Managing and Visualizing Spatial Information of Small Solar-System Bodies by Using WebGIS- and VR-Based Technologies

Zhi Yin *, Jingsheng Zhang , Junsheng Liu , Weiwei Zhou , Mingyao Ji , Hao Yang

Posted Date: 9 March 2026

doi: 10.20944/preprints202603.0556.v1

Keywords: small celestial body; spatial information; visualization; WebGIS; VR

 doi:10.20944/preprints202603.0556.v1

Article

Managing and Visualizing Spatial Information of Small Solar-System Bodies by Using WebGIS- and VR-Based Technologies

Zhi Yin [1,*], Jingsheng Zhang [1], Junsheng Liu [1], Weiwei Zhou [1], Mingyao Ji [2] and Hao Yang [1]

[1] Department of Geomatics Engineering, School of Marine Technology and Geomatics, Jiangsu Ocean University, Lianyungang 222005, Jiangsu, China

[2] School of Ocean Engineering, Jiangsu Ocean University, Lianyungang, China

* Correspondence: zyin@jou.edu.cn

Abstract

Rapid advances in deep-space exploration are drawing increasing attention from geode-sists and creating an urgent demand for effective management and visualization of deep-space object information. This paper presents a comprehensive study on methods for managing and visualizing small celestial bodies' "orbit–shape–field" spatial information, including geometric shape, orbital simulation and gravitational fields. We develop an in-tegrated solution that combines a backend spatial database, a web frontend and a virtual reality (VR) frontend. On the backend, we design and implement a database to catalog heterogeneous small-body information efficiently, where partitioned tables are adopted to support scalable storage, fast querying and convenient updates. On the web frontend, Spacekit.js and Cesium.js are integrated to simulate coordinated solar-system motion of planets, comets and asteroids (both rotation and revolution) while enabling rapid loading and rendering of gravitational vector fields. On the VR frontend, we build a standalone Unreal Engine 5 application that renders orbits with spline curves consistent with the web results and supports cooperative multi-body motion in immersive exploration. Finally, performance tests are conducted to recommend VR hardware configurations and practical data-loading scales for smooth gravitational-vector rendering and interaction. These methods support analysis workflows and have potential value for deep-space science.

Keywords: small celestial body; spatial information; visualization; WebGIS; VR

1. Introduction

Deep space exploration is a crucial means for humanity to explore the universe. In recent years, with technological breakthroughs and international cooperation, it has entered a phase of rapid development. Major spacefaring nations/agencies have invested significant resources in deep space exploration, undertaking challenging missions to the Moon, Mars, Jupiter, and small solar-system celestial bodies (including asteroids and comets), providing valuable data for enhancing humanity's understanding of the universe [1]. Among them, small celestial body exploration is particularly challenging [2]. In 2004, the European Space Agency (ESA) launched the Rosetta spacecraft to Comet 67P, marking the first-ever landing of an artificial probe on an extraterrestrial small celestial body [3]. This sparked an international wave of small celestial body exploration, including National Aeronautics and Space Administration (NASA)'s OSIRIS-REx mission (target: Asteroid Bennu) [4], Japan Aerospace Exploration Agency (JAXA)'s Hayabusa mission (target: Asteroid Itokawa) and Hayabusa2 mission (target: Asteroid Ryugu) [5,6], as well as the China National Space Administration (CNSA)'s Tianwen-2 mission (target: Asteroid HO3) [7].

Deep space exploration is a systematic scientific engineering field that integrates theory and technology, encompassing astronomy, computer science, systems engineering, control engineering,

 doi:10.20944/preprints202603.0556.v1

navigation engineering and more [8,9]. In practice, deep space exploration technology is closely related to the establishment and maintenance of the deep-space spatiotemporal reference frame [10]. For example, navigation and control of spacecraft, including flybys, orbit insertion, and landing, rely on the efficient management and convenient analysis of spatial information of the target celestial body [11]. The spatial information of the target body primarily includes its geometric shape and gravitational field models, both of which are core research topics in deep space geodesy [12]. Deep Space Geodesy is an extension of modern Space Geodesy, which applies geodetic principles and methods to deep space beyond Earth. It is an interdisciplinary science aimed at achieving high-precision measurement and study of parameters such as the position, motion, and gravitational fields of celestial bodies [13]. Currently, an increasing number of deep space geodesists are attempting to conduct interdisciplinary research combining geodesy and deep space science, in order to establish deep--space control networks [14], solve celestial bodies' gravitational fields [15,16],, and invert for planets' internal structures [17].

Although deep space science is developing rapidly, its high entry barriers have, to some extent, hindered the research efficiency of deep space geodesists. The high barriers in deep space science are mainly reflected in two aspects. On the one hand, deep space exploration missions are characterized by technical difficulties and high costs, and the observational data are far fewer compared to those in Earth sciences [18]. As a result, the research outcomes, such as geometric models, gravitational field models, and dynamic parameters, are scattered across the scientific community, making it difficult for geodesists—especially those new to the field—to easily access deep-space data and models [19]. On the other hand, research data/outcomes related to deep-space science are generally of spatial information (related to celestial bodies' positions), such as the force state, geometric shape, surface temperature, rotation parameters, orbital properties, etc. Moreover, these spatial properties are usually varying over time. Studying these time-varying spatial properties requires an intuitive imagination or visualization ability of the coordinated motion of celestial bodies, which can be particularly challenging for a newcomer to the field of deep space exploration [20,21]. Currently, there are a few of publicly accessible, science-communication-oriented websites for visualizing solar system bodies (e.g., NASA's "Eyes on Asteroids" [22] and ESA's "Orbit Visualization Tool" [23]), but they are insufficient to meet the specialized research needs of deep space geodesists (e.g., displaying deep space remote sensing data, plotting the gravitational fields of celestial bodies, etc.). Furthermore, visualization methods based on flat web pages are limited by terminal hardware (i.e., flat browsers) and lack immersive experience effects [24] [25]. How to develop more enriched visualization methods for deep space exploration remains a topic worthy of further investigation.

In summary, deep space exploration technology has become a research hotspot. However, for newcomer (especially deep space geodesists), it is inconvenient to access, manage, and analyze the spatial data of deep space celestial bodies, as well as the multi-body coordinated motions. Hence, there lacks a deep space celestial body spatial data management and visualization platform, dedicated for deep-space geodesists and tailored for research purpose. To address this, we develop a Small Celestial Bodies' Spatial-data Management and Visualization System (referred to as "the System" hereafter), focusing on the research of spatial information management and visualization methods for deep space objects (mainly small celestial bodies in the solar system), and design an integrated solution for both frontend and backend. The content of this paper is arranged as follows. First, the comprehensive solution for the spatial data management and visualization of deep space bodies is presented, including an introduction to the overall architecture of the System, the data and models required for development, a description of the expected functionalities, and the proposed technical approach (Section 2). Next, the design and implementation of the backend database and frontend functionalities are detailed, with the frontend encompassing both Web-based and VR-based technical solutions (Section 3). Then, the System's performance is tested including the geometric consistency between celestial orbits obtained through spline interpolation on the VR frontend and those drawn using the component Spacekit.js on the Web frontend, and on rendering performance to

 doi:10.20944/preprints202603.0556.v1

determine the optimal data scale for smoothly loading gravity vectors on the VR frontend (Section 4). Finally, the research is discussed and summarized (Section 5).

2. Data and Methods

2.1. Overall Architecture

The system adopts a three-tier architecture of "Data Tier, Service Tier and Presentation Tier" (as shown in Figure 1), and is functionally divided into four modules of "data management, data services, Web visualization and VR visualization". The three-tier architecture achieves decoupling through interface contracts as boundaries. The main functions of each tier are as follows: the data tier is responsible for organizing and managing multi-category data of small celestial bodies; the service tier provides unified data access and control interfaces; the presentation tier forms interactive visual analytics of the "orbit-shape-field" spatial information of small celestial bodies, based on different types of terminal devices.

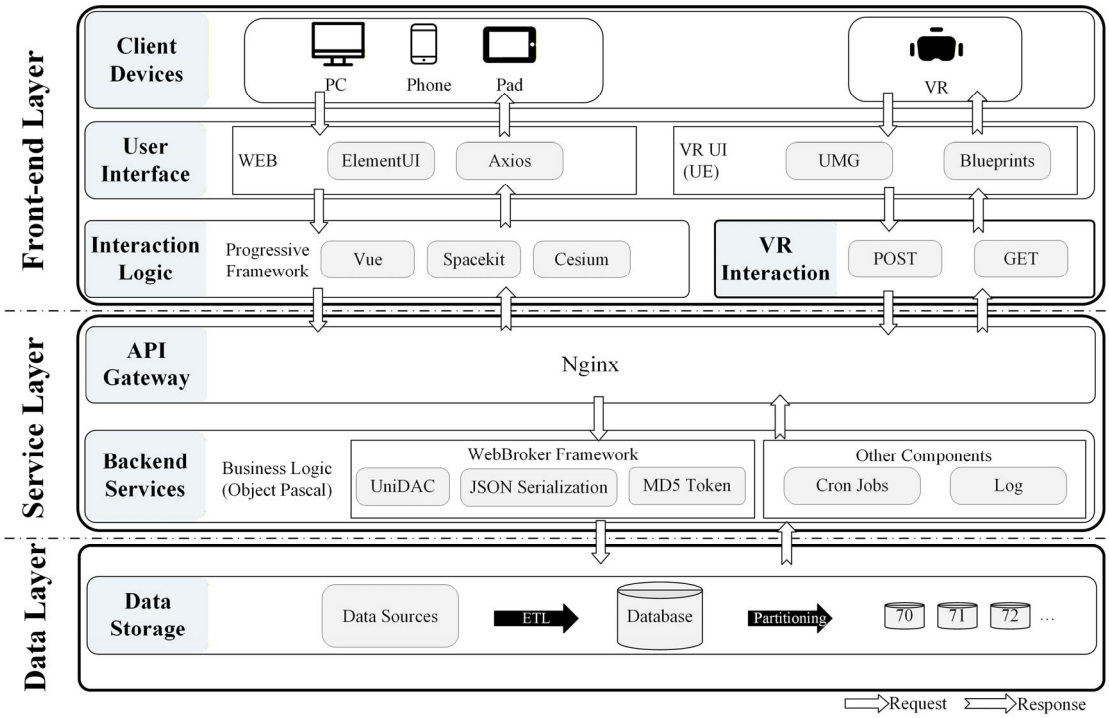

Figure 1. Architecture of the System.

The data tier is primarily responsible for storing and managing the spatial data of small celestial bodies. The number of small celestial bodies in the solar system is large, and their attribute information is extensive, with the total number of discovered asteroids continuing to grow over time. Therefore, the small celestial body database needs to be updated regularly. The data tier first updates the data by extracting, cleaning, structuring, and organizing upstream data (the data source is listed in Table 1). It then improves the data management efficiency through mechanisms such as partitioning, indexing, and views. In this way, the data tier provides stable data support for the upper-tier services. The service tier, built on top of the data tier, exposes standardized REST interfaces, handling common capabilities such as request routing, parameter validation, access control and runtime status tracking, allowing Web/VR clients to access small celestial body data in a unified manner.

The presentation tier provides two visualization solutions, respectively for the web- and VR-based environments. The Web frontend focuses on cross-platform fast access, retrieval and browsing,

 doi:10.20944/preprints202603.0556.v1

while the VR frontend emphasizes immersive (first-person perspective) spatiotemporal experience. Both frontends share the same backend, accessing small celestial body spatial information through the unified data interface provided by the backend. In this way, the data consistency can be ensured when displaying small celestial bodies' geometric elements and physical fields (i.e., the gravitational field in this study) across different terminal devices.

2.2. Data and Models

In the system backend, the data storage layer is primarily responsible for the storage and management of small celestial body spatial information data. The small celestial body attribute data (see Table 1) is sourced from NASA/JPL's Small-Body Database (SBDB). This database is maintained by the Solar System Dynamics (SSD) team and covers a wide range of data, including orbital parameters, physical parameters, naming and discovery information, Near-Earth approaching records, radar observations, and virtual impactors of the asteroids and comets in the solar system. Although the SBDB is comprehensive, it has some limitations, such as a large number of attributes, high update frequency, and complex query patterns. The geometric models of small celestial bodies are mainly sourced from public research websites. For example, the 3D geometric model of Comet 67P is obtained from ESA's Rosetta project (URL: https://sci.esa.int/web/rosetta/; last accessed on January 1, 2026). The gravitational field models of small celestial bodies are obtained from existing researches. For instance, the gravitational field model of Comet 67P comes from the paper previously published by one co-author of this paper, Zhi YIN [26]. The gravitational models are stored in CSV format file, with the data structure of the starting coordinates, vector components and vector magnitude. Such data can be converted into arrow vectors for rendering on the Web/VR frontend. Based on data type and scale, this study selects the software PostgreSQL for the design, construction and maintenance of the object-relational database (see Table 2) [27]. By establishing table partitions, indexes and materialized views, the System efficiently manages large-scale spatial data, providing stable support for the System's service layer (see Figure 1).

Table 1. Main data and models of this study.

Name	Type	Source
Small-Body Database (SBDB)	Attribute data of small bodies	NASA JPL [28]
Horizons ephemeris service	Small-body ephemerides	NASA JPL Horizons [29]
Publicly accessible scientific data repositories	Small-body 3D geometric/shape models	Public 3D model repositories [30]
Small-body force vector-field model (CSV format)	Gravitational-field model	Adopted from the authors' previous study [26,31]

Table 2. Main development software and components of this study.

Software/Framework	Purpose
PostgreSQL	Database design and development
Object Pascal+WebBroker	Backend server development
PrimeVue	Web front-end development
Spacekit	Web-based orbit visualization
Cesium	Web-based field-data visualization
Unreal Engine 5.5	VR front-end development (including scene interaction, orbit rendering, model loading, etc.)

2.3. Key Technical Approach

The system development technical approach can be organized as four steps: (1) data acquisition and update, (2) structured storage and optimization, (3) service encapsulation and interface standardization, and (4) frontend visualization (Figure 1).

In the data acquisition phase, the upstream data from NASA/JPL is transformed into locally structured tables through the ETL (Extract-Transform-Load) process. Additionally, a batch

 doi:10.20944/preprints202603.0556.v1

processing mode and scheduled task mechanism are employed to achieve real-time updates of the local database.

In the data ingestion phase, due to the one-to-many mapping relationship between one small celestial body and its attributes, we adopt the strategy of data partitioning and indexing to optimize the data structure, thereby to meet potential high-concurrency data access. The partition-table technique, using hash partitioning, has the advantage of faster query speeds compared to conventional table query methods. This is one of the key technologies enabling the efficient management of the large-scale small celestial body spatial data [32].

In the service encapsulation phase, the Object Pascal programming language is used to build HTTP/REST services based on the WebBroker (Delphi Web Service Framework); efficient access to the PostgreSQL database is achieved through the UniDAC data access component library. The interface layer uses JSON as the data exchange format and provides a unified HTTP/JSON API. To ensure the security and stability of the interface, an authentication mechanism based on request signatures (referred to as MD5 Token), logging and status tracking components are introduced. Additionally, scheduled tasks are handled using the Cron class to maintain data updates and system operation. Furthermore, Nginx is employed as a reverse proxy and load balancer to provide unified access to backend services, presenting a single and stable REST access point to external clients.

In the frontend visualization phase, the Web frontend uses PrimeVue to develop a component-based interface, routing and interaction features, integrating Spacekit (a 3D orbital/astronomical visualization library) and Cesium (a 3D WebGL visualization engine) to achieve multi-body orbital plotting, 3D model loading and multi-scale spatial browsing. The VR frontend, based on Unreal Engine 5, uses spline curves to render celestial bodies' orbits and simulates celestial bodies' motion. Additionally, it renders the gravitational vector field around celestial bodies with 3D arrows, providing an immersive observation and interaction experience for the user. The VR frontend operates on Meta Quest 2 helmet, with the pipeline running on OpenXR.

3. Realization of System

3.1. Realization: Backend Database

The main goal of the backend development is to efficiently organize and manage multi-source, heterogeneous spatial data of small celestial bodies, ensuring that the System is able to provide stable data services even under a high access volume. The following content describes the realization details.

The SBDB interface supports retrieval of targets using identifier fields (e.g., sstr, spk, and des), and allows optional data blocks (e.g., orbit, phys-par, ca-data, vi-data, radar-obs, discovery, and model-pars) to be enabled as needed, returning the corresponding subsets of attributes. To ensure the completeness and traceability of the spatial information of small celestial bodies, the data is organized and stored according to a two-level data structure (i.e., celestial body and attribute block), where the data is first identified by the key "spkid" and then subdivided by attribute block. Based on this data structure, we designed nine thematic tables in PostgreSQL (as shown in Figure 2), including: basic information (table: object), physical parameters (table: physical), orbital information (table: orbit), orbital elements (table: orbitelements), orbital model parameters (table: orbitmodel), close approach information (table: closeapproach), virtual impactors (table: virtualimpactor), radar observations (table: radarobs), and discovery information (table: discovery). Each table uses "spkid" as the primary key and the smallest unique attribute within the data block (e.g., name, cd+body) as the composite key to identify and manage the data entry of each small celestial body.

 doi:10.20944/preprints202603.0556.v1

Figure 2. Entity-relationship (ER) diagram of the database.

The system retrieves approximately 39,000 near-Earth small celestial body samples from upstream data source, and this number continues to grow over time. Each small celestial body has the abovementioned nine types of attribute information, and the record volume is large (typically in the millions). For such large-scale spatial data, using a linear single-table expansion method would significantly increase I/O query volume, leading to inefficient data management. Therefore, we apply hash partitioning to the thematic tables, with "spkid" as the primary key. The table partitioning is done with a modulus of 7, logically distributing one table into 7 physical sub-tables (numbered 70–76, as shown in Supplementary File S1). Hash partitioning offers advantages such as load balancing, partition pruning and compatibility with parallel schedulers, which ensures millisecond-to-second response time for queries on millions of records. Furthermore, considering that upstream data of small celestial bodies is updated daily and may require corrections, we employ a daily delta mechanism for data records (see the specific ER diagram in Supplementary File S1). For existing records, conflict resolution is performed at the data layer based on UPSERT semantics, while the batch processing task is executed within a single database transaction. In the event of an error, the entire transaction is rolled back, and job scheduling automatically retries, thereby ensuring consistency between local and upstream data during the daily update process.

The backend data service is realized in Object Pascal, using the WebBroker Web framework and UniDAC data access engine for efficient connection and parameterized queries with PostgreSQL. The service is compiled as an Apache dynamic module, deployed on cloud servers, and runs persistently with the system Apache (see Figure 3). This operational mode is simple and stable, facilitating unified logging, rate limiting and reverse proxy, while using precomputed materialized views and read-only access to offload complex query costs to the data layer, ensuring responsive online service performance. The service provides a read-only HTTP+JSON REST API, where data domains or objects are selected using path and query parameters. The query format is http://geojson.top/zionasteroid/api/data?class=<block>&asteroid=<spkid>, where the <block> option can be filled with "object", "orbit", "orbitelements", "orbitmodel", "physical", "virtualimpactor", "radarobs", "discovery", "closeapproach" or "resource". The API returns JSON text with SBDB-equivalent semantics and unified field names for easy parsing and reuse across various terminals.

 doi:10.20944/preprints202603.0556.v1

The client layer supports both Web and VR frontends. The Web frontend uses Axios for GET/POST requests, while the VR frontend accesses backend data using the same URL format via UE's built-in HTTP module or blueprint nodes, enabling orbital queries, data filtering and VR first-person view rendering.

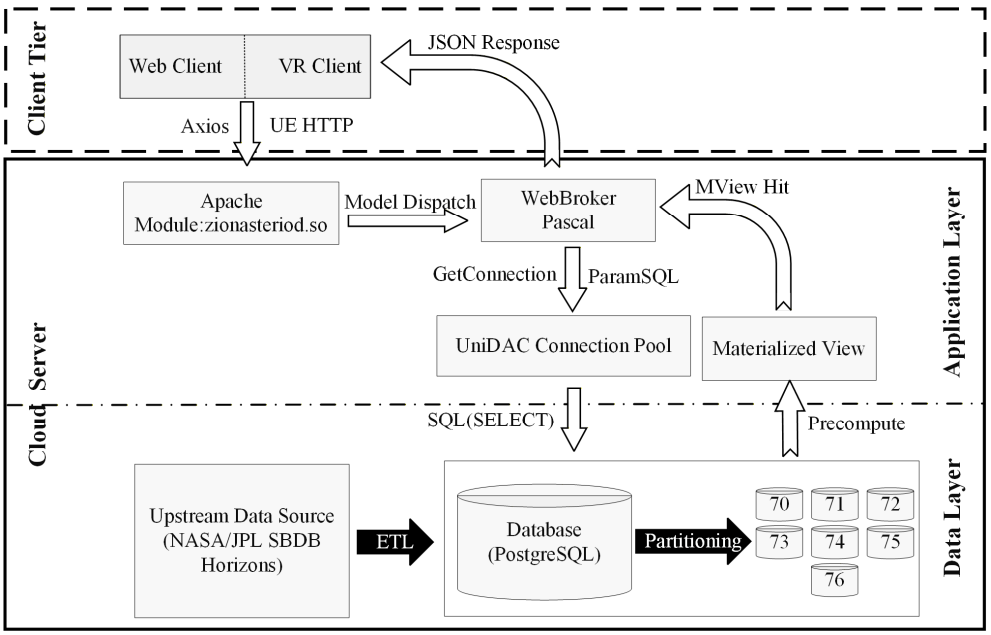

Figure 3. Backend service architecture.

3.2. Realization: Frontend Function

The goal of the frontend development is to build an efficient visualization method and User Interface (UI) interaction mechanism for small celestial bodies' spatial data (i.e., orbit, shape and field data) on frontend devices. To achieve this, we design and realize three basic functions as follows: (1) the geometric shapes and orbits of small celestial bodies in the solar system; (2) the coordinated motion of solar system bodies; (3) the rendering of external gravitational vector fields of small celestial bodies. The VR application developed in this study enables immersive, multi-perspective observation and roaming of the "orbit-shape-field" spatial information and coordinated motion of solar system bodies from a first-person perspective. The following content describes the realization details involved.

3.2.1. Web Client

The Web frontend (URL:http://geojson.top/spacesium/; last accessed on January 1, 2026) follows a single main interface design approach, with the layout divided into three functional areas: small celestial body search, attribute browsing, and 3D view (as shown in Figure 4). The top navigation bar (marked with ① in Figure 4; the similar notation rule is used in the subsequent content) provides global functions such as system introduction, usage instructions, quick access to Near-Earth asteroids, asteroid search, and language switching. The right-side toolbar (②) integrates buttons for 3D scene controls such as layer display, zoom, and pause (③). The left panel (④) displays the attribute information for the small celestial body selected. The bottom timeline component (⑤) controls the real-time/fast-forward parameters for simulating celestial bodies' coordinated motion. The central 3D main view focuses on displaying the position, orbit (⑥), and gravitational field (⑦) of celestial bodies. This interface is developed using the UI component library PrimeVue, ensuring that the system layout (e.g., navigation bar, list, attribute panel, timeline, 3D view, etc.) adapts to terminal browsers with different resolution settings.

 doi:10.20944/preprints202603.0556.v1

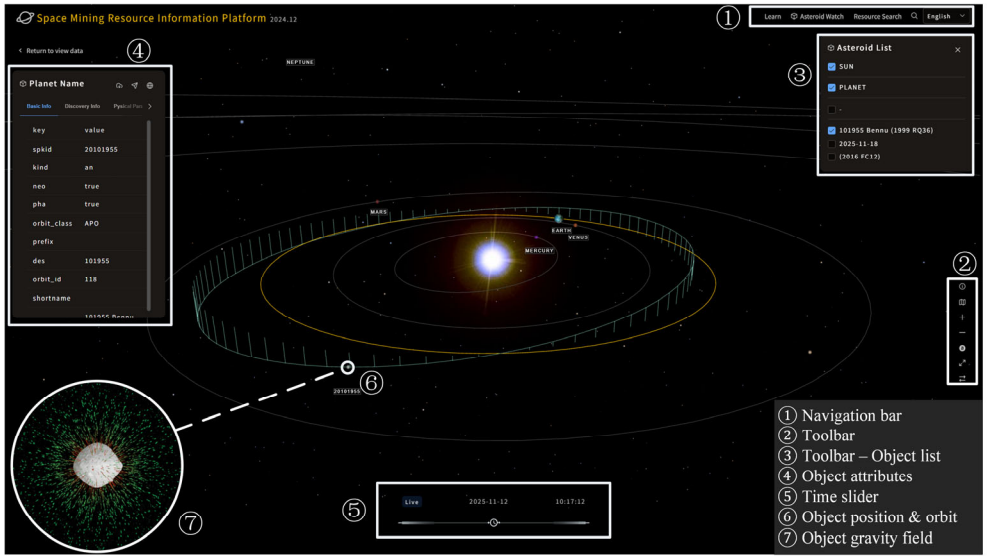

Figure 4. Web client interface of the System.

After entering the system webpage, users can perform precise searches for a small celestial body using the key “spkid” or name in the top search bar. Additionally, users can add the small celestial bodies of interest to the list on the right (③). The celestial body list supports toggling the display/hiding of individual or multiple bodies, as well as multi-selection for “currently selected celestial bodies”. The spatial elements and information of the selected celestial body are simultaneously displayed in other function panels, making it easier to do spatial analysis.. For example, when an asteroid is selected from the list, its basic information, physical parameters, and orbital elements are displayed in the left-side attribute panel (④). As shown, the entire interaction process follows the sequence "search, filter and attribute browsing”, providing multi-dimensional semantic understanding and spatial analysis support for the 3D visualization of celestial bodies’ orbits, geometric models and gravitational fields.

Regarding the orbit visualization, the System’s frontend first retrieves the six orbital parameters of the selected celestial body from the backend database. Then, based on the principle of orbital dynamics, the System uses the component library Spacekit to plot celestial bodies’ orbits as well as their coordinated motion effect. After selecting different celestial body targets from the list (③), the corresponding orbits are highlighted in the main view (⑥), allowing users to visually compare the orbital characteristics of different bodies (e.g., orbital shape, size, inclination, etc.). By dragging the timeline (⑤) or adjusting the playback speed, the relative positions of the bodies at different time points can be shown, providing an intuitive reference for orbit analysis.

For the gravitational field visualization aspect, we designed an independent sub-view entry of gravitational field in the attribute panel on the Web frontend. After clicking the gravitational field button at the top-right corner of the webpage (④), a separate window will pop up, specially displaying the 3D geometric model of the celestial body and the surrounding gravitational vector distribution (⑦), synchronized with the main view. The gravitational vector data in the sub-window is retrieved by using the backend API. After gravitational vectors rendered as arrows, the starting point of the vectors represents the sampling location of the gravitational vector, the direction of the vector represents the gravitational direction, and the length of the vector is proportional to the magnitude of the gravitational vector. The larger the gravitational force, the longer the drawn vector.

3.2.2. VR Client

The VR frontend uses UE 5.0 to create an immersive scene. UE 5.0 retrieves data by using the backend API, and visualizes celestial elements (such as the geometric models, orbits, gravitational

 doi:10.20944/preprints202603.0556.v1

fields, etc.) through Blueprints. The interface follows the design idea of "dual-scene + floating control panel", and the interaction process is structured in the order of "solar system overview", "target celestial body selection", "scene switching" and "close-up gravity display". Considering that the size of one small celestial body is much smaller than the spatial scale of the real solar system, it is difficult to simultaneously perceive both the overall solar system and the local details of small celestial bodies, within one VR field of view. Therefore, we use different scales to render geometric elements in the overview scene of the solar system and the local scene of the small celestial body. Specifically, in the overview, a smaller scale is used to display the orbital distribution of solar system bodies (including planets and small celestial bodies), making it easier for users to compare the relative positions and hierarchical structure of all orbits. In the local view of a selected celestial body, a larger scale is used to render spatial elements centered on the body, such as its geometric shape and gravitational field.

After opening the application on the VR frontend, it defaults to the main interface of the spacecraft (Figure 5a). The main screen provides two entry buttons, "Begin" and "Real Life", for entering the system or exiting the application, respectively. The auxiliary screens on the left and right sides display instructions for the VR controller operation. Clicking the button "Begin" enters the solar system overview scene by default (Figure 5b). As shown in Figure 5, using the VR controller, a floating UI panel can be brought up in the user's view, which mainly includes two functions: celestial body selection (①) and simulation control (②-⑦). The celestial body selection area uses a horizontal scrolling control to display the celestial body currently selected (display format: thumbnail + name; see ① and ②). The simulation control area provides a set of basic browsing control functions based on the celestial body selected, such as toggling the display/hide of orbits (③ and ⑤), and controlling the play/pause and speed adjustment of the solar-system bodies' coordinated motion (④). Additionally, the main interface includes a button for switching the scenes "Solar System Overview" and "Small Celestial Body Close-up" (⑥), as well as a 'Return to Spacecraft' button (⑦).

Figure 5. VR user interface: entry menu and interactive control panel.

In the VR frontend, considering that there is no component library for celestial motion visualization (like Spacekit for developing the Web frontend) available in the VR frontend, we firstly obtain the ephemerides of each celestial body and then visualize the orbits in the VR device by drawing spline curves (Figure 6: a1 and a2). To simulate the coordinated motion of celestial bodies

 doi:10.20944/preprints202603.0556.v1

along the orbits, a unified time system is needed. We use the timeline (Timeline) component, combined with ephemeris data, to map the periodic heliocentric motion (orbital revolution) of each celestial body to a normalized time range of [0,1). In this way, it provides a normalized ephemeris for each celestial body. When simulating the coordinated motion of all bodies, the position of each body at each time frame (corresponding to a point on the normalized time axis) is interpolated based on their normalized ephemeris (see schematic in Supplementary File S2). This allows for the unified control of the positions of all solar system bodies according to the normalized time parameter, simulating their coordinated heliocentric motion

In the near-field scene, after selecting the celestial body of interest, the user can switch to the detailed view of the selected body (e.g., geometric model, gravitational vectors) with a single click (Figure 6b2). In addition, we extend the built-in VR free-roaming capability and design the interaction scheme to balance naturalness and basic operations (e.g., movement, rotation and zoom), thereby maintaining usability during immersive spatial navigation [33]. First, the official VR roaming feature is enhanced by replacing the VRPawn with a Character that has a capsule collider. Then, using Blueprint scripting, the controller joystick (physical hardware) is mapped to translate and elevate the viewpoint (a software functionality). Finally, by applying frame-by-frame Camera-Capsule consistency constraints, we achieve a first-person free-roaming effect with collision avoidance (see schematic in Supplementary File S3) [34].

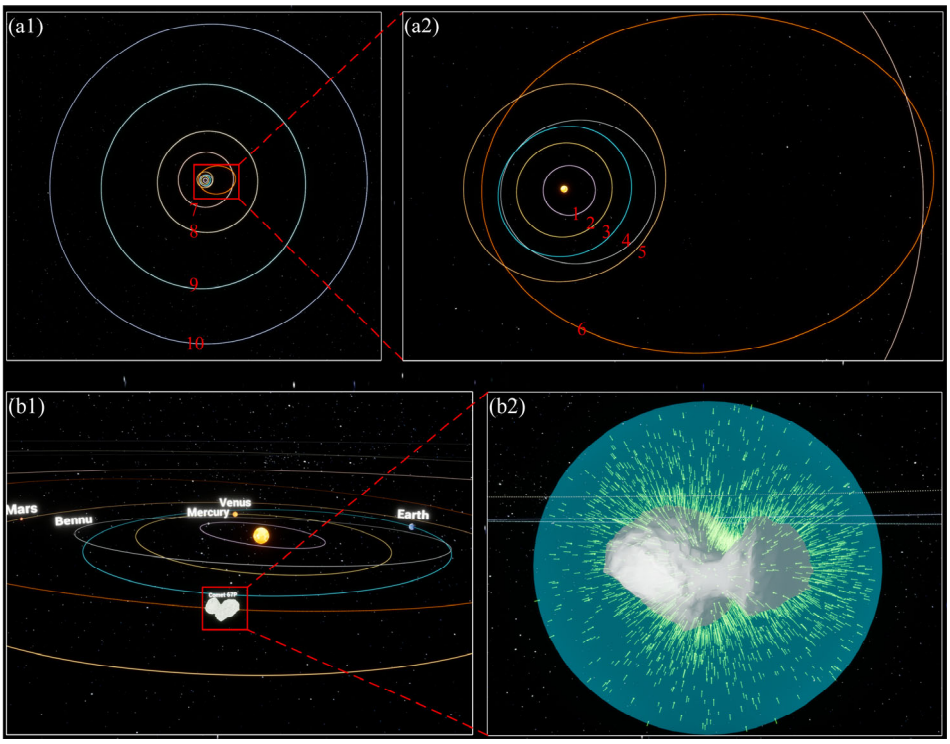

Figure 6. Rendering effect for orbits and small-body gravity vectors. In panel a2, labels 1-6 correspond to the orbits of Mercury, Venus, Earth, Bennu, Mars and Comet 67P, respectively. In panel a1, labels 7-10 correspond to the orbits of Jupiter, Saturn, Uranus and Neptune, respectively. Panel b1 shows the Solar System overview, and panel b2 shows the near-field gravity vectors of Comet 67P.

For the visualization of small celestial bodies' gravitational fields, we represent the gravitational vectors by drawing 3D arrows. Since small celestial bodies usually have irregular shape, using analytical functions (such as spherical harmonic function) to represent their external gravitational fields usually leads to function divergence problem. Therefore, we use numerical gravitational field models of small celestial bodies, drawing 3D arrows to represent the gravitational vectors. In the VR frontend, after reaching the small celestial body of interest, the gravitational field data is first loaded,

 doi:10.20944/preprints202603.0556.v1

and then 3D arrows are drawn near the body, i.e., in the space between the body's surface and a semi-transparent sphere. The center of the sphere is located at the mass center and the radius is approximately 5 times the average size of the small celestial body. Figure 6b2 shows the gravitational field rendering of Comet 67P. These features are realized through UE5 Blueprints, with the core tasks being the drawing of gravitational vector arrows and the semi-transparent sphere. See the corresponding blueprint, please refer to Supplementary File S4.

4. System Performance Test

4.1. Validation of the Orbits in VR Client

The system's architecture can be briefly summarized as "one backend and two frontends", meaning that both (Web and VR) frontends share a common backend (including data and service). Therefore, the spatial elements of celestial bodies, displayed by the two frontends, should be consistent with each other. When simulating celestial bodies' motion, due to the lack of ready-made tools like the Spacekit component used for orbit plotting on the Web end, we plot the orbits in the VR end using spline curves. Next, we test the correctness of the orbit plotting function of the VR end by comparing the orbits from both ends and validate the geometric consistency on orbital position and shape.

As shown in Table 1, the Spacekit of the Web end plots the orbits by calling the Kepler orbital elements provided by SBDB, while the VR end plots the orbits based on the ephemeris released by the Horizons System of NASA. Both the SBDB and the Horizons System databases calculate the orbital elements of each celestial body from the same raw astronomical observation data, but there are slight differences in the realization algorithms (e.g., the number of iterations), resulting in minor numerical differences in the orbital elements provided by these two databases. However, these numerical differences are negligible when verifying orbital consistency.

The steps for plotting celestial body orbits using spline curves in the VR frontend are as follows. First, starting from a unified epoch, the orbit is downsampled according to the orbital period of each celestial body; for example, the orbital period of the Earth is ~365 days, corresponding to 365 sampling points on the periodic orbit. Then, these discrete orbital points, along with the corresponding epochs, are organized into a CSV file and stored in the local backend database. Finally, the VR frontend calls these resampled orbital data through the backend API to plot the spatial spline curves, resulting in the celestial body's orbit. Table 3 lists the orbital resampling parameters for the eight planets in the solar system and two example small celestial bodies. This method simplifies the real orbits into periodic orbits for a specific period, ignoring orbital precession and perturbation effects. This approach avoids frequent calls to the raw ephemeris data, significantly improving the efficiency of celestial orbit plotting. It also avoids substantial orbital deviations, as the real orbital changes of each celestial body are very small. If to account for orbital precession effects, it is simply necessary to change the starting epoch as well as the corresponding orbital parameters, and then redraw the orbit.

Table 3. Orbital sampling parameters of Solar System bodies in the VR frontend.

Body	Orbital Period	Time Span	Step size of Sampling	Amount of Sampling Points
Mercury	88 day	2024-04-25——2024-07-22	1 day	88
Venus	224 day	2024-04-25——2024-12-06	2 day	113
Earth	365 day	2024-04-25——2025-04-25	1 day	365
Mars	687 day	2024-04-25——2026-03-12	2 day	344
Jupiter	11.86 years	2024-04-25——2036-03-27	1 month	144
Saturn	29.5 years	2024-04-25——2053-10-25	2 months	177
Uranus	84 years	2024-04-25——2108-04-25	2 months	504
Neptune	164.8 years	2024-04-25——2189-02-11	2 months	989
Bennu (Asteroid)	1.2 years	2024-04-25——2025-07-07	1 day	439
67P-CG (Comet)	6.42 years	2024-04-25——2030-09-25	1 month	78

 doi:10.20944/preprints202603.0556.v1

To verify the accuracy of the orbits plotted in the VR frontend, we use Asteroid Bennu and Comet 67P as examples. The Kepler orbital elements are retrieved from the celestial orbits plotted in the VR frontend using spline functions, and then compared with those read by the Web-end Spacekit. Table 4 presents the Kepler orbital elements of the orbits of the two small celestial bodies, respectively from SBDB database and NASA's Horizons System. It can be seen that the differences are very small, with relative differences on the order of 1E-10. For the case of Asteroid Bennu, the relative differences in semi-major axis and eccentricity are 5.40×10^{-12} and 4.38×10^{-12}, respectively, and the largest difference in angular elements, the argument of perihelion, is only 8.26×10^{-10}°, much smaller than the angular second scale (1″ ≈ 2.78×10^{-4} °). Similarly, the case of Comet 67P shows comparable results. In conclusion, although the VR frontend uses a different orbital plotting method, the orbital position and shape remain consistent with those of the Web frontend, thus verifying the correctness of the spline-based orbital plotting method in the VR frontend.

Table 4. Comparison of orbital elements between the Web and VR clients (Bennu and 67P as examples).

Object	Keplerian element	Web client (Data source: SBDB database)	VR client (Data source: NASA Horizons System)	Difference ($\times10^{-13}$)	Relative difference ($\times10^{-10}$)
Bennu	epoch	2455562.5	2455562.5	/	/
	a(AU)	1.126391025894812	1.126391025888730	60.82	0.053995
	e	0.2037450762416414	0. 2037450762425329	8.915	0.043756
	i(deg)	6.03494377024794	6.034943770255041	71.01	0.011766
	Ω(deg)	2.06086619569642	2.060866195707317	108.97	0.052876
	ω(deg)	66.22306084084298	66.22306084001706	8259.2	0.124718
Comet 67P	epoch	2457305.5	2457305.5	/	/
	a(AU)	3.462249489765068	3.462249489765064	0.04	0.000011553
	e	0.6409081306555051	0.6409081306555049	0.002	0.000003121
	i(deg)	7.040294906760007	7.040294906760015	0.08	0.000011363
	Ω(deg)	50.13557380441372	50.13557380441362	1.0	0.000019946
	ω(deg)	12.79824973415729	12.79824973415736	0.7	0.000054695

4.2. Efficiency of the Gravitational Vector Rendering in VR Client

As a spatial information visualization system, the 3D rendering efficiency of spatial information elements depends on both the scale of the spatial data and the rendering efficiency of the frontend device. The largest dataset stored in the backend of the System is the small celestial body's gravitational field model. When loaded onto the frontend device, the gravitational vectors need to be downsampled. It is necessary to test the rendering efficiency of the gravitational field vectors on the frontend, particularly the VR frontend, to determine the optimal data loading scale that ensures smooth rendering. It needs to find a balance between visualization quality and the 3D gravitational vector rendering speed represented by the downsampling ratio for gravitational vectors.

To quantify the VR rendering capability, we continuously collect 1800 frames of gravitational vectors for nine different data sizes in the PC-VR mode of the VR device Meta Quest 2 (72Hz), using a fixed interaction script and unified testing viewpoint. The test is conducted on a PC (CPU: Intel Core i9-14900HX; GPU: NVIDIA GeForce RTX 5060 Laptop; Memory: 32GB; Operating system: Windows 11). The frame rate is set to 13.89 ms for 72Hz, with 27.78 ms (=13.89 ms * 2) serving as the stuttering threshold.

The evaluation metrics include: (1) the average value of the frame time per 1000 frames (AvgFPS; unit: fps); (2) the 99th percentile of frame time (FT_P99; unit: ms), typically used to estimate tail latency; (3) the percentage of frames with a frame time greater than 27.78 ms (HitchRate); (4) the average time spent per frame on the game thread (GT; unit: ms), reflecting CPU load; (5) the average time spent per frame on the render thread (RT; unit: ms), reflecting CPU load; (6) the average time spent per frame executing commands on the GPU (GPU; unit: ms), reflecting GPU load. When the metrics GT, RT and GPU are all present, the frame time tends to show a maximum value. These

 doi:10.20944/preprints202603.0556.v1

metrics characterize the impact of changes in gravitational vector rendering scale on smoothness and thread computation load, under the given hardware and refresh rate.

The results in Table 5 show that, as the gravitational vector scale increases, the VR 3D rendering performance exhibits a segmented characteristic of stable, critical and unstable. In the stable segment (amount of arrows: 0-3000), the VR end can consistently maintain the target frame rate of 72fps with no stuttering. The indicator GT dominates the frame time (13.9 ms), closely aligning with the frame budget (13.89 ms). Meanwhile, the indicators RT and GPU steadily increase with the number of arrows, but their maximum values always remain below GT. It indicates that the system is in a state of tight balance, with the GT in control. Although the rendering load continues to rise, it has not yet surpassed the performance threshold. At the critical point (amount of arrows: 3200), although the average frame rate remains close to the target, stuttering (0.28% stutter rate) occurs for the first time, and GT clearly exceeds the frame budget. It indicates that the increased rendering load has started to apply substantial backpressure on the game thread, causing the completion time of trailing frames to surpass the stuttering threshold, thereby breaking the stable state of the system. In the unstable segment (amount of arrows: >3200), the system performance drops sharply. The main reason is the significant increase in GT, which greatly surpasses RT and GPU, becoming the system bottleneck. Notably, the RT and GPU increases relatively smoothly, indicating that they have not yet reached their hardware limits. This suggests a CPU bottleneck caused by GT overload, meaning the instance and data required per frame have exceeded the processing capacity of the CPU.

Table 5. Impact of arrow-glyph rendering scale on PC-VR performance.

ID	Arrows	Frames	AvgFPS	FT_P99	HitchRate	GT	RT	GPU
1	0	1800	71.864	14.643	0.00%	13.904	2.412	3.616
2	500	1800	71.861	14.728	0.00%	13.908	3.211	3.883
3	1000	1800	71.872	14.842	0.00%	13.909	4.237	4.055
4	2000	1800	71.890	15.016	0.00%	13.906	5.976	4.35
5	2500	1800	71.914	15.19	0.00%	13.905	7.038	4.606
6	3000	1800	71.901	15.484	0.00%	13.912	8.211	4.778
7	3200	1800	71.742	15.722	0.28%	13.989	8.487	5.226
8	3500	1800	49.723	30.903	37.94%	22.451	10.406	5.82
9	4000	1800	35.922	28.689	60.33%	27.828	9.836	5.05

In a short summary, under the hardware configuration and frame rate condition, specified in this study, the upper limit for the gravitational vector rendering scale that the VR device (Meta Quest 2) can support is 3000 vector arrows, which ensures smooth rendering with a stable experience. The critical capacity for the number of gravitational vectors is 3200, beyond which the rendering starts to become unstable. When the scale reaches ≥3500, the system enters the bottleneck state of CPU load, leading to a significant degradation of the rendering performance.

5. Discussion and Conclusions

In recent years, an increasing number of geodesists stated to focus on and enter the field of deep space science, leading to the emergence of (known as) deep space geodesy. Compared to original geodesy, which focuses on Earth, deep space geodesy, which focuses on deep space objects, involves a much broader spatial scale. It also requires imagination of the relative positions and coordinated motions of the celestial bodies. This introduces higher demands for the management, analysis and visualization of spatial information of deep-space bodies. Due to the lack of a dedicated tool for managing and visualizing sola-system bodies, it is not convenient for newcomer to conduct relevant researches. This has, to some extent, hindered the development of deep space geodesy.

The system developed in this study is expected to alleviate the challenges mentioned above. The system adopts a three-tier architecture (i.e., data tier + service tier + presentation tier) and a three-module functions (i.e., backend + Web frontend + VR frontend). The backend data tier is designed with tables that efficiently manage different attributes of small celestial bodies, forming the spatial

 doi:10.20944/preprints202603.0556.v1

database. This database is consistent with NASA/JPL's SBDB in terms of data volume, containing the most comprehensive spatial information of small celestial bodies and enabling daily updates. Notably, during the database construction, the table partitioning technology is employed to facilitate the rapid retrieval of large-scale spatial data of small celestial bodies. Additionally, user-customizable interfaces are reserved in the database, enabling subsequent storage, management and analysis of user-defined spatial data for small celestial bodies.

The backend database provides standardized interfaces to offer data access service. Visualization features for small celestial bodies' orbit, shapes and gravitational fields are realized on both Web and VR frontends. Although the visualization technologies differ between the two frontends, the geometric elements plotted on the two frontends should be consistent to each other. Therefore, we performed a geometric consistency test on the orbits plotted by both frontends, and the result confirmed the geometric consistency of the two frontends, as well as the correctness of the spline--curve-based orbital plotting method on the VR end. Furthermore, we tested and determined the optimal data loading scale for gravitational field rendering in the VR end. In the PC-VR mode of the Meta Quest 2 device, the 3D rendering effect is smooth and stable when the number of gravitational vector arrows is ≤3000; it starts becoming unstable at around 3200, and deteriorates significantly with CPU overload when the number exceeds 3500.

Taken together, we provide a comprehensive study on the management and visualization method for the spatial information of small celestial bodies, and have developed a corresponding system. However, there still needs improvement in precise orbital plotting and large-scale gravitational vector rendering, and the backend database could be further researched, especially in expanding the storage and management capabilities for user-customized data such as deep-space remote sensing imagery, control networks, DEM data, etc.

Author Contributions: Conceptualization, Zhi Yin and Jingsheng Zhang; Methodology, Zhi Yin and Jingsheng Zhang; Software (system development and functional optimization), Zhi Yin, Jingsheng Zhang and Junsheng Liu; Data curation (data organization, annotation, cleaning and maintenance), Junsheng Liu, Weiwei Zhou, Mingyao Ji and Hao Yang; Validation (code testing), Junsheng Liu, Weiwei Zhou, Mingyao Ji and Hao Yang; Writing—original draft, Zhi Yin and Jingsheng Zhang; Writing—review & editing, all authors.

Funding: This work was funded by the National Natural Science Foundations of China (No. 42204088).

Data Availability Statement: Small-body attribute data used in this study are available from the NASA/JPL Small-Body Database (SBDB) (https://ssd-api.jpl.nasa.gov/doc/sbdb.html), and ephemerides are available from the NASA/JPL Horizons system (https://ssd.jpl.nasa.gov/horizons/). The shape models of the dwarf planet Ceres and the asteroids Bennu and Itokawa can be obtained from the 3D Asteroid Catalogue (https://3d-asteroids.space/) and the shape model of Comet 67P is available from ESA Rosetta mission resources (https://sci.esa.int/web/rosetta/). The gravity-vector dataset used for near-field visualization is derived from the authors' previously published work and is available from the corresponding author upon reasonable request.

Acknowledgments: We thank all contributors to this article and all participants in the experiment for their valuable support and knowing cooperation.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Peeters, W.; Ehrenfreund, P. Charting the Future of Space: A Collaborative Vision for Innovative Commercial Partnerships and Sustainable Space Exploration. *New Space* **2025**, *13*, 7–21, doi:https://doi.org/10.1089/space.2024.0041.
2. Getzandanner, K.M.; Antreasian, P.G.; Moreau, M.C.; Leonard, J.M.; Adam, C.D.; Wibben, D.R.; Berry, K.; Highsmith, D.E.; Lauretta, D.S. Small Body Proximity Operations & TAG: Navigation Experiences &

 doi:10.20944/preprints202603.0556.v1

Lessons Learned from the OSIRIS-REx Mission. *AIAA SCITECH 2022 Forum* **2022**, doi:https://doi.org/10.2514/6.2022-2387.

3. Glassmeier, K.-H.; Boehnhardt, H.; Koschny, D.; Kührt, E.; Richter, I. The Rosetta mission: flying towards the origin of the solar system. *Space Science Reviews* **2007**, *128*, 1–21, doi:https://doi.org/10.1007/s11214-006-9140-8.
4. Lauretta, D.; Balram-Knutson, S.; Beshore, E.; Boynton, W.; Drouet d'Aubigny, C.; DellaGiustina, D.; Enos, H.; Golish, D.; Hergenrother, C.; Howell, E. OSIRIS-REx: sample return from asteroid (101955) Bennu. *Space Science Reviews* **2017**, *212*, 925–984, doi:https://doi.org/10.1007/s11214-017-0405-1.
5. Fujiwara, A.; Kawaguchi, J.; Yeomans, D.; Abe, M.; Mukai, T.; Okada, T.; Saito, J.; Yano, H.; Yoshikawa, M.; Scheeres, D. The rubble-pile asteroid Itokawa as observed by Hayabusa. *Science* **2006**, *312*, 1330–1334, doi:https://doi.org/10.1126/science.1125841.
6. Watanabe, S.; Hirabayashi, M.; Hirata, N.; Hirata, N.; Noguchi, R.; Shimaki, Y.; Ikeda, H.; Tatsumi, E.; Yoshikawa, M.; Kikuchi, S. Hayabusa2 arrives at the carbonaceous asteroid 162173 Ryugu—A spinning top–shaped rubble pile. *Science* **2019**, *364*, 268–272, doi:https://doi.org/10.1126/science.aav8032.
7. LIANG, Z.; LU, B.; CUI, P.; ZHU, S.; XU, R.; GE, D.; BAOYIN, H.; SHAO, W. Research Progress of Technologies for Intelligent Landing on Small Celestial Bodies. *Journal of Deep Space Exploration* **2024**, *11*, 213–224, doi:https://doi.org/10.15982/j.issn.2096-9287.2024.20240035.
8. YU, D.; ZHANG, Z.; PAN, B.; LIU, C.; DING, L.; ZHU, J.; GAO, H.; LIU, J.; CHEN, P. Development and Trend of Artificial Intelligent in Deep Space Exploration. *Journal of Deep Space Exploration* **2020**, *7*, 11–23, doi:https://doi.org/10.15982/j.issn.2095-7777.2020.20190916001. (in Chinese)
9. WU, W.; YU, D. Development of Deep Space Exploration and Its Future Key Technologies. *Journal of Deep Space Exploration* **2014**, *1*, 5–17, doi:https://doi.org/10.15982/j.issn.2095-7777.2014.01.003. (in Chinese)
10. Ely, T.P., John; Tjoelker, Robert; Burt, Eric; Dorsey, Angela; Enzer, Daphna; Herrera, Randy; Kuang, Da; Murphy, David; Robison, David; Seal, Gabriella; Stuart, Jeffrey; Wang, Rabi; Seubert, Jill. Deep Space Atomic Clock Technology Demonstration Mission Results. In Proceedings of the 2022 IEEE Aerospace Conference (AERO 2022), Big Sky, Montana, USA, 5–12 March 2022, 2022; pp. 1–20, doi:https://doi.org/10.1109/AERO53065.2022.9843303.
11. Antreasian, P.G.; Adam, C.D.; Berry, K.; Geeraert, J.L.; Getzandanner, K.M.; Highsmith, D.E.; Leonard, J.M.; Lessac-Chenen, E.J.; Levine, A.H.; McADAMS, J.V.; et al. OSIRIS-REx Proximity Operations and Navigation Performance at Bennu. *AIAA SCITECH 2022 Forum* **2022**, doi:https://doi.org/10.2514/6.2022-2470.
12. Park, R.S.; Werner, R.A.; Bhaskaran, S. Estimating small-body gravity field from shape model and navigation data. *Journal of guidance, control, and dynamics* **2010**, *33*, 212–221, doi:https://doi.org/10.2514/1.41585.
13. Yin, Z.; Zhang, K.; Duan, Y.; Liu, J.; Mu, Q. heoretical research progress of gravitational field modeling in Earth science anddeep-space exploration. *Reviews of Geophysics and Planetary Physics* **2024**, *55*, 501–512, doi:https://doi.org/10.19975/j.dqyxx.2024-002. (in Chinese)
14. Archinal, B.A.; Lee, E.M.; Kirk, R.L.; Duxbury, T.; Sucharski, R.M.; Cook, D.; Barrett, J.M. A new Mars digital image model (MDIM 2.1) control network. *International Archives of Photogrammetry and Remote Sensing* **2004**, *35*, B4.
15. Duan, Y.; Zhang, K.; Yin, Z.; Zhang, S.; Li, H.; Wu, S.; Zheng, N.; Bian, C.; Li, L. A novel gravitational inversion method for small celestial bodies based on geodesyNets. *Icarus* **2025**, *433*, 116525, doi:https://doi.org/10.1016/j.icarus.2025.116525.
16. Kikuchi, S.; Saiki, T.; Takei, Y.; Terui, F.; Ogawa, N.; Mimasu, Y.; Ono, G.; Yoshikawa, K.; Sawada, H.; Takeuchi, H.; et al. Hayabusa2 pinpoint touchdown near the artificial crater on Ryugu: Trajectory design and guidance performance. *Advances in Space Research* **2021**, *68*, 3093–3140, doi:https://doi.org/10.1016/j.asr.2021.07.031.
17. Scheeres, D.; McMahon, J.; French, A. The dynamic geophysical environment of (101955) Bennu based on OSIRIS-REx measurements. *Nature Astronomy* **2019**, *3*, 352–361, doi:https://doi.org/10.1038/s41550-019-0721-3.

 doi:10.20944/preprints202603.0556.v1

18. Keane, J.T.; Sori, M.M.; Ermakov, A.I.; Austin, A.; Bapst, J.; Berne, A.; Bierson, C.J.; Bills, B.G.; Boening, C.; Bramson, A.M.; et al. Next-Generation Planetary Geodesy: Results from the 2021 Keck Institute for Space Studies Workshops. In Proceedings of the 53rd Lunar and Planetary Science Conference, The Woodlands, Texas, March 01, 2022, 2022; p. 1622.
19. YAN, Y. 2024 Annual Progress Review in the Field of Deep Space Exploration. *High-Technology & Commercialization* **2025**, *31*, 73–78, doi:https://doi.org/10.26927/j.cnki.hitech.2025.01.022. (in Chinese)
20. Scheeres, D. Orbital mechanics about small bodies. *Acta Astronautica* **2012**, *72*, 1–14, doi:https://doi.org/10.1016/j.actaastro.2011.10.021.
21. Bottke Jr, W.F.; Vokrouhlický, D.; Rubincam, D.P.; Nesvorný, D. The Yarkovsky and YORP effects: Implications for asteroid dynamics. *Annu. Rev. Earth Planet. Sci.* **2006**, *34*, 157–191, doi:https://doi.org/10.1146/annurev.earth.34.031405.125154.
22. NASA Jet Propulsion Laboratory (JPL). Eyes on Asteroids. Available online: https://eyes.nasa.gov/apps/asteroids/#/home (accessed on 2025-06-21).
23. ESA NEO Coordination Centre (NEOCC). Orbit Visualisation Tool. Available online: https://neotools.neo.s2p.esa.int/ovt (accessed on 2025-12-01).
24. Pavelka, K.; Landa, M. Using Virtual and Augmented Reality with GIS Data. *ISPRS International Journal of Geo-Information* **2024**, *13*, 241, doi:https://doi.org/10.3390/ijgi13070241.
25. Levin, E.; Shults, R.; Habibi, R.; An, Z.; Roland, W. Geospatial Virtual Reality for Cyberlearning in the Field of Topographic Surveying: Moving Towards a Cost-Effective Mobile Solution. *ISPRS International Journal of Geo-Information* **2020**, *9*, 433, doi:https://doi.org/10.3390/ijgi9070433.
26. Zhi, Y.; Nico, S. Modeling the gravitational field by using CFD techniques. *Journal of Geodesy* **2021**, *95*, 68, doi:https://doi.org/10.1007/1345_2019_72.
27. Han, B.; Zhang, Y.-X.; Zhong, S.-B.; Zhao, Y.-H. Astronomical data fusion tool based on PostgreSQL. *Research in Astronomy and Astrophysics* **2016**, *16*, 178, doi:https://doi.org/10.1088/1674-4527/16/11/178.
28. JPL Solar System Dynamics (SSD). SBDB API. Available online: https://ssd-api.jpl.nasa.gov/doc/sbdb.html (accessed on 2025-12-01).
29. Giorgini, J.D., JPL Solar System Dynamics Group. Horizons System. Available online: https://ssd.jpl.nasa.gov/horizons/app.html#/ (accessed on 2025-12-01).
30. Planetary Data System (PDS) Small Bodies Node; Asteroid/Dust Subnode. PDS SBN Asteroid/Dust Subnode. Available online: https://sbn.psi.edu/pds/shape-models/ (accessed on 2025-12-01).
31. Duan, Y.; Yin, Z.; Zhang, K.; Zhang, S.; Wu, S.; Li, H.; Zheng, N.; Bian, C. Modeling the gravitational field of the ore-bearing asteroid by using the CFD-based method. *Acta Astronautica* **2024**, *215*, 664–673, doi:https://doi.org/10.1016/j.actaastro.2023.12.039.
32. Avula, S.B. PostgreSQL Table Partitioning Strategies: Handling Billions of Rows Efficiently. *International Journal of Emerging Trends in Computer Science and Information Technology* **2024**, *5*, 23–37, doi:https://doi.org/10.63282/3050-9246.IJETCSIT-V5I3P103.
33. Xiao, W.; Lv, X.; Xue, C. Dynamic Visualization of VR Map Navigation Systems Supporting Gesture Interaction. *ISPRS International Journal of Geo-Information* **2023**, *12*, 133, doi:https://doi.org/10.3390/ijgi12030133.
34. Bardwell, J. Adding Smooth Locomotion To Unreal Engine 5.1+ For VR (Open XR). Available online: https://www.gdxr.co.uk/learning/adding-smooth-locomotion-to-unreal-engine-5-1 (accessed on 2025-06-21).