Informatics Large Practical

Stephen Gilmore and Paul Jackson
School of Informatics, University of Edinburgh

Document version 1.0.0

First issued on: September 21, 2020

About

The Informatics Large Practical is a 20 point Level 9 course which is available for Year 3 undergraduate students on Informatics degrees. It is not available to visiting undergraduate students or students in Year 4 or Year 5 of their undergraduate studies. It is not available to postgraduate students. Year 4, Year 5 and postgraduate students have other practical courses which are provided for them.

Scope

The Informatics Large Practical is an individual practical exercise which consists of one large design and implementation project, with two coursework submissions. Coursework 1 involves creating a new project and implementing one important component of the project. Coursework 2 is the implementation of the entire project together with a report on the implementation.

Courseworks	Deadline	Out of	Weight
Coursework 1	16:00 on Friday 9th October 2020	25	25%
Coursework 2	16:00 on Friday 4th December 2020	75	75%

Please note that the two courseworks are not equally weighted. There is no exam paper for the Informatics Large Practical so to calculate your final mark out of 100 for the practical just add together your marks for the courseworks.

Introduction

The task for the Informatics Large Practical is to program an autonomous drone which will collect readings from air quality sensors distributed around an urban geographical area as part of a (fictitious) research project to analyse urban air quality. The research project which is collecting and analysing the air quality readings is beginning with a small-scale feasibility study around the University of Edinburgh's Central Area. If this feasibility study is successful, the researchers will apply for funding to conduct their research on a much larger scale, collecting and analysing air quality readings all across the city of Edinburgh.

— ◇ —

Your task is to program the drone to fly around and collect the sensor readings of air quality and to produce scientific visualisations of the data which can be shown to funding councils in order to convince them to fund the large-scale version of the project. Your implementation of the drone control software should be considered to be a prototype only in the sense that you should think that it is being created with the intention of passing it on to a team of research assistants and PhD students who will maintain and develop it in the months and years ahead. For this reason, the clarity and readability of your code is important; you need to produce code which can be read and understood by others.

— ◇ —

The drone is fitted with a receiver which can download readings from sensors over-the-air, provided that it is within 0.0002 *degrees* of the air quality sensor. Degrees are used as the measure of distance throughout the project instead of metres or kilometres to avoid unnecessary conversions between one unit of measurement and another. The latitude and longitude of a sensor are expressed in degrees, so we stay with this unit of measurement throughout all our calculations. As a convenient simplification, locations expressed using latitude and longitude are treated as though they were points on a plane, not points on the surface of a sphere. This simplification allows us to use Pythagorean distance as the measure of the distance between points. That is, the distance between (x_1, y_1) and (x_2, y_2) is just

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

In all, there are 99 sensors distributed around the University of Edinburgh's Central Area but not all of them need to be read each day. On any given day, a list of 33 sensors is produced which enumerates the sensors which need to be read today. The sensors are battery-powered, and when a receiver takes a reading from a sensor it has two components:

- the *reading*: this is a character string which should represent a real value between 0.0 (no air pollution was detected) and 256.0 (maximum possible air pollution reading; the sensor capacity is at its limit); and

- the *battery*: this is a real value expressing the percentage battery charge between 0.0 and 100.0.

However, if the battery has less than 10% charge then the sensor reading cannot be considered to be trustworthy because the sensor hardware is known to give false or misleading readings at low power levels. The sensor reading at less than 10% charge might be "`null`" or "`NaN`" (Not a Number) but even if the air quality reading looks like a real number it should be discarded as being essentially noise, and the sensor reported as needing a new battery.

— ◇ —

Replacing the battery in a sensor is a manual process whereby a researcher from the project is sent to the location of a faulty sensor to swap out the drained battery for a fully-charged one. A novel location addressing system is used to help locate the sensor where the battery needs to be replaced: the *What3Words*

encoding[1]. There is never more than one sensor in any What3Words tile so the What3Words address can be used as the location of the sensor and the associated mobile phone app guides the user to the What3Words address. This has the advantage that researchers do not need to key in (longitude, latitude) number pairs such as $(-3.1887, 55.9452)$; they can instead type a What3Words address such as `rooms.lamp.teach` and be taken to the same location. Past experience with handling (longitude, latitude) number pairs on previous projects has shown that it is very easy to transpose digits when keying location information, sending researchers to the wrong location. It is hoped that the use of What3Words addresses on this project will reduce the number of times that a researcher is sent to the wrong place when they are trying to replace spent batteries.

— ◇ —

All sensors which need to be visited have a latitude which lies between 55.942617 and 55.946233. They also have a longitude which lies between -3.184319 and -3.192473. There is no reason for the drone to be outside this area, so these coordinates define the *drone confinement area* as illustrated in Figure 1. The drone must at all times remain *strictly inside* the confinement area. To be clear, if the drone is at location $(55.946233, -3.192473)$ then it is *outside* the confinement area, and is judged to be malfunctioning.

Forrest Hill		KFC
(55.946233, −3.192473)		(55.946233, −3.184319)

The drone is confined to this area.

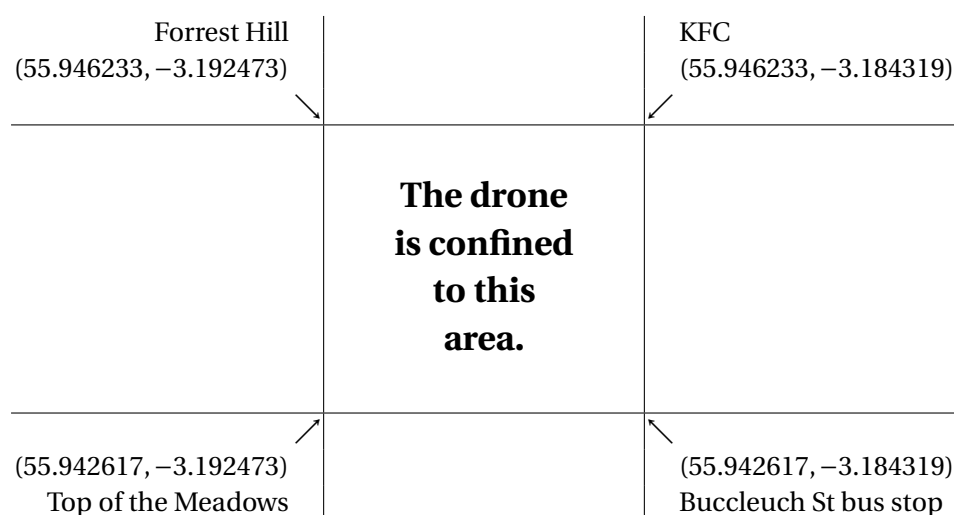| (55.942617, −3.192473)		(55.942617, −3.184319)
Top of the Meadows		Buccleuch St bus stop

Figure 1: The drone confinement area

In order to help with the development of the software for the drone, the researchers on the project have made a web server with synthetic data which represents the sensor and battery level data that a drone would be able to read from a sensor on a given day if it was in range (i.e. within 0.0002 degrees) of the air quality sensor. Other important files are stored on the web server also. The web server has three top-level folders, `maps`, `words`, and `buildings`.

The `maps` folder: The `maps` folder contains files containing lists of sensors to be visited, one for each day in the years 2020 and 2021. This folder is structured with top-level folders 2020 and 2021. In each of these are folders 01 .. 12, representing months of the year. Within these are folders 01 .. 28 (and 29, 30, and 31 as appropriate), representing days of the month. Within each of these folders is a JSON (JavaScript Object Notation) format file named `air-quality-data.json` which specifies the list of sensors which are to be visited on that day, together with the air quality reading and the battery level of that sensor.

An extract from the file `maps/2021/06/15/air-quality-data.json` appears in Figure 2. This is the synthetic data representing the readings that the drone would get if it was flying on 15th June, 2021.

---

[1]The What3Words system maps the earth's surface using 3m × 3m tiles, each with a unique three word address. More information is available at `https://what3words.com/`.

```
[
  {
    "location": "slips.mass.baking",
    "battery": 94.53979,
    "reading": "141.81"
  },
  {
    "location": "sports.topic.clocks",
    "battery": 1.0801673,
    "reading": "null"
  },
  ...,
  {
    "location": "sculpture.shot.melon",
    "battery": 21.519148,
    "reading": "40.8"
  }
]
```

Figure 2: An extract from the file `maps/2021/06/15/air-quality-data.json`

It is a list of 33 JSON records with fields `location` (String), `battery` (numeric), and `reading` (String). There will always be 33 records in these lists, each one representing an air quality sensor.

The `words` folder: The `words` folder contains JSON files which give the What3Words information corresponding to a What3Words address. The details for a What3Words address `"first.second.third"` will be stored in the file `words/first/second/third/details.json`. The most important field in this record for our purposes is the `coordinates` field which allows us to associate a What3Words address with a specific longitude (`"lng"`) and longitude (`"lat"`). In this project, longitudes will always be negative (close to −3) and latitudes will always be positive (close to 56) because Edinburgh is located at 3 degrees West and 56 degrees North.

The What3Words file corresponding to the address `"slips.mass.baking"` is shown in Figure 3.

The `buildings` folder: Until now it might have seemed that the drone had complete freedom of movement in moving from one sensor to another, so long as it remains in the drone confinement area, but that is not the case. Within the confinement area are four regions where the drone is not allowed to fly. These four regions are known as the *No Fly Zones* for the drone. The details of the No Fly Zones are given in the file `buildings/no-fly-zones.geojson`. This file is in Geo-JSON format, which is a standard way of encoding geographic data structures[2].

There are four areas in the No Fly Zones where the drone is not allowed to be. These are the Appleton Tower; the David Hume Tower; the Main Library; and the Informatics Forum / Bayes Centre / Dugald Stewart Building complex, including the enclosed area within the complex. The drone cannot move into any of the No Fly Zones with a move, and it cannot *pass through* any No Fly Zones in moving from one point to the next.

The No Fly Zones are illustrated in Figure 4, together with the drone confinement area showing the limit of the drone's position.

---

[2]For more details, visit `https://geojson.org`. To render Geo-JSON maps, visit `https://geojson.io`.

```
{
  "country": "GB",
  "square": {
    "southwest": {
      "lng": −3.187428,
      "lat": 55.945936
    },
    "northeast": {
      "lng": −3.18738,
      "lat": 55.945963
    }
  },
  "nearestPlace": "Edinburgh",
  "coordinates": {
    "lng": −3.187404,
    "lat": 55.945949
  },
  "words": "slips.mass.baking",
  "language": "en",
  "map": "https://w3w.co/slips.mass.baking"
}
```

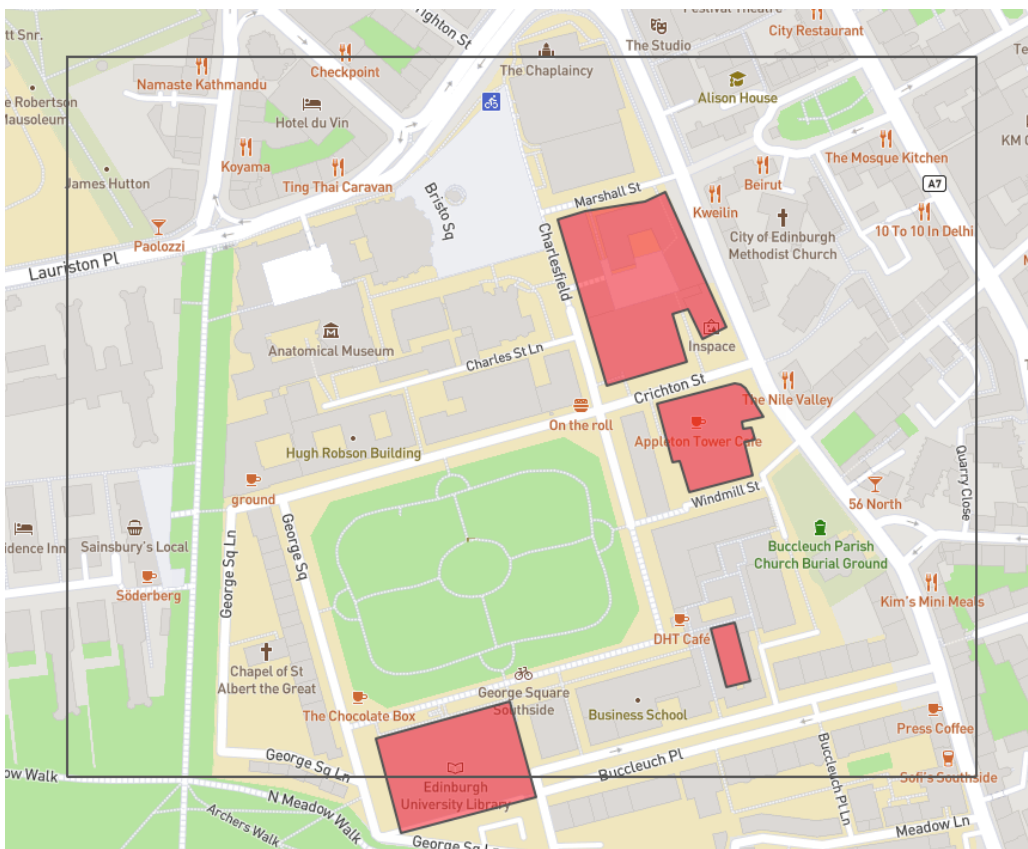Figure 3: The content of the file `words/slips/mass/baking/details.json`

Figure 4: A Geo-JSON map rendered by the website http://geojson.io/. The map shows the drone confinement area (the outer grey rectangle) and the four no-fly zones (the red polygons over the Informatics Forum, the Appleton Tower, the David Hume Tower, and the Main Library).

Your task is to program the drone to fly around the confinement area, getting in range of each of the air quality sensors to collect readings, while avoiding the no-fly zones. The flight of the drone is subject to the following stipulations:

- the drone flight path has at most 150 moves, each of which is a straight line of length 0.0003 degrees;

- the drone *cannot fly in an arbitrary direction*: it can only be sent in a direction which is a multiple of ten degrees where, by convention, 0 means go East, 90 means go North, 180 means go West, and 270 means go South, with the other multiples of ten between 0 and 350 representing the obvious directions between these four major compass points[3];

- as near as possible, the drone flight path should be a closed loop, where the drone ends up close to where it started from; and

- the drone life-cycle has a pattern which iterates (i) making a move; and (ii) taking one sensor reading (if in range).

To clarify the last point, the drone must move before taking a reading, even if a sensor is in range of the initial point on the flight path. Additionally, the drone can choose to connect to *any* sensor which is in range of its current position, not necessarily the nearest one, and it might choose not to connect to a sensor, even if it is in range, but it cannot connect to two or more sensors without making a move between each connection.

— ◇ —

In order to help communicate the behaviour of the drone to others, the drone flight path is to be plotted to a Geo-JSON map which also includes the locations of the sensors plotted using coloured markers which represent the levels of air pollution detected by the sensors. The markers on the map have four properties:

- `location` — the What3Words location string;

- `rgb-string` — the HTML colour of this marker represented as a hexadecimal Red-Green-Blue string;

- `marker-color` — identical to `rgb-string`, but will be rendered by `http://geojson.io`; and

- `marker-symbol` — a symbol from the Maki icon set (`https://labs.mapbox.com/maki-icons/`).

The markers on the map have a symbol which is either a lighthouse (`"lighthouse"` — representing safe levels of air pollution); a skull-and-crossbones (`"danger"` — representing dangerous levels of air pollution); or a cross (`"cross"` — representing low battery). A sensor not visited has no `marker-symbol`. The marker colours are assigned according to the colour mapping for an air quality reading of x as given in Figure 5.

Range	RGB string	Marker colour	Colour name	Marker symbol
$0 \le x < 32$	#00ff00		Green	lighthouse
$32 \le x < 64$	#40ff00		Medium Green	lighthouse
$64 \le x < 96$	#80ff00		Light Green	lighthouse
$96 \le x < 128$	#c0ff00		Lime Green	lighthouse
$128 \le x < 160$	#ffc000		Gold	danger
$160 \le x < 192$	#ff8000		Orange	danger
$192 \le x < 224$	#ff4000		Red / Orange	danger
$224 \le x < 256$	#ff0000		Red	danger
low battery	#000000		Black	cross
not visited	#aaaaaa		Gray	no symbol

Figure 5: The mapping from air quality sensor readings to marker colours and marker symbols.

---

[3]Negative angles are not allowed, and angles greater than 350 are not allowed.

A sample output map is given in Figure 6 for the 15/06/2021 map. The drone flight path is shown as a dark grey line. This is a slightly unsatisfactory drone flight path; other flight paths could improve on this one. The drone starts in the middle of the map (at longitude −3.1878 and latitude 55.9444) near the "On the roll" food cart and visits all of the air quality sensors to be read that day. That part is successful, but unfortunately, the drone does not manage to return to the "On the roll" cart because its algorithm fails to work out how to fly around the Informatics Forum on its way back, leaving it stuck flying back and forth along the side of the Informatics Forum. It seems that a better algorithm is needed for the drone! This image was rendered using the `http://geojson.io` website.

Figure 6: A sample output map for the date 15/06/2021.

Note that this drone was unable to return to (−3.1878, 55.9444) but it might have been able to make the required closed-loop flight path if it had been started at another location on the map. However, the researchers on the air quality project have asked that the drone should return close to its point of origin no matter where it is started on the map[4] so for this reason, the flight path shown is Figure 6 is slightly unsatisfactory. If a drone returns "close to"[5] its initial position in less than 150 moves then the flight path can terminate at that point, there is no need to artificially extend the flight path to 150 moves.

— ◇ —

The information on map marker properties[6] can be accessed by clicking on the markers in the `geojson.io` interface. The property values for the three markers corresponding to the three JSON records shown in Figure 2 can be seen in Figure 7. The first sensor (at location `"slips.mass.baking"`) registered dangerous air quality readings. The second sensor (at location `"sports.topic.clocks"`) had a low battery so no reading was obtained. The third sensor (at location `"sculpture.shot.melon"`) and fourth sensor (at location `"hurt.green.filer"`) registered safe air quality readings.

---

[4]The drone is always started within the drone confinement area and is never started within a no-fly zone.
[5]The precise meaning of "close to" here is "less than 0.0003 degrees from".
[6]Including a fifth property, `marker-size`, which defaults to `"medium"`.
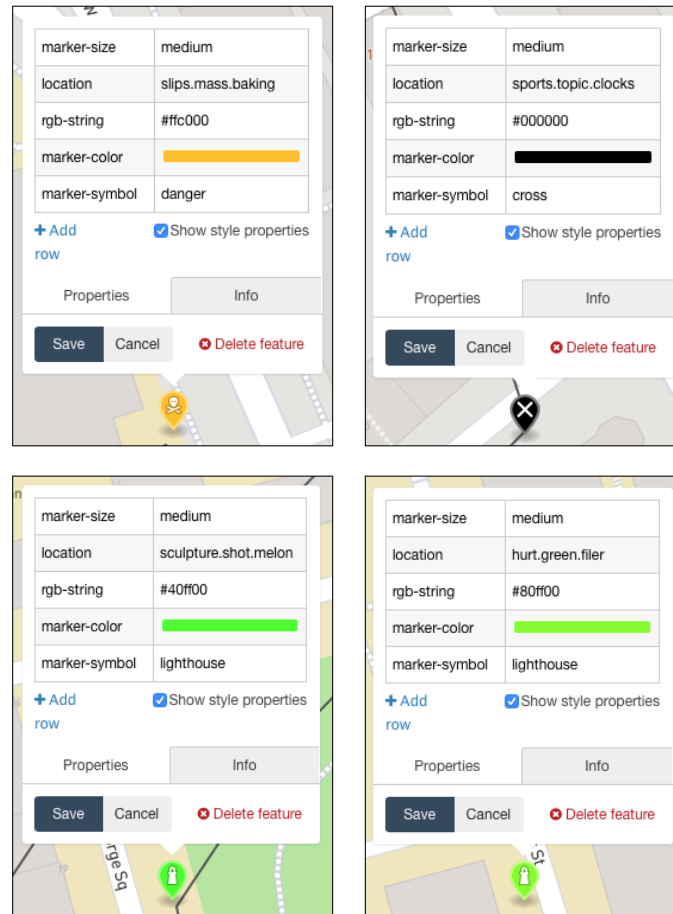
Figure 7: Four sample markers on the map for the date 15/06/2021.

Weather conditions, wind speed, and wind direction will not be a factor to consider when computing the flight path of the drone. The researchers flying the drone will choose a time of day when the flying conditions are most favourable. This will mean that we can proceed as though the drone is flying on a perfectly still day with no wind. This means that the drone travels the same distance no matter which direction it is flying in. Obstacles such as trees, street lights, and even buildings also need not be considered. We can think that the drone is flying high enough that it flies over all the buildings in the George Square area—with the exception of the buildings specified as no-fly zones, which it must fly around.

— ◇ —

When the drone is flying, it is necessary to determine its next location from its current location and the direction of travel (expressed as a multiple of 10 degrees; 0 being East, 90 being North, etc). When considering small changes of latitude or longitude, as we are doing here, it is acceptable to approximate the earth's surface as a plane. This means that, knowing for each move that the drone travels a distance r, which is 0.0003 degrees, we can calculate the drone's next position using simple planar trigonometry. We calculate a new position by considering right-angled triangles with hypotenuse r, and other sides giving the differences in latitude and longitude caused by travelling in this direction.

The implementation task

As the main part of this practical exercise, you are to develop an application which, given the initial starting location of the drone and the date, calculates a flight path which visits as many of the sensors listed for that date as possible, and returns as close to its initial starting location once all sensors have been visited. The application produces a report on the drone's flight which takes the form of a Geo-JSON map and a log file which lists where the drone has been and which sensors it connected to. (As a warm-up exercise before this, you are asked to create an application which generates a Geo-JSON map in order to learn about the Geo-JSON language and the software which is available for creating Geo-JSON maps. See Coursework 1 for more details.)

— ◇ —

Recall that the drone flight path which you generate should be 150 moves in length; the drone is also a battery-powered device so the number of moves that it can make is limited, and 150 is the limit here. Your application should load the `air-quality-data.json` file from the project's webserver[7] for the specified date and produce the Geo-JSON map and the log file for that date.

— ◇ —

Assuming that your project is called `aqmaps` (for "Air Quality Maps") and stored in the file `aqmaps.jar` then when your project is run with the command-line arguments:

```
15 06 2021  55.9444 -3.1878  5678  80
```

or the command

```
java -jar aqmaps.jar  15 06 2021  55.9444 -3.1878  5678  80
```

it should load the `air-quality-data.json` file for 15/06/2021 from the webserver, connecting at port 80. It should start the drone at a latitude and longitude of $(55.9444, -3.1878)$ and use the number 5678 as the random number *seed* for the application[8].

— ◇ —

Your application may write any messages which it likes to the standard output stream but it should also write two text files in the current working directory. These are named `flightpath-DD-MM-YYYY.txt` and `readings-DD-MM-YYYY.geojson` where *DD*, *MM*, and *YYYY* are replaced by the day, month and year of the relevant `air-quality-data.json` file.[9] Please use hyphens (not underscores) in the file names and use only lowercase letters. The content of these two files is the following.

flightpath-*DD-MM-YYYY*.txt This file should be 150 lines long. Each line is numbered, starting at 1. It contains each move of the drone in terms of the longitude and latitude of the drone *before* the move, the direction it chose to move, the longitude and latitude of the drone *after* the move, and the location of any sensor that is connected to after the move, or `null` otherwise. For example, the first line of this file could be:

```
1,-3.1878,55.9444,110,-3.187902606042998,55.94468190778624,hurt.green.filer
```

---

[7]Instructions for running the webserver are given in Appendix B of this document.

[8]Your algorithm for controlling the drone might make use of some (controlled) randomness, and for this reason we use a seed to make executions repeatable.

[9]For example, `flightpath-15-06-2021.txt` and `readings-15-06-2021.geojson` if these results were obtained when processing the `air-quality-data.json` file for June 15th, 2021.

This says that the drone was initially at (−3.1878, 55.9444), then decided to move in a north-westerly direction (110 degrees) to (−3.187902606042998, 55.94468190778624), and after that move was completed it connected to the sensor at What3Words location "hurt.green.filer" to take readings from that sensor for plotting on the map.

Note: Please follow this file format carefully because these TXT files will be processed by an application later. The format of each line is *int,double,double,int,double,double,string*. Be careful not to exchange latitude and longitude values; the expected format for each drone position is *longitude,latitude*. Do not put the location (or null) in quotation marks.

readings-*DD-MM-YYYY*.geojson This is a map in Geo-JSON format which contains 33 markers at the locations of the 33 air-quality sensors which are specified in the file which is located on the web server at maps/*YYYY*/*MM*/*DD*/air-quality-data.json. Each of these markers is a Geo-JSON Feature of type Point, coloured according to the convention presented in Figure 5.

In addition to these 33 markers, there is one additional Feature of type LineString, which plots the flightpath of the drone as a list of *longitude,latitude* pairs.

Programming language

The programming language to be used for your software is Java. The researchers on the project have chosen Java version 11 as the version of Java which will be used throughout the project. This version of Java has been selected because it provides local variable type inference and other helpful features and is the most recent LTS (Long Term Support) version of Java which is available. You should ensure that the code which you submit can be compiled and run on a Java 11 installation.

— ◇ —

The version of the Java language which is the default on DICE Ubuntu is Java 14. If you are developing your application on a version of Java which is greater than 11 you should take care not to use language features or APIs which are not available in a Java 11 installation. You can set the Java language version to 11 in Eclipse to prevent yourself from using APIs from higher versions of Java.

Using third-party software and libraries

This practical exercise allows you to use free software, but not commercial software which you must pay for or license. One free software development kit (SDK) which you should find useful is the Mapbox Java SDK which provides classes and methods for parsing and generating Geo-JSON maps. Instructions for adding the Mapbox Java SDK to your project are available at https://docs.mapbox.com/android/java/overview/.

Informatics Large Practical

Coursework 1

Stephen Gilmore and Paul Jackson
School of Informatics, University of Edinburgh

1.1 Introduction

This coursework and the second coursework of the ILP are for credit; weighted 25%:75% respectively. Please now read Appendix A for information on good scholarly practice and the School's late submission policy.

— ◇ —

This coursework is a warm-up exercise before the main exercise in Coursework 2. In this coursework you are learning about Geo-JSON maps by visualising a "heat map" over the drone confinement area.

— ◇ —

In this project you are creating a Java application which is built using the Maven build system. We will begin by using Eclipse to create the project structure.

1.2 Getting started

If you are working on your own laptop you should begin by downloading Eclipse, if you do not already have it. Download it from `https://www.eclipse.org`. On DICE, Eclipse is available via the `eclipse` command.

— ◇ —

Next, create a new Maven project in Eclipse by choosing File → New → Project ..., and choosing Maven Project as the option. Leave unchecked the option which reads "Create a simple project (skip archetype selection)"; this is unchecked by default. On the following page, choose the archetype "maven-archetype-quickstart" (this is the default). On the final page, fill in the options as shown below:

<div align="center">

Group Id: uk.ac.ed.inf
Artifact Id: heatmap

</div>

Find "JRE System Library" in the Package Explorer and select "Properties". Change "Execution environment" to be "JavaSE-11".

— ◇ —

You should now have a working Maven project structure. Note that there are separate folders for project source and project tests. Note that there is an XML document named `pom.xml` where you can place project dependencies. Two Java files have been automatically generated for you: `App.java` and `AppTest.java`.

1.3 Setting up a source code repository

(This part of the practical is not for credit, but it strongly recommended to help to protect you against loss of work caused by a hard disk crash or other laptop fault.)

— ◇ —

In the Informatics Large Practical you will be creating Maven project resources such as XML documents and Java files which will form part of your implementation, to be submitted in Coursework 2. We recommend that these resources be placed under version control in a source code repository. We recommend using the popular Git version control system and specifically, the hosting service *GitHub* (`https://github.com/`). GitHub supports both public and private repositories. You should create a *private* repository so that others cannot see your project and your code.

— ◇ —

Check your current Maven project into your GitHub repository. Commit your work after making any significant progress, trying to ensure that your GitHub repository always has a recent, coherent version of your project. In the event of a laptop failure or other problem, you can simply check out your project (e.g. into your DICE account) and keep working from there. You may have lost some work, but it will be a lot less than you would have lost without a source code repository.

— ◇ —

A tutorial on Git use in Eclipse is here: `https://eclipsesource.com/blogs/tutorials/egit-tutorial`

1.4 The implementation task

Research projects in science often need to *visualise* data and results in order to make the information that they contain easier to understand both for the scientists on the project and in communicating the results of the project to members of the general public, science journalists, politicians, and other stakeholders. One commonly-used form of scientific visualisation is a *heat map* in which data values are represented as colours across a surface or area. The intention of this heat map is to visualise the predictions that the project researchers have made for the highest sensor reading which will be seen in each area of the drone confinement area, partitioned into a regular 10 × 10 grid.

— ◇ —

The predictions of the researchers are input into your heat map application as a text file with 10 lines of text, each of which has 10 integer values separated by commas, as seen in the example file `predictions.txt` in Figure 8. The values are understood to be in the order of the most northerly values down to the most southerly values, reading each row of data containing the values from the west to the east, in order. These values are to be used to generate an output file in the default output directory called `heatmap.geojson`.

— ◇ —

The output Geo-JSON document will use the values from the input to produce a 10 × 10 grid which covers the drone confinement area whose dimensions were specified in Figure 1. Each of the rectangles in the grid is represented as a Geo-JSON `Polygon`[10] with properties `"fill-opacity"` being 0.75, and properties `"rgb-string"` and `"fill"` both being the colour specified for the relevant input prediction for this area, coloured according to the colour mapping which is specified in Figure 5.

— ◇ —

If your application is in the file `heatmap.jar`, the command `java -jar heatmap.jar predictions.txt` should produce a `heatmap.geojson` output file. A sample input and output are in Figures 8 and 9.

---

[10]Geo-JSON does not have a `Rectangle` geometry.

```
200, 255, 200, 255, 255, 200, 255, 200, 255, 255
200, 255, 255, 255, 255, 255, 255, 255, 255, 255
255, 255, 255, 220, 200, 220, 255, 255, 220, 255
200, 200, 200, 140, 140, 140, 200, 255, 255, 255
80, 80, 80, 100, 100, 140, 180, 255, 255, 220
40, 40, 60, 60, 60, 80, 120, 220, 255, 220
40, 40, 60, 60, 40, 40, 120, 200, 225, 255
40, 40, 60, 60, 40, 40, 120, 180, 220, 255
0, 0, 50, 75, 75, 100, 165, 200, 200, 255
0, 0, 50, 50, 75, 200, 200, 255, 255, 255
```

Figure 8: A file, `predictions.txt`, containing predictions of the maximum sensor readings which will be seen in each rectangle of the 10×10 grid over the drone confinement area.
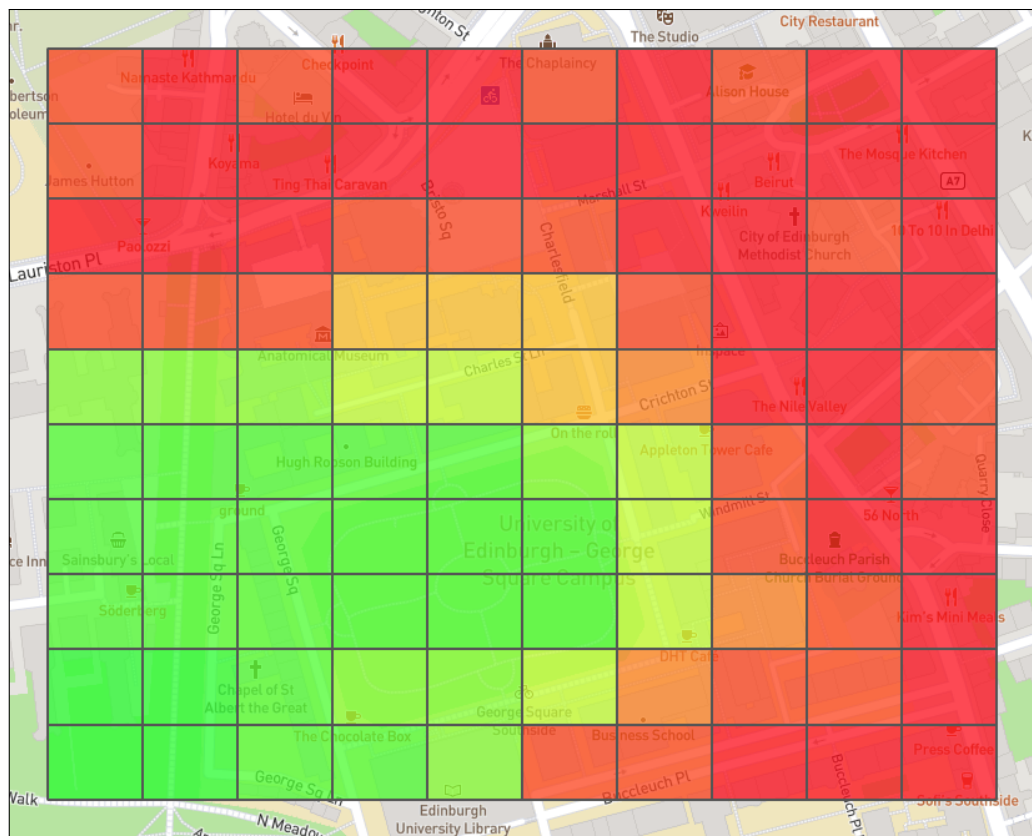

Figure 9: A Geo-JSON heat map grid from the file `heatmap.geojson` which visualises the predictions given in Figure 8 using the colour mapping which is specified in Figure 5. This Geo-JSON map was rendered using the `http://geojson.io` website.

Using the `http://geojson.io` website you can visualise your Geo-JSON output file and check that the colour value assigned to each polygon in the heat map grid is the expected colour value, as shown in Figure 10.
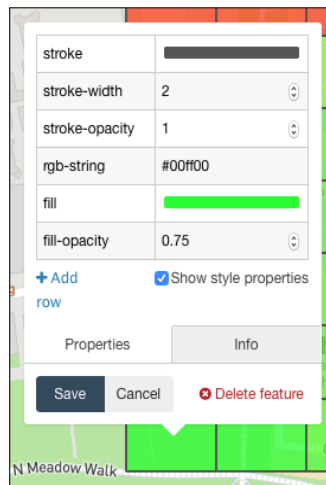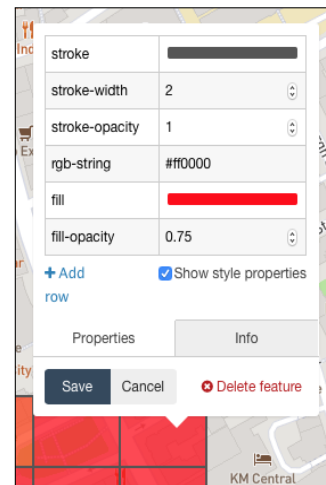
Bottom left corner   Top right corner

Figure 10: The rectangle in the bottom-left corner of the heat map grid is green in colour with RGB string #00ff00. The rectangle in the top-right corner of the heat map grid is red in colour with RGB string #ff0000. The `rgb-string` value and the `fill` value are identical in the Geo-JSON document.

1.5   Allocation of marks

A total of 25 marks are allocated for Coursework 1 according to the following weighting.

Correctness (15 marks): Your application should correctly render the heat map corresponding to the predicted data values in the input text file.

Readability (10 marks): Your application should be well-structured and clear. You should consider the readability of your code, thinking that it will be passed on to the members of the air quality research project to extend and maintain as their needs change.

Informatics Large Practical

Coursework 2

Stephen Gilmore and Paul Jackson
School of Informatics, University of Edinburgh

2.1   Introduction

As noted above, Coursework 1 and Coursework 2 of the ILP are for credit; weighted 25%:75% respectively. Information on good scholarly practice and the School's late submission policy is provided in Appendix A.

— ◇ —

This coursework consists of the report, the implementation, and a collection of output files written by your implementation. The code which you submit for assessment should be readable, well-structured, and thoroughly tested.

2.2   Report on your implementation

You are to submit a report documenting your project containing the following. Your report should have three sections as described below.

1. *Software architecture description.* This section provides a description of the software architecture of your application. Your application is made up of a collection of Java classes; explain why you identified *these classes* as being the right ones for your application.

2. *Class documentation.* Provide concise documentation for each class in your application. Explain each class as through providing documentation for a developer who will be maintaining your application in the future.

3. *Drone control algorithm* This section explains the algorithm which is used by your drone to control their flight around the air-quality sensors and back to the start location of their flight, while avoiding all of the no-fly zones.

   This section of your report should contain two graphical figures (similar to Figure 6 in this document) which have been made using the `http://geojson.io` website, rendering the flights of your drone on two dates of your choosing.

— ◇ —

The maximum page count of your project report is 15 pages, with title pages, table of contents, references, appendices, and all other material included in the page total. You cannot submit reports which are over 15 pages in length, but shorter submissions will be accepted. The choice of font, font size, and margins is up to you but please consider the readability of your submission, and avoid very small font sizes and very small margin sizes. Reports of few pages tend to attract few marks; consider 15 pages to be a *goal*, as well as a limit.

2.3 Source code of your application

You are submitting your source code for review where your Java code will be read by a person, not a script. You should tidy up and clean up your code before submission. This is the time to remove commented-out blocks of code, tighten up visibility modifiers (e.g. turn `public` into `private` where possible), fix and remove TODOs, rename variables which have cryptic identifiers, remove unused methods or unused class fields, fix static analysis problems, and to refactor your code as necessary. The code which you submit for assessment should be well-structured, readable and clear.

2.4 Results from your drone

In addition to submitting your source code for assessment, you should also submit 24 output files giving the results of trying your drone against the sensor visits to be made on specific days of the year[11] while starting the drone at a latitude and longitude of $(55.944425, -3.188396)$. You should submit a ZIP file archive `ilp-results.zip` containing the following files:

```
flightpath-01-01-2020.txt   readings-01-01-2020.geojson
flightpath-02-02-2020.txt   readings-02-02-2020.geojson
             ⋮                          ⋮
flightpath-12-12-2020.txt   readings-12-12-2020.geojson
```

All of the files submitted should have been generated using the same version of your application.

2.5 Things to consider

- Your submitted Java code will be read and assessed by a person, not a script. It is not a waste of time to add comments to your code, documenting your intentions. Your submitted code should be readable and clear.

- Your output files `*.txt` and `*.geojson` will be processed by a script, not read by a person. Your text file must be formatted as described above, and your Geo-JSON file must have the content as described above.

- Logging statements and diagnostic print statements (using `System.out.println` and friends) are useful debugging tools. You do not need to remove them from your submitted code; it is fine for these to appear in your submission. You can write whatever you find helpful to `System.out`, but the content of your output files must be as specified above. An excessive level of logging can be counter-productive, causing the user not to read the log output, thereby defeating the purpose. Consider what should be logged, and log sparingly.

- Your application should be robust. Failing with a `NullPointerException`, `ClassCastException`, `ArrayIndexOutOfBoundsException` or other run-time error will be considered a serious fault.

2.6 Allocation of marks

A total of 75 marks are allocated for Coursework 2 according to the following weighting.

Report (30 marks): You are to provide a document describing your implementation. Your document should be a clear and accurate description of your implementation as described in Section 2.2 above. The three sections of the report are equally weighted, with 10 marks for each section.

---

[11]DD/MM/2020 where DD=MM.

Implementation (30 marks): Your submission should faithfully implement the drone behaviour described above, hosted in a framework which allows the drone to make a maximum of 150 moves on any day. Your application should be usably efficient, without significant stalls while executing. Your code should be readable and clear, making use of private values, variables and functions, and encapsulating code and data structures. Where it is appropriate to do so, your classes should be structured in a way which makes use of the Java class hierarchy mechanism. All else being equal, code with comments should receive a higher mark than code without comments. Everyone thinks that their code is self-documenting, but it isn't.

Output (15 marks): The output files which you submit will be tested to ensure that the moves made by the drone are legal according to the description given above, considering the drone confinement area and the no-fly zones described above. The efficiency of the drone will be considered: the fewer the number of moves required to visit all the sensors and return close to the initial location on the flight path, the better the quality of the drone.

Appendix A

Coursework Regulations

Good scholarly practice

Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

```
http://web.inf.ed.ac.uk/infweb/admin/policies/
academic-misconduct
```

This also has links to the relevant University pages. You are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately.

— ◇ —

The Informatics Large Practical is not a group practical, so all work that you submit for assessment must be your own, or be acknowledged as coming from a publicly-available source such as Mapbox sample projects, answers posted on StackOverflow, or open-source projects hosted on GitHub, GitLab, BitBucket or elsewhere.

Late submission policy

It may be that due to illness or other circumstances beyond your control that you need to submit work late. The School of Informatics late submission policy aligns with the university's Assessment Regulations which applies the following penalty: *5 percentage points will be deducted for every calendar day (or part thereof) it is late, up to a maximum of 7 calendar days.* However, to this University policy, the School of Informatics also adds the constraint that *late submission will only be accepted if no submission in time has been made.* For more information, see

```
http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/
coursework-projects/late-coursework-extension-requests
```

The staff of the Informatics Teaching Organisation will approve extension requests up to 7 days in accordance with the "good reasons" which are defined in the University's assessment regulations.

— ◇ —

Please note that things that would *not* be considered good reasons include (among other things) last-minute computer problems and loss of work through failing to backup your source code. Frequent commits of your work into a remote source code repository such as GitHub provide an off-site backup of your work which can allow you to continue from where you left off in the case of a disk or other hardware failure, so we recommend committing your work frequently.

Appendix B

Running the web server

Introduction

In the description above, we made several references to content which was stored on a web server. To ensure efficient access to this web server, and to avoid problems with firewalled access to the internet, you will run the web server on the same machine that you are using to run your Java application. The web server is not needed for Coursework 1 but it is needed for Coursework 2.

— ◇ —

Download the web server application and its content from the "Course materials" section of the ILP course web page at

    http://course.inf.ed.ac.uk/ilp

We will use a very lightweight web server which is implemented entirely in Java and has a very small memory footprint. Unpack the ZIP file containing the web server and the web server content into a directory on your machine. You can now run the web server using the command

    java -jar WebServerLite.jar

This will start the web server on the default port (80) and it will serve the content in the maps, words, and buildings folders. You can check the web server is running by visiting the address

    http://localhost:80

in your preferred web browser.

— ◇ —

All HTTP requests are logged to the console with output like this:

```
[Sun Sep 13 20:17:34 BST 2020] ... "GET /buildings/ HTTP/1.1" 200
[Sun Sep 13 20:17:36 BST 2020] ... "GET /buildings/no-fly-zones.geojson HTTP/1.1" 200
```

If you cannot run the web server on port 80 on your machine (say, because you already have a web server running there) then you can start the web server on a different port like this:

    java -jar WebServerLite.jar */path/to/web/server/content* 9898

You can now check that the web server is running on your preferred port by visiting the address:

    http://localhost:9898

and you would start your Java application with a command like

    java -jar aqmaps.jar 15 06 2021 55.9444 -3.1878 5678 9898

passing the port number as the last of the command-line arguments.

You will need to have the web server running every time that you are running your Coursework 2 code. If the web server is not running then any attempt to connect to the web server to get JSON or Geo-JSON files will throw a Java exception such as

```
java.net.ConnectException: Connection refused
```

Credits

If you would like more information about the web server which we are using, you can find it at the author's web page at

```
http://www.jibble.org/jibblewebserver.php
```

together with the source code of the web server.

Appendix C

Using the Piazza Forum

Details

The Informatics Large Practical has a discussion forum on Piazza. This is available online at the address `https://piazza.com/ed.ac.uk/fall2020/infr09051ilp/home`. You can register yourself to this forum at `https://piazza.com/ed.ac.uk/fall2020/infr09051ilp` — please enrol with your own name, not a pseudonym or screen name.

Guidelines

Subscribing to the Informatics Large Practical Piazza forum is optional, but strongly encouraged. Questions posted to the forum may be answered by the course lecturers or by another student on the course. Please read the following notes to ensure that you have the best experience with the forum. These guidelines are based on several years of experience with course fora, where issues such as those below have arisen.

- Anonymous and pseudonymous posting on the forum is not allowed so please enrol for the course Piazza forum with your own name. Please be aware that, however they may appear to you, posts on the forum are not anonymous to the course lecturers. The forum is available only to students who are enrolled on the ILP this year. The course lecturers reserve the right to delete the enrolment of anyone who is not (or appears not to be) registered for the ILP.

- Especially when commenting on another student's work, please consider the feelings of the person receiving your message. Please refrain entirely from comments criticising the progress of another student. Each of us works at our own pace and there are many different possible orders in which to tackle the work of the ILP. Perhaps you finished implementing something in Week 4, but that does not mean that everyone did.

- If you find some content on the forum helpful, or think that it is making a useful contribution to the course, please acknowledge this by clicking "Good question" or "Good answer" as appropriate; this encourages continued participation in the forum. The course lecturers will endorse answers which they believe to be helpful.

- Forum postings which intend to correct factual errors or resolve ambiguities in the practical specification are welcome. If necessary, the course lecturers will update this coursework document to correct the error/resolve the ambiguity.

- When asking for help with fixing a run-time error, such as an exception, please include what seems to be the most relevant part of the diagnostic error message that you receive, but please include as little of your code as possible. The course lecturers may edit or delete your post if you include too much program code.

- The Informatics Large Practical is an individual programming project so you are not allowed to share your code with others. Please bear this in mind when answering questions on the forum; do not post your solution as an example for someone else to borrow from. *Piazza is not StackOverflow:* please do not post minimal working examples for others to copy and use.

- Many questions on Piazza tend to be of the form "Do we need to do V for Coursework 1?" or "Are we expected to do W for Coursework 2?". You already have the answers to these questions. This document, the one you are reading right now, contains the definitive statement of what is required for each coursework. It is the *coursework specification*. If this document does not say that it is necessary to do V for Coursework 1, or to do W for Coursework 2, then you do not need to do those things.

- Forum postings which ask for part of the solution to the practical are strongly discouraged. Examples in this category include questions of the form "What is the best way to implement X?" and "Can I have some hints on how to do Y?"

- Questions about the marking scheme for the practical are strongly discouraged. Examples in this category include questions of the form "Which of the following alternatives would get more marks?" and "How much detail is required for Z?"

Appendix D

Document version history

1.0.0 (September 21, 2020): Initial version of this document issued.