**GEOG5990 Assessment 2 - White Star Line project analysis**

This Assessment 2 project was created to provide a program for shipping company White Star Line to identify icebergs from radar and lidar images. This program was required as the company wished to send out iceberg-towing tug boats with each ship in order than potentially harmful icebergs could either be tugged out of the ship’s path, or the route altered as necessary. The program therefore needed to read in two files containing information on the texture of objects (radar file) and how high identified objects are (lidar file) in order to return information about iceberg size.

**Designing the program**

Before writing the program, a mind map was created, detailing the steps that would be compulsory to the program and beginning to place them in an order. It was concluded that the radar file would firstly need to be scanned for the presence of icebergs, and then the location recorded or returned. Icebergs that had been found in the radar file would need to be counted and its metadata assigned to the equivalent iceberg in the lidar file using the previously recorded location. It was also noted that there must be the establishment of a method which meant icebergs and iceberg metadata that had already been recorded was not counted again. This mind map, combined with flow charts which showed interactions between functions and sections of code, which later was developed into the program’s UML diagram, became the framework for the software development.

**Software development and resolved issues**

All outcomes detailed in the Assessment 2 White Star Line criteria have been achieved and the final program can be viewed using icebergchecker\_gui.py. In the program, a radar image of 300m by 300m of ocean is assessed for areas with the presence of icebergs. The total volume of the iceberg is then calculated using data obtained from the lidar file (dimension, total mass and height values) and the mass density of ice (900 kg/m3) which was provided in the assessment brief. The total mass, volume and tow-ability of all identified icebergs are then displayed on a GUI. These can be viewed either by choosing the ‘Show tow-ability values’ in the GUI drop-down menu, or clicking the ‘Show tow-ability values’ button. This textual information is also saved to a text file called icebergmetadata.txt. The additional assessment criteria have also been met as the program can read in a file with multiple icebergs and return a tow-ability map which is colour coordinated to show towable icebergs in green and un-towable ones in red. (Further discussion on this is found in the ‘Known errors and areas for improvements’ section). If the user wishes to use a personal radar and lidar files, they MUST change the name of the input file in icebergchecker\_gui.py. (Change line 22 for the radar file and line 41 for the lidar file.) The number of rows and columns will automatically update based on the input radar file provided.

Due to altering the IPython graphics backend throughout development, and that the final program displays information onto a GUI, one of the initial outcomes - displaying the original two data files - is only possible if using working code before the GUI was implemented. This file is provided and is called icebergchecker.py. On icebergchecker.py lines 242 and 246 can be commented out (to prevent the display of the berg tow-ability map) and either the radar or lidar file displayed instead. This can be done by uncommenting lines 34-36 or lines 48-50 respectively.

Throughout software development, the program was written in sections to ensure that new ideas were working and to reduce the risk of errors. Additionally, print statements were used in development to check loops we were working as planned and functions were being called when needed. Some of these print statements remain so that the user can continue to understand the structure of the code, others were only relevant during development and have since been removed. Writing this program also required learning programming techniques that were not required for the agent-based modelling program in Assessment 1. For example, it was learnt that variables which are altered as part of a nested loop have to be defined within their relevant loop in order to loop successfully, else the counters that loop through each variable will not have access to the updated variable. This was learnt in order that lines 156-162 (of icebergchecker\_gui.py) of the berg\_footprint function would look at every cell in the iceberg. Another learning point was understanding how to set up a blank 2D array. This was not necessary in Assessment 1 however it was necessary in order to set up a 300 by 300 grid of ones which became the basis of the figure of berg tow-ability (line 246 in icebergchecker\_gui.py).

**General sources**

In addition to the resources provided on the ‘Programming for Geographical Information Analysts: Core Skills’ webpages (<https://www.geog.leeds.ac.uk/courses/computing/study/core-python/>), a couple of online sources were used throughout documentation. These included both the official python documentation and Tutorials Point’s web pages. These were particularly helpful when understanding alterable options in methods, for example, the Tkinter pack() method, and when setting up textual outputs such as the write() method and the Tkinter.Label() method. Forums on Stack Exchange (<https://stackexchange.com/>) were helpful as they were discussions regarding similar, but not identical, complexities in development. This resource was particularly used when embedding the figure of berg tow-ability onto the GUI window and writing single lines of text with multiple unspecified values to both the GUI and a text file.

**Known errors and areas for improvements**

The program was written so that the output figure of berg tow-ability has three values – 0 (not towable), 1 (ocean) and 2 (towable). Each value has been assigned a corresponding colour – red (not towable), blue (ocean) and green (towable). This framework always displays a figure but only has the correct colour formatting if all three outcomes are present. In the event that there is only one type of iceberg (towable or un-towable) then the output figure shows as red and green, the two colours at either end of the colour scale. For clarity, future work should ensure that the correct colours are still assigned even if only one type of iceberg is identified.

This code was written with the intention that it could read in any radar and lidar files with unknown quantities of icebergs and still compute an output figure of iceberg tow-ability and save the iceberg metadata to a text file. This code however assumes that all icebergs will only ever be uniformly square in shape. A counter of the number of rows should be added if it was expected that icebergs could be other shapes.