# CS Capstone Handoff Document

June 7, 2019

# Creating Immersive Experiences on the Web using VR and AR.

Prepared for

## Intel

Alexis Menard _____ _____
*Signature* *Date*

Prepared by

## Group 47

Brooks Mikkelsen _____ _____
*Signature* *Date*

Evan Brass _____ _____
*Signature* *Date*

Jonathan Jones _____ _____
*Signature* *Date*

Brandon Mei _____ _____
*Signature* *Date*

Tim Forsyth _____ _____
*Signature* *Date*

**Abstract**

This document contains all the documentation generated over the course of this project. It explains the project itself, the requirements, the design, the tech reviews, blog posts and reference guides for understanding this project.

# CONTENTS

# 1 Introduction

The WebXR Physics project was requested by our client Alexis Menard at Intel. Its purpose is to provide both an educational physics VR application and to be an example for future WebXR applications. It is most important in its demonstration of the WebXR capabilities, specifically in its ease of use. The OSU students tasked with this project were Jonathan Jones, Tim Forsyth, Brooks Mikkelsen, Brandon Mei and Evan Brass. Neither team member had any specific role for this project but each contributed towards implementing the WebXR API, graphical visuals, physics and project management / integration tools. The client was very active, communicative and passionate about this project. What follows is all the documentation that was created throughout the course of this project.

# 2 Requirements Document

CS CAPSTONE REQUIREMENTS DOCUMENT

OCTOBER 30, 2018

# CREATING IMMERSIVE EXPERIENCES ON THE WEB USING VR AND AR

PREPARED FOR

INTEL

ALEXIS MENARD _____ _____
*Signature* *Date*

PREPARED BY

GROUP 47- WEBPHYSICSVR

JONATHAN JONES _____ _____
*Signature* *Date*

EVAN BRASS _____ _____
*Signature* *Date*

BROOKS MIKKELSEN _____ _____
*Signature* *Date*

TIM FORSYTH _____ _____
*Signature* *Date*

BRANDON MEI _____ _____
*Signature* *Date*

**Abstract**

This document is a Software Requirements Specification (SRS) that outlines the purpose, scope, and technical requirements of the WebPhysicsVR immersive web experience hosted by the Intel 01.org open source portal. This document describes the website functionality and provides a tentative timetable for the fulfillment of project goals and requirements.

# CONTENTS

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this project is to provide a technical demonstration of the WebXR API capabilities. It will be an example of how WebXR can be used to create an entertaining, interactive VR experience within a web browser. It will have a focus on education and act as a virtual physics lab.

## 1.2 Scope

WebPhysicsVR will be an interactive physics simulation that utilizes the WebXR API. Its primary use is as tech demo and a platform for education.

As a tech demo, our project will incorporate all the major features of WebXR. We'll want to faithfully present the physical concepts in an educational and engaging fashion while also focusing on designing a quality WebXR experience.

At a minimum, users should be able to visualize the effects of gravity. If we have time to do so, we could visualize other forces like adding friction between objects or giving certain objects magnetism as well as mass.

We intend to support a spectrum of VR devices though we will only test on a few selected devices that we have access to.

## 1.3 Product overview

### *1.3.1 User Interfaces*

Users with compatible hardware will be able to experience an immersive/enhanced version of the website right when they access it, with a click of a button.

Within the VR experience, we will have interfaces for the user to manipulate the objects and physical constants. These interfaces will be displayed in the most contextually meaningful location in the environment. This might mean different floating icons around the 3d objects, floating interfaces or control objects that are within the environment.

### *1.3.2 Hardware Interfaces*

WebXR is compatible with a range of XR devices, web browsers and operating systems. This includes mobile devices. For the full intended experience, an HMD with body and controller tracking devices should be used.

### *1.3.3 Software Interfaces*

- WebXR API
- WebGL 2.0
- (Some physics API)
- If needed:

  - Web Workers
  - Web Assembly (for physics / processing intensive code or porting non-JavaScript libraries)

#### *1.3.4 Functions*

The main functions of this website are:

- VR Experience accessible without need to download client side software
- Interactive physics environment

### 1.4 Definitions

- XR : Family of hardware devices capable of Virtual Reality and Augmented Reality.
- API : Application Program Interface.
- HMD : Head mounted display. There is an optic lens for each eye which runs at 90Hz.
- WebXR : Open source web API for accessing virtual reality (VR) and augmented reality (AR) devices, including sensors and head-mounted displays.
- WebGL : Open graphics API.
- Web Worker : API for performing and communicating between parallel computation on the Web.

## 2 SPECIFIC REQUIREMENTS

### 2.1 External interfaces

- Users may interface with the website using:
  - Mobile devices with positional tracking
  - Head mounted displays, whether they are opaque, transparent, or utilize video pass-through
  - Fixed displays with head tracking capabilities
- Users will be able to interact with objects in the environment through their controller (or simulated/virtual controller)
  - Translate objects
  - Scale objects
  - Rotate objects
  - Pick up/grasp objects
  - Drop/throw objects
- Users must be able to change physical constants. Potential mechanisms for doing that:
  - Physical Constants Dashboard with levers/switches/dials to change the environment
  - Floating user interfaces around the objects with icons/bars and other traditional UI features
- User can ”pause/play”.

  - During pause mode, objects are not animated and are not affected by the physics engine.
  - During play mode, objects are animated and are affected by physics.
- User can ”freeze” objects, stopping all kinematic forces that the object is experiencing
- User can spawn in a multitude of objects stored on the website through a spawn menu

### 2.2 Functions

- Utilize all core features of WebXR including:
  - Headset location and orientation
  - Controller location and orientation
  - Multi-platform compatibility
  - Parsing input from XR devices
- The website will load the correct VR environment depending on the hardware that the user has.
- Website will pause when a controller has been disconnected and resume when the connection is reestablished.
- Website will notify user if web browser/hardware is not compatible with WebXR
- Render environment and physics in real time

### 2.3 Usability requirements

Offer a way of manipulating objects in the simulation for all supported platforms. On a phone this might mean having a virtual manipulator at a fixed location from the user’s face. On devices like the Oculus Rift, this would probably mean the hand controllers.

We want to pair a non-technical-user friendly physics demonstration with a technical-user friendly code base.

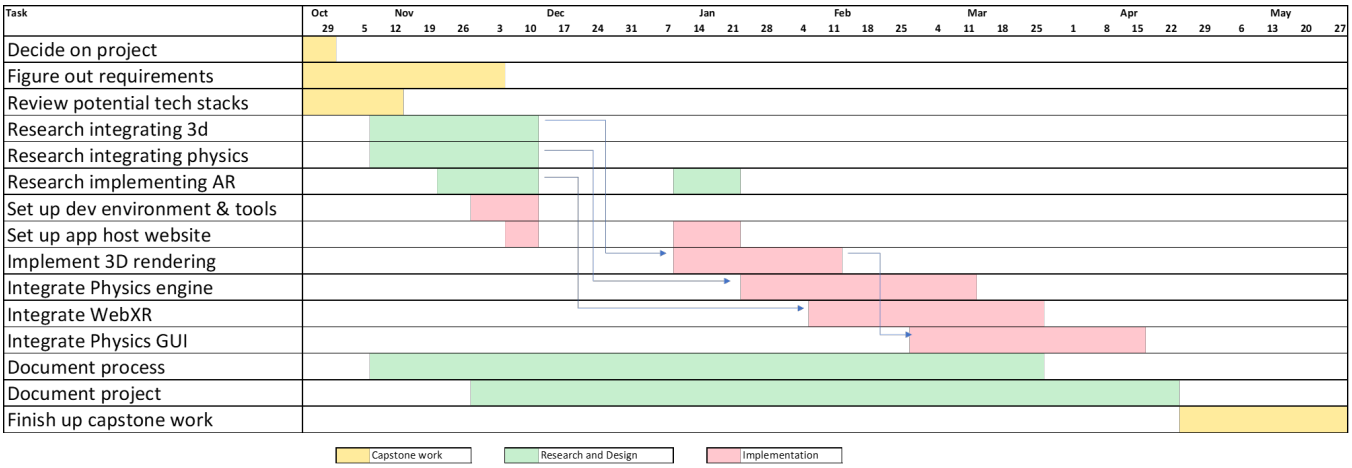
### 2.4 Performance requirements

- Latency must not exceed 11 milliseconds within the simulation.
- Re-factor to distribute to multiple web workers if needed to achieve performance requirements.
- Meet an average 60 frames per second, appropriately degrading the content to meet that budget on mobile phones and other low powered devices.
- HMDs should be running at 90 fps

### 2.5 Software system attributes

- Landing page will be compatible with all evergreen browsers
- Website will be compatible with browsers that support WebXR
- Website will be compatible with (List VR/AR devices here)
- Website will run on multiple devices and operating systems (Windows, Android, Mac, iOS)

# 3 GANTT CHART



# 3 Design Document

CS CAPSTONE DESIGN DOCUMENT

APRIL 17, 2019

# CREATING IMMERSIVE EXPERIENCES ON THE WEB USING VR AND AR.

PREPARED FOR

INTEL

ALEXIS MENARD _____ _____
*Signature* *Date*

PREPARED BY

GROUP 47

BROOKS MIKKELSEN _____ _____
*Signature* *Date*

EVAN BRASS _____ _____
*Signature* *Date*

JONATHAN JONES _____ _____
*Signature* *Date*

BRANDON MEI _____ _____
*Signature* *Date*

TIM FORSYTH _____ _____
*Signature* *Date*

**Abstract**

This document is a software design specification that is intended to outline the various components used in the creation of the WebPhysicsVR simulation. There is a list of requirements that must be met for this project, their technical solutions, descriptions of those technologies and rational for using them.

# CONTENTS

| Section | Original | New |
|---|---|---|
| 2.1.2 | Objects in the scene must be interactive through translation, scaling, rotation and collision. | Objects in the scene must be interactive through translation, rotation and collision. |
| 2.4.1 | Stretch: Content may be ”installable” to make it available offline. | (Removed)<br>• Content is technically installable via use of the dev environment and localhost but an installable feature from the website itself is a stretch goal that is not needed for this project. |
| 2.5.1 | Users must be able to choose to display information about an object, such as velocity, acceleration, and mass. | Users must be able to visualize velocity vectors on moving objects in the kinematics scene.<br>• The original idea was to use a GUI interface such as DATGUI VR but an API such as this is constructed for the WebVR API and not WebXR.<br>• We settled on using a custom text display system but it was not optimized enough to handle updates for every frame. For that reason, attribute displays were mostly omitted for objects. |
| 2.5.2 | A menu must be available for users to switch stations or select objects. | Users can interact with the scene to traverse through it and into other scenes.<br>• Rather than traversing through the scenes via a user interface, we found a better alternative in users actually interacting with objects in the scene to traverse. |

| | | |
|---|---|---|
| 2.6.1 | A scene can be paused/frozen. | (Removed)<br>• It no longer seemed practical to add this functionality since we were limited to a single button press on a controller. We decided that the user experience would be too low to trying to aim their controller at an interface on the wall to pause the scene. |
| 2.6.2 | An object can be paused/frozen. | (Removed)<br>• It no longer seemed practical to add this functionality since we were limited to a single button press on a controller. We decided that the user experience would be too low to trying to aim their controller at an interface on the wall to pause the scene. |
| 2.7.4 | Users must be able to interact with objects in the scene using a mobile touchscreen interface. | Users will be able to select objects in the scene using a mobile touchscreen interface.<br>• Unfortunately, the WebXR API does not support drag and drop events for touchscreens, so we are only able to implement select functionality (tapping objects). |
| 2.8 Web Hosting | Users must be able to host directly from the GitHub repository. | The project will be published on a website using a Now deployment.<br>• We are not using GitHub pages, instead we're using Now deployment. |
| 2.9 Version Control | Users must be able to edit, push, pull request, and the changes are live. | (Removed)<br>• This was meant for the developers, not the users. Does not need to be in the design document. |

| | | |
|---|---|---|
| 3.3 | • In this section, ways of optimizing our site were explored.<br>• At the time, we were unsure of the web capability for VR experiences and sought ways to prevent exceeding our frame budget. | (Removed)<br>• During the course of the project, we quickly discovered that tools like ThreeJS and the WebXR API took care of multi-threading and optimization for us.<br>• There was no need on our end to implement performance boosting tools. |
| 3.4 | Service Workers expose the browser's caching system to developers. Our service worker can prime the browser cache with VR assets and respond to asset requests when disconnected. To describe to the browser what modules and assets will be needed later, we will use HTML meta elements. On parallel serving protocols like (G)QUIC and HTTP 2 the browser can load those assets over a low priority stream or maintain a connection to the server while waiting for the app to request those assets. We will also use responsive images to give the browser multiple options for an image source and it will pick whichever one would load best in terms of resolution, supported format, etc. | Many 3D and 2D assets will be used to style this site from a variety of third party sources. In the interest of providing a smooth user experience, there needs to be some overhead when loading assets into a scene. For this task, we'll employ a script of our own to asynchronously load into our scenes the assets that we need. During the time in which the scene assets are loading, a loading screen will be displayed to keep the user experience clean.<br>• Service workers and web assembly were an overcomplication of a simple task. Loading assets into the scene was easily achieved through the use of the ThreeJS API. Caching our objects was as easy as keeping a reference to their loaded instances between scenes. |
| 3.5 | Spatial audio is required to achieve full immersion into the physics simulation. Audio clips must be played when events happen in the scene.<br>• This section originally handled the Audio component, which has been omitted from the final product. | (Removed)<br>• This project no longer uses audio components. |

| | | |
|---|---|---|
| 3.6 | The web hosting services must able to host static web pages for GitHub users to publish content. GitHub Pages can also be served securely which is a requirement for using service workers. | The site will be hosted on the NOW hosting service provided by ZEIT.CO for testing and deployment.<br>• We opted for using NOW deployment instead of github pages as it offers more testing capability and a wider selection of deployments to view throughout development. |
| 3.8 | • Originally referenced Jekyll as the publishing system. | • Changed Jekyll to ZEIT as that has been our publishing system throughout deployment. |
| 3.9 | • Old version specified typescript as the primary language for this project. | • Types are not available for WebXR<br>• Changed to Javascript |
| 3.12 | • Listed AFrame as a component. | • AFrame was never used. Unnecessary component.<br>• Removed AFrame. |
| 3.13 | • Listed AFrame as a component. | • AFrame was never used. Unnecessary component.<br>• Removed AFrame. |

| | | |
|---|---|---|
| 3.14.3 Personal Computer | The product should be playable on a computer browser with the ability to switch to a full VR experience if the appropriate hardware is detected. | The product can be run on a computer browser with the ability to switch to a full VR experience if the appropriate hardware is detected.<br>• Removed mouse interactions since the WebXR APIs input source did not support mouse clicks. Since this project is focusing on the WebXR API, we decided it was not necessary. |
| 4.3 | A physics simulation should have sound for it to be immersive in the VR world. The applicaiton will use the WebAudio API to deliver event driven spatial sounds to the simulation. HowlerJS is an API for WebAudio that makes its implementation easier. Spatial audio is supported as well as event playback. | (Removed)<br>• This requirement was removed so we could better focus on demonstrating the capabilities of WebXR.<br>• Therefore, this Design View was removed as well. |
| 4.4 | The hosting will need to host static web pages for GitHub users, blogs, and project documentation. We will be using GitHub pages as there are no required databases to setup and no server to configure. It also needs to serve all of the project site from a personal URLs that is tied to an organization or GitHub account. The GitHub pages will automatically build and deploy our site. | Zeit Now will be used to host the website as we develop it so that our progress can be shared with the client and other team members. It allows us to set up continuous deployment for static websites easily. Zeit now also allows us to create a unique and lasting deployment for each commit that is pushed to GitHub. This will allow us to share individual progress with stakeholders in the project.<br>• Changed from using GitHub to host the website to Zeit Now.<br>• Now offered us better customization and the ability to create individual deployments per commit, so we could reference a specific version of the website. |

| | | |
|---|---|---|
| 4.6 | The hosting sites must converts the files into a static website that can be deliver with any standard web servers. That means it needs to automatically generate the HTML code in the background if there are changes that are made to the files. Jekyll is a engine of GitHub Pages that can be use with each other, while it provides updates and HTML rendering. | (Removed)<br>• Bundler - Parcel - Does this, and we tested hosting our project from another server called Caddy<br>• This optional requirement has been removed.<br>• It was originally for if we wanted to blog (Jekyll's primary goal) about our experiences using WebXR. |
| 4.7 | The application must run in modern browsers that have implemented the WebXR API. That means that the code must compile down to JavaScript (or be written in JavaScript). We will be using TypeScript to write code for the client side application. TypeScript is a language written by Microsoft that provides static typing on top of traditional JavaScript syntax. It can be compiled down to JavaScript for use in both Node.js and in browsers.<br>Rust is a systems programming language founded by Mozilla for designing programs with safe memory sharing and simpler parallelism without the usual trade offs of a garbage collector or heavy runtime. | We will be using JavaScript for the client side application because it is what is supported by WebXR and ThreeJS. Unfortunately, ThreeJS does not have updated TypeScript types and WebXR has no TypeScript type definitions at all, so its not practical to use for this project.<br>• Our program is tested using Chrome which is the only browser with an implementation of WebXR<br>• We opted to use JavaScript instead of TypeScript because there are no TypeScript definitions for WebXR.<br>• We decided against using Rust because Three.JS has excellent support for vector math, the most computationally rigorous code we wrote. |
| 4.10 | User interfacing needs some sort of graphical representation that users will be able to view and interact with. This will be accomplished by using Three.js along with A-Frame to create the necessary models to be used as a graphical interface. A-Frame provides primitives that simplifies creating objects, such as a flat plane, that can be used in the creation of user menus or graphical displays. Three.js can also achieve this at a lower level, which will be used in adding finer details. | • Since A-Frame simplifies the task of using WebXR API, we decided to make our user interface manually.<br>• This is so we can stick to the primary goal of demonstrating how to use WebXR. |

| | | |
|---|---|---|
| 4.11 | Many of the graphical user interfaces will need some sort of functionality, otherwise they will just be objects that do nothing. The main user interface functionality is menu functionality. The user must be able to interact with a menu to select an experiment station, pause and play objects or the environment, spawn or delete objects, and others. Nearly everything the user can interact with requires some sort of interface. While the direct functionality may not be defined with A-Frame, A-Frame components will be used to apply the functionality to objects, in this case the interface models. | • Removed references to A-Frame (see above).<br>• Added portion about technology used to integrate User Interaction into our project. |

| | | |
|---|---|---|
| 4.12 | HMDs must be supported, therefore some sort of interfacing is needed to set up the peripherals such as the controllers and head tracking. This will be done using Gamepad API and A-Frame. A-Frame provides a very high level built in component, laser-controls, that provides tracked controls with a laser or ray cursor. Gamepad API is a much lower level API that these A-Frame components are built from. Gamepad API will be used for finer details in creating specific controls and gamepad setup.<br>Mobile devices should be supported, in particular those with Google Daydream support. The high level A-Frame component, laser-controls, includes support for Daydream controls. This means it will be possible to set up support for various control schemes at once by using the laser-controls component. Touchscreen should also be supported for devices without Daydream support. This will be achieved by using various A-Frame components with touchscreen support, such as the look-controls component that allows the camera to be moved by using a touchscreen with touch-drag support.<br>Computers should be supported if they are running in a compatible web browser. It should display the environment in their web browser with the ability to move with a keyboard and interact with the environment by using a mouse. This will be achieved by using several A-Frame components: wasd-controls, look-controls, and cursor. The component wasd-controls will provide the ability to move an object by using the wasd keys, most likely the camera, look-controls allows the user to rotate an object when the mouse is moved and cursor will hover and click functionality for the user to interact with other entities. | • Removed references to A-Frame (see above)<br>• We used the WebXR controller API instead.<br>• Removed requirements about compatibility with non-XR devices such as laptops. This is because we wanted to focus the scope of this project on WebXR. |

| | | |
|---|---|---|
| 5.2.1 | Web Workers will be efficient utilization of modern multi-core CPU's for better battery life and response times | (Removed)<br>• Performance management is no longer necessary since the APIs we are taking advantage of, ThreeJS and WebXR, handle optimization automatically. |
| 5.2.2 | WebAssembly will be faster and more efficient execution with a smaller runtime and response times. | (Removed)<br>• Performance management is no longer necessary since the APIs we are taking advantage of, ThreeJS and WebXR, handle optimization automatically. |
| 5.2.3 | Service Worker will browser cache priming, pruning, and offline service. | (Removed)<br>• Our site is static which allows it to be cached well by HTTP's default caching algorithms.<br>• Performance management is no longer necessary since the APIs we are taking advantage of, ThreeJS and WebXR, handle optimization automatically. |
| 5.4 Audio | HowlerJS Handles audio events and playback on call. | (Removed)<br>• HowlerJS is not part of our main WebXR features. |
| 5.5 Web Hosting | GitHub Pages will be used for hosting static websites and able to edit, push, and make live changes directly from a GitHub repository. | • Using Zeit Now to host website instead of GitHub.<br>• This is because it provides better support for hosting multiple versions of the site at the same time. |

| | | |
|---|---|---|
| 5.6 Blog Publishing System | Jekyll will be use for taking content and spits out static website to be ready to serve a web hosting server.Jekyll is part of GitHub Pages, which can be used to host sites from the GitHub repositories. | (Removed)<br>• No longer require for our WebXR experiences. |
| 5.8.1 TypeScript | TypeScript will be used to write all client side code for the app, except for that interacting with the WebXR API. | JavaScript will be used to write all client side code for the app.<br>• TypeScript was no longer viable to user due to WebXR not having type definitions and ThreeJS types being out of date. |
| 5.8.2 Rust | Safe systems programming with a small runtime and a WebAssembly cross-compilation target. | (Removed)<br>• We decided against using Rust because Three.JS has excellent support for vector math, the most computationally rigorous code we wrote. |
| 5.11.2 A-Frame | A-Frame will assist Three.js in creating these menus by using A-Frame specific tools, such as primitives. | (Removed)<br>• A-Frame was not utilized since we wanted the control that ThreeJS had to offer. |
| 5.12.2 A-Frame | A-Frame will be coupled with JavaScript to apply functional code to specific objects, in this case the user interface models, by using A-Frame components. | (Removed)<br>• A-Frame was not utilized since we wanted the control that ThreeJS had to offer. |
| 5.13.2 A-Frame | A-Frame provides built in components that aid in setting up controls for a wide range of devices. This includes HMDs, mouse and keyboard, and mobile devices with gaze interaction. | (Removed)<br>• The WebXR API dealt with controls through the Gamepad API. |

| | | |
|---|---|---|
| 6.3 Audio | There were not a lot of options for Audio APIs, still, HowlerJS stood out among its peers. Its abstraction of the WebAudioAPI and support for event driven spatial audio was much desired for this simulation. This API makes the use of theWebAudio API much easier and in the event that it is unsupported on certain browsers, it falls back to HTML | (Removed)<br>• We are no longer using audio in this project as we wanted to focus more on implementing the WebXR features. |
| 6.4 Web Hosting | The web hosting services we choose to use GitHub Pages because it will offer us static web pages for GitHub users. Static web page is a web pages that can deliver the users exactly what is stored. GitHub Pages has intergrated with Jekyll to automatically update GitHub Pages servers and regenerate the site. To host a website from GitHub Pages we will need to create a repository with a repository name plus github.io. Then enter the project folder and add/push an index.html file in order to host the website easily. | • Change GitHub to Zeit Now for static web hosting<br>• This is because it supports multiple versions of the website being deployed at the same time. |
| 6.6 Blog Publishing System | Since Jekyll is intergrated with GitHub Pages and repository of GitHub. We chose to use Jekyll as a very simple, blog-aware, and static site generator for our project. While not have to worry with needless complexity and configuration, it allows us to concentrate on the content instead. | (Removed)<br>• We are no longer using any Blog publishing system as GitHub handles our project fine and focus more on implemeting the WebXR features. |
| 6.10 User Interface Visualization | • Had references to A-Frame. | (Removed)<br>• We decided against using A-Frame as ThreeJS gave us more control. |
| 6.11 User Interface Functionality | • Removed any references to A-Frame | (Removed)<br>• Had references to A-Frame. |

| | | |
|---|---|---|
| 6.12 Design Rational - Physics | • We had chosen to use Cannon in part because it was integrated with A-Frame | • Removed reference to A-Frame |
| 6.13 Design Rational - Graphics | (Three.js) has VR support and is built into the A-Frame library. | • Removed reference to A-Frame |
| 6.14 Design Rational - Asset Bundler | | • Added sentence about how Parcel's bundling allows us to use multiple module formats. |
| 6.15 Design Rational - Hardware Interfacing | Had a sentence about using A-Frame | • Replaced that sentence with one about the hardware API functionality which was finalized in the spec. |

## 1 INTRODUCTION

This software design document is intended for the WebXR Senior Capstone Project for Oregon State University. The stakeholder of this project is Alexis Menard of Intel's open source software division. The goal of the project is to provide an example project for other developers interested in using the up and coming WebXR API built for creating VR and AR experiences on the web. Another goal of the project is to showcase an application that can be impossible to perform without a VR and AR device while progressively enhancing the experience to users. The website will serve an additional purpose as an immersive educational physics application.

We will be building a virtual reality physics simulation for use within classrooms to demonstrate how to use the API while creating something valuable for physics students at the same time. The project can be divided into individual components for the purpose of project planning, including: project setup, site hosting, physics, graphics, audio, and user interface. This document will list project requirements, then address these using design viewpoints, views, and elements, and rationales.

# 2 Design Stakeholders and their Concerns

## 2.1 Physics

### *2.1.1*

The physical constants for a scene or object can be changed.

### *2.1.2*

Objects in the scene must be interactive through translation, rotation and collision.

## 2.2 Graphics

### *2.2.1*

3D content is rendered onto the connected devices.

### *2.2.2*

Content is tailored to mobile devices when accessed through one.

## 2.3 Performance

### *2.3.1*

Simulation is rendered in real time.

### *2.3.2*

Rendering the content should fit within frame budget in VR, typically within 11 milliseconds.

### *2.3.3*

Frame rate is 60-90fps on connected HMDs.

## 2.4 Client Reach

### *2.4.1 Mobile Networks and Offline Use*

Content must be accessible from mobile speed networks.

### *2.4.2 Inexpensive and Available Hardware*

Content must render comfortably on cell phones manufactured later than two or three years ago.

Content must be interact-able without an external controller.

### 2.5 User Interface Visualization

*2.5.1*

Users must be able to visualize velocity vectors on moving objects in the kinematics scene.

*2.5.2*

Users can interact with the scene to traverse through it and into other scenes.

### 2.6 User Interface Functionality

*2.6.1*

Users can bring objects into the scene through interfaces in the simulation.

### 2.7 Hardware Interfacing

*2.7.1*

Users will be able to interface using mobile devices.

*2.7.2*

Users will be able to interface using HMDs whether they are opaque, transparent or utilize video pass-through.

*2.7.3*

Users must be able to interact with objects in the scene using position tracked controllers.

*2.7.4*

Users will be able to select objects in the scene using a mobile touchscreen interface.

*2.7.5*

Simulation pauses when a device has been disconnected and waits until reconnection to resume.

*2.7.6*

Website will notify user if the web browser or hardware is not compatible with WebXR.

### 2.8 Web Hosting

*2.8.1*

The project will be published on a website using a Now deployment.

## 3 Design Viewpoints

### 3.1 Physics

**Concerns:** 2.1.1, 2.1.2, 2.2.2, 2.3.2, 2.6.1, 2.6.2
**Elements:** Cannon.js
**Analytical Methods:** The simulation must be animated in a realistic way. Objects must interact with each other and be interactive with the user.
**Viewpoint Source:** Jonathan Jones

### 3.2 Graphics

**Concerns:** 2.2.1, 2.2.2, 2.4.2
**Elements:** Three.js
**Analytical Methods:** 3D content needs to be rendered onto the website and tunneled to the respective devices accessing the website. VR must supported.
**Viewpoint Source:** Jonathan Jones

### 3.3 Installability and Caching

**Concerns:** 2.2.2, 2.4.1, 2.4.2
**Elements:** Loading Script
**Analytical Methods:** Many 3D and 2D assets will be used to style this site from a variety of third party sources. In the interest of providing a smooth user experience, there needs to be some overhead when loading assets into a scene. For this task, we'll employ a script of our own to asynchronously load into our scenes the assets that we need. During the time in which the scene assets are loading, a loading screen will be displayed to keep the user experience clean.
**Viewpoint Source:** Evan Brass

### 3.4 Web Hosting

**Concerns:** 2.8.1
**Elements:** Now-ZEIT
**Analytical Methods:** The site will be hosted on the NOW hosting service provided by ZEIT.CO for testing a deployment.
**Viewpoint Source:** Brandon Mei

### 3.5 Version Control System

**Concerns:** 2.9.1

**Elements:** GitHub

**Analytical Methods:** The service must host models assets, including audio, texture, models, and video files.

**Viewpoint Source:** Brandon Mei

### 3.6 cloud platform System

**Concerns:** 2.8.1

**Elements:** Now-ZEIT

**Analytical Methods:** The site will be hosted on the NOW hosting service provided by ZEIT.CO for testing a deployment.

**Viewpoint Source:** Brandon Mei

### 3.7 Language

**Concerns:** Project Setup

**Elements:** JavaScript

**Analytical Methods:** The product must compile to browser-executable, bug free JavaScript code.

**Viewpoint Source:** Brooks Mikkelsen

### 3.8 Asset Bundler

**Concerns:** Project Setup

**Elements:** Parcel

**Analytical Methods:** The product must download all required assets on page load or before they are required to be displayed.

**Viewpoint Source:** Brooks Mikkelsen

### 3.9 Continuous Integration

**Concerns:** Project Setup

**Elements:** Circle CI

**Analytical Methods:** The product be written with correct and maintainable code.

**Viewpoint Source:** Brooks Mikkelsen

### 3.10 User Interface Visualization

**Concerns:** 2.5.1, 2.5.2

**Elements:** Three.js

**Analytical Methods:** Various user interfaces, such as a menu, are required for users to interact with the simulation.

These user interfaces need some sort of graphical visualization.

**Viewpoint Source:** Tim Forsyth

### 3.11 User Interface Functionality

**Concerns:** 2.6.1, 2.6.2, 2.6.3

**Elements:** JavaScript

**Analytical Methods:** User interfaces need some sort of functionality for users to interact with.

**Viewpoint Source:** Tim Forsyth

### 3.12 Hardware Interfacing

#### *3.12.1 Head Mounted Display*

**Concerns:** 2.7.2, 2.7.3

**Elements:** Gamepad API

**Analytical Methods:** The product should be compatible with VR HMDs and controllers for users to interact with the environment with 6 DoF.

**Viewpoint Source:** Tim Forsyth

#### *3.12.2 Mobile Device*

**Concerns:** 2.7.1, 2.7.4

**Elements:** Gamepad API, Reticulum

**Analytical Methods:** The product should be compatible with most mobile devices, including support for those with Daydream compatibility to offer a wide variety of experiences.

**Viewpoint Source:** Tim Forsyth

#### *3.12.3 Personal Computer*

**Concerns:** 2.7.5, 2.7.6

**Elements:** WebXR

**Analytical Methods:** The product can be run on a computer browser with the ability to switch to a full VR experience if the appropriate hardware is detected.

**Viewpoint Source:** Tim Forsyth

## 4 Design Views

### 4.1 Physics

A physics simulation needs a physics engine to function properly. Cannon.js is a lightweight 3D web physics engine written entirely in Javascript. It provides rigid body dynamics, discrete collision detection, friction, restitution, object constraints, etc.

### 4.2 Graphics

WebGL will be used to render graphics to the connected devices and the website. The website will use a JavaScript library to implement WebGL features. Three.js is a lightweight cross-browser JavaScript library/API used to create and display animated 3D computer graphics on a Web browser. Three.js scripts may be used in conjunction with the HTML5 canvas element, SVG or WebGL. The library provides Canvas 2D, SVG, CSS3D and WebGL renderers.

### 4.3 Web Hosting

Zeit Now will be used to host the website as we develop it so that our progress can be shared with the client and other team members. It allows us to set up continuous deployment for static websites easily. Zeit now also allows us to create a unique and lasting deployment for each commit that is pushed to GitHub. This will allow us to share individual progress with stakeholders in the project.

### 4.4 Version Control Systems

The hosting service must have a repository to sort all the files that can be accessed with a unique URL. We will be using GitHub to manage repositories and it also will be used to host open-source projects. GitHub provides access control and features such as bug tracking, feature request, and task management.

### 4.5 Cloud Platform System

ZEIT Now is a cloud platform for serverless deployment. It enables us to host websites and web services that deploy instantly, scale automatically, and require no supervision, all with minimal configuration.

### 4.6 Language

The application must run in modern browsers that have implemented the WebXR API. That means that the code must compile down to JavaScript (or be written in JavaScript). We will be using a modern ES2018 version of JavaScript to write code for the client side application. It can be compiled down to an older version of JavaScript for use in both Node.js and in browsers.

### 4.7 Asset Bundler

The product must download all required assets on page load or before they are required to be displayed. We will be using Parcel to bundle all of our assets, including code, images, 3D object files, and textures. Parcel takes a single HTML file as an entrypoint, and then resolves all dependencies in a tree format without needing any configuration. It also supports code splitting using dynamic imports with no configuration necessary. If we need more configuration in our asset bundler, we can use Webpack, as provides all this functionality, but requires more configuration.

### 4.8 Continuous Integration

The product must be written in a clean, easily maintainable fashion. To ensure this, we will use Circle CI to build and test our application. Circle CI is a continuous integration service that can hook into GitHub and can be configured to build and test our application after each push to GitHub and before each pull request is merged to ensure the quality of our code.

### 4.9 User Interface Visualization

User interfacing needs some sort of graphical representation that users will be able to view and interact with. This will be accomplished by using Three.js to create the necessary models to be used as a graphical interface. Three.js provides primitives that simplifies creating objects, such as a flat plane, that can be used in the creation of user menus or graphical displays. Three.js can also be used to display text in virtual reality, which will be helpful for buttons and help text.

### 4.10 User Interface Functionality

Many of the graphical user interfaces will need some sort of functionality, otherwise they will just be objects that do nothing. The main user interface functionality is menu functionality. The user must be able to interact with a menu to select an experiment station, pause and play objects or the environment, spawn or delete objects, and others. Nearly everything the user can interact with requires some sort of interface. We will use WebXR's controller API, its Magic Window feature, and Three.js to create our user interface.

### 4.11 Hardware Interfacing

#### *4.11.1 Head Mounted Display*

HMDs must be supported, therefore some sort of interfacing is needed to set up the peripherals such as the controllers and head tracking. This will be done using WebXR controller API. The WebXR controller API is a low level API that is built into WebXR. WebXR controller API will be used for finer details in creating specific controls and gamepad setup.

#### *4.11.2 Mobile Device*

Mobile devices should be supported, in particular those with Google Daydream support. This will also be done via the WebXR controller API. Touchscreen should also be supported for devices without Daydream support. This will be achieved by using various components with touchscreen support, such as a movement-controls component that allows the camera position to be moved by using a touchscreen with touch-drag support.

# 5 Design Elements

## 5.1 Physics

### *5.1.1 Cannon.js*

**Type:** Physics Engine API

**Purpose:** Physics engine written in JavaScript. Animates objects in the scene with realistic physical movements.

## 5.2 Graphics

### *5.2.1 Three.js*

**Type:** WebGL Framework

**Purpose:** Renders 3D content onto the site and connected devices for viewing.

## 5.3 Web Hosting

### *5.3.1 Zeit Now*

**Type:** Global Serverless Deployments

**Purpose:** Zeit Now will be used for hosting static websites and able to edit, push, and make live changes directly from a GitHub repository.

## 5.4 Cloud Platform System

### *5.4.1 Now-Zeit*

**Type:** Cloud Platform

**Purpose:** Global Serverless Deployments. Now makes serverless application deployment easy.

## 5.5 Version Control Systems

### *5.5.1 GitHub*

**Type:** Hosting Platform (Version Control)

**Purpose:** GitHub will manage code and track version control changes of the code.

## 5.6 Language

### *5.6.1 JavaScript*

**Type:** Language

**Purpose:** JavaScript will be used to write all client side code for the app.

## 5.7 Asset Bundler

### *5.7.1 Parcel*

**Type:** Build tool (Asset Bundler)
**Purpose:** Parcel will be used to bundle all of our assets, including code, images, 3D object files, and textures.

## 5.8 Continuous Integration

### *5.8.1 Circle CI*

**Type:** Continuous Integration
**Purpose:** Circle CI will be used to build and test our application after every push to GitHub and before each pull request is merged to maintain code quality.

## 5.9 User Interface Visualization

### *5.9.1 Three.js*

**Type:** API
**Purpose:** Three.js will be used to create graphical models that will be used for user interfaces, such as menus or graphical displays.

## 5.10 User Interface Functionality

### *5.10.1 JavaScript*

**Type:** Language
**Purpose:** JavaScript will be used to write functional code that performs requested actions from a user interacting with a menu.

## 5.11 Hardware Interfacing

### *5.11.1 Gamepad API*

**Type:** API
**Purpose:** Gamepad API provides tools to connect and map controllers to perform specific actions.

### *5.11.2 WebXR*

**Type:** API
**Purpose:** WebXR provides capabilities for detecting connected hardware which is useful in determining compatibility.

# 6 Design Rationale

## 6.1 Physics

Cannon.js was chosen as the physics engine because of its enormous feature set and VR support. Cannon.js has an active community and extensive documentation. There are lots of examples with source code that could be helpful during implementation of this project.

## 6.2 Graphics

The decision to use Three.js is mostly due to its popularity as a 3D graphics API in the web development world. It has a proven record of quality and efficiency along with a large community that supports it.

## 6.3 Web Hosting

The web hosting services we chose to use is Zeit Now because it will offer us static web pages. A static web page is a web page that can deliver the users exactly what is stored, without any server side rendering required. Zeit Now will allow us to publish a new deployment of the website for each commit that is pushed to GitHub, so that we can share that specific version of the website with other teammates and our client. Now also has easy integration with GitHub repositories.

## 6.4 Version Control Systems

We will be using GitHub for our project repository because it will allow us to have Git repository hosting service and publishes web sites. GitHub host code repositories for all types of language, not just HTML and CSS. It also has a feature to make it easy to create a multi-file website hosted at GitHub Pages. GitHub also provides access control and collaboration features like fork, pull, and merge.

## 6.5 Cloud Platform System

We will be using Now for global deployment network. It makes our team productive by removing servers and configuration, giving us a seamless developer experience to build modern scalable web application.

## 6.6 Language

We will be using JavaScript for the client side application because it is what is supported by WebXR and ThreeJS. Unfortunately, ThreeJS does not have updated TypeScript types and WebXR has no TypeScript type definitions at all, so its not practical to use for this project.

### 6.7 Asset Bundler

We chose to use Parcel to bundle our client-side assets because of how simple it is to set up. Parcel also is able to harmonize the different module formats including the old Common.js format which is the module format used for the Three.js examples. We need may need the Three.js orbit controls or perhaps it's model loaders. There is essentially no configuration - we will only need to include the root TypeScript file in our HTML page, and Parcel will figure out the dependencies based on our import statements. If we determine that we need more granularity of control in our configuration, we will switch to using Webpack, since it provides the same features, but requires (and allows) more configuration.

### 6.8 Circle CI

We chose to use Circle CI because it is simple to configure and free for open source projects. It also has built in webhooks with GitHub, so it will be simple to configure it to build and test our application after pushing changes to GitHub and before merging pull requests. Travis CI is very similar in that it is free for open source projects and easily configurable, but build tasks tend to run somewhat slower on Travis CI than Circle CI.

### 6.9 User Interface Visualization

Three.js is used since it is a well regarded API for 3D modeling. The user interface will be using 3D modeling rather than some 2D user interface toolkit, such as Open Source Qt, because it will be in a VR environment. Menus and other graphical user interfaces cannot exist on the forefront of the screen, like it would on a normal program, in a VR simulation without causing strain on the user. This is because the user's eyes will be very close to the screens due to the nature of HMDs. Instead, spatial UI is needed, which is essentially UI that exists within the environment of the simulation. Therefore, some sort of 3D graphics modeling will be needed for creating the graphical user interfaces, in the case of this project being Three.js.

### 6.10 User Interface Functionality

User interface functionality is vital to the project working correctly. Without some sort of user interface functionality, it would be impossible for the user to interact with a lot of the entities in the simulation. Many things, such as changing the properties of objects, pausing an object or the environment, changing laws of physics, and spawning or deleting items depend on some sort of user interface. JavaScript in combination with the WebXR API and Three.js is how these functionalities will be created.

### 6.11 Hardware Interfacing

Hardware interfacing is an absolute necessity; without it, the user would be unable to do anything. The choice of supporting low end devices, such as mobile devices using Google Cardboard, stems from the goal of the project: providing an affordable method of physics education. Those who aren't able to afford the expensive materials required to perform some physics experiments likely won't be able to afford a 500 dollar VR headset, not to mention a computer

that can effectively support it. Despite the WebXR specification being fairly stable, it's implementations may not be so using any library between us and the API could block our development when changes require that library to issue a patch. Additionally, we want to demonstrate the API's functionality so we choose to write our own layer ontop of the WebXR API.

# 4 Tech Reviews

## 4.1 Jonathan

CS CAPSTONE TECHNOLOGY REVIEW

DECEMBER 2, 2018

# CREATING IMMERSIVE EXPERIENCES ON THE WEB USING VR AND AR

PREPARED FOR

INTEL

ALEXIS MENARD

PREPARED BY

GROUP 47- WEBPHYSICSVR

JONATHAN JONES

**Abstract**

This document is a technical review of the components needed for the project graphical API, physics API, and Audio API. Each section explains the need for each respective component and three possible technologies that could be used in its place. Reviewed are the physics APIs: Cannon.js, Ammo.js and Oimo.js; the graphical APIs: ThreeJS, BabylonJS and Blend4Web; the audio APIs: WebAudio, Howler JS and Waud.

## CONTENTS

# 1 INTRODUCTION

Building virtual reality applications is hard and so is building a physics simulation. Developing both at the same time is even more difficult. There are lots of moving parts to a project such as this. For the visual/audio experience to materialize, there needs to be graphical, physics engine, and audio components. Graphical APIs get images on the webpage, physics APIs create realistic image movement and audio APIs immerse the user with sound. Implementing these features on their own is difficult enough. Thankfully, web developers have been hard at work creating a multitude of libraries that provide these services. Choosing the right libraries for the job, however, can be difficult. These libraries need to work together and most importantly, with WebXR. Under the assumption that WebXR developers have been building upon their past API, WebVR, the ideal libraries will be those that have built in support for WebVR. There must also be sufficient documentation associated with each library.

# 2 PHYSICS API

Physics simulations are very math intensive and require a lot of calculations behind the scenes. When VR is factored in, the process becomes much more difficult. Interactive actions like grabbing, swinging and throwing objects extends beyond simple 3D physics. There are a variety of physics engines that have been developed with WebVR in mind. Each brings its own advantages and disadvantages to the table. The physics engine used for this project needs to have VR support built in, be well documented, efficient and extensive in its features.

## 2.1 Cannon.js

Cannon.js is a lightweight 3D web physics engine written entirely in Javascript [1]. The library is open source and has been developed under an MIT license. Cannon.js primarily serves as a rigid body simulation library that is able to make objects move and interact in a realistic way [1]. It has a multitude of features that can be used to create just about any kind of physics situation. These include rigid body dynamics, discrete collision detection, friction, restitution, object constraints, and much more. There are also various shapes and collision algorithms for those shapes included in the library [1].

Cannon.js provides just about every physics function that this project requires. It also has an enormous amount of documentation. Equally as ideal is the fact that Cannon.js is used by many of the technologies that are being considered including BabylonJS and A-Frame [2].

## 2.2 Ammo.js

Ammo.js is a direct port of the bullet physics engine into Javascript. Bullet was originally written in C++ and is almost identical to ammo. It is open source and has been developed under a zlib license [3]. The Bullet physics engine has been a long time favorite for C++ developers. Bullet provides most, if not more features than Cannon.js does and even includes VR support. Bullet has been around for a long time and has a vibrant community, support and documentation. The same cannot be said for ammo which seems to rely on documentation for legacy Bullet. It also is not clear whether or not Ammo.js has VR supportive functions or not.

Included with Ammo.js is a compiler that is able to take Bullet code and convert it to Ammo.js friendly code. This means that it is possible to take the latest Bullet libraries and convert them into Javascript.

### 2.3 Oimo.js

Oimo.js, like Cannon.js, is a lightweight 3D physics engine for Javascript [4]. Oimo.js is a port of the OimoPhysics physics engine which was written in C++. It is open source and has been developed under an MIT license and like Ammo, it has poor documentation. Its features include rigid body motion, fast collision and bounding volume hierarchy, contacts with friction and restitution, multiple collision geometries, joints with springs, limits and motors, breakable joints and constraint solvers [4]. Oimo.js is a powerful yet simple 3D graphics library. The WebGL graphical framework, BabylonJS, uses Oimo.js in conjunction with Cannon.js under the hood for its integrated physics. This makes up for the lack of documentation on the Oimo.js website as it is already provided in the Babylon.js documentation.

## 3 WebGL Frameworks - Graphics

For most graphics programmers, OpenGL is the tool of choice. It works universally across most systems and is capable of just about anything that can be thought of. OpenGL though, is not embedded in web browsers, without the use of plugins. For that, there is WebGL; a cross-platform, royalty-free API used to create 3D graphics in a Web browser [5]. WebGL by itself can do amazing things. These things are hard to program though and have already been done by others on the web. VR, especially, is difficult to implement but these open source libraries have taken care of the boilerplate already. The graphical APIs explored in this document all use WebGL as their backend.

### 3.1 ThreeJS

Three.js is a lightweight cross-browser JavaScript library/API used to create and display animated 3D computer graphics on a Web browser. Three.js scripts may be used in conjunction with the HTML5 canvas element, SVG or WebGL [6]. The library provides Canvas 2D, SVG, CSS3D and WebGL renderers [7]. Three.js has been used to make hundreds of graphical simulations on the web and a large number of them include the WebVR API. Three.js is feature heavy and includes animation, shaders, effects, objects, debugging, virtual reality and more. It is compatible on all modern browsers and can even run on older browsers as it is able to fall back to one of its multiple renderers if one is not available on the current browser [8].

Three.js has an incredible amount of documentation and a vibrant community that is constantly making improvements to the API. It has the most documentation out of the three graphical APIs that are discussed here. There are hundreds of examples and demos on the Three.js website that can be used for reference when developing this project.

### 3.2 BabylonJS

BabylonJS is a graphical web framework for WebGL that was developed by Microsoft. It has a strong focus on the development of 3D games. It is about as feature heavy as Three.js and includes some features not found in the other. BabylonJS is relatively new and is much older than Three.js. It has an integrated physics engine that uses cannon.js and

oimo.js as well as an integrated spatial audio system that uses WebAudio. Essentially, this framework is able to take care of most of the features that need to be implemented by itself. BabylonJS is highly optimized and looks to perform well on most systems [9].

Like ThreeJS, Babylon has an amazing community of developers working on it who are always actively improving the framework. It is extensively documented and also boasts many demos and tutorials.

### 3.3 Blend4Web

Blend4Web is an open source, free GPLv3 licensed WebGL graphical framework that has a built in material editor with heavy support for Blender [10]. This is the most complete framework out of this list; it includes everything that Three.js and BabylonJS have and more. What is most ideal about this framework is how it was created with Blender in mind. Asset development is hard, Blender makes it easy, and Blend4Web makes implementation of Blender products even easier. Like Babylon, Blend4Web has an integrated physics engine, spatial audio systems and support for VR animation [10]. It has been described as a tool for interactive 3D visualization on the Internet [10]. There are lots of examples of Blend4Web projects and many pages of documentation on the web. Still, it remains lesser known than the previously listed frameworks which unfortunately can play into its usefulness as there is still a much better community tied to the other two.

## 4 Audio API

The goal of VR is immersion and audio is an incredibly important part of this process. For this project to be immersive, it needs event driven spatial audio and an audio API that can provide it. Spatial audio is a full-sphere surround sound technique that utilizes the three dimensions to mimic real life audio [11]. This is the only way to correctly convey sound to the user in a virtual reality simulation. Much like the other components in this review, this part of the project does not need to be created from scratch as other developers have done that job already.

### 4.1 WebAudio

The WebAudio API provides a powerful and versatile system for controlling audio on the Web that gives developers the ability to choose audio sources, add effects to audio, create audio visualizations, apply spatial effects and much more [12]. WebAudio works by using a series of audio nodes in an audio routing graph to bring organized audio framework to a webpage [12]. It is just about the only audio API that is used world wide on the internet. Part of the reason for this is the fact that it was developed by the W3C (World Wide Web Consortium). As far as VR development goes, Mozilla has said that It's really useful for WebXR and gaming. In 3D spaces, it's the only way to achieve realistic audio. [13] There is an enormous amount of documentation associated with WebAudio, mostly from the Mozilla developer site.

By itself, web audio is already easy to use and implement into web applications. Still, there are many libraries that simplify its use even further and add features themselves. The following APIs wrap around basic WebAudio and abstract it.

### 4.2 HowlerJS

HowlerJS makes working with audio and WebAudio in Javascript easy. It is a single API for all audio needs that defaults to WebAudio and falls back to HTML5 if needed [14]. It is well optimized, easy to use, modular and lightweight. Additionally, it is compatible with most modern browsers because of its fallback to HTML5 [14]. HowlerJS has stereo panning and 3D spatial audio support which is very important to this project. Finally, it has full codec support, meaning that all audio file types are compatible [14]. As a WebAudio API, it is one of the most popular in its field and has developed a large community. As a result, this API has extensive documentation.

### 4.3 Waud

Waud is essentially the same as HowlerJS. It has all the same features and support for spatial audio. Like Howler, it abstracts WebAudio, defaults to it, and falls back on HTML5 on unsupported browsers [15]. Both have zero dependencies and both are fully modular. One key difference however is the absence of volume fading in Waud. It is important to keep as many options open as possible when dealing with audio so this is a clear disadvantage to HowlerJS.

## 5 Conclusion

For this project to be successful, the right technologies must be chosen for the right components. For the graphical, physical engine and auditory components there are many options. Each with its own advantages and disadvantages. The sheer amount of Javascript libraries that area available for 3D visualizations is staggering and can make this choice unclear. For this project, however, there are clear choices for each component.

The physics engine should be Cannon.js. While Oimo.js and Ammo.js are both great physics engines, only one Cannon.js built directly into the A-Frame API. As it seems that this project will be making use of A-Frame, it only makes sense to use what works with it. This along with the extensive documentation that Cannon.js has makes this an easy choice.

The graphics API must be Three.js. This API stands out as a definitive choice among the three APIs listed. Like Cannon.js, Three.js is built into A-Frame which makes it highly desirable. Its incredibly feature richness and vibrant community is another factor in this recommendation. It is important that there is lots of support behind each component. Success is made all too easy when the entire world of web developers is at the back of the API being used by a project.

For the audio API it was another easy choice. HowlerJS is the favorite among web developers for 3D audio and spatialization. Its wide compatibility and abstraction of WebAudio make it simple and easy to use for beginners in the audio arena. Like the others, HowlerJS has the most extensive documentation among its competitors.

Each technology has its advantages and disadvantages but these technologies will only serve to improve the development experience of this project.

## REFERENCES


[1] "Cannon.js readme," https://github.com/schteppe/cannon.js/blob/master/README.markdown, accessed: 2018-11-4.

[2] "Cannon a-frame article," https://hacks.mozilla.org/2017/05/having-fun-with-physics-and-a-frame/, accessed: 2018-11-4.

[3] "Ammo.js readme," https://github.com/kripken/ammo.js/blob/master/README.markdown, accessed: 2018-11-4.

[4] "Oimo.js github," https://github.com/lo-th/Oimo.js, accessed: 2018-11-4.

[5] "Webgl wiki," https://www.khronos.org/webgl/wiki/Getting_Started, accessed: 2018-11-4.

[6] "Three.js stack overflow," https://stackoverflow.com/questions/tagged/three.js, accessed: 2018-11-4.

[7] "Three.js readme," https://github.com/mrdoob/three.js/blob/dev/README.md#threejs, accessed: 2018-11-4.

[8] "Three.js browser support," https://threejs.org/docs/#manual/en/introduction/Browser-support, accessed: 2018-11-4.

[9] "Babylonjs site," https://www.babylonjs.com/, accessed: 2018-11-4.

[10] "Blend4web readme," https://github.com/TriumphLLC/Blend4Web/blob/master/README.rst, accessed: 2018-11-4.

[11] "Spatial audio how to," https://blog.storyhunter.com/how-to-use-spatial-audio-to-create-immersive-360-videos-e0805b51c143, accessed: 2018-11-4.

[12] "Mozilla web audio introduction," https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API, accessed: 2018-11-4.

[13] "Mozilla web audio spatialization," https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Web_audio_spatialization_basics, accessed: 2018-11-4.

[14] "Howlerjs readme," https://github.com/goldfire/howler.js/blob/master/README.md, accessed: 2018-11-4.

[15] "Waud github," https://github.com/waud/waud, accessed: 2018-11-4.

## 4.2 Brandon

CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 10, 2018

# CREATING IMMERSIVE EXPERIENCES ON THE WEB USING VR AND AR

PREPARED FOR

INTEL

ALEXIS MENARD _____ _____

*Signature* *Date*

PREPARED BY

GROUP 47- WEBPHYSICSVR

JONATHAN JONES _____ _____

*Signature* *Date*

EVAN BRASS _____ _____

*Signature* *Date*

BROOKS MIKKELSEN _____ _____

*Signature* *Date*

TIM FORSYTH _____ _____

*Signature* *Date*

BRANDON MEI _____ _____

*Signature* *Date*

**Abstract**

This document is a technical review of researching different web hosting technologies options for the group 47 - WebPhysicsVR team's as a foundation. It will include the web hosting typed used for web pages to be viewed via the internet. This document outlines the Hosting and Publishing, Types of Web Hosting Technologies, and Content Delivery Network for the WebPhysicsVR. The site will then be deploy immersive web experience hosted by the Intel 01.org open source portal.

# CONTENTS

# 1 Introduction

We want virtual reality to be easier for developers can make something quickly and able to share it with everyone, without compatible issue or what device they're on. My focus of this research will be for the various web hosting technology and access immersive version of the website. Web hosting is a service that allows individuals or organization to host a website or web page onto the internet for the community to see. This project will be demoing in a host website and will later ultimately be deployed in the OpenSource portal of Intel O1.org. The main goal is to promote VR and AR technologies through the web browsers and help display the immersive web. However, there are plenty of services to deploy and host a site like AWS, GitHub, or self-hosting. With the foundation of WebXR of enhanced VR web experience, we need to pick and choose a web hosting that works best for this immersive VR web experience.

# 2 Hosting and Publishing

## 2.1 Overview

For the implementation of the virtual reality physics simulator onto the internet, some kind of web hosting will be required for a storage center that houses the information, images, video, and other content that comprises a physics simulator. By deciding different sites that serve SSL and HTTPS, due to a restriction of the browser's WebVR API. The goal is to use a web hosting technologies to host our physics simulation VR through the web, allowing them to view VR through the browser.

## 2.2 GitHub Pages

GitHub Pages is a web hosting service that is offered by the creators of GitHub. Currently, there are lots of options to build a website but these options can be excessive. GitHub provides an easier way to host static web pages for GitHub users, blogs, and project documentation. There are no required databases to set up and no servers to configure. It works by serving all of your projects sites from a personal URLs that is tied to an organization or GitHub account, and then it looks for any web content on a specific branch called GitHub Pages When you are ready to publish any changes you can merge them in GitHub [1].

This allows users or organizations to host a website by creating a repository with your GitHub pages URL as its name and adding web content to the master branch. GitHub Pages automatically builds and deploys your site for you. The websites are generated through [1]. This could beneficial for the WebXR demo project since this frees up our team to focus on the content of the project and website instead of worrying about how the team will get it from our computers to everyone else browsers.

## 2.3 Amazon Web Services

Amazon Web Services or AWS is an cloud web hosting solutions that provides static websites. Static websites can deliver HTML, JavaScript, video, images, and other files to your website hosts. It also contains no application code. The static websites are very low cost, has high-levels of reliability, and able to scale to handle high-level traffic with no additional

work. AWS will help accomplish our project that delivers data, videos, application, and most importantly API's with low latency and higher transfer speeds. This enable the website to load quickly through Amazon CloudFront. ClouldFront is integrated with AWS and allows content delivery network (CDN) to host website content in close proximity to users [2].

### 2.4 Surge

Surge is another tool to publish web sites but with a simple single command from the command line. This is a great tool if youre comfortable with the command line. This is useful for static web publishing for Front-End Developers. It allows the front end Developers to use HTML, CSS, and JavaScript to code the website inside the user's browsers instead of a back end developer that runs on the web server. This could be beneficial for deploying projects to a production-quality CDN (Content Delivery Network) through Grunt, Gulp, and npm tools to host assets externally [3]. Surge seemed to be simple because of the single-command web publishing. However, it is a little easier to publish HTML, CSS, and JS for free, without leaving the command line. The cost of using Surge is free and unlimited applications with custom domains [1].

### 2.5 Neocities

Neocities is another free web hosting service. Not only it is another free and easy way to create and publish a site from within the browser, but it also operates their own caching CDN to quickly serve our website. Neocities is very similar to Surge in that it lets users upload assets into the project directory versus a CDN (Content Delivery Network). It makes Neocities at least better at hosting models. With Neocities, it is a free and open source and we can create and edit files. There are paid offers that offer more services for $5 per month. [1].

## 3 Content Delivery Network

### 3.1 Overview

Hosting models is not an easy task. The Models usually come as groups of files in a folder, where the model file relatively references other files such as images. Thus, models have to be uploaded as a single folder in the same directory. Content delivery network purpose is to serve to end-users with high availability and speed. This allows content delivery could help drastically reduce server lag by storing static resources on a network of fast-loading servers.

### 3.2 GitHub

GitHub's workflow allows hosting models assets easily and URL hosted on GitHub. In GitHub, you can drag files to add them to your repository or choose specific files and then commit to master branch or create a new branch. The branching exist to help manage the workflow and allows you to try new ideas without affecting the master branch, so we can experiment and commit the changes [1].

### 3.3 A-Frame

A-Frame has become the largest and welcoming virtual reality communities. We don't have to worry much about hosting assets when A-Frame can use relative URLs to reference the assets as they are on the same domain. The A-Frame assets uploader uses UploadCare to host via CDN. This allows A-Frame site to fetch the assets easily to display in the scene[1].

### 3.4 Glitch

Glitch is the easiest and fastest way to create a site within the browser. Glitch allows us to add code, files, upload assets, edit other peoples code, define our own URL name, and immediately deploy any changes. The Glitch editor has a panel to upload assets and provides content delivery network URLs in return. To fetch an asset to display you need to use **¡a-assets¿** and served with cross-origin resource sharing (CORS) to have any permission to access the selected resource from a server at a different origin [1].

## 4 Types of Web Hosting Technologies

### 4.1 Overview

As our group finish researching, designing, and developing, we need host our demo project for the community to see. Because web hosting will help us deploy, host, and publish physics simulator site and its assets onto the web for the world to see.

### 4.2 Shared Web Hosting

Shared Web Hosting is a way to allow multiple websites to utilize a single server and maintained by the hosting service. Each user on a shared server will get a share of the server's total available bandwidth, memory, and power[4]. This allows shared hosting setup easy to use and maintain by the hosting provider. WebPhysicsVR will now have to be responsible for setting up and running their own site, and a single account to set up multiple sites. Shared web hosting is the cheapest and easy to create and maintain. However, there can be advantages and disadvantages of Shared Web Hosting. The advantages it is easy to manage the website as it has a built-in control panel to manage content. The disadvantage is the Shared Resources have limits of the server total resources, including bandwidth, memory, and CPU power [4].

### 4.3 Dedicated Web Hosting

Dedicated Web Hosting is a type of web hosting technology that leases an entire server and is not shared with anyone else. This would be more flexible than shared web hosting because WebPhysicsVR will have full control over the servers, including a type of operating system, hardware, etc. However, there can be advantages and disadvantages of Dedicated Web Hosting. The advantages of dedicated web hosting are that no one can steal others resources It also have an amount and type of memory and hardware to choose for our project. This disadvantage is the cost of maintaining the dedicated web hosting, managing malware, and any additional cost. [4].

### 4.4 Cloud Web Hosting

Cloud hosting is very similar to VPS web hosting as it also has a website on a virtual machine. However, instead of one physical server, it can be many networks of computers which can acquire more power dynamically as it needs and relies on sharing of resources. Cloud web hosting allows WebPhysicsVR to accommodate all aspect of hosting by load balance and hardware resources are available virtually when needed. This could be beneficial for the WebXR project as scalability is a way to grow the virtual and augmented reality. However, there can be advantages and disadvantages of Cloud Web Hosting. The advantages of cloud web hosting are it allows multiple computers to combine into a powerful virtual server that accommodates its resources on a need basis. This allows our project dynamically expand on the amount of power for physics simulation. The disadvantage is that cost vary and may experience traffic spikes to cover the cost [4].

### 4.5 VPS Web Hosting

VPS stands for Virtual Private Server and it is more stable and reliable than shared web hosting. It is very balanced and stable than the other types of web hosting. A VPS server is considered a shared environment, but the way it is set up is very distinct. VPS share one physical server and it can host multiple virtual machines. This would allow WebPhysicsVR to be in a shared web hosting while getting our own dedicated web server. However, there can be advantages and disadvantages of VPS Web Hosting. The advantages are that VPS is more stable as it has a limit on the website per server, provide more flexibility and customize without affecting others. The disadvantages of VPS web hosting is the cost and maintaining the hosting [4].

### 4.6 Conclusion

There are many web hosting services that can be used for our project's hosting and publishing, content delivery network, web hosting technologies. For WebPhysicsVR to be showcase shared with the web community, it needs a type of internet hosting service or technology that would be a benefit for our project and hosting immersive Virtual and augmented reality through the web. The objective is to demo on a web hosting website and then deploy it to Intel Open Source (01.org). In conclusion, there are several hosting services that will help our project but GitHub Pages stands out with its edit, push, and your changes are live through GitHub repository. GitHub allows our project to be easily hosted directly from the GitHub repository and to be demoed.

## REFERENCES

[1] A-Frame, “Hosting and publishing.” https://aframe.io/docs/0.8.0/introduction/hosting-and-publishing.html, 2018.

[2] aws, “Host a static website.” https://aws.amazon.com/getting-started/projects/host-static-website/, 2018.

[3] C. Chapman, “Exactly what you need to know to be a front end developer.” https://skillcrush.com/2016/02/11/skills-to-become-a-front-end-developer/, 2018.

[4] J. Stevens, “Different types of web hosting.” https://hostingfacts.com/different-types-of-web-hosting/, 2018.

### 4.3 Tim

Oregon State University

CS 461: Senior Software Engineering Project I

**Group 47**

**Immersive Web VR/AR:**

# Tech Review

Tim Forsyth

**Abstract**

This document is a review of the potential technologies that could be used in the development of the project for Capstone Group 47, WebPhysicsVR. Inside is a review of potential frameworks, tools, libraries and APIs that could be used in generating the user interfaces, as well as implementing functionality to said interfaces and user controller support. A few of the technologies that will be reviewed are A-Frame, ReactVR, Three.js, Gamepad API, Reticulum, and JavaScript.

Fall 2018

# 1 Introduction

WebPhysicsVR, developed by Group 47, will be an interactive physics simulation which utilizes the WebXR API at its core. The goal of this project is to provide a tech demo of how WebXR can be used to develop entertaining and interactive VR experiences within a web browser and act as an example for future developers to reference in their own endeavors. Tim Forsyth will be responsible for the creation and functionality of the various interfacing within WebPhysicsVR. This can be broken up into three pieces: user interface visualization, user interface functionality, and controller interfacing. This document will go over the various technologies available that may be useful for each individual feature.

# 2 UI Visualization

The user interface is an essential aspect of most interactive programs, WebPhysicsVR is no different. Several different menus will likely be needed, such as a menu for picking the room or selecting objects from a tool box. Objects themselves will also need some sort of interface that displays various statistics about the object's motion, such as velocity, acceleration and height.

In most cases, the UI is on the forefront of the screen, in direct view of the user. However, this doesn't work in VR. Placing the UI too close to the user can lower the clarity and make it nearly impossible to read as well as strain the user. The solution is to use spatial UI. Spatial UI is a form of UI that exists within the virtual space. This essentially means that the UI will not be on the forefront of the screen, instead it will live within the VR environment. Doing this makes it much easier for the user to read and overall interact with the interface. The first step in achieving this is by being able to generate the UI model. There are several frameworks available that can aid in this process; this review will look at three potential tools: A-Frame, ReactVR, and Three.js.

## 2.1 A-Frame: Primitives

A-Frame is a high level web framework originally developed by Mozilla that has turned into an open source project dedicated to helping developers create virtual reality experiences. Being based on top of HTML, it makes it a framework that is easy to jump straight into. A-Frame provides several primitives that make modeling in a VR environment easy. Primitives are a convenience layer on top of the core A-Frame API that provides a more familiar interface with HTML attributes maping to a single value.[1] Of the several primitives available, the most useful would likely be `<a-box>`, `<a-plane>` and `<a-text>`. These respectively create box like shapes, create flat surfaces, and wrap text onto an object.[2, 3, 4]

## 2.2 ReactVR

Similar to A-Frame, ReactVR is a high level framework that has been created for the use of developing user interfaces in a VR environment. ReactVR is essentially React, a JavaScript library for building user interfaces, except it operates within a VR environment. This means it uses the same syntax as React, making it second nature to use for experienced React developers. The caveat is that ReactVR is still young and is not as fully fleshed out as a tool such as A-Frame. With that being said, it still does provide useful tools for creating UI, such as OVRUI which is a library that contains several geometries used in building UIs in VR.[5]

## 2.3 Three.js

Both A-Frame and ReactVR share at least one thing in common: the underlying library they work on top of, Three.js. Three.js is a project that aims at creating a lightweight, easy to use, 3D library. It sits on top of WebGL and simplifies the rather complex API.[6] Even though there exist other tools such as A-Frame and ReactVR that simplify it even further, it can still be useful to directly use Three.js. Using a high level framework makes it easier to work with, but it can also be more restrictive. Three.js has several geometry classes used for creating 3D models; in the case of UI, PlaneGeometry would the most useful for generating a flat surface to work with and build upon.[7]

# 3 UI Functionality

A menu needs more than just a user interface model, it needs to actually have some sort of interactive functionality. There are several ways to be able to detect and handle user inputs, such as button presses or clicks, and perform the requested actions. The few tools that will be discussed in this review are JavaScript, Reticulum, and A-Frame.

## 3.1 JavaScript

Many tools that have already been discussed use JavaScript underneath their overarching framework. In fact, most modern web browsers and APIs use JavaScript, including the WebXR API that will be used. Therefore, JavaScript will be used in some manner, whether it is through some sort of framework or not. With that being said, there is reason to use JavaScript directly for event handling. Being such a popular language, most people know JavaScript and have experience with it to some extent. Luckily, event handling with JavaScript event listeners in WebXR is exactly the same as with any other web API. This means previous knowledge and experience of JavaScript will be able to be used in helping process user clicks and perform actions.

## 3.2 Reticulum

Reticulum is a simple gaze interaction manager that is powered by Three.js. It boasts gaze and click events for targeted objects, allowing us to have the option of implementing a gaze based UI in which users could simply look at what they want to click instead of aiming with a controller. It also has built in fuse support, which adds a timer or a countdown that lets the user know a click event is going to occur from staring at the object. This helps WebPhysicsVR become easier to use for those who are using VR head mounted displays that do not support controllers, such as Google Cardboard. Reticulum is also compatible with all of the other technologies that have been looked at so far.[8]

## 3.3 A-Frame: Components

A-Frame has the concept of components, which are reusable chunks of data that can be simply plugged into an entity or object to add functionality by way of JavaScript. While A-Frame itself my not be implementing the event listening, per say, it can make it a lot easier to apply JavaScript event listeners functionality to entities within the VR environment. This means A-Frame components could be used to add functionality to a UI menu.[9]

Custom components are able to be created by registering the component you wish to create using `AFRAME.registerComponent`. The first parameter of `registerComponent` is the name of the component and the second is a schema object that defines the functionality of the component. Components also have several unique properties that help make the JavaScript programming easy. These properties include `schema`, `data`, and `el`. The property `schema` contains the component names, types and overall functionality. Next, the `data` property is the parsed data that is derived from the defined schema of the component. Finally, the `el` property is an extremely useful tool as it is what references an entity element. This makes it extremely easy to apply components to essentially any entity.[10]

# 4 Controller Interfacing

User interaction is a large portion of WebPhysicsVR. This means the users will need to have some sort of control scheme, whether it be a mouse and keyboard, HMD and controllers, or even a touchscreen in the case of mobile users. User inputs, such as button presses, clicks, or taps, need to be parsed and the appropriate actions need to be performed in return. The three technologies that will be reviewed in this section are the Gamepad API, the legacy Gamepad Extensions API, and once again, A-Frame.

### 4.1 Gamepad API

The Gamepad API is an API dedicated to helping developers access and respond to signals and inputs from gamepads in a simple way. It is able to respond to gamepads being connected and disconnected as well as access other information about the gamepads, such as what buttons are actively being held down. Gamepad API was developed by Mozilla in an attempt to simplify controller interfacing. This API makes implementing controller support trivial.

There are several interfaces provided by the API: `Gamepad`, `GamepadButton`, `GamepadEvent`, `GamepadPose`, and `GamepadHapticActuator`. Most of these are rather self explanatory: `Gamepad` represent the actual controller that is connected, `GamepadButton` represents a single button on a controller, `GamepadEvent` is an object that represents events related to gamepads, `GamepadPose` represents the orientation of a controller in a 3D space, and `GamepadHapticActuator` represents the hardware which provides haptic feedback, such as vibrations. All of these interfaces are important and have their uses, but for a VR project like WebPhysicsVR, `GamepadPose` would most definitely be useful.[11]

### 4.2 Legacy Gamepad Extensions API

WebXR also supports the legacy Gamepad Extensions API, which expands upon the Gamepad API to supply support for more complex devices. Controllers paired with virtual reality headsets are generally more complex and rely on advanced features, such as motion tracking. Thus, using the Gamepad Extensions would make sense as it would ease the ability to interface with VR controllers correctly. Similarly to Gamepad API, it also has a `GamepadPose` interface, however it is expanded on by being able to define the gamepad's linear and angular velocity and acceleration in addition to its position. It is clear that this could be extremely useful in a physics simulator like WebPhysicsVR.[12]

### 4.3 A-Frame

Once again, A-Frame as a prime option. A-Frame components can be taken advantage of to implement controller support as well as support for mouse and keyboard and even mobile touchscreens. In particular, the look-controls, wasd-controls, laser-controls, and cursor components are some of the most critical components that would aid in implementing user controls. For starters, the look-controls component implements the ability to control the rotation of the camera with all three control options: VR HMD, mouse, and touchscreen.[13] The cursor component provides interaction for hovering and clicking the mouse and the wasd-controls component allows control of an entity, generally the camera, with the WASD or arrow keys, making mouse and keyboard support possible.[14, 15] Finally, the laser-controls component implements tracked controls that shoot a laser from a VR HMD paired controller that can be used for interactions. Laser-based interactions scale the best across different pieces of hardware as it can range across several different degrees of freedom (DoF). Common Degrees of Freedom include 0 DoF (Google Cardboard), 3 DoF (GearVR with controller wands), and 6 DoF (HTC Vive).[16]

## 5 Conclusion

WebPhysicsVR will be a physics simulation that relies heavily on user interaction. Thus, it is important to include an interactive user interface that makes it an easy and enjoyable experience for the user. The first step in doing so is generating the interface. Several technologies are available for doing so, such as A-Frame primitives, ReactVR and Three.js. Next is to supply functionality to the UI, which can be done using JavaScript, Reticulum or A-Frame components. Finally, controller support is able to be implemented using the Gamepad API, the Gamepad Extensions API, or A-Frame. An obvious and common theme here is A-Frame. A-Frame has the capabilities of providing essentially everything needed in terms of creating and implementing an interactive user interface due to the fact that it operates on a high level. It operates on top of some of the other technologies referenced, such as JavaScript and Three.js. In conclusion, A-Frame is the best choice of technology going forward in the development of the user interfacing.

# References


[1] "Introduction: Getting started," https://aframe.io/docs/0.8.0/introduction/, 2018, accessed: 11/2/2018.

[2] "Primitives: `<a-box>`," https://aframe.io/docs/0.8.0/primitives/a-box.html, 2018, accessed: 11/2/2018.

[3] "Primitives: `<a-plane>`," https://aframe.io/docs/0.8.0/primitives/a-plane.html, 2018, accessed: 11/2/2018.

[4] "Primitives: `<a-text>`," https://aframe.io/docs/0.8.0/primitives/a-text.html, 2018, accessed: 11/2/2018.

[5] M. Argmstrong and A. Imm, "Introducing the react vr pre-release," https://developer.oculus.com/blog/introducing-the-react-vr-pre-release/, 2016, accessed: 11/2/2018.

[6] mrdoob, "three.js," https://github.com/mrdoob/three.js/, 2018, accessed: 11/2/2018.

[7] "three.js: Planegeometry," https://threejs.org/docs/index.html\#api/en/geometries/PlaneGeometry, 2018, accessed: 11/2/2018.

[8] skezo, "Reticulum," https://github.com/skezo/Reticulum, 2017, accessed: 11/2/2018.

[9] "Component," https://aframe.io/docs/0.2.0/core/component.html, 2018, accessed: 11/2/2018.

[10] "Component: Under the hood," https://aframe.io/docs/0.2.0/core/component.html\#under-the-hood, 2018, accessed: 11/2/2018.

[11] "Gamepad api," https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API, 2018, accessed: 11/2/2018.

[12] "Gamepad extensions," https://w3c.github.io/gamepad/extensions.html, 2018, accessed: 11/2/2018.

[13] "Components: look-controls," https://aframe.io/docs/0.8.0/components/look-controls.html, 2018, accessed: 11/2/2018.

[14] "Components: cursor," https://aframe.io/docs/0.2.0/components/cursor.html, 2018, accessed: 11/2/2018.

[15] "Components: wasd-controls," https://aframe.io/docs/0.2.0/components/wasd-controls.html, 2018, accessed: 11/2/2018.

[16] "Components: laser-controls," https://aframe.io/docs/0.8.0/components/laser-controls.html, 2018, accessed: 11/2/2018.

## 4.4 Brooks

CS CAPSTONE TECHNOLOGY REVIEW

JUNE 7, 2019

# CREATING IMMERSIVE EXPERIENCES ON THE WEB USING VR AND AR

PREPARED FOR

INTEL

ALEXIS MENARD

PREPARED BY

GROUP 47

BROOKS MIKKELSEN

**Abstract**

This document is a Technology Review paper that focuses on the project setup for Capstone group 47, Creating Immersive Experiences on the Web using VR and AR. This document reviews potential languages (and subsets of languages) used for development, including plain JavaScript, TypeScript, and Flow. It also reviews potential Asset Bundlers, including Webpack, Parcel, and Rollup. In addition to this, it reviews three options for continuous integration - Travis CI, Circle CI, and Jenkins.

# CONTENTS

# 1 INTRODUTION

The goal of our project is to provide an example for future developers interested in interfacing with the WebXR API. This API gives developers access to end users' Virtual Reality and Augmented Reality systems, so that they can make interactive VR and AR experiences for the web.

One of my focuses in the project will be project setup. This includes choosing the best language suited to our style of development, the build tools (in our case an asset bundler and development environment), and the continuous integration tool we will use to build, test, and deploy our application.

# 2 LANGUAGE

## 2.1 Plain ES2018 JavaScript

One of the requirements for our project is to have the Virtual Reality experience hosted within modern web browsers. Since all modern web browsers use JavaScript, and the WebXR API is implemented in JavaScript, we will need to use JavaScript or some derivation of it to create our project.

Plain JavaScript does not have any compile-time type checking. The only type checking it does is if there is an error. As with other dynamically typed language, plain JavaScript allows developers to rapidly create projects and get products out the door quickly. However, although it is easy to quickly create projects in plain JavaScript, it is much more challenging to write code that is maintainable over the long run. This is because without static types, much more documentation must be written for more than one person to write code in the same code base, since it all must interact with itself.

Every year, the ECMA standards organization makes updates to the JavaScript standard that is implemented in each browser. However, it takes time for browsers to implement these standards, and for users to actually update their browser. Therefore, if developers want to use the latest standard, they will need to use a preprocessor such as Babel to create polyfills for them so that user on older browsers can still use all the functionalities of the website [1].

## 2.2 TypeScript

TypeScript was created by Microsoft as a way to provide static typing for JavaScript. It does this by forcing users to add type definitions, either explicitly or implicitly (if possible in the situation) to all of their variables, functions, classes, and objects. It also adds interfaces, which are used to tell the compiler (and the developer) about the shape of JavaScript objects and classes, including whether properties are optional or mandatory, and what the type of each of those properties can be [2].

TypeScript does add some overhead to the developer. First, the code must be compiled into regular JavaScript to be run in the browser. Also, each library that is referenced within a TypeScript file must have a type definitions file. This means that for any smaller libraries that don't have type definitions, developers won't be able to get the benefits of typescript when interacting with that library [2].

### 2.3 Flow

Flow is similar to TypeScript in that both tools are used to provide static types to JavaScript. While TypeScript was made and is supported by Microsoft, Flow is made by Facebook. The type annotation system that Flow uses is very similar to TypeScript, but there are a few differences between the tools as well. Flow will build a deeper understanding of the code and does interprocedural analysis on the code. TypeScript, on the other hand, helps developers by providing intellisense (auto complete), code navigation, and refactoring [2].

## 3 Asset Bundlers

### 3.1 Webpack

Webpack has been the most used bundler for the past few years, which means that there is widespread support for it, as well as plenty of tutorials on configuring it. In the past, it has had a bad reputation for being complicated to configure, but in more recent versions, it is significantly easier to have work right out of the box. However, there is configuration required for features such as development server and code splitting. One thing Webpack excels at is allowing the developer full control over their build process, including specifying specific tools for specific filetypes (JavaScript, CSS, Images, etc.) [3].

### 3.2 Parcel

Parcel is a newer asset bundler that requires almost no configuration. It works by having the user directly include the entry file (that still needs to be compiled) in their HTML file, and then it resolves the dependency tree from there and outputs a modified HTML file, along with other dependencies. This could beneficial for the WebXR demo project, since the team wouldn't have to worry about spending the time to configure Webpack. However, it doesn't have as widespread of support since it is a newer package. It does have out of the box support for TypeScript and Rust, along with many other languages [3].

### 3.3 Rollup

Rollup is a recent addition to the Asset Bundler scene. It requires some configuration, and has works very similar to Webpack in that users specify an entry point, a destination, and an array of plugins that will be applied to the assets. However, Rollup's configuration is a little bit easier to use, but has less documentation than Webpack, especially when it comes to third party tutorial articles. Rollup also seemed to be much quicker than Webpack and Parcel, from tests done by X-Team in their article titled Rollup Webpack Parcel Comparison [3].

## 4 Continuous Integration

### 4.1 Circle CI

Circle CI (Continuous Integration) is a tool that automatically runs unit tests on developers' code whenever a commit is pushed to a branch on a source control server. In our case, we will be using GitHub. It is compatible with Node.js

(which will help for running unit tests of client-side code), along with many other languages. It works by listening for notifications from GitHub (or other source control providers), then pulling the branch into their cloud cluster and preforming a set of user-specified tasks. Typically, these tasks include building, testing, and publishing. These tasks are configured within a YAML config file [4]. Circle CI is free for open source and closed source projects, but one can pay to upgrade the resources the build is given.

Using CI such as Circle CI allows the developer to ensure code quality and that the tests are all passed. Developers can also create releases from within continuous integration, which means that the developer doesn't have to do it themselves manually [4].

### 4.2 Travis CI

Travis CI is very similar to Circle CI in that users can specify a YAML config with the tasks they want to execute on their code (typically building, testing, and publishing), and then have Travis run them after every source control push or merge. Although Travis CI has a different config schema, they both allow users to do similar tasks the same way. Both also run in the cloud, rather than on one's local computer or build server so there is little setup required. There are a few things that Travis supports that Circle CI does not. Travis CI has build matrixes, which means that one can specify different environments (such as language versions and operating systems) to build and test their code on [4]. Travis is free for open source projects.

### 4.3 Jenkins

Jenkins is one of the most well-known continuous integration systems in existance, partly because it has been around for a long time. Unlike Travis CI and Circle CI, it is a program that developers download to a build machine. This means that to use Jenkins, we would have to have our own build machine to run it on, then configure it to hook into GitHub. This could be a benefit for closed source projects that do not want to risk their source getting out into the world, but for open source projects, this is just another cost. Jenkins is also known to be challenging and time consuming to configure, so there is a time cost associated with using it as well. Other than those costs, it is very configurable and has a plugin system built into the tool. The software itself is also free [4].

## 5 Conclusion

One of the requirements of our project is for it to be able to run in the browser. Hence, the language we will need to use must compile to JavaScript. Three options for this are plain JavaScript (using the latetest specification so we get the most features), TypeScript, or Flow. My personal recommendation is to use TypeScript since there is widespread support for it, and it makes the code more maintainable in the long run. That will be beneficial to us since our project will be spanning the entire school year.

In terms of Asset Bundlers, there are three leading options in the community currently - Webpack, Parcel, and Rollup. I believe we should try Parcel first to see if it works for us, since there is no configuration and everything we need should

work right out of the box. If Parcel does not give us enough granularity of control, My second recommendation would be Webpack because it is widely used and has great documentation.

For Continuous Integration, there are several options. I narrowed it down to three popular choices: Circle CI, Travis CI, and Jenkins. Circle CI and Travis are both reasonable options for us because they come at no cost for open source projects and they do not require the developer to supply the build machine. They are also highly similar, and we do not need to worry about the code running on multiple versions of node, since we will be putting our code through Babel to compile it to ES5. Jenkins, on the other hand, is free for the software but requires a build machine, which will increase our costs. Therefore, it is not the best option for an open source project.

## REFERENCES

[1] Babel, "What is babel?." https://babeljs.io/docs/en/, 2018.

[2] M. Schulz, "Typescript vs. flow." https://blog.mariusschulz.com/2017/01/13/typescript-vs-flow, 2017.

[3] A. Gerard, "Rollup v. webpack v. parcel." https://x-team.com/blog/rollup-webpack-parcel-comparison/, 2018.

[4] O. Shaporda, "Continuous integration. circleci vs travis ci vs jenkins." https://hackernoon.com/continuous-integration-circleci-vs-travis-ci-vs-jenkins-41a1c2bd95f5, 2017.

## 4.5 Evan

# Tech Review: Performance

Demonstrating WebXR through an educational physics experience

Group 47: Evan Brass (brassev) – Web Developer

**Abstract**

Our goal is to build an educational physics simulation that demonstrates the defining features of a WebXR experience. Of extra concern is keeping our experience running smoothly across our target hardware while running our physics simulations and rendering frames for the head-mounted display. We don't want to cause discomfort or nausea because we are lagging. To do this, we may need to incorporate other new web technologies that bring native-like performance to modern web applications. This document describes some of our initial plans for how we will run our application smoothly.

## I. Goal of the project

We will build a virtual reality experience for multiple devices that demonstrates the feature set exposed in the new WebXR standard. WebXR allows a unified interface for the variety of VR headsets, controllers, and also reserves surface area for future devices including artificial and mixed reality devices which haven't been invented yet. We want our project to serve the community by being a well-rounded use case. A use case whose code patterns and design solutions could transfer to other VR projects. We want to realize this new technology into something that other developers can learn from and reference to gauge the effort required to bring virtual reality to their own projects and inspire viewers to learn more about the physics presented or the technology we present it with.

## II. Introduction

My research is about how we will improve the experience if we find it becoming too much for one or more of our target hardware platforms. Each device's hardware is not something we can change. We must either utilize it more effectively or tune the content to match the available hardware. Utilizing the hardware more efficiently is usually something we can do before our app reaches the client. After it reaches the client we can only decrease the content that we show if we're lagging and to do that we need to be able to detect when we're lagging. The last experience that we can improve is the loading experience by making it efficient and comprehensible to the browser.

## III. Running Efficiently

JavaScript is the language of the web. Its design makes it especially suited to a fast-paced industry where the freedom of dynamic typing, interpretation, and garbage-collection work well. However, these features slow JavaScript. There're a few methods of getting around that. Note: I've read many times, "that premature optimization is the root of all evil", but I believe that premature optimization is the expression of loving your tools more than

your product. I think we'll need some of these technologies, but I imagine that we'll only need them in some sections of our project and not every single line.

The first method of improving our execution speed is to follow a set of rules that make for fast JavaScript [1]. These rules include not changing the types of items stored in arrays or of a field of on an object, not changing the prototype of objects after creation, etc. The performance gains we can get are upper bounded by how many rules we can remember or lint for. An alternative to that is using Google's Closure compiler. Closure can rename variables, remove dead code paths, and even inline functions [2]. But it's also largely an internal Google tool so it's harder to work with and the annotations that help inform it are clunky [3].

If writing JavaScript that's fast enough for our critical loops is too challenging, then perhaps we can utilize the new Web Assembly standard which is much faster than JavaScript. Web Assembly is a specification for a binary format that has functionality common to most modern CPUs. It has static types, manual memory management, and gets compiled to the native architecture's instruction format making it very fast and efficient. It has none of the garbage collection, and interpretation costs that JavaScript does [4].

It's still early enough that there isn't much tooling however. Debugging would be challenging. It also can't completely replace JavaScript because it has no direct access to any of the Browser APIs (yet). We would still need to call those from within JavaScript. There's a compiler which can do all of that for C++/C called Emscripten. It does all the Web Assembly compilation automatically and contains a runtime of JavaScript to affect the browser [5]. C and C++ aren't memory safe, though, so there's a reasonable chance that we'll encounter memory errors. Tracking those could be painful without better Web Assembly tooling like source maps that don't exist yet [6].

There's a language called Rust that is memory safe and can be compiled to Web Assembly, but it doesn't include the automatic JavaScript runtime that Emscripten includes [7]. The safety means we don't need to fear memory issues and might be able to manage without the debugging tools much better. But Rust's Web Assembly target doesn't have a runtime which means we would be writing a lot of glue code between our Rust and the Browser APIs we might need.

Unfortunately, we can't take advantage of the data race freedom inherent to Rust because JavaScript (which hosts the Web Assembly modules) is single-threaded. To get parallelism in JavaScript we need to use web workers and Rust's native multi-threading and thread safe memory sharing can't be directly turned into web workers. Web workers use an event-based channel mechanism for data sharing which is the only system we can use to communicate between workers [8]. The web has no alternative parallelism strategy. Workers are our only option.

## IV. Lagging Gracefully

Lag is inevitable and, like congestion in a network, indicates we're using the hardware to its best capacity. We don't want to overload the end user's hardware, but we don't want to underutilize when an improved experience can run on their computer's extra power.

### *A. Detecting Lag*

There's two ways that I found to detect if we're lagging and accommodate for it. The simplest way is to check the timestamps whenever we submit a frame. If the current timestamp is too far past our previous timestamp, then

we're lagging, and the user's hardware is likely accommodating for us. This method is accurate but means that we are using our render loop to check if our render loop is too slow. We want our render loop to be as fast as possible, so this doesn't seem like the best option.

An alternative is to use a feature called "requestIdleCallback" which lets us utilize the browser's scheduler to run something at a set timeout unless the CPU has some "free time" earlier [9]. This way we can run our evaluation function to check if we are getting lots of idle cycles and improve the content of our app or we can trim our content to relieve CPU pressure if we're consistently hitting that timeout.

### *B. Responding to Lag*

We have several features we can turn on or off to make an experience which runs smoothly for all of our target users. We can reduce the number of objects the user is allowed to have in the scene at any given time to speed up physics and rendering times. We could fallback to a more approximated and faster physics engine or engine setting. We could decrease or remove peripheral effects like shadows, particles, or lighting elements.

## V. Loading Efficiently

There's a couple of things we can do to make entering our experience an easy and quick operation. Two of them are to prefetch resources that we'll need later and to, in the case of Web Assembly modules, precompile them. In the case of ECMAScript modules (If we end up using them) we can fetch and the browser can even evaluate the module before we lazily load it later. The prefetching and evaluation system for HTML uses links in the header with different relationships. The three most appropriate like types for our project would be:

1) <link rel="preload"> - Requests that the browser fetch and load the resource
2) <link rel="modulepreload"> - For use with ES6 modules. The browser may be able to process the module before you even import it.
3) <link rel="preconnect"> - Open a connection to be used for the resource without sharing any information until it requested by the page.

There are other link types, but they mostly reflect relationships across domains, or between pages rather than describing the resources of a single page [10]. In the case of Web Assembly, we can prime the browser's cache using these links, but we'll still need to compile those modules after they arrive. This can be a low priority task scheduled with the idle callback from above or something else. Once compiled, we can share the same module across multiple web workers (via the channels) meaning less overall compilation.

## VI. All together

I'm recommending that when we encounter performance problems, we migrate some of our performance critical JavaScript to Rust that we compile to Web Assembly and run across – perhaps – a pool of web workers. I recommend that we experiment with both methods of lag detection – timestamps and idle callbacks – and pick the one that fits best with the rest of our design or is most efficient. I recommend that we try splitting our code into individual modules and utilize the browser's preload, pre-connect, and module-preload link elements to inform the browser

of those dependencies so that it can prime its cache. I also recommend that we use a tool like Closure to optimize and minify our JavaScript.

## REFERENCES


[1] A. Osmani, "How to write fast, memory-efficient javascript." https://www.smashingmagazine.com/2012/11/writing-fast-memory-efficient-javascript/, 2012.

[2] Google, "Advanced compilation and externs." https://developers.google.com/closure/compiler/docs/api-tutorial3, 2017.

[3] sisyphus, "Why isn't google closure more popular?." https://www.reddit.com/r/javascript/comments/k0xsu/why_isnt_google_closure_more_popular/, 2011.

[4] j. l. f. P. a. A. r. S. p. u. johnstevenson, chrisdavidmills, "Webassembly." https://www.reddit.com/r/javascript/comments/k0xsu/why_isnt_google_closure_more_popular/, 2018.

[5] E. Contributors, "emscripten." http://kripken.github.io/emscripten-site/, 2015.

[6] j. d. k. t. r. m. s. j. d. f. b. l. sunfishcode, lukewagner, "Will webassembly support view source on the web?." https://github.com/WebAssembly/design/blob/master/FAQ.md#will-webassembly-support-view-source-on-the-web, 2018.

[7] R. W. contributers, "Rust and webassembly." https://rustwasm.github.io/book/, 2018.

[8] A. listed in the website footer¿, "Using web workers." https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers, 2018.

[9] P. Lewis, "Using requestidlecallback." https://developers.google.com/web/updates/2015/08/using-requestidlecallback, 2015.

[10] A. listed in the website footer¿, "Link types." https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types, 2018.

# 5 Blog Posts

## 5.1 Jonathan

### *5.1.1 Fall Term - Week 3*

Progress:

My group and I drove up to the Intel Jones Farm campus in Hillsboro, OR on Thursday. There we were able to meet and discuss project details with our client Alexis Menard. Additionally, we were able to meet members of the open source development group from Mexico that will be assisting with our development. We learned that we will be working with the WebXR API to develop an interactive VR experience on the web that does not require client-side software. This application will be a leading example for other developers that wish to use the WebXR API and should serve to bring awareness to this cutting-edge technology. During our meeting we brainstormed ideas for several different VR experiences that we could develop. I was able to get a good deal of notes from the meeting which I distributed to my group members. After we returned from Intel I created a Github repository and organization in which the source code for our project will be stored.

Problems:

Things have been running very smoothly for our group. The only issues we've encountered are phantom group members and long commutes. Time conflicts have been an issue as one of our group members, Evan, was unable to attend the Intel meeting because it would have caused him to miss a quiz.

Plans:

As discussed in the client meeting, we will be brainstorming ideas for what our project will be, drafting up assets and contacting UX resources within Intel through at the latest, mid-November. A Google doc has been set up where we can write down our ideas for the project so it can be shared with the entirety of the group (including those from Intel).

### *5.1.2 Fall Term - Week 4*

Progress:

Completed our problem statements. No new progress besides this.

Problems:

Combining all the problem statements.

Plans:

Make small edits to the final problem statement.

### *5.1.3 Fall Term - Week 5*

Progress:

Group members and client voted on project ideas from a google doc. We have decided to make a VR Physics simulation for educational purposes. With this, we can start working on the requirements document.

Problems:

We have not really started our requirements document and it is due very soon. We probably will not have time to contact our client with the draft.

Plans:

Meet with team whenever possible to discuss possible requirements for the project. Finish the requirements draft before Tuesday.

#### *5.1.4 Fall Term - Week 6*

Progress:

Client is now completely up to date with the progress of the group and the assignments that we currently have. He has actively been providing feedback on our current and previous work. Each group member has been assigned some components of the project to write about for their tech review. All of our documentation is now being written in our shared google drive with the client where he can collaborate with us.

Problems:

Not all of the team members have been using the drive as of yet. I will try to get everyone using it so that the client is happier and more productive in his assistance with our project.

Plans:

I still need to complete my tech review and I am still researching the components I have chosen, physics, graphics and audio.

#### *5.1.5 Fall Term - Week 7*

Progress:

Finished tech review and contacted client about what their preferred technologies are for the components I was writing about. It seems that we are on the same page.

Problems:

No issues as of last week. Really, nothing to talk about here.

Plans:

Continue to communicate with client about their preferences for the project. Review team member's tech reviews to gain better understanding of the project as a whole. Mess around with some of the technologies to try and learn how to use them before implementation of the project begins.

#### *5.1.6 Fall Term - Week 8*

Progress:

Not much going on this week. I've been building a personal website that uses Cannon.js, Three.js and HowlerJS for fun and for practice. Getting myself familiar with these tools will certainly help things to get started more smoothly when implementation actually starts.

Problems:

I won't be present for Alexis's lecture and subsequent group meeting on Tuesday. I have moved my flight to Sunday and hope that I can video conference with my group while Alexis is there.

Plans:

Continue programming my website, eat too much next week.

#### *5.1.7 Fall Term - Week 9*

Progress:

The team met with the client, Alexis Menard on Tuesday after his lecture in the capstone meeting. Much was discussed concerning the future of the project. This includes: Physics simulation content, plans for the break and hardware.

Problems:

I was not present for the meeting with the client and despite multiple attempts to contact my team members by Discord and phone I was unable to connect.

Plans:

Continue to catch up with my teammates on what was discussed Tuesday. Work on the design document and complete before the due date. Contact client with approval for the design document.

#### *5.1.8 Winter Term - Week 1*

Progress:

Met with client via Google Hangouts to discuss how to move forward with implementation. This included project metrics and management tools. Our team began implementing these metrics and management services on our github page. For our management tool we decided to use Github Projects. We've begun adding issues to the github and are slowly implementing those issues. A standing appointment with our client has been setup for Thursdays at 9:30am.

Problems:

No problems so far. I think that as we move further into implementation we may have conflicts with our code.

Plans:

We are working to figure out standing appointments to meet with the OSU team members at least twice a week. As for avoiding code conflicts, it is important that we keep adding issues to the github.

#### 5.1.9 Winter Term - Week 2

Set up a SCRUM development cycle with group. Kanban board, stories, tasks and sprints. Currently in sprint 1. Met with client, discussed some properties of the WebXR API.

Problems:

Trying to setup frequent standups with group. Nobody seems to be able to settle on a time to meet. Need to get everyone's schedules and figure out what time is best.

Plans:

Make sure everyone continues to understand their tasks and responsibilities. Figure out what time is best to meet for frequent standups.

#### 5.1.10 Winter Term - Week 3

Progress:

Sprint 1 is halfway complete. Website contains a three rendering canvas with html/css styling around it that explains what the project is and introduces the experience. Room switching, enter vr button, magic window setup is currently underway. Everything is on track and running smoothly. Group members are performing fantastic in their tasks.

Problems:

Some of us have conflicting programming styles which makes it hard to keep consistent code. All the code they write is great, I'm just not familiar with some of it.

Plans:

Continue to work on sprint one and delegate work as needed between team members. Hopefully everyone will have some comfort with the API's that this website is using by the end of this sprint.

#### 5.1.11 Winter Term - Week 4

Progress:

Nearly completed sprint 1. Should have been done by Thursday but I'll get to that in problems. Team met with client Thursday morning like we always do. Discussed some changes in the WebXR API and the possibility of coming up to the Intel campus for a "Hackathon" to burn through most of the project's code. After client meeting, met with group to discuss how we should tackle the next sprint. We decided that we didn't know enough about long term development practices with the APIs that we're using.

Problems:

The WebXR API is unstable and is constantly changing. Some of the code we've already written has become deprecated and so we must handle that. As a result, getting the Magic Window setup for Sprint 1 was a failure and was not completed on time. The rest of the team is also unsure about how to develop with this API in the long term.

Plans:

The team will take this weekend to research each of their assigned topics for the project. By the end of the weekend we should all have some understanding of what our code should look like moving forward and what sort of program specific tasks we need to complete. On Monday we will reconvene and plan sprint 2. Sprint 2 will most likely consist of finalizing the WebXR Setup, testing for VR input and setting up the CannonJS foundation.

#### *5.1.12 Winter Term - Week 5*

Progress:

Completed sprint 1, research and planning for sprint 2. Sprint 2 is currently in progress.

Magic window is complete and functional on the site. Switched from using the WebXR polyfill to using the actual API. XR boilerplate is mostly setup.

Problems:

Nowhere near alpha readiness, need to get lots of work done in the coming two weeks.

Plans:

Complete work on sprint 2 and modify tasks as needed to meet deadline.

#### *5.1.13 Winter Term - Week 6*

Progress:

Not much change from last week on my end. I'm still working to implement the two eye rendering but have run into some issues with that. Parts of the API appear to be non-functional with the way we've structured some of our code. A refactor may be possible.

Problems:

Forgot this blog post. Everyone's work seems sort of disconnected. Unlike the last sprint, I'm not really seeing much progress on the repo. Everyone seems to have a task that is kept to themselves. I am worried about bringing all of our work together in the coming week.

Plans:

Get the two eye rendering to work.

#### *5.1.14 Winter Term - Week 7*

Progress:

Completed sprint 2, setup sprint 3. Immersive two eye rendering set up. Can interface through VR and cardboard headsets.

Problems:

No big issues right now. Everyone has completed their tasks. Need to make sure that everyone has time to meet to work on the alpha write up which should take some time to complete.

Plans:

Need to complete alpha write up and make sure code is clean for alpha deployment. Working to complete big tasks in sprint 3 such as VR input, movement and experiment setups.

#### *5.1.15 Winter Term - Week 8*

Progress:

I've got some basic VR input working. The controller position and rotation is transformed onto a basic cube model. Sprint 1 is halfway done. There is still a lot of work left to be done. We met with our client on Thursday and discussed going to Intel for a hackathon on April 5th. By that time, most of our project will be done. This will be a great opportunity to test our code and have it reviewed by our client.

Problems:

Translating the position of the controller is proving to be difficult. The matrix transformation code that I previously used to translate the projection matrix of the headset is strangely not applicable to the controller matrix. This will take some thought...

Plans:

Meet with group on weekend for a standup and to bust through most of this sprint. There is a lot of work left to be done and only one week left in this sprint to do it so we're gonna need to shift into overdrive for this to be complete.

#### *5.1.16 Winter Term - Week 9*

Progress:

Finally got the controller to translate with the user throughout the scene. It looks like the issue was that the user position value that we use to translate the viewMatrix is inverted because world translations need to be inverted so that the camera moves to where you want it to. When I was using the user position to translate the controllers, they were to opposite ends of the room. Fixed all that, started rendering controllers and now I can move on to doing some raycasting from the controller. Otherwise, the sprint is still incomplete.

Problems:

Sprint 3 was supposed to end on Friday but I had to extend it another week because nobody (including myself) was able to get our tasks done in time. We're all pretty busy, working on capstone is a challenge.

Plans:

Finish up this sprint, no more delays. Finish off the webxr implementation once and for all so that we can all focus on getting the physics to work.

#### *5.1.17 Winter Term - Week 10*

Progress:

Sprint 3 is completed. All tasks for that sprint have been completed and moving onto spring term will be easy. Most of the functionality for the site is there.

Problems:

Working on capstone has been a challenge this week because its dead week. Lots of projects are due and there are finals to be done next week.

Plans:

Enjoy spring break and pass finals.

#### *5.1.18 Spring Term - Week 1*

Progress:

Most of the functionality is there. Still some requirements that need to be met. Just finishing up the features now.

Problems:

There are some features which have yet to be implemented.We are running low on time so hopefully we can get those done.

Plans:

Get all features implemented before the code freeze. Update requirements and design document.

#### *5.1.19 Spring Term - Week 2*

Progress:

Finished up VR interactions and global raycasting unification. Each team member is now working on their individual scenes.

Problems:

Not a lot of time left, some features won't be implemented before the code freeze. Still need to finish editing requirements document.

Plans:

Finish editing requirements document. Bring project to beta stage for the code freeze.

#### *5.1.20 Spring Term - Week 3*

Progress:

Completed beta version of project. Signed off with client on design and requirements documents. All core functionality of the site is ready.

Problems:

Team members are losing motivation. Now that the code freeze deadline has come and gone, getting meetings/sprints set up is a challenge.

Plans:

Generate one more sprint before the expo to polish the project.

#### *5.1.21 Spring Term - Week 4*

Progress:

No real progress since last week. We polished up our poster for the final poster review by mostly adding screenshots and taking a group photo. Only code changes have been a patch to handle a recent WebXR API update that broke one of our deployments.

Problems:

Senioritus. Everyone is pretty unmotivated at this point since we've mostly completed the project.

Plans:

Our client plans to demo our project to the google.io group so he would like us to polish up our code/experiences so that they are the best they can be when he shows them.

#### *5.1.22 Spring Term - Week 5*

Problems:

Nothing going on this week. Everybody has been working on other classes.

Progress:

No progress. Everything is done now. Just waiting for the expo.

Plans:

Prepare for the expo presentation.

#### *5.1.23 Spring Term - Week 6*

Progress:

Still no progress this week. Our client went to Google io to show off our stuff.

Problems:

We were placed outside for expo which may be an issue. It’s going to be raining that day so hopefully we have a tent.

Plans:

Talk to client today, coordinate getting our equipment for expo. Prepare for expo presentation.

## 5.2 Brandon

### *5.2.1 Fall Term - Week 3*

Once the group was assigned we all contacted our team members and introduce ourselves via email. As far as communication, we agree that email is a bit slow so we all decided to use a discord server to communicate better. Then we contacted Kevin and Kirsten before contacting our client. After, we got a go ahead to contact him. Our client wanted to come over at Hillsboro to discuss face to face to do some brainstorming and feedback of our capstone project ideas. Everyone but one couldn't make it to the meeting because of scheduling conflict. I had the time and went to discuss couple ideas of how we can take it along with my group members. We discuss how we can take the VR/AR immersive project. For example, Exploring 01.org content in VR and selecting 3-4 projects and map then into a virtual set of rooms, VR events, Battleship games, Physics simulator, Stadium/Theater seating position, a better way to see complex graph/data, and product viewer. When our group went to the meeting to brainstorm ideas and understand problems to solve, we have many ideas. Right now we want to narrow or come up with a better ideas that our group will agree on. There was a minor confusion when another student contacted our client saying he was in our group but it was later resolved when we contacted the instructors and was not actually in our group. The current plan is that our client share a document of the ideas we all brainstorm and we allow to edit and add more project ideas while we later narrow how we can make VR/AR immersive to put it in the 01.org.

### *5.2.2 Fall Term - Week 4*

During week 4, we added some more ideas and refine the details of the project ideas document. Our group met up at Thursday to work on the group problem statement assignment. We all share each of our problem statement and figure out which format style since we all have different template. Once we have a default template to use, we work on adding/revising our group problem statement while also using individual peer review feedback to add and fix. On Friday, our group met up with our TA and talk about project progress and discuss plans and problems. Then finish up the group problem statement and adding/revising. Our current problems right now is that we need to do some research once we know more of the requirements of our project so we can move on to the next step. Then create detail document of what we are going to work on for this year. We already have some of the requirements that is included in the document like putting it in the 01.org portal. After that, we need to get in touch of our client and then have our client approve the document. The current plan is create a requirements document. We already have some ideas of the requirement when our group went to Intel to brainstorm VR/AR capabilities and email the client with further more detail of the requirements for the client needs. While still brainstorming and add more project ideas while getting the requirements for our project and later narrow how we can make VR/AR immersive to put it in the 01.org portal.

### *5.2.3 Fall Term - Week 5*

Our client emailed us to incorporate our feedback on the document and then vote number one and number 2 on each of the demos. Then extend on them with our ideas. We meet up with our TA and discuss our progress and problems of the documents. We started the requirements document and created all of the basic outline based on the IEEE stuff. Our current problems right now is for the tech review draft because we have to a little more research since VR/AR

technology is still cutting edge and not easy to defend the choices to others outside of our project. Since the requirements document and tech review document got pushed back. I need to plan ahead because I two midterms at week 6 and figure out. We talked our TA about the tech review requirements an will know more soon.

#### *5.2.4 Fall Term - Week 6*

Our group met up on Monday to work on the requirements document and discuss the different option requirement of a physics simulator. On Tuesday class we plan and assign tech review criteria. While continue discussing on discord. Our TA is currently out of town this week but we didn't have any problems yet. We haven't encounter any problems this week. Although we did email our client about what we have been working on like the software requirements specification document and tech review coming up. We asked him to review and give us feedback on it. He later email us to post on the google group that he has created. This would help to have a history of our discussion and create a folder to manage all the capstone assignments in google docs format so he can comment on it. Alexis Menard has scheduled on 11/20 to be giving a presentation + live coding and asked us to meet before an hour of class and discuss the project.

#### *5.2.5 Fall Term - Week 7*

On Tuesday we had our in-class Tech Review. I have some good comments and feedback on my tech review document. My client also peer review my tech review draft and made some comments. My current problem is that revising my Tech Review from my client requirements while researching more tech option without being out of scope. Revising my in class peer preview for my tech review. We gotten a Fall Mid-Term Checklist to make sure our project is not behind. We already have our Team Name including our group number, need to remember of naming convention of files. Our team will plan to start a list of potential items and source of item while becoming familiar with our tech and any new languages, skills, etc as soon as possible.

#### *5.2.6 Fall Term - Week 8*

Not much going on this week but discussing Design Document soon and still waiting for my Tech Review to be graded. My only confusion was the Design Document requirements description but later cleared up with our TA via email. On 11/20 our client will come before class and hopefully discuss on Design Documents and other things.

#### *5.2.7 Fall Term - Week 9*

On Tuesday we met with client to discuss more of our project requirements. The live code demo provide us a better understanding what what we have to do. Currently there are no problems yet for our design document. After meeting with our client we come up with more ideas and more narrow direction of our project. We plan to have 3-4 experience of physics simulator and want to showcase to like middle school, high school physics.

#### *5.2.8 Winter Term - Week 1*

Progress
On Thursday our group used google hangouts to discuss and start getting organize the work coming up. Plan what to

begin and provided with some document for us to look at. We starting to get the project setup and running on localhost and start developing.

Problems

Our current problems is to decide the rest of the physics simulation experiments. We only decided two right now.

Plans

Our plan is to create issues of the physics experiments to work on and use Github Projects development progress. Alexis manage to lend us a android phone and a VR headset for us to test. But we won't get it until next week. One of our group has a android phone but the rest has Iphones.

#### *5.2.9 Winter Term - Week 2*

Progress

Our group create a sprint 1 list of task that gets the basic structure of our experience running before we move on to sprint 2. On Thursday we had a meeting with our client to talk about our progress.

Problems

We had a problem getting organized enough to jump headfirst into developing so we decided a SCRUM system with sprints to iterate and decide a task list for a 1-2 week sprint.

Plans

We plan to meet with our group to talk about the Elevator pitch and figure out what to do. Our current plan is to finish sprint 1 at least till the next meeting with our client.

#### *5.2.10 Winter Term - Week 3*

Progress

We are about to finish our sprint 1 and start sprint 2 soon.

Problems

Currently we don't have any major problems but we are slowly handling our organization.

Plans

once sprint 1 is done we going to start planing on sprint 2 when we meet and figure out what's next.

#### *5.2.11 Winter Term - Week 4*

Progress

We are finishing up Sprint 1, talked to our client of our progress and more about WebXR API. I implement some of the HTML homepage and also looking at input controller.

Problems

We had a little problem encountering magic window to get it working. For sprint 2 we not sure what to tackle first so we are going to split task on research and findings to have a better understanding to more forward.

Plans

Our current plan is to divide task for us to research and come back on Tuesday to plan on building up Sprint 2.

#### *5.2.12 Winter Term - Week 5*

Progress

This week, our group had a meeting to discuss our research over the weekend. After that we started creating issue for sprint 2 of our research while getting magic window work. Our group work on other their experiments, Keyboard and Mouse controls, drag and drop For me, I researched Cannonjs, so I started working on creating a scene for the cannon world and start adding physics to the room so I can work on the gravity experiments.

Problems

We got magic window to work but some aspect ratio on the resizing kept messing it up.

Plans

My plan is to try work on the physics library and starting to understanding it.

#### *5.2.13 Winter Term - Week 6*

Progress

This week I got some of my task done. I install cannon.js to the node modules, I setup cannon world to the xr-scene.

Problems

my only issue is to get the physics working but I got the 3D to display in the room.

Plans

Since the due dates of the rough draft of the final progress report was push back to Monday week 8. This will give us a bit more time to work on our alpha build and rough draft. We need to plan ahead on asking our client before the weekend cause he rarely response during the weekends.

#### *5.2.14 Winter Term - Week 7*

Progress

I finished setting up cannon and got the physics working in the scene with threejs. Got 2 eye rendering to work and controls.

Problems

There was an issue recently where I got the physics working fine but it effected another scene. I think it has something to do with the timestep. I hopefully can figure out the problem over the weekend.

Plans

Once we finish sprint 2, we are starting to plan for sprint 3. we are meeting Saturday 2/22 to work on alpha progress report.

#### *5.2.15 Winter Term - Week 8*

Progress

This week I haven't do much progress myself this week. I was working on the Pop-up menu and displaying a visualization for velocity. Our group register for expo and we planned for a hackathon on next term first week of spring to get most of the bugs iron out.

Problems

Our current problem is the two eye rendering broke on one of the eye on the planets scene and need to figure out what causing it.

Plans

I should finish my issue before next Friday. I plan to work on it on Wednesday to get the pop-up menu interface working so we can change the gravity or spawn objects.

#### *5.2.16 Winter Term - Week 9*

Progress

My current progress I work on is to get the menu interface gui to able to change the gravity or add velocity vector visualization on the scene. I got the menu to display and change the gravity. I will later add velocity later.

Problems

My current problem is that the cursor from the mouse is not accurate from the actual camera.

Plans

I am trying to find a time to work on capstone but I should have more time on finals week than week 10.

#### *5.2.17 Winter Term - Week 10*

Progress

My current progress I worked on is moving my menugui to a separate file so we can import the menuGUI when ever our experience needs it. I was testing it in the falling object to see if it was working.

Problems

My only problem for me was the camera stuttering but I manage to figure out today that it was my video card in SLI mode. I had to disable that to fix it.

Plans

Our next plans are to plan our next sprint and start getting assets for our experiments.

#### *5.2.18 Spring Term - Week 1*

Progress

On Tuesday we gotten together and create new issues for our sprint 4. After class on Friday we went to Intel to meet with our client and discuss/plan/code.

Problems

Our current problems is getting enough stuff done before the code/freeze.

Plan

We should have everything ready and working before code/freeze. Our Sprint 4 will have our core components ready.

#### *5.2.19 Spring Term - Week 2*

Progress

Our group is finishing up their part enough for a code freeze. I work on displaying an arrow on a object vector and still figuring out updating the vector direction base on object vector.

Problems

Me and Alex were working on the datgui for VR trying to get the interaction within the xr-scene but we found out that the addObjectInput works well for webVR.

Plan

We plan to update our document as early as possible and make sure our code works when code freeze.

#### *5.2.20 Spring Term - Week 3*

Progress

This week we completed our code freeze and the main WebXR experience works fine. We still have to work on any issue of the other VR devices and add more feature to make the experience better. Next sprint I plan to work on add more stuff to the kinematics room like some physics settings like pausing time, changing mass etc.

Problems

I wanted to make datguivr work for WebXR to make the kinematics experience more user friendly but its only works with WebVR and not WebXR. Im not sure if we going to make it work.

Plan

Our group decided to not meet for the next sprint and just add more issues to github and make the experience better since the core project is finish just need to work on the poster, fix problems to make Vive work again.

#### *5.2.21 Spring Term - Week 4*

Progress

This week has been light and mainly work on the poster with our group picture taken. We gotten feedback from our client about our project and suggest what needed added or change.

Problems

Currently there are no major problems.

Plan

I am going to work on what our client suggest on the Kinematics scene changing position of the object spawning and

buttons.

#### *5.2.22 Spring Term - Week 5*

Progress

The Kinematics scene I decided to not use Datguivr since it doesn't work and instead use a slider to change scalar value of gravity, with the value showing overhead and a sphere that you can grab to change the direction of gravity based on the direction of the user's controller. The Planets scene has a exit home on all planets.

Problems

We currently only have problems applying texture on the pendulum scene.

Plans

I think will just plan and prepare for Expo.

#### *5.2.23 Spring Term - Week 6*

Progress

The only thing we had to do is the pendulum texture.

Problems

We are basically ready for Expo and should have a backup plan for wifi cause we are outside.

Plan

On Friday we talk to our client about Expo and how we going to prepare for it. Alexis is going to bring two chromebooks, 2 HMD, PC, some cords.

## 5.3 Tim

### *5.3.1 Fall Term - Week 3*

This week went rather well considering we just found out our project this last Monday. With that being said, we still did accomplish a decent bit as a team. I got in touch with my teammates through email and we eventually set up a discord server for communication. Brooks, one of my teammates, got in contact with our client, Alexis Menard. We struggled a bit trying to find a good time to meet up and go over the project, but eventually, we came to Thursday at 2 PM. The group went up to the Intel offices in Hillsboro and met up with Alexis as well as the Intel team that works on maintaining the 01.org website that our project will be hosted on. There, we went through a brainstorming session in order to come up with ideas for what the project will be on. We came up with some interesting ideas, such as a virtual reality Battleship game, a program that would allow users to interact with seats in a stadium to see what the view might be like in a virtual environment, a physics simulator that lets users mess with and change the laws of physics, and others. We were also given a small tour of Alexis' office space and meet some of his co-workers that may be able to help us along the way. All in all, I believe this was as productive of a week 1 as you could get. We hope to continue to collaborate and come up with a final project idea soon.

### *5.3.2 Fall Term - Week 4*

We worked on compiling our problem statements into one final draft. We also met with our TA and talked a bit about the project and what we plan on doing in the future. This was another quiet week, although there wasn't much really to do. There weren't any problems really, we worked well together using Overleaf to share and write the final draft of the problem statement together. Next week, we plan on getting together again to work on the requirements document. Alexis, our client, will also be coming down to Corvallis to talk with us some more about the WebXR API that we will be using sometime in the future.

### *5.3.3 Fall Term - Week 5*

Another quiet week. We voted on what will we be doing for our Immersive WebVR experience. It looks like the most likely option is a physics simulation based on the votes, but at the end of the day, Alexis, our client, has the final say. We started on our requirements and have a few ideas brainstormed. We plan on getting those done sometime between Saturday and Monday. We also plan on each doing our tech reviews next week. A date has been scheduled for when Alexis will end up coming to talk with us about WebXR (November 20th, he'll be speaking in class and then talk with us 5 alone for a bit after).

### *5.3.4 Fall Term - Week 6*

We finished the requirements document and completed our individual tech review documents. For the tech review, I personally wrote about user interfaces, including the generation and functionality, as well as controller interfacing/support. So far, we have yet to run into problems. We are still in constant contact with Alexis and updating him of our progress through a shared Google Drive. He went out of the way to comment on our documents and give us feedback as well as suggestions. Next week we will be finalizing our tech review.

#### *5.3.5 Fall Term - Week 7*

This week was focused more around individual work. We each finished up our final drafts of our tech reviews. Personally, I fixed grammatical errors as well as got rid of any first person language, such as I, we, or our, to make it more formal. As mentioned in last weeks progress report, I wrote about the UI visuals/functionality and controller interfacing. We have still yet to run into any problems, our group works together pretty well. It's getting closer to our second physical meeting with our client, coming up on the 20th. I'm looking forward to it as it will hopefully clear up any questions we have regarding the WebXR API that we are going to be using.

#### *5.3.6 Fall Term - Week 8*

We got a break this week as nothing really happened. We talked about the design document at our meeting. No problems have come up. Next week, we will meet with Alexis after his guest lecture and will likely continue going over the design document and draw on him for some additional input.

#### *5.3.7 Fall Term - Week 9*

We met with our client, Alexis Menard on Tuesday and went over what he would like us to get figured out by the end of the term. We plan on figuring out what physics experiments we will set up in our simulator before the term ends.

#### *5.3.8 Winter Term - Week 1*

We met with our client, Alexis, via a Google Hangouts call and talked for around 45 minutes. We discussed our plan of action for documenting and organizing our development of the project going forward. We plan on using Github Projects to keep track of our issues and progress made as well as who is working on what. We have begun creating the issues on Github and will hopefully have most, if not all of them, set up by the end of the weekend. I personally plan to start working on implementing my assigned issue this next Monday. We also decided to schedule a weekly meeting with Alexis at 9:30 AM on Thursdays to talk about our weekly progress and our plans for the next days. This way Alexis can be up to date on what we are doing and we are able to come to him with any questions or concerns we may have. We haven't run into any problems, thankfully, considering it's only been a few days into the term.

#### *5.3.9 Winter Term - Week 2*

We continued in getting ourselves organized. We have started a sudo scrum process with 1-2 week sprints. We plan on trying to meet at least 3 times a week. Thursday, we met with our client at the scheduled time and he provided more resources for our development as well as some suggestions for organization. As for the development of the project, we have most of the tools connected, such as Parcel and Three.js, and have written some boilerplate Three.js code for creating a scene and room within the scene.

We got our first sprint in order after the client meeting and have tasks assigned that should result in basic rooms with the ability to move between them with a door that theoretically loads the connected room. This will be used for traveling between the multiple experiment stations that will eventually be implemented. We also aim to implement a basic control scheme for keyboard and mouse.

#### *5.3.10 Winter Term - Week 3*

Things are still running smoothly. We are halfway through our first sprint and on track to finish it a bit early. I have finished my responsibilities and will be looking to help out others as well as look towards tasks that I can work on for sprint 2 and get a jump start. Our weekly meetings with Alexis are going well and will surely be a great way of getting help once we get into the meatier parts of the project. Next week, we will finish up the sprint and layout our plan for the next one.

#### *5.3.11 Winter Term - Week 4*

We finished sprint 1 and started planning for sprint 2. The tasks have not been decided yet, but each of us have chosen a topic to research and we will regroup on Tuesday to create and assign tasks. I am researching XR input to learn better about reading and handling input from VR controllers and the like.

#### *5.3.12 Winter Term - Week 5*

We assigned tasks for sprint 2. I will be working on controller inputs, including controls for keyboard and mouse, HMD/Daydream (VR headsets with controllers), and touchscreen. Currently, Im working on keyboard and mouse first as getting these controls done early will be useful for testing purposes. I plan on then working on touch controls and then will likely end up working with Evan on teleportation controls for HMDs. However, our client has notified us that a change is being made to the API with the intention of making dealing with XR controller input easier. This means changes would have to be made whenever these changes publish. With that being said, Ill still continue with the current plan as we arent sure when these changes will be finalized.

#### *5.3.13 Winter Term - Week 6*

I continued work on the keyboard and touch controls. I got basic keyboard movement working, I still want to get object interaction implemented, such as being able to drag and drop objects using the mouse. The touch controls still need work as there is a slight issue with the website since it tries to scroll down when trying to use the touch controls. I plan on fixing these touch controls issues as well as the keyboard and mouse drag and drop controls which can hopefully double down and be used in part with the touch controls as well. Theres also some problems with the two eye rendering which need to be resolved before teleportation controls can be implemented since without the immersive VR mode there is no way of testing/using the teleportation controls.

#### *5.3.14 Winter Term - Week 7*

I worked on fixing scrolling issues with the touch screen controls. This was solved by implementing a full screen view of the canvas. I also implemented the full screen canvas for the keyboard and mouse as well since the option was available. There are a few issues that need to be solved. Some devices, specifically laptops that can act as tablets, arent getting keyboard controls. This is because they do support magic window gyro control as they can also operate as tablets. The issue is I would like to be able to give them the option of using touch controls or keyboard controls, but am unsure of a way to check what device is in use as someone on a mobile phone would not want to use keyboard controls. Another

issue is with the full screen and the event listener attached to full screen changing not being fired off when some devices exit full screen.

Over the next sprint, I hope to finish up any issues surrounding the non-immersive controls and get the immersive VR controls setup. I believe Evan is working on getting interactivity working with VR controls, so I would like to use some of his work to implement the same interactivity into the non-immersive controls. Finally, I would like to get started on the falling objects experiment and get some basic interactable objects set up and spawnable.

#### *5.3.15 Winter Term - Week 8*

I began work on the falling objects experience. At the moment I am trying to implement the ability to spawn objects by clicking a static object that will be sitting on a 3D table within the scene. So far, I have been able to successfully spawn an object that is linked with a Cannon.js body, however I still have to implement some sort of event listener for when the static object is clicked to call the spawn function.

Next week, I plan on having the spawning fully implemented and will be moving on to creating some sort of interface for the user to use to manipulate gravity. At the moment, I have a simple keyboard event listener setup so when they press G it turns gravity off or on. Eventually, the goal is a fleshed our UI that lets the user manipulate the direction and value of gravity. This should be possible thanks to Cannon.js taking in a 3D vector for the gravity property.

#### *5.3.16 Winter Term - Week 9*

I worked on getting objects spawning working on mobile devices. This was successfully done, however the table model Im loading into the scene still isnt loading completely properly, including an issue with the textures. Next week, I plan on getting the home room set up with doors that will essentially connect all of the experiences together. I also then plan on getting the falling objects scene working with immersive vr controls, which will hopefully be implemented by the time I get to it.

#### *5.3.17 Winter Term - Week 10*

I implemented some basic features that we put on our requirements document that havent been implemented yet, such as navigating between scenes with doors, pausing and playing animation/physics of objects, and manipulating gravity, like being able to reverse it. Next week, we are going to try to put everything together, so anything that hasnt been merged into the master branch will hopefully be pulled into it before the end of the term.

#### *5.3.18 Spring Term - Week 1*

We got together and set up what will be our final sprint before the code freeze. I will be working on the laser experience, creating a puzzle sort of thing with a laser, ending goal, and mirrors that can be used to reflect the laser to the end goal. Friday, we went up to Intel for the day and worked with Alexis to pump out some progress. We got a decent amount of work done by the end. This next week will be a push to make sure everything we want finished is finished by the freeze.

#### *5.3.19 Spring Term - Week 2*

This week has been a push to get the finish line. I've been focused on the laser reflection experience and have gotten it nearly completed. At the moment, mirrors are able to be spawned in, moved around, rotated, and deleted. There's also a pre-set laser in a fixed position that will reflect off of any mirrors it intersects with. A goal is randomly spawned in the scene with the intention that the user will try to make the laser reflect off of mirrors to make the laser intersect with the goal (it lights up green to let the user know they were successful). Finally, teleportation movement is also finally implemented and added to the scene. All that's left is to add some text to better label the buttons used for creating, moving and deleting mirrors as well as display the angle of reflection of the laser above any mirrors that are reflecting the laser. If time allows, I would also like to tweak how the users rotate the mirror, as currently, they can only rotate it in a single direction, which can be slightly frustrating if they overshot the angle they were aiming for by a tiny margin.

#### *5.3.20 Spring Term - Week 3*

We finished up the project for the most part as well as revised our design and requirement documents. I'm pretty happy with the overall result of the project, however we did run into an unfortunate issue: it's not working on HMDs (VIVE/Oculus Rift, etc). It seems to be an issue with the WebXR API which is out of our hands, so hopefully, the grader understands this. Our client knows about the issue and is trying to look into finding a solution, which he can hopefully do soon (definitely before expo). Going forward, we're mostly just going to add finishing touches to the project, such as a few new textures, or fix a couple of minor bugs/features that could increase user experience.

#### *5.3.21 Spring Term - Week 4*

Our project is essentially complete with just a few finishing touches being added here and there, so it's quite relaxed. We worked on our expo poster and got it basically finished. Personally, I only made a minor tweaks to the laser scene, those being the mirror placement outline now reflects the laser, angles no longer can go negative, mirrors are prevented from being moved outside the room, and the most obvious/useful being a brief guide on the control scheme on one of the walls that users can read so they actually know how to use the menu. We're still working with Alexis to find a fix to the issue with using a VIVE, but I'm confident it'll be resolved within a week or so. Alexis left some feedback and gave us a few suggestions for polishing the project which we'll probably work to get done next week.

#### *5.3.22 Spring Term - Week 5*

Not too much happened this week. I did a bit more work on the kinematics and laser scene, but apart from that, nothing was done. The project is essentially complete, so we are just waiting for expo at this point. I've added all of the finishing touches I wanted to add to the project, so I don't plan on doing anything for it next week.

#### *5.3.23 Spring Term - Week 6*

The project is basically done, so I didn't do anything this week regarding capstone apart from our weekly meeting with our client where we talked about expo and checking the boxes to make sure everything is in order. Next week is expo!

Our client will be meeting with us the morning of to help us set up all of the hardware that he is supplying for us to use for demos.

## 5.4 Brooks

### *5.4.1 Fall Term - Week 3*

This week, I contacted my group mates and we met after class to discuss how we wanted to stay in touch and how to contact our client. We ended up making a discord server so that we could message, call, and share files. I sent an email to our client introducing myself and the rest of my team, and he invited us up to Intel to brainstorm ideas for the project.

We discussed potential ways to showcase the Web XR API, and decided together that we would take until the beginning of November to continue flushing out our ideas and then vote on them. If there are two smaller ones, we may split into subgroups to work on them separately.

So far we haven't had any problems, and we hope to keep it that way.

### *5.4.2 Fall Term - Week 4*

This week, our team got together to merge our problem statements into one final draft of the report. We used Overleaf to collaborate on the document at the same time, so we could pull parts from our individual problem statement essays and combine them. We also met with our TA for the first time and described a little about our project and what we had done so far. I also peer reviewed someone else's problem statement so they could get some feedback before the final draft.

We didn't encounter any problems this week.

We plan to meet with our client virtually to discuss the actual project we're going to use to showcase the WebXR API before the end of the month. From there, we will plan our our next steps.

### *5.4.3 Fall Term - Week 5*

This week, our team was planning to work on the requirements document, but the deadline got pushed back so we decided to put it off until this coming weekend. We voted on the direction our project would take, and have decided that we will be making a VR physics simulation.

We didn't encounter any problems this week.

This weekend, we plan to work on and complete the draft of our requirements document so we can send it to our client to review on Monday. We also plan to complete the technology review next week (all separately), but first we need to divvy up the parts of the project to focus on.

### *5.4.4 Fall Term - Week 6*

This week, our group created the work agreement. We also met up after Capstone class to divide up the topics for our technology review. I got assigned to Project Setup, and chose to separate my topics out into Languages, Asset Bundlers, and Continuous Integration.

We had some troubles coming up with three separate topics for everyone, since we have five people.

I plan to get to work setting up the repository and build tools some time in the next week.

#### *5.4.5 Fall Term - Week 7*

This week, I finished my Technology review and had it peer reviewed in class. I updated it to fix a few spelling and grammar mistakes that the peer reviewers pointed out.

We didn't encounter any problems this week, since pretty much everything we did this week was individual.

We plan to get into contact with our client next week to go over the documents we have been writing and get feedback before we submit the final requirements document to him to sign.

#### *5.4.6 Fall Term - Week 8*

This past week, my teammates and I did not really work on anything for our capstone project.

We didn't have any problems.

This coming week, our client (Alexis Menard) is coming to class to give a guest lecture about the technology we've been working with. While he is down in Corvallis, we are going to meet with him to discuss the requirements and our proposed technology stack. We haven't talked to him face to face for a while, so this will help make sure everyone is on the same page.

#### *5.4.7 Fall Term - Week 9*

This tuesday, our client Alexis Menard came down from Intel to give a guest lecture to our Capstone class about WebXR. He stayed afterward and we discussed our next steps for the project.

The only trouble we had was that one of our group members had already left for Thanksgiving break, so he wasn't able to be there in person for the client meeting.

We decided that we would each brainstorm 2 ideas for the physics simulation over Thanksgiving weekend, then get together early next week once everyone is back from vacation to write our design document.

#### *5.4.8 Winter Term - Week 1*

We met with our client on thursday and discussed the current state of the project and our next steps. We also created tasks to track our progress in GitHub Projects.

We didn't run into any problems this week.

We decided that we would make 4 different VR physics experiences, and have 4 team mates work on a separate experience. We will have the 5th person work on the interface and tying everything together.

#### *5.4.9 Winter Term - Week 2*

This week we met with our client again to talk about our progress and plan out the next week. We set up an agile system with 2 week sprints and planned out our first sprint.

No problems this week.

My tasks for the next two weeks are to set up the routing for our app, and to set up how to navigate between rooms in virtual reality.

#### *5.4.10 Winter Term - Week 3*

This week, I worked on the poster with my group and elevator pitch by myself. We also had our weekly meeting with our client, Alexis. We didn't have much to talk about, though, so we ended it short.

No problems

I am planning to meet with my group mate Evan to work on the routing and room navigation later tonight. We probably won't get it all done tonight, so we will work on it throughout next week and hope to get it done before the end of our first sprint on next Thursday.

#### *5.4.11 Winter Term - Week 4*

This week We completed our first sprint. In the sprint, I added functionality to switch different rooms in the vr app by clicking on links in html.

We had a discussion about whether some of the refactor that I added was the way we wanted to take the app, but after talking to the client, he liked what I had changed.

This weekend, I plan on researching how to integrate the webxr api into what we already have in the app

#### *5.4.12 Winter Term - Week 5*

We met with our client and he helped us work out the kinks of getting the magic window set up. We finished that, and are now focused on building an alpha version of our app.

No problems - we resolved a lot of them though.

We plan to get an alpha version of our simulations ready by the deadline, including some basic functionality like interactivity and the objects in each scene.

#### *5.4.13 Winter Term - Week 6*

This week we met with our client and updated him on our progress. We also worked on getting WebXR to render two frames when a phone is turned sideways so we can put it in a daydream or similar VR headset.

We encountered some problems when updating to the newest version of chrome dev build, but our client was able to help us out with those.

I plan to get a working demo of my planets simulation before the alpha deadline.

#### *5.4.14 Winter Term - Week 7*

This week we finished sprint 2, and set up our stories for sprint 3 as a group. Personally, I finished the alpha version of the planets demo.

I had an issue when merging master into my branch since all of our branches were pretty large. In the future, we should work on using smaller branches.

This coming week, we plan to finish up the alpha document and continue working on our stories for sprint three. For me, than includes adding user interaction to the planets demo.

#### *5.4.15 Winter Term - Week 8*

This week I worked on separating out the information about the project into a home screen, and then from there you can go to the home vr room. We also finished up the progress report.

I didnt have any problems

Next week I plan to work on my planets demo and make it more interactive, with the ability to add and remove planets from a system

#### *5.4.16 Winter Term - Week 9*

Last week, I moved the welcome screen away from the bottom of the VR page. This helped with the UX because being able to scroll on the VR page was annoying.

No problems

Next week, I plan to work on the statistics display (text above everything else that follows each planet when enabled) for my planets demo.

#### *5.4.17 Winter Term - Week 10*

This past week, I worked on the progress report video with my teammates. I also implemented a loader that will load all the necessary assets (textures, 3d object files, etc.) for the next scene before they are needed.

I had an issue with the loader, where it wouldn't wait for the assets to be loaded to start rendering the scene, but my teammate luckily caught it in the pull request review.

We still need finish piecing our parts of the progress report video together, and update our requirements document and get it signed off by Alexis, our client.

#### 5.4.18 Spring Term - Week 1

This week we went to Intel to work on the project with our client for the day. I got the planet textures and built my planet objects so that they look more realistic (with rings and textures etc.).

I ran into problems with the solar system model I was using. I found one online that had textures and an object file for the whole solar system, and I hoped I could extract each planet individually from that without too much trouble. As it turned out, it was a little harder than I had hoped, so I decided to manually create the planets from spheres in code.

This coming week I plan to finish up my demo with a way to move throughout the planets and see stats about them.

#### 5.4.19 Spring Term - Week 2

I got the planets to orbit in the same manner they do in our solar system. I also got the planet details to display next to each planet.

Ran into a small issue with displaying 2d text in the 3d environment, but I fixed it by using a canvas to render text to, then using that canvas as a texture for a sprite.

I plan to finish up the planets demo with a next and previous planet feature by Monday, the code freeze deadline. As a team, we also need to update our requirements document.

#### 5.4.20 Spring Term - Week 3

This week my team and I met to update the requirements document and design document. We sent them to our client and he approved them.

We didnt have any problems.

We plan to do a few bug fixes, such as being able to drag and drop objects without a headset.

#### 5.4.21 Spring Term - Week 4

This week I met with my group to finish up our poster.

There were no problems.

We plan to do a few bug fixes to our project before expo at the request of our client.

#### 5.4.22 Spring Term - Week 5

This past week, we didnt do very much. I did one bug fix that our client Alexis pointed out.

Didnt run into any problems

Dont have any plans to work on it, since the planets demo is pretty much done.

### *5.4.23 Spring Term - Week 6*

This week, we didnt do anything since our project is pretty much done.

We didnt have any problems.

We plan on prepping/setting up for expo next week (as well as attending!).

## 5.5 Evan

### *5.5.1 Fall Term - Week 3*

Progress:

Weve met with our clients including a team from Mexico that well likely be working with (I had a quiz on Thursday and had to stay back, but hopefully Ill be able to go to Hillsborough sometime during our capstone to see Intel and meet Alexis). We cleared up some of the disagreements we had about what the point of the project was. Weve scheduled our weekly TA meeting. Weve created our GitHub organization and repository. We each came up with problem statements and mine is wrong, but thats ok/expected. Weve setup our Discord server and everybodys active. Lastly, we found a few videos made by our client about the technology that well be using (WebXR).

Problems:

I was in class when our TA reached out to us for the weekly meeting. My group Discorded it and came up with a time and responded to the TA. But it was a time I couldnt do. I let them know after my class and we found a good time. Honestly, Im really happy that theyre excited (as am I). Ill just need to be assertive and quick on chat draw.

Plans:

We have a Google doc with a few different ideas for what the actual project will be. Its more of a tech demo than I thought it would be, which is great because Im very excited and interested in the technologies. What else do we need to do? 1. Organize our repo for the code + Tex docs 2. Continue researching WebXR/WebGL/Gamepad API, etc.

### *5.5.2 Fall Term - Week 4*

Progress:

We had our first TA meeting and wrote our problem statement today. I havent done much research this week into the technologies that well be using and whatnot. We also still dont know what project were going to build. Honestly, any one of them could be used to showcase WebXR. I would like to see us build a super cool VR application that somehow also has a non-VR option. I believe that thats the missing demo. Theres plenty of web-based VR experiments but I havent seen a VR experience that showed the same content as the main page. We probably wont do that I would guess, which is perfectly fine. It will be fun and useful either way.

Problems:

Nothing really this week. We had a miss communication about when we were going to meet up for the problem statement and Alex showed up at the Library on the wrong day. Poor guy, I hate wasted trips for myself and empathize for others.

Plans:

Since were going to be writing our requirements document soon, we should probably narrow down what project were going to do. I also need to commit my problem statement from before.

Midterms are this week.

#### *5.5.3 Fall Term - Week 5*

Progress

Not much to report here. We voted on the list of possible projects to demonstrate our WebVR with. Its looking like well do a physics simulation with objects and different physical parameters.

Problems

Midterms. It's been a very stressful week.

Plans

I cant wait for the tech review. Im learning Rust (when I should be doing other things. Thats why I forgot about the PPP blog) and I hope well need to use it in the final project. Especially if were working in physics, perhaps well need to port/modify an existing physics engine that compiles to Web Assembly.

#### *5.5.4 Fall Term - Week 6*

Progress

Weve been communicating more with Alexis. He setup a Google group that we hadnt been using until this week when we had the opportunity to send him our Requirement documents which he had a lot feedback for. We also now are syncing a google drive folder for all the content of our project and using our repository for all the latex documents. We also all finished our tech reviews. Mine was on performance mostly though I ended up adding lazy loading to that to meet the word count. I still have more options to consider for languages that compile to Web Assembly because I just heard that Go has WebAssembly support.

Problems

I wish we didnt have data in two places, that always ends poorly in my experience, but I think its the best we can do for now. I suppose its actually in three places because we have Overleaf, Google Drive, and GitHub.

Im really glad that Kirsten extended the due dates for the Tech Review because it would have been extremely difficult to do with my class load during midterm season.

Plans

Were hoping to meetup with Alexis again when he comes to talk about WebXR. As long as I dont have any quizzes that day, then I really want to be at the meeting and be able to listen to his goals for our project. My teammates gave me pretty good notes from the previous meeting, but I still want to hear what Alexis has to say.

#### *5.5.5 Fall Term - Week 7*

Progress

I just submitted my final tech review. Thats about it. I havent done much else. Midterms are over but Im still doing lots of assignments for my classes.

Problems

Alex brought up in our TA meeting that he thinks we need to communicate more with Alexis because it has felt like this is our project and its his project as well.

Plans

Wed like to play around with the stuff we talked about in our tech reviews since we have some time before our next assignment is due. Also our meeting is still on for when Alexis comes to Corvallis though Alex might not be able to make it.

#### *5.5.6 Fall Term - Week 8*

Progress

We didnt really do anything this week.

Problems

Were still figuring out how to do our meeting with Alexis next Wednesday. Alex looks like hell need to remote in to the meeting.

Plans

We need to work on our design document which seems pretty straight forward given our tech reviews and whatnot.

#### *5.5.7 Fall Term - Week 9*

Progress

We had a great meeting with Alexis talking more about his expectations and goals for the project. I think we know how to do the design document now. I uploaded my notes (or at least what I can remember) from our meeting to our drive folder.

Problems

Alex went home early for the holiday and asked us to video the lecture and skype him in for our meeting. None of us committed to doing that. I tried to video the lecture, but my phone ran out of space. Tim recorded the audio though. And then we all forgot to setup Discord for our meeting. He was (is) very disappointed.

Plans

Were going to pick the exact physics experiments that were going to do and start the design document, hopefully immediately. We want to know what experiments we will do so that we can figure out how we are going to acquire them.

#### 5.5.8 Winter Term - Week 1

Progress

We had a Google Hangouts meeting with our Client this past Thursday. Alexis gave us more resources to read while were working, and it felt like we officially kicked it off.

Problems

Nothing really.

Plans

Were transferring our previous assignments (requirements, scheduling, etc.) into GitHub which well be using for this term. All the talk is going to start being do.

#### 5.5.9 Winter Term - Week 2

Progress

I did a bunch of programming this week. Ive mostly been recreating a demo that our client Alexis made with the goal of getting VR magic window support. Im not quite there. I also did some UI prototyping. Not that it needs to be how the project looks in the end, but it was fun to hack out some CSS.

Problems

We had to stop using Typescript because we couldnt find typing files for the WebXR API and would have to write them ourselves to make the compiler happy. I usually prefer a native programming environment anyway without all the building because I have an older computer and it slows my development cycle. I got around this by working in a computer lab on campus and I think Im going to need to do more of that this term.

Plans

We assigned issues for our first sprint of work. Im supposed to do some CSS and teleport functionality. The CSS seems like just migrating my personal coding to the main repo, but Im concerned that were not ready for the teleportation feature yet.

#### 5.5.10 Winter Term - Week 3

Progress

I did a bunch of work but at the same time, it wasnt a bunch of work? I got magic window working by following Alexis code. I did this however in my own, Parcel, and all the other automation free branch. It was useful in that it let me figure out a little bit of how our code will work but it needed to be worked back into the rest of the project. I did that for the page styling that I did (I actually spent a fair amount of time on that part) but I didnt do that for the magic window (yet?). I also had a sample enter VR button going that integrated with my personal branch.

I also met with Brooks and we talked about how were going to do the room changing functionality. We have an idea of how thats going to work. I talked a little bit with Alex about how to get magic window working.

Problems

Doing that programming was a joy. I am very familiar with doing web development, but Ive always shied away from build tools. Thats included Typescript, React JSX, bundlers, etc. I find that I can get away with plain JavaScript (well, ES 6) and it integrates super well with Caddy and Chrome Canarys inspecting tools. Ive not been very motivated to work on our projects main stuff when I know that, due to using source maps, I cant inspect the code as I would like. Or that its going to take 3  5 seconds before Ill see the changes in the browser (unless I wait longer to find that I need to refresh the page, or restart Parcel.)

During the time that I was talking with Alex I learned that he doesnt know much about Promises. Theyre a sizable feature in the WebXR specification so I think it would be helpful for him to know how to use the whole syntactic sugar of async, await, etc. You can work using the .then and .catch functions but it isnt quite as nice. Im also ok with not using those new language features if needs be, however if were still shooting to display the WebXR API in a way that other people would use it in their projects, then I think we should use them.

Plans

Im hoping to do some 3d modelling soon. I learned some Blender 3D awhile back and it turns out that theres an exporter from blender to Three.jss preferred format. I modified the GLTF loader to support being loaded as a module as part of the programming I did. I want to make sure that I have some real assets to load so that I can start on my experience because I want it to serve as a model and discussion point for our team.

#### *5.5.11 Winter Term - Week 4*

Progress

I havent done much this week. I did a bit of 3d modelling for my section of the project. I think that 3d modeling some of our own assets will end up being cheaper and faster than finding them.

Problems

Nothing to report. If were going to switch to programming directly against the api then my phone will likely be unusable for our project. With the polyfill, I was able to get some of the stuff working.

Plans

Were doing research so that we know what we need to do next. My team thinks that this will likely be that last sprint before we move into personal implementation of our respective experiments, however I think it would be great to start blocking out at least one of them so that we can get an idea of what each one will look like. I think that would make individual implementation much easier.

#### 5.5.12 Winter Term - Week 5

Progress

I did some 3d modelling this week. I now have: a draft pendulum, a table, the start of the room that will hold all of the objects, the lunar floor, a door with a doorframe, and thats it. None of these are textured, though. I suspect Ill just end up giving each a different color.

Problems

Its always been a struggle to focus on getting the project fully up and running and not polishing the UI which will probably change or deciding the best way of adding event listeners to the links in a document when the current way works with only slight annoyance.

Plans

So much I still need three planet surface floors. I need to assemble the whole scene. I need to mock out the user input and make some interactions. Luckily, most of those interactions will be slight modifications of each other. Teleport, is just drag and drop that jumps to the location rather than moving the object, starting the pendulum is like drag and drop but its just rotation along one axis, etc. Once those work, I should probably get the physics equations hooked up to the objects. Lastly there should probably be a way of seeing the properties of the pendulum: screen with digits / velocity + acceleration arrows / current length of the pendulum / etc.

Then theres tones of polishing that we can do after the alpha: textures, performance monitoring and adaptation, arrows to display

#### 5.5.13 Winter Term - Week 6

Progress

I havent colored my models but I have put them all into the room(s) that theyll probably stay in. Ive got them loading into the browser and rendering. I didnt really do anything else.

Problems

Plans

My plans havent really changed: Still 3 floors, still need to do all the input but now that things are loaded I have objects to interact with.

#### 5.5.14 Winter Term - Week 7

Progress

Im still troubleshooting the offset issue and I have a hypothesis, but honestly I havent worked on it much.

Problems

I found out that the way Ive been doing things is not the future of the spec. Secondly, I rebased my code onto master and now most of the WebXR stuff is broken. Im not really sure whats going on. My hypothesis about the offset is that the scene root object is offset by the two eye rendering but I havent tested it yet. I had been using canary but canary is broken and I switched to dev but my rebase seems to have broken more stuff. Hopefully I can find something in my git history that works and then figure out whats going on.

Plans

I signed myself up for lots of issues  the ones that sound interesting. Im interested in dom to texture and Ill need it eventually for the UIs in my pendulum experience but Im not ready for that yet.

#### *5.5.15 Winter Term - Week 8*

Progress

Im still troubleshooting the offset issue and I have a hypothesis, but honestly I havent worked on it much.

Problems

I found out that the way Ive been doing things is not the future of the spec. Secondly, I rebased my code onto master and now most of the WebXR stuff is broken. Im not really sure whats going on. My hypothesis about the offset is that the scene root object is offset by the two eye rendering but I havent tested it yet. I had been using canary but canary is broken and I switched to dev but my rebase seems to have broken more stuff. Hopefully I can find something in my git history that works and then figure out whats going on.

Plans

I signed myself up for lots of issues  the ones that sound interesting. Im interested in dom to texture and Ill need it eventually for the UIs in my pendulum experience but Im not ready for that yet.

#### *5.5.16 Winter Term - Week 9*

Progress

I gave up on the offset and just started working on drag and drop. In doing so I got a better idea of what is wrong with the raycasting. Turns out that I was setting the proper position in the world matrix but then the scene would move and put it right back where it started. I was also able to get a drag and drop system together and work it into two objects in the pendulum scene. The pendulum can be picked up and moved around. The swing on the pendulum can also be rotated which is how youll eventually put energy into the pendulum. I also made snap points so that the pendulum snaps onto the table. I think Ill be able to use this to decide which value to use for gravity (each table is in a different room and each room represents a different planet). Since Im submitting this later, I might as well add the stuff I did during finals week. I added the equations to the pendulum and made it so that it gets the gravity value based on what table the pendulum is set down on. I havent fully tested this because I cant get to any of the other tables (teleport sends me backwards instead of forwards).

Problems

Im not sure where our project is going to end up. I intend to work on it during spring break to hopefully get the pendulum scene fairly functional.

Plans

Finish integrating the interaction stuff. 3d model a couple more world floors. Get teleport working.

#### *5.5.17 Spring Term - Week 1*

Progress

I switched us back to using originOffset which Chrome had implemented before and Id tried to use but hadnt been able to get it working. One reason for switching to originOffset is that theres a bug I want to squash after the code freeze where when you switch from being in VR to magic window You start facing a different direction. Im hoping to copy the direction you were facing from one to the other and vice versa.

I finished unifying the Interactions (mostly). This meant implementing Alexs triggermesh using my own interaction system. Triggermesh had a few problems largely in that you couldnt have one trigger mesh be a child of another trigger mesh which wouldnt have worked for my pendulums where the pendulum swing can be interacted with and the pendulum base can be moved around. I also got most of the controller, laser, and cursor rendering back after I removed them to unify the interaction implementations.

We went up to Intel and discussed our project with Alexis as well as tried it out using a Vive. He changed some of what I was working on as well as everyone else which I think is ok. I was pretty close to finished with the pendulum functionality as Id planned but the way he wants it isnt too different and much of the code will carry over.

Problems

Ive struggled to get my code into master. Currently my branch has all the pendulum stuff and interactions but some issue (which I cant figure out and which my groupmates first suggestion hasnt fixed) is causing lint to fail and linting is a prereq for merging into master. Honestly, I think its mostly gotten in our way rather than improving our code. Maybe slightly but if I had a nickel for every time Ive just turned an error into a warning because I disagreed with the linter then Id be a rich person.

Plans

I asked the NMC department if I can ask someone to borrow the VR lab mostly so that we can try using two controllers again. Alexis noticed a bug while he was using the project which I had anticipated and hope is fixed now that the interactions are unified but I hadnt finished while at Intel.

I just need to reorganize the pendulum scene and then Im free! Instead of having the different rooms indicate planets, Alexis wants the planet surfaces to swap out from underneath you. He also wants the quiz part so Ill need to use blender to create text objects for that.

#### 5.5.18 Spring Term - Week 2

Progress

I have the pendulum quiz working and moved from individual rooms to swapping out the floors as Alexis had requested. Theres plenty of polish that I could do but I really need to do assignments for other classes so Im not sure how far Ill get. I also did some simple object throwing mechanics but only implemented it for the ball. It should be an easy copy and paste or simple refactor for the falling object people to make anything they have throwable. Lastly, I think it was sometime this week, Alex and I got together to fix the issues that Id introduced while merging the interaction things together. I dont have a daydream so I could only test the gaze interactions and it was good because there were multiple things that we needed to fix.

Problems

Couldnt figure out what the problem was with Circle CI and other team members have had building errors so I just used admin rights to force the merge into master. It felt good to actually have what Ive been working on available to my teammates.

Plans

Im going to finish my other assignments and then I hope to fix some bugs, maybe write up a little something about how to use the pendulums. I should also probably clean up the home room with the door models, and remove the testing boxes.

#### 5.5.19 Spring Term - Week 3

Progress

Our code freeze was a little hectic but we did a lot and Im very happy with where were at now. We also did our documentation and that is in aswell.

Problems

Id tried to access the VR labs to make sure that none of our changes broke two controller input since we tested it with our client at Intel last week. Unfortunately, the VR lab didnt open even though I was pretty sure I was there during their open lab time. Anyway, this meant that the next time we tried it on the Vives was right before the code freeze for like 5 min and it didnt work on the Vives. Looking through the Chrome logs there looks to be a change to how Chrome handles external sessions but we didnt have enough time in the lab (and depending on what the underlying issue was: time before the code freeze) to get it fixed.

Plans

Nothing. I went to my required session, this morning (though I havent done the right up) and Im hoping to catch up on the backlog from my other classes. After I catch up then I might try and do more polishing. Theres still lots of improvements that we could do.

#### *5.5.20 Spring Term - Week 4*

Progress

We slightly updated our poster and added a picture.

Problems

We may need to constrain to using Chrome Dev again because something in Canary is causing issues again. I havent looked into it so I dont know how big of a fix it would be, but my groupmates seem to know whats going on.

Plans

I just need to add textures to the pendulum room. The rest of the pendulum functionality seems to have been good, but its always looked bland and our client requested that it be textured.

#### *5.5.21 Spring Term - Week 5*

Progress

I added a few textures: wood textures to the table, and pendulum bases, as well as a moon texture to the lunar floor.

Problems

Unfortunately, the UV unwrapping that I did doesnt seem to have been exported so the textures dont line up with the objects (as far as I can tell) so the earth one with the city is all messed up.

Plans

Either fix the UVs or model the city as lots of different objects so that I can texture them individually.

#### *5.5.22 Spring Term - Week 6*

Progress

I added textures to the pendulums which took learning how blender works. I also found out how to export lights so I tuned the lighting a little bit to make it look better. Alexis also mentioned people getting lost from the platform so I disable the ability to teleport out around the planet surfaces. You can still teleport around the platform, though. I also added a little plane with some instructions behind the table with the pendulums.

Problems

Something about parenting wasnt making the GLB exporter happy. In blender the objects were right where I wanted them and then in the GLB they were offset so I put the pendulum swings at the center and that seemed to fix it. I dont know why they need to be at the center now because they didnt need it before but, ya.

Plans

Do expo.

## PROJECT BACKGROUND

WebXR is an exciting new API that will let developers bring virtual and augmented reality experiences into their websites. This means that users won't have to download a mobile app to see the VR or AR content. It can also seamlessly deploy across different operating system and browsers without compromising capabilities.

The WebXR specification is still in development, so there are not a lot of resources out there for developers to go off of for making new AR and VR apps for the browser. One of the primary goals was to build a working example project that future developers could reference when building their applications.

That goal left us with a wide variety of possibilities for what the project could look like. Our next goal was to make an open source project that could help solve a real-world problem. Since VR simulations have proven to be so beneficial to learning, we wanted to make a way for high school students to experience physics labs without the need for expensive lab equipment. Another goal of the project is to showcase an application that can be impossible to preform without a VR device while progressively enhancing the experience to users.

API
- Application Programming Interface

VR
- Virtual Reality

XR
- **V**irtual/**A**ugmented Reality

Oregon State University

# WEBXR PHYSICS

In-browser virtual reality physics simulation with a focus on an interactive educational experience

## EXPERIENCES

### PLANETARY DIORAMA

A simulation of the solar system, with info about each planet as you visit it. The user can navigate from planet to planet and look around in virtual reality.

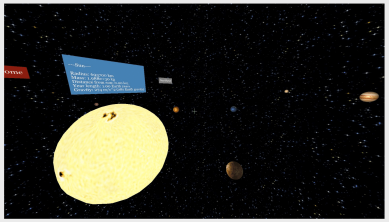


Figure 1. A view of the planetary diorama.

### LASER REFLECTION

This experiment demonstrates how light reflects off mirrors. Users can create mirrors that reflect a laser with the objective of hitting a target. The mirrors can also be rotated to change the direction of the reflecting laser.

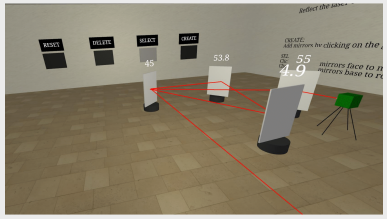



Figure 3. Using mirrors to reflect the laser into the goal in the laser reflection experience.

### KINEMATIC SANDBOX

Manipulate and toss various objects around in an open area environment. Observe how different levels of gravity affects these objects. Visualize the velocity vectors of objects in motion throughout the scene.

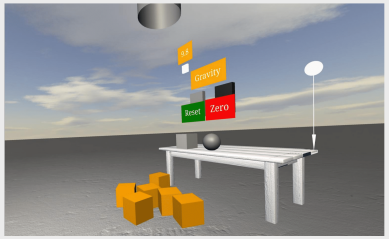



Figure 2. Tossing around boxes in the kinematic sandbox.

### PENDULUM PERIOD PUZZLE

Simple pendulums only have two factors which determine their period. Most physics labs change the length of the pendulum but ours changes the length and the value for local gravity by virtually taking the pendulum to different planets. The experimenter can also put themselves in a quiz where they use their previous tests on each planet to guess which planet they're on from a closed room.

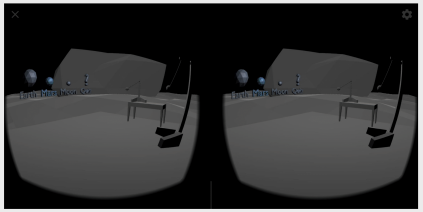


Figure 4. Swinging pendulums on the moon in the pendulum experience.

## PROJECT DESCRIPTION

WebXR Physics is a virtual reality physics simulation built into a web browser using the latest web virtual reality technology, WebXR, with the intent of showcasing WebXR's potential use cases. In this case, we created a virtual physics lab that can be accessed straight from a web browser and without additional downloads.

Our project incorporated the major facets of the VR and controller portions of the WebXR API and includes physically accurate experiment stations ready to be explored. This makes it a good reference for physics students without fancy equipment and engineers considering incorporating WebXR into their own products.

WebXR Device API can fully support FireFox, Chrome, and Edge.


Figure 5. The home room, a main hub with doors leading to all the different experiences.

## TEAM MEMBERS & SPONSOR

Jonathan Jones (jonesjon@oregonstate.edu)
Tim Forsyth (forsytht@oregonstate.edu)
Brandon Mei (meib@oregonstate.edu)
Brooks Mikkelsen (mikkelsb@oregonstate.edu)
Evan Brass (brassev@oregonstate.edu)

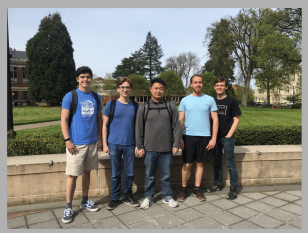


Figure 6. From left to right: Jonathan Jones, Tim Forsyth, Brandon Mei, Brooks Mikkelsen, Evan Brass

Sponsor: Alexis Menard (Intel)

# 7 Project Documentation

## 7.1 Structure

Our is entirely run on the end user's web browser. Therefore, we had to use JavaScript to build the whole thing, including interacting with the VR devices and rendering 3D objects. To render the scenes/experiences, we used Three.js, a popular javascript graphics library.

We built a parent class that has all of the shared code that is needed for the scenes. We then made children classes for each scene, which contained code unique to that scene. We used a custom-made router to tie all of these scenes together, and allow the user to navigate throughout the app using URLs.

The WebXR API is what allowed us to bring VR to life on this website. The API itself serves to collect input from connected VR devices and make it available to anyone using the API. Provided with this input, we were able to translate the movements of the headsets and the controllers into 3D matrix transformations that we could apply to our 3D scenes.

Use our readme (below) to install, set up peripherals, and run our project. There is also a small guide to each of the experiences.

# Web XR Physics

The WebXR API is a brand new spec that gives developers the ability to create VR and AR experiences that run natively in the browser. Our physics demo uses this API to create an immersive experience that students can use to learn physics fundamentals.

This is an OSU Senior Capstone project for group 47. The goal is to demonstrate the abilities of the WebXR API by creating a sample project that can be referenced by others later. The project is sponsored by Intel.

## Building and running the project

- Ensure the latest versions of Node.js and Node Package Manager (npm) are installed on your machine, and are accessible from the terminal.
- After cloning the project, open up the root project repo and install the dependencies using

```
$ npm i
```

- Start the development server using `$ npm run dev`
- The project will be running on http://localhost:1234. The development server will watch for changes in the source files and automatically reload the page after compiling them.

## Working with mobile devices (Daydream)

Install Chrome Dev. We tested using Chrome 75.0.3759.4, but later versions should work as well.

Open the inspect tab: chrome://inspect
Configure the port forwarding by selecing the Port forwarding... button. Add the following ports (such as the default 8080): - localhost:1234 for the http that forwards to localhost:1234 - localhost:64320 which is how Parcel does it's live updating stuff

Be sure to check the Enable port forwarding box at the bottom of the Port forwarding... modal.

In order for the port forwarding to work on your device, there are some settings you need to enable.

Please follow the instructions at this site to enable USB debugging on your android device.

Now, connect an Android device with Daydream support via a USB cable and install and open Chrome Dev on your device. There are a couple flags that need to be enabled. Go to chrome://flags and ensure the following flags are enabled: - WebXR Device API - WebXR Hit Test

Go to localhost:1234. Click the 'check it out' button and you're in! There should be an 'EnterVR' button in the top left of the screen. Click it to enter immersive VR mode.

## Working with HMDs (VIVE/Oculus)

Currrently, there are some issues with using/setting up HMDs. We met with our client in person on April 5th and he was able to get it working on his VIVE. However, we are unable to manage to get it working with the steps he provided us (down below). At the moment it appears to be an issue with the WebXR API itself. For now, we recommend only testing using a Daydream device, although feel free to attempt to get it working using the instructions given.

In order to use an HMD (head mounted display), such as a VIVE or Oculus Rift, you first need to download Steam and SteamVR. Assuming you have connected your device, start up SteamVR and then open localhost:1234 in your Chrome Dev browser. Ensure that the following flags are enabled by going to chrome://flags and enabling: - WebXR Device API - WebXR Hit Test - WebXR orientation sensor device - OpenVR hardware support

At localhost:1234 click 'check it out' to load the home room. There should be an 'EnterVR' button in the top left of the screen. Click it to enter immersive VR mode.

## Scene not loading?

For unknown reasons, the scene you are trying to enter may not load the first time you try to enter it. If you observe a blank screen when entering a room, all you have to do is back navigate to the previous room/page then try again.

# How to use the experiments:

## Movement

To move around a scene (except the planets scene) in magic window sessions, drag the small circle in the bottom left portion of the screen. In immersive sessions, you can move about the scene by selecting a point on the floor to teleport instantaneously to that spot.

## The Home Scene

This room can be accessed by selecting `Check it out!` on the landing page. In this room you will find four doors. Each are labled with their respective scene. Selecting a door (either through touch or VR controller) will navigate you to the scene behind the door.

## The Kinematic Sandbox Scene

Within the kinematic sandbox scene you should be able to see a table with a cube and a sphere on it. Clicking either of these will generate a new object that you can interact with that falls from a chute in the ceiling. The newly created objects can be grabbed with your laser, thrown around and used to interact with the other objects in the scene. Beyond the table is a pair of two buttons that enable and disable earth gravity. Selecting the ON button sets the gravity in the scene to 9.8 m/s while selecting the OFF button sets the gravity in the scene to 0.0 m/s.

## The Pendulum Scene

Once you enter the pendulum scene, you should see two pendulums on a table and a list of planet's on your left. The lighter blue planet icon is the planet that you are currently on. You can drag the pendulum's swings to start them swinging. If you want to stop the pendulums swinging you can either switch planets or pickup the pendulum and then place it back on the table.

Once you've gained a feeling for how the pendulums swing on each planet, hit the quiz icon which will put a box around you so that you can't see what planet you're on. Then you can play with the pendulums and when you're ready, click the icon that you think is the planet you are on. The correct planet's icon will light up green and if you're guess was incorrect then it will be shown as red.

Once you're ready to go back to the other experiments, there should be a door on the opposite side of the platform that will take you back to the home room.

## The Planets Scene

Upon entering the solar system simulation, you should see the sun, along with some information about it. You can explore the planets in the solar system either by clicking the "Next Planet" and "Previous Planet" buttons, or by clicking on another planet.

You can exit the simulation by returning to the Sun and clicking "Exit to Home".

## The Lasers Scene

Inside the laser room, there should be a wall with 4 large buttons on it. 3 of the buttons set the controls to a specific mode: - CREATE: While in creation mode, you are not able to teleport, but will instead see an outline of a mirror where it would be placed if you selected the ground. Select the ground, and a mirror will be created in it's place. You can create as many mirrors as you would like while in this mode.

- SELECT: Selection mode allows the user to move mirrors via dragging them (holding the select button). Release the select button to place the mirror. Mirrors can also be rotated by dragging their base left or right.
- DELETE: Any mirror clicked on is this mode is deleted instantly, so be careful!

The final button, RESET, resets the entire scene, removing any placed mirrors and repositioning the goal.

The goal is the small box randomly placed in the scene. You're mission is to place and rotate mirrors in such a way to reflect the laser into the white cylinder on the goal.

Finally, to exit the room, simply click on the door and you'll be transported back to the home room.

# 8 Recommended Technical Resources for Learning More

The following websites are helpful in understanding our project (listed in order of helpfulness):

1) Our GitHub repo: https://github.com/Group-47-Capstone-2019/Immersive-Web-VR-AR
2) WebVR Explained Mini Series - Alexis Menard: https://software.intel.com/en-us/vr/webvr-series
3) Immersive Web Article - Alexis Menard: https://medium.com/@darktears/vr-concepts-and-the-immersive-web-dd2604e3e338
4) WebXR Device API: https://www.w3.org/TR/webxr/
5) Three.js docs: https://threejs.org/docs/
6) Cannon.js docs: https://schteppe.github.io/cannon.js/docs/

There are not really any books or people on campus we used as resources in making this project, although our client Alexis Menard gave us a lot of helpful pointers throughout the process.

# 9 Conclusions and Reflections

## 9.1 Jonathan

This project was an enormous learning experience. I had the chance to learn more about VR on the web, web graphics, development practices and team management. I had some graphics experience to begin with but I had never worked on a graphics project of this magnitude before. I'm very comfortable with the Three.js API, and the WebXR API now. What I learned most was how to manage a team and the tasks associated with a project. I took on the role of setting up sprints in a SCRUM format. Generally, things went well. Our project was well organized and we were able to delegate tasks pretty efficiently. A successful team has great communication and we were lucky to have that. If I could do this project over, I would have studied our tech stack more before starting it. Some tasks would have been completed much quicker and more smoothly had we known about some of the particulars of the tech stack beforehand.

## 9.2 Brandon

Overall this project was a great learning experience for me. I learned a lot working with virtual reality through the web. At the beginning, I had little or no prior knowledge of graphics and virtual reality. I learned how to use three.js and cannon.js libraries for our project. I learned enough to be comfortable with the three.js and cannon.js APIs while using WebXR API to provide an immersive web experience.

As for non-technical information, I learned a lot about dealing with unstable APIs and spectrum of hardware, applications, and techniques used for Virtual Reality, and other related technologies.

Working on this project has taught me how to manage a team and planning a schedule of ahead of time. It makes sure that our project will stay on track and fix any issue before we moved on.

Our project management we used an agile development process with a SCRUM workflow. We had a weekly meetings with our client to give updates and issue we had. I learned that splitting up the work can be a challenge without overlapping others people code. We used GitHub projects to manage and track our development progress.

As for working as teams, I learned that constant communication and feedback was the key of our success to the project.

Overall, if I could do this all over again, I would learned all these technologies like cannon.js, three.js, and WebXR API ahead of time so we can just focus on building our experience. I would also like to see our group to work on one big VR experience, so we can mostly contribute on the same experience.

### 9.3 Tim

I had little to no experience in virtual reality and graphics programming prior to this project, so it provided an opportunity to learn more about these topics. I learned a lot about three.js and would say I'm competent enough to be able to list it as a new skill. I also gained experience using a developmental API while using the WebXR API.

In terms of non-technical information, I learned a bit about working with a client. I had never had any real experience with a client, so working with Alexis gave me an opportunity to learn how to properly communicate and update the client on your progress.

This project also taught me the importance of planning ahead and keeping to a schedule. We ended up making a huge push near the end to ensure everything was finished and ready for the final deployment.

As far as project management goes, I gained experience using a combine board as we used GitHub projects to track our development progress.

We used an agile development process with biweekly scrum meetings. I learned that splitting up work evenly can be harder than it seems. It's also hard to predict how long it'll take to complete some task. We had moments were sprint tasks had to be carried over to the next sprint due to it taking longer than we had initially estimated.

Overall, if I had to redo this project, I would first and foremost study more about the technologies we were using, like three.js and the WebXR API as a lot of time was spent combing through the documentation trying to figure out how it worked. Secondly, I would try to create a more concrete plan and schedule way ahead of time. If we were able to visualize a development path with tasks laid ahead of us, I feel like it would have been easier to create the sprints and assign tasks accordingly. Finally, I would have chosen a different VR experience. We were a bit ambitious in choosing to do something that essentially split us up over 4 projects. I believe we could have produced one amazing experience if we had instead worked on a simpler project that we could put more time into polishing and adding fine details to.

### 9.4 Brooks

I learned a lot about working with graphics programming and virtual reality, especially where it intersects with web development. We learned how to use three.js, which is one of the most popular 3D graphics libraries for the web.

On the less technical side, I learned a lot about the pitfalls of working with incomplete/work-in-progress APIs. We had a lot of trouble keeping our project up-to-date with the WebXR API as it progressed, since there were a lot of breaking changes over the course of the year.

I also learned about working on projects - specifically about time management on large projects. We spent the last few days working very hard on the project, since we had taken a pretty large break before that. In the future, I'll make sure to spread out my work more evenly.

We tried to use an Agile development process so that we could have everyone working on the project at the same time. While this was definitely the right idea, we had trouble splitting up the code so that we didn't step on each others' toes.

The same advice here applies to working on teams in general: splitting up the work properly is very important to make sure everyone has something to do.

If I could do it all over, I would have pushed our group to work on one single VR experience, rather than 4, so that it would be more polished. It seems like we had 4 pretty good ones, where we could have had 1 really complete and fun-to-use one.

### 9.5 Evan

I didn't know much about the WebXR api or graphics programming in general. I'd copy-pasted the usual spinning cube using WebGL before but that was it. I also wasn't familiar with much of our building / deploying tools until we used them for the first time.

Much of my learning came from programming and engineering as a group. We accomplished a lot together and it required discussing what we'd done and needed to do to make sure we didn't double up on segments. While it was important to come up with a good design, it was as important to be able to explain that design clearly and convincingly so that our whole team wanted and was able to integrate with our work.

Projects require good leadership especially in software. Communicating, coordinating, and keeping on task is a full time job of its own. There's so many non-programming things to be done and it can be discouraging to spend so much time doing them. Every document can feel like the enemy of a feature. Having a good view of those tasks and their importance was something I had to learn.

If I did it over again, I would spend less time jumping into the writing assignments and more time finding Alexis' vision for the project. The assignments would have been easier if they'd flowed from the project instead of the project being derived from the written assignments. I think using Latex and the complications therein contributed to our primary focus on the documents and secondary focus on the project (at least for the first term or so).