#include <SFML/Graphics.hpp>

#include <iostream>

#include <vector>

#include <cmath>

#include <limits>

#include <sstream>

#include <iomanip>

struct Point {

float x, y;

Point(float xCoord, float yCoord) : x(xCoord), y(yCoord) {}

};

float distance(const Point& a, const Point& b) {

return std::sqrt((a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y));

}

struct Edge {

int u, v;

float dist;

Edge(int u, int v, float dist) : u(u), v(v), dist(dist) {}

bool operator < (const Edge& e) const {

return dist < e.dist;

}

};

void primMST(std::vector<Point>& points, std::vector<Edge>& result) {

int n = points.size();

std::vector<float> min\_e(n, std::numeric\_limits<float>::max());

std::vector<int> sel\_e(n, -1);

std::vector<bool> used(n, false);

min\_e[0] = 0;

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

int v = -1;

for (int j = 0; j < n; ++j) {

if (!used[j] && (v == -1 || min\_e[j] < min\_e[v]))

v = j;

}

used[v] = true;

if (sel\_e[v] != -1)

result.push\_back(Edge(sel\_e[v], v, min\_e[v]));

for (int to = 0; to < n; ++to) {

float dist = distance(points[v], points[to]);

if (dist < min\_e[to] && !used[to])

{

min\_e[to] = dist;

sel\_e[to] = v;

}

}

}

}

int main() {

sf::RenderWindow window(sf::VideoMode(800, 600), "Network Optimization");

sf::Font font;

if (!font.loadFromFile("C:\\Windows\\Fonts\\arial.ttf"))

{

std::cout << "Could not load font\n";

return 1;

}

std::vector<Point> points;

std::vector<Edge> edges;

std::string inputBuffer;

sf::Text text("", font, 20);

text.setFillColor(sf::Color::White);

bool inputComplete = false;

float totalCableLength = 0.0f;

sf::Text totalCableText("", font, 20);

totalCableText.setFillColor(sf::Color::White);

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event)) {

if (event.type == sf::Event::Closed)

window.close();

else if (event.type == sf::Event::TextEntered) {

if (event.text.unicode == '\b') {

if (!inputBuffer.empty()) {

inputBuffer.pop\_back();

}

}

else if (event.text.unicode < 128) {

inputBuffer += static\_cast<char>(event.text.unicode);

}

}

else if (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::Enter) {

std::istringstream ss(inputBuffer);

int count;

ss >> count;

float x, y;

points.clear();

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

ss >> x >> y;

points.emplace\_back(x, y);

}

inputComplete = true;

edges.clear();

primMST(points, edges);

totalCableLength = 0.0f;

for (const auto& edge : edges) {

totalCableLength += edge.dist + 16.0f;

}

std::stringstream totalCableStream;

totalCableStream << "Total Cable Length: " << std::fixed << std::setprecision(2) << totalCableLength << " ft";

totalCableText.setString(totalCableStream.str());

float centerX = 0, centerY = 0;

for (const auto& point : points) {

centerX += point.x;

centerY += point.y;

}

centerX /= points.size();

centerY /= points.size();

sf::View view(sf::Vector2f(centerX, centerY), sf::Vector2f(800, 600));

window.setView(view);

inputBuffer.clear();

text.setString("");

}

}

window.clear(sf::Color::Black);

for (const auto& edge : edges)

{

sf::Vertex line[] = {

sf::Vertex(sf::Vector2f(points[edge.u].x, points[edge.u].y), sf::Color::Red),

sf::Vertex(sf::Vector2f(points[edge.v].x, points[edge.v].y), sf::Color::Red)

};

window.draw(line, 2, sf::Lines);

std::stringstream ss;

ss << std::fixed << std::setprecision(2) << edge.dist;

sf::Text distText(ss.str(), font, 15);

distText.setPosition((points[edge.u].x + points[edge.v].x) / 2, (points[edge.u].y + points[edge.v].y) / 2);

distText.setFillColor(sf::Color::Cyan);

window.draw(distText);

}

for (const auto& point : points)

{

sf::CircleShape circle(5);

circle.setPosition(point.x - 5, point.y - 5);

circle.setFillColor(sf::Color::Green);

window.draw(circle);

std::stringstream ss;

ss << "(" << point.x << ", " << point.y << ")";

sf::Text coordText(ss.str(), font, 15);

coordText.setPosition(point.x + 10, point.y - 5);

coordText.setFillColor(sf::Color::White);

window.draw(coordText);

}

text.setString(inputComplete ? "" : "Enter points: " + inputBuffer);

text.setPosition(10, 10);

if (!inputComplete)

{

window.draw(text);

}

totalCableText.setPosition(window.getSize().x - totalCableText.getLocalBounds().width - 10, 10);

window.draw(totalCableText);

window.display();

}

return 0;

}