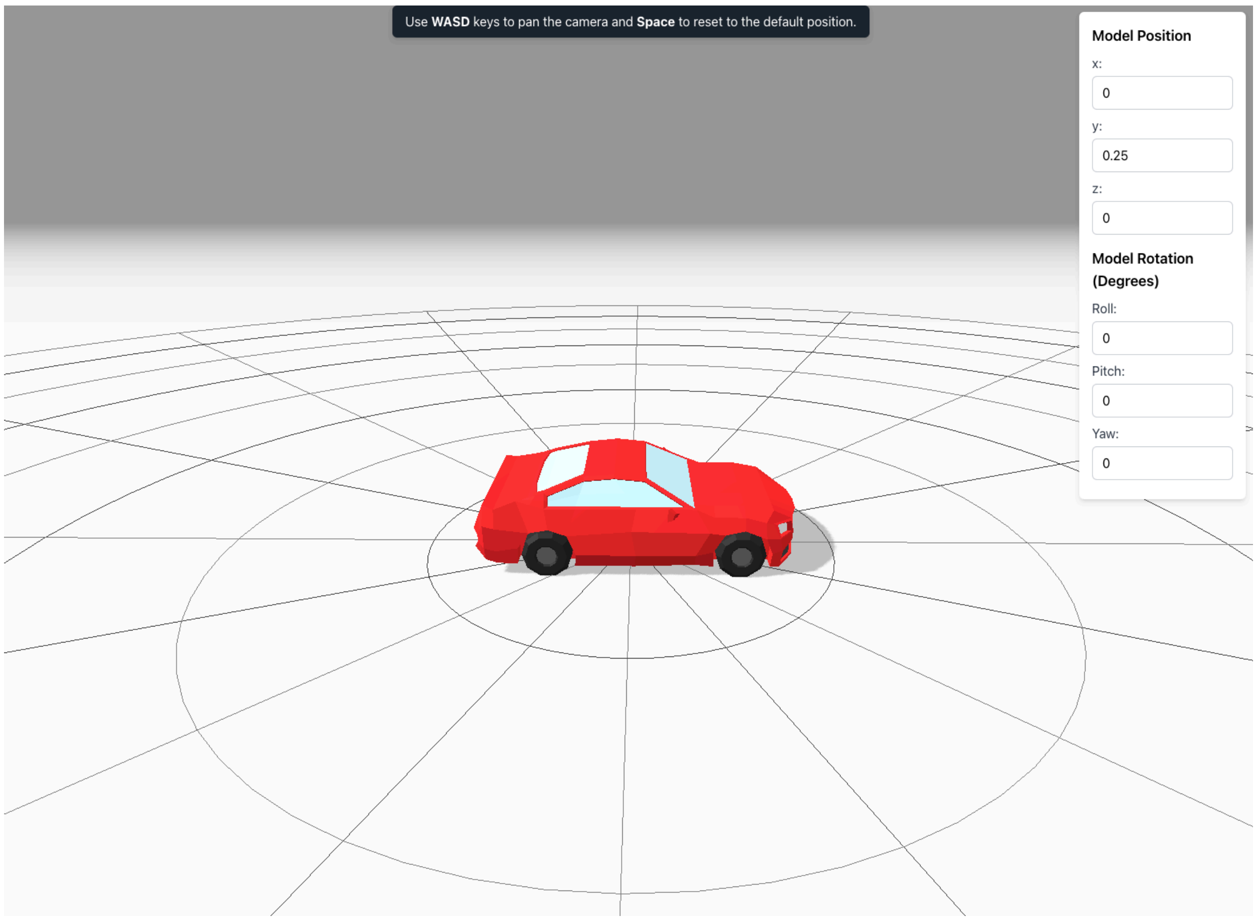
Coding Assignment Report - Seoul Robotics

Introduction

This report outlines my approach and thought process for the Frontend Engineer coding assignment at Seoul Robotics. The main objective of the assignment was to set up a 3D scene in a React application and create an interactive user interface for modifying the model's position and rotation.


Use **WASD** keys to pan the camera and **Space** to reset to the default position.

Model Position

x: 0

y: 0.25

z: 0

Model Rotation (Degrees)

Roll: 0

Pitch: 0

Yaw: 0

Demo Link: https://seoul-robotics-benjamin.netlify.app

Technologies Used

For this project, I utilized the following technologies:

- **React**: Chosen for building the user interface due to its component-based architecture and my previous experience.
- **Three.js**: Used to create and render the 3D scene, as it is powerful and simpler to use than Babylon.js for smaller projects, and I had experience with it from previous work.
- **Tailwind CSS**: Implemented for styling the interface quickly and efficiently.
- **Jotai**: Used for state management to keep the application state simple and reactive. Although not required for the project, I chose it to facilitate data sharing between the user interface and the Three.js scene, based on my past experience with it.
- **Netlify:** Used for deployment, which made it easier to share the project, even though it was not a requirement.

Resources and References

I only used the official Three.js documentation and example projects to guide me and refresh my knowledge on setting up 3D scenes and camera controls. These resources were very helpful during development.

Approach

Part 1: Setting Up the 3D Scene

- **Scene Creation**: I initialized a Three.js scene and configured it with a perspective camera and responsive renderer.
- **Grid Helper**: Added to help with scene orientation.
- **Model Loading**: I initially used a simple cube for the 3D model but later switched to a more complex car model for better visual results. Loading this detailed model was challenging, so I opted for a simpler car model (found here). I also had to update the lighting and shadow casting setup compared to the simple cube.
- **Lighting**: I added directional and hemisphere lighting for a balanced and realistic visual effect.
- **Camera Controls**: Integrated OrbitControls to enable user interaction with the camera through mouse and keyboard inputs.

Part 2: Building the User Interface

- **Position Controls**: Developed input fields for modifying the x, y, and z coordinates of the model.
- **Rotation Controls**: Added input fields for adjusting roll, pitch, and yaw angles. This was easier to implement with Three.js as it provides functions to rotate the model directly.
- **Real-Time Updates**: Ensured that changes in the UI reflect immediately in the 3D scene by using Jotai state management.