Experiment No. 7
Liang Barsky Line Clipping Algorithm
Name: AARYAN CHANDRAKANT GOLE
Roll Number: 12
Date of Performance:
Date of Submission:

Experiment No. 7

Aim: To implement Line Clipping Algorithm: Liang Barsky

Objective:

To understand the concept of Liang Barsky algorithm to efficiently determine the portion of a line segment that lies within a specified clipping window. This method is particularly effective for lines predominantly inside or outside the window.

Theory:

This Algorithm was developed by Liang and Barsky. It is used for line clipping as it is more efficient because it uses more efficient parametric equations to clip the given line.

These parametric equations are given as:

$x = x1 + tdx$

$y = y1 + tdy, 0 <= t <= 1$

Where $dx = x2 – x1$ & $dy = y2 – y1$

Algorithm:

1. Read 2 endpoints of line as p1 (x1, y1) & p2 (x2, y2).

2. Read 2 corners (left-top & right-bottom) of the clipping window as (xwmin, ywmin, xwmax, ywmax).

3. Calculate values of parameters pi and qi for i = 1, 2, 3, 4 such that

$p1 = -dx, q1 = x1 – xwmin$

$p2 = dx, q2 = xwmax – x1$

$p3 = -dy, q3 = y1 – ywmin$

$p4 = dy, q4 = ywmax – y1$

4. if pi = 0 then line is parallel to ith boundary

if qi < 0 then line is completely outside boundary so discard line

else, check whether line is horizontal or vertical and then check the line endpoints with the corresponding boundaries.

5. Initialize t1 & t2 as

t1 = 0 & t2 = 1

6. Calculate values for qi/pi for i = 1, 2, 3, 4.

7. Select values of qi/pi where pi < 0 and assign maximum out of them as t1.

8. Select values of qi/pi where pi > 0 and assign minimum out of them as t2.

9. if (t1 < t2)
{
xx1 = x1 + t1dx

xx2 = x1 + t2dx

yy1 = y1 + t1dy

yy2 = y1 + t2dy

line (xx1, yy1, xx2, yy2)
}

10. Stop.

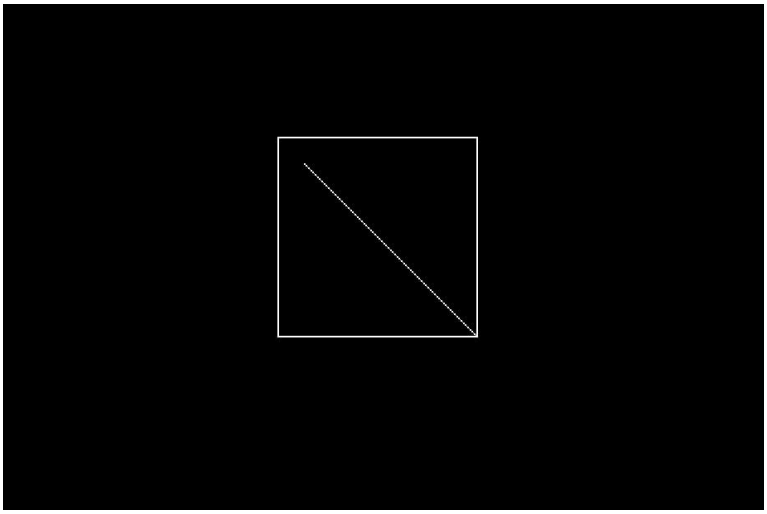
Program:

```
 #include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
 int main()
```

```
{
int i,gd=DETECT,gm;
int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
float t1,t2,p[4],q[4],temp;
x1=120;
y1=120;
x2=300;
y2=300;
xmin=100;
ymin=100;
xmax=250;
ymax=250;
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 clipping boundary");
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[2];
xx2 = x1 + t2 * p[2];
yy1 = y1 + t1 * p[3];
yy2 = y1 + t2 * p[3];
line(xx1,yy1,xx2,yy2);
}
delay(5000);
closegraph();
return 0; }
```

Output:



Conclusion:

In conclusion, the Liang-Barsky algorithm is a powerful and efficient method for line clipping in computer graphics. It provides a systematic way to determine which portion of a line lies within a specified clipping window, helping optimize rendering and improve the visual representation of objects on the screen.

By efficiently discarding portions of lines that are outside the viewing area, it reduces unnecessary computational overhead, making it a valuable tool for real-time graphics applications