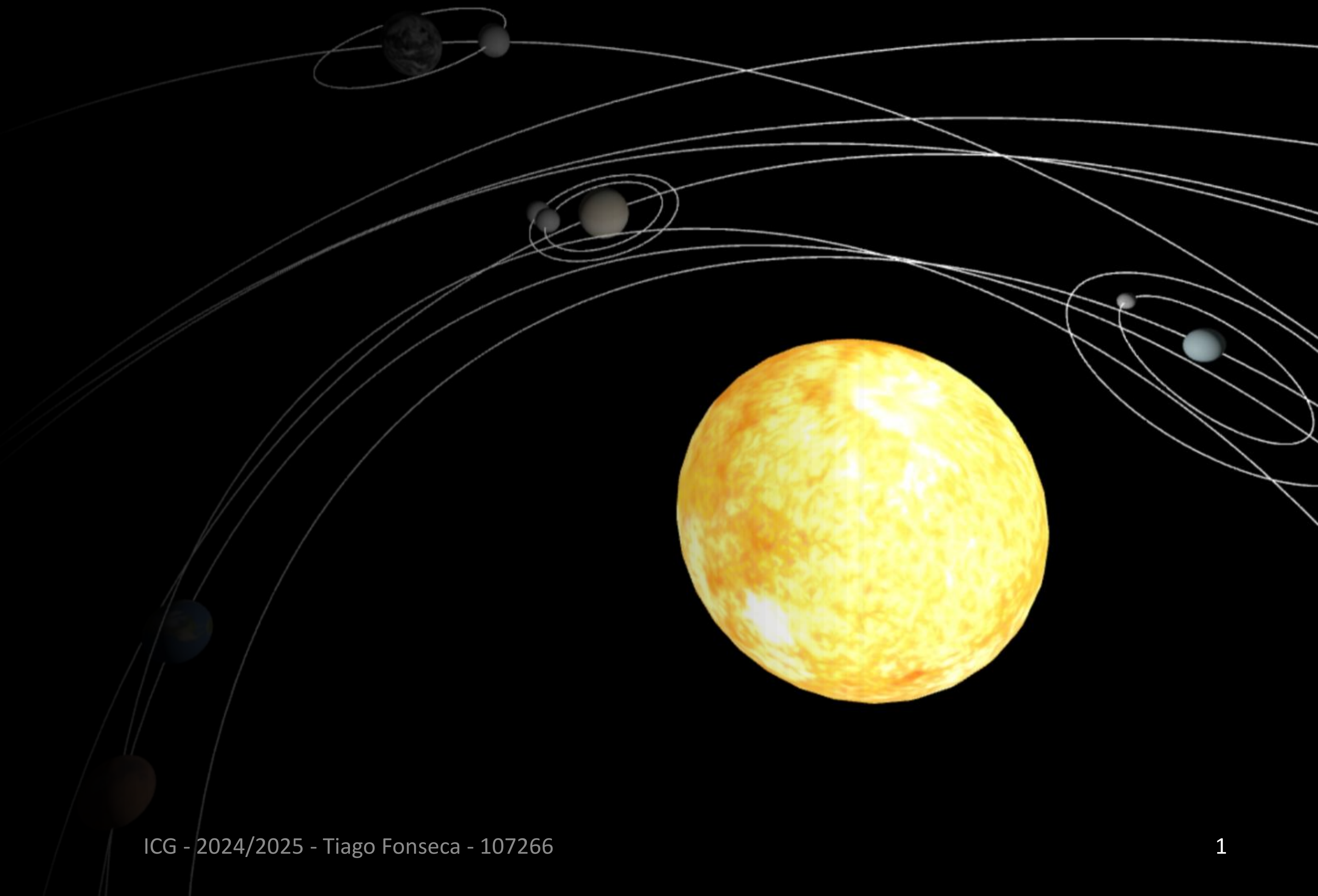
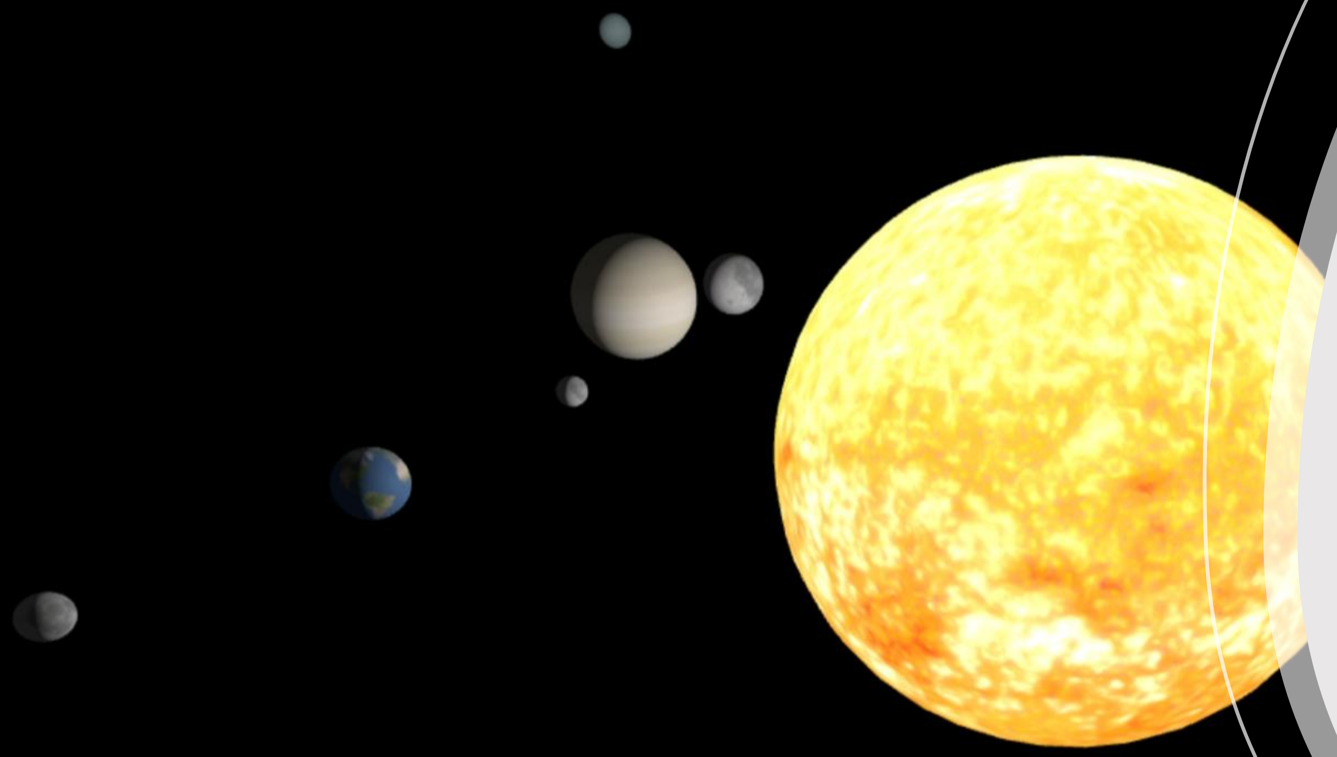
# Solar Systems





# Main ideas

The project focuses on exploring various solar systems and being able to create their own.

What can the user do?

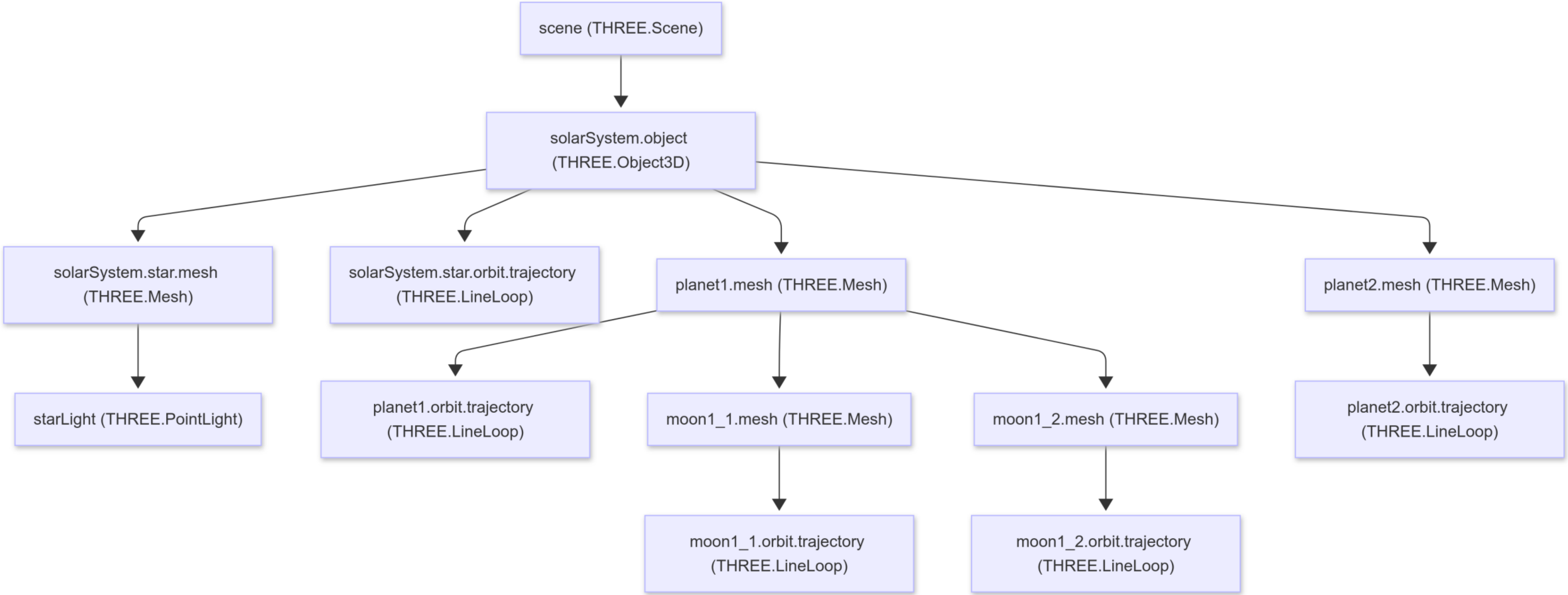
- Navigate through the solar system;
- Visualize the Celestial Bodies;
- Download and Upload Solar Systems;
- Toggle the orbits mesh;
- Track his position using the mini map.

Available at [https://www.solarsytems.pt/](https://www.solarsytems.pt/)

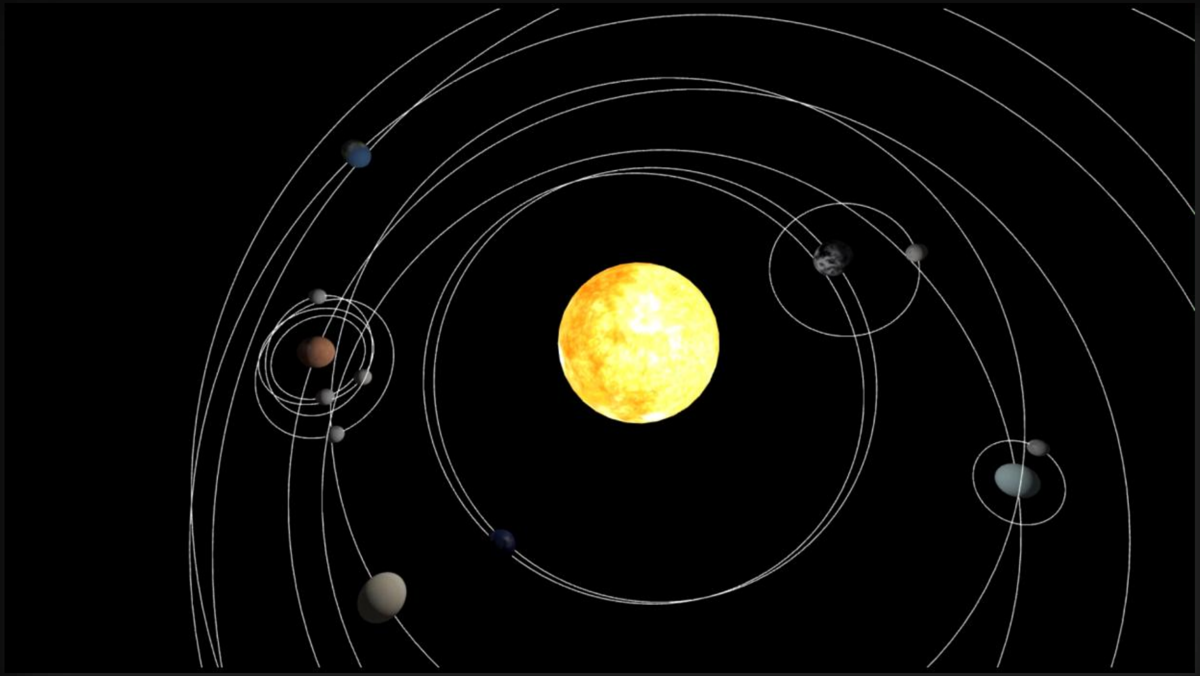
# Models and the scene graph

- **SphereGeometry**: Used to create the spherical shape for planets, moons, and stars.
- **Object3D**: Used as a general-purpose object to group other objects, specifically for the Orbit class to hold the trajectory and for the Solar System to group all its celestial bodies.
- **BufferGeometry**: Used in the Orbit class to create the trajectory line from a set of points.
- **LineLoop**: Used in the Orbit class to draw the orbital path as a continuous loop.

# Scene Graph


scene (THREE.Scene)
solarSystem.object (THREE.Object3D)
solarSystem.star.mesh (THREE.Mesh)
solarSystem.star.orbit.trajectory (THREE.LineLoop)
planet1.mesh (THREE.Mesh)
planet2.mesh (THREE.Mesh)
starLight (THREE.PointLight)
planet1.orbit.trajectory (THREE.LineLoop)
moon1_1.mesh (THREE.Mesh)
moon1_2.mesh (THREE.Mesh)
planet2.orbit.trajectory (THREE.LineLoop)
moon1_1.orbit.trajectory (THREE.LineLoop)
moon1_2.orbit.trajectory (THREE.LineLoop)

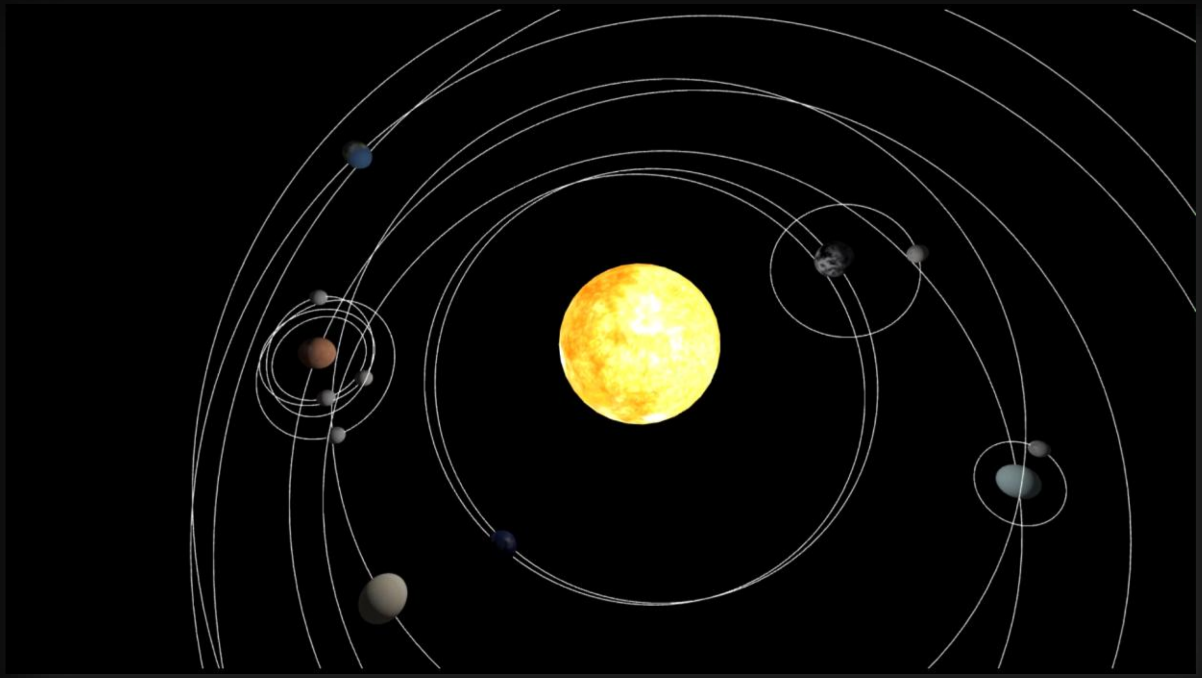
# Animation



## Celestial Body Movement (Rotation and Orbit)

- Rotation: Each planet and moon spins on its axis by updating its rotation in the Y axis based on a calculate delta time (time since last frame) and its rotation speed
- Orbit: Planets and moons orbit their respective centers (star for planets, planet for moons) by calculating a new elliptical position in each frame. This involves incrementing an orbit angle and applying the orbit's radius, eccentricity and inclination to update its position

# Animation



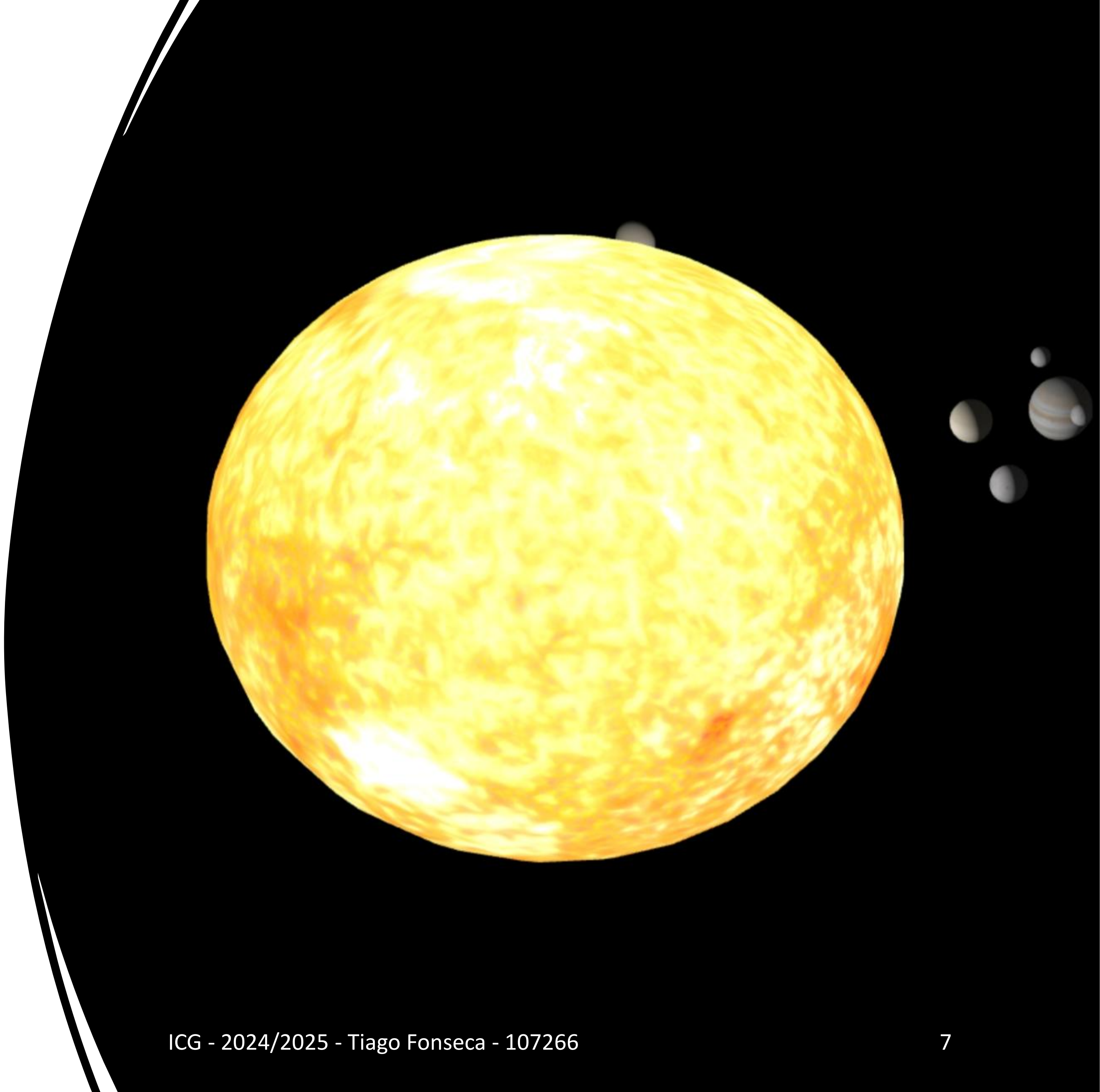
## Camera Movement and Mini Map Updates

- Player Movement: The main camera moves based on user input (WASD for horizontal, Space/Shift for vertical).
- Minimap Camera Tracking: A separate mini map camera follows the main camera's horizontal position and always looks straight down.
- Minimap Camera Orientation
- Minimap Zoom: Zoom level of the mini map can be smoothly adjusted with the mouse wheel.
- Planet Info Cards: HTML cards for planets/moons appear only when within view and close enough. Their positions are updated in real-time using 3D-to-2D projection to stay aligned with their objects.

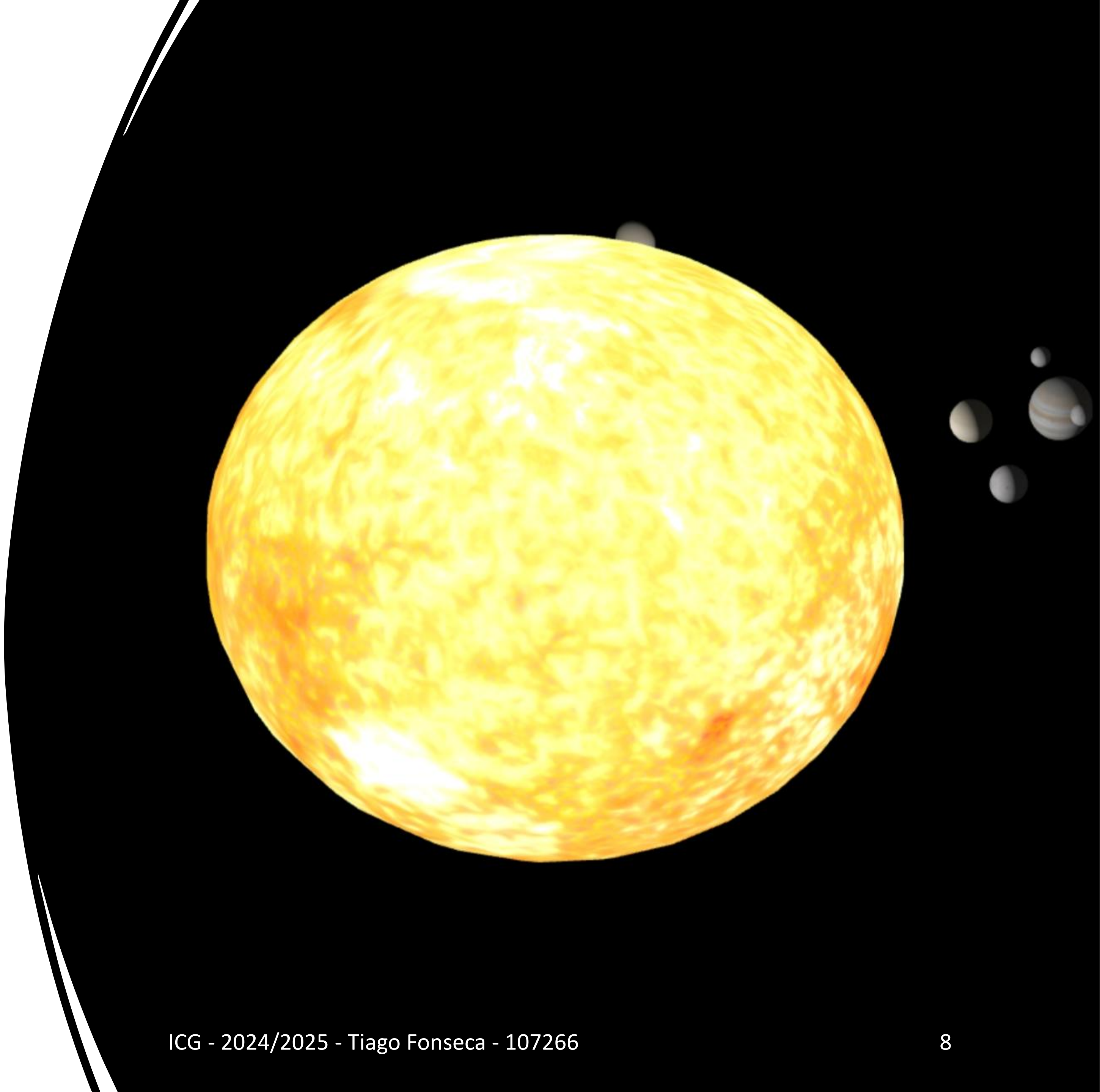
# Illumination

### Star Light (THREE.PointLight)

- Main Light Source - A PointLight simulates the star (e.g., Sun), illuminating planets and moons with clear light and shadow effects.
- Light Properties - Uses white light (0xFFFFFF) with high intensity (100) for strong illumination.
- Dynamic Position - The light's position updates to match the star mesh, ensuring accurate lighting origin.
- Shadow Casting - Enables shadows from planets and moons; uses a 2048x2048 shadow map and adjusts shadow camera range based on the star's radius.
- Star Appearance - The star mesh uses Phong material with emissive properties and texture to appear self-illuminated and glowing.



# Illumination

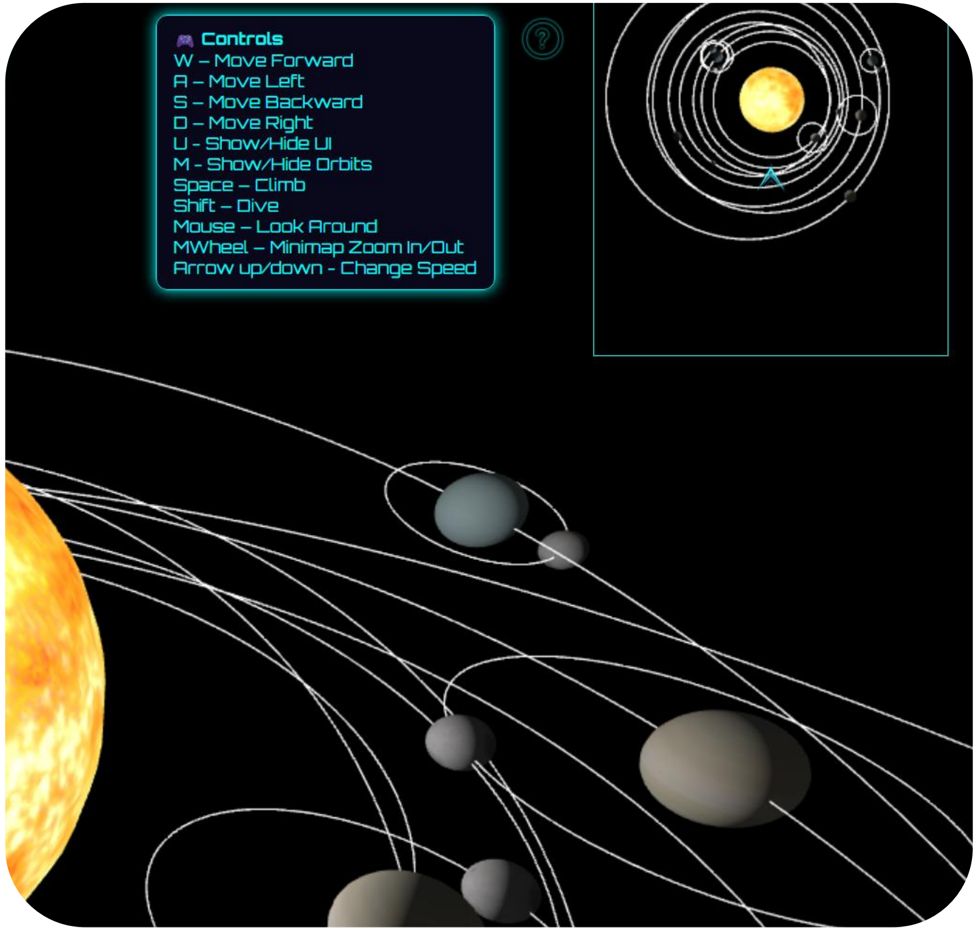


### Ambient Light (THREE.AmbientLight)

- Purpose - Provides soft, uniform fill lighting across the entire scene.
- Light Properties - Uses a dark grey color (0x333333) with an intensity of 1.5 to prevent completely black shadows.
- Effect - Adds subtle illumination to shadowed sides of planets and moons, maintaining visual detail even without direct star light.

# User Interaction

| Action | Control |
|---|---|
| Move Forward | **W** |
| Move Backward | **S** |
| Move Left | **A** |
| Move Right | **D** |
| Show/Hide UI | **U** |
| Show/Hide Orbits | **M** |
| Move Up | **Space** |
| Move Down | **Shift** |
| Rotate Camera | **Mouse Drag** |
| Zoom In/Out Minimap | **Scroll Wheel Up/Down** |
| Increase/Decrease Orbit Speed | **Up/Down Arrow** |



🎮 **Controls**
W – Move Forward
A – Move Left
S – Move Backward
D – Move Right
U - Show/Hide UI
M - Show/Hide Orbits
Space – Climb
Shift – Dive
Mouse – Look Around
MWheel – Minimap Zoom In/Out
Arrow up/down - Change Speed

# Development

## Code organization & Implementation

Approach - Follows an object-oriented programming (OOP) structure to encapsulate logic for celestial bodies, movement, and rendering.

Structure - Code is modularized across multiple files for clarity and separation of concerns (e.g., planets, stars, utilities).

CI/CD - A DevOps pipeline is in place for continuous integration and deployment, streamlining updates and testing.

## Challenges Faced

Eclipse Simulation - Accurately simulating eclipses (e.g., alignment and shadow casting) presented technical difficulties, particularly in ensuring realistic light/shadow behavior.

All the goals were achieved except for the eclipses…