Orbital Mechanics Simulator

Lucius Kwok (VC1B)
<lk@felttip.com>
Supervisor: Brian Papa
<bpapa@icloud.com>

Project Purpose

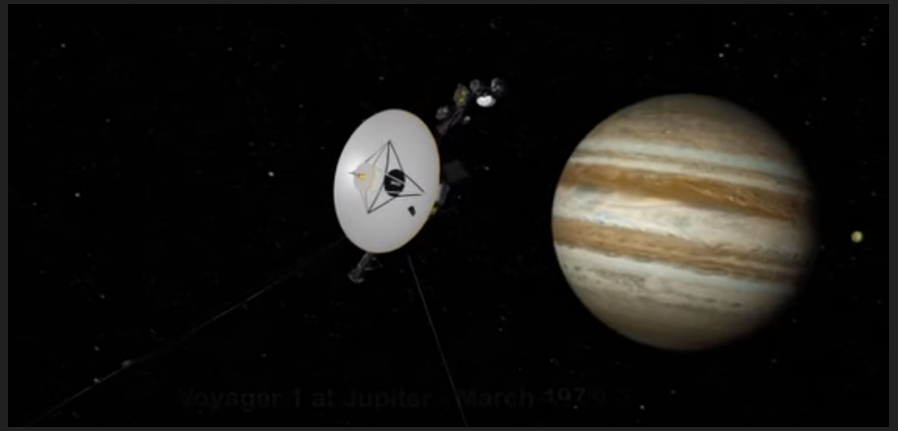
This 3D spaceflight simulator will be part of a web page where users will learn about the orbital mechanics topic within the topic of spaceflight. It will allow students and people without a background in physics to understand the physics required to go from an Earth orbit to a Moon orbit. It allows users to plan out and execute a series of velocity change maneuvers to go from an Earth orbit to a Moon orbit based on an approximation of realistic orbits of celestial bodies called patched conics, which uses multiple spheres of influence to represent discrete gravity wells for each body. It will also present some historical context in the form of the Apollo Moon missions, the Voyager missions, and more recent robotic missions to the Moon.

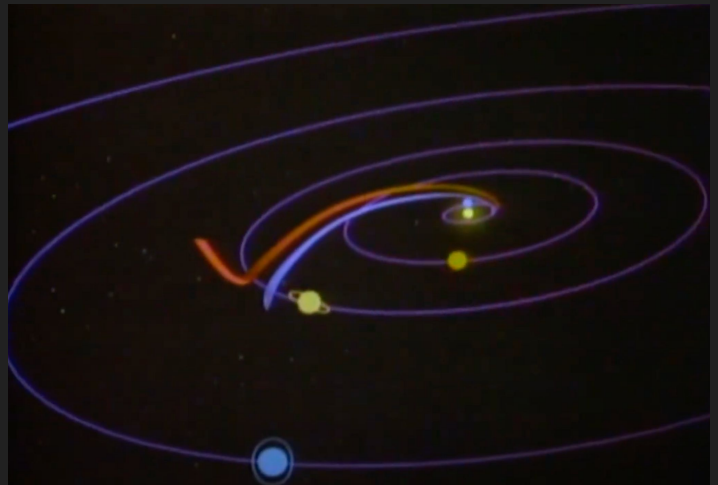
Target Audience

This experience is targeted at users who are:

- College-level students interested in intro-level physics and astronomy, but not necessarily having a physics background
- Casual gamers who enjoy playing spaceflight simulators such as Kerbal Space Program
- Learners who watch documentaries about historic spaceflight missions to the Moon and beyond, such as those about Apollo or Voyager missions

Concept Art: Voyager missions



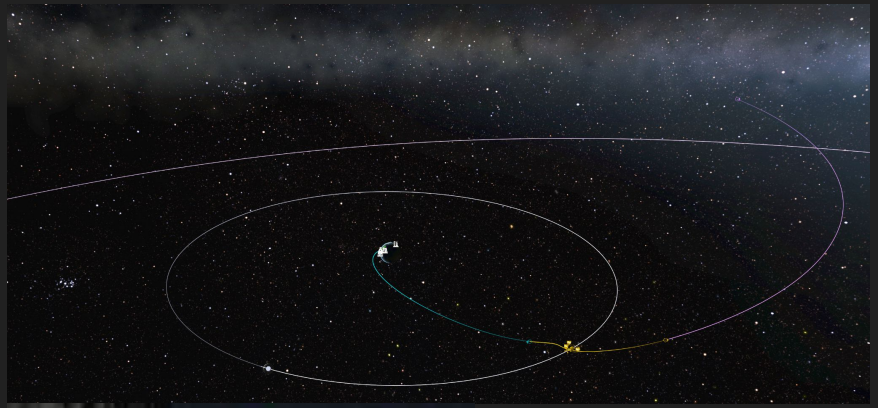


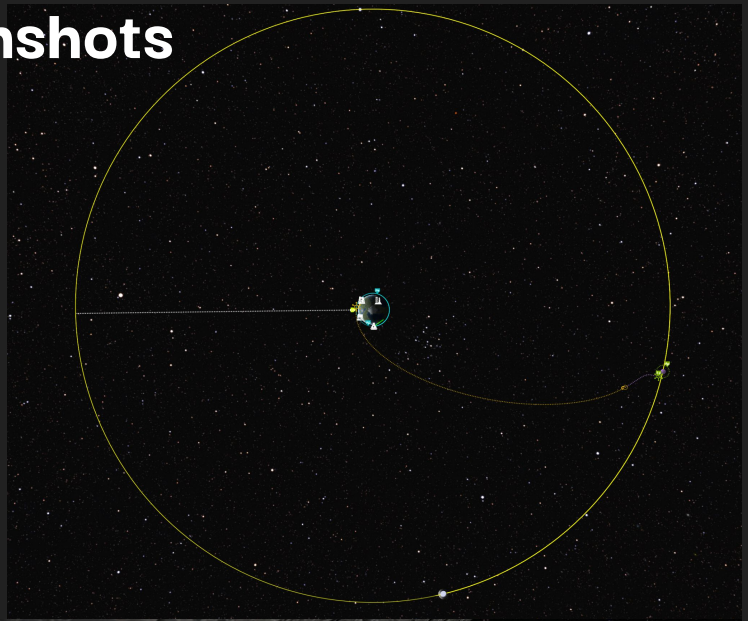
Reference Example:

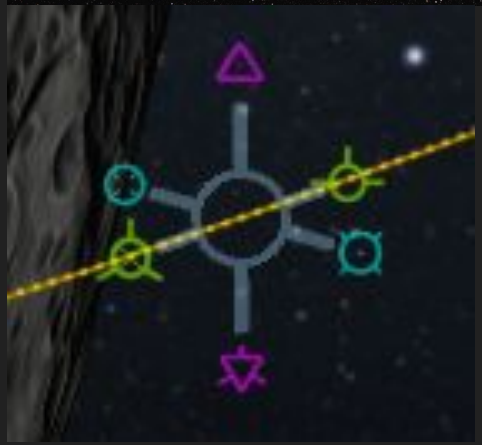
KSP (version 1):

- Started in 2010 as a project by one person, HarvesteR (Felipe Falanghe)
- Uses Unity engine
- Continuously developed by a team until about 2021, to focus on KSP 2
- Changes in ownership and direction led to failure of KSP 2 in the market and closure of the game studio that was developing it
- Vacuum left by implosion of KSP 2 means there is a market for a new spaceflight simulator to take its place.

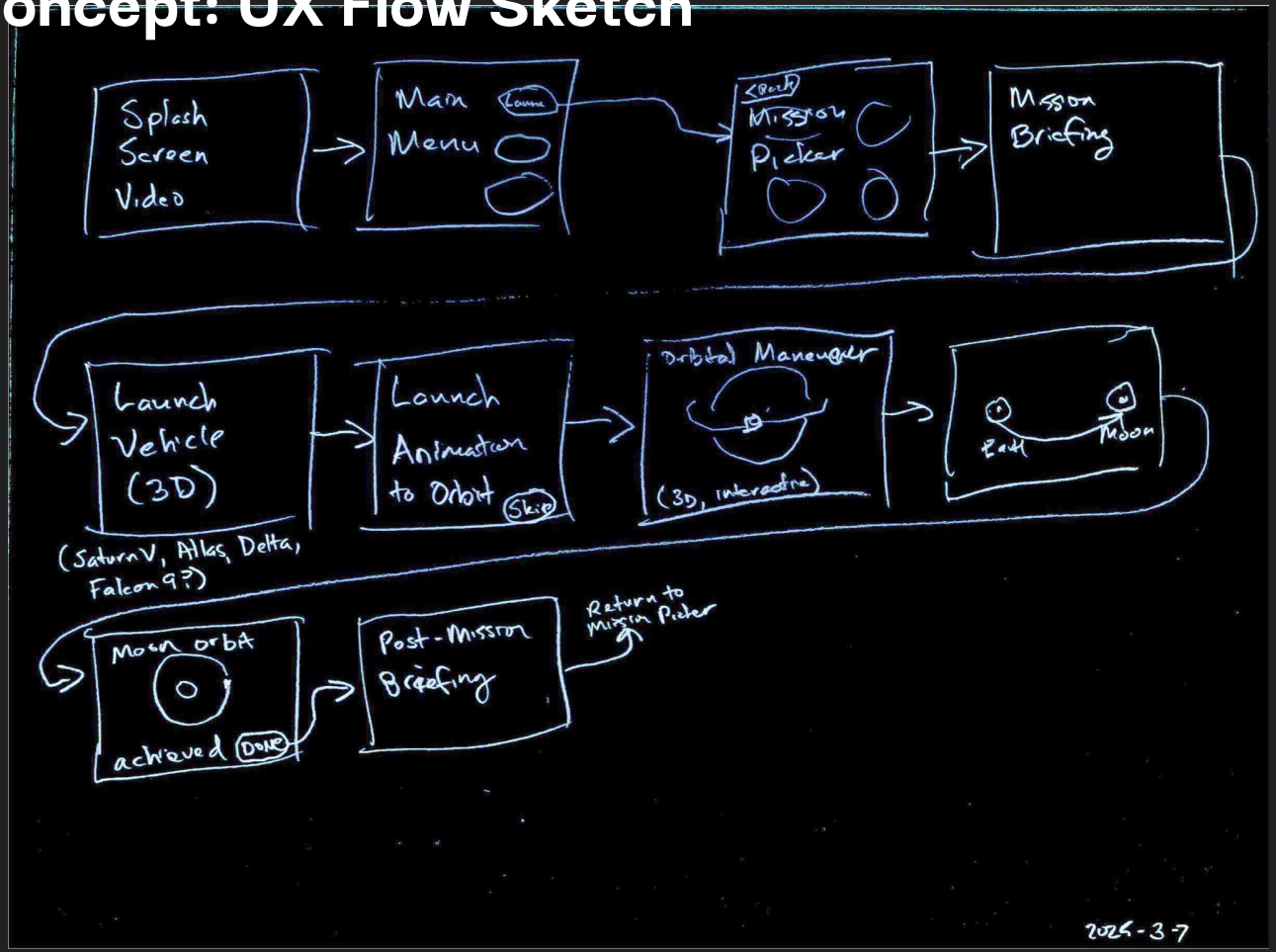
Reference Example: KSP Screenshots



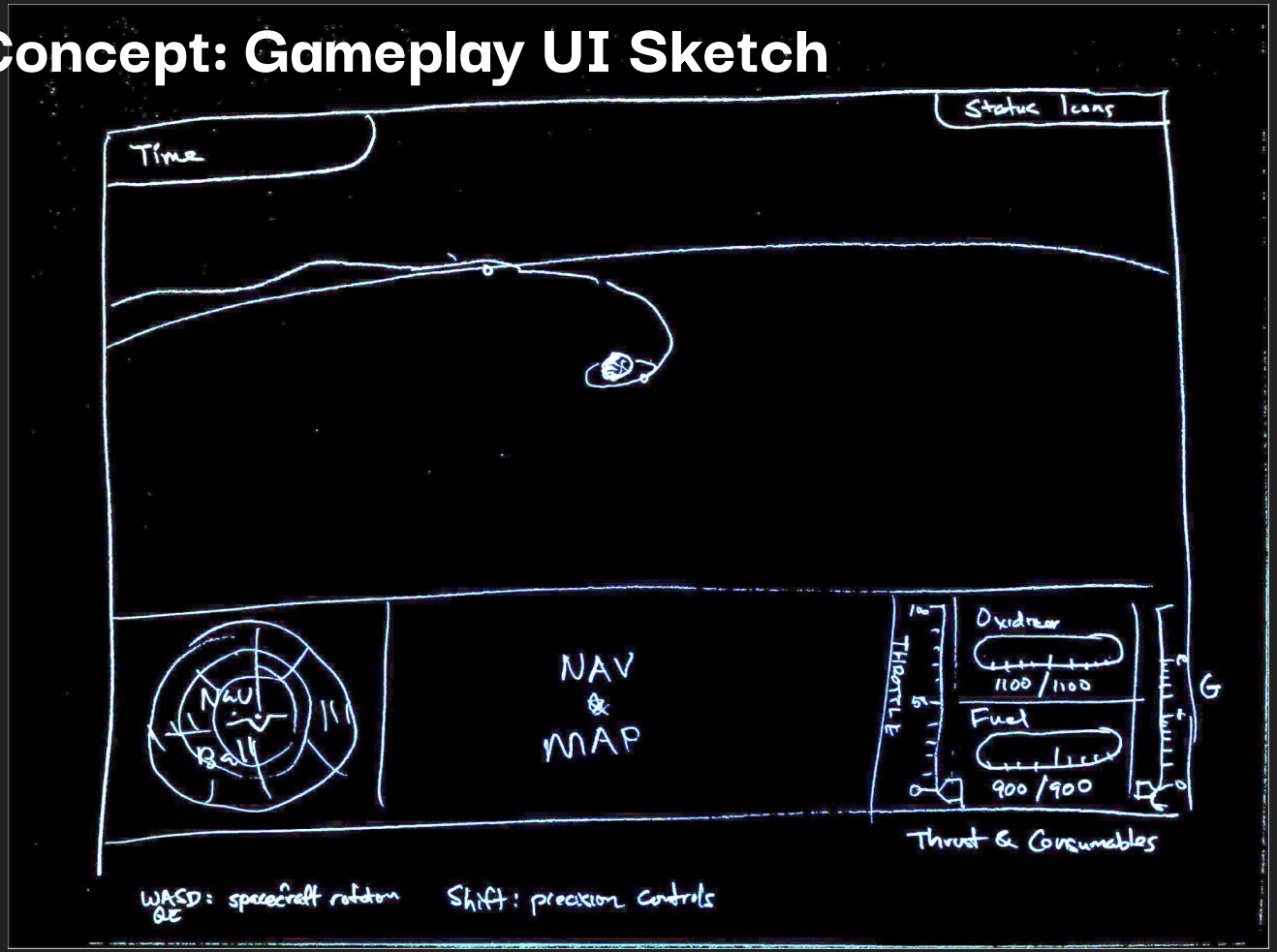




Initial Concept: UX Flow Sketch



Initial Concept: Gameplay UI Sketch

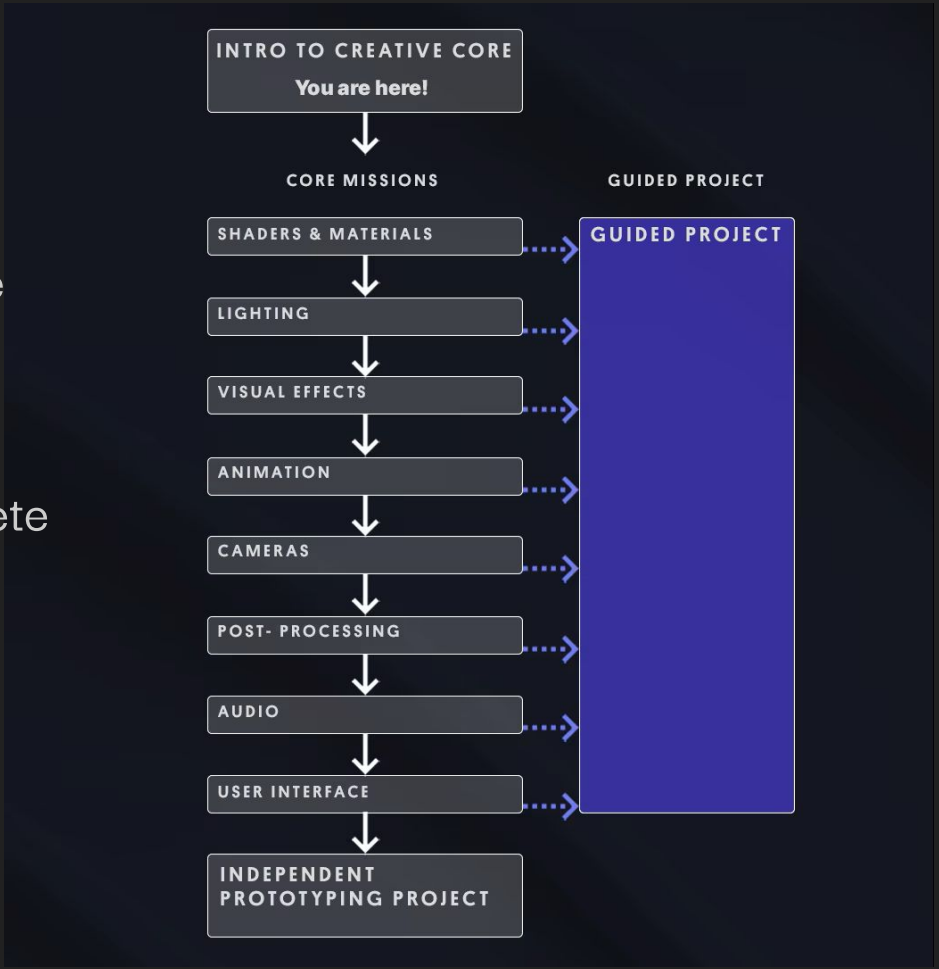


Challenges & Roadblocks: Scope

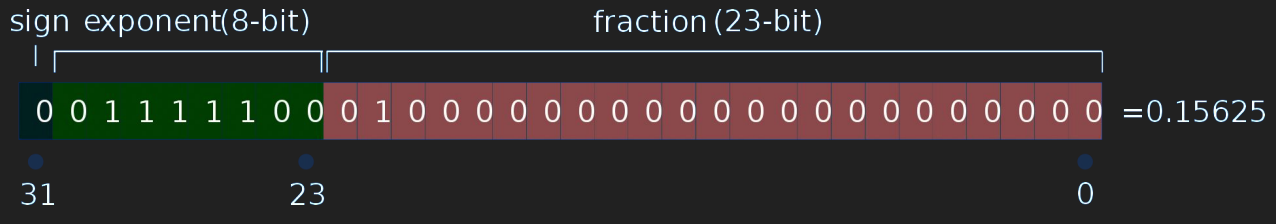
- KSP was developed over 10 years, I have only 15 weeks.
- I have experience only with making productivity apps, not with making games or using Unity engine.
- Game vs. Simulation: I want to lean towards a simulation with realistic physics, but still making an engaging and enjoyable experience.
- Technical challenges such as 32-bit float precision and 3D calculations complicate the project.

Challenges & Roadblocks: Unity

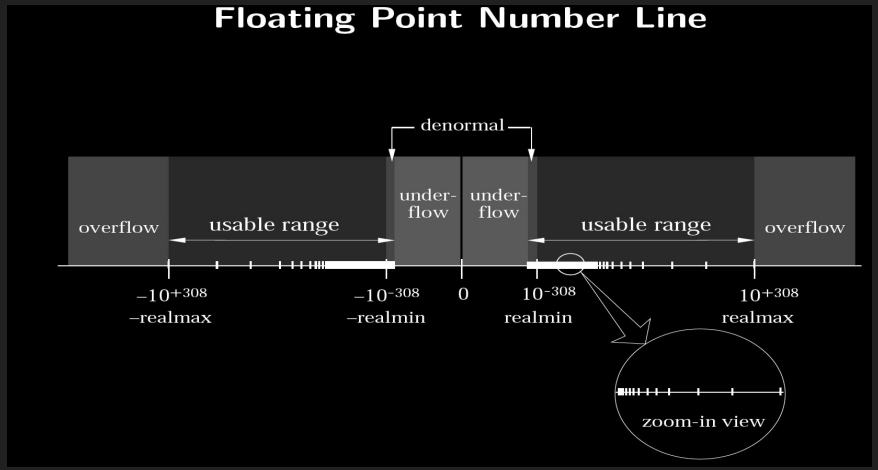
- Unity is a very complex game engine with a steep learning curve.
- I have almost no experience in it.
- Spent 3 weeks / 50 hours to complete Creative Core pathway tutorials.
- Will need to spend as much time to complete the Junior Programmer pathway tutorials.



Technical Challenge: 32-bit Floating Point Accuracy



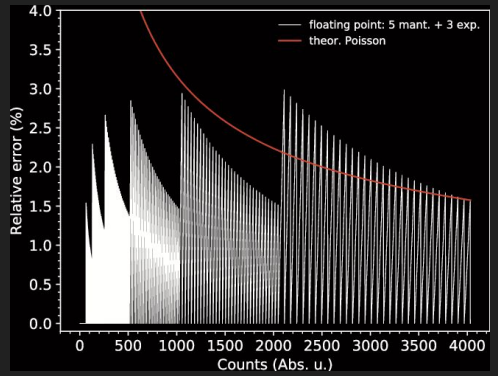
There is problem where the precision of a floating-point number decreases as the value is farther from zero. When representing a location in space, the positions of points far away from the origin of the world become less precise.



Source: https://courses.grainger.illinois.edu/cs357/fa2023/notes/ref-4-fp.html

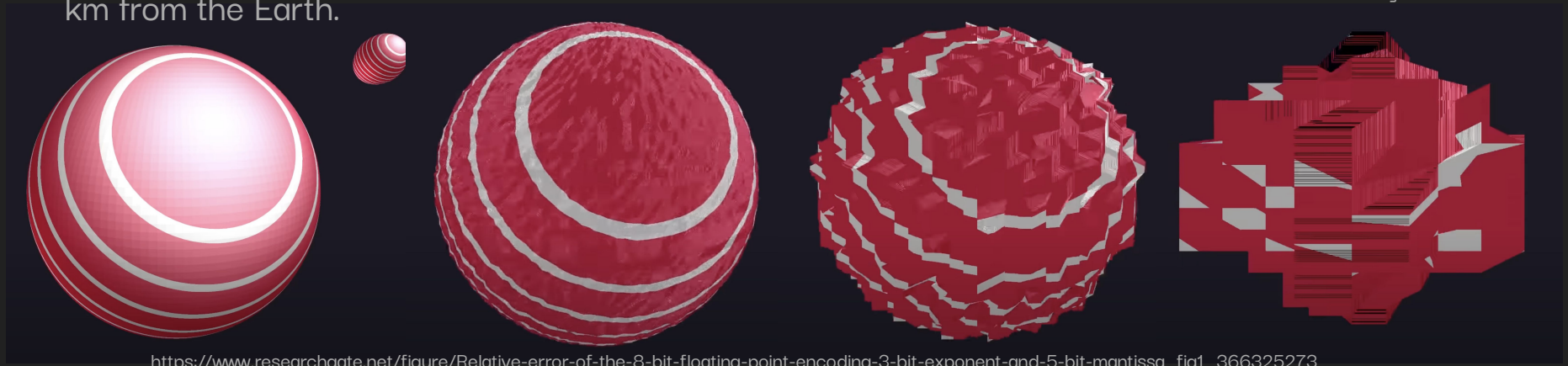
Technical Challenge: 32-bit Floating Point Accuracy

As objects move farther away from the origin of worldspace, their geometries develop glitches due to precision errors. For 32-bit floats, at a distance of 10km, the accuracy is only 1 to 10 meters. For reference, the Earth's diameter is 12,756 km, and the Moon is 382,500 km from the Earth.



Left: Rounding error in 8-bit floating point representation

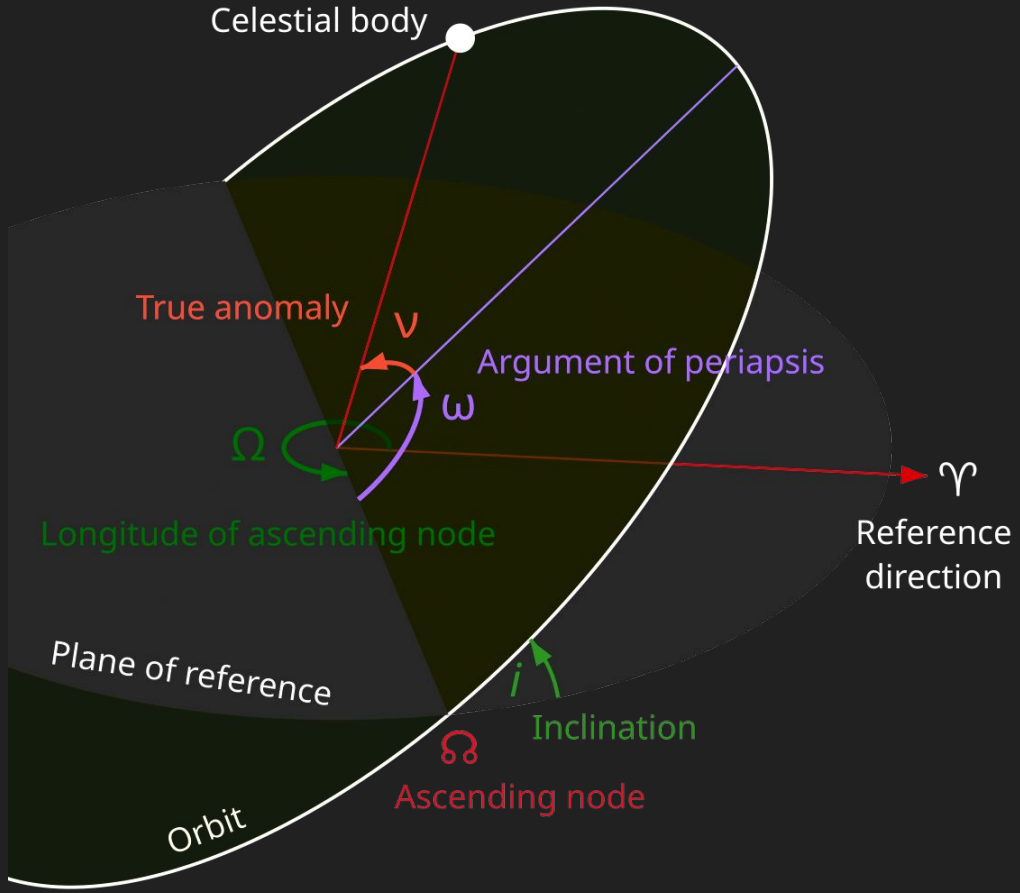
Below: Effects on 3D geometry as a sphere moves away from the origin.



https://www.researchgate.net/figure/Relative-error-of-the-8-bit-floating-point-encoding-3-bit-exponent-and-5-bit-mantissa_fig1_366325273

https://www.youtube.com/watch?v=wGhBjMcY2YQ

Technical Challenge: Orbital Elements in 3D Space



Orbital elements are parameters that define an orbit in 3D space.

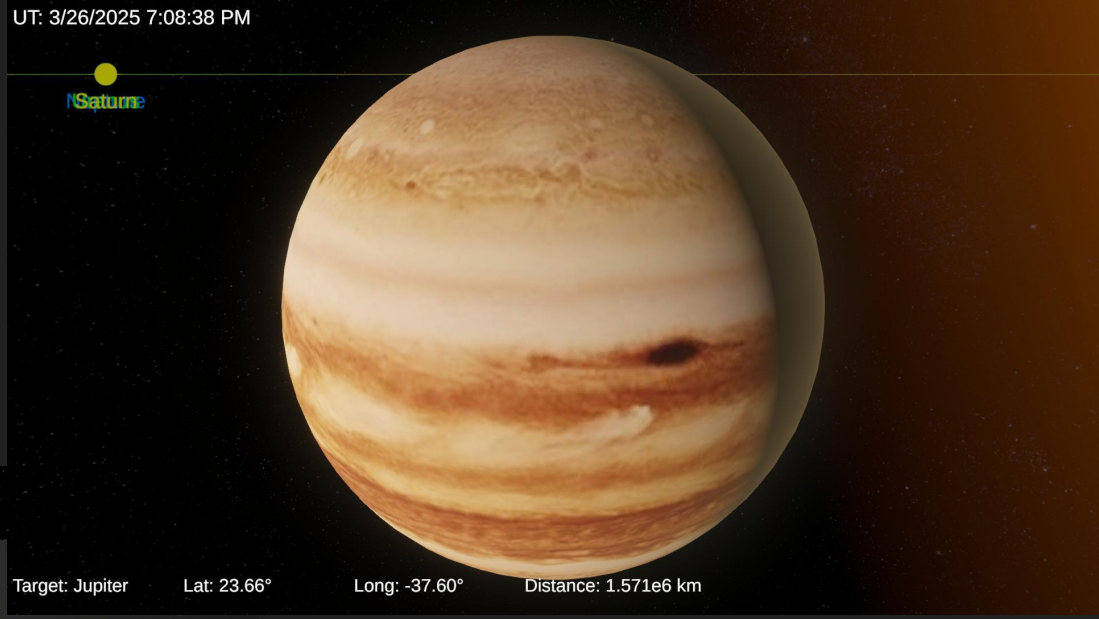
- Eccentricity (*e*)
- Semi-major axis (a)
- Inclination (*i*)
- Longitude of the ascending node (Ω)
- Argument of periapsis (ω)
- Orbital period (*P*)

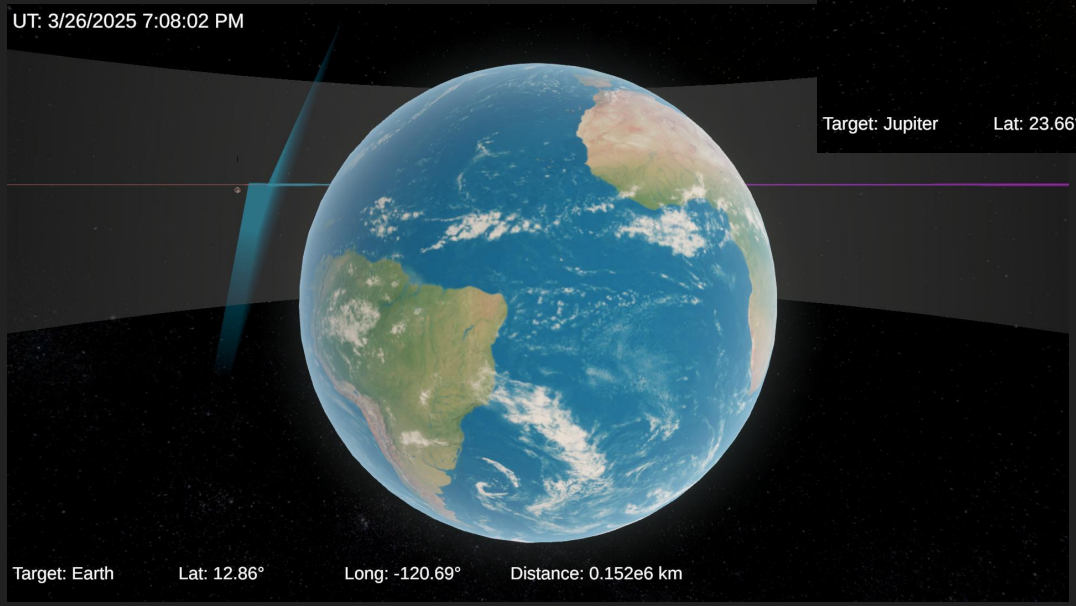
Understanding and applying physics in a 3D space is much more complicated than in 2D.

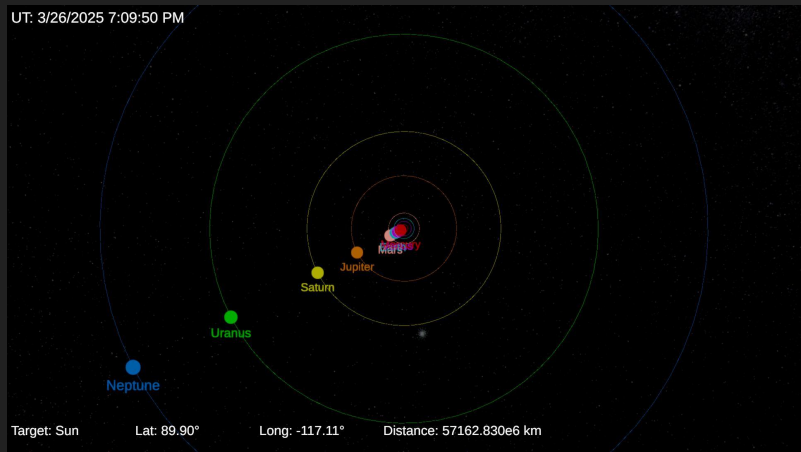
https://en.wikipedia.org/wiki/Orbital_elements

Current Project Status

- Running in Unity
- Displays planet shaders & orbit lines
- Camera control using mouse
- Switch between target planets







Next Steps: Overcoming Challenges

- Scope: will need to focus on one core gameplay element at first (orbital mechanics and maneuver nodes).
- Realism: accurate simulation will be the goal, but some things may need to be simplified.
- Technical challenge: 32-bit float. Several solutions are possible, including a floating origin and multiple rendering contexts.
- Technical challenge: 3D vs 2D. May need to constrain orbits to 2D plane that exist in a 3D space.

Next Steps: Schedule

Week 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project planning & research. Set up project repo & issue tracker.															
		Identify users, use cases, and proposed solutions. Sketch UI/UX.													
				Develop prototype/demo. User testing, presentation.					Further testing			Final testing			
					Refine and build out UI and features based on user and presentation feedback.										
								Triage features and issues to decide on MVP							
											Feature lock & focus on fixing issues.				
													Prepare final presentation & submission		

Endnotes

GitHub repository & project tracker:
https://github.com/luciuskwok/CISC-4900-Unity
https://github.com/users/luciuskwok/projects/2

Data sources: Kerbal Space Program, NASA, Scott Manley, Unity Learn, University of Illinois, University of Texas, Wikipedia.

Tools: Unity, Visual Studio, Maya, Photoshop, Panic Nova, GitHub, and Google suite. Developed on a Windows PC with a Core i7-11700 CPU & a NVIDIA GTX 1080 Ti GPU.