**NEA : Space Sandbox Simulation**

**Name: Scott Render**

**Candidate number:**  **8023**

[1 Introduction 1](#_Toc246691680)

[2 Analysis 2](#_Toc1825647971)

[2.1 Computational Approach 3](#_Toc994672829)

[2.1.1 Thinking Abstractly 3](#_Toc1436902175)

[2.1.2 Thinking Ahead 3](#_Toc1937262813)

[2.1.3 Object Oriented Programming - OOP 3](#_Toc1277855039)

[2.1.4 Thinking Procedurally 4](#_Toc338327789)

[2.1.5 Thinking Logically 4](#_Toc682588463)

[2.1.6 Thinking Concurrently 4](#_Toc1237745255)

[2.1.7 Use of Database 4](#_Toc1694574898)

[2.1.8 Libraries 4](#_Toc1859247754)

[2.2 Stakeholders 4](#_Toc475318054)

[2.2.1 Overview 4](#_Toc181001039)

[2.2.2 Survey 5](#_Toc214695230)

[2.2.3 Other surveyed features 6](#_Toc846698940)

[2.3 Research 6](#_Toc1529969961)

[2.3.1 Overview 6](#_Toc677364121)

[2.3.2 Program #1 – Kerbal Space Program 6](#_Toc1446068981)

[2.3.3 Program #2 – Universe Sandbox 9](#_Toc158173447)

[2.3.4 Program #3 – 11](#_Toc1171919856)

[2.3.5 Summary of research 11](#_Toc1209315431)

[2.4 Essential Features 11](#_Toc599851669)

[2.4.1 Core features 12](#_Toc1721030026)

[2.4.2 Stakeholder desired features 13](#_Toc1219089592)

[2.4.3 Researched desired features 14](#_Toc1880209684)

[2.5 Potential Limitations and Solutions 15](#_Toc830286738)

[2.6 Requirements 16](#_Toc764332367)

[2.6.1 Making the program 17](#_Toc646374226)

[2.6.2 Using the program 17](#_Toc1701920291)

[2.7 Success Criteria 17](#_Toc1393637043)

[3 Design 19](#_Toc778459955)

[3.1 Design Plan 20](#_Toc785642806)

[3.2 Program Structure 21](#_Toc1027238550)

[3.3 Post Development Testing 22](#_Toc1069393461)

[4 Cycle #1 25](#_Toc1608980344)

[4.1 Design 25](#_Toc90441995)

# Introduction

I will produce a program intended to allow A Level Physics students to view and experiment with planetary physics / orbital mechanics. Users will be able to navigate a two-dimensional solar system with a choice between an accurate or simplified scale. Users will also have the option to alter the state of; add new; remove or learn more about bodies in the system whilst they use the program. The motion of the bodies will be accurately modelled (and the user will be able to see the future path of selected bodies and how this changes as they change things about the system). The program will feature dynamic audio, reflecting the situation on-screen. Multiple users will (hopefully) be able to access the program, with each of their saved systems being stored (locally? In DB? System / Challenge sharing / rating system). Users will be able to add and control a spacecraft into the system. (with varying modifiable characteristics e.g. fuel, power, food/water to allow for creativity and challenge) A keyboard and mouse or a gamepad will be used for navigating systems and menus or controlling spacecraft.

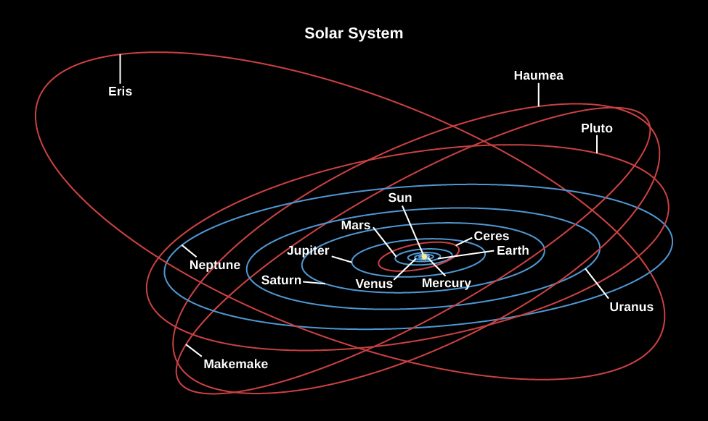
Upon starting the program, the user (will be able to login to their account?) and will be able to configure settings such as volume. From there, the user will have the choice between opening a new system (...? info section?) (+career / challenge mode?) or loading a previously saved one.

# Analysis

## Computational Approach

### Thinking Abstractly

There are many characteristics of the solar system which can be abstracted away in order to simplify the design and processing requirements of the program. This also simplifies the user experience, allowing the user to more easily use the program. One such abstraction is to ignore the inclination of the bodies as the program is a 2D simulation so cannot simply represent inclination. Demonstrated here, [[solar system inclinations demo img - Search]:](https://www.bing.com/images/search?FORM=IRPRST&adlt=strict&ccid=iSTllC30&cdnurl=https%3A%2F%2Fth.bing.com%2Fth%2Fid%2FR.8924e5942df4da0f5aa0961506f71f34%3Frik%3DxACkDReo2mtUvQ%26pid%3DImgRaw%26r%3D0&ch=828&ck=099AAFA61ED9A85A98E712AD1DC1B2AB&cw=1721&exph=582&expw=975&id=BD6B06A9842F346E8082BD546BDAA8170DA400C4&itb=0&mediaurl=https%3A%2F%2Fpressbooks.bccampus.ca%2Fastronomy1105%2Fwp-content%2Fuploads%2Fsites%2F235%2F2017%2F08%2FOSC_Astro_07_01_Orbits-1.jpg&mode=overlay&q=solar+system+inclinations+demo+img&safesearch=strict&selectedIndex=27&simid=608049310136237195&thid=OIP.iSTllC302g9aoJYVBvcfNAHaEa&view=detailV2)

This is one of the reasons a 2D approach to the simulation was chosen as it greatly reduces the number of necessary calculations for an accurate simulation. Another, more necessary for computation abstraction is to ignore the composition & unnecessary characteristics of the bodies in the system, not calculating forces acting on each particle or molecule, instead each body will be abstracted to acting as if it were a single particle of evenly distributed mass. This will greatly reduce processing time to a feasible amount from an impossible amount.

### Thinking Ahead

The bulk of the program is to accurately simulate the motions of bodies in the system; to achieve this, each body must have certain characteristics (More about this in the OOP section) The simulation must have some global time, to be able to run the simulation at speeds different to real time. Finally, the change to any non-constant characteristics of each body in the system must be accurately updated according to the situation of the rest of the bodies in the system with each step in time. Computational Methods are suitable for this task as while each individual calculation is trivial, many must take place in a very short time to be useful to the user.

### Object Oriented Programming - OOP

OOP is suited to this project as the essential characteristics (or attributes) of bodies are identical – position, mass, velocity, radius – and some other aesthetic characteristics e.g. image to be displayed in the body’s position. OOP is similarly well suited to a computational approach as functions and procedures can be designed to receive one or a list of body objects and perform calculations or update attributes of one or all of them sequentially. One possible improvement could be to perform such calculations in parallel, e.g. using the GPU as this would greatly reduce the total computation time of lengthy algorithms being applied to many inputs, however I will not be making use of this as the program should run in the browser and be accessible to as many users as possible.

### Thinking Procedurally

The program will be split into subprograms, with 2 notable procedures being “update()” and “draw()” being ran once per frame each in the stated order by the q5.js library. Other subprograms will be executed according to the logic of the program (Logic to be expanded upon in the Thinking Logically section) with inputs, processes and outputs (of each?) to be defined in the design section of this document.

### Thinking Logically

Within the update procedure, different subprograms will be executed based on a selection of the global state of the program. E.g. A menu state will lead to the execution of subprograms which will display the buttons which the user can select to exit, access settings etc. This main global program state and certain other variables must be alterable by the user’s chosen input method, handled by a procedure which handles user input.

### Thinking Concurrently

Whilst JavaScript does not allow for parallel processing (?), the illusion of states changing concurrently can be created as the user can only see changes of state once per frame. So, any calculations or changes performed within each update loop appear to be concurrent. This effect will be useful for example in allowing the motion of bodies to appear to move at the same time as opposed to one at a time.

### Use of Database

A database will be used to store user’s login data and preferences as well as saved simulations / scenarios. Othe data about the simulations will also be stored such as ratings & descriptions if they are made public – viewable to other users of the program. Users should be able to utilize a username and password to login, with a hash of the password being stored in the database as it is more secure than plaintext.

### Libraries

The program will run on the browser using JavaScript with a library called “q5” to display the program to the user. This allows the user to use the program without downloading any additional software or libraries to their device, given that they have access to a web browser. This is beneficial as it makes the program accessible to students who may only have access to a browser on a school computer.

## Stakeholders

### Overview

This program is aimed at A level physics students in the UK and feature suitable for all. This allows for the use of terminology at this level. However, in order to allow for a wider range of users, the program will features ways for users to learn about any terms they don’t understand. The students are likely to appreciate terminology to be representative of what they are supposed to learn as a part of their course. All users are likely to appreciate realistic graphics, however the intended stakeholders of physics students will be using the program for educational purposes so will desire a more simple representation if it leads to better understanding. The program will be accessible via a web browser, this suits many students, many of whom will have access to a college computer, itself having access to a web browser and the internet.

### Survey

#### Overview

I produced a form for potential stakeholders to fill out to survey what users may want to see in the program. In total, twelve people completed the form with two thirds of them studying A level physics and the other third studying some other A level or equivalent at college. 50% of the surveyed said that they had used a space simulation/sandbox previously. Of these, their opinions are as follows.

#### Desirable Features

|  |
| --- |
| The ability to drag things around with the mouse. |
| Ability to add new planets and stars. |
| Ability to change features of planets and seeing the impact it had in its surrounding. Adding objects or events found in space into a setting they are not normally seen in and seeing its impact such as a black hole in our solar system. |
| Ability to blow up planets. |
| An intuitive Ui system with sliders on most properties, such as gravity and initial velocity and mass of certain objects. If the simulations around space I think being able to see the paths that objects will take would be useful in order to be able to create certain trajectories by tweaking values. |
| Make numbers like mass really really big. like x10^100. |
| Place new planets into orbit of other things. |
| Add in object it evens like dying stars or supernovas into our solar system. Just mess around with planets and seeing them explode. Maybe futuristic stuff like adding a ray that you can shoot and like cut the planets. |
| Create your own planets. |
| See planet's dimensions Know accurate distances between celestial bodies Be able to adjust forces (such as gravity) to see the effects on space. |

#### Undesirable features

|  |
| --- |
| Lack of simplicity. |
| Was slow to run. |
| They didn’t really have any negatives other than having to pay for them. |
| Easily boring |

#### Features overview

Here, the undesirable features complement the desired features as it seems that the stakeholders have identified what they would like to see in the program in order to fix what they did not like. For example, the “Easily boring” being remedied by the creative responses such as “Ability to blow up planets”. Where adding features such as blowing up planets for recreational users, these features should still have some educational aspect if they are to be implemented at all. Furthermore, there will be no issue with cost with this program as it will be available for free via a web browser with connection to the internet. The program should also offer a range of things to do for users who desire an experience of different complexities; from adding in a second sun to our solar system and witnessing the ensuing chaos to predicting the effect of altering a planet’s mass and confirming the result with the program or reading definitions of physics terms, users should be able to find features of their desired complexity. With users stating that a slow running program is not desirable, it should be an objective for this program to run smoothly, making use of necessary algorithms and data structures to do so. To achieve this, it may not be possible to allow users to change attributes at the desired scale of “x10^100” however the range of possible values should be as large as possible.

### Other surveyed features

Users were asked to rank their preferred method of interfacing with the program. Keyboard and mouse were favoured the most with no users rating it the worst method with the other options being, in rank order: Mouse and Keyboard, Touchscreen, Gamepad. This means that the program should primarily be controlled with mouse and keyboard as nobody rated it the worst and other input methods should be implemented if there is ample development time.

63% of users stated that they would prefer to use a space sandbox/simulation for recreation while the other 27% would use it for education. Given the intended purpose of the program, the educational aspects should not be sacrificed to provide a more recreational experience. However, the program should be as entertaining as is reasonably possible to capture as many users as possible who will inevitably learn whilst they use the program, even in recreation.

75% of the survey respondents said they would like simplified graphics as shown in the adjacent image, whilst one noted in the ‘other’ response that they would prefer images to vary depending on distance/situation. This is a good idea and will be implemented if possible as it provides a more varied experience and shows to users that planetary bodies look vastly different at different scales. Though for the most part, these simplified graphics are simple to display and require very little memory to store making them fitting for a browser program.

58% of respondents said that they would prefer a dynamic star pattern as a background, while 25% said they would prefer black. Perhaps allowing this feature to be a configurable option will be the most beneficial for all users, allowing them to see which one they prefer in the context of the program. Again, 58% said that they would prefer dynamic audio, with the remaining respondents being largely split between constant music or silence. Here a default of dynamic audio would be ideal with the option for the user to decrease the volume or silence the sound.

## Research

### Overview

Broadly, the problem this program seeks to solve can be split into three parts. Firstly, to accurately simulate the relevant physics to a reasonable degree; secondly, to educate the user about the relevant physics (including limitations of the simulation) and finally to be captivating enough to retain the user so that they can learn while they enjoy the program.

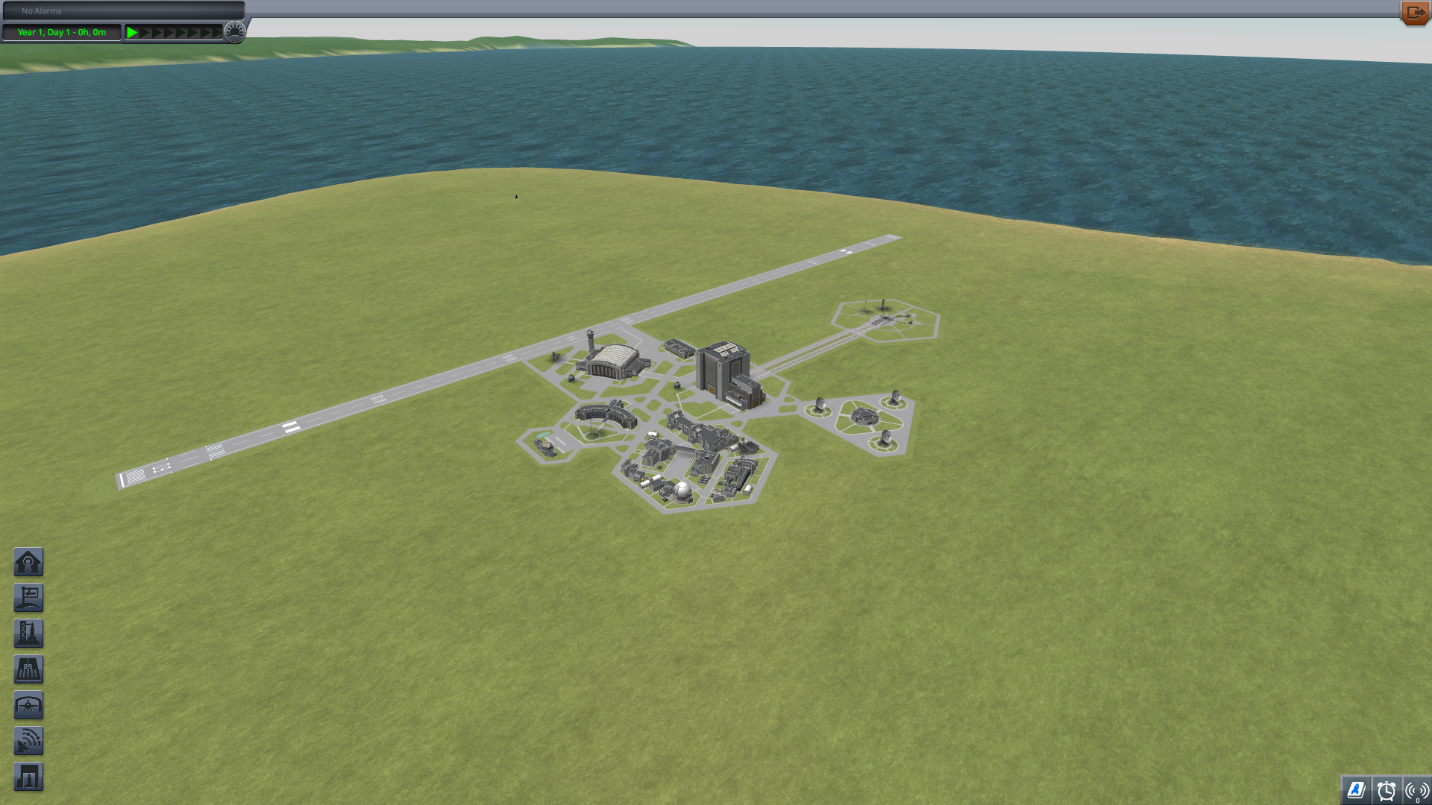
### Program #1 – Kerbal Space Program

Commonly referred to as “KSP”; KSP effectively portrays a good solution to the third part of the problem in this genre as it prioritizes having a creative sandbox where users can construct their own spacecraft part by part. In doing this, KSP allows users to learn the fundamentals of the related physics by trial and error and introducing more complicated parts as the user explores more of a small-scale solar system via the use of a ‘career’ mode. I hope to achieve a similar effect with my program however not using the same method as KSP. However, I will similarly have user creativity being the main reason to keep users on the program whilst having the choice at any point to learn about the relevant physics at their own pace. This has the benefit of users having the choice to learn as much or as little as they choose to.

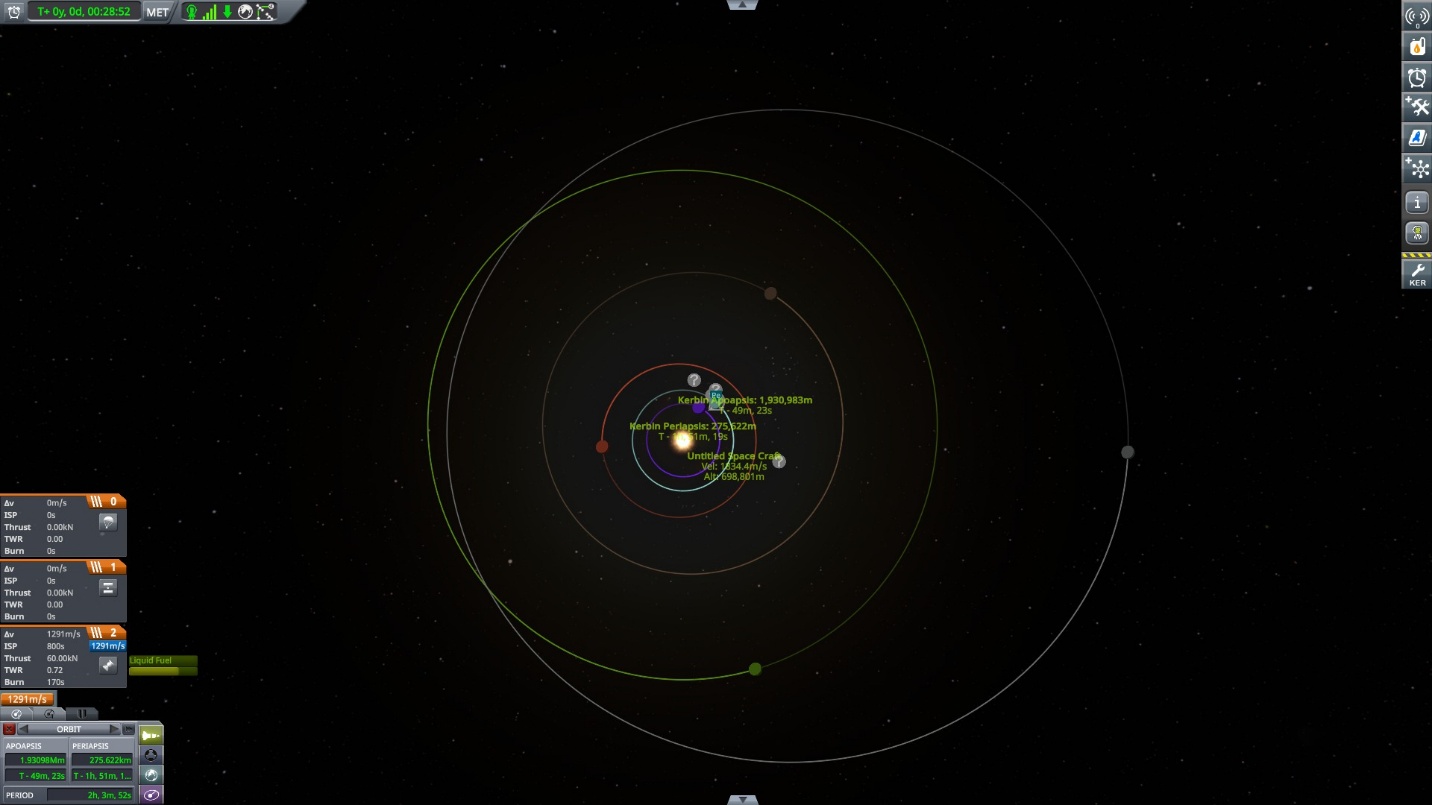
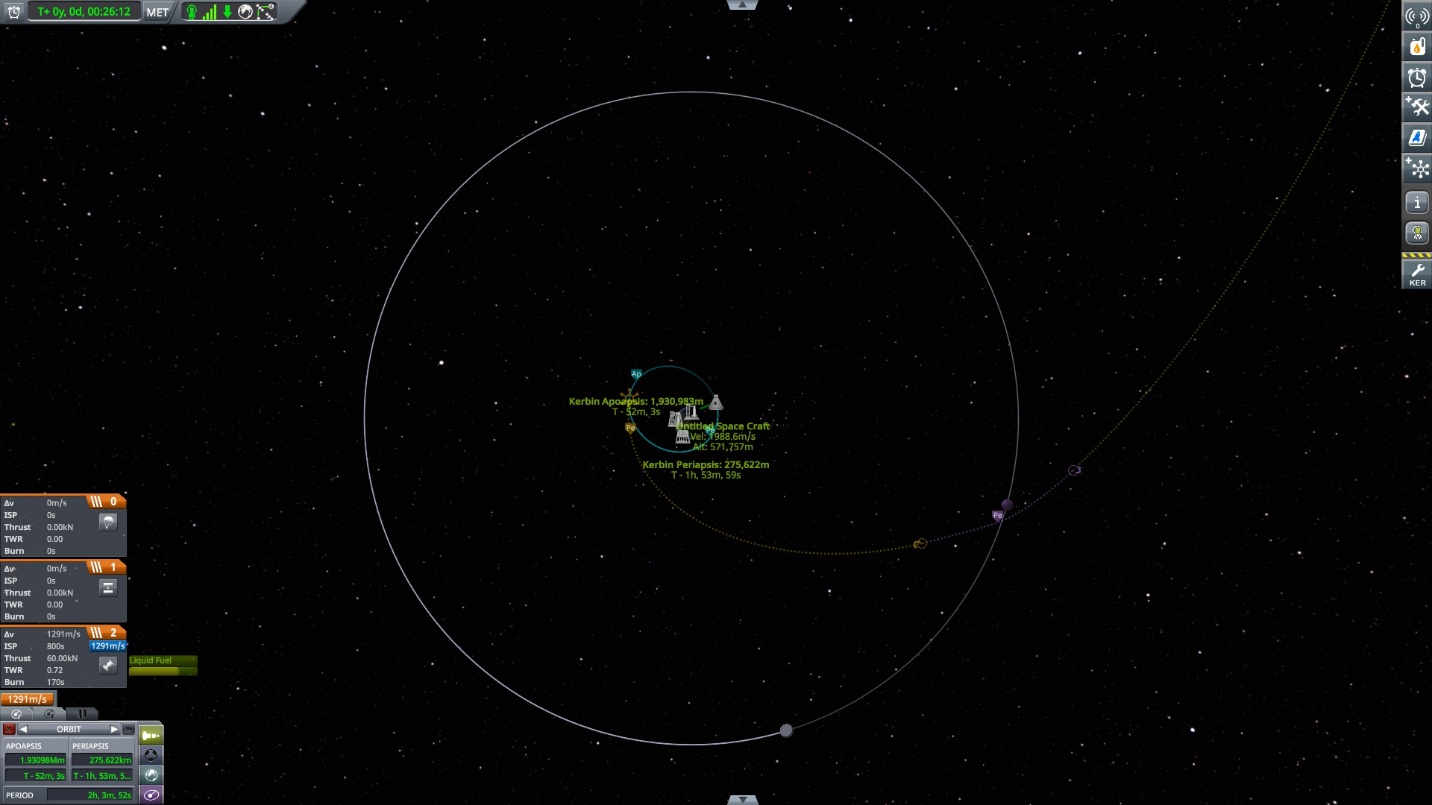
A similar menu design to KSP could be incorporated into my program. As it features essentially the same initial options available to the user.

* Start Game à new / load simulation.
* Settings à configure volume etc.
* Community à View other users’ public systems / challenges.
* Credits à list of credits to art sources and me.
* Quit à exit program

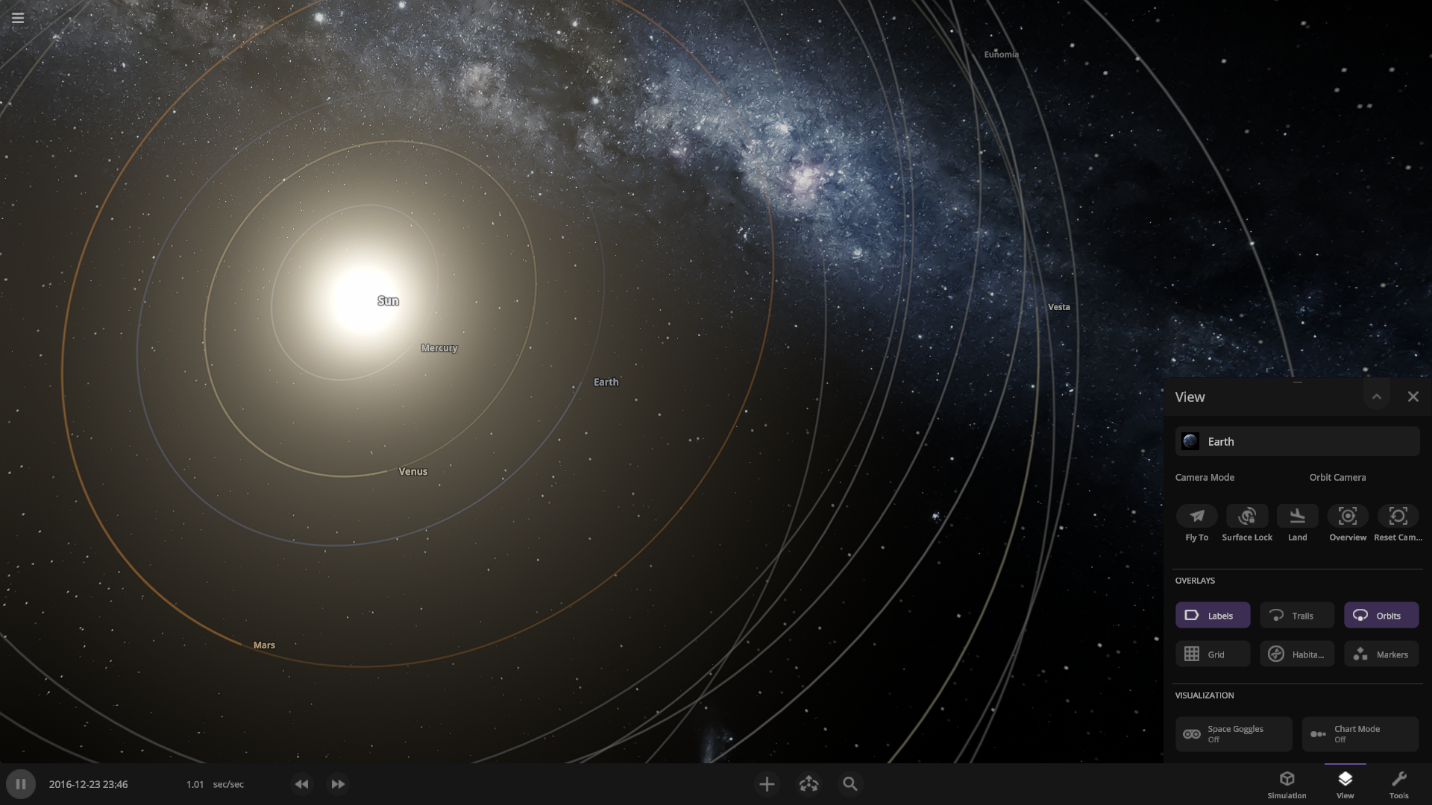
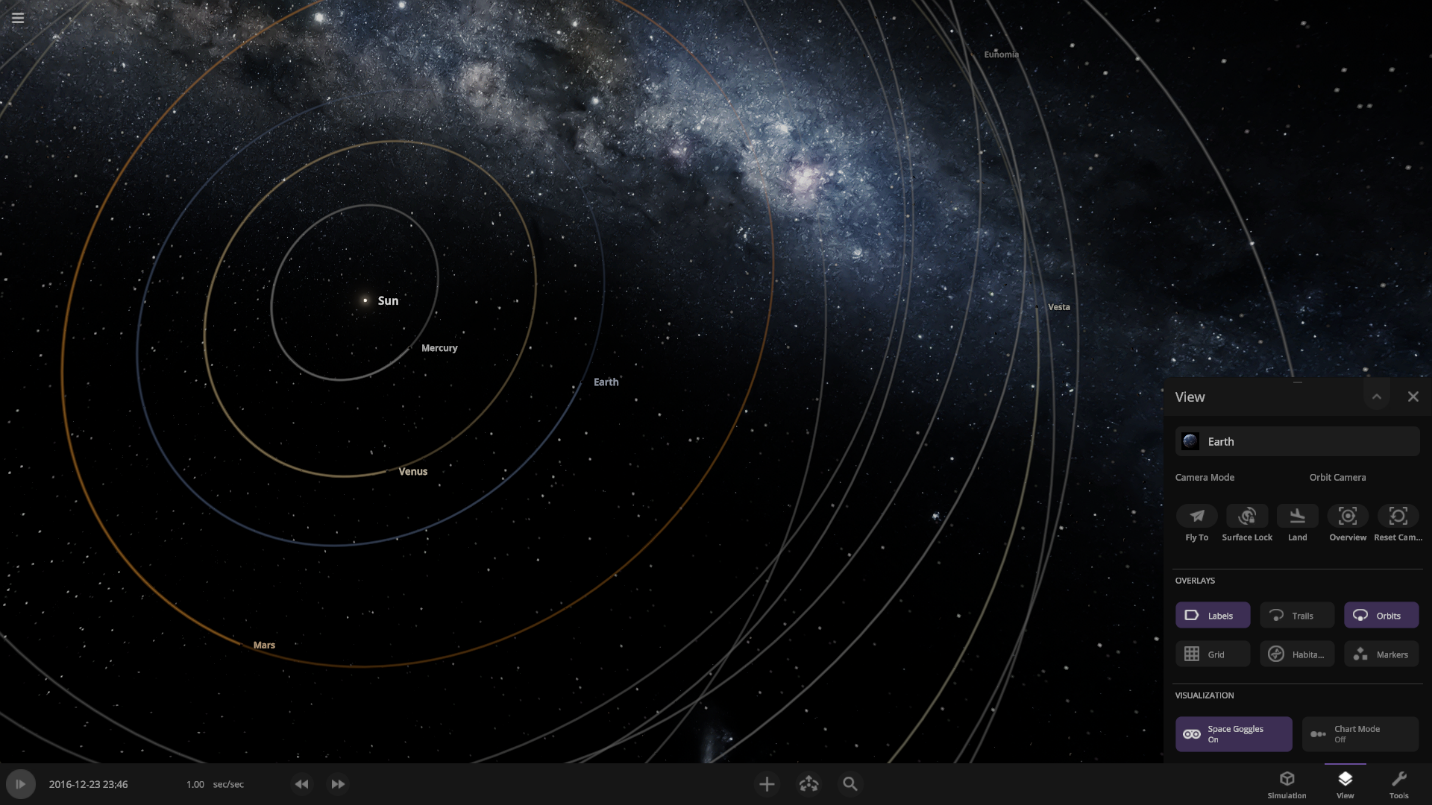
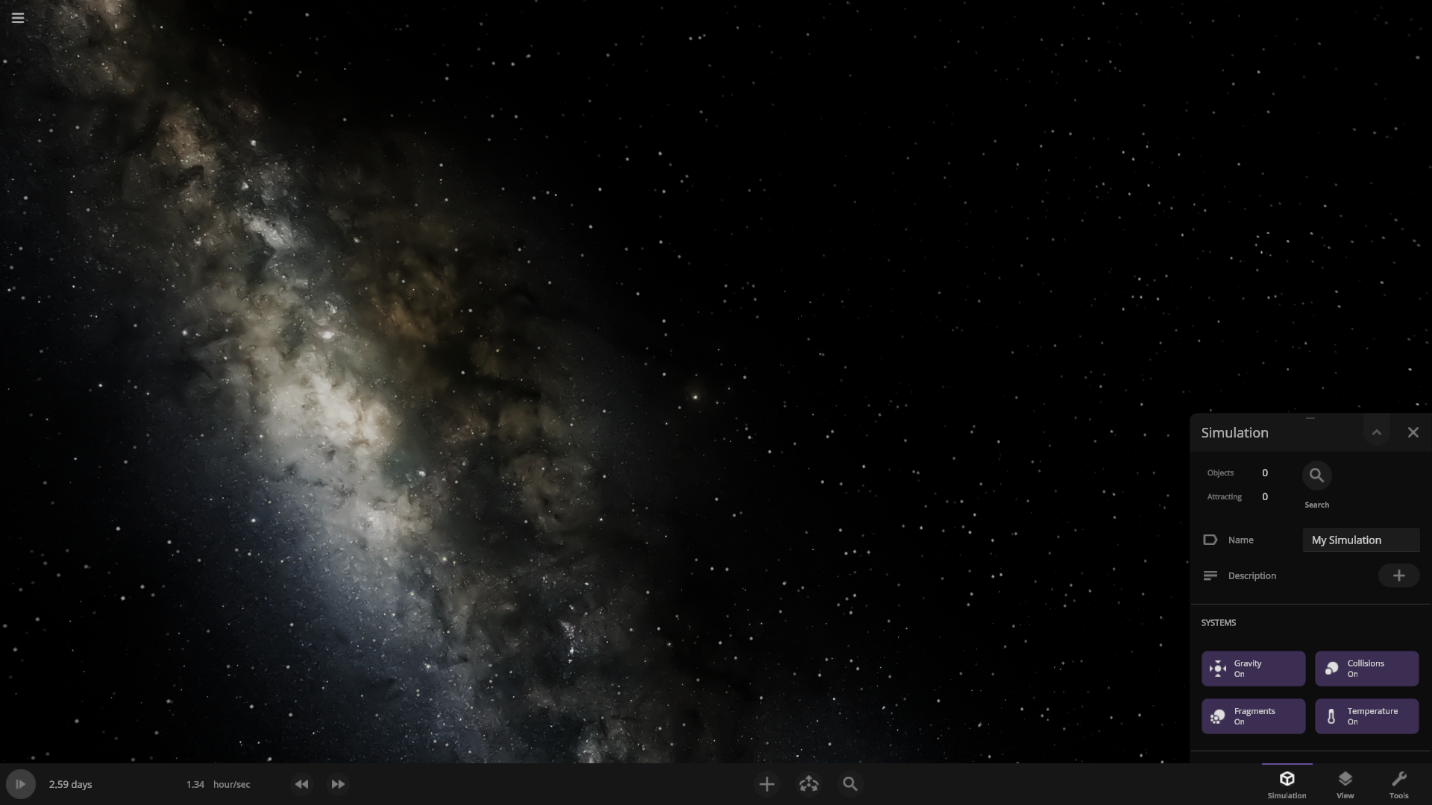
Some features from KSP’s career mode could be introduced as configurable attributes of a controllable rocket, e.g. allowing users to create a challenge such as getting from low earth orbit to mars orbit using a limited fuel load, with other users being able to attempt the challenge and give it a rating. However, there are some notable features of KSP that will not feature in my program with good justification. For example, KSP features a single 3D system which cannot be altered, the only thing that the user can alter is the design and control of a spacecraft being launched from it’s Earth equivalent. My program will focus more on the user’s ability to alter aspects of anything in the system and will be a 2D simulation. This differs from KSP as KSP does not accurately model the motion of the bodies in the system apart from the spacecraft, they are intentionally ‘on rails’ in elliptical orbits designed by the developers to provide a game like experience for the user. My program will accurately model this motion however, though to support this, a 2D simulation will instead be used.

Features of KSP that my program will not include 3D Graphics (...), upgradeable space center (...), simple “planets on rails” physics.

Features of KSP that my program will include:



### **Program #2 – Universe Sandbox**

Unlike KSP, Universe Sandbox provides an ‘N body’ simulation of bodies in a system using Newtonian mechanics. As Universe Sandbox appears to have an accurate enough model for the scope of my project therefore, I shall take a similar approach to providing an accurate simulation. Universe Sandbox has some more desirable features for a sandbox experience that would be useful to emulate in my program. Firstly, the non-static background which moves as the user pans their camera. This would be beneficial to emulate as having a dynamic background can allow the user to view similar things in the simulation and hence learn by experimentation whilst having a subconsciously different experience, hopefully leading to the user not becoming bored and closing the program.

### Program #3 –

### Summary of research

## Essential Features

From the previous analysis, a list of features follows:

### Core features

#### Compulsory

* Ability to navigate the program with keyboard and mouse.
  + The simplest way to implement user input and the most common method of interfacing with computers. Most users will have access to this input method.
  + Includes moving the camera within the simulation
* Ability to view information about planets, stars and other physics info within the simulation.
* To be able to view definitions and explanations of selected physics terminology
  + This and the previous feature are necessary as they both fit the aim of the program: to educate users about physics.
* Users able to create account to save simulations and continue when logged in
  + User preferences and saved simulations stored in database.
* Pause menu.
  + Save / Save as.
    - Present user with opportunity to save the simulation’s current state and give it a name to be loaded later.
  + Settings
    - Allow user to change program and simulation settings without exiting to the main menu to use its settings screen.
    - Volume.
    - Graphics settings.
  + Menu
    - Brings the user back to the program’s main menu.

#### Desirable

* Ability to navigate the program with a gamepad controller.
  + This would benefit the program as it would be more accessible. This is desirable though is not necessary as only one method of input is necessary for the program.
* Ability to view future path of selected planet / star / spaceship.
  + This would benefit the user experience of the program as they would be able to see the impact of their changes to the state of the simulation on the future path of selected planets / stars.
* A ‘Learn’ tab in the main / pause menu where the user can learn about the relevant physics to the program.
  + This fits the aim of the program, to educate users about physics.
* A settings menu allowing the user to alter the key bindings for the main simulation environment.
  + This allows for more user personalisation and/or accessibility of the program.

#### Additional

* Ability to navigate the program with a touchscreen.
  + This feature would allow for maximum accessibility to the program as users would be able to use the program on a mobile device for example. However, the difficulty of implementing this feature with the relevant libraries makes it a lesser priority than many other features.
* Variable level of detail of definitions and explanations in settings.
  + This would allow for a wider range of users to be able to benefit from the educational aspects of the program.
* Ability to view future path of all planets / stars / spaceships in the simulation simultaneously.
  + Expanding this feature to all bodies in the simulation would be preferrable. However, it is likely to be computationally intensive and would require compromising on other more compulsory features e.g. maintaining a smooth-running program.

### Stakeholder desired features

#### Compulsory

* Ability to add / remove new planets and stars to a simulation.
  + This was requested by the survey respondents and was suggested in the introductory idea of the program so shall be included.
* Ability to change attributes of planets and stars to a simulation.
  + Like the previous feature, this is a core part of the idea of the project and was also suggested by survey participants who were not briefed with even the project introduction.
* View dimensions of planets and stars within a simulation
* Variable simulation speed
  + This feature would allow users to witness the change to the system within their simulation over a very short or very long period, within the same amount of their time.

#### Desirable

* Ability to move objects in a simulation with the mouse / controller
  + This feature would be beneficial as it is an additional way to experiment with their simulations.
* Variable simulation graphic detail depending on zoom
  + This would allow for users to have a better sense of scale as when zoomed in
* Variety of default/preloaded simulations to choose to start from
  + Allowing users to have the choice between a blank simulation for the user to decide what will go there and preloaded / community made starting points of interesting events occurring e.g. the solar system in its current state or with every planet having double the mass.

#### Additional

* Ability to destroy / explode planets and stars
  + Some survey participants suggested the ability to destroy bodies in the simulation using things such as giant lasers. While this does not fit the theme of the project, it could be implemented to allow for even more recreational creativity

### Researched desired features

#### Compulsory

* Main menu with the following features
  + Start new simulation
  + Settings
    - Volume
    - Graphical details
  + Community
    - View users’ public simulations / challenges
  + Credits
  + Exit
* Toolbar at bottom of window interact with simulation
  + Option to vary simulation speed here
  + Button / dropdown to add new planets / stars here

#### Desirable

* Togglable “Space goggles” to change visible scale of planets and stars to be more easily seen depending on distance
  + This feature would be desirable as it allows users to switch between a realistic scale and a more visible, user-friendly scale.

#### Additional

* Altitude display – above focussed planet / star – when piloting spaceship.
  + Beneficial for the user to navigate the spaceship to specific orbits.

## Potential Limitations and Solutions

| Limitation | Explanation | Justification | Potential Solutions -> design later |
| --- | --- | --- | --- |
| Computation demand | Running the simulation at higher speeds will require a lot of computation to maintain accuracy.  Any compromise that reduces the amount of computation will lead to a decrease in the accuracy of the simulation of the movement of the bodies.  With the program running on the browser, processing power is greatly limited compared to if it were to run as an executable, locally on the user’s device. | It is reasonable for this limitation to persist due to the nature of the program running on the browser. | Implementing parallel processing could greatly increase the speeds of simulation before this problem becomes significant.  Limiting simulation speed could prevent this impacting accuracy. |
| Drawing predicted path of bodies | Drawing the path which a selected body will take requires running the simulation to some specified depth, ahead of what is being shown to the user.  This would require vastly more processing to occur each frame, something which is unlikely to be manageable at higher simulation speeds. | It is reasonable for this to be a limitation of the program as the program should prioritise simulation of the bodies’ positions over aesthetically pleasing visuals of their future paths. | A different model of the system could be used to display future elliptical/hyperbolic paths – Keplerian as opposed to Newtonian for the rest of the simulation. |
| Public / shared simulations | If simulations are shared to other users, the users will be unable to alter the simulation simultaneously and see each other’s changes. | This is because the simulations will be saved into a database and loaded into the local program when being used. | - |

Other limitations of the program could result from external circumstances such as illness or injury. These factors could limit the time where design and implementation can effectively be carried out, potentially leading to fewer requirements being met than would have otherwise been.

## Requirements

### Making the program

Some hardware and software are necessary to produce and use a space sandbox/simulation program. I should aim for the usage requirements of the program to not be to intensive. This is because the program is aimed at students who may not have the resources to use a powerful computer with additional peripherals or costly software packages. These software packages students may also not have the permissions to run on their school computers.

In making the program it is necessary for me to use a computer to produce and test code. To make this process easier, I will also be using an integrated development environment; namely, VS Code. I will also require DB management software as this is where user preferences & simulations will be stored & managed. The computer used to produce the program should meet the usage requirements as follows.

### Using the program

The user will require a computer with a monitor to run and display the web browser on which the program will run. Additionally, some hardware to interface with the program, namely a keyboard and mouse and/or a gamepad. The user’s computer must have a compatible operating system as well as sufficient processing power and memory to run a web browser as a web browser tab is the limit of the program’s processing intensity.

No libraries will have to be downloaded alongside the program should the user decide to download the program and use it locally, however to do this they must have a connection to the internet. This is because within the code, the libraries can be linked via a web address or to a local folder/file. This means that the user can download the program and libraries and use it offline if they want to.

## Success Criteria

| # | Criteria | Justification |
| --- | --- | --- |
| 01 | Is main menu displayed on program start, allowing the user to select between the following choices?   * new simulation * load simulation * view community simulations * learn * change settings * exit program | User should be able to choose which aspect of the program they would like to use at their own pace.  This is a suitable criterion as it can be easily determined whether the program has a main menu or not, additionally any missing choices from the main menu can be identified / measured. |
| 02 | Is the user able to move the camera’s position – x,y – within a simulation using the following input methods?   * keyboard and mouse * gamepad * touchscreen | The user should be able to navigate the simulation with at least one input method with more being preferable.  This is a suitable criterion as it can be objectively determined if the user can move the camera’s position and measured with how many input methods. |
| 03 | Is the user able to select from a range (3+) of default starting situations from a simulation when selecting ‘new simulation’? | The user should have the option to determine the start position of their new simulations. The number of choices they have is discrete and measurable so it can easily be determined if this criterion has been met. |
| 04 | Is the user able to view the current simulation time as well as stop, change and alter the rate of the simulation time? | The user should be able to view their simulations at various speeds. This criterion is suitable as the number of the stated criteria met can be counted. |
| 05 | Is the user able to pause the simulation, displaying a pause menu where the user can choose from the following options?   * Continue * Save / Save as simulation * Settings * Main menu | The user should be able to view the main menu alongside the other stated choices whilst running a simulation by using a pause menu.  The number of the stated choices available to the user in the pause menu can be measured and compared to this success criterion. |
| 06 | Do the objects within the simulation act reasonably accurately for a simulation of motion? | This criterion can be measured by comparing motion within the simulation to what is expected of real bodies of the same attributes. E.g. this could be done by calculating the escape velocity of an object from earth and placing an object of lesser and greater than this velocity near earth and viewing if it escapes earth in each situation. |
| 07 | Is the user able to manipulate the state of the simulation in the following ways?   * Add new objects * Remove existing objects * Alter the following attributes of existing bodies: mass, position, velocity, diameter | Users should be able to manipulate the simulation to enjoy it as a ‘sandbox’ recreationally or as a creative way to learn about physics.  Of these ways to manipulate the simulation, successfully implemented ones can be counted and compared to this success criterion. |
| 08 | Is the user able to log into the program on launch with a username and password with their user preferences being saved E.g. volume and graphical settings? | Users should be able to log into the program with their username, hash of their password and user preferences being stored in a database to be loaded.  Whether users log in to use the program and have saved preferences or not is measurable, making this a suitable criterion. |
| 09 | Are users able to save the state of their simulations to the program’s database and load them in various sessions of program use? | Users should be able to create and store multiple different simulations to the database and continue using them in future uses of the program. Whether users can save and load simulations or not is measurable, making this a suitable criterion. |
| 10 | Is the user able to view definitions and explanations of physics terms at differing levels of detail via the following methods?   * Clicking / selecting highlighted terms to view definitions * Using a ‘learn’ section to find explanations / definitions | Achieving these criteria allows users to learn with multiple methods, whilst using the simulation or in a more dedicated learning section devoted solely to explanations and definitions. |
| 11 |  |  |

# Design

## Design Plan

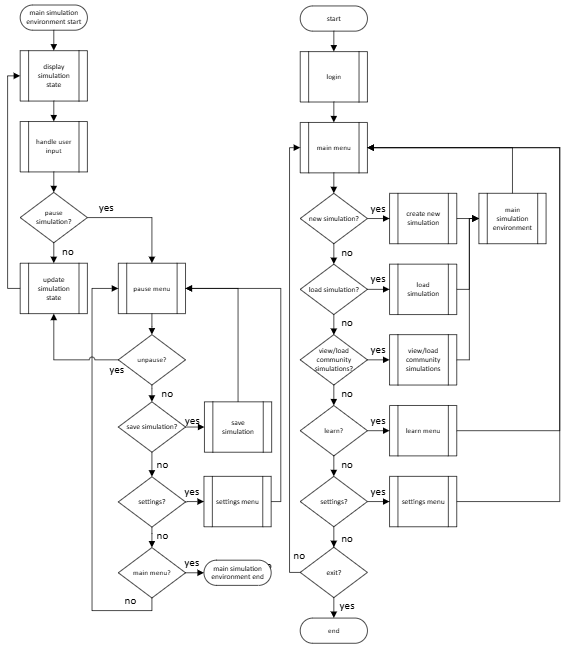
The program requirements broken into problems with relevant explanation of solution design with justifications are as follows:

| # | Problem | Explanation & Justification |
| --- | --- | --- |
| 01 | Produce main simulation environment | Implementing the following features:   1. Camera class with x,y position 2. Store array of objects holding information of bodies in the system 3. Accurately display objects in the system to the canvas 4. Accurately update position and velocity of each body in the system   This provides the core features of the program, necessary to produce a sandbox-simulation. |
| 02 | Allow user to manipulate main simulation environment | Implementing the following features:   1. User input to change camera position / zoom 2. User input to create various new bodies into the system 3. User input to remove bodies from the system 4. User input to determine simulation time rate – speed of simulation 5. User input to alter attributes (mass, position, velocity, diameter) of bodies in the system   This provides features making the program a sandbox-simulation as opposed to a simulation. |
| 03 | Allow user to investigate information / attributes about/of bodies in the simulation environment | Implementing the following features:   1. User input to select body to display its attributes 2. User ability to view text / image based definitions / explanations of relevant physics |
| 04 | Produce learning environment | Sub menu accessible from the pause or main menu allowing the user to choose to choose sections of educational text and images regarding the relevant physics. |
| 05 | Main menu | Display text describing the options which can be selected/clicked, taking the user to the relevant sub-menu. The following are the options:   * New simulation * Load simulation * View/load community simulations * View/change settings * Learn * Exit program |
| 06 | Pause menu | Same requirements as the main menu, accessible by pressing escape or equivalent whilst in the main simulation environment with the following options:   * Continue / un-pause * Learn * Settings * Main menu |
| 07 | User management | Allow users to create and log into an account. Details, saved simulations and preferences of the account stored in a database. |
|  |  |  |

## Program Structure

A diagram of a company

AI-generated content may be incorrect.This hierarchy diagram shows the order of the program flow. Note that a ‘\*’ connotes that the element appears elsewhere, redirecting the flow of the program to a previous position. The following diagram is a flowchart describing the same abstracted program flow.



## Post Development Testing

For each of the success criteria stated in the [analysis section](#_Success_Criteria), a post development test plan must be created to identify where any features may not meet their respective criteria.

The overall quality and usability of the program can be reviewed with a stakeholder usage test, collecting quantitative and qualitative information on what has and has not met their expectations and what they would like to see altered if any further development were to take place. This could be done by performing an alpha test by sending a link to use the program to the survey respondents alongside a complementary survey.

The robustness of the program can be tested by rigorously testing inputs which are within, at the boundary and outside of the boundary of what is expected. Reviewing the program’s reaction to these fringe inputs will allow the robustness of the program to be judged. Ideally, fringe inputs should be handled accordingly and not cause a crash or other undesirable results.

| # | Criteria | Explanation | Expected result |
| --- | --- | --- | --- |
| 01 | Is main menu displayed on program start, allowing the user to select between the following choices?   * new simulation * load simulation * view community simulations * learn * change settings * exit program | Menu should display the stated choices and selecting each option should cause that section’s menu to be shown | If the mouse’s x,y position overlaps an option’s displayed text and mouse presses left mouse button, that option’s menu should be displayed / program exit.  Likewise for gamepad and touchscreen, navigation to desired option and pressing a button or touching the desired option. |
| 02 | Is the user able to move the camera’s position – x,y – within a simulation using the following input methods?   * keyboard and mouse * gamepad * touchscreen | The user should be able to alter the position of the camera within the simulation | If user inputs the arrow keys or the corresponding w,a,s or d keys, the value of the camera’s x and y position should alter accordingly to some variable camera speed and amount of time the keys are held |
| 03 | Is the user able to select from a range (3+) of default starting situations from a simulation when selecting ‘new simulation’? | The user should be shown the option of at least three starting states of their sandbox-simulation | Upon selecting ‘new simulation’ the user is presented with a menu containing titles and descriptions of starting states for a simulation. They can select their desired start by clicking on the section with the title/description. They are then taken to the main simulation environment with that starting state. |
| 04 | Is the user able to view the current simulation time as well as stop, change and alter the rate of the simulation time? | The user should be able to start and stop the simulation speed discretely and choose to vary the simulation speed within a reasonable range. | User can use spacebar to start & stop the simulation time and use slider or equivalent to vary speed in a range should they choose to. |
| 05 | Is the user able to pause the simulation, displaying a pause menu where the user can choose from the following options?   * Continue * Save / Save as simulation * Settings * Main menu | The user should be presented with the stated text options when opening the pause menu and the corresponding menus should be shown when the user selects each option. | User can use the mouse to highlight and click the stated options in the menu |
| 06 | Do the objects within the simulation act reasonably accurately for a simulation of motion? | The motion of the bodies in the simulation should be accurate to a reasonable degree. This is to teach and most importantly not mislead users on the relevant physics. | Massive bodies should accelerate towards each other. If the difference in velocity is sufficient, they should orbit and not collide. Compare necessary speed in simulation to calculated necessary speed in reality. |
| 07 | Is the user able to manipulate the state of the simulation in the following ways?   * Add new objects * Remove existing objects * Alter the following attributes of existing bodies: mass, position, velocity, diameter | Adding, altering and removing bodies from the system should not cause the rest of the simulation to fail with a crash or unexpected result. | Adding, removing and altering the characteristics of the bodies in the simulation should affect the motion of the rest of the simulation as expected, not causing crash or other bugs. |
| 08 | Is the user able to log into the program on launch with a username and password with their user preferences being saved E.g. volume and graphical settings? | Users’ should be able to save their preferences so that they don’t need to visit the settings menu each use of the program. | Upon starting the program, users should be presented with a functional choice to log into or create a new account with a username and password. Whilst logged in, changed settings should be remembered upon their next logged in session. |
| 09 | Are users able to save the state of their simulations to the program’s database and load them in various sessions of program use? | Users should be able to save their simulations, linked to their account so that they can continue using them in later use. | When a simulation is saved, the state should be preserved with no noticeable changes upon saving and loading multiple times. |
| 10 | Is the user able to view definitions and explanations of physics terms at differing levels of detail via the following methods?   * Clicking / selecting highlighted terms to view definitions * Using a ‘learn’ section to find explanations / definitions |  |  |
| 11 |  |  |  |

# Cycle #1

## Design

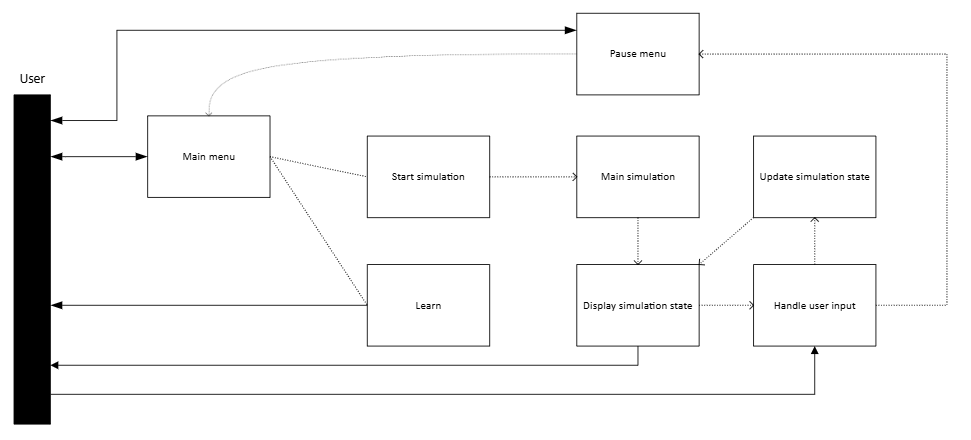
### Problem decomposition

The following is a table detailing the features to be implemented in the first cycle of development. These features when implemented will result in the simple structure of the program, that being a main menu leading to the simulation or a learning menu.

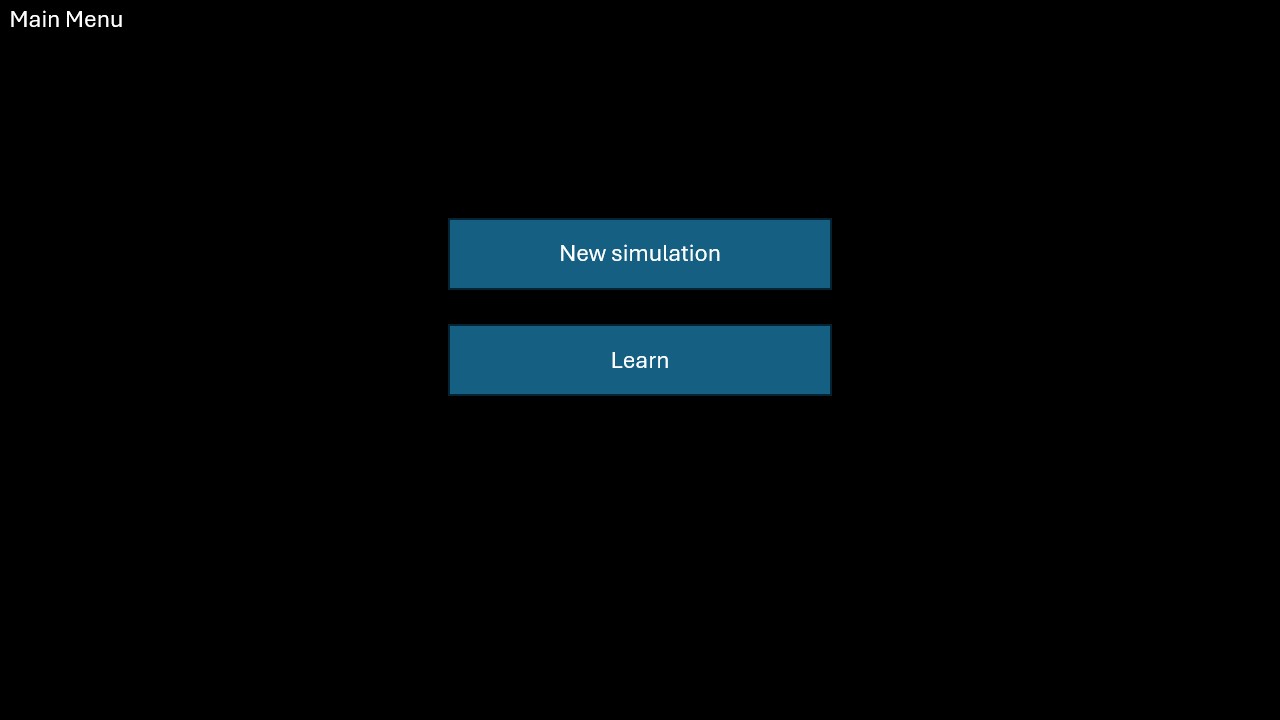
|  |  |  |
| --- | --- | --- |
| # | Feature | Explanation & Justification |
| 01 | Program state: main menu | Display simple background with 2 rectangular buttons containing ‘start simulation’ and 'Learn’ text.  These options have been chosen to be the first to be implemented as they are  Clicking each button changes the program’s state to the corresponding value.  Initial program state. |
| 02 | Program state: main simulation | Display simple background with state of the simulation overlaying it.  Simulation start state has camera showing a centred earth with orbiting moon. |
| 03 | Output: display simulation state | Display simple circular images of corresponding bodies at accurate position and size relative to camera position. Precise algorithm to be described later.  To be done each frame of the main simulation. |
| 04 | Input: camera movement | Adjust camera’s position in the simulation based on some variable scale factor and user key input. E.g. w,a,s and d, moving the mouse whilst clicking or arrow keys.  To be checked each frame of the main simulation. |
| 05 | Process: Update bodies’ physics | Each time step, every body must have its physics updated. This involves calculating the resultant force on the body, calculating its acceleration vector based on its resultant force then updating its velocity based on its acceleration then updating its position based on its velocity. |
| 06 | Program state: Pause menu | Accessible by pressing the escape key  Menu displaying rectangular buttons for the following options:  - Main menu  - Learn  - Un-pause  Where clicking on each button changes the game state to the corresponding value. |
| 07 | Program state: Learn | Sub-menu accessible from buttons in the main menu and pause menu.  Buttons leading to sub-menus for the following sections:  - Simulation tutorial  - Info on how to use the simulation e.g. camera and other controls  - Physics info  - Initially one sub-menu titled “Newtonian Mechanics”, with a second sub-menu titled ‘Gravity’ which when selected displays text containing information on gravity related to the program. |
| 08 | Input: Simulation speed | Adjust simulation speed based on user inputting a scale factor / selecting from useful preselected choices. |
| ? | ? Alter body attributes | ? Adjust body characteristics such as mass, position and velocity with button / slider / input |

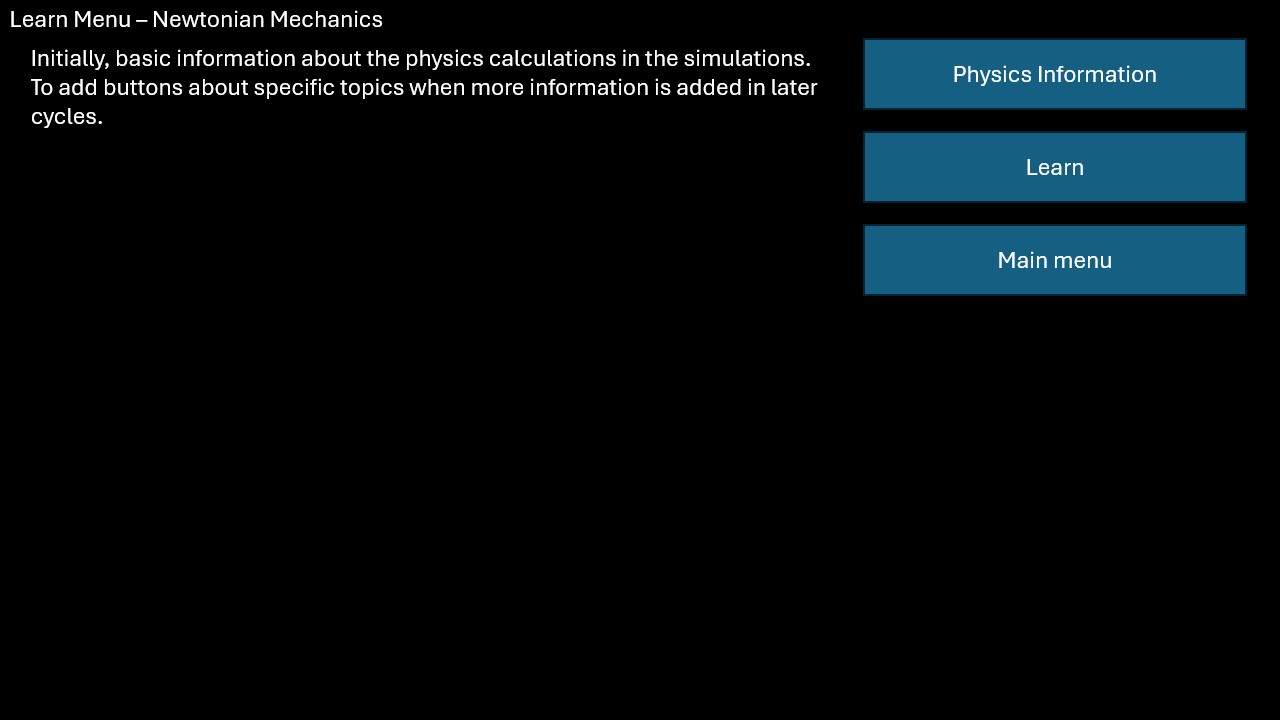
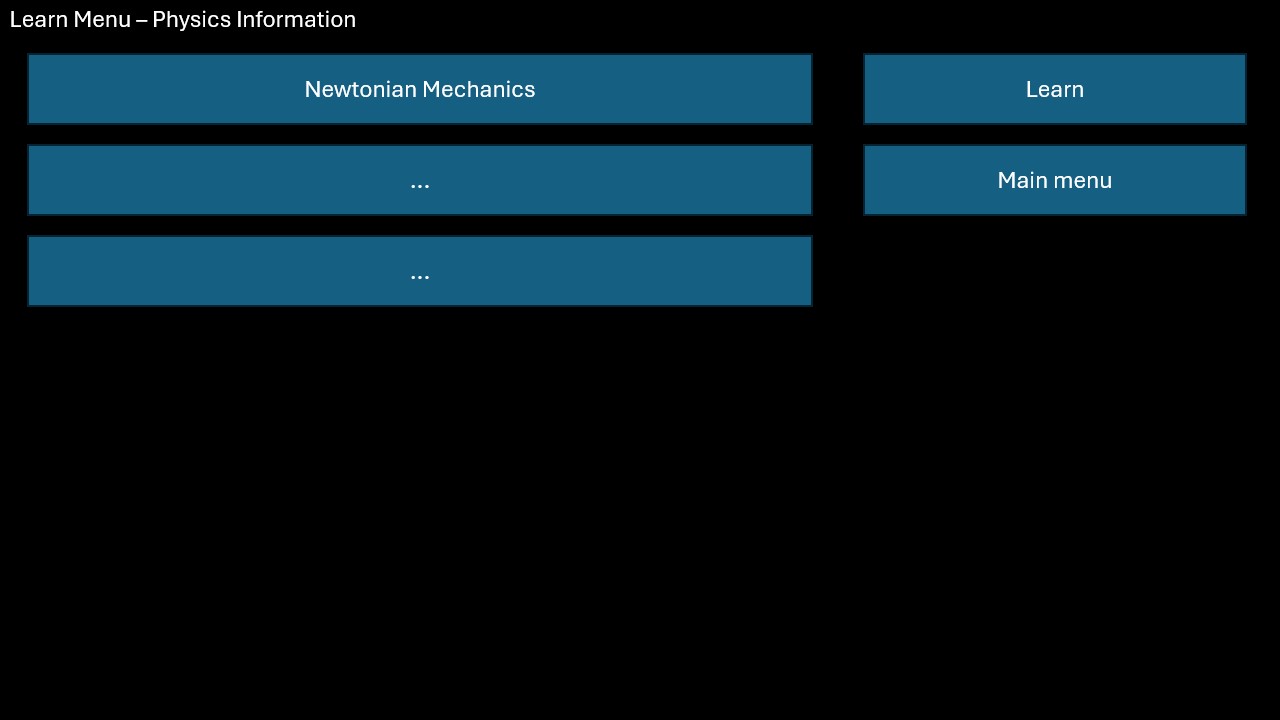
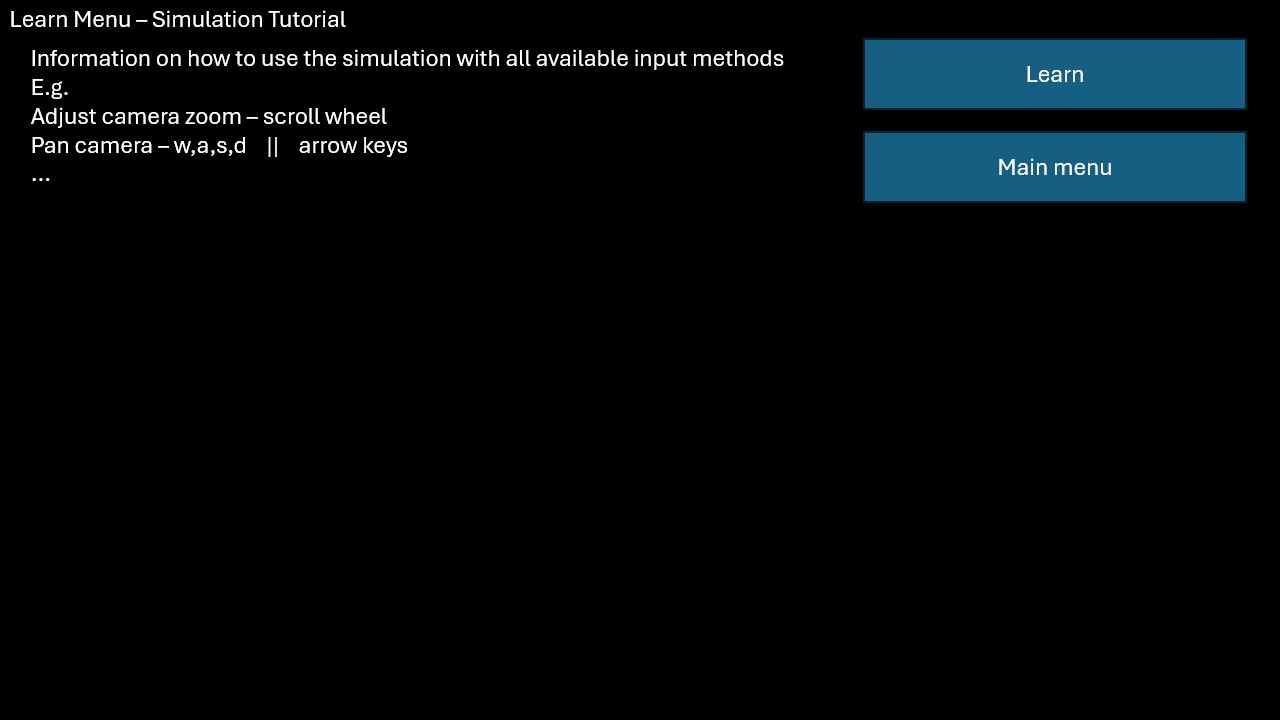
### Use Case Diagram

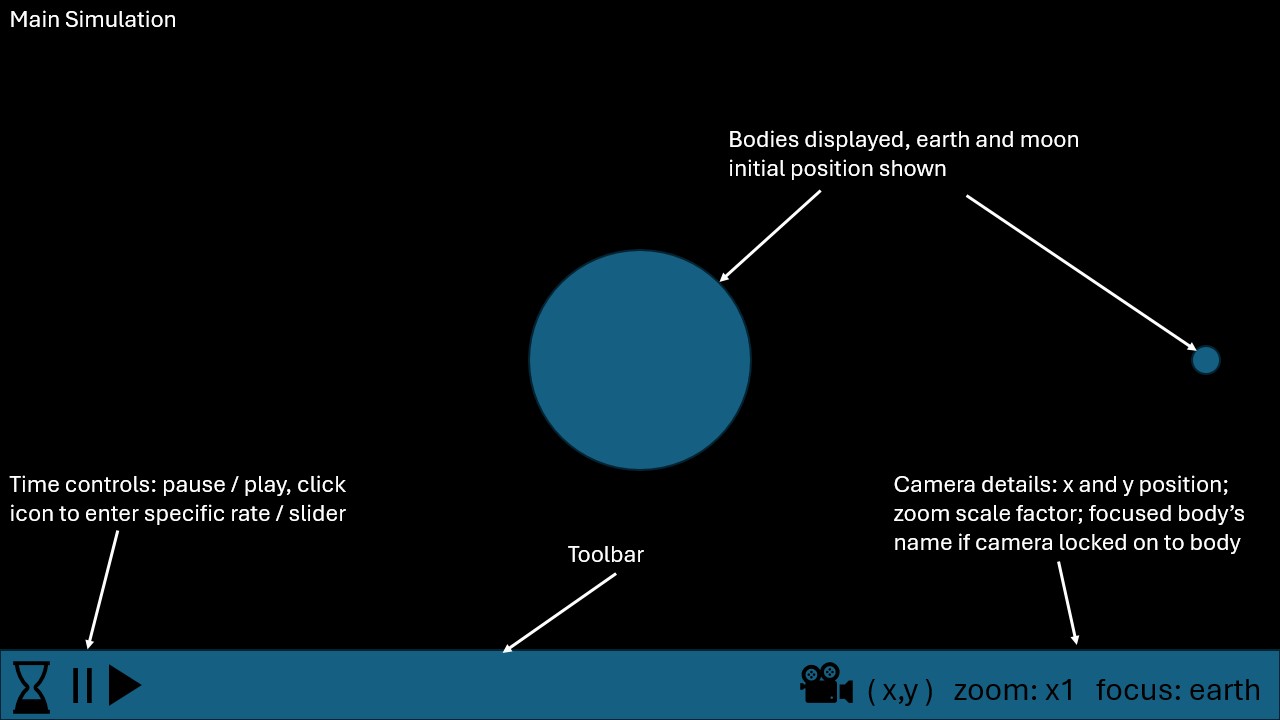
This diagram outlines the flow of the user’s use of the program with arrows to and from the user indication when input and outputs of the program are occurring. Specific details as to how the processes will be performed have been omitted to demonstrate more clearly when input and outputs are happening.



### Usability features

The following images demonstrate the general layout of the canvas for the program when it is in the given states: 





Having a text description of the currently displayed state to the user in the corner of the canvas is beneficial to user experience of navigating the program. Combined with having simple and consistent menu and HUD (heads up display – toolbar) design leads to less confusion for first time users.

### Assets

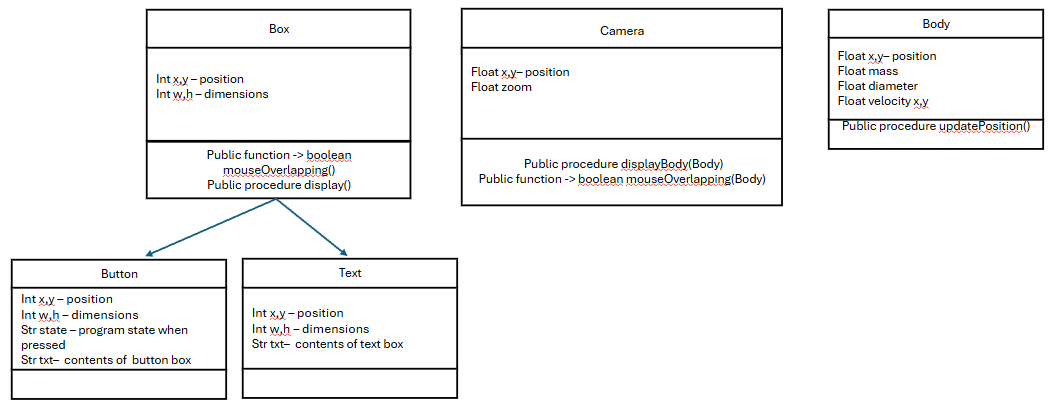
|  |  |  |
| --- | --- | --- |
|  | Simple pixelated planets & moon | https://opengameart.org/content/planets-4 |
|  | Simple pixelated sun | https://opengameart.org/content/sun-earth-and-moon |
|  | Menu font | https://www.dafont.com/a-space.font |
|  | Dark starry background | https://opengameart.org/content/space-assets |
| To be confirmed | Icons for buttons main simulation | https://opengameart.org/content/free-buttons-gui-pack-1 |
|  |  |  |

### Variable Table

Important variables along with a description of their use and data type:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data type** | **Description** |
| state | integer | Hold program state e.g  State = 1 -> main menu displayed  State = 2 -> learn menu  ... |
| box | class | Parent class of menu objects |
| button | Class – inherits from box | Button to change state in menus  Method to display button & string contents to canvas |
| text | Class – inherits from box | Display information in menus  Method to display text to canvas |
| camera | Class | Holds position and zoom of camera.  Methods to display bodies to canvas |
| body | Class | Holds position, mass, velocity, diameter of body |
| time | float | Holds the elapsed time of the simulation in seconds |
| timeRate | float | Holds the rate that simulation time is elapsed (simulation seconds per second) |
|  |  |  |

### Class Diagrams



### Use of data structures

After instantiation, button and text boxes for each menu will be stored in an array of objects. This is so the objects in the array can be iterated through and have the same methods executed each frame when necessary. Similarly, for the main simulation, the initial and added bodies will be stored in an unordered list, allowing for linearly searching through the bodies for specific characteristics and/or iteratively performing similar processes on each body.

### General Input Process Output table

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Precondition** | **Process** | **Output** |
| Click mouse left | Mouse overlapping button, button is being displayed | Program state changes to state stored in button object | Program state changes, different menu screen / main simulation displayed |
| w,a,s,d / arrow keys | Program state main simulation | Update camera position | Displayed bodies appear to move on canvas relative to camera |
| Array of bodies, timeRate | Program state main simulation, timeRate not currently paused | Update velocity & position of each body in array according to position and mass of the others | Bodies appear to accelerate and move as simulation time advances |
|  |  |  |  |

### Process Pseudocode

|  |  |  |
| --- | --- | --- |
| **Process** | **Pseudocode** | **Explanation & justification** |
| Mouse overlaps box | IF mouseX > button.x - button.width / 2  AND mouseX < button.x + button.width / 2  AND mouseY > button.y - button.height / 2  AND mouseY < button.y + button.height / 2  Return true  Else  Return false | Compares mouse’s co-ordinates with the box’s co-ordinates and returns true if the mouse’s position is within the rectangular bounds of the box.  As all menu items are rectangular boxes, this method will be sufficient to determine whether the mouse is within a given box or not. |
| Camera movement | If keyPressed(‘a’)  Camera.pos.x -= cameraSpeed  Else if keyPressed(‘d’)  Camera.pos.x += cameraSpeed  Else if keyPressed(‘s’)  ... | This updates the camera’s position according to keys being pressed on the keyboard by the user. This specific method prioritises inputs in the listed order in the source code, potentially leading to some user confusion due to apparent inconsistency. |
| Update body characteristics | For i = 0 to bodyArray.length  For j = i + 1 to bodyArray.length  // between each pair of bodies in array  // calculate force (N) between them  ForceMagnitude = (GravitationalConstant \* massi \* massj ) / dist(positioni , positionj) \*\* 2  //find unit vector in direction between them  //find difference between positions, the magnitude of this vector then divide each element in the difference vector by the found magnitude  Difference = posj – posi  Magnitude = sqrt(difference[0]\*\*2 + difference[1]\*\*2)  UnitVector = difference / magnitude  // calculate force vector by multiplying unitvector by the force magnitude  ForceVector = unitVector \* forceMagnitude  // find acceleration vectors (m/s^2) by dividing force vector by mass of bodies, using –1 \* force vector for second body  AccelerationVectori = forceVector / massi  AccelerationVector = -forceVector / massj  // add the respective velocities to the bodies  Bodyi.addVelocity(accelerationVector)  Bodyj.addVelocity(accelerationVector) | // indicates comments which explain the pseudocode. |
|  |  |  |

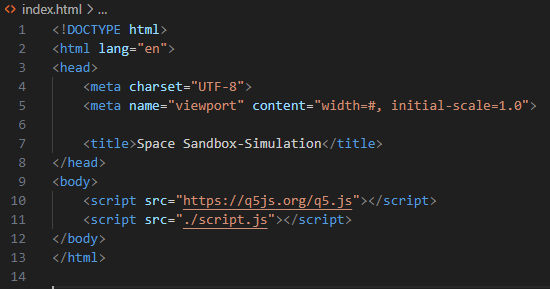
### Test Plan

|  |  |  |
| --- | --- | --- |
| **Feature tested** | **Test process / data** | **Expected outcome** |
| Menu buttons | Position mouse within button box bounds; click left mouse button; program state should visible change, displaying chosen menu / main simulation.  Observe programState variable with debugger tools. | Program state should be updated to that of the button object and the program should display the corresponding state. |
| Camera movement | Enter main simulation using buttons, using designated camera movement keys, observe x,y variables of camera and movement of bodies on the canvas | Arrow keys and w,a,s,d should alter x and y values of camera accordingly while bodies should move in the corresponding direction, camera moves right, body appears to move left. |
| Body physics | Initialise 2 bodies, earth and moon at accurately scaled position, velocity and mass and observe movement.  Vary simulation speed and observe impact on accuracy | Moon should orbit earth once ever 27.3 simulation days |
|  |  |  |

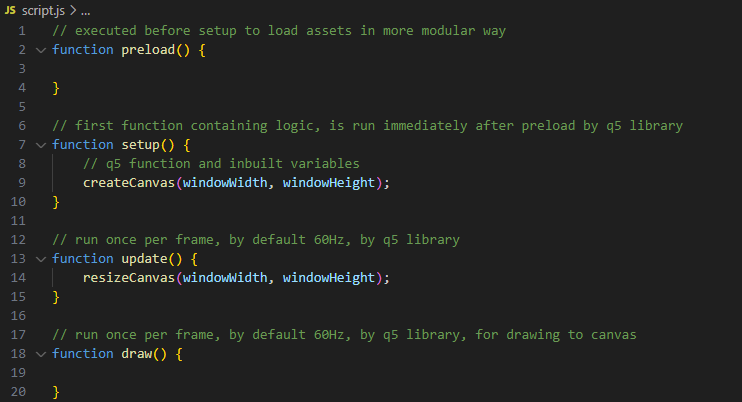
## Implementation

### Milestone #1 – Main Menu

Implement main menu / title screen state with buttons changing program state to their corresponding values.



This is the necessary html to link the q5.js library and the following is javascript to display a simple grey background on a canvas.



This results in the following when index.html displayed by browser:

A white rectangular object with a black border

AI-generated content may be incorrect.

Here I’ve implemented a parent ‘box’ class to be used for menu navigation