#include <iostream>

#include <GL/glut.h>

#include <cmath>

using namespace std;

int x, y, x1, x2, y1\_new, y2, dx, dy;

float m;

// Function to draw a single pixel

void drawPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

// Function to draw a continuous line using DDA Algorithm

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

dx = x2 - x1;

dy = y2 - y1;

m = dy / (float)dx;

if (fabs(m) < 1) {

float x\_incr = 1;

float y\_incr = m;

float y = y1;

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

drawPixel(round(x), round(y));

y += y\_incr;

}

} else {

float x\_incr = 1 / m;

float x = x1;

float y\_incr = m;

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

drawPixel(round(x), round(y));

x += x\_incr;

}

}

}

// Function to draw a dotted line using DDA Algorithm

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

dx = x2 - x1;

dy = y2 - y1;

m = dy / (float)dx;

int dotSize = 2; // Size of the dots

int gapSize = 4; // Size of the gaps between dots

if (fabs(m) < 1) {

float x\_incr = 1;

float y\_incr = m;

float y = y1;

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

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

drawPixel(round(x), round(y));

}

y += y\_incr \* dotSize + gapSize;

}

} else {

float x\_incr = 1 / m;

float x = x1;

float y\_incr = m;

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

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

drawPixel(round(x), round(y));

}

x += x\_incr \* dotSize + gapSize;

}

}

}

// Function to draw a dashed line using DDA Algorithm

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

dx = x2 - x1;

dy = y2 - y1;

m = dy / (float)dx;

int dashSize = 8; // Size of the dashes

int gapSize = 4; // Size of the gaps between dashes

if (fabs(m) < 1) {

float x\_incr = 1;

float y\_incr = m;

float y = y1;

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

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

drawPixel(round(x), round(y));

}

y += y\_incr \* dashSize + gapSize;

}

} else {

float x\_incr = 1 / m;

float x = x1;

float y\_incr = m;

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

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

drawPixel(round(x), round(y));

}

x += x\_incr \* dashSize + gapSize;

}

}

}

void init(void) {

glClearColor(0.7, 0.7, 0.7, 0.7);

gluOrtho2D(-100, 100, -100, 100);

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.1, 0.0);

glPointSize(2.0f);

// Draw the line based on the selected line type

int lineType;

cout << "Select line type (0: Continuous, 1: Dotted, 2: Dashed): ";

cin >> lineType;

switch (lineType) {

case 0: // Continuous line

drawContinuousLineDDA(x1, y1\_new, x2, y2);

break;

case 1: // Dotted line

drawDottedLineDDA(x1, y1\_new, x2, y2);

break;

case 2: // Dashed line

drawDashedLineDDA(x1, y1\_new, x2, y2);

break;

default:

cout << "Invalid line type.";

break;

}

glFlush();

}

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

cout << "\n : Enter X1,Y1,X2,Y2 :\n";

cout << "x1: ";

cin >> x1;

cout << "Y1: ";

cin >> y1\_new;

cout << "X2: ";

cin >> x2;

cout << "Y2: ";

cin >> y2;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("DDA Line Algorithm");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}