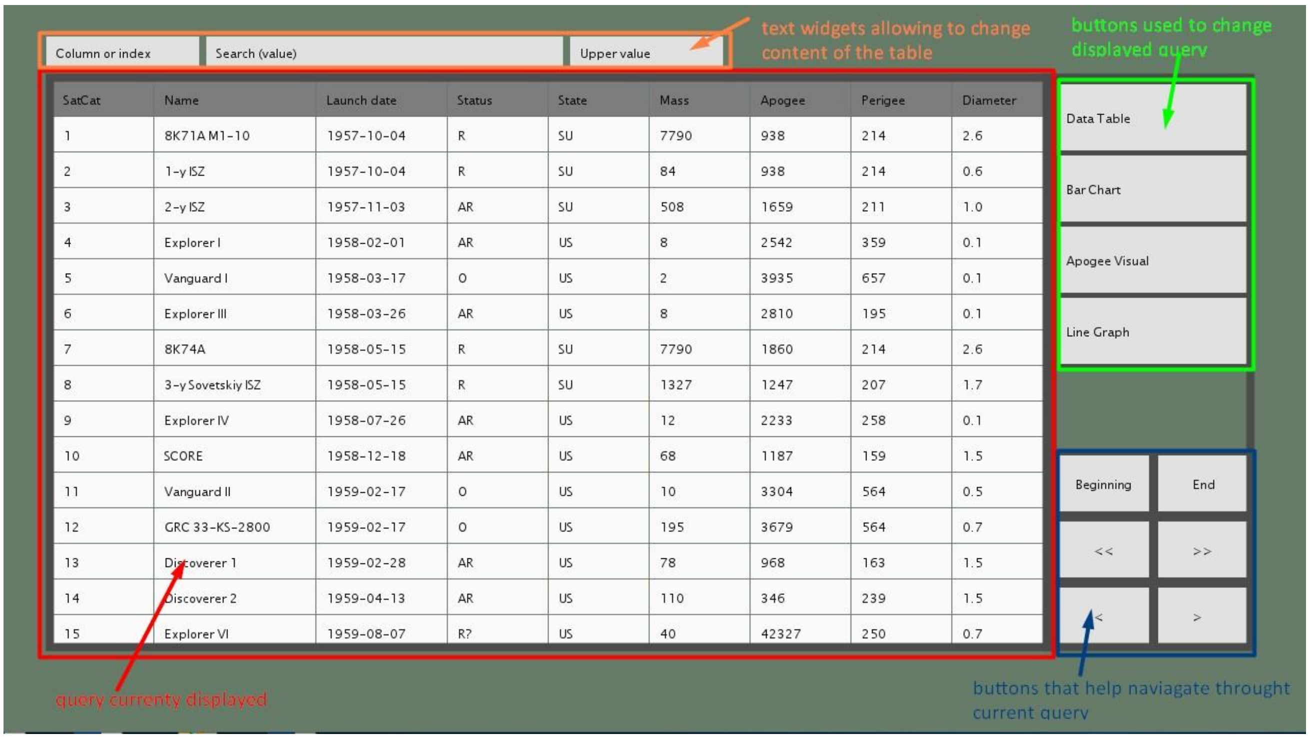
Report - Group 12

Group Members

Antoni Zapedowski
Noah McNamara
Liam Murphy
James Doyle

Outline of the design

We have decided that the program which we would create will have one screen, with the ability to change what is currently displayed.
After starting the program the first query noticeable will be the table with the most vital information about certain space objects (e.g name, mass, state, status). The table displays up to 15 objects at once.
Above the table, there are 3 text widgets responsible for searching for space objects containing a certain value/range of values that modify content of that query.
Next to the right of the table, there is another section in which there are a few interactive widgets that allow changing what query is currently displayed by the program. Additionally below the query buttons if the user is currently displaying data table there are interactive buttons that allow the user to change what is displayed in the table. (e.g see the next 15 space objects in the table).
Below you can see an overview of how the program looks split into the most important sections.


Column or index
Search (value)
Upper value
text widgets allowing to change content of the table
buttons used to change displayed query
Data Table
Bar Chart
Apogee Visual
Line Graph
Beginning
End
<<
>>
<
>
query currenty displayed
buttons that help naviagate throught current query

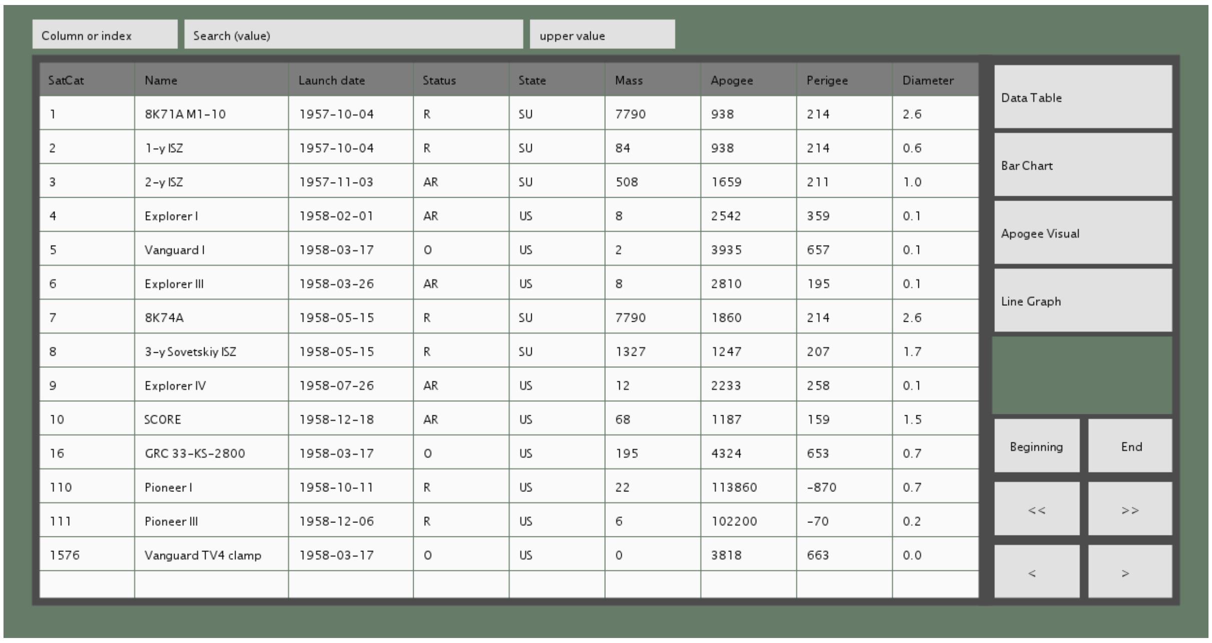
SatCat	Name	Launch date	Status	State	Mass	Apogee	Perigee	Diameter
1	8K71A M1-10	1957-10-04	R	SU	7790	938	214	2.6
2	1-y ISZ	1957-10-04	R	SU	84	938	214	0.6
3	2-y ISZ	1957-11-03	AR	SU	508	1659	211	1.0
4	Explorer I	1958-02-01	AR	US	8	2542	359	0.1
5	Vanguard I	1958-03-17	O	US	2	3935	657	0.1
6	Explorer III	1958-03-26	AR	US	8	2810	195	0.1
7	8K74A	1958-05-15	R	SU	7790	1860	214	2.6
8	3-y Sovetskiy ISZ	1958-05-15	R	SU	1327	1247	207	1.7
9	Explorer IV	1958-07-26	AR	US	12	2233	258	0.1
10	SCORE	1958-12-18	AR	US	68	1187	159	1.5
11	Vanguard II	1959-02-17	O	US	10	3304	564	0.5
12	GRC 33-KS-2800	1959-02-17	O	US	195	3679	564	0.7
13	Discoverer 1	1959-02-28	AR	US	78	968	163	1.5
14	Discoverer 2	1959-04-13	AR	US	110	346	239	1.5
15	Explorer VI	1959-08-07	R?	US	40	42327	250	0.7

Table and table search

As previously mentioned the data table was our first core query where we just wanted to display as much basic data to the user as possible and to allow them to quickly create additional queries to find out more information.
The tables structure was created in the TableGUI class which contained methods to generate the table and sort its values. To create the grid we used a 2d array of widgets to create headers, rows, and columns that could be adjusted using the constructors input values. These widgets would then get a starting space object index and occupy each row after that with the next corresponding space objects data separated into each column category.
The first way of interaction with the data we implemented was the use of buttons/scrolling to navigate the query which would increment the starting space object index and regenerate the table. Secondly, the user can select any header to sort the data from highest to lowest, lowest to highest, or back to default. To do this we had to overwrite a sort method and then display a temporary array of sorted space objects depending on which header detected a click event.
Where we went beyond the specification:

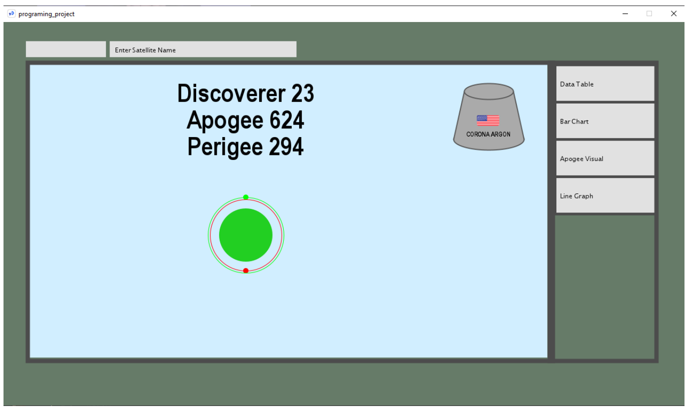
- Sorting by header allowed an extremely simple way to query data in 18 different ways all in the same table.
- Furthermore, the different types of navigation buttons were another quality of life feature that made interaction with our project easier

Above the table, there are 3 text widgets used to search for a certain range (of space objects) or space objects that meet the condition. The first widget asks the user to input the column name (case ignores) or the index of the table. This was done to make search function more efficient, especially in case of the data with 50k entries. The second widget is the value input (e.g. a certain date in right format, or an int while looking for mass). The last widget is used when the user wishes to display a range of values (e.g space objects with date 1957-10-04 to 1959-01-01 - result of this will be shown below), however, this can be used only in certain columns (containing date or numbers, not strings). Once the results are displayed they can still be sorted.



SatCat	Name	Launch date	Status	State	Mass	Apogee	Perigee	Diameter
1	8K71A M1-10	1957-10-04	R	SU	7790	938	214	2.6
2	1-y ISZ	1957-10-04	R	SU	84	938	214	0.6
3	2-y ISZ	1957-11-03	AR	SU	508	1659	211	1.0
4	Explorer I	1958-02-01	AR	US	8	2542	359	0.1
5	Vanguard I	1958-03-17	O	US	2	3935	657	0.1
6	Explorer III	1958-03-26	AR	US	8	2810	195	0.1
7	8K74A	1958-05-15	R	SU	7790	1860	214	2.6
8	3-y Sovetskiy ISZ	1958-05-15	R	SU	1327	1247	207	1.7
9	Explorer IV	1958-07-26	AR	US	12	2233	258	0.1
10	SCORE	1958-12-18	AR	US	68	1187	159	1.5
16	GRC 33-KS-2800	1958-03-17	O	US	195	4324	653	0.7
110	Pioneer I	1958-10-11	R	US	22	113860	-870	0.7
111	Pioneer III	1958-12-06	R	US	6	102200	-70	0.2
1576	Vanguard TV4 clamp	1958-03-17	O	US	0	3818	663	0.0

Apogee Visualizer



Discoverer 23
Apogee 624
Perigee 294

The goal behind the apogee visualizer was to display orbital data graphically in a more meaningful manner than just data in a table. We decided on a 2D representation where we displayed earth and then two ellipses around it, one for perigee and one for apogee.

To calculate these heights we first needed a specific space object's data so two search bars were implemented, the query would be compared with the entire database, and if a match was found that space object's important data would be stored in the class. The orbits and name of the object are then displayed by adding to the width of an ellipse the values scaled down by 100.
This worked well for most entries but one problem in this design was with extreme values as certain objects would have too large of an apogee which would place the orbit of the screen.

The second part of the query is satellite visualisation. This was done by getting the shape, diameter, length, and state data from the selected object. We created graphics for the 5 most common shapes and would then display them with altered dimensions depending on the diameter and length. This also worked out pretty well however we didn't have enough time to create designs for every shape, and we couldn't accurately depict the dimensions as some satellites would be too small to see so a base size was decided on and then the object would get its length added to that value.

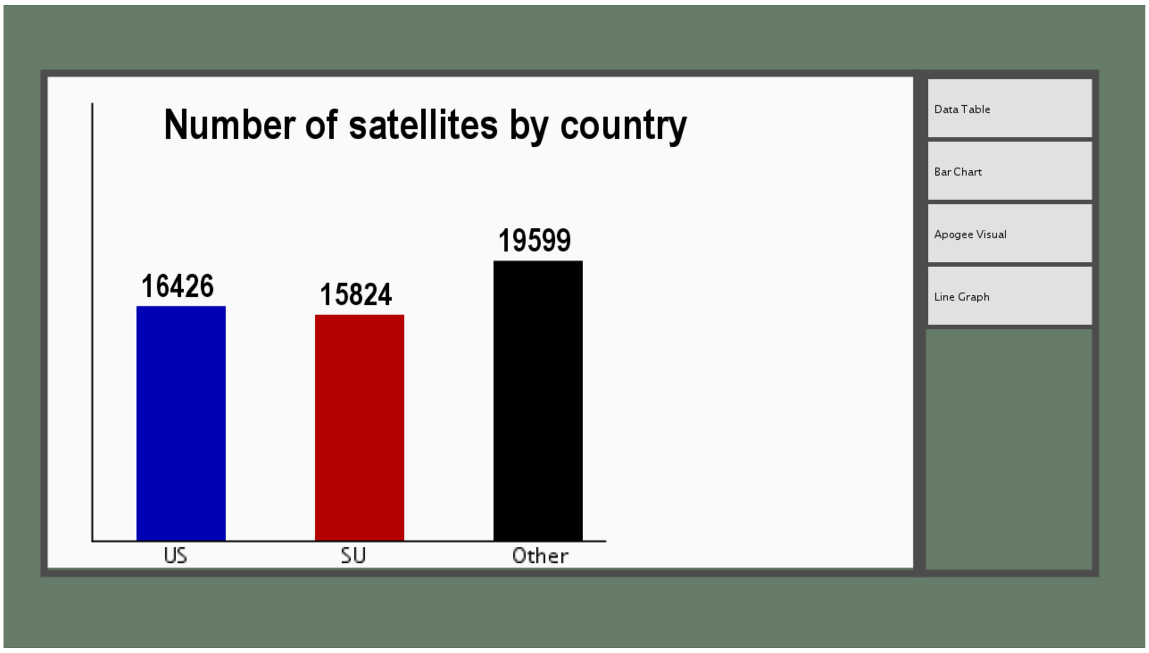
Where we went beyond the specification:

- Extra focus on making our graphical display detailed enough to get across as much information while also being simple to understand
- This specific case was also great as being to see orbits and what the satellite actually looked like gives the user an extra level of understanding that other graphical representations might not give

Bar Chart

The bar chart is designed to show a breakdown of the number of launches per country over time. This is achieved by storing all the countries to be counted in a hashmap and then moving through the entire dataset and incrementing each value as they are detected. We decided to choose three values to be displayed as bars: satellites launched by the US, satellites launched by the Soviet Union, and any satellites launched by other countries.

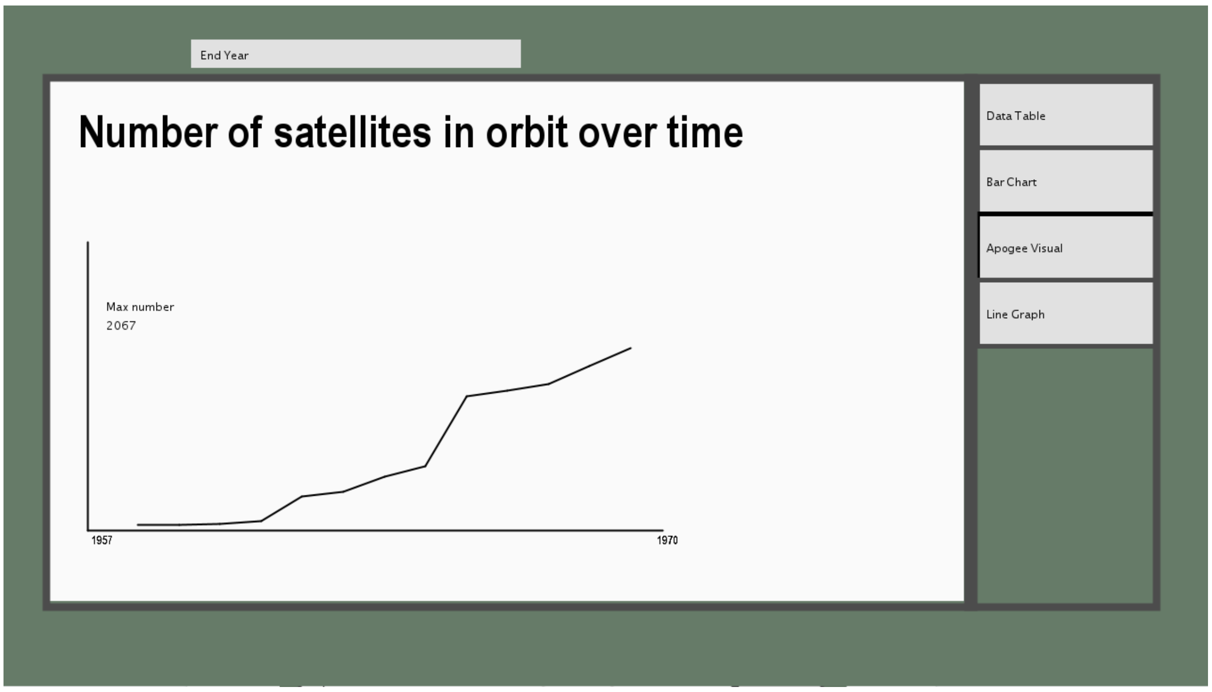
These values were selected as they equate to bars of roughly equal size.
To calculate the size of the bars we took the count for each of the data values selected and multiplied it by the scale chosen in the constructor.



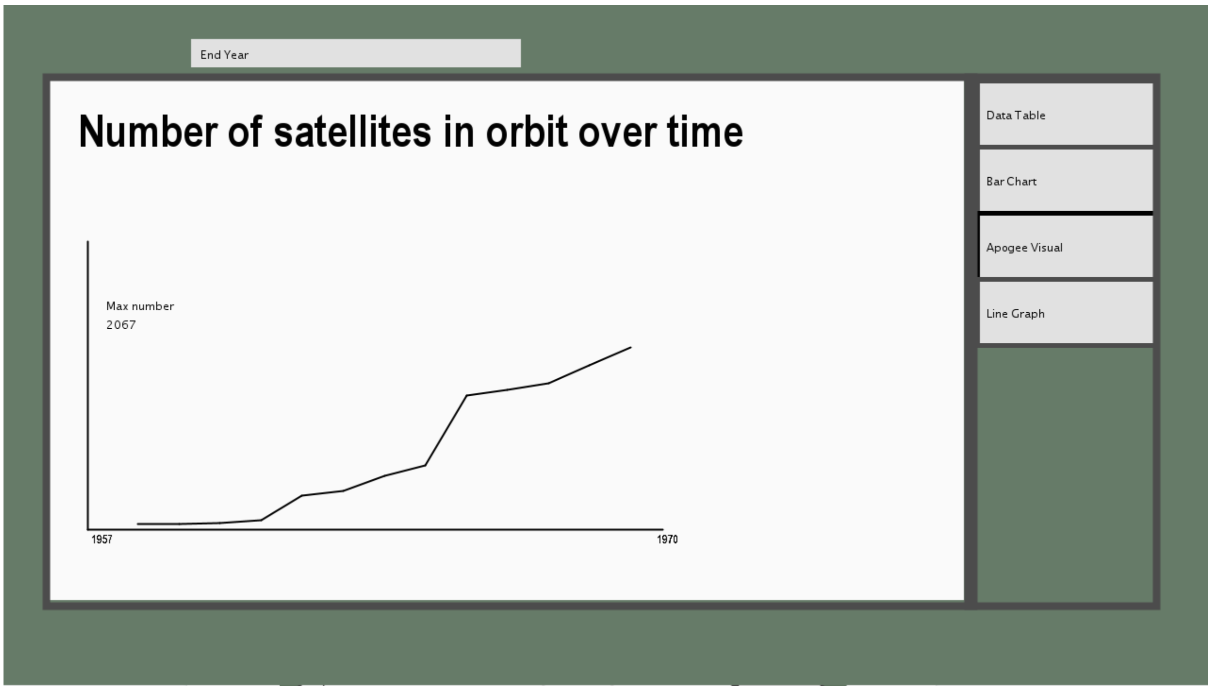
Line Graph

The line graph displays the number of satellites that are currently in orbit during any given year. The user can then set their desired end year for the graph and it will dynamically change to reflect this input and the size of the graph will remain constant.

To create the graph a loop is created that adds a key to a hashmap for each year in the date range selected, it then goes through the entire dataset and adds to the key if there are any satellites with a launch date in that year and subtracts if there are any with a decommission date. Another hashmap is then created that stores the X value as the key and the corresponding Y value which is calculated using the count of how many satellites are in orbit that year. The lines are then drawn in a loop that takes the previous coordinate and the current coordinate and multiplies them by a scaler that is calculated by dividing the X scale of the graph by the number of years that are in the date range selected. This allows the graph to stay the same size when a new date range is selected. This is where we went above and beyond with this query as we could have let the graph change size but we decided it would be more user-friendly to have it stay the same size.



End year 1970 selected.



End year 2022 selected.

Overall our project went beyond by focusing on having intuitive user interaction, UI and design while also containing more than enough meaningful and cohesive ways to query and display the provided data.