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:

Step1: Start Algorithm

Step2: Declare x_1,y_1,x_2,y_2,dx,dy,x,y as integer variables.

Step3: Enter value of x_1,y_1,x_2,y_2.

Step4: Calculate dx = x_2-x_1

Step5: Calculate dy = y_2-y_1

Step6: If ABS (dx) > ABS (dy)
Then step = abs (dx)
Else

Step7: x_{inc}=dx/step
y_{inc}=dy/step
assign x = x_1
assign y = y_1

Step8: Set pixel (x, y)

Step9: $x = x + x_{inc}$
$y = y + y_{inc}$
Set pixels (Round (x), Round (y))

Step10: Repeat step 9 until $x = x_2$

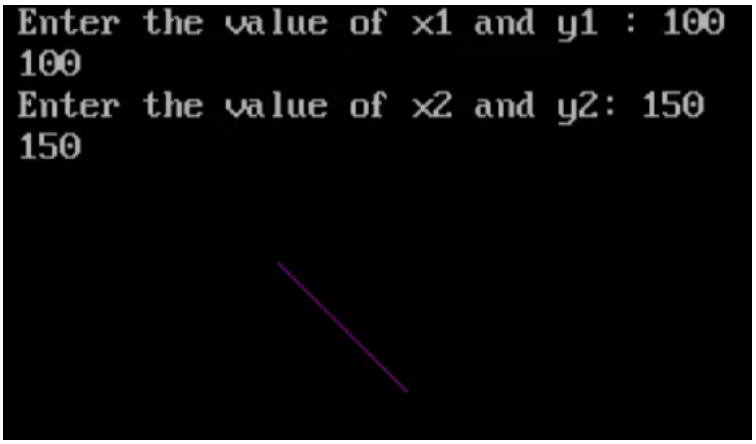
Step11: End Algorithm

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
2. Equation for line
3. Need of line drawing algorithm
4. Slow or fast

1. A pixel, short for "picture element," is the smallest and indivisible unit of a digital image, representing a single point in a grid and typically containing color or grayscale information.

2. The equation of a line in two-dimensional Cartesian coordinates is typically represented in slope-intercept form as:
\[y = mx + b\]
Where:
- \(y\) is the vertical coordinate of a point on the line.
- \(x\) is the horizontal coordinate of a point on the line.
- \(m\) is the slope of the line, indicating its steepness or inclination.
- \(b\) is the y-intercept, representing the point where the line intersects the y-axis.

This equation describes a straight line and provides a way to calculate the value of y for any given x on the line, based on the slope m and the y-intercept b. The slope m determines the line's angle or steepness, and the y-intercept b specifies where the line crosses the y-axis.

3. The Digital Differential Analyzer (DDA) algorithm plays a crucial role in computer graphics by addressing the need for efficient and accurate line rendering on digital displays. It simplifies the process of drawing straight lines between two specified points, making it especially valuable for real-time applications like video games and simulations. DDA is designed for digital displays with a grid of pixels, ensuring precise alignment with pixel boundaries. Moreover, it can be extended to support antialiasing, enhancing the visual quality of graphics. DDA's simplicity and accessibility make it an excellent tool for educational purposes, helping students understand the fundamental concepts of pixel-based graphics and serving as a foundation for more complex rendering algorithms. While DDA is highly efficient for basicline drawing, it's important to note that other algorithms may be more suitable for complex curves or scenarios requiring higher accuracy.

4. The DDA algorithm is relatively slower compared to more optimized techniques like Bresenham's line algorithm for fast line drawing in computer graphics. Bresenham's algorithm is faster because it uses integer arithmetic and incremental calculations, making it more efficient for real-time or performance-critical applications.