IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 9, September 2017

# Efficient Plotting Algorithm

**Sushant Ipte[1], Riddhi Agarwal[1], Murtuza Barodawala[1], Ravindra Gupta[1], Prof. Shiburaj Pappu[1]**

Computer Department, Rizvi College of Engineering, Mumbai, Maharashtra, India[1]

**Abstract:** This article describes various plotting algorithm and hence concludes the efficient algorithm which can be used to speed up the plotting process. Various graphical algorithms will be analysed and the best algorithm will speed up the plotting process adopted by various plotter. We will also have a look on the advantages and disadvantages of all the algorithms.

**Keywords:** Computer Graphics, Plotting, Digital Differential Analyzer, Bresenham.

## I. INTRODUCTION

There are various algorithm which can be used for plotting such as Line drawing algorithm, Digital Differential Analyzer Line algorithm (DDA), Midpoint Circle algorithm, Midpoint Line algorithm and Brenham's algorithm individually and find out the most efficient plotting algorithm which can be used by plotters and also we would take a glance at the properties, its applications, advantages and disadvantages in details. These algorithms are the base for any plotters which are available in the market worldwide. Let us study about plotter and its basic plotting techniques.

## II. LINE DRAWING ALGORITHM

Line Drawing algorithm is the algorithm which is used by the plotters and many other computer graphics. A line connects two points. It is a basic element in graph. A line drawing algorithm is a graphical algorithm for approximating a line segment on a discrete graphical media. To draw a line we need two points between which you can draw a line. On continuous media, by contrast, no algorithm is necessary to draw a line. For example, oscilloscopes use natural phenomena to draw line and curves.

### A. Basic Line Drawing Algorithm

The simplest method to represent a natural line drawing algorithm is the

```
dx = x2 - x1
dy = y2 – y1
for x from x1 to x2 {
 y = y1 + dy * (x – x1)/dx
Plot(x, y)  }
```

It is here that the points have already been ordered so that x2 > x1. This algorithm works just fine when dx >= dy (i.e., slope is less than or equal to 1), but if dx < dy (i.e., slope greater than 1), the line becomes quite sparse with lots of gaps, and in the limiting case of dx = 0, only a single point is plotted.

### B. Digital Differential Analyser Line Algorithm (DDA)

A digital differential analyzer (DDA) is hardware or software used for interpolation of variables over an interval between start and end point. DDAs are used for rasterization of lines, triangles and polygons. They can be extended to nonlinear functions, such as perspective correct texture mapping, quadratic curves, and traversing vowels. The DDA method can be implemented using floating-point or integer arithmetic.

The indigenous floating-point practice requires 1 addition & 1 rounding operation per interpolated value and output result. This process is only dexterous when an FPU with fast add and rounding operation will be gettable.

In its simplest implementation for linear cases such as lines, the DDA algorithm interpolates values in interval by computing for each xi the equations xi = xi−1+1/m, yi = yi−1 + m,

Where Δx = x end − x start

And Δy = y end – y start

And m = Δy/Δx.

Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step below.

**Step 1** − Get the input of two end points from the user (X0,Y0) and (X1,Y1)

**Step 2** – Difference between two points is calculated as,

dx = X1 – X0

```
dy = Y1 – Y0
```

**Step 3**– According to the calculated difference in step-2, we will identify the number of steps that are placed in the pixel. If dx > dy, then we need more steps in x coordinate; than in y coordinate.

```
if (absolute (dx) >absolute (dy))
Steps = absolute(dx);
else
Steps = absolute (dy);
```

**Step 4** – Calculating the increments of x coordinate and y coordinate

```
X increment = dx / (float) steps;
Y increment = dy / (float) steps;
```

**Step 5** − Incrementing x and y coordinates by successfully putting the pixel and completes the drawing of the line.

```
For(int v=0; v < Steps; v++)
{
x = x + X increment;
y = y + Y increment;
Putpixel (Round(x), Round(y));
}
```

## C. Midpoint Line algorithm

The midpoint line algorithm is a line incremental line plotting algorithm in which at each step one can make incremental calculation based on the preceding step to find next value of variable y, which will lead to the formation of an approximate straight line using two points.
Let us understand by taking a paradigm,
Given coordinate of two points A(x1, y1) and B(x2, y2) such that x1 < x2 and y1 < y2. The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.
For any given or calculated previous pixel P(Xp, Yp), there has to be two candidates for the next pixel which are closest to the line, E (Xp+1, Yp) and NE (Xp+1, Yp+1) (E stands for East and NE stands for North-East).
In Mid-Point algorithm we will do the following steps:-

1. Find the midpoint of two points. Middle of E(Xp+1, Yp) and NE (Xp+1, Yp+1) is M (Xp+1, Yp+1/2).
2. If M is above the line, then choose E as next point.
3. If M is below the line, then choose NE as next point.

The main advantages of mid-point line algorithm can be listed below,

- This algorithm determines the closet pixel to the line with definiteness, undeviated and with exact accuracy.
- This algorithm is very simple to understand and only requires integer data and simple mathematical operations
- To eliminate error this algorithm does not use division and multiplication operations.

Let us discuss a few basic steps of Midpoint Line algorithm

```
Step 1:- take input from the user as (x0,y0) and (x1,y1).
Step 2:- Now calculate the value of dx and dy with the help of two points.
Step 3:- d=dx-(dy/2)
Step 4:- set x=x0 and y=y0
Step 5:- plot the point(x,y)
Step 6:- Declare a while loop with a condition (y<y1) and in that loop do y = y+1
Step 7:- In the while loop we will make if condition of d<0 and if yes do d= d+dx.
Step 8:- if not do d=d+dx-dy and x=x+1.
Step 9:- plot the point(x, y).
```

## D. Bresenham's Line Drawing Algorithm.

Bresenham's Line Drawing algorithm is a type of algorithm which determines the points of n-dimension ratherthat should be selected in sequence to form a close approximate to straight line between two points. It's an accrual error algorithm. This algorithm is commonly used to draw primitives in a bit map image as it only uses basic mathematical operation like integer addition, subtraction and bit shifting and some are also be found in many software graphics libraries. Because the algorithm is very lucid it is often applied either the firmware or the graphics hardware of modern graphics cards.
Given coordinate of two points a(x1, y1) and B(x2, y2). The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.
Below are some assumptions to keep algorithm

We draw line from left to right.
1. x1 < x2 and y1< y2
2. Slope of the line is between 0 and 1. We draw a line from lower left to upper right.

The Brenham's algorithm can be explained as slightly modified digital differential analyser

The principle of using an incremental error in place of division operations has other applications in graphics. It is possible to use this technique to calculate the U, V co-ordinates during raster scan of texture mapped polygons. The voxel height map software-rendering engines seen in some PC games also used this principle.Bresenham's also published a Run-Slice (as opposed to the Run-Length) totalling algorithm. There is an extension to the algorithm that handles thick lines, which is as follows:-

Algorithm of Bresenham's line drawing algorithm

Step 1:- the start point of the line is f(x0, y0)

Step 2:- If the slope of the line is less than zero then the next point would be (x0+1,y0) and if the slope of the line is equal to zero then the next point would be (x0+1,y0+1).

Step 3:- the next point should be chosen on the basis of the line of x0+1

Step 4:- If the line is closer to the x line then plot it closer to x line or else plot it near y line

Step 5:- Evaluating the line function at the midpoint of these two points,
F(x0+1, y0+1/2)

Step 6:- If the value from the Step 5 is positive then the ideal line would be below the midpoint and closer to (x0+1, y0+1).

Step 7:- If the value from the Step 5 is not positive then the ideal line has not advanced to the midpoint and y coordinate and this line is closer to (x0+1, y0).

## III.CIRCLE DRAWING ALGORITHM

To draw a circle on the screen is little bit complex than drawing line on the screen. In Circle Drawing algorithm we have two basic and efficient methods, Bresenham's circle drawing algorithm and Midpoint circle drawing algorithm. We also have Midpoint ellipse generation which can also be used as an algorithm for plotters for plotting various designs.

### A. Midpoint Circle Drawing Algorithm

The midpoint circle drawing algorithm is an algorithm which needs points to rasterize a circle. Brenham's circle drawing algorithm is derived from the midpoint circle drawing algorithm. This algorithm can be used to draw eight octants simultaneously each with a cardinal angle and extends both way to the nearest multiple of 45 degrees. When it x=y then this algorithm determines to stop and it has reached 45 degree. As y increases, it does not skip nor repeat any y value until reaching 45°.In loop iteration y increments by 1, and x decrements by 1, it never exceeds more than 1 in a single iteration. The main objective of this algorithm is to find a path of the pixel grid using the formula $x^2 + y^2 = r^2$.

Let us discuss a few basic steps of Midpoint Circle algorithm,

Step 1: Get the input of the coordinates for the centre of the circle and radius of the circle and store them as x, y and R respectively. Initialize two new variables p and q and set them as P = 0 and Q = R.

Step 2: Input the initial value of the decision parameter as
D = 5/4 – R.

Step 3: Repeat through step 8, while x<y.

Step 4: Call the function of drawing the circle (X, Y, P and Q).

Step 5: Increment the value of P.

Step 6: If D<0 then D = D + 4P + 6

Step 7: Else set Y = Y – 1 and D = D+ 4(P-Q) + 10.

Step 8: Call draw circle (X, Y, P and Q).

### B. Bresenham's Circle Drawing Algorithm

Bresenham's algorithm is derived from mid-point algorithm which is used to determine the points used for rasterizing. This algorithm is called incremental, because the position of the next pixel is calculated on the basis of the last plotted one, instead of just calculating the pixels from a global formula. Such logic is faster for computers to work on and allows plotting circles without trigonometry. The algorithm use only integers, and that's where the strength is: floating point calculations slow down the processors.

This algorithm draws all eight octants simultaneously, starting from each cardinal direction (0°, 90°, 180°, 270°) and extends both ways to reach the nearest multiple of 45° (45°, 135°, 225°, 315°). It can determine where to stop because when y = x, it has reached 45°. The reason for using these angles is shown in the above picture: As y increases, it does

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 9, September 2017

not skip nor repeat any y value until reaching 45°. So during the while loop, y increments by 1 each iteration, and x decrements by 1 on occasion, never exceeding 1 in one iteration. This changes at 45° because that is the point where the tangent is rise=run. Whereas rise>run before and rise<run after.
The second part of the problem, the determinant, is far trickier. This determines when to decrement x. It usually comes after drawing the pixels in each iteration, because it never goes below the radius on the first pixel. Because in a continuous function, the function for a sphere is the function for a circle with the radius dependent on z (or whatever the third variable is), it stands to reason that the algorithm for a discrete(voxel) sphere would also rely on this Midpoint circle algorithm. But when looking at a sphere, the integer radius of some adjacent circles is the same, but it is not expected to have the same exact circle adjacent to itself in the same hemisphere. Instead, a circle of the same radius needs a different determinant, to allow the curve to come in slightly closer to the center or extend out farther.
As circle is of 360 and symmetrical to x-axis the calculation along it is not done next, we see that it's also symmetrical about the y axis, so now we only need to calculate the first 90 degrees. Finally, we see that the circle is also symmetrical about the 45 degree diagonal axis, so we only need to calculate the first 45 degrees. Bresenham's circle algorithm calculates the locations of the pixels in the first 45 degrees. It assumes that the circle is centered on the origin shifting the original center coordinates (centerx, centery). So for every pixel (x, y) it calculates, we draw a pixel in each of the 8 octants of the circle:
putpixel (centerx + x, center y + y)
putpixel(centerx + x, center y - y)
putpixel(centerx - x, center y + y)
putpixel(centerx - x, center y - y)
putpixel(centerx + y, center y + x)
putpixel(centerx + y, center y - x)
putpixel(centerx - y, center y + x)
putpixel(centerx - y, center y - x)

Now, consider a very small continuous arc of the circle interpolated below, passing by the discrete pixels as shown
As can be easily intercepted, the continuous arc of the circle cannot be plotted on a raster display device, but has to be approximated by choosing the pixels to be highlighted. At any point (x, y), we have two choices – to choose the pixel on east of it, i.e. N(x+1, y) or the south-east pixel S(x+1, y-1). To choose the pixel, we determine the errors involved with both N & S which are f (N) and f(S) respectively and whichever gives the lesser error, we choose that pixel. Let di = f (N) + f(S), where d can be called as "decision parameter", so that if di<=0, then, N(x+1,y) is to be chosen as next pixel i.e. xi+1 = xi+1 and yi+1 = yi, and if di>0, then, S(x+1,y-1) is to be chosen as next pixel i.e. xi+1 = xi+1 and yi+1 = yi-1. We know that for a circle, x 2 + y2 = r2, where r represents the radius of the circle, an input to the algorithm. Errors can be represented as f (N) = (xi + 1)2 + yi 2 - r 2, - (1) f(S) = (xi + 1)2 + (yi - 1)2 - r 2 - (2) as di = f (N) + f(S), _di = 2(xi+1)2 + yi 2 + (yi-1)2 – 2r2 - (3)
Calculating next decision parameter,
Di+1 = 2(xi+2)2 + yi+1 2 + (yi+1-1)2 – 2r2 - (4)
From (4)- (3), we get,
di+1 – di = 2((xi+2)2 -(xi+1)2 ) + (yi+1 2 – yi 2 ) + ((yi+1-1)2 + (yi-1)2 ) _di+1 = di + 2((xi+2+xi+1)(xi+2-xi-1)) + ((yi+1+yi)(yi+1-yi)) + ((yi+1-1+yi-1)(yi+1-1-yi+1))
Di+1 = di + 2(2xi+3) + ((yi+1+yi)(yi+1-yi)) + ((yi+1-1+yi-1)(yi+1-1-yi+1))
Now, if (di<=0), xi+1=xi+1 and yi+1=yi so that
Di+1 = di + 2(2xi + 3) + ((yi+1+yi)( yi-yi)) + ((yi-1+yi-1)(yi-1-yi+1)) _ di+1 = di + 2(2xi + 3) + ((yi+1+yi)(0)) + ((yi-1+yi-1)(0))
_ Di+1 = di + 4xi + 6 else di+1 = di + 2(2xi+3) + ((yi-1+yi) (yi-1-yi)) + ((yi-2+yi-1) (yi-2-yi+1)) _ di+1 = di + 4xi+6 + ((2yi-1) (-1)) + ((2yi-3)(-1)
Di+1 = di + 4xi+6 - 2yi - 2yi + 1 + 3 _ di+1 = di + 4(xi - yi) + 10 to know di+1, we have to know di first. The initial value of di can be obtained by replacing x=0 and y=r in (3). Thus, we get, do = 2 + r2 + (r - 1)2 -2r2 _do = 2 + r2 + r2 + 1 -2r – 2r2 _ do = 3 – 2r

## CONCLUSION

Here we will discuss about the advantages and disadvantages of all the mentioned algorithms in this paper and will come to a decision on an algorithm which is the most efficient algorithm from them and can be used as a plotter algorithm.

First we will discuss the advantages and disadvantages of the DDA algorithm and then all the other algorithm thereafter according to occurrence of the algorithm in this paper

1. DDA
Advantages of DDA algorithm;
a) Faster than the direct use of line equation and it does not need any floating point multiplication.

Disadvantages of DDA algorithm;
a) Floating point Addition is still needed
b) Precession loss is possible because of rounding of the points.
c) The algorithm is orientation dependent.
d) Rounding-off in DDA is time consuming.

2. Midpoint Line Drawing
Advantages of Midpoint Line Drawing algorithm;
a) High Accuracy
b) High Speed
c) Draws the line between any two points

Disadvantages of Midpoint Line Drawing algorithm;
a) Time Consumption is high
b) The distance between the pixel is not equal, so we don't get a smooth line

3. Bresenham's Line Drawing
Advantages of Bresenham's Line Drawing algorithm;
a) It is a fast incremental algorithm.
b) It is fast because it uses arithmetic operation for precision marking of pixels.
c) It is less time consuming then other circle drawing algorithm
d) It uses only integer calculation such as addition, subtraction and bit shifting.

Disadvantages of Bresenham's Line Drawing algorithm;
a) This algorithm is only meant for basic line drawing algorithm.
b) The advance topic of anti-aliasing is not the part of the Bresenham's line drawing algorithm, so drawing a drawing a smooth line, one needs to find a different algorithm.

4. Midpoint Circle Drawing
Advantages of Midpoint Circle Drawing algorithm;
a) This algorithm is efficient for scan conversion for drawing geometric curves on raster display.

Disadvantages of Midpoint Circle Drawing algorithm;
a) It is time consuming
b) Distance between the pixels is not equal so we won't a smooth circle.

5. Bresenham's Circle Drawing
Advantages of Bresenham's Circle Drawing algorithm;
a) It is less time consuming then other circle drawing algorithm
b) It is fast because it uses arithmetic operation for precision marking of pixels

Disadvantages of Bresenham's Circle Drawing Algorithm;
a) The advance topic of anti-aliasing is not the part of the Bresenham's circle drawing algorithm, so drawing a drawing a smooth circle.

After understanding all the algorithm and their various techniques, we have concluded that Bresenham's Line Drawing Algorithm is the best algorithm which is to be used for plotters.

## REFERENCES

[1] Improved DDA line drawing anti-aliasing algorithm based on embedded graphics system. INSPEC Accession Number: 11537581. DOI: 10.1109/ICACTE.2010.5579418
[2] A Novel Method for Projection Based on Bresenham like Algorithm. INSPEC Accession Number: 12657144. DOI: 10.1109/ISdea.2012.623.
[3] Parallelization of Bresenham's line and circle algorithms. INSPEC Accession Number: 3790564. DOI: 10.1109/38.59038
[4] An embedded real-time fish-eye distortion correction method based on Midpoint Circle Algorithm. INSPEC Accession Number: 11851521. DOI: 10.1109/ICCE.2011.5722533