Introduction to DDA Algorithm

## THEORY:

The DDA algorithm is based on calculating either Δx or Δy. Sampling of line is done at unit intervals in one co-ordinate and corresponding value nearest to the line path is determined in another coordinate.

For lines with positive slope less than or equal to 1, we sample at unit x intervals Cox = 1) and compute each Successive y values as

Yk+1 = mxk+1 + b

= m (xk + Δx) + b

= mxk + m + b

Yk+1 = yk +m --①; where k = 1,2,3…

For lines with positive slope greater than 1, a step in x creates a slip in y that is greater than 1. Thus, we reverse the roles of x and y. That is we sample at unit y interval (Δy=1) and calculating each succeeding x values as,

Xk+1 = Xk + 1/m---(ii) (|m|>1)

Equations (i) and (ii) are based on the assumption that lines are to be processed from the left end point to right end point. If we reverse the processing, i.e. from right end point to left," we have,

Δx=-1 & Δy = -1, such that

yk+1 = Yk -m --(iii)

Xk+1=Xk – 1/m---(iv)

The same equations explained above can be used for a line with negative slope.

For |m|<1, Δx = 1; yk+1 = yk+ m

Iml>1, Δy = 1; Xk+1 = xk + 1/m

(Start end point at left)

For |m|<1, Δx = -1; yk+1 = yk - m

Iml>1, Δy = -1; Xk+1 = xk - 1/m

(Start end point at right)

## Algorithm:

1. Read end points (x1,y1), (x2,y2)

2. Approximate length of line i.e.

if (abs (x2-x1)> abs (y2-y1)) then,

length = abs (x2-x1)

else

length = abs (y2-y1)

3. Select raster unit

i.e. Δx

Δy

4. Round value

X = x1+0.5 \* sign (Δx)

Y = y1 + 0.5 \* sign (Δy)

5. Now, plot points

i=1

while (i<=length)

{

Plot (integer (x), integer(y))

X= X+ Δx

Y= Y+ Δy

i= I + 1

}

6. Stop.

## Program:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void dda (int, int, int, int);

void main()

{

int gd = DETECT,gm;

initgraph (&gd, &gm, "c:\\tc\\bgi");

dda (100,100,200,100);

dda (200,100,200,200);

dda (200,200,100,200);

dda (100,200,100,100);

dda (100,100,200,200);

dda (100,200,200,100);

dda (100,125,200,125);

dda (100,150,200,150);

dda (100,175,200,175);

dda (175,100,175,200);

dda (150,100,150,200);

dda(125,100,125,200);

dda(100,150,150,100);

dda(150,100,200,150);

dda(200,150,150,200);

dda(150,200,100,150);

getch();

closegraph();

}

void dda(int x1,int y1, int x2, int y2)

{

int i,stepsize, dx,dy;

float x,y,xinc, yinc;

dx=x2-x1;

dy-y2-y1;

x=x1;

y=y1;

if (abs(dy)>abs(dx))

{

stepsize=abs(dy);

}

else

{

stepsize=abs(dx);

}

xinc=dx/ (float) stepsize;

yinc=dy/ (float) stepsize;

putpixel(x,y,RED);

for(i=0; i <stepsize;i++)

{

x =x+xinc;

y=y+yinc;

putpixel((int)(x+0.5), (int)(y+0.5),RED);

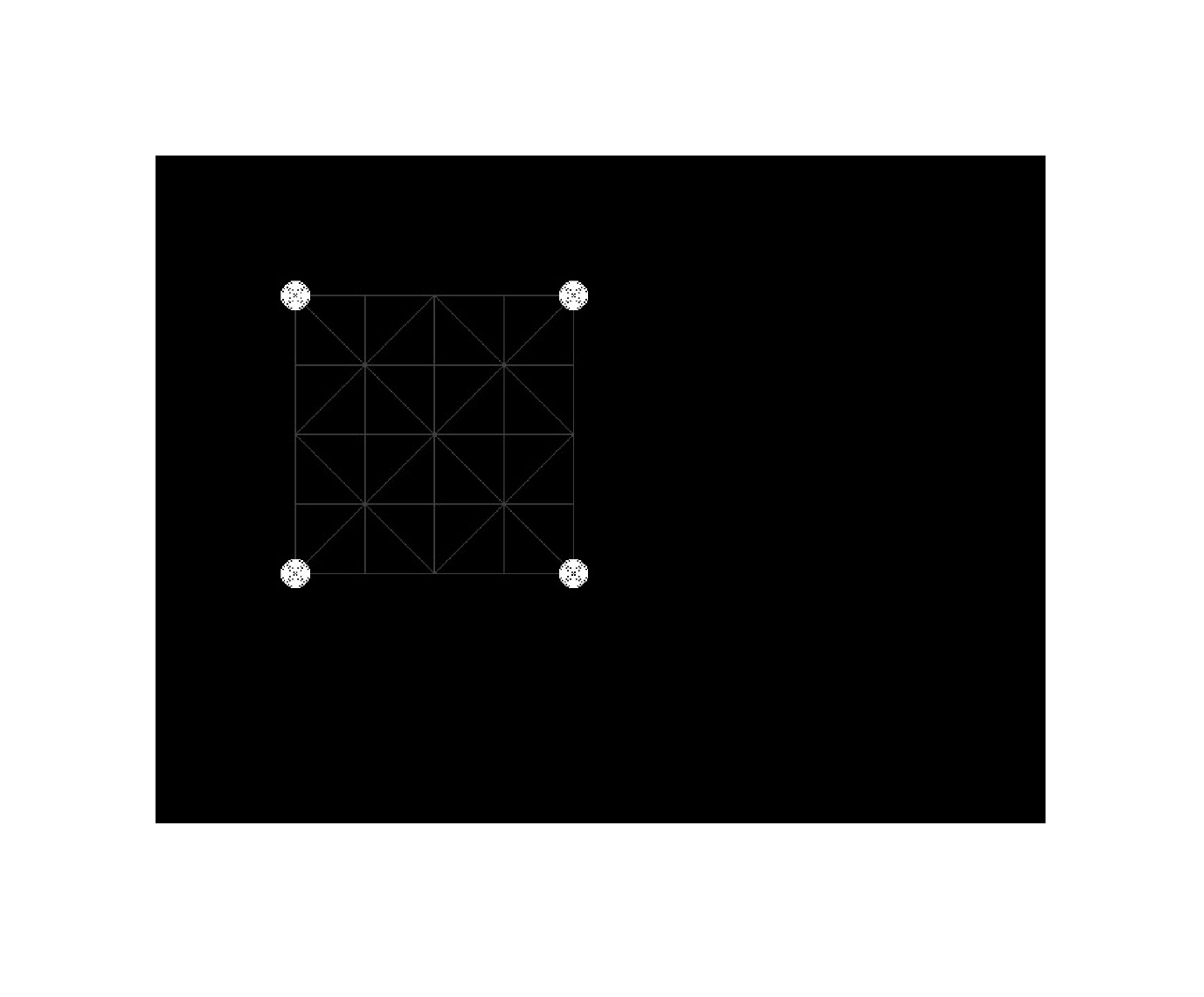
delay(10);

}

}

## Output:

The program initializes graphics mode and uses the DDA algorithm to draw a square with vertices at (100, 100), (200, 100), (200, 200), and (100, 200). It then adds the diagonals and several horizontal and vertical lines inside the square, creating a detailed grid pattern with intersecting lines, all drawn with a slight delay to animate the drawing process.



## Discussion:

In this lab, we used DDA (Digital Differential Analyzer) algorithm to produce required shape as an output. We Created a function called dda and passed the end-points as an argument and then drew a straight line from one end point to another end point

## Conclusion:

Hence, we plotted multiple straight line using DDA algorithm which as a whole look like our desired image.