#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();

}

// DDA Line Drawing Algorithm with Different Patterns void drawLineDDA(int x1, int y1, int x2, int y2, int flag) {

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

cout << "Coordinates cannot be the same!"; return;

}

int dx = x2 - x1; int dy = y2 - y1;

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

steps = (int)steps;

float xInc = dx / (float)steps; float yInc = dy / (float)steps; float x = x1, y = y1;

for (int i = 0; i <= steps; i++) { if (flag == 1) { // Solid Line

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

}

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

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

}

} x += xInc;

y += yInc;

}

}

// Function to draw an improved boat void drawBoat() {

// Base of the boat (Curved bottom)

drawLineDDA(300, -300, 750, -300, 1); // Top horizontal base

drawLineDDA(300, -300, 250, -350, 1); // Left Slant

drawLineDDA(750, -300, 800, -350, 1); // Right Slant

drawLineDDA(250, -350, 525, -450, 1); // Left Curve

drawLineDDA(800, -350, 525, -450, 1); // Right Curve

// Mast

drawLineDDA(525, -300, 525, -100, 1); // Vertical mast

// Flag

drawLineDDA(525, -100, 600, -150, 1); // Upper flag diagonal

drawLineDDA(600, -150, 525, -200, 1); // Lower flag diagonal

// Deck (Upper part of the boat)

drawLineDDA(400, -275, 650, -275, 1); // Small horizontal line (deck top)

drawLineDDA(400, -275, 350, -300, 1); // Left connection

drawLineDDA(650, -275, 700, -300, 1); // Right connection

// Small circular windows on the boat drawPixel(450, -320);

drawPixel(500, -320);

drawPixel(550, -320);

drawPixel(600, -320);

cout << "Boat drawn successfully!" << endl;

}

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) {

drawLineDDA(x1, y1, x2, y2, 1);

}

else if (index == 2) { drawLineDDA(x1, y1, x2, y2, 2);

}

else if (index == 3) { drawLineDDA(x1, y1, x2, y2, 3);

}

else if (index == 4) { drawLineDDA(x1, y1, x2, y2, 4);

}

else if (index == 5) { drawBoat();

}

}

// Initialize OpenGL environment void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glColor3f(0.0, 0.0, 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("DDA Line Drawing Algorithm"); glutCreateMenu(menu);

glutAddMenuEntry("Simple Line", 1);

glutAddMenuEntry("Dotted Line", 2);

glutAddMenuEntry("Dashed Line", 3);

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

glutAddMenuEntry("Boat", 5);

glutAttachMenu(GLUT\_RIGHT\_BUTTON); init();

glutDisplayFunc(display); glutMainLoop();

return 0;

}