Overall Project

In our project, “Handsy Hands”, we experimented with the technologies needed to build virtual reality applications. To accomplish this, we used WebXR, which allows us to build web-based VR applications. Since we sought to explore the many directions this could go, we built many scenes, most prominently, a lighting demo, a playable piano, and “hands on hands”.

We implemented our scene selector as a simple webpage with a dropdown menu allowing the user to select their desired scene. With the click of a button, the scene loads up, immersing the user in the VR experience.

WebXR is an API that allows websites to interface with VR and AR devices. While this is somewhat general, we focused on developing for the Oculus Quest 2 and experimenting with its Hand Tracking API. WebXR allows for the creation of content in WebGL.

For this project, we implemented some basic objects as well as an entity component system. We built cubes, cylinders, and spheres. Our lights utilized the Phong illumination model and are colored as shown in the lighting demo. The ECS allowed us to nest objects within other objects in a tree hierarchy, easing development. The ECS also supported “containers”, which contained many other objects. Notable examples of this are the piano and hands.

We also implemented a generic vertex and fragment shader that every model uses. Every object has to opt in to colored shading and lighting when it’s first created.

The hand tracking necessary for our hands use the Hand Tracking API provided by the device. The device takes care of detecting the world space positions and orientations of every joint in our hands and provides a full transformation matrix through the API. We placed spheres at the joints of the hands and cylinders between them to represent bones, emanating from the wrist.

Our piano was built out of a series of scaled cubes. Keys are “pressed when a user’s fingertip intersects the key’s bounding box. To visually indicate this, the key is rotated slightly down (as in a real piano) and marked in blue. To play the notes of the piano, we used the Web Audio framework Tone.js’s piano library.

For testing, we used the Chrome Extension “WebXR API Emulator” and for later interactions involving hands, we used a Oculus Quest 2. We found that for testing, it was useful to place a piece of tape over a sensor so the Quest would believe it was being worn. Additionally, we could watch the results by screencasting to a TV.

Technical Detail

WebXR provided access to the headset, perspective/view matrices, and hand transformation matrices. The key difference between typical WebGL projects and WebXR is that for VR compatability, we must render two images from slightly different camera angles, one for each eye. Using the matrices associated with each eye, we render each half of the view, creating an immersive, 3D environment for the user.

The hands were provided as 25 points on each hand, 5 joints on each finger, 4 joints on the thumb, and one central wrist joint. These positions were provided in global coordinates, requiring some simple math to move them, as done in the “hands on hands” scene.

Cubes were built with a simple trimesh while cylinders were built as extruded n-gons with interpolated normals. To build spheres, we built a NxNxN lattice cube, then normalized all the vertices to magnitude 1, building a rough sphere. Using the vertices as the normal vectors, we were able to interpolate a smooth surface.

- All scenes can be found in src/scenes
- Basic solids can be found in src/render/entity/solids
- Code related to hands, lights, and the piano can be found in src/render/entity/common
- All shaders can be found in src/render/shaders
    - VS_verts is a generic vertex shader
    - FS_generic is a generic fragment shader
    - We used a GLSL plugin to allow shader snippet imports
        - https://www.npmjs.com/package/vite-plugin-glsl

Limitations

There were a number of limitations, largely related to the VR technology itself. The Quest 2 is an untethered headset, so it performs all its computation on its own hardware. Unfortunately, the associated weight/size requirements result in hardware with fairly limited computational power. Additionally, any lag in the VR experience was noticeable and disorienting, so dropped frames are a much larger problem than in general graphics. Ideally, as processors improve, this will be less of an issue.

These limited resources meant that we had to limit the complexity of our scenes. We had to make sure that we weren’t using too many triangles. This resulted in relatively simple scenes.

We didn’t light the piano scene due to the computational complexity, but the scene still looked good without proper lighting. Additionally, there are some typical usability issues, where the lack of tactile feedback make it somewhat difficult to actually play the piano.

The biggest limitation was the difficulty of debugging. While using the Chrome extension helped, when crashes occurred on the headset, it was unclear how to resolve them.