



SAPIENZA
UNIVERSITÀ DI ROMA

DIAG
Dipartimento di Ingegneria
informatica, automatica e gestionale
Antonio Ruberti

SPACE EXPLORATION

AN INTERACTIVE 3D SOLAR SYSTEM SIMULATION

Gianmarco Donnesi
Matr. n. 2152311

A.Y. 23/24

01

PROJECT OVERVIEW

This project is a 3D simulation of a solar system and includes various celestial bodies such as the Sun, planets, asteroids, and the International Space Station (ISS).

02

PURPOSE

To provide a visually engaging and interactive representation of our solar system.

To allow users to explore and learn about different celestial bodies and their movements.

03

KEY FEATURES

- Realistic sun with dynamic lighting effects.
- Detailed planets with realistic textures and orbits.
- A belt of asteroids with random movements.
- An orbiting ISS with controllable view.

04

TECHNOLOGIES USED

- **Three.js**: A powerful JavaScript library for 3D graphics.
- **GLTFLoader**: For loading 3D models of the ISS.
- **Custom Shaders**: For realistic visual effects on the sun and planets.

SUN LIGHTING

The sun is the central light source in the solar system simulation. It is designed to be visually dynamic and realistic to enhance the overall experience.

Lighting Techniques:

- **Point Light:**

- A high-intensity point light is placed at the sun's position.
- This light casts shadows and illuminates other objects in the scene.
- Configured with decay to simulate the natural falloff of light over distance.

- **Emissive Material:**

- The sun's material is emissive, meaning it emits light and creates a glow effect around the sun, making it appear bright and radiant.

SUN SHADERS

Vertex Shader:

- The vertex shader modifies the positions of the vertices to create a pulsating effect.
- It uses a sine function based on time to simulate the dynamic surface of the sun.

Fragment Shader:

- The fragment shader handles the color and lighting of the sun's surface.
- It blends a base texture with noise to create a realistic, turbulent surface.
- Uses a sine function to adjust the brightness over time, simulating solar activity.

Noise for Realism:

- **Noise Function:**

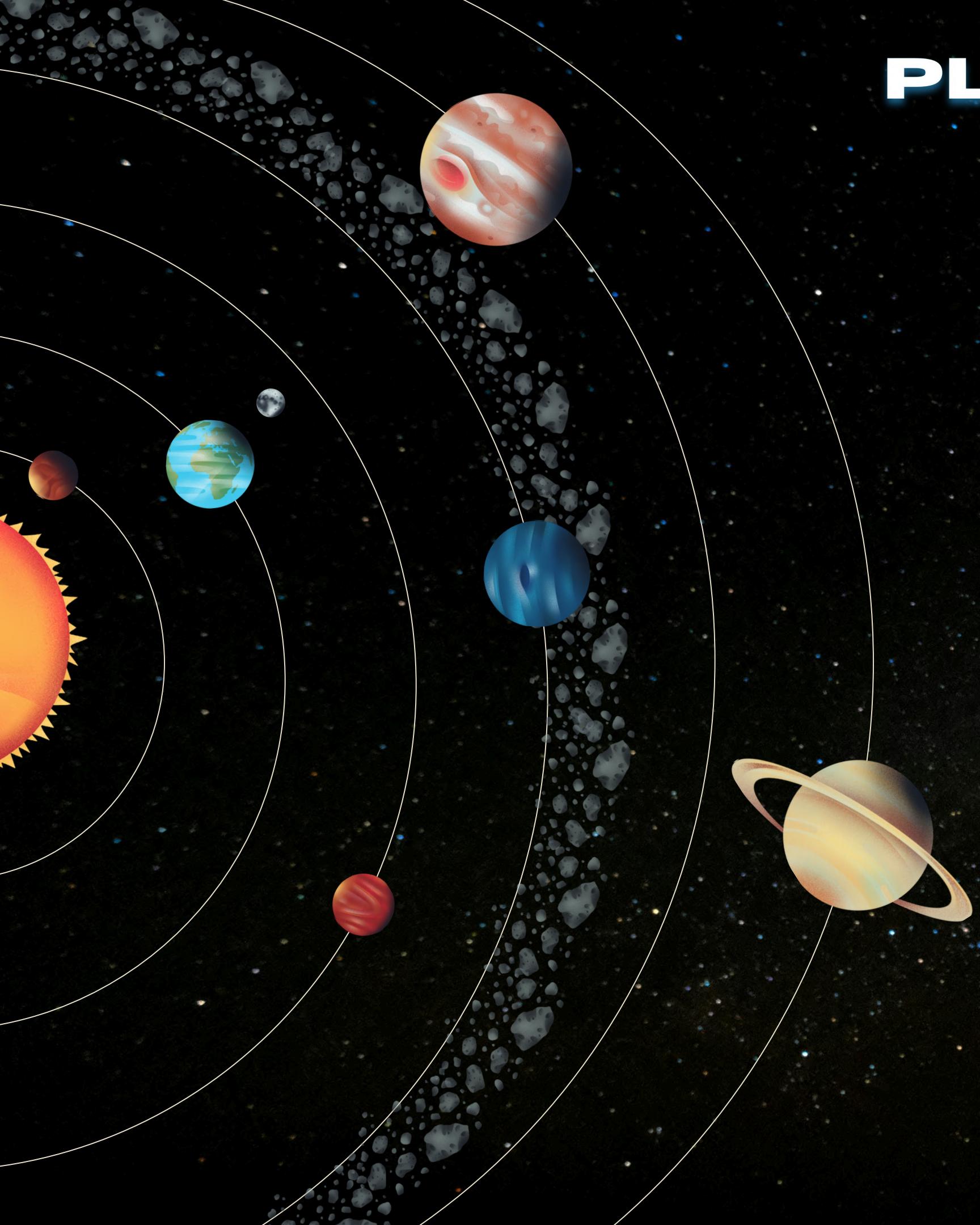
- A custom noise function is implemented to create irregularities on the sun's surface.
- This noise adds small variations to the color and brightness, making the sun look more natural.

- **Parameters:**

- Noise Frequency: Controls the scale of the noise patterns.
- Noise Amplitude: Controls the intensity of the noise effect.

```
vertexShader:  
    varying vec2 vUv;  
    varying vec3 vNormal;  
    varying vec3 vPosition;  
  
    void main() {  
        vUv = uv;  
        vNormal = normal;  
        vPosition = position;  
        gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);  
    }  
  
fragmentShader:  
    uniform float time;  
    uniform sampler2D sunTexture;  
    uniform float noiseFrequency;  
    uniform float noiseAmplitude;  
    varying vec2 vUv;  
    varying vec3 vNormal;  
    varying vec3 vPosition;  
  
    float noise(vec3 p) {  
        return sin(p.x + p.y + p.z + time);  
    }  
  
    void main() {  
        vec3 texColor = texture2D(sunTexture, vUv).rgb;  
        float noiseValue = noise(vPosition * noiseFrequency + time * 0.5);  
        float intensity = 0.5 + 0.5 * noiseValue * noiseAmplitude;  
        vec3 color = mix(vec3(1.0, 0.3, 0.0), vec3(1.0, 1.0, 0.0), intensity);  
        gl_FragColor = vec4(texColor * color, 1.0);  
    }
```

PLANETS



The simulation includes several planets with realistic textures and properties. Each planet orbits the sun based on its distance and revolution time.

Planet Data:

- Each planet is modeled with specific data including distance from the Sun, eccentricity, revolution time, and surface temperature.
- This data provides realistic orbital paths and behaviors for each planet in the simulation.

Dynamic Lighting:

- Each planet has its own dynamic point light source, enhancing realism by simulating how light interacts with the planet's surface.
- Light intensity and direction are adjusted based on the distance from the Sun.

Interactive Elements:

- Users can click on planets to get detailed information displayed in a tooltip.
- The camera can lock onto planets for close-up views, providing an interactive and educational experience.
- Planets move in their respective orbits (visually represented) around the Sun with varying speeds and eccentricities.

DYNAMIC LIGHTING

Dynamic lighting plays a crucial role in enhancing the realism and visual appeal for the 3D Solar System Simulator.

Point Lights for Planets:

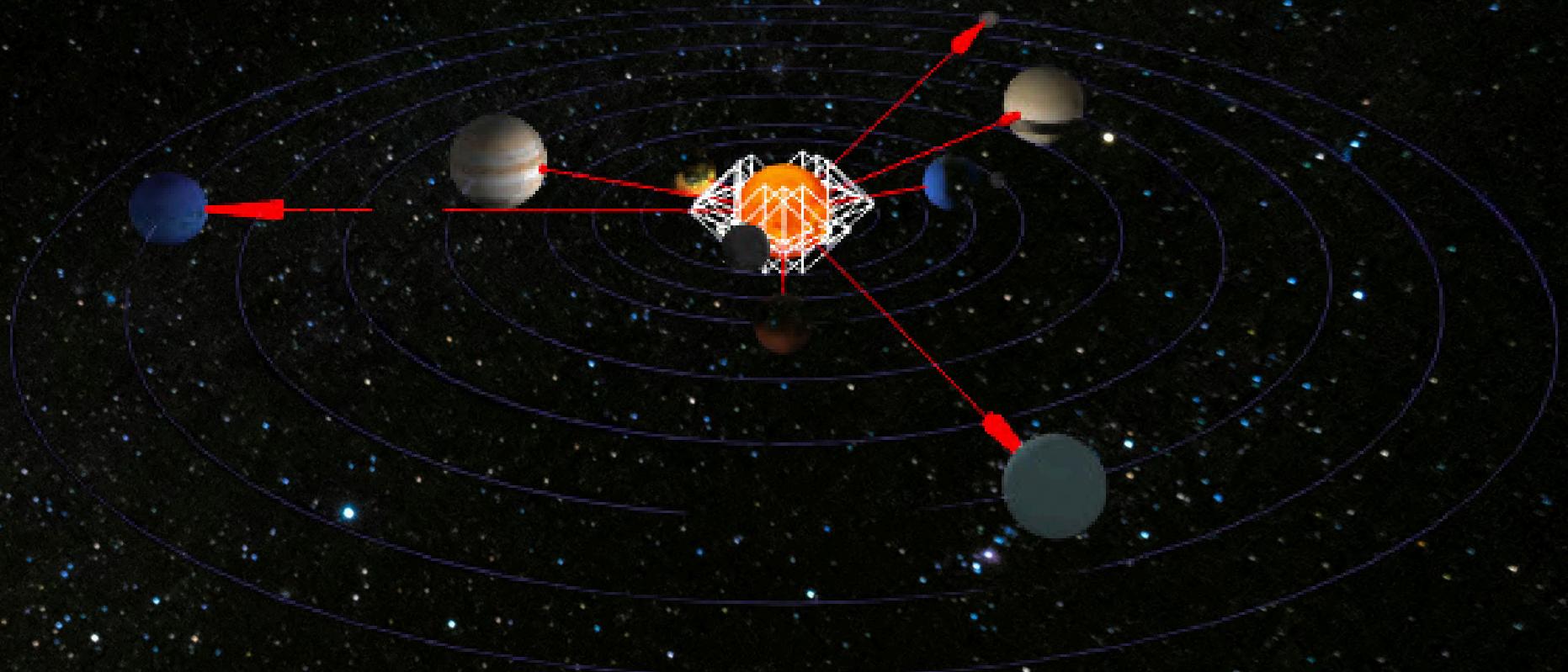
- The point lights simulate how sunlight interacts with each planet's surface, creating realistic lighting and shadow effects.
- Intensity and direction of light are adjusted based on the planet's distance from the Sun.

Adjusting Light Intensity:

- Light intensity is calculated dynamically based on the distance of the planet from the Sun.
- This ensures that planets farther from the Sun receive less light, mimicking real-world conditions.

Visualization Helpers:

- Light helpers and arrow helpers are used to visualize the direction and position of the lights.
- These tools are useful for development and debugging purposes, ensuring lights are correctly placed and oriented.



SHADOW MAPPING

Enabling Shadow Mapping:

- Implemented to create realistic and dynamic shadows for planets and other objects.
- Enabled in the Three.js renderer to allow shadows to be cast and received by objects in the scene.

```
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap; // Soft shadows for better quality
```

```
const pointLight = new THREE.PointLight(0xffffff, 3, 1000);
pointLight.castShadow = true;
pointLight.shadow.mapSize.width = 4096; // High-resolution shadow map
pointLight.shadow.mapSize.height = 4096; // High-resolution shadow map
pointLight.shadow.bias = -0.0001; // Shadow bias to reduce artifacts
scene.add(pointLight);
```

Dynamic Shadows:

- Shadows dynamically change based on the position of the light source and objects. As planets orbit the Sun, their shadows shift accordingly, adding a layer of dynamism and realism to the simulation.

Configuring Light Sources:

- Dynamic point lights for each planet are configured to cast shadows, enhancing the depth and realism of the scene.
- Shadow bias and shadow map quality are adjusted to minimize artifacts and ensure detailed and accurate shadows.

```
const time = Date.now() * 0.001;
planets.forEach((planet, index) => {
  const distance = planet.userData.distance;
  const angle = time * settings.orbitSpeed * (1 + index);
  planet.position.x = distance * Math.cos(angle);
  planet.position.z = distance * Math.sin(angle);

  // Update shadow position
  const pointLight = planet.userData.pointLight;
  if (pointLight) {
    const direction = new THREE.Vector3();
    direction.subVectors(planet.position, sun.position).normalize();
    const lightPosition = direction.clone().multiplyScalar(1).add(sun.position);
    pointLight.position.copy(lightPosition);
    pointLight.target.position.copy(planet.position);
  }
});
```

SPACE STATION AND ASTEROIDS

3D Model of the ISS:

- The ISS is represented by a detailed 3D model positioned near Earth.
- The model is scaled realistically to fit the simulation environment.
- Users can lock the camera view onto the ISS for close-up observation.
- The ISS orbits Earth, mimicking real orbital mechanics.
- Users can press 'I' to lock/unlock the camera view on the ISS.



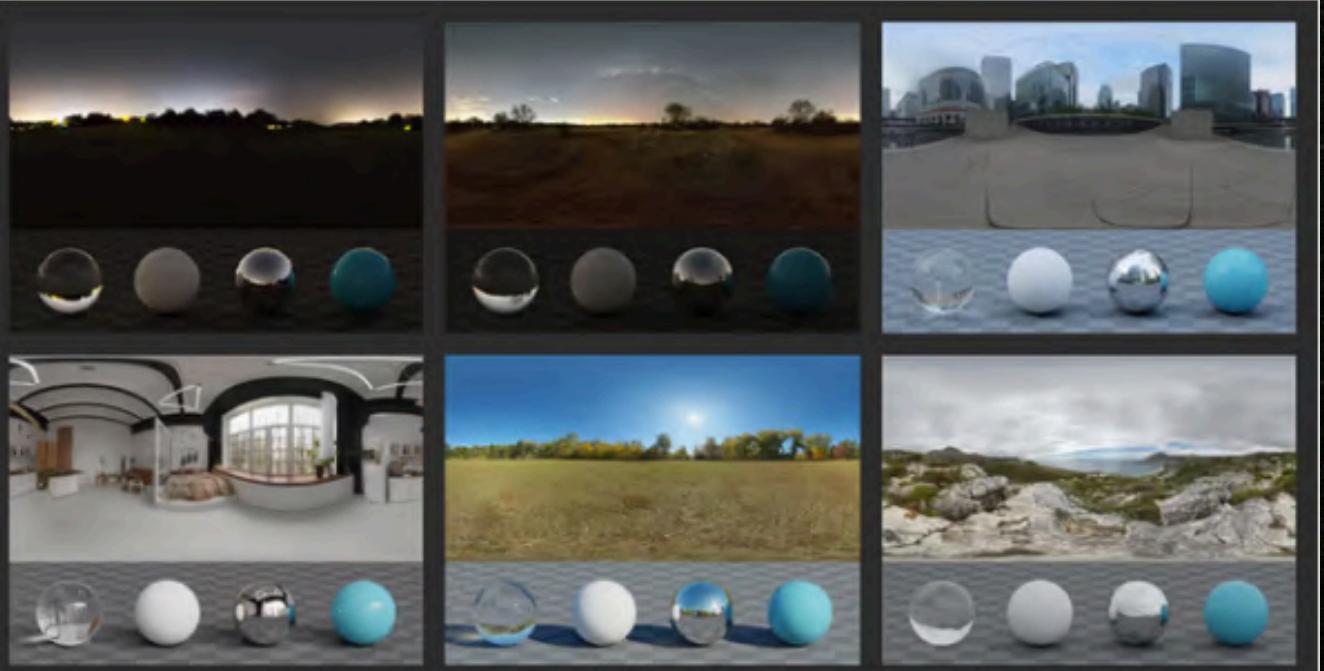
Asteroid Belt:

- The asteroid belt is simulated with numerous small 3D models, each representing an asteroid.
- Positions and movements are randomized for a natural appearance.
- Asteroids move randomly within the belt, simulating the dynamic environment of space.
- Each asteroid has its own speed and path, creating a realistic belt.

BACKGROUND AND ENVIRONMENT

HDRI Environment for Realistic Lighting:

- An HDRI (High Dynamic Range Imaging) environment map is used to simulate realistic lighting conditions.
- This environment map ensures that light reflections and ambient lighting match the space setting.



HDRI • Poly Haven

Previously known as HDRI Haven. Hundreds of free HDRI environments, ready to use for any purpose. No login required.

[Poly Haven](#)

Ambient Lighting:

- Soft ambient lighting is added to simulate the faint light present in space.
- This helps in highlighting objects without creating harsh shadows.

```
function loadHDRIEnvironment(THREE, renderer, scene, textureLoader) {  
  const pmremGenerator = new THREE.PMREMGenerator(renderer);  
  pmremGenerator.compileEquirectangularShader();  
  
  textureLoader.load('textures/space_hdr.hdr', (texture) => {  
    const envMap = pmremGenerator.fromEquirectangular(texture).texture;  
    scene.environment = envMap;  
    scene.background = envMap;  
    texture.dispose();  
    pmremGenerator.dispose();  
  });  
}  
  
loadHDRIEnvironment(THREE, renderer, scene, textureLoader);
```

CONCLUSION

Future Improvements:

1. Enhanced Planetary Details:

- Adding more surface details and features to planets and moons.
- Implementing atmospheric effects for planets with atmospheres.

2. Expanded Interactivity:

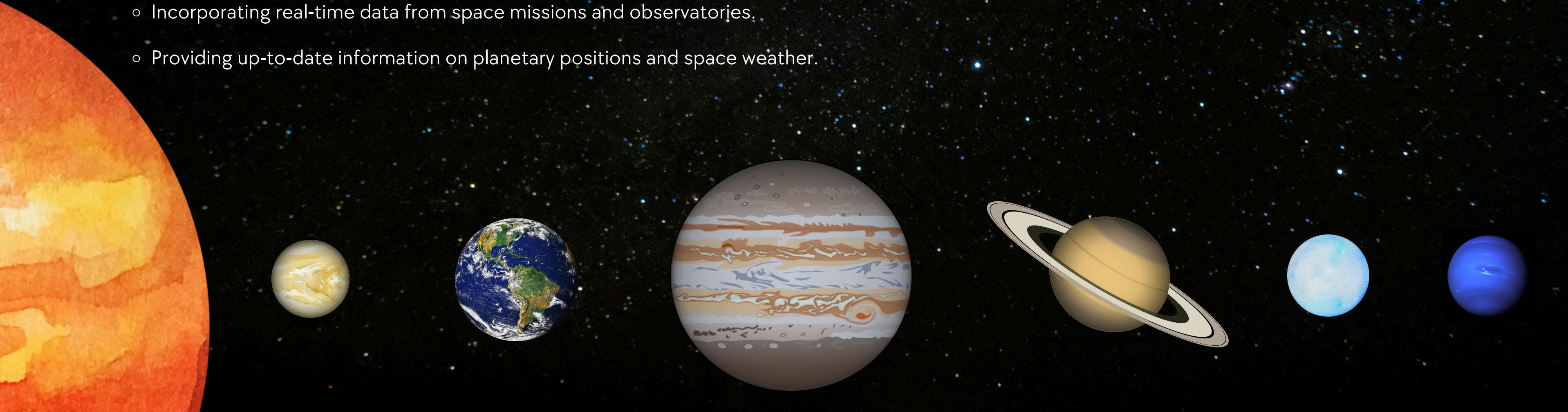
- Allowing users to take virtual tours on the surfaces of planets and moons.
- Adding more interactive elements such as comets and additional spacecraft.

3. Scientific Data Integration:

- Incorporating real-time data from space missions and observatories.
- Providing up-to-date information on planetary positions and space weather.

Key Achievements:

- Realistic 3D Models
- Dynamic Lighting and Shadows
- Interactive Elements





SAPIENZA
UNIVERSITÀ DI ROMA

DIAG

Dipartimento di Ingegneria
informatica, automatica e gestionale
Antonio Ruberti

**THANK YOU FOR
LISTENING!**

Gianmarco Donnesi
Matr. n. 2152311