Michelle Song
Chen Ren
Final Project – Option 2

Problem Statement

Develop a graphic representation of the graph problem using PyQt5's QPainter device and overwriting the paintEvent() (in a frame).

Your application needs to provide the following functionality:

1. Read a graph (air traffic problem) from an SQL database using sqlite3.
2. Design and implement a graphical user interface (GUI) that allows you to
    1. Plot the entire graph (2D drawing using black or blue lines and points)
    2. Select start and end points from pull-down menus (work from Assignment #6)
    3. Display a list of shortest path or found paths, depending on user selection (utilize your graph class and add methods to get the information, if not yet implemented)
    4. If the user selects "shortest path", plot that path using a red line
    5. Make sure scaling the window does not mess up your plot!
    6. All output (text) has to go through your GUI. **NO OUTPUT TO THE CONSOLE!**
    7. **TWO PERSON VERSION**: Draw a map image/bitmap (USmap.png) behind the flight paths. You will need to load the map and scale as needed. Make sure map and plot align (scaling problem). The provided map requires an alternative mapping of longitude and latitude as follows:

        $x = R * (longitude_0 - longitude)$
        $y = R * (\tan(latitude) - \tan(latitude_0))$

        with R the radius of Earth and $longitude_0$ the longitude of the left edge of the map, and $latitude_0$ the latitude of the bottom of the map.

Deliverables:

Michelle Song
Chen Ren
Final Project – Option 2

1. Functioning code – demonstration to Prof. Mackenzie required before submission
2. Detailed disclosure of who did which part of the project (if done as two person project).
3. Full documentation including
   - Interface layout – screenshot and all panels labeled with their respective variable name
   - Discussion on how sizers and plot scaling work
   - Screenshots for routes SEA-MIA and MIA-SEA
   - **TWO PERSON PROBLEM**: discussion on how image scaling and matching of map and plot was realized.

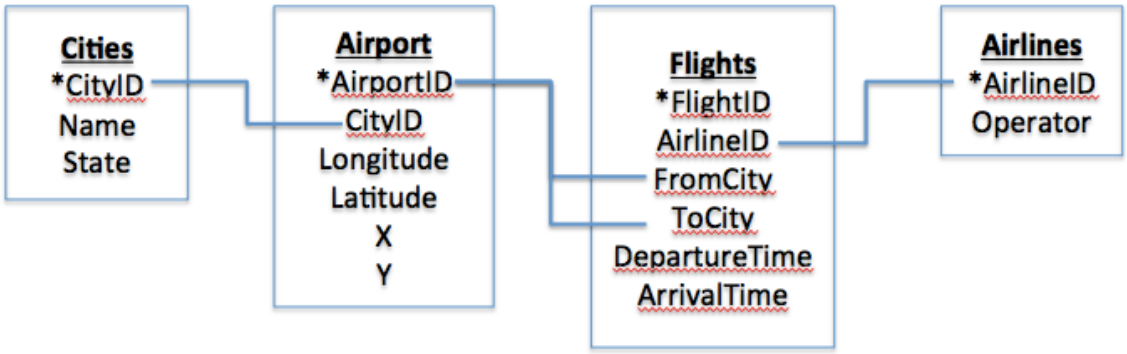
Algorithm

Graph.py

Graph.py has been used in multiple previous assignments so the algorithm will not be discussed in this report. Please reference (Michelle Song's) previous reports for Graph.py's algorithm.

Import.py

This was the same Import.py that was used for the midterm. Please reference (Michelle Song's) midterm report for algorithm.



Final.py

The class in Final.py is called Airport (QWidget)
The algorithm is similar to Assignment 6's algorithm. Here are the key differences.

1. Insert the qwidget created in Map.py into gui layout
   a. This is inserting the photo of the US map
2. Search shortest path (def search_shortest)
   a. Finds shortest path (using Graph.py)
   b. Output text (of shortest path)
   c. Calls Map.py to draw the shortest flight path
3. Search all flights (def search_all)

   a. Uses Graph.py to search all flight paths

Map.py

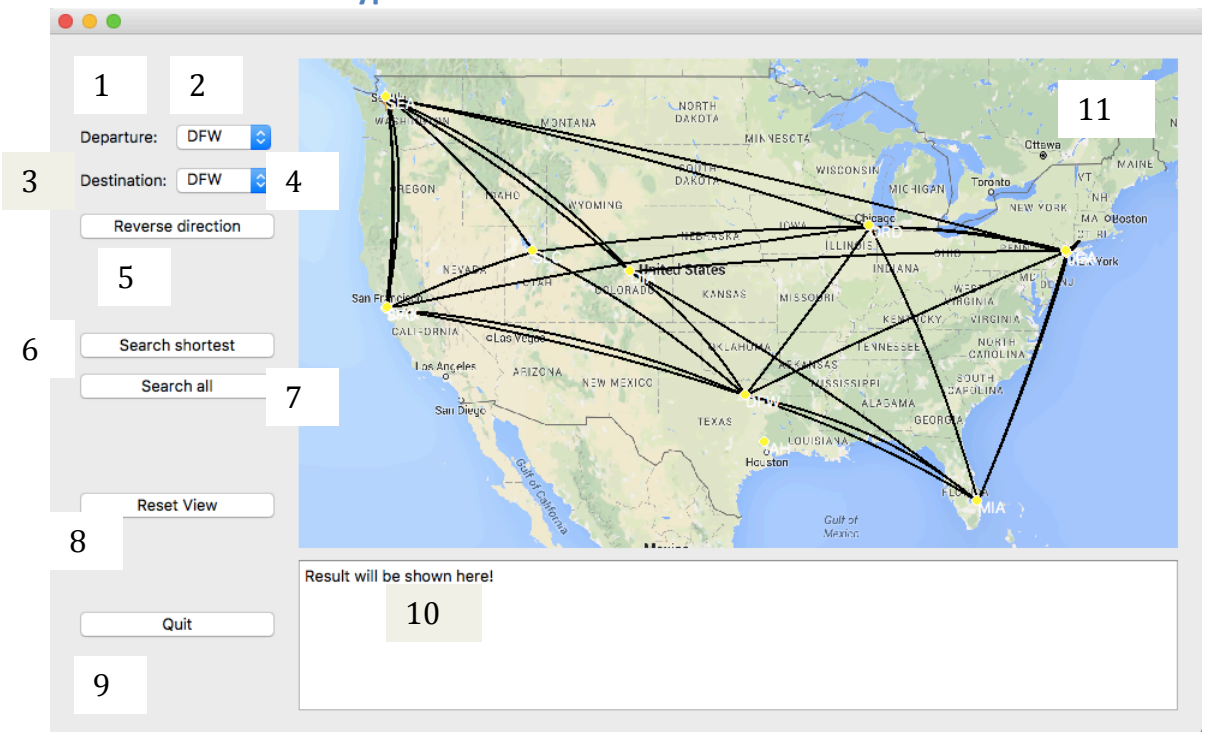
The class in Map.py is called Map (QWidget).

1. Initialize
   a. Def init (self, parent)
   b. Define self. variables
      i. self.shortest_flights = []
      ii. self.x = []
      iii. self.y = []
2. Set minimum size
3. def paintEvent
   a. Call QPainter
   b. Import USmap.png
   c. Call drawFlights (next def)
4. def drawFlights
   a. Call QPainterPath
   b. Input (self) size, height, and width into variables size, h, and w
   c. Define Rx, Ry in terms of w and h (used for longitude/latitude to x/y calculations)
   d. Set Longitude0 and Latitude0
   e. Call Graph.py to return a list of flights and list of airport positions
   f. Calculated x, y from longitude and latitudes
   g. Calculate mid points and adjust so that drawPaths draws curved paths
   h. Draw flight paths and plot airport positions
   i. Label airports with airport code
5. def drawshortestflight
   a. Call Graph.py for list of flights in shortest path
   b. Call self.draw to draw shortest path
6. def draw
   a. Takes list of shortest flights and draws them (in red). Uses same calculation/process as drawFlights (see number 4 above).
7. def resetFlights
   a. Set self.shortest_flights = []

Michelle Song
Chen Ren
Final Project – Option 2

Screenshots and Explanations

Variable names and type

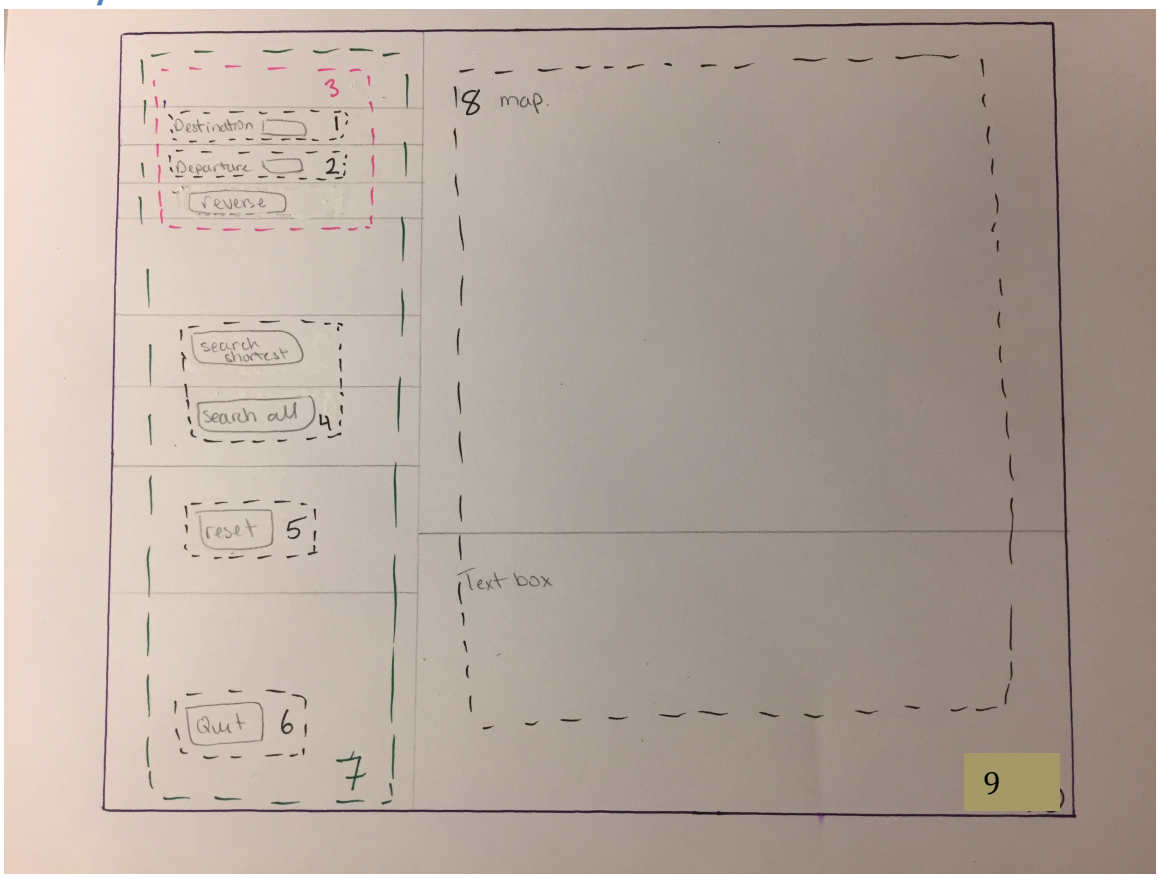


[1] label_1 (QLabel)
[2] self.departure (QComboBox)
[3] label_2 (QLabel)
[4] self.destination (QComboBox)
[5] pbtn_reverse (QPushButton)
[6] ptbn_short (QPushButton)
[7] ptbn_all (QPushButton)
[8] ptbn_reset (QPushButton)
[9] ptbn_quit (QPushButton)
[10] self.result (QTextEdit)
[11] self.map = Map(self)
Map.py used QPixMap

This is the main screen of our gui; this is what appears when you run Final.py. Airport names printed next to airport in white. Airports are denoted used a yellow circle. Flights are drawn using curved lines.

Michelle Song
Chen Ren
Final Project – Option 2

Gui Layout



[1] lyt_1_1 (QHBoxLayout)
[2] lyt_1_2 (QHBoxLayout)
[3] lyt_2_1 (QVBoxLayout)
[4] lyt_2_2 (QVBoxLayout)
[5] lyt_2_3 (QVBoxLayout)
[6] lyt_2_4 (QVBoxLayout)
[7] lyt_left (QVBoxLayout)
[8] lyt_right (QVBoxLayout)
[9] lyt (QHBoxLayout)

Scaling

To ensure that the drawn flight paths and plotted airport locations would scale appropriately, we calculated (x, y) in terms of the width and height of the Map widget. We used self.height and self.width to determine the width and height of the widget. The picture of the map was the only thing in the widget so we did not have to try and determine the size of the picture. The next section gives a more detailed breakdown of how we calculated x and y values using longitude and latitude values.

Michelle Song
Chen Ren
Final Project – Option 2

We used Map.py to create the map widget and also to draw on the map widget. This way, we did not have to consider how the overall gui scaled and only needed to consider how the map widget would scale.

Calculating (X, Y)

We used each airport's longitude and latitude values to calculate x and y values to plot on the map.

$$x = R_x * (Long0 - Long_{airport}) \quad \text{(Eq. 1)}$$

$$y = R_y * (\tan\left(Lat_{airport} * \frac{pi}{180}\right) - \tan\left(Lat0 * \frac{pi}{180}\right) \quad \text{(Eq. 2)}$$

We used the longitude and latitude values of SFO and LGA to calculate Rx and Ry. We determined (x, y) for LGA in terms of w and h. The latitude and longitude values for Ry were converted into radians.

$$R_x = \frac{x_{LGA} - x_{SFO}}{Lat0 - Lat_{LGA}} \quad \text{(Eq. 3)}$$

$$R_y = \frac{y_{LGA} - y_{SFO}}{\tan\left(Long_{SFO} * \frac{pi}{180}\right) - \tan\left(Long0 * \frac{pi}{180}\right)} \quad \text{(Eq. 4)}$$

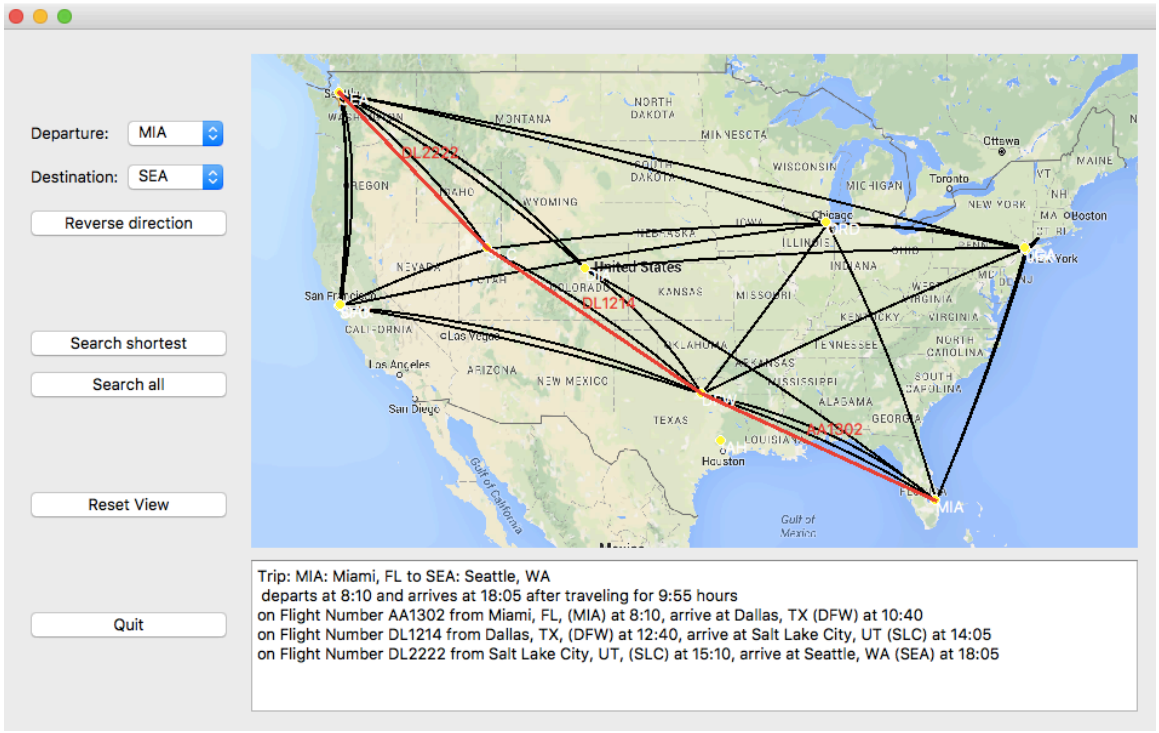
We used the SFO (San Francisco airport) location as our reference point of "0, 0".
We determined (x, y) for SFO in terms of the width (w) and height (h) of the map widget.
Thus Eq.1 and Eq 2 became

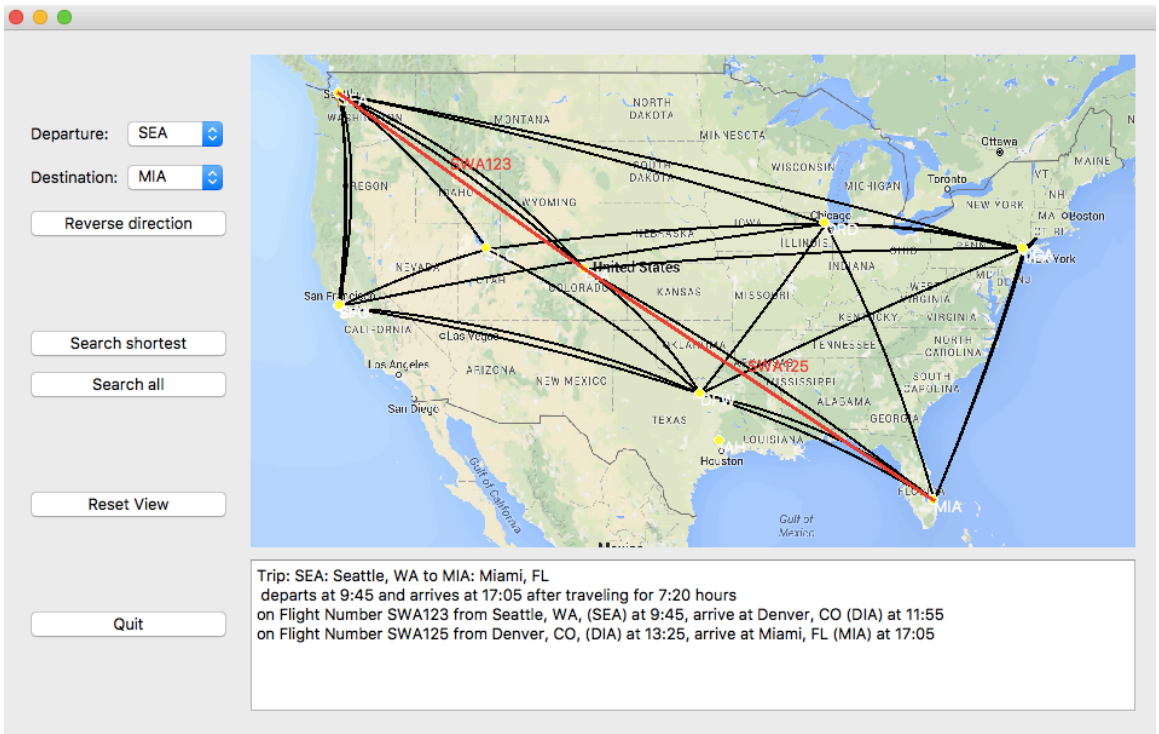
$$x = x_{SFO} + (R_x * (Long0 - Long_{airport})) \quad \text{(Eq. 5)}$$

$$y = y_{SFO} + (R_y * (\tan\left(Lat_{airport} * \frac{pi}{180}\right) - \tan\left(Lat0 * \frac{pi}{180}\right)) \quad \text{(Eq. 6)}$$

Michelle Song
Chen Ren
Final Project – Option 2

MIA-SEA Screenshots





Michelle Song
Chen Ren
Final Project – Option 2

Project Contribution

Much of the code was written together by Chen and Michelle. Chen created the gui layout and determined how to calculated (x, y) from (Long, Lat). Michelle determined how to draw all the flight paths (and later how to draw curved paths). We worked together for most of the project and would bound ideas off of each other. Therefore it is difficult to determine things that only 1 person contributed towards.