ESCUELA DE INGENIERÍA DE FUENLABRADA

INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CONTROL CON MANOS EN REALIDAD EXTENDIDA USANDO TECNOLOGÍA VUE.JS

Autor : Samuel Cobos Correa

Tutor : Dr. Jesús María González Barahona

Curso académico 2023/2024

Dedicado a

mi familia / mi pareja / mis amigos

II

Agradecimientos

Después de un largo tiempo, el camino llega a su final.

Estaré eternamente agradecido a mi familia, por siempre estar conmigo, desde el principio hasta el final.

A mi pareja, por apoyarme en los momentos difíciles, haciendo de mi una persona mas fuerte y confiada para lograr todas mis metas y sueños.

Finalmente, a todos mis amigos, todos los amigos del Grado que han ido pasando a lo largo de los años, y a mis compañeros de trabajo, por enseñarme tanto y apoyarme cuando lo necesitaba.

Resumen

Este proyecto se ha centrado en la creación de un detector de poses que permite guardar y gestionar diferentes gestos realizados con la mano, mostrando en tiempo real las poses detectadas y las guardadas, a través del uso de WebXR Hand. Además, se ha desarrollado un panel de control en un entorno virtual, donde los usuarios pueden coger objetos, moverlos y pulsar botones, que a su vez generan nuevas interacciones en la escena. También, ha sido diseñado para explorar la integración de Vue.js, un framework de JavaScript para crear interfaces de usuario interactivas, con A-Frame, una herramienta de código abierto para construir experiencias de realidad virtual y aumentada directamente en el navegador.

La combinación de Vue.js y A-Frame ofrece una manera innovadora de desarrollar entornos virtuales interactivos y accesibles. Vue.js proporciona una estructura clara y reactiva, ideal para el desarrollo de interfaces dinámicas, mientras que A-Frame facilita la creación de entornos 3D inmersivos. Esta integración permite a los desarrolladores familiarizados con frameworks modernos, como Angular o React, utilizar sus habilidades para crear aplicaciones VR y AR sin necesidad de aprender herramientas completamente nuevas.

Este proyecto demuestra hasta dónde puede llegar el manejo de las manos en realidad virtual y aumentada. También muestra las nuevas posibilidades que ofrecen estas tecnologías cuando se combinan con frameworks modernos de desarrollo web, proporcionando una plataforma robusta para la creación de aplicaciones inmersivas y accesibles directamente desde el navegador.

Summary

This project has focused on the creation of a pose detector that allows saving and managing different gestures made with the hand, showing in real-time the detected and saved poses through the use of WebXR Hand. Additionally, a control panel has been developed in a virtual environment, where users can grab objects, move them, and press buttons, which in turn generate new interactions in the scene. It has also been designed to explore the integration of Vue.js, a JavaScript framework for creating interactive user interfaces, with A-Frame, an open-source tool for building virtual and augmented reality experiences directly in the browser.

The combination of Vue.js and A-Frame offers an innovative way to develop interactive and accessible virtual environments. Vue.js provides a clear and reactive structure, ideal for developing dynamic interfaces, while A-Frame facilitates the creation of immersive 3D environments. This integration allows developers familiar with modern frameworks, such as Angular or React, to use their skills to create VR and AR applications without needing to learn entirely new tools.

This project demonstrates the extent to which hand tracking can go in virtual and augmented reality. It also shows the new possibilities these technologies offer when combined with modern web development frameworks, providing a robust platform for creating immersive and accessible applications directly from the browser.

Índice general

1. Introducción **1**

1.1. Contexto 2

1.2. Objetivos 3

1.2.1. Objetivo general 3

1.2.2. Objetivos específicos 3

1.3. Estructura de la memoria 4

2. Tecnologías utilizadas **5**

2.1. Tecnologías principales 5

2.1.1. HTML5 5

2.1.2. A-Frame 7

2.1.3. WebXR 9

2.1.4. Vue JS 10

2.1.5. JavaScript 12

2.1.6. Three.js 13

2.2. Tecnologías auxiliares 14

2.2.1. Visual Studio Code 14

2.2.2. GitHub 15

2.2.3. LaTeX 16

3. Desarrollo del proyecto **17**

3.1. Sprint 0 17

3.2. Sprint 1 18

3.2.1. Objetivo principal 18

3.2.2. Implementación del desarrollo . . . 18
3.2.3. Resultados obtenidos . . . 20
3.2.4. Planificación . . . 20
3.3. Sprint 2 . . . 20
3.3.1. Objetivo principal . . . 20
3.3.2. Implementación del desarrollo . . . 20
3.3.3. Resultados obtenidos . . . 21
3.3.4. Planificación . . . 22
3.4. Sprint 3 . . . 22
3.4.1. Objetivo principal . . . 22
3.4.2. Implementación del desarrollo . . . 22
3.4.3. Resultados obtenidos . . . 25
3.4.4. Planificación . . . 25
3.5. Sprint 4 . . . 25
3.5.1. Objetivo principal . . . 26
3.5.2. Implementación del desarrollo . . . 26
3.5.3. Resultados obtenidos . . . 28
3.5.4. Planificación . . . 30
3.6. Sprint Final . . . 30
3.6.1. Objetivo principal . . . 30
3.6.2. Implementación del desarrollo . . . 30
3.6.3. Resultados obtenidos . . . 31

4. Resultados **33**
4.1. Descripción funcional . . . 33
4.2. Manual de usuario . . . 35
4.2.1. Panel de Control . . . 35
4.2.2. Detector de poses . . . 36
4.3. Descripción técnica e implementación . . . 37
4.3.1. Vue.js y A-Frame . . . 37
4.3.2. HandTracking y control de manos . . . 40

4.3.3. Implementación 43
4.3.4. Creación de nuevos componentes 43
4.3.5. Funcionalidades de A-Frame 45

5. Conclusiones **47**
5.1. Aplicación de lo aprendido 48
5.2. Lecciones aprendidas 49
5.3. Trabajos futuros 50

Bibliografía **51**

Índice de figuras

2.1. Imagen de un escenario básico con A-Frame 8
2.2. Imagen de los diferentes frameworks 13
2.3. Imagen en 3D realizada con Three.js 14
2.4. Diagrama de ramas en GitHub 15

3.1. Escenario básico con A-Frame 21
3.2. Estructura principal del código 23
3.3. Escenario implementado en Vue.js y A-Frame 25
3.4. Escenario Editor 28
3.5. Escenario Panel de Control 29
3.6. Escenario Panel de Control después presionar un botón 29
3.7. Escenario final sin poses guardadas 31
3.8. Escenario final sin poses guardadas 32

4.1. Moviendo el panel de control con las manos 34
4.2. Pulsando el botón para guardar pose 35
4.3. Estructura de componentes 38
4.4. Estructura de entidades 38
4.5. Ejemplo del componente pulsador en la escena 39
4.6. Guardado de una pose 41
4.7. Definición de los puntos de la mano de XRHand 42
4.8. Guardado y detección de poses 43
4.9. Correr aplicación en local 43

Capítulo 1

Introducción

Este trabajo de fin de grado se centra en el desarrollo de una aplicación que permita el manejo y control de manos en entornos de realidad virtual y aumentada. La capacidad de interactuar con objetos virtuales utilizando nuestras propias manos representa un avance significativo en la manera en que experimentamos estos entornos inmersivos. A través del seguimiento y detección de gestos y poses de las manos, los usuarios pueden interactuar de forma más natural, mejorando así la inmersión en realidad virtual. Este proyecto no solo busca implementar estas funcionalidades, sino también explorar las mejores prácticas y tecnologías disponibles para lograrlo.

A-Frame es un framework de código abierto desarrollado por Mozilla que facilita la creación de experiencias VR en la web. Utilizando tecnologías como HTML, CSS y JavaScript [8], A-Frame permite a los desarrolladores construir entornos VR sin necesidad de profundos conocimientos en gráficos 3D. Esto ha democratizado el acceso al desarrollo de VR, permitiendo que una mayor diversidad de desarrolladores participe en la creación de experiencias inmersivas.

A-Frame ha simplificado enormemente el desarrollo de aplicaciones VR, proporcionando herramientas accesibles para la creación de entornos virtuales de manera sencilla. Antes, el desarrollo de experiencias VR requería conocimientos avanzados de programación y gráficos 3D, mientras que ahora, los desarrolladores pueden crear facilemente experiencias inmersivas con herramientas como esta.

Vue.js, por otro lado, es un framework progresivo de JavaScript que se utiliza para construir interfaces de usuario. Su capacidad para crear componentes reutilizables y su fácil integración con otras bibliotecas lo han convertido en una herramienta popular entre los desarrolladores

web. Integrar A-Frame con Vue.js ofrece una poderosa combinación para desarrollar aplicaciones VR interactivas y escalables.

El proyecto se estructura en varias escenas progresivas que permiten evaluar y mejorar el control de manos en VR. Cada escena sirve como un paso hacia el objetivo final: un escenario capaz de detectar y guardar poses con suficiente precisión. Estas etapas de desarrollo permiten probar y ajustar las funcionalidades implementadas, desembocando en una demostración funcional del sistema de control de manos. Se puede acceder al proyecto a través del siguiente enlace: https://samuelcoboscorrea.github.io/final-project

1.1. Contexto

Durante los últimos años, el desarrollo y la adopción de tecnologías de realidad virtual y realidad aumentada han transformado radicalmente la forma en que interactuamos con entornos digitales. Lo que antes era impensable sin las herramientas adecuadas, hoy es una realidad gracias al creciente interés y la popularidad de estas aplicaciones. Empresas líderes como Google, Apple, Facebook y Microsoft han invertido significativamente en adquisiciones estratégicas en este sector.

Uno de los campos más fascinantes de esta tecnología es el handtracking o seguimiento de manos, que permite a los usuarios interactuar de manera intuitiva y natural con entornos virtuales. La capacidad de detectar y seguir los movimientos de las manos ha revolucionado la experiencia de usuario en aplicaciones de VR y AR, facilitando desde gestos simples hasta interacciones complejas en tiempo real. Esta innovación no solo mejora la experiencia de usuario, sino que también abre nuevas posibilidades para aplicaciones en campos como la educación, la simulación médica y el diseño industrial.

Estas tecnologías no solo se limitan al ámbito del entretenimiento, sino que también están transformando industrias como la arquitectura, el diseño, la medicina y más. La aplicación que hemos desarrollado tiene aplicaciones tanto profesionales, en sectores como la arquitectura y el diseño, como personales, proporcionando experiencias de ocio inmersivas y educativas.

Este proyecto surge como respuesta al creciente potencial de estas tecnologías emergentes y su impacto en diversos sectores. A medida que continuamos avanzando en este camino de innovación, queda claro que la realidad virtual y aumentada están en constante evolución,

ofreciendo oportunidades sin precedentes para la creación de nuevas aplicaciones y servicios.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo principal de este proyecto es explorar y avanzar en el manejo y control de las manos en entornos de realidad virtual y aumentada. Se centra en la detección precisa de poses y gestos utilizando la integración de A-Frame con Vue.js. Se pretende superar las limitaciones actuales en la precisión y la respuesta de los sistemas de seguimiento de manos disponibles en A-Frame. Este enfoque no solo busca mejorar la interactividad y la naturalidad de las experiencias virtuales, sino también ampliar las capacidades de desarrollo de aplicaciones inmersivas y accesibles directamente desde el navegador.

1.2.2. Objetivos específicos

Dentro del contexto del foco principal de proyecto, y para conseguir cumplir el objetivo principal, se llevarán a cabo una serie de objetivos específicos indicados a continuación:

- Investigar y evaluar las tecnologías existentes para el manejo de gestos y poses en entornos virtuales.
- Integrar de manera efectiva A-Frame dentro del marco de Vue.js aprovechando sus capacidades para la creación de interfaces de usuario en realidad virtual.
- Facilitar la reutilización y la modularidad del código en el desarrollo de escenarios interactivos en realidad virtual utilizando componentes nativos de Vue.js y A-Frame.
- Estudiar en profundidad el componente de WebXR Hand y adaptarlo para su integración fluida en la detección y gestión de poses.
- Demostrar el potencial de interacción natural mediante gestos y poses en aplicaciones de realidad virtual.
- Facilitar la interacción natural y fluida del usuario mediante el control avanzado de las manos en entornos WebXR, mejorando la accesibilidad y la experiencia del usuario final.

- Implementar un sistema flexible que permita cambiar dinámicamente entre diferentes escenarios dentro del entorno principal de la aplicación.
- Desarrollar todas las funcionalidades y componentes bajo el estándar WebXR para garantizar la compatibilidad y accesibilidad en múltiples dispositivos y plataformas de realidad virtual.

1.3. Estructura de la memoria

- **Segundo capítulo:** Teconologías utilizadas, agrupando las herramientas necesarias a nivel tecnológico para la viabilidad del desarrollo cumplir los objetivos del proyecto.
- **Tercer capítulo:** Desarrollo del proyecto, siguiendo la metodología `AGILE` [10], se describe paso por paso el proyecto, determinando las fases por las que ha ido pasando, determinando estas etapas según la metodología como `Sprints`.
- **Cuarto capítulo:** Resultados, consta de una descipción funcional junto a un manual de usuario. En este capítulo se presentan los resultados obtenidos durante el desarrollo del proyecto, destacando las funcionalidades implementadas, así como las lecciones aprendidas. El manual de usuario que guiará a los usuarios sobre cómo interactuar con la aplicación.
- **Quinto capítulo:** Conclusiones, se evalua el éxito del proyecto en función de los objetivos planteados, reflexiona sobre los desafíos enfrentados y ofrece recomendaciones para futuras investigaciones y desarrollos relacionados con el proyecto.

Capítulo 2

Tecnologías utilizadas

En esta sección se exponen las diferentes tecnologías empleadas durante el desarrollo del proyecto.

2.1. Tecnologías principales

2.1.1. HTML5

HTML5 (*HyperText Markup Language 5*) es la versión más reciente del estándar HTML, introducido en 2014 por el World Wide Web Consortium (W3C). Esta versión trajo consigo nuevas características y mejoras significativas para el desarrollo web moderno. A continuación se detallan algunas de las características más destacadas de HTML5:

HTML5 incorporó etiquetas semánticas como `<header>`, `<footer>`, `<article>`, y `<section>`, que proporcionan una estructura más clara y significativa al contenido de la página. Estas etiquetas facilitan la accesibilidad y optimización para motores de búsqueda (SEO), ayudando a los desarrolladores a definir mejor el propósito de cada sección del documento.

Además de las mejoras semánticas, HTML5 introdujo soporte nativo para audio y video mediante las etiquetas `<audio>` y `<video>`. Esto eliminó la necesidad de plugins externos como Flash Player para reproducir contenido multimedia, mejorando la compatibilidad y la experiencia del usuario.

Otra característica importante de HTML5 es la API Canvas, que permite la renderización

de gráficos dinámicos, incluyendo gráficos 2D. La combinación de Canvas con JavaScript permite crear visualizaciones interactivas y juegos directamente en el navegador sin necesidad de plugins adicionales. Además de Canvas, HTML5 incluye la API WebGL [3], que proporciona soporte para gráficos 3D dentro del navegador. WebGL utiliza el estándar OpenGL ES para renderizar gráficos 3D de alto rendimiento, abriendo nuevas posibilidades para aplicaciones de realidad virtual, juegos y simulaciones interactivas en la web.

HTML5 también mejoró el soporte para aplicaciones web progresivas mediante APIs como Service Workers y Web Storage. Estas APIs permiten a las aplicaciones funcionar offline, enviar notificaciones push y almacenar datos localmente, proporcionando una experiencia de usuario más fluida y similar a la de una aplicación nativa.

El DOM (Document Object Model) es una representación estructurada de los elementos de un documento HTML organizados en forma de nodos, formando un ”árbol de nodos”. Gracias al DOM, se puede acceder y manipular los elementos del documento usando JavaScript, permitiendo un mayor dinamismo y funcionalidad en las páginas web.

Estructura Básica de un Documento HTML

Un documento HTML básico puede estructurarse de la siguiente manera:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>TFG</title>
</head>
<body>
    <p>¡Hola, main!</p>
</body>
</html>
```

En este ejemplo, `<html>` representa el contenedor principal del documento, `<head>` contiene metadatos y el título, mientras que `<body>` incluye el contenido visible.

2.1.2. A-Frame

A-Frame [5] es un framework de código abierto desarrollado por Mozilla VR en 2015. Su propósito es facilitar la creación de experiencias de realidad virtual directamente en HTML, eliminando la necesidad de conocimientos avanzados en WebGL. Además de HTML, A-Frame también utiliza otras tecnologías como WebXR y JavaScript.

Este proyecto se basa en A-Frame debido a su importancia fundamental. A-Frame permite a los desarrolladores crear escenas de realidad virtual de manera sencilla e intuitiva, utilizando HTML. Una de las características destacadas de A-Frame es su arquitectura basada en un modelo de entidad-componente, que se integra con Three.js.

Aquí se muestra un ejemplo básico de cómo crear figuras en A-Frame usando HTML:

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.6.0/
      aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box position="-1 0.5 -3" rotation="0 45 0"
        color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25"
        color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5"
        height="1.5" color="#FFC65D"></a-cylinder>
      <a-plane position="0 0 -4" rotation="-90 0 0"
        width="4" height="4" color="#7BC8A4"></a-plane>
      <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```

Este código HTML renderiza una escena en una página web con varias figuras geométricas. El resultado se puede ver en el navegador como una escena básica de A-Frame, como se muestra

en la figura 2.1.

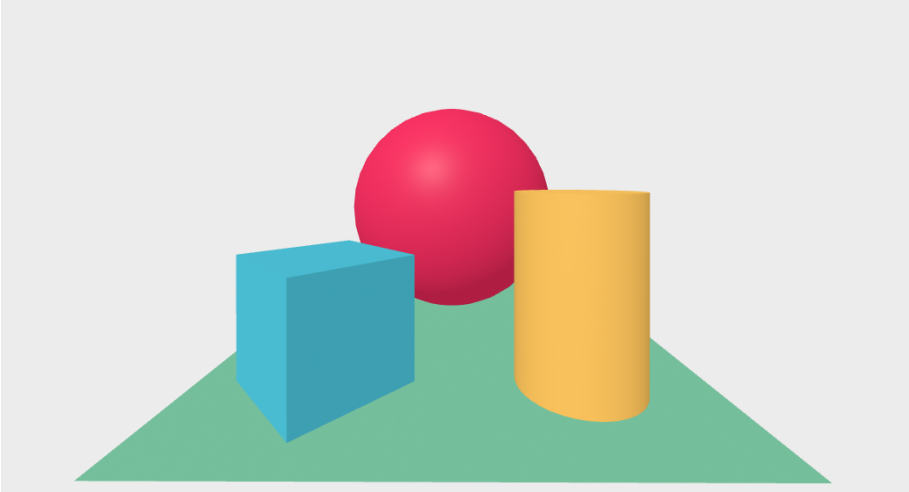

Figura 2.1: Imagen de un escenario básico con A-Frame

A-Frame utiliza un modelo de entidad-componente. Una entidad es el objeto HTML que se crea, y el componente es el comportamiento o funcionalidad que se le asigna, generalmente mediante JavaScript. Los componentes siguen una estructura estándar que incluye elementos como `schema`, que define las propiedades del componente, e `init`, que se ejecuta al inicializar el componente. Para añadir un componente a una escena, se lo incorpora al elemento correspondiente como si fuera un atributo.

También, es altamente extensible y modular, permitiendo a los desarrolladores añadir y compartir componentes personalizados, siendo compatible en todos los navegadores web modernos, lo que lo hace ideal para desarrollar en aplicaciones web.

Componentes y Entidades en A-Frame

A-Frame es un framework web para construir experiencias de realidad virtual (VR) en HTML. En A-Frame, los componentes y las entidades son conceptos fundamentales que permiten definir y manipular objetos y comportamientos en un entorno de realidad virtual.

Inicialización de Componentes En A-Frame, los componentes son bloques de construcción que encapsulan la lógica y el comportamiento de las entidades (objetos) en una escena de realidad virtual. Para inicializar un componente, se siguen estos pasos:

- **Definición del Componente:** Primero, se define el componente mediante la creación de un nuevo archivo JavaScript que contiene la lógica del componente. Este archivo define cómo se comportará el componente y qué atributos aceptará.

- **Registro del Componente:** Luego, el componente se registra en A-Frame utilizando el método 'AFRAME.registerComponent()'. Aquí se especifica el nombre del componente y se proporciona un objeto con métodos que definen su comportamiento.

- **Uso del Componente:** Una vez registrado, el componente puede ser utilizado en cualquier entidad dentro de la escena de A-Frame. Se añade como un atributo en HTML y puede configurarse con valores específicos según sea necesario.

Inicialización de Entidades Las entidades en A-Frame son los elementos visuales que conforman la escena de realidad virtual. Pueden representar objetos 3D, cámaras, luces y más. Para inicializar una entidad:

- **Creación de Entidades:** Las entidades se crean utilizando elementos HTML personalizados como `<a-entity>`. Dentro de estos elementos se pueden definir componentes, atributos y eventos que controlan el comportamiento y la apariencia de la entidad.

- **Añadir Componentes:** Los componentes se añaden como atributos a la entidad. Por ejemplo, si se tiene un componente de color, se puede añadir a una entidad con color, y actuará sobre este modificandolo.

- **Relación entre Componentes y Entidades:** Los componentes definen el comportamiento y las características de las entidades. Cada entidad puede tener múltiples componentes que trabajan juntos para definir cómo interactúan con el usuario y el entorno virtual.

2.1.3. WebXR

WebXR [1] es una API diseñada para desarrollar experiencias inmersivas de realidad aumentada (AR) y virtual (VR) directamente en navegadores web compatibles. Reemplazando a

la anterior WebVR, WebXR amplía su alcance para incluir tanto AR como VR, proporcionando un marco unificado y accesible para crear aplicaciones inmersivas.

Esta API utiliza tecnologías web estándar como HTML, JavaScript y CSS, lo que facilita a los desarrolladores crear experiencias inmersivas sin la necesidad de instalar aplicaciones nativas adicionales. Esto significa que los usuarios pueden acceder a estas experiencias directamente desde sus navegadores, mejorando significativamente la accesibilidad y la difusión de la realidad virtual y aumentada.

WebXR es compatible con una amplia gama de dispositivos, incluyendo auriculares de realidad virtual y dispositivos móviles con soporte AR como smartphones y tablets. Esta versatilidad asegura que los desarrolladores puedan llegar a una amplia audiencia sin depender de plataformas específicas de hardware.

Integrando también WebRTC, WebXR permite la transmisión de audio y video entre dispositivos, facilitando la colaboración y la comunicación en entornos virtuales. Esto no solo mejora la experiencia del usuario final, sino que también abre nuevas posibilidades para aplicaciones colaborativas y educativas en VR y AR.

2.1.4. Vue JS

Vue.js [6] es un framework progresivo de JavaScript utilizado principalmente para construir interfaces de usuario interactivas y dinámicas en aplicaciones web. A diferencia de otros frameworks más monolíticos como Angular o React, Vue.js se destaca por su enfoque gradual y su capacidad para ser adoptado de manera incremental en proyectos existentes.

Facilita la creación de aplicaciones web modernas al proporcionar una sintaxis clara y concisa que permite la creación rápida de componentes reutilizables. Estos componentes encapsulan HTML, CSS y JavaScript, lo que facilita la gestión y mantenimiento del código, especialmente en proyectos de gran escala.

Un ejemplo de estructura de código sería el siguiente:

```
<script setup>
  import { ref } from 'vue'
  const count = ref(0)
</script>
```

```
<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
  button {
    font-weight: bold;
  }
</style>
```

Se beneficia de una comunidad activa y un ecosistema de herramientas y bibliotecas complementarias que amplían su funcionalidad y facilitan el desarrollo. Esto incluye herramientas para la gestión del estado de la aplicación, enrutamiento, pruebas unitarias y más, haciendo que Vue.js sea una opción versátil para desarrolladores que buscan eficiencia y productividad en el desarrollo web moderno.

Características esenciales de Vue.js

- **Reactividad:** Vue.js ofrece un sistema de reactividad[1] bidireccional que actualiza automáticamente el DOM cuando los datos cambian, sin necesidad de recargar la página. Esto mejora la usabilidad y la experiencia del usuario al proporcionar interacciones rápidas y fluidas.

- **Componentes:** Se centra en el uso de componentes[2] como bloques de construcción fundamentales para interfaces de usuario. Estos componentes encapsulan código reutilizable que incluye fragmentos de HTML, JavaScript y CSS, facilitando la modularidad y escalabilidad de los proyectos.

- **Virtual DOM:** Utiliza un Virtual DOM para optimizar el rendimiento al realizar cambios en la vista. Este enfoque garantiza actualizaciones eficientes del DOM real, mejorando el rendimiento y la experiencia del usuario.

[1] https://vuejs.org/guide/extras/reactivity-in-depth.html
[2] https://vuejs.org/guide/essentials/component-basics.html

- **Eventos y transiciones:** Vue.js permite reaccionar a eventos del DOM de manera intuitiva y eficiente, facilitando la implementación de interacciones dinámicas. Además, proporciona un sistema sencillo para gestionar transiciones y animaciones, aprovechando las capacidades de CSS.

2.1.5. JavaScript

JavaScript es un lenguaje de programación interpretado que se utiliza principalmente para crear contenido dinámico en las páginas web. Es una de las tecnologías fundamentales de la web junto con HTML y CSS, y permite a los desarrolladores implementar funciones complejas en sus sitios web.

JavaScript fue desarrollado originalmente por Netscape como un medio para añadir programas a las páginas web en el navegador Netscape Navigator. Hoy en día, es un lenguaje ampliamente utilizado, tanto en el lado del cliente (frontend) como en el lado del servidor (backend).

Es un lenguaje interpretado, se ejecuta directamente en el navegador web facilmente y sin ninguna complicación, lo que permite a los desarrolladores y a la comunidad, ver los cambios de inmediato. Aunque no es un lenguaje orientado a objetos puro, también soporta la programación orientada a objetos. JavaScript es de tipado débil y dinámico, que destaca por su rapidez de desarrollar y su flexibilidad.

Es conocido por su capacidad para manejar eventos y manipular el DOM (Document Object Model), permitiendo la creación de interfaces de usuario interactivas, y es compatible con todos los navegadores web modernos, lo que lo hace ideal para el desarrollo web multiplataforma A continuación, se muestra un ejemplo básico de un script en JavaScript que muestra un mensaje en la consola del navegador:

```
console.log('¡Hola, main!');
```

Uso de JavaScript en el Desarrollo Web

JavaScript permite a los desarrolladores web mejorar la experiencia del usuario a través de la creación de contenido dinámico y de la manipulación del DOM. Esto incluye tareas como

la validación de formularios, la creación de animaciones, la carga de contenido sin recargar la página (AJAX) y la interacción con APIs externas.

Además, con la aparición de frameworks y bibliotecas como React, Angular y Vue.js, JavaScript se ha convertido en una herramienta muy útil para el desarrollo de aplicaciones web modernas y complejas. En el backend, Node.js ha permitido que JavaScript se utilice también del lado del servidor, facilitando el desarrollo de aplicaciones web de pila completa (full-stack) con un solo lenguaje de programación.

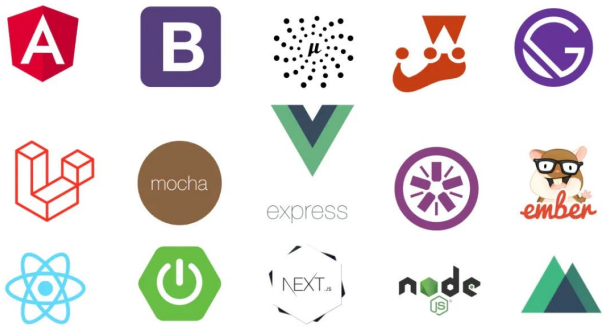

Figura 2.2: Imagen de los diferentes frameworks

2.1.6. Three.js

Three.js [7] es una poderosa biblioteca de JavaScript para la creación de gráficos 3D en navegadores web. Utiliza WebGL para renderizar escenas 3D en tiempo real de manera eficiente, permitiendo a los desarrolladores crear aplicaciones interactivas y visualmente atractivas sin necesidad de profundizar en los detalles técnicos de WebGL.

Three.js simplifica el proceso de desarrollo proporcionando una API de alto nivel que abstrae la complejidad de WebGL. Esto incluye la creación y manipulación de geometrías 3D como cubos, esferas y planos, así como la aplicación de materiales con texturas y efectos de iluminación realistas. A continuación se muestra una imagen en 3D realizada con Three.js.

La biblioteca ofrece herramientas para gestionar cámaras y controles de usuario, facilitando la navegación y la interacción dentro de las escenas 3D. Además, soporta animaciones suaves y dinámicas para objetos y cámaras, permitiendo crear efectos visuales impresionantes y juegos interactivos.

Figura 2.3: Imagen en 3D realizada con Three.js

Three.js es apoyado por una activa comunidad de desarrolladores y ofrece una amplia gama de recursos adicionales como extensiones y plugins que amplían su funcionalidad. Esto hace de Three.js una opción versátil y robusta para proyectos que requieren gráficos 3D avanzados en la web.

2.2. Tecnologías auxiliares

2.2.1. Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es de código abierto y está disponible para Windows, macOS y Linux. VS Code es ampliamente utilizado por desarrolladores de software debido a su versatilidad, flexibilidad y amplia gama de características que facilitan la edición y depuración de código de programación.

Es altamente personalizable a través de extensiones que los usuarios pueden instalar a través del Marketplace. Desde extensiones para autocomplementar código o funciones predefinidas, hasta correctores y compiladores.

Dentro de las herramientas y características productivas que VS Code ofrece, caben destacar varias que se han utilizado a lo largo del proyecto para facilitar su desarrollo y depuración:

- **Source Control:** Integración con GitHub, permitiendo la gestión de versiones del código,

y manteniendo el despliegue continuo.

- **LaTeX Workshop (Extensión):** Realización de la memoria, facilitando la escritura, compilación y previsualización de documentos.

- **Prettier ESLint (Extensión):** Formateo de código JavaScript, asegurando un formato consistente y la adherencia a las mejores prácticas.

- **A-Frame Completion (Extensión):** Proporcionando autocompletado y snippets específicos para el desarrollo de experiencias de realidad virtual.

2.2.2. GitHub

GitHub es una plataforma basada en Git [9] (sistema de control de versiones de código fuente), diseñada para alojar proyectos de desarrollo de software y facilitar la colaboración entre desarrolladores. Permite almacenar y gestionar repositorios de código de manera eficiente, proporcionando herramientas para el control de versiones y la revisión de código, entre otras.

Es ampliamente utilizado en la comunidad de desarrollo debido a su capacidad para soportar proyectos de diferentes magnitudes, ofreciendo funciones clave como ramificaciones (branches) para el desarrollo paralelo, pull requests para la revisión de cambios y la integración continua para automatizar el proceso de construcción y despliegue. En la figura 2.4 se muestra un ejemplo de la organización habitual de las ramas de un repositorio.

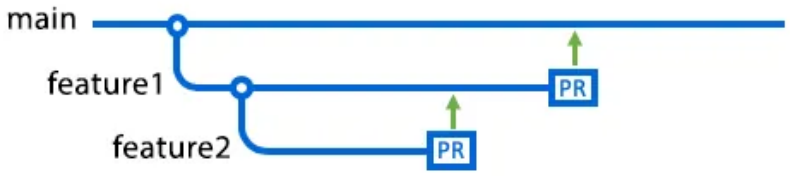

Figura 2.4: Diagrama de ramas en GitHub

Permite a los desarrolladores trabajar simultáneamente en proyectos, mantener un historial detallado de cambios y gestionar contribuciones mediante permisos de acceso y roles definidos. Entre las características destacadas de GitHub se incluyen:

- **Control de Versiones:** Mantiene un registro de todos los cambios realizados en el código, facilitando la reversión a versiones anteriores.
- **Gestión de incidencias:** Permite a los equipos registrar y gestionar problemas, tareas y solicitudes de nuevas características de manera estructurada.
- **Integración con Herramientas de Desarrollo:** Integra fácilmente con herramientas de desarrollo populares y servicios de integración continua para automatizar pruebas y despliegues.
- **Comunidad y Colaboración:** Fomenta la colaboración abierta al permitir que los desarrolladores contribuyan a proyectos de código abierto y participen en discusiones a través de comentarios y revisiones de código.

Ha permitido estructurar y mantener de manera eficiente el código fuente utilizado en el proyecto. Ha sido esencial para gestionar cambios, experimentar con nuevas funcionalidades y asegurar la estabilidad del proyecto a lo largo del desarrollo.

2.2.3. LaTeX

LaTeX es un sistema de composición tipográfica basado en TeX, ideal para la creación de documentos científicos y técnicos que incluyen textos y fórmulas matemáticas. Funciona como un procesador de macros, permitiendo un alto control sobre la estructura y el diseño del documento final.

Es conocido por su capacidad de gestionar referencias, bibliografías y otros elementos complejos de manera eficiente. Aunque su aprendizaje inicial puede ser exigente, ofrece la posibilidad de componer fórmulas con calidad de imprenta y exportar el resultado a PDF. LaTeX es software libre, distribuido bajo la licencia LPPL, lo que facilita su uso, modificación y distribución libremente. Esta propia memoria esta realizada con LaTeX.

Capítulo 3

Desarrollo del proyecto

En este capítulo se detalla el proceso de desarrollo del proyecto de fin de grado, desde su concepción hasta la implementación y evaluación de resultados. Se describen los distintos sprints, metodologías utilizadas y las decisiones clave tomadas durante el desarrollo.

3.1. Sprint 0

Puesta en marcha del proyecto, incluyendo conversaciones iniciales con el tutor del TFG y el inicio formal del proyecto.

Objetivo principal

El objetivo principal de este sprint es identificar las necesidades del proyecto y planificar su duración, dividiendo este en fases para facilitar su desarrollo. Esto incluye la definición de los objetivos específicos del proyecto, la identificación de los recursos necesarios.

Definición de necesidades y requisitos

La identificación de las necesidades y requisitos del proyecto fue fundamental para asegurar que todos los aspectos importantes se consideraran desde el principio. El objetivo principal de este proyecto se centró en la integración de A-Frame con Vue.js para el manejo de gestos y posturas.

Para lograr esto, se definieron claramente los requisitos funcionales y no funcionales, prestando especial atención a las capacidades de A-Frame para soportar la interacción a través

de gestos. Se realizó un análisis exhaustivo de las herramientas y tecnologías a utilizar, como Vue.js y A-Frame, para asegurarse de que eran adecuadas para los objetivos del proyecto.

Planificación de la duración del proyecto y división en sprints

La planificación de la duración del proyecto y su división en sprints, fue un punto clave para partir desdes una buena base, y organizar el trabajo en curso. Decidiendose así, la duración aproximada de Sprint siguiendo la metodología SCRUM [2], de dos semanas. Cada sprint funcionó con reuniones finales de Sprint y principio del siguiente, lo cual fue de gran utilidad para evaluar el progreso y ajustar la planificación.

3.2. Sprint 1

El objetivo principal de este sprint fue la puesta a punto del proyecto y el comienzo oficial del mismo.

3.2.1. Objetivo principal

En esta fase inicial, se creó el proyecto en GitHub y se estableció el tablero Kanban donde se organizaban todas las tareas. Además, se instalaron las dependencias necesarias para el desarrollo, incluyendo Vue.js en local, utilizando Vue CLI para generar un nuevo proyecto.

También se trabajó en comprobar la compatibilidad entre A-Frame y Vue.js, asegurando una integración fluida entre ambas tecnologías.

Por último, se prepararon los entornos de desarrollo en local para garantizar un flujo de trabajo eficiente y organizado.

3.2.2. Implementación del desarrollo

La implementación del desarrollo comenzó con la creación del repositorio en GitHub[1], configurando el control de versiones.

Como herramienta de trabajo utilizamos Visual Studio Code, y de manera mas intuitiva, la extensión que te proporciona de git para que sea mas intuitivo el proceso.

[1] https://github.com/samuelcoboscorrea/final-project

Una vez teniendo la herramienta para desarrollar el código de programación, comenzamos con crear el nuevo proyecto de Vue.js y añadirlo al repositorio de codigo inicial

Utilizando los comandos:

```
vue create final-project
```

para crear el proyecto en Vue.js[2], se añade a una nueva rama llamada `develop`, donde tendremos todos los desarrollos finales, asegurando que todas las dependencias necesarias estuvieran correctamente instaladas.

Se configuró el tablero Kanban en GitHub Projects, donde se definieron las tareas iniciales y se organizó el flujo de trabajo.

El siguiente paso fue la integración de A-Frame en el proyecto de Vue.js. Esto se logró importando A-Frame a través del archivo `main.js` y configurando el entorno para que soportara la creación de componentes VR.

```
import 'aframe'
```

Previamente, se instaló A-Frame utilizando npm:

```
npm install aframe
```

También se configuraron los archivos de configuración de Vue.js para asegurar una compatibilidad total con A-Frame. Finalmente, se configuraron los entornos de desarrollo en local, incluyendo la instalación de editores de código recomendados, extensiones útiles y la configuración de servidores locales para pruebas rápidas y eficientes.

Ramas y flujo en GitHub

En la metodología de trabajo llevada a cabo durante el proyecto, cada tarea se creaba como una `issue` que proporciona el Kanban de GitHub, haciendo mas llevadero y organizado el proyecto.

Tenemos una rama principal `main`, donde desemboca el proyecto final, y desde donde parte la rama `develop`, donde se pretenden desarrollar los desarrollos.

Posteriormente a esto, desde `develop` partiran las ramas `feature/task-1`, `feature/task-2`, `etc`, que corresponden con las diferentes tareas llevadas a cabo, desembocando nuevamente en `develop` cada vez que la tarea llegaba a su fin.

[2] https://cli.vuejs.org/guide/creating-a-project.html

3.2.3. Resultados obtenidos

Los resultados obtenidos fueron satisfactorios. Se logró establecer una base sólida para el desarrollo del proyecto con un repositorio bien configurado y un tablero Kanban organizado. La instalación de dependencias se completó sin inconvenientes y se creó el nuevo proyecto de Vue.js. Además, se logró importar y configurar A-Frame en el entorno de Vue.js, permitiendo la creación de componentes de realidad virtual.

3.2.4. Planificación

Para el siguiente sprint, se planificó continuar con la línea de trabajo establecida. El enfoque principal será implementar una puesta a punto más avanzada con A-Frame y familiarizarse profundamente con la tecnología. Se planificaron tareas específicas para crear componentes de realidad virtual básicos, explorar más funcionalidades de A-Frame y mejorar la integración con Vue.js.

3.3. Sprint 2

En el segundo sprint, se enfocó en explorar y familiarizarse con el framework A-Frame, así como en implementar escenarios simples utilizando esta tecnología.

3.3.1. Objetivo principal

El objetivo principal fue adquirir un conocimiento profundo del framework A-Frame y su integración con entornos de desarrollo local. Se propuso aprender a manipular el DOM básico con JavaScript dentro de A-Frame y crear escenarios de realidad virtual simples.

3.3.2. Implementación del desarrollo

Se comenzó por configurar el entorno de desarrollo local para A-Frame, asegurándose de tener las herramientas necesarias instaladas, como el servidor local y editores de código compatibles.

Para familiarizarse con el manejo del DOM en A-Frame, se implementaron ejemplos básicos que incluían la creación y manipulación de entidades en un entorno de realidad virtual. Se

utilizaron componentes estándar de A-Frame como `a-box`, `a-sphere`, y `a-plane` para construir escenarios simples que demostraran el uso práctico de la biblioteca[3].

3.3.3. Resultados obtenidos

Se obtuvieron resultados satisfactorios al implementar diferentes escenarios utilizando A-Frame. En la figura 3.1 se presenta un ejemplo de código que muestra cómo crear un simple escenario con algunos objetos en un entorno VR:

```
<a-scene background="color: #ECECEC">
  <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9">
  </a-box>
  <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E">
  </a-sphere>
  <a-plane position="0 0 -4" rotation="-90 0 0" width="4"
  height="4" color="#7BC8A4"></a-plane>
</a-scene>
```

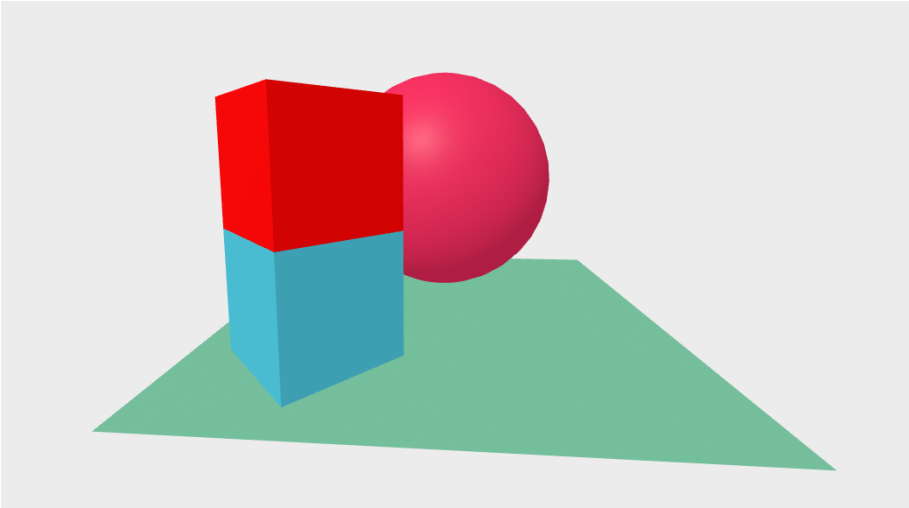

Figura 3.1: Escenario básico con A-Frame

[3] https://aframe.io/docs/1.6.0/introduction/

3.3.4. Planificación

Con el entorno de desarrollo establecido y una comprensión básica de A-Frame, se planea avanzar hacia la creación de escenarios más complejos que integren interactividad y funcionalidades mas completas, poniendo el punto de mira hacía el control de manos. En el próximo sprint, se centrará en mejorar la experiencia del usuario mediante la implementación de interacciones con eventos e integrar A-Frame con Vue.js.

3.4. Sprint 3

En el tercer sprint, continuamos avanzando con el conocimiento adquirido en A-Frame, el manejo del DOM y Vue.js, integrando estos elementos en una aplicación compuesta siguiendo ciertas pautas.

3.4.1. Objetivo principal

El objetivo principal fue componetizar la aplicación Vue.js utilizando componentes simples y unificando los componentes propios de Vue.js con los de A-Frame. Se instaló y utilizó dependencias de A-Frame dentro de un entorno Vue.js para facilitar su integración y reutilización.

3.4.2. Implementación del desarrollo

Para lograr la componetización, se comenzó creando componentes Vue.js que representan entidades básicas de A-Frame. Una forma muy básica de organizar el código, dentro del paquete components de vue, dentro de src, se crea un paquete llamado aframe, el cual se constituye de todo lo relacionado con la integración.

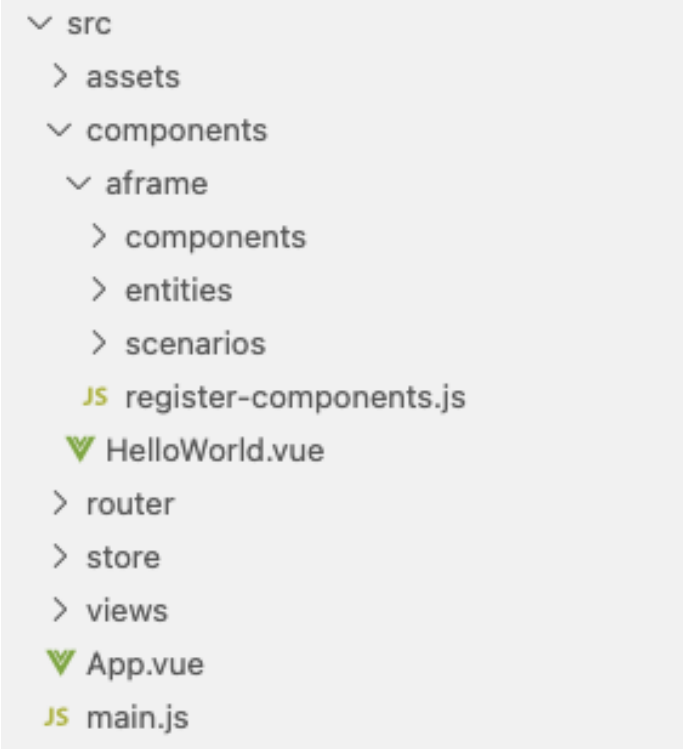

Figura 3.2: Estructura principal del código

El componente `Entity` se diseñó para reflejar un `a-entity` de A-Frame dentro de Vue.js, además es una referencia ya que se podrá reutilizar para la creación de cualquier otra entidad de A-Frame, siendo esta una base para el resto del proyecto.

```
<script setup>
import { watch } from 'vue'
const emit = defineEmits(['onClickEvent', 'onChangeProps'])

const props = defineProps({
  aprops: Object
})

watch(props, (newValue, oldValue) => {
  emit('onChangeProps')
});

const handleClickEvent = () => {
```

```
  emit('onClickEvent')
}

</script>

<template>
  <a-entity v-bind="props.aprops" @click="handleClickEvent">
    <slot></slot>
  </a-entity>
</template>
```

Además, se crearon otros componentes similares para entidades como `Assets.vue`, `Box.vue`, `Cylinder.vue`, `Grid.vue`, `Ocean.vue`, `Plane.vue`, `Sky.vue`, y `Sphere.vue`, adaptando cada uno para facilitar su uso como componentes Vue.js nativos.

Se configuró un escenario llamado `BasicScenario`[4], equivalente al `a-scene` de A-Frame, donde se integraron estos componentes:

```
  ...
      <Scene embedded :aprops="basicSceneProps">
        <Sky :aprops="skyProps" />
        <Entity id="camera" :aprops="cameraEntityProps">
          <Camera />
        </Entity>
        <Grid :aprops="gridProps" />
        <Entity ref="entity-box" :aprops="entityProps">
          <Box ref="first-box" :aprops="boxProps"
          @onClickEvent="handleBoxClickEvent(boxProps)"/>
        </Entity>
      </Scene>
  ...
```

[4] https://samuelcoboscorrea.github.io/final-project/scenario/basic

3.4.3. Resultados obtenidos

Esta fue una parte clave en el proceso, ya que, de haber encontrado incompatibilidades entre los dos frameworks diferentes, nuestros objetivos del proyecto no podrían haberse cumplido.

Conociendo la posibilidad de realizar todo tipo de componentes en realidad virtual utilizando esta integración fue muy satisfactorio.

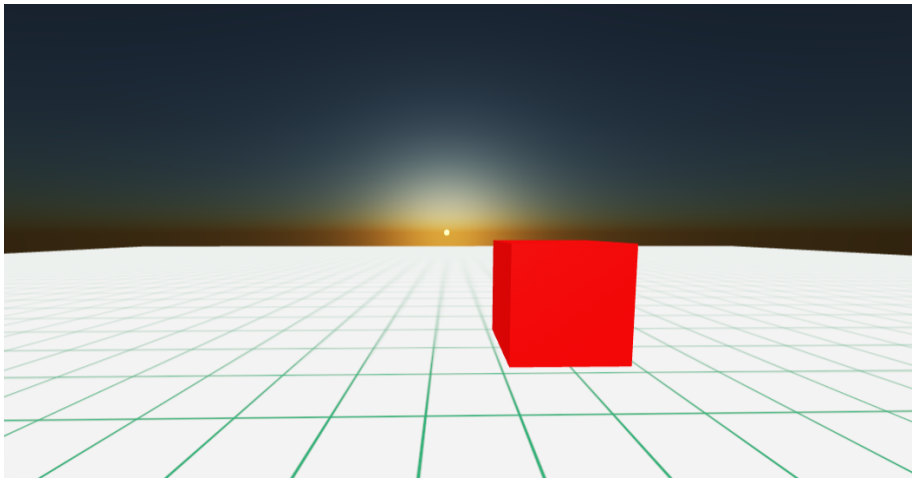

Figura 3.3: Escenario implementado en Vue.js y A-Frame

Se obtuvieron resultados satisfactorios al implementar la compenetización de la aplicación Vue.js con componentes A-Frame. En la siguiente figura 3.3 se presenta un ejemplo del escenario 'BasicScenario', siendo el resultado de la integración entre ambos frameworks.

3.4.4. Planificación

Con la base de la aplicación compenentizada establecida, el siguiente paso será interactuar con componentes como `hand-controls`, `hand-tracking-controls`, `hand-tracking-grab-controls` y otros, para integrar interacciones avanzadas con las manos en entornos de realidad virtual.

3.5. Sprint 4

Una vez establecidos los conocimientos y la integración básica de A-Frame con Vue.js, el siguiente paso es crear escenarios y componentes específicos de A-Frame

dentro de Vue. El enfoque principal en este sprint fue el uso de componentes de A-Frame que permiten la interacción con las manos, como `hand-tracking-controls` y `hand-tracking-grab-controls`.

3.5.1. Objetivo principal

El objetivo principal de este sprint fue la creación de un escenario completo llamado `Panel.vue`, junto con la implementación de los componentes clave y las vistas necesarias. Además, se integraron controles manuales utilizando `hand-tracking-controls` y `hand-tracking-grab-controls`.

3.5.2. Implementación del desarrollo

La implementación del desarrollo se llevó a cabo de manera estructurada, comenzando con la creación del componente `Panel.vue`. A continuación se muestra el código de este componente:

```
...
  <Scene :aprops="basicSceneProps">
    <Sky :aprops="skyProps" />
    <Entity id="camera" :aprops="cameraEntityProps">
      <Camera>
      </Camera>
      <Entity :aprops="leftHandProps"/>
      <Entity :aprops="rightHandProps"
        @pinchstarted="handlePinchStarted"
        @pinchended="handlePinchEnded"
        @pinchmoved="handlePinchMoved"
      />
    </Entity>

    <ControlPanel @selectItem="handleSelectItem"
    :position="'0 1.5 -0.5'" :handsData="text"/>
```

```
      <Grid id="grid" :aprops="gridProps">
        <Tatami :items="items"/>
      </Grid>
    </Scene>
  ...
```

En este código, se utilizan eventos como `pinchstarted`, `pinchended` y `pinchmoved` para detectar interacciones con las manos. Estos eventos permiten que, al realizar un gesto de pinza, se lance un evento controla la posición de la mano en cada momento y la muestra por consola.

El `ControlPanel` es un componente crucial dentro de `Panel.vue`:

```
  ...
    <Entity id="control-panel" :aprops="menuProps">
      <Entity :aprops="backgroundMenuProps" >
        <ButtonLabel v-for="(buttonLabel, index) in buttonLabels"
        :key="index" v-bind="buttonLabel"
        @selectItem="handleSelectItem(buttonLabel)"/>
      </Entity>
    </Entity>
  ...
```

Este componente contiene `buttonLabel`, que representa cada uno de los botones del panel. Estos botones tienen el componente de A-Frame `pressable`:

```
AFRAME.registerComponent('pressable', {
  ...
```

El componente `pressable` controla el evento de clickar un objeto con las manos, lanzando un evento desde los componentes hijos hasta el principal del panel. Una vez llegado a este último, se mostrará un objeto u otro dependiendo del botón que haya pulsado, en el componente mencionado anteriormente `Tatami`.

3.5.3. Resultados obtenidos

Para el diseño del Panel de control, se realizó un escenario llamado `Editor`[5] mostrado en las figuras 3.4, que recoge diferentes elementos, en ellos el precesor al panel de control. Esto sirvió de base para la creación y avance del proyecto.

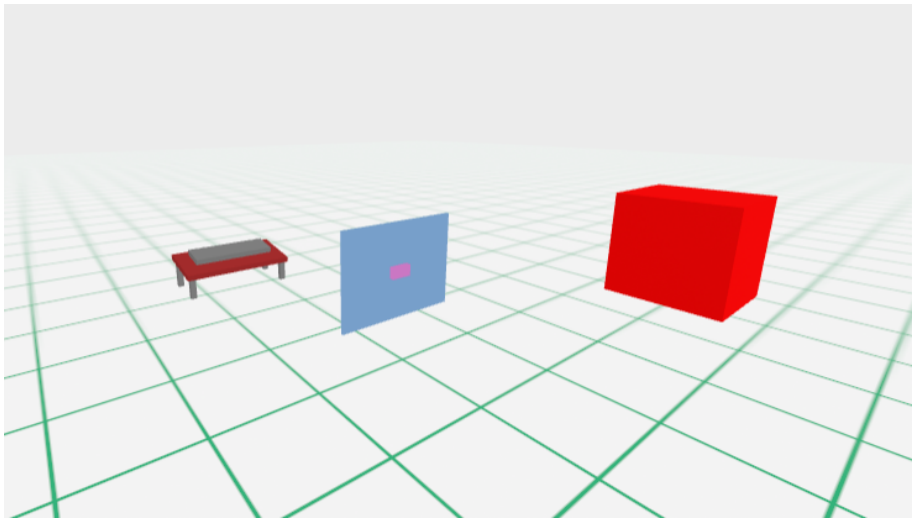

Figura 3.4: Escenario Editor

Se logró crear un escenario interactivo completo en `Panel`[6] reflejado en las figuras las 3.5 3.6, donde se pueden manipular fácilmente un escenario utilizando las manos, donde puedes mover el panel de control y clickar en cada una de las opciones para que aparezcan objetos en el `Tatami`

Después de presionar en alguno de los objetos, observamos en la siguiente imagen como aparecen dentro del `Tatami`

[5] https://samuelcoboscorrea.github.io/final-project/scenario/editor
[6] https://samuelcoboscorrea.github.io/final-project/scenario/panel

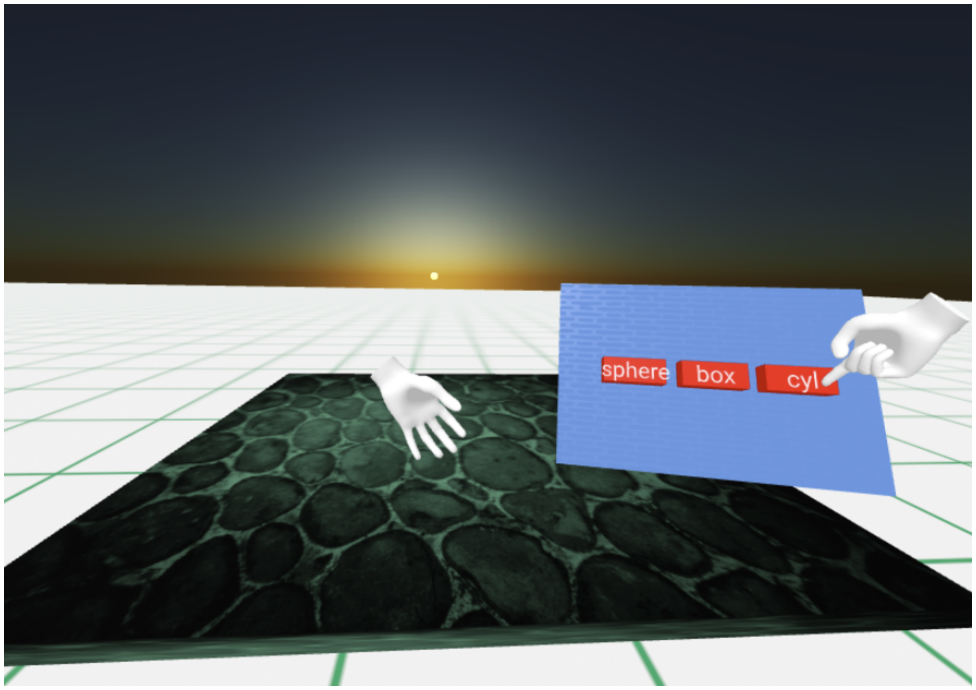

Figura 3.5: Escenario Panel de Control

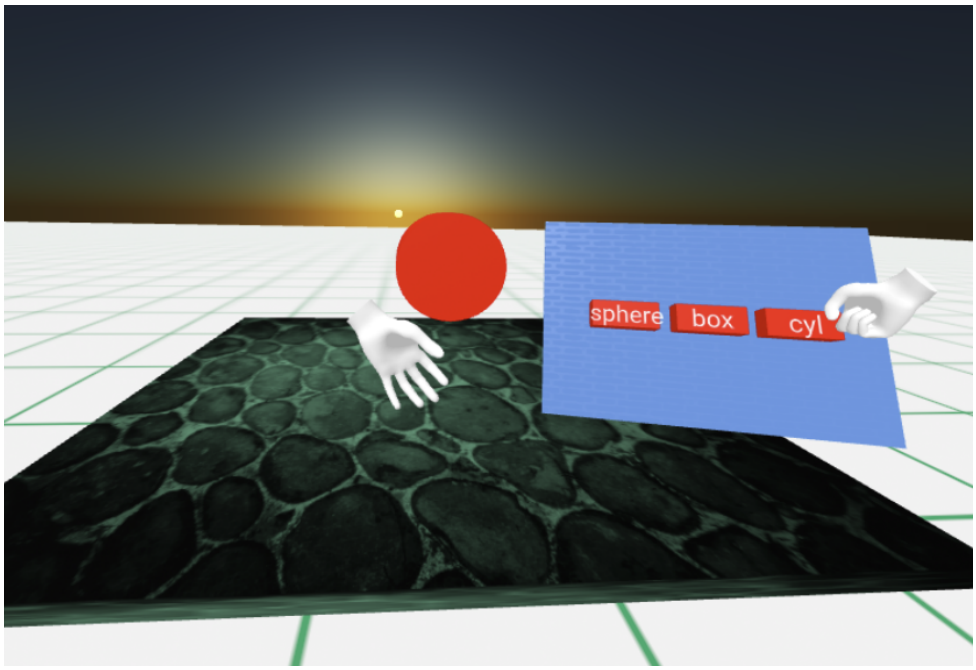

Figura 3.6: Escenario Panel de Control después presionar un botón

Estas imágenes muestran cómo el objeto aparece en el `Tatami` tras realizar el gesto de pinza, demostrando la funcionalidad de los componentes implementados.

3.5.4. Planificación

La siguiente fase del proyecto se centrará en estudiar y avanzar con la creación de componentes específicos de A-Frame para la detección de poses, ampliando las capacidades de interacción manual más allá de los gestos básicos actualmente soportados.

3.6. Sprint Final

En la fase final del proyecto, con toda la base de integración y el conocimiento necesario, el enfoque se centra en la creación de un detector de poses capaz de reconocer e interactuar con poses realizadas por nuestras propias manos en una escena de realidad virtual.

3.6.1. Objetivo principal

El objetivo principal de esta última fase es la creación de un escenario y los componentes necesarios para realizar un detector de poses utilizando WebXR Hand [4]. Este detector debe ser capaz de analizar cada movimiento de las manos y reconocer distintas poses en tiempo real.

3.6.2. Implementación del desarrollo

Para esta implementación, se crearon dos componentes principales en Vue.js. El primero es el escenario, donde se aloja la lógica de negocio encargada de manejar la escena y la interacción con el usuario. Este componente se define de la siguiente manera:

```
...
  <Scene id="save-pose-scene" :aprops="basicSceneProps">
  ...
    <SavePosePanel :id="'save-panel'"
    @selectItem="handleSavePose" :handsData="text"/>
  ...
  </Scene>
...
```

El segundo componente principal es `SavePosePanel`, que es el encargado de guardar

las poses detectadas. Este componente maneja la lógica de interacción con los controles de las manos y proporciona una interfaz para guardar y revisar las poses detectadas.

En el escenario, la entidad que hace referencia a las manos en realidad virtual incluye el componente nativo de A-Frame `detect-pose`. Este componente se crea con la intención de detectar las poses y mostrar en el escenario dos puntos importantes: la pose actual que está detectando nuestra mano, y las poses que hemos ido guardando.

Además, en el escenario existe un botón de guardar que permite ir almacenando las poses detectadas. Este botón es parte del `SavePosePanel` y se activa mediante eventos de presionar detectados por los controles de mano.

3.6.3. Resultados obtenidos

Como resultado final del proyecto, se logró crear un entorno de realidad virtual donde los usuarios pueden interactuar con el escenario utilizando sus manos. El detector de poses es capaz de reconocer diferentes gestos y almacenarlos en una lista visual. En las figuras 3.7 3.8 se presentan algunas capturas del escenario final.

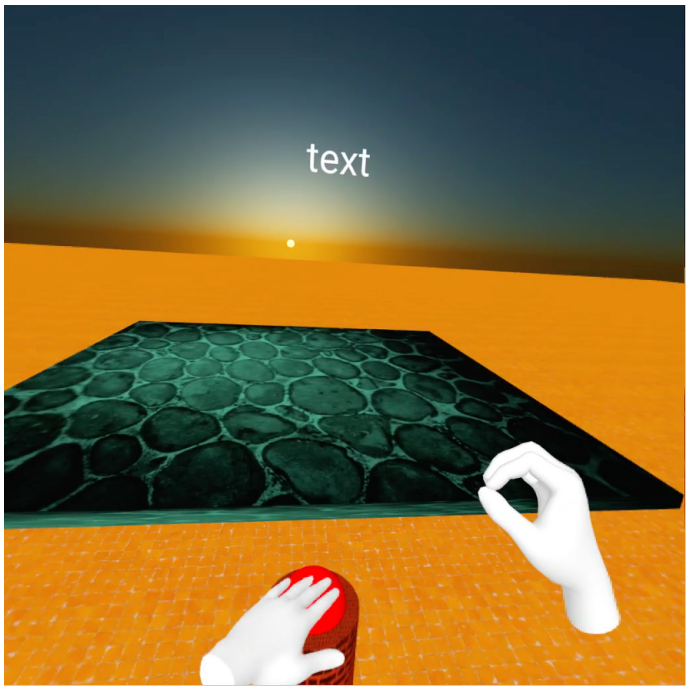

Figura 3.7: Escenario final sin poses guardadas

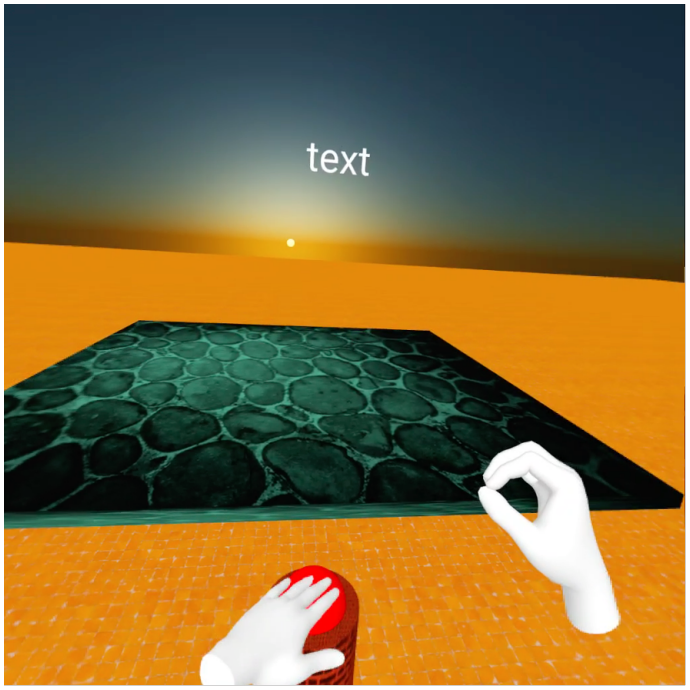

Figura 3.8: Escenario final sin poses guardadas

En estas imágenes, se puede observar el panel de control y el área de detección de poses. La primera imagen muestra el escenario inicial sin poses guardadas, mientras que la segunda imagen muestra el escenario después de que varias poses han sido detectadas y almacenadas. El botón de guardar, que se activa con la mano izquierda mientras se realiza la pose con la mano derecha, facilita la interacción y la gestión de las poses guardadas. Este sistema permite un reconocimiento preciso de gestos y proporciona una interfaz intuitiva para la interacción en realidad virtual.

Capítulo 4

Resultados

En esta sección se incluyen los resultados obtenidos a lo largo del proyecto, así como una descripción funcional del mismo, desarrollando las partes básicas de la conclusión final a nivel de funcionalidad. También destaca por un manual de usuario, que mostrará de manera sencilla como utilizar la aplicación desde un punto de vista de usuario y finalmente los detalles técnicos e implementación del proyecto.

4.1. Descripción funcional

En la primera área, el enfoque se encuentra en la creación y uso de componentes básicos de A-Frame que facilitan la interacción con las manos. Estos componentes incluyen funcionalidades predefinidas para manejar objetos, lo que permite a los usuarios realizar acciones como coger y mover objetos, así como interactuar con elementos en el entorno virtual.

En el escenario del Panel de Control, se demuestra cómo estas funcionalidades básicas se pueden aplicar de manera práctica. Los usuarios pueden coger un objeto con sus manos, moverlo dentro del escenario, y pulsar botones que, al ser activados, hacen aparecer nuevos objetos en la pantalla. Este panel es esencial para mostrar cómo las manos pueden utilizarse de manera intuitiva y eficiente en tareas comunes de interacción.

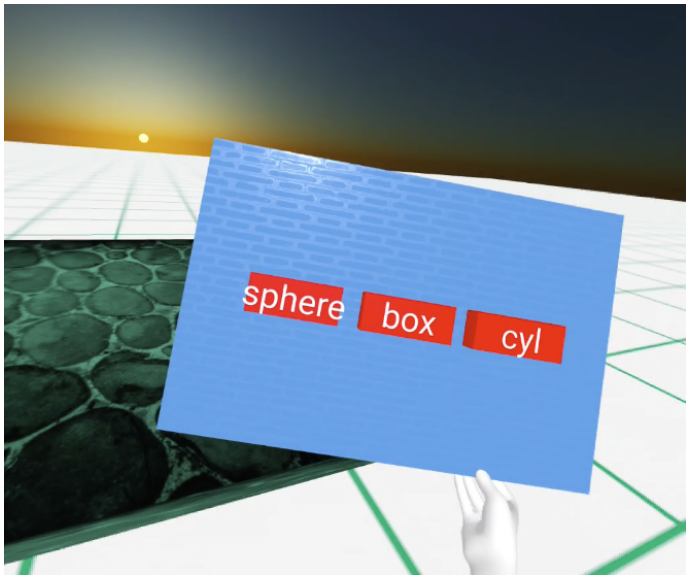

Figura 4.1: Moviendo el panel de control con las manos

La segunda área del proyecto se enfoca en la utilización avanzada de WebXR Hand, en la detección y almacenamiento de poses de las manos. En el escenario del Detector de Poses, se permite al usuario registrar diferentes poses de la mano de manera sencilla e intuitiva.

Cada vez que el usuario realiza una pose con la mano derecha, el sistema detecta y muestra la pose en tiempo real. Si el usuario desea guardar esta pose, puede hacerlo presionando un botón que se pulsa con la mano izquierda. Las poses guardadas se visualizan en una lista vertical, facilitando la revisión y gestión de las mismas.

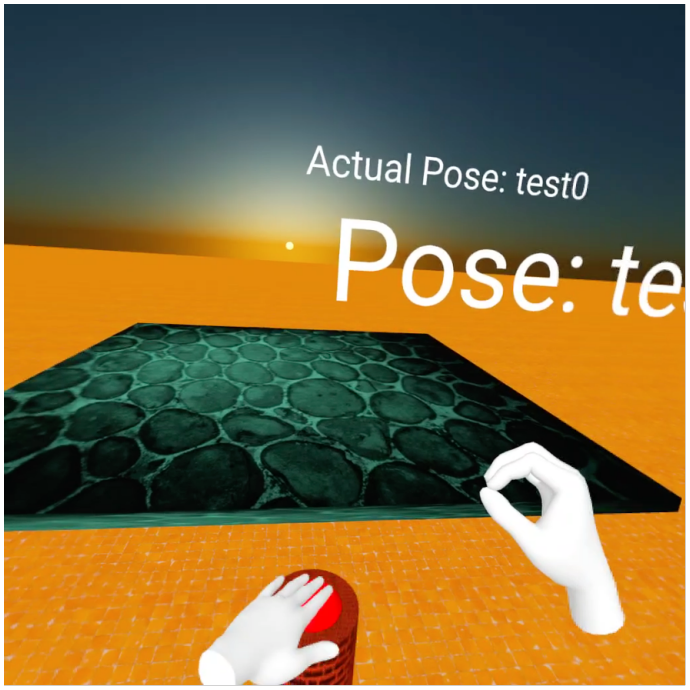

Figura 4.2: Pulsando el botón para guardar pose

4.2. Manual de usuario

Existen dos tipos de escenarios principales, uno de ellos es el panel de control, y el otro el escenario de detectar las poses. Partiendo de esta base, se explicaran de manera diferenciada los dos tipos de interacción con el usuario individualmente.

4.2.1. Panel de Control

En la pantalla principal del escenario, nos encontramos con diferentes componentes que lo forman. El tatami, que se muestra en el suelo de color azul verdoso, será el componente donde aparecerán las diferentes figuras, este está fijo siempre a nivel del suelo, y no se podrá mover. El panel de control, es el componente azul que aparece a mano derecha, es el encargado de manejar la escena dependiendo de la entrada (que en este caso, es la interacción con las manos). Dentro de este, tenemos tres botones, cada uno de ellos corresponde con un objeto diferente.

Para utilizar e interactuar sobre la escena, podemos desplazarnos por esta hasta acercarnos al tatami, para ver o interactuar con las figuras que aparecerán posteriormente. Utilizaremos las manos que nos aparecen de color blanco (nuestras propias manos), y tenemos varias

funcionalidades:

- **Mover el panel de control:** Con nuestra mano derecha, serermos capaces de, con el gesto de .ªgarrar", sobre el mismo panel de control, lo tendremos en nuestra mano y lo podremos desplazar con el propio movimiento del mismo. Este componente tiene la particularidad de `grabbable`, que hace que este objeto pueda ser agarrado gracias a sus propiedades.

- **Elegir un objeto:** Con nuestra mano izquierda, y pudiendo mover el panel de control al mismo tiempo, podemos pinchar con el gesto de "pinchar", la mano cerrada con el dedo índice estirado, sobre cualquiera de los tres botones de objetos que nos aparecen de color rojo. Cada uno, lleva su propia etiqueta para diferenciarse del resto. Una vez que interactuemos con el panel de control, se mostrarán dichos objetos en el Tatami.

También, nos podemos acercar hacia el Tatami, para interactuar con las figuras y cambiarlas de posición en la escena, esta parte esta limitada por el espacio fisico real donde esté el usuario en ese momento de la inmersion de realidad virtual.

4.2.2. Detector de poses

Dentro del escenario detector de poses, nos encontramos con un escenario parecido al anterior, el usuario podrá moverse por la escena hasta llegar a la zona principal donde podrá:

- **Guardar una pose:** Existe en pantalla el objeto de color ladrillo y una semiesfera roja que podremos pulsar con nuestra mano izquierda para guardar la pose. Antes de presionar el botón, se debe preparar la pose con la mano derecha, esta pose puede ser cualquiera, pero debe ser robusta y diferenciada para que el margen la detecte con claridad. Mientras mantenemos la pose con la mano derecha preparada, pulsamos el botón rojo con la mano izquierda para guardar.

- **Visualizar pose guardada:** De manera vertical vemos un listado de `Pose: ...` con todas las poses que hayamos guardado durante la experiencia de inmersión.

- **Detección y visualización de poses:** Una vez hayamos guardado poses, podremos visualizar al fondo como aparece `Actual Pose: ....` Este detector estará activo siempre y podemos ir cambiando las poses mientras nos fijamos en indicador para comprobar la pose que está detectando.

4.3. Descripción técnica e implementación

Nos centraremos en dos partes clave para reflejar los resultados obtenidos a lo largo del proyecto, la integración entre Vue.js y A-frame, y la detección de poses en un escenario de realidad virtual.

4.3.1. Vue.js y A-Frame

A lo largo del proyecto se ha creado una base estable de componentes básicos de A-Frame integrados directamente en Vue.js, lo que ha facilitado el desarrollo del mismo.

Se han diferenciado estas partes troncales para la creación de escenas en realidad virtual de cara a la integración de los frameworks:

- **Componentes**

  Para que A-Frame funcione dentro del marco de la aplicación de Vue.js, debemos importar el componente directamente creado en un archivo JavaScript en el archivo principal `main.js`

  ```
  import '@/components/aframe/components/components.js'
  ```

  Estos componentes (A-Frame) podrán ser reutilizados en cualquier parte de la aplicación, teniendo la ventaja de poder utilizarse en cualquier escena creada dentro del marco de la aplicación. En la figura 4.3 se muestra la estructura de componentes.

- **Entidades**

  En el caso de las entidades, se crearon diferentes tipos siguiendo una estructura 4.4, ya sean entidades básicas de A-Frame, como customizadas teniendo diferentes propiedades dependiendo de la utilidad.

  En cualquier escenario, simplemente con importar la entidad sería suficiente para utilizar este de manera fácil y sencilla. Estas entidades, a nivel de DOM realmente se están

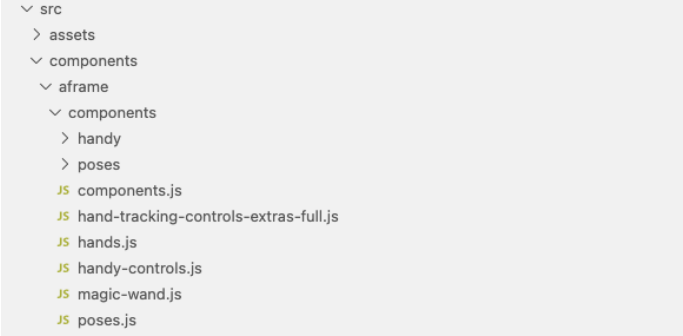

Figura 4.3: Estructura de componentes

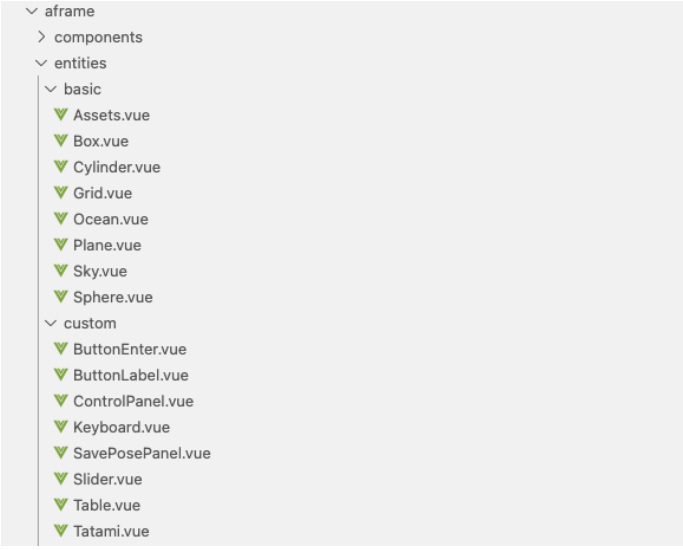

Figura 4.4: Estructura de entidades

interpretando con el lenguaje de A-Frame, permitiendo visualizar las escenas sin ningún inconveniente.

En la figura 4.5 se muestra el ejemplo del componente pulsador.

- **Escenarios**

Los escenarios son la parte central donde se integran las entidades y componentes para crear escenas completas de realidad virtual. Dentro de un escenario, se pueden incluir múltiples entidades y componentes, definiendo sus propiedades y comportamientos.

Para crear un escenario, se define una estructura en un componente Vue.js que utiliza las entidades de A-Frame.

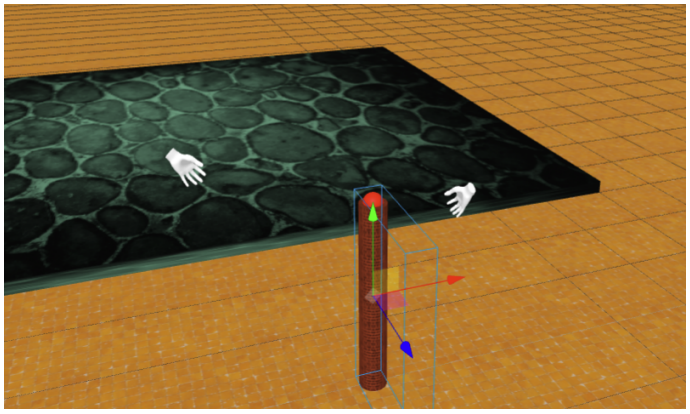

Figura 4.5: Ejemplo del componente pulsador en la escena

```
<template>
  <div class="basic-scenario">
    <div class="scene-container">
      <Scene id="save-pose-scene"
        :aprops="basicSceneProps">
        <Sky :aprops="skyProps" />

        <SavePosePanel :id="'save-panel'"
          @selectItem="handleSavePose"
          :handsData="text"/>

        <Entity @obbcollisionstarted="handleNewPoseAdded"
          :aprops="leftHandProps"/>
        <Entity @pinchstarted="inputKeyboard"
          @newpose="handleNewPoseAdded"
          :aprops="rightHandProps"/>

      </Scene>
    </div>
  </div>
</template>
```

En este ejemplo, se definen diversas entidades y componentes dentro de una escena.

Las propiedades de cada entidad se pasan a través de aprops, lo que permite una gran flexibilidad en la configuración del escenario

A continuación, se muestra cómo se definen las propiedades en el script del componente:

```
...
const poses = ref([])

const keyboardProps = ref({
  position: { x: 0, y: 1.076, z: -0.5 }
})

const leftHandProps = ref({
  id: 'leftHand',
  'hand-tracking-grab-controls': {
    hand: 'left'
  },
  'obb-collider': 'showColliders: true',
  'hand-positions': '',
})
...
```

En este código, se definen las propiedades de las entidades como referencias reactivas `ref`, lo que permite actualizar y gestionar fácilmente los estados de las entidades dentro del escenario.

4.3.2. HandTracking y control de manos

La detección de poses es un componente fundamental del proyecto y para implementarla de manera efectiva se siguieron varias etapas clave.

Primero, fue necesario crear un componente nativo de A-Frame llamado `detect-pose`, que se registra de la siguiente manera:

```
AFRAME.registerComponent('detect-pose', {
```

```
...
});
```

Este componente trabaja en conjunto con otro componente subyacente llamado `hands-tracking-controls`, específicamente con el subcomponente `tracked-controls`. El objetivo principal se logró calculando las distancias euclídeas de todos los puntos de la mano para determinar la pose actual. La figura 4.6 muestra el escenario al guardar una pose con las manos.

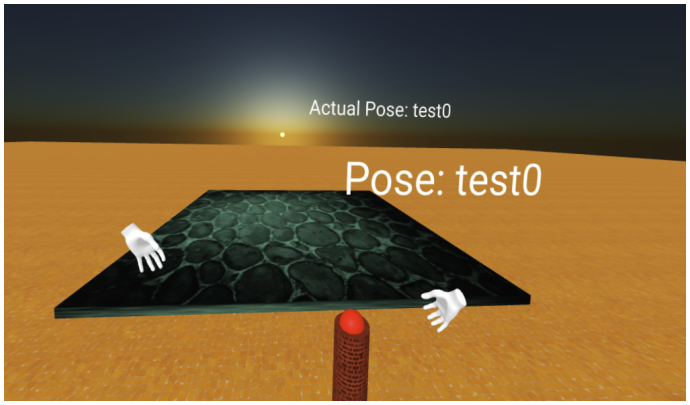

Figura 4.6: Guardado de una pose

La mano humana está compuesta por varias articulaciones clave, las cuales se enumeran a continuación:

```
var JOINTS = [
'wrist',
'thumb-metacarpal',
'thumb-phalanx-proximal',
...
'pinky-finger-phalanx-distal',
'pinky-finger-tip'
];
```

Estos son todos los puntos de la mano que detectamos. Finalmente, calculamos la distancia euclídea de cada uno de estos puntos con respecto al punto fijo de la muñeca, denominado `wrist`.

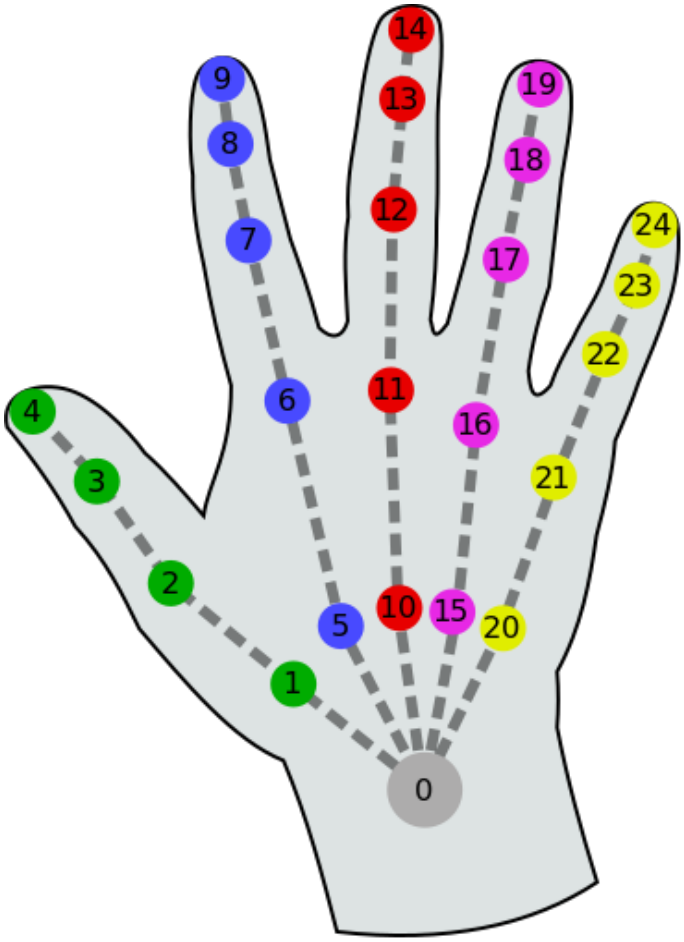

Figura 4.7: Definición de los puntos de la mano de XRHand

Una vez que tenemos estos cálculos, guardamos las poses en nuestro escenario dentro del componente `SavePosePanel.vue`, lo que nos permite mostrar las poses en pantalla.

Para visualizar la pose actual detectada, se crea un elemento dentro del componente `detect-pose` llamado `poseTextEl`. Este elemento se actualiza continuamente para reflejar la pose detectada en tiempo real, utilizando la función `tick`.

Con este enfoque, logramos detectar y visualizar las poses en tiempo real, proporcionando una interacción fluida y precisa dentro del entorno de realidad virtual. La figura 4.8 ilustra las poses detectadas y mostradas en pantalla:

Esta implementación ha sido crucial para el desarrollo del proyecto, permitiendo una interacción natural y precisa basada en la detección de poses de la mano del usuario.

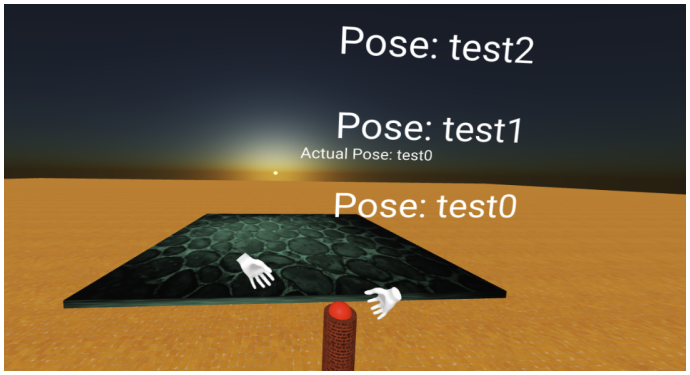

Figura 4.8: Guardado y detección de poses

4.3.3. Implementación

Desde el punto de vista técnico, y para desarrolladores con experiencia en Vue.js o frameworks de alto nivel basados en JavaScript, es muy intuitivo la creación de escenas y la integración y manejo de componentes nativos de A-Frame.

En primer lugar como en cualquier otro proyecto similar, tenemos que instalar las dependencias de node lanzando el siguiente comando:

```
npm install
```

A continuación, y después de instalar todas las dependencias instaladas para correr el proyecto, seguimos con el siguiente comando:

```
npm run serve
```

Vemos que nos aparece lo siguiente, y podremos acceder a la aplicación en local:

```
DONE  Compiled successfully in 1172ms

App running at:
- Local:   https://localhost:8085/
```

Figura 4.9: Correr aplicación en local

4.3.4. Creación de nuevos componentes

Para crear entidades de A-Frame encapsuladas dentro de un componente de Vue.js, nos desplazamos al paquete `src/components/aframe/entities` donde se podrán crear archivos con extensión `.vue`, para crear entidades customizadas, ya que las básicas de A-Frame

están cubiertas.

Para la creación de una entidad básica se estructura de la siguiente forma:

```
<template>
  <Entity :aprops="entityProps"
    @click="handleClickEvent">
    <slot></slot>
  </Entity>
</template>
```

En la sección de template añadiremos un componente base llamado Entity, capaz de convertir todo en una entidad de aframe. Dentro de este podemos añadir tanto código A-Frame nativo como una representación de componente con la capa superficial de Vue.js, como propiamente se muestra en el ejemplo.

Todas las entidades que partan de la base del componente principal `Entity.vue`, tienen como propiedad `aprops`, que es el encargado de convertir un simple objeto en propiedades nativas de aframe para su inyección.

La parte de `<slot></slot>`, sirve para que, en un componente padre donde se utilice esta nueva entidad (componente en Vue.js), se puedan añadir mas componentes siguiendo la estructura lógica de HTML5.

En la parte de script, se añadirá lo básico, posteriormente pudiendose desarrollar:

```
<script>
import { ref, onMounted } from 'vue'
import Entity from '@/components/aframe/entities/Entity.vue'

export default {
  name: 'NewEntity',
  components: {
    Entity
  },
  props: {
    id: String,
    text: String,
```

```
      width: String,
      color: String,
      position: Object,
      items: Array
    },
    setup(props, { emit }) {
      const entityProps = ref({
        position: "0 4 0",
        rotation: "0 0 0",
        width: 5
      })
      const handleClickEvent = (event) => {
        console.log(event)
      }
      onMounted(() => {
      })

      return { entityProps, handleClickEvent }
    }
  }
  </script>
```

Las partes básicas son los componentes que vayamos a utilizar en el mismo, importando en la parte de arriba como se hace con `Entity.vue`, propiedades que podemos pasarle al propio componente para utilizarlas dentro y setup donde tenemos las funciones y los manejos de la propia funcionalidad del mismo.

4.3.5. Funcionalidades de A-Frame

Una vez teniendo nuestro primer componente creado, se añadirá una lógica y un componente nativo de A-Frame para utilizarlo en el mismo.

En el paquete `src/components/aframe/components` estarán ubicados los archivos corresponientes en JavaScript para registrar los componentes nativos de A-Frame. Por ejemplo,

en el archivo `src/components/aframe/components/helloWorld.js` tendremos lo siguiente:

```
AFRAME.registerComponent('hello-world', {
  schema: {},
  multiple: false,
  init: function () {
    console.log('componente iniciado correctamente')
  },
  update: function (oldData) {
    console.log(oldData)
  },
  remove: function () { },
  // tick: function (t) { },
  pause: function () { },
  play: function () { },
  events: {
  }
})
```

Para poder utilziar este componente en cualquier parte de la aplicación, nos desplazamos al archivo principal `main.js` ubicado en la raíz del proyecto e importamos el archivo JavaScript previamente creado:

```
import '@/components/aframe/components/helloWorld.js'
```

Para utilizar este componente dentro de una entidad, básicamente nos iremos al objeto de propiedades que mandamos a las entidades con `aprops` y lo añadimos.

Capítulo 5

Conclusiones

En esta sección, se visualizarán los objetivos presentados al principio del proyecto y su grado de cumplimiento, así como problemas encontrados a lo largo de este.

Con respecto al objetivo general, la aplicación realizada cubre con el objetivo propuesto, la detección de poses por un lado, con la base principal de WebXR Hand, creándose componentes en A-Frame para detectar y guardar poses. Por otro lado, la integración entre los dos frameworks Vue.js y A-Frame se realizó de manera satisfactoria, siendo los componentes reutilizables y facil de implementar a nivel de desarrollador. Podríamos indicar que se ha cumplido lo siguiente de forma resumida:

- Detección y guardado de poses en realidad virtual
- Integración de frameworks Vue.js y A-Frame

Por otro lado, tenemos los objetivos específicos.

Se evaluaron las posibles tecnologías que existen dentro del contexto de A-Frame, siendo como mejor opción directamente desarrollar basándose en el componente de mas bajo nivel WebXR Hand, cubriendo también las necesidades básicas de conocimiento con respecto a esta tecnología y todas sus posibilidades.

Con respecto a la integración de frameworks, la aplicación cumple los requisitos para realizar escenarios, componentes, y entidades de manera compacta y sencilla, así como la reutilización del código y de los propios componentes en diferentes contextos.

En la escena de detección de posees, se puede demostrar el potencial de la interacción de las manos mediante poses y gestos en una escena de realidad virtual, cumpliendo también con

este objetivo específico definido al comienzo del proyecto.

Gracias a Vue.js y como podemos observar, se facilita la mezcla de una misma página web que nos sirve de alojamiento para todas las escenas implementadas con A-Frame de base. Además, todas las fucnionalidades y componentes están bajo el estandar WebXR, cumpliendo así el último de los objetivos específicos, para garantizar la compatibilidad y accesibilidad en multiples dispositivos y plataformas de realidad virtual.

5.1. Aplicación de lo aprendido

A lo largo del grado, he adquirido conocimientos en diversas áreas, entre ellas la programación en diferentes asignaturas. Esto me ha permitido desarrollar este proyecto, aplicando lo aprendido y adquiriendo las habilidades necesarias para su realización. Entre las asignaturas mencionadas anteriormente, destacan:

1. **Fundamentos de la programación**: Esta asignatura fue mi primera toma de contacto con el mundo de la programación. Aquí senté las bases para un conocimiento técnico fundamental en programación.

2. **Programación de sistemas de telecomunicación**: En esta asignatura aprendí aplicaciones básicas de programación utilizando Python. Desarrollé habilidades para escribir scripts eficientes y realizar tareas de procesamiento de datos.

3. **Sistemas operativos**: El curso se centró en la programación en C y el uso de Bash. Aprendí a gestionar el control de errores y a entender el funcionamiento interno de los sistemas operativos.

4. **Ingeniería de sistemas de información**: Enfocado en la programación orientada a objetos con Java, esta asignatura proporcionó fundamentos básicos y esenciales para el desarrollo de software complejo.

5. **Servicios y aplicaciones telemáticas**: Aprendí a utilizar JavaScript y Python con Flask para entender cómo funcionan los frameworks y desarrollar aplicaciones web. También adquirí habilidades en la manipulación del DOM, estructuras HTML5 y control de eventos en el navegador.

6. **Ingeniería de sistemas de información**: Este curso avanzó en el uso de Java con frameworks para desarrollar aplicaciones web sencillas siguiendo el patrón MVC. Además, aprendí a utilizar GitHub para subir y mantener el código de manera colaborativa.

En definitiva, las asignaturas relacionadas con la programación han aportado gran valor a nivel de conocimiento, ayudando de manera directa a la realización de este proyecto.

5.2. Lecciones aprendidas

A lo largo del desarrollo de este proyecto, he tenido la oportunidad de aplicar los conocimientos adquiridos en diversas asignaturas como las mencionadas anteriormente, proporcionando las bases necesarias y la capacidad de enfrentar y resolver los desafíos presentes en el proyecto. Gracias a este Trabajo de Fin de Grado, se han adquirido nuevos concomientos de diferentes areas y mejorado los que ya tenía, obteniendo una serie de lecciones aprendidas:

1. En el contexto de A-Frame y el mundo de la realidad virtual, aprendí a crear escenas inmersivas e interacciones con el usuario. Esto incluyó el control de poses y el manejo de las manos en entornos virtuales, así como el manejo del DOM y la manipulación de elementos HTML para crear experiencias interactivas. La integración de estos elementos me permitió entender mejor cómo los usuarios pueden interactuar con objetos virtuales y cómo diseñar experiencias más intuitivas y naturales.

2. Con Vue.js y su integración en el proyecto, he aprendido a estructurar y desarrollar aplicaciones de forma eficiente. Vue.js me permitió crear interfaces de usuario reactivas y manejar el estado de la aplicación de manera efectiva, mejorando la interacción en las aplicaciones de realidad virtual.

3. He mejorado mi uso de GitHub como herramienta de desarrollo manteniendo el código limpio y organizado, utilizando ramas para manejar diferentes características y flujos de trabajo.

4. El uso de LaTeX para la documentación del proyecto me ha permitido producir documentos técnicos de alta calidad. Aprendí a estructurar la memoria del proyecto de

manera clara y profesional, mejorando la presentación y legibilidad del contenido.

5.3. Trabajos futuros

El desarrollo de este proyecto ha abierto numerosas oportunidades para futuras mejoras y ampliaciones. A continuación, se presentan algunas mejoras y complementos que podrían implementarse en el futuro:

- **Desarrollo de componentes Vue.js para A-Frame**: Se podrían crear más componentes que cubran las funcionalidades básicas de A-Frame y convertirlos en nativos de Vue.js ya que facilitaría la integración y reutilización de estos componentes en otras aplicaciones, simplificando el desarrollo y mejorando la consistencia del código.
- **Mejora en el escenario de poses**: Se podría implementar un teclado virtual que permita guardar el nombre específico de cada pose detectada. Esto añadiría un nivel adicional de personalización y organización, haciendo que el sistema sea más intuitivo.
- **Enlace de poses a acciones específicas**: Podría desarrollarse la funcionalidad de asociar una pose a una acción concreta. Por ejemplo, al realizar el gesto de un número, un objeto específico podría aparecer en pantalla para que el usuario interactúe con él.
- **Mayor precisión en la detección de poses**: Se podrían mejorar los algoritmos de reconocimiento de poses y la sensibilidad de las detecciones para hacer las interacciones más naturales y precisas. Esto permitiría una experiencia de usuario más realista.
- **Integración con tecnologías de realidad aumentada**: Explorar la integración con otros dispositivos y tecnologías de realidad aumentada permitiría que las poses y gestos se utilicen en entornos AR, ampliando las aplicaciones posibles que ofrece el proyecto.

Bibliografía

[1] Fundamentals of WebXR. https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals.

[2] Scrum guide. https://scrumguides.org/scrum-guide.html.

[3] WebGL Fundamentals. https://web.archive.org/web/20121130002131/http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals.

[4] WebXR Hand Input Module - Level 1. https://www.w3.org/TR/webxr-hand-input-1.

[5] What is A-Frame. https://aframe.io/docs/1.6.0/introduction.

[6] What is Vue. https://vuejs.org/guide/introduction.html.

[7] J. Dirksen. *Learn Three.js*. Packt Publishing, 4th edition, 2023.

[8] D. Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, 7th edition, 2020.

[9] P. P. . J. Loeliger. *Version Control with Git*. O'Reilly Media, 3rd edition, 2023.

[10] D. Rigby. *Doing Agile Right: Transformation Without Chaos*. Harvard Business Review Press, illustrated edition, 2020.