



ESCUELA DE INGENIERÍA DE FUENLABRADA

**GRADO EN INGENIERÍA EN SISTEMAS
AUDIOVISUALES Y MULTIMEDIA**

TRABAJO FIN DE GRADO

**VISUALIZACIÓN DE DATOS MULTIUSUARIO EN
REALIDAD VIRTUAL**

Autor : Flavio Andrés Cuichan Flores

Tutor : Dr. Jesús María González Barahona

Curso académico 2024/2025

Trabajo Fin de Grado

Visualización de datos multiusuario en realidad virtual

Autor : Flavio Andrés Cuichan Flores

Tutor : Dr. Jesús María González Barahona

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

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente **Calificación:**

Fuenlabrada, a de de 2024

©2024 Flavio Andrés Cuichan Flores

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>

*Dedicado a la persona
que no sabía si podría lograrlo
pero que persistió y nunca se dio por vencido...*

Yo.

Agradecimientos

Quiero agradecer a mis padres por todo lo que han sacrificado para que yo pueda convertirme en ingeniero. Su esfuerzo en trabajos duros y su constante apoyo han sido fundamentales para alcanzar mis metas. Aprecio profundamente su dedicación al trabajar incansablemente, esforzándose siempre por brindarme un futuro mejor. Espero que todos sus sacrificios hayan valido la pena.

También quiero agradecer a mi hermana y a Raquel, quien siempre han estado a mi lado consintiéndome y espero que continúen haciéndolo en el futuro. Y por último, quiero dedicar unas palabras a la última incorporación a la familia, mi querida sobrina Alaia. Desde tu tío favorito, tampoco es que tengas tantos, quiero que sepas que cualquier cosa que te propongas, lo puedes alcanzar.

Quiero extender mi reconocimiento a mi tutor, Jesús María González Barahona, por su valiosa guía a lo largo de este arduo proceso de realizar mi Trabajo Fin de Grado. Su apoyo, paciencia y conocimientos han sido esenciales para superar todos los desafíos que han surgido en el camino

Resumen

Este Trabajo Fin de Grado se centra en desarrollar un sistema de visualización de datos multiusuario en realidad virtual que permita a los usuarios crear y configurar elementos de visualización. Facilitando así la interacción y exploración de conjuntos de datos de forma inmersiva y compartida. Para ello, se ha diseñado una interfaz de usuario intuitiva que optimiza la experiencia de interacción en entornos virtuales, permitiendo nuevas formas de análisis y comprensión de la información.

La visualización de datos en realidad virtual es un campo emergente que, aunque promete mejorar significativamente la interacción y comprensión de la información, todavía enfrenta diversos desafíos. Entre ellos, destacan las dificultades para acceder a la realidad virtual desde diversos dispositivos y la necesidad de interfaces más intuitivas y accesibles, aspectos que este proyecto aborda de manera directa.

El sistema de visualización de datos desarrollado en realidad virtual se basa en el framework A-Frame, lo que permite la creación y configuración de elementos de visualización de datos mediante la librería BabiaXR. Además, se ha integrado Networked A-Frame, lo que habilita la creación de escenas multiusuario, donde múltiples personas pueden conectarse simultáneamente, interactuando y manipulando datos en tiempo real, a la vez que se comunican entre sí.

Summary

This Bachelor's Thesis focuses on developing a multi-user data visualization system in virtual reality that allows users to create and configure visualization elements. This facilitates interaction and exploration of data sets in an immersive and shared manner. To achieve this, an intuitive user interface has been designed to optimize the interaction experience in virtual environments, enabling new ways of analyzing and understanding information.

Data visualization in virtual reality is an emerging field that, while promising to significantly enhance interaction and understanding of information, still faces various challenges. Among these are difficulties in accessing virtual reality from different devices and the need for more intuitive and accessible interfaces—issues that this project addresses directly.

The data visualization system developed in virtual reality is based on the A-Frame framework, which allows for the creation and configuration of data visualization elements using the BabiaXR library. Additionally, Networked A-Frame has been integrated, enabling the creation of multi-user scenes where multiple people can connect simultaneously, interact, and manipulate data in real time, while communicating with each other.

Índice general

1. Introducción	1
1.1. Contexto	2
1.2. Objetivos	3
1.2.1. Objetivo general	3
1.2.2. Objetivos específicos	4
1.3. Estructura de la memoria	5
1.4. Disponibilidad del software	6
2. Tecnologías utilizadas	7
2.1. Tecnologías principales	7
2.1.1. WebGL	8
2.1.2. WebXR	8
2.1.3. Three.js	9
2.1.4. HTML	9
2.1.5. JavaScript	10
2.1.6. A-Frame	11
2.1.7. Networked A-frame	13
2.1.8. BabiaXR	15
2.2. Tecnologías auxiliares	17
2.2.1. GitHub	17
2.2.2. Visual Studio Code	18
2.2.3. Meta Quest 2 y 3	18
2.2.4. LaTeX	19

3. Sistema resultante	21
3.1. Descripción funcional	21
3.1.1. Inicio del sistema de visualización de datos	22
3.1.2. Interfaz de usuario para componentes de tipo query	24
3.1.3. Interfaz de usuario para componentes de tipo filtro	28
3.1.4. Interfaz de usuario para componentes de tipo gráfica	31
3.2. Arquitectura técnica	35
3.2.1. Implementación de la escena	35
3.2.2. Componentes principales	39
3.2.3. Componentes complementarios	47
4. Desarrollo del proyecto	53
4.1. Sprint 1: Primeros pasos en realidad virtual	54
4.1.1. Principios básicos de A-frame	54
4.1.2. Desarrollo de programas de prueba	55
4.1.3. Exploración y pruebas con gafas de realidad virtual	56
4.2. Sprint 2: Escenas multiusuario y extensiones de A-Frame	56
4.2.1. Análisis de Networked A-Frame	57
4.2.2. Introducción de Superhands y A-Frame Extras	57
4.2.3. Desarrollo de programas de prueba	58
4.3. Sprint 3: Etapa final de formación	59
4.3.1. Desarrollar un minijuego	60
4.3.2. Introducción de BabiaXR	61
4.3.3. Desarrollo de programa de prueba	61
4.4. Sprint 4: Inicio del proyecto	62
4.4.1. Estructura principal del proyecto	63
4.4.2. Crear una primera versión de la interfaz de usuario	64
4.5. Sprint 5: Progreso de la interfaz de usuario	66
4.5.1. Optimizar el diseño de la interfaz	67
4.5.2. Implementar componentes de tipo de gráficas	69
4.6. Sprint 6: Sistema de visualización de datos	70

4.6.1. Evolucionar la interfaz de usuario	71
4.6.2. Implementar componentes de tipo filtro	73
4.7. Sprint 7: Sistema completo	75
4.7.1. Implementar Networked A-frame a la escena	76
4.7.2. Agregar avatares y controles	78
5. Conclusiones	81
5.1. Consecución de objetivos	81
5.2. Aplicación de lo aprendido	82
5.3. Lecciones aprendidas	83
5.4. Trabajos futuros	84
Bibliografía	85

Índice de figuras

2.1. Escenario básico de A-Frame	11
2.2. Escena multiusuario con Networked A-frame	13
2.3. Visualización de datos con componentes de BabiaXR	15
3.1. Escena con varios usuarios conectados.	22
3.2. Escena con el panel de instrucciones.	23
3.3. Escena con el menú inicial de creación.	24
3.4. Escena con el menú de creación de tipo query.	25
3.5. Escena de los menús de opciones para componentes creados.	26
3.6. Escena de configuración de babia-queryjson.	27
3.7. Escena de información de babia-queryjson.	27
3.8. Escena de configuración de babia-querycsv.	28
3.9. Resultado final: Integración de un babia-queryjson y un babia-querycsv a la escena.	28
3.10. Escena con el menú de creación de tipo filtro.	29
3.11. Escena de configuración de babia-filter.	30
3.12. Escena de información de babia-filter.	30
3.13. Resultado final: Integración de un babia-filter a la escena.	31
3.14. Escena con el menú de creación de tipo gráfica.	31
3.15. Escena de configuración de babia-doughnut.	32
3.16. Escena de configuración de la propiedad From de babia-doughnut.	33
3.17. Escena de información babia-doughnut.	34
3.18. Escena compartida de configuración de babia-pie.	34
3.19. Resultado final: Integración de un babia-doughnut y un babia-pie a la escena.	35

3.20. Estructura de los archivos del proyecto.	36
4.1. Resultado del programa de prueba.	55
4.2. Escena agarrando una caja con superhands.	58
4.3. Escena compartida con avatares.	59
4.4. Escena compartida del minijuego.	61
4.5. Escena con gráficas de datos.	61
4.6. Escena compartida de la interfaz interactiva.	62
4.7. Escenario con ícono de creación.	64
4.8. Escenario con iconos de componentes tipo queries.	65
4.9. Escenario de configuración del componente babia-queryjson.	65
4.10. Interfaz de usuario interactiva mejorada	67
4.11. Interfaz de usuario de tipo query	68
4.12. Interfaz de usuario de tipo query y gráfica.	70
4.13. Escena de la etapa de creación del componente query.	72
4.14. Escena con los tres menús de opciones.	73
4.15. Escena de la etapa de creación del componente filtro.	74
4.16. Escena del menú de configuración del componente filtro.	75
4.17. Escena con dos usuarios conectados.	78

Capítulo 1

Introducción

En el ámbito de la visualización de datos, los gráficos en 2D han sido cruciales para las herramientas analíticas más comunes en la actualidad, como Tableau, Power BI y Excel, entre otros. Estas plataformas han facilitado a los usuarios una interpretación rápida y eficiente de los datos. Sin embargo, con el incremento en la complejidad y multidimensionalidad de los conjuntos de datos, las limitaciones de las representaciones bidimensionales tradicionales se vuelven cada vez más evidentes. Estas limitaciones incluyen la dificultad para mostrar múltiples variables de manera clara, la potencial pérdida de información crucial y la sobrecarga cognitiva derivada de gráficos excesivamente complejos. Además, las capacidades de interactividad y escalabilidad en el formato 2D a menudo resultan insuficientes para un análisis en profundidad y dinámico. Estas limitaciones destacan la necesidad de adoptar nuevas metodologías de visualización, como las representaciones en 3D y realidad virtual, que brindan mejores oportunidades de interacción y una representación más detallada y completa de los datos.

Mi Trabajo de Fin de Grado se centra en superar las limitaciones de la visualización tradicional en 2D mediante la integración de realidad virtual, utilizando principalmente el framework A-Frame. Para ello, se ha desarrollado un sistema que permite la creación dinámica de escenas para la visualización de datos con el apoyo de la biblioteca BabiaXR. Además, se ha implementado la funcionalidad de realidad virtual multiusuario utilizando Networked-Aframe, lo que permite que múltiples usuarios interactúen simultáneamente en el entorno virtual. Esta capacidad multiusuario no solo enriquece la experiencia de visualización de datos al permitir la colaboración en tiempo real, sino que también mejora la interacción y el análisis compartido de la información, facilitando una comprensión más profunda y dinámica de los datos.

1.1. Contexto

La realidad virtual (RV) tiene sus raíces en 1963, cuando Hugo Gernsback presentó The Teleyeglasses, un prototipo pionero de gafas inmersivas. Este dispositivo fue una televisión portátil que se sujetaba a la cabeza, lo que permitía a los usuarios sentirse casi dentro del mundo televisado. Otro avance crucial llegó en 1968 con Ivan Sutherland del MIT, que desarrolló The Sword of Damocles. El auténtico precursor de las gafas de realidad virtual actuales. Este sistema, compuesto por un casco sostenido por un brazo mecánico fijado al techo, utilizaba dos pantallas CRT para mostrar gráficos simples que seguían la rotación de la cabeza del usuario. Tanto The Teleyeglasses como The Sword of Damocles fueron fuentes de inspiración para el desarrollo continuo de la tecnología de realidad virtual.

La realidad virtual experimentó un avance significativo desde que Palmer Luckey lanzó la campaña de Kickstarter para Oculus Rift, recaudando 2,4 millones de dólares y atrayendo la atención de Facebook, que compró Oculus VR por 2.000 millones en 2014. Desde entonces, el mercado ha evolucionado con la aparición de dispositivos como HTC Vive, Oculus Quest, y más recientemente, Apple Vision Pro y Meta Quest 3. Estos desarrollos han potenciado la adopción de la realidad virtual en campos como el entretenimiento, la educación y la salud. Empresas como Meta y Apple están a la vanguardia de este sector, invirtiendo fuertemente y desarrollando tecnologías que están haciendo que la realidad virtual sea cada vez más accesible y poderosa.

Una de las aplicaciones más importantes de la realidad virtual es la visualización de datos. Esta aplicación permite crear entornos digitales totalmente inmersivos en los que los usuarios pueden explorar y analizar información de manera tridimensional y altamente interactiva. Mediante la realidad virtual, los datos se presentan en contextos virtuales que facilitan una comprensión más profunda y dinámica. Los usuarios pueden interactuar con gráficos y conjuntos de datos en un entorno digital, lo que mejora la capacidad de interpretar y manipular información compleja de formas innovadoras. Esta aplicación se puede encontrar en varios campos clave como en negocios para análisis de datos inmersivos, en educación para una comprensión interactiva, en investigación científica para simular datos detalladamente, y en salud para mejorar diagnósticos y planificación de tratamientos.

La realidad virtual ha sido tradicionalmente asociada con experiencias inmersivas pero so-

litarias. No obstante, los avances tecnológicos han permitido el desarrollo de experiencias de realidad virtual multiusuario, donde varias personas pueden interactuar en entornos virtuales compartidos de manera simultánea. Este tipo de interactividad plantea ciertos desafíos, como la necesidad de sincronización precisa para que todos los participantes experimenten el mismo entorno en tiempo real, la minimización de la latencia de red para evitar demoras en la comunicación, y la importancia de garantizar la seguridad y la privacidad de los datos de los usuarios en estos espacios virtuales. A diferencia de la RV individual, esta modalidad se centra en actividades colaborativas, tales como juegos multijugador, donde los usuarios pueden competir o colaborar en equipos, herramientas de trabajo colaborativo que facilitan la cooperación en proyectos profesionales, y aulas virtuales donde estudiantes y docentes pueden interactuar en un entorno educativo compartido.

En mi Trabajo de Fin de Grado, profundizo en la convergencia de la realidad virtual multiusuario con la visualización de datos, explorando cómo estas tecnologías pueden transformar la manera en que interpretamos y colaboramos con la información. Me centro en cómo la realidad virtual proporciona una plataforma inmersiva que permite a múltiples usuarios interactuar con los datos de forma simultánea, superando las limitaciones de las herramientas tradicionales de análisis de datos. Este enfoque no solo facilita una comprensión más intuitiva de la información compleja, sino que también fomenta una mayor interactividad y participación entre los usuarios. Así, mi TFG destaca el potencial de la realidad virtual multiusuario para revolucionar la visualización de datos, proporcionando un sistema de visualización de datos en la que los participantes puedan explorar y analizar las visualizaciones de datos de manera compartida en tiempo real.

1.2. Objetivos

En esta sección se exponen los objetivos del Trabajo de Fin de Grado, que guiarán la investigación y la implementación desarrolladas.

1.2.1. Objetivo general

El propósito central de este proyecto es desarrollar una interfaz de usuario que permita la creación dinámica de escenas de visualización de datos en realidad virtual, permitiendo que los

elementos puedan ser configurados simultáneamente por múltiples usuarios.

1.2.2. Objetivos específicos

Para lograr el objetivo principal del proyecto, se implementarán diversos objetivos secundarios, los cuales se describen a continuación:

- **Implementar una herramienta de visualización de datos interactiva:** Construir una aplicación en el entorno de BabiaXR para ofrecer una interfaz de visualización de datos en realidad virtual, utilizando A-Frame como base tecnológica. La interfaz permitirá crear y configurar componentes de BabiaXR, incluyendo componentes de queries para extraer datos, componentes de filtros para refinar los datos, y componentes de gráficas para visualizar los datos.
- **Integrar escenas multiusuario:** Desarrollar escenas compartidas en realidad virtual donde los usuarios, representados por avatares con diferentes apariencias, puedan interactuar con los componentes de visualización de datos y comunicarse en tiempo real utilizando la tecnología de Networked A-Frame.
- **Desarrollar una interfaz para componentes queries de BabiaXR:** Crear una interfaz que permita crear y configurar componentes queries que actúen como fuentes de datos, extrayendo información desde archivos JSON y CSV.
- **Desarrollar una interfaz para componentes filtros de BabiaXR:** Implementar una interfaz que facilite la creación y configuración de componentes filtros para seleccionar y filtrar partes específicas de los datos.
- **Desarrollar una interfaz para componentes gráficas de BabiaXR:** Diseñar una interfaz que permita la creación y configuración de componentes gráficas, incorporando varios tipos de gráficas para representar los datos.
- **Aplicar estrategias basadas en metodologías ágiles:** Utilizar prácticas inspiradas en la metodología Scrum para mejorar la eficiencia de la gestión y coordinación del desarrollo del proyecto.

- **Compatibilidad Multi-Dispositivo:** Garantizar la compatibilidad del programa con diversos dispositivos, incluyendo aquellos sin capacidades de realidad virtual, asegurando que se pueda acceder y utilizar sin necesidad de más complementos.

1.3. Estructura de la memoria

A continuación se presenta la estructura de la memoria, organizada en los siguientes capítulos:

- **Tecnologías utilizadas:** Este capítulo explora todas las tecnologías empleadas en el proyecto, dividiéndolas en categorías de tecnologías principales y auxiliares. Con el fin de explicar la funcionalidad de cada herramienta utilizada y de ofrecer una visión detallada de su impacto en el proyecto.
- **Sistema resultante:** Se presenta el resultado final del proyecto, que culmina en la creación de un sistema de visualización de datos. Este capítulo tiene como objetivo explicar la funcionalidad completa del sistema y todas las posibles acciones que un usuario puede realizar dentro de la escena de realidad virtual. Además, se describe la arquitectura de los componentes desarrollados que hicieron posible la implementación de este sistema.
- **Desarrollo del proyecto:** En este capítulo se examina todo el proceso de desarrollo que condujo a la creación del sistema resultante de visualización de datos. Guiado por enfoques basados en la metodología ágil Scrum, que ha estructurado cada fase del proyecto. Las etapas del desarrollo se explican, abordando las necesidades y desafíos que surgieron a lo largo del trabajo.
- **Conclusiones:** En este último capítulo se evalúa el cumplimiento de los objetivos planteados, analizando si se han desarrollado de manera efectiva. Además, se reflexiona sobre las nuevas habilidades adquiridas a lo largo del proyecto y las aplicaciones de conocimientos obtenidos en la carrera que han sido fundamentales para completar este trabajo. También, se contemplan futuras implementaciones que podrían optimizar aún más el proyecto.

1.4. Disponibilidad del software

Para acceder y explorar el proyecto, se han desarrollado varios recursos:

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

<https://andrescuichanflores.github.io/>

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

<https://github.com/AndresCuichanFlores/TFG>

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. Tecnologías principales

En esta sección se presentarán las tecnologías esenciales empleadas para la construcción del proyecto. La base fundamental del desarrollo ha sido A-Frame, un framework que ha definido los cimientos y la estructura global del trabajo fin de grado. A-Frame se desarrolla dentro del navegador apoyándose en tecnologías clave de realidad virtual a través de la web como WebGL, que permite renderizar gráficos en 3D, WebXR, que facilita la integración de experiencias de realidad virtual y aumentada en el navegador, y Three.js, una biblioteca que simplifica la creación de escenas 3D. Además, se utilizan dos librerías fundamentales que amplían las capacidades de A-Frame, y son esenciales para el desarrollo del proyecto. La primera, BabiaXR, sobre la cual se articula toda la lógica de la interfaz de usuario para la creación y configuración de los componentes relacionados con la visualización de datos. La segunda, Networked-AFrame, que permite la sincronización en tiempo real de las escenas entre varios usuarios conectados. El código del proyecto se construye principalmente en JavaScript, el lenguaje que A-Frame utiliza para su funcionamiento, junto con HTML para la estructura en el navegador.

2.1.1. WebGL

WebGL[3] (Web Graphics Library) es una tecnología multiplataforma de bajo nivel para la renderización de gráficos tridimensionales y bidimensionales dentro de cualquier navegador web compatible.

Creado por el grupo Khronos, WebGL se fundamenta en OpenGL ES, una variante simplificada de OpenGL diseñada para dispositivos móviles. Desde su lanzamiento en 2011, WebGL ha experimentado una evolución continua, con múltiples actualizaciones que han mejorado su funcionalidad y compatibilidad tanto en navegadores de escritorio como en dispositivos móviles.

La API de WebGL está diseñada para interactuar directamente con la Unidad de Procesamiento Gráfico (GPU) del dispositivo, permitiendo así el aprovechamiento de la computación para generar gráficos detallados y de alta calidad. Además, WebGL se destaca por su capacidad de interoperabilidad, integrándose de manera fluida con otros estándares web como HTML, CSS y DOM.

WebGL aporta al proyecto la renderización de gráficos 3D, proporcionando la base necesaria para crear entornos tridimensionales detallados y de alta calidad en navegadores web.

2.1.2. WebXR

WebXR[4] es una API (application programming interface) que proporciona la funcionalidad necesaria para llevar la realidad mixta, compuesta por todo el espectro de experiencias entre realidad Aumentada (AR) y Realidad Virtual (VR), a sitios web mediante el uso de WebGL.

WebXR se concibió a partir de la API WebVR, desarrollada por Mozilla en 2014 para explorar el mundo de la realidad virtual en navegadores web. La necesidad de integrar tanto la realidad aumentada como la realidad virtual hizo que en 2018 la API WebVR fuera sustituida por WebXR. Esto permitió que los principales navegadores empezaran a implementar WebXR, haciendo accesibles experiencias tridimensionales en diversas aplicaciones, desde juegos hasta educación.

WebXR está diseñada para interactuar con hardware de realidad mixta, tales como cascos y gafas de realidad virtual, para contenido web y aplicaciones. Esta API abarca la administración del proceso de renderización de las vistas requeridas para simular experiencias 3D, como de-

tectar el movimiento de los cascos de realidad virtual o suministrar los datos para actualizar las imágenes visualizadas por el usuario.

WebXR contribuye al proyecto mediante la capacidad de integrar y gestionar de manera nativa experiencias de realidad virtual y aumentada en diversos dispositivos como gafas de realidad virtual, dispositivos móviles y ordenadores portátiles, los cuales se utilizaron para probar y validar el proyecto.

2.1.3. Three.js

Three.js[2] es una biblioteca de código abierto de JavaScript para crear y mostrar gráficos tridimensionales en la web. Funciona directamente sobre WebGL que permite la generación de animaciones 3D aceleradas por GPU sin necesidad de complementos del navegador.

Tree.js fue creado por el español Ricardo Cabello en 2010, también conocido como "Mrdoob". Las primeras versiones de Tree.js tenían como objetivo simplificar drásticamente el proceso del manejo de WebGL. Three.js empezó a obtener gran popularidad rápidamente debido a la comunidad activa de desarrolladores que creaban escenas en 3D de forma más rápida y sencilla. A lo largo de los años, Three.js se ha establecido como una herramienta esencial para gráficos 3D en la web, utilizada en videojuegos, visualizaciones científicas, arte interactivo y marketing digital. Three.js ofrece una amplia gama de herramientas para definir texturas, crear geometrías y manipular la iluminación. Con esta biblioteca, los desarrolladores tienen la opción de construir escenas en 3D mucho más ricas con animaciones que simulan el movimiento de los objetos gracias a la generación de gráficos 3D en tiempo real y la posibilidad de importar modelos 3D desde distintos formatos de archivo.

Three.js fue esencial para proporcionar una capa de abstracción sobre WebGL, simplificando el proceso de creación y manipulación de gráficos 3D. Permitiendo entornos 3D más dinámicos y con mayor control para el proyecto.

2.1.4. HTML

HTML[7] (HyperText Markup Language) es un conjunto de símbolos de marcado estándar utilizado por los navegadores web para representar páginas en la World Wide Web (W3C). El marcado indica a los navegadores cómo mostrar las palabras e imágenes de una página web.

A inicios de los años 90, el físico Tim Berners Lee, publicó una obra conocida como el origen de HTML, «HTML Tags». Esta publicación permitió a los científicos del CERN intercambiar información de manera rápida y fácilmente referencial a través de un sistema de hipertexto. En 1995 se publicó la segunda versión, HTML 2.0, que fue la primera propuesta oficial de estandarización de HTML. En 1997, evolucionó significativamente con HTML 3.2, que introdujo mejoras como soporte para estilos y scripts. El gran cambio llegó con HTML5, versión estándar utilizada en la actualidad, que incorporó soporte nativo para multimedia, gráficos y almacenamiento local.

HTML está formado por etiquetas predefinidas que se encuentran entre corchetes angulares (<>). Estas etiquetas definen diferentes elementos de una página web, como encabezados, párrafos, formularios, enlaces, imágenes y más. El conjunto de todas estas etiquetas estructuran el HTML y facilitan la lectura eficiente del contenido a las personas y a los navegadores. HTML permite crear hipervínculos para navegar entre páginas web y es compatible con todos los navegadores y sistemas operativos. Además, ofrece características como consultas de medios y marcos responsivos para asegurar una buena visualización en diferentes dispositivos y tamaños de pantalla. También, permite poder agregar CSS (Cascading Style Sheets), así como de JavaScript al documento digital, lo que proporciona una combinación visual impactante y una experiencia dinámica e interactiva para el usuario.

HTML fue una pieza muy importante en el trabajo, al ofrecer una estructura esencial y una semántica adecuada para mostrar los elementos visuales en realidad virtual.

2.1.5. JavaScript

JavaScript es un lenguaje de programación compilado interpretado, ligero y multiplataforma para crear contenido dinámico para páginas web. Crea elementos para mejorar la interacción de los usuarios en páginas web. Además, se puede utilizar tanto para desarrollos del lado del cliente como para desarrollos del lado del servidor.

JavaScript, creado por Brendan Eich de Netscape en 1995, comenzó como Mocha, luego se renombró a LiveScript y finalmente a JavaScript. Este nombre coincidió con la integración de Java en Netscape Navigator, lo que generó confusión sobre su relación con Java. En 1997, JavaScript fue estandarizado como ECMAScript por ECMA y poco después como un estándar ISO, lo que facilitó su adopción global. A partir de 2000, JavaScript se expandió al lado del

servidor con tecnologías como Node.js, y su popularidad creció con la llegada de AJAX. ECMAScript 6, publicado en 2015, introdujo características avanzadas y actualizaciones anuales, que han consolidado a JavaScript como uno de los lenguajes de programación más importantes en la web.

JavaScript es un lenguaje de alto nivel, su programación se parece al lenguaje humano y evita muchas dificultades de la máquina. El lenguaje está orientado a objetos mediante prototipos en lugar de clases tradicionales como en otros lenguajes. Además, está basado en eventos como clics, movimientos del ratón y teclas pulsadas, lo que lo hace perfecto para crear interfaces de usuario interactivas. JavaScript puede vincularse directamente a un archivo HTML escribiendo el código dentro de la etiqueta script. También, podemos escribir código JavaScript en otros archivos que tengan la extensión .js y luego vincular este archivo dentro de la etiqueta head del archivo HTML.

La mayor parte del desarrollo del proyecto fue escrito en JavaScript, lo que permitió implementar completamente la lógica y controlar los comportamientos de las escenas de realidad virtual del trabajo.

2.1.6. A-Frame

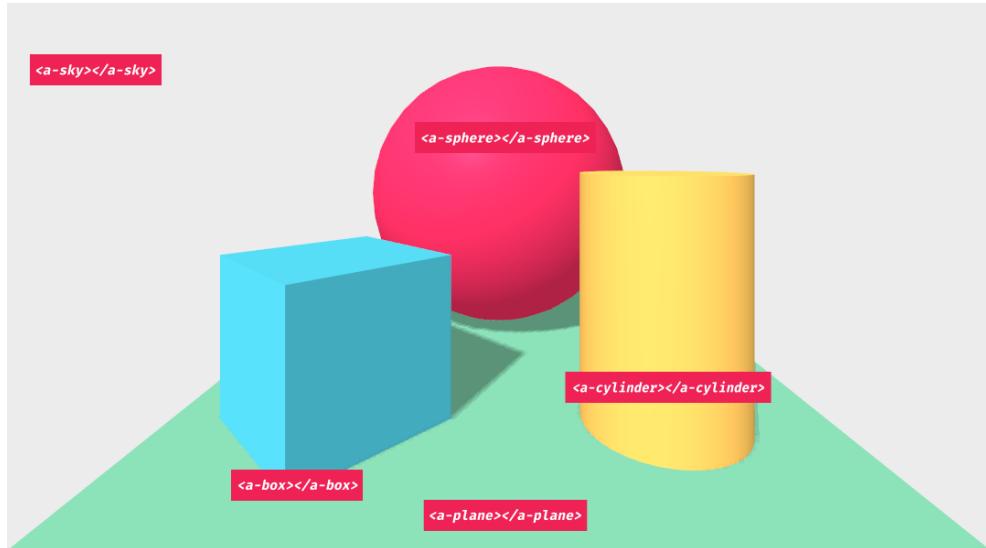


Figura 2.1: Escenario básico de A-Frame

A-frame[1] es un framework en JavaScript para construir realidades virtuales, ofreciendo infinitas posibilidades sin necesidad de conocimientos avanzados en gráficos 3D. Este framework

usa la arquitectura ECS (Entity Component System), que se aplica en el desarrollo de juegos donde cada objeto es una entidad. Además, A-Frame emplea HTML para definir escenas, entidades y componentes, permitiendo a los desarrolladores reutilizar y crear nuevos elementos con eficacia. El navegador sirve como fuerza impulsora detrás del funcionamiento del framework, ya que amplifica la accesibilidad de realidad virtual y realidad extendida al llegar a una amplia gama de dispositivos.

A-Frame fue desarrollado por Mozilla y se lanzó por primera vez en 2015. La intención era ofrecer una herramienta que facilitara la creación de experiencias de realidad virtual en la web mediante el uso de tecnologías comunes como HTML y JavaScript. A-Frame ganó rápidamente popularidad debido a su enfoque intuitivo y accesible. La comunidad de A-Frame fue una pieza clave por las múltiples contribuciones regulares al código base y una creciente cantidad de proyectos y recursos creados por usuarios. Con la evolución de la especificación WebXR, A-Frame se actualizó para soportar tanto realidad virtual como aumentada, mejorando la compatibilidad y la funcionalidad. En la actualidad, A-Frame sigue recibiendo actualizaciones y mejoras, centrándose sus esfuerzos en asegurar la compatibilidad con las tecnologías web más recientes y en optimizar continuamente la experiencia del desarrollador.

A continuación se muestra el código necesario para crear una entidad de una caja 3D usando HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <!-- Entidad de un caja -->
      <a-entity geometry="primitive: box; depth: 2; height: 2; width: 2"
        material="color: blue; position="0 1 -5" rotation="0 45 0">
      </a-entity>
    </a-scene>
  </body>
</html>
```

Para construir una escena en 3D, solo es necesario importar la biblioteca de A-Frame e

incluir la etiqueta `a-scene` dentro del código `body` del HTML, y el programa estará listo para generar una escena en realidad virtual. Dentro de las etiquetas `a-scene`, podemos definir las entidades, objetos contenedores donde los componentes son ligados para otorgarles propiedades. También, podemos identificar componentes, propiedades que hacen que una entidad sea diferente a otra modificando su comportamiento, apariencia y funcionalidad. En A-Frame, existen componentes predefinidos que facilitan el desarrollo de escenas tridimensionales, tales como los componentes `geometry`, `material` y `position`, entre otros. Estos componentes permiten configurar de manera sencilla las propiedades de los elementos en la escena, como su forma, aspecto y posición. Además de los componentes predefinidos, A-Frame permite la creación de componentes personalizados. Estos componentes personalizados te permiten añadir funcionalidades específicas y complejas que no están abarcadas por los componentes predefinidos. La creación de estos componentes es donde reside la parte más desafiante del desarrollo en A-Frame, ya que requiere un buen entendimiento de JavaScript y de la arquitectura de A-Frame.

Aframe, como la tecnología principal del TFG, no solo permitió la creación y gestión de entornos de realidad virtual, sino que también proporcionó un marco absoluto y flexible para integrar componentes 3D, manejar interacciones complejas y optimizar el rendimiento de escenas.

2.1.7. Networked A-frame

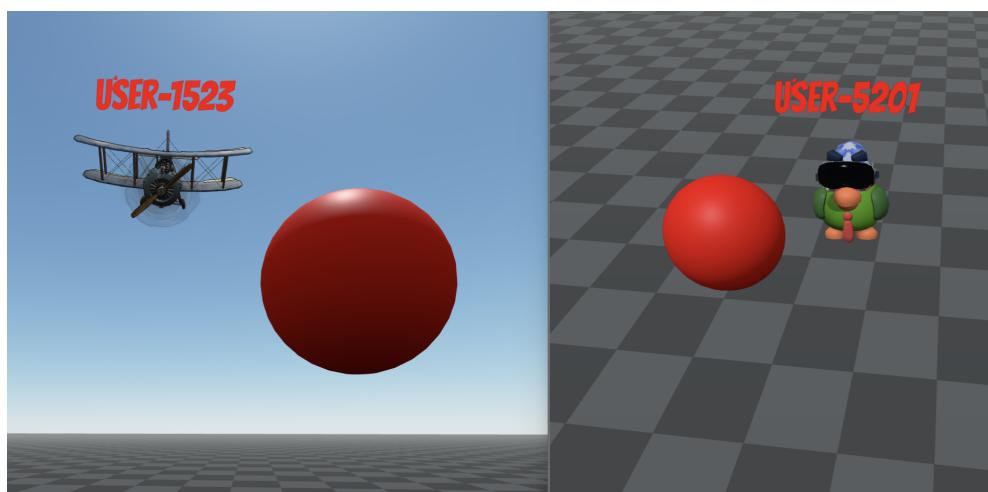


Figura 2.2: Escena multiusuario con Networked A-frame

Networked A-frame[6] (NAF) es un framework web para el desarrollo de escenas de realidad virtual multiusuario construido sobre A-Frame. Funciona sincronizando entidades y componentes con los usuarios conectados a un servidor local. NAF simplifica la implementación de la infraestructura de red, servidores y protocolos, permitiendo crear aplicaciones de realidad virtual para múltiples usuarios con herramientas ya conocidas.

Debido a la popularidad de A-frame y a la creación de escenas VR cada vez más complejas, surgió la necesidad de crear experiencias multiusuario. Esto requería que varios usuarios pudieran interactuar simultáneamente en el mismo entorno de realidad virtual, una capacidad que A-frame no ofrecía de forma nativa. Para solucionar esta limitación, se desarrolló Networked A-frame. Desde su inicio, la comunidad de desarrolladores ha mantenido y mejorado continuamente el framework. El código fuente está disponible en GitHub, permitiendo a los desarrolladores colaborar y contribuir al proyecto. La comunidad que rodea a A-frame y Networked A-frame sigue expandiéndose, impulsando innovaciones y nuevas características en aplicaciones como juegos multijugador, herramientas educativas colaborativas, simulaciones y programas de entretenimiento.

A continuación se muestra el código necesario para crear una escena compartida con una entidad de una esfera 3D:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://aframe.io/releases/1.6.0/aframe.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.5.0
      socket.io.slim.js"></script>
    <script src="/easyrtc/easyrtc.js"></script>
    <script src="https://unpkg.com/networked-aframe@^0.12.0/dist/networked
      -aframe.min.js"></script>
  </head>
  <body>
    <a-scene networked-scene="room:sala1;adapter:wseasyrtc;audio:false">
      <a-assets>
        <template id="esfera-template">
          <a-sphere position="0 4 0" color="red"></a-sphere>
        </template>
      </a-assets>
```

```

<a-entity id="esferaCompartida" networked="template:#esfera-template">
</a-entity>
</a-scene>
</body>
</html>

```

Para conectarse a una sala compartida se debe agregar el componente networked-scene al elemento a-scene, donde podemos configurar propiedades como el nombre de la sala, el tipo de adaptador de conexión y la transmisión de audio o video, entre otros. Las entidades instanciadas con el componente networked serán una copia compartida del elemento template agregado, visible para todos los usuarios conectados a la sala. Por defecto, los componentes de posición y rotación se sincronizan automáticamente, aunque para otros componentes ya predefinidos o creados se debe definir un esquema por elemento plantilla. NAF optimiza el uso del ancho de banda al enviar actualizaciones de red únicamente cuando hay cambios, y es compatible con todos los navegadores modernos tanto en dispositivos de escritorio como móviles.

Networked A-Frame permitió al proyecto crear experiencias de realidad virtual multijugador en tiempo real, al sincronizar interacciones y estados entre varios usuarios, facilitando así el desarrollo de entornos compartidos y dinámicos.

2.1.8. BabiaXR

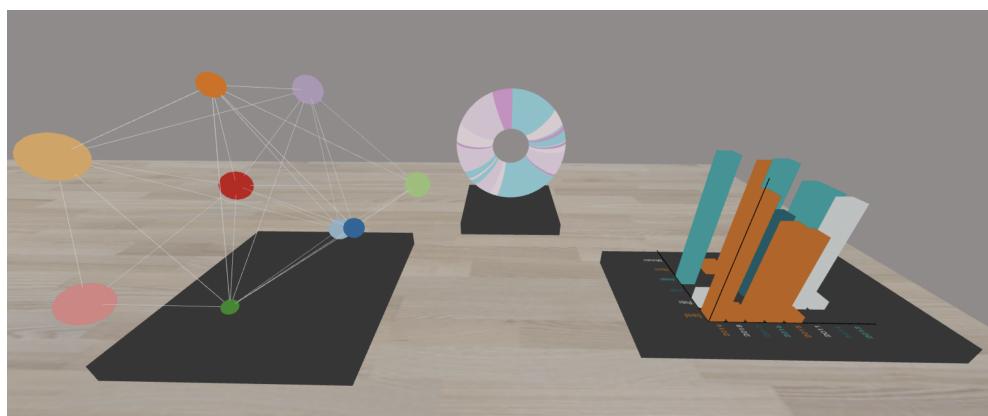


Figura 2.3: Visualización de datos con componentes de BabiaXR

BabiaXR[5] es una librería escrita en JavaScript de visualización de datos en realidad virtual diseñada para su uso en navegadores web. Se basa en el framework de A-Frame para propor-

cionar componentes y herramientas especializadas para visualizar datos complejos de manera inmersiva en entornos VR.

BabiaXR es de código abierto y ofrece documentación sobre sus componentes y tutoriales para crear escenas básicas. Esta librería fue desarrollada por el Dr. Jesús María González Barahona, profesor de la universidad Rey Juan Carlos. BabiaXR está formado por componentes de datos como babia-querys, que extraen datos desde archivos JSON, y babia-filters, que trabajan junto con babia-querys para seleccionar y filtrar partes específicas de esos datos según una clave o filtro definido. Además, existen componentes de visualización, que incluyen múltiples gráficos para representar los datos de entrada extraídos, como los gráficos babia-pie, babia-donut, babia-bars, entre otros. Estos componentes cuentan con varias propiedades que permiten seleccionar datos específicos y modificar la apariencia de las gráficas.

A continuación se muestra el código necesario para crear una gráfica de pastel:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
    <script src="https://unpkg.com/aframe-babia-components/dist/aframe-
      babia-components.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-entity id="data" babia-queryjson="url:./data.json"></a-entity>
      <a-entity babia-pie="from:data; key:model; size:sales;
        legend:true; animation:true; palette:blues">
      </a-entity>
    </a-scene>
  </body>
</html>
```

Para construir una escena con una gráfica de pastel básica, primero es necesario importar las bibliotecas de A-Frame y BabiaXR. Dentro de las etiquetas a-scene, debemos definir una entidad con un componente que actúe como fuente de datos, como babia-queryjson. Este componente se encarga de extraer los datos de un archivo JSON, cuya ubicación se especifica en la propiedad url. Una vez que los datos han sido extraídos, se asignan a la entidad por medio de

un nombre identificador. Para visualizar la gráfica de pastel, se debe implementar otra entidad que utilice el componente babia-pie. Este componente recoge los datos del babia-queryjson a través de la propiedad from, donde se especifica el identificador de la entidad que contiene los datos. Además, se necesita dar valor a las propiedades de key y size con las claves del archivo JSON, para determinar el nombre y el tamaño de cada segmento del pastel. Existen propiedades adicionales, como animation, legend y palette, entre otras, que permiten personalizar el comportamiento, la apariencia y la funcionalidad de las diferentes gráficas.

BabiaXR, la tecnología central del TFG, sirvió como base para todo el desarrollo y representó el objetivo principal del proyecto, crear y configurar visualizaciones inmersivas de datos en realidad virtual.

2.2. Tecnologías auxiliares

En esta sección se describen las herramientas complementarias que, aunque no intervienen de manera directa en el desarrollo del proyecto, han jugado un papel crucial en garantizar su correcta gestión, organización, documentación y depuración del código, facilitando el flujo de trabajo y optimizando el desarrollo.

2.2.1. GitHub

GitHub es una plataforma que ofrece un servicio de alojamiento de repositorios Git basado en la nube. Git es un sistema de control de versiones de código abierto creado por Linus Torvalds en 2005. Este sistema ayuda a los desarrolladores a tener disponible en el ordenador todo el historial del código desarrollado. Los usuarios de GitHub pueden crear cuentas, subir archivos y crear proyectos de codificación. GitHub centraliza todo el código y la documentación, lo que facilita el acceso y contribución de cualquier desarrollador en un proyecto. Además, detecta los errores antes de enviar el código, facilita el seguimiento de los cambios y retorna a versiones anteriores de un proyecto, lo que optimiza el flujo de trabajo de desarrollo.

Esta plataforma ha proporcionado al código del proyecto una estructura organizada, accesible, eficiente y segura, que ha sido fundamental para el desarrollo del trabajo.

2.2.2. Visual Studio Code

Visual Studio Code (VS Code) es un editor de código abierto y potente que está disponible para Windows, macOS, Linux, Raspberry Pi OS y versión web. Viene con soporte integrado para JavaScript, TypeScript y Node.js, aunque con su rico ecosistema de extensiones se puede codificar en cualquier lenguaje de programación.

VS Code cuenta con IntelliSense, una herramienta que ofrece autocompletado, resaltado de sintaxis y edición de código. Estas funcionalidades agilizan y facilitan la escritura de código. La depuración en Visual Studio Code también ayuda a los desarrolladores a detectar errores en el código, establecer puntos de interrupción, ver variables en tiempo real y ejecutar comandos en una consola integrada. Además, viene integrado con control de versiones de Git, lo que permite confirmar, extraer y enviar los cambios de su código a un repositorio Git remoto.

El código del proyecto ha sido desarrollado en Visual Studio Code, que ha demostrado ser una herramienta invaluable debido a su adaptabilidad con JavaScript. Además, sus múltiples herramientas y extensiones han optimizado el flujo de trabajo y mejorado la eficiencia de la codificación.

2.2.3. Meta Quest 2 y 3

Meta Quest es un dispositivo de realidad virtual (RV) inalámbrico compuesto por unas gafas, micrófono y auriculares integrados en un solo dispositivo que se coloca en la cabeza del usuario. Además, integran controladores Touch para interactuar mediante gestos con las manos. Fue desarrollado por Reality Labs, una división de Meta Platforms, que lanzó a la venta en octubre de 2020 la primera versión de las gafas.

Para este proyecto se han utilizado las Meta Quest 2 y Meta Quest 3. Las Meta Quest 2 cuentan con un procesador Snapdragon XR2, 6 GB de RAM, pantallas LCD con una resolución por ojo de 1832×1920 y una frecuencia de actualización de hasta 120 Hz. En comparación, las Meta Quest 3, las gafas más recientes, ofrecen 8 GB de RAM, una resolución de pantalla mejorada de 2064×2208 por ojo para una mayor nitidez visual, y mejoras en el campo de visión (FOV), lo que aumenta la inmersión. Además, presentan un diseño más liviano y ergonómico para una experiencia de usuario más cómoda.

Las Meta Quest han desempeñado un papel crucial para realizar pruebas exhaustivas y re-

finamientos del código programado, contribuyendo significativamente a la experimentación, evolución y mejora continua del proyecto final.

2.2.4. LaTeX

LaTeX[8] es una herramienta para la creación de documentos con aspecto profesional. Este sistema se emplea principalmente para documentos científicos o técnicos de tamaño medio a grande, aunque es versátil y puede adaptarse a prácticamente cualquier tipo de publicación. Utiliza comandos LaTeX para expresar los resultados tipográficos deseados, convirtiendo estos comandos con el texto del documento en un archivo PDF profesional gracias a su software llamado motor TeX. Cuenta con capacidad para manejar matemáticas complejas, lo que lo convierte en una herramienta indispensable para las ciencias físicas. Además, LaTeX ofrece facilidades avanzadas para la inclusión de índices, listas de figuras, referencias y gestión de bibliografías, simplificando la organización y citación en documentos extensos.

Para la elaboración del trabajo fin de grado, he utilizado LaTeX debido a que permite centrarse únicamente en el contenido textual, mientras que el sistema se encarga del diseño y formato del documento. Esto no solo asegura un resultado estéticamente profesional, sino que también facilita la gestión de elementos complejos y la consistencia del estilo a lo largo del documento.

Capítulo 3

Sistema resultante

En este capítulo se presentará de forma minuciosa el resultado final del proyecto, que abarca el desarrollo de un sistema para la visualización de datos en un entorno multiusuario de realidad virtual. Se ofrecerá una descripción exhaustiva de las funcionalidades disponibles para los usuarios dentro de la escena. Asimismo, se explorará la arquitectura técnica del proyecto, desglosando la lógica y el propósito de cada uno de los componentes creados.

3.1. Descripción funcional

La aplicación está diseñada para generar una escena compleja de visualización de datos multiusuario, que puede incluir múltiples componentes, query, que extraen los datos, filtros, que permiten seleccionar y restringir la información, y gráficas, que presentan visualmente los datos.

Esta descripción tiene como objetivo crear una escena básica que incluya al menos un componente de cada tipo. Por ello, se explicará el proceso completo que el usuario debe seguir para llegar a esta escena final de visualización de datos. Además, para que la experiencia de visualización sea más eficiente e inmersiva, se detallará cómo se integran varios usuarios en la misma escena. Todas las interacciones realizadas por un usuario en la escena se sincronizan en tiempo real, lo que permite a los demás usuarios conectados observar las acciones de otros.

3.1.1. Inicio del sistema de visualización de datos

El primer paso para visualizar una escena es iniciar el servidor proporcionado por Networked A-Frame, lo que habilita el uso de escenas compartidas. Se puede acceder a la URL por medio de un navegador web o en realidad virtual, y cada acceso conectará un nuevo usuario a la escena compartida. La plataforma permite que se conecten tantos usuarios como deseen, facilitando una experiencia de colaboración en tiempo real. Los usuarios conectados se generarán de forma aleatoria en un área central de la escena, cada uno con un avatar 3D y un nombre de usuario único visible sobre su cuerpo. Estos avatares se inicializan con animaciones de movimiento en bucle, aunque se desactivan para mejorar el rendimiento de la escena. La movilidad del avatar está definida de tal manera que en la versión del navegador, los usuarios podrán desplazarse utilizando las teclas de flecha y mover la cámara con el ratón. En realidad virtual, podrán mover y rotar la cámara con los joysticks, así como inclinar la cabeza. El avatar contará con movilidad total, permitiendo desplazarse en cualquier dirección y volar libremente.



Figura 3.1: Escena con varios usuarios conectados.

Al acceder a la escena, se presentará primero un panel de instrucciones que detalla los controles permitidos para la generación e interacción del menú inicial de creación de un componente, tanto para los usuarios en realidad virtual como para aquellos que se conectan a través de la plataforma web. En el caso de los usuarios de realidad virtual, se utiliza los controles de las gafas Oculus. Para generar el menú inicial de creación, deberán presionar simultáneamente los botones B e Y. Además, se podrá interactuar con todos los menús del sistema utilizando el gatillo de ambos mandos. Cualquier otra configuración de botones no tendrá efecto, esta restric-

ción se implementa para profundizar en el funcionamiento de los mandos de Oculus. Por otro lado, los usuarios en el navegador web podrán generar el menú inicial de creación pulsando la tecla de espacio y podrán interactuar con él haciendo clic con el ratón. En ambas versiones, la interfaz se generará en el suelo, frente a la dirección en la que esté mirando el avatar, independientemente de la posición. Una vez que se han comprendido las instrucciones, los usuarios deben hacer clic en el ícono de verificación para cerrar el panel. Es importante destacar que este panel solo será visible para el usuario conectado, no se compartirá con los demás usuarios.

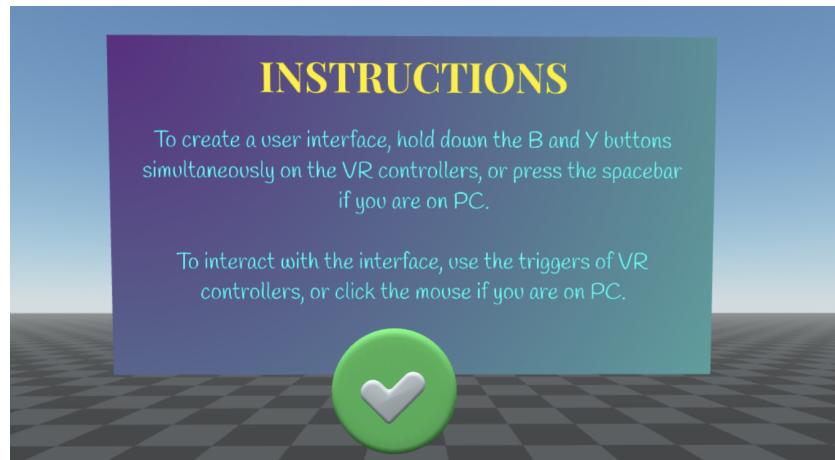


Figura 3.2: Escena con el panel de instrucciones.

El menú inicial de creación, generado al pulsar los controles mencionados anteriormente, puede generarse tantas veces como quiera. Este menú es el comienzo de la etapa de creación de tres tipos de componentes de BabiaXR, cada uno diseñado para un uso específico y con varios subtipos. Los componentes de tipo queries permiten recuperar datos de archivos JSON Y CSV, incluyendo babia-queryjson y babia-querycsv. Por otro lado, los filtros son responsables de filtrar los datos, con el componente babia-filter. Finalmente, los componentes de gráficas actúan como visualizadores de datos, como babia-pie y babia-doughnut. Todos estos componentes se pueden crear y configurar tantas veces como se deseé.

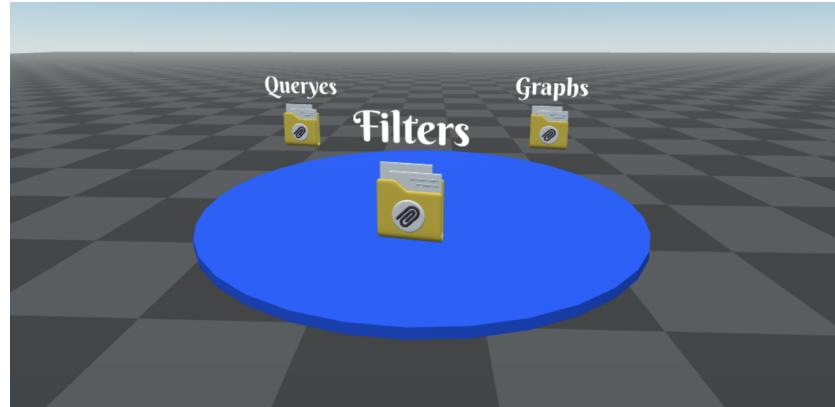


Figura 3.3: Escena con el menú inicial de creación.

3.1.2. Interfaz de usuario para componentes de tipo query

Al seleccionar el ícono para crear un componente de tipo query, aparece un nuevo menú interactivo, una plataforma en movimiento con varios íconos seleccionables. En esta vista, el usuario puede elegir el tipo de componente query que desea crear, babia-queryjson o babia-querycsv. Estos componentes están diseñados para recuperar datos de cualquier archivo JSON o CSV disponible en el equipo. Una vez elegido el tipo de componente para crear, este se inserta en la entidad del ícono seleccionado, aunque aún no está configurado por lo que sus propiedades se encuentran vacías. A través de una animación fluida, los menús utilizados desaparecen de la vista mientras el ícono seleccionado desciende hacia el suelo, agrandándose progresivamente. Este cambio visual indica que la etapa de creación del componente ha concluido, dejando listo el componente para su próxima etapa. En todos los menús interactivos, se presenta una mini plataforma central que muestra claramente el valor seleccionado del menú anterior, permitiendo al usuario visualizar en todo momento su elección previa.

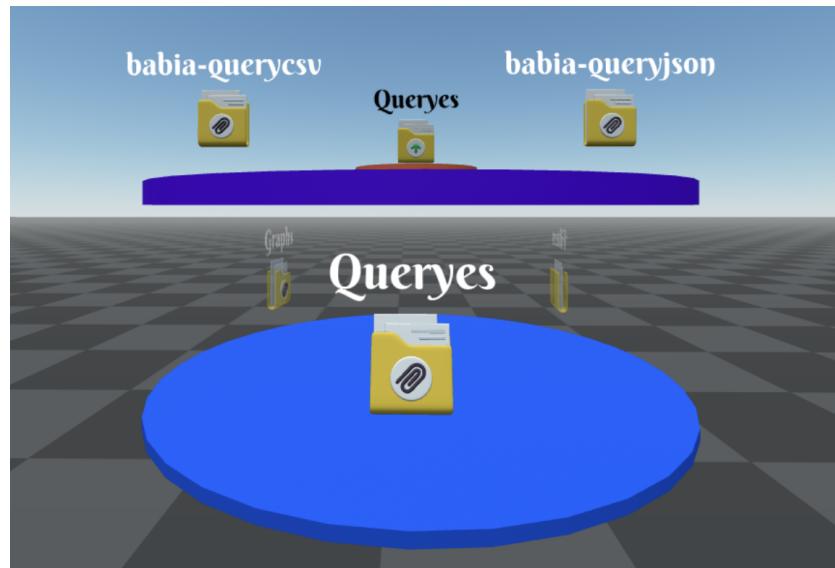


Figura 3.4: Escena con el menú de creación de tipo query.

Una vez creado el componente seleccionado, aparecerán tres menús interactivos alrededor del ícono seleccionado. Estos menús ofrecen opciones comunes para cualquier tipo de componente creado, ya sean queries, filtros o gráficas. Los menús de opciones pueden ser borrados o generados al hacer clic en la zona central del ícono del componente creado previamente. Los tres menús disponibles son:

- **Menú de Información:** Este menú despliega un panel informativo que proporciona detalles sobre el componente creado, como su descripción, qué usuario lo creo y diferentes variables importantes de configuración del componente. Inicialmente, las variables de información están vacías, dado que el componente aún no ha sido configurado. El panel se actualizará automáticamente a medida que se configuren las propiedades del componente. Además, si el avatar se mueve mientras el panel de información está abierto, el panel rotará para mantenerse siempre alineado con el campo de visión del usuario.
- **Menú de Eliminar:** Al seleccionar esta opción, se eliminará completamente la entidad del DOM, como si nunca hubiera sido creada. Esta acción es inmediata e irreversible, y puede modificar los resultados de otros componentes.
- **Menú de Configuración:** Este es el menú más importante, ya que permite asignar valores a las propiedades del componente creado. Dependiendo del tipo de componente, será posible configurar una mayor o menor cantidad de propiedades. Al hacer clic en cualquiera

de los valores configurables, el componente se actualizará de inmediato, reflejando los cambios en tiempo real. Todas las propiedades pueden ser modificadas en cualquier momento, y los usuarios pueden ajustar los valores tantas veces como deseen.

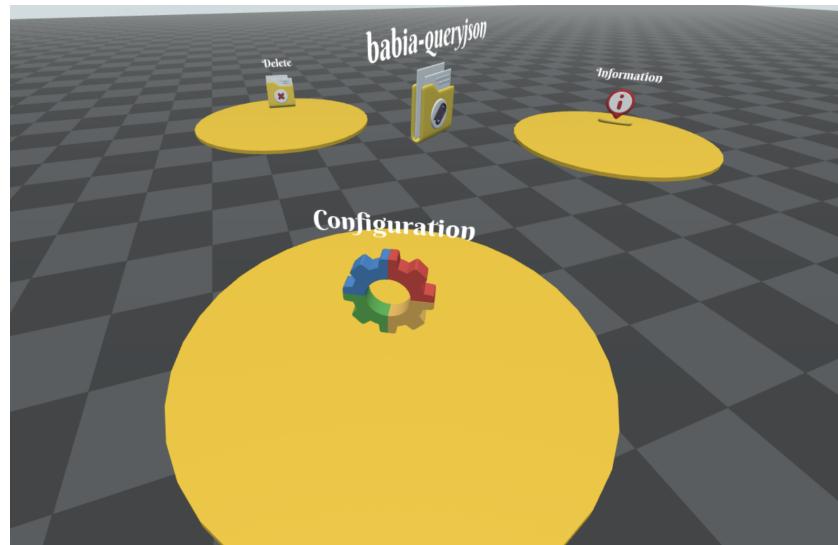


Figura 3.5: Escena de los menús de opciones para componentes creados.

En este caso, se configura un componente `babia-queryjson`. Al seleccionar el ícono de configuración, los menús de opciones restantes se borrarán, y se desplegará un nuevo menú en la parte superior del menú configuración. Este menú cuenta con una única propiedad configurable del tipo query, la propiedad URL. Esta propiedad requiere la ubicación del archivo desde el cual deseamos extraer los datos. Al hacer clic en el ícono de la propiedad URL, aparecerá un nuevo menú por encima que mostrará una lista de archivos JSON disponibles para extraer datos. Una vez seleccionado el archivo deseado, el componente se actualizará automáticamente con el nuevo valor. Esto también provocará una actualización inmediata en el panel de información y añadirá el título del archivo vinculado debajo del ícono del componente, facilitando su identificación visual. Una vez completada la configuración, es posible cerrar el menú de configuración y volver a abrir todos los menús de opciones haciendo clic en el ícono del componente.

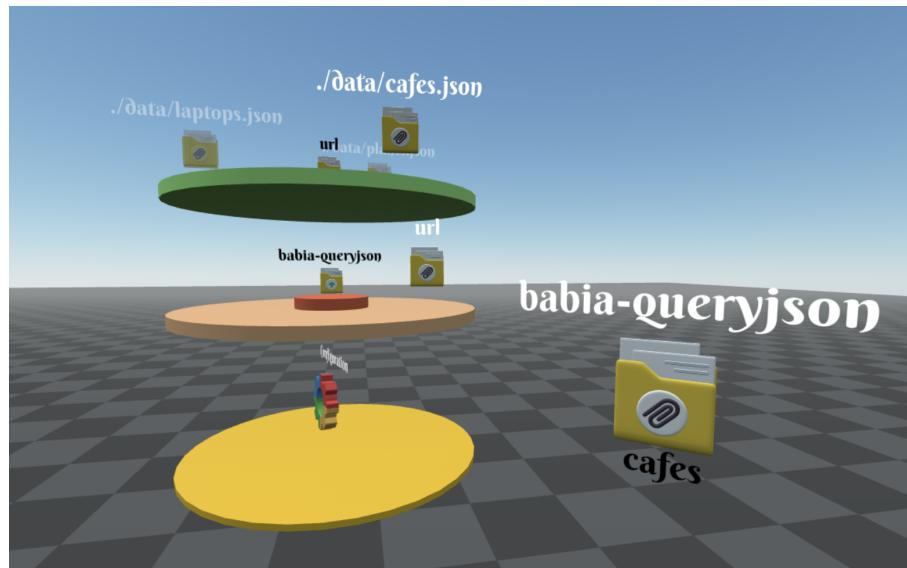


Figura 3.6: Escena de configuración de babia-queryjson.

Si seleccionamos ahora el ícono de Información, se desplegará un panel que muestra una descripción detallada de la función del componente babia-queryjson. Además, se mostrarán datos relevantes como las variables asociadas al creador del componente y la URL del archivo previamente seleccionado. Este panel proporciona una visión clara y completa de la configuración actual del componente, facilitando el acceso a la información clave de manera organizada.

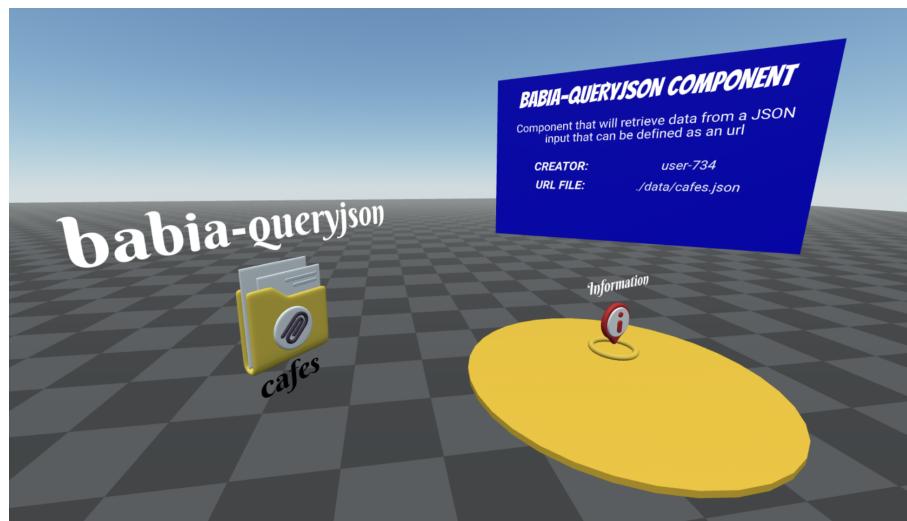


Figura 3.7: Escena de información de babia-queryjson.

La creación y configuración para el componente babia-querycsv sigue el mismo proceso que para babia-queryjson. Ambos comparten una única propiedad configurable, la URL. Sin embar-

go, en el caso de babia-querycsv, los resultados muestran una lista de archivos CSV disponibles en el equipo.



Figura 3.8: Escena de configuración de babia-querycsv.

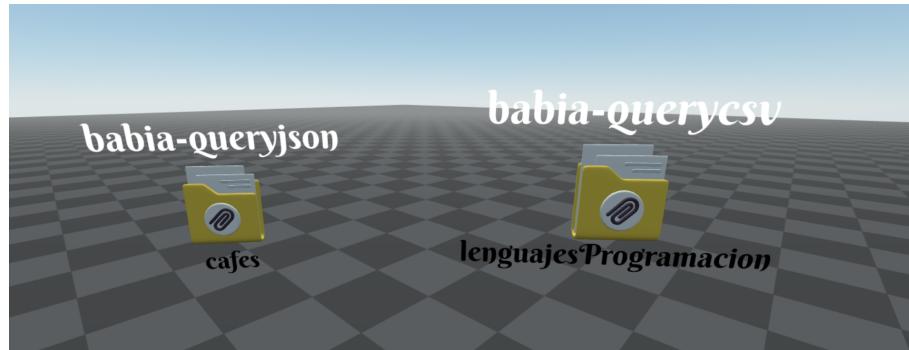


Figura 3.9: Resultado final: Integración de un babia-queryjson y un babia-querycsv a la escena.

3.1.3. Interfaz de usuario para componentes de tipo filtro

El componente de filtros depende directamente de los componentes de tipo query, por lo que es necesario haber creado previamente un babia-queryjson o babia-querycsv para poder generar un filtro, ya que sin datos no hay nada que filtrar. Así, al seleccionar el componente de filtros en el menú inicial de creación, se mostrará un menú con todos los componentes babia-queryjson y babia-querycsv disponibles en el DOM. Si no se ha creado ningún componente query con anterioridad, este menú aparecerá vacío.

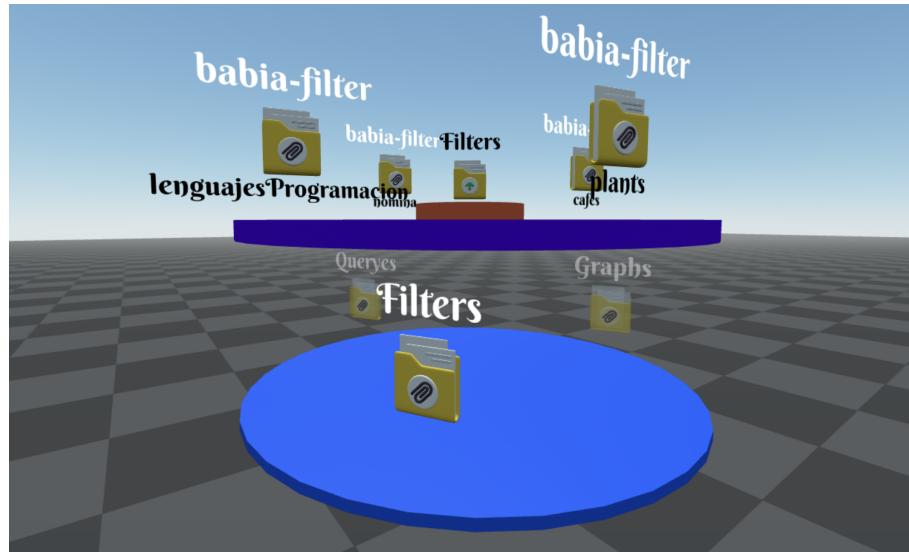


Figura 3.10: Escena con el menú de creación de tipo filtro.

Al seleccionar el componente query para filtrar, se creará un componente babia-filter vinculado a la query seleccionada. Una vez completada la etapa de creación, aparecerán los tres menús de opciones, comunes para todos los tipos de componentes. En la etapa de configuración, se desplegará un menú que mostrará todos los campos disponibles del archivo de datos seleccionado, permitiendo elegir el campo por el que se va a filtrar dicho archivo. Al seleccionar un campo, aparecerá un nuevo menú con todos los valores posibles de ese campo. Toda esta fase es posible gracias a un algoritmo propio que previamente analiza el archivo de datos seleccionado, procesando todos los campos de los objetos presentes en el archivo junto con sus posibles valores. Esto permite que los campos y sus resultados estén disponibles para la configuración y el filtrado posterior. Una vez acabada la etapa de configuración se actualizará el panel informativo con los nuevos valores y se añadirá el título del archivo junto con su filtro debajo del icono del componente.

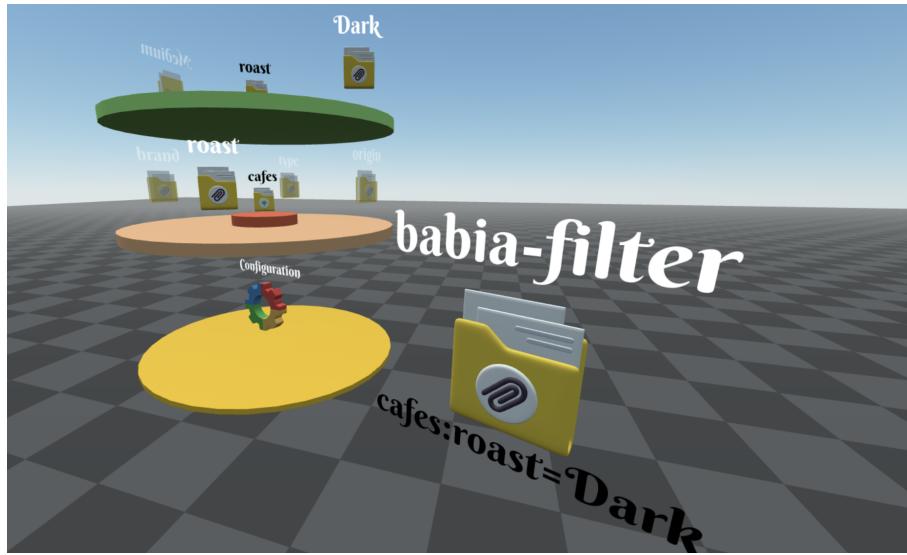


Figura 3.11: Escena de configuración de babia-filter.

El panel de información de los componentes de tipo filtro muestran una descripción del componente creado, el usuario que lo generó, y la URL del archivo de datos vinculados al queryjson o querycsv correspondiente. Además, incluye una nueva variable que refleja el filtro configurado para ese archivo. Esta variable indica el campo filtrado del archivo y el valor seleccionado. Por lo tanto, solo se mostrarán los objetos del archivo de datos que coincidan con ese campo y valor específicos. Cualquier cambio en la configuración del filtro se actualizará automáticamente en el panel de información.

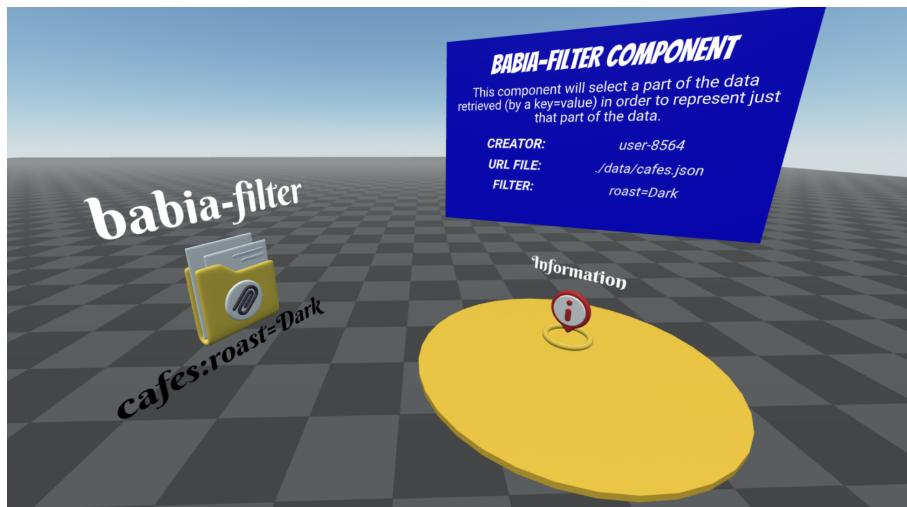


Figura 3.12: Escena de información de babia-filter.



Figura 3.13: Resultado final: Integración de un babia-filter a la escena.

3.1.4. Interfaz de usuario para componentes de tipo gráfica

Los componentes de gráficas son los más importantes, ya que visualizan los resultados de los demás componentes creados, mostrando los datos en gráficos tridimensionales. A través de estas gráficas, se pueden explorar los datos de cada sección individual. Estas gráficas dependen de los componentes queries y filtros, los cuales proporcionan los datos necesarios para generar las visualizaciones. En el menú inicial de creación, los componentes gráficos disponibles son babia-pie y babia-doughnut, que representan los datos en gráficos circulares.

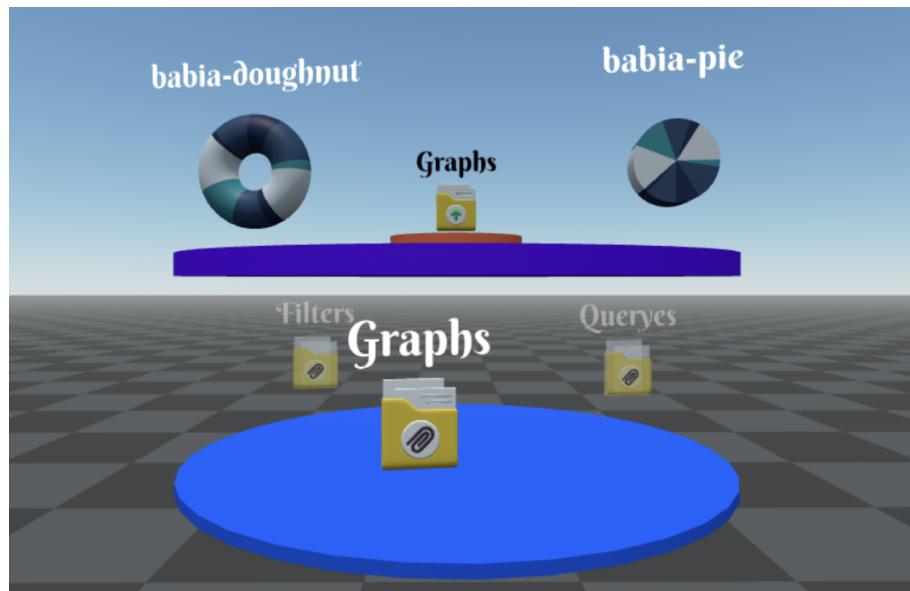


Figura 3.14: Escena con el menú de creación de tipo gráfica.

En la etapa de configuración de los componentes de gráficas, se disponen de cuatro propiedades clave. La primera Legend, que permite activar o desactivar la visualización de los valores de las secciones de la gráfica cuando el ratón o el controlador VR pasa sobre ellas. Animation,

que habilita o desactiva la animación de la gráfica cuando los valores se actualizan. Palette, que ofrece distintas opciones de colores para personalizar las gráficas. Y From, la propiedad más importante, que permite seleccionar qué datos se visualizarán en las gráficas.

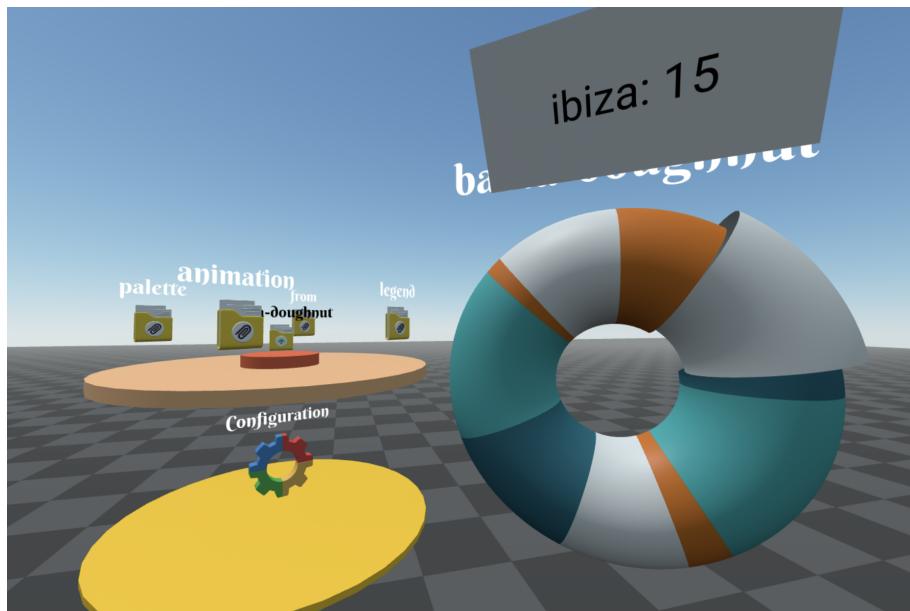


Figura 3.15: Escena de configuración de babia-doughnut.

Al seleccionar la propiedad From, se generará un nuevo menú con todos los archivos babia-queryjson, babia-querycsv y babia-filter previamente creados por cualquier usuario conectado. Después de elegir el archivo de donde se extraerán los datos, aparecerá un nuevo menú con dos subpropiedades, Key y Size. Al seleccionar Key, se mostrará un menú con todos los campos que contengan valores de tipo string, ya que estos definirán cada porción del gráfico circular. Por otro lado, al seleccionar Size, se desplegará un menú con los campos que contengan valores numéricos, los cuales determinarán el tamaño de las porciones del gráfico. Ambas subpropiedades son obligatorias para que la gráfica se actualice correctamente con los datos seleccionados. Se puede elegir primero Key y luego Size, o viceversa, pero ambos deben tener valores asignados. Todas las propiedades de las gráficas pueden configurarse cuantas veces sea necesario, ya sea de manera conjunta o por separado.

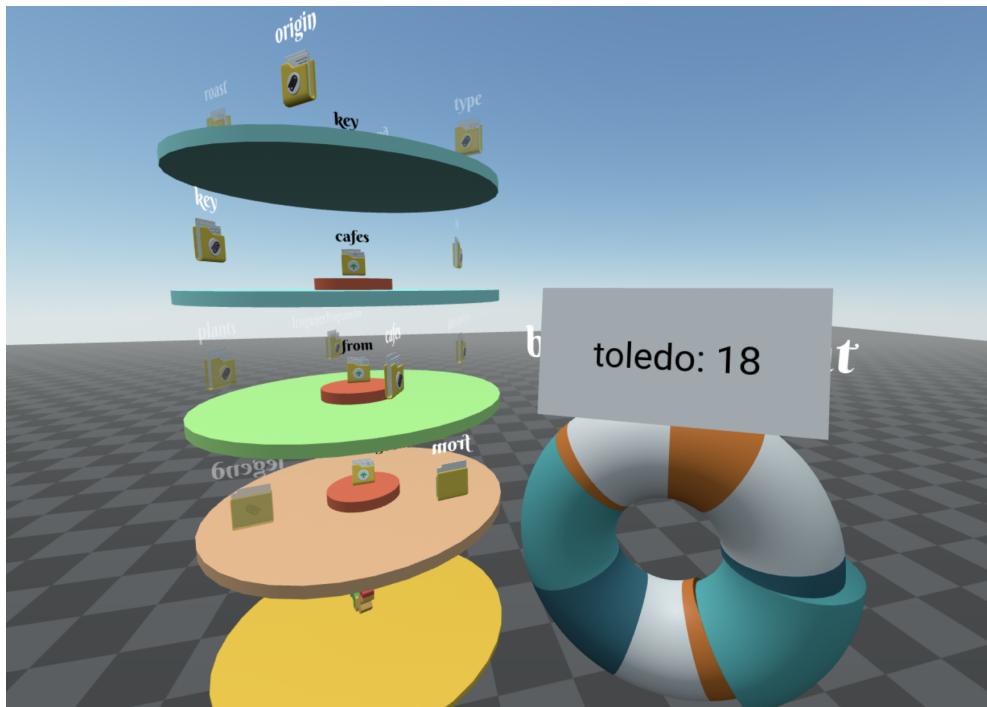


Figura 3.16: Escena de configuración de la propiedad From de babia-doughnut.

El panel de información proporciona una descripción detallada del componente creado, incluyendo el nombre del usuario que lo generó, la URL de la ubicación del archivo de donde se obtendrán los datos, y el filtro aplicado en caso de que se haya vinculado a un babia-filter la gráfica. Además, se muestra un nuevo atributo denominado KEY/SIZE, donde se reflejan los valores seleccionados durante la etapa de configuración. Esto facilita la interpretación de los datos al visualizar las gráficas, ayudando a identificar claramente qué representan los distintos segmentos.



Figura 3.17: Escena de información babia-doughnut.

Para el componente babia-pie, el proceso de configuración es el mismo que para babia-doughnut, con cuatro propiedades disponibles para ajustar la configuración de la gráfica. El panel de información también mostrará los mismos atributos que en el caso de babia-doughnut. Es importante destacar que todos los componentes creados son visibles para todos los usuarios conectados a la escena. Sin embargo, lo único que no se sincroniza es la interacción cuando un usuario pasa el ratón o el controlador sobre la gráfica para ver un dato específico. Esto se ha diseñado así para que cada usuario pueda explorar los datos de manera individual según sus preferencias.

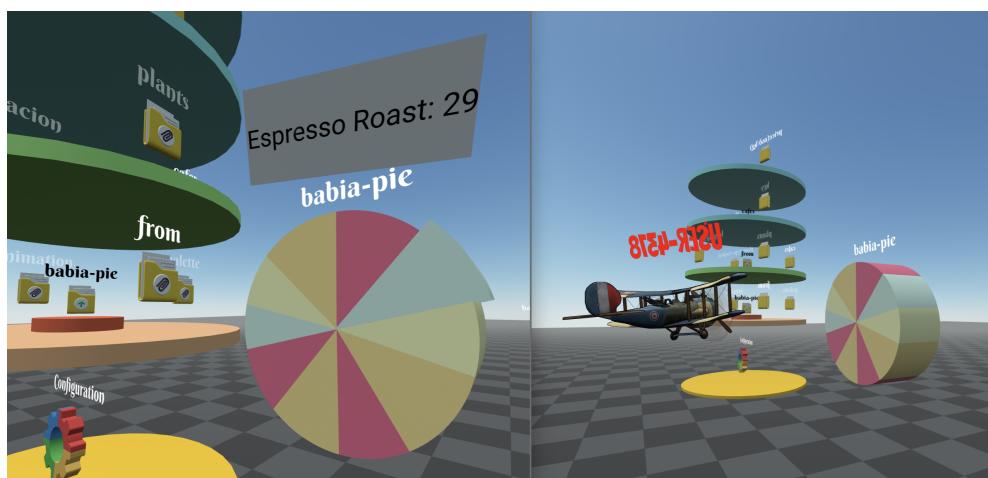


Figura 3.18: Escena compartida de configuración de babia-pie.

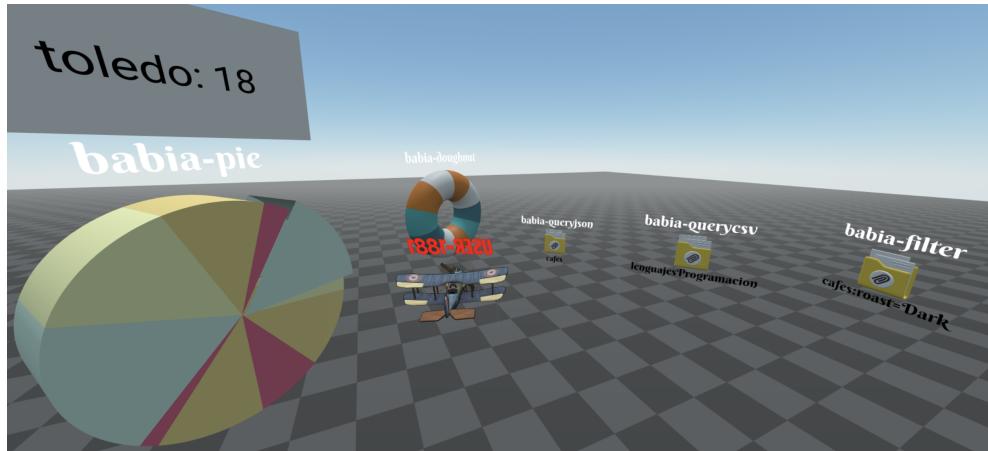


Figura 3.19: Resultado final: Integración de un babia-doughnut y un babia-pie a la escena.

3.2. Arquitectura técnica

En esta sección se detallará todo lo relacionado con la arquitectura técnica utilizada para la creación de la escena final del proyecto. Se realizará un recorrido por la estructura de archivos y carpetas. Además, se explicará la lógica detrás de los componentes desarrollados, desglosando cómo cada uno contribuye al correcto funcionamiento del sistema resultante y a la experiencia colaborativa en la escena. Finalmente, se abordará la implementación del archivo HTML, que constituye la base estructural de la aplicación.

3.2.1. Implementación de la escena

En esta sección, se va a detallar la implementación de la escena 3D interactiva del proyecto. En primer lugar, se explicará la estructura de carpetas y archivos del proyecto, describiendo el contenido y propósito de cada una. A continuación, se profundizará en el funcionamiento de A-Frame, un framework que permite renderizar escenas 3D directamente a través de un archivo HTML. Finalmente, se analizará el archivo HTML del proyecto, explicando el diseño y la función de toda la estructura.

Estructura del proyecto

El proyecto ha sido desarrollado utilizando el editor Visual Studio Code, con una arquitectura organizada de forma ordenada y clara. La carpeta principal del proyecto se llama TFG, y

en su interior encontramos varias subcarpetas y un único archivo HTML.

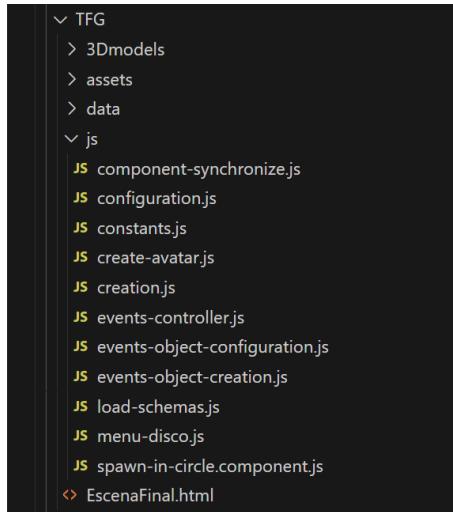


Figura 3.20: Estructura de los archivos del proyecto.

Las subcarpetas son:

- **3Dmodels:** Aquí se almacenan los modelos 3D en formato .glb, incluidos los iconos y avatares que aparecen en la escena.
- **assets:** Contiene imágenes de texturas para los paneles de información, además del archivo favicon.ico, el pequeño ícono que se muestra en las pestañas del navegador.
- **data:** Esta carpeta almacena archivos de datos en formato JSON y CSV, de los cuales se extraen los datos para la visualización de las gráficas.
- **js:** Se encuentran todos los componentes JavaScript creados específicamente para el funcionamiento de la escena final.

Por último, tenemos el archivo HTML, que es el encargado de definir la escena 3D. Este HTML incluye la estructura de A-Frame para cargar y gestionar los elementos en el entorno de realidad virtual. Aquí es donde se integran los componentes creados y se configuran los elementos visuales y funcionales.

El archivo HTML de la escena

A-Frame facilita la creación de escenas 3D al permitir definir tanto la escena como los objetos directamente en un archivo HTML, dejando que el framework gestione la complejidad

técnica de la renderización y el control de la escena. Para que A-Frame funcione correctamente en conjunto con las librerías utilizadas en este proyecto, es necesario incluir todas las bibliotecas necesarias dentro de la etiqueta head del HTML, agregando los scripts correspondientes. Además, para garantizar que todos los componentes creados que hemos desarrollado para el proyecto sean visibles en la escena, también se deben importar especificando la ubicación de los archivos JavaScript donde están definidos.

Definición de las librerías del proyecto:

```
<!-- Libreria de Aframe 1.5.0. sin problema del grabbable -->
<script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
<script>
  delete AFRAME.components["grabbable"];
</script>
<!-- Libreria de Networked Aframe 0.12.0-->
<script src="https://unpkg.com/networked-aframe@^0.12.0/dist/networked
-aframe.min.js"></script>
<!-- Libreria de Socket 2.5.0 -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.5.0/
socket.io.slim.js"></script>
<!-- Libreria de EasyRTC -->
<script src="/easyrtc/easyrtc.js"></script>
<!-- Libreria de BabiaXR -->
<script src="https://unpkg.com/aframe-babia-components/dist/
aframe-babia-components.min.js"></script>
<!-- libreria Aframe-extra -->
<script src="https://cdn.jsdelivr.net/gh/donmccurdy/aframe-extras@v7.2.0/
dist/aframe-extras.min.js"></script>
//MAS LIBRERIAS
...
...
```

A-Frame amplía el uso de HTML mediante un sistema basado en entidades y componentes, lo que permite definir elementos 3D utilizando etiquetas como a-scene, a-entity, y otros componentes que permiten agregar geometrías, cámaras, entre otros. El archivo HTML en A-Frame incluye una etiqueta principal a-scene, que actúa como el contenedor de la escena 3D, inicializando el motor gráfico y gestionando la renderización de todos los objetos 3D. Dentro de esta etiqueta, se añaden las distintas entidades que compondrán la escena, y a cada una se le pueden asignar componentes que definen su comportamiento o propiedades. Estos componentes pueden ser predefinidos como position o rotation, o personalizables para una lógica más compleja.

En el proyecto, dentro de la etiqueta a-scene utilizamos el componente environment pa-

ra crear un entorno predefinido y el componente networked-scene, fundamental para habilitar la colaboración multiusuario. Este componente permite especificar el nombre de la sala y el adaptador de red, que en nuestro caso es easyrtc, lo cual habilita la comunicación de audio en tiempo real entre los usuarios conectados. En nuestra escena, solo existen dos entidades iniciales, una con el componente create-avatar, que gestiona la apariencia y los controles del usuario conectado, y otra con el componente events-controller, que restringe los controles disponibles, tanto para los navegadores web como para los controladores de realidad virtual. Estos dos únicos componentes se activan automáticamente cuando se carga el HTML. El componente create-avatar habilita los controles, mientras que el componente events-controller, al ejecutarse un evento con los botones específicos de los controles, añade una entidad con el componente creation al HTML. A partir de ahí, creation puede inicializar y gestionar todos los demás componentes, ya sea directamente o a través de los nuevos componentes activados.

Definición de la etiqueta a-scene:

```
<a-scene environment="preset: default; playArea: 3" networked-
scene="room: user-interface; debug: true; adapter: easyrtc;">
  <a-entity create-avatar></a-entity>
  <a-entity id="entityEventsController" events-controller ></a-entity>
</a-scene>
```

Los componentes creados en el proyecto tienen la capacidad de generar nuevas entidades, personalizarlas y agregarles varios componentes que definen su comportamiento. Sin embargo, para que estas entidades sean visibles en la escena, es fundamental que sean añadidas como hijas dentro de la etiqueta `<a-scene>`. Esto se logra utilizando el método `appendChild`, que inserta dinámicamente estas entidades en la escena 3D. Una vez que la escena ha sido configurada con todas sus entidades y componentes, la visualización final se realiza accediendo a un navegador web. Para ello, simplemente se introduce la URL del servidor seguida del nombre del archivo HTML, que contiene la definición de la escena. Esto cargará la escena completa en el navegador, permitiendo la interacción en tiempo real, tanto para el usuario local como para los demás usuarios conectados.

3.2.2. Componentes principales

Estos componentes son los encargados de gestionar la lógica central del sistema resultante, actuando como el eje fundamental en el proceso de creación y configuración de cualquier tipo de componente de BabiaXR. Están diseñados de manera general y flexible, permitiendo que puedan adaptarse a múltiples casos de uso y asegurar que la creación y personalización de los elementos sea sencilla y eficiente. Además, cada entidad creada mediante estos procesos incluye el componente de Networked A-Frame, lo que garantiza que todas las acciones y cambios realizados sean visibles y sincronizados en tiempo real para todos los usuarios conectados a la escena compartida.

Creation

El componente Creation es el más crucial de todos, ya que contiene una de las lógicas más compleja del sistema. Su función principal es gestionar la creación de componentes BabiaXR y orquestar el flujo de trabajo de los menús de opciones como el menú de información y eliminación del componente.

Primero, es importante explicar la estructura lógica que se ha implementado. El componente Creation se añade a la entidad padre que va a contener todos los menús. En este contexto, llamamos menú a cada plataforma circular que gira sobre su eje y contiene objetos seleccionables en su superficie. El componente Creation se encarga de invocar a otros componentes principales para generar los discos y los objetos correspondientes a cada menú. Cada disco y los objetos que contiene estarán asociados a un único tipo, por lo que cada menú representará un tipo específico de elementos. Al seleccionar un objeto dentro de un menú, se activará nuevamente el componente Creation, enviando dos de sus propiedades clave, typeObjectSelected y valueObjectSelected. La propiedad typeObjectSelected tendrá valores restringidos, como MainOpcion o TypeCreation, mientras que valueObjectSelected contendrá el nombre del objeto seleccionado. Una vez que se reciban estos datos en Creation, se seguirá un flujo de decisiones diferente en función del tipo de objeto seleccionado. Esta estructura se ha diseñado así para permitir la creación de componentes BabiaXR de cualquier tipo utilizando un único componente en solo una entidad.

Definición del componente Creation:

```
AFRAME.registerComponent('creation', {
  schema: {
    typeObjectSelected: { type: 'string', oneOf: [CONSTANTS.MAINOPCION,
      CONSTANTS.TYPECREATION] },
    valueObjectSelected: { type: 'string', default: '' },
  },
  init: function () {
    this.initializeParameters();
    createNewMenuDisco(this, Object.keys(this.dashboard), CONSTANTS.MAINOPCION,
      '#0061FF', false);
  },
});
```

El componente Creation va a generar entidades con el componente menu-disco, encargado de crear un disco con objetos de tipo MainOpcion o TypeCreation. Estos objetos incluyen el componente Events-object-creation, el cual maneja la lógica de interacción al hacer clic sobre ellos. Este clic invoca nuevamente al componente Creation con nuevos valores de sus propiedades. Además, al crear un menú, también se genera un minidisco que contiene el valor seleccionado en el menú anterior.

Creación de la entidad Menú y Minidisco:

```
let entityMenuDisco = document.createElement('a-entity');
entityMenuDisco.setAttribute('networked', 'template:#platoInit-template');
entityMenuDisco.setAttribute('id', 'Menu-' + objectType);
entityMenuDisco.setAttribute('position', { x: 0, y: 3 * self.numeroDiscosCreados, z: 0 });
entityMenuDisco.setAttribute('menu-disco', {
  'objectsStage': objects,
  'objectType': objectType,
  'colorDisco': colorDisco
});

if (miniDisco) {
  let entityMiniDisco = createMiniDisco(self.valuesSelectded[CONSTANTS.MAINOPCION]);
  entityMenuDisco.appendChild(entityMiniDisco);
}
```

El primer menú generado es de tipo MainOpcion que presenta objetos con valores como queries, filtros y gráficas. Según la opción seleccionada, se genera un segundo menú de tipo TypeCreation con objetos de los componentes de BabiaXR. Todos los nuevos valores que llegan al componente Creation se almacenan en una variable global dentro del componente, permitiendo su uso posterior para seguir un flujo lógico coherente y organizado. Esto facilita la gestión de

datos y asegura que las decisiones tomadas durante el proceso de creación se basen en la información previamente recibida.

Definición del método Update del componente Creation:

```
update: function () {
    this.valuesSelectded[this.data.typeObjectSelected] = this.data.valueObjectSelected;

    if (this.valuesSelectded[CONSTANTS.MAINOPCION] === CONSTANTS.QUERYES) {
        this.executeMenuQueryes();
    } else if (this.valuesSelectded[CONSTANTS.MAINOPCION] === CONSTANTS.GRAPHS) {
        this.executeMenuGraphs();
    } else if (this.valuesSelectded[CONSTANTS.MAINOPCION] === CONSTANTS.FILTERS) {
        this.executeMenuFilters();
    }
},
}
```

Cuando se selecciona un objeto del menú TypeCreation, se crea el componente correspondiente y se ejecuta el método executeProcesoCreacion. Este método incluye animaciones para eliminar los menús previos, expandir el ícono del objeto creado, y activar los menús de opciones de información, borrado y configuración. Estos menús pueden ser eliminados o añadidos otra vez haciendo clic en el ícono del componente creado.

Método para la creación de los menús opciones:

```
addMenusOption: function () {
    const radius = 8.5;
    const angles = [120, 240, 360];

    angles.forEach(angle => {
        const rad = angle * (Math.PI / 180);
        const posX = radius * Math.cos(rad);
        const posZ = radius * Math.sin(rad);

        if (angle === 120) {
            this.createMenuConfiguration(posX, posZ);
        } else if (angle === 240) {
            this.createMenuDelete(posX, posZ);
        } else if (angle === 360) {
            this.createMenuInformation(posX, posZ);
        }
    });
}
```

La lógica de los menús de información y borrado está contenida dentro del propio componente Creation, ya que su lógica es más simple y no requiere un componente independiente. Sin embargo, para el menú de configuración, sí se ha desarrollado un componente específico. Al seleccionar el objeto del menú de borrado, se identifica la entidad correspondiente y se elimina de la escena. Por otro lado, al seleccionar el menú de información, se despliega un panel con detalles específicos según el tipo de TypeCreation seleccionado. En términos generales, este componente gestiona toda la lógica inicial para la creación de entidades y componentes, asegurando que estén listos para su posterior configuración. Es el núcleo del proceso de creación, estableciendo las bases para que cada componente funcione correctamente.

Menu-disco

Este componente es invocado repetidamente por los componentes Creation y Configuration para crear un nuevo menú en forma de un disco con objetos en su superficie. La entidad que tenga este componente se encargará de personalizar una forma geométrica de disco, aplicando colores distintos, aumentando la altura cada vez que se cree un nuevo menú y permitiendo que gire sobre su propio eje. Además, se inicializa el proceso de creación de las entidades hijas correspondientes a los objetos que se encuentran sobre la superficie del disco, los cuales serán seleccionables.

Definición del componente Menu-disco:

```
AFRAME.registerComponent('menu-disco', {
  schema: {
    colorDisco: { type: 'string', default: '' },
    objectsStage: { type: 'array', default: [] },
    objectType: { type: 'string', default: '' },
  },
  init: function () {
    this.customizeDiscoStage();
    this.createObjectsStage();
  },
})
```

Las entidades de los objetos para los valores de los componentes de tipo query y filtro van a tener un componente llamado Events-object-creation. Este componente será responsable de gestionar la lógica y personalizar cada objeto seleccionable. En cuanto a los componentes de

tipo gráfico, será necesario crear entidades con los componentes babia-pie y babia-doughnut, utilizando una configuración y datos por defecto. Esto se hace porque, en este punto, solo nos interesa mostrar el ícono de los gráficos, sin procesar datos reales.

Método para la creación de entidades de objetos:

```
createObjectsStage: function (nameObjectGLB) {
    let self = this;
    let radius = 3.2;
    let posicionNueva = { x: 0, y: 0, z: 0 };
    let complementsGraph = [CONSTANTS.BABIAPIE, CONSTANTS.BABIADOUGHNUT];

    this.data.objectsStage.forEach(function (object, index) {
        let angle = (Math.PI * 2 / self.data.objectsStage.length) * index;
        posicionNueva = { x: radius * Math.cos(angle), y: 0, z: radius * Math.sin(angle) };
        let entityObject;
        if (complementsGraph.includes(object)) {
            entityObject = createObjectGraph(object, posicionNueva);
        } else {
            entityObject = createObject(self, object, nameObjectGLB, posicionNueva);
        }
        self.el.appendChild(entityObject);
    })
},
```

Events-object-creation

Cada objeto de los menús, durante su etapa de creación, contiene este componente. El componente Menu-disco es el encargo de activar el componente Events-object-creation con los valores de sus propiedades, como el nombre del objeto, el tipo al que pertenece y el nombre del modelo 3D que se le adjudica. La función del componente es personalizar el objeto, asignándole un ícono en forma de modelo 3D, ajustando su escala y rotación, y añadiendo un título para identificarlo. Lo más importante de este componente es que gestiona el evento de clic de la entidad del objeto. Cuando se selecciona un objeto del menú, este componente contiene la lógica de qué hacer tanto con el objeto seleccionado como con los demás objetos del mismo menú.

Al hacer clic en un objeto, su valor se guarda, y los demás objetos del menú cambian con una opacidad reducida, indicando que no fueron seleccionados. Finalmente, el objeto seleccionado invoca al componente Creation mediante el método setAttribute, y se le asignan las propiedades del valor del objeto y el tipo de menú en el que se encuentra.

Método para personalizar el objeto:

```
//Añadir un ícono 3D al objeto
this.el.setAttribute('gltf-model', '3Dmodels/' + this.data.nameObjectGLB
+ '.glb');
this.el.setAttribute('scale', '0.2 0.2 0.2');
this.el.setAttribute('animation', {
  'property': 'rotation',
  'to': '0 360 0',
  'dur': '15000',
  'easing': 'linear',
  'loop': 'true'
});
//Titulo del objeto
let entityObjectChildren = document.createElement('a-entity');
entityObjectChildren.setAttribute('networked', 'template:#textInit-template');
entityObjectChildren.classList.add("topNameObject");
entityObjectChildren.setAttribute('text', {
  'value': this.data.objectStage,
  'align': 'center',
  'side': 'double',
  'color': 'WHITE',
  'shader': 'msdf',
  'font': 'https://raw.githubusercontent.com/etiennepinchon/
aframe-fonts/master/fonts/berkshireswash/BerkshireSwash
-Regular.json'
});

```

Configuration

Este componente es uno de los más importantes y contiene una lógica más compleja que el resto. Después de completar la etapa de creación del componente, se despliegan tres menús de opciones, entre los cuales se encuentra el menú de configuración. Al seleccionar el ícono de configuración, se invoca el componente Configuration, que recibe dos propiedades, typeObjectSelected y valueObjectSelected, las cuales representan el tipo de menú en el que se encuentran y el valor del objeto seleccionado.

La estructura lógica de este componente es similar a la del componente Creation. Se genera un primer menú que presenta los objetos correspondientes, en este caso, los nombres de las propiedades configurables del componente. Durante la etapa de configuración, se generan menús gracias al componente Menu-disco, y a los objetos de este disco se añade el componente events-object-configuration, que gestiona la lógica del evento selección del objeto. Al seleccionar un

objeto, se crean menús jerárquicos de manera ascendente hasta que cada propiedad queda completamente configurada, momento en el cual el componente se actualiza con los nuevos valores.

En la etapa de configuración, se requiere un mayor número de menús en comparación con la etapa de creación, por lo que se incrementaron los tipos de menús. Cada uno de estos menús es único y se utiliza según la función que deba cumplir. El flujo de ejecución empieza cuando se selecciona un objeto del menú, el componente events-object-configuration actualiza el componente Configuration con nuevos valores en sus propiedades. Estos valores se pasan al método update del componente, y dependiendo de la selección realizada, se sigue un camino de configuración u otro.

Definición del método Update del componente Configuration:

```
update: function () {
    this.valuesSelected[this.data.typeObjectSelected] = this.data.valueObjectSelected;

    if (this.valuesSelected[CONSTANTS.MAINOPCION] === CONSTANTS.QUERYES) {
        this.executeMenuQueryes();
    } else if (this.valuesSelected[CONSTANTS.MAINOPCION] === CONSTANTS.GRAPHS) {
        this.executeMenuGraphs();
    } else if (this.valuesSelected[CONSTANTS.MAINOPCION] === CONSTANTS.FILTERS) {
        this.executeMenuFilters();
    }
},
```

Es fundamental prever todos los posibles caminos para cada propiedad de todos los componentes de BabiaXR, lo que añade un nivel considerable de complejidad al componente. Cada tipo de componente BabiaXR tiene configuraciones únicas, lo que implica que se deben gestionar múltiples rutas lógicas y procesos de actualización. Además, cualquier cambio realizado debe reflejarse en el menú de información para que su panel muestre siempre información actualizada. Es importante destacar que algunas propiedades configurables dependen directamente de los campos presentes en los archivos de datos. Por ejemplo, el componente de filtros necesita acceder a todos los campos del archivo, así como a los posibles valores de cada uno para configurarse correctamente. De manera similar, el componente de gráficos requiere separar los campos de datos en valores numéricos y strings, ya que estos son los que se mostrarán en las gráficas.

Para recopilar los campos del archivo de datos seleccionado, se diseñaron dos métodos específicos, uno para archivos JSON y otro para archivos CSV. Estos métodos recorren todos los

campos del archivo recopilando, en una variable global del componente, los campos y sus valores únicos. Además, se dividen los campos en dos categorías según sus valores sean numéricos o strings. Estos métodos se ejecutan antes de configurar cualquier propiedad que dependa de los campos de datos del archivo, asegurando una correcta configuración de las propiedades.

Método collectFieldsFromJSON para extraer campos de un archivo JSON:

```
let collectFieldsFromJSON = (self, idDocument) => {
    fetch('./data/' + idDocument + '.json')
        .then(response => response.json())
        .then(data => {
            const keys = Object.keys(data[0]);
            const stringKeys = [];
            const numberKeys = {};
            const uniqueValues = {};
            keys.forEach(key => {
                const value = data[0][key];
                if (typeof value === 'string') {
                    stringKeys.push(key);
                } else if (typeof value === 'number') {
                    numberKeys[key] = [];
                }
            });
            stringKeys.forEach(key => {
                const uniqueSet = new Set();
                data.forEach(obj => {
                    uniqueSet.add(obj[key]);
                });
                uniqueValues[key] = Array.from(uniqueSet);
            });
            const fields = {
                key: uniqueValues,
                size: numberKeys
            };
            self.documentsCreated[idDocument] = fields;
        })
        .catch(error => console.error('Error al cargar el archivo:', error));
}
```

Events-object-configuration

Este componente comparte una lógica similar al componente Events-object-creation, pero este está diseñado para ser añadido a los objetos de la etapa de configuración. Su función principal es asignar un ícono 3D a cada objeto en todos los menús de configuración, junto con un

título que permite identificarlos claramente. Cuenta con tres propiedades relacionadas con el objeto, su valor, el tipo, y el nombre del modelo 3D que se le va a asignar. Además, gestiona el evento de clic para elegir el objeto. Al seleccionar un objeto, este componente enviará los valores correspondientes al método update del componente Configuration a través del método setAttribute para continuar con el proceso de configuración.

Definición del componente events-object-configuration:

```
AFRAME.registerComponent('events-object-configuration', {
  schema: {
    objectStage: { type: 'string', default: '' },
    objectType: { type: 'string', default: '' },
    nameObjectGLB: { type: 'string', default: '' },
  },

  handleClick: function (evt) {
    //Enviamos los datos del objeto al componente Configuration
    this.menuConfiguration.setAttribute('configuration', {
      typeObjectSelected: this.data.objectType,
      valueObjectSelected: this.data.objectStage,
    });
  }

  //Lógica
}
}
```

3.2.3. Componentes complementarios

Estos componentes, con objetivos muy específicos, son esenciales para asegurar el funcionamiento fluido y eficiente del proyecto. Se encargan de gestionar el proceso de inicio de sesión de los usuarios cuando acceden por primera vez a la escena, configurando su avatar y controles. Además, garantizan la correcta sincronización de los distintos componentes en entornos multiusuario, permitiendo que todas las acciones y modificaciones realizadas por un usuario se reflejen en tiempo real para el resto de los participantes.

Create-avatar

Este componente se activa cuando un usuario se conecta a la escena. Su objetivo principal es crear la entidad del usuario, integrando todo lo relacionado con el movimiento del avatar, la cámara y la apariencia. En un primer momento, se selecciona aleatoriamente un modelo

3D para el avatar, al que se le añaden componentes de cámara centrados en la cabeza para lograr una representación lo más realista posible. Además, se implementan controles para el movimiento, ya sea mediante teclado y ratón en PC o utilizando controladores VR, así como todos los elementos necesarios para interactuar con la interfaz.

También se asigna un nombre de usuario único generado aleatoriamente, que aparece sobre el avatar para permitir la identificación entre los distintos usuarios conectados. Además, a la entidad del avatar se le incorpora el componente networked-audio-source, lo que permite la comunicación mediante el micrófono con otros usuarios en la escena. Por último, se incluye la entidad del panel de instrucciones de los controles enfrente del avatar, de modo que al conectarse a la escena sea lo primero que vea el usuario.

Definición de la entidad del avatar:

```
let entityRig = document.createElement('a-entity');
entityRig.setAttribute('networked', 'template', '#auxInit-template');
entityRig.setAttribute('id', 'rig-player');
entityRig.setAttribute('movement-controls', 'fly', 'true');
entityRig.setAttribute('spawn-in-circle', 'radius', '15');

let entityAvatar = document.createElement('a-entity');
entityAvatar.setAttribute('networked', 'template', '#auxInit-template');
entityAvatar.setAttribute('networked-audio-source', '');
entityAvatar.setAttribute('id', 'player');
entityAvatar.setAttribute('camera', '');
entityAvatar.setAttribute('look-controls', '');

let entityAvatarGLB = this.addModelGLB(randomModel);
let entityAvatarTitle = this.addTitleModelGLB(randomModel);

entityAvatar.appendChild(entityAvatarGLB);
entityAvatar.appendChild(entityAvatarTitle);
entityRig.appendChild(this.addPanelInstruc())
entityRig.appendChild(entityCursor);
entityRig.appendChild(entityAvatar);
```

Events-controller

Este componente se activa automáticamente al entrar en la escena, ya que está integrado en el HTML principal. Una vez que los controles del avatar están habilitados, tanto para navegador web como para controladores VR, solo queda definir qué controles van a realizar las diferentes

interacciones en la escena.

En el caso de los usuarios en ordenador, se agrega un evento global al documento que detecta cuando se presiona la tecla de espacio. Al hacerlo, se genera una entidad con el componente Creation, que toma la posición del avatar y la rotación de la cámara para crear el menú inicial de creación justo enfrente del avatar. De manera similar, en los controladores VR se debe presionar simultáneamente los botones B e Y para generar el menú inicial de creación. Una vez activado el componente Creation, todos los demás componentes pueden ser inicializados y gestionados tanto desde él como desde los nuevos componentes activados.

Para interactuar con la escena en VR, se implementó un evento que activa la interacción al presionar los gatillos de cualquier controlador con los objetos de la escena. En el caso del navegador web, esta interacción se realiza con un clic del ratón. Estas asignaciones de teclas y botones se establecieron intencionalmente para explorar a fondo el control de los botones tanto en los controladores VR como del teclado.

Definición de los eventos de los controles:

```
//INTERACTUAR TRIGGER VR
document.addEventListener('triggerdown', () => {
  console.log("CLICK abuttondown")
  self.data.pressbuttonTrigger = true;
});

//GENERADOR MENÚ INICIAL DE CREACIÓN PARA VR
self.bButtonPressed = false;
self.yButtonPressed = false;

document.addEventListener('bbuttondown', function () {
  self.bButtonPressed = true;
  self.checkBothButtons();
});
document.addEventListener('bbuttonup', function () {
  self.bButtonPressed = false;
});
document.addEventListener('ybuttondown', function () {
  self.yButtonPressed = true;
  self.checkBothButtons();
});
document.addEventListener('ybuttonup', function () {
  self.yButtonPressed = false;
});
```

```
//GENERADOR MENÚ INICIAL DE CREACIÓN PARA PC
document.addEventListener('keydown', function (event) {
  if (event.code === 'Space') {
    self.createMenuInit();
  }
});
```

Spawn-in-circle

Este componente se añade a la entidad del usuario cuando se activa el componente create-avatar. Su función es garantizar que los usuarios conectados aparezcan por primera vez en diferentes posiciones y orientaciones dentro de la escena, evitando así que todos los avatares se generen en el mismo lugar al conectarse. El componente cuenta con una propiedad llamada radius, que define el área en la que los usuarios pueden aparecer, cuanto mayor sea el valor de esta propiedad, mayor será el radio dentro del cual se distribuyen los avatares en la escena.

Load-schemas

Este componente es fundamental para sincronizar los componentes entre todos los usuarios en la escena. Networked A-Frame permite crear entornos de realidad virtual multiusuario, pero para lograrlo es necesario definir schemas que especifiquen qué componentes se compartirán. Para que una entidad sea visible y sincronizada con otros usuarios, debe incluir el componente Networked A-Frame con una propiedad template. Es necesario definir un schema para cada template que se desee compartir, y especificar qué componentes se sincronizarán. Todos los componentes que deban estar sincronizados con los demás usuarios deben estar incluidos en el schema correspondiente a su template. Si no se incluyen en el schema, el componente solo será visible para el usuario local y no se compartirá en la escena global. Por tanto, este componente contiene todos los schemas necesarios para la correcta sincronización en la escena.

Definición de los schemas de templates de los menús y objetos de los menús:

```
NAF.schemas.getComponentsOriginal = NAF.schemas.getComponents;
NAF.schemas.getComponents = (template) => {
  if (!NAF.schemas.hasTemplate('#platoMenu-template')) {
    NAF.schemas.add({
      template: '#platoMenu-template',
      components: [
        'position',
```

```

'scale',
'geometry',
'animation',
'component-synchronize',
'material',
]
});
}

if (!NAF.schemas.hasTemplate('#objetoMenu-template')) {
NAF.schemas.add({
template: '#objetoMenu-template',
components: [
'position',
'scale',
'gltf-model',
'animation',
'component-synchronize',
'babilia-queryjson',
'babilia-querycsv',
'babilia-filter',
'look-at',
]
});
}

const components = NAF.schemas.getComponentsOriginal(template);
return components;
};
}

```

Component-synchronize

Este componente tiene como objetivo sincronizar acciones que no están integradas en un componente específico. La sincronización de componentes se realiza utilizando el método `setAttribute`, que actualiza las propiedades del componente. Sin embargo, cuando se desea sincronizar una acción que no utiliza `setAttribute`, esta no se reflejará en todos los usuarios conectados. Al eliminar un componente de una entidad con `removeAttribute`, el cambio solo afecta al entorno local, sin actualizarse para los demás usuarios. Esto mismo sucede al intentar modificar la opacidad de un modelo 3D. Para resolverlo, este componente cuenta con dos propiedades clave, `componentShare` y `valueShare`. Dependiendo del valor asignado a `componentShare`, se encargará de eliminar un componente o actualizar la opacidad del modelo 3D. La propiedad

valueShare define qué componente será eliminado o qué nivel de opacidad se debe ajustar, permitiendo que los cambios se sincronicen correctamente entre todos los usuarios.

Definición del componente component-synchronize:

```
AFRAME.registerComponent('component-synchronize', {
  schema: {
    componentShare: { type: 'string', default: '' },
    valueShare: { type: 'string', default: '' },
  },
  update: function () {
    let self = this;

    if (self.data.componentShare == 'opacity') {
      self.el.object3D.traverse((value) => {
        if (value.type === 'Mesh') {
          const material = value.material;
          material.transparent = true;
          material.opacity = self.data.valueShare;
        }
      })
    } else if (self.data.componentShare == 'remove') {
      self.el.removeAttribute(self.data.valueShare);
    }
  },
});
```

Constants

Este componente contiene únicamente variables constantes de tipo String que se utilizan en todos los componentes del proyecto. Su propósito es reducir la redundancia de escribir cadenas de texto repetidas en cada componente, lo que además minimiza errores. Al centralizar los Strings en variables constantes, facilita la modificación de cualquier cadena de texto, ya que solo es necesario actualizar la variable correspondiente en lugar de cambiarla en múltiples lugares del código. Esto mejora la mantenibilidad y coherencia del proyecto.

Capítulo 4

Desarrollo del proyecto

Tras haber expuesto las características y funcionalidades del sistema resultante de visualización de datos, en este capítulo se dedicará a detallar el proceso de investigación y desarrollo que dio origen a este sistema multiusuario en realidad virtual.

A este proceso se aplicó estrategias basadas en metodologías ágiles Scrum para organizar el trabajo, un marco que facilita la gestión de proyectos complejos mediante ciclos iterativos conocidos como sprints. El proyecto se llevó a cabo a lo largo de nueve meses, dividiéndose en sprints de aproximadamente un mes cada uno. Durante este tiempo, se equilibró el desarrollo del proyecto con un empleo a tiempo completo, dedicando las tardes entre semana, tras la jornada laboral, y los fines de semana a avanzar en el proyecto. El equipo de trabajo estuvo compuesto por dos miembros. Mi tutor Jesús María González Barahona, actuando como Scrum Master, su responsabilidad fue guiar mi trabajo y mantener una supervisión constante del proyecto. Y mi rol, como desarrollador, consistió en implementar los objetivos acordados en las reuniones periódicas.

En cada sprint se definieron objetivos específicos y las actividades necesarias para alcanzarlos. Las reuniones con el tutor, llevadas a cabo a través de Microsoft Teams, se realizaban cada 1 o 2 semanas para resolver dudas y evaluar el avance de las actividades acordadas. Además, al finalizar cada sprint, aproximadamente una vez al mes, se realizaban reuniones más extensas donde se revisaba lo aprendido durante el sprint, se discutían las direcciones y ajustes necesarios para el próximo sprint.

4.1. Sprint 1: Primeros pasos en realidad virtual

En este primer sprint se aborda el estudio inicial de la tecnología principal del proyecto, A-Frame. Explorando sus características fundamentales y comprendiendo su estructura básica. Además del estudio de A-Frame, se dedica tiempo a familiarizarse con las gafas de realidad virtual y a configurar GitHub como la herramienta principal para almacenar y gestionar el código del proyecto.

A continuación se detalla el plan de trabajo para este sprint:

■ **Objetivos:**

1. Dominar el uso básico de A-Frame.
2. Integrar el código desarrollado en un repositorio GitHub.
3. Testear las gafas de realidad virtual.

■ **Actividades:**

1. Principios básicos de A-frame.
2. Desarrollo de programas de prueba.
3. Exploración y pruebas con gafas de realidad virtual.

Duración: 1 mes

4.1.1. Principios básicos de A-frame

Se comenzó con la revisión de la documentación oficial de A-Frame, realizando la sección del tutorial, donde se aprendió a instalar y configurar el framework. Se creó un archivo HTML simple, incluyendo los scripts necesarios para ejecutar A-Frame en un navegador web. Además, se añadieron entidades geométricas primitivas directamente en el HTML como cajas, esferas y planos.

Se experimentó con estas entidades básicas modificando sus atributos como la posición, color y tamaño de las geometrías. También, se utilizó el inspector de A-Frame para obtener una vista alternativa de la escena y observar el efecto de las modificaciones en tiempo real.

Se avanzó en la obtención de entidades en el DOM mediante queries, lo que permitió manipular directamente las propiedades de los objetos en la escena. Además, se crearon componentes propios básicos, definiendo nuevas funcionalidades que no estaban incluidas en los componentes predeterminados de A-Frame. Estos componentes personalizados se integraron en las entidades primitivas, permitiendo la introducción de nuevos atributos y la gestión de eventos como clics o temporizadores. Esto permitió añadir funcionalidades específicas y mejorar la interacción con las entidades dentro de la escena.

4.1.2. Desarrollo de programas de prueba

En esta actividad, se desarrolló varios componentes personalizados desde cero. Se puso especial énfasis en comprender la estructura básica de la definición de un componente, abordando métodos clave del ciclo de vida como `.init()`, que se ejecuta cuando el componente se adjunta por primera vez a su entidad, y `.update()`, que se invoca después del `.init()` y cada vez que se actualiza el componente.

El primer objetivo fue crear un componente que permitiera cambiar el color de una esfera al hacer clic sobre ella, seleccionando un color aleatorio en cada interacción. Además, se implementó otro componente que al hacer clic sobre una caja generaba tres cajas hijas directamente en el DOM con una posición más arriba respecto a la entidad padre.

Por último, se creó un repositorio específico para centralizar todo el código desarrollado durante el TFG. Todos los programas de prueba y las implementaciones del proyecto final fueron versionados y almacenados en GitHub.

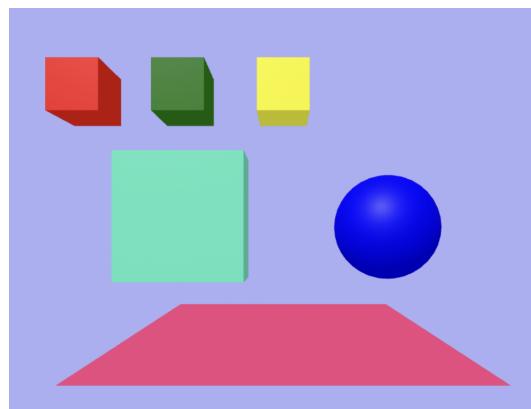


Figura 4.1: Resultado del programa de prueba.

4.1.3. Exploración y pruebas con gafas de realidad virtual

En esta fase, se llevaron a cabo pruebas con las gafas de realidad virtual Meta Quest, comenzando por explorar sus capacidades a través de apps y juegos de la plataforma. Se familiarizó con el uso de los mandos, evaluando la experiencia inmersiva y la respuesta de los controles en diferentes aplicaciones.

Posteriormente, se accedió al navegador web desde las gafas de realidad virtual, ingresando la URL del servidor local para visualizar y probar el funcionamiento de los programas desarrollados previamente. Para adaptarse a los mandos, fue necesario cambiar la entidad cursor, que permitía el funcionamiento del ratón, al controlador de Oculus para navegar y seleccionar elementos mediante los mandos.

4.2. Sprint 2: Escenas multiusuario y extensiones de A-Frame

En este segundo sprint, se realiza un estudio profundo de Networked A-Frame, una de las tecnologías clave para el proyecto. Esta tecnología permitirá implementar escenas multiusuario, donde todos los participantes conectados podrán ver y experimentar los cambios en tiempo real. Además, se investigarán diversas bibliotecas de A-Frame para explorar funcionalidades avanzadas y complejas que enriquecerán al proyecto.

A continuación se detalla el plan de trabajo para este sprint:

■ **Objetivos:**

1. Familiarizarse con Networked A-Frame.
2. Explorar librerías adicionales para A-Frame.

■ **Actividades:**

1. Análisis de Networked A-Frame.
2. Introducción de Superhands y A-Frame Extras.
3. Desarrollo de programas de prueba.

Duración: 1 mes

4.2.1. Análisis de Networked A-Frame

Se comenzó estudiando el repositorio oficial de Networked A-Frame (NAF), siguiendo el tutorial para configurar una experiencia desde cero. Inicialmente, se instalaron las dependencias necesarias y se configuró un servidor local propio. Posteriormente, se puso en marcha el servidor para permitir la conexión de múltiples usuarios y se descargó un ejemplo básico en HTML para observar el funcionamiento de Networked A-Frame. Se abrieron dos navegadores web distintos, se conectaron a la misma URL del servidor local y se interactuó con los avatares prediseñados incluidos en el ejemplo. Además, se probaron todos los ejemplos disponibles del repositorio para explorar las capacidades completas del framework.

Una vez comprendidas las capacidades del NAF, se procedió al análisis detallado del código. Se examinó el componente principal networked-scene, responsable de la conexión a una sala. Este componente incluye varios atributos cruciales que permiten configurar la escena compartida, como el tipo de servicio de red, el nombre de la sala, y la posibilidad de habilitar audio y video entre avatares.

A continuación, se investigó la creación de entidades en red. Para ello, se debe añadir el componente networked a una entidad estándar junto con una plantilla correspondiente, lo que permite a otros usuarios ver la entidad en la red. También, se estudió la sincronización de componentes personalizados que requieren definir un esquema por plantilla para habilitar su sincronización, aunque para la posición y rotación de la entidad se sincronizan por defecto.

4.2.2. Introducción de Superhands y A-Frame Extras

Para profundizar en A-Frame, se estudiaron algunas de sus extensiones más importantes que no están integradas de forma nativa en el framework. Se comenzó con la librería Superhands, instalando sus dependencias y probando sus demostraciones con gafas de realidad virtual. Esta librería permite la funcionalidad de interacción manual con los objetos de la escena, como agarrar, lanzar, estirar o arrastrar objetos. Tras explorar las capacidades de Superhands, se analizó su código para comprender su funcionamiento. El componente principal, super-hands, es el encargado de permitir mover las entidades, basándose en la interacción del usuario y colisiones. Generalmente, este componente se aplica a las entidades de los controladores y depende de un componente de detección de colisiones, como sphere-collider de la librería A-Frame Extras,

que se debe incluir en la misma entidad que super-hands.

Otra extensión importante que se estudió fue A-Frame Extras, que incluye componentes y utilidades adicionales como controles, cargadores de modelos y nuevas entidades primitivas. Entre los componentes probados, está el movement-controls, que permite mover la cámara del usuario con las teclas WASD y las flechas del teclado. El gltf-model, utilizado para cargar modelos 3D en formato .glb, y animation-mixer, que habilita la animación en bucle de los modelos 3D. También, se experimentó con nuevas entidades primitivas como a-ocean, un océano con olas animadas, y a-tube, un tubo que sigue una ruta personalizada.

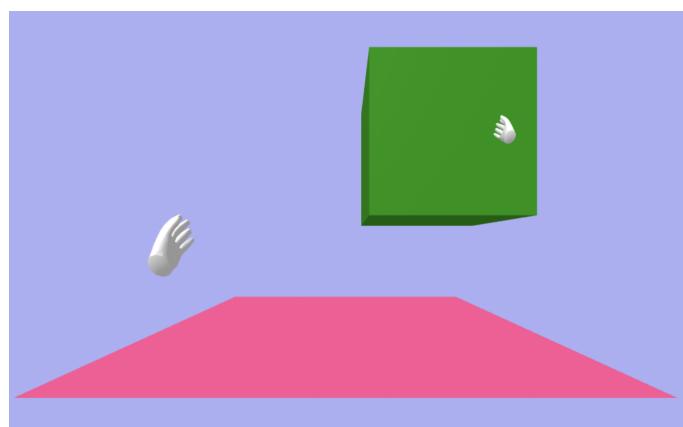


Figura 4.2: Escena agarrando una caja con superhands.

4.2.3. Desarrollo de programas de prueba

El primer programa realizado fue una prueba del componente de animation con varias entidades. Se experimentó con la animación de la propiedad scale, haciendo que las entidades crecieran o disminuyeran de tamaño durante un tiempo determinado. También, se utilizó la propiedad rotation para rotar las entidades de manera continua en diferentes ángulos, y la propiedad position para animar el movimiento de una entidad hacia una posición específica.

Se creó una nueva escena para integrar dos usuarios con Networked A-Frame, permitiendo que cada usuario que se conectara al servidor iniciara con un avatar con geometría esférica y color aleatorio. Posteriormente, se creó una plantilla para una caja, esta plantilla fue utilizada con el componente networked, lo que permitía compartir esta entidad con los demás usuarios conectados.

El siguiente paso fue añadir el componente animation a la caja pero se observó que el mo-

vimiento solo se reproducía en el navegador de uno de los usuarios. Esto ocurrió porque los componentes no se sincronizan entre usuarios hasta que se define un esquema por plantilla, especificando qué componentes deben sincronizarse. Una vez realizado esto, la animación de la caja se sincronizó correctamente y fue visible en tiempo real en los navegadores de ambos usuarios.

En el último ejercicio, se creó una nueva escena compartida utilizando la extensión Super-hands. El objetivo fue que uno los usuarios conectados, utilizando las gafas de realidad virtual, pudiera agarrar y arrastrar una esfera por la escena con las manos. Mientras el otro usuario observaba en tiempo real cómo la entidad se movía. Para lograrlo, la esfera tenía el componente `grabbable`, permitiendo que fuera manipulada, y el usuario contaba con el componente `super-hands` y un colisionador asignado a los mandos.

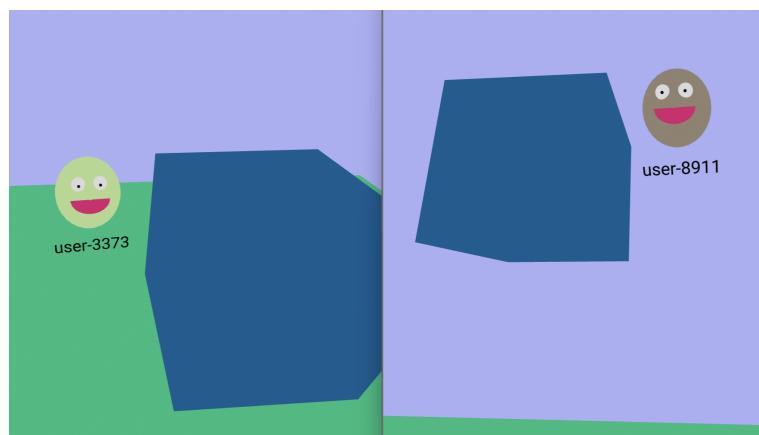


Figura 4.3: Escena compartida con avatares.

4.3. Sprint 3: Etapa final de formación

Este sprint marca la fase final de formación en las tecnologías clave del proyecto. Durante este sprint, se consolidará los conocimientos en A-Frame, con especial énfasis en la creación de escenas multijugador. Además, se iniciará en el estudio de BabiaXR, una librería esencial que permite generar visualizaciones gráficas a partir de datos. Con el dominio adquirido durante estos tres primeros sprints, se estará plenamente capacitado para abordar el desarrollo del proyecto final.

A continuación se detalla el plan de trabajo para este sprint:

■ Objetivos:

1. Profundizar en el uso de A-Frame y Networked A-Frame.
2. Aprender BabiaXR.

■ Actividades:

1. Desarrollar un minijuego.
2. Introducción de BabiaXR.
3. Desarrollo de programa de prueba.

Duración: 1 mes

4.3.1. Desarrollar un minijuego

Se diseñó un minijuego de memoria multiusuario para profundizar en el uso práctico de A-Frame y Networked A-Frame. El juego consiste en una cuadrícula de cartas que los usuarios deben voltear para emparejar pares. Los usuarios conectados tienen como avatar un modelo 3D de un hombre. En este caso, las cartas son paneles negros situados en el suelo. Los usuarios, ya sea usando el ratón o los mandos de las gafas de realidad virtual, pueden hacer clic en estos paneles y transformarlos en cajas de colores aleatorios pares. Los usuarios tienen dos intentos para emparejar las cajas del mismo color, si tienen éxito, obtienen otros dos intentos para descubrir más pares de colores. Si no logran emparejar las cajas correctamente, se transformarán en panales otra vez y pasará el turno al siguiente usuario.

Desarrollar este minijuego permitió perfeccionar la creación de componentes con lógica moderada y controlar la integración de entidades compartidas, sincronizando tanto componentes personalizados como predefinidos.

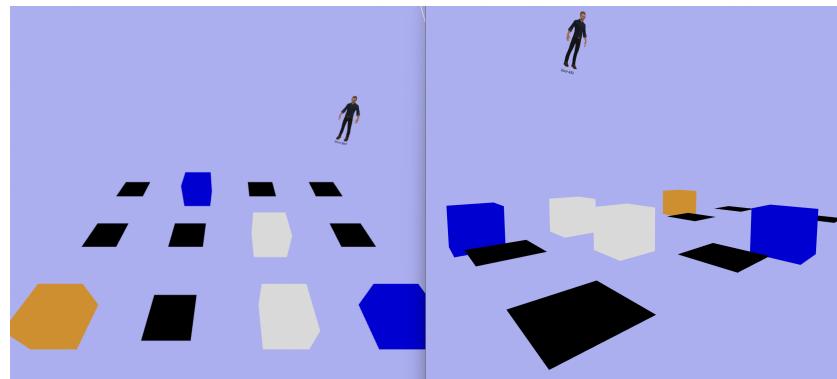


Figura 4.4: Escena compartida del minijuego.

4.3.2. Introducción de BabiaXR

Se comenzó a estudiar la librería BabiaXR, diseñada para crear visualizaciones gráficas a partir de datos sin procesar. Se consultó su página web oficial para revisar ejemplos y evaluar sus capacidades. Posteriormente, se siguió el tutorial para crear gráficas sencillas, se creó un archivo HTML con el script de BabiaXR. Se empezó a explorar los componentes para la recuperación de datos desde archivos JSON o CSV y se experimentó con filtros para refinar los datos. Finalmente, se probaron los componentes gráficos de visualización, como gráficos de tarta, barras y donut, ajustando varias propiedades de los componentes para observar cómo cambiaban los resultados.

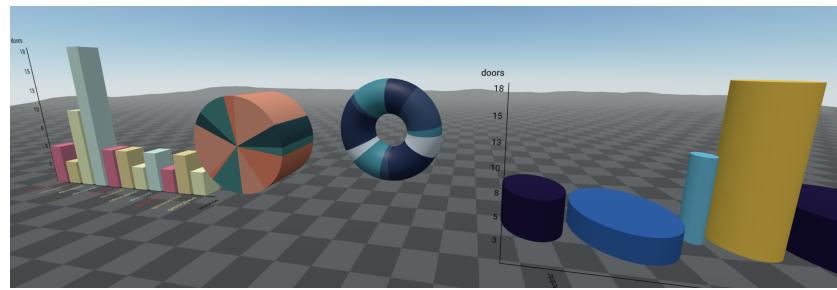


Figura 4.5: Escena con gráficas de datos.

4.3.3. Desarrollo de programa de prueba

Tras familiarizarse con los conceptos básicos de BabiaXR, se llevó a cabo un ejercicio práctico para desarrollar una interfaz de usuario simple que permitiera cambiar dinámicamente el archivo JSON de los datos, actualizando en consecuencia una gráfica de pastel. Primero, se

creó un archivo HTML que incluía el script de BabiaXR. A continuación, se implementaron tres componentes babia-queryjson, cada uno vinculado a un archivo JSON diferente. Se configuró un componente de gráfica de pastel babia-pie, estableciendo como valor predeterminado uno de los archivos JSON. Posteriormente, se creó el componente ui-querys que gestionaría la interfaz de usuario. Este componente genera una fila de cuatro paneles interactivos. El primer panel de color violeta muestra el nombre de DATA para indicar que los datos son modificables, y los tres paneles restantes en azul muestran los nombres de los archivos JSON disponibles. Al hacer clic en uno de estos paneles, se actualiza automáticamente la gráfica con los nuevos datos del archivo seleccionado, y el panel cambia su color a gris para indicar la selección. Además, esta interfaz está compartida para todos los usuarios conectados, por lo que cualquier cambio en la gráfica se actualizará en tiempo real para todos los usuarios.

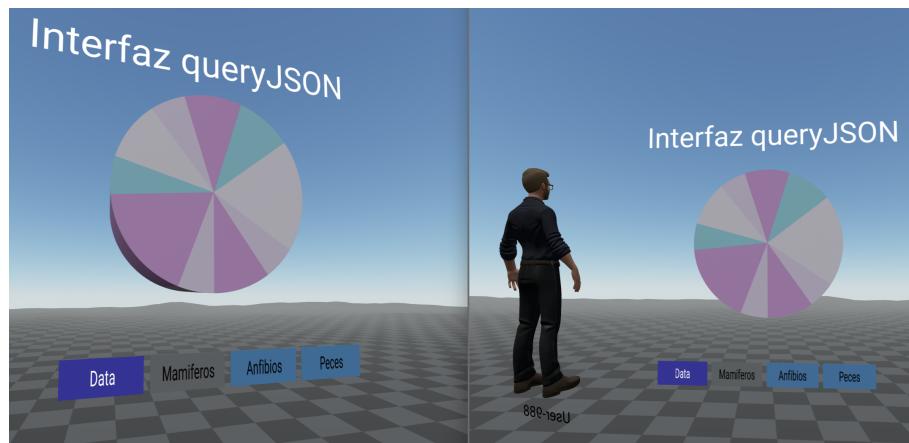


Figura 4.6: Escena compartida de la interfaz interactiva.

4.4. Sprint 4: Inicio del proyecto

Este sprint marca el punto de partida para el desarrollo y la estructuración del proyecto final. Con los conocimientos adquiridos en los sprints previos, se establece la base para lo que será una aplicación multiusuario de visualización de datos. En este primer paso, se comienza a definir la estructura principal del proyecto y se da inicio a la creación de una primera versión de la interfaz de usuario, la cual permitirá a los usuarios conectados crear y configurar solamente componentes queries de BabiaXR.

A continuación se detalla el plan de trabajo para este sprint:

■ Objetivos:

1. Diseñar la estructura del proyecto.
2. Iniciar la creación de la interfaz de usuario.

■ Actividades:

1. Estructura principal del proyecto.
2. Crear una primera versión de la interfaz de usuario.

Duración: 1 mes

4.4.1. Estructura principal del proyecto

La idea principal es desarrollar una interfaz de usuario que permita crear y configurar componentes de BabiaXR. Para ello, se dividió en tres tipos: componentes de queries, de filtros y de gráficas. Los componentes de queries, como babia-queryjson y babia-querycsv, recuperarán la información desde archivos. Los filtros, representados por el componente babia-filter, permiten refinar estos datos. Por último, los componentes de gráficas, como babia-pie, babia-doughnut y babia-bars, se encargarán de visualizarlos.

En la primera etapa de la interfaz, los usuarios podrán ver los tres tipos de componentes. Al hacer clic en uno de ellos, se desplegarán los componentes específicos de la categoría seleccionada. Cada vez que se elija un componente, se creará una entidad en el DOM con el componente seleccionado y con su respectivo ícono en la escena, representando visualmente el componente de BabiaXR.

En la segunda etapa, se accederá a la configuración del componente creado, donde se podrán modificar sus propiedades. Cada componente tiene varias propiedades configurables que aparecerán en la interfaz con sus valores correspondientes. El mayor desafío de esta idea radica en gestionar las propiedades de todos los componentes creados y garantizar la correcta interacción entre ellos, especialmente en el caso de los componentes de gráficas, que dependerán tanto de los componentes de queries como de los filtros previamente creados. Además, los filtros dependerán también de los componentes de queries para su correcto funcionamiento. A todo esto se suma la necesidad de que todo lo creado y configurado se sincronice en tiempo real con todos

los usuarios conectados, permitiendo que cada uno de ellos pueda crear y configurar sus propios componentes, visibles para el resto de usuarios.

4.4.2. Crear una primera versión de la interfaz de usuario

En esta primera versión de la interfaz, el enfoque está en la planificación de la interfaz y en la lógica para los componentes solamente de tipo queries. Se comienza creando un nuevo archivo HTML e incluyendo los scripts necesarios, como A-Frame, Networked A-Frame, BabiaXR y sus extensiones correspondientes. Se incorpora una plantilla de entidad avatar para que, al iniciar sesión, cada usuario tenga un avatar masculino predeterminado. También, se añade una plantilla de dos discos apilados de diferentes colores para representar un escenario con el componente de animación para que giren sobre sí mismos con una velocidad moderada. Además, se coloca un ícono de creación en la parte superior del escenario. Al ser hijo de la entidad escenario, este ícono girará junto con los discos.

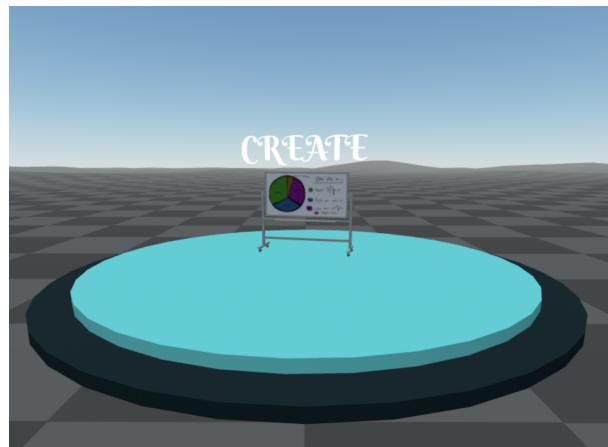


Figura 4.7: Escenario con ícono de creación.

Si un usuario conectado hace clic en el ícono de creación, este desaparecerá y se desplegarán tres paneles por encima, cada uno con una imagen que representa diferentes tipos de componentes: queries, gráficos y filtros. Al seleccionar uno de estos paneles, se generarán íconos correspondientes a los componentes que se pueden crear, ubicados sobre el escenario dando vueltas.

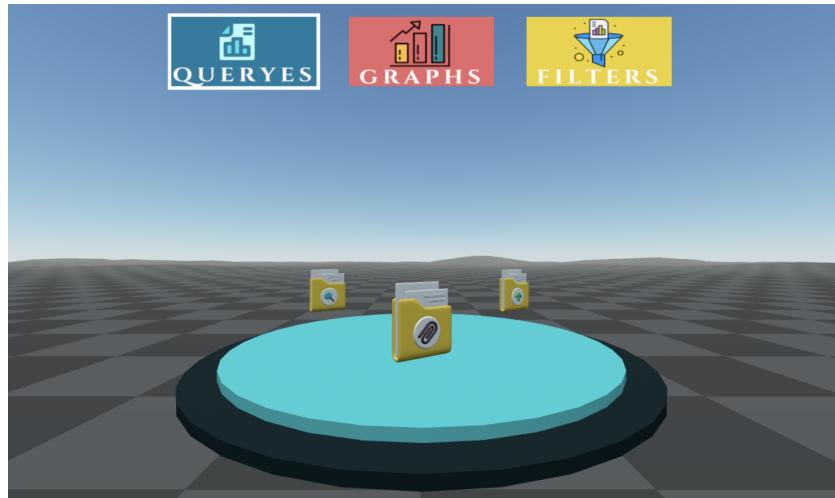


Figura 4.8: Escenario con iconos de componentes tipo queries.

Al hacer clic en uno de estos iconos, se creará una entidad con el componente elegido. Después de una transición de movimiento de los discos, todos los elementos desaparecen y solo permanece visible el ícono seleccionado encima de los discos con otro color diferente. Si se vuelve a hacer clic en el ícono, aparecerá un pequeño ícono de configuración a la derecha. Al accionar este ícono, se abrirá un panel con las propiedades configurables del componente y sus valores. Una vez configurado, se podrá hacer clic en un ícono de guardar configuración, lo que aplicará los valores seleccionados al componente creado.

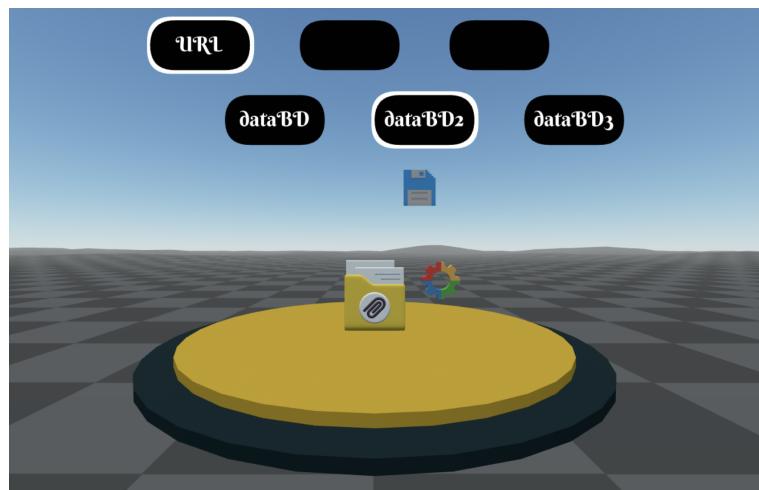


Figura 4.9: Escenario de configuración del componente babia-queryjson.

En este caso, se seleccionó el componente babia-queryjson. En la interfaz de configuración, la primera fila de paneles con esquinas redondeadas muestra su única propiedad configurable,

url, que sirve para especificar el nombre del archivo JSON de datos. En la segunda fila se presentan los posibles valores de archivos que se pueden asignar a la url.

Cabe destacar que los paneles con esquinas redondeadas no están disponibles como entidades primitivas en A-Frame. Por lo que se investigó si existía una librería que hiciera esto, y se logró encontrar la librería A-Frame Rounded Primitive. Esta librería proporciona un componente que permite configurar paneles con esquinas redondeadas. Sin embargo, al probar el componente se descubrió que no funcionaba correctamente debido a que la librería, creada hace siete años, no era compatible con la versión actual de A-Frame. Se descargó el código oficial de la librería y se pudo encontrar el problema. La causa fue que la librería utilizaba la función THREE.ShapeBufferGeometry, que ya no estaba soportada. El problema se solucionó actualizando la función a una versión compatible, permitiendo que los paneles con esquinas redondeadas funcionaran correctamente.

4.5. Sprint 5: Progreso de la interfaz de usuario

Tras establecer la estructura básica de la interfaz de usuario y definir el plan de desarrollo, en este sprint se enfoca en mejorar la interactividad de la interfaz para los usuarios. Se probaron varias versiones hasta lograr una más estable y funcional. Además de rediseñar la interfaz, también fue necesario ajustar la lógica del código al integrar los componentes de gráficos de datos. Los componentes desarrollados desde cero que gestionan toda la lógica de la interfaz de usuario, ahora son más versátiles y permiten crear y configurar los dos tipos principales de componentes de BabiaXR, queries y gráficas, lo que ha elevado el desafío técnico y de diseño.

A continuación se detalla el plan de trabajo para este sprint:

- **Objetivos:**

1. Rediseñar la interfaz de usuario.
2. Visualizar una gráfica de datos.

- **Actividades:**

1. Optimizar el diseño de la interfaz.
2. Implementar componentes de tipo de gráficas.

Duración: 1 mes

4.5.1. Optimizar el diseño de la interfaz

En esta actividad se desarrollaron mejoras para el diseño de la interfaz de usuario. Se comenzó identificando un problema clave, la interfaz parecía estar dividida en dos flujos desacoplados. Por un lado, los iconos en 3D seleccionables y por otra, paneles planos de opciones, lo que afectaba negativamente la interactividad y cohesión de la experiencia. Para mejorar este aspecto, se decidió unificar toda la interacción mediante iconos en 3D, manteniendo una experiencia visual y de flujo consistente. A partir de esta mejora, los paneles de queries, gráficos y filtros se representarán con iconos en lugar de paneles planos. Además, los paneles de configuración también seguirán este esquema, empleando iconos interactivos. Los iconos seleccionables estarán organizados sobre discos rotatorios que simularán menús.

Durante la etapa de creación de un componente, aparecerá un menú con los diferentes tipos de componentes disponibles para crear. Al seleccionar uno, el ícono se posicionará en el centro de la escena, mientras los demás íconos desaparecerán para resaltar la selección. A continuación, surgirá un segundo menú por encima del primero con las opciones correspondientes al componente seleccionado. Una vez elegido el componente específico, se generarán más menús laterales que indicarán las propiedades configurables del componente. Al seleccionar una propiedad, aparecerá otro menú por encima con los posibles valores para esa propiedad. Hasta este punto, la lógica de creación y modificación de componentes BabiaXR se ha implementado únicamente para los componentes de queries, sentando las bases para extender esta interacción a los demás tipos de componentes.

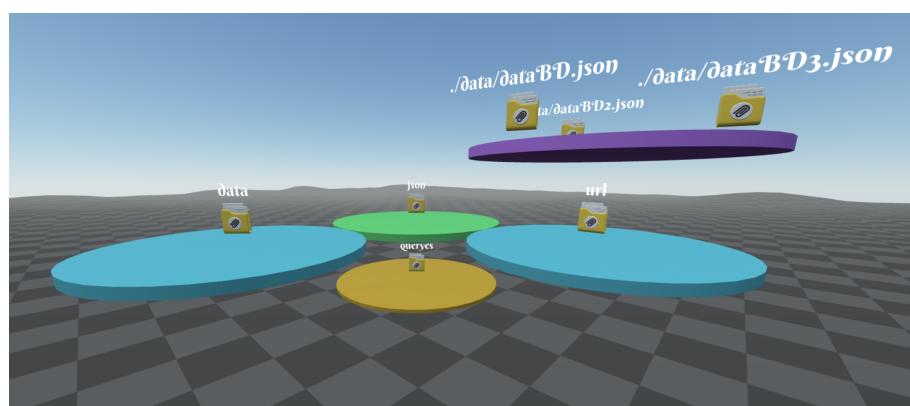


Figura 4.10: Interfaz de usuario interactiva mejorada

Tras las revisiones semanales, se implementó una nueva mejora en la interactividad de la interfaz. Anteriormente, los menús de configuración se creaban al lado del menú del componente seleccionado, pero ahora tanto en la fase de creación como en la de configuración, todos los menús se dispondrán uno sobre otro formando una estructura visual similar a una torre. Este enfoque refuerza la coherencia visual y facilita la navegación entre opciones. Además, se optimizó la visualización de los iconos no seleccionados. En lugar de desaparecer por completo, ahora se vuelven semitransparentes, permitiendo a los usuarios identificar claramente qué opciones no han sido seleccionadas.

En la fase de configuración, se incorporó un pequeño ícono de rueda multicolor junto al componente creado, con el fin de indicar claramente que desde ese punto se puede acceder a la etapa de configuración. Además, para los componentes de tipo queries, se añadió la capacidad de actualizar los valores de las propiedades en cualquier momento. Aunque se haya seleccionado y configurado un archivo para el componente, es posible volver a interactuar con cualquier ícono transparente, lo que permitirá actualizar el componente con el nuevo valor elegido. Al realizar esta acción, el ícono seleccionado recupera su opacidad completa, mientras que los demás se vuelven transparentes, indicando visualmente el cambio de estado y la nueva selección.

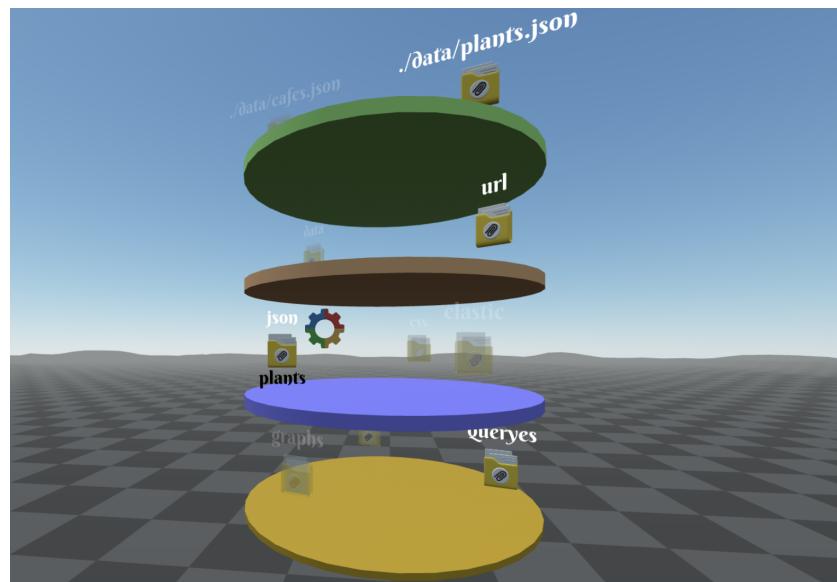


Figura 4.11: Interfaz de usuario de tipo query

4.5.2. Implementar componentes de tipo de gráficas

Después de establecer y probar la estructura básica de la interfaz con los componentes de tipo queries, se procede a implementar los componentes de tipo gráficas. Esta etapa requiere una reestructuración significativa del código debido a la complejidad adicional de las gráficas, que cuentan con más propiedades que los componentes de queries. Además, las gráficas dependen de la existencia de componentes query para funcionar correctamente, ya que sin ellos no se mostraría ninguna información. En primer lugar, se implementaron los componentes de gráficas de pastel y de dona. La propiedad clave de estos componentes es la propiedad from, que debe listar los íconos de los componentes query previamente creados, ya que estos proporcionarán los datos necesarios para la gráfica. Si no hay ningún componente query disponible, no se mostrará ningún ícono.

Una vez que se selecciona un componente query, se presentan dos propiedades adicionales esenciales para la configuración de la gráfica. La primera propiedad es key, que debe corresponder a los campos de datos que son cadenas de texto (string). La segunda propiedad es size, que debe estar relacionada con los campos numéricos de los datos. Aquí surge un desafío, es necesario leer y procesar los archivos JSON o CSV para dividir los campos en key y size antes de configurarlos. Una vez que se han seleccionado los valores de las propiedades key y size, la gráfica resultante se mostrará en el suelo de la escena.

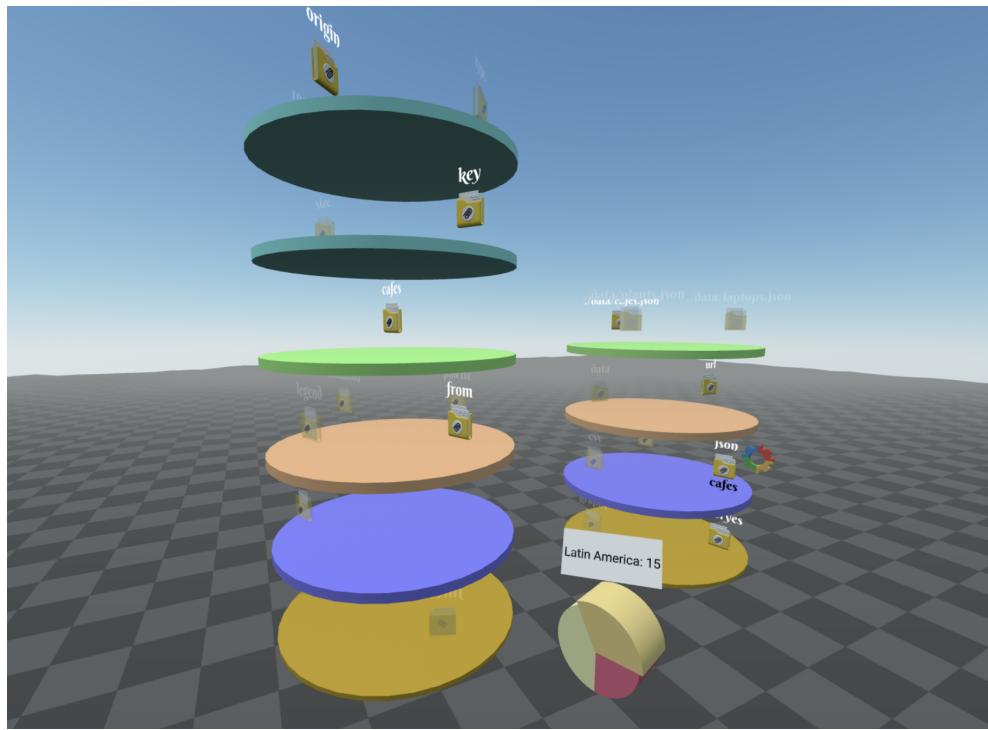


Figura 4.12: Interfaz de usuario de tipo query y gráfica.

4.6. Sprint 6: Sistema de visualización de datos

Con una estructura clara para la formación de menús con un formato jerárquico, y tras haber establecido la lógica de configuración de los componentes de query y gráficos, en este sprint se procederá a desacoplar las etapas de creación y configuración. Esto permitirá un enfoque más intuitivo sobre el inicio y final de cada fase del proceso.

Además, se han integrado nuevas acciones que los usuarios pueden realizar una vez que el componente ha sido creado. Finalmente, se lleva a cabo la implementación del último componente de visualización de datos, los filtros. Con todo esto, la interfaz de usuario, inicialmente concebida como un objetivo de solo crear y configurar, ha evolucionado hacia un sistema de visualización más complejo donde también se pueda eliminar el componente creado y obtener información acerca del componente.

A continuación se detalla el plan de trabajo para este sprint:

- **Objetivos:**

1. Desacoplar etapas de creación y configuración.

2. Integrar nuevos menús para el componente.
3. Generar un componente de tipo filtro.

■ **Actividades:**

1. Evolucionar la interfaz de usuario.
2. Implementar componentes de tipo filtro.

Duración: 1 mes

4.6.1. Evolucionar la interfaz de usuario

Se decidió mejorar la estructura de los menús para ofrecer una experiencia más clara y fluida. Hasta el momento, la etapa de creación y la etapa de configuración compartían la misma estructura, primero se creaba un menú para seleccionar el componente a crear y luego, mediante menús adicionales de forma ascendente, se realizaba la configuración. Esta organización generaba cierta confusión respecto a cuándo terminaba la fase de creación y cuándo comenzaba la de configuración. Para solucionar este problema, se optó por separar los dos menús, asignando uno exclusivamente para la creación y otro para la configuración. Además, se planteó la posibilidad de ofrecer más acciones una vez que se haya creado un componente, más allá de simplemente configurarlo. Esto llevó a la inclusión de dos nuevos menús, uno para eliminar el componente del DOM y otro de información, donde se muestra detalles importantes sobre el componente creado. Esta nueva estructura aporta mayor claridad y flexibilidad a la interacción con los componentes.

La nueva estructura quedó así, la etapa de creación muestra un primer menú donde el usuario puede seleccionar entre los tres tipos de componentes de BabiaXR que se pueden crear. Una vez seleccionado el tipo, aparece un nuevo menú sobre el anterior, desde el cual se elige el componente específico. Tras la selección, ambos menús desaparecen mediante una animación, y el ícono del componente elegido se desplaza hacia abajo, aumentando de tamaño para señalar el final de la fase de creación.

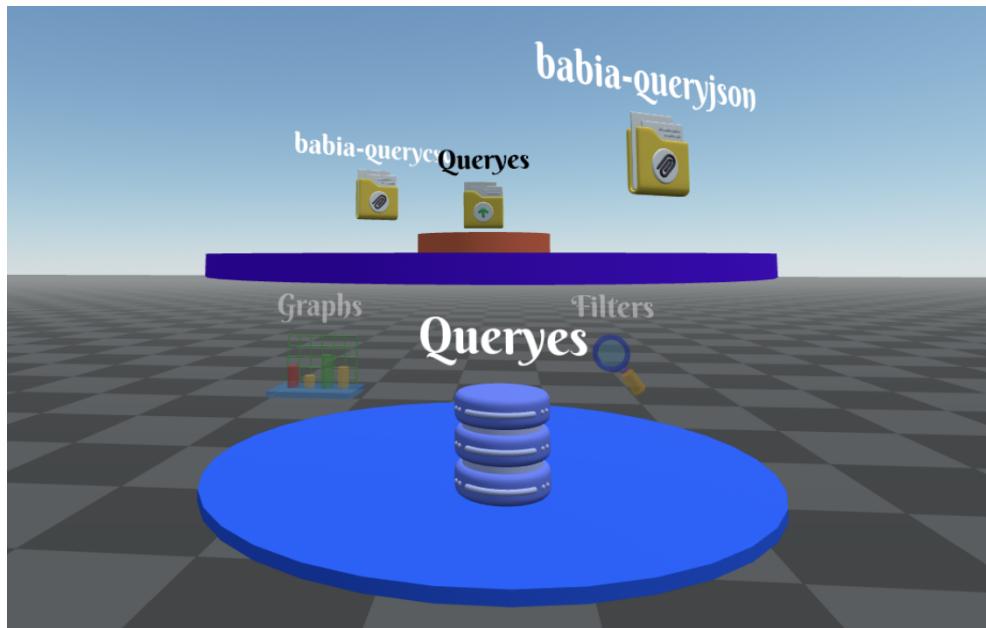


Figura 4.13: Escena de la etapa de creación del componente query.

En este punto, aparecen tres menús de opciones alrededor del ícono del componente creado. Cada menú tiene un único ícono rotando sobre el eje del disco que lleva el título del menú correspondiente. Estos menús son:

- **Menú de eliminación:** Al hacer clic en el ícono de eliminar, el componente y los menús desaparecen de la escena mediante una animación, y la entidad que contiene el componente se elimina por completo del DOM.
- **Menú de información:** El menú de información despliega un panel con una descripción detallada del componente creado, incluyendo las variables clave conectadas al menú de configuración. Estas variables varían según el tipo de componente y sus propiedades configurables.
- **Menú de configuración:** El menú de configuración mantiene su estructura original pero con una mejora, al actualizar las propiedades del componente, el panel de información también se actualiza automáticamente, proporcionando información actualizada en tiempo real.

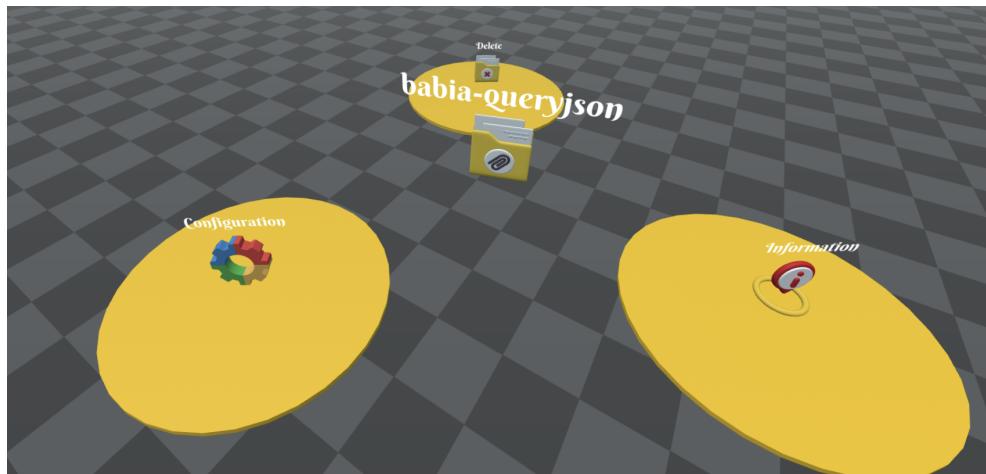


Figura 4.14: Escena con los tres menús de opciones.

Además, los tres menús pueden ocultarse o mostrarse de nuevo haciendo clic en el ícono del componente creado, lo que permite al usuario alternar entre las opciones y mantener una vista despejada de la escena cuando sea necesario.

Como parte de las mejoras visuales, se han añadido nuevos iconos 3D que reflejan con mayor claridad la función descrita por su título, facilitando la comprensión de cada menú. Asimismo, se ha incorporado una mini plataforma en el centro de cada disco que muestra el valor de la selección anterior, lo que ayuda al usuario a seguir el flujo de selecciones. Con todos estos cambios, la interfaz de usuario ha evolucionado hacia un sistema de visualización de datos, aunque aún no esté completamente terminada, superando la propuesta inicial y ofreciendo una experiencia más clara, coherente e intuitiva.

4.6.2. Implementar componentes de tipo filtro

Con la estructura del sistema de visualización de datos ya consolidada y la creación y configuración de componentes de tipo queries y gráficos completada, quedaba por implementar un último componente, los filtros. Estos componentes de tipo filtro dependen de la existencia de componentes query, ya que sin datos no se puede aplicar ningún filtro. Por lo tanto, al seleccionar un componente de tipo filtro en la etapa de creación, el sistema buscará todos los componentes de tipo query creados hasta ese momento. Tras seleccionar uno de esos archivos de datos, se generará el componente de filtro vinculado al archivo seleccionado, permitiendo así aplicar filtros sobre la información contenida en dicho archivo.

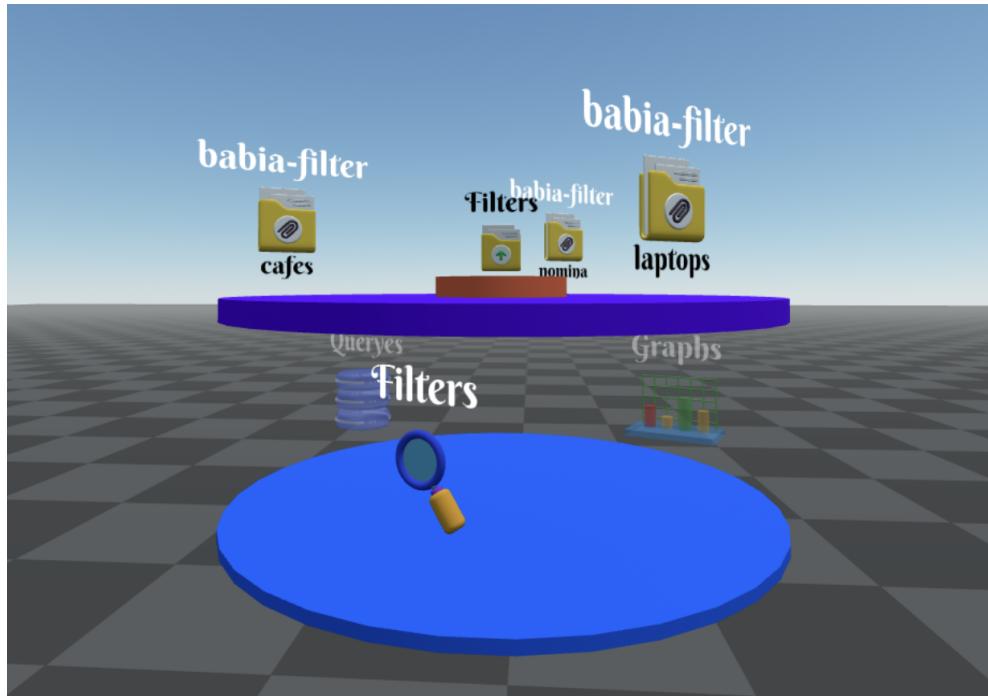


Figura 4.15: Escena de la etapa de creación del componente filtro.

En la etapa de configuración del componente filtro, el primer paso fue seleccionar un menú que muestra todos los campos disponibles del archivo de datos. Después, se elige el valor correspondiente a ese campo que servirá como filtro. Para que esto fuera posible, se implementó un algoritmo propio que primero identifica el formato del archivo de datos, ya sea JSON o CSV, y luego recorre el archivo completo, extrayendo tanto los campos como los valores únicos de cada uno de ellos. Esta información se almacena en una variable global que contiene todos los campos y sus posibles valores. Una vez que esta etapa está completa, el filtro puede configurarse. Tras configurar el filtro, el nombre del archivo de datos junto con el valor filtrado aparecerá debajo del componente creado, ofreciendo una referencia visual clara de los datos filtrados. Además, estas variables se actualizan automáticamente en el panel de información del menú correspondiente, manteniendo todo sincronizado y accesible para el usuario.

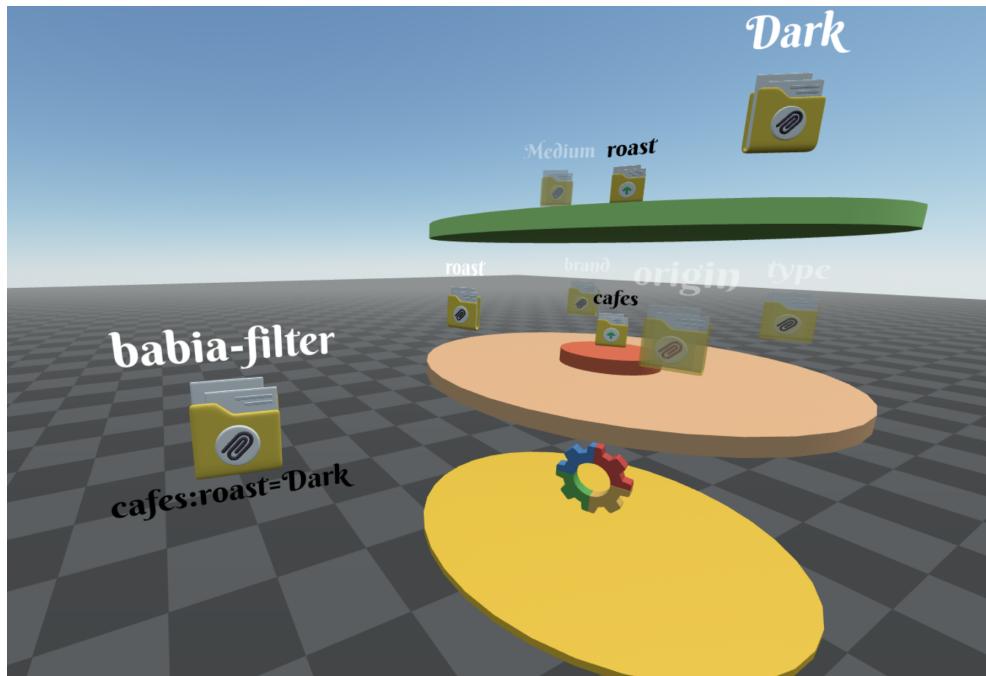


Figura 4.16: Escena del menú de configuración del componente filtro.

Una vez implementado el componente de filtro, fue necesario añadir nueva lógica para los componentes de gráficos, ya que estos dependen tanto de los componentes de tipo query como los tipos filtro. Esto permitió que las gráficas pudieran anclarse a un componente de tipo filtro, lo que posibilita la visualización de datos más refinados y específicos. De esta manera, los gráficos pueden representar no solo los datos generales, sino también aquellos que han sido filtrados, ofreciendo un análisis más detallado y personalizado. Esta funcionalidad mejora la capacidad de los usuarios para interactuar con conjuntos de datos complejos, proporcionando herramientas para explorar la información desde diferentes perspectivas.

4.7. Sprint 7: Sistema completo

Tras haber transformado la interfaz de usuario en un sistema de visualización de datos más complejo, este último sprint se centra en completar el sistema con la implementación de escenas multiusuario. Esta funcionalidad permite que múltiples usuarios interactúen con el sistema de visualización de datos en tiempo real y se comuniquen entre sí a través del micrófono. Además, se han integrado modelos 3D de avatares para los usuarios y controles más específicos tanto para el navegador web como para los controladores de las gafas Oculus. Con todas estas me-

joras implementadas, se da por finalizado el sistema de visualización de datos multiusuario en realidad virtual.

A continuación se detalla el plan de trabajo para este sprint:

■ **Objetivos:**

1. Incorporar escenas multiusuario.
2. Completar el sistema de visualización de datos.

■ **Actividades:**

1. Implementar Networked A-frame a la escena.
2. Agregar avatares y controles.

Duración: 1 mes

4.7.1. Implementar Networked A-frame a la escena

Con la estructura y lógica del sistema de visualización de datos completamente desarrollada, el último paso consistía en implementar escenas compartidas utilizando Networked A-Frame, permitiendo que múltiples usuarios participen simultáneamente en la misma escena. En este punto, un usuario puede crear, configurar, eliminar un componente y visualizar información sobre él. El objetivo de esta fase es permitir que múltiples usuarios realicen estas mismas acciones de forma simultánea, compartiendo la escena y observando en tiempo real lo que los demás están haciendo. Lo que antes se había explorado en sprints previos de formación con Networked A-Frame, ahora se aplicará esos conocimientos de manera más avanzada, posibilitando una interacción más compleja y fluida entre los usuarios dentro de la misma escena.

El componente clave para la compartición de la escena es networked, que permite sincronizar entidades entre múltiples usuarios. Cada entidad de la escena debe incluir este componente para que sea visible y manipulable por todos los usuarios conectados en tiempo real. Este componente requiere un "template" que defina cómo se replicará la entidad en las distintas instancias de la escena compartida. Inicialmente, se crean varios templates vacíos dentro de la etiqueta assets del archivo HTML, lo que permite mayor flexibilidad para agregar entidades dinámicamente según las necesidades del flujo de la lógica.

Si bien sería posible personalizar un template con entidades predefinidas y una estructura concreta, se optó por mantener los templates vacíos para añadir componentes de manera progresiva. De esta forma, todas las entidades en la escena llevan el componente networked junto con su template correspondiente. Dado que la escena utiliza una variedad de componentes, tanto personalizados como predefinidos, es necesario crear un archivo JavaScript que defina un "schema" por cada template. Esto es crucial, ya que por defecto networked-aframe solo sincroniza componentes básicos como la posición y rotación. Si se desea una sincronización completa de la escena en tiempo real, se deben incluir en estos schemas todos los componentes que cada entidad va a utilizar. Cualquier componente que no esté presente en el schema no será sincronizado y solo será visible para el usuario local.

Schema del template de los iconos de los menús:

```
NAF.schemas.getComponentsOriginal = NAF.schemas.getComponents;
NAF.schemas.getComponents = (template) => {
  if (!NAF.schemas.hasTemplate('#objectIcon-template')) {
    NAF.schemas.add({
      template: '#objectInit-template',
      components: [
        'position',
        'scale',
        'gltf-model',
        'animation',
        'component-synchronize',
        'babia-queryjson',
        'babia-querycsv',
        'babia-filter',
        'look-at',
      ]
    });
  }
  const components = NAF.schemas.getComponentsOriginal(template);
  return components;
}
```

Este proceso requirió una cantidad considerable de tiempo, ya que al principio no se tenía plena certeza sobre cómo implementar networked-aframe en una escena de esta complejidad. Fue necesario aprender su funcionamiento a fondo, recurriendo a un enfoque de prueba y error para asegurar que toda la escena pudiera ser compartida correctamente. En este punto, no se puso un enfoque especial en los avatares y controles de los usuarios. Para garantizar que todas las acciones e interacciones dentro de la escena se transmitieran en tiempo real, se realizaron

pruebas exhaustivas. La integración de networked-aframe resultó ser una de las tareas más desafiantes del proyecto. Primero, fue crucial entender con precisión cómo funcionaba esta tecnología antes de poder implementarla de manera efectiva. Aunque la lógica del proyecto ya estaba terminada, fue necesario abordar cada punto de forma meticulosa para integrar la compartición de escenas de manera exitosa. A pesar de las dificultades, finalmente se logró el objetivo de sincronizar las escenas en tiempo real.

4.7.2. Agregar avatares y controles

Inicialmente, los usuarios conectados a la escena solo contaban con una cámara en una entidad sin ningún avatar visible para los demás, lo cual fue mejorado en esta fase. Se realizó una búsqueda en línea para encontrar modelos 3D ligeros que se alinearan con la idea de avatares. Se desarrolló un algoritmo que asigna aleatoriamente un avatar 3D a cada usuario cuando se conecta a la escena, proporcionando una representación visual personalizada. También, se añadió un nombre de usuario único sobre cada avatar para que los demás participantes pudieran identificarlo fácilmente. Además, al usuario conectado se le incorpora el componente networked-audio-source, lo que permite la comunicación mediante el micrófono con otros usuarios en la escena.

Durante las pruebas, se detectó que los cuerpos de los avatares, al tener diferentes tamaños, no coincidían correctamente con la posición de la cámara, lo que afectaba a la inmersión. Para resolver esto, se ajustó manualmente la posición de la cámara en la cabeza de cada avatar, logrando así una experiencia más realista y coherente para todos los usuarios.

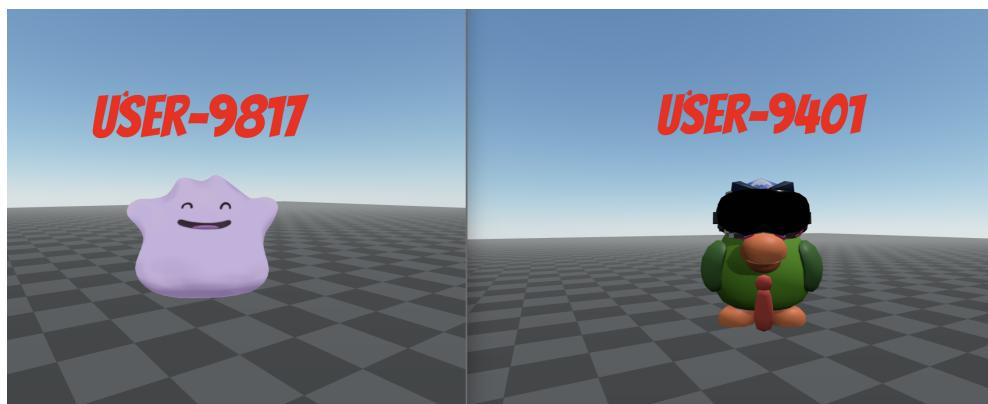


Figura 4.17: Escena con dos usuarios conectados.

Otra implementación significativa fue la activación de los controles para generar el menú inicial de creación, tanto para el navegador web como para la realidad virtual. Una vez completado el desarrollo del sistema de visualización de datos, se identificó la necesidad de crear un mecanismo para que el usuario pueda generar por sí mismo el menú inicial de creación. Este mecanismo puede generar tantos menús de creación como el usuario quiera. Por lo tanto, se implementó que en la versión del navegador web, al hacer clic en el espacio, se genera un menú inicial de creación frente a la vista del avatar, permitiendo la interacción a través de un clic. En el caso de la realidad virtual, se utilizó las gafas Oculus, lo que llevó a una investigación sobre cómo manipular los botones de los controladores VR. Se decidió que el menú de creación se activaría al presionar simultáneamente los botones B e Y de ambos controladores, y permitiendo la interacción con los objetos de la escena a través de los gatillos. Esta combinación de botones se implementó intencionalmente para profundizar en el control de los mecanismos de los controladores. Además, para garantizar que los usuarios comprendieran estos controles, se diseñó un panel informativo que proporciona las instrucciones necesarias. Este panel es lo primero que se muestra al usuario cuando se conecta por primera vez a la escena.

Capítulo 5

Conclusiones

Este capítulo finaliza el proyecto presentando un resumen de las conclusiones obtenidas y una reflexión sobre el proceso seguido. Se destaca el aprendizaje adquirido a lo largo del desarrollo y se evalúa el cumplimiento de los objetivos establecidos, ofreciendo además perspectivas para futuras mejoras y aplicaciones del sistema.

5.1. Consecución de objetivos

En esta sección, se evalúa si se han cumplido los objetivos planteados al inicio del proyecto. El objetivo general de desarrollar una interfaz de usuario para la visualización de datos en escenas compartidas en realidad virtual se ha alcanzado con creces. No solo se ha creado una interfaz funcional, sino que se ha desarrollado un sistema más complejo para la visualización de datos. Inicialmente, la interfaz de usuario tenía como propósito principal la creación y configuración de un componente de BabiaXR. Sin embargo, se implementaron acciones adicionales, como la eliminación de componentes del DOM y la obtención de información actualizable sobre el componente, en función de la configuración previa realizada por el usuario.

El objetivo de crear y configurar tres tipos de componentes de BabiaXR, query, filtro y gráfica, se logró completamente. El sistema de visualización de datos permite que los componentes de cualquier tipo puedan ser manipulados e integrados en la escena. Asimismo, se logró el objetivo de que la escena sea multiusuario, permitiendo a los participantes conectarse y observar en tiempo real las acciones de los demás. Cada jugador recibió un modelo de avatar 3D, acompañado de un nombre de usuario único. Además, se logró incorporar la funcionalidad de

comunicación entre los usuarios a través del micrófono.

Todo este proceso fue realizado según el objetivo de organizar el proyecto con estrategias basadas en metodologías ágiles. Y por último, se cumplió con la meta de asegurar la adaptabilidad de realidad virtual en múltiples dispositivos mediante el uso de A-Frame.

5.2. Aplicación de lo aprendido

Esta sección detalla cómo los conocimientos obtenidos a lo largo de las asignaturas de la carrera de Ingeniería en Sistemas Audiovisuales y Multimedia se han puesto en práctica en la realización de este proyecto.

- **Informática I y II:** Estas asignaturas me proporcionaron una base sólida en los principios fundamentales de la programación, como la lógica algorítmica, la resolución de problemas mediante código y orientación a objetos.
- **Construcción de servicios y aplicaciones audiovisuales en internet y Laboratorio de tecnologías audiovisuales en la Web:** Me permitieron profundizar en los lenguajes de programación web, con un enfoque especial en JavaScript, HTML5, CSS y la manipulación dinámica del DOM. También, adquirí conocimientos sobre el desarrollo de los mecanismos de interacción tanto del lado del servidor como del cliente.
- **Gráficos y Visualización en 3D:** Esta asignatura fue fundamental para dominar los principios de la visualización tridimensional. Además, me proporcionó un entendimiento sólido de WebGL y Three.js, herramientas esenciales para la creación y manipulación de gráficos 3D.
- **Arquitectura de internet y Sistemas telemáticos para medios audiovisuales:** Me proporcionaron una base sólida en el funcionamiento de redes de ordenadores e Internet, cubriendo en detalle las arquitecturas de protocolos esenciales para la comunicación de datos. Estos conocimientos fueron fundamentales para entender cómo opera el servidor de Networked A-Frame, permitiéndome comprender mejor la comunicación y el intercambio de datos en un entorno de realidad virtual multiusuario.

5.3. Lecciones aprendidas

A lo largo del desarrollo de este Trabajo Fin de Grado, se ha adquirido un profundo conocimiento de las tecnologías involucradas en la creación del sistema de visualización de datos multiusuario en realidad virtual.

El aprendizaje inicial incluyó una comprensión profunda del funcionamiento de la realidad virtual en navegadores web a través de A-Frame. A partir de esta tecnología, se adquirieron conocimientos esenciales sobre herramientas clave como WebGL, que permite el renderizado eficiente de gráficos 3D, WebXR, que facilita la integración de la realidad virtual en los navegadores, y Three.js, una librería fundamental para la creación de entornos tridimensionales interactivos. Otra tecnología base que se aprendió fue BabiaXR. Inicialmente, fue necesario entender a fondo el funcionamiento de esta librería, lo que implicó crear y experimentar con diversos componentes de visualización de datos para dominar su uso de manera efectiva.

Un reto significativo fue la integración de escenas multiusuario en el sistema. Esta implementación resultó ser particularmente compleja, requiriendo un extenso trabajo de perfeccionamiento y múltiples revisiones de la documentación para comprender correctamente cómo sincronizar las escenas entre varios usuarios. Durante la ejecución, fue necesario realizar una gran cantidad de pruebas en diferentes escenarios para verificar el comportamiento de la sincronización con otros usuarios conectados. El proceso de prueba y error fue clave para garantizar que todas las acciones del sistema resultante se sincronizasen correctamente en tiempo real entre todos los participantes.

Otro desafío importante fue garantizar que el sistema de visualización de datos funcionara sin problemas tanto en realidad virtual como en el navegador web. Mientras que la interacción con el ratón y el teclado en el navegador se comportaba como se esperaba, surgieron problemas al interactuar con los controladores de realidad virtual. Se tuvo que realizar un trabajo exhaustivo de pruebas para integrar correctamente los mandos de Oculus, asegurando que los controladores pudieran interactuar con los menús del sistema de visualización. A través de estas pruebas, se adquirió un conocimiento avanzado sobre cómo hacer que los controladores reconocieran y seleccionaran objetos en la escena, perfeccionando la interacción y logrando un control preciso sobre los botones de los mandos para implementar la lógica deseada.

En cuanto a la programación, se ha fortalecido notablemente el dominio de JavaScript, el

lenguaje principal de A-Frame, así como de HTML para la estructuración de las escenas. Todos los componentes desarrollados para el sistema fueron escritos en JavaScript, lo que requirió alcanzar un nivel avanzado en la escritura de código para implementar con éxito la lógica propuesta.

Estas lecciones aprendidas han sido fundamentales en la culminación de un sistema robusto y eficiente, proporcionando una valiosa experiencia tanto a nivel técnico como en la resolución de problemas en entornos de realidad virtual.

5.4. Trabajos futuros

En esta sección se sugieren nuevas ideas y direcciones que podrían ampliar las capacidades del proyecto.

- **Servidor en línea:** Implementar un servidor online en Networked A-Frame para permitir a los usuarios conectarse a escenas de realidad virtual desde cualquier ubicación, facilitando una experiencia multijugador global en la visualización de datos.
- **Expansión de fuentes de datos:** Ampliar la interfaz para permitir la extracción de datos desde ElasticSearch y de repositorios GitHub, enriqueciendo así las opciones de visualización y análisis.
- **Ampliación de componentes de visualización:** Incorporar nuevos tipos de gráficos para la visualización de datos, como gráficos de cilindros, burbujas y mapas, para brindar una representación más rica y detallada de los datos.
- **Incorporación de interacción con gestos:** Habilitar la interacción con los menús del sistema de visualización a través de gestos manuales mientras se utilizan las gafas de realidad virtual.
- **Mejora de soporte para realidad aumentada:** Optimizar la compatibilidad con tecnologías de realidad aumentada, permitiendo que el sistema de visualización de datos se muestre correctamente en entornos mixtos.

Bibliografía

[1] A-Frame. Documentación web oficial:

<https://aframe.io/docs/1.6.0/introduction/>

[2] Three.js. Documentación web oficial:

<https://threejs.org/>

[3] WebGL. Documentación estándar WebGL:

<https://registry.khronos.org/webgl/specs/latest/1.0/>

[4] WebXR. Documentación web disponible:

https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API

[5] BabiaXR. Documentación web oficial:

<https://babiaxr.gitlab.io/>

[6] Networked-Aframe. Documentación web disponible:

<https://github.com/networked-aframe/networked-aframe>

[7] HTML5. Documentación web disponible:

<https://www.w3.org/TR/2011/WD-html5-20110405/>

[8] Latex. Documentación web oficial:

<https://www.latex-project.org/>

[9] Visual Studio Code. Documentación web oficial:

<https://code.visualstudio.com/docs>

[10] Flanagan, D. *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media, 2020.

- [11] Pilgrim, M. *HTML5: Up and Running* (1st ed.). O'Reilly Media, 2010.
- [12] Lamport, L. *LaTeX: A Document Preparation System* (2nd ed.). Addison-Wesley, 1994.
- [13] Chacon, S., & Straub, B. *Pro Git* (2nd ed.). Apress, 2014.
- [14] Dirksen, J. *Learn Three.js* (4th ed.). Packt Publishing, 2023.
- [15] Parisi, T. *Programming 3D Applications with HTML5 and WebGL* (1st ed.). O'Reilly Media, 2014.