KING'S College LONDON

Virtual Reality Solar System

Supervisor: Prof. Eugene Lim

Project 1: Jaymin Shah, Sahildeep Kapour
Project 2: Mohammad Shikha, Jiahua Fan
Project 3: Agha Khan, Mateusz Nowak

Objective 1: Build a Scale Model Solar System on the Unity Engine

Objective 2: Simulate Orbits using Newtonian Mechanics

Objective 3: Using Unity, implement a Virtual Reality Experience to explore the Scale Model Solar System

Newtonian Physics

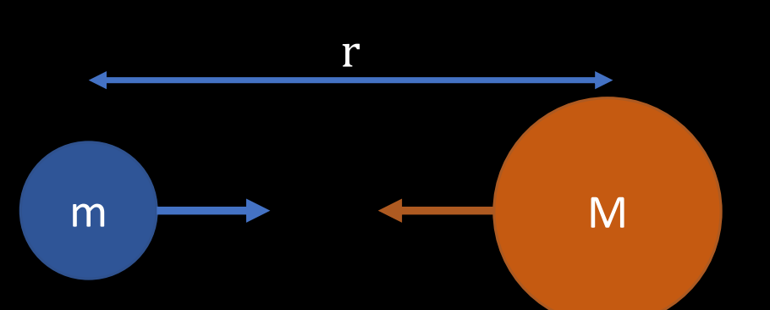
Newton's Law of Universal Gravitation

Newton compared the acceleration of the Moon to the acceleration to a falling object on Earth and deduced these are the same.
Newton's 2nd Law states:

$$\vec{F} = m \cdot \vec{a}$$

This mutual force was inversely proportional to the distance between the objects, therefore coming to his Universal Law of Gravitation:

$$F = \frac{GMm}{r^2}$$



This figure showing how the masses would be oriented with the distance shown (R)

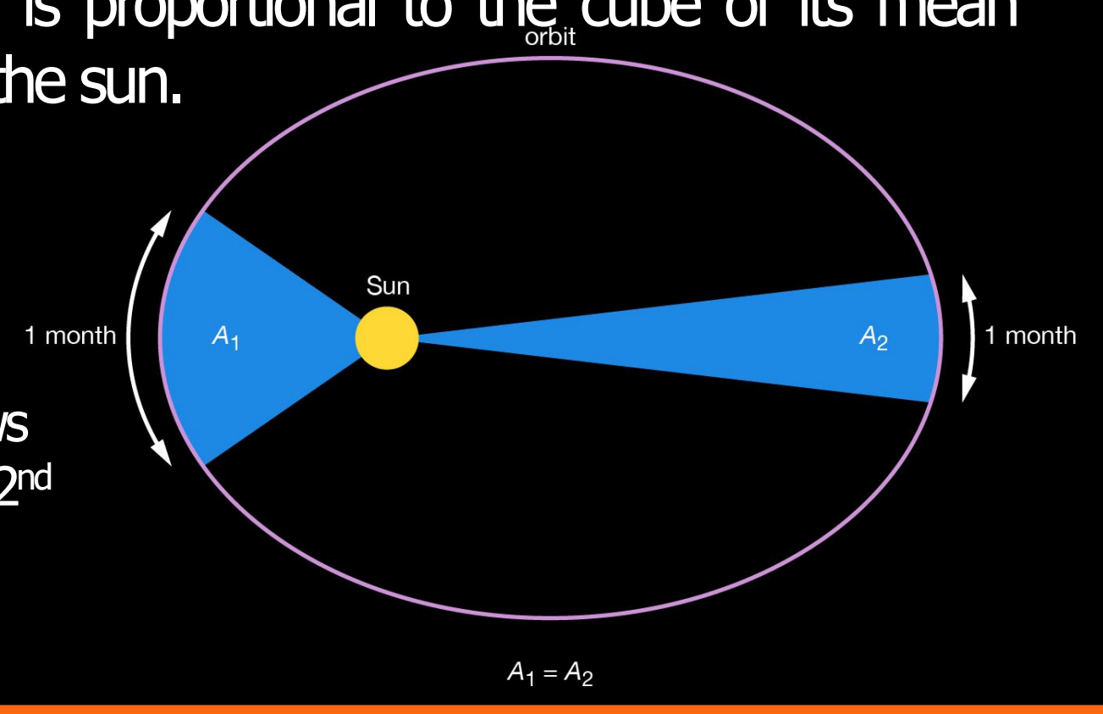
Kepler's Laws of Planetary Motion

Johannes Kepler and Tycho Brahe used observations to derive the following laws, which inspired Newton:

1) Every planet travels round the sun in an elliptical orbit, with the sun at one focus

2) The line joining a planet with the sun sweeps out equal areas in equal times.

3) The square of the time taken by any planet to make a complete orbit is proportional to the cube of its mean distance from the sun.



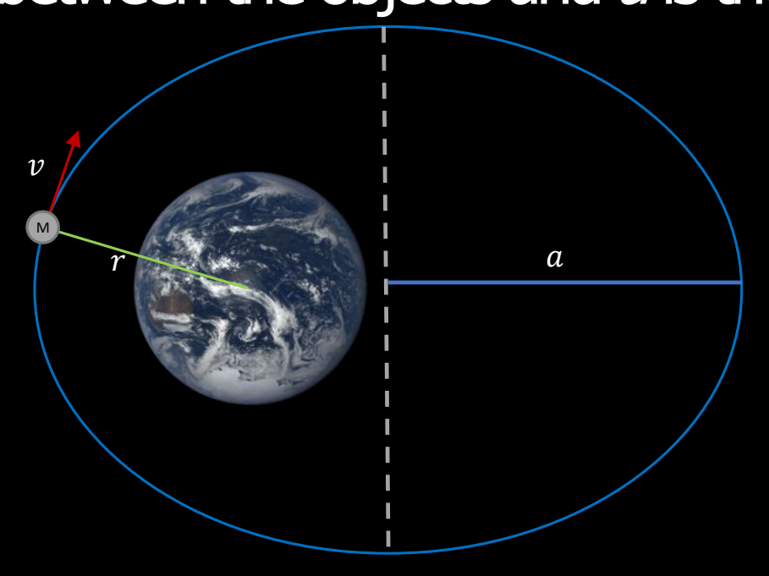
This figure shows Kepler's 1st and 2nd laws.

Vis Viva Equation

The Vis Viva Equation, or the Orbital-Energy-Invariance Law, is an equation [1] that can be used to calculate velocities of orbiting bodies in elliptical orbits. The equation is:

$$v^2 = \mu\left(\frac{2}{r} - \frac{1}{a}\right),$$

Where μ is the geocentric gravitational constant, r is the distance between the objects and a is the semi-major axis of the orbit.



For example, this is used in Binary-system simulations to calculate the velocities of Alpha Centauri A and B as they process around each other.

The Gravitational Constant

Taking Newton's Law of Gravitation and Centripetal Force felt by an orbiting object, we arrive at Kepler's third law.

Thus we find an equation for the Gravitational constant:

$$G = \frac{4\pi^2}{T^2(M+m)}a^3$$

This governs how quickly simulations evolve and depends on the time period of one planet that we use as a control, i.e. the Earth.

Unity

Basic Unity functionality explained

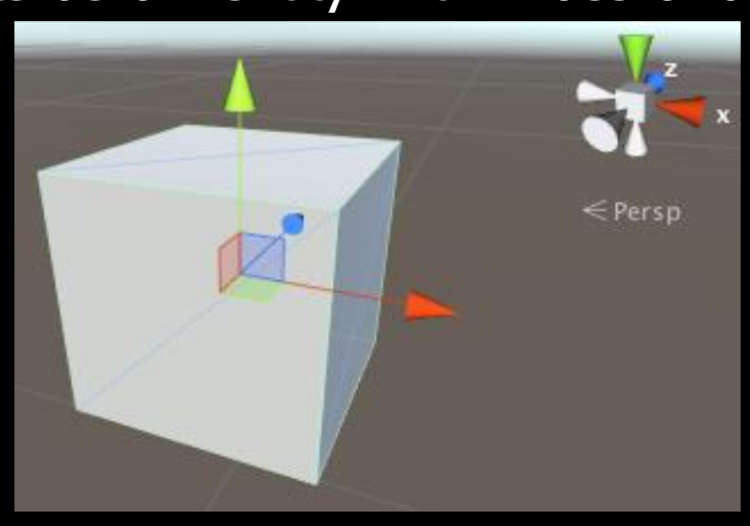
Some keywords will be mentioned a lot when explaining the project, the following are:

- **GameObjects:** Fundamental objects within Unity used for props, like lighting and cameras which are used to build your scene.

- **Components:** Each GameObject can have components attached via the Unity Editor. These give additional functionalities like custom scripts for specific behaviours.

An example used in all projects would be a **RigidBody**. If an object has a **RigidBody** component, it acts as an entity with mass and abides by Newton's laws.

This is the Cube GameObject in Unity.
The components attached to it are:
- Transform
- Renderer



Unity XR

XR: An umbrella term for "Extended Reality" grouping together **Virtual Reality** (VR), **Mixed Reality** (MR) and **Augmented Reality** (AR).

Virtual Reality allows you to isolate the user from a real environment and immerse them in new and completely digital environment.

Unity supports all 3 with the **Unity XR plug-in framework** used for easy interfacing between device and engine.

The **Oculus Integration Pack** provided by the creators of Oculus enhances this interfacing.

This means Unity can easily communicate with the Oculus Rift CV1 device used for the project.

Scaling between Reality and Unity

Game objects in Unity experience **jitter at large distances** away from the origin due to floating point error.

If we were to keep all quantities in SI units, jittering becomes so large that behavior of the simulation can change.

Unity units	Real life unit conversion
1 Length unit	0.01AU – 1.496×10^8 m
1 Mass unit	1 earth mass - 5.972×10^{24} kg
1 Time unit	1 earth day – 86,400 seconds

Developing for VR

Existing scenes would have to be "migrated" so that Unity can output video to the **Head-Mounted-Display** (HMD).

Through the **Oculus Integration Pack**, virtual camera rigs made as pre-fabricated models (PreFabs) were used to aid this.

Also included were PreFab models that would be virtually interacted with.

This level of VR interaction can also be done from a distance, allowing the user to pull distant objects or interact with UI to edit/restart the simulation

Superposition of Forces

For each astronomical body, the net force felt is a superposition of all the gravitational forces acting on it so to create spherical Newtonian Gravity we do the following:

- Assign mass for every RigidBody.
- Use Newton's Law of Universal:
$$\vec{F} = \frac{GMm}{r^2}\vec{r}$$
- Loop through each object.
- Add the vector Forces.
- Prevent recalculating: Use Newton's Third Law.
- Use AddForce() on RigidBody to apply effect.

Implementation

Elliptical Orbits

- Use real life data from Horizons System.
- Ellipse: Type of Conic
- Conic equation [2]:
$$Ax^2 + \mathrm{B}xy + \mathrm{C}y^2 + Dx + Ey + F = 0$$
- Record Five points and solve.
- Centre of ellipse:
$$X_c = \frac{BE - 2CD}{4AC - B^2}, \qquad Y_c = \frac{BD - 2AE}{4AC - B^2}$$
- Semi major axis
$$a = \sqrt{\frac{2(AX_c^2 + CY_c^2 + BX_cY_c - 1)}{A + C + \sqrt{(A-C)^2 + B^2}}}$$
- Velocity vector tangent to ellipse.
- Find by differentiating ellipse equation.
- Use vis-viva to find magnitude.

Project 1: Binary Systems & Collision Courses

The purpose of this project was to analyse whether the Unity Engine was suitable to simulate close encounters of systems. This was done by:

- Modelling the Alpha Centauri system as a binary star system (mass of Proxima Centauri is overshadowed by its companion stars).

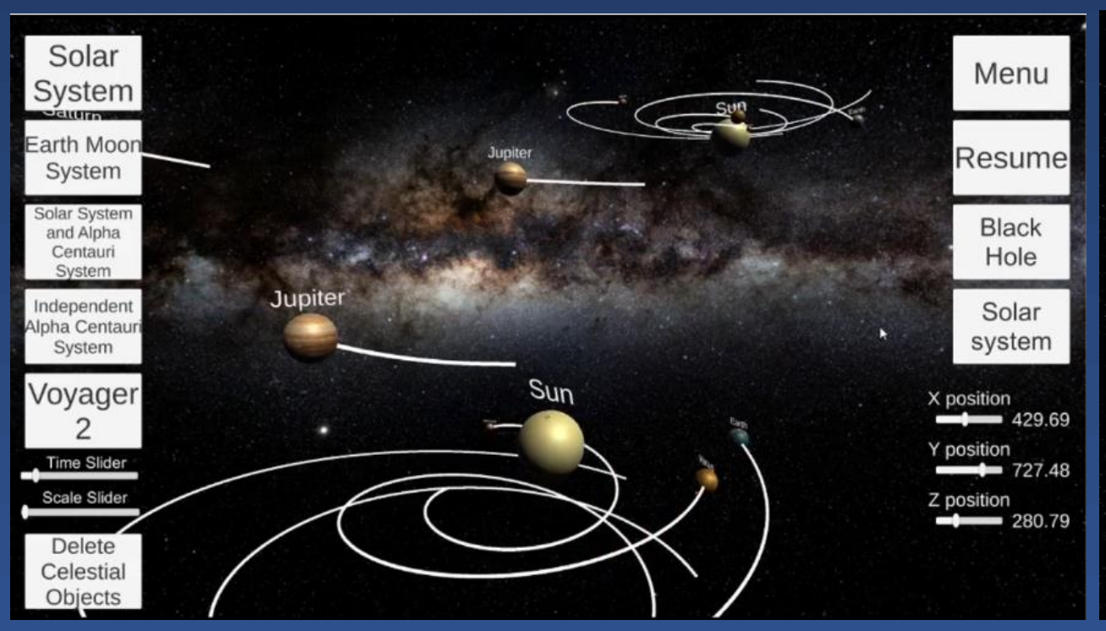
- Finding the initial position and velocity of the system through the Vis-viva equation, with centre of mass at the origin.

- Calculating a new gravitational constant to speed up the simulation (since the orbit period of the Alpha Centauri system, $\sim 79\,years$, is much larger than that of the inner planets of the Solar System)

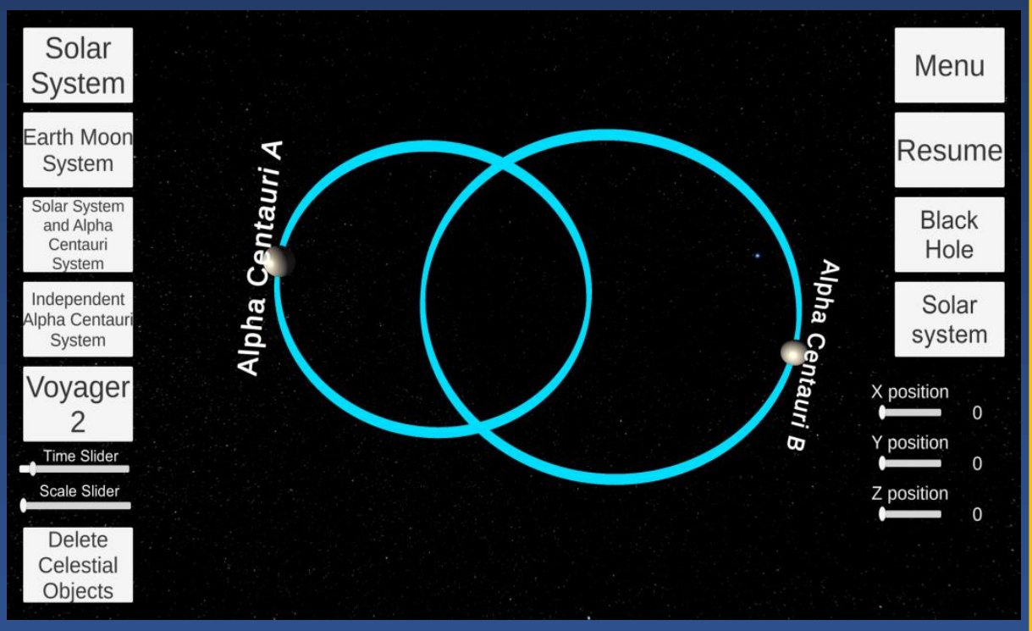
- Simulating a collision course of the binary system and Solar System, with the binary star system approaching at $3kms^{-1}$.

After the encounter of the systems only the inner planets continue to orbit the Sun.

Outer planets are pulled away by the Alpha Centauri system, since the gravitational force is inversely proportional to the distance between two bodies.



The collision between two Solar Systems in Project 1



Orbit of the binary star system and UI of Project 1

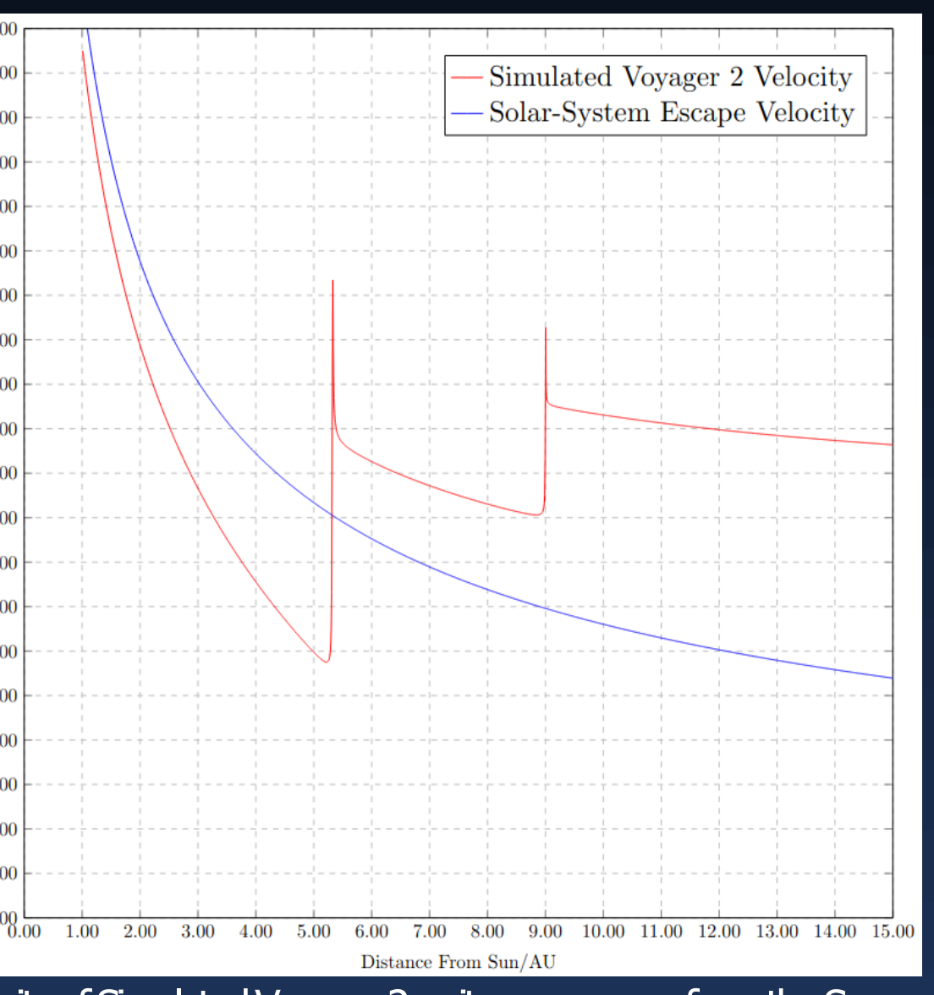
Project 2: The Grand Tour

The grand tour of the solar system is visiting all four of the outer-planets in the solar system. This simulation will focus on the trajectory of Voyager 2 which has accomplished this.

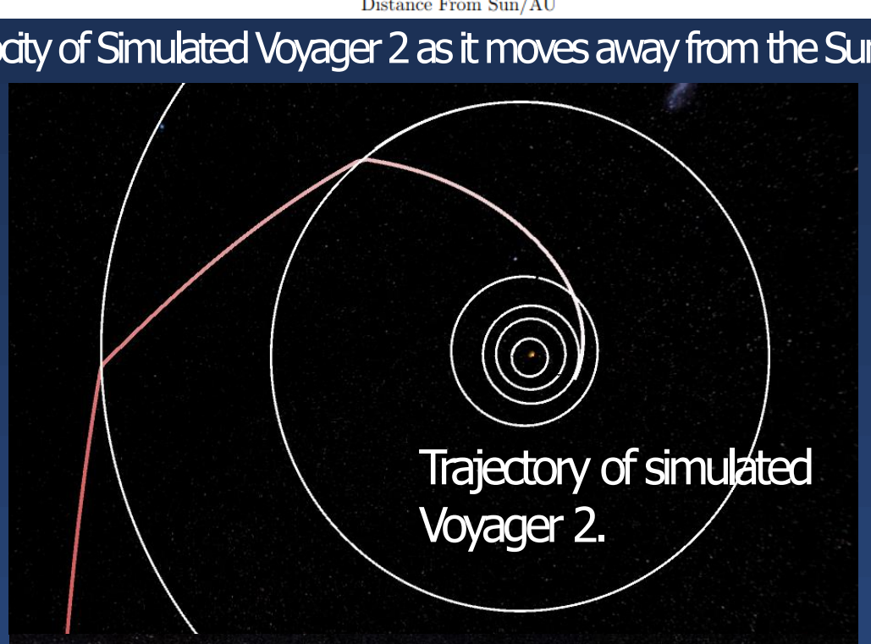
With current technology, the alignment of the outer planets that enables a grand tour, occurs once every 176 years.

This project allows the user to visualize the effects of gravity assist with a flyby of Jupiter and Saturn.

Not only can users see this from a top down 2D view, they can also experience the awe of seeing Jupiter and Saturn close from a First-Person Virtual Reality view centred on the spacecraft.



Velocity of Simulated Voyager 2 as it moves away from the Sun.

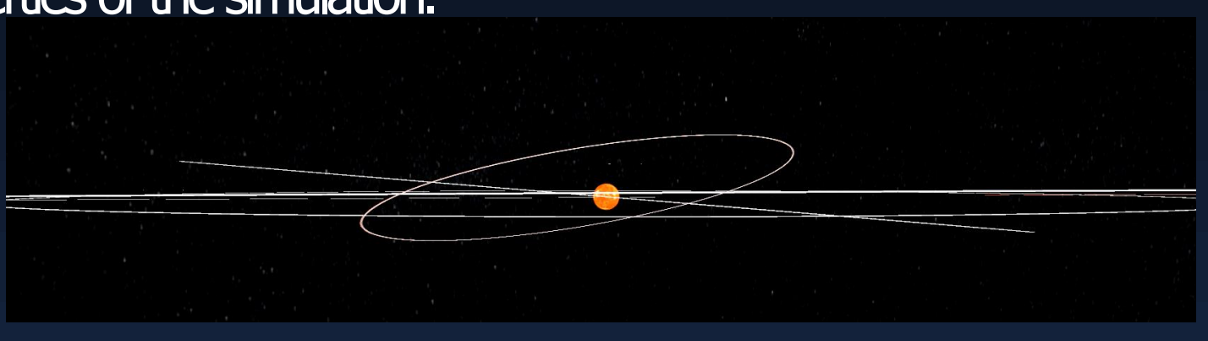


Trajectory of simulated Voyager 2.

Capture of Jupiter during flyby

Project 3: Realistic Orbits & Zero-G Sandbox

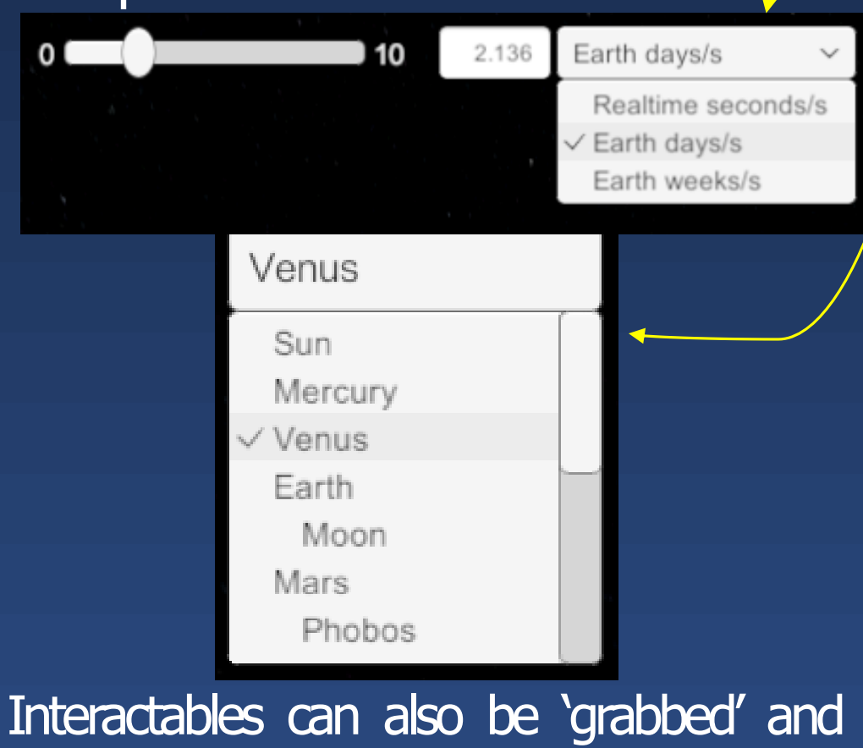
There were 2 points of focus for this project:
1) Keeping orbit representations as realistic as possible .
2) Give the user the ability to play with properties of the simulation.



Representing Orbits

Planetary data was taken from NSSDC fact sheets from, where it was then converted into custom units.

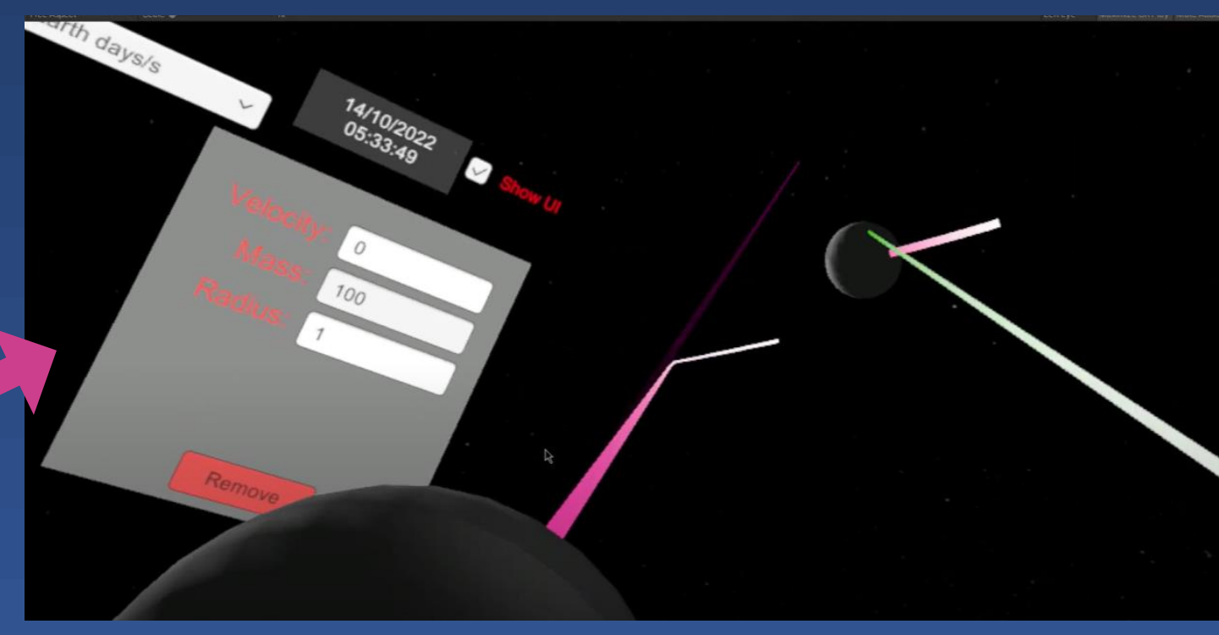
Initial conditions were set from converted data by starting positions at periapsis points, which were entered into their custom script components.



Implementing UI

UI was implemented to allow real-time editing of the simulation, some of those include:
- **Time scale controls** to slow down or speed up the simulation.
- **Celestial bodies menu** to choose which object the camera will follow.
- **Ability to add and remove celestial bodies** to simulate how the solar system would behave if new planets were introduced.
- **Velocity and position display** to let the user see how each celestial moves across the solar system.
- **Restart button** to let the user start from the beginning if something goes wrong.

Interactables can also be 'grabbed' and thrown to induce orbits with physical motion.



Additional functionality was provided through the Oculus Integration Package.

Review

Accuracy of Project 1 and the stability of the orbits were assessed by calculating percentage difference in the magnitude of position vectors obtained from the Horizons On-Line Ephemeris System (HES) [3] and the Unity Simulation ($percentage\,\Delta|\boldsymbol{r}|$), as seen by equation (1)

$$percentage\,\Delta|\boldsymbol{r}| = \left|\frac{|\boldsymbol{r}_{HES}| - |\boldsymbol{r}_{SIM}|}{|\boldsymbol{r}_{HES}|} \times 100\right|, \qquad (1)$$

where $|\boldsymbol{r}_{HES}|$ and $|\boldsymbol{r}_{SIM}|$ are the magnitudes of position vectors of a celestial body obtained from the Horizons On-Line Ephemeris System (HES) and Unity Simulation respectively.

The $percentage\,\Delta|\boldsymbol{r}|$ was found for a simulation comprising of the Sun, eight planets, and Pluto after timesteps (Δt) of 1, 10, and 100 years, see in Figure 1.

As seen from Table 1 the accuracy of the simulation is quite high, and it is suitable to accurately simulate the motion of celestial bodies under the influence of gravity.

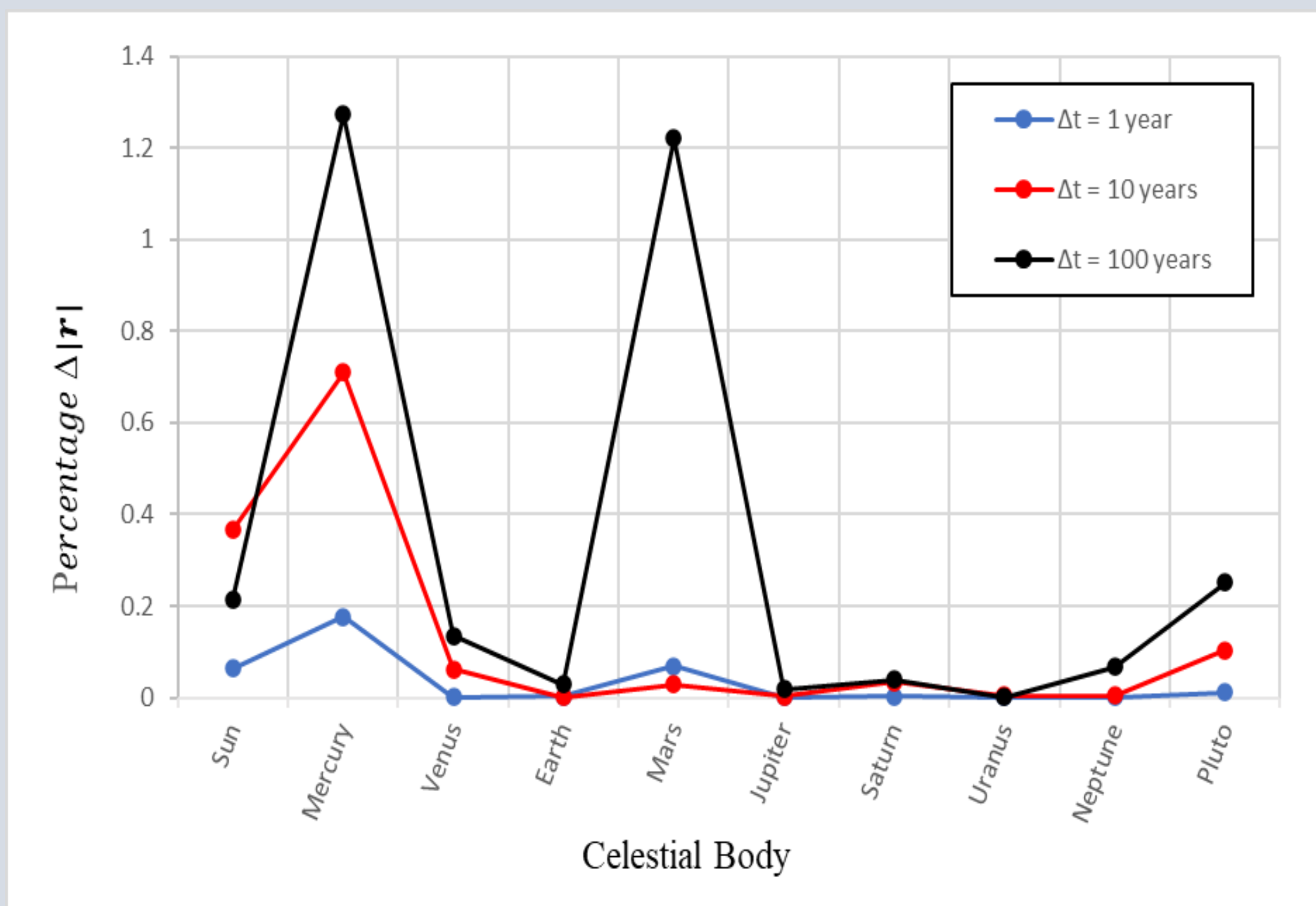


Figure 1. Percentage difference in the magnitude of the position vectors obtained from the Horizons On-Line Ephemeris System and the Unity simulation ($percentage\,\Delta|\boldsymbol{r}|$) for each Celestial Body. The blue, red, and black lines represent the percentage $\Delta|\boldsymbol{r}|$ after certain timesteps (Δt) in the simulation where 1 year represents 365 days. The error bars are too insignificant to be seen.

Table 1: Mean $percentage\,\Delta|\boldsymbol{r}|$ of the Eight Planets, Sun, and Pluto after certain timesteps in the simulation.

Timesteps (Δt) in simulation where 1 year = 365 days	Mean $percentage\,\Delta|\boldsymbol{r}|$ ($\times 10^{-1}$)
1 year	0.331 ± 0.007
10 years	1.132 ± 0.006
100 years	3.25 ± 0.01

- It is clear as the time running the simulation increases the uncertainty also increases

- This is due to floating point numbers having limitations in their range and decimal values, limiting the precision of the number

- Thus, if a value from a calculation has been rounded and used in further calculations it will result in a cascading effect of accumulating a larger uncertainty.

- To mitigate floating point errors in the future, a method can be used to find the upper and lower bound on the rounding errors, this would give a more accurate representation of the real uncertainty due to the errors.

Conclusion

Where the aims of the project met?

- Objective 1: Create Scale Model of Solar System ✓
- Objective 2: Simulate Newtonian Gravity ✓
- Objective 3: Implement a VR experience ✓

Can features be improved?

- Accuracy could be further improved with better hardware.
- Stability can be improved with optimised code and assets.

References

[1] www.fxsolver.com. (n.d.). *Vis-Viva Equation, and How To Go To Space - fxSolver*. [online] Available at: https://www.fxsolver.com/blog/2015/11/25/vis-viva-equation-and-how-go-space/ [Accessed 23 Apr. 2022].

[2] Mathews, J.H. (1995). The Five Point Conic Section: Exploration with Computer Software. *School Science and Mathematics*, 95(4), pp.206–208.

[3] D.Giorgini, J., 2022. *Horizons System*. [online] Ssd.jpl.nasa.gov. Available at: <https://ssd.jpl.nasa.gov/horizons/app.html#/> [Accessed 21 April 2022].