Experiment No. 1
Implement DDA Line Drawing algorithm.
Name: Aryan Gaikwad
Roll Number: 09
Date of Performance:
Date of Submission:

Experiment No 1.

Aim: To implement DDA algorithms for drawing a line segment between two given end points.

Objective: Draw the line using (vector) generation algorithms which determine the pixels that should be turned ON are called as digital differential analyzer (DDA).It is one of the techniques for obtaining a rasterized straight line. This algorithm can be used to draw the line in all the quadrants.

Theory:
DDA algorithm is an incremental scan conversion method. Here 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. Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

Algorithm:

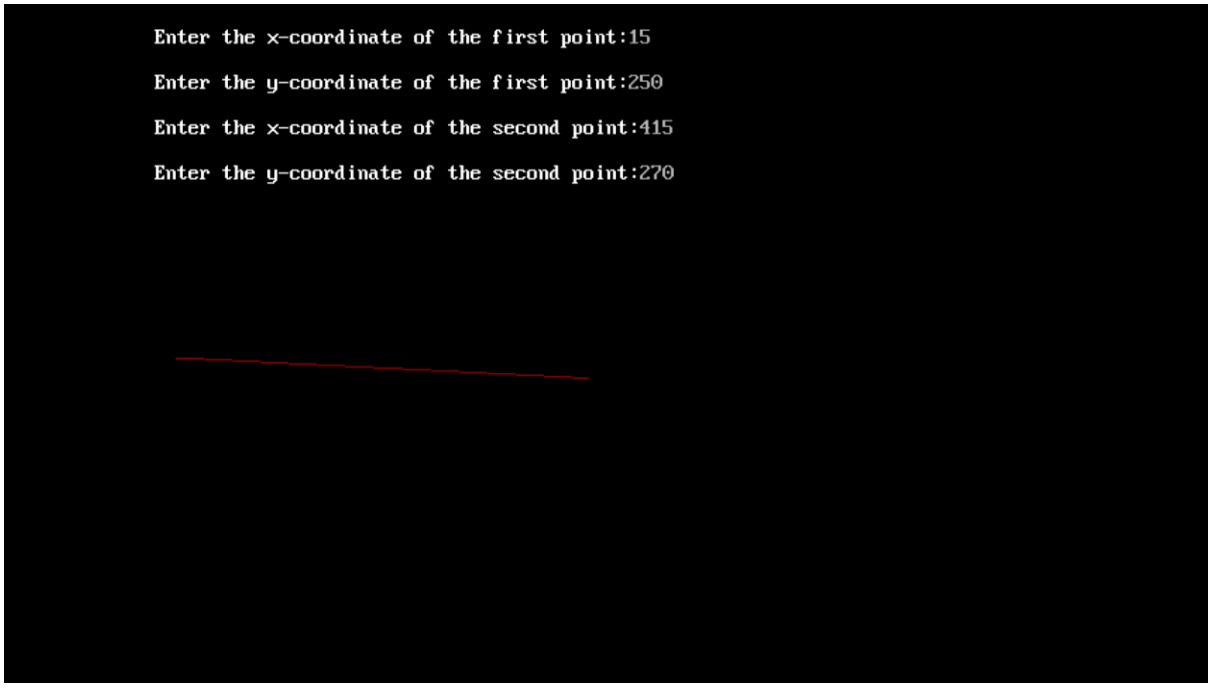
1. Input two endpoints: (x1, y1) and (x2, y2).
2. Calculate the differences in the x and y coordinates:
3. dx = x2 - x1 dy = y2 - y1
4. Determine the number of steps required to draw the line. You can use the maximum difference between dx and dy:
5. steps = max(abs(dx), abs(dy))
6. Calculate the increments for x and y:
7. x_increment = dx / steps y_increment = dy / steps
8. Initialize the current position (x, y) as the starting point (x1, y1):
9. x = x1 y = y1
10. For each step from 1 to steps:
11. a. Round the current coordinates to the nearest integer since pixel positions are discrete. b. Plot the pixel at the current position (x, y). c. Update the current position:
12. x = x + x_increment y = y + y_increment
13. Continue the loop until you have plotted all the necessary pixels to draw the line segment.

Program:

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
int main()
{
   float x,y,x1,y1,x2,y2,dx,dy,step;
   int i,gd=DETECT,gm;
   //detectgraph(&gd,&gm);
   initgraph(&gd,&gm,"");
   printf("\nEnter the x-coordinate of the first point:");
   scanf("%f",&x1);
   printf("\nEnter the y-coordinate of the first point:");
   scanf("%f",&y1);
   printf("\nEnter the x-coordinate of the second point:");
   scanf("%f",&x2);
   printf("\nEnter the y-coordinate of the second point:");
   scanf("%f",&y2);
   dx=abs(x2-x1);
   dy=abs(y2-y1);
   if(dx>dy)
   {
      step=dx;
   }
   else
   {
      step=dy;
   }
```

```
    dx=dx/step;
    dy=dy/step;
    x=x1;
    y=y1;
    i=1;
    while(i<=step)
    {
        putpixel(x,y,14);
        x=x+dx;
        y=y+dy;
        i=i+1;
        delay(100);
    }
    getch();
    closegraph();
}
```

Output:



Conclusion: Comment on –

1. Pixel : A pixel is the smallest unit of a digital image or display, typically a tiny square or dot that can be illuminated with a specific color. In graphics programming, the screen is divided into a grid of pixels, and each pixel can be individually controlled to create images or drawings.

2. Equation for a Line : The equation for a straight line in the Cartesian coordinate system is typically represented as `y = mx + b`, where 'm' is the slope of the line, and 'b' is the y-intercept. In the code, the algorithm does not explicitly use this equation but calculates the next pixel position along the line using the Bresenham Line Drawing Algorithm.

3. Need for Line Drawing Algorithm **:** Computers use pixels on a grid to represent images. To draw continuous lines, you need an algorithm to determine which pixels to color.