CGPractcal2

```
#include <GL/glut.h>
#include <iostream>
#include <vector>
using namespace std;

struct Point {
    int x, y;
};

vector<Point> points;
int lineType = 1; // 1: Simple, 2: Dotted, 3: Dashed, 4: Solid

void setPixel(int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
    glFlush();
}

// DDA Line Drawing Algorithm
void DDA(Point p1, Point p2) {
    int dx = p2.x - p1.x, dy = p2.y - p1.y;
    int steps = max(abs(dx), abs(dy));
    float xInc = dx / (float)steps;
    float yInc = dy / (float)steps;
    float x = p1.x, y = p1.y;

    for (int i = 0; i <= steps; i++) {
        if (lineType == 1 ||                    // Simple
            (lineType == 2 && i % 5 == 0) ||      // Dotted
            (lineType == 3 && i % 10 < 5) ||      // Dashed
            lineType == 4)                        // Solid
        {
            setPixel(round(x), round(y));
        }
        x += xInc;
        y += yInc;
    }
}

// Bresenham's Line Algorithm (Alternative, not used in drawLine by default)
void Bresenham(Point p1, Point p2) {
    int dx = abs(p2.x - p1.x);
    int dy = abs(p2.y - p1.y);
    int sx = (p1.x < p2.x) ? 1 : -1;
    int sy = (p1.y < p2.y) ? 1 : -1;
    int err = dx - dy;
    int count = 0;

    while (true) {
        if (lineType == 1 ||
            (lineType == 2 && count % 5 == 0) ||
            (lineType == 3 && count % 10 < 5) ||
            lineType == 4)
        {
            setPixel(p1.x, p1.y);
        }

        if (p1.x == p2.x && p1.y == p2.y) break;

        int e2 = 2 * err;
        if (e2 > -dy) { err -= dy; p1.x += sx; }
        if (e2 < dx) { err += dx; p1.y += sy; }

        count++;
    }
}

// Draw line after selecting 2 points
void drawLine() {
    glColor3f(1.0, 1.0, 1.0);
    if (points.size() == 2) {
        DDA(points[0], points[1]);
        // Bresenham(points[0], points[1]); // Uncomment to use Bresenham instead
        points.clear();
    }
}

// Mouse click to collect points
void mouse(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        points.push_back({x - 250, 250 - y});
        if (points.size() == 2) drawLine();
    }
}
```

```
// Draw X and Y axes
void display() {
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0.0, 1.0, 0.0); // Green axes
    glBegin(GL_LINES);
        glVertex2i(-250, 0); glVertex2i(250, 0); // X-axis
        glVertex2i(0, -250); glVertex2i(0, 250); // Y-axis
    glEnd();

    glFlush();
}

// Set up 2D coordinate system
void init() {
    glClearColor(0.0, 0.0, 0.0, 1.0);  // Black background
    gluOrtho2D(-250, 250, -250, 250); // Coordinate system
}

// Keyboard input for line style
void keyboard(unsigned char key, int, int) {
    switch (key) {
        case '1': lineType = 1; break; // Simple
        case '2': lineType = 2; break; // Dotted
        case '3': lineType = 3; break; // Dashed
        case '4': lineType = 4; break; // Solid
    }
    display();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("DDA Line Drawing with Styles");
    init();
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```