

**MOI UNIVERSITY**

**SCHOOL OF BIOLOGICAL AND PHYSICAL SCIENCES**

***(Department of statistics and computer science)***

**NAME**: BIWOTT K. ANDREW

**REG NO.:** COM/018/12

**COURSE CODE**: COM 420

**COURSE TITLE:** COMPUTER GRAPHICS II

**TASK:** CAT 1

**LECTURER:** MRS.MILSORT KEMBOI

**SIGNATURE: K.B.A**

**DATE: 18TH OCTOBER 2015**

**DISCUSS DDA AND BRESENHAM’S ALGORITHMS USED IN GRAPHICS**

* **DDA algorithm** - [**Digital Differential Analyzer**](https://en.wikipedia.org/wiki/Digital_differential_analyzer) (**DDA**) is hardware or software used for [linear interpolation](https://en.wikipedia.org/wiki/Linear_interpolation) of [variables](https://en.wikipedia.org/wiki/Variable_%28computer_science%29) over an [interval](https://en.wikipedia.org/wiki/Interval_%28mathematics%29) between start and end point. DDAs are used for [rasterization](https://en.wikipedia.org/wiki/Rasterization) of lines, triangles and polygons.
* DDA algorithm used for drawing lines on the discrete plane, such as computer monitor, it is one of the fundamental algorithms in computer graphics.DDA algorithm is an incremental scan conversion method. In this algorithm, we perform calculations at each step using the results from the preceding step. The characteristic of the DDA algorithm is to take unit steps along one coordinate and compute the corresponding values along the other coordinate.
* The unit steps are always along the coordinate of greatest change, e.g. if dx = 10 and dy = 5, then we would take unit steps along x and compute the steps along y.

**Example:**

If the end points of the line is given as (x1,y1)= (2,2) and (x2, y2)= (9,5). Here we will calculate y2-y1 and x2-x1 to find which one is greater. Here y2-y1 =3 and x2-x1 =7;

Therefore here the major axis is the x axis. So here we need to sample the x axis at unit intervals i.e.? x = 1 and we will find out the y values for each x,in the x axis using the slope equation.

**In DDA we need to consider two cases:**

1. **When Slope of the line less than or equal to one (|m| ? 1).**

**When |m| <=1**, means y2-y1 = x2-x1 or y2-y1 <x2-x1.

In both these cases we assume x to be the major axis. Therefore we sample x axis at unit intervals and find the y values corresponding to each x value.

We have the slope equation as ? y = m ? x   y2-y1 = m (x2-x1) In general terms we can say that y i+1 - yi = m(x i+1 - xi ). But here ? x = 1; therefore the equation reduces to y i+1= yi + m = yi + dy/dx.

1. **Slope of the line greater than one (m| > 1).**

**When m| > 1,** means y2-y1> x2-x1 and therefore we assume y to be the major axis. Here we sample y axis at unit intervals and find the x values corresponding to each y value.

We have the slope equation as:

? y = m ? x

y2-y1 = m (x2-x1)

In general terms we can say that y i+1 - yi = m(x i+1 - xi ). But here ? y = 1; therefore the equation reduces to 1 = m(x i+1 - xi).

Therefore : x i+1=xi+ 1/mx i+1=xi+ dx/dy

**DDA Algorithm is given below:**

procedure DDA( x1, y1, x2, y2: integer);

var

            dx, dy, steps: integer;

            x\_inc, y\_inc, x, y: real;

begin

            dx := x2 - x1; dy := y2 - y1;

            if abs(dx) > abs(dy) then

            steps := abs(dx); {steps is larger of dx, dy}

else

            steps := abs(dy);

            x\_inc := dx/steps; y\_inc := dy/steps;

            {either x\_inc or y\_inc = 1.0, the other is the slope}

            x:=x1; y:=y1;

            set\_pixel(round(x), round(y));

            for i := 1 to steps do

begin

             x := x + x\_inc;

             y := y + y\_inc;

              set\_pixel(round(x), round(y));

end;

end; {DDA}

**TYPICAL EXAMPLE OF DDA ALGORITHM:**

***Consider the line from (0,* 0) *to* (-6, -6). *Use the simple* DDA *algorithm line.***

Sol. Evaluating steps 1 to 5 in the DDA algorithm we have

Xl = 0                          Y1 = 0

X2 = - 6                       Y2 = - 6

Length = l X2-X1l |Y2-Y1l = 6

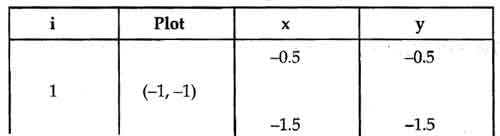
                           ∆X = ∆Y = -1

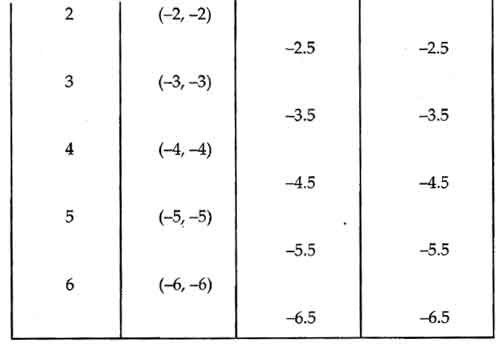
Initial values for

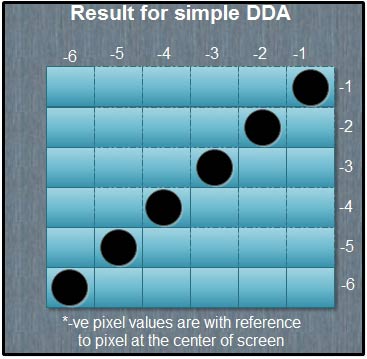
                                       X = 0 + 0.5 \* Sign (-1) =-0.5

                                       Y = 0 + 0.5 \* Sign (-1) =-0.5

Tabulating the results of each iteration in the step 6 we get,

[](http://ecomputernotes.com/images/DDA-Algorithm-Line.jpg)

[](http://ecomputernotes.com/images/DDA-Algorithm-Line2.jpg)

[](http://ecomputernotes.com/images/Result-for-simple-DDA.jpg)

The results are plotted as shown in the Fig above. It shows that the rasterized line lies on the actual line and it is 45° line.

**Advantages of DDA Algorithm**

1. It is the simplest algorithm and it does not require special skills for implementation.
2. It is a faster method for calculating pixel positions than the direct use of equation

y = mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or v direction to find the pixel positions along the line path.

**Disadvantages of DDA Algorithm**

1. Floating point arithmetic in DDA algorithm is still time-consuming.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.

**Bresenham’s Algorithm** is the basic ”line drawing” algorithm used in computer graphics. This algorithm was developed to draw lines on digital plotters, but has found wide-spread usage in computer graphics.

Unlike DDA Algorithm, this algorithm is fast – it can be implemented with integer calculations only – and very simple to describe.

This algorithm also provides the means for the fast and efficient way to represent continuous abstract lines onto discrete plane of computer display through process called **rasterization.**

**Description:**

Consider a line with initial point (x1, y1) and terminal point (x2, y2) in device space.

If \_x = x2 −x1 and \_y = y2 − y1, we define the *driving axis* (*DA*) to be the x-axis if |\_x| ≥ |\_y|, and the y-axis if |\_y| > |\_x|.

The *DA* is used as the “axis of control” for the algorithm and is the axis of maximum movement. Within the main loop of the algorithm, the coordinate corresponding to the *DA* is incremented by one unit. The coordinate corresponding to the other axis (usually denoted the *passive axis* or *PA*) is only incremented as needed.

**Bresenham’s Algorithm**

The points (x1, y1) and (x2, y2) are assumed

not equal and integer valued.

\_ is assumed to be real.

**Let** \_x = x2 − x1

**Let** \_y = y2 − y1

**Let** m = \_y

x

**Let** j = y1

**Let** c = m − 1

**for** i = x1 to x2 − 1

**illuminate** (i, j)

**if** (c >= 0)

j + = 1

c − = 1.0

**end if**

i + = 1

c+ = m

**next** i

**finish .**

**Other Algorithms for Breseham’s are:**

1. **Bresenham’s Algorithm using Integer Arithmetic**

Bresenham’s Algorithm, as given in the previous section, requires the use of floating point arithmetic to

calculate the slope of the line and to evaluate the error term.

The points (x1, y1) and (x2, y2) are assumed

not equal and integer valued.

¯c is assumed to be integer valued.

**Let** \_x = x2 − x1

**Let** \_y = y2 − y1

**Let** j = y1

**Let**  c = \_y − \_x

**for** i = x1 to x2 − 1

**illuminate** (i, j)

**if** (c>=0)

j + = 1

c− = \_x

**end if**

i + = 1

c + = \_y

**next** i

**finish.**

1. **Bresenham’s Algorithm for Lines with Arbitrary Endpoints :**

The points (x1, y1) and (x2, y2) are assumed not equal and have arbitrary real coordinates c is assumed to be real.

**Let** \_x = x2 − x1

**Let** \_y = y2 − y1

**Let** m = \_y

x

**Let** i1 = bx1c

**Let** j = by1c

**Let** i2 = bx2c

**Let** c= −(1 − (y1 − j) − \_y (1−(x1−i1))

x

**for** i = i1 to i2

**illuminate** (i, j)

**if** (>=0)

j + = 1

c − = 1.0

**end if**

i + = 1

c\_ + = m

**next** i; **finish.**

**Advantages For Bresenham’s Algorithm**

1. Fast incremental algorithm.
2. Only integer calculation.
3. Does not have rounding and floating point operation.
4. Provide high speed.

**Disadvantages of Bresenham’s Algorithm:**

* The disadvantage of this simple algorithm is that it is meant for basic line drawing. The "advanced" topic of anti aliasing isn't part of Bresenham's algorithm, so to draw smooth lines, you'd want to look into a different algorithm but this algorithm.

**Difference between DDA and Bresenham Algorithm.**

1. DDA uses floating points where as Bresenham algorithm use fixed points.
2. DDA round off the coordinates to nearest integer but Bresenham algorithm does not.
3. Bresenham algorithm is much accurate and efficient than DDA.
4. Bresenham algorithm can draw circles and curves with much more accuracy than DDA.
5. DDA uses multiplication and division of equation but Bresenham algorithm uses subtraction and addition only.

**REFERENCES**

1. William M. Newman, Robert F. Sproull, (2000), *Principles of Interactive Computer Graphics*, Tata-McGraw Hill,.
2. Donald Hearn, M. Pauline Baker( 2007) ,*Computer Graphics – C Version*, Pearson Education,
3. ISRD Group( 2006), *Computer Graphics*, McGraw Hill,.
4. .J.D. Foley, A. Dam, S.K. Feiner, J.F. Hughes (1997),*Computer Graphics – principles and* *practice*, Addison-Wesley,
5. [www.answers](http://www.answers) .com.