**SCHOOL OF COMPUTER SCIENCE**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**COMPUTER GRAPHICS**

**LABORATORY FILE**

**(2024-2025)**

**For**

**Vth Semester**

**Submitted To: Submitted By:**

Mr. Dinesh Akshat Negi

Assistant Professor 500106533(SAP ID)

[Vth Semester] R2142220414(Roll No.)

School of Computer Science B.Tech. CSF (Batch-1)

**LAB EXPERIMENT – 2**

**DRAWING A LINE**

**[Usage of Open GL]**

***# Take the input from user for all the three scenarios i.e. value of (x1, y1) and (x2, y2).***

1. **Draw a line using equation of line Y=m\*X+C.**

#include <GL/freeglut.h>

float m = 2.0f;

float C = 1.0f;

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

for (float x = -1.0f; x <= 1.0f; x += 0.01f)

{

float y = m \* x + C;

glVertex2f(x, y);

}

glEnd();

glFlush();

}

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glColor3f(1.0, 1.0, 1.0);

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Line: Y = mX + C --- Akshat Negi");

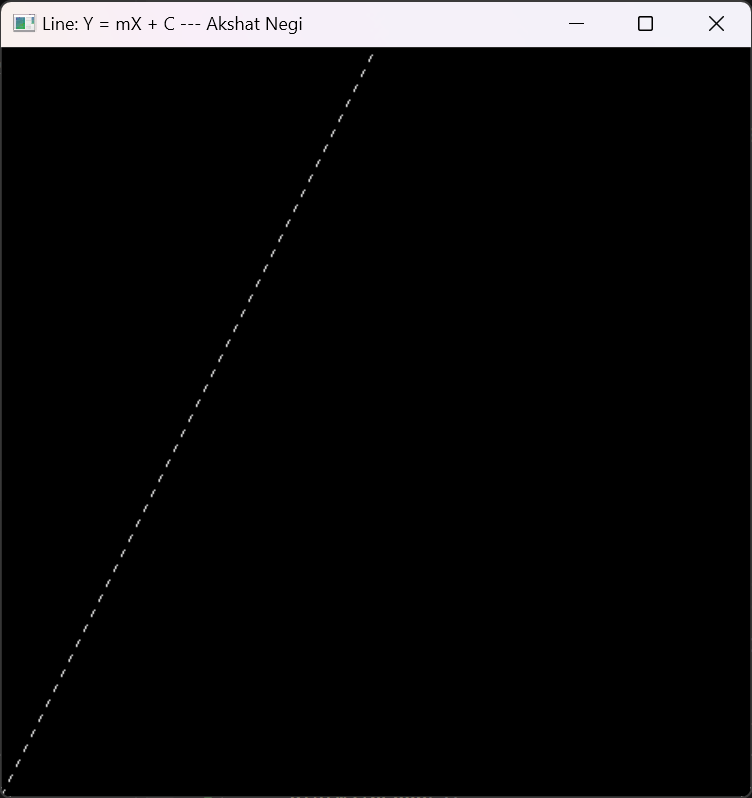
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

****

1. **Draw a line using DDA algorithm for slope m<1 and m>1.**

#include <GL/freeglut.h>

#include <iostream>

#include <cmath>

using namespace std; // Function to plot points

void plot(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

} // DDA Line Drawing Algorithm

void DDA(int x1, int y1, int x2, int y2) {

int dx = x2 - x1;

int dy = y2 - y1;

int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy); // Maximum steps

float xIncrement = dx / (float)steps;

float yIncrement = dy / (float)steps;

float x = x1;

float y = y1; // Draw the line by plotting points

for (int i = 0; i <= steps; i++) {

plot(round(x), round(y));

x += xIncrement;

y += yIncrement;

}

}// Function to get input from the user and call DDA

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

int x1, y1, x2, y2;

cout << "Enter the coordinates of the first point (x1, y1): ";

cin >> x1 >> y1;

cout << "Enter the coordinates of the second point (x2, y2): ";

cin >> x2 >> y2;

DDA(x1, y1, x2, y2);

} // Initialize the OpenGL Graphics

void init() {

glClearColor(1.0, 1.0, 1.0, 0.0); // Background color

glColor3f(0.0, 0.0, 0.0); // Drawing color

glPointSize(2.0); // Point size

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0); // Define the drawing area

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500); // Window size

glutInitWindowPosition(100, 100); // Window position

glutCreateWindow("DDA Line Drawing Algorithm --> Akshat Negi");

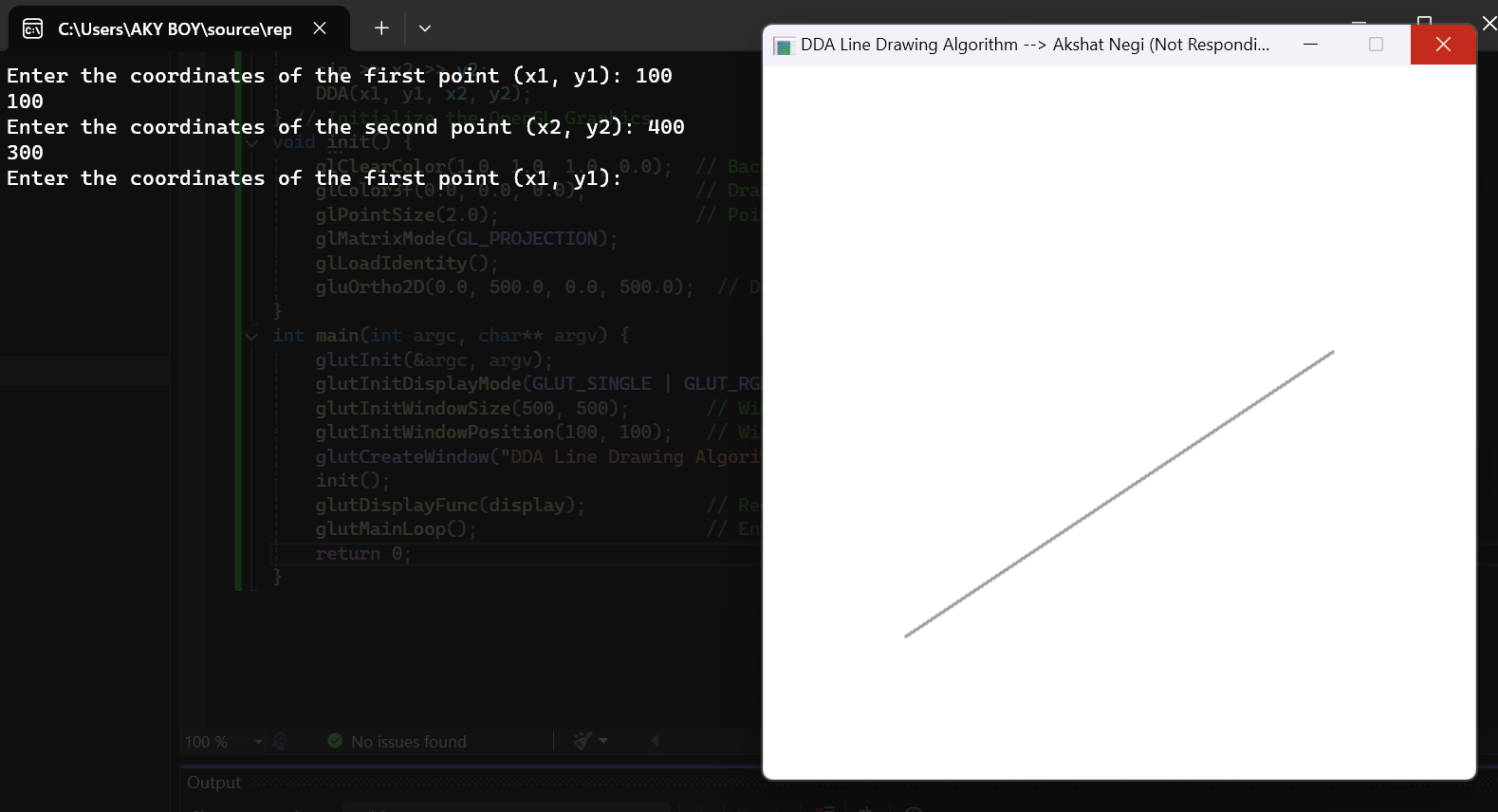
init();

glutDisplayFunc(display); // Register display function

glutMainLoop(); // Enter the event-processing loop

return 0;

}

****

1. **Draw a line using Bresenham algorithm for slope m<1 and m>1.**

#include <GL/freeglut.h>

#include <stdio.h> // Function to set pixel at (x, y)

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

} // Bresenham's algorithm for slope |m| < 1 (dy < dx)

void bresenhamLineLow(int x1, int y1, int x2, int y2) {

int dx = x2 - x1;

int dy = y2 - y1;

int D = 2 \* dy - dx;

int y = y1;

for (int x = x1; x <= x2; x++) {

setPixel(x, y);

if (D > 0) {

y += (y2 > y1) ? 1 : -1; // Increase/decrease y depending on the slope direction

D = D + (2 \* (dy - dx));

}

else {

D = D + 2 \* dy;

}

}

}// Bresenham's algorithm for slope |m| > 1 (dy > dx)

void bresenhamLineHigh(int x1, int y1, int x2, int y2) {

int dx = x2 - x1;

int dy = y2 - y1;

int D = 2 \* dx - dy;

int x = x1;

for (int y = y1; y <= y2; y++) {

setPixel(x, y);

if (D > 0) {

x += (x2 > x1) ? 1 : -1; // Increase/decrease x depending on the slope direction

D = D + (2 \* (dx - dy));

}

else {

D = D + 2 \* dx;

}

}

} // Main function that checks the slope and calls the appropriate function

void drawLine(int x1, int y1, int x2, int y2) {

if (abs(y2 - y1) < abs(x2 - x1)) {

if (x1 > x2) {

bresenhamLineLow(x2, y2, x1, y1); // Line from (x2, y2) to (x1, y1)

}

else {

bresenhamLineLow(x1, y1, x2, y2); // Line from (x1, y1) to (x2, y2)

}

}

else {

if (y1 > y2) {

bresenhamLineHigh(x2, y2, x1, y1); // Line from (x2, y2) to (x1, y1)

}

else {

bresenhamLineHigh(x1, y1, x2, y2); // Line from (x1, y1) to (x2, y2)

}

}

} // User input handling and initialization

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

int x1, y1, x2, y2;

printf("Enter coordinates of the first point (x1, y1): ");

scanf\_s("%d %d", &x1, &y1);

printf("Enter coordinates of the second point (x2, y2): ");

scanf\_s("%d %d", &x2, &y2);

drawLine(x1, y1, x2, y2);

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glColor3f(0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 500, 0, 500); // Set the orthographic projection

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Bresenham's Line Algorithm --> Akshat Negi");

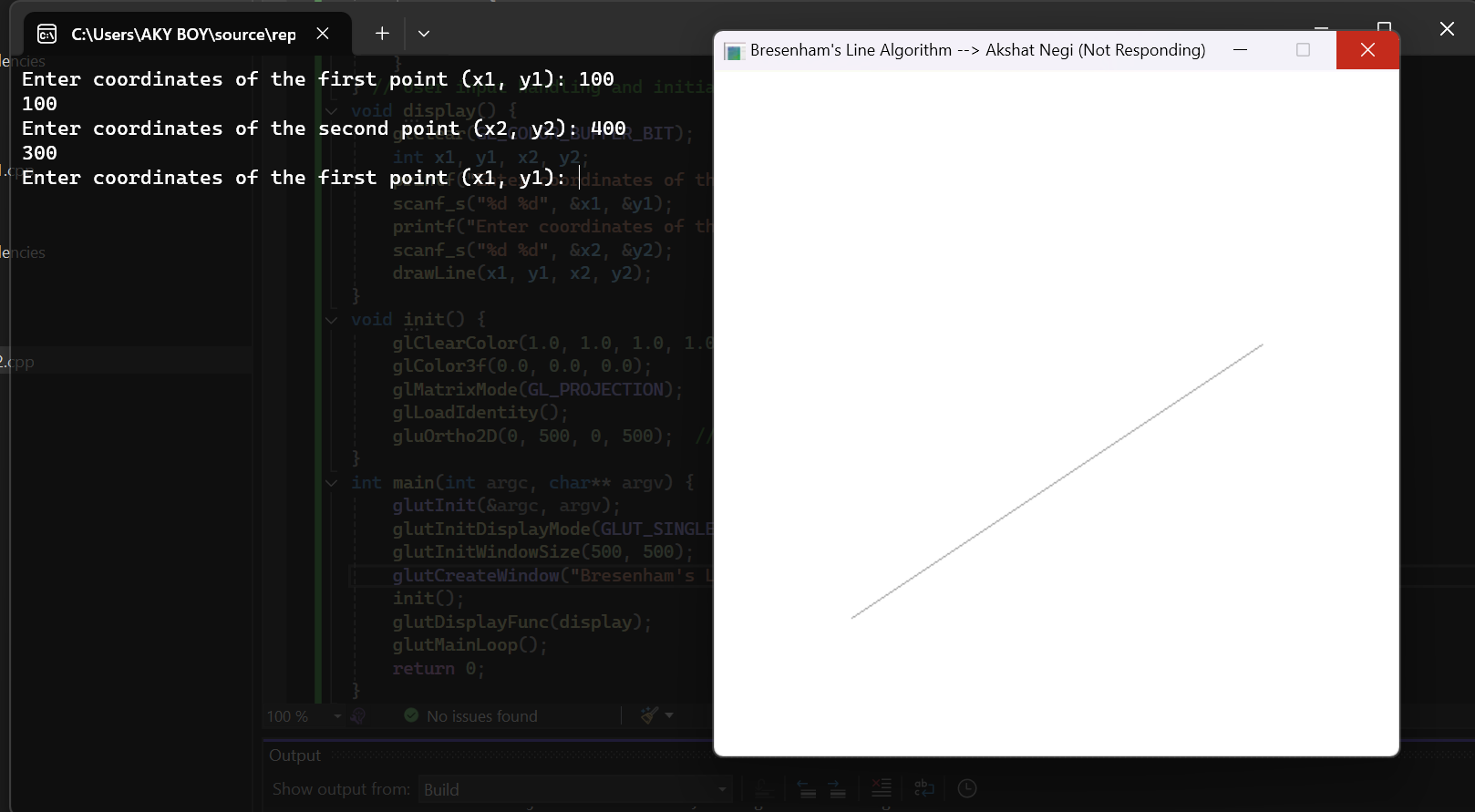
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

****