Universidad
Rey Juan Carlos

ESCUELA DE INGENIERÍA DE FUENLABRADA

INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

INTERFAZ DE USUARIO BASADO EN SEGUIMIENTO DE MANOS PARA REALIDAD EXTENDIDA

Autor : José Antonio Bejarano Sigüenza

Tutor : Jesús María González Barahona

Curso académico 2023/2024

Trabajo Fin de Grado

Interfaz de basado usuario basado en seguimiento de manos para realidad extendida

Autor : José Antonio Bejarano Sigüenza
Tutor : Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2024, siendo calificada por el siguiente tribunal:

Presidente:

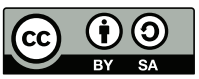
Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2024



Dedicado a mi familia,
mis hermanos,
mis amigos,
pero sobre todo a mi madre

Agradecimientos

Han sido muchos años de altibajos, de momentos que parecía que no lo iba a conseguir, momentos que estuve cerca de abandonar, pero al final aquí estoy escribiendo la memoria de mi TFG para dar por término esta etapa de mi vida en cuál tengo que agradecer a la gente que ha estado a mi lado, pues si ellos esto no sería posible.

En primer lugar, y más importante a mi Madre, por ser la persona que más cerca ha estado de mí estos años, apoyándome e insistiendo en estudiar más para poder lograrlo y por supuesto todo el esfuerzo económico que conlleva estudiar una carrera universitaria. A mi hermano, mi hermana, mi padre y Vero por su apoyo durante todo este tiempo. Gracias a toda mi familia por demostrarme que aunque falle siempre estarán ahí para apoyarme incondicionalmente.

A mis compañeros, que ahora son mis amigos, este no ha sido un camino que he hecho solo, sin ellos y su fuerza no podría haberlo logrado. Hemos pasado juntos por momentos buenos y momentos complicados, pero siempre hemos estado juntos, pase lo que pase.

También a mi tutor Jesús María González Barahona por dedicar tanto tiempo a guiarme en este proyecto, sin él no lo hubiera podido llevar a cabo.

Y por último, pero no menos importante, quiero agradecerme a mí mismo por creer en mí, por hacer todo el trabajo duro que he tenido que hacer para llegar hasta este momento, por no rendirme y, en definitiva, por confiar en mí.

Resumen

Este proyecto se enfoca en la construcción de un conjunto de componentes para crear menús interactivos en realidad virtual, utilizando el seguimiento de manos como método exclusivo de interacción. La importancia radica en que permite la interacción sin controladores, haciendo uso únicamente de las manos. Esto fue posible gracias a la reciente disponibilidad de la API de seguimiento de manos (WebXR) en los navegadores web, que garantiza la compatibilidad de esta tecnología en cualquier dispositivo que soporte WebXR. Además, dispositivos de bajo coste han comenzado a incorporar el hardware necesario para esta funcionalidad. Para comprobar la integración y funcionalidad de los componentes, se desarrollaron tres interfaces de usuario.

Las interfaces de usuario propuestas se basan en menús tridimensionales, los cuales tendrán objetos que el usuario puedo coger con la mano y de esta forma interactuar con el entorno virtual.

El desarrollo del proyecto se llevó a cabo utilizando el framework A-Frame, que facilita el acceso a la API de WebXR y asegura que las interfaces funcionen en el navegador. Esto simplifica la creación de escenas de realidad virtual mediante JavaScript y HTML.

Summary

This project focuses on the construction of a set of components to create interactive menus in virtual reality, using hand tracking as the exclusive method of interaction. The importance lies in the fact that it allows interaction without controllers, using only the hands. This was made possible by the recent availability of the hand tracking API (WebXR) in web browsers, which ensures compatibility of this technology on any device that supports WebXR. In addition, low-cost devices have started to incorporate the necessary hardware for this functionality. To test the integration and functionality of the components, three user interfaces were developed.

The proposed user interfaces are based on three-dimensional menus, which will have objects that the user can pick-up by hand and thus interact with the virtual environment.

The development of the project was carried out using the A-Frame framework, which facilitates access to the WebXR API and ensures that the interfaces work in the browser. This simplifies the creation of virtual reality scenes using JavaScript and HTML.

Índice general

Índice de figuras

Capítulo 1

Introducción

La realidad virtual ha sido objeto de investigación y desarrollo durante varias décadas. El término "realidad virtual" fue acuñado en 1987 por Jaron Lanier. Desde su creación, la realidad virtual se ha empleado en aplicaciones médicas, simuladores de vuelo, diseño industrial y el ámbito militar, evidenciando su capacidad en la educación, la medicina, la arquitectura y el diseño.[1]

La realidad virtual se encuentra en una fase de auge. Las gafas Oculus Rift, introducidas en el mercado en 2010, impulsaron el desarrollo de esta tecnología debido a sus innovadoras capacidades técnicas en dispositivos portátiles. Originalmente diseñadas para videojuegos, la compra de Oculus Rift por Facebook en 2014 motivó a otras compañías, como Google y Sony, a desarrollar tecnologías de realidad virtual enfocadas en el entretenimiento.

En los últimos dos años, dispositivos como Meta Quest y PICO han experimentado una notable evolución. Meta Quest, conocido por su sistema de realidad virtual independiente, ha mejorado significativamente en términos de rendimiento y accesibilidad, incorporando sistemas de seguimiento de manos basados en cámaras y tecnologías de machine learning. Estos avances han permitido a los usuarios interactuar de manera más intuitiva y precisa en entornos virtuales sin necesidad de controladores físicos. Por otro lado, PICO ha emergido como un competidor fuerte en el mercado de la realidad virtual, ofreciendo dispositivos asequibles y de alta calidad con características similares de seguimiento de manos y capacidades autónomas, ampliando el acceso a experiencias de realidad virtual inmersivas a un público más amplio.

En febrero de 2024, Apple lanzó las Apple Vision Pro, unas gafas también de alto costo con características destacadas como la interacción directa mediante gestos manuales en una interfaz de usuario que integra elementos del mundo real con menús virtuales, permitiendo su uso tanto en entornos domésticos como al aire libre. Estas innovaciones demuestran el continuo crecimiento y diversificación del mercado de la realidad virtual y aumentada, ofreciendo opciones que van desde el entretenimiento hasta aplicaciones profesionales altamente especializadas.

Este trabajo de fin de grado (TFG) se centra en la exploración de la API de seguimiento de manos (WebXR) en el navegador. Utilizando A-Frame, el proyecto crea experiencias de realidad

virtual que permiten la interacción sin controladores, mejorando la inmersión y la accesibilidad del usuario. Se desarrollaron prototipos para demostrar la funcionalidad de estas interfaces, mostrando cómo las escenas interactivas pueden facilitar una interacción más natural y efectiva.

1.1. Objetivo principal

El objetivo es construir una biblioteca de componentes que utilicen el reconocimiento de manos, funcionando en realidad virtual y permitiendo la interacción de las aplicaciones.

1.2. Objetivo instrumentales

Para conseguir el objetivo principal, he tenido que completar los siguientes objetivos instrumentales:

1. **Creación de Menús y Submenús**: Desarrollar una estructura de menús y submenús que permita al usuario navegar y seleccionar opciones de manera intuitiva y fácil.
2. **Construir componentes de seguimientos de manos**: Estos componentes están basados en el componente hand-tracking de A-Frame, utilizan este componente por debajo.
3. **Desarrollar Componentes con A-Frame**: Crear y personalizar componentes en A-Frame para construir interfaces de usuario eficientes y efectivas con menús.
4. **Creación prototipos**: Crear demostradores en realidad virtual que usen mis componentes para comprobar la funcionalidad.

1.3. Objetivos Relacionados

Aunque no llevan directamente al objetivo principal, existen ciertos objetivos relacionados que son importantes para el desarrollo integral de este proyecto. Estos objetivos complementarios permiten enriquecer la experiencia del usuario y asegurar que las interfaces desarrolladas sean completas y funcionales. Entre estos objetivos se incluyen:

1. **Funcionar en navegadores**: Desarrollar componentes de realidad virtual que funcionen directamente en navegadores web, garantizando así su portabilidad y accesibilidad.
2. **Herramientas similares a front-end**: Desarrollar componentes utilizando herramientas y tecnologías de front-end, específicamente JavaScript.
3. **Utilizar A-Frame**: Utilizar el Framework A-Frame para desarrollar la interfaz de usuario en realidad virtual porque permite acceder a la API de seguimiento de manos (WebXR).

4. **Enfoque en la Usabilidad de los Componentes**: Optimizar la usabilidad de la interfaz mediante pruebas con usuarios, asegurando que sea intuitiva y efectiva.

5. **Personalización de la interfaz**: Desarrollar herramientas dentro de la interfaz que permitan al usuario modificar diversas características del entorno virtual. Estas herramientas incluirán la capacidad de cambiar el color, tamaño y posición de las entidades presentes en la escena.

6. **Edición de Entornos Virtuales**: Posible uso de mi biblioteca, de esta manera se puede demostrar con aplicaciones razonablemente reales las capacidades del sistema construido.

1.4. Estructura de la memoria

La memoria está organizada de la siguiente manera:

El capítulo 2 proporciona una descripción detallada de las tecnologías empleadas en el proyecto, tanto directa como indirectamente. El capítulo 3 explica el proceso de desarrollo del proyecto, incluyendo la planificación temporal y las etapas clave. En el capítulo 4 se analiza los resultados, ofreciendo descripciones tanto para el usuario final que utiliza las interfaces como para usuarios técnicos que deseen comprender y modificar su funcionamiento. Finalmente, el capítulo 5 presenta las conclusiones del trabajo de fin de grado.

1.5. Disponibilidad de código y otra información

He realizado una página web para el trabajo final de grado en esta URL jabejarano.github.io y el repositorio del código fuente se encuentra aquí https://github.com/jabejarano/TFG-AFRAME

Capítulo 2

Tecnologías utilizadas y estado del arte

En este capítulo se detallarán las principales tecnologías utilizadas para el desarrollo del proyecto. En primer lugar, se describirá el dispositivo utilizado en el proyecto. A continuación, se explicará el seguimiento de manos y la forma en la que esta tecnología representa las manos, dependiendo de cada dispositivo. Luego, se abordará el software utilizado, comenzando con A-Frame, Framework de código abierto que permite el desarrollo de escenas en realidad virtual construido sobre, Three.js, y finalmente WebXR, una tecnología crucial para el seguimiento de manos en A-Frame. Por último, se comentarán otras tecnologías auxiliares, como WebGL, utilizada para renderizar gráficos 3D en el navegador; JavaScript, el lenguaje de programación principal para la lógica de la aplicación y la manipulación de elementos VR; HTML5, que proporciona la estructura y base de las páginas web; Visual Studio Code, el IDE utilizado para el desarrollo del proyecto; y LaTeX, empleado para escribir la memoria del proyecto.

2.1. Meta Quest

Oculus VR, fundada en 2012, fue adquirida por Facebook en 2014, lo que impulsó el desarrollo de su tecnología de realidad virtual. En 2016, Oculus lanzó sus primeras gafas de realidad virtual, las Oculus Rift, que requerían conexión a un ordenador. Con el tiempo, la empresa evolucionó hacia dispositivos autónomos con el lanzamiento de la serie Oculus Quest en 2019. Estas gafas autónomas, que no requieren conexión a ordenadores o consolas, permiten a los usuarios sumergirse en mundos virtuales de manera más accesible. En 2021, como parte de una reestructuración más amplia, Facebook cambió su nombre corporativo a Meta, y las gafas de realidad virtual pasaron a llamarse Meta Quest, desarrolladas por Facebook Reality Labs.

La serie Quest se destaca por su capacidad de operar de manera autónoma, con un hardware integrado potente y un software optimizado que permite ejecutar una amplia gama de aplicaciones y juegos de realidad virtual directamente en el dispositivo. Uno de los principales avances introducidos por los dispositivos Meta Quest es su sistema de seguimiento posicional. Utilizando múltiples cámaras externas, estos dispositivos pueden rastrear los movimientos del usuario

en el espacio físico sin necesidad de sensores externos adicionales.

Además, la serie Meta Quest incorpora tecnologías avanzadas de interacción, como el seguimiento de manos. Esta función permite a los usuarios interactuar con el entorno virtual utilizando solo sus manos, sin necesidad de controladores físicos. Al utilizar cámaras y algoritmos avanzados de visión por computadora y machine learning, las gafas Meta Quest pueden detectar y seguir con precisión los movimientos de las manos, facilitando una interacción más intuitiva y natural con los objetos virtuales.

Las Meta Quest 2 Lanzadas en octubre de 2020, están equipadas con un procesador Qualcomm Snapdragon XR2 y 6 GB de RAM, una pantalla LCD de alta resolución (1832 x 1920 píxeles por ojo) y la tasa de refresco de hasta 90 Hz. El sistema de seguimiento posicional, basado en cuatro cámaras externas, permite una inmersión completa y una interacción precisa con los controladores táctiles. Además, la opción de seguimiento de manos mejora la naturalidad de la interacción, eliminando la necesidad de controladores físicos en muchas aplicaciones.

Las Meta Quest 3, lanzadas posteriormente, incluyen una interfaz en realidad aumentada en color, a diferencia de sus antecesoras. Están equipadas con el procesador Qualcomm Snapdragon XR3. Estas gafas cuentan con una pantalla OLED de mayor resolución (2160 x 2160 píxeles por ojo) y una tasa de refresco de hasta 120 Hz. El campo de visión ha sido ampliado a 110 grados, mejorando la inmersión del usuario. El sistema de seguimiento posicional utiliza seis cámaras externas para una mayor precisión en el seguimiento de los movimientos. Además, el seguimiento de manos ha sido optimizado para reducir la latencia y mejorar la precisión, facilitando interacciones más naturales y precisas.

2.2. Seguimiento de manos

A principios de la década de 2010, se comenzaron a desarrollar los primeros sistemas utilizando sensores infrarrojos y cámaras para detectar y rastrear los movimientos de las manos y dedos. Uno de los pioneros en esta área fue Leap Motion, que en 2013 lanzó dispositivos específicos para el seguimiento de manos. Con el tiempo, la incorporación de algoritmos de visión por computadora y machine learning ha mejorado la precisión y la velocidad del seguimiento.

Las tecnologías de seguimiento de manos han sido integradas en dispositivos de grandes compañías, como Meta (anteriormente Facebook) y Microsoft. Los dispositivos Meta Quest, por ejemplo, utilizan cámaras integradas y la API Oculus Hand Tracking para proporcionar un seguimiento de manos preciso y en tiempo real. Del mismo modo, HoloLens de Microsoft emplea la API Windows Mixed Reality para el seguimiento de manos.

Las cámaras utilizadas en el seguimiento de manos son generalmente cámaras RGB o infrarrojas que pueden captar movimientos detallados de los dedos y las manos. Estas cámaras funcionan en conjunto con algoritmos de visión por computadora que analizan las imágenes

capturadas para identificar la posición y el movimiento de las manos. Los datos de las cámaras se combinan para crear un modelo tridimensional en tiempo real de las manos del usuario.

El seguimiento de manos en realidad virtual y aumentada se centra en rastrear la posición y orientación de las falanges de los dedos para proporcionar una representación precisa de las manos del usuario. Este proceso implica la identificación y seguimiento de múltiples articulaciones del esqueleto de la mano, conocidas como “skeleton joints”.

Una forma de representar las manos con seguimiento de manos sería la siguiente, cada articulación del esqueleto tiene un radio que es el radio de una esfera colocada en su centro para que toque aproximadamente la piel en ambos lados de la mano. Las articulaciones de la “punta” deben tener un radio no nulo adecuado para que las colisiones con la punta del dedo funcionen. Las implementaciones pueden desplazar el origen de la articulación de la punta para que pueda tener una forma esférica con radio no nulo.

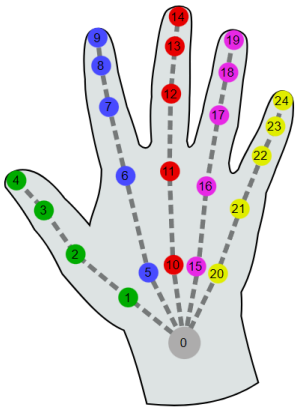


Figura 2.1: Mano enumerada con las articulaciones del esqueleto definidas

El seguimiento de manos permite una interacción natural sin controladores físicos, permitiendo a los usuarios manipular objetos y realizar gestos con precisión. Esta tecnología enriquece la experiencia inmersiva al integrar las manos en entornos virtuales, logrando que sea más intuitiva, directa y aumentando el realismo.

2.3. A-Frame

A-Frame[2][1] es un framework[2] web basado en código abierto, es una estructura de sistema de componentes de entidad para Three.js donde los desarrolladores pueden crear escenas 3D y WebVR usando HTML. A-Frame es compatible con la mayoría de los cascos de realidad virtual, como Vive, Rift, Windows Mixed Reality, Cardboard, Oculus Go, e incluso se puede

[1]Framework `https://aframe.io`

[2]Framework `https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/`

utilizar para realidad aumentada. A continuación se muestra un ejemplo básico de una escena en A-Frame:

```
<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>
```

Figura 2.2: Ejemplo inicial A-Frame versión HTML

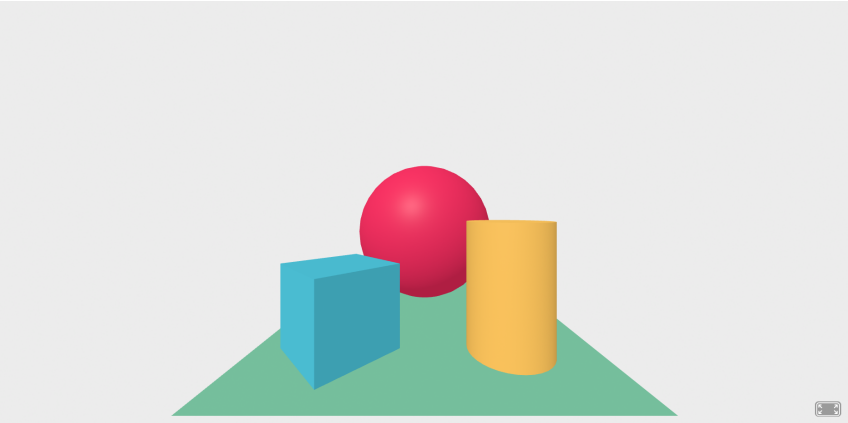


Figura 2.3: Ejemplo inicial A-Frame en el navegador corresponde con el HTML anterior

Las características más importantes de A-Frame son las siguientes:

1. **VR Simplificada:** Una de las ventajas más significativas de A-Frame es su capacidad para simplificar la creación de contenido VR. Solo necesitas añadir una etiqueta `<script>` y una `<a-scene>` en tu HTML para empezar. La simplicidad de A-Frame permite a los desarrolladores concentrarse en crear experiencias VR en lugar de preocuparse por la configuración técnica.

2. **HTML Declarativo:** A-Frame utiliza HTML para describir escenas y objetos en 3D, lo que lo hace accesible para cualquier persona familiarizada con el desarrollo web. HTML es fácil de leer y comprender, y su uso en A-Frame permite a los desarrolladores crear y manipular escenas VR de manera intuitiva.

3. **Arquitectura de Entidad-Componente:** A-Frame es una poderosa herramienta basada en Three.js que ofrece una estructura de entidad-componente que es declarativa, modular y reutilizable. Aunque se utiliza HTML para la configuración básica, los desarrolladores también pueden acceder completamente a JavaScript, a la API DOM, a Three.js, a WebVR y a WebGL para crear experiencias más complejas y personalizadas.

4. **VR Multiplataforma:** A-Frame permite el desarrollo de aplicaciones de realidad virtual compatibles con una amplia variedad de dispositivos, incluidos auriculares VR y sus respectivos controladores. Además, A-Frame es compatible con computadoras de escritorio y teléfonos inteligentes, lo que garantiza la accesibilidad de las experiencias de realidad virtual, incluso en ausencia de hardware especializado.

5. **Rendimiento Optimizado:** A-Frame está diseñado específicamente para WebVR desde el inicio. Aunque A-Frame utiliza el DOM, sus elementos no interfieren con el motor de diseño del navegador.

6. **Inspector Visual:** A-Frame incluye un inspector visual 3D integrado que facilita la inspección y depuración de escenas VR.

7. **Componentes:** Los componentes en A-Frame son una característica fundamental que permite la modularidad y reutilización de código en el desarrollo de experiencias de realidad virtual. Los componentes definen comportamientos específicos que se pueden adjuntar a las entidades dentro de la escena. Esta capacidad de agregar y combinar componentes facilita la creación de animaciones, luces, controles y otras funcionalidades complejas de manera eficiente y estructurada.

8. **Escalabilidad:** A-Frame permite desarrollar aplicaciones y escenas complejas sin perder rendimiento gracias a su arquitectura de componentes reutilizables y modulares.

2.4. Three.js

Three.js[3][3] es una biblioteca JavaScript utilizada para crear y renderizar gráficos en 3D en un navegador, simplifica considerablemente la creación de contenido 3D interactivo.

Esta biblioteca proporciona una abstracción de alto nivel sobre WebGL, permitiendo a los desarrolladores crear escenas complejas con modelos 3D, luces, sombras, efectos especiales y mucho más, utilizando una sintaxis más amigable y accesible. Es ampliamente utilizada en aplicaciones de visualización, juegos, simulaciones y experiencias interactivas en línea debido a su potencia y versatilidad.

A diferencia de WebGL, que es una API bastante técnica y de bajo nivel, Three.js ofrece una serie de clases, métodos y utilidades que facilitan la creación y manipulación de objetos 3D. Esto incluye geometrías (como cubos, esferas, cilindros), materiales (texturas, colores, reflejos), cámaras (para ver la escena desde diferentes perspectivas) y controles de interacción (para mover y girar objetos en la escena).

Una de las fortalezas de Three.js radica en su capacidad para manejar la complejidad del rendering 3D de una manera optimizada para el navegador. Utiliza técnicas como la culling

[3]Sitio Web oficial `threejs`

de objetos (solo renderiza los objetos visibles en la pantalla), optimización de shaders (para mejorar el rendimiento de los efectos visuales) y soporte para técnicas avanzadas de iluminación y sombreado, lo cual es fundamental para crear experiencias inmersivas y realistas.

A-Frame, por otro lado, es una biblioteca construida sobre Three.js que simplifica aún más la creación de contenido 3D utilizando HTML declarativo. A-Frame abstrae gran parte de la complejidad de Three.js al permitir a los desarrolladores crear escenas 3D utilizando etiquetas HTML estándar, lo que facilita la creación de realidad virtual y aplicaciones interactivas sin necesidad de profundos conocimientos en WebGL o Three.js.

En resumen, Three.js y A-Frame representan diferentes niveles de abstracción y complejidad en la creación de contenido 3D para la web. Three.js es una herramienta poderosa para desarrolladores que necesitan un control profundo sobre los gráficos 3D en el navegador, mientras que A-Frame es una opción más accesible y rápida para crear experiencias 3D utilizando HTML y aprovechando la potencia de Three.js en segundo plano.

2.5. WebXR

WebXR[4] es la API del navegador diseñada para proporcionar experiencias de realidad virtual (VR) y realidad aumentada (AR). Su funcionamiento depende del soporte de software y hardware subyacente del dispositivo en uso. WebXR ofrece una API estándar que puede ser utilizada en cualquier navegador compatible, pero se basa en las APIs específicas proporcionadas por cada dispositivo. Esto significa que WebXR trabaja con la API nativa del dispositivo para ofrecer una interfaz unificada y consistente.

Cada dispositivo de seguimiento de manos utiliza una API diferente para gestionar el hardware y los datos de seguimiento. Por ejemplo, los dispositivos Meta Quest utilizan la API Oculus Hand Tracking, mientras que los HoloLens de Microsoft emplean la API Windows Mixed Reality.

WebXR unifica estas diversas APIs bajo una interfaz estándar, permitiendo que las aplicaciones construidas sobre WebXR funcionen sobre cualquier dispositivo compatible. Esto facilita a los desarrolladores la creación de aplicaciones VR y AR sin tener que preocuparse por las diferencias específicas de cada hardware, proporcionando una mayor portabilidad y accesibilidad de las aplicaciones.

A-Frame utiliza la API WebXR para integrar interacciones y experiencias inmersivas directamente en el navegador web. Los desarrolladores pueden definir escenas y objetos 3D utilizando el elemento `<a-entity>` de manera declarativa, y luego manipular y animar estos elementos utilizando JavaScript(Explico este lenguaje en la sección 2.7).

Por último, la forma que he mostrado en la figura 2.1 es como lo hace WebXR[5] y esta tabla muestra el nombre de cada una de esas enumeraciones.

Skeleton joint		Skeleton joint name	Index
Wrist		wrist	0
Thumb	Metacarpal	thumb-metacarpal	1
	Proximal Phalanx	thumb-phalanx-proximal	2
	Distal Phalanx	thumb-phalanx-distal	3
	Tip	thumb-tip	4
Index finger	Metacarpal	index-finger-metacarpal	5
	Proximal Phalanx	index-finger-phalanx-proximal	6
	Intermediate Phalanx	index-finger-phalanx-intermediate	7
	Distal Phalanx	index-finger-phalanx-distal	8
	Tip	index-finger-tip	9
Middle finger	Metacarpal	middle-finger-metacarpal	10
	Proximal Phalanx	middle-finger-phalanx-proximal	11
	Intermediate Phalanx	middle-finger-phalanx-intermediate	12
	Distal Phalanx	middle-finger-phalanx-distal	13
	Tip	middle-finger-tip	14
Ring finger	Metacarpal	ring-finger-metacarpal	15
	Proximal Phalanx	ring-finger-phalanx-proximal	16
	Intermediate Phalanx	ring-finger-phalanx-intermediate	17
	Distal Phalanx	ring-finger-phalanx-distal	18
	Tip	ring-finger-tip	19
Little finger	Metacarpal	pinky-finger-metacarpal	20
	Proximal Phalanx	pinky-finger-phalanx-proximal	21
	Intermediate Phalanx	pinky-finger-phalanx-intermediate	22
	Distal Phalanx	pinky-finger-phalanx-distal	23
	Tip	pinky-finger-tip	24

Figura 2.4: Lista de articulaciones del esqueleto y su orden

WebXR utiliza una serie de articulaciones del esqueleto (skeleton joints) para identificar y rastrear las posiciones de diferentes partes de la mano. Cada articulación se puede identificar de manera única mediante un nombre de articulación, que es un enumerado del tipo XRHand-Joint. Una articulación puede tener un hueso asociado, utilizado para orientar su eje -Z, y este hueso es el que sigue a la articulación cuando se mueve hacia las puntas de los dedos. Las articulaciones de las puntas de los dedos y de la muñeca no tienen huesos asociados. Además, cada articulación tiene un radio que representa el tamaño de una esfera colocada en su centro, tocando aproximadamente la piel de ambos lados de la mano. Las articulaciones de la punta de los dedos deben tener un radio adecuado para que las colisiones con la punta funcionen correctamente.

2.6. WebGL

WebGL[6] es una tecnología basada en OpenGL ES que permite renderizar gráficos 3D interactivos dentro de los navegadores web utilizando la capacidad de procesamiento gráfico (GPU) del dispositivo del usuario. WebGL es un formato vectorial para poder describir es-

cenas 3D que funciona dentro del navegador. Esta tecnología es fundamental para desarrollar aplicaciones y juegos 3D inmersivos directamente en el navegador, sin depender de plug-ins adicionales.

WebGL[7] se basa en estándares como OpenGL ES 2.0 y ECMAScript, asegurando un amplio soporte y compatibilidad en diferentes plataformas y dispositivos.

Entre sus características destacadas se encuentra la interoperabilidad con otros estándares web como HTML, CSS y JavaScript, facilitando la integración fluida de gráficos 3D en aplicaciones web. Esta capacidad permite a los desarrolladores combinar fácilmente contenido visualmente atractivo con elementos interactivos y dinámicos. Además, WebGL es reconocido por su habilidad para el renderizado en tiempo real de gráficos 3D, esencial para aplicaciones que necesitan representar entornos virtuales de manera instantánea y dinámica, como juegos, simulaciones y visualizaciones interactivas. Esta característica permite que los objetos y escenarios 3D respondan rápidamente a las interacciones del usuario, proporcionando una experiencia inmersiva y fluida.

La combinación de WebGL[8] con A-Frame simplifica el desarrollo de aplicaciones VR/AR al proporcionar abstracciones de alto nivel sobre las complejidades de la programación gráfica. Esto hace que sea más accesible para desarrolladores de diferentes niveles de experiencia crear contenido 3D dinámico y visualmente atractivo para la web, impulsando así la adopción y expansión de experiencias inmersivas en línea.

2.7. JavaScript

JavaScript[9] es uno de los lenguajes de programación más utilizados a nivel mundial. Fue creado por Netscape en 1995 y desde entonces ha evolucionado para convertirse en uno de los pilares fundamentales de la programación web moderna. Su característica más destacada es su capacidad para crear aplicaciones interactivas en páginas web.

JavaScript[10] es un lenguaje de programación de alto nivel que se distingue por su flexibilidad y su integración nativa con HTML y CSS. Esto permite a los desarrolladores crear contenido dinámico e interactivo directamente en el navegador del usuario.

Dentro de un entorno, como un navegador web, JavaScript[11] se puede conectar a los objetos de su entorno para proporcionar control programático sobre ellos. Esto significa que en el navegador, JavaScript puede cambiar la apariencia de la página web (a través del DOM). Del mismo modo, el JavaScript de Node.js en el servidor puede responder a solicitudes personalizadas desde el código escrito en el navegador.

JavaScript contiene una biblioteca estándar de objetos, como Array, Date y Math, y un conjunto básico de elementos del lenguaje, como operadores, estructuras de control y declaraciones. El núcleo de JavaScript se puede extender para una variedad de propósitos completándolo con

objetos adicionales. Por ejemplo:

- **JavaScript del lado del cliente:** Extiende el núcleo del lenguaje al proporcionar objetos para controlar un navegador y su Modelo de Objetos de Documento (DOM). Esto permite a una aplicación interactuar con elementos HTML y responder a eventos del usuario, como clics del mouse, formularios para ingreso de datos y navegación de páginas.
- **JavaScript del lado del servidor:** Extiende el núcleo del lenguaje al proporcionar objetos relevantes para ejecutar JavaScript en un servidor. Esto permite que una aplicación se comunique con una base de datos, brinde continuidad de información de una invocación a otra de la aplicación o realice manipulación de archivos en un servidor.

Además, JavaScript se utiliza en diversas plataformas y dispositivos, desde navegadores web hasta servidores y aplicaciones móviles, gracias a su versatilidad y capacidad para integrarse con otras tecnologías y lenguajes de programación.

2.8. HTML5

HTML5[12] es la última versión del lenguaje de marcado para la creación de páginas web. Esta versión no solo mejora y expande las capacidades de HTML, sino que también introduce una serie de nuevas características y APIs que son fundamentales para el desarrollo de aplicaciones web avanzadas. Estas mejoras permiten a los desarrolladores crear sitios web y aplicaciones más dinámicas e interactivas.

En el contexto de este proyecto, HTML5 es esencial porque proporciona la base sobre la cual se integran y funcionan tecnologías clave como el motor de JavaScript y Frameworks como A-Frame(visto en la sección 2.3. Las nuevas APIs de HTML5 permiten una integración más eficiente y potente con estas tecnologías, facilitando la creación de experiencias interactivas y ricas en realidad virtual. Esta capacidad de integración es crucial para el desarrollo de interfaces de usuario avanzadas y aplicaciones de realidad virtual que forman el núcleo de este trabajo.

HTML[13] es esencial para cualquier desarrollo web moderno, proporcionando la base para la creación de páginas accesibles, estructuradas y visualmente atractivas en Internet. Mediante la combinación de elementos, atributos y semántica, los desarrolladores pueden crear sitios web robustos que cumplen con estándares internacionales y ofrecen una experiencia de usuario optimizada.

2.9. Visual Studio Code

Visual Studio Code (VSCode) [14] es un editor de código fuente desarrollado por Microsoft que se utiliza ampliamente en la programación y el desarrollo de software. VS Code es

conocido por su versatilidad y extensibilidad, lo que lo convierte en una herramienta ideal para desarrolladores de todos los niveles. Es compatible con una amplia variedad de lenguajes de programación y plataformas, y ofrece una experiencia de desarrollo altamente personalizable gracias a su amplio ecosistema de extensiones.

Una de las principales ventajas de Visual Studio Code es su interfaz de usuario intuitiva y su capacidad para manejar proyectos de software de cualquier tamaño y complejidad. Ofrece características avanzadas como depuración integrada, resaltado de sintaxis, completado de código inteligente, y control de versiones mediante Git. Además, su integración con numerosas extensiones permite a los desarrolladores adaptar el entorno de desarrollo a sus necesidades específicas, mejorando la productividad y facilitando el proceso de codificación.

Para la ejecución y depuración de la aplicación desarrollada en este proyecto, se instaló Visual Studio Code junto con la extensión ”Live Server”. Esta extensión permite lanzar un servidor local y ejecutar la aplicación en modo de depuración, proporcionando una experiencia de desarrollo más ágil y eficiente. Live Server facilita la visualización de cambios en tiempo real a medida que se edita el código, lo que es particularmente útil para el desarrollo de aplicaciones web, donde es importante ver cómo se renderizan los cambios de inmediato.

2.10. LaTex

LaTeX[15] es un sistema de composición de documentos de alta calidad que se utiliza ampliamente en la elaboración de documentos técnicos y científicos. Basado en el lenguaje de tipografía laTeX, creado por Donald Knuth, LaTeX permite a los usuarios centrarse en el contenido y la estructura de sus documentos, mientras el sistema se encarga del formato y la presentación tipográfica. LaTeX es especialmente popular en campos que requieren una presentación precisa de fórmulas matemáticas, gráficos y tablas, como las ciencias, la ingeniería y la informática.

Una de las principales ventajas de LaTeX[16] es su capacidad para manejar documentos largos y complejos de manera eficiente. Ofrece características avanzadas como referencias cruzadas automáticas, bibliografías gestionadas automáticamente, índices y generación de tablas de contenido. Además, LaTeX utiliza un enfoque basado en comandos para el formato, lo que permite una gran flexibilidad y control sobre el diseño del documento. Esta capacidad es particularmente útil para la creación de artículos científicos, tesis y libros técnicos, donde la precisión y la consistencia del formato son cruciales.

Para la redacción y el manejo del documento, se ha utilizado Overleaf, una plataforma colaborativa basada en la nube que facilita la creación de documentos en LaTeX. Overleaf proporciona un entorno de edición en línea que permite a múltiples usuarios trabajar simultáneamente en un mismo documento, haciendo el proceso de redacción más eficiente y colaborativo. Utilizar Overleaf en combinación con LaTeX ha permitido la creación de un documento técnico, con una presentación profesional.

Capítulo 3

Desarrollo del proyecto

En el desarrollo de este proyecto, se han aplicado algunos principios de metodologías ágiles[17], optando por un enfoque de desarrollo incremental y por fases. Cada fase concluía con un prototipo funcional, lo que permitía obtener retroalimentación y refinar las funcionalidades para el siguiente sprint. Este método facilitó una evolución constante del proyecto, asegurando que cada sprint incorporara mejoras basadas en los resultados y aprendizajes anteriores. Como todas estas técnicas se pueden encontrar en metodologías ágiles, voy a estructurar la explicación de este proyecto siguiendo una estructurar similar a la de las metodologías ágiles.

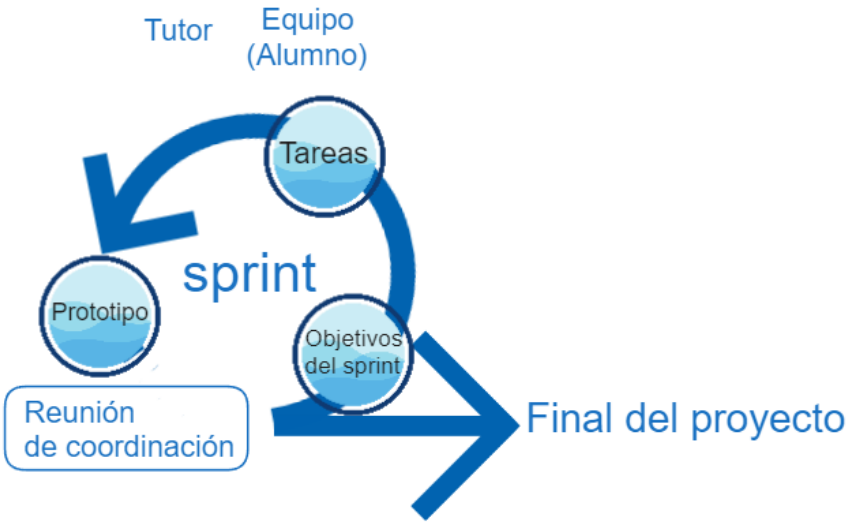


Figura 3.1: Esquema estructura proyecto

En mi proyecto, los roles se distribuyeron de la siguiente manera: el tutor del proyecto asumió las funciones de coordinar y orientar el desarrollo, mientras que yo, como autor del proyecto, fui responsable de la implementación y desarrollo de los objetivos.

Las metodologías ágiles tienen como característica que se organizan en sprints. Estos bloques de tiempo se dividen en periodos de 1 a 4 semanas y tienen como objetivo entregar algo funcional, en nuestro caso lo llamaremos prototipo.

Antes de cada sprint, realizamos una reunión de coordinación donde analizamos el prototipo realizado, revisamos los requisitos previamente discutidos y planificamos el siguiente sprint. Esta reunión es crucial para asegurar que el proyecto se mantiene en el camino correcto y para ajustar cualquier desviación en función de los resultados obtenidos y el feedback recibido.

En el resto de este capítulo se describe todo el proceso que ha llevado a los resultados de este trabajo, estructurado en función de los sprints. Cada sección detallará los objetivos, tareas, prototipos y detalles de cada sprint. Además, se comentarán los aprendizajes obtenidos durante el sprint y los problemas que se han resuelto, proporcionando una visión clara y ordenada del desarrollo incremental del proyecto.

3.1. Sprint 0

3.1.1. Objetivos del sprint

El propósito de este sprint es formativo: aprender y comprender los conceptos básicos de A-Frame. Durante este sprint, se construirán algunos componentes y escenas con ciertos niveles de interactividad para adquirir las habilidades necesarias para el desarrollo del proyecto.

3.1.2. Tareas

- **Aprender a construir un componente en A-Frame:** Familiarizarse con la sintaxis y estructura de A-Frame, comprendiendo cómo se definen y utilizan los componentes en esta tecnología.

- **Construir un componente con temporizador:** Consiste en crear algunos elementos interactivos, cajas, que cambian de color cuando se hace clic sobre ellas. Este componente utiliza eventos y temporizadores para modificar propiedades.

- **Construir un componente duplicador:** Se busca que el componente creado genere dos cubos. Además, permitirá la interacción del usuario con estos cubos, de manera que al hacer clic se creen y eliminen esferas al lado de ellos. Estas esferas serán elementos hijos de sus respectivos cubos, requiriendo una gestión adecuada de la jerarquía de objetos en la escena.

- **Construir un componente más complejo (columna de cubos):** Crear un componente que genere una columna de cubos apilados verticalmente con colores alternados. Los cubos deben ser interactivos, permitiendo al usuario crear y eliminar cubos secundarios adyacentes al cubo seleccionado mediante clic.

3.1.3. Prototipos finales

- **Construir un componente con temporizador:** Para completar la primera tarea, creé múltiples entidades en el HTML, representadas como cajas dispuestas en diversas posiciones dentro de la escena de A-Frame. Cada caja está vinculada al componente cambiador

  Para habilitar la interacción mediante clics, he utilizado "cursor–mouse=rayOrigin: mouse" en una entidad, lo que permite detectar la posición del puntero del ratón y activar eventos de interacción con objetos 3D en la escena a través del raycaster de A-Frame. El raycaster proyecta una línea invisible desde el cursor del ratón hacia los objetos en la escena, permitiendo que las entidades respondan a eventos de clic de manera efectiva.

```
<a-scene>
  <a-entity cursor__mouse = "rayOrigin: mouse"></a-entity>
  <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9" cambiador="color: red; timeout: 5000"></a-box>
  <a-box position="0 1.5 -5" rotation="0 45 0" color="#4CC3D9" cambiador="color: green; timeout: 3000"></a-box>
  <a-box position="1 0.75 -3" rotation="0 45 0" color="#4CC3D9" cambiador="color: blue; timeout: 1000"></a-box>
  <a-sky color="#ECECEC"></a-sky>
</a-scene>
```

Figura 3.2: Representación de los cubos con su componente cambiador

El componente acepta dos parámetros: *color* y *timeout* e incluye un "escuchador de eventos" (event listener) que se activa al hacer clic en cada cubo. Un "escuchador de eventos" es un mecanismo en la programación que espera a que ocurra una acción específica, como hacer clic con el ratón, y luego ejecuta una función en respuesta a esa acción. En este caso, al hacer clic en un cubo, el escuchador de eventos cambia su color después de un tiempo específico definido por el parámetro *timeout*.

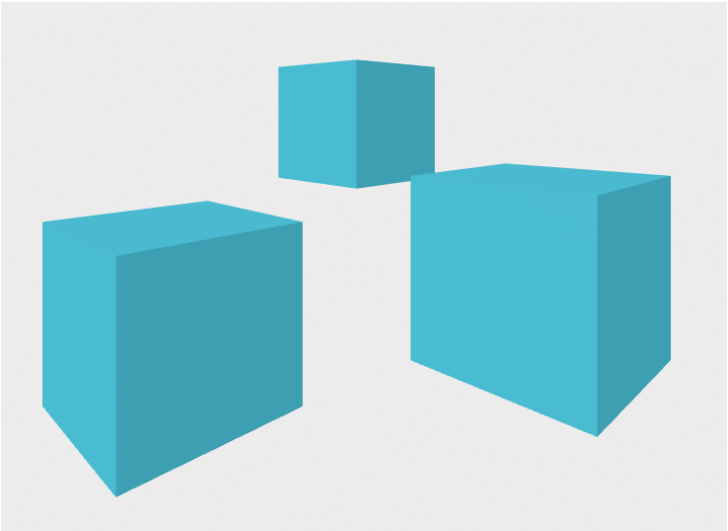


Figura 3.3: Prototipo componente con temporizador

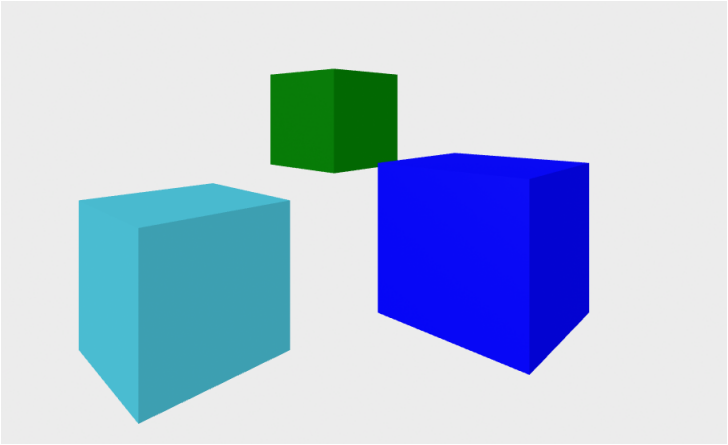


Figura 3.4: Prototipo después de hacer clic y esperar 5 segundos

- **Construir un componente duplicador:** En esta tarea el HTML es muy simple, ya que consta de una entidad en la cual se asocia el componente "duplicador" quien será el encargado tanto de crear los dos cubos en la escena como de gestionar el evento "click" en cada cubo para crear al lado de ellos una esfera, para ello se han añadido dos event listeners uno para cada cubo que verifican si una esfera hija está creada. Al hacer clic en un cubo, se crea una esfera con un color específico y al hacer clic nuevamente se elimina la esfera, alternando su existencia.

```
cubo1.addEventListener('click', () => {
    if (!esferaCreada1) {
        // Si la esfera no está creada, la creo
        var hijoDeCubo1 = document.createElement('a-sphere');
        hijoDeCubo1.setAttribute('position', '0.920 0 0.920');
        hijoDeCubo1.setAttribute('radius', '0.5');
        hijoDeCubo1.setAttribute('color', '#FF00FF');
        cubo1.appendChild(hijoDeCubo1);

        esferaCreada1 = true;
    } else {
        // Si la esfera está creada, la elimino
        var esfera = cubo1.querySelector('a-sphere');
        if (esfera) {
            cubo1.removeChild(esfera);
        }
        esferaCreada1 = false;
    }
});
```

Figura 3.5: Parte importante para crear las esferas hijas con el escuchador de eventos

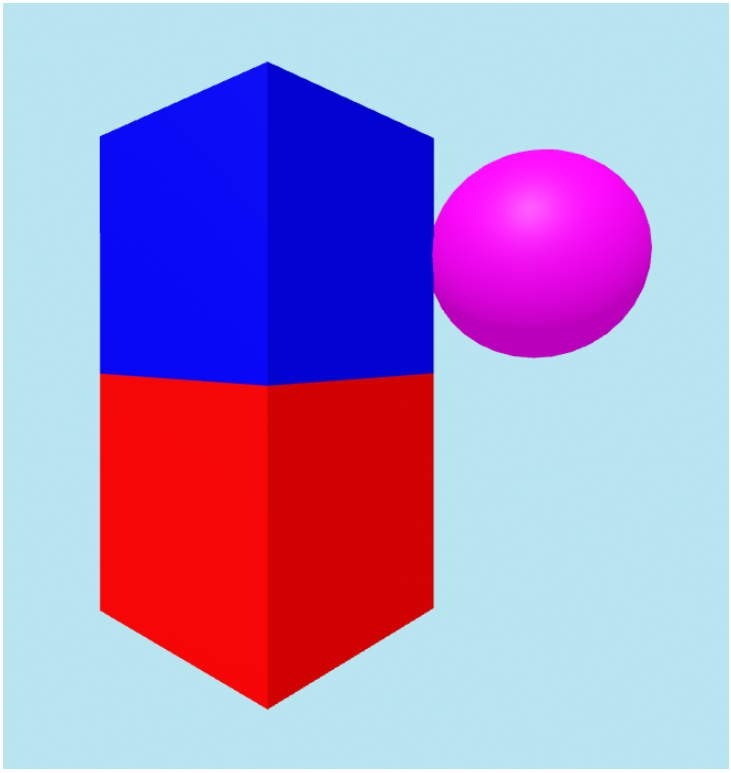


Figura 3.6: Prototipo duplicador

- **Constructor de columna de cubos:** Nuevamente, el HTML es muy sencillo, ya que solo consta de una entidad con el componente *cubos*, este será el encargado de crear dinámicamente cubos en la escena. Tiene dos parámetros *N* (número de cubos) y "colores" (el usuario puede elegir los colores que se le darán a los cubos)

```
AFRAME.registerComponent('cubos', {
    schema: {
        N: { type: 'number', default: 4 },
        colores: { type: 'array', default: ['red', 'blue', 'pink'] }
    },
```

Figura 3.7: Parámetros

La manera de crear los cubos dependiendo del parámetro N es mediante un bucle for, que es una forma sencilla de repetir una acción varias veces. En este caso, el bucle for crea un cubo por cada número desde 0 hasta N-1. Dentro del bucle:

1. Se selecciona un color de una lista de colores, rotando entre ellos.
2. Se crea un cubo y se le asignan propiedades como su posición y color.
3. El cubo se añade a la escena.

```
init: function () {
    var colores = this.data.colores;

    // Crear los cubos iniciales
    for (var i = 0; i < this.data.N; i++) {
        var color = colores[i % colores.length];
        var cubo = document.createElement('a-box');
        cubo.setAttribute('position', '0 ' + i + ' -3');
        cubo.setAttribute('rotation', '0 45 0');
        cubo.setAttribute('color', color);
        cubo.classList.add('box');
        this.el.appendChild(cubo);
```

Figura 3.8: Crear cubos iniciales

Para crear un cubo hijo al lado del padre mediante el evento *click* se gestiona de manera similar al anterior prototipo.

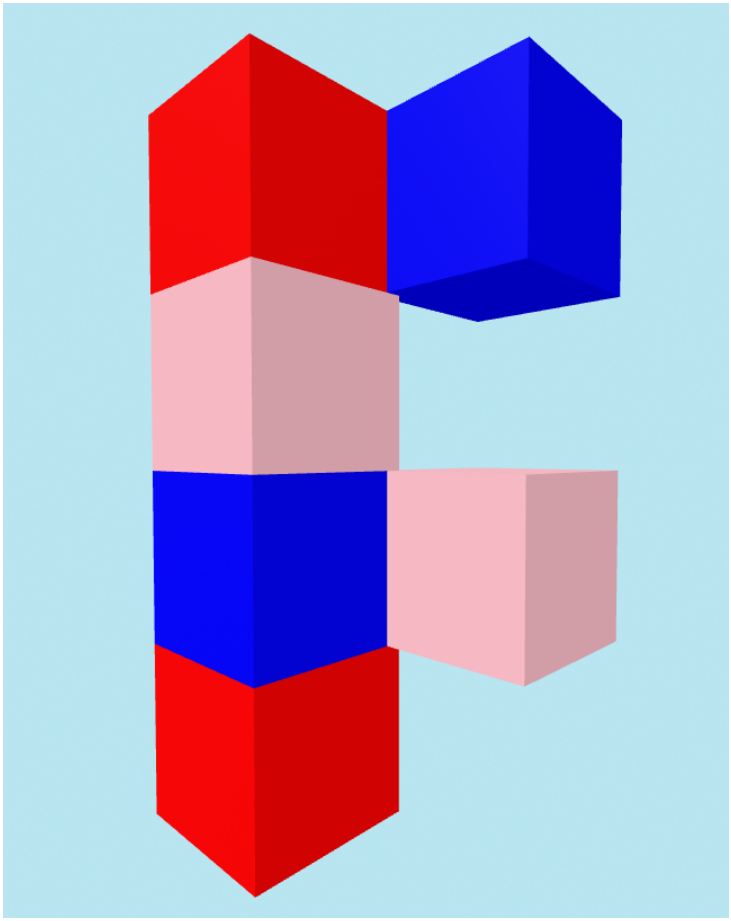


Figura 3.9: Columna de cubos

3.1.4. Detalles del Sprint

- **Uso de Componentes Personalizados**: Entender cómo registrar y utilizar componentes personalizados en A-Frame, lo cual es esencial para añadir comportamientos específicos a los elementos de la escena.

- **Familiarización con HTML y JavaScript**: Reforzar conocimientos previos de HTML y JavaScript, aplicándolos en un contexto práctico y moderno, como el desarrollo de aplicaciones de realidad virtual.
- **Manejo de Eventos**: Aprender a manejar eventos en A-Frame, como el evento de clic, para interactuar con los objetos en la escena. Esto incluye la implementación de escuchadores de eventos y la respuesta a las interacciones del usuario.
- **Despliegue de un entorno de Desarrollo**: Configurar un entorno de desarrollo eficiente usando Visual Studio Code y Live Server para previsualizar los cambios en tiempo real. Además, la gestión del código fuente con GitHub y la publicación del progreso usando GitHub Pages.

3.2. Sprint 1

3.2.1. Objetivos del sprint

El objetivo de este sprint es construir escenas donde la interacción esté controlada por las manos (seguimiento de manos) comprender cómo funciona el componente y familiarizarme con su integración y uso dentro del entorno de A-Frame.

3.2.2. Tareas

- **Modificar el ejemplo de seguimiento de manos que tiene A-Frame:** Esta tarea implica revisar y comprender el ejemplo de hand-tracking[1] proporcionado en A-Frame. Se debe explorar el código y los componentes utilizados, y luego experimentar con pequeñas modificaciones para ver cómo afectan el comportamiento de la escena.
- **Construir un menú de botones usando el seguimiento de manos:** Esta tarea se centra en aplicar el conocimiento adquirido del ejemplo de hand-tracking para crear un menú personalizado de botones interactivos.

3.2.3. Prototipos

Para realizar los prototipos de las tareas primero he tenido que comprender cómo están construidos los componentes del ejemplo de Hand-Tracking en A-Frame. Ejecuté el ejemplo en mis Meta Quest 2 para observar su funcionamiento (este ejemplo lo adjunto a pie de esta página). A continuación, revisé el código[2] HTML y JavaScript para entender como se implementa

[1]Ejemplo Hand-tracking `https://aframe.io/examples/showcase/handtracking/`

[2]Código Hand-tracking `https://github.com/aframevr/aframe/tree/master/examples/showcase/hand-tracking`

hand-tracking.

Lo más importante es el uso del componente *hand-tracking-controls*, que proporciona la entrada de seguimiento de la mano en aplicaciones de realidad virtual. Esto básicamente se hace en el HTML añadiendo dos entidades, una para cada mano. La explicación de este componente está en la documentación de A-Frame[18].

```
<a-entity hand-tracking-controls="hand: left"></a-entity>
<a-entity hand-tracking-controls="hand: right"></a-entity>
```

Figura 3.10: Añadir componentes hand-tracking

Los siguientes componentes se derivan del ejemplo de hand-tracking de A-Frame:

- **pinchable**: La función principal de este componente es permitir que los objetos reaccionen a los gestos de pellizco.

  Tiene un parámetro el cual define la distancia mínima *pinchDistance* a la que se debe estar para que el pellizco sea reconocido. Esta distancia actúa como un umbral que decide si el gesto es válido o no.

  El componente también incluye "escuchadores de eventos" igual que los que hice en el sprint anterior, pero esta vez en vez de utilizar el evento *click* estos escuchan ciertas acciones de la mano como el inicio *pinchstarted*, la finalización *pinchended*, y el movimiento *pinchmoved* de un pellizco. Estos eventos los gestiona y emite el propio componente "hand-tracking-controls" de A-Frame cuando los dedos índice y pulgar se unen.

  Estos eventos por sí solos no hacen nada, simplemente detectan un "pellizco". Cuando se emiten estos eventos, se activan funciones específicas que determinan la proximidad de un objeto a este pellizco. Si el objeto está lo suficientemente cerca, la función lo "coge" y sigue el movimiento del pellizco.

  El componente utiliza funciones de matemáticas en 3D para calcular la distancia entre los puntos donde se encuentran los dedos en el espacio virtual. Esto se hace copiando la posición de los dedos y calculando la distancia entre ellos. Esta medida es crucial para determinar si los dedos están lo suficientemente cerca para formar una pinza o no. Si los dedos están lo suficientemente cerca, se considera que la pinza está activa y se pueden realizar acciones basadas en este gesto.

  Profundizaré en la explicación de este componente en esta sección 4.3.3, ya que utilizaré este componente en las escenas finales del proyecto.

- **pressable**: La función principal de este componente es permitir que los objetos reaccionen a la presión de un dedo.

Tiene un parámetro, *pressDistance*, que define la distancia mínima a la que se debe estar para que la presión sea reconocida. Esta distancia actúa como un umbral que decide si el gesto de presión es válido o no.

El componente pressable utiliza escuchadores de eventos para detectar cuando un dedo (la punta del índice) está lo suficientemente cerca de un objeto. En la fase de inicialización (init), se obtiene la posición mundial del objeto y se busca todos los elementos que utilizan hand-tracking-controls. Durante cada fotograma (tick), se calcula la distancia entre la punta del dedo índice y el objeto. Si esta distancia es menor que *pressDistance*, se emite el evento *pressedstarted* si no estaba ya presionado, y se marca el objeto como *pressed*. Si la distancia es mayor, se emite el evento *pressedended* y se marca como no presionado.

El cálculo de la distancia se realiza mediante la función calculateFingerDistance, que mide la distancia entre la posición del objeto y la punta del dedo índice en el espacio virtual. Este cálculo es esencial para determinar si el dedo está lo suficientemente cerca para activar el gesto de presión.

- **button**: El componente button está diseñado para crear botones interactivos que pueden responder a eventos de presión. Este componente define varias propiedades: *label* (texto del botón), *width* (ancho del botón), *toggleable* (si el botón puede alternar entre estados presionado/no presionado), y *soundSrc* (fuente del audio a reproducir al presionar el botón)

  El método init inicializa el componente configurando la geometría y el material del botón, y creando una entidad de texto para mostrar la etiqueta. Se añade el componente pressable para que el botón pueda detectar eventos de presión. Además, se vinculan métodos al contexto del componente y se agregan escuchadores de eventos para manejar los cambios de estado del botón *stateadded*, *stateremoved*, *pressedstarted*, *pressedended*.

```
// Se vinculan los métodos a 'this' (el componente 'button')
this.bindMethods();
// Se agregan escuchadores de eventos para cambiar el estado del botón
this.el.addEventListener('stateadded', this.stateChanged);
this.el.addEventListener('stateremoved', this.stateChanged);
this.el.addEventListener('pressedstarted', this.onPressedStarted);
this.el.addEventListener('pressedended', this.onPressedEnded);
```

Figura 3.11: Vincular métodos a escuchadores de eventos

El método *onPressedStarted* se activa cuando se ha “tocado” con el dedo índice un botón y es en este método donde gestiono la geometría del botón y la cambio en el momento del toque para simular que está siendo apretado.

- **slider**: El componente slider está diseñado para permitir la interacción deslizante dentro de una interfaz 3D. Este componente cuenta con un parámetro *width* que determina el

ancho del slider. Durante la inicialización (init), se crean dos entidades principales que componen el slider: una pista y un selector.

He realizado un cambio en el componente slider para que, en lugar de utilizar un cilindro como elemento deslizante, se emplee un modelo GLTF. Esto permite mover un objeto 3D personalizado con un gesto de pellizco. El selector se implementa usando un modelo GLTF, que en este caso es un modelo de dinosaurio. Se ha ajustado la posición inicial y la escala del selector para que se integre adecuadamente en el diseño del slider. Además, se ha añadido el componente pinchable al selector para permitir la interacción mediante gestos de pellizco.

Cuando el selector es movido a través de un gesto de pellizco, se activa la función *onPinchedMoved*. Esta función antes actualizaba la posición del selector dentro del sistema de coordenadas local del slider, manteniendo el movimiento dentro de los límites del ancho del slider. Ahora he aplicado un cambio en esta función para permitir mover el objeto por toda la escena

```
// Copiar la posición global del selector al sistema de coordenadas local del slider
localPosition.copy(evt.detail.position);
el.object3D.updateMatrixWorld();
el.object3D.worldToLocal(localPosition);
// Actualizar la posición del selector en todas las coordenadas (X, Y, Z)
this.pickerEl.object3D.position.copy(localPosition);
```

Figura 3.12: Actualizar la posición del selector

- **menu:** El componente menú crea un fondo rectangular que se puede usar para organizar otros elementos de la interfaz en una escena de realidad virtual. Este fondo se define como una caja geométrica de color gris, y se posiciona detrás de la entidad principal a la que se ha agregado el componente. Este enfoque facilita la creación de menús visuales ordenados y accesibles dentro de experiencias de realidad virtual desarrolladas con A-Frame.

- **event-manager:** El componente event-manager gestiona la interacción del usuario con elementos de la escena mediante botones HTML. Al hacer "click" en un botón, se actualiza la visibilidad de las geometrías correspondientes y se puede alternar entre modos de iluminación. Este enfoque permite una manipulación dinámica y responsiva del entorno virtual, mejorando la experiencia del usuario.

Realizar un prototipo basado en el ejemplo de seguimiento de manos que tiene A-Frame

Para terminar de comprender cómo están construidos los componentes, realicé pequeñas variaciones en algunos. En el caso del componente Button, que añade el parámetro `soundSrc` para reproducir un sonido cuando se presiona un botón, también ajusta la profundidad del botón, reduciéndola cuando se presiona simulando que está siendo "presionado" y se restaura al soltar.

Por otro lado, el componente Slider en lugar de usar un cilindro para el selector utiliza un modelo GLTF (para las escenas finales de mi proyecto trabajaré con este tipo de modelos). Además, antes el selectro se movía únicamente por el eje X dentro de un rango específico, ahora la función `onPinchedMoved` permite al selector que se mueva por toda la escena, esto lo seguiré usando más adelante en mi proyecto a la hora de interactuar con menús.

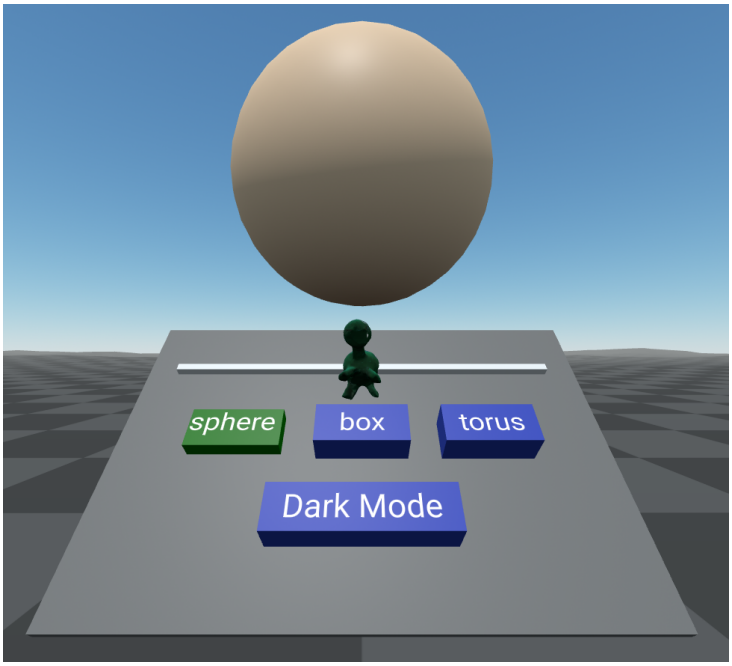


Figura 3.13: Prototipo variación ejemplo A-Frame

Construir un menú de botones usando el seguimiento de manos

Después construí mi propio menú con botones utilizando seguimiento de manos con los conceptos aprendidos en el anterior ejemplo:

Para este menú, he reutilizado varios componentes anteriores, como button.js, menu.js, pressable.js, y event-manager.js. Es un menú muy parecido con ligeros cambios como añadir un botón “dino” y se pinta un modelo GLTF en la escena.

Figura 3.14: Menú botones

3.2.4. Detalles del sprint

- **Interacción gestual:** Entendí cómo implementar interacciones gestuales como presionar y pellizcar en A-Frame, gracias a los componentes pressable y pinchable.
- **Método `bindMethods`:** Aprendí a utilizar `bind` para asegurar que las funciones del componente estén correctamente vinculadas al contexto (this), lo que es crucial para manejar eventos y mantener la coherencia de la lógica del componente.
- **Movimiento de GLTF por la Escena:** Aprendí cómo implementar el movimiento de un modelo GLTF por toda la escena en A-Frame utilizando eventos de gestos. Esto implica copiar la posición global del gesto al sistema de coordenadas local del objeto GLTF y actualizar su posición en todas las coordenadas (X, Y, Z), asegurando así una interacción fluida y precisa con elementos 3D en entornos de realidad virtual.

3.3. Sprint 2

3.3.1. Objetivos del sprint

El objetivo principal de este sprint es diseñar e implementar una interfaz de usuario en A-Frame que incluya menús interactivos. Estos menús permitirán a los usuarios agarrar objetos dentro de ellos, los cuales podrán desencadenar diferentes acciones o eventos dentro de la escena.

3.3.2. Tareas

- **Componentes para modelar los menús:** Crear los componentes necesarios para poder construir los menús y las opciones (objetos) que estos tendrán dé una manera escalable.
- **Componentes para modelar las opciones de los menús:** Crear componente para interactuar con los objetos de los menús y que desencadenen acciones en la escena.

3.3.3. Prototipo

Lo primero que he hecho ha sido reciclar el componente pinchable del anterior sprint (presente en el ejemplo de seguimiento de manos de A-Frame) este componente es clave para mi proyecto, puesto que al final la forma de interactuar con los menús será agarrando los objetos gracias a que la realidad virtual nos permite esta acción.

Los siguientes componentes los construí para esta tarea:

- **shelf:**El componente shelf organiza y gestiona los objetos que se muestran en los menús de la escena. Utiliza dos propiedades, `objects` y `names`, que son `arrays` conteniendo las referencias y nombres de los objetos a mostrar. En el método `init`, se validan estos `arrays` para asegurar que la cantidad de objetos coincide con la cantidad de nombres, mostrando un error en caso contrario. Si la validación es exitosa, el componente almacena los objetos permitidos en un atributo allowed-objects del elemento HTML y lo mismo hace con los nombres allowed-names, lo que facilita el acceso y manejo global de estos objetos y nombres dentro de otros componentes de la escena.
- **option:** El componente option en A-Frame sirve para crear opciones interactivas dentro de un menú o estante virtual, permitiendo que los usuarios interactúen con distintos objetos en una escena de realidad virtual.

  Este componente tiene varias propiedades importantes: `figure` define el modelo 3D a utilizar, `name` es el nombre del objeto, `pinchable` indica si el objeto puede ser "pinchado"(agarrado y movido), `scale` define el tamaño del modelo, `textScale` ajusta el tamaño del texto asociado, y `textPosition` determina la posición del texto sobre el objeto.

  Cuando el componente se inicializa, busca un estante cercano (shelf) para verificar si el objeto puede colocarse en él. Si todo está correcto, se crea una nueva entidad que representa el modelo 3D, se le aplica la escala especificada y se le añade un texto descriptivo encima.

  Si el objeto está marcado como pinchable, se le añade la capacidad de ser agarrado y movido mediante gestos de pellizco. Esto se logra agregando el componente pinchable y configurando un evento que actualiza la posición del objeto cuando es movido con un

pellizco. La función encargada de esto (onPinchedMoved) ajusta la posición del objeto según los movimientos del usuario, asegurando que la interacción sea precisa y fluida.

Combinando los componentes shelf y option en el HTML, se logra una estructura escalable y modular para construir menús interactivos dentro de una escena de A-Frame.

```
<!-- Segunda estantería a la izquierda con esferas y girada para estar de frente -->
<a-entity position="0 0 0s" rotation="0 90 0" shelf="objects: #figura1, #figura3; names: Opcion 1, Opcion 3">
  <a-sphere position="0 1.5 -0.5" radius="0.3" material="color: #654321; side: double; transparent: true; opacity: 0.4">
    <a-entity option="figure: #figura1; name: Opcion 1; pinchable: true" position="-0.15 0 0" creador sound-effects></a-entity>
    <a-entity option="figure: #figura3; name: Opcion 3" position="0.15 0 0"></a-entity>
    <a-entity option="figure: #figura2; name: Opcion 2" position="0 0 0"></a-entity>
  </a-sphere>
</a-entity>
```

Figura 3.15: Estantería que contiene varias opciones representadas por modelos 3D

- **creador:** El componente creador en A-Frame permite a los usuarios crear objetos 3D en la escena mediante gestos de pellizco. Cuando se inicializa, configura los escuchadores de eventos para detectar cuándo comienza y termina un pellizco. Al inicio de un pellizco, el componente crea un nuevo modelo GLTF (un dron) en la escena y lo posiciona. Al finalizar el pellizco, crea un nuevo cubo con una posición y tamaño determinados. Para evitar múltiples creaciones accidentales, el componente utiliza un sistema de enfriamiento (cooldown) que asegura que solo se pueda crear un objeto cada 300 milisegundos.

- **sound-effects:** El componente sound-effects en A-Frame se utiliza para añadir efectos de sonido a las interacciones con los objetos mediante gestos de pellizco. Al inicializarse, el componente selecciona dos archivos de sonido del documento: uno para cuando se agarra `pinchedstarted` y otro para cuando se suelta `pinchedended` un objeto. Los métodos `playGrabSound` y `playReleaseSound` están vinculados al contexto del componente y se activan en respuesta a los eventos de pellizco. Cuando se detecta que se ha iniciado un pellizco, se reproduce el sonido de agarre, y cuando el pellizco termina, se reproduce el sonido de liberación, proporcionando una retroalimentación auditiva al usuario sobre sus interacciones en la escena.

Prototipo menús

Para comprobar la funcionalidad de los componentes hechos durante el sprint he realizado el siguiente prototipo.

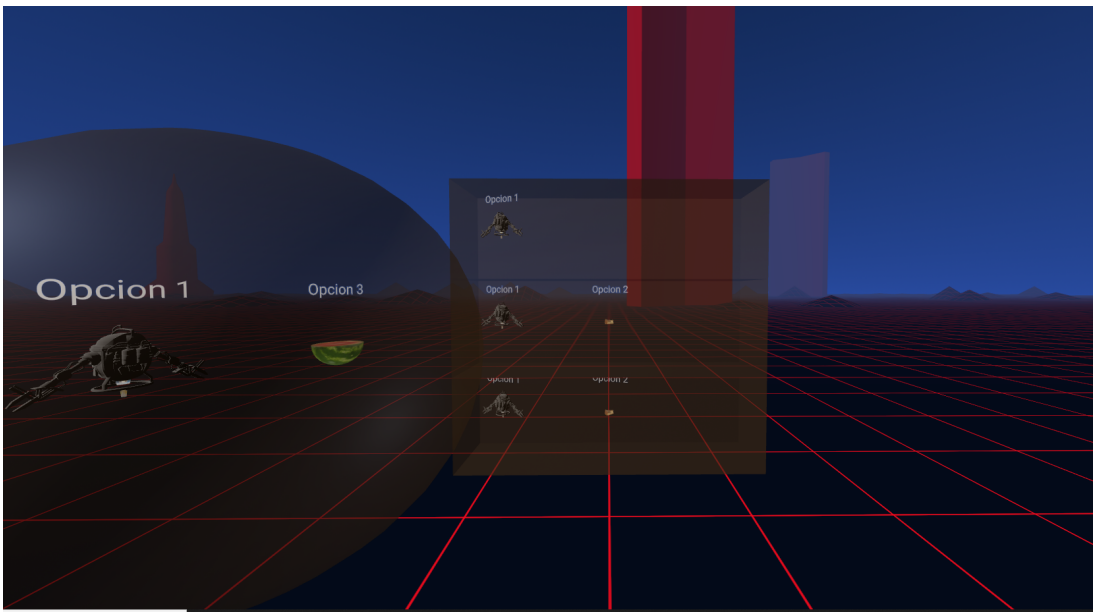


Figura 3.16: Escena básica con menús y objetos agarrables

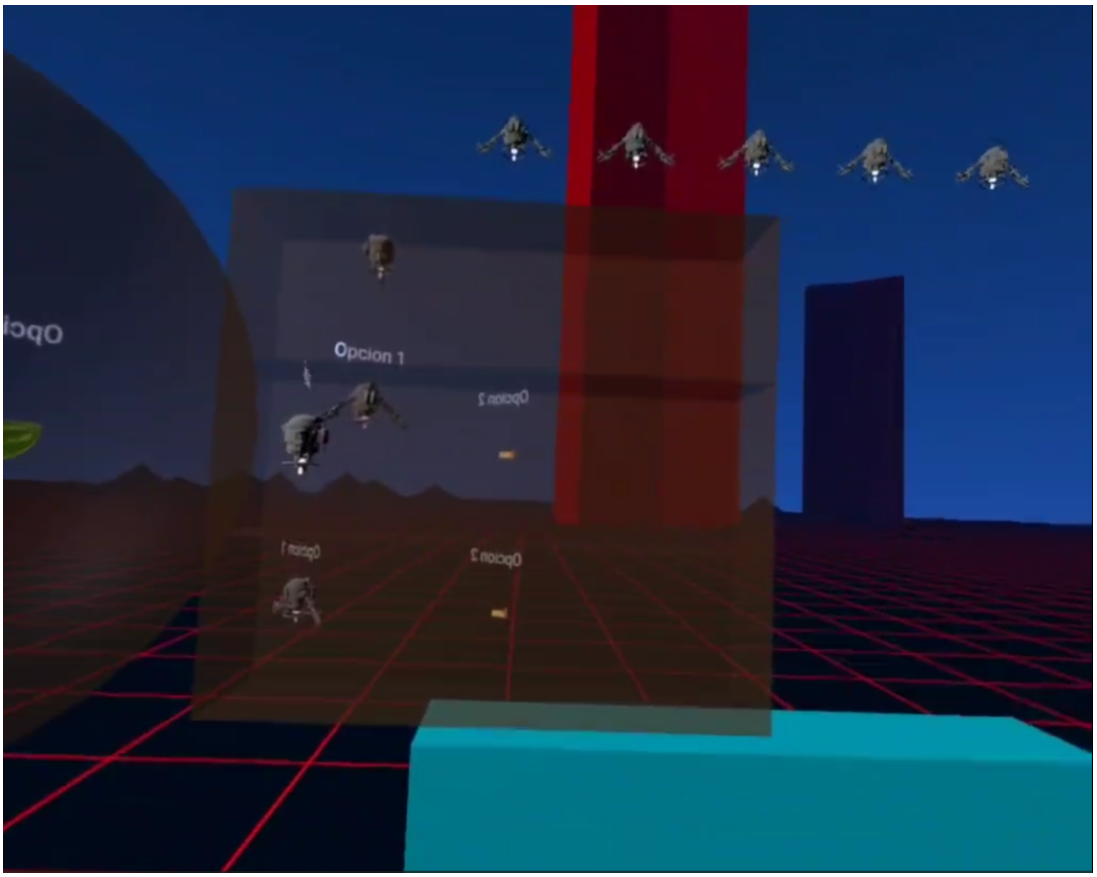


Figura 3.17: Después de interactuar con los objetos

3.3.4. Detalles del sprint

Durante este sprint, se han destacado varias lecciones importantes que han contribuido al desarrollo efectivo de interfaces de usuario interactivas en realidad virtual y la resolución de un

problema:

- **componentes Avanzados:** Implementar componentes personalizados como 'Creador' y 'sound-effects' en A-Frame ha demostrado ser crucial para modular y reutilizar funcionalidades complejas en diferentes partes de la aplicación. Esto no solo facilita el desarrollo, sino que también asegura una estructura modular que es escalable y fácil de mantener a medida que crece el proyecto.

- **Escalabilidad y Flexibilidad con los Componentes option y shelf:** Los componentes option y shelf han sido diseñados con una arquitectura que facilita la escalabilidad. Utilizando atributos de esquema `schema`, estos componentes pueden adaptarse dinámicamente a diferentes configuraciones y cantidades de objetos y nombres. Esto permite agregar fácilmente nuevas opciones (option) y estanterías (shelf) sin necesidad de modificar el código subyacente, lo cual es fundamental para proyectos que requieren expansión y mantenimiento a largo plazo.

- **Resolución de Problemas de Sincronización:** Se identificó un problema donde los componentes option no funcionaban correctamente debido a que el 'shelf' no estaba completamente inicializado. Este problema se resolvió esperando a que el componente 'shelf' emitiera el evento `componentinitialized`, asegurando que todas las propiedades estuvieran disponibles antes de proceder. Esto garantizó una inicialización correcta y evitó errores de acceso prematuro a los atributos del shelf.

3.4. Sprint 3

3.4.1. Objetivos del sprint

El objetivo de este sprint es explorar diferentes interfaces de usuario con los elementos ya desarrollados, combinándolos y añadiendo nuevas funcionalidades para proporcionar una experiencia más rica y dinámica.

3.4.2. Tareas

- **Implementar una interfaz de usuario para la habitación:** Crear una habitación donde un menú esférico contenga varios objetos. Cada objeto estará asociado a un componente que gestionará cambios en la habitación al agarrar y soltar los objetos, activando acciones específicas o modificaciones en el entorno virtual.

- **Diseñar la interfaz de usuario de los menús:** Desarrollar una escena con un menú esférico que contenga varios objetos. Cada objeto estará vinculado a un componente que, al ser

agarrado, creará un submenú diferente en la escena, permitiendo al usuario interactuar con opciones específicas relacionadas con cada objeto.

- **Crear una interfaz de usuario para un creador de objetos:** Configurar una escena con dos menús: uno rectangular para seleccionar el tipo de figura a crear y otro esférico para configurar las propiedades de la figura seleccionada. Esta interfaz permitirá al usuario crear y personalizar objetos dentro del entorno virtual.

- **Interfaz de usuario habitación V2:** Desarrollar una versión mejorada de la primera habitación. Esta nueva versión combinará las funcionalidades de los submenús y de la habitación inicial, pero sin activar eventos al soltar los objetos. Esto proporcionará una experiencia mejorada y más refinada para los usuarios.

3.4.3. Prototipos

En este sprint voy a explicar los componentes construidos directamente durante el desarrollo de cada prototipo y estos componentes están explicados con detalle en esta sección 4.3.

Implementar una interfaz de usuario para la habitación

Se ha desarrollado una interfaz de usuario que permite la interacción con diversos elementos en una habitación virtual. Esta interfaz incluye un menú interactivo en forma de esfera, en el cual los objetos pueden ser agarrados. Cada objeto en el menú tiene asignado un componente específico con escuchadores de eventos, permitiendo personalizar la configuración de la habitación en función del objeto seleccionado.

La estructura del menú sigue el mismo enfoque que las interfaces de usuario anteriores, utilizando los componentes shelf y option para organizar y configurar el menú según las necesidades. Además, se han realizado mejoras en estos componentes para gestionar tanto los objetos permitidos como los nombres de manera global. Ahora, el componente shelf almacena y verifica tanto los objetos como los nombres permitidos, y el componente option utiliza esta información para asegurar que solo los objetos y nombres permitidos sean utilizados, garantizando una configuración coherente y segura del menú.

Cada objeto dentro del menú está asociado a un componente específico que permite interactuar con la habitación. Los siguientes componentes son simples, ya que esperan que ocurra el evento `pinchedstarted` (cuando un objeto es agarrado) para realizar una acción en la escena, y el evento `pinchedended` (cuando se suelta el objeto) para revertir dicha acción:

- **change-wall-color**: Al agarrar el objeto asociado a este componente, las paredes se pintan de azul. Al soltarlo, las paredes vuelven a su color original.

- **toggle-box**: Al agarrar el objeto asociado a este componente, se crea una caja en el suelo de la habitación. Al soltarlo, esta caja se elimina.
- **remove-roof**: Al agarrar el objeto asociado a este componente, el techo desaparece. Al soltarlo, el techo vuelve a aparecer.
- **toggle-light**: Al agarrar el objeto asociado a este componente, la luz de la habitación se apaga. Al soltarlo, la luz se enciende.
- **sound-effects**: Al agarrar el objeto asociado a este componente, empieza a sonar música de fondo. Al soltarlo, la música se detiene.

Estos componentes permiten una personalización dinámica e interactiva de la habitación virtual, proporcionando una experiencia inmersiva y ajustable según las preferencias del usuario. Este prototipo pasó a llamarse ”La habitación de las interacciones” y lo explico en detalle en esta sección 4.1.1

Diseñar la interfaz de usuario de los menús

Esta interfaz permite al usuario seleccionar qué objeto crear en la escena (caja, cilindro y cono) desde el menú ubicado a la derecha. Mientras tanto, el menú esférico a la izquierda funciona como un configurador para personalizar la escala, color y rotación de estos objetos.

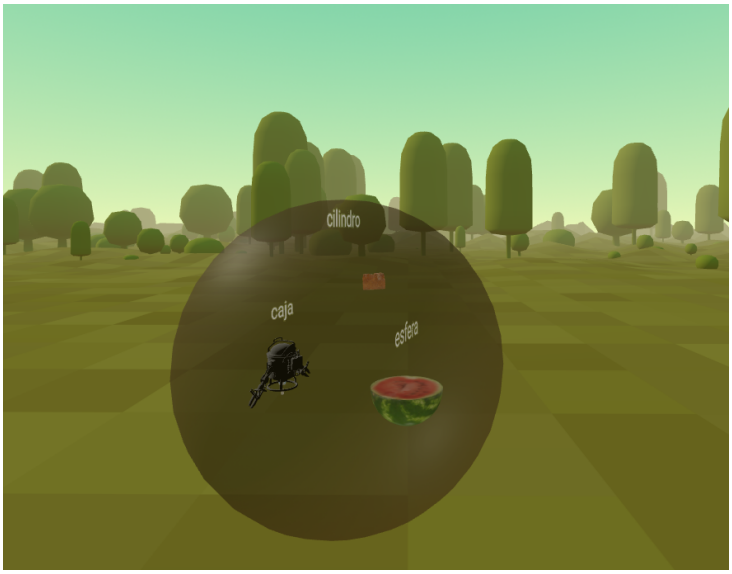


Figura 3.18: Interfaz de usuario de los menús

Esta interfaz de usuario sigue la dinámica establecida por las versiones anteriores, pero introduce una diferencia significativa: los componentes asignados a los objetos tienen la función de generar nuevos menús en lugar de modificar directamente la escena virtual. Este enfoque permite una organización más estructurada y dinámica de las opciones disponibles para el usuario.

En detalle, cada objeto dentro del menú principal tiene asociado un componente específico que utiliza los componentes shelf y option para organizar y presentar submenús. Cuando el usuario interactúa agarrando uno de estos objetos, se activa la creación de un nuevo menú contextualizado, con nuevas opciones que pueden incluir ajustes, configuraciones o selecciones adicionales. Esta interactividad en capas facilita una navegación más intuitiva y profunda dentro de la aplicación de realidad virtual.

Esta evolución en el diseño de la interfaz no solo mejora la usabilidad al organizar de manera eficiente las opciones disponibles, sino que también enriquece la experiencia del usuario al proporcionar múltiples niveles de interacción dentro del entorno virtual.

Componentes de esta escena:

- **menu-cylinder**: Al agarrar el objeto asociado a este componente, se crea un menú con forma de cilindro que contiene dos opciones (objetos) que se pueden agarrar. En este prototipo, estas opciones no tienen funcionalidad adicional.
- **menu-sphere**: Al agarrar el objeto asociado a este componente, se crea un menú con forma de esfera que contiene dos opciones (objetos) que se pueden agarrar. En este prototipo, estas opciones no tienen funcionalidad adicional.
- **menu-box**: Al agarrar el objeto asociado a este componente, se crea un menú con forma de caja que contiene dos opciones (objetos) que se pueden agarrar. En este prototipo, estas opciones no tienen funcionalidad adicional.

Crear una interfaz de usuario para un creador de objetos

Esta interfaz permite al usuario seleccionar qué objeto crear en la escena (caja, cilindro y cono) desde el menú ubicado a la derecha. Mientras tanto, el menú esférico a la izquierda funciona como un configurador para personalizar la escala, color y rotación de estos objetos..

Finalmente, esta interfaz utiliza dos componentes:

- **creador**: Facilita la creación dinámica de nuevos objetos 3D en la escena al recibir un gesto de pinzado `pinchedstarted`. Si ya existe un objeto previamente creado, este componente lo elimina antes de instanciar uno nuevo según el tipo especificado (’box’, ’cylinder’ o ’cone’). Los objetos recién creados se colocan automáticamente a una altura predeterminada y se configuran con una escala inicial y un color estándar, listos para ser personalizados con el componente ’configurador’.
- **configurador**: Permite ajustar propiedades como color, escala y rotación de los objetos seleccionados mediante gestos de pinzado `pinchedstarted`. Al detectar un gesto, realiza cambios específicos en el objeto actual, como cambiar su color a través de una lista predefinida de colores, aumentar su escala de manera incremental y rotar el objeto

en incrementos de 45 grados en el eje Y. Este componente asegura una interactividad fluida y directa para personalizar objetos en tiempo real dentro de la escena.

Explico en detalle este prototipo en esta sección 4.1.3

Interfaz de usuario habitación de los submenús

En esta versión mejorada, la interfaz de usuario de la habitación incluye nuevas funcionalidades y una mayor cantidad de opciones interactivas para personalizar la habitación virtual. Se han añadido nuevos submenús que proporcionan al usuario más opciones de interacción.

Los cambios más significativos en esta versión son los componentes `menu` asignados a cada objeto. Estos componentes permiten que, al interactuar con un objeto específico, se generen submenús contextuales, proporcionando una navegación más detallada y opciones adicionales para los usuarios.

Estos submenús permiten realizar acciones más específicas dentro de la escena, tales como:

- **Iluminación:** Permite ajustar la intensidad y el color de la luz en la habitación, ofreciendo una mayor personalización del ambiente. Este menú tiene 3 componentes:
  - **toggle-light-intensity:** Este componente cambia la intensidad de la luz de la habitación cuando se produce un evento de pinzado `pinchedstarted`(). La luz alterna entre dos intensidades predefinidas
  - **toggle-light:** Este componente apaga o enciende la luz de la habitación cuando se produce un evento de pinzado `pinchedstarted`.
  - **toggle-red-light:** Este componente alterna el color de la luz de la habitación entre blanco y un tono rojizo suave cuando se produce un evento de pinzado `pinchedstarted`.
- **Color pared:** Al seleccionar esta opción, se despliega un submenú con varias opciones de colores, permitiendo al usuario personalizar el entorno de manera más precisa.
  - **change-wall-color-blue:** Este componente cambia el color de las paredes de la habitación a azul cian cuando se produce un evento de pinzado `pinchedstarted`.
  - **change-wall-color-green:** Este componente cambia el color de las paredes de la habitación a verde suave cuando se produce un evento de pinzado `pinchedstarted`.
  - **change-wall-color-yellow:** Este componente cambia el color de las paredes de la habitación a amarillo suave cuando se produce un evento de pinzado `pinchedstarted`.
- **Quitar pared:** Los submenús pueden incluir opciones para añadir música de fondo o efectos de sonido específicos, mejorando la inmersión en la escena virtual.

  - **remove-back-wall:** Este componente alterna la visibilidad de la pared trasera cuando se produce un evento de pinzado `pinchedstarted`.
  - **remove-front-wall:** Este componente alterna la visibilidad de la pared frontal cuando se produce un evento de pinzado `pinchedstarted`.
  - **remove-roof:** Este componente alterna la visibilidad del techo cuando se produce un evento de pinzado `pinchedstarted`.

- **Agregar objeto:** Los submenús pueden incluir opciones para añadir diferentes tipos de muebles o decoraciones, cada uno con sus propias configuraciones de color, escala y posición.

  - **toggle-box:** Este componente crea una caja (a-box) en la escena cuando se produce un evento de pinzado `pinchedstarted` si ya está creada esta desaparece.
  - **toggle-cone:** Este componente crea un cono (a-cone) en la escena cuando se produce un evento de pinzado `pinchedstarted` si ya está creado este desaparece.
  - **toggle-cylinder:** Este componente crea un cilindro (a-cylinder) en la escena cuando se produce un evento de pinzado `pinchedstarted` si ya está creado esta desaparece.

- **Música:** Los submenús pueden incluir opciones para añadir música de fondo o efectos de sonido específicos, mejorando la inmersión en la escena virtual.

  - **sound-mario-bros:** Este componente alterna la reproducción del sonido de ”Mario Brosçuando se produce un evento de pinzado `pinchedstarted`
  - **sound-mario-jump:** Este componente alterna la reproducción del sonido de ”Mario Jumpçuando se produce un evento de pinzado `pinchedstarted`.
  - **sound-stranger-things:** Este componente alterna la reproducción del sonido de ”Stranger Thingsçuando se produce un evento de pinzado `pinchedstarted`.

Explico con más detalle este prototipo en esta sección 4.1.2

3.4.4. Detalles del sprint

Durante este sprint, se han destacado varias lecciones importantes que han contribuido al desarrollo efectivo de interfaces de usuario interactivas en realidad virtual:

- **Modularidad en el Diseño de Componentes:** La creación de componentes modulares como `shelf`, `option`, `creador` y `configurador` facilitó la reutilización y la escalabilidad del código. Esta práctica permitió combinar diferentes funcionalidades y añadir nuevas características de manera eficiente.

- **Gestión Dinámica de Submenús:** Inicialmente, la gestión de submenús en la escena se implementó mediante la manipulación de la visibilidad de los menús en el HTML, configurándolos como invisibles (`visible=false`) y alternando su visibilidad mediante componentes específicos. Sin embargo, esta aproximación presentó un problema: los menús invisibles seguían siendo interactivos, permitiendo al usuario interactuar con ellos incluso cuando no eran visibles. Además, al hacerlos visibles nuevamente, los objetos dentro de los menús permanecían en las posiciones y estados anteriores, lo que no era deseable para la experiencia del usuario.

  Para resolver este problema, se adoptó un enfoque diferente donde los componentes de los submenús no solo alternan la visibilidad, sino que crean y eliminan dinámicamente los menús y sus objetos asociados. Esto garantiza que los menús solo existan en el DOM cuando son necesarios y se eliminen completamente cuando no se usan, asegurando que cada interacción comience desde un estado inicial limpio. Este enfoque no solo mejora la usabilidad al evitar interacciones no deseadas con menús invisibles, sino que también facilita una mejor gestión de los recursos y el estado de la escena. Esta gestión de los submenús la explicaré en detalle en el próximo capítulo.

Capítulo 4

Resultados

El resultado final de este proyecto ha sido el desarrollo de un conjunto de componentes diseñados para permitir la interacción con una escena virtual utilizando seguimiento de manos en A-Frame. Estos componentes forman una "caja de herramientas" que facilita la creación de interfaces de usuario inmersivas y altamente interactivas. Para demostrar la funcionalidad y versatilidad de estos componentes, se han construido varios prototipos, cada uno ilustrando diferentes aspectos y aplicaciones del seguimiento de manos en entornos de realidad virtual.

En este capítulo, se describirán en detalle los prototipos desarrollados, destacando sus características principales. Después, se explicará cómo utilizar la caja de componentes, proporcionando instrucciones claras y precisas para su implementación en otras aplicaciones. Por último, se discutirá la estructura interna de los componentes, ofreciendo una visión técnica de su construcción y las decisiones de diseño que se tomaron durante el desarrollo del proyecto.

4.1. Descripción de los prototipos

En esta sección se detallan las escenas desarrolladas, explicando cómo los usuarios finales deben interactuar con ellas. Cada prototipo se carga simplemente ingresando una URL específica. Estos prototipos son accesibles desde diversos dispositivos, incluidos ordenadores, móviles y gafas de realidad virtual. Sin embargo, la interacción completa, aprovechando el seguimiento de manos, está hecha para dispositivos con soporte para la API de WebXR con seguimiento de manos, en mi caso he utilizado las Oculus Quest.

La guía incluirá instrucciones sobre cómo navegar por las escenas, describiendo las funcionalidades disponibles y cómo interactuar con los elementos presentes en el entorno virtual.

4.1.1. La habitación de las interacciones

Este prototipo consiste en personalizar la habitación mediante la interacción de los objetos que están dentro de un menú, la interacción ocurre tanto al agarrar un objeto como al soltarlo.

Al ingresar a la escena, te encontrarás en una habitación virtual equipada con paredes, suelo y techo. Utiliza el movimiento de tu cabeza para mirar a tu alrededor y explora la habitación moviéndote libremente. La exploración de la habitación te permitirá familiarizarte con el entorno y localizar el menú esférico interactivo que contiene diversos objetos flotando dentro de él.

Tus manos virtuales están representadas en la escena y se pueden utilizar para agarrar y mover los objetos del menú esférico. Para interactuar con los objetos, acerca tus manos virtuales hasta ellos y cierra los dedos índice y pulgar en forma de "pinza" para cogerlos. Esta forma de interacción intuitiva permite una manipulación precisa y natural de los elementos virtuales, proporcionando una experiencia inmersiva y realista.

En la escena se ve un menú esférico transparente que contiene cinco objetos diferentes, cada uno asociado con una acción específica. Al agarrar y soltar estos objetos, puedes personalizar diversos aspectos de la habitación. a continuación, hay un listado de estos objetos según la etiqueta que tienen en la escena:

- **Cambiar Color:** Al interactuar con este objeto, las paredes de la habitación cambiarán a un color azul cian. Al soltar el objeto, las paredes volverán a su color original.
- **Apagar Luces:**: Este objeto controla la iluminación de la habitación. Al agarrarlo, las luces se apagarán. Al soltarlo, la iluminación se restaurará a su nivel original.
- **Activar sonido:** Al agarrar este objeto, se activará un efecto de sonido que se reproducirá. Al soltar el objeto, el sonido se pausará.
- **Quitar Techo:** Esta interacción permite modificar la estructura de la habitación. Al agarrar el objeto, el techo de la habitación desaparecerá, ofreciendo una vista despejada del cielo virtual. Soltar el objeto restaurará el techo a su posición original, permitiendo alternar entre una vista al aire libre y una habitación cerrada.
- **Agregar Objeto:** Al agarrar el objeto, se generará un cubo que se colocará en una posición predeterminada dentro de la habitación. Al soltarlo, el cubo desaparecerá, ofreciendo una forma sencilla de añadir y eliminar objetos dentro del entorno virtual.

Ejemplo de Interacción: Al ingresar a la habitación virtual, el usuario utiliza sus manos virtuales para explorar el entorno. Al levantar la mano derecha y formar una pinza con los dedos índice y pulgar, el usuario agarra el objeto Çambiar Color"del menú esférico. Inmediatamente, las paredes de la habitación se tornan azul cian, permitiendo al usuario visualizar el cambio de color. Tras explorar un poco más, suelta el objeto y las paredes vuelven a su tono original. Luego, el usuario agarra el objeto "Quitar Techo", haciendo que el techo desaparezca y revelando el cielo virtual. Aprovechando la nueva vista, agarra el objeto .Activar Sonido"para reproducir una música ambiental que complementa la experiencia inmersiva. Cada acción y su correspondiente

reversión al soltar los objetos ilustran la intuitiva interacción y personalización que ofrece la escena.

Figura 4.1: Interfaz de usuario habitación

4.1.2. La habitación de los submenús

Este prototipo avanza en la personalización de la habitación, integrando submenús contextuales que permiten al usuario interactuar de manera más detallada y específica con el entorno. La interacción se realiza tanto al agarrar como al soltar objetos del menú principal, generando submenús con opciones adicionales.

La Habitación de los submenús se presenta como una habitación virtual con paredes, suelo y techo, similar a la versión anterior. El usuario puede explorar libremente el entorno utilizando el movimiento de su cabeza y las manos virtuales.

Dentro de la escena, las manos del usuario están representadas virtualmente y permiten interactuar con los objetos del menú principal. Al formar una pinza con los dedos índice y pulgar, el usuario puede agarrar y mover los objetos, activando sus respectivas funcionalidades.

Un menú principal transparente contiene diversos objetos, cada uno asociado con un submenú específico. Al interactuar con estos objetos, se generan submenús que ofrecen opciones adicionales para personalizar la habitación.

Descripción de los objetos del menú principal, listados según su etiqueta:

- **Iluminación:** Al agarrar este objeto, se despliega un submenú esférico con opciones para controlar las luces de la habitación, incluyendo encender/apagar las luces, cambiar a luz roja y ajustar la intensidad de la luz.
- **Color Pared:**: Este objeto genera un submenú que permite cambiar el color de las paredes a azul, verde o amarillo. Cada opción es accesible a través de un objeto flotante dentro del submenú.

- **Pared:**Al interactuar con este objeto, se abre un submenú con opciones para eliminar el techo, la pared frontal o la pared trasera de la habitación.
- **Agregar Objeto:** Este objeto permite acceder a un submenú que ofrece la opción de agregar diferentes formas geométricas (cubo, cono, cilindro) en la habitación. Estos objetos aparecen en posiciones predeterminadas y pueden ser eliminados al volver a agarrar el objeto del menú.
- **Música:** Al seleccionar este objeto, se despliega un submenú con varias opciones de efectos de sonido, incluyendo música de Stranger Things, Mario Bros y el sonido de salto de Mario. Agarrar un objeto del submenú, reproduce el sonido correspondiente y volver a agarrarlo lo detiene.

Ejemplo de Interacción: El usuario ingresa a la habitación virtual y utiliza sus manos para explorar. Decide cambiar la iluminación, por lo que agarra el objeto “Iluminación” del menú principal. Un submenú esférico aparece, y el usuario selecciona la opción “Luz roja”. Inmediatamente, la luz de la habitación cambia a un tono rojizo. Luego, el usuario decide agregar un objeto decorativo, agarra el objeto “Añadir Objeto” y selecciona la opción “Crear cubo” en el submenú correspondiente. Un cubo aparece en la habitación.

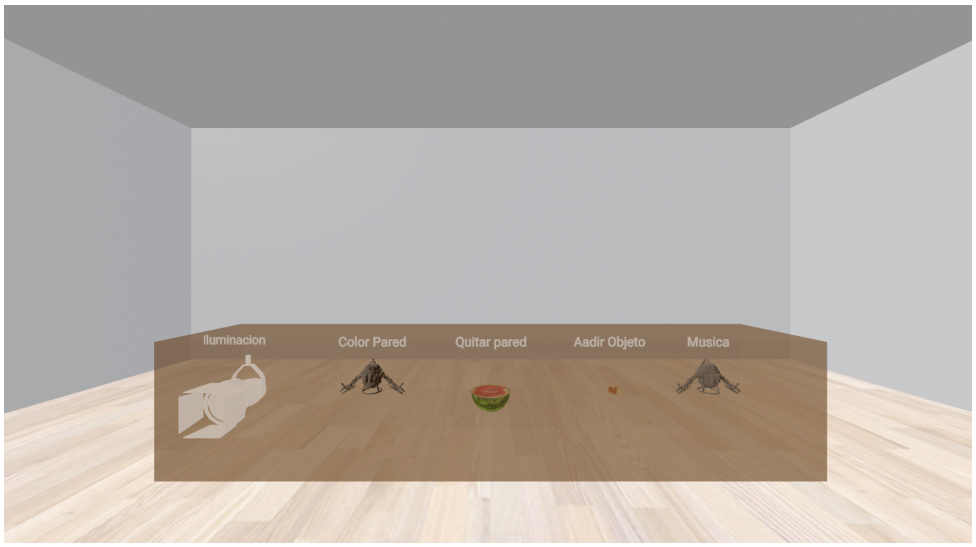


Figura 4.2: Habitación de los submenús

4.1.3. Creador de objetos

Este prototipo está diseñado para permitir a los usuarios crear y personalizar objetos 3D dentro de una escena virtual. La interacción se realiza mediante el seguimiento de manos, permitiendo a los usuarios seleccionar, crear y configurar objetos utilizando gestos naturales.

La escena del Creador de Objetos se presenta en un entorno virtual donde los usuarios pueden explorar libremente utilizando el movimiento de su cabeza. Las manos del usuario están

representadas virtualmente y permiten interactuar con los objetos del menú principal.

En la escena, frente al usuario, se encuentran dos menús principales: uno para la creación de objetos y otro para la configuración de los mismos. Al formar una pinza con los dedos índice y pulgar, el usuario puede agarrar y mover los objetos del menú, activando sus respectivas funcionalidades.

El menú de la derecha permite seleccionar el tipo de objeto a crear (caja, cilindro o cono). Una vez creado, el objeto aparece en una posición predeterminada en la escena. El menú de la izquierda proporciona opciones de configuración para cambiar el color, la escala y la rotación del objeto creado.

Menú creación

- **Box:** Al seleccionar este objeto, se crea una caja en la escena en una posición predeterminada. La caja se puede personalizar utilizando las opciones del menú de configuración.

- **Cylinder:**: Al seleccionar este objeto, se crea un cilindro en la escena. El cilindro se puede personalizar utilizando las opciones del menú de configuración.

- **Cone:** Al seleccionar este objeto, se crea un cono en la escena. El cono se puede personalizar utilizando las opciones del menú de configuración.

Menú de configuración

- **Change Color:** Este objeto permite cambiar el color del objeto actualmente seleccionado en la escena. Cada vez que se agarra, el color del objeto cambia a través de una lista predefinida de colores.

- **Change Scale:**: Este objeto permite aumentar la escala del objeto actualmente seleccionado. Cada vez que se agarra, la escala del objeto incrementa en todas las dimensiones.

- **Rotate:** Este objeto permite rotar el objeto actualmente seleccionado en la escena. Cada vez que se agarra, el objeto rota 45 grados en el eje Y.

Ejemplo de Interacción: El usuario entra en la escena virtual y utiliza sus manos para explorar. Decide crear un objeto, por lo que agarra el objeto `Box` del menú de creación. Una caja aparece en la escena. A continuación, el usuario selecciona `Change Color` del menú de configuración para cambiar el color de la caja. La caja cambia a un nuevo color cada vez que se agarra el objeto `Change Color`. Satisfecho con el color, el usuario procede a ajustar la escala agarrando el objeto `Change Scale`, incrementando el tamaño de la caja. Finalmente, el usuario selecciona Rotate"para rotar la caja en el eje Y.

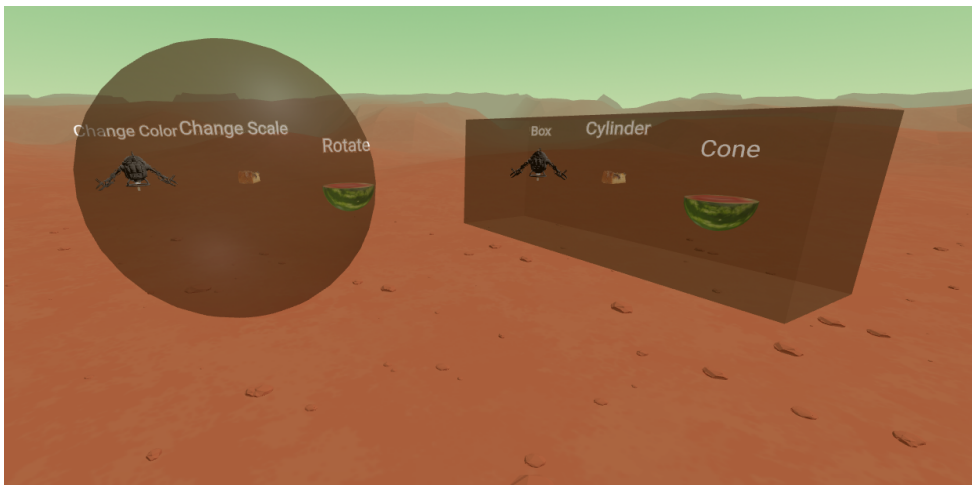


Figura 4.3: Creador de objetos

4.2. Descripción del conjunto de componentes

En esta sección se proporciona una descripción detallada destinada a usuarios que desean componer escenas utilizando los componentes desarrollados en este proyecto. Se explica el funcionamiento de cada componente, permitiendo comprender cómo utilizarlos y combinarlos para añadir nuevas funcionalidades a las escenas.

El conjunto de componentes es lo suficientemente genérico y flexible para permitir la construcción de una variedad de escenas y tipos de interacción, facilitando la creación de experiencias de realidad virtual personalizadas y ricas en funcionalidades.

Componentes para crear menús

Estos componentes se repiten en todas las escenas y son fundamentales para la creación de menús principales utilizando shelf y option, aprovechando las capacidades de pinchable para la manipulación intuitiva de objetos en el entorno virtual.

- **Componente shelf:** El componente shelf gestiona las opciones disponibles en la interfaz de usuario. Recibe un array de objetos (`objects`) y nombres (`names`) que define las figuras y acciones permitidas en el menú. Al inicializarse, shelf almacena estos objetos y nombres para su uso global, asegurando que solo las figuras y acciones permitidas sean accesibles.

- **Componente option:**: El componente option define una opción interactiva dentro de la interfaz de usuario. Este componente verifica si la figura especificada está permitida según lo definido en el componente shelf y, si es así, la añade a la escena con la configuración especificada. Sus parámetros incluyen `figure` (URL del modelo GLTF), `name` (nombre de la acción), `pinchable` (indica si puede ser pinchada y movida), `scale` (escala del

modelo), `textScale` (escala del texto) y `textPosition` (posición del texto). Al inicializarse, `option` también configura si la entidad puede ser movida mediante gestos de pinzado, basado en el parámetro pinchable.

- **Componente pinchable:** El componente pinchable permite que las entidades sean movidas mediante el gesto de pinzar. Gestiona eventos de inicio (pinchedstarted), fin (pinchedended) y movimiento (pinchedmoved) del gesto de pinzado, actualizando la posición del objeto en consecuencia. Este componente se añade a cada entidad que requiere manipulación intuitiva, y su comportamiento se controla mediante el parámetro `pinchDistance`, que define la distancia a la que se activa el evento `pinchedstarted`.

Componentes menús específicos

Los componentes de menús específicos en este proyecto están diseñados para desplegar opciones interactivas dentro el/los menús principales. Cada componente, al ser activado mediante un gesto de pinzado `pinchedstarted`, muestra una esfera interactiva con múltiples opciones.

Primero, se crea la entidad shelf, que define qué modelos 3D (figura1, figura2, figura3) y nombres asociados (Sub Opción A, Sub Opción B, Sub Opción C) se incluirán en el menú. Luego, se crea la entidad `menuSphere` (dependiendo del tipo de menú que se esté utilizando) como hijo de shelf, la cual representa visualmente la esfera interactiva en el espacio virtual. Dentro de `menuSphere`, se añaden un texto descriptivo y tres entidades option, cada una configurada para mostrar un modelo 3D específico junto con su nombre y funcionalidad asociada según el tipo de menú.

Este enfoque modular permite gestionar de manera eficiente la creación y personalización de menús interactivos dentro del entorno de realidad virtual. Todos los componentes de las entidades creadas en cada menú serán explicados después en los componentes interactivos.

- **Componente menu-light:** Cuando se interactúa con la entidad asociada, este componente creará una esfera con 3 entidades option toggle-light, toggle-red-light, toggle-light-intensity. controla la intensidad de la luz ambiental.

- **Componente menu-music:** Cuando se interactúa con la entidad asociada, este componente creará una esfera con 3 entidades, sound-stranger-things, sound-mario-bros, sound-mario-jump.

- **Componente menu-object:** Cuando se interactúa con la entidad asociada, este componente creará una esfera con 3 entidades, toggle-box, toggle-cone, toggle-cylinder.

- **Componente menu-wall-color:** Cuando se interactúa con la entidad asociada, este componente creará una esfera con 3 entidades, change-wall-color-blue, change-wall-color-green, change-wall-color-yellow.

- **Componente menu-wall:** Cuando se interactúa con la entidad asociada, este componente creará una esfera con 3 entidades, remove-back-wall, remove-roof, remove-front-wall.

Componentes actuadores

Con componentes actuadores me refiero que son los componentes que desencadenan una acción en la escena, apagar luces, pintar paredes, encender la música etc. Estos primeros cinco componentes tienen la particularidad de activar una acción al agarrarlos `pinchedstarted` y revertirla al soltarlos `pinchedended`.

- **Componente toggle-light:** controla la intensidad de la luz ambiental. Apaga la luz al agarrar el objeto y cuando se suelta vuelve a encenderse la luz, con un breve periodo de enfriamiento para evitar activaciones rápidas sucesivas.
- **Componente remove-roof:** Este componente permite quitar el techo de la habitación. Se elimina mediante un gesto de pinchar y el techo vuelve a ser visible cuando se suelta.
- **Componente change-wall-color:** Este componente permite cambiar el color de las paredes. Cuando se activa, modifica el material de las paredes para reflejar el nuevo color al soltarse, vuelve al color original.
- **Componente toggle-box:** Este componente permite agregar o quitar objetos en la habitación. Se activa mediante un gesto de pinchar, añadiendo una caja al suelo de la habitación, soltar el objeto elimina la caja.
- **Componente sound-effects:** Este componente controla la reproducción de efectos de sonido. Se activa mediante un gesto de pinchar y reproduce el sonido cuando se suelta el objeto, el sonido se pausa y se reinicia.

A continuación, cada componente responde a un gesto de pinchar `pinchedstarted` para activar una acción particular:

- **Componente change-wall-color-blue:** Este componente cambia el color de las paredes (excepto techo y suelo) a un tono de azul cian cuando se activa mediante un gesto de pinchar.
- **Componente change-wall-color-green:** Al activar este componente, las paredes adquieren un tono verde suave. Funciona de manera similar al componente anterior, aplicando el color a las mismas paredes definidas.
- **Componente change-wall-color-yellow:** Este componente cambia el color de las paredes a un tono de amarillo suave cuando se activa. También afecta a las mismas paredes que los componentes anteriores.

- **Componente remove-back-wall:** Controla la visibilidad de la pared trasera. Al activarse con un gesto de pinchar, alterna la visibilidad de esta pared entre visible e invisible.
- **Componente remove-front-wall:** Este componente gestiona la visibilidad de la pared frontal. Al activarse, cambia la visibilidad de esta pared de manera similar al componente anterior.
- **Componente remove-roof:** Permite controlar la visibilidad del techo. Al activarse mediante un gesto de pinchar, cambia la visibilidad del techo entre visible e invisible.
- **Componente sound-mario-bros:** Este componente controla la reproducción de un sonido específico. Al activarse con un gesto de pinchar, alterna entre reproducir y pausar el sonido. Utiliza el elemento de audio identificado como “mario-sound”.
- **Componente sound-mario-jump:** Similar al componente anterior, este componente gestiona la reproducción de otro sonido. También se activa mediante un gesto de pinchar y utiliza el elemento de audio “mario-jump-sound”.
- **Componente sound-stranger-things:** Controla la reproducción de un sonido asociado a ”Stranger Things”. Al activarse con un gesto de pinchar, inicia o detiene la reproducción del sonido utilizando el elemento de audio “grab-sound”.
- **Componente toggle-box:** Este componente permite añadir o quitar un objeto tipo caja `<a-box>` en la escena. Al activarse con un gesto de pinchar, alterna la presencia de este objeto en la posición especificada.
- **Componente toggle-cone:** Controla la creación y eliminación de un objeto tipo cono `<a-cone>` en la escena. Al activarse mediante un gesto de pinchar, añade o remueve este objeto en la ubicación especificada.
- **Componente toggle-cylinder:** Gestiona la creación y eliminación de un objeto tipo cilindro `<a-cylinder>` en la escena. Al activarse con un gesto de pinchar, controla la presencia de este objeto en la posición especificada.
- **Componente toggle-light-intensity:** Este componente ajusta la intensidad de la luz ambiente en la habitación virtual. Al activarse con un gesto de pinchar, alterna entre dos niveles predefinidos de intensidad de luz.
- **Componente toggle-light:** Controla el encendido y apagado de la luz ambiente en la habitación virtual. Al activarse con un gesto de pinchar, cambia el estado de la luz entre encendido y apagado.
- **Componente toggle-red-light:** Este componente gestiona un ajuste específico de la luz ambiente, cambiando su color entre blanco y un tono rojizo suave. Se activa mediante un gesto de pinchar.

Estos dos últimos componentes suelen utilizarse en la misma escena, el componente creador cuando crea un objeto guarda la referencia de este objeto en una variable global y después el componente configurador usa esta variable global para cambiar los atributos de ese objeto que quiera el usuario.

- **Componente creador:** El componente creador es responsable de la creación dinámica de objetos 3D en la escena virtual. Este componente permite a los usuarios generar diferentes tipos de geometrías (e.g., caja, cilindro, cono) dependiendo con el objeto que hayas interactuado. El parámetro que tiene este componente es `type` el cual especifica el tipo de geometría a crear.

- **Componente configurador:** El componente configurador es responsable de aplicar configuraciones dinámicas a los objetos creados en la escena. Este componente permite a los usuarios modificar propiedades como color, escala y rotación de los objetos a través de gestos de interacción. El parámetro que tiene es `action` el cual especifica la acción de configuración a realizar.

4.2.1. Composición de Componentes

Para componer una escena completa utilizando los componentes desarrollados, es necesario entender cómo se integran y configuran en el HTML. La composición de componentes sigue un enfoque modular y reutilizable, permitiendo la creación de menús interactivos y funcionalidades específicas en la habitación virtual.

1. **Definir los ítems:** En primer lugar, definir el entorno y los ítems necesarios como modelos 3D y sonidos, en la sección `<a-assets>`. Esto asegura que todos los recursos estén cargados antes de que se utilicen en la escena.

```
<a-scene environment="preset: tron;">
  <a-assets>
    <a-asset-item id="figura1" src="figuras3d/color_.glb"></a-asset-item>
    <a-asset-item id="figura2" src="figuras3d/fruit.glb"></a-asset-item>
    <a-asset-item id="figura3" src="figuras3d/watermelon.glb"></a-asset-item>
    <a-asset-item id="figura4" src="figuras3d/focus.glb"></a-asset-item>
    <a-asset-item id="figura5" src="figuras3d/tejas.glb"></a-asset-item>
    <audio id="grab-sound" src="sounds/stranger-things.mp3"></audio>
    <img id="floor" src="assets/floor-texture.jpg">
  </a-assets>
```

Figura 4.4: Definir entorno e ítems

2. **Crear la Escena Principal:** Define Los elementos base de la escena, en el caso de utilizar los componentes que realizan cambios en la habitación es importante definir las paredes, el suelo y el techo.

```
<!-- Habitacion -->
<a-entity id="room">
  <a-plane id="wall1" position="0 2 -5" rotation="0 0 0" width="10" height="4" material="color: #B0B0B0; side: double;"></a-plane>
  <a-plane id="wall2" position="5 2 0" rotation="0 -90 0" width="10" height="4" material="color: #B0B0B0; side: double;"></a-plane>
  <a-plane id="wall3" position="0 2 5" rotation="0 180 0" width="10" height="4" material="color: #B0B0B0; side: double;"></a-plane>
  <a-plane id="wall4" position="-5 2 0" rotation="0 90 0" width="10" height="4" material="color: #B0B0B0; side: double;"></a-plane>
  <a-plane id="floor" position="0 0.15 0" rotation="-90 0 0" width="10" height="10" material="src: #floor; side: double;"></a-plane>
  <a-plane id="ceiling" position="0 4 0" rotation="90 0 0" width="10" height="10" material="color: #A0A0A0; side: double;"></a-plane>
</a-entity>
```

Figura 4.5: Crear habitación

3. **Agregar el Menú Principal:** Para ello utiliza el componente shelf para definir los objetos que puede tener el menú y los nombres, después crea una entidad primitiva para dar la forma que quieras al menú (esfera, caja, cilindro etc.) y por último dentro de esta entidad puedes escribir las entidades que quieras utilizando el componente option y rellena todos sus parámetros para definir la figura, el nombre, si puede ser agarrado o no, la escala de la figura, la escala del texto y la posición del texto. El menú principal debería verse así:

```
<!-- Menú principal -->
<a-entity position="0 0 0" shelf="objects: #figura1, #figura2, #figura3, #figura4, #figura5; names: Cambiar Color, Apagar luces, Activar sonido,
Agregar Objeto, Quitar Techo">
  <a-sphere position="-0.2 1.5 -0.5" radius="0.4" material="color: #654321; side: double; transparent: true; opacity: 0.4">
    <a-entity option="figure: #figura1; name: Cambiar Color; pinchable: true; scale: 0.02 0.02 0.02; text-scale: 25 25 25; text-position: 0 6 0"
    position="-0.25 0 0" change-wall-color></a-entity>
    <a-entity option="figure: #figura4; name: Apagar luces; pinchable: true; scale: 0.1 0.1 0.1; text-scale: 7 7 7; text-position: 0 1.3 0"
    position="0 0 0" toggle-light></a-entity>
    <a-entity option="figure: #figura3; name: Activar sonido; pinchable: true; scale: 0.3 0.3 0.3; text-scale: 3 3 3; text-position: 0 0.3 0"
    position="0.25 0 0" sound-effects></a-entity>
    <a-entity option="figure: #figura5; name: Quitar Techo; pinchable: true; scale: 0.01 0.01 0.01; text-scale: 55 55 55; text-position: 0 9 0"
    position="0 0.25 0" remove-roof></a-entity>
    <a-entity option="figure: #figura2; name: Agregar Objeto; pinchable: true; scale: 0.25 0.25 0.25; text-scale: 3 3 3; text-position: 0 0.5 0"
    position="0 -0.25 0" toggle-box></a-entity>
  </a-sphere>
</a-entity>
```

Figura 4.6: Crear menú principal

4. **Configurar Componentes actuadores o menús específicos:** Cada opción en el menú se configura con un componente interactivo que define su comportamiento específico. Por ejemplo, el componente toggle-light para encender/apagar luces, se puede asociar un componente menú si se quiere añadir un submenú cuando se interactúe con el objeto. Los componentes interactivos con añadirlos es suficiente, mientras que los menús hay que rellenar el parámetro `position` para situar al submenú en la posición que queramos y es importante recordar que si queremos utilizar el componente configurador en la escena hay que usar el componente creador para crear los objetos que el configurador va a personalizar.

```
<!-- Menú principal -->
<a-entity position="0 0 0" shelf="objects: #figura1, #figura2, #figura3, #figura4, #figura5; names: Cambiar Color, Apagar luces, Activar sonido,
Agregar Objeto, Quitar Techo">
  <a-sphere position="-0.2 1.5 -0.5" radius="0.4" material="color: #654321; side: double; transparent: true; opacity: 0.4">
    <a-entity option="figure: #figura1; name: Cambiar Color; pinchable: true; scale: 0.02 0.02 0.02; text-scale: 25 25 25; text-position: 0 6 0"
    position="-0.25 0 0" change-wall-color></a-entity>
    <a-entity option="figure: #figura4; name: Apagar luces; pinchable: true; scale: 0.1 0.1 0.1; text-scale: 7 7 7; text-position: 0 1.3 0"
    position="0 0 0" menu-light="position: -0.75 1.6 -0.5"></a-entity>
    <a-entity option="figure: #figura1; name: Box; pinchable: true; scale: 0.20 0.20 0.20; text-scale: 3 3 3; text-position: 0 0.5 0"
    creador="type: box" position="-0.2 0 0"></a-entity>
  </a-sphere>
</a-entity>
```

Figura 4.7: Ejemplo menú principal con componente interactivo, submenú y creador

4.3. Detalles de implementación

Esta sección está orientada a desarrolladores que desean entender y posiblemente modificar la implementación del proyecto. A continuación, se proporciona una descripción detallada de todos los componentes desarrollados, centrándose en su implementación técnica. Aunque ya se ha explicado la funcionalidad y la composición de estos componentes, ahora se analizarán sus estructuras internas y los métodos utilizados para lograr sus comportamientos específicos.

En A-Frame, un componente es una unidad de código reutilizable que define el comportamiento y las propiedades de los objetos en una escena de realidad virtual.

Se construyen utilizando un objeto en el que se redefinen las propiedades y métodos necesarios para su funcionamiento. Cada componente incluye un `schema`, que define los parámetros que el componente aceptará, permitiendo configurar adecuadamente sus propiedades y comportamiento. El método `init` se invoca cuando la entidad que contiene el componente se inicializa, estableciendo la configuración inicial del componente y preparando cualquier lógica necesaria para su operación. Por último, el método `bindMethods` se utiliza para asegurar que el contexto de `this` se mantenga correcto en los métodos que se asignan a eventos, garantizando que las funciones se ejecuten en el contexto adecuado. En programación, `this` se refiere al objeto actual desde el cual se está llamando un método o accediendo a una propiedad, asegurando que el código funcione correctamente dentro del contexto del componente.

4.3.1. Componente shelf

Schema

Los parámetros son `objects` y `names` que son arrays que contienen las referencias a las figuras 3D y sus nombres correspondientes. Estos parámetros son esenciales para configurar correctamente el menú

Método ‘init”

Primero, el método asigna los valores de los parámetros objects y names a las variables locales objects y names respectivamente.

A continuación, se realiza una validación de longitud para asegurarse de que la cantidad de objetos y nombres coinciden. Si las longitudes de los arrays `objects` y `names` no son iguales, se muestra un mensaje de error en la consola y el método termina su ejecución con return. Esta validación es importante para prevenir inconsistencias y errores posteriores en la escena, garantizando que cada figura 3D tenga un nombre asociado.

Finalmente, el método almacena las figuras y nombres permitidos como atributos de la entidad shelf.

```
init: function () {
  var objects = this.data.objects;
  var names = this.data.names;

  if (objects.length !== names.length) {
    console.error('La cantidad de objetos y nombres no coincide');
    return;
  }
  // Almacena las figuras y nombres permitidos en el elemento shelf para acceso global
  this.el.setAttribute('allowed-objects', objects.join(','));
  this.el.setAttribute('allowed-names', names.join(','));
```

Figura 4.8: Método init componentes shelf

4.3.2. Componente option

schema

El *schema* define los parámetros que el componente aceptará. En este caso, *figure* y *name* son parámetros de tipo *string*, que representan la referencia al modelo GLTF y su nombre, respectivamente. El parámetro *pinchable* es un booleano que indica si el objeto puede ser manipulado mediante gestos de pinza. Además, se han añadido tres parámetros adicionales: *scale*, *textScale* y *textPosition*, todos de tipo *string*. *scale* define la escala del modelo GLTF, *textScale* define la escala del texto que se mostrará encima del modelo, y *textPosition* define la posición del texto en relación con el modelo. Estos parámetros son esenciales para configurar correctamente la representación visual y la interactividad del objeto en la escena de A-Frame.

Método *init*

Primero, el método asigna los valores de los parámetros definidos en el *schema* a la variable *data*. También se guarda una referencia a la entidad (*el*) y se inicializa una variable *localPosition* como una nueva instancia de *THREE.Vector3*. A continuación, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos.

El método busca el elemento *shelf* más cercano en la jerarquía del DOM. Si no encuentra un elemento *shelf*, muestra un mensaje de error en la consola y termina la ejecución del método. Esta validación asegura que cada *option* esté asociada a un *shelf* válido.

Se añade un *listener* de eventos para esperar a que el componente *shelf* esté completamente inicializado. Una vez que se inicializa, se obtienen los objetos permitidos desde el atributo *allowed-objects* del *shelf*. Si la figura especificada en los datos del componente no está en la lista de objetos permitidos, se muestra un mensaje de error y el método termina su ejecución. Esta validación es crucial para asegurar que solo se creen entidades que están permitidas en el contexto del *shelf*.

Si la figura es permitida, se crea una entidad *a-entity* que representa el modelo GLTF, apli-

cando la escala proporcionada en los datos del componente. Luego, se crea otra entidad *a-entity* para el texto, configurando su valor, alineación, lado, posición y escala según los datos del componente. Este texto se añade como hijo de la entidad del modelo GLTF.

Finalmente, si el parámetro *pinchable* es verdadero, se añade el componente *pinchable* a la entidad del modelo y se configura un *listener* para el evento *pinchedmoved*, llamando al método *onPinchedMoved* cuando se detecta el evento. Esta configuración permite que el objeto sea interactivo mediante gestos de pinza

Método *onPinchedMoved*

El método *onPinchedMoved* (código en figura 4.9) se ejecuta cuando se detecta que el objeto ha sido movido mediante un gesto de pinza. Se copia la posición global del modelo GLTF al sistema de coordenadas local del objeto, actualizando así la posición del objeto en todas las coordenadas (X, Y, Z). Este método permite que el modelo GLTF se mueva correctamente en la escena cuando se interactúa con él mediante gestos de pinza.

```
bindMethods: function () {
  this.onPinchedMoved = this.onPinchedMoved.bind(this);
},

onPinchedMoved: function (evt) {
  var el = evt.target;
  var localPosition = this.localPosition;

  localPosition.copy(evt.detail.position);
  this.el.object3D.updateMatrixWorld();
  this.el.object3D.worldToLocal(localPosition);

  // Actualizar la posición del objeto en todas las coordenadas (X, Y, Z)
  el.object3D.position.copy(localPosition);
}
```

Figura 4.9: Método onPinchedMoved

4.3.3. Componente pinchable

schema

En este caso, el parámetro *pinchDistance* tiene un valor por defecto de 0.1. Este parámetro especifica la distancia mínima a la que debe estar el gesto de pinza para que se considere que ha iniciado una interacción con el objeto.

Método *init*

Primero, se obtiene una referencia a la escena (*sceneEl*) y se inicializa una variable *worldPosition* como una nueva instancia de *THREE.Vector3*. Además, se inicializa una variable *pinched* en *false* para rastrear el estado de la interacción.

A continuación, se añaden listeners para los eventos *pinchstarted*, *pinchended* y *pinchmoved* a la escena. Estos eventos permiten que el componente reaccione a los gestos de pinza que ocurren en la escena, iniciando, terminando o moviendo la interacción.

Método *onPinchStarted*

El método *onPinchStarted* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Calcula la distancia de la pinza utilizando el método *calculatePinchDistance*. Si esta distancia es menor que el valor especificado en *pinchDistance*, se emite un evento *pinchedstarted*, se marca la variable *pinched* como *true*.

Método *calculatePinchDistance*

El método *calculatePinchDistance* calcula la distancia entre la posición del objeto en el mundo y la posición de la pinza en el mundo. Para ello, copia la posición del objeto en el mundo y la convierte al sistema de coordenadas del mundo utilizando *updateMatrixWorld* y *localToWorld*. Luego, calcula y devuelve la distancia entre estas dos posiciones. Este cálculo clave para determinar si la pinza está lo suficientemente cerca del objeto para iniciar una interacción.

```
calculatePinchDistance: function (pinchWorldPosition) {
  var el = this.el;
  var worldPosition = this.worldPosition;
  var pinchDistance;

  worldPosition.copy(el.object3D.position);
  el.object3D.parent.updateMatrixWorld();
  el.object3D.parent.localToWorld(worldPosition);

  pinchDistance = worldPosition.distanceTo(pinchWorldPosition);

  return pinchDistance;
},
```

Figura 4.10: Método calculatePinchDistance

Método *onPinchEnded*

El método *onPinchEnded* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *pinched* es *true*, se marca como *false* y se emite un evento *pinchedended*. Esto permite que el objeto responda adecuadamente al final de una interacción de pinza.

Método *onPinchMoved*

El método *onPinchMoved* se ejecuta cuando se detecta que un gesto de pinza ha movido el objeto. Calcula la distancia de la pinza utilizando el método *calculatePinchDistance*. Si la variable *pinched* es *false*, el método termina su ejecución. Si la distancia de la pinza es menor

que el valor especificado en *pinchDistance*, se emite un evento *pinchedmoved* con los detalles del evento. Si la distancia es mayor, se marca la variable *pinched* como *false* y se emite un evento *pinchedended*. Esto asegura que el objeto responda correctamente a los movimientos de la pinza y termine la interacción si la pinza se aleja demasiado.

```
onPinchMoved: function (evt) {
  var el = this.el;
  var pinchDistance = this.calculatePinchDistance(evt.detail.position);
  if (!this.pinched) { return; }
  if (pinchDistance < this.data.pinchDistance) {
    el.emit('pinchedmoved', evt.detail);
  } else {
    this.pinched = false;
    el.emit('pinchedended');
  }
```

Figura 4.11: Método onPinchEnded

4.3.4. Componente menu-light

Los componentes que crean submenús (menu-light, menu-music, menu-object, menu-wall-color y menu-wall) en la escena se implementan de manera similar. Por tanto, explicaré la implementación de uno de ellos.

schema

El parámetro *position* es de tipo *string*, este parámetro especifica la posición en la que se ubicará la esfera del menú en la escena.

Método *init*

Primero, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos. Luego, se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *toggleMenu* cuando se detecte el evento. Además, se inicializan las variables *cooldown* en *false* y *menuSphere* en *null* para gestionar el estado del menú.

Método *toggleMenu*

El método *toggleMenu* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la escena (*sceneEl*). Si ya existe una esfera del menú (*menuSphere*), se elimina de la escena y se establece *menuSphere* en *null*. Si no, se crea una nueva entidad *shelf* y se configuran sus atributos con los objetos y nombres permitidos.

Se crea una entidad *menuSphere* con una geometría de esfera, material transparente y la posición especificada en los datos del componente. A continuación, se añade un texto encima de la esfera con las propiedades de alineación, color y ancho configuradas.

Se crean tres entidades option de submenú, cada una representada por una entidad configurada con los atributos correspondientes, como la figura, el nombre, la interactividad (*pinchable*), la escala, la escala del texto y la posición del texto. Cada opción tiene un componente específico para la acción correspondiente, como *toggle-light*, *toggle-red-light* y *toggle-light-intensity*.

```
var option1 = document.createElement('a-entity');
option1.setAttribute('option',
  'figure: #figura1; name: Apagar/Encender luz; pinchable: true; scale: 0.15 0.15 0.15; text-scale: 2 2 2; text-position: 0 0.5 0');
option1.setAttribute('position', '0 0.1 0');
option1.setAttribute('toggle-light', '');

var option2 = document.createElement('a-entity');
option2.setAttribute('option',
  'figure: #figura2; name: Luz roja; pinchable: true; scale: 0.15 0.15 0.15; text-scale: 2 2 2; text-position: 0 0.5 0');
option2.setAttribute('position', '0 -0.1 0');
option2.setAttribute('toggle-red-light', '');

var option3 = document.createElement('a-entity');
option3.setAttribute('option',
  'figure: #figura3; name: Bajar intensidad luz; pinchable: true; scale: 0.15 0.15 0.15; text-scale: 2 2 2; text-position: 0 0.5 0');
option3.setAttribute('position', '0 0 0.1');
option3.setAttribute('toggle-light-intensity', '');
```

Figura 4.12: Entidades option

Estas opciones se añaden como hijos de *menuSphere*, y *menuSphere* se añade como hijo de la entidad *shelf*. Finalmente, *shelf* se añade a la escena y se almacena la referencia de *menuSphere*.

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.5. Componente toggle-light

Método *init*

Se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto. A continuación, se añaden listeners para los eventos *pinchedstarted* y *pinchedended* en la entidad, que llamarán a los métodos *toggleLightOff* y *toggleLightOn*, respectivamente, cuando se detecten los eventos.

Método *toggleLightOff*

El método *toggleLightOff* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad de luz de la habitación (*roomLight*) utilizando *document.querySelector*. Se comprueba si la luz está encendida verificando si la intensidad de

la luz es mayor que 0. Si la luz está encendida, se establece su intensidad a 0 para apagarla.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

Método *toggleLightOn*

El método *toggleLightOn* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad de luz de la habitación (*roomLight*) utilizando *document.querySelector*. Se comprueba si la luz está encendida verificando si la intensidad de la luz es mayor que 0. Si la luz está apagada, se establece su intensidad a 0.85 para encenderla.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.6. Componente remove-roof

Método *init*

Se añaden listeners para los eventos *pinchedstarted* y *pinchedended* en la entidad, que llamarán a los métodos *hideRoof* y *showRoof*, respectivamente, cuando se detecten los eventos. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto.

Método *hideRoof*

El método *hideRoof* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad del techo (*ceiling*) utilizando *document.querySelector*. Si la entidad existe, se establece su atributo *visible* en *false* para ocultar el techo.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

Método *showRoof*

El método *showRoof* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad del techo (*ceiling*) utilizando *document.querySelector*. Si la entidad existe, se establece su atributo *visible* en *true* para mostrar el techo.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.7. Componente change-wall-color

Método *init*

Se añaden listeners para los eventos *pinchedstarted* y *pinchedended* en la entidad, que llamarán a los métodos *changeToNewColor* y *changeToOriginalColor*, respectivamente, cuando se detecten los eventos. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto, y se definen los colores *originalColor* y *newColor*.

Método *changeToNewColor*

El método *changeToNewColor* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a las entidades de las paredes (*walls*) utilizando *document.querySelectorAll* para seleccionar *wall1*, *wall2*, *wall3* y *wall4*. Para cada pared, se cambia el atributo *material* para establecer el color a *newColor*, cambiando así el color de las paredes a azul cian.

Método *changeToOriginalColor*

El método *changeToOriginalColor* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a las entidades de las paredes (*walls*) utilizando *document.querySelectorAll* para seleccionar *wall1*, *wall2*, *wall3* y *wall4*. Para cada pared, se cambia el atributo *material* para establecer el color a *originalColor*, devolviendo así las paredes a su color original.

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.8. Componente toggle-box

Método init

Se añaden listeners para los eventos *pinchedstarted* y *pinchedended* en la entidad, que llamarán a los métodos *addBox* y *removeBox*, respectivamente, cuando se detecten los eventos. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto, y se inicializa la variable *box* en *null* para almacenar la referencia de la caja creada.

Método *addBox*

El método *addBox* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la escena (*sceneEl*) y se crea una nueva entidad *a-box*. Se configuran sus atributos de posición, anchura, altura, profundidad y material (color). La caja se añade a la escena mediante *sceneEl.appendChild(this.box)*.

```
var sceneEl = this.el.sceneEl;
this.box = document.createElement('a-box');
this.box.setAttribute('position', { x: 0.6, y: 0.4, z: -1 });
this.box.setAttribute('width', 0.4);
this.box.setAttribute('height', 0.4);
this.box.setAttribute('depth', 0.4);
this.box.setAttribute('material', 'color', '#4CC3D9');
sceneEl.appendChild(this.box);
```

Figura 4.13: Crear entidad caja y añadir a la escena

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

Método *removeBox*

El método *removeBox* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Si la variable *box* no es *null*, la caja se elimina de la escena utilizando *this.box.parentNode.removeChild(this* y se establece *box* en *null*.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.9. Componente sound-effects

Método *init*

Primero, se obtiene una referencia al elemento de sonido (*grabSound*) utilizando *document.querySelector* para seleccionar el elemento con el ID *grab-sound*. Luego, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto.

A continuación, se añaden listeners para los eventos *pinchedstarted* y *pinchedended* en la entidad, que llamarán a los métodos *toggleGrabSoundOn* y *toggleGrabSoundOff*, respectivamente, cuando se detecten los eventos.

Método *toggleGrabSoundOn*

El método *toggleGrabSoundOn* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Si el sonido (*grabSound*) está pausado, se reinicia el tiempo de reproducción (*currentTime* a 0) y se reproduce el sonido utilizando *this.grabSound.play()*.

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

Método *toggleGrabSoundOff*

El método *toggleGrabSoundOff* se ejecuta cuando se detecta que un gesto de pinza ha terminado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Si el sonido (*grabSound*) no está pausado, se pausa el sonido utilizando *this.grabSound.pause()* y se reinicia el tiempo de reproducción (*currentTime* a 0).

Un temporizador de 100 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.10. Componentes que cambian el color de las paredes

Hay 3 componentes que pintan en las paredes un color determinado, *change-wall-color-blue*, *change-wall-color-green*, *change-wall-color-yellow* voy a explicar uno porque los tres están implementados de la misma manera.

Primero, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos. Luego, se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *changeColor* cuando se detecte el evento. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto, y se define el color *color* como azul cian.

Método *changeColor*

El método *changeColor* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a las entidades de las paredes (*walls*) utilizando *document.querySelectorAll* para seleccionar *wall1*, *wall2*, *wall3* y *wall4*. Para cada pared, se cambia el atributo *material* para establecer el color a *color*, cambiando así el color de las paredes a azul cian.

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.11. Componentes que eliminan paredes

Hay tres componentes que eliminan una pared determinada, *remove-back-wall*, *remove-front-wall*, *remove-roof* voy a explicar uno porque los tres están implementados de la misma manera.

Método *init*

Primero, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos. Luego, se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *toggleWall* cuando se detecte el evento. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto.

Método *toggleWall*

El método *toggleWall* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad de la pared trasera (*backWall*) utilizando *document.querySelector* para seleccionar *wall3*. Si la entidad existe, se obtiene su atributo *visible*

para determinar si la pared es visible o no. A continuación, se alterna el valor del atributo *visible* para ocultar o mostrar la pared.

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.12. Componentes que activan sonidos

Hay 3 componentes que gestionan los sonidos, *sound-mario-bros*, *sound-mario-jump*, *sound-stranger-things* voy a explicar uno porque los tres están implementados de la misma manera.

Primero, se obtiene una referencia al elemento de sonido (*grabSound*) utilizando *document.querySelector* para seleccionar el elemento con el ID *mario-sound*. A continuación, se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *toggleSound* cuando se detecte el evento. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto.

Método *toggleSound*

El método *toggleSound* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Si el sonido (*grabSound*) está pausado, se reinicia el tiempo de reproducción (*currentTime* a 0) y se reproduce el sonido utilizando *this.grabSound.play()*. Si el sonido no está pausado, se pausa el sonido utilizando *this.grabSound.pause()* y se reinicia el tiempo de reproducción (*currentTime* a 0).

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.13. Componentes que crean objetos

Hay 3 componentes que crean objetos en la escena, *toggle-box*, *toggle-cone*, *toggle-cylinder* voy a explicar uno porque los tres están implementados de la misma manera.

Método *init*

Primero, se llama al método *bindMethods* para asegurar que el contexto de *this* se mantenga correcto en los métodos que se asignan a eventos. Luego, se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *toggleBox* cuando se detecte el evento. Además, se inicializan las variables *cooldown* en *false* para gestionar el estado de activación del gesto, y *box* en *null* para almacenar la referencia de la caja creada.

Método *toggleBox*

El método *toggleBox* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la escena (*sceneEl*). Si la variable *box* no es *null*, la caja se elimina de la escena utilizando *sceneEl.removeChild(this.box)* y se establece *box* en *null*. Si la variable *box* es *null*, se crea una nueva entidad *a-box* y se configuran sus atributos de posición, anchura, altura, profundidad y material (color). La caja se añade a la escena mediante *sceneEl.appendChild(box)* y se almacena la referencia de la caja en la variable *box*.

```
var sceneEl = this.el.sceneEl;
if (this.box) {
  sceneEl.removeChild(this.box);
  this.box = null;
} else {
  var box = document.createElement('a-box');
  box.setAttribute('position', { x: 0, y: 0.4, z: -2 });
  box.setAttribute('width', 0.4);
  box.setAttribute('height', 0.4);
  box.setAttribute('depth', 0.4);
  box.setAttribute('material', 'color', '#4CC3D9');
  sceneEl.appendChild(box);
  this.box = box;
}
```

Figura 4.14: Creación de objeto

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.14. Componentes que gestionan la iluminación

Hay 3 componentes que gestionan la intensidad de la iluminación de la escena, *toggle-light-intensity*, *toggle-light*, *toggle-red-light* voy a explicar uno porque los tres están implementados de la misma manera.

Método *init*

Se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *toggleIntensity* cuando se detecte el evento. Además, se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto.

Método *toggleIntensity*

El método *toggleIntensity* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activacio-

nes en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la entidad de la luz de la habitación (*roomLight*) utilizando *document.querySelector* para seleccionar *room-light*. Se obtiene la intensidad actual de la luz (*currentIntensity*) y se alterna el valor de la intensidad. Si la intensidad actual es mayor que 0.5, se establece la intensidad en 0.5. Si es menor o igual a 0.5, se establece la intensidad en 0.85.

```
var roomLight = document.querySelector('#room-light');
var currentIntensity = roomLight.getAttribute('light').intensity;

if (currentIntensity > 0.5) {
  roomLight.setAttribute('light', 'intensity', 0.5);
} else {
  roomLight.setAttribute('light', 'intensity', 0.85);
}
```

Figura 4.15: Gestión iluminación

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

4.3.15. Componente creador

schema

El parámetro *type* es de tipo *string* y especifica el tipo de geometría (por ejemplo, *box*, *sphere*, *cylinder*, etc.) que se creará en la escena. Esta definición permite personalizar el tipo de objeto que se generará.

Método *init*

Se inicializan las variables *cooldown* en *false* para gestionar el estado de activación del gesto, y *createdObject* en *null* para almacenar la referencia del objeto creado. Se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *createObject* cuando se detecte el evento.

Método *createObject*

El método *createObject* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia a la escena (*sceneEl*). Si ya existe un objeto creado (*window.currentCreatedObject*), se elimina de la escena y se establece *window.currentCreatedObject* en *null*.

```
var sceneEl = this.el.sceneEl;

// Si ya existe un objeto creado, elimínalo
if (window.currentCreatedObject) {
  window.currentCreatedObject.parentNode.removeChild(window.currentCreatedObject);
  window.currentCreatedObject = null;
}
```

Figura 4.16: Eliminar objeto creado

A continuación, se crea una nueva entidad *a-entity* y se configuran sus atributos de geometría (utilizando el tipo especificado en los datos del componente), escala, posición y material (color). La entidad se añade a la escena mediante *sceneEl.appendChild(object)* y se guarda la referencia del objeto creado en la variable global *window.currentCreatedObject*.

```
// Crear el nuevo objeto
var object = document.createElement('a-entity');
object.setAttribute('geometry', `primitive: ${this.data.type}`);
object.setAttribute('scale', '0.5 0.5 0.5');
object.setAttribute('position', '0 1 -0.5');
object.setAttribute('material', 'color: #4CC3D9');
sceneEl.appendChild(object);

// Guardar la referencia al objeto creado en la variable global
window.currentCreatedObject = object;
```

Figura 4.17: Crear objeto y referenciar variable global

Un temporizador de 300 milisegundos se establece para desactivar el estado de

4.3.16. Componente configurador

schema

En este caso, el parámetro *action* es de tipo *string* y especifica la acción que se aplicará al objeto seleccionado. Las acciones pueden incluir *color*, *scale* o *rotate*.

Método *init*

Se inicializa la variable *cooldown* en *false* para gestionar el estado de activación del gesto, y se añade un listener para el evento *pinchedstarted* en la entidad, que llamará al método *applyConfiguration* cuando se detecte el evento.

Método *applyConfiguration*

El método *applyConfiguration* se ejecuta cuando se detecta que un gesto de pinza ha comenzado. Si la variable *cooldown* es *true*, el método termina su ejecución para evitar múltiples activaciones en poco tiempo. Si no, se establece *cooldown* en *true* para activar el tiempo de espera.

Luego, se obtiene una referencia al objeto 3D actualmente creado y almacenado en la variable global *window.currentCreatedObject*. Si no hay un objeto seleccionado, se muestra una advertencia en la consola y se desactiva el estado de *cooldown*.

Dependiendo del valor de *this.data.action*, se llama al método correspondiente para cambiar el color (*changeColor*), escalar (*changeScale*) o rotar (*rotateObject*) el objeto. Si la acción no es reconocida, se muestra una advertencia en la consola.

Un temporizador de 300 milisegundos se establece para desactivar el estado de *cooldown*, permitiendo nuevas activaciones después de un breve período de espera.

Método *changeColor*

El método *changeColor* cambia el color del objeto seleccionado. Primero, se define un array de colores (*colors*). Luego, se obtiene el color actual del objeto y se determina su índice en el arreglo de colores. Se selecciona el siguiente color en el arreglo y se aplica al objeto utilizando *object.setAttribute('material', 'color', newColor)*.

```
changeColor: function (object) {
  var colors = ['#FF0000', '#00FF00', '#0000FF', '#FFFF00', '#FF00FF', '#00FFFF'];
  var currentColor = object.getAttribute('material').color;
  var currentIndex = colors.indexOf(currentColor);
  var newColor = colors[(currentIndex + 1) % colors.length];
  object.setAttribute('material', 'color', newColor);
},
```

Figura 4.18: Cambiar color objeto

Método *changeScale*

El método *changeScale* aumenta la escala del objeto seleccionado en 0.1 en cada dirección (x, y, z). Se obtiene la escala actual del objeto y se calcula la nueva escala. Luego, se aplica la nueva escala al objeto utilizando *object.setAttribute('scale', newScale)*.

```
changeScale: function (object) {
  var currentScale = object.getAttribute('scale');
  var newScale = {
    x: currentScale.x + 0.1,
    y: currentScale.y + 0.1,
    z: currentScale.z + 0.1
  };
  object.setAttribute('scale', newScale);
},
```

Figura 4.19: Aumentar tamaño objeto

Método *rotateObject*

El método *rotateObject* rota el objeto seleccionado 45 grados en el eje y. Se obtiene la rotación actual del objeto y se calcula la nueva rotación. Luego, se aplica la nueva rotación al objeto utilizando *object.setAttribute('rotation', newRotation)*.

```
rotateObject: function (object) {
  var currentRotation = object.getAttribute('rotation');
  var newRotation = {
    x: currentRotation.x,
    y: currentRotation.y + 45,
    z: currentRotation.z
  };
  object.setAttribute('rotation', newRotation);
```

Figura 4.20: Rotar objeto

Esta sección proporciona una visión completa de la implementación de los componentes. Los desarrolladores pueden utilizar esta información para entender y modificar la implementación según sus necesidades. La explicación de cómo se componen estos componentes se encuentra en la sección 4.2.1. Con estas dos explicaciones, se puede comprender cómo usar los componentes entre sí y lo que hacen internamente a un nivel más bajo.

Capítulo 5

Conclusiones

5.1. Consecución de objetivos

El principal resultado de este proyecto es una caja de herramientas, que es un conjunto de componentes que trabaja con el seguimiento de manos. Además, se ha cumplido el objetivo de trabajar en A-Frame, que es el entorno utilizado. Este proyecto permite a los usuarios interactuar con el entorno virtual utilizando sus manos de manera natural y precisa.

Para alcanzar este objetivo principal, se han completado los siguientes objetivos instrumentales:

1. **Creación de entornos virtuales:** Se han desarrollado entornos virtuales en navegadores web que permiten a los usuarios experimentar y explorar espacios tridimensionales de manera inmersiva. Estos entornos proporcionan la base para la interacción avanzada mediante el seguimiento de manos.

2. **Creación de menús y submenús:** Se ha diseñado una estructura de menús y submenús intuitiva y fácil de usar. Esta estructura permite a los usuarios navegar y seleccionar opciones de manera eficiente.

3. **Desarrollo de componentes con A-Frame:** Se han creado y personalizado componentes en A-Frame para construir interfaces de usuario eficientes y efectivas. Estos componentes permiten realizar tareas específicas dentro de las escenas, como cambiar colores, añadir objetos y controlar la iluminación, todo ello mediante gestos de la mano.

4. **Creación de prototipos:** Para comprobar que el conjunto de componentes funciona de manera integrada, se han realizado varios prototipos.

En resumen, este proyecto no solo ha cumplido con sus objetivos principales e instrumentales, sino que también ha demostrado el potencial de las tecnologías de seguimiento de manos y realidad virtual en la creación de interfaces de usuario interactivas y avanzadas en entornos web.

5.2. Utilización de recursos

Los recursos fundamentales en este trabajo han sido para desarrollo y para prueba de la realidad virtual. Para el desarrollo se ha utilizado mi ordenador de sobremesa en casa, y para las pruebas, el dispositivo de realidad virtual empleado ha sido las Meta Quest 2.

Material utilizado

- **Ordenador**: Utilizado para el desarrollo del proyecto.
- **Gafas Meta Quest**: Utilizadas para las pruebas de la realidad virtual.

El otro recurso principal ha sido el tiempo dedicado al desarrollo y aprendizaje, que se describe en el cuadro 5.1.

Tiempo	Sprint 0	Sprint 1	Sprint 2	Sprint 3
Horas de esfuerzo	65	40	30	60
Días de calendario	90	20	15	30

Cuadro 5.1: Proceso temporal

5.3. Aplicación de lo aprendido

Para realizar este proyecto se ha partido de algunos conocimientos adquiridos durante el grado.

1. **Expresión oral y escrita y búsqueda de la información:** Esta asignatura me ha ayudado a aplicar técnicas de presentación de documentos técnicos y aprender técnicas tanto de expresión oral como escrita que me han ayudado a la elaboración de la memoria y a la presentación de mi proyecto.

2. **Construcción de servicios y aplicaciones audiovisuales en internet:** Esta asignatura me introdujo a los conceptos básicos de las tecnologías web, aprendiendo los fundamentos de HTML y CSS. Más adelante los mecanismos del lenguaje JavaScript.

3. **Informática I:** Aquí aprendí las bases de la lógica de programación.

4. **Informática II:** En esta asignatura continué con mi formación sobre la programación aprendiendo depurar código, un código más limpio y eficiente.

5.4. Lecciones aprendidas

Durante este proyecto he adquirido diferentes conocimientos.

- Trabajar con dispositivos de realidad virtual.
- Aumentar mis conocimientos de lenguajes de programación para front-end
- Construir componentes para la creación de elementos en entornos de realidad virtual con A-Frame.
- El uso del seguimiento de manos.
- El uso de metodologías ágiles para gestionar este proyecto.
- Realizar memorias técnicas
- El uso de LaTeX para escribir la memoria

5.5. Trabajos futuros

Esta tecnología es muy moderna y por supuesto que este proyecto no está acabado. Al final he explorado el uso de seguimiento de manos con menús muy simples, pero se puede continuar explorando diferentes gestos de la mano para enriquecer los menús, no solamente agarrar, sino pulsar o deslizar, a continuación dejo una posible lista:

- Se podrían implementar gestos adicionales como pulsar, deslizar, girar, hacer Zoom, etc. Esto enriquecería significativamente la experiencia del usuario y permitiría una mayor variedad de interacciones.
- Permitir a los usuarios definir y personalizar sus propios gestos. Por ejemplo, un usuario podría definir un gesto específico para abrir un menú o realizar una acción específica dentro de la aplicación.
- Desarrollar capacidades de interacción multitáctil donde los usuarios puedan usar ambas manos simultáneamente para realizar acciones complejas, como manipular objetos en 3D, estirarlos o combinarlos.
- Crear juegos utilizando la tecnología de hand-tracking, permitiendo a los usuarios interactuar de manera intuitiva y envolvente. Los juegos pueden aprovechar los gestos para realizar acciones como lanzar objetos, resolver puzzles o controlar personajes, ofreciendo una experiencia de juego única y dinámica.

Bibliografía

[1] Xperimenta Cultura. Historia de la realidad virtual. https://xperimentacultura.com/historia-de-la-realidad-virtual/, 2024.

[2] A-Frame. A-Frame documentation. https://aframe.io/docs/1.6.0/introduction/, 2024.

[3] Three.js. Three.js documentation. https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene, 2024.

[4] MDN Web Docs. Webxr device api fundamentals. https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals, 2024.

[5] W3C. WebXR documentation. https://www.w3.org/TR/webxr-hand-input-1/, 2024.

[6] Khronos Group. WebGL 2.0 Specification. https://registry.khronos.org/webgl/specs/latest/2.0/, 2024.

[7] MDN Web Docs. WebGL API. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API, 2024.

[8] Tony Parisi. *WebGL Up and Running*. O'Reilly & Associates Inc., Sebastopol, CA, 2012.

[9] ECMA International. Ecmascript 2023 language specification. https://262.ecma-international.org/14.0/, 2023.

[10] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, Sebastopol, California, 7th edition, 2020.

[11] Gerd Wagner. *Building Front-End Web Apps with Plain JavaScript*. Web-Engineering.info, 2015. Available at https://e-booksdirectory.com/details.php?ebook=10585.

[12] Emilio A. Franganillo. Html5: Guía para desarrolladores web. https://franganillo.es/html5.pdf, 2024.

[13] W3Schools. HTML Tutorial. https://www.w3schools.com/html/default.asp, 2024.

[14] Microsoft. Visual studio code documentation. https://code.visualstudio.com/docs, 2024.

[15] LaTeX Project. LaTeX documentation. https://www.latex-project.org/help/documentation/, 2024.

[16] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1994.

[17] Carrera Profesional. Desarrollo agile. https://carreraprofesional.com/tutorial/desarrollo-agile/, 2024.

[18] A-Frame. Hand Tracking Controls. https://aframe.io/docs/1.6.0/components/hand-tracking-controls.html, 2024.