SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 01/08/2021

EX NO: 2

Drawing 2D Primitives –Line – DDA Algorithm

1. To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using the DDA line drawing algorithm.

Aim:

To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using the DDA line drawing algorithm.

Algorithm:

Reading all 8 pairs of points from the user that satisfies all the cases and its subdivisions given in the question.

- Find dx(difference in x coordinates) and dy(difference in y coordinates).
- Now find which one is greater and assign it to step. Doing this will make sure that no gap is in between the points in the line.
- Now find the x increment and y increment.
- Now run a loop from 0 to step. For each iteration, using GL_POINTS draw a point in the plane and increment the x and y coordinates.

8 points are
Case 1: +ve slope Left to Right line

1. |m|<= 1 - (300, 150) & (200, 50)
2. |m|>1 - (500, 150) & (600, 50)

Case 2: +ve slope Right to Left line

1. |m|<= 1 - (500, 200) & (700, 150)
2. |m|>1 - (300, 200) & (100, 150)

Case 3: -ve slope Left to Right line

1. |m|<= 1 - (500, 350) & (700, 400)
2. |m|>1 - (300, 350) & (100, 400)

Case 4: -ve slope Right to Left line

1. |m|<= 1 - (500, 400) & (600, 550)
2. |m|>1 - (300, 400) & (200, 550)

(Note : x-axis is from top-left to top-right, y-axis is from top-left to bottom-left like the paint app.)

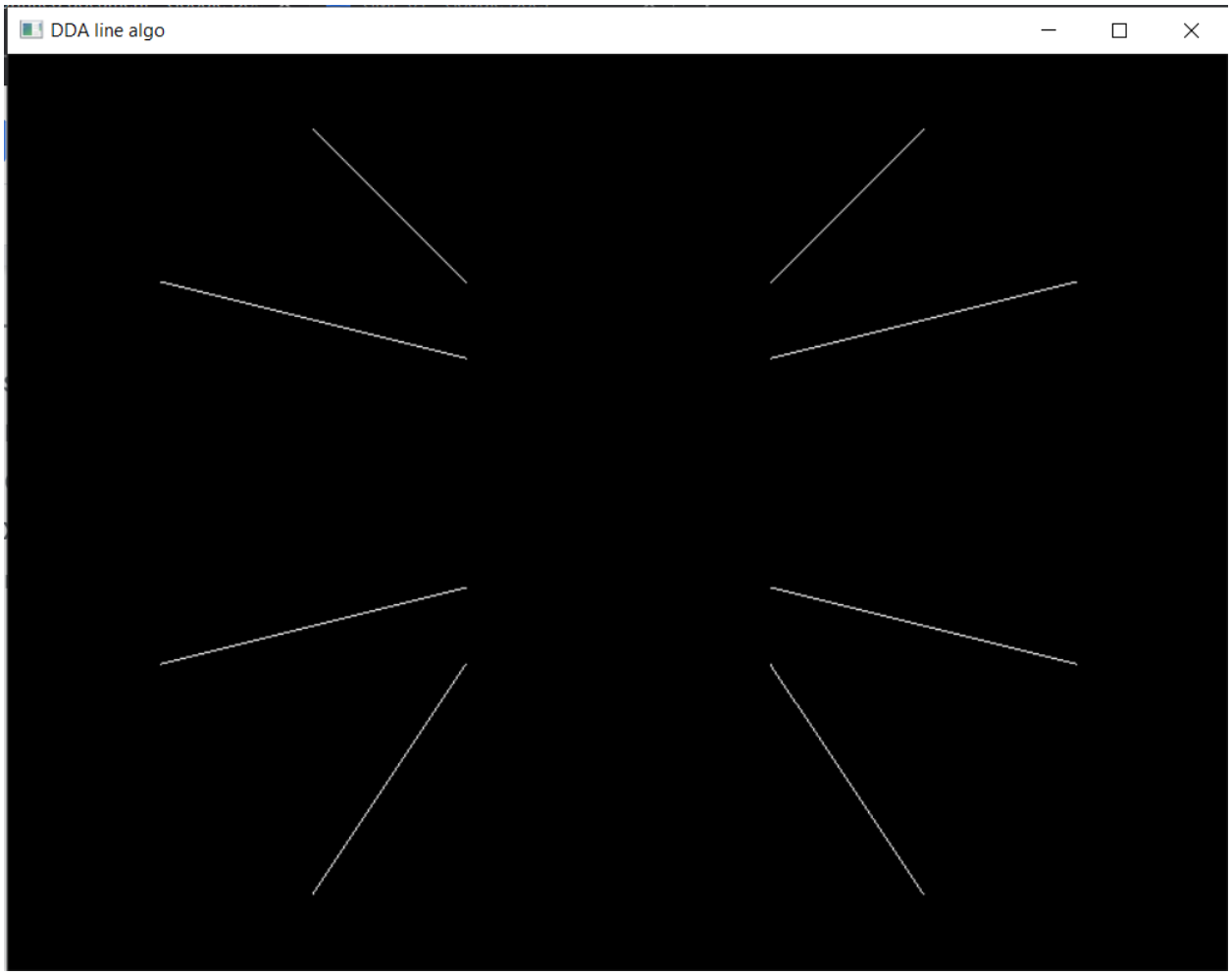
Code:

```
#include<windows.h>
#include<gl/glut.h>
#include<cstdlib>
#include<iostream>
using namespace std;

float x1[8], Y1[8], x2[8], y2[8];

void myInit()
{
	glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
	glPointSize(1);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 800.0, 600.0, 0.0);
	glClear(GL_COLOR_BUFFER_BIT);
}

void display(void)
{
    float dy, dx, step, x, y, k, Xin, Yin;

    for (int i = 0; i < 8; i++)
    {
        dx = x2[i] - x1[i];
        dy = y2[i] - Y1[i];

        if (abs(dx) > abs(dy))
        {
```

```
            step = abs(dx);
        }
        else
            step = abs(dy);

        Xin = dx / step;
        Yin = dy / step;

        x = x1[i];
        y = Y1[i];
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();

        for (k = 1; k <= step; k++)
        {
            x = x + Xin;
            y = y + Yin;

            glBegin(GL_POINTS);
            glVertex2i(x, y);
            glEnd();
        }
    }

    glFlush();
}

int main(int argc, char** argv)
{
    for (int i = 0; i < 8; i++)
    {
        cout << "x1 : ";
        cin >> x1[i];
        cout << "y1 : ";
        cin >> Y1[i];
        cout << "x2 : ";
        cin >> x2[i];
        cout << "y2 : ";
        cin >> y2[i];
    }

    glutInit(&argc, argv);                  // Initialize GLUT
```

```
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);   // Set the window's initial width &
height
    glutInitWindowPosition(50, 50); // Position the window's initial
top-left corner
    glutCreateWindow("DDA line algo"); // Create a window with the given
title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for
window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
    return 0;
}

// 500 150 600 50 300 150 200 50 500 200 700 150 300 200 100 150 500 350
700 400 300 350 100 400 500 400 600 550 300 400 200 550
```

Output Screenshot:



Result:

Lines are drawn by using the given endpoints for all the cases and its subdivisions and by implementing the DDA algorithm.

2. Replicate the following pattern using DDA algorithm:

Aim:

To replicate the given pattern using the DDA algorithm.

Algorithm:

1. This program is divided into 2 subdivisions. One for plotting patterns for the upper half(x,y - (0,0) to (0, 500)) and the other for the lower half(from (0,0) to (500, 0)).
2. Running an outer loop that increments the **y coordinate** value from 0 to 500 in random increments.

2.1. Now make the previous point as x1, y1 and find a random increment and add it to x1 and y1 to get x2 and y2. There is no need for any float values since the slope(m) value is 1(45 degree).
2.2. Repeat this until x1, y1 or x2, y2 reaches 500.

3. Running an outer loop that increments the **x coordinate** value from 0 to 500 in random increments.
3.1. Now make the previous point as x1, y1 and find a random increment and add it to x1 and y1 to get x2 and y2. There is no need for any float values since the slope(m) value is 1(45 degree).
3.2. Repeat this until x1, y1 or x2, y2 reaches 500.
4. Now plot the lines by using the DDA algorithm that is implemented and explained in the previous program.

Code:

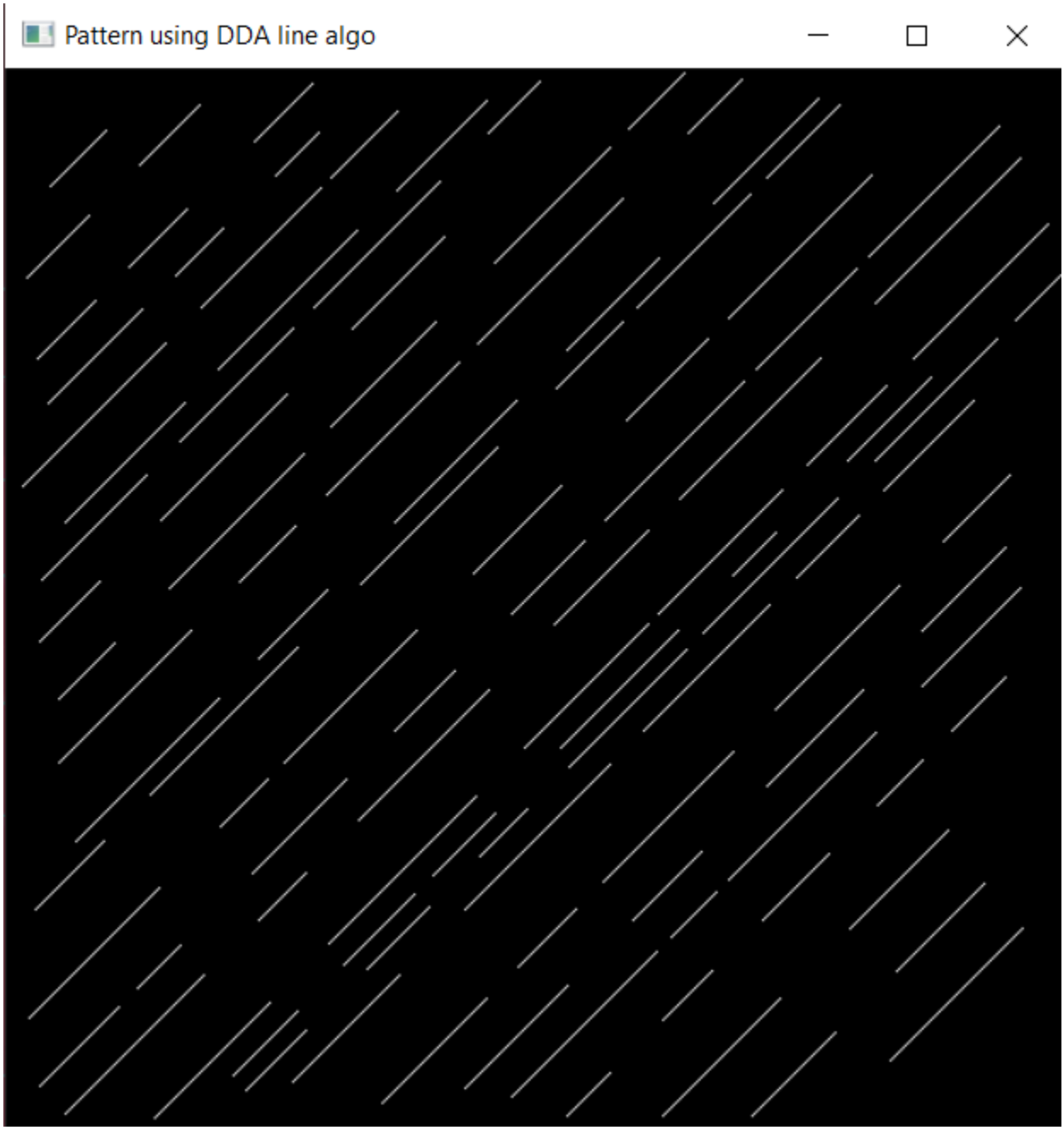
```
#include<windows.h>
#include<gl/glut.h>
#include<cstdlib>
#include<iostream>

using namespace std;

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 500.0, 0.0, 500.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

void display(void)
{
    float dy, dx, step, x, y, k, Xin, Yin;
    float x1, Y1, x2, y2;
    float current_y, current_x, offset1, offset2;
    bool canDraw = false;


    x1 = x2 = 0;
    Y1 = y2 = current_y = 5;
```

```
while(current_y < 500)
{
    while(true){
        offset2 = (rand() % 30) + 5;
        x1 = x2 + offset2;
        Y1 = y2 + offset2;
        offset2 = (rand() % 50) + 20;
        x2 = x1 + offset2;
        y2 = Y1 + offset2;
        dx = x2 - x1;
        dy = y2 - Y1;
        //cout << x1 << " " << Y1 << " " << x2 << " " << y2;
        if (x1 < 500 && Y1 < 500 && x2 < 500 && y2 < 500)
        {
            if (abs(dx) > abs(dy))
            {
                step = abs(dx);
            }
            else
                step = abs(dy);

            Xin = dx / step;
            Yin = dy / step;

            x = x1;
            y = Y1;
            glBegin(GL_POINTS);
            glVertex2i(x, y);
            glEnd();

            for (k = 1; k <= step; k++)
            {
                x = x + Xin;
                y = y + Yin;

                glBegin(GL_POINTS);
                glVertex2i(x, y);
                glEnd();
            }
        }
        else
        {
            //cout << "break\n";
```

```
            break;
        }
    }
    offset1 = (rand() % 50) + 10;
    Y1 = y2 = current_y + offset1;
    current_y = Y1;
    x1 = x2 = 0;
    //cout << current_y << "\n";
}

x1 = x2 = current_x = 20;
Y1 = y2 = 0;
while (current_x < 500)
{
    while (true) {
        offset2 = (rand() % 30) + 5;
        x1 = x2 + offset2;
        Y1 = y2 + offset2;
        offset2 = (rand() % 50) + 20;
        x2 = x1 + offset2;
        y2 = Y1 + offset2;
        dx = x2 - x1;
        dy = y2 - Y1;
        if (x1 < 500 && Y1 < 500 && x2 < 500 && y2 < 500)
        {
            if (abs(dx) > abs(dy))
            {
                step = abs(dx);
            }
            else
                step = abs(dy);

            Xin = dx / step;
            Yin = dy / step;

            x = x1;
            y = Y1;
            glBegin(GL_POINTS);
            glVertex2i(x, y);
            glEnd();

            for (k = 1; k <= step; k++)
            {
```

```
                    x = x + Xin;
                    y = y + Yin;

                    glBegin(GL_POINTS);
                    glVertex2i(x, y);
                    glEnd();
                }
            }
            else
            {
                break;
            }
        }
        offset1 = (rand() % 50) + 10;
        x1 = x2 = current_x + offset1;
        current_x = x1;
        Y1 = y2 = 0;
    }

    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);   // Set the window's initial width &
height
    glutInitWindowPosition(50, 50); // Position the window's initial
top-left corner
    glutCreateWindow("Pattern using DDA line algo"); // Create a window
with the given title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for
window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
    return 0;
}
```

Output Screenshots:



Result:

The given pattern is replicated by using the DDA algorithm.