#include <GL/glut.h> #include <iostream> #include <cmath>

#define h 1500

#define w 1500

using namespace std;

// Function to draw a pixel at (x, y)

void drawPixel(int x, int y) { glBegin(GL\_POINTS); glVertex2i(x, y);

glEnd();

}

// Function to determine the sign of a number int sign(int x) {

if (x > 0) return 1;

if (x < 0) return -1;

return 0;

}

// Bresenham's Line Drawing Algorithm with Pattern Support void drawLineB(int x1, int y1, int x2, int y2, int flag) {

int x = x1, y = y1;

int dx = abs(x2 - x1); int dy = abs(y2 - y1); int interchange;

int s1 = sign(x2 - x1); int s2 = sign(y2 - y1);

if (dy > dx) {

int temp = dx; dx = dy;

dy = temp;

interchange = 1;

} else {

interchange = 0;

}

int e = 2 \* dy - dx;

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

// Pattern logic

if (flag == 1) { // Solid Line drawPixel(x, y);

}

else if (flag == 2) { // Dotted Line if (i % 10 == 0) {

drawPixel(x, y);

}

}

else if (flag == 3) { // Dashed Line

int dashLength = 35; // Longer dash int gapLength = 35; // Bigger gap

int patternLength = dashLength + gapLength;

if (i % patternLength < dashLength) { drawPixel(round(x), round(y));

}

}

else if (flag == 4) { // Center-Dotted Line: Dash - Space - Dot - Space - Dash int patternLength = 140;

int dashLength = 35; int dotPosition = 70;

int mod = i % patternLength;

if(mod < dashLength || mod == dotPosition){ drawPixel(round(x), round(y));

}

}

// Bresenham's algorithm logic

while (e >= 0) {

if (interchange == 1) { x = x + s1;

} else {

y = y + s2;

}

e = e - 2 \* dx;

}

if (interchange == 1) { y = y + s2;

} else {

x = x + s1;

}

e = e + 2 \* dy;

}

}

// Function to draw a house

// Function to draw the ⊕ symbol in the 4th quadrant void drawGeometricPattern() {

glColor3f(1.0, 1.0, 1.0);

// Draw the square drawLineB(600,-100,1200,-100, 1);

drawLineB(1200,-100,1200,-700, 1);

drawLineB(1200, -700, 600, -700, 1);

drawLineB(600,-700, 600, -100, 1);

drawLineB(600,-400,900,-100, 1);

drawLineB(900,-100,1200,-400, 1);

drawLineB(1200, -400, 900, -700, 1);

drawLineB(900,-700, 600, -400, 1);

drawLineB(750,-250,1050,-250, 1);

drawLineB(1050,-250,1050,-550, 1);

drawLineB(1050, -550, 750, -550, 1);

drawLineB(750,-550, 750, -250, 1);

cout << "Geometric pattern (⊕) drawn successfully in the 4th quadrant!" << endl;

glColor3f(1.0, 1.0, 1.0);

}

void plotAxis() {

glBegin(GL\_POINTS);

for (int i = -h; i <= h; i++) {

glVertex2i(0, i); // Vertical axis glVertex2i(i, 0); // Horizontal axis

}

glEnd(); glFlush();

}

// Display callback function void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); plotAxis();

glFlush();

}

void menu(int index) {

int x1, y1, x2, y2; if (index != 5) {

cout << "Enter the coordinates of the starting point: "; cin >> x1 >> y1;

cout << "Enter the coordinates of the ending point: "; cin >> x2 >> y2;

if (x1 == x2 && y1 == y2) {

cout << "Co-ordinates cannot be the same!"; cout.flush();

return;

}

}

if (index == 1) {

drawLineB(x1, y1, x2, y2, 1); // Solid Line

}

else if (index == 2) {

drawLineB(x1, y1, x2, y2, 2); // Dotted Line

}

else if (index == 3) {

drawLineB(x1, y1, x2, y2, 3); // Dashed Line

}

else if (index == 4) {

drawLineB(x1, y1, x2, y2, 4); // Center-Dotted Line

}

else if (index == 5) {

drawGeometricPattern(); // Draw geometric pattern

}

}

// Initialize OpenGL environment void init() {

glClearColor(0.0, 0.0, 0.0, 1.0);

glColor3f(1.0, 1.0, 1.0); glMatrixMode(GL\_PROJECTION); glLoadIdentity();

gluOrtho2D(-w, w, -h, h);

}

int main(int argc, char\*\* argv) { glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); glutInitWindowSize(1000, 800);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bresenham's Line Drawing Algorithm with Patterns");

glutCreateMenu(menu); glutAddMenuEntry("Solid Line", 1);

glutAddMenuEntry("Dotted Line", 2);

glutAddMenuEntry("Dashed Line", 3);

glutAddMenuEntry("Center-Dotted Line", 4);

glutAddMenuEntry("Geometric Pattern", 5); glutAttachMenu(GLUT\_RIGHT\_BUTTON);

init(); glutDisplayFunc(display); glutMainLoop();

return 0;

}