Demo Environment Development Process:

- Fusing THREE.js and React:
    - I had experience with THREE.js, but I had never used React or any other JSX library for UI.
    - I discovered React-Three-Fiber, a JSX Library that combines THREE.js into a React environment
    - Although I wasn’t used to the tag-based nature of JSX, I was able to understand it with practice.

- CSS
    - My experience with CSS was limited since most of my work was focused on Javascript and THREE.js. However, being a fairly simple language, I was able to understand it’s workings and combine it with JSX in order to align my UI and controllers where I wanted them inside the React-Three-Fiber scene, as well as modify the appearance of my HTML.

- Creating scripts that spoke to each other
    - Connecting the Experience.jsx script, which contained my THREE.js elements, and the Interface.jsx script was one of the processes that took the longest to implement. Although I first tried “moving the data up” from Interface.jsx into index.jsx, and then pulling it into Experience.jsx in order to update my meshes’s values, I found it more efficient to use a state-management solution called “zustand”. Although the logic in order to implement it requires a lot of function calling, eventually I figured out how to pass arguments and values from Interface.jsx into a GlobalState.jsx file, and pulling those parameters into Experience.jsx to implement them.

- Finishing touches
    - I added other features such as moving the position of the Light, as well as casting shadows. Moreover, I added OrbitControls, as well as PivotControls for the sphere.
    - Also, I added HTML text into the React-Three-Fiber environment and anchored it to the sphere. In addition, I included a more aesthetically pleasing text that is animated to move as if it was floating.
    - Finally, I included a Title, my name, the MIT logo, as well as the title of the project and a personal logo I made some time ago on the HTML head tag.

- Features that could be implemented
    - Camera raycast boxes that allow you to zoom into each object when it is clicked
        - Could be done creating a raycastManager class that handles all the raycasting logic
        - Workload estimate: 5-6 hours
    - Limiting the OrbitControls to stop the camera when it is level with the floor
        - Could be done by setting the minPolarAngle of the OrbitControls
        - Workload estimate: 30 minutes
    - Adding particles that move around randomly in the scene
        - Would probably require the use of shaders, which are complex, but once done, it would be fairly easy to make them randomly spawn and despawn with the Math.random class
        - Workload estimate: 4-4.5 hours
    - Making UI controllers to move around the scene
        - Adding arrows that allow you to move up, down, left, and right, forwards, and backwards in the THREE.js environment
        - Would be fairly easy to implement using the same globalState logic implemented in the project. Would only need good looking buttons from a React library such as MUI.
        - Workload estimate: 1 hour
    - Toggling the UI as visible or invisible for a more aesthetic view (or a timer that makes the UI invisible after a certain amount of time without interacting with the environment and would reinstantiate it once it is interacted with)
        - Using Event Listeners would be fairly simple to implement a timer that starts when the user stops interacting with the scene, and a reset once they interact with it again
        - Worload estimate: 2-2.5 hours