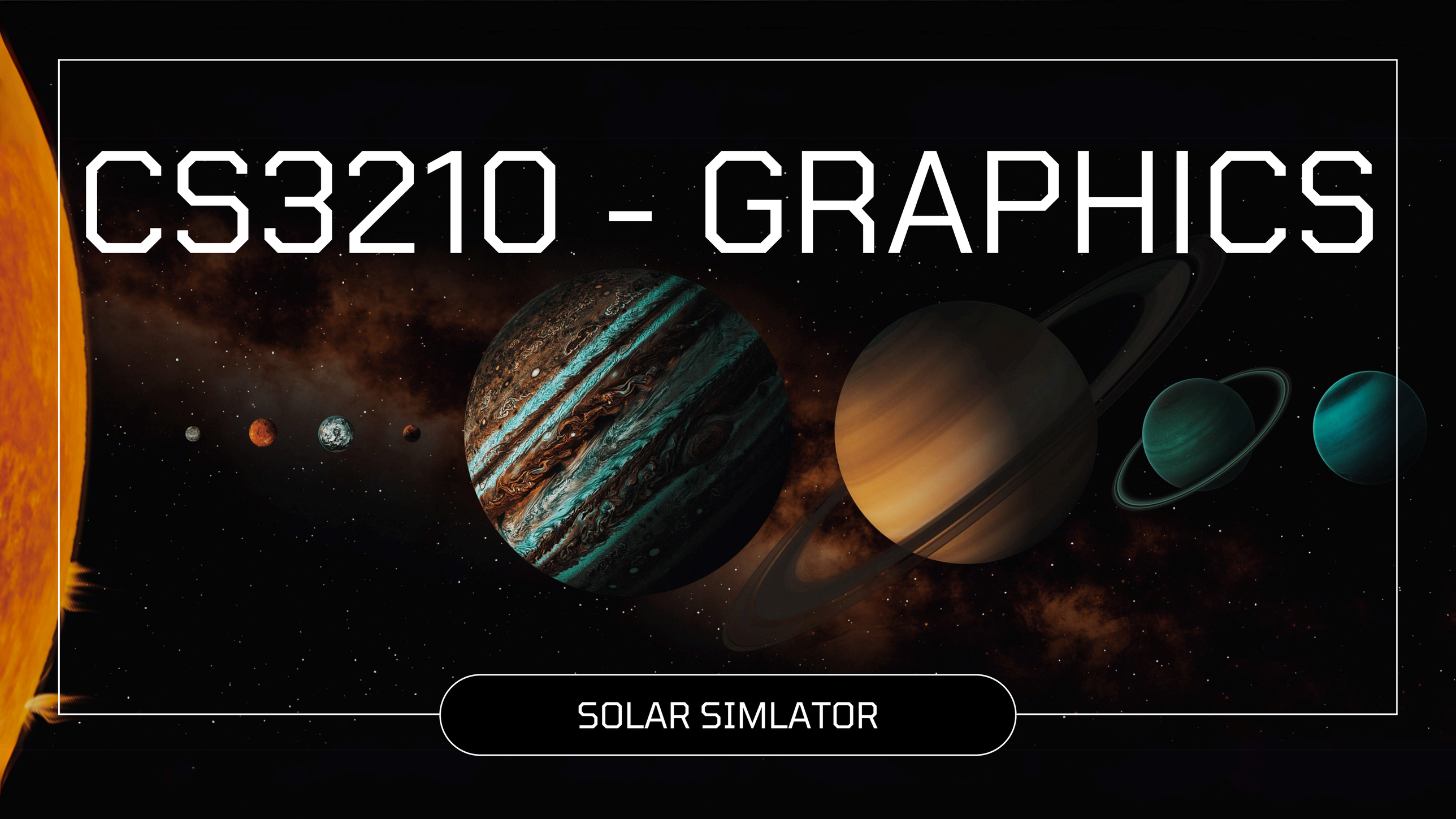
CS3210 - GRAPHICS



SOLAR SIMLATOR

01 PROJECT OVERVIEW

OVERVIEW

Objective: Create an interactive 3D visualization of the solar system with realistic scale and textures

Key Features:

- **Navigation**: Traverse through planets and view their orbits and relative positions.
- **Realism**: Accurate sizes, distances, orbits and textures for all celestial bodies.
- **Interactivity/UI**: Display key stats for each planet, including size, orbital speed, and distance from the Sun.
- **Time Manipulation:** Adjust the simulations time scale to observe orbits and rotation

TECHNOLOGIES

FRAMEWORKS

- **Three.js**: Enables 3D rendering and creation of detailed solar system scenes in browser.
- **WebGL**: Delivers hardware-accelerated graphics for smooth performance and realistic visuals.

DEVELOPMENT

- **JavaScript (ES6+):** Core logic and interactive functionality.
- **HTML5 & CSS3:** Provide the application's structure and styling.
- **Node.js & npm:** Streamline package management and build processes.

ADDITIONAL

- **Git LFS:** Manages large assets like high-resolution textures.

Git LFS extends Git to efficiently manage large files by replacing large objects in your repository with lightweight pointers. These pointers reference the actual files, which are stored in a remote LFS server or alongside your Git hosting service (e.g., GitHub).

02 FEATURES / DEMO

GRAPHICS/VFX

Enhancing the visual fidelity and immersive experience of the simulator

- High - Resolution Textures
- Surface Detailing
- Shaders
- Realistic Shadows/Lighting
- Lighting Effects (Sunflare)

SIMULATION

Creating accurate representation of celestial mechanics and physical behaviors within the solar system.

- Keplerian Orbital Calculations
- 3D Position Calculations
- Time Manipulation
- Visualized Orbit Paths

UI/UX

Prioritizing usability of the simulator and how users interact with it to explore and learn.

- Information Panels
- Camera Orbit Controls
- Toggle UI Elements
- Shortcuts
- Responsive UI
- Navigation System

GRAPHICAL FEATURES

High-Quality Texturing

- Realistic planetary surfaces using NASA imagery and high-resolution textures.
- Nearly all textures used are 8K quality, allowing for consistently high LOD (Neptune's and Uranus's are 2K, Pluto's is fictitious)
- Realistically sized and textured rings for Saturn and Uranus

Dynamic Lighting

- Simulated Sun as a light source
- Real-time lighting and shadows.
- Day-night cycles with smooth transitions.
- Simple sunflare effect adds to space immersion

UI Enhancement

- Thematic fonts and icons integrated with the visual design for a cohesive look.
- UI Shaders to create unique and high quality feeling sci-fi inspired UI

SIMULATION FEATURES

PHYSICS

- Simplified but realistic orbital mechanics.
- Accurate axial tilts and rotation speeds for planets. (Orbit Speed/Period is not)
- Orbits are derived from Kepler's first law, leveraging public solar system orbital data

ACCURATE SCALE

- Distances and sizes of planets are scaled down proportionally to preserve realism while maintaining interactivity.
- True-to-life representation of celestial hierarchy (e.g., relative size of the Sun vs. Earth).

TIME MANIPULATON

- Variable time controls:
  - Pause, slow down, or speed up planetary motion.
  - Observe long-term orbital changes in seconds.

UI/UX FEATURES

Planet Info

- Interactive Dynamic display of planetary stats when selected:
- Name, size, orbital speed, distance from the Sun, and other key data.
- State management to preserve menu states when cycling through planets

Navigation

- Intuitive mouse and keyboard support for panning, zooming, and rotating the camera.
- Quick navigation access for jumping between planets and sun.
-

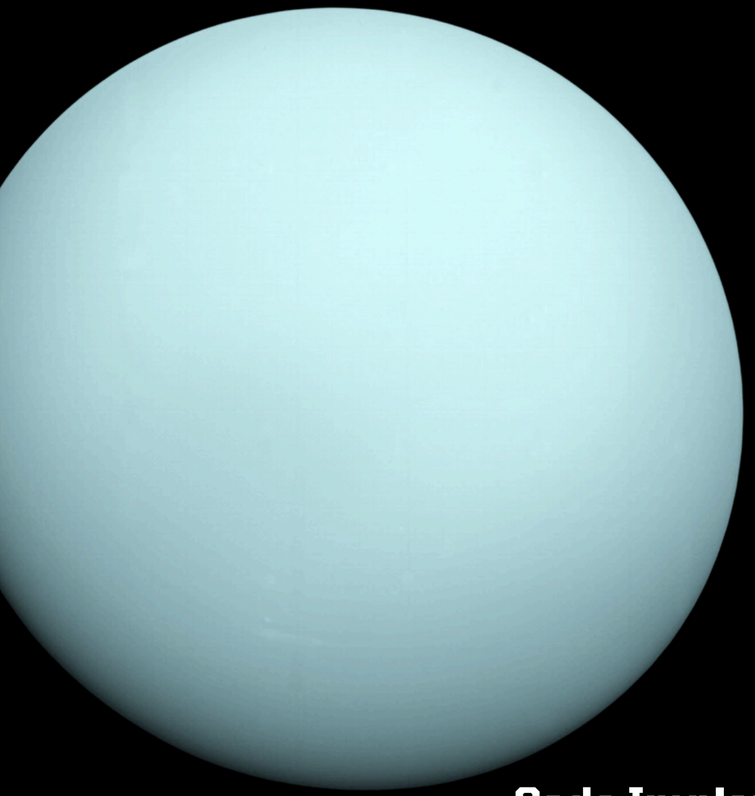
Customization

- Toggle visibility of orbit lines to suit user preferences.
- Allow user to “minimize” the info panels on screen enhancing view and usability

LIVE DEMO

SOLAR SIMULATOR

03 TECHNICAL CHALLENGES



Accurate Orbits

Challenge

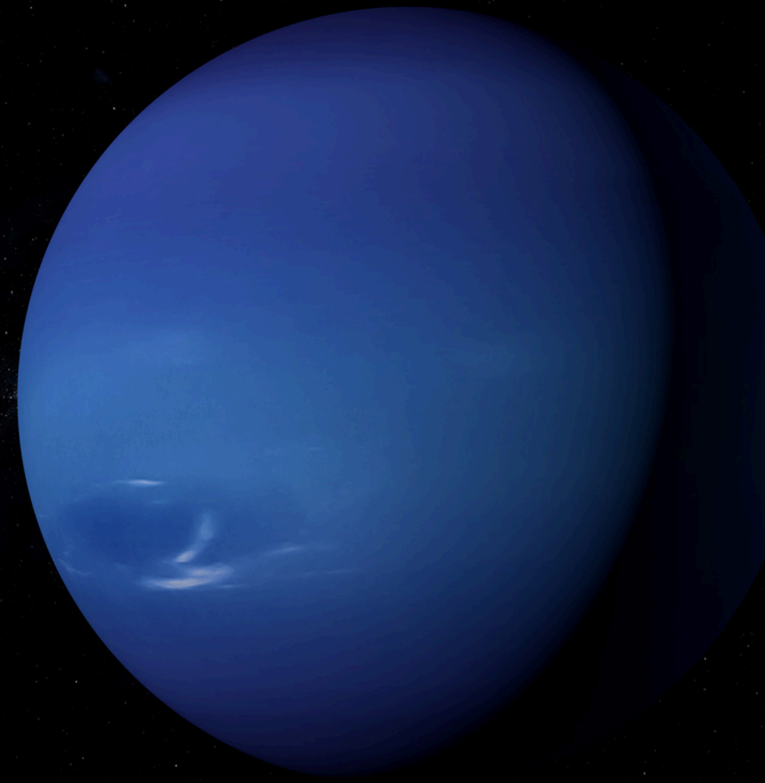
- Simulating realistic planetary motions using complex orbital parameters while balancing performance and accuracy.

Code Implementation

- Kepler's Equation: Solved numerically in calculateOrbitalPosition() using iterative methods for precision.
- Orbit Path Resolution: Adjusted resolution to balance computational performance and visual accuracy.

Design Decision

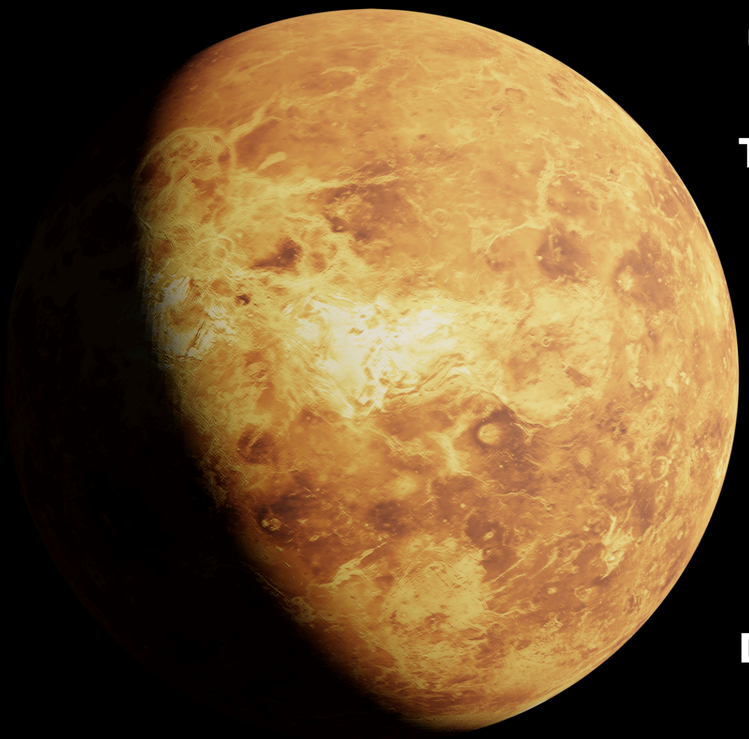
- Simplified orbital calculations where precision was less critical to optimize rendering performance while maintaining realism.



VISUAL REALISM / SHADERS

Challenge

- Achieving lifelike appearances of celestial bodies, including terrain details, atmosphere, and lighting effects.



Technical Solutions:

- Normal and Bump Maps:
  - Used for planetary textures (e.g., Earth) to enhance terrain details.
  - Leveraged MeshPhongMaterial properties (normalMap, bumpMap) for realistic shading.
- Custom Shaders:
  - Developed shaders for advanced effects like atmospheric glow and the Sun’s corona, and sunlight/normal calculations for earths day/night map blending.

Design Decision

- Invested significant time in shader development to enhance visual appeal, accepting the added complexity for a more immersive experience.

UI/UX DESIGN

Challenge

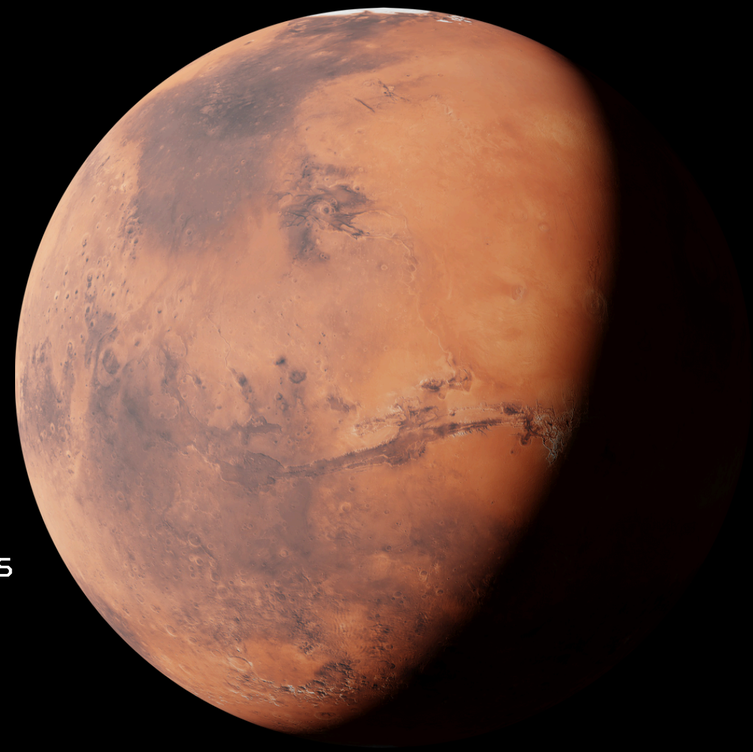
- Designing an intuitive, responsive UI that integrates seamlessly with the 3D simulation.

Implementation Details

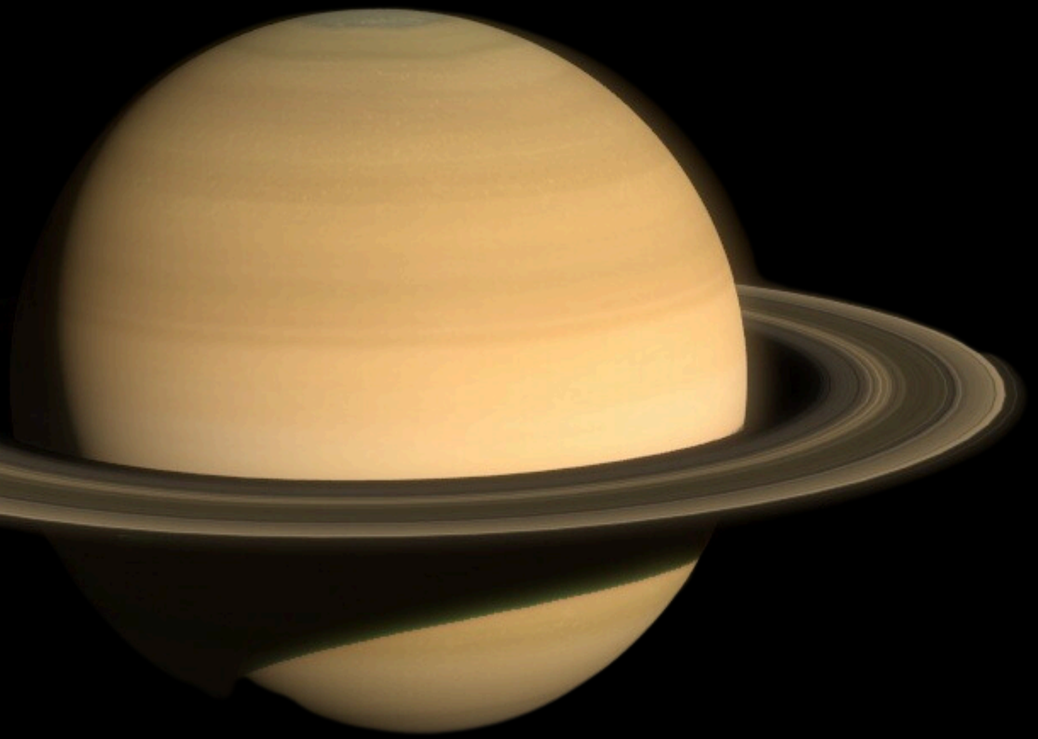
- HTML and CSS:
  - Structured UI elements using semantic HTML.
  - Used CSS Grid and Flexbox for adaptable layouts across screen sizes.
  - Applied a cohesive design theme with space-inspired fonts and colors.
- Dynamic Updates:
  - UI panels update in real-time based on user interactions, such as selecting planets or adjusting simulation speed.

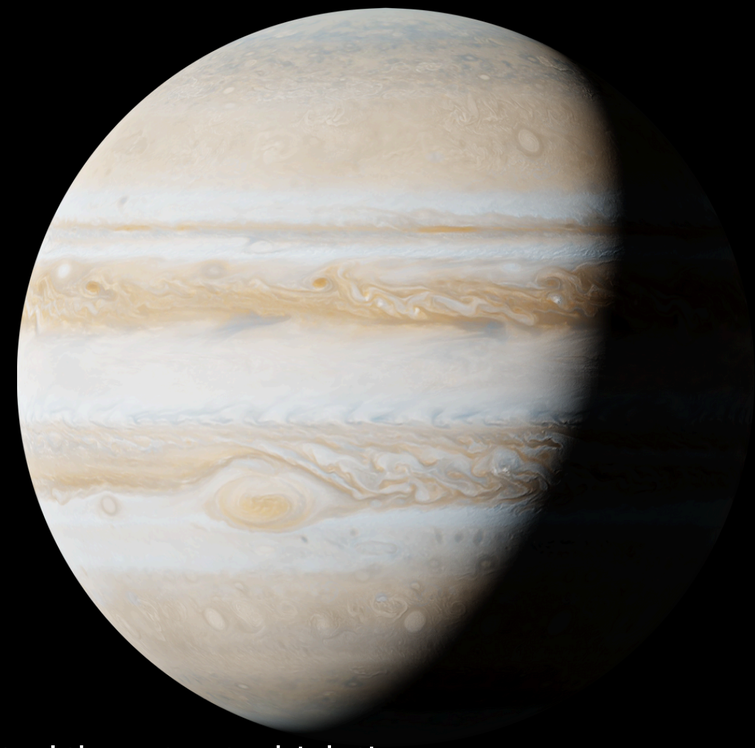
Design Decision

- Kept the interface on borders to avoid distracting from the 3D scene while maintaining functionality.



TIME MANIPULATION





Challenge

Allowing users to control simulation speed, including pausing time, without breaking orbital physics.

Technical Approach

- Time Scaling:
  - Introduced a timeScale variable to control the rate at which time progresses or if its does at all with an isPaused flag.
- Animation Loop Adjustment:
  - Synchronized all celestial body movements using a consistent elapsedTime variable.

Design Decision

- Offered a range of time scales, including reverse time, to enhance interactivity and user engagement while preserving simulation integrity.
- Major core refactor to modularize time control into its own file (TimeManager.js)


04
RESOURCES

RESOURCES

Textures:

- [Solar System Textures (Attribution 4.0 License)](solarsystemscope.com/textures)
- [JHT's Planetary Pixel Emporium (Non-Redistributable)](planetpixelemporium.com/)

Fonts:

- [Google Fonts](https://fonts.google.com/)

Data:

- [Basic Data / Orbits (NASA)](https://nssdc.gsfc.nasa.gov/planetary/factsheet/)
- [Solar System Data (JPL/NASA)](https://ssd.jpl.nasa.gov/horizons/app.html#/)

THANK YOU!