**Liang Barsky**

|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No.**

**8**



Semest

er



S.E. Semester III



–



Computer Engineering



Subj



Computer Graphics



Subject Professor In



-



char



Prof. Sachin Deshpande



Assisting

Teachers



Prof. Sachin Deshpande



|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**EXPERIMENT DETAILS:**

|  |  |
| --- | --- |
| Title | Write a program to implement Liang Barsky Line Clipping |
| Tools/Language Used | Dev C++ |

**Liang Barsky**

The Liang-Barsky algorithm is a line clipping algorithm. This algorithm is more efficient than Cohen–Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping. This algorithm is considered to be the faster parametric line-clipping algorithm. The following concepts are used in this clipping:

The parametric equation of the line.

The inequalities describing the range of the clipping window which is used to determine the intersections between the line and the clip window.

The parametric equation of a line can be given by,

X = x1 + t(x2-x1)

Y = y1 + t(y2-y1)

Where, t is between 0 and 1.

Then, writing the point-clipping conditions in the parametric form:

xwmin <= x1 + t(x2-x1) <= xwmax

ywmin <= y1 + t(y2-y1) <= ywmax

The above 4 inequalities can be expressed as,

tpk <= qk

Where k = 1, 2, 3, 4 (correspond to the left, right, bottom, and top boundaries, respectively).

The p and q are defined as,

p1 = -(x2-x1), q1 = x1 - xwmin (Left Boundary)

p2 = (x2-x1), q2 = xwmax - x1 (Right Boundary)

p3 = -(y2-y1), q3 = y1 - ywmin (Bottom Boundary)

p4 = (y2-y1), q4 = ywmax - y1 (Top Boundary)

When the line is parallel to a view window boundary, the p value for that boundary is zero.

When pk < 0, as t increase line goes from the outside to inside (entering).

When pk > 0, line goes from inside to outside (exiting).

When pk = 0 and qk < 0 then line is trivially invisible because it is outside view window.

When pk = 0 and qk > 0 then the line is inside the corresponding window boundary.

Using the following conditions, the position of line can be determined:

Condition Position of line

pk = 0 parallel to the clipping boundaries

pk = 0 and qk < 0 completely outside the boundary

pk = 0 and qk >= 0 inside the parallel clipping boundary

pk < 0 line proceeds from outside to inside

pk > 0 line proceeds from inside to outside

Parameters t1 and t2 can be calculated that define the part of line that lies within the clip rectangle.

When,

pk < 0, maximum(0, qk/pk) is taken.

pk > 0, minimum(1, qk/pk) is taken.

If t1 > t2, the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter t.

**Algorithm :**

Step 1:

Accept Window Extents (Xwmin, Ywmin) (Xwmax, Ywmax)

Accept End point coordinates of a line segment (x1, yq) (x2, y2)

Step 2:

Calculate pk qk and rk where r k= pk/q k

Step 3:

During the calculation for any k if pk = 0 and qk < 0 implies line is parallel to on of the edge and is outside, therefore reject the line and STOP

Step 4:

Calculate the two point of intersections say u1 and u2 as given below

u1 = max{0, rk} for all pk < 0

u2 = min(0, rk } for all pk > 0

Step 5:

If u1 > u2 implies line is totally outside, therefore reject and STOP

Step 6:

1. Else Calculate the point of intersection I`(x`,y`) and I``(x``,y``) by using u1 and u2 respectively as

x =x 1+u1( Δ(x )) y =y1+u 1(Δ( y ))

x =x 2+u2( Δ(x )) y =x 2+u2( Δ( y))

Step 7:

Display the line between I` and I``

Step 8:

return.

**Program**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

#include<conio.h>

int main()

{

int i,gd,gm;

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

float t1,t2,p[4],q[4],temp;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,(char\*)"");

printf("Enter Starting point coordinates x1,y1 of line AB ");

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

printf("Enter ending point coordinates x2,y2 of line AB ");

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

setcolor(GREEN);

line(x1,y1,x2,y2);

setcolor(RED);

printf("Accept window boundary xwmin,ywmin,xwmax,ywmax");

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

rectangle(xmin,ymin,xmax,ymax);

setcolor(WHITE);

dx=x2-x1;

dy=y2-y1;

p[0]=-dx; q[0]=x1-xmin;

p[1]=dx; q[1]=xmax-x1;

p[2]=-dy; q[2]=y1-ymin;

p[3]=dy; 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[1];

xx2 = x1 + t2 \* p[1];

yy1 = y1 + t1 \* p[3];

yy2 = y1 + t2 \* p[3];

line(xx1,yy1,xx2,yy2);

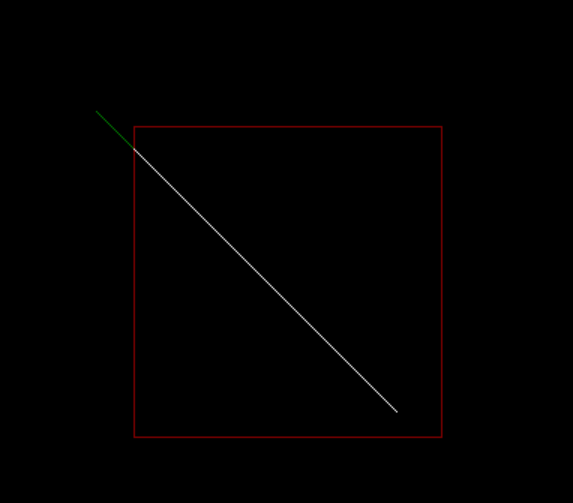
}

delay(10);

getch();

}

**Output**

****

**Advantages**

* Since intersection calculations are reduced.
* Each update of parameters require only one division & window intersection of lines are calculated once, when final values have been computed.

**Disadvantages :**

* Liang-Barsky algorithm involves the parametric equation of line.
* We have to derive equations to check whether the line is inside or outside the clipping window.