Experiment No. 1
Topic :DDA algorithm
Name: Harsh Tripathi
Roll Number: 59
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:
Input the two endpoints of the line segment, (x1,y1) and (x2,y2).
Calculate the difference between the x-coordinates and y-coordinates of the endpoints as dx and dy respectively.
Calculate the slope of the line as m = dy/dx.
Set the initial point of the line as (x1,y1).

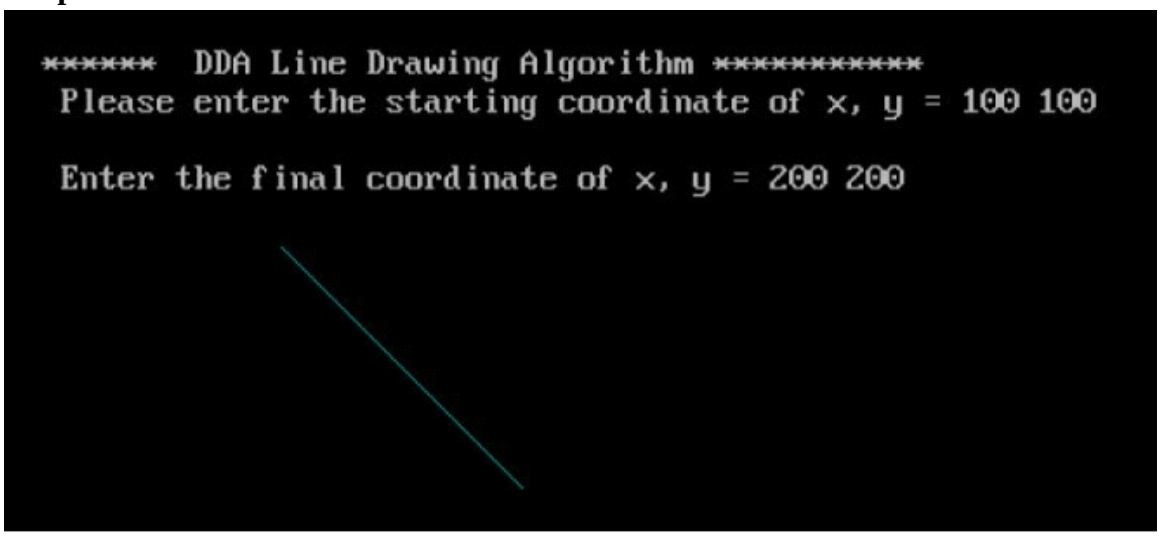
Loop through the x-coordinates of the line, incrementing by one each time, and calculate the corresponding y-coordinate using the equation y = y1 + m(x – x1).
Plot the pixel at the calculated (x,y) coordinate.
Repeat steps 5 and 6 until the endpoint (x2,y2) is reached.

Program:

```
#include<graphics.h>
#include<math.h>
#include<conio.h>
void main()
{
int x0,y0,x1,y1,i=0;
float delx,dely,len,x,y;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** DDA Line Drawing Algorithm ***********");
printf("\n Please enter the starting coordinate of x, y = ");
scanf("%d %d",&x0,&y0);
printf("\n Enter the final coordinate of x, y = ");
scanf("%d %d",&x1,&y1);
dely=abs(y1-y0);
delx=abs(x1-x0);

if(delx<dely)
{
len = dely;
}
else
{
len=delx;
}
delx=(x1-x0)/len;
dely=(y1-y0)/len;
x=x0+0.5;
y=y0+0.5;
do{
putpixel(x,y,3);
x=x+delx;
y=y+dely;
i++;
delay(30);
}while(i<=len);
getch();
closegraph();
}
```

Output:



Conclusion: Comment on -

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

Pixel: In the context of the DDA algorithm, a pixel refers to the smallest unit of a digital image that can be displayed on a screen. The DDA algorithm is used for efficient rendering of lines on a digital display, which involves determining which pixels to activate to represent the desired line.

Equation for line: The DDA algorithm uses the equation of a line to calculate the coordinates of the points on the line. The equation used is typically the slope-intercept form of a line, which is y = mx + c, where 'm' is the slope of the line and 'c' is the y-intercept.

Need for line drawing algorithm: Line drawing algorithms like DDA are necessary for efficient and accurate rendering of lines on digital displays, such as computer monitors or screens. Without such algorithms, it would be challenging to accurately represent lines and shapes in computer graphics and various applications that involve graphical representations.

Slow or fast: The DDA algorithm is relatively simple and straightforward, making it easy to implement. However, it is not the most efficient algorithm for drawing lines, especially when dealing with lines that are steep or near-horizontal. DDA can be considered relatively slow compared to some of the more advanced line drawing algorithms like Bresenham's line algorithm, which provides better performance by using integer increments and avoiding floating-point arithmetic.