Advanced algorithms mini-project

Benoît Piranda

1 Introduction

1.1 The initial code

The initial code is available on GitHub at the following address:
https://github.com/docben/DroneAndRooms.git.
Clone the source on your system and run the program. You should see the following result (Figure 1), in which we can identify:

- The servers marked by a circles in the window and associated with the name of a town.
- The drones that are marked by a drone icon in the window and associated with a surname.

The JSON file "*simple.json*" is provided in the "*json*" directory. The file contains all the informations about server and drone objets.
Each `server` has a name, a position and a color for its area. An id will be automatically added; it is the server's position in the list starting from 0.
Each `drone` has a name, an initial position, and a target name corresponding to a server name.

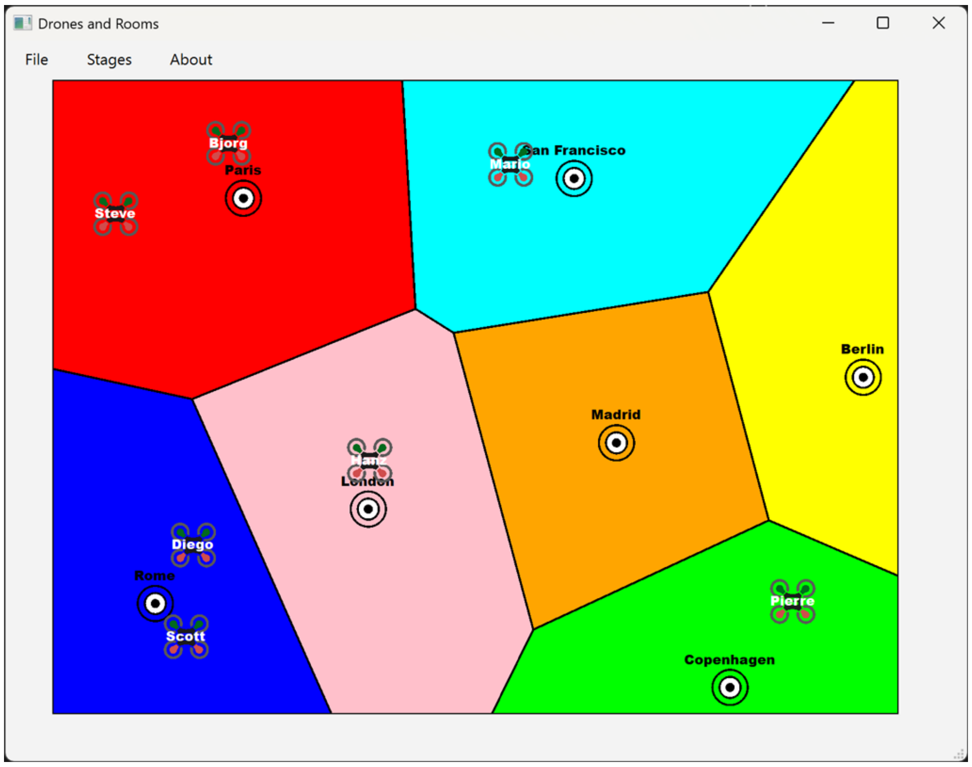

Figure 1: Initial window

As we can see in the Figure 1, the area is divided into colored regions (areas) defined by the server position using the Voronoi algorithm.

1.2 The interface

The interface is based on the definition of a `MainWindow` containing an instance of `Canvas`. This object stores and displays all the data (the lists of `servers`, `drones` and `links` that define the graph. The main menu consists of:

- `File/Open` load a ”*.json*” file. Two examples are available in the ”*json*” directory.
- `File/Quit` is for closing the application.
- `Stages/Show graph` Toggle the drawing of the links betwwen servers (if `links` is fill.
- `Stages/Move Drones` Run the motions of all the drones according to the `destination`.
- `About/Credits` Informations about the software and the author.

2 Exercises

The purpose of these exercises is to complete the program so that each drone can find the server that corresponds to it. Each drone can only communicate with the server in the zone in which it is located and can only cross zones at the gates in the middle of the segments.

Exercise 1: Adding doors Doors are placed between two zones in the middle of a polygon segment. Their width is defined by the `doorWidth` constant, and they must be centered on the middle of the segment and aligned with the axis of the polygon edge.

Question 1: Write the mathematical expressions for the two positions Q_0 and Q_1 of the extremity of a door segment placed in the middle of the edge P_iP_{i+1}.

Question 2: Complete the code in the `Polygon::draw(...)` method to add the drawing of the doors. Use a white pen with a width of 7 for that. The figure 2 shows the result.

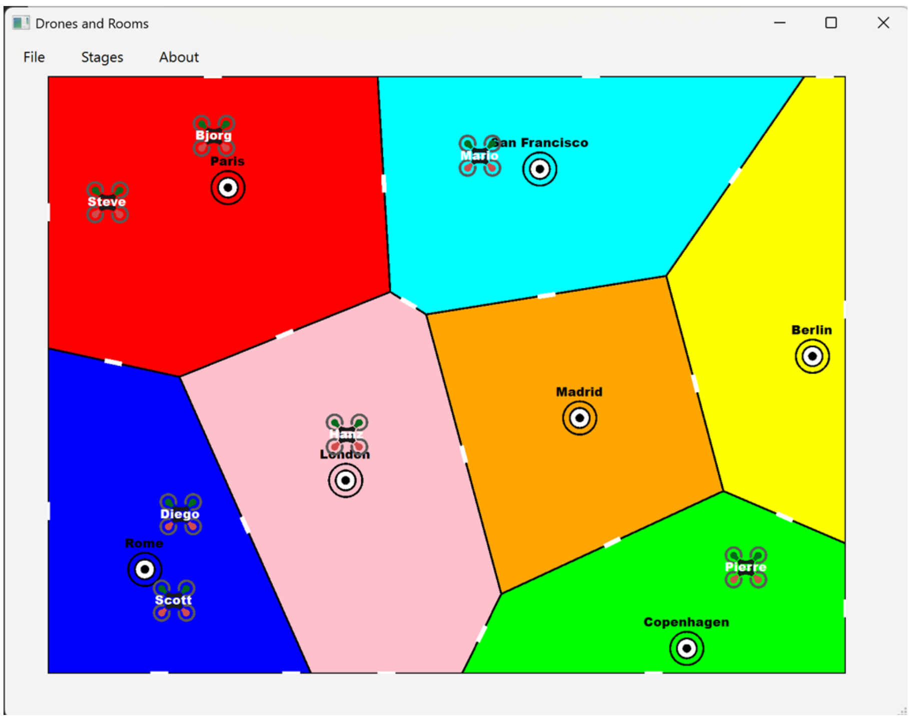

Figure 2: Doors in the middle of the segment

Exercise 2: Computation of the graph

We consider that servers can communicate with each other if they are in areas that share a common edge, such as Paris and London. We will therefore define a graph where the servers are the nodes of the graph. They are connected by links (`Link` object) that store the length of the link. This length represents the distance between the servers passing through the middle of the edge.

Question 1: In the `MainWindow::createServersLinks` method, create the graph connecting the servers in defining all the links between servers that are in two neighboring areas. Use defined polygons to search for the one with a common edge. Your result should be similar to the one proposed Figure 3.

Question 2: What is the complexity of your algorithm?

Question 3: The distance associated with a link is the length of the link as drawn in figure 3. It is the sum of the lengths of the segments from the server position to the middle of the shared edge plus the distance from that point to the second server position.
Propose an algorithm that computes all the minimum distance from each server to all the others. This distance is store in the array `server.bestDistance`. For example `server.bestDistance[0].second` stores the distance from the current server to the server number *id* = 0. And `server.bestDistance[0].first` is the link to follow to reach server 0 using the shortest path.

Question 4: Implement this algorithm in the `MainWindow::fillDistanceArray()` method.

Question 5: Print a table that gives all the computed distances.

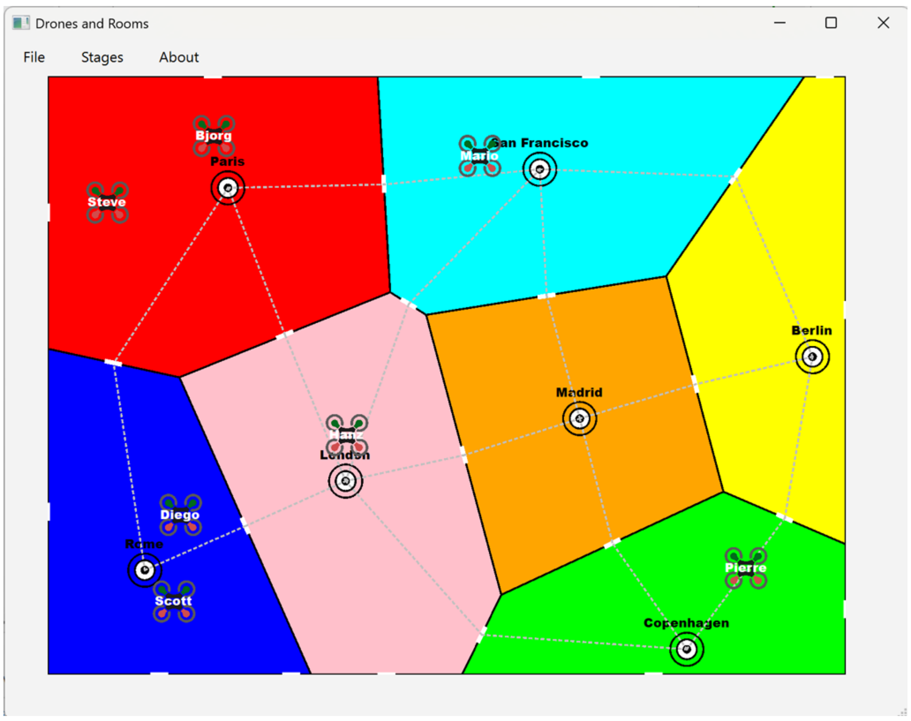

Figure 3: The graph of servers

Exercise 3: Animation of drones The motion rule of the drone is as follows:

1. First all the drones are associated with the server in the area where they are placed (use the `overflownArea` method).
2. The first motion consists in reaching the associated server by setting the `destination` to the server's position.
3. When a drone reaches a server, the server provides the shortest path to the target server by setting the `destination` attribute to the middle position of the edge crossed by the path.
4. When a drone reaches the center of an edge, its `destination` is the position of the server in the opposite area.

The video (https://youtu.be/cYU5AZrY9B4) shows an example of motions following these rules.

Question 1: Complete the method `Drone::move(dt)` to apply the motion rules. This function is call during the animation to update the position of the drone every *dt* seconds.

Question 2: Propose your own configuration (json file) and test your algorithm in the new case.