Programming Project - Group 23

Peter, Vaibhav, Seán, Daire, and Kate

# **Overview**

As a group, we set out to create a program that fit the project specification. In short, an application that reads in data from a file “gcat.tsv”, renders it, and allows the user to interact with it. More specifically, we wanted to make the data as interactive as possible by allowing the user to filter the data by almost any query imaginable.

Our program implements a dataset of 50k entries, but also allows the user to choose smaller versions of the dataset. We avoided the use of libraries as much as possible other than a few ones necessary in order to demonstrate our coding capabilities.

# **Design**

In the first meeting, we discussed what we thought would be the best way to store the data, and which fields we wanted to focus on. Peter suggested the idea of using a hashmap for the dataPoint class, which would make it easy to store and access the data, as well as allowing us to read in all of the fields, meaning we could decide which ones we wanted to use or ignore later, without having to make any changes.

The final design was decided on from the second meeting. We decided on a simple design in order to fit the project outline. A sketch of the initial design is included below.

We agreed on a single screen design that allows the user to interact with the data using different widgets such as checkboxes, radio buttons, drop down menus, and sliders. As seen in the screenshots further below, we closely stuck to this design.

We changed the colour scheme a few weeks into the project to make the interface aesthetically pleasing and readable. After some research we found the best colour schemes didn’t use bold colours and didn’t contain too many different colours. Before we changed the colour scheme the interface mainly consisted of a black background and a deep red secondary colour. We decided on a grey background as it is not too distracting and goes with any secondary colour. The rest of the colour scheme we avoided any colours that didn’t contrast drastically from the background this made the user interface readable and aesthetic.

When the program is launched, a loading screen appears. When the program is fully loaded, the main screen appears with the default set of queries.

The main program gives an array of possibilities to the user when it comes to interacting with the data and filtering through it. Here are some of the main ones:

#### Graphs:

The main visual representation of the data in the program is the bar chart in the top left corner of the screen. It displays the results to whatever filters that the user applies. There is also the option to visualise the data in a pie chart.

#### Drop Down Menus:

There are seven separate drop down menus in our program. Four of them allow the user to modify the barchart, allowing us to choose how we would like the satellites to be grouped, to choose the dataset to use, to view the total or average of any variable by any other variable.

Another two are associated with the search bar, allowing the user to choose what variable to search through, and whether the results should start with or contain the search query.

The last one allows the user to choose between a bar chart or a pie chart.

#### Scrolling:

As there is a large amount of features, we allow the user to scroll through lists that are too big to fully draw to the screen. Examples of this are the graph, some of the drop down menus, and the checkboxes. The graph and the checkboxes include scrollbars which allows the user to scroll by dragging the mouse.

#### Sliders:

Each slider has two slider widgets which can be dragged using the mouse, and collide with each other. The data key is used to determine which field the slider would interact with (e.g. apogee), and it uses this key to loop through the array of data points to find the minimum and maximum values for the field. Interacting with the slider causes the data to be filtered, based on the current min/max values.

#### Checkboxes and Radio buttons:

The checkboxes and radiobuttons further allow the user to filter through the data, the radio buttons also feature a disappearing description of the flags when the mouse is hovering over the flag labels.

Search Bar:

The search bar allows the user to provide input to filter the data, and has options “contains”, “starts with”, and “ends with”.

DataPoint displayer and Pop-up window:

The datapoint displayer is used to display all of the data points contained within a particular bar that the player has selected on the graph. The displayer can be selected by the user, which causes the pop-up window to appear, showing all of the data contained within that particular datapoint.

# Teamwork

After receiving our group allocation for this project, Peter created a discord for our group which we then used to organise meetings.

Our teamwork for this project was mainly based on bi-weekly in-person meetings outside of the labs. Peter is the most experienced programmer in our group and delegated tasks to the rest of the team. This dynamic allowed us to get a lot of work done as each of us had clearly defined jobs to do.

# Individual Contributions

#### Peter

I worked on the backend (e.g. processing data, filtering data) and on the frontend (e.g. graphs, dataPointDisplayer). I added a dropDownMenu class, which knows its current option, hides the unselected options when not active, and can be scrolled either by mouse wheel or by a scrollbar (as can the graph). I created the functions to read in the file and then process the data, which is stored in dataPoint objects. Each one has a hashmap of the variable name to that variable’s value for easier processing. I created a graph class to represent these, either as a bar or pie chart, and added a scale for the bars and key for the pie chart. Below I added a dataPointDisplayer, showing all non-filtered space objects, which can be clicked to create a window which is easier to read. I added a drop down menu on the top left to control the x variable, which is also the variable by which the objects are sorted. I worked to ensure that the program was efficient, e.g. only sorting when the file / variable was changed (all files were read and cached at the start using a hashmap). I made the bars hoverable (for the exact value) and clickable (so only those bar’s values appeared in the dataPointDisplayer. To make the filters work efficiently, I gave each dataPoint a hashset of related inactiveFilters. Whenever a filter’s state was changed (e.g. a checkbox clicked, a slider released), I iterated through the dataPoints, and if that filter would deactivate an active dataPoint, it was added to the list of inactive filters. A dataPoint is active if it has no inactiveFilters. I also added to the string filter / checkboxes, e.g. making them scrollable, and fixed a number of bugs. I added two more drop down menus between the search bar and the sliders, which allow for more advanced queries. The first allows you to choose between having the y variable be the total of the variable in the second drop menu, or the average, by the x variable. By default, the y variable is a count of the x variable, but you can for example choose to have a graph of total/average mass by state. The y variable can be any numeric variable.

#### Vaibhav

For this project, I added two classes mainly: the checkbox class, the search bar class, and a little bit of the StringFilter.Firstly, In the checkbox class, I made a single checkbox with an "X" inside it and when pressed upon the checkbox all the space objects whose string filter variable matches the inactive checkbox will be hidden. Then I created an ArrayList of checkboxes which is implemented in the StringFilter class. Secondly, the search bar class allows the user to filter the data by providing input. Lastly, the String Filter class allows us to filter out the objects based on strings such as state, status, etc. I had just made the beginning of the class and the constructor. It was later on completed by peter.

#### Seán

As part of the project I added the scrollBar class, the date class and added the alien png. The scrollBar class added the three implementations of a scroll bar two of which were vertical and one horizontal.The date class organised dates from lowest to highest by separating days, months and years into integer values and adding them all together. I found it difficult to find an image of an alien that was suitable for the colour scheme as many online did not have transparent backgrounds and didn’t fit the aesthetic of the rest of the project. I decided the best way to solve this problem was to create my own image using drawing software and make the background the same colour as the colour scheme of the project.

#### Daire

For the project, I made the bar class and the slider class. The bar class draws simple bars with a width, height, colour and label, which is drawn below the bar. The bar class is used in the graph class, which was implemented by Peter. I also made the slider class. The slider can be used for any field with a type of int, float or date. Since date is quite different to int and float, I had to use different variables as well as using a slightly different method to display the current date based on the position of the slider by using the date class that Seán made.

#### Kate

My contributions to the project were mainly focused around the flag filters in the main program. I began by creating individual radio widgets, and then a boolean filter class that implements them. As I don't believe that the use of the flags is common knowledge, I added a feature that shows more information about them if the mouse is hovering over the boolean filter's label.

# Issues

* Fixed issue with max date on slider relating to object references
* Fixed issue with pop-up text box crashing main program
* Fixed issues with efficiency, mainly when changing to largest file
* Fixed issue with running the program on mac OS (due to differences in storing new lines)
* Fixed issues with scrolling - sometimes the last few elements could not be seen, while at other times you could scroll past the array, causing a null pointer exception.

# Conclusion

Overall, we were very happy with how the project turned out. With our initial design plan being a single screen that would display a graph type, as well as a large number of filters, we feel the result is exactly what we had hoped for. We feel that the filters are all intuitive, and despite the screen containing such a large number of options, it isn’t overwhelming thanks to clear labels and nice combinations of colours. The group worked very well together over the course of the project, with us running into no real problems regarding collaboration. Everyone attended weekly scheduled meetings and contributed to the program.



