Yalini Nadar
CSCI 6810

Galactic Explorer

Demo Link: https://youtu.be/PbjOScsf1BQ

Problems:

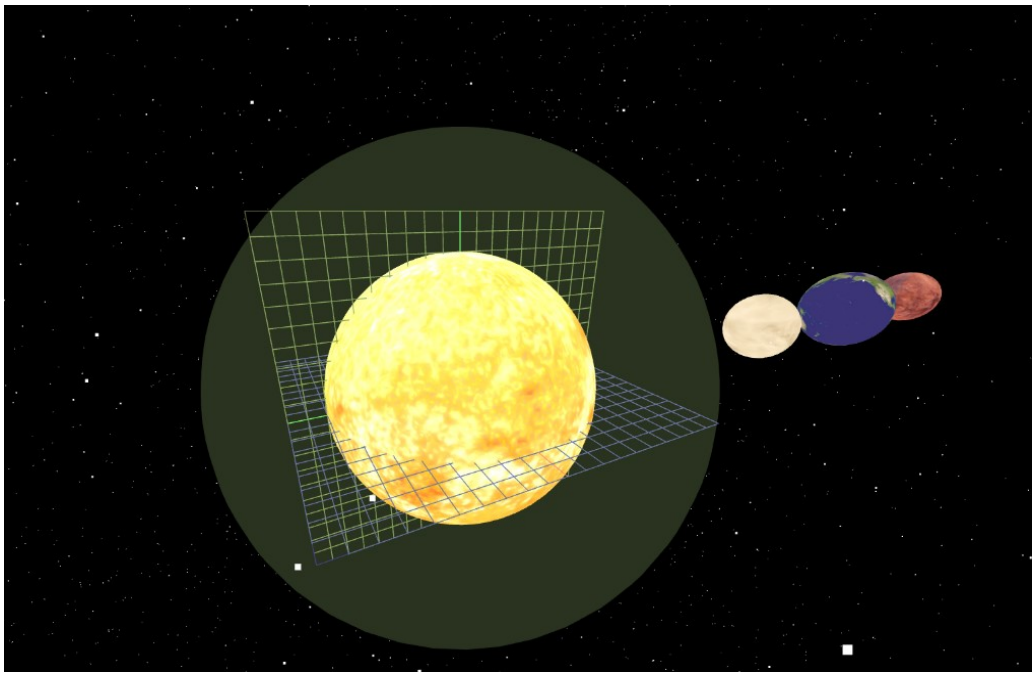
This project involves an immersive Galactic Explorer using THREE.js, showcasing various celestial bodies and interactive elements. The specific challenges addressed include implementing custom shaders for the sun's halo lighting effect, advanced lighting techniques for realism, importing a GLTF model for a spacecraft, interactive controls for its navigation, and incorporating particle systems for stars/asteroids, random galaxy generations with the ability to control a galaxies parameters dynamically. In this paper, I will break down the project's key components and functionality from a technical perspective.

Approach:

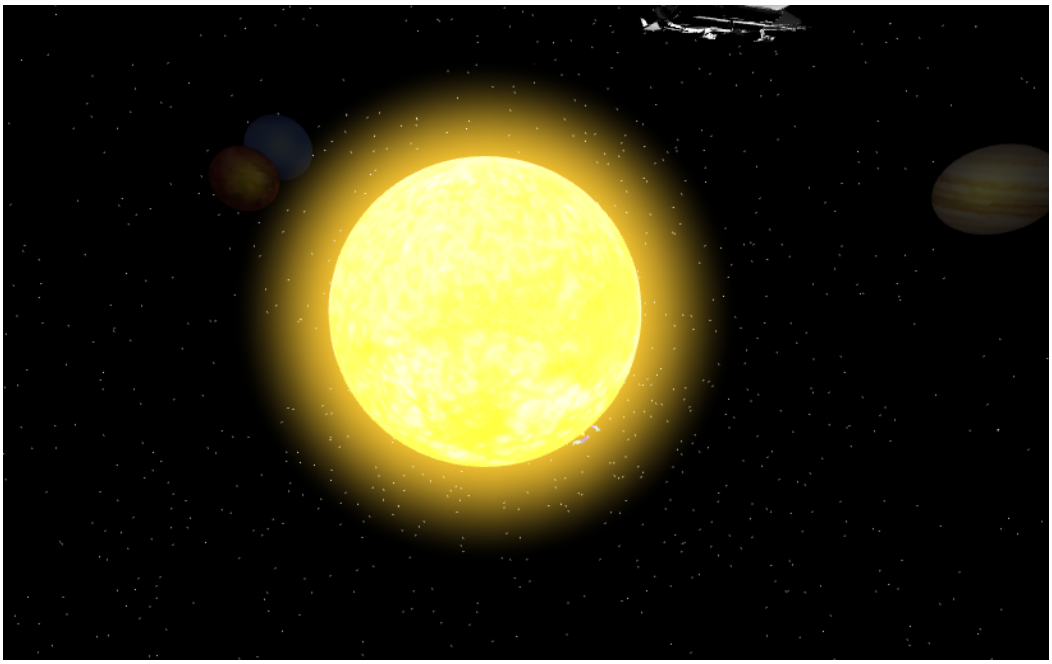
I organized the project by importing essential libraries and creating a modular structure, including a 'Planet' class for encapsulated logic and shader code stored in separate GLSL files. Adopting this modular approach ensures code reusability, facilitates changes, and enhances readability, with shaders fetched dynamically for efficient management.

Next, I added some more camera controls for the ease of camera manipulation. The user can control and move the camera using the arrow keys and also use the touchpad to zoom in and out. In addition to this, I imported a GLTF spacecraft model from Sketchfab with the ability to "fly" the spacecraft. Users can maneuver the spacecraft using arrow keys and 'W/S' keys, facilitated by the 'onKeyDown' function, dynamically updating the 'spacecraftDirection' vector. Upon key release, it resets the corresponding component of the spacecraft direction vector back to 0, indicating that the spacecraft should stop moving in that direction. I update the spacecraft's position in the update function based on the direction vector and default speed and also update the camera to follow the spacecraft's movement. To add another level, I added a button to reverse the spacecraft's body by space by 180 degrees.

Then I started creating the solar system with a sun positioned at the origin. I added an intense PointLight at the origin to represent the light that is emitted from a single point in all directions. For the glowing effect, I created another transparent and opaque mesh called 'halo' at the same position and scaled it to be 1.5x the sun mesh. This is what the halo was looking at that time:



I wanted the glowing effect to be intense near the origin and gradually decrease towards the outside so that the outline of the halo isn't as harsh as this picture. To do this, I created a custom vertex and fragment shader. The fragment shader calculates the intensity of light by considering the direction of the pixel's surface normal in relation to the light source. It also incorporates a smooth falloff effect based on the distance from the light source, creating a natural attenuation of light. The overall intensity is then adjusted and softened for a realistic appearance, and the final color is a blend of the base sun color and the calculated intensity, resulting in this:



As for the planets, I created the eight planets, each with their own textures that I found online. I did plan to implement custom shaders for each planet as well for realistic planetary surfaces; however, I later opted for a more straightforward approach and used the standard MeshPhongMaterial instead because the process of creating and managing custom shaders for the sun appeared to be more intricate and time-consuming than anticipated. I animated each planet to simulate the rotations of the celestial bodies around the sun by simplifying the math and incrementing each celestial body's 'rotation.y' by a fraction of an Earth year so that all rotations are synchronized to fractions of an Earth year.

Deviating from my initial plan of implementing custom shaders for planets, I shifted the project focus to particle systems. I wrote a function to generate a particle cloud, utilizing a buffer geometry for positions, and randomizing particle sizes and coordinates within a designated cubic space. These coordinates were then applied to the buffer geometry, and the particles were visualized with 'THREE.Points'. I incorporated mouse-based animation, allowing larger particles to simulate asteroid-like movement as the scene is navigated. This approach added dynamism and visual interest to the particle system, aligning with the overall project's creative direction.

I expanded the project by introducing a galaxy generator function, utilizing mathematical computations for point positions and colors. Polar coordinates, including the distance variable `x`, `branchAngle` for even distribution, and `spinAngle` for a spin effect, contribute to the position calculations. Random offsets (`randomX`, `randomY`, `randomZ`) introduce positional irregularity, while color interpolation between `colorInside` and `colorOutside` based on radial distance enhances the visual appeal. The `randomnessPower` parameter controls irregularity levels, ensuring a realistic and dynamic galaxy representation. GUI controls were implemented for dynamic parameter updates and real-time visualization of galaxy changes. With this galaxy generator function, I created another function to randomly generate galaxies with random parameters around the environment. This time introduced a rotated positioning, ensuring that galaxies form at unique angles, resulting in the creation of vibrant and beautiful superclusters each time the scene is rendered.

Finally, I added the 'UnrealBloomPass' as a post-processing effect to simulate an exaggerated bloom around the bright areas in the scene. I also added some GUI controls to dynamically change its parameters and see its effect on the scene.

Challenges Faced and Overcame

The first challenge was determining the optimal camera position for observing the static galaxy. I wanted to have a button that automatically adjusted the camera to focus on the galaxy. However, setting the camera directly at the galaxy's center resulted in an unsatisfactory zoomed-in view. To refine the position, I tried to manually adjust the x, y, and z values to find the best angle, which was annoying. Then I streamlined this process by adding a log camera button to the GUI which

would log the current camera position to the console, facilitating efficient position discovery and integration into the code.

To achieve realistic lighting effects, I removed all the lights, keeping only the sun's point light. However, I had issues loading the spacecraft, yielding a blank console. Suspecting loader issues, I implemented error handling with logs, only to discover that the model loaded correctly. It was invisible due to the absence of ambient lighting which I had to remove since it was affecting planetary illumination. To fix this, I strategically inserted point lights surrounding the spacecraft, dynamically updating their positions as the craft moved. This adjustment not only restored visibility but also simulated headlights effectively, enhancing the overall visual experience.

Effort Level/Justification

The completion of the project involved integrating complex elements such as 3D models, shaders, and interactive controls. The effort level is significant, considering the diverse functionalities and the need to manage complex interactions between all the components. The code base is also well-organized, leveraging functions and modular structures to enhance maintainability and readability. Here is a detailed list to demonstrate that my project aligns with our course objectives:

1. **Shader Programming:** Custom shaders written in GLSL within Three.js, specifically for the sun's halo lighting effect

2. **Lighting Techniques:** The strategic use of point lights for the spacecraft and a complex shader for the sun's glow, contribute to scene realism by emphasizing shadows, reflections, and natural light attenuation. Use of Phong material for planets to showcase distinct dark and light sides.

3. **Visual Effects:** The inclusion of particle systems for stars as well as the dynamic generation of galaxies with varied parameters, showcases creative visual effects integral to the immersive Galactic Explorer experience.

4. **Interactive Elements:** The project incorporates interactive elements through user-controlled camera movements, spacecraft navigation using arrow keys and reverse button.

5. **Animations:** Celestial bodies, including planets and stars, exhibit fluid rotations and movements.

6. **Physics and Mathematics:** Mathematical models, including polar coordinates and random offsets, are employed for the generation of galaxies, demonstrating the integration of physics and mathematics for realistic and procedurally generated celestial patterns.

7. **Rendering Optimization:** The project employs optimization strategies to ensure efficient rendering, considering the capabilities and limitations of the Three.js framework, contributing to a smooth and responsive user experience.

8. **Graphical User Interface (GUI):** A visually cohesive GUI is designed and implemented with interactive buttons for camera positioning, spacecraft control, and galaxy and bloom parameter adjustments.

9. **Post-Processing:** Implemented UnrealBloomPass and integrated GUI for real-time adjustment of bloom intensity, threshold, and radius for a polished and refined visual experience.

Overall I spent about 2 weeks (started on 11/30) working on this project and 5 hours per day. So I did approximately put in 1.75 times the effort of P3.

Future Work Plans/Reflections

My approach was mostly successful as I was able to deliver my proposed project on time with most of the features I had planned. For future enhancements, I can incorporate more sophisticated planetary shaders to achieve realistic surface details. I can also explore integrating more diverse and interactive elements such as planetary rings, asteroid belts, or interactive constellations. Furthermore, I could delve into enhancing the procedural galaxy generation by incorporating additional parameters and controls, allowing users to customize and explore an even broader range of celestial formations. I do plan to continue researching and working on this project beyond this class.