CS3210 Final - Solar Simulator

User Manual

December 12, 2024

Contents

1 Introduction **1**

2 Background Information **1**
2.1 Key Concepts . . . 1

3 Installation **1**
3.1 Prerequisites . . . 1
3.2 Git LFS Setup . . . 2
3.3 Cloning the Repository . . . 2
3.4 Installing Dependencies . . . 2
3.5 Running the Simulator . . . 3

4 Getting Started **3**
4.1 User Interface Overview . . . 3

5 Using the Simulator **3**
5.1 Navigating the Solar System . . . 3
5.2 Interacting with Planets . . . 4
5.3 Adjusting Time Speed . . . 6
5.4 Controlling Orbits . . . 6

6 Technical Challenges and Design Decisions **7**
6.1 Accurate Orbital Mechanics . . . 7
6.1.1 Design Decision . . . 8
6.2 Performance Optimization . . . 8
6.2.1 Design Decision . . . 8
6.3 Visual Realism . . . 8
6.3.1 Design Decision . . . 8
6.4 User Interface Design . . . 8
6.4.1 Design Decision . . . 9
6.5 Time Manipulation . . . 9
6.5.1 Design Decision . . . 9

7 Features Overview **9**
7.1 Realistic Planet Textures . . . 9
7.2 Accurate Orbits . . . 9
7.3 Interactive Controls . . . 9
7.4 Informative Panels . . . 9
7.5 Visual Effects . . . 9
7.6 Responsive Design . . . 10

8 Troubleshooting **10**
8.1 Planets Appear Without Textures . . . 10
8.2 Performance Issues . . . 10
8.3 Controls Not Responding . . . 10
8.4 Orbit Paths Not Visible . . . 11

9 Credits and Resources **11**

9.1 Textures 11
9.2 Libraries and Tools 11
9.3 Further Reading 11

10 Conclusion **11**

1 Introduction

Welcome to the **Solar System Simulator**, an interactive 3D visualization of our solar system built with Three.js. This user manual will guide you through installing, using, and understanding the features of the simulator. Whether you're an astronomy enthusiast or just curious about our cosmic neighborhood, this simulator is designed to provide an immersive educational experience.

2 Background Information

Understanding the vastness and complexity of our solar system can be challenging. This simulator aims to bridge that gap by visualizing the positions, movements, and characteristics of the planets in an interactive 3D environment.

2.1 Key Concepts

- **Orbital Mechanics**: The simulator uses Keplerian orbital elements to calculate the positions of planets over time, providing an accurate representation of their movements.
- **Axial Tilt**: Each planet's rotation axis is tilted relative to its orbital plane, affecting the simulation of seasons and day/night cycles.
- **Astronomical Units (AU)**: Distances in space are immense, so the simulator scales down measurements using AU, where 1 AU is the average distance from the Earth to the Sun (149.6 million kilometers).

3 Installation

3.1 Prerequisites

Before installing the Solar System Simulator, ensure that you have the following software installed on your system:

- **Node.js** (latest LTS version)
- **npm** (comes with Node.js)
- **Git LFS** (Git Large File Storage)
- **Modern Web Browser** (with WebGL 2.0 support, e.g., Chrome, Firefox)

3.2 Git LFS Setup

This project uses Git LFS to manage large texture files. Follow these steps to set up Git LFS:

1. **Install Git LFS**:
   - **Windows**:
     ```
     winget install Git.LFS
     ```
   - **macOS**:
     ```
     brew install git-lfs
     ```
   - **Linux**:
     ```
     sudo apt install git-lfs
     ```
2. **Initialize Git LFS**:
   ```
   git lfs install
   ```

3.3 Cloning the Repository

Open your terminal and execute the following commands:

1. **Clone the repository**:
   ```
   git clone https://github.com/justinfeatherstone/
   featherstoneplesciafarhat_3210_final.git
   ```
2. **Navigate to the project directory**:
   ```
   cd featherstoneplesciafarhat_3210_final
   ```
3. **Pull LFS files (if not already done)**:
   ```
   git lfs pull
   ```

3.4 Installing Dependencies

Install the required npm packages:

```
npm install
```

3.5 Running the Simulator

Start the development server:

```
npx vite
```

Open your web browser and navigate to http://localhost:5173 to view the simulator.

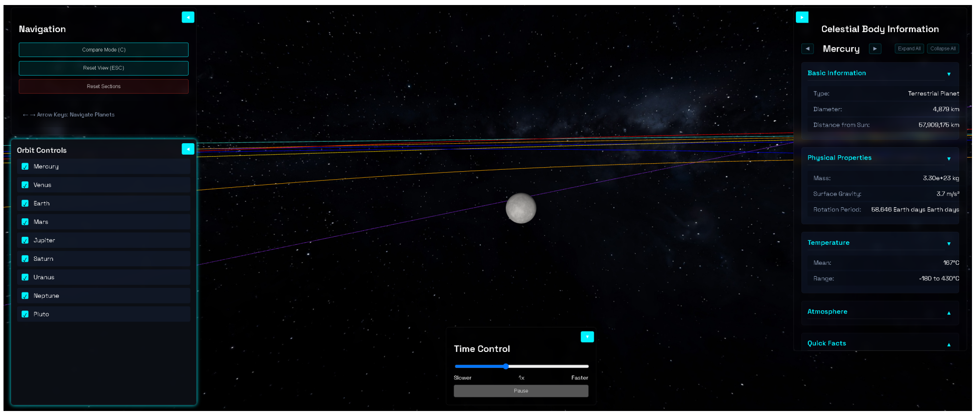
4 Getting Started

Upon launching the simulator, you will see the Sun at the center, surrounded by the orbiting planets.

4.1 User Interface Overview

The simulator's interface consists of four main panels:

1. **Navigation Panel**: Located on the left side, allows quick selection of planets and reset view.
2. **Planet Information Panel**: On the right side, displays detailed information about the selected planet.
3. **Time Control Panel**: At the bottom center, controls simulation speed and allows pausing/resuming time.
4. **Orbit Controls Panel**: Next to the planet information panel, allows toggling the visibility of orbital paths.



Type:	Terrestrial Planet
Diameter:	4,879 km
Distance from Sun:	57,909,175 km
Mass:	3.30e+23 kg
Surface Gravity:	3.7 m/s²
Rotation Period:	58.646 Earth days Earth days
Mean:	167°C
Range:	-180 to 430°C

Figure 1: Simulator Interface Overview

5 Using the Simulator

5.1 Navigating the Solar System

- **Rotate View**: Click and drag to rotate around the current focus point.
- **Zoom**: Use the mouse wheel or trackpad scrolling to zoom in and out.
- **Pan**: Right-click and drag to pan the camera.

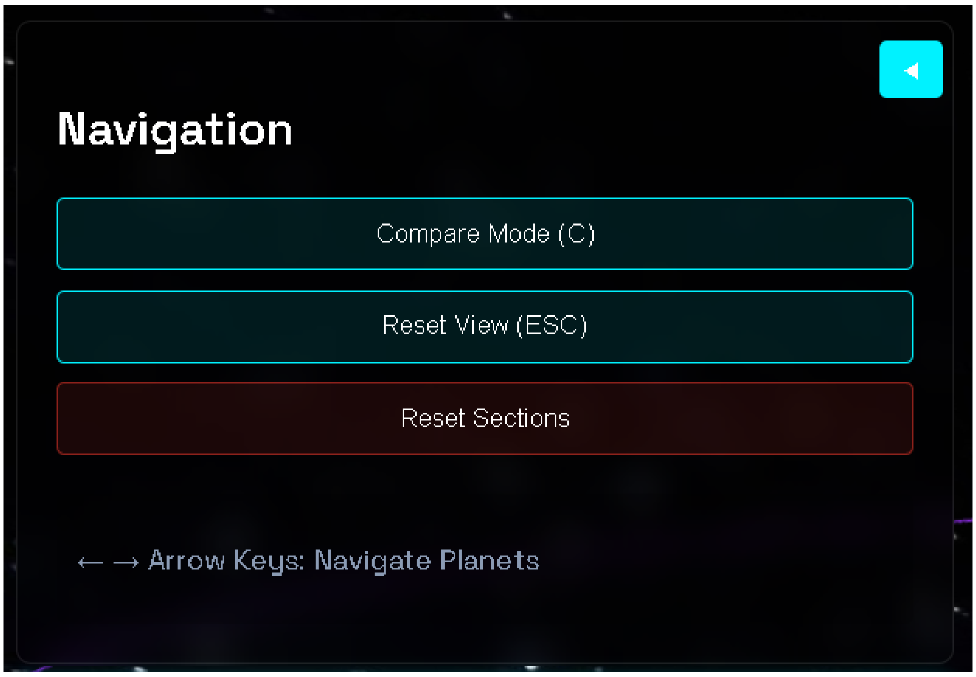


Figure 2: Navigation Controls

5.2 Interacting with Planets

- **Select a Planet**: Click on a planet or select it from the navigation panel to focus on it.
- **Access Planet Information**: View detailed information in the planet information panel.
- **Navigate Between Planets**: Use the left (<) and right (>) arrows in the planet information panel.

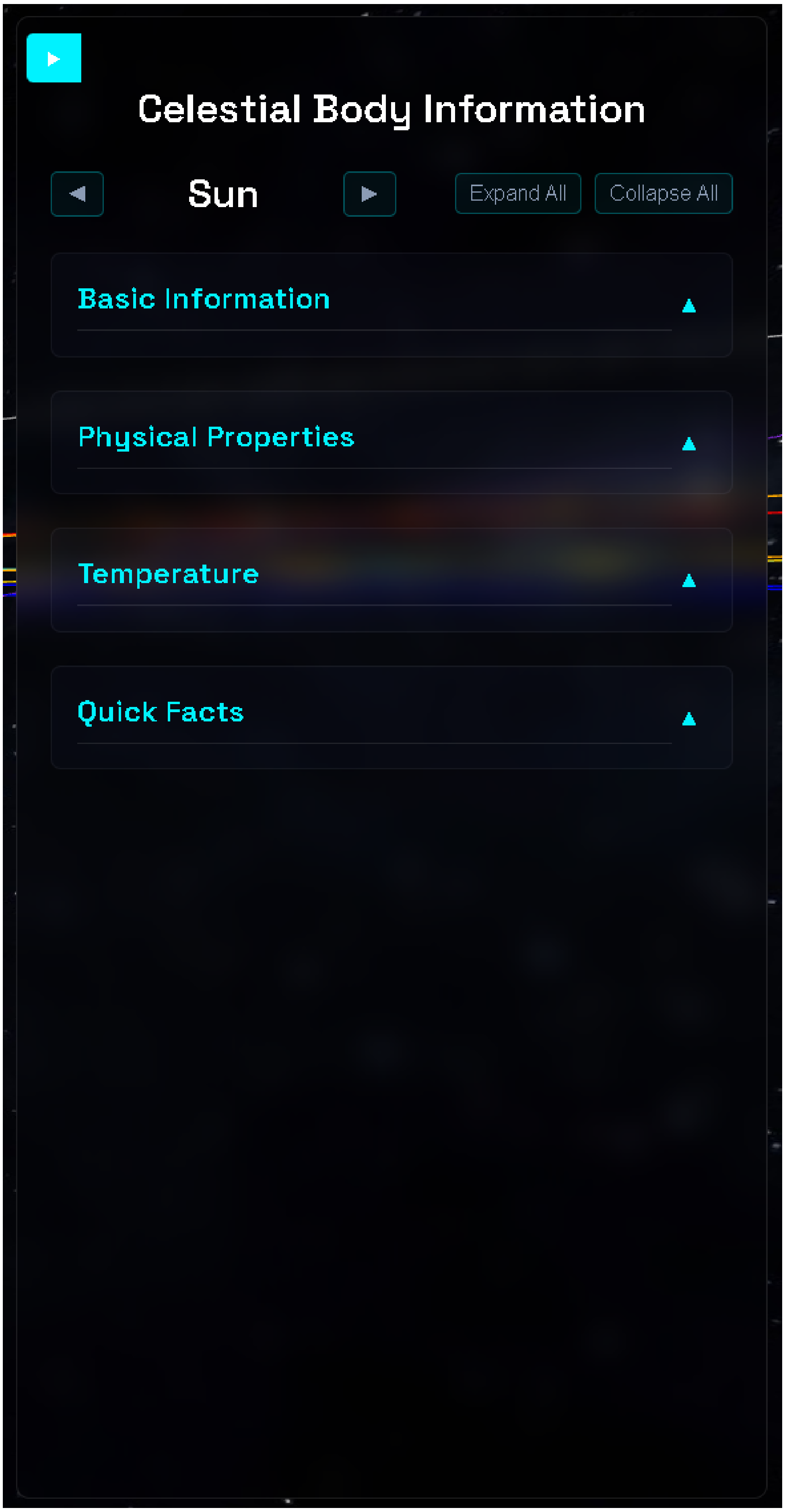


Figure 3: Planet Information Panel

5.3 Adjusting Time Speed

- **Time Slider**: Adjust the simulation speed using the slider in the time control panel.
- **Pause/Resume**: Click the "Pause" button to halt time progression; click again to resume.

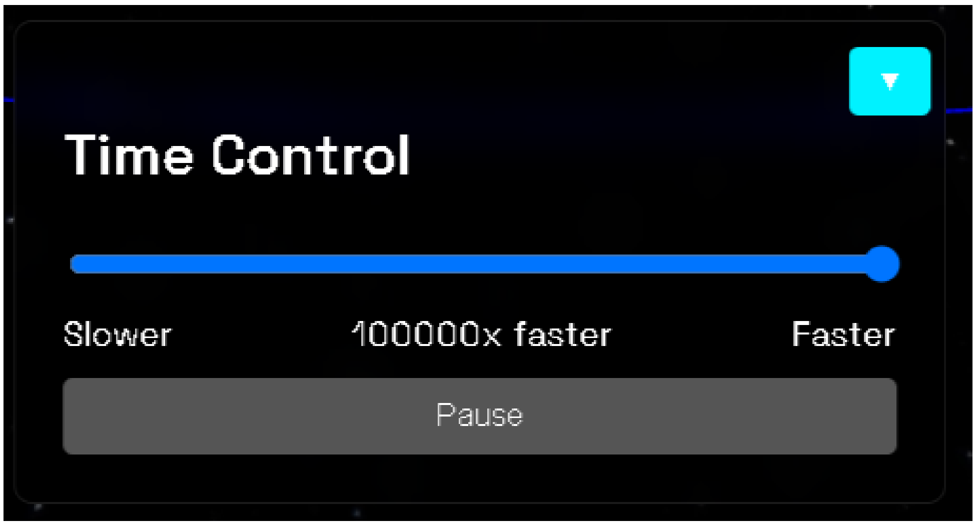


Figure 4: Time Control Panel

5.4 Controlling Orbits

- **Toggle Orbit Visibility**: Use the checkboxes in the orbit controls panel to show or hide orbital paths.
- **Customize Orbits**: Select specific planets whose orbits you wish to view or hide.

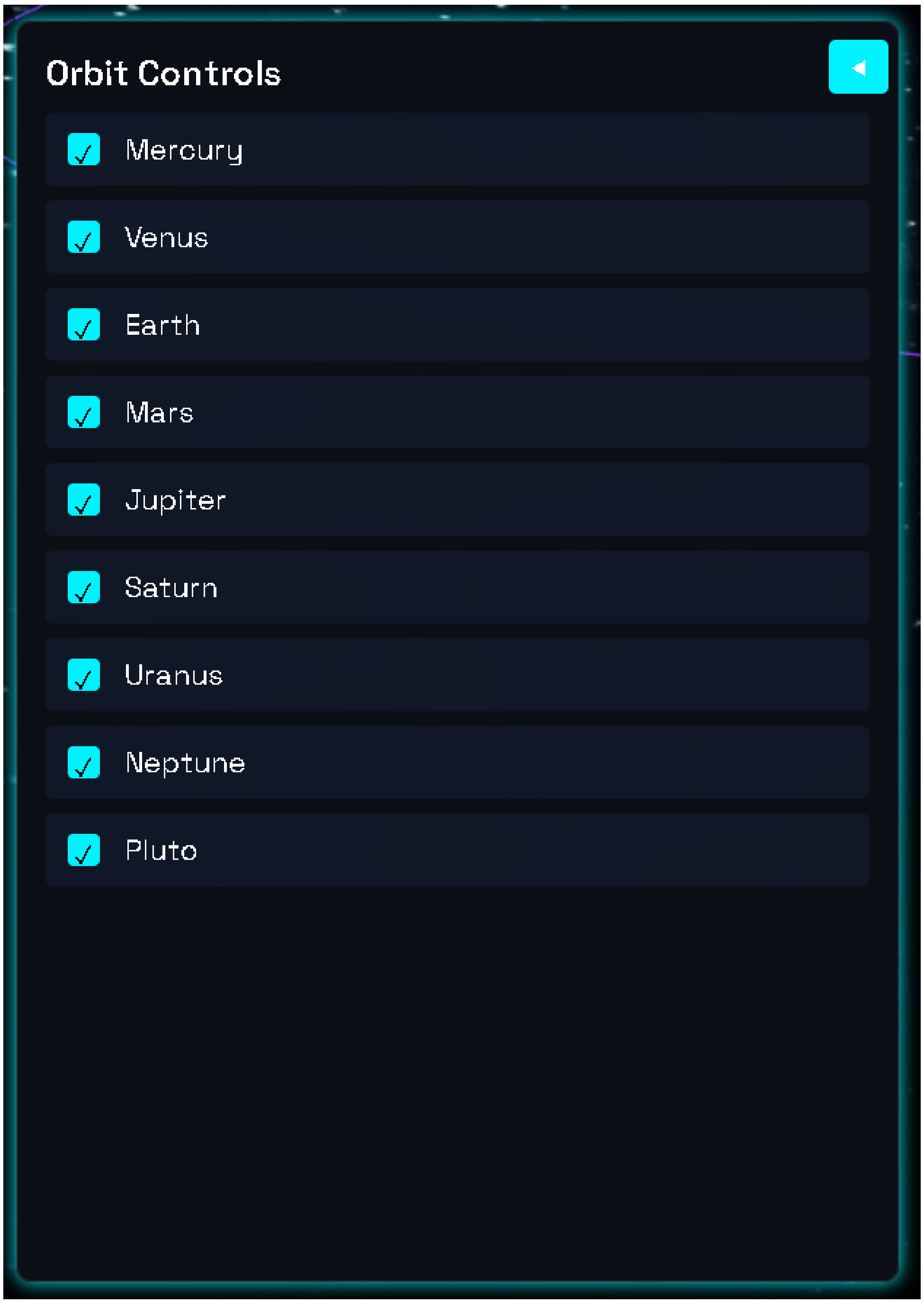


Figure 5: Orbit Controls Panel

6 Technical Challenges and Design Decisions

6.1 Accurate Orbital Mechanics

Challenge: Implementing realistic planet movements using Keplerian orbital elements.

Solution: Calculated each planet's position based on its orbital parameters, accounting for eccentricity, inclination, and other factors. Solved Kepler's Equation numerically to simulate true orbital paths.

6.1.1 Design Decision

Chose a balance between accuracy and performance by adjusting the number of segments in orbit lines based on each planet's eccentricity.

6.2 Performance Optimization

Challenge: Rendering complex 3D scenes with detailed textures and effects without compromising performance.

Solution:

- Implemented efficient algorithms for orbital calculations.
- Applied Levels of Detail (LOD) techniques to manage resource-intensive elements.
- Optimized textures and utilized Git LFS to handle large files.

6.2.1 Design Decision

Prioritized essential visual elements and introduced performance monitoring to ensure smooth operation across devices.

6.3 Visual Realism

Challenge: Achieving visually appealing and realistic representations of celestial bodies.

Solution:

- Used high-resolution textures for planet surfaces.
- Created custom shader materials for effects like Earth's day/night cycle and the Sun's glow.
- Added visual effects such as lens flares, atmospheric haze, and rotating clouds on Earth.

6.3.1 Design Decision

Implemented custom shaders for key planets to enhance realism, accepting the added complexity for improved user experience.

6.4 User Interface Design

Challenge: Designing an intuitive and responsive UI that provides necessary information without overwhelming the user.

Solution:

- Organized information into collapsible sections within the planet information panel.
- Implemented dynamic controls that update based on user interaction.
- Ensured responsiveness across different screen sizes using CSS Grid and flexible layouts.

6.4.1 Design Decision

Placed essential controls within easy reach and made advanced options accessible but unobtrusive.

6.5 Time Manipulation

Challenge: Allowing users to control the simulation speed while maintaining accurate physics.

Solution:

- Implemented a time scaling system that adjusts the simulation's time step based on user input.
- Ensured that orbital calculations are synchronized with the time scale for consistency.

6.5.1 Design Decision

Allowed negative time scales for reverse simulation and provided intuitive feedback on time changes.

7 Features Overview

7.1 Realistic Planet Textures

High-quality visuals provide an immersive experience, using textures sourced from reputable astronomical databases.

7.2 Accurate Orbits

Planets move according to real-world astronomical data, calculated using Keplerian orbital elements.

7.3 Interactive Controls

Full control over navigation, time manipulation, and visual elements enhances user engagement.

7.4 Informative Panels

Detailed information about each planet is presented in an organized and accessible manner.

7.5 Visual Effects

Incorporation of lens flares, glowing sun effects, rotating clouds, and atmospheric phenomena add to the realism of the simulation.

7.6 Responsive Design

The user interface is optimized for various screen sizes, ensuring a consistent experience across devices.

8 Troubleshooting

8.1 Planets Appear Without Textures

Cause: Texture files are missing due to Git LFS not being properly configured.

Solution:

Ensure Git LFS is installed and initialized before cloning the repository.

1. Install Git LFS (*see Section 3.2*).
2. Initialize Git LFS:

   ```
   git lfs install
   ```

3. Clone the repository and pull LFS files:

   ```
   git clone https://github.com/yourusername/solar-system-simulator.git
   git lfs pull
   ```

8.2 Performance Issues

Cause: The simulation may be resource-intensive on some devices.

Solution:

- Close unnecessary applications to free up system resources.
- Use a modern browser with good WebGL support (e.g., latest versions of Chrome or Firefox).
- Lower the simulation speed using the time control panel.

8.3 Controls Not Responding

Cause: The canvas may not have focus, or there could be an error in the application.

Solution:

- Click on the canvas area to ensure it has focus.
- Open the browser console to check for any error messages.
- Ensure all dependencies are properly installed.

8.4 Orbit Paths Not Visible

Cause: Orbit visibility may be toggled off in the orbit controls panel.

Solution:

- Open the orbit controls panel.
- Ensure the checkboxes next to the desired planets are checked.

9 Credits and Resources

9.1 Textures

- **Solar System Scope Textures**: www.solarsystemscope.com/textures (Attribution 4.0 International License)
- **JHT's Planetary Pixel Emporium**: planetpixelemporium.com (Used with permission)

9.2 Libraries and Tools

- **Three.js**: JavaScript 3D library used for rendering.
- **Vite**: Development server and build tool for modern web projects.
- **Git LFS**: Git extension for versioning large files.
- **Font Awesome**: Icon library for UI elements.
- **Google Fonts**: Typography resources for the user interface.

9.3 Further Reading

- **Keplerian Orbital Elements**: Wikipedia Article
- **Astronomical Units**: NASA Solar System Exploration
- **Three.js Documentation**: threejs.org/docs

10 Conclusion

We hope this simulator provides you with an engaging way to explore our solar system. The combination of accurate data and interactive features aims to both educate and inspire curiosity about the cosmos.

For any questions or feedback, please reach out!

Enjoy your journey through space!