Technical Report: Interaction and Simulation

Analysis of the WebGL "Space Scene" Codebase

Computer Science & 3D Graphics Department

January 5, 2026

1 Introduction

This report provides a deep technical analysis of the `glb.html` project. The application is a sophisticated 3D simulation that leverages the Three.js library to render a cosmic environment. It moves beyond static rendering by implementing real-time physics simulations, camera interpolation, and a robust user-interaction system.

2 Simulation Mechanics

The code implements several simulation patterns to mimic astronomical behaviors without the overhead of a full physics engine.

2.1 Gravitational Attraction Simulation

The simulation of the asteroid belt follows a specific vector-based logic. Each asteroid identifies the center of the scene $(0, 0, 0)$ as its attractor.

- **Vector Normalization:** The code calculates the direction vector from the asteroid's current position to the center.
- **Directional Velocity:** By multiplying the normalized vector by a speed constant, the asteroids move consistently toward the black hole.
- **State Reset:** A conditional logic check is implemented: `if (dist < threshold)`, the asteroid's position is reset to its `initialPos`, creating a continuous loop of matter "falling" into the singularity.

2.2 Procedural Rotation and Orbitals

Planetary bodies utilize an independent rotation constant (`planet.rotation.y += 0.005`). This separates the "visual time" of the scene from the frame rate, ensuring a smooth, persistent motion across all objects.

3 Interaction System

Interaction in the 3D space is handled via a bridge between 2D screen coordinates and 3D world space.

3.1 Raycasting and Selection

The primary interaction tool is the **THREE.Raycaster**.

1. The code captures the mouse click and maps it to a Normalized Device Coordinate (NDC) between -1 and 1.
2. A ray is projected from the camera's position through the mouse coordinates.
3. The system checks for `intersectObjects(planets)`.
4. If a planet is hit, a **Highlight Box** (using `THREE.Box3`) is dynamically positioned to encompass the selected object, providing vital visual feedback to the user.

3.2 User Interface (UI) Integration

The project employs a hybrid approach where an HTML overlay communicates with the 3D scene. This allows for:

- **Dynamic Instantiation:** Functions like `addAsteroid()` allow for real-time injection of new objects into the rendering buffer.
- **Parameter Tweaking:** UI sliders and color pickers modify the `mesh.material` properties instantly via event listeners.

4 Conclusion

The `glb.html` file serves as an excellent case study for interactive 3D environments. By combining **mathematical simulation** (normalization and interpolation) with **user agency** (raycasting and UI), the project achieves a high degree of immersion and utility. The essential takeaway is the use of smooth camera transitions and immediate visual feedback, which define a professional user experience.