An Interactive Orbital Visualization System for Educational Purposes*

Yuseong Choi
Purdue University
Computer Science - Computer Graphics and Visualization
West Lafayette, Indiana, United States
choi865@purdue.edu

***Abstract*—This paper introduces an interactive 3D orbital visualization system designed as an educational tool and explains about fundamental principles of orbital mechanics. The system, developed in Python using VTK for graphics and PyQt5 for the user interface, allows users to observe celestial body motion and crucially to interactively modify core physical parameters such as solar mass, the gravitational constant, and orbital elements like eccentricity and inclination. By providing immediate visual feedback on these modifications, the tool aims to make abstract astrodynamics concepts more tangible and understandable for students.**

***Index Terms*—Orbital Mechanics, Visualization, Educational Tool, Python, VTK, PyQt5, Interactive Simulation, Astrodynamics**

I. Introduction

Understanding the dynamics of celestial bodies in orbit is a cornerstone of astrophysics and space engineering. However, the underlying principles of orbital mechanics, governed by complex gravitational interactions and mathematical formulations can be challenging to understand especially for new learners. Traditional teaching methods often rely on static diagrams and equations, which may not fully convey the dynamic and interconnected of orbital parameters. There is a need for accessible tools that can bridge the gap between theoretical concepts and intuitive understanding.

This work addresses this need by presenting an interactive 3D visualization system, 'orbital_fw.py'. The primary contribution is an educational tool that not only renders the orbits of planets and moons within our solar system but also empowers users to actively experiment with fundamental physical laws. Users can adjust parameters such as the Sun's mass, the gravitational constant (G), and the eccentricity and inclination of planetary orbits. The system then recalculates and re-renders the orbits in real-time, providing immediate visual feedback on the consequences of these changes. This interactive "what-if" capability is the core makes intuitive understanding of how these parameters change the orbital paths. The system is built using Python, Visualization Toolkit (VTK) for 3D rendering and PyQt5 for the graphical user interface making it a accessible and extensible platform.

II. Related Work

The visualization of orbital mechanics is not a new endeavor. Numerous professional software packages, such as Systems Tool Kit (STK) or GMAT, offer highly sophisticated simulation and analysis capabilities, but these are often complex and may have a steep learning curve making them less suitable for introductory educational purposes. On the academic front, many researchers have developed custom tools or applets for specific educational scenarios.

The foundational calculations for planetary positions in our system draw upon established astronomical algorithms, notably the methods detailed by Paul Schlyter in "Computing planetary positions" [1], which provides a practical guide to implementing Keplerian orbital mechanics. Data for the physical properties and initial orbital elements of celestial bodies are sourced from reputable institutions like NASA's Jet Propulsion Laboratory (JPL) Solar System Dynamics Group [2] and the Planetary Fact Sheet [3].

While many visualization tools exist, the emphasis of our project is on direct, interactive manipulation of fundamental physical constants and orbital characteristics within a simplified Keplerian framework for educational impact. Some advanced research tools, like REBOUND, provide highly accurate N-body simulations, which represent a more complex and computationally intensive approach than the two-body (Keplerian) model primarily used in our educational tool for clarity and responsiveness. Our system aims to provide a balance between reasonable physical accuracy for demonstrating core principles and a high degree of interactivity focused on cause-and-effect learning.

III. Method/Approach

The interactive orbital visualization system is designed around three core components: the orbital mechanics model, the visualization engine, and the user interface.

A. Orbital Mechanics Model

The system primarily implements a direct calculation of Keplerian orbits for the Sun, the eight major planets, and Earth's Moon.

1) **Data Initialization**: Physical properties (mass, radius, rotation period, axial tilt) are sourced from NASA [3]. Initial orbital elements (semi-major axis a, eccentricity e, inclination i, longitude of the ascending node Ω, argument of perihelion ω, and mean anomaly M at the J2000 epoch) are based on data from NASA/JPL [2].

2) **Orbital Element Propagation**: The ‘update_orbital_elements‘ method calculates time-dependent orbital elements using formulas from Schlyter [1]. The time d is measured in days from the J2000 epoch.
3) **Position Calculation**: The ‘calculate_position_from_elements‘ method determines the 3D Cartesian coordinates of celestial bodies from their orbital elements. This involves iteratively solving Kepler's equation ($M = E - e \sin E$) to find the eccentric anomaly E from the mean anomaly M, and subsequently the true anomaly ν.
4) **Interactive Physics Modification**: A key feature is the ability for users to modify fundamental parameters. The ‘update_orbital_elements_with_phy‘ method applies user-defined multipliers:
   - **Sun's Mass (M_{sun})**: A ‘sun_mass_scale‘ multiplier directly impacts the gravitational parameter $\mu = GM_{sun}$ and, to maintain specific orbital energy for a given period in a simplified view, can be modeled as affecting the semi-major axis ($a \propto (M_{sun})^{-1/3}$ if period is to be conserved, or more directly μ changes affecting period $T^2 \propto a^3/\mu$). The visual size of the Sun is also scaled.
   - **Gravitational Constant (G)**: A ‘G_multiplier‘ scales G, directly impacting μ and thus orbital periods and, potentially, semi-major axes ($a \propto G^{-1}$ if energy/period relations are maintained).
   - **Eccentricity (e)**: An ‘ecc_multiplier‘ scales the eccentricity of each planet's orbit relative to its original value, capped to prevent unstable orbits.
   - **Inclination (i)**: An ‘inc_multiplier‘ scales the inclination of each planet's orbit.
5) **Moon's Orbit**: The Moon's orbit is calculated geocentrically using elements propagated via Schlyter's formulas (‘update_moon_orbital_elements‘), and its final position is derived by adding its position relative to Earth to Earth's heliocentric position.

Rationale: Direct Keplerian calculation was chosen for this project that allows users to directly observe the consequences of parameter changes without the added complexity of N-body perturbations. This focus serves the primary educational goal of understanding fundamental two-body dynamics.

B. Visualization

The Visualization Toolkit (VTK) is used for 3D graphics rendering.

1) **Celestial Bodies**: Planets and the Sun are rendered as textured spheres (‘vtk.vtkSphereSource‘, ‘vtk.vtkTextureMapToSphere‘). Textures are sourced from Solar System Scope [4] for realistic appearance. The Sun's visual size is dynamically updated based on the ‘sun_mass_scale‘.
2) **Orbital Paths**: Orbit lines (‘vtk.vtkPolyLine‘) generated from a series of calculated points along each orbit, created by the ‘create_orbit_paths‘ method.
3) **Lighting and Camera**: A primary light source represents the Sun, with ambient lighting configured via VTK. The standard VTK interactor allows user-controlled zoom, pan, and rotation. The ‘focus_camera_on_planet‘ method centers the view on a selected body.
4) **Axial Tilt**: Visualized by rotating the celestial body's actor and a separate axis actor (‘add_celestial_body‘, ‘update_actor_position‘).
5) **Background**: Starry background is implemented using a big textured sphere (‘add_stars_background‘).

C. User Interface (UI)

PyQt5 [5] is used as the GUI framework.

1) **Main Window**: The UI (‘SolarSystemApp‘ class) integrates the VTK render window with control panels.
2) **Interaction Design**:
   - **Planet Selection**: A dropdown menu allows users to select a celestial body, which becomes the focus of the camera and the subject of the information display panel.
   - **Time Controls**: A slider, play/pause buttons, simulation speed selection, a reset button, and a date display allow users to control the flow of time in the simulation.
   - **Information Display**: A ‘QTextEdit‘ area shows physical and orbital data for the selected planet, along with current physics parameter multipliers.
   - **Physics Parameters**: A dedicated group box contains ‘QDoubleSpinBox‘ controls for Sun Mass, G, Eccentricity, and Inclination multipliers. ”Apply Physics Changes” and ”Reset Physics”.

D. Implementation Details and Challenges

Key implementation aspects include the robust propagation of orbital elements and the iterative solution of Kepler's equation. Dynamic updates to VTK actors ensure that changes in position, Sun size, and orbital paths are reflected promptly. A significant challenge was managing the application of user-defined physics modifications, ensuring that orbital elements are correctly updated and that the simulation remains stable, especially when extreme parameter values are chosen. Handling coordinate transformations and ensuring the Moon's geocentric orbit correctly translates to the heliocentric view also required careful implementation. Furthermore, an attempt to visualize the Sun's gravitational field using glyphs presented difficulties accurately representing the field's strength and direction in a clear and intuitive visual manner proved challenging with the chosen methods, leading to this feature not being included in the final interactive system.

IV. Results

System provides an interactive and visual for exploring orbital mechanics.

A. Final Visualization Capabilities

Users can observe a 3D representation of the solar system, including the Sun, eight planets, and Earth's Moon. The user interface (described as Figure 3) provides comprehensive controls for time, focus, and physics parameters.

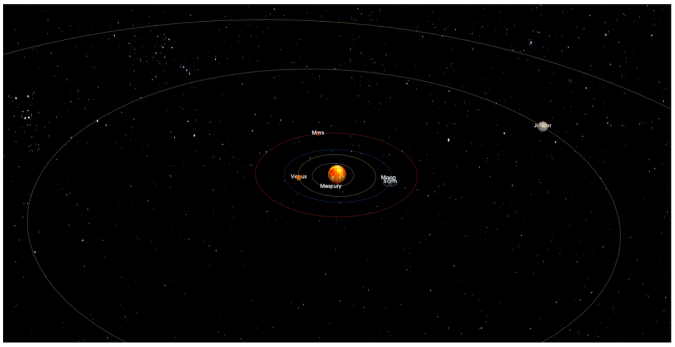

Fig. 1. Wide-angle view of the inner solar system.

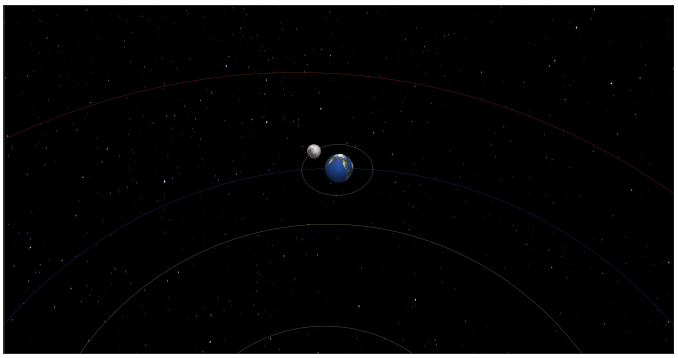

Fig. 2. Focused view of the Earth-Moon system.

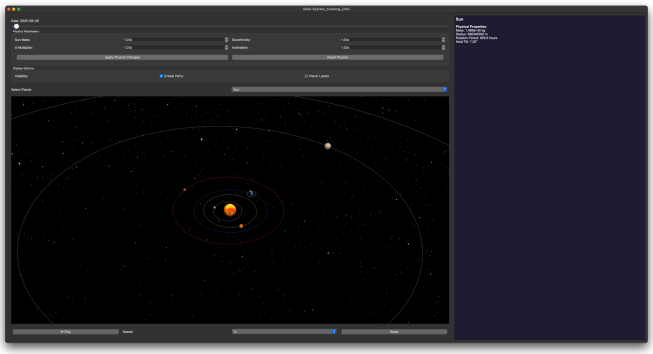

Fig. 3. User Interface showing controls and visualization.

A core result is the system's ability to demonstrate the immediate impact of altering fundamental physical constants and orbital elements. For example:

- **Modifying Sun's Mass**: Increasing the 'sun_mass_scale' multiplier results in planets adopting smaller, faster orbits. The Sun's visual representation also enlarges.
- **Modifying Eccentricity**: Increasing the 'ecc_multiplier' for a planet results in a visibly more elliptical orbit.
- **Modifying Gravitational Constant (G)**: Altering G has a profound impact on the gravitational forces and thus orbital sizes and periods.
- **Modifying Inclination**: Changing the 'inc_multiplier' alters the tilt of a planet's orbital plane relative to the ecliptic.

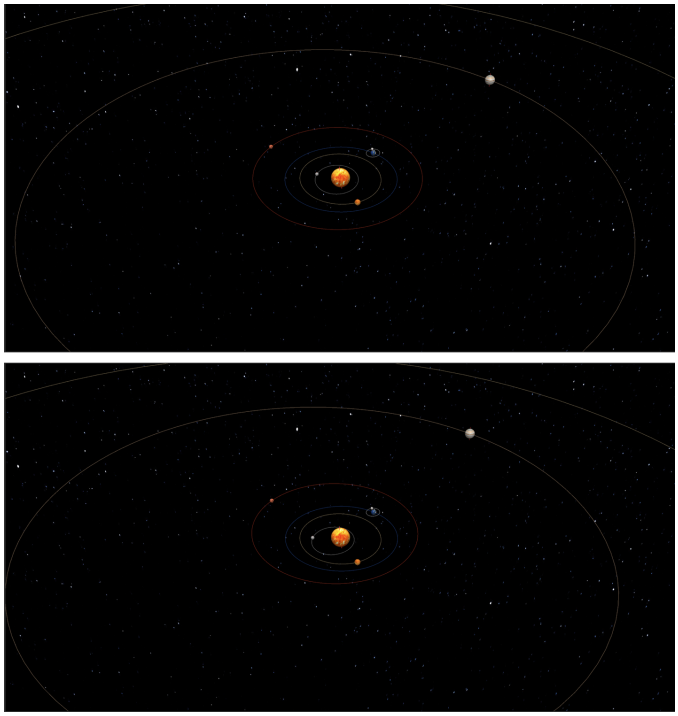

Fig. 4. Effect of physics parameter modification (Increased Sun mass and increased eccentricity)

B. Interactive Exploration

The system effectively meets its goals of interactive exploration:

- **Rendering and Propagation**: Accurate Keplerian orbits are rendered and propagated over time.
- **Time Control**: Users can control the simulation's pace and specific date.
- **Flexible Focus**: Easy selection of celestial bodies for focused observation and data display.
- **Parameter Modification**: The "Physics Parameters" panel allows direct experimentation, with changes reflected instantly in the visualization. This interactive feedback is particularly valuable for understanding the relationships between parameters and orbital geometry/dynamics.
- **Representation**: Planets are textured, and their axial tilts are visualized, adding to the realism and educational value.

The visualizations clearly demonstrate cause and effect. When a user increases the Sun's mass they immediately see the planet

orbits shrink and their orbital speeds increase reinforcing the concept of gravitational influence. Similarly, adjusting eccentricity provides a clear visual understanding of how this parameter shapes an orbit from nearly circular to highly elliptical.

V. Discussion

A. Strengths and Achievements

The primary strength is that system lies in its educational efficacy through interactivity. It successfully translates abstract orbital mechanics principles into a tangible, explorable 3D environment.

- **Educational Impact**: The ability to directly manipulate parameters like M_{sun}, G, e, and i and observe the immediate consequences provides a powerful learning experience that surpasses static diagrams or non-interactive simulations.
- **Keplerian Clarity**: By focusing on the Keplerian two-body problem the system clearly demonstrates orbital characteristics without the complexities of N-body perturbations.
- **User-Interface**: The PyQt5 GUI offers intuitive controls for time, focus, and parameter modification.
- **Modular Design**: The Python-based implementation with UI, VTK setup, and orbital calculations facilitates understanding and potential future extensions.
- **Accuracy**: Within the Keplerian model, the system accurately propagates orbits based on established formulas [1] and data [2], [3].

B. Weaknesses and Limitations

- **Keplerian Model Simplification**: The system does not account for N-body interactions or perturbations which are significant in the real solar system over long timescales. This was a deliberate choice for educational clarity but limits its use for high-precision scientific simulation.
- **Stability with Extreme Parameters**: Allowing users to freely modify fundamental constants can lead to extreme or unstable orbital configurations. While some caps are in place (for eccentricity), further robustness could be added.
- **Visual**: While functional and clear, the visual realism (atmospheric effects, shadows).

C. Future Work

1) **Enhanced Visual**: Incorporating more realistic planetary textures, atmospheric scattering effects, and dynamic inter-body shadows.
2) **Advanced Physics Models**: Optionally introducing simplified N-body perturbation models or allowing the import/visualization of data from more advanced simulators like REBOUND.
3) **Performance Optimization**: Exploring GPU acceleration for rendering orbital paths.
4) **Broader Data Support**: Allowing users to import orbital elements for other celestial bodies (e.g., asteroids, comets) or custom scenarios.
5) **WebGL Port**: Migrating the visualization to a web-based platform using WebGL could significantly increase accessibility.

VI. Conclusion

This paper presented an interactive 3D orbital visualization system, designed primarily as an educational tool. By allowing users to not only observe pre-defined solar system orbits but also to actively modify fundamental physical parameters (Sun's mass, G) and orbital characteristics (eccentricity, inclination), the system provides a unique, hands-on approach to learning orbital mechanics. The immediate visual feedback on parameter changes is a key contribution fostering an intuitive grasp of complex gravitational principles. While currently based on a Keplerian model, the system successfully demonstrates core concepts and offers a solid foundation for future enhancements in visual fidelity and physics modeling, interactive visualization in science education.

References

[1] P. Schlyter, "Computing planetary positions - a tutorial with worked examples,"

[2] Jet Propulsion Laboratory (JPL), NASA, "Solar System Dynamics." [Online]. Available: https://ssd.jpl.nasa.gov/?bodies#elem

[3] D. R. Williams, "Planetary Fact Sheet - Metric," NASA Goddard Space Flight Center.

[4] Solar System Scope, "High-resolution textures." [Online]. Available: https://www.solarsystemscope.com/textures/

[5] Riverbank Computing Limited, "PyQt5." [Online]. Available: https://www.riverbankcomputing.com/static/Docs/PyQt5//