Informatics Large Practical

Coursework 2 Report

Haoshi Wang
December 3 2021

Contents

1 Software Architecture Description

1.1 Overview

This section will introduce all the java classes that integrated as a whole program, explaining their role and importance.

1.2 Aim

The aim of the program is to plan the flight path for a drone that let the drone collecting and delivery food while not fly cross the non-fly zone, once all orders is delivered, is should go back to the start position.

1.3 DatabaseFunction

The class DatabaseFunction is responsible for all interaction with the Derby database, including check connection with the database, get Order information, creating tables like Deliveries and Flightpath, insert data into table Deliveries and Flightpath.

1.4 Drone

This class is the most important class, it indicates an object that perform the task of delivery. It contains other classes objects that is used to store and get information like: orders need to be delivered, Non-fly Zone objects, Landmarks and etc. With this information, the Drone class's most important functionality 'Drone Control Algorithm' could plan the drone's flightpath with all requirements satisfy. When the program is started, we just need one object of Drone class, with its function 'start()', the flightpath could be output.

1.5 LandMarks

Class LandMarks is used to parse GeoJSON get from http server that stores the coordinates of land marks, using these land marks as Transfer station allows the drones wo avoid most situations that the flightpath will cross the non-fly zone, which is an important in the drone control algorithm.

1.6 LongLat

The class LongLat contain the Longitude and Latitude that indicating a position on the map, objects position information is stored as a LongLat object. It also contains methods to find distance, bearing angle between current LongLat object and another LongLat object, also able to using a direction angle to calculate the next position if a move is performed with this direction angle.

1.7 NonFlyZone

Class NonFlyZone is used to parse GeoJSON get from http server that stores the information of non-fly zone, the Polygon feature is parsed into list of coordinates so that they can be used to create line2D object, which are used to check weather the drone flight path will cross the non-fly zone. Checking if the path cross the non-fly zone is very important in drone control algorithm, the it is necessary to have NonFlyZone class.

1.8 Orders

The instance of class orders store details of an order, these details is obtained by request to the Derby Database. It will contain information like order's receiver position, items that order asked to buy, and order number. These information helps the drone control algorithm to find movement targets and to plan the most effective path, to obtain largest monetary value.

1.9 ServerConnection

ServerConnection has methods responsible for communicate with the http server, request the http serve to get JSONs and GeoJSONs, then parse these JSON file into their class instances, allow drone control algorithm to use these objects to plan the drone path.

1.10 Shop

The class shop is used to parse Json file menu.json that get from the http server, it holds location, name and menu of a shop.

1.11 W3W location

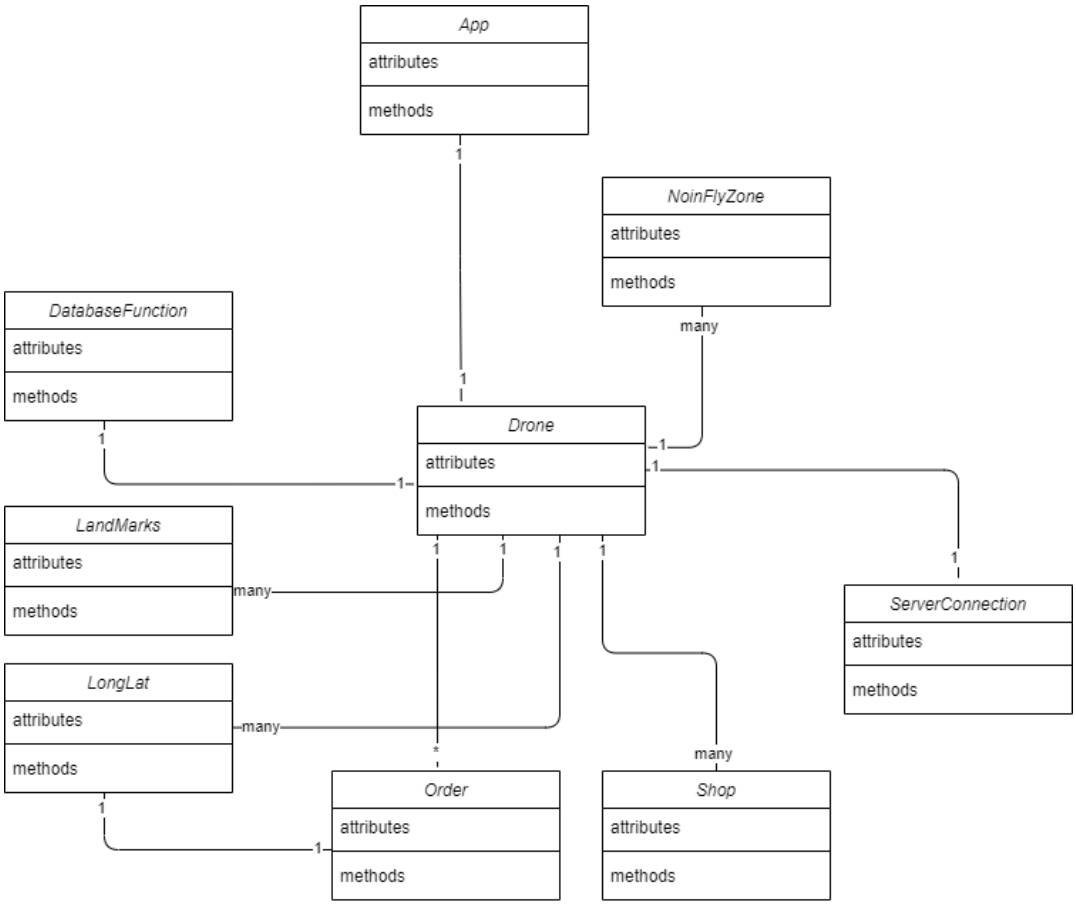
The class W3Wlocation is used to parse what 3 word address json file that get from the http server, allows drone to find the order's receiver's location, also find the shops location.

1.12 App

Class App contains the method main(), where the whole program integrate and run to provide a geojson file of the flightpath of a given day. It will receive parameters like http server port, database port and the date. It will create a Drone class object and execute method start(), then use function getAllPath() to get the planed path of the drone, write the path into geojson file.

1.13 Simple class diagram:

This is a simple class diagram that used to represent the relationship between class in the program, the class attributes and methods are omitted. Check Javadoc for details of class attributes and methods.



1.14 Third-Party Libraries

Following 3rd party libraries are used in the program to solve problems:

- Mapbox Java SDK GeoJSON, this is used handle and create GeoJSON features [1]
- Derbyclient, This is used to read and write data from derby database [2]
- Gson API, This is used to parse the JSON files to Java classes [3]

2 Drone Control Algorithm

In this section I will discuss the algorithm used to control the order in which the drone visits the sensors and other logical decisions the drone must make on its flight, along with 2 example flights with analyses.

2.1 The Algorithm Explanation

The general logic sequence of the Algorithm is:

1. Select the order to deliver next
2. Select path
3. Move And Record Path in database with selected path.
4. If not all order delivered or drone ran out of moves, back to step 1

In the following subsections from 2.1.1 to 2.1.5, I will explain each of the sequence of the drone control algorithm that mentioned above.

2.1.1 Select best order

The next order to deliver is selected with following steps of algorithm:

1. Loop though all orders that is not delivered
2. Assign first order as best order
3. Calculate value of the order divide distance between shops and the receiver location of the order to get monetary value of order
4. Choose the order with highest monetary value as the next order.

2.1.2 Select Path

1. Calculate the direct Path between Drone current position and its next destination.
    a) As drone always land on interest point, like shops, delivery point, so it is actually calculating path between interest points.
2. Check if the directed path crosses the non-fly zone
3. If cross, try to use landmarks to avoid, if not, fly direct path

2.1.2.1 calculate direct path

When calculating direct path, a method **calculateAngle()** in class Longlat, it calculate the best bearing angle between current LongLat object and a given LongLat object as destination.

So, the direct path calculator works with these steps:

1. Initialize an array that store the path.
2. Add start position into path
3. Loop while current position is not close to destination
4. Get bearing angle between current position and destination, perform move with the bearing angle
5. Add new position in to array path

6. Back to loop.
7. Loop finished, return path.

2.1.2.2 check path cross non-fly zone or not

The checking algorithm use two consecutive LongLat objects in array of path to from a lin2D object, which indicating a unit move of the drone. Then, use coordinates of the non-fly buildings to from several line2D objects indicating the borders of all the ono-fly building, check if the unit move lin2D cross the border lines of all the non-fly zone.

2.1.2.3 Path decision

If the direct path between drone and its destination crosses the non-fly zone, drone will use landmark to avoid cross the non-fly zone. Using landmarks cannot avoid all the case that the path of drone crosses the non-fly zone, but most of them will be avoided.

The algorithm of selecting path of the drone will works with following steps

1. If the direct path not cross non-fly zone, perform fly with direct path.
2. If cross non-fly zone, loop though all the landmarks to find the landmark that the paths between drone and landmark, landmark and destination do not cross non-fly zone.
   a) If cannot find a landmark satisfy conditions in step2, use the last landmark in landmark list.
   b) If there are some landmarks satisfy conditions in step2, use the landmark closest to drone

Note: there is a possibility that path using landmark still cross non-fly zone in all the case, it problem is handled in method in **moveAndRecordPatn()** in section

2.1.3 Move And Record Path

When path is decided in section 2.1.2, path passed to function moveAndRecordPatn() that will actually change drone's position and record the path to the database.

There is a case that mentioned in section 2.1.2.3, that using landmark still can not avoid flying though non-fly zone, **moveAndRecordPatn()** handle this problem.
moveAndRecordPatn() will have following steps:
Loop though the path, check if line formed by 2 consecutive LongLat object cross the non-fly zone.

- If not, get the angle that cause this move from PathAngles array and let drone do the move with that angle.
- If cross, recalculate the flying angle using function **getAvoidingAngle()** which will return angle, using this new angle to move the drone.

Then record the moved path into database.

Note: method getAvoidingAngle() works with logic of: it takes a destination and

current position. Checking all moves with angles from 0 to 350 from current position, select an angle that makes moved position closest to destination while not cross the non-fly zone or border.

2.1.4 Fly Back To Appleton Tower

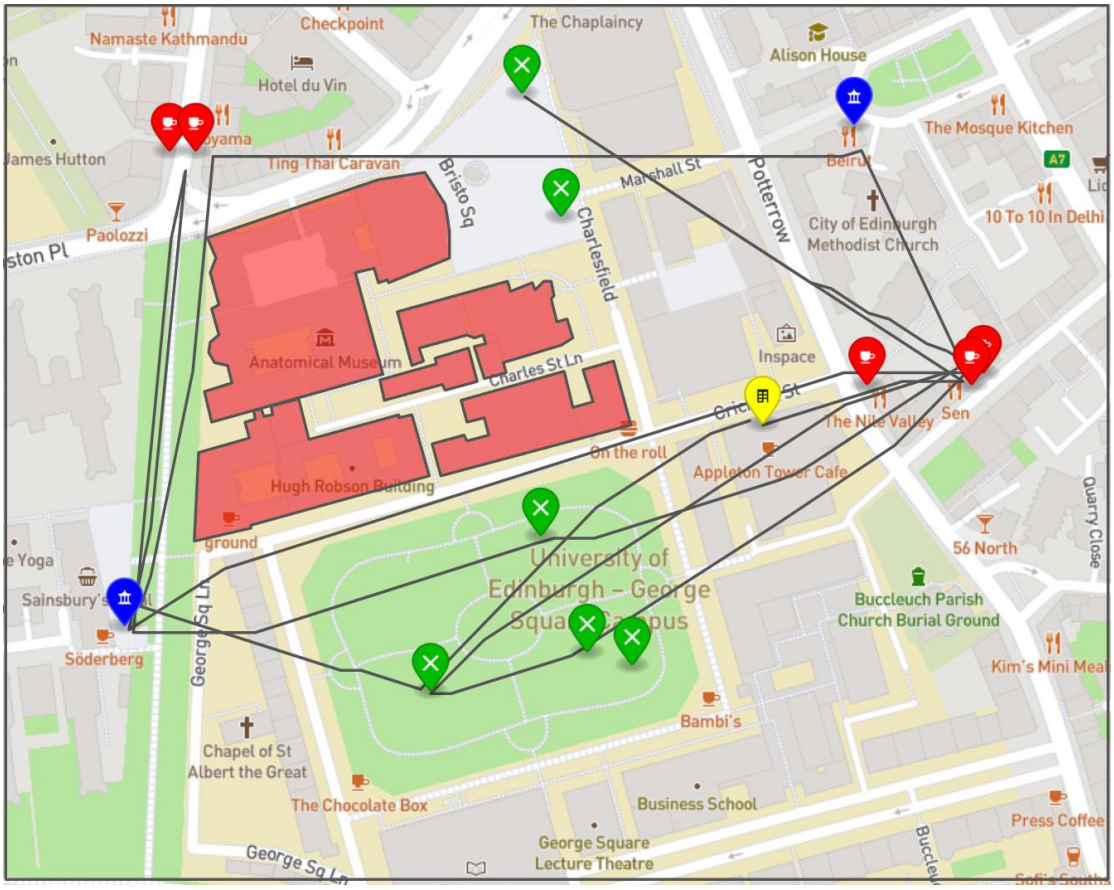
Case 1: all the orders delivered, enough battery:

In this case, after all order is delivered, fly back to Appleton Tower.

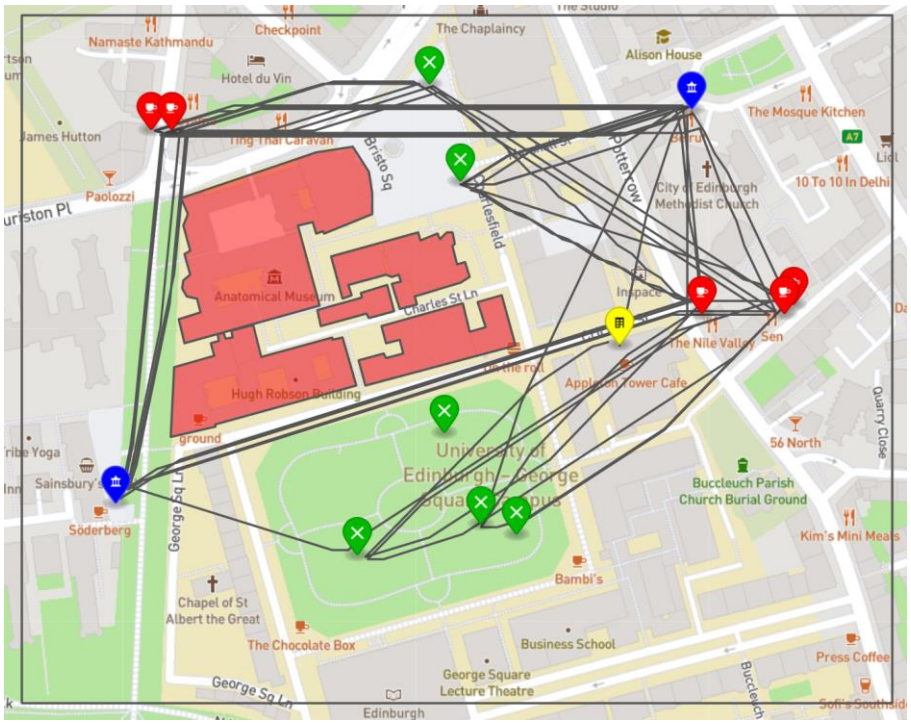
Case 2: during delivering, the drone run out of battery:

During the delivery, before the drone move to next shop or the receiver, there will be a process that check if the move is done, whether the drone are able to go beck to Appleton Tower, if not, do not move to next position, move back to Appleton Tower.

2.2 Examples of Program Output



This picture shows the path plan for orders in 01-01-2022, it is clear that drone start from Appleton Tower, flying cross interest points, avoiding fly across non-fly zone(see the corner around “on the roll”). When all order is delivered, back to Appleton Tower. Run time is 0.8 second on my PC.



This graph is the path plan for orders at 11-11-2022, with total 1193 moves, delivered 14 orders and finally back to Appleton Tower. Run time is 1.7 second on my PC.

So, my algorithm work for both large number of orders and small number of orders, but there are still some of weakness:

1. Monetary vlaue for select next order is not optimised:
   When calculating the monetary value to decide which order to go first, I did not consider landmarks, I just use the direct disdance between interest point, it is possible to improve the order selection algorithm that consider the movement to landmark when calculating whole distance, which will increase efefctiveness.
2. The algorithm of 'getAvoidingAngle':
   The algorithm of getAvoidingAngle use greedy search, but there is a possibility that causing the drone to stuck in borders of noin-fly zone, this is because the movement of the drone always has longth 0.00015 degree. Though all of my out put did not show this problem, but I think it exist.
3. Uncesserly using landmark
   It is obvious that using landmarks cause the drone to take more moves to travel to its destination, it is possible to improve the path selection algorithm so that drone can use landmark only wiser.

Reference

1. MapBox Java SDK, https://docs.mapbox.com/android/java/api/libjava-geojson/5.6.0/index.html
2. Derby database https://mvnrepository.com/artifact/org.apache.derby/derbyclient
3. Gson https://javadoc.io/doc/com.google.code.gson/gson/latest/com.google.gson/module-summary.html