Shivcharan T
205001100
CSE - B

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Lab Exercise 2 : DDA Line Drawing Algorithm in C++ using OpenGL

To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using DDA line drawing algorithm.
Case 1: +ve slope Left to Right line
Case 2: +ve slope Right to Left line
Case 3: -ve slope Left to Right line
Case 4: -ve slope Right to Left line
Each case has two subdivisions
(i) |m|<= 1 (ii) |m|>1
Note that all four cases of line drawing must be given as test cases.

Aim:
To implement DDA line drawing algorithm

Algorithm:

1. Accept two endpoints `P1(x1, y1)` and `P2(x2, y2)` of the line to be drawn.
2. Calculate the differences in x and y coordinates:
   dx = x2 - x1
   dy = y2 – y1
3. Determine the number of steps required for drawing the line. Use the larger of `dx` and `dy` as the number of steps:
   steps = max(abs(dx), abs(dy))
4. Calculate the incremental values for `x` and `y`:
   x_increment = dx / steps
   y_increment = dy / steps
5. Initialize a loop and start drawing the line by repeatedly adding the incremental values to `x` and `y`:
   x = x1
   y = y1
   for i from 1 to steps:
      Plot the point (x, y)
      x = x + x_increment
      y = y + y_increment
6. The loop will draw the line from `P1` to `P2` by plotting points along the line at regular intervals.
7. End the algorithm.

Code:

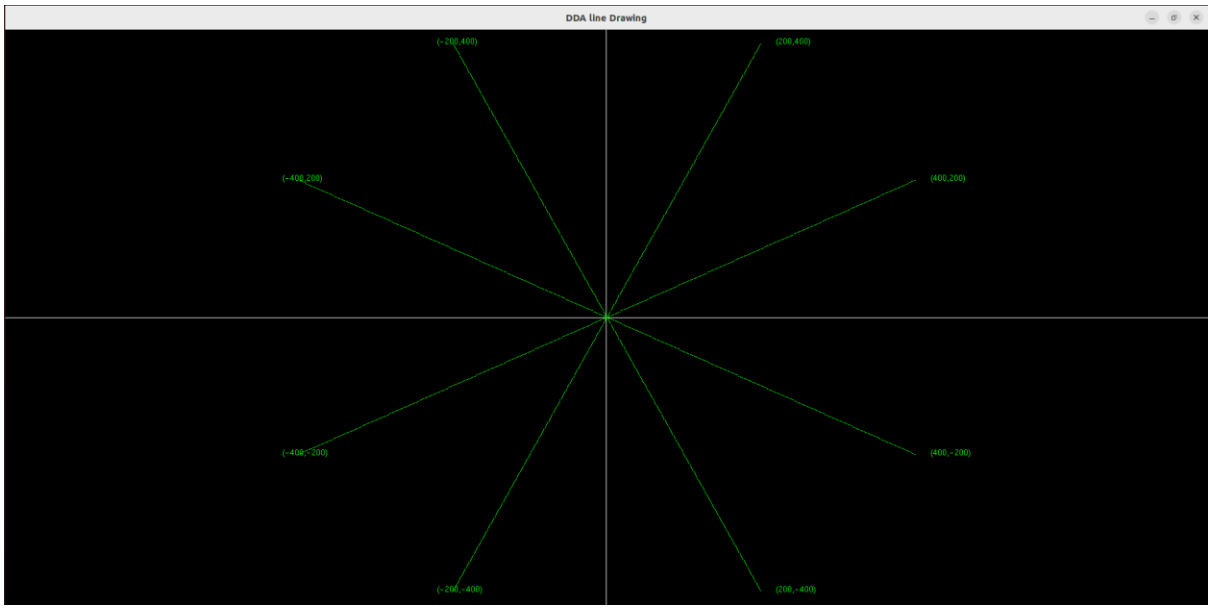
```
#include <stdio.h>
#include <GL/glut.h>
#include <math.h>
#include <cstring>
#define pi 3.142857

void output(int x, int y, const char *string)
{
    glRasterPos2f(x, y);
    int len, i;
    len = (int)strlen(string);
    for (i = 0; i < len; i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, string[i]);
    }
}
void draw_pixel(int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
void myInit(void)
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(0.0, 1.0, 0.0); // making picture color green (in RGB mode), as middle
argument is 1.0
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-780, 780, -420, 420);
}
void drawLineDDA(float x0, float y0, float xn, float yn)
{
    // glClear(GL_COLOR_BUFFER_BIT);
    float dx = xn - x0;
    float dy = yn - y0;
    float steps = fabs(dx) > fabs(dy) ? fabs(dx) : fabs(dy);
    float xIncrement = dx / steps;
    float yIncrement = dy / steps;
    float x = x0;
    float y = y0;
    for (int i = 0; i <= steps; ++i)
    {
        draw_pixel(static_cast<int>(x + 0.5), static_cast<int>(y + 0.5));
        x += xIncrement;
        y += yIncrement;
    }
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
```

```
    glColor3f(1.0, 1.0, 1.0);

    glBegin(GL_LINES);
    glVertex2d(0, 420);
    glVertex2d(0, -420);
    glEnd();

    glBegin(GL_LINES);
    glVertex2d(780, 0);
    glVertex2d(-780, 0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    drawLineDDA(0, 0, 400, 200);
    output(420, 200, "(400,200)");

    drawLineDDA(0, 0, 200, 400);
    output(220, 400, "(200,400)");

    drawLineDDA(-400, 200, 0, 0);
    output(-420, 200, "(-400,200)");

    drawLineDDA(-200, 400, 0, 0);
    output(-220, 400, "(-200,400)");

    drawLineDDA(0, 0, 400, -200);
    output(420, -200, "(400,-200)");

    drawLineDDA(0, 0, 200, -400);
    output(220, -400, "(200,-400)");

    drawLineDDA(-400, -200, 0, 0);
    output(-420, -200, "(-400,-200)");

    drawLineDDA(-200, -400, 0, 0);
    output(-220, -400, "(-200,-400)");

    glFlush();
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    // giving window size in X- and Y- direction
    glutInitWindowSize(1366, 768);
    glutInitWindowPosition(0, 0);
    // Giving name to window
    glutCreateWindow("DDA line Drawing");
    myInit();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

run.sh:

```
g++ 2.cpp -lGL -lglut -lGLU
./a.out
```

Sample I/O:



Learning Outcomes:

Thus, DDA line drawing algo has been implemented with OpenGL and GLUT frameworks.