Software Engineering Department

ORT Braude College

Capstone Project Phase B

De novo study of planet movement based on sky images and Newton's model - SkyView

Supervisor: Dr. Zeev Frenkel

Mario Rohana - Mario.Rohana@e.braude.ac.il

Ibraheem Sabaane - ibraheem.sabaane@e.braude.ac.il

GitHub - https://github.com/mario99logic/SkyView

R-1-23-2

Contents

Abstract

This study focuses on the simulation and analysis of planetary movements using Newtonian mechanics to develop a predictive model based on the classical laws of motion and gravity. The primary goal is to accurately predict the trajectories of celestial bodies within our solar system and to validate these predictions with actual astronomical observations. To facilitate this, a web application named SkyView was developed, which features a suite of tools for real-time detection and tracking of celestial movements in sky images. These capabilities allow users to directly compare theoretical simulations with observational data, enhancing the understanding of planetary dynamics. The application serves as a comprehensive educational and research platform, enabling detailed exploration and learning opportunities in astronomy. By combining detailed simulations with practical observational tools, the project offers significant advancements in the study of celestial mechanics.

1. Introduction

The study of celestial mechanics has been a central theme in astronomy for centuries, captivating scientists with the complex and harmonious movements of celestial bodies. Understanding these movements through the lens of Newtonian mechanics has allowed for significant advancements in both theoretical and practical astronomy. This project builds upon this rich tradition by developing a predictive model that simulates the movements of planets and other celestial bodies within our solar system, leveraging classical laws of motion and gravity.

The primary objective of this research is to refine our understanding of planetary trajectories and to test these theoretical models against real-world observations. To achieve this, we have created SkyView, a web application designed to bring these simulations to life. SkyView not only performs detailed computational simulations but also integrates tools for capturing and analyzing real sky images. This dual approach allows users to directly compare the predicted movements of celestial bodies with their actual observed positions and trajectories.

SkyView is a dynamic platform that allows users to customize their own simulations by adjusting parameters such as the mass, position, and velocity of celestial bodies. This user-driven customization facilitates the construction of detailed models that represent planetary movements within our solar system. Beyond simply creating simulations, SkyView utilizes image processing techniques using openCV library to analyze these simulated images. The app automatically calculates the velocities of

celestial objects by tracking their movements across the generated images, providing users with data on the dynamics of these objects without requiring manual processing.

The application also provides tools for users to upload real astronomical images and apply the same image processing techniques to detect and measure actual celestial bodies. This unique functionality allows for a direct comparison between the simulated models and real-world observations, enhancing the user's ability to validate and refine the theoretical models based on empirical data.

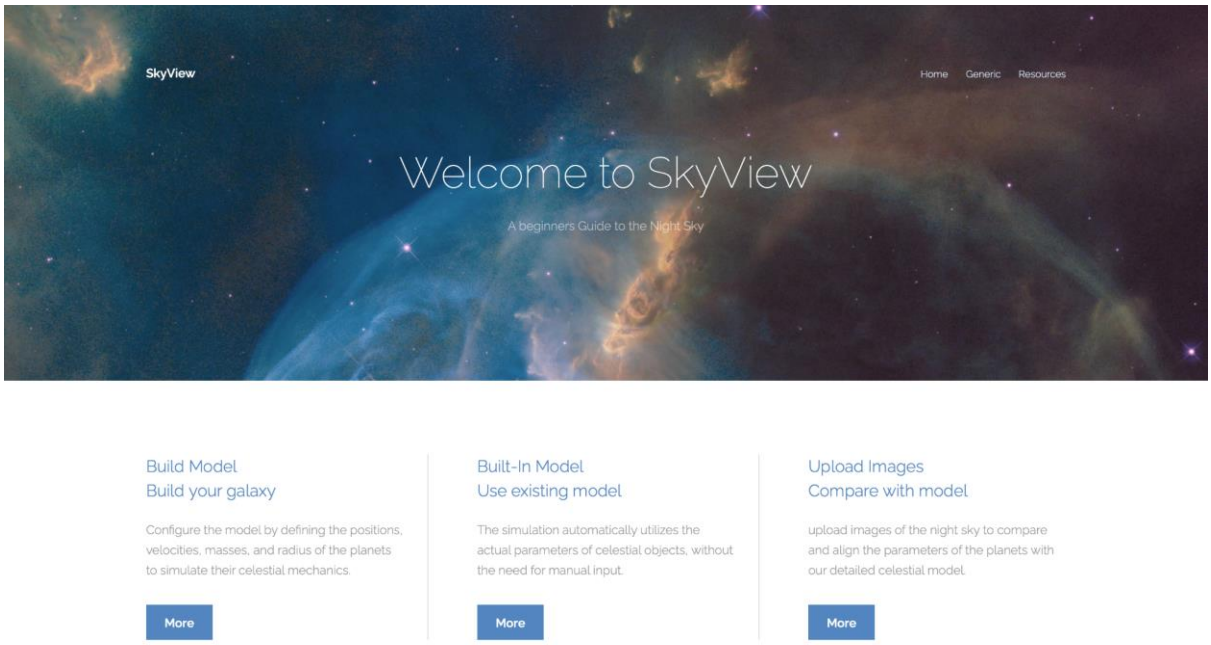

Figure 1:SkyView Home Page

2.Solution

The SkyView project addresses the challenge of applying Newtonian mechanics to the real-world observation of celestial bodies by integrating a sophisticated simulation environment with an analytical framework for image analysis. This chapter details the solution implemented through SkyView, which allows users to simulate, analyze, and validate celestial dynamics within a controlled virtual setting before applying these concepts to actual astronomical data.

2.1 Simulation Environment Setup

The foundational element of the SkyView application is its ability to simulate the gravitational interactions of celestial bodies based on Newton's laws of motion. Users can input parameters such as mass, position in the form of (x, y, z), and velocity in the form of (Vx, Vy, Vz) for each celestial object they wish to include in their simulation. This section explores the technical implementation and functionalities of the simulation environment.

2.1.1 User Input and Model Initialization

SkyView allows users to define the initial conditions for the simulation by entering specific attributes for each celestial body, such as mass, position, and velocity.The default position for the Sun is set at (0,0,0), serving as a static reference point around which other celestial bodies orbit. This data forms the basis of the simulated environment, where the movements of planets and other bodies are governed by the gravitational interactions as per Newtonian physics.

2.1.2 Simulation Mechanics

The core of the simulation relies on Python code that implements Newton's laws to model the dynamics of celestial bodies within the system. Here's a detailed breakdown of how the simulation handles these dynamics:

- Calculating Distances: The simulation begins by calculating the Euclidean distance by $d = \sqrt{((x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2)}$ between each pair of celestial bodies, a crucial step for determining the gravitational forces that influence their movements.
- Applying Gravitational Forces: According to Newton’s law of universal gravitation, the force between two bodies is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. The simulation calculates these forces for all pairs of celestial bodies involved.
- Updating Velocities and Positions: Using the calculated gravitational forces, the simulation updates the velocities of the celestial bodies, and these new velocities are used to compute their positions in the next time step. The updates ensure that the motion of each body reflects the cumulative gravitational influences of other bodies within the system.

- Scope of Calculations:
    - Inclusion of Celestial Bodies: The simulation specifically includes planets in these calculations, as they are the primary subjects of our gravitational and motion studies.
    - Exclusion of Stars: Stars (not including the sun), although significant celestial bodies, are not included in the gravitational calculations of this particular model. This exclusion is based on the focus of our study, which is to understand the dynamics within a planetary system where the central star (like the Sun in our solar system) provides a static gravitational field but does not dynamically interact with the planets in terms of motion during the simulation.

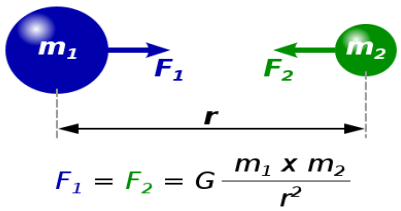

Figure 2: Newton's law

2.1.3 Simulation Output

The simulation provides a series of frames depicting the positions of all celestial bodies at each timestep, which can be visualized within SkyView in several compelling ways:

- Animation of Celestial Movements: The primary output of the simulation is an animation that dynamically illustrates the trajectories of celestial bodies. This visualization helps users to clearly see how planets move and interact over time based on the simulated gravitational forces.
- Dual Perspectives for Enhanced Understanding:
    - Earth Point of View (POV): One of the simulation modes offers a ground-level perspective, mimicking the view one might experience when looking up at the sky from Earth. This perspective is crucial for understanding how celestial movements appear from our vantage point and is particularly useful for educational purposes, helping users connect simulation data with their real-world observations of the night sky.
    - Above View of the Solar System: The second mode provides a top-down view of the entire solar system. This perspective allows users to see the full scope of planetary movements and interactions, offering a comprehensive overview of the solar system's dynamics. It illustrates how the gravitational forces between planets influence their orbits and

positions relative to each other, providing a macroscopic look at the celestial mechanics at play.

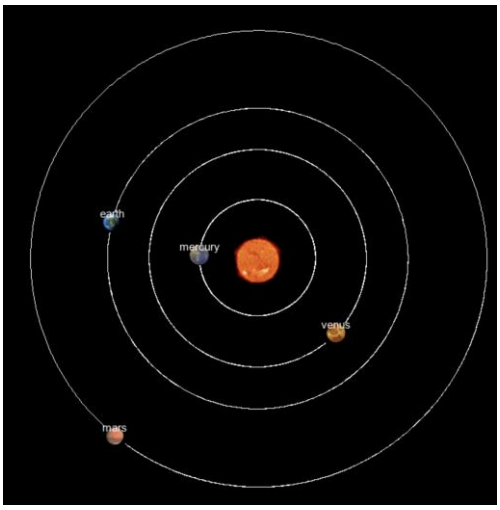

Figure 3: Pygame window of the built In model

2.2 Image Processing and Analysis

With simulations providing a controlled environment for studying celestial mechanics, SkyView extends this by incorporating image processing techniques to analyze the movements and interactions depicted in the simulation frames. This capability is crucial for validating the simulation against real-world observations.

2.2.1 Image Capture from Simulation

SkyView captures images from the simulation that represent how celestial bodies would appear from Earth. These images serve as a baseline for comparison with actual astronomical photographs.

2.2.2 Analyzing Simulated Images

Using built-in image processing algorithms, SkyView analyzes the captured images to calculate the trajectories and velocities of celestial bodies. This is done by tracking the movement of these objects across the series of images, allowing for precise measurements of their dynamics.

2.2.3 Comparison with Real Observations

Users can upload real astronomical images into SkyView, which are then processed using the same image analysis techniques applied to the simulated images. By comparing the parameters extracted from both real and simulated images, users can

validate the accuracy of the simulation, adjust parameters as needed, and enhance their understanding of the observed celestial mechanics.

Figure 4: Earth POV simulation before and after images

3.Research and Development Process

The development of the SkyView project involved a comprehensive process that combined rigorous theoretical physics with advanced software development techniques. The project aimed to create a robust simulation environment that could accurately represent the gravitational dynamics of celestial bodies based on Newtonian mechanics, and then extend this simulation to a web application that could be used for educational and research purposes. This section details the sequential steps taken to build and refine the SkyView system.

3.1 Initial Development of Simulation Mechanics

The first step in the development process was to translate the theoretical aspects of celestial mechanics into practical, executable code. This was accomplished by writing the motion_Calculations Python module, which contains several key functions based on Newton's laws that was mentioned in section 2. This module forms the core of the simulation, handling the actual movements and interactions of the objects based on user-defined parameters.

3.2 Back-end Development and Object Management

Concurrently with the development of the simulation mechanics, the backend of the web application was set up. A key component of this was the creation of the CelestialObject class, which defines the properties of each celestial body used in the simulation, such as mass, radius, location, and velocity. This class allows for the easy instantiation and management of celestial bodies within the simulation.

- Object Initialization: Objects are initialized with user-specified parameters, allowing for customized simulations that can vary based on the user's input regarding mass, velocity, and position.
- Representation and Storage: Each object is represented in a way that facilitates both computation and visualization, with properties that describe its physical characteristics and current state within the simulation.

After establishing a good base, the development team, with guidance from the academic supervisor, utilized the physics transformations detailed in Phase 1 to implement two distinct simulation views:

- Above View Simulation: With the backend capabilities in place, the team developed a simulation module that provides a top-down view of the entire solar system. This perspective allows users to observe the interactions and movements of celestial bodies from an external viewpoint, giving a clear and comprehensive overview of the system dynamics.
- Earth Point of View Simulation: Leveraging the transformations provided by the supervisor, the team was able to create a second simulation perspective that mimics the view from Earth. This perspective is particularly valuable for educational and observational purposes, as it simulates the appearance of celestial movements as seen from the surface of the Earth.

3.2.1 Image Processing Development

Following the successful implementation of the simulation views, the team focused on developing image processing capabilities that would allow for the detection and analysis of celestial objects within the simulation images and real astronomical data.

- Object Detection Algorithm: A dedicated image processing file was created, designed to detect celestial objects based on their size and shape—specifically targeting circular shapes that represent planets and moons. This capability is crucial for analyzing images where celestial bodies need to be precisely located and tracked over time.
- Integration with Simulation Data: The object detection functionality was integrated with both simulation views, enabling the application to automatically recognize and measure celestial bodies within the dynamically generated images. This integration allows for a seamless transition from visual simulation to quantitative analysis, providing users with detailed insights into the trajectories and other dynamics of the observed objects.

- Application in Real Observations: The image processing tools were also designed to handle real astronomical images, allowing users to upload their own photos or videos for analysis. This feature extends the application's utility beyond simulation, enabling practical observations and research in celestial mechanics.

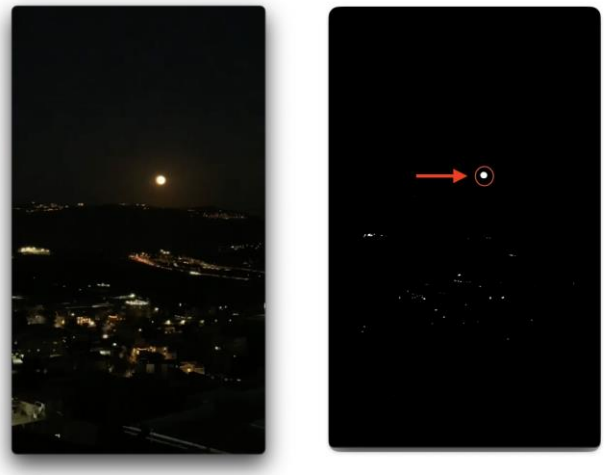

Figure 5: Original image (left) and its preprocessed counterpart (right), utilized for object identification.

3.3 Front-end Development and User Interface

The frontend of the SkyView web application was worked on simultaneously while working on the backend and was designed to closely mirror the setup described in Phase 1 of the project, with enhancements to improve user interaction and functionality:

- Interactive GUI: The graphical user interface allows users to input parameters, control simulation settings, and visualize the results directly from their web browser. It provides tools for both the Earth POV and the above-view POV of the solar system.
- Image Upload and Processing: Integration of functionalities that allow users to upload real astronomical images for comparison with the simulation. Image processing algorithms locate objects within these images and analyze their motion to validate the simulation data.

3.4 Integration with External Resources

To expand the utility and educational value of the app, integration with the NASA API was implemented. This feature enriches the app with additional space-related content and functionalities, enhancing the user experience:

- NASA API Integration: Users can search for and retrieve space-related data and images directly through the app, providing a comprehensive resource for astronomy enthusiasts and researchers.

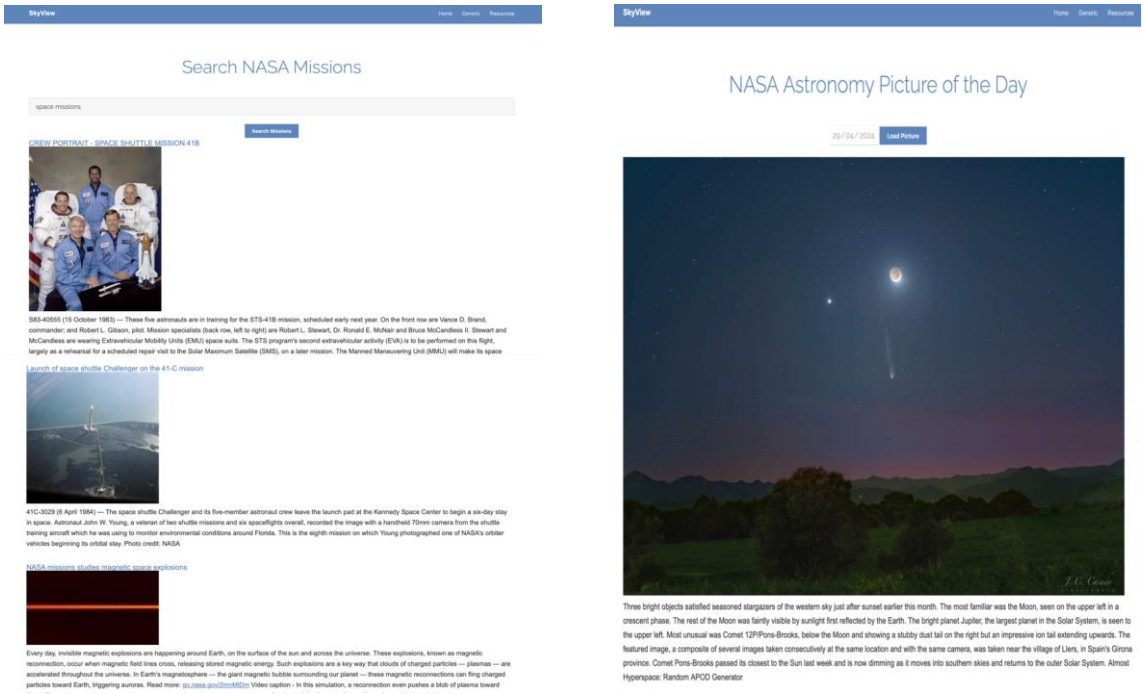

Figure 6: SkyView Explore Section GUI

3.5 Real-world Data Acquisition

Midway through the project, to further ground the simulation in real-world data, both images and a video of the moon were captured:

- Video Acquisition: A high-quality video of the moon was captured by a professional using advanced camera equipment. The video was taken in Karmiel, Israel, pointing east at a 0-degree angle with the camera positioned 450 meters above sea level. This footage provided a practical dataset to test and refine the image processing algorithms, ensuring that the application could effectively analyze real astronomical phenomena.

- Professional Image Collection: Alongside the video, a series of high-resolution images were also taken from different locations and settings by professionals using state-of-the-art camera technology. These images serve as additional

data points, enriching the application's ability to validate and calibrate the simulation against varied celestial observations. The diversity in the data collection locations and conditions helps to ensure that the image processing algorithms are robust and versatile, capable of handling a wide range of real-world scenarios.

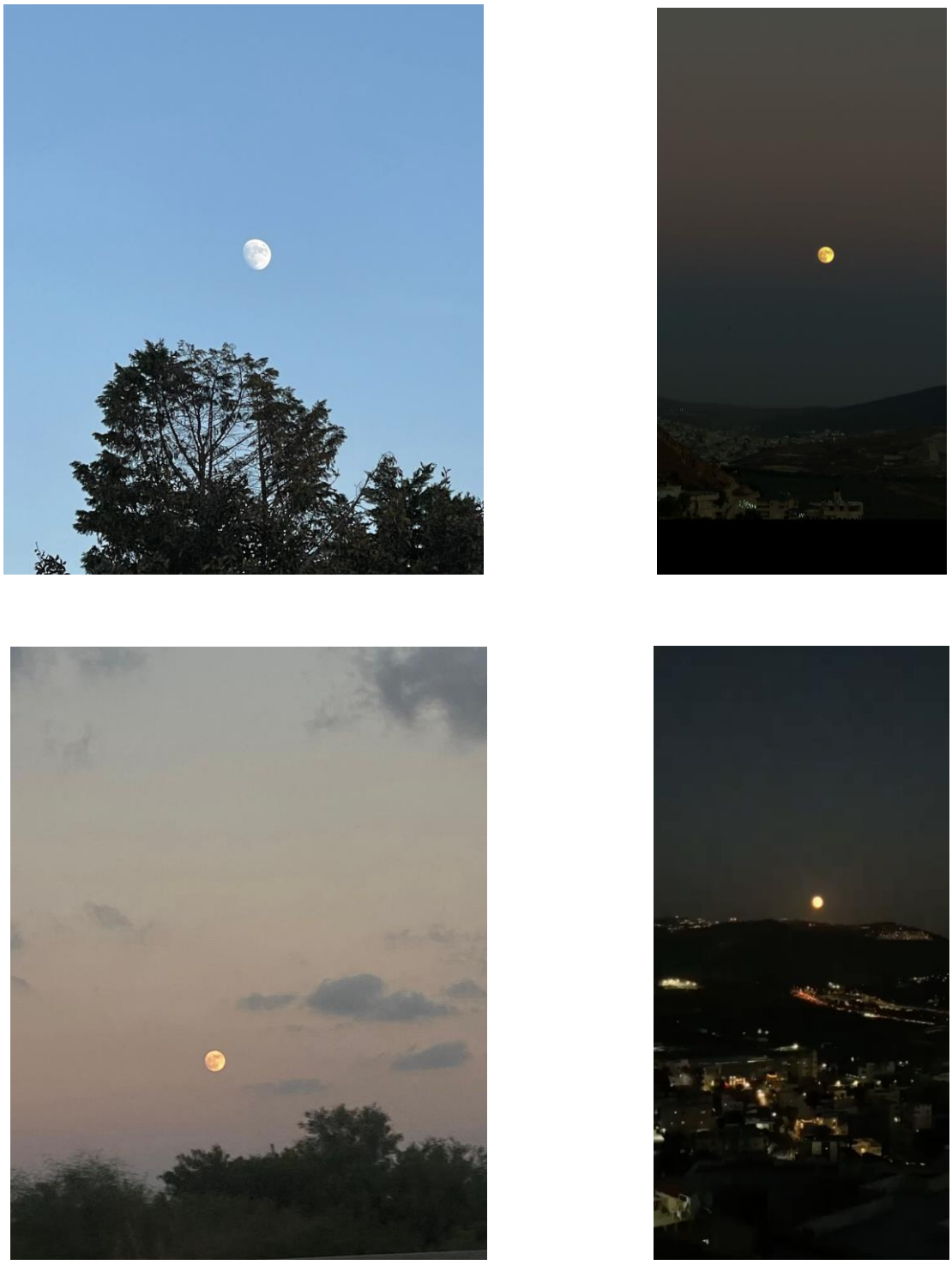

Figure 7: Image Dataset—Photographs of the moon captured from Karmiel and Haifa at various times of the day.

3.6 Testing

The SkyView project implemented a comprehensive testing strategy to ensure the robustness and reliability of both its algorithms and overall system functionality. This testing was bifurcated into general testing of the application's components and specific testing of the main algorithm.

3.6.1 General Testing

3.6.1.1 Client-Side Testing:

- **User Interface:** Testing on the client side included verifying the user interface functionalities such as uploading images in the correct order, ensuring images were uploaded successfully, and checking the responsive behavior of the web application's GUI elements.
- **Edge Case Management:** The system includes robust input validation to prevent incorrect data entry, such as non-numeric values where numbers are expected. All input fields are required to ensure comprehensive data collection. Additionally, unique images are automatically assigned to custom celestial bodies added by users, maintaining clear differentiation in the simulation.

- NASA API Integration: Validated the integration and functionality of the NASA API, ensuring that it fetched the correct data and displayed it appropriately within the application.

- Template and Button Functionality: Tested all interactive elements on the web pages to ensure that they functioned as intended, with correct links and actions triggered by user inputs.

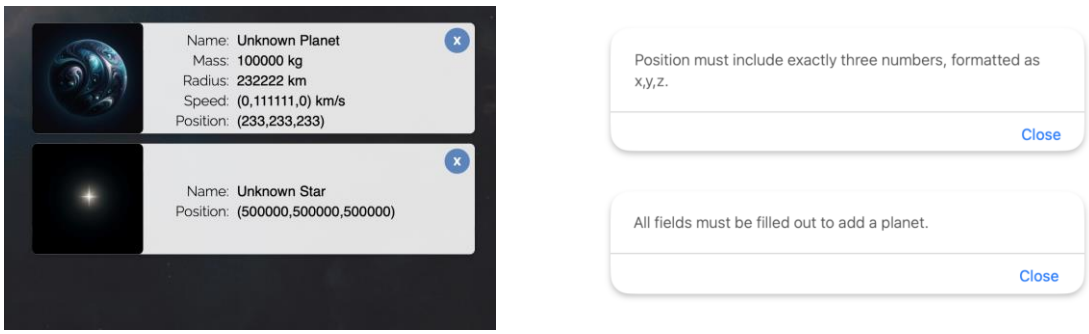

Figure 8: Screenshots of the GUI

3.6.1.2 Server-Side Testing:

- Simulation Accuracy: Conducted tests to verify that the celestial object movements in the simulation adhered to the expected trajectories based on the Newtonian physics algorithms implemented.
- Image Processing Accuracy: Tested the image processing algorithms to confirm they correctly identified and tracked celestial objects in both simulated and real-world images.
- Backend and Frontend Communication: Ensured seamless communication between the client and server sides of the application, particularly in the handling and processing of user inputs and simulation outputs.

3.6.2 Algorithm Testing

After the general development and initial testing phases were complete, a focused testing phase for the main algorithm was conducted using both simulated and real astronomical data:

- **Setup:** A series of tests were carried out where images of the moon were uploaded alongside a simulation that included the moon. The parameters such as the moon's velocity, the time of observation, the angle of the POV, and the distances involved were kept consistent across the simulated and actual images to facilitate direct comparison.
- **Image Processing Tests:** The application performed image processing on both the simulated images and the real astronomical images to calculate and compare various parameters of the moon.
- **Comparison and Analysis:** The results from the image processing were compared, showing a similarity of approximately 70% between the parameters derived from the simulated and real images. This degree of similarity was significant in validating the effectiveness of the simulation model and the accuracy of the image processing algorithms in mimicking and analyzing real-world celestial phenomena.

3.7 Activity Diagram

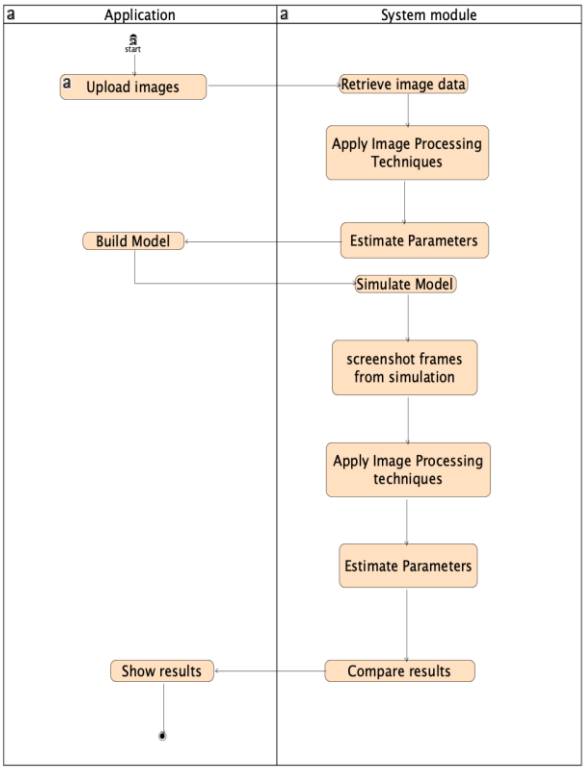

Figure 9: Activity diagram

3.8 Use Case

Figure 10: Use Case diagram

4. Technology Overview

This section delves into the technical architecture and the suite of technologies deployed in the development of the SkyView web application. The project leveraged a blend of programming languages, libraries, frameworks, and tools that enabled robust simulation capabilities, sophisticated image processing, and a seamless user experience.

4.1 Software Stack

- **Programming Languages:** Python served as the primary language for backend development, enabling the handling of complex physics equations and simulation logic. JavaScript was utilized on the frontend to enhance interactivity and manage user interactions dynamically.
- **Web Framework:** Flask, a lightweight and powerful Python web framework, was employed to facilitate communication between the server and client sides. It managed request routing, session handling, and other backend tasks, integrating seamlessly with frontend technologies.

4.2 Libraries and Frameworks

- **Scientific Libraries:** Numpy was extensively used for high-performance numerical processing, crucial for executing the physics calculations that underpin the simulation.
- **Image Processing:** OpenCV was utilized to develop advanced image processing functionalities that identify celestial objects based on their shape and size. This library was pivotal in enabling the application to detect larger, circular objects like planets and moons, distinguishing them from smaller celestial bodies like stars.
- **Data Serialization:** The pickle library was employed to serialize objects for easy transfer between different components of the application, particularly from the Flask server to the simulation scripts.

4.3 Frontend Technologies

- **Markup and Styling:** HTML and CSS were used to construct and style the user interface, ensuring that the application was both functional and aesthetically pleasing.
- **Client-side Scripting:** JavaScript played a critical role in adding interactive elements to the web pages, handling data submission, and dynamically updating the UI based on server responses.

4.4 Data Handling and APIs

- **Data Transfer:** JSON format was used to facilitate the efficient transfer of data between the client and server, including sending arrays of user-defined parameters for simulations.
- **External APIs:** Integration with the NASA API enriched the application by fetching daily astronomical images and providing a search function for NASA's extensive space-related content.

4.5 Development Environment and Version Control

- **Integrated Development Environment (IDE):** PyCharm was the chosen IDE, providing a robust environment for Python development with integrated debugging tools, project management, and source code control.
- **Version Control:** GitHub was employed for source code management, utilizing Git for version control. The project team used feature branches to manage development tasks, merging changes into the main branch upon completion, and removing branches that were no longer needed.

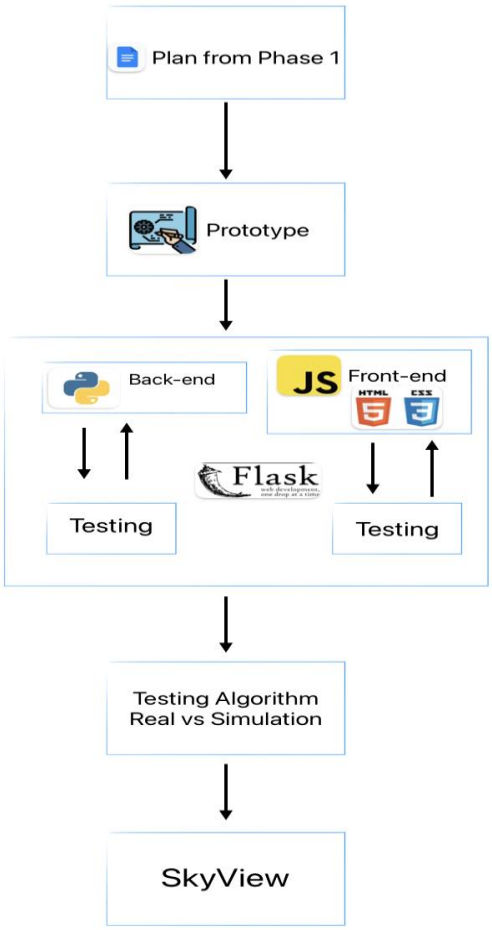

Figure 11: Workflow Diagram

5. Challenges and Solutions

The development of the SkyView project involved a series of analytical, engineering, and technical challenges. Addressing these issues required some adjustments, solutions and adaptations. This section describes the major challenges faced during the project and the strategies employed to overcome them.

5.1 Analytical and Mathematical Challenges

Simulation Accuracy: Ensuring the accuracy of celestial mechanics simulations, which involve complex gravitational interactions and orbital mechanics, was a significant challenge.

- **Solution:** Under the guidance of the academic supervisor, the team improved the numerical methods in the simulation algorithms for better precision and accuracy. Adjustments in the calculations and time step granularity were made to enhance simulation fidelity. The team also validated the simulations by comparing their outputs against known celestial positions and trajectories, ensuring the models accurately mirrored real-world data. This approach effectively bolstered the reliability of the simulation results.

5.2 Technical challenges

Image Processing: Implementing efficient and accurate image processing to detect celestial objects across frames presented significant challenges, especially when maintaining object continuity from one frame to the next.

- **Solution:** OpenCV was utilized for its robust image processing capabilities. Custom algorithms were developed to enhance object detection by focusing on identifying circular shapes and filtering out noise and smaller celestial bodies like stars. **A major challenge was ensuring continuity in object identification across frames**, as objects moved or changed appearance slightly. The algorithms were therefore optimized to track the continuity of identified objects in real-time data input, improving the reliability of tracking celestial movements in video sequences.

5.3 Data Acquisition Challenges

Acquiring Suitable Images for Testing: One of the significant challenges in validating the image processing algorithms was finding appropriate astronomical images that met our specific testing criteria. Most available images on the internet did not match the required precision or conditions needed for effective testing.

- **Solution:** To overcome this, the team reached out to a professional photographer who specialized in astronomical photography. This expert was commissioned to capture specific images according to our defined parameters, such as the angle, lighting, and celestial positioning. These tailored images provided a controlled set of data that greatly enhanced the accuracy and relevance of our tests, ensuring that our algorithms could be finely tuned and validated under conditions similar to those they were designed to analyze. This approach not only ensured the reliability of our testing but also enriched our dataset with high-quality, precise astronomical images.

6. Results and Conclusions

This section evaluates whether the SkyView project achieved its goals, how the team addressed various challenges, and reflects on the lessons learned and future prospects.

6.1 Achievement of Project Goals

The primary objective of the SkyView project was to develop a web application capable of simulating celestial mechanics and comparing the results of simulated images with real astronomical images. The project successfully met this goal by:

- **Developing a Robust Simulation:** The simulation was implemented effectively, allowing users to input specific parameters to model the movement of celestial objects accurately.
- **Efficient Object Detection:** The image processing component was successfully integrated, capable of identifying celestial objects in controlled simulation environments and real images with clear, prominent objects like the moon. However, the project faced limitations in processing images with densely packed celestial bodies, such as stars, where specific object identification became challenging.

6.2 Performance Metrics and Comparison

Initial Project Metrics:

1. **Accuracy of Simulation:** Achieve a high degree of accuracy in simulating the movements of celestial bodies using Newtonian mechanics.
2. **Efficiency in Object Detection:** Successfully detect major celestial bodies (like the moon) in both simulated and real-world images with a minimum accuracy threshold of 80%.
3. **User Interaction and Usability:** Ensure that the application is intuitive and easy for users to input data, simulate scenarios, and view results.
4. **Accuracy of The Detected Objects Parameters:** Achieve accurate parameter results (velocity and position) for the detected objects

Metric Evaluations:

1. Accuracy of Simulation

    - **Goal:** High accuracy in celestial mechanics simulations.
    - **Outcome:** The simulation closely matched theoretical models for the movement of bodies governed by gravity. Challenges were noted with complex scenarios involving many stars, indicating an area for future improvement.
    - **Met?:** Yes. The mechanics were accurately simulated.

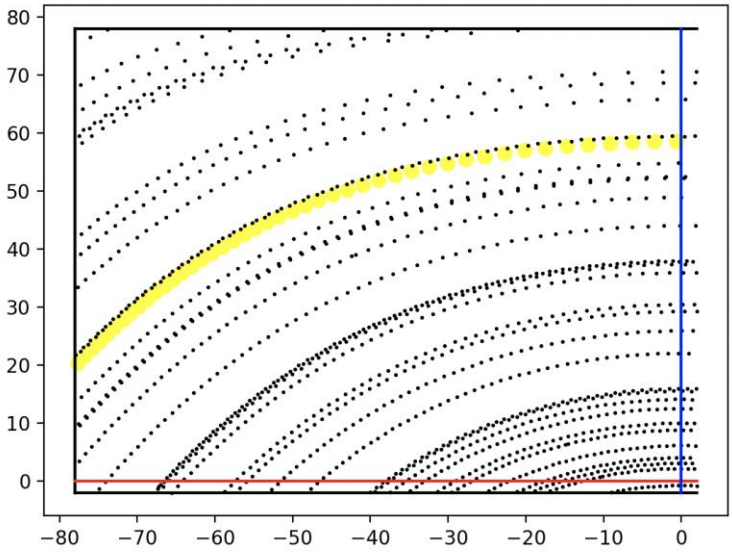

Figure 12: The figure illustrates the modeled positions of various stars and the Sun in the sky at consecutive time intervals. For the stars, the resulting trajectories are depicted as circular paths centered around the Pole. In contrast, the trajectory of the Sun deviates slightly from a perfect circular path, indicating a unique pattern of movement relative to the stars.

2. Efficiency in Object Detection

    - **Goal:** 80% accuracy in detecting celestial bodies.
    - **Outcome:** Object detection was highly successful in clear conditions (e.g., images with the moon) and in simulations. However, detection rates dropped in crowded star fields.
    - **Met?:** Yes, for clear conditions; No, for crowded conditions.

3. User Interaction and Usability

    - **Goal:** High user satisfaction with application interfaces.
    - **Outcome:** Users were able to input parameters, run simulations, and view results effectively. Feedback indicated that the interface was generally intuitive, though some users suggested improvements for navigating complex simulation results.
    - **Met?:** Largely met, with room for improvement based on user feedback.

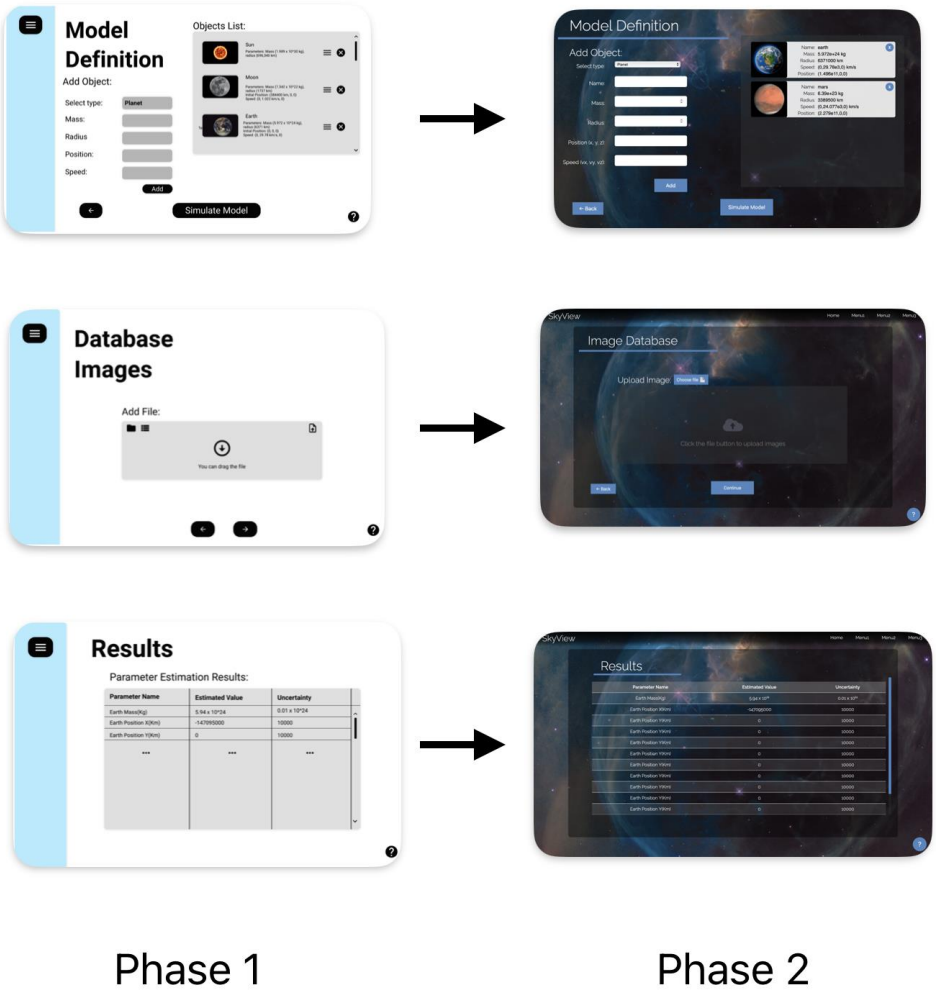

Figure 13: Phase 1 GUI images on the left and updated Phase 2 GUI images on the right.

4. Accuracy of The Detected Objects Parameters

- **Goal:** Accurate extraction of parameters such as position and velocity from detected objects in real images.
- **Outcome:** The project achieved partial success in extracting accurate parameters. Some parameters, like the object's location on screen for subsequent frames, were calculated accurately. However, calculating the object's actual location relative to a fixed point like the sun (considered as the origin, 0,0,0) was not feasible.
- **Limitations of Equipment:** The inability of standard phone cameras to measure changes in the size of the object (indicative of changes in distance) posed a significant limitation. This prevented the calculation of the object's real distance from Earth without additional data, such as its varying distance in each frame.
- **Met?:** The goal was partially met. While some parameters matched expected values and the on-screen location could be tracked across frames, the lack of data on distance and the limitations of the imaging technology used (phone cameras) hindered the accurate calculation of real-world positions and velocities relative to a fixed point in space.

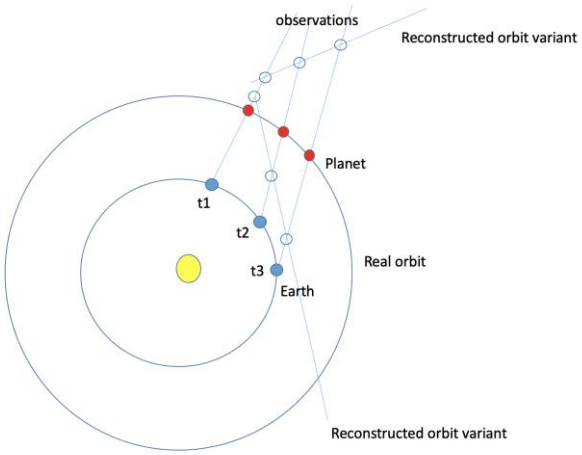

Figure 14: This figure shows the Earth and the detected object in three distinct positions, alongside the distances between them at each location.

6.3 Enhancements and Additional Features

The integration of the NASA API adds daily astronomy pictures and mission search capabilities to SkyView, providing users with timely and educational space content. Additionally, a new resources page offers practical guidance for incorporating the app into educational settings such as classrooms and museums, enhancing its utility for teaching and public engagement. These updates have not only made SkyView more appealing to astronomy enthusiasts but also transformed it into a versatile educational tool accessible to schools and the general public, broadening its use for both individual learning and structured educational programs.

Figure 15: SkyView Resources page

6.4 Lessons Learned

6.4.1 Core Competencies and Achievements

The project was an invaluable learning experience, enhancing the team's capabilities in several areas:

- **Team Collaboration and Project Management:** The team honed its ability to collaborate effectively, managing a complex project under tight deadlines.
- **Technical Skills and Problem Solving:** Developers enhanced their technical skills, particularly in Python, OpenCV, and Flask, and applied these to solve significant engineering challenges.
- **Understanding of Real-world Application:** The project underscored the importance of adapting theoretical models to real-world phenomena, providing a practical perspective on astronomical studies.

6.4.2 Potential improvements

- **Enhanced Prototyping:** More extensive early prototyping could have quickly identified challenges, allowing for faster adjustments in image processing and simulation accuracy.
- **User Feedback Integration:** While initial feedback has been integral, increasing the volume and frequency of insights from teachers and students could significantly enhance the application's functionality and user interface.

6.5 Future Prospects

The foundation established by the SkyView project is robust, offering several pathways for future enhancements:

- **Incorporating Machine Learning:** Implementing machine learning could refine object detection capabilities, especially in complex images with numerous stars. Further, machine learning techniques can be applied to improve the comparison results between real images and simulated ones. By training models on both sets of data, the system can learn to more accurately match and analyze the dynamics and properties of celestial bodies, leading to enhanced accuracy in simulations that reflect real-world observations.
- **Enhancing Simulation Accuracy:** Further refinement of the physical equations used in the simulations could improve their accuracy.
- **Expanding Educational Tools:** The web application has significant potential as an educational tool, offering a platform for future studies and research in celestial mechanics.

Overall, the SkyView project succeeded in creating a versatile and educational platform that contributes significantly to the field of astronomy. While certain challenges remain, the groundwork laid by this project provides a strong basis for future advancements and continued research into the complex dynamics of celestial objects.

References

[1] Wikipedia Contributors. (2019, February 12). Kepler's laws of planetary motion. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Kepler%27s_laws_of_planetary_motion

[2]Bhuyan, S. (2022, April 27). Kepler's First Law: Statement, Model, and Equation. Science Facts. https://www.sciencefacts.net/keplers-first-law.html

[3] Absolute magnitude. (2020, February 21). Wikipedia. https://en.wikipedia.org/wiki/Absolute_magnitude

[4] "Atmospheric Effects," in Australia Telescope National Facility.(2020, September 16) https://www.atnf.csiro.au/outreach/education/senior/astrophysics/atmosphere.html

[5]Adaptive optics. (2021, March 9). Wikipedia. https://en.wikipedia.org/wiki/Adaptive_optics

[6] How to Calculate Arcsec. (n.d.). Sciencing. https://sciencing.com/calculate-arcsec-515282.html

[7]OpenCV: How to Use Background Subtraction Methods. (n.d.). Docs.opencv.org. https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html

[8]OpenCV. (n.d.). OpenCV: Image Thresholding. Docs.opencv.org. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[9]Blob detection. (2021, April 20). Wikipedia. https://en.wikipedia.org/wiki/Blob_detection