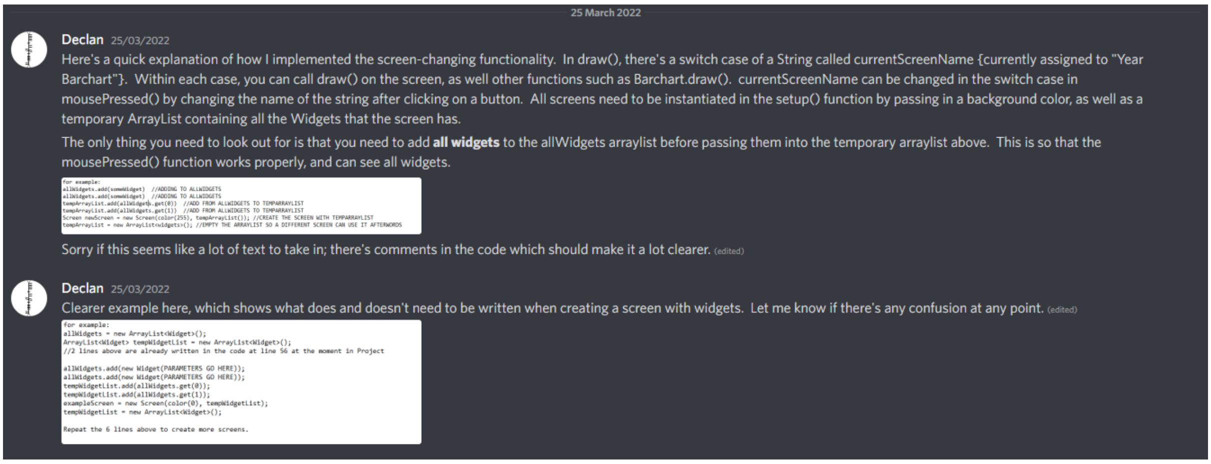
Programming Project - Group 8

Declan Michael McCabe + Mia Cimas + Darius Pop + Kostiantyn Ohorodnyk

Introduction

Our team consists of Declan McCabe, Mia Cimas, Darius Pop and Kostiantyn Ohorodnyk. We communicated using a Discord server and also had in-person meetings every Tuesday after the lectures and online meetings on Wednesdays. In those meetings we would decide on weekly goals, and split up the work amongst ourselves, in order to make sure we had most of the work done before the start of the Thursday labs. Over the weekends, we worked on finishing off and polishing the work we had done in that week. We documented our code using an "explaining-code" channel in Discord, where we would tell the rest of the team how to implement and work with the code we have written. Below is one such example of documentation from that channel.



25 March 2022

Declan 25/03/2022

Here's a quick explanation of how I implemented the screen-changing functionality. In draw(), there's a switch case of a String called currentScreenName {currently assigned to "Year Barchart"}. Within each case, you can call draw() on the screen, as well other functions such as Barchart.draw(). currentScreenName can be changed in the switch case in mousePressed() by changing the name of the string after clicking on a button. All screens need to be instantiated in the setup() function by passing in a background color, as well as a temporary ArrayList containing all the Widgets that the screen has.

The only thing you need to look out for is that you need to add **all widgets** to the allWidgets arraylist before passing them into the temporary arraylist above. This is so that the mousePressed() function works properly, and can see all widgets.

```
for example:
allWidgets.add(someWidget)  //ADDING TO ALLWIDGETS
allWidgets.add(someWidget)  //ADDING TO ALLWIDGETS
tempArrayList.add(allWidgets.get(0))  //ADD FROM ALLWIDGETS TO TEMPARRAYLIST
tempArrayList.add(allWidgets.get(1))  //ADD FROM ALLWIDGETS TO TEMPARRAYLIST
Screen newScreen = new Screen(color(255), tempArrayList()); //CREATE THE SCREEN WITH TEMPARRAYLIST
tempArrayList = new ArrayList<widgets>(); //EMPTY THE ARRAYLIST SO A DIFFERENT SCREEN CAN USE IT AFTERWORDS
```

Sorry if this seems like a lot of text to take in; there's comments in the code which should make it a lot clearer. (edited)

Declan 25/03/2022

Clearer example here, which shows what does and doesn't need to be written when creating a screen with widgets. Let me know if there's any confusion at any point. (edited)

```
for example:
allWidgets = new ArrayList<Widget>();
ArrayList<Widget> tempWidgetList = new ArrayList<Widget>();
//2 lines above are already written in the code at line 56 at the moment in Project

allWidgets.add(new Widget(PARAMETERS GO HERE));
allWidgets.add(new Widget(PARAMETERS GO HERE));
tempWidgetList.add(allWidgets.get(0));
tempWidgetList.add(allWidgets.get(1));
exampleScreen = new Screen(color(0), tempWidgetList);
tempWidgetList = new ArrayList<Widget>();

Repeat the 6 lines above to create more screens.
```

Screen Organisation

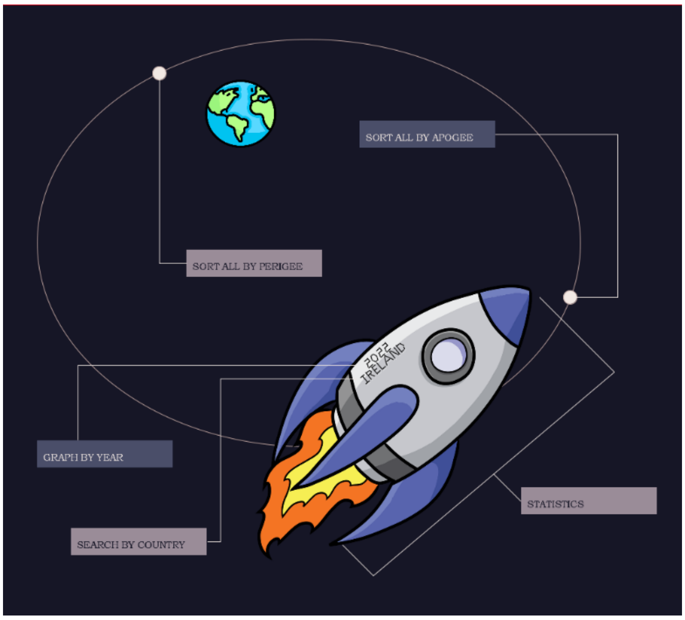
The screen class written by Declan determined a lot of how we would structure our program. This class is used to organise each screen's widgets and the background colour. The class also checks if the mouse is hovering over any of the widgets, changing the cursor image from an arrow to a hand if that is the case.

The setup for each Screen takes place in the setup() function, called at the start of the file, where an ArrayList of the widgets is passed into the Screen's constructor. A switch statement in the draw() function determines which screen should currently be drawn. It is at this stage where we first noticed a problem in our implementation. As convenient as it is to have the Screen handle the drawing of the widgets, it does have its limitations in terms of control over what order items are drawn in. For this reason, Declan decided to add support for widgets being added to a screen, but drawn somewhere that isn't the Screen's own draw() method. This allowed us to have much more control of the layers on which widgets were drawn, which was particularly helpful in the Statistics Screen. A simple event handler in the mousePressed() function was used to take care of screen switching, as well as handling user queries.

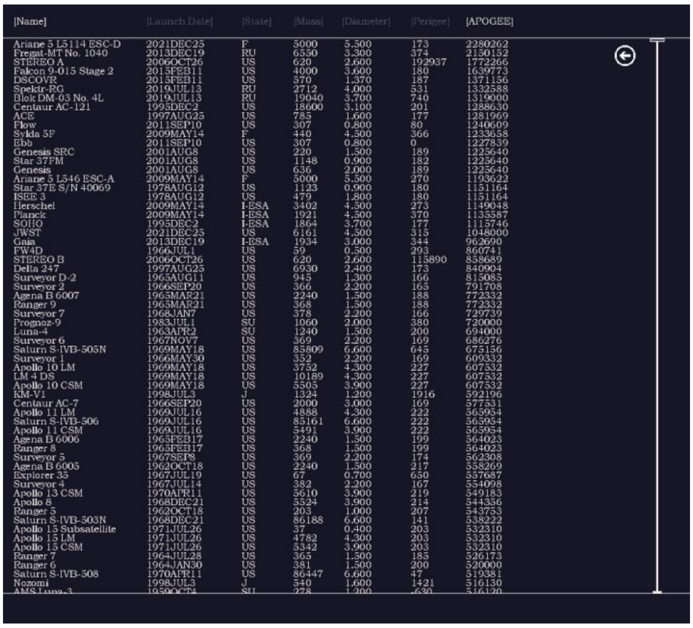
To create return buttons, Mia created a PhotoWidget subclass of Widget. The subclass inherits all instance variables and methods from its superclass and overrides the draw() method by displaying a selected picture instead of a default rectangular button.

Start Screen

When we first started the project, we were more focused on the functionality of the program than on its visual aspects. Around week 4, after we finished implementing all the queries, Darius and Mia decided to focus on creating a more visually appealing layout and colour palette. They worked on the design of the start screen and added pictures, widgets, and lines to make it look more topic related. All the created widgets open different screens, which present data in various ways.



Sort by Apogee/Perigee Screens

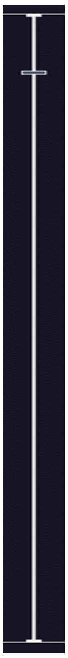


[Name]	[Launch Date]	[State]	[Mass]	[Diameter]	[Perigee]	[APOGEE]
Ariane 5 L5114 ESC-D	2021DEC25	F	5000	5.500	173	2280262
Fregat-MT No. 1040	2013DEC19	RU	6550	3.300	374	2150152
STEREO A	2006OCT26	US	620	2.600	192937	1772266
Falcon 9-015 Stage 2	2015FEB11	US	4000	3.600	180	1639773
DSCOVR	2015FEB11	US	570	1.370	187	1371156
Spektr-RG	2019JUL13	RU	2712	4.000	531	1332588
Blok DM-03 No. 4L	2019JUL13	RU	19040	3.700	740	1319000
Centaur AC-121	1995DEC2	US	18600	3.100	201	1288630
ACE	1997AUG25	US	785	1.600	177	1281969
Flow	2011SEP10	US	307	0.800	80	1240609
Sylda 5F	2009MAY14	F	440	4.500	366	1233658
Ebb	2011SEP10	US	307	0.800	0	1227839
Genesis SRC	2001AUG8	US	220	1.500	189	1225640
Star 37FM	2001AUG8	US	1148	0.900	182	1225640
Genesis	2001AUG8	US	636	2.000	189	1225640
Ariane 5 L546 ESC-A	2009MAY14	F	5000	5.500	270	1193622
Star 37E S/N 40069	1978AUG12	US	1123	0.900	180	1151164
ISEE 3	1978AUG12	US	479	1.800	180	1151164
Herschel	2009MAY14	I-ESA	3402	4.500	273	1149048
Planck	2009MAY14	I-ESA	1921	4.500	370	1135587
SOHO	1995DEC2	I-ESA	1864	3.700	177	1115746
JWST	2021DEC25	US	6161	4.500	315	1048000
Gaia	2013DEC19	I-ESA	1934	3.000	344	962690
FW4D	1966JUL1	US	59	0.500	293	860741
STEREO B	2006OCT26	US	620	2.600	115890	858689
Delta 247	1997AUG25	US	6930	2.400	173	840904
Surveyor D-2	1965AUG11	US	945	1.300	166	815085
Surveyor 2	1966SEP20	US	366	2.200	165	791708
Agena B 6007	1965MAR21	US	2240	1.500	188	772332
Ranger 9	1965MAR21	US	368	1.500	188	772332
Surveyor 7	1968JAN7	US	378	2.200	166	729739
Prognoz-9	1983JUL1	SU	1060	2.000	380	720000
Luna-4	1963APR2	SU	1240	1.500	200	694000
Surveyor 6	1967NOV7	US	369	2.200	169	686276
Saturn S-IVB-505N	1969MAY18	US	85809	6.600	645	675156
Surveyor 1	1966MAY30	US	352	2.200	169	609332
Apollo 10 LM	1969MAY18	US	3752	4.300	227	607532
LM 4 DS	1969MAY18	US	10189	4.300	227	607532
Apollo 10 CSM	1969MAY18	US	5505	3.900	227	607532
KM-V1	1998JUL3	J	1324	1.200	1916	592196
Centaur AC-7	1966SEP20	US	2000	3.000	169	577531
Apollo 11 LM	1969JUL16	US	4888	4.300	222	565954
Saturn S-IVB-506	1969JUL16	US	85161	6.600	222	565954
Apollo 11 CSM	1969JUL16	US	5491	3.900	222	565954
Agena B 6006	1965FEB17	US	2240	1.500	199	564023
Ranger 8	1965FEB17	US	368	1.500	199	564023
Surveyor 5	1967SEP8	US	369	2.200	174	562308
Agena B 6005	1962OCT18	US	2240	1.500	217	558269
Explorer 35	1967JUL19	US	67	0.700	650	557687
Surveyor 4	1967JUL14	US	382	2.200	167	554098
Apollo 13 CSM	1970APR11	US	5610	3.900	219	549183
Apollo 8	1968DEC21	US	5524	3.900	214	544356
Ranger 5	1962OCT18	US	203	1.000	207	543753
Saturn S-IVB-503N	1968DEC21	US	86188	6.600	141	538222
Apollo 15 Subsatellite	1971JUL26	US	37	0.400	203	532310
Apollo 15 LM	1971JUL26	US	4782	4.300	203	532310
Apollo 15 CSM	1971JUL26	US	5342	3.900	203	532310
Ranger 7	1964JUL28	US	365	1.500	185	526173
Ranger 6	1964JAN30	US	381	1.500	200	520000
Saturn S-IVB-508	1970APR11	US	86447	6.600	47	519381
Nozomi	1998JUL3	J	540	1.600	1421	516130
AMS Luna-3	1959OCT4	SU	278	1.200	-630	516120

The first query we implemented was a simple list of the data, sorted by perigee or apogee. This screen was created by Declan. The data was loaded from the file using the loadStrings() method in processing. This array was then passed into a method written by Declan called generateObjects(), which created an ArrayList of “spaceObjects” containing the data for each line.

The ArrayList of objects is passed into a method made by Declan called sortSpaceObjects(). This method implements the method Collections.sort(), along with custom

Comparator subclasses to sort the objects by various properties, which can be found int java.util.Collections and java.util.Comparator respectively.

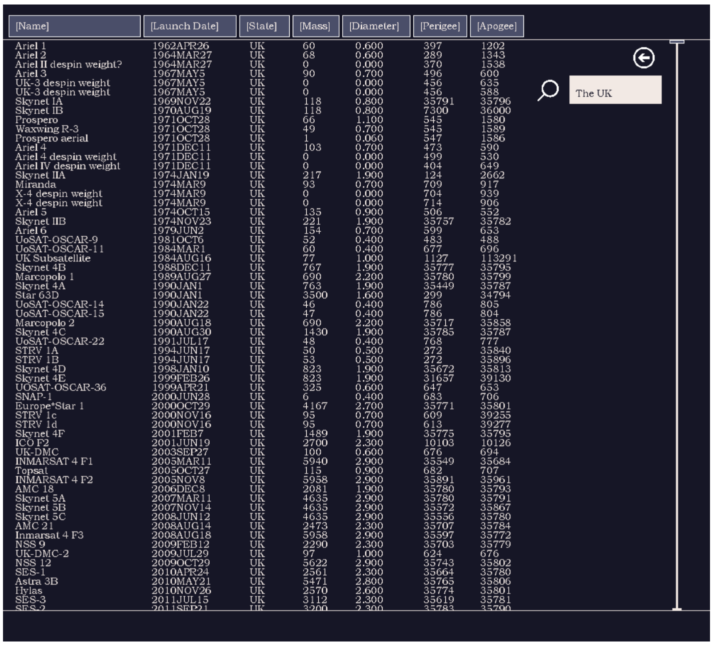


The UI itself is drawn by a method called drawSortedData(), written by Declan, which displays headers at the top of the screen, and draws the properties of each spaceObject. This screen makes use of the Slider class created by Mia. The slider consists of the track, two lines to indicate its ending and beginning, and the thumb used to indicate our position on the track.

The thumb moves when the mouse is pressed and dragged along the track or if the mouse is pressed somewhere within the slider bounds. It is used on multiple screens to aid displaying large data sets by allowing the user to scroll through a list.

Initially, the drawSortedData() method called draw() for the entire list of objects, which was a simple solution, but led to performance issues when working with larger files. In order to fix this, Declan made it so that only the lines with a Y-position within the bounds of the screen had the draw() method called for them.

Search by Country Screen



The UK

[Name]	[Launch Date]	[State]	[Mass]	[Diameter]	[Perigee]	[Apogee]
Ariel 1	1962APR26	UK	60	0.600	397	1202
Ariel 2	1964MAR27	UK	68	0.600	289	1343
Ariel II despin weight?	1964MAR27	UK	0	0.000	370	1538
Ariel 3	1967MAY5	UK	90	0.700	496	600
UK-3 despin weight	1967MAY5	UK	0	0.000	456	635
UK-3 despin weight	1967MAY5	UK	0	0.000	456	588
Skynet IA	1969NOV22	UK	118	0.800	35791	35796
Skynet IB	1970AUG19	UK	118	0.800	7300	36000
Prospero	1971OCT28	UK	66	1.100	545	1580
Waxwing R-3	1971OCT28	UK	49	0.700	545	1589
Prospero aerial	1971OCT28	UK	1	0.060	547	1586
Ariel 4	1971DEC11	UK	103	0.700	473	590
Ariel 4 despin weight	1971DEC11	UK	0	0.000	499	530
Ariel IV despin weight	1971DEC11	UK	0	0.000	404	649
Skynet IIA	1974JAN19	UK	217	1.900	124	2662
Miranda	1974MAR9	UK	93	0.700	709	917
X-4 despin weight	1974MAR9	UK	0	0.000	704	939
X-4 despin weight	1974MAR9	UK	0	0.000	714	906
Ariel 5	1974OCT15	UK	135	0.900	506	552
Skynet IIB	1974NOV23	UK	221	1.900	35757	35782
Ariel 6	1979JUN2	UK	154	0.700	599	653
UoSAT-OSCAR-9	1981OCT6	UK	52	0.400	483	488
UoSAT-OSCAR-11	1984MAR1	UK	60	0.400	677	696
UK Subsatellite	1984AUG16	UK	77	1.000	1127	113291
Skynet 4B	1988DEC11	UK	767	1.900	35777	35795
Marcopolo 1	1989AUG27	UK	690	2.200	35780	35799
Skynet 4A	1990JAN1	UK	763	1.900	35449	35787
Star 63D	1990JAN1	UK	3500	1.600	299	34794
UoSAT-OSCAR-14	1990JAN22	UK	46	0.400	786	805
UoSAT-OSCAR-15	1990JAN22	UK	47	0.400	786	804
Marcopolo 2	1990AUG18	UK	690	2.200	35717	35858
Skynet 4C	1990AUG30	UK	1430	1.900	35785	35787
UoSAT-OSCAR-22	1991JUL17	UK	48	0.400	768	777
STRV 1A	1994JUN17	UK	50	0.500	272	35840
STRV 1B	1994JUN17	UK	53	0.500	272	35896
Skynet 4D	1998JAN10	UK	823	1.900	35672	35813
Skynet 4E	1999FEB26	UK	823	1.900	31657	39130
UOSAT-OSCAR-36	1999APR21	UK	325	0.600	647	653
SNAP-1	2000JUN28	UK	6	0.400	683	706
Europe*Star 1	2000OCT29	UK	4167	2.700	35771	35801
STRV 1c	2000NOV16	UK	95	0.700	609	39255
STRV 1d	2000NOV16	UK	95	0.700	613	39277
Skynet 4F	2001FEB7	UK	1489	1.900	35775	35795
ICO F2	2001JUN19	UK	2700	2.300	10103	10126
UK-DMC	2003SEP27	UK	100	0.600	676	694
INMARSAT 4 F1	2005MAR11	UK	5940	2.900	35549	35684
Topsat	2005OCT27	UK	115	0.900	682	707
INMARSAT 4 F2	2005NOV8	UK	5958	2.900	35891	35961
AMC 18	2006DEC8	UK	2081	1.900	35780	35793
Skynet 5A	2007MAR11	UK	4635	2.900	35780	35791
Skynet 5B	2007NOV14	UK	4635	2.900	35572	35867
Skynet 5C	2008JUN12	UK	4635	2.900	35556	35780
AMC 21	2008AUG14	UK	2473	2.300	35707	35784
Inmarsat 4 F3	2008AUG18	UK	5958	2.900	35597	35772
NSS 9	2009FEB12	UK	2290	2.300	35703	35779
UK-DMC-2	2009JUL29	UK	97	1.000	624	676
NSS 12	2009OCT29	UK	5622	2.900	35743	35802
SES-1	2010APR24	UK	2561	2.300	35664	35780
Astra 3B	2010MAY21	UK	5471	2.800	35765	35806
Hylas	2010NOV26	UK	2570	2.600	35774	35801
SES-3	2011JUL15	UK	3112	2.300	35619	35781
SES-2	2011SEP21	UK	3200	2.300	35783	35790

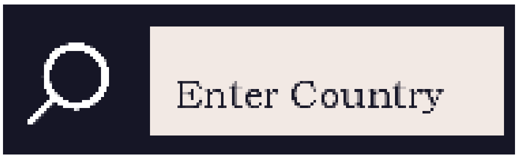
In this screen, the user is able to search for, and scroll through information pertaining to, objects of a particular country. To get user input, Darius, Kostiantyn and Declan created and implemented a search-box widget, which allowed the user to type in their query. Declan created a method called getObjectsByCountry(), which returns a subset of the full object list, only including the objects in the given country. This method supports searching by the country abbreviation (eg: UK, US, RU, CN) and the country's actual name (eg: Russia, France, Germany). Common variations on country names are also supported (eg: The United States, The USA, The US).

To increase user-interaction, Declan implemented sorting in this screen too. The user can click on the various headers to sort the selection of objects by various parameters. This required an extension in the functionality of the sortSpaceObjects() method, such as sorting alphabetically and sorting by date. Clicking on the headers multiple times toggles between sorting high-to-low and low-to-high.

This screen makes use of the same drawSortedData() method as in the Sort by Apogee/Perigee screens, allowing us to save on code duplication.

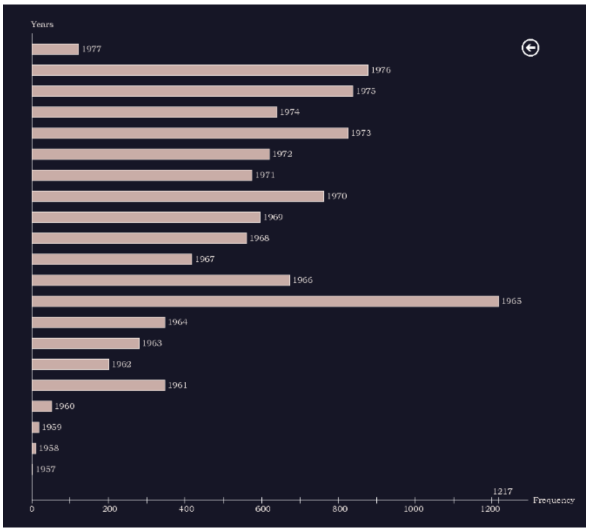
Other features were also introduced to make the UI seem more professional. Kostiantyn made it so that if no objects were being drawn (such as when the screen is first loaded, or if a search query returns empty), the slider wouldn't appear on the screen. This was done by checking the length of the searchedObjects ArrayList and hiding the slider if it was equal to zero. Darius also made it so that if a search query was made while the slider was already scrolled down, it would jump back to the top of the list; a feature which makes the design much more intuitive. Darius made the slider reset its position in the "search by country" screen. The problem was that after a user entered a valid country, the specific data would appear. The user could then scroll down and see more information, but if the user left the scroll at a position different from its initial one and entered a different country, the slider would remain in the same position, even though new information appeared. In order to solve this, he reset the slider's Y-position to the initial value it had.

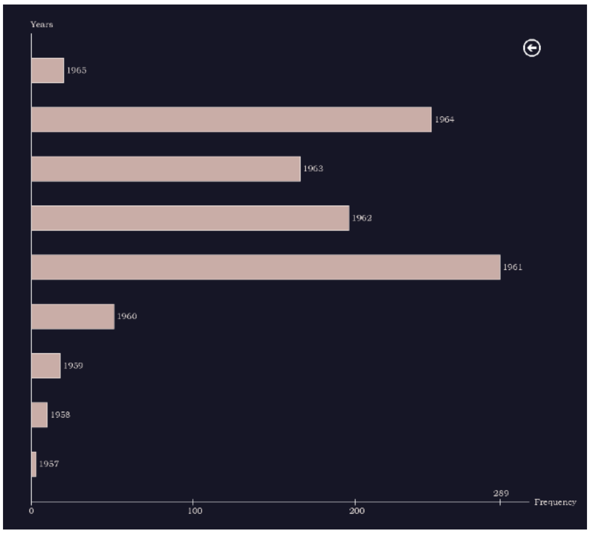
Finally, Mia used the PhotoWidget subclass to create a magnifying glass search widget which can be used to submit user input instead of pressing the enter key.



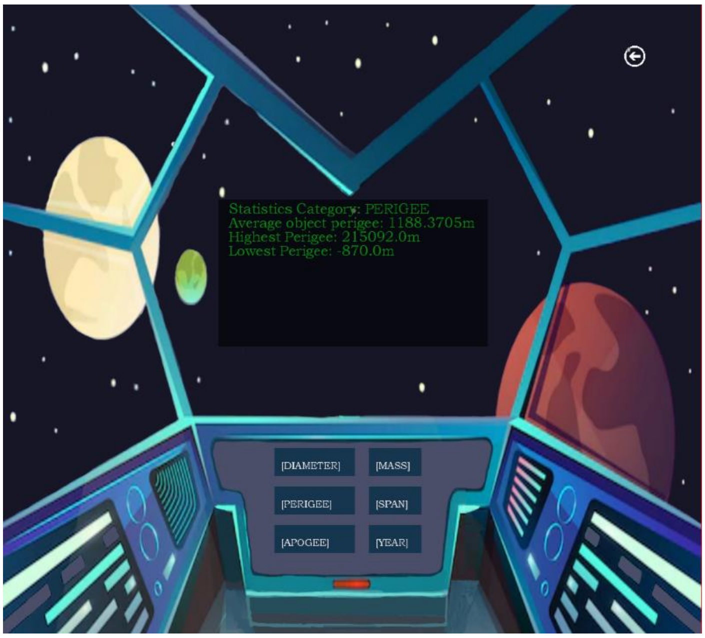
Year Bar Chart Screen

The YearBarChart class is used to create bar charts, which show the number of launched space objects for each year within an interval (e.g., 1967 – 1987). To evenly space out the bars, Mia used a scale, which made it possible to draw any number of bars spaced out evenly on the y axis. As for the frequency points, if the highest number of launches is a four-digit number, the code displays the frequency points as multiples of 200 (e.g., 200, 400, 600, 800 etc.), otherwise it shows the frequency points as multiples of 100 (e.g., 100, 200, 300, 400 etc). Finally, the animation is achieved by incrementing a temporary width variable of the bars every time the draw method is called. The temporary variable stops incrementing when it reaches its highest width value for a certain bar.





Statistics Screen



Statistics Category: PERIGEE
Average object perigee: 1188.3705m
Highest Perigee: 215092.0m
Lowest Perigee: -870.0m

The statistics screen is a screen created and implemented by Declan which gives some brief summary statistics about the collection of space objects. To make this screen more visually interesting, the UI was made to look like the inside of a spaceship, with shooting stars visible outside the windows, and a statistics display made to look like a semi-transparent, futuristic computer terminal projected onto the window with green text.

To achieve this look, a PNG image of the inside of a spaceship was found online, and paint.net was used to erase the outside of the ship, leaving only the stars and planets. A ShootingStar class was also created by Declan, which moves a small white circle back and forth randomly across the screen. Since the PNG was made transparent in some areas, the stars pass behind the image when moving across the planets and parts of the ship, giving the appearance that they are outside of the ship. This was an instance where we had to make us of the feature mentioned in the “Screen Organisation” section for drawing widgets outside of the Screen’s draw() method, since we needed complete control over what layers objects were being drawn on. Otherwise, we would’ve had trouble making sure that the shooting stars moved behind the correct objects. The statistics display was made to look semi-transparent by editing the alpha value of the fill colour.

The statistics were calculated using a class written by Mia called SummaryStatistics, as well as a couple of extra statistics calculated by Declan. The buttons below the window allow the user to choose what statistics are shown.

Polishing + Efficiency + Finishing Touches

In the last week, Mia organised the code to make it easier to follow by adding a “constants” tab, deleting unused code which was not specified by the weekly goals, and modifying certain variable names.

Declan added UI features, such as button highlighting if the mouse is hovering over a widget.