Lab Report

Course Code: CSE 422

Course Title: Computer Graphics Lab

Report No: 04

Title: Implementation of DDA line drawing algorithm.

Submitted To:

Syada Tasmia Alvi

Senior Lecturer

Department of Computer Science and Engineering (C.S.E)

Submitted by:

Name: Mahmudul Hasan Piash

ID: 221-15-5606

Section: 61_D1

Department of Computer Science and Engineering (C.S.E)

Date of submission: 28th November, 2025

Title

Implementation of DDA line drawing algorithm.

Introduction

In this lab task, I implemented the DDA (Digital Differential Analyzer) Line Drawing Algorithm using OpenGL and GLUT. The objective of the experiment was to draw a straight line between two user-defined points by incrementally calculating intermediate positions. This program demonstrates how computer graphics systems generate lines using floating-point arithmetic and coordinate-based plotting. Through this task, we learned how to initialize an OpenGL window, take input from the user, apply the DDA algorithm, and use OpenGL functions to visualize the computed line. This experiment strengthens the understanding of line rendering concepts and how digital devices convert mathematical line equations into pixels on the screen.

Contents

In this lab task :

1. Functions used
   - glClear() – clears the screen
   - glColor3f() – sets the drawing color
   - glBegin() / glEnd() – starts and ends drawing operations
   - glVertex2i() – plots individual points of the line
   - glFlush() – displays the rendered output
   - drawLineDDA() – implements the DDA algorithm to draw a line by calculating intermediate pixel coordinates.
2. Input Taken:
   - Starting point (x_0, y_0)
   - Ending point (x_1, y_1)

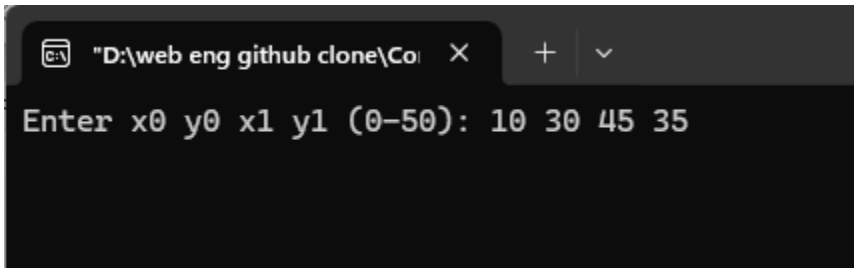
Code

```
#include <GL/gl.h>

#include <GL/glut.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>


int x0_user, y0_user, x1_user, y1_user;
```

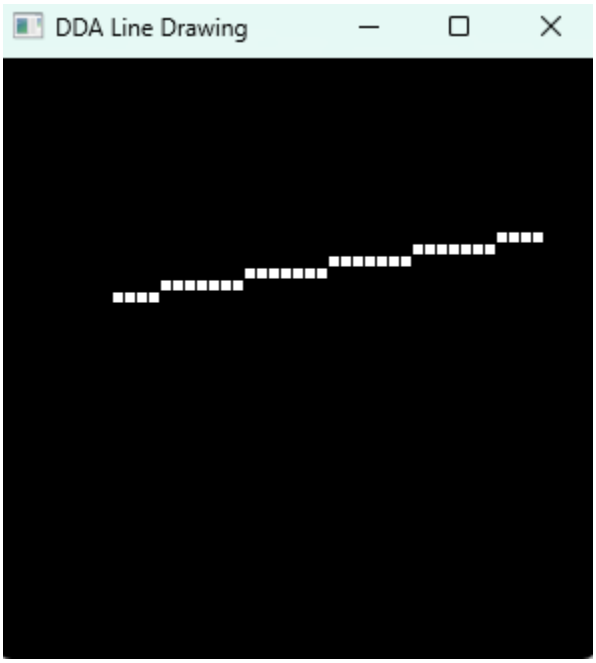
```
void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   gluOrtho2D(0, 50, 0, 50);
   glPointSize(5.0);
}

void drawLineDDA(int x0, int y0, int x1, int y1)
{
   float dx = x1 - x0;
   float dy = y1 - y0;
   float steps = fabsf(dx) > fabsf(dy) ? fabsf(dx) : fabsf(dy);
   float xInc = dx / steps;
   float yInc = dy / steps;
   float x = x0;
   float y = y0;

   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_POINTS);
   for (int k = 0; k <= steps; k++)
   {
      glVertex2i((int)roundf(x), (int)roundf(y));
      x += xInc;
      y += yInc;
   }
   glEnd();
```

```
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    drawLineDDA(x0_user, y0_user, x1_user, y1_user);
    glFlush();
}

int main(int argc, char** argv)
{
    printf("Enter x0 y0 x1 y1 (0-50): ");
    scanf("%d %d %d %d", &x0_user, &y0_user, &x1_user, &y1_user);

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(300, 300);
    glutCreateWindow("DDA Line Drawing");

    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Input



Output



Discussion

In this lab task, I successfully implemented the DDA Line Drawing Algorithm using OpenGL. The program reads two endpoints from the user and calculates intermediate points along the line using incremental floating-point steps in both x and y directions. Each calculated point is plotted using glVertex2i(), and glColor3f() is used to set a clear visual color for the line. The implementation demonstrates how DDA performs step-by-step pixel plotting based on the slope of the line without relying on complex arithmetic. Overall, this task enhanced my understanding of how lines are generated in computer graphics and how OpenGL can be used to visualize algorithmic output efficiently.