



ESCUELA DE INGENIERÍA DE FUENLABRADA

INGENIERÍA EN SISTEMAS DE
TELECOMUNICACIONES

TRABAJO FIN DE GRADO

DESARROLLO DE INTERFACES DE USUARIO PARA
REALIDAD EXTENDIDA CON CONTROL POR GESTOS

Autor : Alberto Sánchez Santos

Tutor : David Moreno Lumbreras

Curso académico 2024/2025

©2025 Alberto Sánchez Santos

Algunos derechos reservados.

Este documento se distribuye bajo la licencia

“Atribución-CompartirIgual 4.0 Internacional”

de Creative Commons, disponible en

<https://creativecommons.org/licenses/by-sa/4.0/deed.es>

Resumen

Este Trabajo de Fin de Grado presenta el diseño y desarrollo de un menú interactivo y personalizable basado en la librería A-Frame, dirigido a entornos de realidad virtual (VR) y realidad extendida (XR). El objetivo es crear una librería modular que pueda ser fácilmente importada y utilizada por la comunidad de desarrolladores, facilitando la incorporación de interfaces gráficas funcionales y adaptables en proyectos inmersivos. El menú está estructurado mediante botones que permiten la navegación entre distintos niveles o submenús, con la posibilidad de personalizar el número de botones en cada submenú, desde uno hasta cuatro. Esta configuración flexible busca adaptarse a diversas necesidades de interacción y organización de contenido, ofreciendo una experiencia intuitiva para el usuario final.

El componente se empaquetará como un archivo npm, lo que facilitará su integración en proyectos web que utilicen A-Frame, y será compatible con múltiples dispositivos de realidad extendida. Para validar su funcionalidad y fomentar su adopción, se desarrollarán demostraciones interactivas junto con documentación técnica detallada, que guiará a los desarrolladores en la implementación y personalización del menú.

Este proyecto aborda la actual carencia de repositorios actualizados que ofrezcan una interfaz gráfica modular y personalizable para menús en A-Frame, proporcionando una solución moderna y accesible que simplifica la creación de interfaces en aplicaciones de VR y XR.

Abstract

This Final Degree Project presents the design and development of an interactive and customizable menu based on the A-Frame library, targeting virtual reality (VR) and extended reality (XR) environments. The objective is to create a modular library that can be easily imported and used by the developer community, facilitating the incorporation of functional and adaptable graphical interfaces into immersive projects. The menu is structured using buttons that allow navigation between different levels or submenus, with the ability to customize the number of buttons in each submenu, from one to four. This flexible configuration aims to adapt to diverse interaction and content organization needs, providing an intuitive experience for the end user.

The component will be packaged as an npm file, facilitating its integration into web projects using A-Frame, and will be compatible with multiple extended reality devices. To validate its functionality and encourage adoption, interactive demos will be developed along with detailed technical documentation, guiding developers through the implementation and customization of the menu.

This project addresses the current lack of up-to-date repositories that offer a modular and customizable graphical interface for A-Frame menus, providing a modern and accessible solution that simplifies the creation of interfaces in VR and XR applications.

Índice general

1. Introducción	11
1.1. Contexto	12
1.2. Objetivos	13
1.2.1. Objetivos generales	13
1.2.2. Objetivos específicos	14
1.3. Estructura de la memoria	14
2. Tecnologías utilizadas	17
2.1. Desarrollo Web	17
2.2. Frameworks y Bibliotecas	20
2.3. Entornos de Desarrollo	21
2.4. Realidad Extendida	22
2.5. Desarrollo de Documentación	24
2.6. Modelado y Formatos 3D	24
3. Desarrollo del proyecto	27
3.1. Sprint 1: Exploración Inicial de A-Frame y JavaScript	27
3.1.1. Introducción a A-Frame y su integración con HTML	27
3.1.2. Eventos e interacciones básicas	29
3.1.3. Primer entorno: creación de una habitación virtual	30
3.2. Sprint 2: Desarrollo del Menú Estático	32
3.2.1. Primer diseño de interfaz fija	32
3.2.2. Incorporación de elementos interactivos	33
3.2.3. Utilización de las manos para la interacción	36

3.3.	Sprint 3: Experimentación con Menús Dinámicos	37
3.3.1.	Generación dinámica de menús	37
3.3.2.	Lectura de datos externos y generación del menú	37
3.3.3.	Gestión de eventos de botones y navegación	39
3.3.4.	Optimización y detección de errores en el sistema dinámico	40
3.4.	Sprint 4: Implementación Final y Optimización	41
3.4.1.	Objetivos y mejoras planteadas en este sprint	41
3.4.2.	Rediseño estructural del menú y modificación del formato JSON . .	42
3.4.3.	Nuevo sistema de detección de interacción	44
3.4.4.	Creación modular y escalable de botones y submenús	45
3.4.5.	Mejoras visuales: rediseño del menú e incorporación de imágenes .	46
4.	Descripción del resultado	49
4.1.	Interfaz de Usuario	49
4.1.1.	Características principales	49
4.1.2.	Requisitos técnicos	50
4.1.3.	Instalación y estructura de archivos	50
4.1.4.	Configuración de los menús	51
4.1.5.	Integración en la escena de A-Frame	52
4.1.6.	Parámetros de configuración disponibles	53
4.1.7.	Sistema de interacción	53
4.1.8.	Opciones de personalización	53
4.1.9.	Recomendaciones de uso	54
4.2.	Arquitectura del menú	54
4.2.1.	Estructura general del sistema	54
4.2.2.	Funcionamiento interno	55
4.2.3.	Diagrama de arquitectura	56
4.2.4.	Distribución de elementos en el espacio	58
4.2.5.	Personalización y escalabilidad	58
4.3.	Casos de Uso	59
4.3.1.	Demo 1: Habitación personalizable	59

ÍNDICE GENERAL	7
4.3.2. Demo 2: Menú en el desierto	61
4.3.3. Demo 3:Realidad aumentada	63
4.4. Disponibilidad del software	64
5. Conclusiones	67
5.0.1. Consecución de objetivos	67
5.0.2. Aplicación de lo aprendido	67
5.0.3. Lecciones aprendidas	68
5.0.4. Trabajos futuros	68
Bibliografía	71

Índice de figuras

3.1. Ejemplo básico de A-Frame	28
3.2. Generacion de habitacion sin texturas	32
3.3. Ejemplos de texturas	32
3.4. Ejemplo de objeto 3D	35
3.5. Ejemplo de uso de manos en realidad virtual	37
3.6. Diseño de Menu versión inicial	39
3.7. Sistema de vectores de la mano en A-frame	45
3.8. Diseño de menú versión final	48
4.1. Esquema de generación del menú	56
4.2. Esquema de interacción usuario-menú	57
4.3. Habitación con una cama	60

Capítulo 1

Introducción

El desarrollo de interfaces en entornos de realidad virtual (VR) y en realidad extendida (XR) se ha convertido en un campo de creciente interés dentro de la ingeniería y la informática, debido a su capacidad para ofrecer experiencias inmersivas, interactivas y altamente personalizables. Este Trabajo Fin de Grado se enmarca dentro de esta tendencia, proponiendo el diseño y construcción de un menú interactivo y personalizable en realidad virtual utilizando tecnologías web, en particular A-Frame y JavaScript.

Los motivos de la creación de este proyecto surge por la creciente necesidad de ir creando una serie de herramientas personalizables , que permitan enriquecer el entorno de los programadores , desarrolladores e investigadores , permitiéndoles centrarse en el desarrollo de sus actividades sin tener que preocuparse en otro aspectos mas técnicos , que van fuera de su área de desarrollo. Actualmente, muchos proyectos en A-Frame carecen de soluciones reutilizables que gestionen menús u opciones interactivas de forma clara y adaptable. Este trabajo pretende cubrir ese vacío mediante la construcción de una librería robusta, flexible y fácil de integrar.

El procesos de desarrollo del proyecto se ha basado en una metodología por ciclos , a lo largo de varios sprints , en lo que se ha pasado desde una exploración y entendimiento de las tecnologías hasta la creación y implementan de un sistema completo para la creación de un menú dinámico y personalizable. Se ha trabajado no solo en la parte funcional del menú sino en su integración como una herramienta para poder ser reutilizada por la comunidad mediante su despliegue con tecnologías como npm.

En esta memoria, se explican las diversas etapas del proyecto , se presentan en detalle

respaldadas por una base teórica, y las decisiones llevadas a cabo , sus razones y así como los resultados obtenido de estas. Además, se incluyen propuestas de mejora y líneas de trabajo futuras para continuar el progreso en la interfaz de realidad virtual a la que se puede acceder y personalizar.

1.1. Contexto

La interacción en entornos tridimensionales , ha sufrido un gran evolución en los últimos año con la aparición de nuevas tecnologías y la normalización de estas , y especialmente con el auge que han tenido la realidad aumentada y la realidad virtual. Estas nuevas tecnologías han permitido representar espacios en entornos inmersivos pudiéndose usar en distintos ámbitos ya sean académicos como la educación, entornos laborales como en la industria , la medicina y/o entornos personales como el entretenimiento.

En este contexto , la creación de entornos y interfaces gráficas se ha convertido en un punto clave , para esta industria , que a diferencia de los entornos clásicos digitales como las pantallas 2D , se requiere nuevos enfoques para adaptarnos a las nuevas oportunidades que surgen en formas de interacción , el movimiento del usuario o de los objetos así como adaptarse de un entorno en dos dimensiones a un entorno tridimensional.

A-Frame,es una librería basada en HTML y JavaScript , se ha convertido en una herramienta gratuita y poderosa para desarrollares de entornos XR y VR directamente a través de los navegadores navegadores web , ya que permite tener entornos de realidad virtual sin necesitar de herramientas complejas o entornos pesados.

El proyecto presentado en este trabajo proporcionando una solución para crear un menú muchas veces utilizando interacciones en distintos proyectos de A-Frame. La falta de componentes actualizados flexibles como interfaz para el desarrollo de otros proyectos , ha promovido la necesidad de desarrollar una librería que le permita crear un menú descentralizado y configurable, lo que pueda ayuda a usuarios experimentados y nuevos. Además, se busca que el sistema sea extensible, con capacidad para incorporar mejoras futuras y ser utilizado como base en otros proyectos VR más complejos.

Este contexto tecnológico y académico justifica la relevancia y la pertinencia del proyecto, el cual se sitúa en la intersección entre el desarrollo web, la ingeniería de software

y la experiencia de usuario en entornos inmersivos.

1.2. Objetivos

El presente proyecto tiene como finalidad el diseño y desarrollo de una librería de menús interactivos y personalizables en entornos de Realidad Extendida (XR), utilizando la tecnología A-Frame. Se pretende facilitar la creación de interfaces modulares y configurables, que puedan integrarse en cualquier proyecto de Realidad Virtual (VR) o XR basado en tecnologías web.

Para alcanzar este propósito general, se han definido una serie de objetivos generales y específicos que guían el desarrollo del proyecto.

1.2.1. Objetivos generales

- Desarrollar una librería modular y reutilizable de menús interactivos para entornos XR basados en A-Frame.
- Crear un sistema personalizable que permita al usuario modificar la estructura y apariencia de los menús mediante archivos de configuración externos (**JSON**).
- Garantizar la compatibilidad con dispositivos de XR, especialmente aquellos que permiten la interacción mediante hand-tracking, como las gafas Meta Quest 2 o Meta Quest 3.
- Proporcionar una solución de código abierto, fácilmente integrable por otros desarrolladores, que contribuya a enriquecer los recursos disponibles para el desarrollo de interfaces en realidad virtual.
- Implementar un sistema dinámico de generación de botones y submenús, configurables mediante archivos **JSON**.
- Diseñar un sistema de navegación fluida entre menús, mediante flechas de transición y botones enlazados.
- Desarrollar mecanismos de interacción natural basados en la detección de la posición de los dedos, sin necesidad de controladores físicos.

- Crear una estructura modular en el código que facilite su uso en otros proyectos de A-Frame.
- Elaborar documentación técnica detallada y ejemplos prácticos que permitan a otros desarrolladores integrar y personalizar fácilmente la librería en sus propios entornos XR.

1.2.2. Objetivos específicos

1.3. Estructura de la memoria

Esta memoria se ha organizado de forma que refleje tanto el proceso de desarrollo como los conocimientos adquiridos durante el proyecto. La estructura sigue un orden lógico que permite al lector comprender el contexto, la evolución del trabajo, las decisiones técnicas adoptadas y los resultados obtenidos.

El documento comienza con el capítulo de introducción, donde se presenta el contexto general del proyecto, los objetivos planteados, la organización del contenido y la disponibilidad del software desarrollado.

A continuación, el segundo capítulo está dedicado a las tecnologías utilizadas. En él se describen las herramientas de desarrollo web, los frameworks aplicados, los entornos de programación y ejecución, así como los dispositivos y recursos empleados para trabajar en realidad extendida.

El tercer capítulo constituye el núcleo del trabajo y expone el desarrollo del proyecto a lo largo de diferentes fases, organizadas en sprints. Cada sprint representa una etapa con objetivos concretos y avances progresivos, desde los primeros experimentos hasta la implementación final del sistema.

En el cuarto capítulo se detallan los resultados obtenidos mediante distintas demostraciones funcionales. Estas demos ilustran los escenarios desarrollados y permiten evaluar el rendimiento y la usabilidad del sistema en condiciones reales.

Finalmente, el capítulo de conclusiones recoge una valoración global del proyecto. Se analizan el grado de cumplimiento de los objetivos, los aprendizajes adquiridos durante el proceso, y se plantean posibles líneas de trabajo futuras que podrían ampliar o mejorar

la propuesta actual.

Capítulo 2

Tecnologías utilizadas

En este capítulo, se llevará a cabo un completo repaso de todas las tecnologías empleadas en el proyecto final. Se abordarán en detalle cada una de las herramientas y plataformas utilizadas, explicando su funcionalidad y la manera en que contribuyen al éxito del proyecto.

2.1. Desarrollo Web

El desarrollo web es la base tecnológica sobre la cual se desarrollan la interfaz del proyecto y la lógica. Esta área cubre la programación seleccionada y la manipulación de contenido dinámico, que es esencial para crear una experiencia interactiva. Esta sección describe las tecnologías más importantes utilizadas en esta área, con detalles de su papel y cómo están integradas para apoyar la aplicación desarrollada.

2.1.0.1. HTML

HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web, es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. En el contexto de este proyecto, HTML ha sido la base sobre la que se ha construido la escena virtual, ya que A-Frame está diseñado para integrarse directamente en este tipo de documentos.

Uno de los aspectos más destacados de trabajar con HTML en combinación con A-Frame es la posibilidad de definir entornos tridimensionales mediante una sintaxis sencilla.

y declarativa. Por ejemplo, los objetos virtuales, cámaras o luces pueden añadirse utilizando etiquetas específicas dentro del propio HTML, lo que reduce la complejidad inicial para quienes no están familiarizados con motores gráficos más avanzados. Un ejemplo de esta sintaxis sería para definir un documento en HTML se puede ver a continuación.

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

Además, el uso de HTML, ha facilitado la claridad y desarrollo del código en las primeras etapas debido a su sencillez, esto ha sido muy importante en las etapas iniciales. donde el foco estaba en experimentar y aprender los principios básicos para programar en el entorno virtual.

2.1.0.2. JavaScript

JavaScript es un lenguaje de programación interpretado que nos permite garantizar un dinamismo y interacción en las páginas Webs. En este proyecto juega un papel fundamental, ya que se encarga de administrar los comportamientos interactivos en el entorno de realidad aumentada, así como de controlar la lógica relacionada con el menú, los eventos de usuario y la manipulación de elementos virtuales. Una de las principales ventajas de JavaScript en este contexto es su estricta integración con el marco con A-Frame. Gracias a esto, es posible acceder y cambiar los objetos de la escena en tiempo real, responder a eventos creados por el usuario, como gestos o colisiones de la mano con objetos y practicar la lógica para ajustar la experiencia de acuerdo con las acciones del usuario. En el proceso de desarrollo, se han utilizado diferentes funciones y estructuras de JavaScript originales para crear elementos dinámicos en forma de un menú automático dependiendo de algunos parámetros. En resumen, JavaScript es un motor que da vida al entorno virtual, lo que permite la experiencia no solo intuitiva sino también interactiva y adaptativa. Su flexibilidad y compatibilidad con las tecnologías modernas de Internet lo convierten en una herramienta muy adecuada para desarrollar aplicaciones avanzadas.

2.1.0.3. DOM

El modelo de objetos del documento, más conocido como DOM, es la forma en que los navegadores interpretan y organizan el contenido de una página web. Básicamente, convierte el HTML en una estructura jerárquica que se puede recorrer y modificar mediante JavaScript. Esto es especialmente útil cuando desea actualizar algún contenido sin recargar toda la página, esto es muy popular en los sitios web modernos.

Durante la realización del proyecto , el DOM ha sido un herramienta indispensable a la que ha sido necesaria acceder para controlar diferentes elementos , como entidades en la escena para modificar y controlar una serie de eventos que realizaba el usuario con su interacción. Por ejemplo cuando se genera el menú , lo que se estaba realizando en realidad era una modificación del DOM en tiempo real.

Un aspecto especialmente relevante es que, al trabajar con A-Frame, todos los elementos de la escena —como cajas, cámaras o luces— se integran como parte del DOM. Esto facilita enormemente su manipulación mediante JavaScript, sin necesidad de uti-

lizar APIs gráficas complejas. Esta integración ha contribuido a agilizar el proceso de desarrollo, permitiendo iterar y experimentar de manera más eficiente.

En definitiva, el DOM ha sido una especie de columna vertebral para todo lo que implicaba interacción y dinamismo en el proyecto. Sin él, habría sido muy complicado implementar ciertas funcionalidades como la generación automática de menús o la respuesta a eventos generados por los gestos del usuario.

2.2. Frameworks y Bibliotecas

2.2.0.1. A-Frame

A-Frame es un frameworks sencillo y intuitivo de utilizar que sirve para crear entornos en realidad virtual directamente desde el navegador. En vez de tener que aprender un gran numero de herramientas para crear entornos virtuales , tenemos un herramienta comprimida , que permite montar escenas 3D usando etiquetado HMTL , que resulta común en los desarrolladores web como punto de inicio.

En este proyecto, A-Frame ha sido la base para construir los espacios virtuales. Por ejemplo, agregar un objeto tan simple como un cubo es cuestión de escribir unas pocas líneas y ajustar algunas propiedades como tamaño o color.A continuación hay un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Esfera en A-Frame</title>
<script src="https://aframe.io/releases/1.7.0/aframe.min.js"></script>
</head>
<body>
<a-scene>
<!-- Esfera -->
<a-box position="0 1.25 -3" depth="1.25" height="1.25" width="1.25" color="#EF2D5E"></a-box>
</a-scene>
</body>
</html>
```

Otra cosa , destacarle de A-Frame , es su capacidad de integración con WebXR , que mediante el uso de las gafas Meta Quest 2 , te permite probar la experiencia directamente sin tener que añadir nuevas configuraciones y con una gran rapidez y sencillez

La complejidad de Aframe radica en entender como funcionan los eventos y las interacciones con JavaScript

2.2.0.2. JSON

JSON (Notación de Objetos de JavaScript) es un formato ligero para el intercambio de datos. Su principal objetivo es ser fácil de leer y escribir para los humanos, así como fácil de analizar y generar por las máquinas. Aunque se deriva del lenguaje JavaScript, es independiente del lenguaje y se utiliza ampliamente en entornos que van desde aplicaciones web hasta servicios móviles y APIs. JSON utiliza una sintaxis basada en texto para representar estructuras de datos como objetos y arreglos (arrays), permitiendo transmitir datos estructurados entre un servidor y un cliente, o entre distintos sistemas. Un ejemplo de esta tecnología es :

```
{  
  "usuario": "maria123",  
  "nombre": "María Pérez",  
  "email": "maria@example.com",  
  "activo": true,  
  "roles": ["admin", "editor"]  
}
```

2.3. Entornos de Desarrollo

2.3.0.1. Visual Studio Code

Visual Studio Code, también denominado como VS Code, es un editor de código ampliamente adoptado en el ámbito del desarrollo de software. Fue creado por Microsoft con la intención de ofrecer una herramienta ligera, rápida y adaptable a diferentes lenguajes de programación. Está disponible para diversos sistemas operativos, como Windows, Linux y macOS, lo que favorece su uso en distintos entornos de trabajo desde la programación de software hasta el desarrollo de APIs , entornos webs entre otros.

Una característica que destaca en esta herramienta es su sistema de extensiones, que permite a los usuarios añadir funcionalidades específicas según el tipo de proyecto que estén desarrollando. Una de estas extensiones que es de gran importancia para el desarrollo web es la de live server que permite visualizar directamente en redes locales archivos HTML para su posterior depuración, ademas tambien destacan extensiones relacionadas con autocompletado de código o git para su conexión directa con Github.

Visual Studio Code tambien cuenta con una terminal integrada , soporte para múltiples pestañas y vista dividida , herramientas de ayuda para asistencia de programación.Por todo estas herramientas y Además de su apariencia sencilla, Visual Studio Code cuenta con una terminal integrada, soporte para múltiples pestañas y vista dividida, así como herramientas de asistencia al programador que facilitan la escritura de código limpio y eficiente. También permite trabajar con archivos ubicados en servidores remotos o dentro de contenedores, lo cual es especialmente útil en proyectos que requieren entornos virtualizados o distribuidos.

Por su capacidad de personalización, su constante evolución y la amplia comunidad de desarrolladores que lo respalda, VS Code se ha consolidado como uno de los entornos preferidos tanto por programadores principiantes como por profesionales con experiencia.

2.3.0.2. GitHub

Github es una plataforma gratuita utilizada ampliamente en el desarrollo de software para la gestión de proyectos , control de versiones , estructura de código , entre otros , basado en Git. Su empleo para este TFG ha permitido llevar un desarrollo sostenido de los distintos Spints que se han realizado , de manera fácil y rápida , teniendo un registro organizado y seguro de la evolución del código a lo largo de las distintas fases. Ademas se ha utilizado como una unidad de respaldo y forma de compartir el código hacia la comunidad que junto a sus numerosas herramientas ha permitido , un desarrollo mas sencillo y colaborativo con la comunidad.

2.4. Realidad Extendida

2.4.0.1. Realidad Extendida

La Realidad Extendida (XR, por sus siglas en inglés) es un nuevo campo tecnológico emergente que esta conformado por la realidad Virtual(VR) , la Realidad Aumentada(AR) y la realidad Mixta(MR). Es un nuevo concepto innovador que permite crear nuevos escenarios a través del ordenandor , ya sea creando todo el entorno como en la VR , superponiendo información con contenido generado como ocurre con la AR o creando un mix de contenido real y contenido generado permitiendo interacciones bidireccionales

entre objetos virtuales y objetos físicos como en el caso de la MR.

En el contexto del presente Trabajo Fin de Grado, la Realidad Extendida ha sido empleada como un medio para facilitar la visualización e interacción con una interfaz gráfica de tres dimensiones en el entorno virtual. Este desarrollo tecnológico responde a la necesidad de representar diferentes escenarios de una forma sencilla, comprensiva e intuitiva. La XR ofrece ventajas con respecto a otras tecnologías debido a que permite representar elementos de forma espacial, superando los límites de visualización de cualquier tecnología anterior, y favoreciendo una comprensión de los elementos más intuitivo y/o habitual para todos.

Además, la XR permite diseñar entornos adaptativos, donde el usuario no solo observa, sino que también actúa e influye directamente sobre los elementos del sistema, lo que resulta especialmente útil en tareas como la configuración de menús interactivos, selección de opciones o manipulación de objetos.

2.4.0.2. Gafas de Realidad Extendida Meta Quest

Las gafas de realidad virtual Meta Quest, desarrolladas por Meta, representan una de las soluciones más completas y accesibles dentro del ecosistema XR actual. Su principal ventaja radica en su carácter autónomo: no requieren estar conectadas a un ordenador ni depender de cables, lo que facilita la movilidad del usuario y mejora la experiencia inmersiva. Además, cuentan con soporte nativo para WebXR, lo que las hace compatibles con entornos virtuales creados directamente para su ejecución en navegadores web, como los desarrollados en A-Frame.

Durante el desarrollo del proyecto, las Meta Quest 2 han sido utilizadas como plataforma principal para la validación funcional del menú interactivo implementado. Gracias a su capacidad de ejecutar escenas WebXR sin necesidad de otras herramientas extras que ha permitido evaluar el desempeño de los distintos desarrollos, y a su vez comprobar la respuesta de la interfaz ante diversos eventos distintos, así como comprobar el comportamiento de los botones interactivos en un entorno simulado fiel a una aplicación final.

Además, el sistema de seguimiento de manos y controladores implementado con las gafas, han posibilitado que se pueda investigar en distintas modalidades de interacción,

lo que ha enriquecido el proceso de desarrollo. Teniendo la posibilidad de cambiar entre mandos físicos y gestos naturales con las manos lo que ha ofrecido una mayor flexibilidad en el diseño de la interfaz, permitiendo ajustar y optimizar la experiencia para seleccionar la mejor forma de implementar este desarrollo del TFG.

2.5. Desarrollo de Documentación

2.5.0.1. LaTeX

LaTeX es un sistema de composición de textos , utilizado en ambientes académicos y científicos , debido a la capacidad y facilidad de generar documentos con un formato estandarizado. Su uso es altamente recomendado para la generación de documentos técnicos y de investigación , donde son aspectos fundamentales la estructura del contenido o la inclusión de formulas matemáticas o código

LaTeX , ha sido utilizado para la redacción de este trabajo , no solo por un estándar dentro de la escuela de la ETSIT , sino por la calidad tipográfica que ofrece a la hora de permitir representar código , mantener una organización y una coherencia en el documento , ya que con el uso de los capítulos , secciones y subsecciones , se permite tanto una fácil lectura como una posible actualización futura de los contenidos.

2.6. Modelado y Formatos 3D

Los objetos tridimensionales constituyen una importante base de este proyecto , ya que son necesarios para la representación precisa de objetos en el espacio tridimensional.Para conseguir estos objetos modelados en 3D , se ha buscado esta información en repositorios especializados y se han adaptado los distintos objetos según las necesidades específicas del menu interactivo.

En este proceso, se ha trabajado con diversos formatos estándar de archivos tridimensionales, entre los que destacan ‘glTF’ y ‘obj’. Su compatibilidad con A-Frame ha sido esencial para facilitar la carga dinámica de modelos y su integración en la interfaz.

Este componente del trabajo no solo ha contribuido al aspecto visual del sistema desarrollado, sino que también ha permitido estudiar cómo integrar contenido externo en

una estructura modular y reutilizable dentro del ecosistema A-Frame, fortaleciendo así el carácter práctico y extensible del proyecto.

Capítulo 3

Desarrollo del proyecto

3.1. Sprint 1: Exploración Inicial de A-Frame y JavaScript

En este primer sprint , el enfoque esta orientado a un estudio de la tecnologia que se utilizara durante la mayor parte del proyecto , A-Frame , ademas se realizara un aprendizaje de otras tecnologias adyacentes como es Github y la gafas de realidad virtual. En este sprint los objetivos son los siguientes: -Aprendizaje de uso basico de A-Frame - Creacion y integracion de codigo en un repositorio Github -Aprendizaje de funcionamiento y utilizacion de la gas Oculus Quest 2

3.1.1. Introducción a A-Frame y su integración con HTML

Durante las primeras fases del desarrollo se llevo a cabo una toma de contacto con la tecnologia de Aframe el conocimiento previo en tecnologias web , particularmen HTML y Javascript jugo un papel importante en esta etapa que supuso una curva de aprendizaje menos elevada.

El primer contacto se inicio con pequeñas pruebas con la tecnología que consistió en generar distintas entidades básicas como por ejemplo un cubo utilizando la etiqueta <a-box>, todas ellas situándolas dentro de un contenedor <a-scene>. Esta prueba inicial sirvió no solo para validar la instalación del entorno, sino también para visualizar estas escenas el uso de las gafas Oculus Quest 2 los distintos escenarios que se iban generando.Se puede observar el ejemplo a continuación:

```
<html>
```

```
<head>
<script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
</head>
<body>
<a-scene>
  <a-box position="0 1 -3" color="#4CC3D9"></a-box>
  <a-box position="0 2 -3" color="black"></a-box>
</a-scene>
</body>
</html>
```

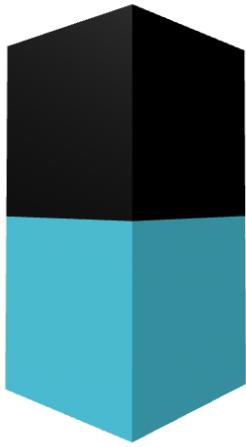


Figura 3.1: Ejemplo básico de A-Frame

A partir de esta prueba simple, se comenzó a investigar en distintas formas de avanzar y se decidió continuar con un sistema de colisiones entre las distintas entidades para crear interacciones básicas.

La detección de colisiones , fue un paso importante ya que se empezó a aprender el uso de librerías y también supuso un paso para poder integrar Javascript en A-Frame. La incorporación de este lenguaje permitió profundizar en sus conceptos fundamentales y avanzados, lo cual se tradujo en la implementación de eventos interactivos basados en la detección de colisiones entre distintas entidades del entorno virtual. Esta integración presentó desafíos iniciales, especialmente en lo referente a la manipulación de entidades

mediante scripts externos y a la articulación entre la estructura del DOM en HTML y el comportamiento interno definido en A-Frame.

Con el fin de ampliar los conocimientos desarrollados , se incorporaron otras tecnologías como son Hubiste que permitió tener un sistema de versiones y forma de compartir el código de una manera mas sencilla ,y también otras librerías para aprender otros funcionamientos de colisiones como es el repositorio de Ammo.js , que permitió realizar un sistema de de comportamientos de colisiones entre objetos mucho mas realistas añadiendo gravedad o respuesta ante colisiones. La inclusión de estas herramientas supuso un salto cualitativo en el desarrollo, pero también requirió un mayor grado de comprensión técnica y adaptación a nuevos modelos de programación.

Habiendo realizado un desarrollo y comprensión de A-Frame junto con el uso de Javascript , se comenzó con uno de los primero elementos a construir para una de las demos , creando un entorno básico con paredes y delimitaciones espaciales. La combinación de HTML junto a Javascript permitió desarrollar un sistema que generaba de forma automática una habitación virtual , que serviría como sala de pruebas , para avances mas complejos en los próximos sprints.

Fue en este punto, tras completar las pruebas de colisiones y comenzar a construir un entorno básico con paredes y delimitaciones espaciales, cuando se consolidó una comprensión más profunda de la estructura lógica de A-Frame. La combinación de etiquetas HTML con funciones JavaScript permitió desarrollar un sistema que generaba automáticamente una habitación virtual, lo que evidenció la flexibilidad del enfoque y sentó las bases para futuros avances más complejos, como la creación de menús interactivos y componentes personalizables.

3.1.2. Eventos e interacciones básicas

Tras los primeros pasos exploratorios con A-Frame y la construcción de un entorno básico, el siguiente objetivo fue dotar a las escenas de interactividad. En este contexto, se comenzó a trabajar con eventos, una funcionalidad esencial en entornos inmersivos para generar respuestas dinámicas ante acciones del usuario.

La incorporación de eventos en A-Frame se realizó principalmente mediante el uso de atributos HTML como onclick, mouseenter o mouseleave, y su gestión desde JavaScript.

Al principio, los eventos se limitaban a acciones simples como cambiar el color de un objeto al pasar el puntero por encima o eliminar un objeto al hacer clic sobre él. Estas pequeñas pruebas fueron fundamentales para entender cómo vincular el DOM de HTML con las capacidades reactivas de A-Frame.

Uno de los mayores retos fue comprender cómo gestionar correctamente el flujo de eventos dentro de una escena tridimensional, especialmente cuando varias entidades podían estar involucradas. También se aprendió a diferenciar entre los eventos generados por interacción directa (como un clic) y aquellos basados en la posición o movimiento (como las colisiones o la permanencia dentro de un área determinada).

Posteriormente, se implementaron funciones personalizadas para crear respuestas más complejas, como mover entidades dentro del espacio virtual, aplicar transformaciones a tiempo real, o activar submenús en función del comportamiento del usuario. Para lograrlo, se hizo un uso intensivo del sistema de componentes de A-Frame, permitiendo que los objetos fueran más dinámicos y reutilizables.

Asimismo, se comenzaron a explorar eventos más avanzados vinculados al sistema de físicas de la escena, como los que se generan mediante librerías externas (por ejemplo, `ammo.js`), lo que permitió ampliar el tipo de interacciones posibles, añadiendo efectos como rebote, colapso o desplazamiento por gravedad al interactuar con los elementos del entorno.

En resumen, esta fase del desarrollo no solo sirvió para enriquecer la experiencia del usuario, sino también para afianzar el conocimiento técnico sobre el funcionamiento interno de A-Frame, sentando las bases para la construcción de un sistema de menús y herramientas interactivos más robustos en fases posteriores.

3.1.3. Primer entorno: creación de una habitación virtual

El primer paso en el desarrollo de este proyecto fue la utilización de javascript en un código que sirviese para la futura demo de la habitación , en un entorno virtual que sirviese como base para posteriores sprints. Para ello, se implementó una habitación básica compuesta por suelo, techo y cuatro paredes, todo ello construido mediante entidades de *A-Frame* insertadas de forma automática , con una colocación y creación de estas de forma personalizable mediante código.

Este componente, denominado `room-resizer`, permite definir las dimensiones de la habitación mediante tres parámetros configurables: `altura`, `ancho` y `largo`. Su funcionamiento se basa en la creación de entidades directamente dentro de la escena, sin necesidad de que el desarrollador las ubique manualmente en el marcado HTML.

En el siguiente fragmento de código se muestra el uso de dicho componente:

```
<a-entity room-resizer="altura: 5; ancho: 12; largo: 12"></a-entity>
```

El componente `room-resizer` se encarga de insertar las paredes y el suelo de la habitación mediante el uso de entidades con geometría de caja. Una vez generadas se encarga de ubicar las distintas entidades mediante una serie formulas matemáticas para que encajen tanto el suelo , paredes como techo , creando así un habitación básica. Por ultimo con fines de depuración y visualización , se asignaron colores diferentes a cada uno de los elementos para poder diferenciarlos para un futuro de manera sencilla.

El código del componente está estructurado en tres partes fundamentales:

-Creación de entidades: Se generan dinámicamente las paredes, el suelo y el techo, y se añaden a la escena.

-Ajuste de dimensiones: A partir de los parámetros configurados, se calcula la posición y tamaño de cada elemento, garantizando que las paredes se alineen correctamente y la habitación tenga las proporciones deseadas.

-Flexibilidad: El sistema permite modificar las dimensiones de la habitación simplemente alterando los valores del atributo `room-resizer`, sin necesidad de reescribir el código además estos objetos permiten introducir texturas de manera sencilla.

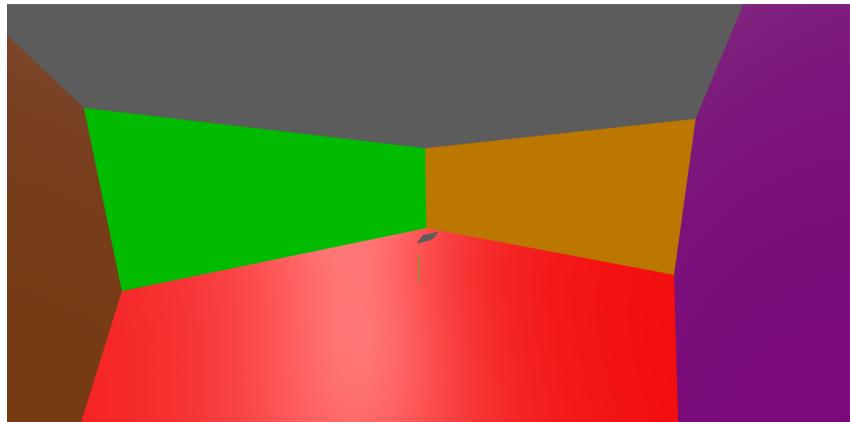


Figura 3.2: Generacion de habitacion sin texturas



Figura 3.3: Ejemplos de texturas

Este primer entorno tiene como objetivo servir de contenedor para las futuras pruebas del menú personalizable e interactivo que constituye el núcleo del trabajo. Asimismo, su construcción dinámica representa un primer acercamiento a la manipulación avanzada de entidades en *A-Frame* mediante JavaScript, sentando las bases para los desarrollos posteriores.

3.2. Sprint 2: Desarrollo del Menú Estático

3.2.1. Primer diseño de interfaz fija

Durante el segundo sprint del proyecto se abordó el diseño y construcción del primer prototipo de menú dentro del entorno virtual. Este menú tiene como finalidad servir de punto de partida para las futuras funcionalidades interactivas, permitiendo al usuario realizar acciones dentro de la escena mediante botones accesibles.

Para ello, se implementaron varios componentes personalizados en *A-Frame* que permitieron estructurar tanto la parte visual como la lógica del menú. En primer lugar, se desarrolló el componente `menu`, encargado de generar la base o fondo sobre el que se sitúan los botones. Esta objeto consiste en un entidad con una geometría de un plano con forma rectangular , lo que nos permite tener una interfaz sencilla en donde poder tomar de base para empezar a generar el menu.

Una vez creado el menu, se creó el componente `button`, que permite generar entidades que actuaran como botones en nuestro menu , estos botones , tendrán su correspondiente etiqueta de texto. Los botones son instancias de entidades *A-Frame* con geometría en forma de caja, a las que se les asocia un texto centrado indicando su función. Además, desde esta fase temprana se integró el sistema de detección de interacción mediante gestos de pellizco, gracias al componente `pinchable`, el cual detecta si el usuario realiza un gesto de pinza sobre el botón y emite un evento personalizado llamado `pinchstarted` a la escena de Aframe que puede ser detectado , con el que mediante su detección en otro parte del código , se puede capturar el gesto y/o colisión con el que se puede interactuar sobre el menu.

Adicionalmente, durante este sprint se avanzó en la carga dinámica de objetos 3D externos al entorno, utilizando ficheros en formato `.obj` y `.mtl`. Esta funcionalidad se incorporó dentro del `event-manager`, un componente que gestiona los eventos generados por los botones. Así, al pulsar determinados botones se puede insertar dentro de la escena modelos como camas u otras figuras complejas, demostrando la capacidad de cargar recursos externos y ampliar el contenido del entorno.

En resumen , este sprint permitió , iniciar con las bases para generar una interfaz para el usuario , teniendo un sistema modular para crear distintos menús personalizados y interactivos.

3.2.2. Incorporación de elementos interactivos

Una vez diseñado la base del menú estático con el fondo y sus botones, el siguiente paso consistió en dotarlo de un administrador que se encargase de ejecutar las distintas acciones y dotase de interactividad a los botones, permitiendo al usuario ejecutar acciones dentro del entorno virtual de forma intuitiva. Para ello, se desarrolló el componente `event-`

manager..

Mediante este componente, se implementó un sistema que permite al usuario, a través de la interacción con los botones detectados previamente, activar diferentes funciones asociadas a cada botón. Estos eventos no solo provocan cambios visuales, sino que permiten insertar de manera dinámica objetos en la escena, demostrando la interacción directa entre el menú y el entorno.

Durante esta fase se incorporaron las siguientes funcionalidades:

- **Generación de objetos primitivos:** El menú permite al usuario insertar cubos, esferas, cilindros o conos en la escena, situándolos en posiciones concretas con dimensiones predefinidas.
- **Carga de modelos externos:** Se añadió la posibilidad de insertar modelos 3D externos, como una cama, utilizando archivos en formato .obj y .mtl. Esta funcionalidad demuestra la flexibilidad del sistema para integrar recursos externos y enriquecer el entorno virtual.(Véase próxima imagen)
- **Gestión de estados:** Se implementó un sistema de cambio de estado mediante flechas direccionales, que modifica el comportamiento de los botones en función del modo seleccionado. Con esto se quería llegar a tener un menú interactivo y rotativo en donde se pudiesen tener varias capas de profundidad , por ejemplo, en un primer estado los botones insertan objetos, mientras que en un segundo estado permitirían modificar dimensiones o insertar distintos objetos que en ese primer estado



Figura 3.4: Ejemplo de objeto 3D

La incorporación de estos elementos interactivos supuso un avance significativo en la evolución del proyecto, proporcionando al usuario un primer nivel de control dentro de la escena y sentando las bases para un sistema de interacción más complejo y dinámico en fases posteriores.

3.2.3. Utilización de las manos para la interacción

Una de las principales innovaciones implementadas en este sprint es la posibilidad de interactuar directamente con los elementos del menú utilizando únicamente las manos, sin necesidad de dispositivos externos como mandos o controladores físicos. Esta funcionalidad se logró gracias al desarrollo e integración del componente personalizado `pinchable`, diseñado específicamente para detectar gestos de pinza realizados por el usuario.

El componente `pinchable` calcula en tiempo real la distancia entre la posición de la mano del usuario y los distintos elementos interactivos del menú. Cuando el sistema detecta que esta distancia es inferior a un umbral predefinido, se interpreta como que el usuario ha realizado un gesto de pinza sobre dicho elemento, emitiendo un evento específico denominado `pinchedstarted`.

Este evento es capturado por otros componentes del sistema, como el `event-manager`, que se encarga de ejecutar la acción correspondiente asociada al botón o entidad interactiva. De esta forma, el menú deja de ser solo un conjunto de varias entidades y pasar a ser una interfaz en realidad virtual con la que el usuario puede interactuar, mediante un gesto sencillo como es el de "pincharçon los dedos".

La incorporación de la interacción manual aporta un valor diferencial al proyecto, permitiendo una experiencia de usuario mucho más inmersiva y realista. El control por gestos, sin necesidad de mandos, acerca el funcionamiento del sistema a los estándares actuales de interfaces avanzadas en entornos de realidad virtual y realidad extendida, alineándose con las tendencias que esta habiendo en donde el control sea lo mas natural y cercano a la realidad consiguiendo así una accesibilidad para todo el mundo.

Ademas , el enfoque de únicamente usar las manos sin necesidad de mandos u otros elementos de hardware , mejora la libertad de movimiento y las posibilidades de integrar el sistema en dispositivos VR que soporten la detección de manos. Con todo esto , llegamos a la conclusión de que la implementation del control de gestos representa un hito clave dentro del desarrollo del proyecto ya que permite una mejora en su accesibilidad y punto de innovación con respecto al desarrollo de otros menus interactivos. A continuación se puede observar un ejemplo del uso de manos en realidad virtual.

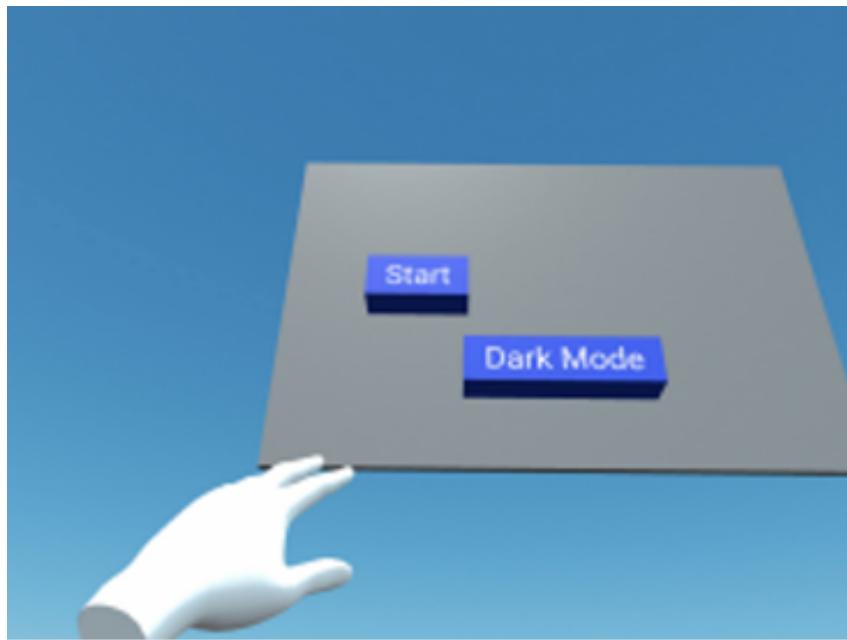


Figura 3.5: Ejemplo de uso de manos en realidad virtual

3.3. Sprint 3: Experimentación con Menús Dinámicos

3.3.1. Generación dinámica de menús

Durante este tercer sprint se abordó la idea de realizar el sistema del menú , completamente dinámica a través de un archivo con datos personalizables que permita crear una estructura adaptable.Hasta este punto los sistemas de menus que se crearon se realizaban de forma estática configurando cada menu de forma manual , con cada botón y elemento definido explícitamente en el código, que limitaba la escalabilidad y la personalización

Para solventar estas limitaciones, se implementó un sistema de generación dinámica de menús basado en la lectura de ficheros .JSON

3.3.2. Lectura de datos externos y generación del menú

Una de las principales novedades incorporadas en este sprint fue la transición de un sistema de menús estáticos a un modelo completamente dinámico, capaz de adaptarse en función de datos externos ,permitiendo definir la configuración de los botones de forma flexible y sin necesidad de modificar el código fuente.

La información de los botones se almacena en un archivo denominado `menu_data.json`, donde se especifican propiedades clave como el identificador, la etiqueta de texto, las dimensiones y el color de cada botón. El siguiente fragmento muestra un ejemplo de dicho archivo:

```
[
  {
    "id": "boxButton1" ,
    "label": "Boton 1",
    "width":5
  },
  {
    "id": "boxButton2" ,
    "label": "Boton 2" ,
    "height": 7
  },
  {
    "id": "boxButton3" ,
    "label": "Boton 3",
    "depth": 15 ,
    "color": "black"},
  {
    "id": "boxButton4" ,
    "label": "Boton 4" ,
    "evento": "spawn_bed"
  }
]
```

La lectura de este archivo se realiza dentro del componente de Javascript `menu.js`, mediante el uso de la función `fetch()`. Una vez obtenidos los datos, se procede a la creación dinámica de cada botón, generando entidades en tiempo de ejecución y asignándoles los atributos que hayan sido declarados en el archivo JSON.

Además, se implementó un algoritmo que distribuye los botones de forma automática en una cuadrícula dentro del menú, calculando la posición de cada elemento para mantener una disposición simétrica y organizada, independientemente del número total de botones. Este cálculo se basa en determinar el número óptimo de filas y columnas, distribuyendo los botones alrededor del origen para garantizar un aspecto centrado y equilibrado.

Este enfoque aporta una notable flexibilidad al sistema, ya que permite modificar el contenido y la apariencia del menú simplemente editando el archivo JSON, sin necesidad de intervenir en el código JavaScript. De esta forma, se facilita la personalización, es-

calabilidad y mantenimiento del sistema, abriendo la posibilidad de generar diferentes configuraciones de menús adaptadas a las necesidades de cada escenario.

Con todo esto explicado se llega a un resultado de un menu que se genera de forma dinámica que es el siguiente:

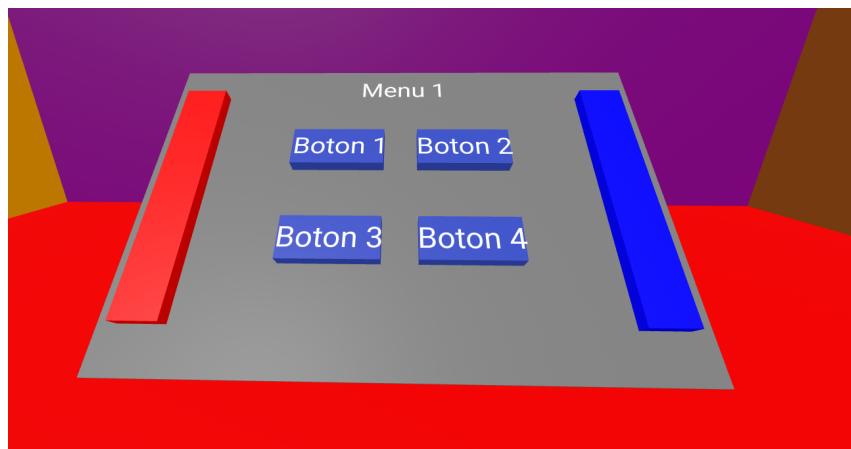


Figura 3.6: Diseño de Menu versión inicial

El uso de datos externos en el proceso de generación de una interfaz para el usuario supone un gran avance en la evolución del proyecto , adaptándose a los principios de modularidad y reutilización de código ,orientado hacia los entornos de realidad virtual.

3.3.3. Gestión de eventos de botones y navegación

Una vez implementado el sistema de generación dinámica de menús mencionado anteriormente, el siguiente paso fue dotar a estos elementos de interactividad, permitiendo al usuario ejecutar acciones y navegar entre distintos niveles del menú. Para ello, se diseñaron e integraron dos componentes clave: `event-manager` y `arrow_button_manager`.

El componente `buttons_event_manager.js` es el encargado de gestionar los eventos asociados a los botones generados dinámicamente. A través de la detección de gestos de pinza, implementada previamente mediante el componente `pinchable`, se identifican las interacciones del usuario sobre cada botón y se desencadenan las eventos o acciones designadas para cada botón.

Este sistema funciona mediante un sistema de estados en el que existe una serie de botones que van cambiando de estados para permitir insertar distintos objetos en la escena

como cubos, esferas, cilindros o modelos 3D externos, en función del botón pulsado y del estado actual en el que se encuentra el menú. Los estados se gestionan de forma dinámica, lo que otorga a los botones un comportamiento contextual y polivalente.

Por otro lado, el componente `arrow_button_manager` controla la navegación entre los diferentes estados del menú. Mediante dos flechas de dirección situadas en los laterales del menú, el usuario puede cambiar el modo de funcionamiento de los botones. Al pulsar una flecha, se actualizan los estados internos de todos los botones y se modifica el texto de cabecera para reflejar el nuevo modo activo.

Este sistema de navegación por estados permite mejorar con respecto al anterior sprint , las posibilidades del menu , permitiendo que un único botón pueda tener infinitos estado , según el contexto que se programe. Además, se implementaron mecanismos de control y propagación de estados, garantizando que todos los elementos del menú se mantengan sincronizados y actualizados en todo momento.

En conjunto, la gestión de eventos y navegación dota al sistema de un alto grado de interactividad y flexibilidad, ofreciendo al usuario una experiencia más rica y funcional dentro del entorno virtual, pero estando limitado a tener que generar cada uno de los estados y las acciones de los botones a través de código y no de una manera sencilla y interactiva , por lo que se conseguir cierta modularidad en el menu pero no al nivel máximo que se busca con este proyecto.

3.3.4. Optimización y detección de errores en el sistema dinámico

El proceso de desarrollo del menú dinámico y su integración en el entorno virtual no estuvo exento de dificultades y situaciones imprevistas que revelaron la complejidad inherente a los sistemas generados de forma automática en tiempo de ejecución.

uno de los principales problemas fue con la lectura del archivo `menu_data.json`.En los casos en donde el archivo no estaba disponible , contenía errores de formato o la ruta de acceso no estaba disponible.Todo esto producía que la pagina no cargase o que el menu se quedase incompleto , haciéndolo inoperativo.Para evitar estos errores se incluyeron una serie de mecanismo para mitigar los errores , como la validación de los datos o la aparición

de mensajes de alerta que permiten al usuario conocer los errores.

Otro aspecto critico fue el control de los estados de los botones , en el que no se sincronizaban correctamente o no respondían a los eventos de interacción , demostraba la necesidad de una correcta inicialización de menu correcta

También se vi caso en donde la existencia de múltiples botones generaba solapes entre ellos haciendo que no mantuviesen una estructura correcta o se pulsase botones que nos deseasen por estar muy próximos entre ellos lo que obligo a realizar una serie de modificaciones en los parámetros del componente de `pinchable` para evitar interferencias

Por ultimo , se identificaron problemas con la sincronización de los cambios de estados , ya que ocurría que los botones no actualizaban sus estados de forma coherente, para solucionar esto se reforzó el control interno en los componentes de `arrow_button_manager` y `event-manager` , asegurándose la propagación de los estados de forma uniforme

Este proceso de optimizan y detección de errores ha sido clave para consolidar un sistema de menús dinámicos robusto, fiable y preparado para ser ampliado en futuras fases del proyecto, minimizando los fallos y garantizando una experiencia de usuario coherente y funcional.

3.4. Sprint 4: Implementación Final y Optimización

3.4.1. Objetivos y mejoras planteadas en este sprint

El cuarto y último sprint del proyecto se centró en la consolidación y optimización del sistema de menús desarrollado en fases anteriores, con el objetivo de obtener una solución final más robusta, escalable y visualmente atractiva. A diferencia de las fases previas, este sprint no solo planteó mejoras funcionales, sino también un rediseño profundo tanto a nivel técnico como estético.

Entre los principales objetivos de esta fase se encuentran:

- **Mejorar la precisión y la fiabilidad del sistema de interacción manual:** Se identificaron limitaciones en el sistema basado en el componente `pinchable`, por lo que se procedió a su sustitución por el nuevo componente `pressable`, capaz de detectar la proximidad del dedo índice ya que es mas natural pulsar un botón utilizando el dedo indice.

- **Optimizar la estructura y el funcionamiento interno del menú:** Se rediseñó el archivo de configuración `menu_data.json`, permitiendo una mayor flexibilidad y soporte para menús enlazados, submenús y navegación entre diferentes niveles de interfaz.
- **Mejorar la estética y usabilidad del menú:** Se incorporaron nuevas características visuales para pasar de un menú tosco con figuras simples a un menú más estético y configurable con el uso de el componente `rounded-plane` que permite que el menú tenga bordes redondeados y la posibilidad de poner imágenes personalizadas en los botones , logrando un menú más atractivo , con una interfaz más moderna y profesional.
- **Garantizar la escalabilidad del sistema:** Se desarrolló un nuevo enfoque de generación dinámica de botones y submenús, permitiendo que la cantidad de opciones de menús y como se conectan entre ellos se realice de manera automática a través del archivo de configuración sin necesidad de realizar modificaciones estructurales en el código base.

Con estas mejoras, se persiguió como meta final disponer de un sistema de menús completamente funcional, atractivo y adaptado a las necesidades de interacción en entornos de realidad extendida, capaz de integrarse de forma eficiente en aplicaciones VR/AR que requieran interfaces accesibles y personalizables.

3.4.2. Rediseño estructural del menú y modificación del formato JSON

Uno de los cambios más significativos introducidos en este sprint fue la completa reorganización interna del sistema de menús y la modificación de la estructura del archivo `menu_data.json`. Hasta este punto, la generación de los menús se basaba en esquemas más simples, con un único nivel de botones y configuraciones limitadas, lo que restringía la escalabilidad y la capacidad de navegación.

Para solventar estas limitaciones, se rediseñó el sistema bajo un modelo de menús enlazados y submenús dinámicos, soportando múltiples niveles de interfaz y relaciones jerárquicas entre menús. Esta evolución implicó un cambio en el formato del archivo `menu_data.json`, y también esto supuso un cambio en la lógica de como generar el menú ya que pasamos de un botón que múltiples estado a tener todos los botones generados y

ir llevando y trayendo los botones al fondo del menu en función del menu que se solicitase con esto se pasó a tener una estructura más compleja, pero mucho más flexible y potente.

En el nuevo diseño, cada menú se define como un objeto independiente dentro del archivo JSON, con atributos que determinan su identificador, el nombre del menú en la etiqueta, el menú al que pertenece, los posibles menús anterior y siguiente para un sistema rotativos entre los distintos menús, y el listado de botones asociados a ese menú. Cada botón, a su vez, tiene atributos como su identificador, nombre que tendrá su etiqueta, una imagen asociada que funcionara como su caratula, y luego una acción o un submenu en caso de ser un botón para llegar a otro menú que se activara con la pulsación del botón.

El siguiente fragmento muestra un ejemplo simplificado de esta nueva estructura:

```
{
  "menuId": "menuPrincipal",
  "menuLabel": "Menú inicial",
  "activo": true,
  "submenu": null,
  "nextMenu": null,
  "previousMenu": null,
  "buttons": [
    {
      "id": "boton1",
      "label": "Botón 1",
      "action": "generico",
      "img": "./assets/cama_generica.png",
      "open_submenu": null
    }
  ]
}
```

Este enfoque aporta varias ventajas clave:

- **Navegación flexible:** El sistema permite enlazar menús principales, submenús y menús adicionales, con esto se consigue gestionar el movimientos entre menus mediante las flechas de transiciones y mediante los botones que abren otros submenus.
- **Escalabilidad:** Se pueden añadir nuevos botones, submenús o niveles adicionales sin necesidad de modificar el código fuente, únicamente editando el archivo JSON.
- **Separación de datos y lógica:** La configuración visual y estructural del menú queda completamente separada de la lógica del sistema, lo que mejora el mantenimiento, la personalización y la reutilización en diferentes proyectos o entornos.

Este rediseño de la lógica del código y la forma de plantearse , representa una mejora notable con respecto al anterior sprint ya que se consigue que mediante la modificación del fichero JSON , tengamos todo lo necesario para crear una gran cantidad de menus de distintas formas y formatos

3.4.3. Nuevo sistema de detección de interacción

Uno de los objetivos fundamentales de este proyecto ha sido la utilización de la detección de las manos en realidad virtual con el objetivo de permitir una interacción más natural del usuario y no depender de hardware como mandos u otros dispositivos. En este contexto esta interacción de las manos cobra un papel esencial en el proyecto

En sprints anteriores, la detección de interacción manual se basaba en el componente **pinchable**, que identificaba gestos de pinza realizados por el usuario. Sin embargo, este sistema presentaba un problema ya que no era una forma de interacción natural con los botones y otros objetos aparte generaba a veces activaciones de otras funciones como agarrar objetos de forma accidental , por ello se decidió cambiar el gesto y hacer la interacción únicamente con la punta del dedo.

Para arreglar estas limitaciones y mejorar la fiabilidad del sistema, en este sprint se implementó un nuevo componente denominado **pressable**. A diferencia del sistema anterior, **pressable** basa su funcionamiento en el seguimiento de la distancia entre los elementos interactivos y la posición de la punta del dedo índice, utilizando la información proporcionada por el sistema de *hand tracking*.

Cuando la punta del dedo se aproxima a un botón o a un elemento interactivo a una distancia inferior a un umbral configurable, se emite un evento **pressedstarted**, interpretado como una pulsación , esta punta del dedo esta ya mapeada en la tecnología de A-Frame en donde la mano esta formada por una serie de vectores y uniones denominadas **Joints** y la punta del dedo es la unión de vectores que es la numero 9.Esto se puede observar en la imagen a continuación. Este mecanismo resulta más intuitivo y natural, ya que simula el gesto de presionar físicamente un botón, lo que resulta familiar para el usuario y elimina la necesidad de realizar gestos complejos. La introducción de **pressable** supone una mejora significativa tanto en precisión como en accesibilidad. Al tratarse de un sistema basado en proximidad, se minimizan los falsos positivos y se incrementa la robustez

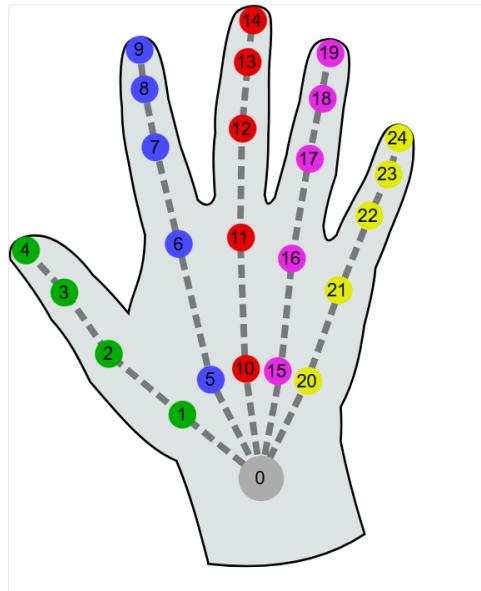


Figura 3.7: Sistema de vectores de la mano en A-frame

de la detección, especialmente en entornos con alta densidad de elementos interactivos.

Ademas esta solución se encuentra mucho mas adaptado como herramienta en otros entornos de realidad virtual por lo que todo el código se alinearía con los estándares actuales de interfaces de realidad virtual que existen mejorando la accessibilidad y entendimiento de los usuarios.

En conjunto, la transición de `pinchable` a `pressable` representa un paso clave en la evolución del sistema, consolidando una interacción precisa, fluida y adaptada al uso exclusivo de las manos, eliminando barreras y proporcionando una experiencia inmersiva de alta calidad.

3.4.4. Creación modular y escalable de botones y submenús

Con el objetivo de garantizar un sistema flexible, personalizable y preparado para futuras ampliaciones, durante este sprint se diseñó e implementó un nuevo enfoque modular y escalable para la generación de botones y submenús dentro del entorno virtual.

Hasta fases anteriores, la estructura del menú limitaba las posibilidades de expansión, ya que la creación de botones estaba vinculada a configuraciones rígidas o se realizaba de forma manual, lo que dificultaba la inclusión de nuevos niveles, submenús o relaciones entre elementos teniendo la necesidad de modificar el código base.

Para solucionar estas limitaciones, el sistema actual permite generar de forma dinámica todos los botones y submenús a partir de los datos definidos en el archivo `menu_data.json`, sin necesidad de modificar el código fuente. El componente `menu` es el encargado de procesar este archivo y, mediante el método `AppendButtons`, se crean en tiempo real los botones con sus propiedades, acciones y atributos de navegación.

Del mismo modo, los submenús se gestionan de forma modular. Cuando el usuario selecciona un botón que abre un submenú, el sistema oculta los botones y etiquetas del menú actual y activa los elementos correspondientes al submenú solicitado. Este proceso se realiza mediante transiciones espaciales, desplazando los elementos fuera del área visible y reubicándolos cuando se solicitan.

Además, se ha incorporado la navegación entre menús mediante flechas direccionales, gestionadas por el componente `arrow`, lo que permite moverse entre diferentes niveles de menú de forma sencilla, sin necesidad de salir del entorno o recargar la escena.

Esta estructura modular aporta múltiples ventajas:

- **Escalabilidad:** Se pueden añadir nuevos menús, botones o subniveles de forma ilimitada, simplemente editando el archivo de configuración.
- **Mantenimiento sencillo:** Al separar los datos de la lógica, se facilita la actualización o modificación de los menús sin necesidad de intervenir en el código.
- **Reutilización y portabilidad:** El sistema es fácilmente adaptable a otros proyectos o entornos que requieran menús similares, gracias a su diseño independiente y parametrizable.

En conjunto, esta solución garantiza un sistema de menús robusto, preparado para entornos complejos y capaz de adaptarse a nuevas necesidades sin comprometer la estabilidad ni la usabilidad del entorno virtual.

3.4.5. Mejoras visuales: rediseño del menú e incorporación de imágenes

Además de las mejoras funcionales y estructurales, este sprint final introdujo un conjunto de cambios orientados a tener un menu mas profesional y con una interfaz con un aspecto visual y estetico , mas moderno y intuitivo.

La primera mejora destacable fue la incorporación del componente `rounded-plane`, desarrollado específicamente para generar planos con esquinas redondeadas , una función la cual no existe en aframe ya que todos sus entidades son entidades geométricas con ángulos rectos. Gracias a esta modificación, el fondo del menú adopta una forma más orgánica y estilizada, alejándose de los diseños rectangulares rígidos de fases anteriores. Esta mejora supone que el acabado visual sea mucho mejor sino que el menú permita dar una sensación de inmersión y realismo en el entorno visual mucho mas superior.

Por otro lado , se mejora la apariencia de los botones , pasando de objetos con geometrías de cajas a objetos con forma de circulo , en donde se podían personalizar poniéndoles una imagen , pasando de entidades normales y corrientes a botones , que presentan imágenes y puede generar una mejor conexión en el usuario de la acción del botón con lo que ven haciendo toda la experiencia mas intuitiva para cada uno de los distintos menus que pueda generar los distintos usuarios.

Para ello, se amplió el archivo `menu_data.json`, permitiendo asociar a cada botón una ruta de imagen. Durante la creación dinámica de los botones, el sistema aplica automáticamente la textura correspondiente, logrando una interfaz mucho más visual y accesible. Un ejemplo de esta configuración es el siguiente:

```
{
  "id": "boton1",
  "label": "Botón 1",
  "accion": "generico",
  "img": "./assets/cama_generica.png",
  "abreSubmenu": null
}
```

También, se adoptaron paletas de colores modernas y equilibradas, diferenciando claramente los distintos elementos del menú. Por ejemplo, se utilizaron tonos azulados para los fondos, imágenes de flechas en las flechas de transición entre los distintos menús y textos de alto contraste para poder leer las distintas etiquetas de manera cómoda.

Toda estas series de cambios visuales se pueden observar en la siguiente imagen:

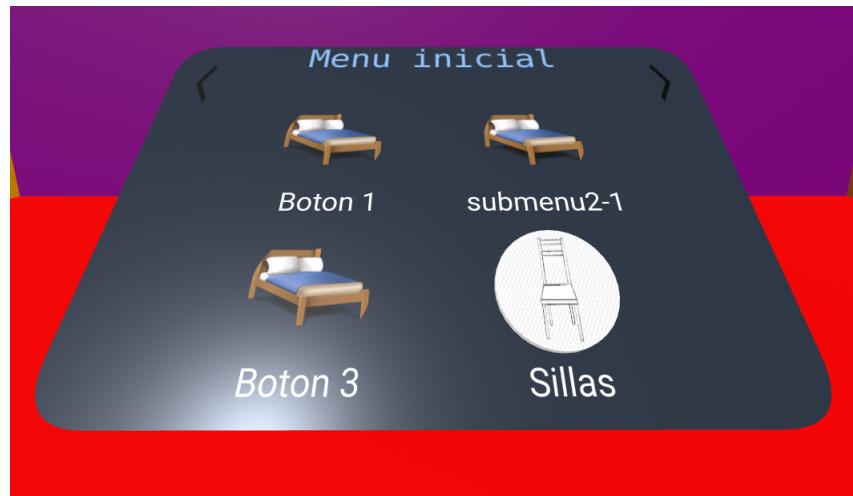


Figura 3.8: Diseño de menú versión final

Estas mejoras visuales no solo optimizan la apariencia del menú, sino que contribuyen directamente a la usabilidad, al reducir la carga cognitiva del usuario mejorando así su interactividad.

Capítulo 4

Descripción del resultado

4.1. Interfaz de Usuario

Como parte de este Trabajo de Fin de Grado y del proyecto **BabiaXR**, se ha desarrollado una librería de menús interactivos y personalizables para entornos de Realidad Extendida (XR) utilizando la tecnología de A-Frame en la que se ha creado un archivo `user_guide_spanish.md` para ayudar al usuario con el uso de esta librería.

4.1.1. Características principales

La interfaz de usuario implementada ofrece las siguientes funcionalidades:

- Menús completamente configurables mediante archivos JSON externos.
- Generación dinámica de botones y submenús enlazados.
- Interacción natural a través de la punta del dedo, compatible con dispositivos como Meta Quest 2/3.
- Soporte para imágenes personalizadas dentro de los botones.
- Estética mejorada con botones redondeados y fondos personalizables.
- Navegación fluida entre diferentes menús mediante flechas de dirección.
- Sistema modular y escalable, fácilmente integrable en otros proyectos basados en A-Frame.

4.1.2. Requisitos técnicos

Para el correcto funcionamiento de la interfaz, se requiere:

- Navegador compatible con WebXR (se recomienda Google Chrome o Microsoft Edge).
- Dispositivo que disponga de soporte para *hand-tracking*.
- Conocimientos básicos de A-Frame y estructura de escenas en HTML.
- Editor de texto para la edición de archivos de configuración en formato JSON.

4.1.3. Instalación y estructura de archivos

El sistema puede integrarse de dos formas:

1. Clonando directamente el repositorio:

```
git clone https://github.com/ALSASA12/Menu-interactivo-aframe.git
```

2. Descargando manualmente los siguientes archivos esenciales:

- arrow_menu.js
- pressable.js
- rounded-plane.js
- menu.js
- sub-menu.js
- Carpeta assets/ con las imágenes necesarias para botones y navegación

Ademas de estos archivos sera necesario crear un archivo .json con una configuracion especifica explicada en el siguiente apartado.

4.1.4. Configuración de los menús

Los menús se definen mediante un archivo .json que sigue la siguiente estructura:

```
[  
 {  
   "menuId": "menuPrincipal",  
   "menuLabel": "Menú inicial",  
   "activo": true,  
   "submenu": null,  
   "nextMenu": null,  
   "previousMenu": null,  
   "botones": [  
     {  
       "id": "boton1",  
       "label": "Muebles",  
       "action": null,  
       "img": "./assets/sofa_generico.png",  
       "open_submenu": "submenu2-1"  
     }  
   ]  
 }]
```

Campo	Descripción
menuId	Identificador único del menú, no puede repetirse.
menuLabel	Texto que aparece como título del menú.
activo	Si es <code>true</code> , este será el menú visible al iniciar la escena. Solo uno debe tener esta propiedad activada.
submenu	Si es un submenú, indica el <code>menuId</code> de su menú padre.
nextmenu	<code>menuId</code> del siguiente menú en la secuencia. Puede ser <code>null</code> .
previousmenu	<code>menuId</code> del menú anterior. Puede ser <code>null</code> .
botones	Array con los botones asociados a este menú.

Cada botón se configura de la siguiente manera:

Campo	Descripción
id	Identificador único del botón.
label	Texto visible del botón.
action	Acción a ejecutar al pulsar el botón (opcional).
img	Ruta de la imagen que se mostrará en el botón.
opensubmenu	Si abre un submenú, se indica su <code>menuId</code> .

4.1.5. Integración en la escena de A-Frame

El siguiente ejemplo muestra cómo insertar el menú en una escena:

```
<a-scene>
<a-entity id="menuContainer"
menu="json: ./menu_data.json";
menu_width: 0.6;
menu_height: 0.4;
button_height: 0.025;
button_color: #3B82F6;
button_label_color: #FFFFFF;
menu_color: #374151;
menu_label_color: #93C5FD;
arrow_color: #F59E0B">
</a-entity>

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

4.1.6. Parámetros de configuración disponibles

Parámetro	Descripción	Valor por defecto
json	Ruta al archivo .json de configuración	./menu_data.json
menu_width	Anchura del menú (metros)	0.6
menu_height	Altura total del menú (metros)	0.4
button_height	Altura de los botones (metros)	0.025
button_color	Color de fondo de los botones	#3B82F6
button_label_color	Color del texto de los botones	#FFFFFF
menu_color	Color de fondo del menú	#374151
menu_label_color	Color del título del menú	#93C5FD
arrow_color	Color de las flechas de navegación	#F59E0B

4.1.7. Sistema de interacción

El sistema utiliza el componente `pressable`, que detecta la proximidad de la punta del dedo índice y emite eventos como `pressedstarted` y `pressedended`, que se pueden gestionar mediante scripts personalizados.

Ejemplo de detección de pulsación:

```
AFRAME.registerComponent('mi-componente-interactivo', {
  init: function () {
    this.onPressStart = this.onPressStart.bind(this);
    this.el.addEventListener('pressedended', this.onPressStart);
  },
  onPressStart: function (evt) {
    if (evt.target === this.el) {
      console.log("Botón pulsado, ejecutar acción personalizada.");
    }
  }
});
```

4.1.8. Opciones de personalización

- Estilos visuales modificables mediante el componente `rounded-plane`.
- Iconos, imágenes, menús, número de botones y sus acciones configurables desde el

archivo `menu_data.json`.

- Navegación entre menús mediante los campos `menuSiguiente` y `menuAnterior`.
- Ajustes de color, tamaños y estética general a través de atributos en el HTML.

4.1.9. Recomendaciones de uso

Para un uso óptimo se sugiere:

- Optimizar las imágenes utilizadas para mejorar el rendimiento.
- Verificar el correcto funcionamiento del *hand-tracking* antes de su despliegue.
- Evitar sobrecargar la escena, especialmente en dispositivos con recursos limitados.
- Comprobar que las rutas de las imágenes y archivos en el `menu_data.json` sean relativas y estén correctamente definidas.

4.2. Arquitectura del menu

El sistema de menús interactivos desarrollado se basa en una arquitectura modular utilizando la librería A-Frame y componentes personalizados. A continuación, se detalla la estructura interna, el flujo de funcionamiento y los principales módulos que componen el sistema.

4.2.1. Estructura general del sistema

El componente principal se llama `menu` y es el encargado de inicializar la interfaz, generar los botones y etiquetas de los menús, cargar la configuración desde archivos JSON y gestionar los eventos de interacción.

El sistema se apoya en los siguientes módulos:

- `menu.js`: Componente principal que crea el menú, las flechas de navegación y los botones, y distribuye los elementos en el espacio.

- **sub-menu.js**: Controla la apertura de submenús al pulsar botones configurados con esa funcionalidad.
- **arrow_menu.js**: Permite la navegación entre menús mediante flechas izquierda y derecha.
- **pressable.js**: Detecta la interacción de la punta del dedo índice con los botones, basado en *hand-tracking*.
- **rounded-plane.js**: Genera los paneles con bordes redondeados que componen la interfaz visual.

Todos los elementos visuales son entidades de **A-Frame** que se insertan dinámicamente en la escena 3D y cuyas propiedades (posición, visibilidad, colores, etc.) se definen en función de la configuración que elija el usuario al personalizar el menu.

4.2.2. Funcionamiento interno

El flujo de funcionamiento del código para generar el componente de menu en su totalidad es:

- El componente **menu** crea el panel visual que servira de base para todo el menu.
- Mediante la función **Process_menu**, se carga el archivo **menu_data.json** que define los menús, submenús y botones.
- Cada botón puede ejecutar una acción personalizada o abrir un submenú, según lo configurado en el JSON.
- Los botones y etiquetas de cada menú se posicionan fuera del campo de visión inicialmente.
- El menú activo se muestra ubicando sus botones y etiquetas en posiciones predefinidas mediante la función **ubicar_menu**, en la que el componente con el campo activo a true se lleva a la ubicación del panel visual del menu.

- El usuario interactúa tocando los botones con la punta del dedo, lo que dispara eventos como `pressedended`, gestionados por los módulos `pressable`, `sub-menu` o `arrow_menus`.

4.2.3. Diagrama de arquitectura

A continuación se presenta dos diagramas personalizados ,el primero para explicar la lógica de la generación del menú de forma dinámica y un segundo para ver la interacción del usuario con el menú.

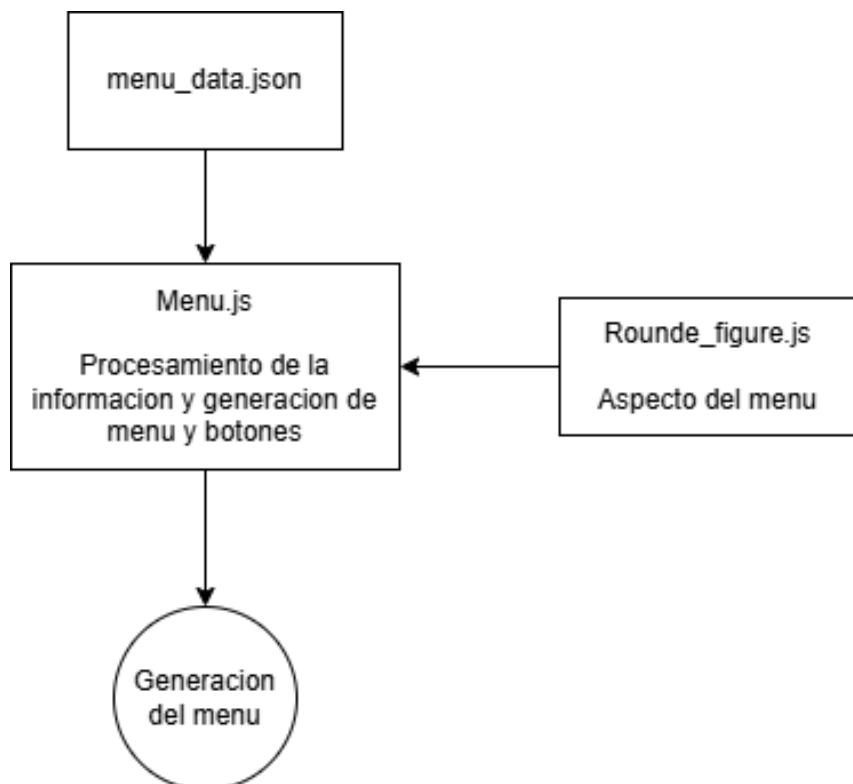


Figura 4.1: Esquema de generación del menú

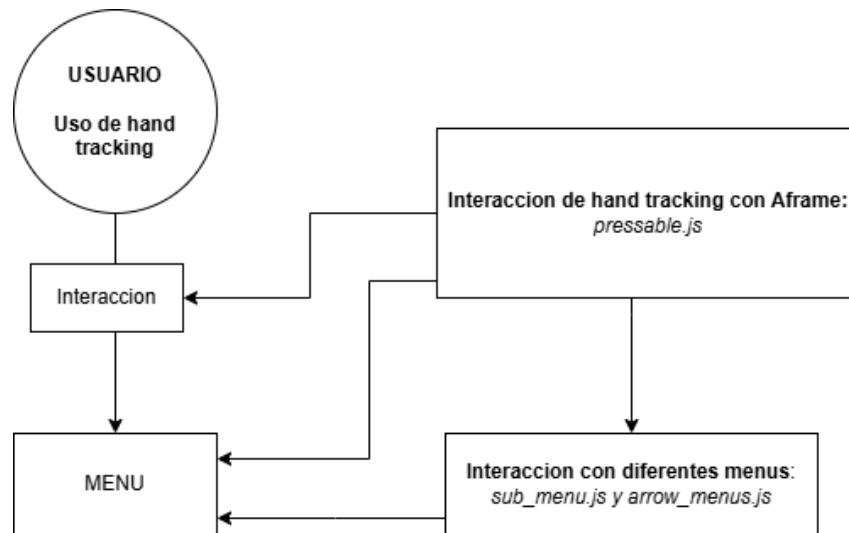


Figura 4.2: Esquema de interacción usuario-menú

En el primer esquema podemos ver como se genera el menú , en donde partimos del archivo .json y a partir de el mediante el procesamiento con el código de menú.js se generan tanto los botones como el menú , apoyado para darle formato a través del código de Rounded_figure.js

En el segundo esquema vemos ya un flujo de interacción del usuario mediante el uso de la tecnología del hand tracking como interactúa con el menú directamente mediante el componente de `pressable.js` que permite detectar cuando las manos colisionan con el menú y los objetos , usando la diferencia de la distancia del objeto con la punta del dedo indice y cuando esta diferencia es inferior a un umbral determinado se producen el inicio de dos eventos, estos eventos son `preseedstarted` y `pressedended` y se propagan a la escena y para poder ser detectados y iniciar un desencadenante de una acción al pulsar el botón y luego el funcionamiento del menú para ir desplazándose a través de los distintos submenus que forman el componente del menú mediante los componentes de `sub_menu.js` y `arrow_menu.js`.

4.2.4. Distribución de elementos en el espacio

El sistema utiliza una distribución predefinida de los botones en función del número de elementos activos en cada menú, permitiendo hasta 4 botones por menú. La ubicación y visibilidad de los elementos se controla dinámicamente:

- Menús activos: Botones y etiquetas posicionados en coordenadas visibles.
- Menús inactivos: Elementos desplazados fuera de la vista para ocultarlos.

La posición de los elementos se define en arrays dentro de las funciones `ubicar_menu`, presentes tanto en `menú.js` como en los componentes secundarios.

4.2.5. Personalización y escalabilidad

El sistema permite:

- Configurar los menús mediante un único archivo `.json`, permitiendo flexibilidad y mantenimiento sencillo.
- Adaptar el diseño del menú mediante cambios de colores y personalización de los botones pudiendo elegir las imágenes que se deseen.
- Variar el número de menús y submenús sin modificar la estructura interna del código , mediante rápidas modificaciones en el archivo `.json` .
- Incorporar nuevas acciones o funcionalidades en los botones de manera modular mediante la etiqueta de `action` que permite ponerle un atributo para una acción en particular a un botón.

Esta arquitectura modular y escalable facilita la integración del sistema en distintos entornos XR dentro del ecosistema **BabiaXR**.

4.3. Casos de Uso

4.3.1. Demo 1: Habitación personalizable

El primer caso de uso implementado es una demostración funcional de una habitación virtual personalizable, desarrollada íntegramente mediante la librería A-Frame y tecnologías complementarias como JavaScript y JSON. El objetivo principal es ofrecer al usuario la posibilidad de configurar el mobiliario de una estancia de manera interactiva, utilizando únicamente el seguimiento de manos, sin necesidad de controladores físicos.

4.3.1.1. Descripción del entorno

La escena se genera automáticamente al cargar la aplicación, incorporando un suelo y cuatro paredes que definen el espacio tridimensional. La geometría de la habitación es configurable, y en esta demostración se establece una altura de 5 unidades y dimensiones de 12x12 para los ejes x e z.

Ademas el sistema integra un controlador de cámara con movimiento para ver la escena en el navegador y el menú interactivo que permanece suspendido dentro de la habitación y como se ha explicado a lo largo del proyecto puede ser manipulado mediante gesto. Además, el sistema integra un controlador de cámara con movimiento libre y un menú flotante interactivo, el cual permanece suspendido dentro de la habitación y puede ser manipulado mediante gestos.

4.3.1.2. Interfaz de usuario: menú interactivo

El menú se implementa como una entidad personalizada de A-Frame, combinando planos redondeados (`rounded-plane`) y botones cilíndricos. Su diseño se adapta a criterios de accesibilidad, incluyendo colores de alto contraste y etiquetas visibles. Los botones son sensibles al tacto gracias al componente `pressable`, el cual detecta la proximidad del dedo del usuario utilizando la extensión `hand-tracking-controls`.

El menú principal presenta cuatro categorías: *Muebles*, *Camas*, *Mesas* y *Sillas*. Cada categoría despliega un submenú independiente, permitiendo al usuario navegar entre las diferentes opciones mediante flechas laterales.

4.3.1.3. Inserción de mobiliario

Cada elemento del menú representa un mueble que puede ser instanciado en la escena. Al seleccionar un botón, el sistema genera el modelo 3D correspondiente a partir de archivos .obj y .mtl ubicados en la carpeta de **assets**. Los modelos disponibles en esta demo incluyen:

- **Camas:** Cama genérica de dibujo animado y cama moderna (**cama_1** y **cama_2**).
- **Mesas:** Tres modelos distintos de mesas (**mesa_1**, **mesa_2**, **mesa_3**).
- **Muebles mixtos:** Frigorífico y mesilla auxiliar.
- **Sillas:** Modelo de silla básica.



Figura 4.3: Habitación con una cama

Cada objeto incluye un *collider* invisible que permite agarrar y mover el mueble dentro del espacio. Los *colliders* se generan dinámicamente y adaptan su tamaño y posición al modelo correspondiente.

4.3.1.4. Interacción sin controladores

La aplicación prescinde de controladores físicos, utilizando exclusivamente el seguimiento de las manos. El sistema detecta la posición de la punta del dedo índice y emite eventos de interacción cuando se aproxima lo suficiente a un botón, permitiendo así la navegación por los menús y la manipulación de los objetos en la escena.

4.3.1.5. Navegación entre submenús

La navegación entre submenús se implementa mediante las flechas laterales (`leftArrow` y `rightArrow`) situadas en el menú. Estas permiten recorrer los submenús de manera circular, manteniendo siempre visible solo el conjunto de botones correspondiente a la categoría seleccionada.

4.3.1.6. Conclusión

Esta demostración evidencia la viabilidad de un entorno virtual personalizable accesible mediante interacción natural, sin dispositivos externos. El sistema constituye la base para desarrollos más complejos que incluyan catálogos ampliados de objetos, validación de diseño o entornos multiusuario. Esta demo está disponible en YouTube en la ruta: Link demo.

4.3.2. Demo 2: Menú en el desierto

La segunda demostración tiene como finalidad trasladar el sistema de menú interactivo a un entorno temático: un paisaje desértico. Esta escena, además de ilustrar la portabilidad y adaptabilidad de la librería desarrollada, introduce nuevos objetos, modelos y funciones avanzadas de interacción.

4.3.2.1. Descripción del entorno

La ambientación simula un desierto, con un cielo claro y una fuente de luz ambiental, así como modelos tridimensionales que representan flora, fauna y elementos característicos de este bioma.

La arquitectura de la escena se mantiene consistente con el diseño modular de la primera demo, pero se adapta el contenido visual para ajustarse a la temática específica.

4.3.2.2. Estructura del menú

El menú principal en este caso se divide en tres categorías:

- **Seres vivos:** Permite acceder a submenús de *Plantas* y *Animales*.

- **Luz:** Incorpora un botón para alternar entre modo día y noche (`dark_mode`).
- **Paisaje:** Incluye elementos inertes del entorno como rocas y pozos.

La navegación entre los distintos menús se realiza mediante flechas laterales, de forma circular, garantizando la usabilidad en cualquier punto del sistema.

4.3.2.3. Configuración de colores del menu

La apariencia visual del menú en la Demo 2, correspondiente al escenario del desierto, se ha adaptado mediante una paleta de colores temática acorde al entorno. El sistema permite modificar fácilmente los siguientes parámetros, los cuales se definen directamente como atributos en la entidad que contiene el componente `menu`:

- `menu_color`: Color de fondo del menú principal.
- `button_color`: Color de los botones interactivos.
- `button_label_color`: Color del texto en los botones.
- `menu_label_color`: Color del texto que identifica los submenús.
- `arrow_color`: Color de las flechas de navegación laterales.

En esta demostración se emplea la siguiente configuración, diseñada específicamente para integrar el menú en el ambiente árido del desierto:

Parámetro	Color (Hexadecimal)	
<code>menu_color</code>	#C2B280	(arena clara)
<code>button_color</code>	#D97706	(ocre cálido)
<code>button_label_color</code>	#FFFFFF	(blanco puro)
<code>menu_label_color</code>	#8B5E3C	(marrón terroso)
<code>arrow_color</code>	#FBBF24	(amarillo dorado)

Esta paleta, además de ser coherente con la estética del desierto, garantiza un contraste adecuado y una experiencia de usuario clara y accesible.

Cabe destacar que, al ser un sistema completamente parametrizable, el usuario o desarrollador puede modificar estos colores en tiempo de desarrollo, adaptando el menú a diferentes temáticas o entornos sin necesidad de reescribir el código interno.

4.3.2.4. Modelos y funcionalidades

Al igual que en la demostración anterior, los botones del menú instancian modelos 3D a partir de archivos .obj y .mtl, gestionados dinámicamente mediante el sistema de assets de A-Frame. Los objetos disponibles en esta demo son:

- **Plantas:** Cactus y palmera.
- **Animales:** Serpiente y camello.
- **Otros elementos:** Pozo, roca.

Cada modelo incluye un *collider* invisible que permite su manipulación mediante el seguimiento de manos. Adicionalmente, el botón de *Luz* activa la función `dark_mode`, que alterna dinámicamente entre un entorno diurno y nocturno, modificando tanto el color del cielo como la intensidad de la iluminación.

4.3.2.5. Modo día/noche

El modo nocturno oscurece el cielo (#0d1b2a), atenúa la luz ambiental y oculta el sol. Al revertir la acción, se restablecen los valores diurnos, creando así una experiencia inmersiva y adaptable, controlada de forma natural a través de la interfaz táctil del menú.

4.3.2.6. Conclusión

Esta segunda demo no solo amplía el catálogo de modelos, sino que demuestra la versatilidad de la solución al integrarse en un escenario temático diferente. Asimismo, la funcionalidad de cambio ambiental en tiempo real supone un avance en términos de interacción y personalización, alineándose con los objetivos de accesibilidad y realismo inmersivo. Esta demo está disponible en YouTube en la ruta: [Link demo](#).

4.3.3. Demo 3:Realidad aumentada

Esta demostración corresponde a la adaptación de la Demo 1 , es decir , la habitación personalizable al ámbito de la realidad aumentada (AR). El objetivo es comprobar la versatilidad del sistema, permitiendo su uso tanto en entornos completamente virtuales

como en escenarios del mundo real, sin necesidad de realizar modificaciones profundas en la estructura del menú o la lógica de interacción.

A diferencia de la versión en realidad virtual, en esta demo no se genera un entorno cerrado compuesto por un suelo y unas paredes que simulen una habitación. En su lugar, los elementos tridimensionales —principalmente el menú interactivo y los objetos insertables— se proyectan directamente sobre la imagen capturada por la cámara del dispositivo, integrándose visualmente en el entorno que nos rodea , haciendo que el usuario lo sienta más realista.

La activación de la AR se realiza de forma sencilla gracias a las capacidades nativas de *A-Frame*.

```
<a-scene xr-mode-ui="XRMode: xr">
```

Esta configuración permite que la escena funcione en modo de realidad aumentada, siempre que se acceda desde un navegador compatible con *WebXR* y el dispositivo disponga de cámara. Una ventaja destacable de este enfoque es que no requiere la instalación de aplicaciones adicionales ni dependencias externas complejas, ya que está integrado dentro del propio *A-Frame* facilitando el acceso a la experiencia desde multitud de dispositivos.

A nivel funcional, se conserva el mismo sistema de menú interactivo implementado en la versión de realidad virtual. El usuario puede visualizar el menú suspendido en el espacio real y utilizar gestos de las manos para interactuar con los botones, navegar entre submenús e insertar objetos tridimensionales. Estos objetos se posicionan sobre la superficie física visible a través de la cámara, creando la ilusión de que forman parte del entorno del usuario.

Esta demo está disponible en YouTube en la ruta: [Link demo](#).

4.4. Disponibilidad del software

Esta demostración confirma que el sistema es flexible y compatible tanto con entornos virtuales como con aplicaciones de AR, abriendo la posibilidad de utilizarlo en contextos como la visualización de mobiliario, catálogos de productos o configuradores interactivos en el espacio real. Para acceder y explorar el proyecto, se han desarrollado varios recursos:

- Pagina web: Ofrece una visión completa del proyecto, incluyendo toda la documentación relevante y vídeos con demostraciones realizadas.

Link pagina web

- Repositorio de GitHub: Alberga el código fuente del proyecto:

Link GitHub

- Disponibilidad en npm: npm install design-room

Este proyecto forma parte de **BabiaXR**, una iniciativa de software libre enfocada en el desarrollo de experiencias inmersivas en realidad extendida (XR). El código se distribuye bajo la licencia **Apache License 2.0**, lo que garantiza que se trata de un software de código abierto, permitiendo su uso, modificación y distribución, siempre respetando los términos de dicha licencia.

Capítulo 5

Conclusiones

5.0.1. Consecución de objetivos

El presente Trabajo de Fin de Grado ha cumplido satisfactoriamente los objetivos planteados al inicio. Se ha logrado desarrollar una librería de menús interactivos y personalizables para entornos de Realidad Extendida (XR) empleando *A-Frame* y tecnologías web. El sistema permite la generación dinámica de menús configurables mediante archivos JSON, integrando funcionalidades como submenús enlazados, imágenes personalizables y navegación mediante flechas. Asimismo, se ha garantizado la compatibilidad con dispositivos como *Meta Quest 2*, incorporando interacción mediante seguimiento de manos, sin necesidad de controladores externos.

Además, se ha conseguido que el sistema sea modular, escalable y fácilmente integrable en otros proyectos, cumpliendo el propósito inicial de crear una herramienta reutilizable que facilite el trabajo de otros desarrolladores en el ámbito de XR.

5.0.2. Aplicación de lo aprendido

Durante el desarrollo del proyecto se han puesto en práctica y consolidado múltiples competencias adquiridas a lo largo de la titulación. Destacan especialmente los conocimientos en:

- Programación web avanzada, combinando HTML, JavaScript y manipulación dinámica del DOM.

- Uso y personalización de la librería A-Frame, creando componentes propios como `menu`, `pressable` o `rounded-plane`.
- Integración de sistemas de interacción natural mediante *hand-tracking* en entornos XR.
- Metodologías de desarrollo iterativo (sprints), planificación, detección de errores y optimización progresiva.
- Control de versiones y colaboración mediante GitHub.
- Redacción técnica y documentación en L^AT_EX.

5.0.3. Lecciones aprendidas

Este proyecto ha supuesto un importante reto técnico y ha permitido identificar varios aspectos clave:

- La importancia de diseñar sistemas flexibles y escalables desde el inicio, para evitar limitaciones en fases avanzadas del desarrollo.
- La complejidad que implica trabajar en entornos tridimensionales inmersivos, donde se deben considerar no solo aspectos visuales, sino también la ergonomía y la naturalidad de la interacción.
- El valor de la interacción sin mandos, utilizando exclusivamente las manos, que aporta mayor realismo, como mecanismo de desarrollo.

En definitiva, el proyecto ha reforzado la capacidad para abordar problemas complejos de forma estructurada, así como para combinar distintos conocimientos y tecnologías en un producto funcional y de utilidad real.

5.0.4. Trabajos futuros

- Incorporar más tipos de interacción, como gestos avanzados o reconocimiento de voz..

- Optimizar aún más el sistema de detección de proximidad para evitar problemáticas al tener un gran número de elementos.
- Ampliar el sistema de menús para permitir un mayor grado de personalización como tipos botones , como un botón deslizable.
- Crear una interfaz gráfica externa que permita a usuarios sin conocimiento en HTML ni en A-Frame puedan desarrollar sus escenarios visuales.

Bibliografía

- [1] A-Frame. (s.f.). *A-Frame – Make WebVR*. Disponible en: <https://aframe.io/>
- [2] Mozilla. (2023). *WebXR Device API*. MDN Web Docs. Disponible en: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API
- [3] Khronos Group. (2023). *WebXR Overview*. Disponible en: <https://www.khronos.org/webxr/>
- [4] OpenJS Foundation. (s.f.). *Three.js Documentation*. Disponible en: <https://threejs.org/docs/>
- [5] MDN Web Docs. (2023). *Using JSON*. Disponible en: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>
- [6] ISO/IEC. (2018). *ISO/IEC 18039:2018 Virtual Reality and Augmented Reality — Terminology and Concepts*. Disponible en: <https://www.iso.org/standard/70669.html>
- [7] Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, 77(12), 1321–1329.
- [8] A-Frame Documentation. (2024). *A-Frame Official Documentation*. Recuperado de: <https://aframe.io/docs/>
- [9] AR.js. (2024). *Efficient Augmented Reality for the Web*. Recuperado de: <https://ar-js-org.github.io/AR.js-Docs/>
- [10] Mozilla. (2024). *WebXR: Immersive Experiences on the Web*. Recuperado de: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API

- [11] W3C. (2024). *WebXR Hand Input Module - Editor's Draft*. Recuperado de: <https://immersive-web.github.io/webxr-hand-input/>
- [12] BabiaXR. (2024). *Framework de Realidad Extendida Abierta*. Recuperado de: <https://babiaxr.github.io/>
- [13] Three.js. (2024). *3D JavaScript Library Documentation*. Recuperado de: <https://threejs.org/docs/>
- [14] ISO/IEC. (2018). *ISO/IEC 18039:2018 Virtual Reality and Augmented Reality — Terminology and Concepts*. Recuperado de: <https://www.iso.org/standard/70669.html>
- [15] Normand, J. M., et al. (2012). *The Role of Natural Interaction in Virtual Reality*. Journal of Virtual Reality and Broadcasting, 9(2). Recuperado de: <http://www.jvrb.org/past-issues/9.2012/3693>
- [16] Khronos Group. (2024). *WebXR Overview*. Recuperado de: <https://www.khronos.org/webxr/>
- [17] Don McCurdy. (2024). *A-Frame Super Hands Component*. Recuperado de: <https://github.com/wmurphyrd/aframe-super-hands-component>