + t. Δx <= Xwmax

Ywmin <= y1 + t. Δy <= Ywmax

These two conditions can be written as-

x1 + t. Δx >= Xwmin

x1 + t. Δx <= Xwmax

y1 + t. Δy >= Ywmin

y1 + t. Δy <= Ywmax

We can take a common expression for above four conditions. It will be-

t.pk <= qk (Here the value of k is multiple)

t = qk / pk

p1 = -Δx q1 = x1 - xwmin (For left boundary)

p2 = Δx q2 = xwmax - x1 (For right boundary)

p3 = -Δy q3 = y1 - ywmin (For bottom boundary)

p4 = Δy q4 = ywmax – y1(For top Boundary)

Algorithm of Liang-Barsky Line Clipping:

Step 1: Set the endpoints of the line (x1, y1) and (x2, y2).

Step 2: Calculate the value of p1, p2,p3, p4 and q1, q2, q3,q4.

Step 3: Now we calculate the value of t

t1 = 0 (For initial point)

t2 = 1 (For final point)

Step 4: Now, we have to calculate the value of pk and qk

If

pk = 0

then

{The line is parallel to the window}

If

Qk < 0

then

{The line is completely outside the window}

Step 5: If we have non zero value of pk -

If

pk < 0

then

t1 = max (0, qk / pk)

If

pk > 0

then

t2 = min (1, qk / pk)

Now, if t1 < t2 {If t1 value is changed

Then the first point is outside the window.

If t2 value is changed

Then the second point is outside the window}

else

t1 > t2

then

{Line is completely outside the window}

Step 6: Stop.

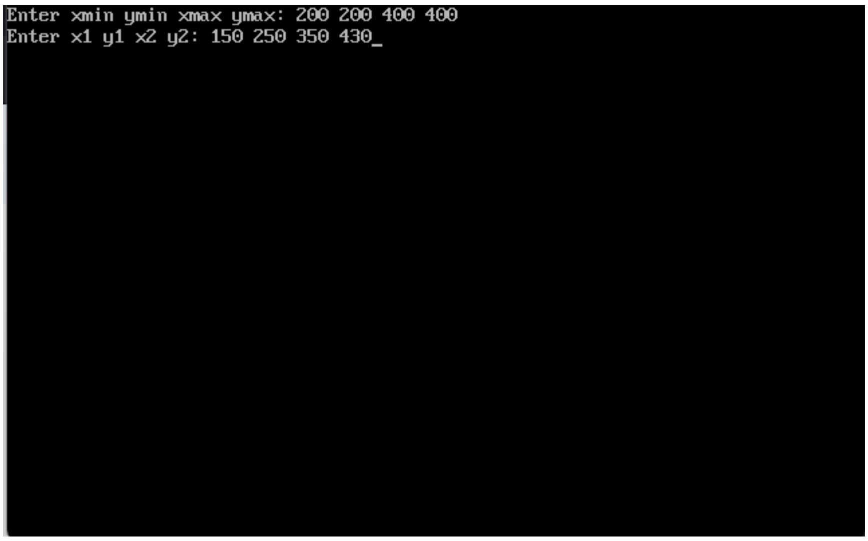
Code

```
#include<stdio.h>
#include<graphics.h>
int main()
{
	int i,gd=DETECT,gm,x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
	float t1,t2,p[4],q[4],temp;
	clrscr();
  printf("Enter xmin ymin xmax ymax: ");
	scanf("%d %d %d %d",&xmin,&ymin,&xmax,&ymax);
	printf("Enter x1 y1 x2 y2: ");
	scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
	initgraph(&gd,&gm,"c:\\TURBOC3\\BGI");
	rectangle(xmin,ymin,xmax,ymax);
```

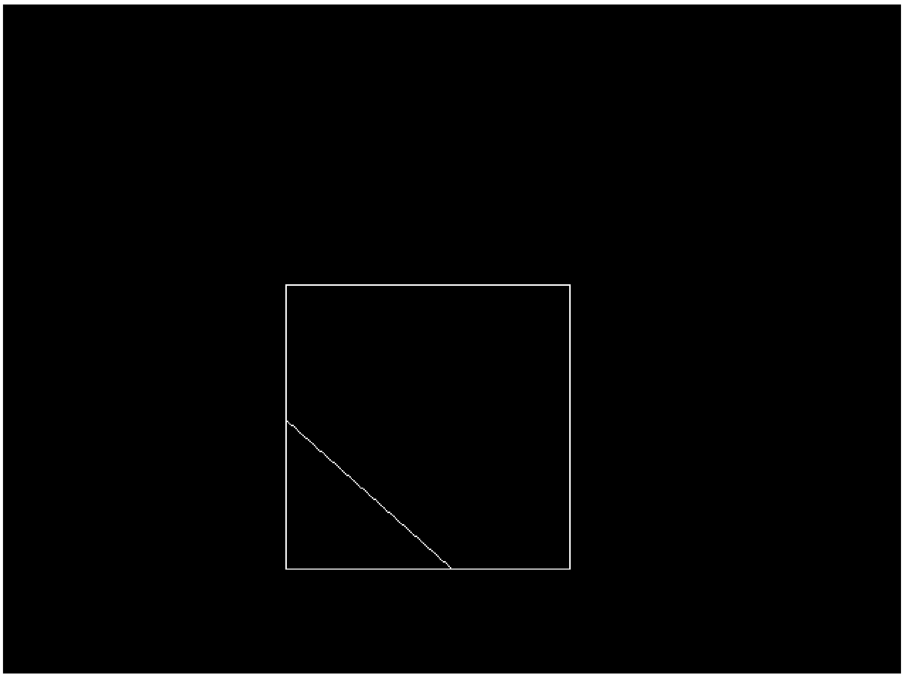
```
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;
for(i=0;i<4;i++)
{
if(p[i]==0)
{
printf("Line is parallel to one of the window boundaries");
if(q[i]>=0)
{
if(i<2)
{
	if(y1<ymin)
	{
	y1=ymin;
	}
	if(y2>ymax)
	{
	y2=ymax;
	}
	line(x1,y1,x2,y2);
```

```
}
if(i>1)
{
	if(x1<xmin)
	{
	x1=xmin;
	}
	if(x2>xmax)
	{
	x2=xmax;
	}
	line(x1,y1,x2,y2);
}
}
}
}
t1=0;
t2=1;
for(i=0;i<4;i++)
{
temp=q[i]/p[i];
if(p[i]<0)
{
if(t1<=temp)
t1=temp;
}
else
{
```

```
	if(t2>temp)
	t2=temp;
	}
	}
	if(t1<t2)
	{
	xx1=x1+t1*p[1];
	xx2=x1+t2*p[1];
	yy1=y1+t1*p[3];
	yy2=y1+t2*p[3];
	line(xx1,yy1,xx2,yy2);
	}
	getch();
	return 0;
}
```



Aim: Implement Line Clipping Algorithm: Liang Barsky.



Conclusion: