Comparative Analysis of Line Drawing Techniques in Computer Graphics

Jaspreet Singh[#1]

[#1]Assistant Professor, Department Of Computer Science & Engineering
Chandigarh University, Gharuan, Punjab, India
[1]jaspreet.rajal@gmail.com

Abstract— Computer graphics involves creating pictures, films on computers with creative programming. It involves various manipulations, transformations and mathematical operations on dataset. Line is considered as basic element in computer graphics during scan conversion. Scan conversion is the process in which object is represented by collection of discrete pixels. This feature can be implemented with the help of Digital Differentiation Analyzer (DDA) algorithm's line equations and Bresenham Algorithm. This paper performs comparative analysis of basic line drawing algorithm to conclude which algorithm is best .

Keywords—Bresenham, DDA, Pixel, Scan Conversion.

I. INTRODUCTION

A line is used to connect two points. It is considered as a primitive element in computer graphics. In order to draw a line, two points are required i.e starting point and end point. A line can be straight or any other desired shape. Straight line segments are very useful in block diagrams, bar charts, graphs, drawings, architecture plans etc.

The basic objective of any line drawing algorithm is to draw a satisfactory line with minimum computation and minimum possible time. It can be only be achieved by reducing the complex calculations.

Most of the algorithms involve floating point arithmetic which is time consuming than integer arithmetic. A line segment has many pixels and every pixel is calculated through computations. Thus, it is required to reduce complex calculations and a single arithmetic operation is important. If one computation per pixel is reduced, it will save many computations in generation of an object and it will reduce the time required to generate the complete image on the screen.

Computer graphics involves raster scan and random scan display devices. Raster scan devices display lines using pixel by pixel and Random scan display devices display a straight line smoothly from one point to another end point.

There are certain requirements that a good line drawing algorithms must follow:

1) A Line must appear as a straight line.
2) It should start and end at accurate position, matching end points with connecting lines.
3) Lines should has constant brightness.
4) Lines should be drawn as rapidly as possible.
5) Line drawing must be applicable for line with any slope.

LINE DRAWING ALGORITHMS

There are various line drawing algorithms.DDA and Bresenham algorithms for line drawing are the most common.

A. Digital Differential Analyzer (DDA)

DDA algorithm is an incremental scan conversion method. Digital Differential Analyzer (DDA) algorithm is considered as basic line generation algorithm.

It involves following steps:

Step 1: Input the line coordinates of two end points
i.e. X(X1, Y1) and Y(X2, Y2) for the line XY such that points X and Y are not equal. If they are equal then it is a point. Plot the point and exit.

Step 2: Find the difference between two end points of the line

i.e. $dx = X_2 - X_1$

$dy = Y_2 - Y_1$

Step 3: Based upon the difference in step-2, find the number of steps to generate a pixel
.i.e. Determine length of line 'L'

if abs(dx)>= abs(dy)
then L= abs(dx)
else L=abs(dy)

Step4: find the increments in terms of :x,y coordinates.

$X_{increment} = dx / L$

$Y_{increment} = dy / L$

This step makes either $X_{increment}$ or $Y_{increment}$ equal to 1.Thus,a step increment in X or Y direction is equal to 1.

Step5: Initialize the inital point on the line and plot

i.e. $X= X_1+0.5$

$Y= Y_1+0.5$

Value 0.5 is added to plot values in closest integer form by truncating fraction part.

Step 6: Compute the next coordinate position along the line

$X= X+ X_{increment}$

$Y=Y +Y_{increment}$

Step 7: Plot the values of x,y coordinates.

Step 8: Repeat steps 6 and 7 for L times.

Step9: Stop.

DDA algorithm is considered a simple algorithm and it does not requires any specialized skills for implementation.
DDA has certain disadvantages.

Disadvantages of DDA Algorithm

1. DDA is simplest algorithm but its floating point arithmetic is time-consuming in terms of compuataions.
2. The algorithm is orientation dependent. Hence end point accuracy is very poor.

B. Bresenham Line Drawing Algorithm

It is an incremental scan conversion algorithm developed by Bresenham.It uses only integer calculations. Fig. 1 shows the actual line location and nearest pixels on the raster screen.

D_A and D_B represent the distance between true line and plotted pixel.

Decision variable= $D_B - D_A$ Or $D_A - D_B$
If $D_B > D_A$, then
plotted pixel above the line will be closer to the true line.
If $D_B < D_A$, then
plotted pixel below the line will be closer to the true line.

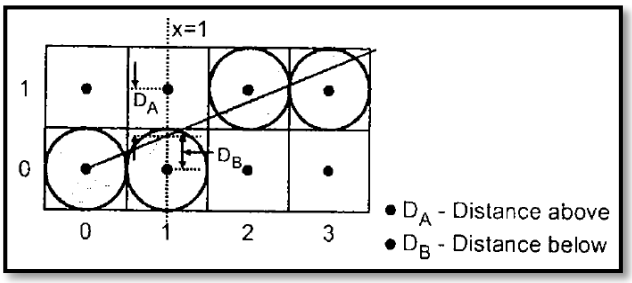

Figure 1: True Line and Pixel Representation

Step 1: Input the coordinates for two end points $X(X_1, Y_1)$ and $Y(X_2, Y_2)$ for the line XY such that points X and Y are not equal. If they are equal then it is a point. Plot the point and exit.

Step 2: Find the difference between two end points of the line

i.e. $dx = X_2 - X_1$

$dy = Y_2 - Y_1$

Step 3: Initialize starting points $X=X_1, Y=Y_1$

Step 4: Find the value of constants 2d*y*, (2d*y* – 2d*x*) and calculate the value of initial decision parameter

i.e.: $p_0 = 2dy - dx$

Step 5: For each point, x_k along the line, starting at k = 0, test the following conditions:

If $p_k < 0$,

$X_{next}=X_{k+1}=X_k+1$

$Y_{next}=Y_{k+1}=Y_k$

the next point to plot is (x_k+1, y_k)

and $p_{k+1}=p_k+2dy$

Otherwise, $P_k>=0$

then

$X_{next}=X_{k+1}=X_k+1$

$Y_{next}=Y_{k+1}=Y_k+1$

the next point to plot is (x_k+1, y_k+1)

$p_{k+1}=p_k+2dy-2dx$

Step 6: Display the point

Step 7: Repeat steps 5 and 6 until $X_{k+1}= X_2, Y_{k+1}=Y_2$

Step 8: Stop

Advantages of Bresenham Line Algorithm

1. This algorithm is very efficient since it use only incremental integer calculations. The computer can perform integer operations very rapidly.
2. By checking the sign of decision parameter a user can identify which is the better pixel to represent path of the line.

Limitations of Bresenham Line Algorithm

1. It will work only in case of value of $|m| < 1$.It is not applicable for negative line slopes.
2. It works only for first octant. It has to be modified for other octants.

C. Generalized Bresenham Line Algorithm

Basic Bresenham Line Algorithm is modified to make it applicable for all octants as well as for any type of slope.

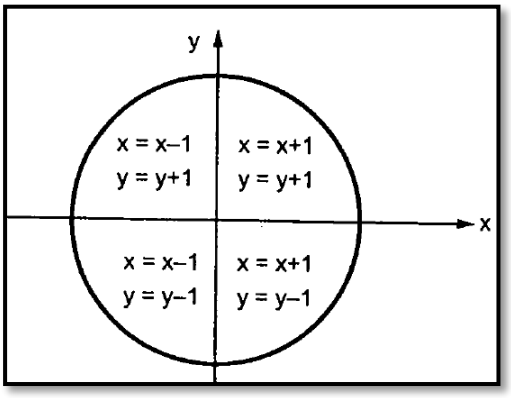

Figure 2: Conditions for Generalized Bresenham Line Algorithm

Step 1: Input the coordinates of two end points $X(X_1, Y_1)$ and $Y(X_2, Y_2)$ for the line XY such that points X and Y are not equal. If they are equal then it is a point. Plot the point and exit.

Step 2: Find the difference between two end points of the line

i.e. $dx = X_2 - X_1$

$dy = Y_2 - Y_1$

Step 3: Intialize starting points $X=X_1, Y=Y_1$

Step 4: $S_1=sign(X_2 - X_1)$

$S_2=sign(Y_2 - Y_1)$

sign is a function created by user that will return -1,0,1 as per argument values.

Step 5: If dy>dx
then interchange the values of dx and dy.
Set flag=1
else
Set flag=0
end if

Step 6: Calculate the constants 2d*y* and find the initial value of the decision parameter i.e. $p_0=2dy-dx$

*Step 7:*Inialize counter, k=1

*Step 8:*Plot the pixel values for X and Y.

Step 9: Test the conditions for parameter P_k
while($p_k \geq 0$)
{
if (flag=1)
then $X=X+ S_1$
else $Y=Y+ S_2$
end if
$p_k=p_k-2dx$
}

Step 10: If flag=1
then $Y= Y+ S_2$
else $X=X+ S_1$
end if
$p_k=p_k+2dy$

*Step 11:*k=k+1

Step 12: If ($k \leq dx$) then go to step8

*Step 13:*Stop

IMPLEMENTATION

All algorithms can be implemented in C++ by using in built graphics functions. Generalized Bresenham Algorithm can be implemented in C++ as follows:

```
//Bresenham Algorithm for positive and negative slope//
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main( )
{
int x,y,x1,x2,y1,y2,p,k,dx,dy;
int sign(int);
int sy,sx;
int gd=DETECT,gm;
initgraph(&gd,&gm," ");
cout<<"Enter co-ordinates of Starting point 1:";
cin>>x1>>y1;
cout<<"Enter co-ordinates of Ending point 2:";
cin>>x2>>y2;
sy=sign(y2-y1);
sx=sign(x2-x1);
dx=abs(x2-x1);

dy=abs(y2-y1);
p=(2*dy)-dx;
x=x1;
y=y1;
putpixel(x,y,YELLOW);
delay(100);
for(k=0;k<dx;k++)
{
      x=x+sx;
      if(p<0)
      {
      p=p+(2*dy) ;                  }
else
      {
      y=y+sy;
      p=p+2*(dy-dx);
      }
      putpixel(x,y,RED);
      delay(100);
}
```

```
        getch();
        closegraph();
}
        int sign(int s)
        {
        if(s<0)
        return -1;
        else if(s==0)
        return 0;
        else
        return 1;
        }
```

Outputs:

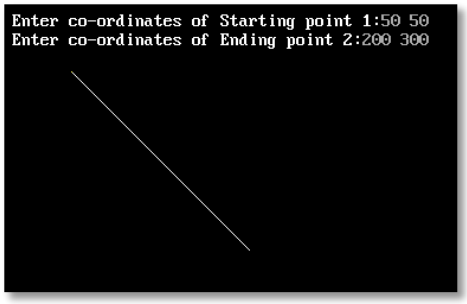

Figure 3: Line with positive slope

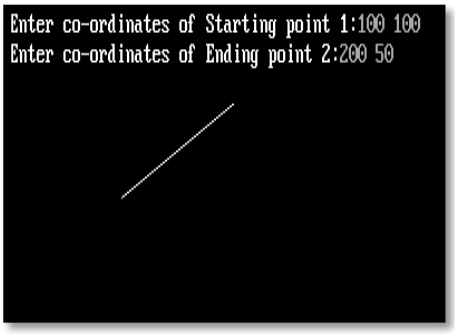

Figure 4: Line with negative slope

COMPARISON OF ALGORITHMS

The following table represents the comparison of line drawing algorithms on the basis of some key terms.

TABLE I: Comparison of Basic Line Algorithms

Key terms	**Digital Differential Analyzer (DDA)**	**Bresenham Line Drawing Algorithm**
Arithmetic Calculations	It uses floating point's values.	It uses only integer arithmetic.
Basic Mode of Operations	multiplication and division for calculations.	subtraction and addition for calculations.
Working Speed	Slow than Bresenhams due to floating point calculations.	Fast than DDA due to integer calculations..
Accuracy & Efficiency	Not accurate and efficient as compared to Bresenham	More efficient and much accurate than DDA
Scope of Drawing	circles and curves can be drawn but not accurate as Bresenham	Circles, curves and other shapes with accuracy than DDA
Round Off	Rounding off the values to integer that is nearest to the true line.	There is no rounding off the values. Incremental Method is used.

CONCLUSION

Line drawing is a basic approach in computer graphics.DDA involves successive addition of floating point increments which causes accumulation of round off error and there may be drift away of the plotted pixels from true line path specially in long line segments. Bresenham is more accurate and efficient as compared to DDA algorithm because it clearly avoids the round function and scan conversion of lines is done using incremental methods of integer calculations. It is also applicable for all types of slopes however it is useful as a driving engine for many other graphics routines.

REFERENCES

[1] http: // en.wikipedia.org/wiki/ Bresenham's_line_algorithm".
[2] Muhammad Usman Khan, Md. Rizwan Beg, Mohd Zunnun Khan. Improved Line Drawing Algorithm: An Approach and Proposal. Int. Conf. on Advances in Computer Science and Electronics Engineering 323-327.
[3] "http://i.thiyagaraaj.com/articles/articles/dda-line-algorithm.
[4] Angel, E., and Morrison, D. Speeding up Bresenham's Algorithm. IEEE Computer Graphics and Application 1991;11(6):16-17.
[5] Donald Hearn & Baker,Computer Graphics C version, Pearson Second Edition.
[6] Mangal Priyanka,Bresenham's Line Drawing Is An Efficient Apporach Than The Dda, Asian Journal of Computer Science Engineering 2015;1(2):1-3
[7] Kenneth I. Joy ,Breshenham's Algorithm, On-Line Computer Graphics Notes, University of California, Davis.
[8] J. Voley et al, Computer Graphics Principles and Practices. New York: Addison Wesley, 1996.

[9] J. D. Foley, et al., “Computer Graphics, Principles and Practice”, 2nd Edn. Addison-Wesley, (1990).
[10] G. Bhatnager, S. Metha, and S. Mitra, Introduction to Multimedia Systems. London: Academic Press, 2001.