Assignment No: 01

Title/ Problem Statement:

Write C++/Java program to draw line using DDA and Bresenham‘s algorithm. Inherit pixel class and Use function overloading.

Prerequisites: Coordinates System, Mathematics

Objectives: To learn use of slope to draw line.

Theory :

DDA (Digital Differential Analyzer):

Scan conversion line algorithm based on Δ x or Δ y. Sample the line at unit interval in one co-ordinate and determine corresponding integer value. Faster method than y= mx+ b using but may specifies inaccurate pixel section. Rounding off operations and floating point arithmetic operations are time consuming.DDA is used in drawing straight line to form a line, triangle or polygon in computer graphics. DDA analyzes samples along the line at regular interval of one coordinate as the integer and for the other coordinate it rounds off the integer that is nearest to the line. Therefore as the line progresses it scan the first integer coordinate and round the second to nearest integer. Therefore a line drawn using DDA for x coordinate it will be x0 to x1 but for y coordinate it will be y=ax+ b and to draw function it will be fn(x, y rounded off).

i	Plot	x	y	e
		5	5	0
1	(5, 5)	6	6	-8
2	(6, 6)	7	6	0
3	(7, 6)	8	7	-8
4	(8, 7)	9	7	0
5	(9, 7)	10	8	-8
6	(10, 8)	11	8	0
7	(11, 8)	12	9	-8
8	(12, 9)	13	9	

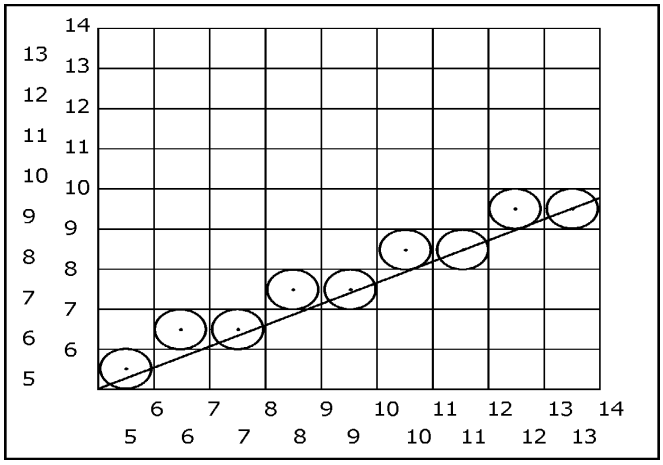

Fig DDA Line Algorithm

Advantages & disadvantages of DDA:-

• Faster than the direct use of the line equation and it does not do any floating point multiplication.

• Floating point Addition is still needed.

• Precision loss because of rounding off.

• Pixels drift farther apart if line is relatively larger.

Bresenham line algorithm:

This was developed by J.E.Bresenham in 1962 and it is much accurate and much more efficient than DDA. An algorithm which determines which order to form a close approximation to a straight line between two given points. It is commonly used to draw lines on a computer screen, as it uses only integer addition, subtraction and bit shifting, all of which are very cheap operations in standard computer architectures. It scans the coordinates but instead of rounding them off it takes the incremental value in account by adding or subtracting and therefore can be used for drawing circle and curves. Therefore if a line is to be drawn between two points x and y then next coordinates will be(xa+1, ya) and (xa+1, ya+1) where a is the incremental value of the next coordinates and difference between these two will be calculated by subtracting or adding the equations formed by them.

	i Plot	x	y	e
		2	5	2
1.	(2, 5)	3	6	-4
2.	(3, 6)	4	6	6
3.	(4, 6)	5	7	4
4.	(5, 7)	6	8	2
5.	(6, 7)	7	8	+8
6.	(7, 8)	8	9	2
7.	(8, 9)	9	10	-4
8.	(9, 10)	10	10	

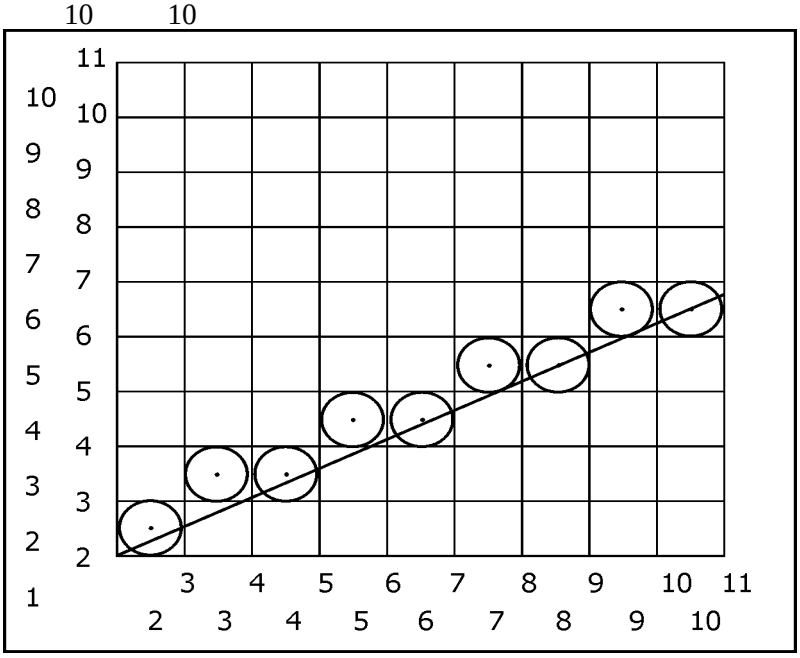

Fig Bresenham'sLine Algorithm

Algorithm :

DDA LINE DRAWING ALGORITHM

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also
   declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer;that is,plot the first point. put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs(dx) > abs(dy), then steps=abs(dx).
8. Otherwise steps= abs(dy).
9. Then xi=dx/steps and yi=dy/steps.
10. Start from k=0 and continuing till k<steps, the points will be
    i. x=x+xi.
    ii. y=y+yi.
11. Plot pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.

BRESENHAM'S LINE DRAWING ALGORITHM WITH SLOPE < 1

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,p,dx,dy and also declare gdriver=DETECT,gmode.
3. Initialize the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer; that is, plot the first point put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1,and obtain the initial value of decision parameter p
   as: a. p=(2dy-dx).
7. Starting from first point (x,y) perform the following test:
8. Repeat step 9 while(x<=x2).
9. If p<0, then

   next point is (x+1,y)
   and calculate p=(p+2dy).

   Otherwise,
   the next point to plot is (x+1,y+1)

and calculate p=(p+2dy-2dx).

10. Plot pixels using putpixel at points (x,y) in specified colour.
11. Close Graph.
12. Stop.

Write GENERALIZED BRESENHAM'S LINE DRAWING ALGORITHM

Input :

```
//Assignment No.1

#include<iostream>
#include<graphics.h>
using namespace std;

class Pixel
{
public:
        void plotPixel(int x,int y)
        {
                putpixel(x,y,WHITE);
        }
};


class Line : public Pixel
{
public:
        void drawLine(int x1,int y1,int x2,int y2,int d)//dda
        {
                int steps,i;
                float dx,dy,xinc,yinc,x,y;
                dx = abs(x2-x1);
                dy = abs(y2-y1);
                if(dx>dy) //gentle
                {
                        steps = dx;
                        xinc = dx/steps;
                        yinc = dy/steps;
                        plotPixel(x1,y1);
                        x = x1;
                        y = y1;
                        for(i=1 ;i <= steps; i++)
                        {
                          if(x1<x2)  //forward
                          {
                                x = x + xinc;
                                if(y1>y2)
                                     y = y - yinc;
```

```
                    else
                            y = y + yinc;
                    plotPixel(x,y+0.5);
                   }
                   else   //backward
                   {
                     x = x - xinc;
                     if(y1>y2)
                         y = y - yinc;
                     else
                             y = y + yinc;
                     plotPixel(x,y+0.5);
                   }
                    delay(50);
               }

          }
          else   //steep
          {
                    steps = dy;
                    xinc = dx/steps;
                    yinc = dy/steps;
                     plotPixel(x1,y1);
                    x = x1;
                    y = y1;
                    for(i=1 ;i <= steps; i++)
                    {
                     if(y1<y2)  //forward
                     {
                      if(x1<x2)
                              x = x + xinc;
                      else
                              x = x - xinc;
                      y = y + yinc;
                      plotPixel(x+0.5,y);
                     }
                     else   //backward
                     {
                      if(x1<x2)
                              x = x + xinc;
                      else
                             x = x - xinc;
                      y = y - yinc;
                      plotPixel(x+0.5,y);
                    }
                     delay(50);
                    }
          }
}//end
void drawLine(int x1,int y1,int x2,int y2,char d)//bresenham
{
          int x,y,i;
          float dx,dy,g;
          x = x1;
          y = y1;
          plotPixel(x,y);
          dx = abs(x2-x1);
          dy = abs(y2-y1);
```

```
if(dx>dy) //gentle
{
        g = 2*dy-dx;
        for(i = 1; i<= dx ; i++)
        {
                if(x1<x2)  //forward
                {
                        x = x + 1;
                        if(g>0)
                        {
                                if(y1<y2)
                                  y = y + 1;
                                else
                                  y = y - 1;
                                g = g + 2 * dy - 2 * dx;
                        }
                        else
                        {
                                g = g + 2 * dy;
                        }
                        plotPixel(x,y);
                        delay(50);
                }
                else  //backward
                {
                        x = x - 1;
                        if(g>0)
                        {
                                if(y1<y2)
                                   y = y + 1;
                                 else
                                   y = y - 1;
                                g = g + 2 * dy - 2 * dx;
                        }
                        else
                        {
                                g = g + 2 * dy;
                        }
                        plotPixel(x,y);
                        delay(50);
                }

        }

}
else  //steep
{
        g = 2*dx-dy;
        for(i = 1; i<= dy ; i++)
        {
                if(y1<y2)  //forward
                {
                        y = y + 1;
                        if(g>0)
                        {
                                if(x1<x2)
                                        x = x + 1;
```

```
                                        else
                                                x = x - 1;
                                        g = g + 2 * dx - 2 * dy;
                                }
                                else
                                {
                                        g = g + 2 * dx;
                                }
                                plotPixel(x,y);
                                delay(50);
                        }
                        else  //backward
                        {
                                y = y - 1;
                                if(g>0)
                                {
                                        if(x1<x2)
                                                x = x + 1;
                                        else
                                                x = x - 1;
                                        g = g + 2 * dx - 2 * dy;
                                }
                                else
                                {
                                        g = g + 2 * dx;
                                }
                                plotPixel(x,y);
                                delay(50);
                        }

                }
        }

    }//end

};


int main()
{
    int gd = DETECT, gm,x1,x2,y1,y2;
    initgraph(&gd,&gm,NULL);
    Line l;
    cout<<"Enter line coordinates";
    cin>>x1>>y1>>x2>>y2;
    l.drawLine(x1,y1,x2,y2,1); //dda
    l.drawLine(x1+10,y1+10,x2+10,y2+10,'b'); //bre
    getch();
    closegraph();
    return 0;
}
```

Output : Line Plotted

Conclusion:

Thus we have implemented program to draw Line using DDA and Bresenham's circle generation algorithms.