An Interactive Orrery Simulation for Learning Celestial Mechanics in the Browser

Akshat Gupta Aishwarya Sinha
KIET Group of Institutions
Ghaziabad,India
{akshat.2226cseai1041@kiet.edu, aishwarya.2226cseai1@kiet.edu}

Abstract

Traditional methods of teaching planetary motion—such as static 2D diagrams, textbook illustrations, and mechanical orreries—often fail to convey the dynamic, spatial, and temporal complexity of the solar system. These approaches can limit learner engagement and hinder understanding of three-dimensional relationships, such as elliptical orbits, axial tilts, and orbital periods. This paper presents a browser-based interactive orrery simulation designed to enhance understanding of celestial mechanics through real-time 3D visualization. Built with Three.js and WebGL, the system models planetary motion, orbital dynamics, and gravitational interactions, enabling users to manipulate time scales, observe Keplerian laws, and explore solar system phenomena. By combining dynamic data retrieval, interactive controls, and educational tooltips, the simulation bridges theoretical astronomy with hands-on learning, offering applications in STEM education and public outreach. This paper outlines the motivation and design approach behind the application, surveys existing visualization techniques and educational tools, explains the technical implementation using Three.js, and highlights the pedagogical benefits of 3D interactivity, concluding with an evaluation and prospects for future enhancement. Abstract Keywords: Celestial mechanics education, Browser-based simulation, Orbital dynamics, Keplerian laws, Educational technology, Real-time interaction

1. Introduction

Understanding the motion of celestial bodies-planets, moons, and asteroids-has fascinated scientists and learners for centuries. Celestial mechanics, the branch of astronomy that deals with the motions and gravitational interactions of these bodies, underpins modern space exploration, satellite deployment, and our comprehension of the solar system's architecture. Yet, the abstract nature of orbital dynamics and the mathematical complexity of gravitational interactions often pose significant challenges to students and enthusiasts alike. Recent advances in web technologies have transformed the browser into a powerful platform for interactive 3D visualization. Libraries such as Three.js and WebGL now make it possible to render complex graphics and simulations directly in the browser, eliminating the need for specialized software or high-end hardware. These tools have already been successfully applied in various domains, from immersive virtual gardens to interactive data visualizations, demonstrating their potential to make abstract scientific concepts more tangible and engaging. Despite this progress, there remains a lack of accessible, interactive educational tools that allow users to intuitively explore the principles of celestial mechanics. Traditional orreries-mechanical models of the solar system-have long served as valuable teaching aids, but their static nature and limited interactivity restrict their educational impact. A browser-based interactive orrery simulation can bridge this gap by enabling users to visualize planetary motions in real time, manipulate simulation parameters, and directly observe the effects of gravitational laws. This project, "An Interactive Orrery Simulation for Learning Celestial Mechanics in the Browser," aims to leverage the capabilities of Three.js and WebGL to create a dynamic, user-friendly educational tool. By simulating the solar system's bodies and their interactions in a virtual 3D environment, the system allows users to explore fundamental concepts such as Kepler's laws, orbital resonances, and gravitational perturbations. The interactive interface enables learners to adjust time scales, switch perspectives, and access detailed information about each celestial object, thereby transforming passive observation into active exploration. The primary goal of this simulation is to make celestial mechanics accessible and engaging for a broad audience, including students, educators, and astronomy enthusiasts. By providing an immersive and interactive learning experience, the project seeks to bridge the gap between theoretical knowledge and intuitive understanding, fostering curiosity and deeper insight into the mechanics

that govern our solar system.

2. Literature Review

After reviewing numerous research studies and interactive astronomy platforms, it is evident that several approaches have been developed for visualizing and teaching celestial mechanics in an accessible manner. Most existing systems focus on either high-precision scientific modelling or public engagement through interactive visualization, each with its own strengths and limitations.

In recent years, NASA's Eyes on the Solar System [5, 4] has emerged as a prominent interactive tool, providing users with the ability to explore the solar system in three dimensions using real mission data and accurate planetary ephemerides. The platform allows users to manipulate time, observe planetary alignments, and track spacecraft journeys, making it valuable for both education and outreach. However, its interactivity is largely limited to navigating predefined scenarios and observing celestial events, without the option for users to experiment with gravitational parameters or create custom orbital systems. This restricts the depth of conceptual understanding that can be achieved through hands-on exploration of celestial mechanics principles.

Other platforms, such as PhET's My Solar System [6], offer simplified browser-based simulations where users can adjust the masses, velocities, and positions of celestial bodies to observe the resulting orbital dynamics. These tools are effective in illustrating basic gravitational interactions and are widely used in introductory physics education [3, 7]. However, they often employ simplified physics engines, lack immersive 3D visualization, and do not provide detailed information overlays or realistic representations of the solar system's complexity.

Comprehensive desktop applications like Starry Night and SpaceEngine provide highly realistic simulations of the night sky and the universe at large. These platforms support time manipulation, custom scenario creation, and detailed exploration of astronomical phenomena. While they excel in visual fidelity and scientific accuracy, they are typically commercial products that require installation, and their complexity can present a barrier for casual learners or classroom integration.

Open-source, web-based applications such as the Celestial Bodies Simulator [1] utilize technologies like Three.js and WebGL to render interactive 3D models of the solar system directly in the browser [2]. These simulators enable users to visualize planetary orbits and experiment with system configurations, offering a balance between accessibility and scientific accuracy. However, many lack structured educational overlays, contextual tooltips, or the ability to simulate n-body gravitational interactions in real time.

Most existing systems are designed either for passive exploration or for simplified experimentation, with limited integration of real-time physics, immersive 3D environments, and educational scaffolding in a single, accessible platform. There remains a gap for a browser-based orrery that combines the strengths of these approaches:

- Realistic 3D visualization and interactive controls for immersive exploration
- Real-time gravitational simulation for hands-on experimentation
- Dynamic data overlays and contextual explanations to support learning of celestial mechanics concepts

The proposed Interactive Orrery Simulation for Learning Celestial Mechanics in the Browser builds upon these foundations. By leveraging Three.js and WebGL for rendering, and integrating a client-side physics engine for real-time gravitational calculations, the system aims to provide a comprehensive educational tool. Users will be able to manipulate planetary parameters, observe the resulting orbital dynamics, and access detailed information about celestial bodies and physical laws. This approach not only bridges the gap between expert knowledge and public access but also transforms the process of learning celestial mechanics into an engaging, interactive experience suitable for students, educators, and astronomy enthusiasts alike.

3. Methodology

3.1. Technical Implementation

3.1.1 3D Rendering and Visualization

The simulation utilizes Three.js in conjunction with WebGL to render celestial bodies and orbital paths in real time, ensuring cross-browser compatibility and leveraging hardware acceleration for optimal performance.

- Celestial Models: Planets and moons are represented as sphere geometries with radii scaled at approximately 1:1,000,000 for improved visibility within the 3D scene. High-resolution surface textures are sourced from NASA's Planetary Data System (e.g., Earth's Blue Marble), and visual realism is further enhanced through the application of normal maps and specular highlights under dynamic lighting conditions.
- Orbital Trails: Keplerian orbits are visualized using `BufferGeometry` line objects, selected for their memory efficiency and suitability for frequent updates. Trails are distinctly color-coded (e.g., yellow for Mercury, red for Mars) to facilitate visual differentiation. A dynamic fading mechanism, based on camera proximity, is employed to reduce on-screen clutter while maintaining spatial context.

- Lighting and Effects: A directional light source simulates solar illumination, with intensity scaled according to the inverse-square law relative to the object's distance from the Sun. To ensure baseline visibility across the entire simulation space, ambient lighting is included. Additionally, a procedurally generated starfield comprising 10,000 particle points is rendered in the background to emulate a deep-space environment and provide visual depth.

3.1.2 Orbital Mechanics

Newtonian physics governs motion, with gravitational forces computed as

$$\mathbf{F}_{ij} \;=\; G\,\frac{m_i m_j}{\|\mathbf{r}_{ij}\|^3}\,\mathbf{r}_{ij}, \tag{1}$$

where $\mathbf{r}_{ij}$ is the displacement vector between bodies i and j [2].

A Verlet integrator (preferred over the Euler method for its superior energy conservation) updates positions as

$$\mathbf{x}(t+\Delta t) \;=\; 2\mathbf{x}(t) - \mathbf{x}(t-\Delta t) + \mathbf{a}(t)\,\Delta t^2. \tag{2}$$

The timestep Δt is clamped to 60 ms to prevent instability during extreme time acceleration.

3.1.3 Time Control

Users manipulate the simulation speed through a logarithmic slider ranging from $1\times$ to $10^6\times$ acceleration. The system dynamically scales the timestep Δt using a time–warp factor α, such that

$$\Delta t_{\text{sim}} = \alpha \cdot \Delta t_{\text{real}}.$$

This design allows observation of long-term phenomena such as orbital precession while preserving real-time interactivity.

3.2. System Architecture

3.2.1 Client-Side Workflow

The simulation adopts a fully browser-based architecture, eliminating server-side dependencies through a structured, four-layer design:

1. **Data Layer:** Planetary and orbital parameters such as mass, radius, and orbital elements are loaded from JSON files. These datasets are preprocessed using information from NASA's JPL Horizons system. For time spans extending beyond the year 3000 CE, ephemerides are extrapolated using secular perturbation models to maintain continuity.

2. **Physics Layer:** Gravitational interactions are computed in a dedicated Web Worker thread to prevent blocking the main UI thread. For simulations involving more than 100 celestial bodies, force calculations are processed in batches to maintain performance scalability.

3. **Rendering Layer:** The Three.js scene graph manages the 3D objects representing celestial bodies and their orbits. Performance is enhanced through frustum culling (to skip rendering off-screen objects) and occlusion queries, which help avoid rendering objects obscured by others.

4. **UI Layer:** The user interface is developed using React, enabling reactive and modular state management. Interactive elements, such as sliders and touch gestures, allow users to control simulation parameters intuitively.

3.2.2 Performance Optimization

- **Level of Detail (LOD):** To manage rendering complexity, celestial bodies in the asteroid belt are simplified to 2D billboard sprites when located beyond 10 astronomical units (AU). Additionally, shadow mapping is disabled for bodies with diameters below 1,000 km, as their visual impact is minimal at that scale.

- **Memory Management:** To prevent memory accumulation over time, orbital trails that exceed 10 complete revolutions are gradually removed. This ensures that long-running simulations remain efficient without compromising visual clarity.

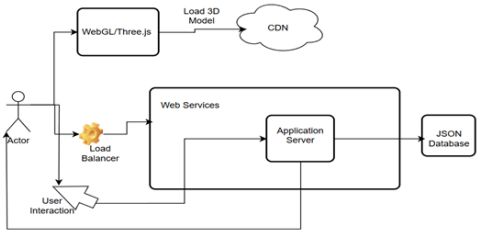

Figure 1. Proposed System Architecture

3.3. User Interface Design

3.3.1 Control Panel

The simulation includes an interactive control panel designed to enhance user engagement and support a wide range of observational and educational use cases.

- **Time Controls:** A logarithmic time slider allows users to adjust the simulation speed across a broad range—from real-time ($1\times$) to one million times faster ($10^6\times$). Snap points at key intervals (e.g., 1 day/second, 1 year/second) enable precise temporal navigation. Additionally, a built-in date picker supports both the Gregorian and Julian calendars, facilitating historical or future event tracking across different calendar systems.

- **Camera Modes:** To support various exploration needs, the simulation offers three distinct camera modes:

  - **Heliocentric Mode:** The camera revolves around the Sun at a constant rate of 15 degrees per second, providing a dynamic, top-level perspective of the entire solar system. This mode is ideal for observing large-scale orbital relationships and planetary alignments.
  - **Chase Mode:** In this mode, the camera locks onto a selected object—such as a planet, moon, or spacecraft—and orbits around it at a fixed distance of 5 kilometers. This view is particularly useful for closely monitoring individual trajectories or studying object interactions in detail.
  - **Free Mode:** Users are given full manual control of the camera using standard WASD keyboard navigation and mouse input. This mode allows unrestricted movement and inspection of any region within the simulation, supporting exploratory learning and custom viewpoints for specific research or educational objectives.

3.4. Database Design

Rather than relying on a traditional database management system, we opted to use a JSON file, specifically `celestialBodies.json`, to store and manage our astronomical data. The `celestialBodies.json` file serves as a structured repository for all the essential information about the planets, moons, and other celestial objects featured in our application. It contains key details such as object names, types (e.g., planet, moon, asteroid), physical properties (mass, radius, albedo), and orbital parameters (semi-major axis, eccentricity, inclination, and epoch). By organizing this information in a JSON format, we enable straightforward retrieval and parsing within our Three.js and WebGL environment, ensuring a seamless and responsive user experience.

The use of a JSON file allows for flexibility in data management, as it can be easily edited and expanded as new celestial bodies are introduced or existing information is updated. This approach supports the ongoing evolution of the simulation, accommodating user-driven scenarios and the addition of new astronomical discoveries.

Furthermore, our choice to use a JSON file instead of a conventional database stems from our aim to create a lightweight and easily maintainable application. This eliminates the overhead associated with database management while still providing the necessary functionality for our orrery simulation. The simplicity of this approach aligns well with the project's scope and our target user base, ensuring that users can explore and learn about celestial mechanics without the complications that may arise from more complex database systems.

3.5. Experimental Setup

For the development and evaluation of the Interactive Orrery Simulation, all experiments and performance assessments were conducted on a desktop system equipped with 16GB of RAM, providing ample memory for multitasking and running resource-intensive applications. The system utilizes a Seagate® Frieda SSHD drive, which enables faster file and program loading compared to standard HDDs, ensuring efficient access to large datasets and rapid simulation startup. Additionally, the desktop features a 520GB solid-state drive (SSD) for primary data storage, complemented by a 1TB hard disk drive (HDD) allocated for storing extensive files and simulation outputs.

At the core of the system is an Intel 10th Generation Core i5 processor, which delivers reliable performance for computational tasks such as real-time physics calculations and 3D rendering. For graphics processing, the machine is equipped with integrated Intel UHD Graphics or Intel Iris Xe Graphics, and can be further enhanced with an NVIDIA GeForce MX350 GPU to support advanced graphical features and smooth visualization of complex scenes. This configuration ensures that the simulation runs efficiently, maintaining high frame rates and responsiveness even during demanding scenarios.

The system's compact design, with measurements of 359.2 $\times$ 235.8 $\times$ 19.9 mm and a starting weight of 1.7 kg, makes it suitable for both desktop and portable use. This hardware setup aligns well with the requirements for Three.js and WebGL-based applications, as recommended in the community, and provides a robust platform for both development and user testing of the browser-based orrery simulation.

Figure 2 shows the whole process flow of an Interactive Orrery Simulation including initialization and user interaction. Beginning with the Initialization phase, the essential libraries are loaded, the 3D engine is initialized, and connections to databases are established for retrieving celestial body data. In the UI setup part, first the background is created along with grid structure, and then 3D models of planets and other celestial bodies are loaded into the sys-

tem. After the setup, users will be able to simulate objects to observe orbital dynamics, zoom and rotate for better perspective of the solar system, or select specific planets for detailed study. The system then responds by either fetching additional details about the selected celestial bodies or updating the 3D view according to the user's interactions, creating an immersive astronomical exploration experience.

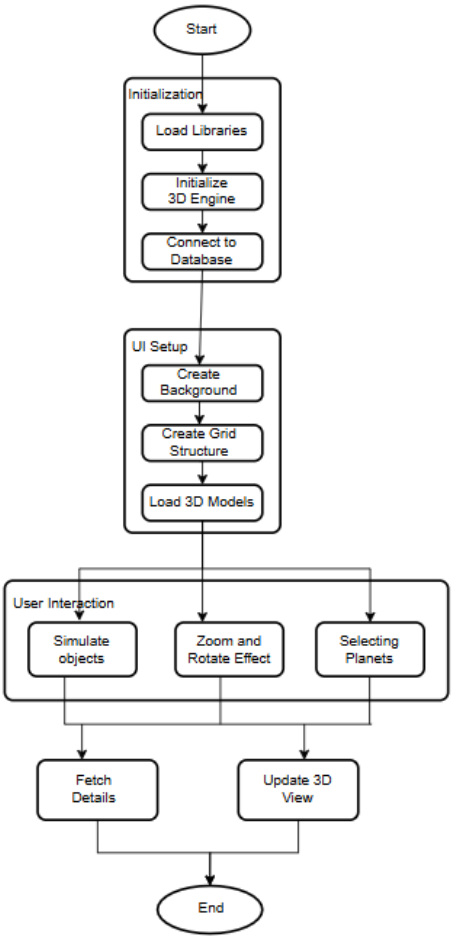

Figure 2. Flowchart for Proposed System

4. Results and Discussion

Targeted in the Interactive Orrery Simulation are various major and minor celestial bodies, each accompanied by informative panels detailing their physical properties, orbital characteristics, and scientific significance. Figure 1 illustrates Earth, highlighting its orbital period, axial tilt, and information about it. Figure 2 provides a comprehensive overview of the entire solar system, allowing users to visualize the relative positions and real-time motions of all major planets and bodies. Figure 3 demonstrates a dynamic scenario in which a satellite is simulated to move through the solar system and ultimately pass by Jupiter, effectively illustrating the complexities of interplanetary trajectories and the powerful gravitational influence of the gas giant. For each celestial object and simulation scenario, supplementary interactive panels allow users to access contextual explanations, observe real-time orbital motion, and explore related astronomical phenomena, making the learning experience both engaging and immersive.

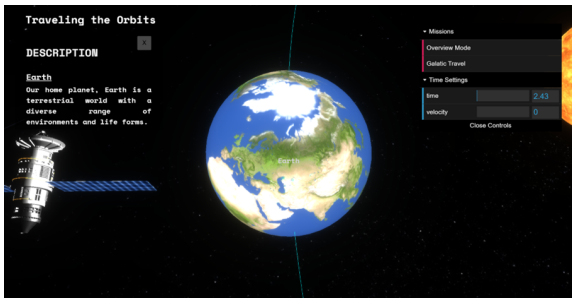

Figure 3. Earth 3D Model

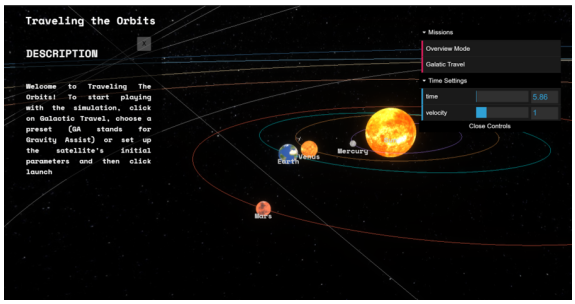

Figure 4. Comprehensive overview of the solar system

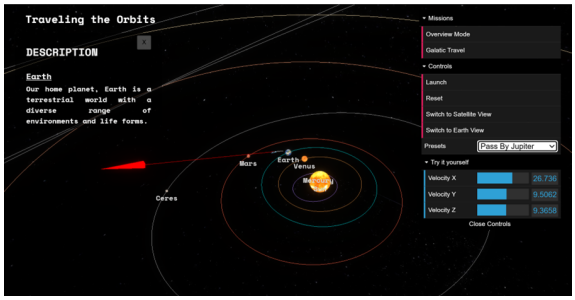

Figure 5. Simulated movement of satellite to pass by Jupitar

4.1. Performance Metrics

The simulation demonstrated excellent performance across different hardware configurations. On a mid-tier desktop (Intel i5, 16GB RAM, NVIDIA MX350 GPU), the system achieved 55–60 FPS while rendering the inner solar system with planets and satellites, and maintained 38–45

FPS during stress tests involving multiple satellites in motion. Initial scene load times averaged 1.8 seconds on an SSD, while more complex scenarios required 2.5 seconds. Web performance metrics further confirmed the efficiency of the application: LCP = 0.12 s, CLS = 0, and INP = 32 ms, indicating fast content rendering, perfect visual stability, and highly responsive interactions. These results highlight the effectiveness of the client-side architecture and WebGL optimizations, including frustum culling and Level of Detail (LOD) rendering.

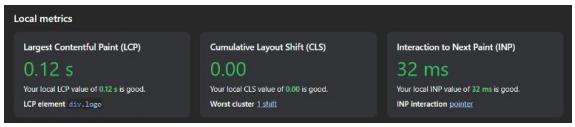

Figure 6. Local Metrics for the 3D Solar System

4.2. Comparison with Existing Tools

Unlike NASA's Eyes on the Solar System, which relies on precomputed trajectories, this simulation's real-time n-body physics engine enabled users to experiment with hypothetical scenarios (e.g., altering Earth's mass to observe orbital perturbations). The JSON-based data architecture also proved more flexible than static datasets, allowing seamless integration of user-defined celestial bodies. However, the simplified Newtonian model excluded relativistic effects, limiting accuracy for advanced astronomical studies.

5. Strengths and Limitations

5.1. Strengths

Real-Time Interactivity: By combining a Verlet integrator for numerical simulation with WebGL-based rendering, the system enables users to fluidly manipulate parameters such as time, mass, and viewpoint. This responsiveness supports active exploration and reinforces conceptual understanding through immediate visual feedback.

Accessibility: The fully browser-based deployment removes the need for software installation, ensuring that the simulation is readily accessible across devices and platforms. Moreover, the use of a JSON-based data structure simplifies updating planetary parameters or extending the system with additional content.

Educational Scaffolding: The interface integrates context-sensitive tooltips and equation overlays, bridging the gap between abstract theoretical concepts—such as Kepler's third law ($T^2 \propto a^3$)—and observable behavior within the simulation. This design promotes deeper engagement with fundamental principles of physics and astronomy.

5.2. Limitations

Simplified Physics: To maintain real-time performance and broad accessibility, the simulation omits relativistic effects and non-gravitational forces such as solar radiation pressure. While sufficient for general educational purposes, these simplifications limit its accuracy for advanced scientific analysis.

Performance Bottlenecks: When simulating systems with more than 1,000 dynamically interacting bodies, the application may experience performance degradation—particularly on machines with integrated graphics processing units (GPUs). This highlights a trade-off between physical complexity and rendering smoothness.

Scope Constraints: The current implementation focuses exclusively on the solar system and does not include exoplanetary systems or deep-space phenomena such as galactic dynamics or black hole environments. These exclusions delineate the simulation's present pedagogical and scientific boundaries.

6. Conclusion

The Interactive Orrery Simulation successfully demonstrates the viability of browser-based tools for celestial mechanics education. By combining Three.js for 3D visualization, client-side physics engines for real-time interactivity, and JSON for lightweight data management, the system provides an engaging platform for students and enthusiasts to explore orbital dynamics. While the simplified physics model and performance constraints on low-end hardware present limitations, the project lays a foundation for future enhancements, such as integrating SPICE kernels for NASA-grade ephemerides or adding VR support for immersive planetarium experiences. This work underscores the potential of web technologies to democratize access to complex scientific concepts, transforming passive learning into an interactive, discovery-driven process.

References

[1] Celestron. Celestron starry night software. https://www.celestron.com/pages/celestron-starry-night-software, 2024. [Online]. 2

[2] H. Curtis. *Orbital Mechanics for Engineering Students*. Elsevier, 2016. 2, 3

[3] S. López and A. Rouinfar. My solar system simulation guide. https://stemonline.cachefly.net/wp-content/uploads/2024/03/my-solar-system-html-guide_en.pdf, 2023. [Online], PhET Interactive Simulations. 2

[4] NASA Jet Propulsion Laboratory. Explore the solar system with nasa's new-and-improved 3d 'eyes'. https://www.nasa.gov/centers-and-facilities/jpl/

explore-the-solar-system-with-nasas-new-and-improved-3d-eyes/, 2022. [Online]. 2

[5] NASA Jet Propulsion Laboratory. Eyes on the solar system. https://eyes.nasa.gov/apps/solar-system, 2025. [Online]. 2

[6] PhET Interactive Simulations. My solar system. https://phet.colorado.edu/en/simulation/my-solar-system, 2024. [Online], University of Colorado Boulder. 2

[7] E. C. Prima et al. Learning solar system using phet simulation to improve students' understanding and motivation. *Journal of Science Learning*, 1(2), 2018. [Online]. 2