Shivcharan T
205001100
CSE - B

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

Lab Exercise 3 : Bresenham's 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. Calculate the differences in the x and y coordinates between the two endpoints, which we'll call Δx and Δy.
2. Determine the direction of the line by checking whether Δx and Δy are positive or negative. This helps decide whether to increment or decrement the x and y coordinates while drawing the line.
3. Initialize an error term to keep track of how far off the line is from the ideal path. This error term is calculated as Δx - Δy.
4. Start at the first point (x1, y1) and draw a pixel at that location.
5. Enter a loop that continues until you reach the second point (x2, y2).
6. In each iteration of the loop, you evaluate the error term. If the error term is greater than or equal to zero, you adjust the y-coordinate (move vertically) and subtract Δy from the error term. If the error term is less than zero, you adjust the x-coordinate (move horizontally) and add Δx to the error term.
7. Continue this loop until you reach the second point, updating the x and y coordinates based on the error term and the direction of the line.

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);
}
int sign(int x)
{
    return (x > 0) - (x < 0);
}
void bresenham(int x1, int y1, int x2, int y2)
{
    int dx, dy, i, e;
    int incx, incy, inc1, inc2;
    int x, y;
    dx = abs(x2 - x1);
    dy = abs(y2 - y1);
    incx = x2 < x1 ? -1 : 1;
    incy = y2 < y1 ? -1 : 1;
    x = x1;
    y = y1;
    if (dx > dy)
    {
        draw_pixel(x, y);
        e = 2 * dy - dx;
        inc1 = 2 * (dy - dx);
        inc2 = 2 * dy;
        for (i = 0; i < dx; i++)
        {
            if (e >= 0)
            {
                y += incy;
                e += inc1;
            }
            else
                e += inc2;
            x += incx;
            draw_pixel(x, y);
```

```
        }
    }
    else
    {
        draw_pixel(x, y);
        e = 2 * dx - dy;
        inc1 = 2 * (dx - dy);
        inc2 = 2 * dx;
        for (i = 0; i < dy; i++)
        {
            if (e >= 0)
            {
                x += incx;
                e += inc1;
            }
            else
                e += inc2;
            y += incy;
            draw_pixel(x, y);
        }
    }
}
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);

    bresenham(0, 0, 400, 200);
    output(420, 200, "(400,200)");
    bresenham(0, 0, 200, 400);
    output(220, 400, "(200,400)");
    bresenham(-400, 200, 0, 0);
    output(-420, 220, "(-400,200)");
    bresenham(-200, 400, 0, 0);
    output(-220, 400, "(-200,400)");

    bresenham(0, 0, 400, -200);
    output(420, -200, "(400,-200)");
    bresenham(0, 0, 200, -400);
    output(220, -400, "(200,-400)");
    bresenham(-400, -200, 0, 0);
    output(-420, -200, "(-400,-200)");
    bresenham(-200, -400, 0, 0);
```

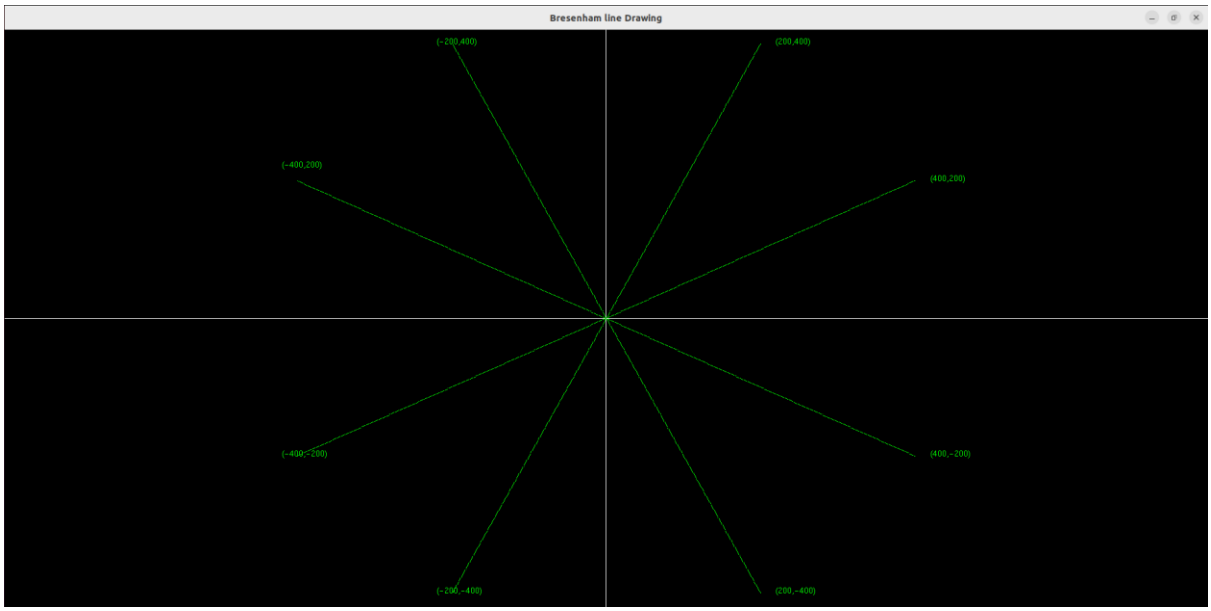
```
    output(-220, -400, "(-200,-400)");

    glFlush();
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1366, 768);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Bresenham line Drawing");
    myInit();
    glutDisplayFunc(display);
    glutMainLoop();
    return 1;
}
```

run.sh:

```
g++ 3.cpp -lGL -lglut -lGLU
./a.out
```

Sample I/O:



Learning Outcomes:

Thus, Bresenham's line drawing algorithm has been implemented with OpenGL and GLUT frameworks.