**SSN College of Engineering**

**Department of Computer Science and Engineering**

## UCS1712 – GRAPHICS AND MULTIMEDIA LAB

**Ex 2 – Drawing 2D Primitives - Line-DDA Algorithm**

SATHEESH KUMAR G R

185001136

AIM:

To draw lines as a series of points using DDA line drawing algorithm.

ALGORITHM:

1. Read the sets of points (x1,y1) and (x2,y2) to plot line.
2. Calculate dx as x1-x2 and dy as y2-y1.
3. Find the number of steps as the maximum of the absolute values of dx and dy.
4. Slope m is dy/dx.
5. xinc is dx/steps and yinc is dy/steps.
6. Plot (x1,y1).
7. Increment x and y with xinc and yinc respectively and plot (x,y).
8. Repeat step 7 for ‘steps’ number of times.
9. Flush the line to output screen.
10. Repeat step 1-9 for all the pair of points given as input.

CODE:

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

double X1, Y1, X2, Y2;

double arrx1[10],arry1[10],arrx2[10],arry2[10];

float round\_value(float v)

{

  return floor(v + 0.5);

}

void LineDDA(void)

{

  glClear(GL\_COLOR\_BUFFER\_BIT);

  int j =0;

  while(j<8)

  {

      X1 = arrx1[j];

      Y1 = arry1[j];

      X2 = arrx2[j];

      Y2 = arry2[j];

      double dx=(X2-X1);

      double dy=(Y2-Y1);

      double steps;

      float xInc,yInc,x=X1,y=Y1;

      /\* Find out whether to increment x or y \*/

      steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));

      float m = dy/dx;

      xInc=dx/(float)steps;

      yInc=dy/(float)steps;

      /\* Plot the points \*/

      glBegin(GL\_POINTS);

      /\* Plot the first point \*/

      glPointSize(4.0);

      glColor3f(1.0,0.0,0.0);

      glVertex2d(x,y);

      int k;

      /\* For every step, find an intermediate vertex \*/

      for(k=0;k<steps;k++)

      {

        x+=xInc;

        y+=yInc;

        /\* printf("%0.6lf %0.6lf\n",floor(x), floor(y)); \*/

        glColor3f(1.0,0.0,0.0);

        glVertex2d(round\_value(x), round\_value(y));

      }

      glEnd();

      j++;

      glFlush();

    }

}

int main(int argc, char \*\*argv)

{

int i = 0;

while(i<8)

{

  printf("Enter two end points of the line to be drawn:\n");

  printf("\nCase %d:",i+1);

  printf("\nEnter Point1( X1 , Y1):\n");

  scanf("%lf,%lf",&X1,&Y1);

  arrx1[i] = X1;

  arry1[i] = Y1;

  printf("\n");

  printf("\nCase %d:",i+1);

  printf("\nEnter Point2( X2 , Y2):\n");

  scanf("%lf,%lf",&X2,&Y2);

  arrx2[i] = X2;

  arry2[i] = Y2;

  i++;

}

  glutInit(&argc,argv);

  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

  glutInitWindowPosition(0,0);

  glutInitWindowSize(1000,1000);

  glutCreateWindow("DDA\_Line");

  glClearColor(1.0,1.0,1.0,0);

  glColor3f(0.0,0.0,1.0);

  gluOrtho2D(-500 , 500 , -500 , 500);

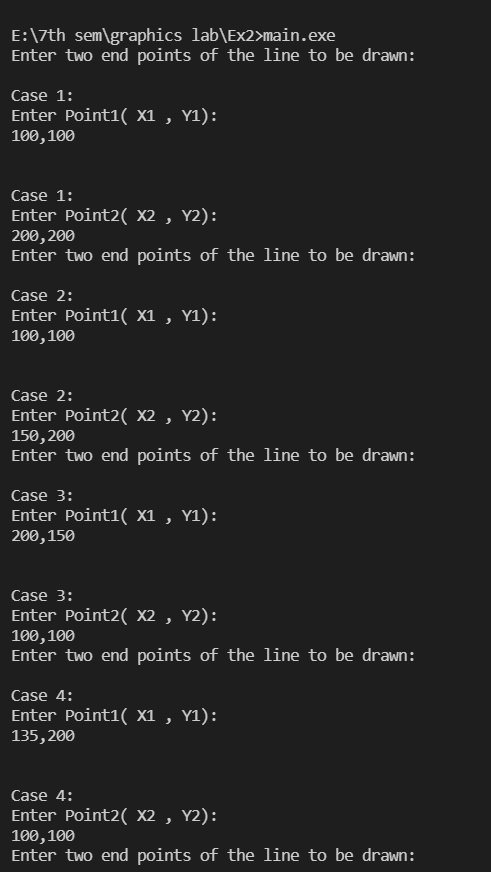
  glutDisplayFunc(LineDDA);

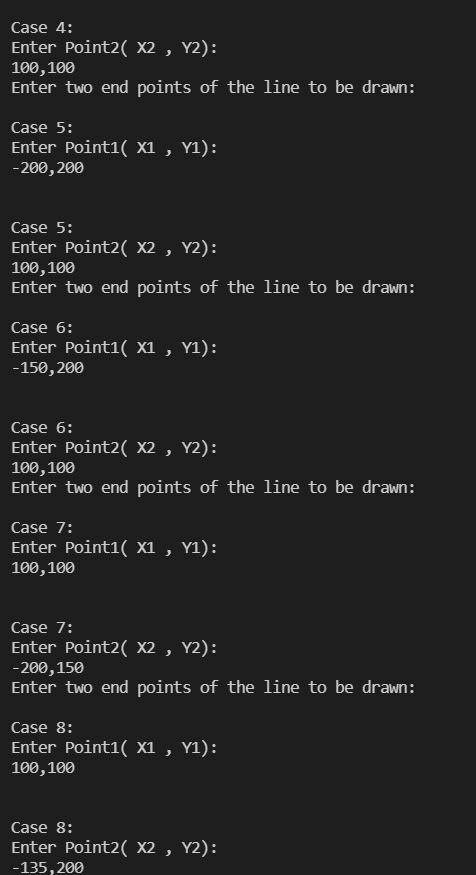
  glutMainLoop();

  return 0;

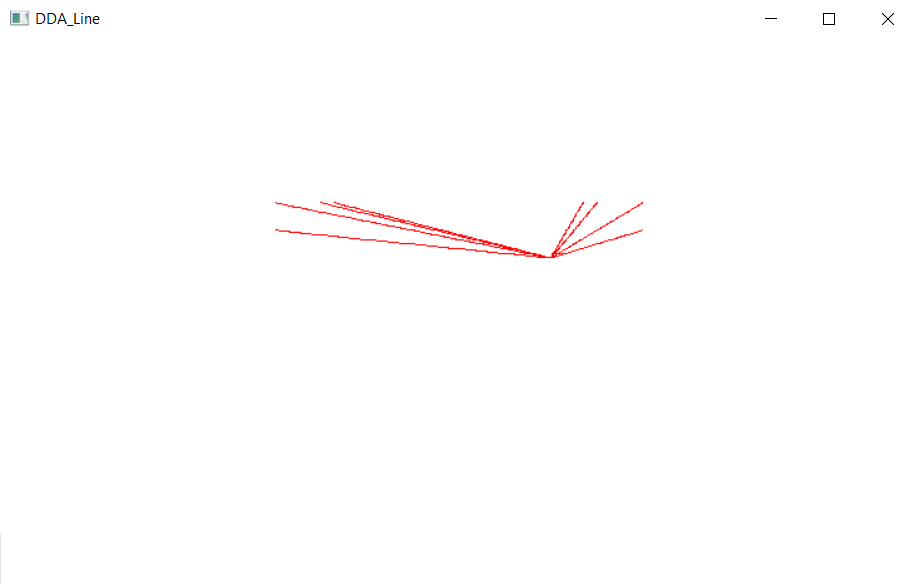
}

INPUT:





OUTPUT:



----------------------------------------------------------------------------------------------------------------

2)

AIM:

To generate a rain like pattern using lines drawn using DDA algorithm.

ALGORITHM:

1. Initialise point1 as x,y where x and y are randomly generated and plot(x,y).
2. Randomly generate the size of the line to lie between 0 to 30.
3. Store the size of the line in a variable called step.
4. According to the size of the line, increment x and y by 1 each and plot each point to construct the line as series of points for ‘steps’ number of times.
5. Flush the line to output screen.
6. Continue the steps 1-3 500 times to generate 500 lines.

CODE:

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#include<time.h>

float x1,x2,y1,y2,m;

void display(void)

{

for(int i=0;i<500;i++)

{

float dy,dx,step,x,y,xinc,yinc;

step = rand()%30;

xinc = 1;

yinc = 1;

x= (rand() % (200)) - 100;

y= (rand() % (200)) - 100;

glBegin(GL\_POINTS);

glColor3ub(100,100,255);

for (int k=0;k<=step;k++)

{

glVertex2i(x,y);

x += xinc;

y += yinc;

}

glEnd();

}

glFlush();

}

void init(void)

{

glClearColor(1.0,1.0,1.0,0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-100,100,-100,100);

}

int main(int argc, char\*\* argv) {

srand(time(0));

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (400, 400);

glutInitWindowPosition (0,0);

glutCreateWindow ("Rain Pattern DDA");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

