Malvern College

Contents

[Analysis 2](#_Toc161043633)

[Introduction to the Problem 2](#_Toc161043634)

[Suitability of a Computational Solution 2](#_Toc161043635)

[Features of My Solution 3](#_Toc161043636)

[Analysis of current solutions 3](#_Toc161043637)

[Summary of Analysis of Current Solutions 11](#_Toc161043638)

[Stakeholders 12](#_Toc161043639)

[Interview Questions 13](#_Toc161043640)

[Interview Response 14](#_Toc161043641)

[Analysis of Interview Responses 17](#_Toc161043642)

[Decomposition 18](#_Toc161043643)

[Abstraction 18](#_Toc161043644)

[Essential Features of Solution 19](#_Toc161043645)

[Requirements 19](#_Toc161043646)

[Stakeholder Requirements 19](#_Toc161043647)

[Additional Requirements 20](#_Toc161043648)

[Success Criteria and Methods of Validation 20](#_Toc161043649)

[Limitations of the Simulation 22](#_Toc161043650)

[Hardware and Software Requirements 22](#_Toc161043651)

[Design 23](#_Toc161043652)

[Algorithms Overview 23](#_Toc161043653)

[List and Explanation of Algorithms 24](#_Toc161043654)

# Analysis

## Introduction to the Problem

I was inspired by the British Computational Physics Olympiad to undertake this problem. The problem involves simulating the solar system and the orbit of planets within it. I thought, why can’t we have any solar system with any gravitational constant? And decided to make this my project.

The target is to create an application/website that allows a user to create their own solar system. They should be able to add planets, change their masses, change the gravitational constant and see what happens. This is a tool that can be used in education, for example teaching pupils astrophysics in A level physics. I take A level physics and so this is a good practice for me to learn more about physics as well.

The system should use a GUI to allow for user to make/edit their solar system with a way of changing it as per described in the above paragraph, and should simulate and display the orbit of those planetary bodies.

The system should have a way of storing different solar systems created by the user, storing data about each planetary bodies’ current position, velocity and acceleration, as well as constants such as their mass. Ideally the system could make use of webserver tools to host itself on a website where people can use this. For adding complexity the system could also make use of login systems and allow other users to view each other’s solar systems and simulate their orbits.

## Suitability of a Computational Solution

This problem requires calculation of acceleration due to the resultant force of gravitational attraction due to the masses of each planet the user adds. This solution needs to run on a computer for the simple fact that no one is capable of doing it faster than a computer, as you need to do multiple integrations to calculate the position vector of a planet at time t, and t ideally needs to be a small number for a suitable degree of accuracy. The solution may calculate the acceleration every 0.1 seconds, as it is impossible to find the acceleration for a time step that is approximately 0 in real time. There are no real way of implementing this without a computer.  
  
Moreover, the user will interact with the solution to add more planets and use user interfaces to change settings, which is quite hard and unthinkable of to implement with anything other than a computational solution.

The problem Involves a many stage calculation that require first finding the position vector of each planet, then using Newton’s law of gravitation to find the force planet A exerts on planet B due to gravitational attraction, then dividing by the mass to find acceleration. Once you have acceleration you need to integrate to find velocity and integrate again to find displacement. This makes it suitable for decomposition, which I will talk about how it applies to my solution later. It is also suitable for abstraction to take place since I will not be modelling any unnecessary parts of the solar system. Due to the two mentioned factors I wish to justify why this solution is suitable to a computational solution.

## Features of My Solution

To summarize the current problem, the program needs to have a GUI for user to add more planets, change the masses of planets and stars, set a time step to run at (e.g. 1 second in real life = 1 year in simulation), and see the orbit of the overall system.

Ideally the solution should have a means of storing what the user has created permanently but is not the main focus of the problem. The program could possibly have a login system and a way of viewing other people’s simulations.

The above are all features that I can think of, and I will consult with stakeholders for further input. To summarize, the features are listed below:

|  |
| --- |
| Have an intuitive GUI for the user to interact with my solution. |
| Have a setting or a way to change the masses of planets and stars |
| Have a setting or a way to change the time step it runs at (e.g. 1 second = 1 year) |
| The user can see the orbits and movement of planets simulated. |
| (Extension) – The user can save what they have created |
| (Extension) – The user can view what others have created |
| (Extension) – Webserver hosting to make it hosted on a webpage |
| (Extension) – Implement thick client and thin server |

More features are to be added after interviewing stakeholders.

## Analysis of current solutions

**List of current solutions (query result from google):**

1. UCL Planetary Orbit Simulation

<https://www.ucl.ac.uk/~zcapg66/work/COMP4/simulations/orbit/orbit.html>

2. UNL Astronomy Education Orbit Simulation

<https://astro.unl.edu/classaction/animations/renaissance/kepler.html>

3. JPL Solar System Dynamics

<https://ssd.jpl.nasa.gov/tools/orbit_viewer.html>

4. planetaryorbits.com

<http://www.planetaryorbits.com>

5. Sebastion Lague’s Coding Adventure (youtube)   
<www.youtube.com/watch?v=7axImc1sxa0>

Analysis of solution number 1 (UCL Planetary Orbit Simulation):

Introduction:

This is a software developed by UCL and this is an orbit simulation I found on their website.

Overall view:   
A screen shot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated

A screenshot of a game

Description automatically generated

List of features

|  |
| --- |
| A display to show the distance of mouse from sun |
| Left click to create anticlockwise planet |
| Right click to create clockwise planet. |
| Mass of the sun can be changed |
| Can display data of planets such as acceleration, speed, and radius. |
| Can pause/play and reset simulation |
| Can change the colour of the planets created. |

While the solution has many features that are suitable for my solution, and the GUI looks clean, there are some missing features that I want to demonstrate. The solution shown above uses a 2-body simulation method that utilizes Keppler’s laws of planetary motion which does not apply for a N-body simulation. My solution requires using N-body simulation methods and so I cannot use the same approach as theirs. They are also missing a way of changing the mass of planets, which they have justified as negligible in a 2-body simulation. The mass of each planets play a significant role on a N-body simulation with non-fixed orbits. However, I like how they can display data of planets, pausing and playing the simulation.

Summary of solution number 1 (Pros):

|  |
| --- |
| Has intuitive GUI |
| Can add planets which is what I want |
| Can change the mass of the sun |
| Can display data of planets |
| Can pause and play |

Summary of solution number 1 (Cons):

|  |
| --- |
| It is a 2-body simulation, which means that it does not apply to my solution |
| Cannot change mass of planets |
| Orbits are fixed |
| Cannot zoom in or zoom out |
| Planets can phase through each other |

I can take inspiration from their UI design as it is quite simple and I think I can make it quite easily. They have a well-documented webpage demonstrating how to use their application and so I think I could do something similar to that. However, I still need to come up with an algorithm myself as theirs does not apply to my solution. The zoom in and out feature is something that I can implement as I will be changing the orbits of planets and those can be quite big. I need to allow for the user to see the whole system or zoom into a specific part. Overall the simulation ran pretty fast with an average of 60 frames per second, and I want to achieve something similar for a smooth end user experience.

Analysis of solution number 2 (UNL Astronomy Education):

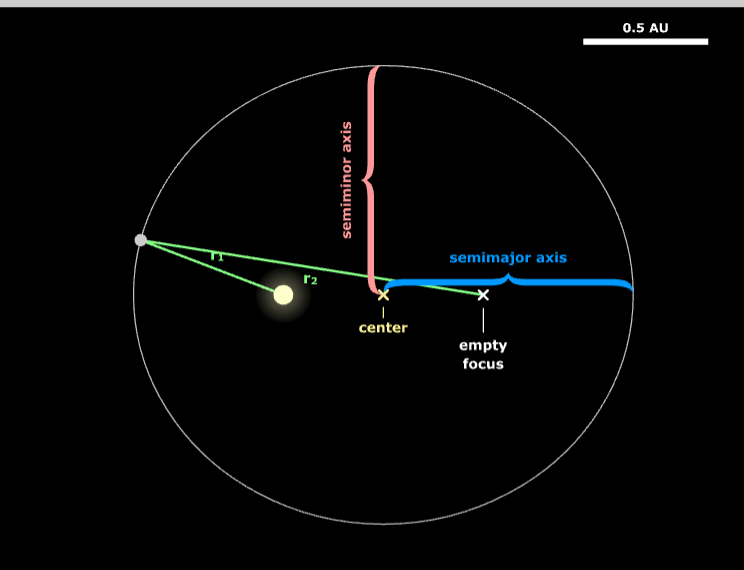
Introduction:

This is a website that hosts a simulation on it. I found this website on google when researching similar simulations such as the one by UCL above.

Overall view

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

A graph and diagram of a function

Description automatically generated

List of features

|  |
| --- |
| Ability to set parameters to a known planet |
| Changing semi-major axis |
| Changing eccentricity |
| Changing animation rate |
| Showing information about orbit such as empty locus and distance from loci |
| Plotting of areas swept in a certain time to demonstrate Keppler’s second law |
| Plotting of period against semi-major axis to demonstrate Keppler’s third law |
| Visualisation of the solar system’s orbits |

This solution has many features that allow for educational purposes. It successfully demonstrates many astrophysics laws such as Keppler’s law. It has a lot of UI going on which may be confusing for the casual user. If there is a way of simplifying the UI then it would be great. I think that overall this is a really good solution but unfortunately this solution does not apply to my scenario because, once again, it is only a 2-body simulation.

Summary of solution number 2 (Pros):

|  |
| --- |
| Lots of features to demonstrate physics |
| Intuitive UI if you know what you are doing |
| Many features to customise the simulation |
| Adaptive zoom that adjusts based on the size of the orbit so that it is always viewed in full, and in a reasonable scale. |
| Can pause and play |

Summary of solution number 2 (Cons):

|  |
| --- |
| GUI may be confusing if you don’t know what the terms mean, though you can easily find out by testing what happens if you move the sliders |
| Does not apply to my scenario |

Overall this solution is really good and once again I can use the GUI as an example as to how I might design my own. However, one problem with this is that it does not achieve what I want to do – to allow users to add more planets and simulate N-body scenarios. The amount of features and educational capabilities is something that I can implement in my solution.

Analysis of solution number 3 (JPL Solar System Dynamics):

Introduction:   
  
I found this website on google, and the 3d aspect of it is something that is worth looking into.

A screen shot of a computer screen

Description automatically generated

List of features

|  |
| --- |
| 3D plot of orbits |
| Viewing of current position of planets (approximation) |
| Clicking on a planet allows you to centre your camera at it. |

I think that this solution is a little bit weaker compared to the other ones as it misses quite a bit of features that the other two both have. The UCL one has the ability for user to add more planets which I want, and the second one has a lot of features for you to plot and change things. It allowed the user to change the orbits and experience Keppler’s three laws. However, for this solution you can only view the orbits. Despite being in 3d, the lack of the features made this a comparatively worse solution to my problem.

Summary of solution 3 (Pros):

|  |
| --- |
| In 3 dimensions |
| Flexible camera, as you can change centre to focus on each of the planets |
| Intuitive, as if you drag you see the camera angle change. |

Summary of solution 3 (Cons):

|  |
| --- |
| Too little features for the user to interact with the system |
| The camera is a little bit jittery and made me feel dizzy |
| Although it is in 3D, it is really hard to tell from a glance, and depth perception is hard due to the weird design of the axis. |
| No way of adding more planets or altering the orbits |
| Doesn’t really allow any demonstration of physics (in other words we can’t do much with it) |
| Slow bootup time |

Overall this solution is less suitable for my solution. I can consider what they did badly with 3 dimensions and try to see what I can do if I choose to move on to 3 dimensions after creating a working example with 2 dimensions. I also want to avoid the long start up time that slowed down the user experience.

Analysis of solution number 4 (planetaryorbits.com):   
  
Introduction:   
  
Yet another website I found on google, this one has more scientific graphs and different view angles. Slightly faster than the 3d one in terms of startup time.

A screenshot of a video game

Description automatically generated

A line with a line in the middle

Description automatically generated with medium confidence

A screen shot of a graph

Description automatically generated

List of features

|  |
| --- |
| 2D view from above and sideways. |
| View of planets from earth with variation of time. |
| Zoom in and out (not really flexible, as you can only zoom in to the inner solar system and zoom out to view the entire solar system) |
| Helpful GUI to show the user what they need to do to zoom in and out, or to change the view. |

This solution suffers the same drawbacks as solution 2. It only shows the solar system and does not allow the user to dynamically change the masses of planets and stars, and it does not allow the user to add more planets. The orbits are fixed and cannot be changed, though it allows for a way for the user to view it from different angles. When compared to the other solutions, this one is arguably the weakest as it has practically no features I am looking for and does not draw it in 3d, which ranks it lower than solution number 3.

Summary of solution number 4 (Pros):

|  |
| --- |
| Can view from different angles (though camera angle is static) |
| Has graph of stars and planets viewed from earth |
| Can zoom in and out |
| UI is pretty simple to use |

Summary of solution number 4 (Cons):

|  |
| --- |
| Uses a 2-body simulation to work out the orbits, presumably based on Keppler’s laws |
| Camera angle is static and fixed, and can only be changed by pressing a button. |
| Does not allow the user to zoom in and out using mouse wheel |
| No feature for user interaction with the orbits (the user cannot change the orbits) |

Overall I think what I can adapt from this solution is limited as most other solutions have enough features that I can try to incorporate in my solution. One thing I can learn not to do is how they manage camera. I can write it such that the camera can be moved with the mouse and the user can zoom in and out with the mouse wheel.

Analysis of solution number 5

Introduction:   
  
This is a youtube video that has been recommended to me by a teacher when I talked to them about my computer science project. It uses a N-body simulation using Newton’s law of gravitation and F=ma to calculate the position of planets using time steps.

A screenshot of a computer

Description automatically generated

List of features

|  |
| --- |
| User is able to freely add more planets |
| Able to change initial velocity of planets |
| Orbit is drawn upon creating planets, allowing for user to visualise what will happen without simulating it |
| The user can actually fly in a spacecraft to visit all of the planets |
| Planets can crash into each other and destroy one another. |

This solution is really cool and has lots of features that I am looking for. My end goal is basically something similar to what he achieved in this video. The user can create more planets and the system simulates the orbit of those planets. He managed to simulate a 3-body system in equilibrium. His approach was suited for a N-body simulation which is exactly what I want.

Summary of solution number 5 (Pros):

|  |
| --- |
| Uses a N-body approach for simulation |
| Seems to run pretty fast |
| Has a way of seeing the orbit before simulating, allowing for the user to make changes without having to run the simulation |
| Graphics are nice, but not sure if they have a way for a user to add more planets while playing the game. |

Summary of solution number 5 (Cons):

|  |
| --- |
| Doesn’t seem to be focused on letting the users create planets, rather is focused on having the user explore a fixed solar system. |
| Graphics may be unnecessary in my situation, as I am only providing a simulation, not a video game. |
| Initially he had problems with the physics breaking as he moved to the rim of the solar system, but he adjusted for that by moving the whole solar system around the user’s loaded area to avoid issues with memory. |
| Not sure if he used a time-step method, but if he did, then I should improve on it because it requires a time step that is extremely small for accuracy and it allows for the system to drift substantially over time. |

What I can take on board from his solution is the N-body approach. He used Newton’s law of gravitation for each planet to calculate acceleration then used a small time step to use during integration to estimate the positions of each planet. I can use the same algorithm to produce a similar result in 2D for my solution.

## Summary of Analysis of Current Solutions

**Identification**

So far, we see a few different approaches. One main approach I see in two of the solutions is to use Keppler’s laws of motion for 2 body simulations. They have used them to determine the position at any time given the semi-major axis, semi-minor axis, and eccentricity.

**Justification**

This is not a suitable solution to my problem, and I wish to justify this claim by saying that the goal of my project is to simulate a N-body situation. This feature of my problem makes this approach unfeasible.

**Identification**

Another approach was just to plot the orbits using currently known data and not allow the user to interact with it.

**Justification**

Again, this approach is unsuitable for my solution as I wish to create a system/application where the user can dynamically modify the system as needed (e.g. changing mass of planets, adding more planets). While the first two solutions which use the first approach have these features, they do not allow for a N-body simulation. The latter two solutions use this approach, but completely fail to meet the needs as specified earlier.

**Identification**

A third approach I have seen is from solution number 5. They have used a N-body approach for calculating acceleration then used a time slice approach for estimating the positions of planets. While this approach is suitable for my solution, I can make it better by solving a higher order of ordinary differential equations using algorithms such as leapfrog integration to ensure that there is no deviation over large amounts of time, and that I do not need to use an unnecessarily small time step which increases computational cost.

**Justification**

This approach is suitable for my solution as it is basically what I want. They have used a N-body approach which is suitable for allowing the user to add more planets and simulate the effects of gravity due to each of the planets.

## Stakeholders

There are a few stakeholders that may be interested in my solution. For example, a physics teacher may be interested in the solution as it can be used in education. They may use the solution to demonstrate the laws of astrophysics. Another stakeholder may be students who are interested in physics and want to use a visual way of learning about astrophysics, or students who are interested in computer science/maths and want to learn more about the backbones of my solution.

Analysing and summarizing what a physics teacher may use the solution for:

|  |
| --- |
| Teaching astrophysics |
| Demonstrating effects of gravity |
| Providing a visual way of learning for students |
| Research into planetary motion |

Reasons why my solution may be appropriate for a physics teacher’s needs:

|  |
| --- |
| An interactive simulation is likely to be helpful when teaching pupils about astrophysics. The teacher is able to change the orbit of planets and this allows the teacher to ask students for key terms such as the semi-major axis of the orbit. The teacher may ask students to draw in axis to demonstrate their knowledge. |
| My simulation aims to be able to change the gravitational constant and simulate its consequence, as well as changing the mass of planets. This is appropriate for the need of demonstrating the effects of gravity as the force of gravity exerted on each planet depends on the mass of the planet that is exerting the force as well as the gravitational constant. Hence my solution may be appropriate for this need. |
| A simulation is in itself a pictural of showing information. This allows for a visual way of demonstrating physics theories such as Newton’s law of gravitation. This means that the solution is highly likely to meet the need of providing a visual way of learning for students. |

Analysing and summarizing what a student may use the solution for:

|  |
| --- |
| Learning astrophysics |
| Entertainment – creating a solar system may be considered fun for some |
| Learning visually as mentioned before |
| Research into planetary motion and the mathematics behind it |

Reasons why my solution may be appropriate for a student’s needs:

|  |
| --- |
| It provides a graphical way of learning about astrophysics |
| Visualisation of solar systems can be entertaining and visually pleasing. |
| It is a good way to learn visually. |
| There is quite a lot of mathematics behind it which I can make the program demonstrate simply, to enhance the learning. |

### Interview Questions

For learning more about my demographic I will be using the following questions.

1. If you had an application that lets you simulate the solar system, add planets, remove planets, and adjust attributes of the planets such as mass and velocity, what would you most likely use it for?
2. Would you prefer a simple to use application with little features, or an application with lots of different features but slightly harder to understand and use?
3. Would it matter if you can store the solar systems you make and load it back?
4. Would it be great if you can load what other people have made?
5. How accurate would you like the simulation to be? Would you rather have it run faster at the cost of accuracy, or be more accurate at the cost of being slower?

Question 1 asks the stakeholders what the use of the application might be. For example they may say that they would use it for teaching astrophysics, and then I can try to tailor it for an educational purpose with explanations to show the physics and mathematics behind it.

Question 2 asks the stakeholders for their preference of ease of use. It helps me understand how complex the simulation should be. It may not be useful to simulate complicated physical scenarios such as nebulas by each individual molecule if the user does not want to deal with setting up a nebula with the application, and so they may be omitted. Some features that are not core may be tucked away into a separate menu to make it more accessible.

Question 3 confronts the stakeholders about the necessity of a permanent storage. I suspect that most of them would agree that it is quite important to store your work, as it may take some time to get the simulation right and for planets not to crash into each other by changing the initial velocity. It may be a useful feature. But if most of my demographic thinks otherwise then I may omit the permanent storage to focus on improving the simulation.

Question 4 asks about the need for a way to view other people’s work. I think that this is a good feature to have, as teachers may then be able to send stuff to students for them to have a play with it, to learn on their own. However, once again, if the stakeholders think otherwise then I may omit it to focus on other areas of the project.

Question 5 deals with the accuracy vs speed problem of a simulation. I basically want to know how accurate I need to make my simulation, and how fast I need my application to run at. If I use a different algorithm I may get a faster application at the cost of accuracy of the simulation. One example is if I use leapfrog integration or if I use a different order of ordinary differential equations to calculate the position vector of a planet at any time. Depending on which one I use I get more accurate results but use more time. The stakeholders may prefer to have a faster application that has a good level of accuracy that can roughly show the physics.

All of the above questions fulfil the basic knowledge I need to have about the demographic and I can analyse from the results what the stakeholders want from my solution and thus what I need to do in order to achieve them.

### Interview Response

Lisa, a student interested in physics.

[Me]: Hi do you have 5 minutes for a few questions. This is for my coursework for compsci.

[Lisa]: Yeah sure I have 5 minutes.

[Me]: If you have an application that lets you simulate the solar system, what would you most likely use it for?

[Lisa]: Science projects or competitions when I need to use a graph.

[Me]: Okay. Would you prefer to have an app that has lots of features but feel harder to use or an app that has simpler features but is easier to use?

[Lisa]: While I like having different features I think that, um, simple features may be better if it makes more sense to me.

[Me]: Would it matter if you can store solar systems and then load it back?

[Lisa]: Yes.

[Me]: Would it be great if you can load what other people have made?

[Lisa]: Yes.

[Me]: How accurate would you like the simulation to be? Like totally accurate but kinda slow, or not so accurate but fast?

[Lisa]: More accurate but at the cost of being slow.

- End of interview –

Steven, a teacher of physics

[Me]: So for my coursework I am planning to do something similar to this [shows solution number 2], where you’ve got like a star, planets orbiting around it, and then I want to be able to change the mass of the planets, change the mass of the sun, and then have this be moving as well.

[Steven]: Ok. By having this be moving you mean have the star moving as well?

[Me]: Yeah yeah, so all of the planets and stars affect each other, and then they move around. And then but this one uses Keppler’s laws of motion which does not apply for a N-body simulation. So basically the questions I have is if you have an application that lets you drag and drop and adjust the mass of the planets to see the orbits, what would you most likely to use the application for.

[Steven]: Uh, interesting. What would I most likely use it for…There are…there’s some GCSE topics that are always nice. There is a simple simulation by Phet. So maybe if you go on to Phet. [I tried to look it up] Well you don’t have time to look now you can just talk about that and what causes the change in the gravitational forces. We can do something like “let’s change the mass of the sun and see how it affects the orbits around the planets” and you can do the same for the Earth and the moon surrounding it. It can have a trail of where it’s recently been to trail the orbit so you can certainly use it to introduce gravitational fields to GCSE classes. And then also for Newton’s law of gravitation for your sixth form classes. And I’m just thinking now, I wonder if you had some values on there as well, like if it actually showed the gravitational force of attraction between two bodies. Then you could get them [students] to actually vary the masses and the distance between them and to maybe see if they can figure out the Newton’s law of gravitation and maybe what the factors are—

[Me]: So you want like, a setting where you can press [select] two planets and then show the force of A exerted on B and B exert on A?

[Steven] Yeah, yeah that would be quite useful, if you can change the distance or mass.

[Me]: I think I can let you like change the initial position of the planets by dragging it around and then you can start the simulation and see the values.

[Steven]: If they could change the distance, they [students] could at least figure out that the force is inversely proportional to r [radius of orbit] squared, like that would be a value right.

[Me]: And then my second question is: would you prefer a simple application with less features or lots of different features but might not like, seem very intuitive to use?

[Steven]: Well, what sort of extra features do you have in mind?

[Me]: Like if I look at the second solution I’ve got, you’ve got so many like different menus and settings. You can change the semi-major axis, you can change its eccentricity and stuff. But you can set it to show these [radial lines from loci, refer to solution number 2]. So when it moves, the lines actually follows it [the planet]. And you can have a setting to show the area. So this one is more based on Keppler’s laws but then I think there is like too many things going on? And it can be confusing if you are trying to learn one thing at a time or something.

[Steven]: Yeah, to be honest, I wonder…is it okay to say both? ‘Cause you can have…I keep going back to these Phet simulations again…you can have a basic layer or you can even have just, for example for circuits, they have “basic” and then “lab”. So yes, if I just want to teach kids about if I increase the mass then what’s going to happen there. Or you know, if we’re doing sixth form then we do want to have an advanced mode.

[Me]: Maybe I can tuck away all of the advanced stuff into like an advanced menu where you can change more stuff but then have like a more simple GUI for the sake of being simple at first.

[Steven]: Yeah absolutely, that’s alright.

[Me]: Ok. And then my third question is, let me just scroll down: do you think it’ll be better if you can like save what you’ve created and then load it back, or is that like not really necessary?

[Steven]: It depends on how easy it is to do...if I am completely honest, I think we’re unlikely to go back to exactly what we did before. Unless, you know that functionality of can we figure out the how of things, like can they figure out r^2, there may be a need of collecting a lot of data and go back to that. If it’s easy sure, but otherwise they can just do it in the lesson and store their data elsewhere. So yeah I wouldn’t say that’s a huge priority.

[Me]: Ok. And because that isn’t a huge priority, I assume that it won’t be a, very important for you to be able look at what other people have made?

[Steven]: No, I don’t think that needs to be on the software. I think-

[Me]: You can just go and look on the screens of the students.

[Steven]: Exactly so even we as a class, I use One Note all the time. But if I were to get you guys [my class] to do your work on One Note, and then try to mark it on that software, it just makes everyone’s life harder. And it’s also harder to show other people the work, which obviously isn’t the most important thing, but it is important.

[Me]: Ok so one final question is like: between the speed and the accuracy of the application, which one do you prefer? Do you prefer the application to be like very accurate at the cost of it running a little bit slow, or do you want it to run fast at the cost of being less accurate?

[Steven]: I think it depends on how extreme those values are, right? I think as people we are getting used to things loading almost immediately these days, and there is a cut off time for people to say this is too slow. And so long that it’s less than that. For these simulations, you [most people] are happy to wait for it to load for, 10 seconds? And so anything within that- I bet you there’s a value out there that people have researched and said right that is the cut off time.

[Me]: That is the point where people get annoyed.

[Steven]: Yeah where the general public get annoyed. But for these Phet simulations, some of them do take a little while to load. If you can get the values [experimental values] you are looking for as a result then I think it’s okay. I think 30 seconds is way too long, and 10 seconds is absolutely fine.

[Me]: Ok, that’s it. Thank you very much.

- end of interview –

### Analysis of Interview Responses

Looking at the response from Lisa, I think that the end users of this application may be used for science-related projects or competitions. She also thinks that it’s better to have simple features to not make everything seem cluttered. She would like to be able to save and load what she has done. One interesting point that contradicts my initial prediction is that she would rather the simulation take more time and be more accurate.

Steven thinks that the application can be used in an educational setting, which I agree. A new point that was made was that I can perhaps make a setting for the user to see the effects of gravity of one planet on another, when the user selects a pair of planetary bodies. While Lisa thinks that she would prefer having less features, Steven thinks that I can have the best of both worlds by hiding away the advanced settings if it may be too confusing. He also believes that loading/saving creations and seeing other people’s solar systems may not be entirely necessary in an educational environment, as teachers can just look at student’s screens physically, and it may in fact make it more inconvenient if they had to do it over the software. Finally he thinks that it is okay to have the simulation load for approximately 10 seconds but any more than that is not acceptable.

Key takeaways from the stakeholder interviews are listed as follows:

|  |
| --- |
| Purpose of the application is for science and education, so I should try to make it useful for that |
| Simple features at first seem appropriate, and then more capabilities of the application may be accessed in a different mode or hidden in a menu that users can click on if they so wish to, such that pupils from an earlier key stage may still be benefitted by the application |
| Saving and loading solar systems created may be useful for students exploring on their own but may not be as useful for teachers. |
| Stakeholders seem to prefer accuracy over speed of the application. Optimisation will be required if I use an accurate simulation, to ensure that the speed is satisfactory while providing a high level of accuracy. |
| An additional point that is not intended from the interview is that it may be good to have settings for the user to select two planets and see the interaction in terms of the force they apply on each other. |

## Decomposition

The problem is suitable for decomposition as you need to apply step-by-step algorithms to achieve the target. An initial view of what the steps I may take are as follows:

* Taking user input to add new planets and place it on the mouse cursor position.
* Allowing the user to drag and drop the planets to where they want it.
* Finding the force other planets exert on the planet.
* Dividing by the mass to find instantaneous acceleration.
* Using either a time-step approach (using very small time steps and calculating acceleration at those time steps then integrating to find velocity and position) or use a differential equation numerical method to solve for new position (such as the leapfrog integration method)
* Changing the attributes of planets (position, velocity)
* Updating the display of the screen to show the planets.
* Detecting if the user pushes the button to pause the simulation.

The entire project together may seem daunting but in their own are doable tasks. This justifies how it is suitable for computational method, such as decomposition.

## Abstraction

The problem involves physical calculations and that might cause some computational problems if I try to simulate everything that is going on in the universe to the molecular level. I can apply abstraction in this case and say that I will ignore what goes on down to a molecular level and focus on how it works on a larger scale that applies to my situation. I have decided to model the planets as perfect spheres which is not the case in reality, but for a simulation this is good enough. I will also model the planets as perfectly uniform with the mass at the centre of the spheres, which is not the case in reality which would affect the accuracy of the model. But since I am after a simple application that demonstrates the effects of gravity and mass, I can ignore the effects of the masses not being uniform, and whatever that happens in the molecular and quantum level.

## Essential Features of Solution

The following are the essential functionality of the solution:

|  |  |  |
| --- | --- | --- |
| Name | Justification | Priority |
| The program shall be able to plot the orbit of planets | The whole purpose of the project is to do this | High |
| The program shall be able to change the mass of planets | The project’s purpose is to allow for interaction with the orbits | High |
| The program shall be able to change the gravitational constant | The aim of the project is to have a mean of interacting with the gravitational constant | High |

## Requirements

### Stakeholder Requirements

From what I gather from the interview, the requirement from the stakeholder’s can be summarized as follows.

|  |  |  |
| --- | --- | --- |
| Name/Description | Justification | Priority |
| The program shall have simple features at first glance, and this includes being able to adjust the distance from each planets. | As suggested by stakeholders. Not as important as the above objectives which are the crux of the project, but still quite important | Medium |
| The program shall have an advanced setting for more display or adjustments such as selecting two planets and seeing the force they exert on each other | As suggested by Steven | Medium |
| The program shall have a way of displaying the force a pair of planets exert on each other | As suggested by Steven | Medium |
| The program shall prioritise accuracy over speed | As extrapolated from the summary of interviews. This objective is not as important as if the overall program works then it is fine, but this enhances the user experience. | Medium-Low. |
| The program shall be accurate to around 10% of the actual value | Accuracy is important as per above objective. | Medium |

That is all of the objectives I can extrapolate from the stakeholder interviews.

### Additional Requirements

Listed here are some additional/optional requirements that the project may need to meet.

|  |  |  |
| --- | --- | --- |
| Name/Description | Justification | Priority |
| The user interface shall be intuitive to use and not confusing | Enhancement of user experience. Not a main priority | Low |
| The program shall have a means of storing what the user have created using the software | As explained by Steven, this is not an important part of the project, and the user may not use this. This can be an extension to the project for anyone who may use it, such as Lisa. | Low |
| The program shall have a means of sharing what the user has created | As explained by Steven this is also not an important feature of the program. However, as users like Lisa may want to have this feature, this could be an optional feature for me to implement after the overall project hits the necessary requirements first. | Low |

### Success Criteria and Methods of Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Justification | Validation | Optional? |
| The program shall be able to plot the orbit of planets | Overall goal of the project | Screenshot of plot | No |
| The program shall be able to calculate the orbits of planet | Required in order to plot | Screenshot of different plots with different values for masses | No |
| The program shall allow the user to change the orbits | Overall aim of the project | Screenshot of user interface.  Video of user using program to change orbit. | No |
| The program shall allow the user to change the mass of planets | Overall aim of the project | Screenshot of user interface  Video of user interacting with the menu to change the mass | No |
| The program shall allow the user to change the gravitational constant | Overall aim of the project | Screenshot of user interface Video of user interacting with menu to change the gravitational constant | No |
| The program shall be accurate within 10% of actual value | Accuracy is more important than speed, as justified in the stakeholder analysis | Plotting of known planets and cross-comparison.  Using the program to calculate the error using known formulas for the algorithm that I use. | No |
| The advanced settings such as gravitational constant shall be tucked away into an advanced menu that the user can open | To prevent cluttering of the main window, and optimising the user experience as justified in the stakeholder analysis.  An optimisation objective, not a main objective. | Screenshot of user interface  Video of user interacting with the advanced menu | Yes |
| The program shall allow for the user to store what they have created | An option for the user to save what they created can be useful, but not necessary for the overall project as explained by Steven | A screenshot of the user interface A video showing the user interacting with the saving feature | Yes |
| The program shall allow for the user to share what they have created | The application may choose to use webserver hosting and user login systems to allow for the users to look at what other people have created | Screenshot of the user interface  Video of a user loading up what other people have made | Yes |
| The user interface shall be intuitive and easy to use | A clean user interface can make the user experience better.  An optimisation objective, not a necessity | Questionnaire/interview with stakeholders for opinion  Screenshot of the user interface | Yes |
| If implementing a user login system:   The database shall be secure and store the passwords as hashes | For security,  not very important for this task as it depends on whether I implement a system for multiple users | Screenshot of code  Screenshot of example password stored in database. | Yes |
| The program shall take no longer than 15 seconds to start up. | As discussed with Steven this may annoy the users. | Video of starting the application showing that it does not take longer to start | Yes |
| The program shall be able to allow for the user to log in to the system | Required for the users to load what they have created | Screenshot of login menu | Yes |
| There should be a play button for the user to press to start the simulation | Part of the GUI design | Screenshot of button | No |
| There should be a textbox or scrollbar for the user to adjust the mass of planets | Part of the GUI design | Screenshot of textbox/scrollbar | No |
| There should be a clear and concise tutorial about how to use the application | To illustrate how to use the software I developed | Screenshot of manual  Video showing how to use it | No |
| There should be a textbox or scroll bar for the user to change the gravitational constant | Part of the GUI design | Screenshot of textbox/scrollbar | No |
| The planets should be displayed as circles on the screen | Part of the graphics design | Screenshot of planets | No |
| There should be a button to open up an advanced menu for the user to adjust additional settings (if applicable) | Part of the GUI design, and something that I discussed with Steven | Screenshot of the menu | Yes |

### Limitations of the Simulation

The simulation is not all powerful. I cannot simulate down to the molecular level, and I cannot model the entire universe. Listed here are the limitations of the simulation.

|  |  |
| --- | --- |
| Limitation | Justification |
| The simulation will not be 100% accurate | It is theoretically impossible to model the position of planets continuously (e.g. with a time step that is basically 0), due to computational power |
| The simulation will not simulate each and every molecule separately | There is not enough computational power and I do not have enough knowledge to do so even when given the computational power |
| The planets will be modelled as perfectly spherical | To abstract away unnecessarily complex difficulties that comes with it.   It is not necessary for me to consider the unevenness of the planets, as the effect is negligible. |
| The planets will be modelled with their masses uniformly distributed within it. | To abstract away unnecessary complexities.  The effect of the distribution of mass is negligible in the scale/grand scheme of my simulation |
| The simulation cannot simulate infinitely many planets | There is not enough computational power to do so, as the computational cost increases exponentially with number of planets. If I had 3 planets and I wanted to add 1 to the system, I need to calculate 12 times, instead of 6 times. If I then add 1 more I need to calculate 20 times instead of 12 times. The increments increase further the bigger the number. This means that it becomes more and more difficult and slower as you add more planets in. |

### Hardware and Software Requirements

|  |  |
| --- | --- |
| Requirement | Justification |
| Windows 8 or later or equivalent operating systems, such as Mac or Linux | Requires python |
| 4GB of RAM minimum | Requires python, and requires pygame.  Pygame requires 4GB. |
| 1.5 GHz processor or equivalent | Need to be able to run python |
| 500 MB of storage | To store code |
| Has python installed | Project runs on python |
| Has pygame library installed | Project uses pygame to plot orbits |

# Design

## Algorithms Overview

The problem involves a couple of steps. I will break them down into different sets of problems for me to solve.

One of the set of problems I need to solve is the actual simulation itself. A step-by-step guide as to how I might tackle this problem is listed below.

|  |
| --- |
| Give a defined list of planets with their attributes defined (such as mass, position, velocity and acceleration). Define them as objects of the class planet. |
| For each planet that is in the simulation, I need to find the force each of the other planets exert on it.  To do this I can use the vector form of Newton’s law of gravitation.  Newton's Law of Universal Gravitation - ppt video online downloadsource <https://slideplayer.com/slide/273197/> |
| To find the force, I need to have the masses of the two planets involved, which can be accessed by the attribute of the planet object. (e.g. planet.mass). |
| I also need to find the distance between the two planets. I can do this by using:  Typed inside of desmos: <https://www.desmos.com/calculator>  Where I can get x1 by taking the first planet and then taking its .x value, and similarly for the rest of the variables. |
| Now that we have the distance between the planets, we need to use vector subtraction to find the vector r.  We can achieve this by doing (x1-x0)i + (y1-y0)j. This would give us the vector r using the standard unit vectors i and j. |
| Now we can find the unit vector r12, which is the unit vector that goes from object 1 to object 2 as shown in the diagram.  This can be done by dividing the vector r by the distance between them. |
| Now that we have the masses, the distance between them, and the unit vector, we can use the formula to find the force planet A exerts on planet B, given a value for gravitational constant. |
| Once I have the force of every other planet on one planet I am considering, I can add them all up to find the resultant force. |
| Once I have the resultant force, I can divide it by the mass of the planet I am considering to find its instantaneous acceleration. |
| Repeat for all planets in the simulation (it is this step that may be computationally expensive) |
| To find the next position I can apply leapfrog integration:  <https://en.wikipedia.org/wiki/Leapfrog_integration>  My acceleration is a function of all the position of the planets. I can write a method to find the acceleration of any planet at any time given all of the position of the planets, using the above steps.  I can then find the new velocity at half a time step away as it is the velocity at half a step before + the acceleration at the current step multiplied by the difference in time between time steps.  The new position is the current position + the velocity at half a step away multiplied by the difference in time between the time steps.  This is summarized in:  Source: Wikipedia, link is shown above |
| Now I just need to feed it to another algorithm for moving the planet to the according position.  I can achieve this by using libraries such as pygame to display the planets. |

Another set of problem is the setting up of the simulation. A step-by-step algorithm is listed below:

|  |
| --- |
| Have a button detect user input. If the user clicks on it, add a planet at the origin. |
| Initialise the planet to have attributes (mass, position, velocity and acceleration) to default values. |
| Have a button on the planet that detects click. If the user clicks on it open a menu that has settings for the user to change the values. |
| Use either a scrollbar or input box for values depending on complexity. If it is simpler then just use a input box. |
| Sanitise the input by ensuring that it does not exceed the limitations of the model (mass not too big, velocity not too big, etc) and does not contain code. |
| Once the user hits enter, check that the input does not contain any unallowed characters such as !@#$%^ and update the attribute of that planet accordingly. |
| Wait until the user hits play before beginning the simulation. |

## List and Explanation of Algorithms

A list of the algorithms I will use is listed below:

|  |
| --- |
| Vector subtraction |
| Finding the unit vector between two points |
| Finding distance between two points in a 2-dimensional grid |
| Finding unit vector between two points in a 2d grid |
| Dividing a vector by a scalar |
| Finding the force a planet exerts on another one |
| Finding resultant force exerted on one planet |
| Finding acceleration on one planet |
| Finding the velocity at half a time step away |
| Finding the position one step away |

Vector subtraction:

|  |
| --- |
| Input the x and y coordinates for the two vectors. In this algorithm we are taking away the first vector from the second vector. We are doing second - first , which would give us the vector that goes from the first vector to the second vector. |
| Subtract the x value of the first vector from the x value of the second vector. |
| Subtract the y value of the first vector from the y value of the second vector. |
| Define/instantiate a new vector with x value of the computed x value, and y value of the computed y value. |
| Return the new vector. |

Finding the distance between two points in a 2d grid:

|  |
| --- |
| Input x and y coordinates for the two vectors. |
| Use the vector subtraction algorithm as shown above, and store the returned vector. |
| Take the square of the x value and the y value of the vector. |
| Take the square root of the sum of the squares (of x and y) |
| Return the value |

Pseudo code for finding the distance between two points:

|  |
| --- |
| x1 = integer(input())  y1 = integer(input())  x2 = integer(input())  x2 = integer(input())  # takes 4 values, 2 for each vector, and turn the input into integers  x\_diff = x2 – x1  y\_diff = y2 – y1  # take the difference in x values and the difference in y values  x\_diff\_square = x\_diff \*\* 2  y\_diff\_square = y\_diff \*\* 2  # take the square of them  sum = x\_diff\_square + y\_diff\_square  # add them up  return sum\*\*(1/2)   # return the square root of the sum of the differences |

Finding unit vector between two points in a 2d grid:

|  |
| --- |
| Take the distance between the two vectors using the algorithm above |
| Use vector subtraction |
|  |
|  |