Implementation of Efficient Line Clipping Algorithm

Deepali Dev, Pooja Saharan

Abstract: ***This paper proposes a new and efficient line clipping algorithm against a two-dimensional rectangular window. As Cohen Sutherland line clipping algorithm uses four-bit binary codes and Liang Barsky line clipping algorithm uses parametric line equation to find the intersection points with rectangular window, the proposed algorithm uses simple line equation which makes the process of finding intersection points and process of clipping the line segment very simple and time efficient. Comparing to other algorithms of line clipping, the proposed algorithm is faster in terms of speed and more efficient for calculations.***

Keywords: ***Line clipping, Intersection points***

I. INTRODUCTION

Clipping is an important part of computer graphics which decides the efficiency of graphics. Clipping is done in two aspects: first, find the location of window and clipping object, second is to calculate the intersection points between rectangular window and object. There are many algorithms such as Cohen Sutherland and Liang Barsky present for clipping a line segment.

Cohen Sutherland line clipping algorithm [1]involves a lot of intersection calculations. It uses four-bit binary code to calculate the intersection points with the rectangular boundary.

Liang Barsky algorithm [3] for line clipping uses a parametric equation.

The outline of paper is as follows. In section II we discuss our proposed algorithm for clipping the line segment. Section III shows implementation details. Section IV concludes our work and discusses the future work.

II. PROPOSED ALGORITHM

# define
Cond1 = ((Xwmin<= X1<= Xwmax) && (Ywmin<=y1<= Ywmax))
Cond 2= ((Xwmin<= X2<= Xwmax) && (Ywmin<=y2<= Ywmax));
1. Take input Wmin:(Xwmin,Ywmin) and Wmax(Xwmax,Ywmax).
2. Take input end point of the line (x1,y1) and (x2,y2).
3 Check **if** ((Xwmin<= x1<= Xwmax) && (Ywmin<=y1<= Ywmax)) && ((Xwmin<= x2<= Xwmax) && (Ywmin<=y2<= Ywmax));
Then Line(x1,y1,x2,y2) and exit.
4 **Else**find the Slope of the line
if ((x2!== x1)&&(y2!==y1))
m=(y2-y1)/(x2-x1).
5 For line (y=mx+c) c= y1 - (m*x1).
6 Find the intersection points as
a[0][0]=xmin
a[0][1]=(m*xmin) + c
a[1][0]=xmax
a[1][1]=(m*xmax)+c
a[2][0]=(ymin-c)/m
a[2][1]=ymin
a[3][0]=(ymax-c)/m
a[3][1]=ymax
7 **If**(!cond1 && !cond2)
Find the two inter section points which satisfy the boundary condition
If there are points then draw line and exit.
8 **If**(cond1 (+) cond2)
Find the two inter section points which satisfy the boundary condition.
Now check (x1<=Xinter<x2)&&(Y1<=Yinter<=Y2)
Draw lineWhereXinter is the intersection point at x-axis
And Yinter is the intersection point at y-axis
9 **If**(x2==x1)
If((xmin<=x1)&&(x1<=xmax)
Then intersection point is (x1,ymin)(x1,ymax)
If((ymin>=y1>=ymax)&&(ymin>=y2>=ymax))
Line(x1,ymin,x1,ymax)
If(ymin<=y1) &&(y1<=ymax)
If (y2>y1)
Line(x1, y1, x1, ymax)
Else
Line(x1, y1, x1, ymin)
Else if (ymin<=y2) &&(y2<=ymax)
If (y1>y2)
Line(x2, y2, x2, ymax)
Else
Line(x2, y2, x2, ymin)
10**. If** (y2==y1)
If ((ymin<=y1) && (y1<=ymax)
Then intersection point is (xmin, y1) (xmax, y1)
If ((xmin>=x1>=xmax) && (xmin>=x2>=xmax))
Line(xmin, y1, xmax, y1)
If (ymin<=y1) &&(y1<=ymax)
If(x2>x1)
Line(x1, y1, xmax, y1)
Else
Line(x1, y1, x1, ymin)

Revised Manuscript Received on May 05, 2019.

Deepali Dev, Department of Information Technology, ABES Engineering College, Ghaziabad, India
Pooja Saharan, Department of Computer Science, ABES Engineering College, Ghaziabad, India

Else if (ymin<=y2) && (y2<=ymax)
If(y1>y2)
Line(x2, y2, xmax, y2)
Else
Line(x2, y2, xmin, y2)
11. Exit

III. IMPLEMENTATION DETAILS

We have implemented the new algorithm on PC and compared its performance with that of Liang Barsky algorithm. Our machine is based on Pentium IV 2.00GHz, and the compiler is Turbo C running on Windows XP

a. CASE I (Both endpoints of line are inside the window)

We have taken the coordinates of rectangular boundary i.e. Wmin (100,100) &Wmax(200,200)and coordinates of line end point i.e. (x1,y1): (120,120),(x2,y2):(180,180)

Now we have checked the condition: if ((Xwmin<= X1<= Xwmax) && (Ywmin<=y1<= Ywmax)) && ((Xwmin<= X2<= Xwmax) && (Ywmin<=y2<= Ywmax))
i.e. (100<120<200) && (100<120<200) && (100<180<200) && (100<180<200)

This condition is true for above mentioned coordinates of line end point and line completely lies in the rectangular boundary and draw a line

Line (120,120,180,180)

Before Clipping

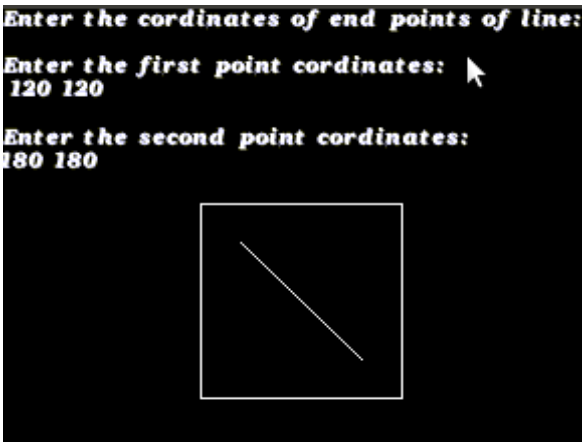

Fig. 1 Both end Points are inside the window

After Clipping

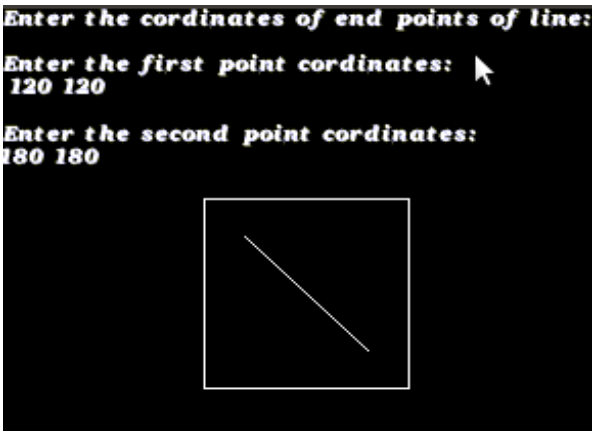

Fig. 2 Results after clipping

b. CASE II (One endpoint is inside the window and another point is outside the window)

Coordinates of rectangular boundary i.e. Wmin (100,100) &Wmax(200,200) and coordinates of line end point i.e. (x1,y1): (120,140) ,(x2,y2):(180,240)

Now we have checked the condition: if ((Xwmin<= X1<= Xwmax) && (Ywmin<=y1<= Ywmax)) && ((Xwmin<= X2<= Xwmax) && (Ywmin<=y2<= Ywmax));
i.e.
(100<=120<=200)&&(100<=140<=200)&&(100<180<200) &&(100<=140<=200) which is false
Now Find slope of line if ((X2!== X1)&&(Y2!==Y1)) m= (240 – 140) / (180-120) = 5/3
intersection to window (xmin, (m*xmin)+ c) ,(xmax `,(m*xmax)+c), ((ymin-c)/m, ymin), ((ymax-c)/m, ymax):
i.e (100,106.66), (200,273.32), (96,100), (156,200)
and now we checked condition: If(!cond1 && !cond2) which is false
then we checked condition: (cond1 (+) cond2)
which is true
then two point which satisfying window condition i.e. (100,106.66),(156,200)
check which point is in between (120,140) and(180,240)
(156,200) satisfying so the intersection point will be
(120,140) and (156,200)

Before Clipping

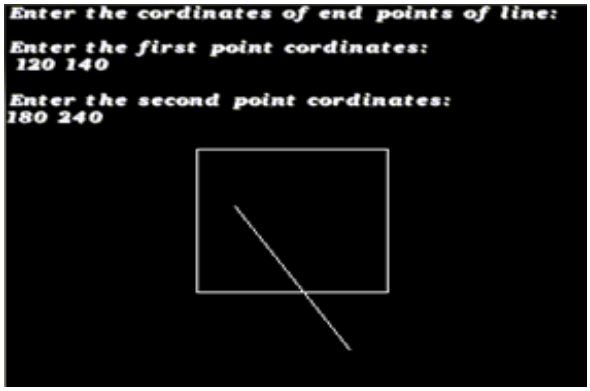

Fig. 3 one endpoint inside the window and other outside

After Clipping

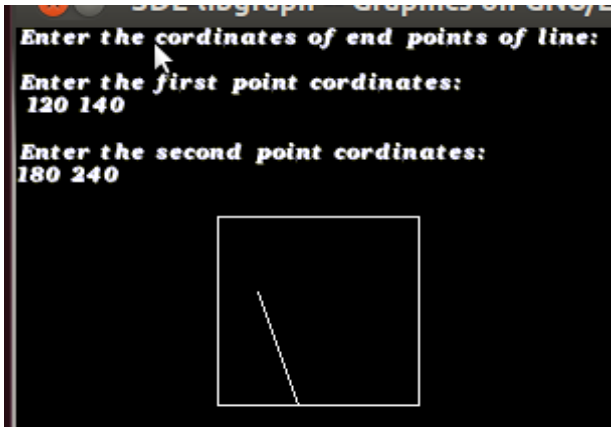

Fig. 4 Results after clipping

c. CASE III (Both endpoints are outside the rectangular window)

We have taken the coordinates of rectangular boundary i.eWmin (100,100) &Wmax(200,200)and coordinates of line end point i.e (x1,y1): (140,240) ,(x2,y2):(240,140)

Now we have checked the condition: if ((Xwmin<= X1<= Xwmax) && (Ywmin<=y1<= Ywmax)) && ((Xwmin<= X2<= Xwmax) && (Ywmin<=y2<= Ywmax));

i.e.

(100<140<200)&&(100<240<200)&&(100<240<200)&&(100<140<200)

this condition is false for above mentioned coordinates of line end point and line completely lies outside the rectangular boundry

now slope of line if ((X2!== X1)&&(Y2!==Y1)) m= (240 – 140) / (180-120) = 5/3

intersection to window (xmin, (m*xmin)+ c) ,(xmax `,(m*xmax)+c), ((ymin-c)/m, ymin), ((ymax-c)/m, ymax): (100,240), (200,140), (240,100),(140,200)

Points (140,200), (200,140) satisfying window condition

Now check is both point is between line point (x1,y1)and(x2,y2)

Line(140,200,240,100)

Before Clipping

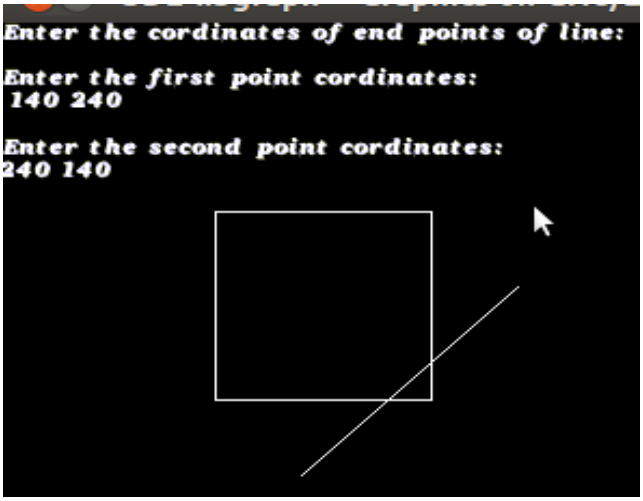

Fig. 5 Both end points are outside the window

After Clipping

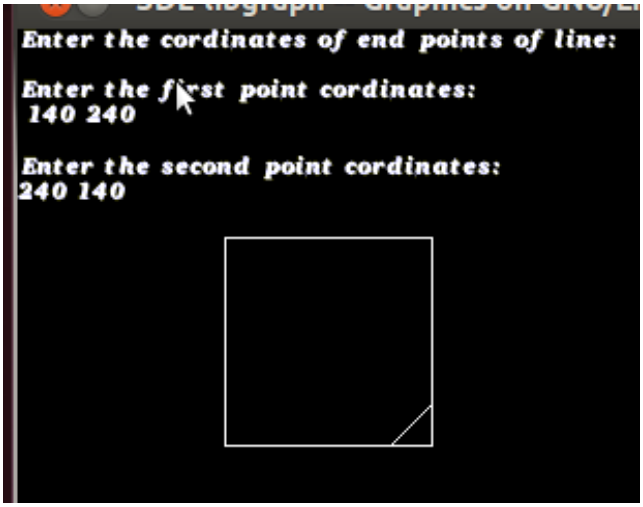

Fig. 6 Results after clipping

Comprehensive comparisons of the new algorithm with liangbarsky algorithm (in seconds)

Time costs for 500 times repeatedly clipping
New algorithm (s)8.23
Liang-Barsky algorithm (s) 7.31

IV. GRAPHICAL RESULT ANALYSIS

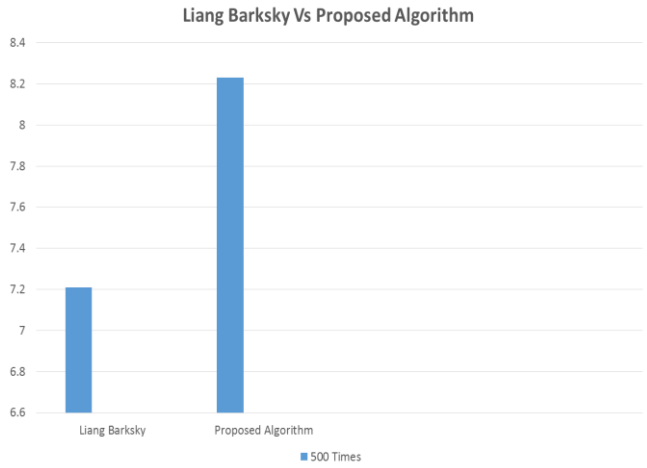

Fig. 7 Graphical Analysis

V. CONCLUSION & FUTURE WORK

In this paper we have introduced a new algorithm that is based on simple line equation. The proposed method is extremely fast for the usually occurring case of line segment which lies completely in the window having both endpoints outside the window, taking only constant time for execution. We can extend this algorithm for the clipping windows which even if they are not square. Moreover for increasing the accuracy of calculation we can use fuzzy.

REFERENCES

1. Cohen-Sutherland, Cohen-Sutherland Line Clipping algorithm ,Computer Graphics,C version ,2nd Ed.1997.11,Prentice Hall,Inc.Tsinghua university press ,p.226.
2. Cyrus,M.,Beck,J.:Generalized Two and Three dimensional clipping ,Computers Graphics ,Vol. 3,No. 1,P .23-28,1978.
3. You-dong Liang,BrianA.Barsky, Liang-Barsky Line Clipping algorithm,ComputerGraphics,C version ,2nd Ed.1997.11,Prentice Hall,Inc.Tsinghua university press .p230.
4. Nicholl ,Lee, Nicholl ,:Nicholl-Lee-Nicholl Line Clipping algorithm ,Computer Graphics,C version ,2nd Ed.1997.11,Prentice Hall,Inc.Tsinghua university press ,p233.
5. Sproull RF, Sutherland IE. A clipping divider. In: Proceedings of the Fall Joint Computer Conference. Washington: Thompson Books, 1968. p. 765–75.
6. Rogers DF. Procedural elements for computer graphics. New York: McGraw-Hill, 1985. p. 111–87.
7. Andreev R, Sofianska E. New algorithm for two-dimensional line clipping. Computers and Graphics 1991;15(4):519–26.
8. Nicholl TM, Lee DT, Nicholl RA. A efficient new algorithm for 2-D line clipping: its development and analysis. Computer Graphics 1987;21(4):253–62.
9. Shi KJ, Edwards JA, Cooper DC. An efficient line clipping algorithm. Computers and Graphics 1990;14(2):297–301.

10. Sobkow MS, Pospisil P, Yang YH. A fast two-dimensional line clipping algorithm via line encoding. Computers and Graphics 1987;11(4):459–67.
11. Sharma NC, Manohar S. Line clipping revisited: two efficient algorithm based on simple geometric observations. Computers and Graphics 1992;16(1):51–4.
12. Day JD. A new two dimensional ling clipping algorithm for small windows. Computer Graphics Forum 1992;11(4):241–5.
13. Wang Haohong, Wu Ruixun, CaiShijie. A new efficient clipping algorithm based on geometric transformation. Journal of Software 1998;9(10):728–33 (in Chinese).
14. Wang Jun, Liang Youdong, PengQunsheng. A 2-D line clipping with the least arithmetic operations. Chinese Journal of Computers 1991;(7):495–504 (in Chinese).