ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA TELEMÁTICA

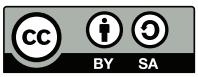
TRABAJO FIN DE GRADO

HABLA Y CREA: ESCENARIOS VR EN EL NAVEGADOR USANDO A-FRAME Y LENGUAJE NATURAL

Autor: Veselin Todorov Hristov

Tutor: Jesús María González Barahona

Curso académico 2024/2025



A mi hermano,

mi primer maestro y eterno referente.

Y a mis padres,

por convertir sus sueños rotos en el suelo firme

donde hoy construyo mi futuro.

Agradecimientos

Este proyecto no habría sido posible sin el apoyo, la compañía y la inspiración de muchas personas que han dejado huella en mi camino.

En primer lugar, gracias a mi familia, que ha sido siempre mi mayor sostén. A mi madre, por su paciencia inagotable y su forma de estar siempre, incluso en silencio. A mi padre, por enseñarme el valor del esfuerzo, de hacer las cosas bien y con constancia. Y a mi hermano, por su ejemplo, su presencia y sus consejos, que siempre llegan en el momento justo.

A mis compañeros de carrera, con quienes he compartido no solo conocimientos, sino también risas, agobios y aprendizajes que van mucho más allá de lo académico. A mis amigos, los de siempre y los nuevos, por ser esa red invisible que sostiene, incluso cuando uno no se da cuenta.

Agradezco especialmente a mi tutor de TFG, por su guía, su disponibilidad y su confianza. Gracias por orientarme sin imponer, por darme libertad y al mismo tiempo estructura, y por tomarse en serio mis ideas.

También quiero extender mi gratitud a todas aquellas personas que, de una manera u otra, han contribuido a que hoy me encuentre aquí, en este punto. Algunas ni siquiera lo sabrán, pero su influencia ha sido decisiva.

Y por último, aunque de forma discreta, gracias a quien me acompaña de forma cercana en lo cotidiano, en lo pequeño y lo grande, con una presencia que reconforta incluso sin palabras.

A todos vosotros: gracias de corazón.

Resumen

Este trabajo presenta una investigación sobre la integración de tecnologías innovadoras para el control de entornos de realidad virtual mediante comandos de voz en lenguaje natural. Los resultados obtenidos demuestran la viabilidad de desarrollar interfaces adaptativas que simplifican la interacción inmersiva, ofreciendo una alternativa más intuitiva frente a los sistemas tradicionales basados en dispositivos físicos.

La solución propuesta combina realidad virtual con modelos de inteligencia artificial capaces de interpretar lenguaje natural, traduciendo órdenes verbales en acciones dentro del entorno tridimensional. El sistema incorpora, además, funcionalidades básicas de manipulación directa mediante *drag & drop*, lo que permite una experiencia de usuario híbrida más versátil y accesible.

Esta integración tecnológica representa un avance significativo en las interfaces humano-máquina aplicadas a entornos de realidad virtual, estableciendo un marco de desarrollo flexible, escalable y adaptable a distintos contextos. Entre sus posibles aplicaciones destacan los ámbitos educativos, el diseño profesional y los entornos industriales, donde la interacción natural con objetos virtuales puede aportar mejoras sustanciales en productividad y accesibilidad.

Summary

This study presents an investigation on the integration of innovative technologies for controlling virtual reality environments through natural language voice commands. The obtained results demonstrate the feasibility of developing adaptive interfaces that simplify immersive interaction, offering a more intuitive alternative compared to traditional physical device-based systems.

The proposed solution combines virtual reality with artificial intelligence models capable of natural language processing, translating verbal commands into actions within the three-dimensional environment. The system additionally incorporates basic direct manipulation functionalities through *drag & drop*, enabling a more versatile and accessible hybrid user experience.

This technological integration represents a significant advancement in human-machine interfaces for virtual reality applications, establishing a flexible, scalable and adaptable development framework. Among its potential applications, it particularly stands out in educational settings, professional design, and industrial environments, where natural interaction with virtual objects can provide substantial improvements in productivity and accessibility.

Índice general

Índice de figuras

Capítulo 1

Introducción

En los últimos años, la realidad virtual (VR) ha dejado de ser una tecnología exclusiva de sectores industriales o de investigación para comenzar a consolidarse en ámbitos como la educación, el entretenimiento, la arquitectura o incluso la formación profesional. Este crecimiento se ha visto favorecido por la progresiva reducción del coste de los dispositivos, el aumento de su disponibilidad comercial, y la creciente inversión por parte de grandes empresas tecnológicas. Paralelamente, el avance de tecnologías basadas en inteligencia artificial, como los modelos de lenguaje (LLM) y los sistemas de reconocimiento de voz, ha abierto nuevas posibilidades para repensar cómo los usuarios pueden interactuar con los entornos digitales.

Este Trabajo de Fin de Grado[1] se sitúa precisamente en la confluencia de estos dos campos emergentes: la realidad virtual accesible desde navegadores web y la interacción mediante lenguaje natural. El objetivo principal es explorar un cambio de paradigma en el diseño de interfaces para VR, proponiendo una solución que sustituye el uso de mandos físicos o dispositivos tradicionales por comandos de voz, entendidos y ejecutados mediante inteligencia artificial. Esta exploración se realiza utilizando tecnologías abiertas, ampliamente accesibles, con el propósito de que cualquier persona —sin necesidad de equipos especializados o conocimientos técnicos avanzados— pueda experimentar con entornos tridimensionales inmersivos.

El proyecto adopta como requisito fundamental su funcionamiento en navegadores web, lo

[1] URL del Trabajo de Fin Grado: https://vescoth.github.io/tfg-site/

que elimina la necesidad de instalaciones complejas y refuerza su carácter accesible. Para ello, se ha utilizado el framework A-Frame sobre WebXR, junto con APIs de captura y procesamiento de voz (MediaStream Recording API) y servicios avanzados de transcripción y generación de texto mediante modelos LLM. El resultado es una interfaz conversacional que permite crear, transformar y gestionar objetos 3D dentro de un entorno VR, utilizando lenguaje hablado como canal principal de interacción.

Este rediseño de la interfaz implica superar retos técnicos y conceptuales: ¿cómo debe estructurarse una interfaz que "escucha" de forma continua?, ¿cómo garantizar que las órdenes se interpreten correctamente?, ¿cómo lograr una experiencia fluida y comprensible para el usuario? El presente trabajo aborda estas preguntas desde un enfoque práctico, desarrollando un sistema funcional que combina control por voz con manipulación directa mediante gestos (drag & drop), ofreciendo una experiencia híbrida más versátil.

Cabe destacar que, hasta la fecha, no existen soluciones previas que integren esta combinación de tecnologías emergentes de manera funcional y accesible. Esta propuesta representa, por tanto, una primera exploración del potencial de la voz como canal principal de interacción en entornos inmersivos tridimensionales, orientada a un público general. El valor del trabajo radica no solo en su componente técnico, sino también en abrir nuevas vías hacia la democratización del diseño 3D, mediante herramientas intuitivas al alcance de cualquier persona.

1.1. Objetivos

El objetivo principal es explorar cómo podría ser un editor de escenas 3D en realidad virtual donde la voz es el medio principal de interacción, utilizando las capacidades que ofrece el navegador web para construir un sistema funcional y accesible.

Los **objetivos instrumentales** derivados son:

- Implementar un editor interactivo que permita crear y modificar escenas de realidad virtual mediante comandos de voz procesados por inteligencia artificial, demostrando cómo estas tecnologías pueden trabajar conjuntamente de manera eficiente.

- Desarrollar una solución para que funcione directamente en navegadores web, aprovechando sus funcionalidades nativas para grabación y procesamiento de audio.
- Interpretar el lenguaje natural de manera general y flexible, sin limitar las posibles órdenes a formatos rígidos, facilitando así una interacción cercana a una conversación real.
- Integrar funcionalidades complementarias, como la manipulación manual mediante *drag & drop*, que permitan mayor precisión en el posicionamiento y edición de objetos.
- Mejorar la accesibilidad, asegurando que usuarios sin experiencia técnica puedan utilizar el sistema con facilidad.

Como **objetivos secundarios** se plantean:

- Realizar pruebas a bajo coste para validar la viabilidad del sistema.
- Garantizar independencia tecnológica frente a posibles variaciones en los servicios externos de inteligencia artificial.
- Justificar las decisiones tecnológicas adoptadas a lo largo del desarrollo, explicando claramente por qué se eligió un método o herramienta en particular.

1.2. Contribuciones

Las contribuciones principales de este trabajo se resumen en:

- Diseño e implementación de un sistema de control por voz para entornos de realidad virtual basados en navegador, integrando tecnologías modernas como A-Frame y modelos de lenguaje alojados en servicios en la nube.
- Desarrollo de una interfaz conversacional que permite manipular objetos 3D mediante lenguaje natural, eliminando la necesidad de conocimientos técnicos avanzados.

- Programación de componentes personalizados en A-Frame que soportan transformaciones complejas sobre modelos tridimensionales a partir de instrucciones verbales interpretadas.
- Introducción de la funcionalidad *drag & drop* para posicionamiento preciso mediante un cursor central controlado por el movimiento del ratón o de la cabeza en VR, complementando el control por voz y mejorando la experiencia del usuario.
- Creación de un sistema de almacenamiento que permite persistir las escenas generadas como archivos reutilizables.
- Implementación de un mecanismo de activación por palabra clave que optimiza el uso de recursos y mejora la interacción.

Este trabajo demuestra la viabilidad de un sistema de edición de escenas VR basado en voz, construido con tecnologías accesibles y aplicables en diversos contextos educativos, creativos y profesionales.

1.3. Estructura del documento

Para comprender plenamente el diseño e implementación del sistema, es necesario conocer las tecnologías y conceptos en los que se apoya. Por ello, el documento se organiza de la siguiente manera:

- **Capítulo 2: Tecnologías utilizadas y trabajos relacionados**. Presenta una revisión de las tecnologías clave utilizadas en este trabajo, incluyendo A-Frame, WebXR, servicios de reconocimiento de voz y modelos de lenguaje, así como trabajos previos relacionados.
- **Capítulo 3: Desarrollo del proyecto**. Este capítulo documenta el proceso de implementación del sistema de control por voz en VR, detallando la arquitectura técnica, decisiones de diseño justificadas y la metodología basada en sprints.

- **Capítulo 4: Descripción del resultado**. Describe el sistema funcional de edición VR por voz, detallando: (1) la interacción natural del usuario mediante comandos hablados, (2) los componentes técnicos implementados (APIs, modelos de IA y entornos VR), y (3) cómo integrarlos para crear aplicaciones adaptables a diversos contextos, demostrando su versatilidad en uso real.

- **Capítulo 5: Pruebas y validación**. Analiza el rendimiento, la usabilidad y la funcionalidad del sistema, y discute sus limitaciones actuales junto con posibles mejoras.

- **Capítulo 6: Conclusiones**. Resume los principales resultados obtenidos, las contribuciones realizadas y plantea líneas futuras de investigación y desarrollo.

Capítulo 2

Tecnologías utilizadas y trabajos relacionados

En este capítulo se presenta el conjunto de tecnologías que han hecho posible el desarrollo del sistema, así como algunos trabajos previos que han servido de referencia o inspiración. El objetivo es ofrecer una visión global del contexto técnico en el que se inscribe el proyecto, justificando las decisiones adoptadas en su implementación.

El sistema desarrollado se basa en la combinación de técnicas de realidad virtual accesible desde navegador, interfaces naturales basadas en voz y procesamiento inteligente del lenguaje. Para ello, se han utilizado múltiples herramientas y marcos tecnológicos interrelacionados, que abarcan desde la construcción de entornos 3D hasta la transcripción de voz y la interpretación semántica de comandos hablados.

Uno de los pilares fundamentales del proyecto es el uso de modelos de lenguaje de gran escala, conocidos como **LLMs**[23] (Large Language Models). Estos modelos, entrenados sobre grandes volúmenes de texto, permiten interpretar instrucciones expresadas en lenguaje natural y generar respuestas coherentes y contextualizadas. En este caso, se ha utilizado un modelo de código abierto de la familia **Gemma 2**[4] (concretamente, Gemma 2 9B), que ofrece capacidades competitivas en tareas de comprensión y razonamiento, manteniendo al mismo tiempo una arquitectura eficiente y adecuada para su uso mediante API.

Tanto el modelo de transcripción automática de voz como el modelo de lenguaje se han utilizado a través de un proveedor de infraestructura que permite la ejecución remota de modelos abiertos mediante una interfaz REST. Esta solución ha facilitado el desarrollo sin necesidad de contar con hardware especializado, además de ofrecer latencias muy reducidas y acceso gratuito en el contexto del proyecto.

Para capturar y enviar el audio del usuario, se ha utilizado la **MediaStream Recording API**[10], una interfaz nativa del navegador que permite grabar audio desde el micrófono de forma sencilla y controlada.

La escena tridimensional se ha construido empleando el framework **A-Frame**[15], que permite definir experiencias de realidad virtual en la web mediante una sintaxis declarativa basada en HTML. Este se apoya internamente en **Three.js**[19], una biblioteca JavaScript ampliamente utilizada para la renderización y manipulación de gráficos 3D en entornos web. Toda la lógica de interacción, comunicación con los modelos y edición dinámica de la escena ha sido implementada íntegramente en **JavaScript**[22].

La interfaz se estructura mediante tecnologías estándar de la web como **HTML5**[12], permitiendo una integración fluida entre los elementos visuales, el comportamiento interactivo y los componentes personalizados definidos en A-Frame. El desarrollo se ha realizado en el entorno **Visual Studio Code**[14], sobre un sistema operativo Windows 10, utilizando el navegador Microsoft Edge para la prueba e iteración del sistema.

Durante el desarrollo, se han utilizado asistentes de inteligencia artificial como ChatGPT o DeepSeek para resolver dudas técnicas puntuales, depurar fragmentos de código o explorar soluciones alternativas. No obstante, el diseño, la arquitectura y la implementación completa del sistema han sido realizados de forma independiente.

En las siguientes secciones se describen con mayor detalle las tecnologías mencionadas, así como varios trabajos previos relacionados con interfaces en realidad virtual controladas por voz, sistemas de interacción por lenguaje natural y editores de entornos tridimensionales en web, con el fin de contextualizar y valorar la originalidad de la propuesta desarrollada.

2.1. Tecnologías utilizadas

2.1.1. A-Frame

A-Frame [1] es un framework de código abierto desarrollado por Mozilla, orientado a la creación de experiencias inmersivas en realidad virtual (VR) y aumentada (AR) directamente desde navegadores web. Está construido sobre la biblioteca Three.js y se caracteriza por ofrecer una sintaxis declarativa basada en HTML, lo que facilita el desarrollo de escenas tridimensionales complejas sin requerir conocimientos avanzados de gráficos 3D o WebGL[7].

Una de las principales fortalezas de A-Frame es su capacidad de ser extendido mediante componentes personalizados, lo que permite incorporar comportamientos específicos y reutilizables, como transformaciones geométricas, animaciones o interacciones. Además, su compatibilidad con el estándar WebXR[11] garantiza que los entornos creados puedan ejecutarse en una amplia variedad de dispositivos VR y AR sin necesidad de instalar software adicional.

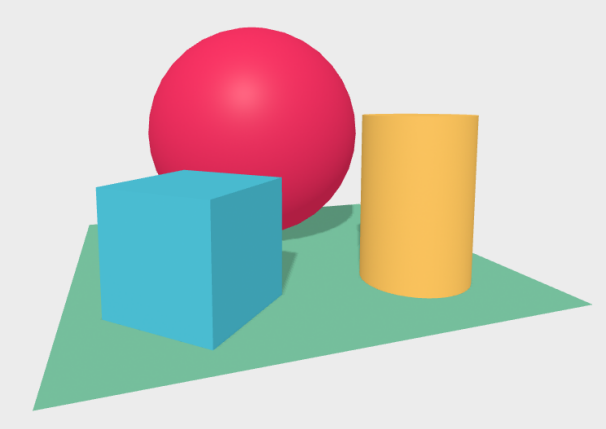


Figura 2.1: Escena básica de A-frame

A-Frame permite definir entidades 3D como primitivas HTML, combinando simplicidad y potencia. Por ejemplo, una escena básica puede incluir cámaras, luces y objetos 3D de forma intuitiva:

[1] Ejemplos A-frame: https://aframe.io/

```
<a-scene>
  <!-- Objeto 3D -->
  <a-box position="0 1 -3" rotation="0 45 0" color="#4CC3D9"></a-box>

  <!-- Cielo -->
  <a-sky color="#ECECEC"></a-sky>

  <!-- Cámara -->
  <a-entity camera look-controls position="0 1.6 0"></a-entity>

  <!-- Luz ambiental -->
  <a-entity light="type: ambient; intensity: 0.5"></a-entity>

  <!-- Luz direccional -->
  <a-entity light="type: directional; intensity: 0.8" position="1 3 1"></
</a-scene>
```

Esta expresividad ha sido clave para este proyecto, en el que se requiere la creación de un espacio tridimensional manipulable por comandos de voz. La elección de A-Frame responde a su capacidad para permitir:

- Representación de modelos 3D interactivos en tiempo real.
- Integración sencilla con bibliotecas JavaScript personalizadas.
- Compatibilidad con dispositivos VR accesibles desde navegador.
- Extensibilidad mediante componentes definidos por el desarrollador.

En el sistema desarrollado, A-Frame constituye la base para construir la escena 3D principal, gestionar la cámara del usuario y permitir la interacción con modelos a través de eventos personalizados. La facilidad para extender su funcionalidad ha permitido adaptar la experiencia a un entorno controlado por lenguaje natural.

2.1.2. Three.js

Three.js[2][19] es una biblioteca JavaScript que proporciona una abstracción de alto nivel sobre la API WebGL, simplificando el desarrollo de gráficos tridimensionales interactivos en el navegador. A través de una colección de clases para geometrías, cámaras, luces, materiales y animaciones, permite construir escenas complejas con relativa facilidad.

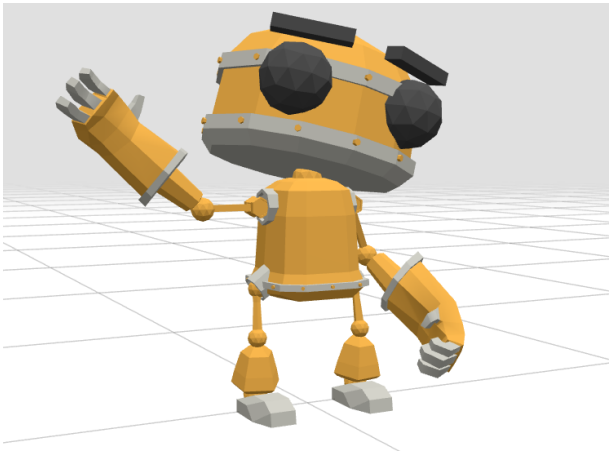


Figura 2.2: Modelo animado de un robot hecho con Three.js

Aunque A-Frame abstrae la mayoría de las complejidades de Three.js, este último sigue siendo el motor gráfico subyacente que realiza las operaciones de renderizado. Es decir, cada objeto definido en A-Frame se traduce internamente en entidades de Three.js que son renderizadas mediante WebGL.

En este proyecto, Three.js se utiliza únicamente para implementar la funcionalidad de *drag & drop*, pero su papel es fundamental: cualquier transformación aplicada a los objetos en la escena (como rotaciones o escalados), cualquier animación o cambio de posición, es ejecutado por medio de esta biblioteca. Por tanto, la estabilidad, rendimiento y compatibilidad multiplataforma del sistema se sustentan en la solidez de Three.js.

[2]Ejemplos Three.js: `https://threejs.org/`

2.1.3. WebXR

WebXR[3][2] es una API estandarizada por el W3C que permite desarrollar experiencias de realidad virtual (VR) y aumentada (AR) directamente desde el navegador. Proporciona una interfaz común para acceder a dispositivos de visualización inmersiva —como gafas de realidad virtual o teléfonos móviles con sensores de movimiento— mediante tecnologías web como HTML y JavaScript.

Entre sus características principales destacan:

- Acceso multiplataforma a hardware XR sin necesidad de plugins.
- Soporte para seguimiento espacial del usuario y de los controladores.
- Compatibilidad con navegadores modernos como Chrome, Firefox o Edge.
- Posibilidad de crear experiencias inmersivas completamente desde el navegador.

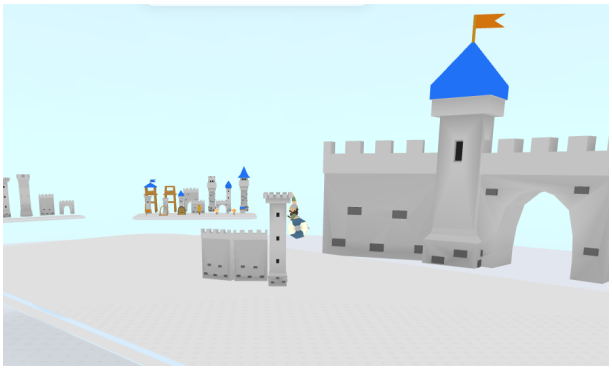


Figura 2.3: Constructor de castillos usando WebXR

En este proyecto, WebXR es la base que permite que el framework **A-Frame** funcione en dispositivos de realidad virtual. A-Frame proporciona una capa declarativa y de alto nivel para construir entornos inmersivos, pero es WebXR quien permite la conexión con el hardware, detectando si el usuario accede desde un visor VR y activando el modo inmersivo. Gracias a

[3]Demos WebXR: https://immersiveweb.dev/

esta integración, la misma escena HTML puede experimentarse tanto en un PC como dentro de unas gafas VR, adaptando automáticamente las capacidades de interacción.

Esta tecnología es esencial para cumplir el objetivo de ofrecer una experiencia multiplataforma sin necesidad de instalaciones ni configuraciones adicionales por parte del usuario.

2.1.4. WebGL

WebGL[4][13] (Web Graphics Library) es una API estándar basada en JavaScript que permite renderizar gráficos 2D y 3D acelerados por hardware directamente en el navegador, sin necesidad de plugins. Está desarrollada y mantenida por el consorcio Khronos Group, y se basa en una versión simplificada de OpenGL ES 2.0.

WebGL permite la ejecución de gráficos interactivos de alta calidad utilizando el hardware gráfico del dispositivo (GPU), lo cual es esencial para el desarrollo de experiencias inmersivas y visualmente exigentes dentro de entornos web.

Entre sus características más destacadas se incluyen:

- Acceso directo al hardware gráfico desde el navegador mediante shaders.
- Compatibilidad con todos los navegadores modernos (Chrome, Firefox, Edge, Safari).
- Integración nativa con HTML5 y el lienzo `<canvas>`.
- Posibilidad de crear escenas 3D complejas y animaciones en tiempo real.

En el contexto de este proyecto, WebGL constituye el motor de renderizado subyacente sobre el que A-Frame y Three.js construyen sus abstracciones. Aunque el desarrollador no interactúa directamente con WebGL, su presencia es crucial: permite que las entidades 3D definidas en HTML mediante A-Frame sean finalmente dibujadas y animadas en el lienzo del navegador con fluidez y eficiencia. De este modo, WebGL es el pilar gráfico que hace posible representar

[4]Página oficial de WebGL: https://www.khronos.org/webgl/

modelos, aplicar transformaciones en tiempo real y garantizar una experiencia visualmente rica y responsiva.

2.1.5. Whisper

Whisper[5][18] es un modelo de reconocimiento automático del habla (ASR) basado en una arquitectura **transformer encoder-decoder** entrenada de extremo a extremo. El proceso comienza con la conversión de la señal de audio en un **espectrograma de log-potencia**, que captura la distribución de energía en distintas bandas de frecuencia a lo largo del tiempo. Este espectrograma sirve como entrada al codificador (encoder), que produce una representación latente del contenido acústico utilizando bloques de atención multi-cabeza y normalización por capas.

El decodificador **(decoder)**, por su parte, es un generador autoregresivo de texto que predice una secuencia de tokens codificados (basados en el vocabulario del modelo, como en GPT). Durante la inferencia, el decoder genera un token a la vez, utilizando como contexto la representación latente del audio producida por el encoder y los tokens previos generados. De este modo, produce una transcripción textual alineada con el contenido del audio.

Whisper ha sido entrenado con un dataset masivo de **680.000 horas de audio multilingüe** con tareas supervisadas como transcripción, traducción, detección de idioma y segmentación.

Entre sus características destacadas se encuentran:

- Soporte para más de 50 idiomas, incluyendo español.
- Robustez ante ruido, acentos y variaciones en el ritmo del habla.
- Capacidad de segmentar audio largo y detectar cambios de idioma automáticamente.

Whisper es especialmente adecuado para este proyecto debido a que:

[5]Ejemplo Whisper: https://openai.com/es-ES/index/whisper/

- Permite el control del sistema mediante comandos de voz sin necesidad de entrenamiento personalizado.
- Aumenta la accesibilidad de la aplicación a usuarios hispanohablantes.
- Reduce el riesgo de errores de transcripción gracias a su diseño robusto.

El modelo se ha integrado a través de una API externa (Groq) que permite enviar fragmentos de audio grabado y recibir como respuesta su transcripción en texto plano. Esta transcripción es luego utilizada para inferir la intención del usuario.

2.1.6. MediaStream Recording API

La *MediaStream Recording API*[10] es una interfaz estándar del navegador definida por HTML5, que permite capturar flujos de audio y video de forma programática. Su principal ventaja es que funciona directamente desde cualquier navegador moderno sin necesidad de extensiones o software adicional.

Esta API permite:

- Acceder al micrófono del usuario con permisos explícitos.
- Grabar y codificar el audio en tiempo real en formatos como WebM o WAV.
- Controlar la duración, eventos y flujo de grabación mediante JavaScript.

Técnicamente, el proceso comienza accediendo a un dispositivo de entrada mediante *navigator.mediaDevices.getUserMedia()*, que retorna una promesa con un objeto *MediaStream*. Este objeto representa el flujo en tiempo real del micrófono y puede ser conectado a un *MediaRecorder*, que codifica el audio conforme al formato y códec especificados por el navegador (normalmente Opus dentro de un contenedor WebM).

El `MediaRecorder` emite eventos como *dataavailable* o *stop*, desde los cuales se pueden recolectar los fragmentos de audio codificados (blobs). Estos blobs pueden concatenarse o

procesarse directamente para su almacenamiento o envío a un servidor. Además, la API permite establecer un intervalo de muestreo mediante *timeslice* para obtener fragmentos parciales periódicamente si se desea.

En este proyecto se utiliza para grabar la voz del usuario mientras interactúa con la escena tridimensional. La grabación es procesada como un *blob* de audio y enviada al backend para su posterior transcripción con Whisper. Esta integración permite un flujo interactivo natural: el usuario activa la grabación con un gesto, pronuncia su orden, y el sistema responde en función del análisis posterior.

2.1.7. Groq

Groq[8] es una plataforma que proporciona acceso a modelos de inteligencia artificial de código abierto mediante una API compatible con la interfaz de OpenAI. Esto permite integrar modelos como Gemma o Whisper fácilmente en herramientas ya diseñadas para OpenAI, sin necesidad de modificar el código base. Aunque su API replica la estructura de OpenAI, los modelos se ejecutan en su propia infraestructura, optimizada mediante unidades especializadas (Language Processing Units) para ofrecer baja latencia.

MODEL ID	DEVELOPER	CONTEXT WINDOW (TOKENS)	MAX COMPLETION TOKENS	MAX FILE SIZE	DETAILS
distil-whisper-large-v3-en	Hugging Face	-	-	100 MB	Details
gemma2-9b-it	Google	8,192	8,192	-	Details
llama-3.1-8b-instant	Meta	131,072	131,072	-	Details
llama-3.3-70b-versatile	Meta	131,072	32,768	-	Details
meta-llama/llama-guard-4-12b	Meta	131,072	1,024	20 MB	Details
whisper-large-v3	OpenAI	-	-	100 MB	Details
whisper-large-v3-turbo	OpenAI	-	-	100 MB	Details

Figura 2.4: Modelos disponibles en Groq para producción

En este proyecto se ha optado por Groq principalmente por su compatibilidad técnica y por la posibilidad de utilizar sus servicios de manera gratuita, lo cual ha permitido desarrollar e iterar sin restricciones presupuestarias, aunque podrían haberse considerado otras alternativas similares.

El acceso a los modelos se realiza mediante peticiones HTTP, con un flujo simplificado como el siguiente:

```
POST https://api.groq.com/openai/v1/chat/completions
Authorization: Bearer <API_KEY>
Content-Type: application/json

{
  "model": "gemma2-9b-it",
  "messages": [...],
  "temperature": 0.2
}
```

En este trabajo, Groq aloja dos servicios fundamentales:

- El modelo de transcripción Whisper, que convierte el audio del usuario en texto.
- El modelo de lenguaje natural Gemma 2 9B, que interpreta las intenciones a partir del texto transcrito.

Su integración mediante peticiones HTTP facilita el desarrollo y despliegue del sistema sin necesidad de gestionar infraestructura local o servidores complejos.

2.1.8. Gemma 2 9B

Gemma 2 9B[4] es un modelo de lenguaje de código abierto desarrollado por Google DeepMind. Se trata de una arquitectura basada en transformadores optimizada para generación e interpretación de lenguaje natural, especialmente útil en tareas como generación de código, respuestas conversacionales o análisis de intención.

Este modelo se entrena sobre grandes volúmenes de datos textuales y está afinado para seguir instrucciones (instruction tuning), lo que lo hace particularmente adecuado para interfaces controladas por comandos naturales.

Entre sus características técnicas destacan:

- 9.4 mil millones de parámetros.
- Versiones optimizadas para inferencia rápida.
- Entrenamiento en múltiples idiomas, incluyendo español.
- Soporte para formatos estructurados como JSON.

En este proyecto, Gemma se emplea para procesar el texto transcrito de la voz del usuario. A través de un *prompt* estructurado, se le instruye para generar respuestas en formato JSON que

representan acciones concretas sobre modelos tridimensionales: rotar, escalar, duplicar, eliminar, etc.

Esta respuesta es posteriormente interpretada por el sistema y ejecutada mediante JavaScript sobre la escena 3D.

2.1.9. Meta Quest 3

Las Meta Quest 3 son un visor de realidad virtual autónomo desarrollado por Meta, que ofrece capacidades avanzadas para experiencias inmersivas sin necesidad de un ordenador o cables externos. Este dispositivo se ha utilizado como principal visor para el desarrollo del proyecto debido a sus características técnicas y facilidades de integración.

Figura 2.5: Gafas de Realidad Virtual Meta Quest 3

Entre sus principales ventajas destacan:

- **Micrófono integrado**: Permite la captura de voz del usuario directamente desde el dispositivo, facilitando la interacción por comandos de voz sin necesidad de hardware adicional.
- **Seguimiento preciso**: Incluye sensores avanzados para rastrear movimientos de la cabeza y las manos, mejorando la interacción natural dentro del entorno virtual.
- **Compatibilidad con WebXR**: Soporta estándares web para realidad virtual y aumentada, lo que permite ejecutar aplicaciones desarrolladas con frameworks como A-Frame de

forma nativa.

- **Portabilidad y comodidad**: Su diseño autónomo y ligero permite al usuario moverse libremente sin cables, incrementando la inmersión y usabilidad.

El uso de las Meta Quest 3 en este proyecto facilita la integración de captura de voz, manipulación de objetos en 3D y navegación inmersiva, constituyendo un entorno adecuado para el desarrollo de herramientas interactivas en realidad virtual.

2.1.10. HTML y JavaScript

El desarrollo del sistema se apoya en dos tecnologías web fundamentales: HTML y JavaScript. Cada una cumple un rol específico en la definición, control e interacción del entorno de realidad virtual basado en A-Frame.

HTML

HTML5 (HyperText Markup Language) es el lenguaje de marcado estándar utilizado para estructurar contenido en la web. En este proyecto, constituye la base de la interfaz y define la organización de los elementos visuales dentro del entorno de realidad virtual. HTML5 destaca por su extensibilidad, compatibilidad con estándares modernos y capacidad de integración con tecnologías como CSS, JavaScript y WebGL.

Una de sus ventajas clave es la posibilidad de describir contenido multimedia e interactivo sin necesidad de plugins externos. En este caso, se ha utilizado para integrar el framework A-Frame, que permite definir escenas tridimensionales mediante una sintaxis declarativa. Gracias a esta integración, es posible construir entornos VR directamente con etiquetas HTML, facilitando la comprensión y mantenimiento del código.

Además, HTML5 permite incorporar scripts externos y elementos de interfaz como botones o menús flotantes, lo que favorece una estructura clara y separada entre presentación y lógica.

JavaScript

JavaScript es un lenguaje de programación interpretado y orientado a eventos, ampliamente utilizado en el desarrollo web para implementar funcionalidades dinámicas e interactivas. Forma parte del núcleo de tecnologías web junto con HTML y CSS, y se ejecuta directamente en el navegador, lo que permite crear aplicaciones cliente sin necesidad de compilación previa.

En este proyecto, todo el código funcional ha sido desarrollado en JavaScript. Su uso permite manejar tanto la lógica de interacción con el usuario como la comunicación con servicios externos, además de controlar el comportamiento de los elementos 3D en tiempo real.

El lenguaje se adapta bien a entornos basados en frameworks como A-Frame, y su flexibilidad ha resultado fundamental para integrar tecnologías como la API de *Meadiastream Recording*, el sistema de reconocimiento de voz y el motor de generación de respuestas, todo en un entorno unificado.

Los aspectos específicos de su aplicación dentro del proyecto se detallan en secciones posteriores.

2.1.11. Otras tecnologías y herramientas

Además de las tecnologías principales, se han empleado otras herramientas de soporte que han contribuido al desarrollo del sistema:

- **Visual Studio Code (VSCode):** Editor de código utilizado para todo el desarrollo, con extensiones útiles para autocompletado, depuración y control de versiones.
- **Microsoft Edge:** Navegador web en el que se ha probado y validado el sistema, elegido por su compatibilidad total con las APIs WebXR y MediaStream.
- **Windows 10:** Sistema operativo base en el que se desarrolló el proyecto.
- **ChatGPT[17] y DeepSeek[5]:** Herramientas utilizadas como apoyo puntual para validación de fragmentos de código o resolución de dudas técnicas. No se ha delegado en ellas

la implementación del sistema.

Estas herramientas han sido elegidas por su compatibilidad con las tecnologías web modernas, su estabilidad y su amplia comunidad de soporte.

2.2. Trabajos relacionados

El uso de la voz como mecanismo de control en entornos virtuales es un área de creciente interés. Numerosos proyectos y prototipos han explorado la posibilidad de integrar reconocimiento de voz con interfaces inmersivas, especialmente en contextos de accesibilidad, formación y creatividad.

A continuación, se presentan algunos ejemplos representativos relacionados con el uso de comandos de voz en entornos virtuales y la edición de escenas 3D. Aunque no se ha realizado una revisión exhaustiva del panorama actual, se incluyen referencias clave que ilustran el contexto y los enfoques más cercanos al sistema propuesto.

2.2.1. Sistemas de control por voz en realidad virtual

Algunos sistemas comerciales y académicos han integrado control por voz en entornos VR. Por ejemplo, plataformas como *Oculus Voice Commands* permiten controlar funciones básicas del sistema mediante órdenes vocales predefinidas. Sin embargo, estos sistemas suelen estar limitados a comandos concretos y no permiten una interacción libre basada en lenguaje natural.

En el ámbito académico, recientes investigaciones han explorado la integración de modelos de lenguaje para realizar acciones en tiempo real dentro de entornos de realidad virtual. Un ejemplo destacado es Voice2Action[6][6], un marco que utiliza agentes impulsados por modelos de lenguaje para extraer comandos de voz y ejecutarlos jerárquicamente en escenarios VR, con soporte para detección de errores y retroalimentación desde el entorno.

[6]Enlace al repositorio de Voice2action https://github.com/yang-su2000/Voice2Action

El sistema propuesto en este trabajo se diferencia por permitir una interacción natural, sin comandos específicos, usando modelos de lenguaje de última generación alojados en plataformas de alto rendimiento. Además, su desarrollo se realiza íntegramente para la web, sin necesidad de instalaciones adicionales ni dependencias externas complejas.

2.2.2. Editores visuales 3D en la web

Existen múltiples herramientas que permiten crear y modificar escenas 3D desde el navegador. Algunas de las más destacadas son:

- **Tinkercad[1]**: Desarrollada por Autodesk, esta aplicación permite a usuarios principiantes crear modelos 3D de forma visual e intuitiva. Aunque está más orientada al diseño para impresión 3D, su facilidad de uso la hace relevante como referencia de interacción simplificada en 3D.

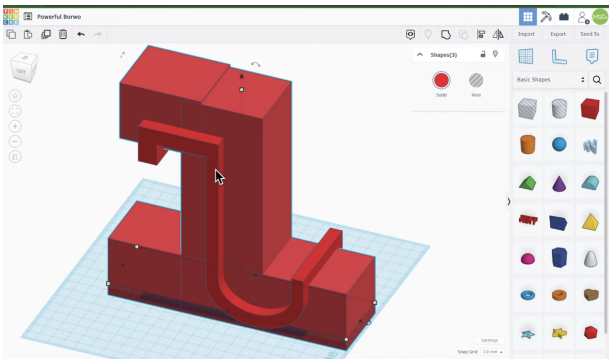


Figura 2.6: Interfaz de Tinkercad

- **Clara.io[3]**: Editor 3D completo en la nube que permite modelado, animación y renderizado sin necesidad de instalar software local. Está pensado para usuarios avanzados y profesionales del diseño, y no incorpora control por voz.

- **A-Frame Inspector**: Herramienta integrada en A-Frame que permite editar visualmente escenas VR directamente en el navegador. Aunque es útil para ajustar parámetros, no permite una interacción mediante lenguaje natural.

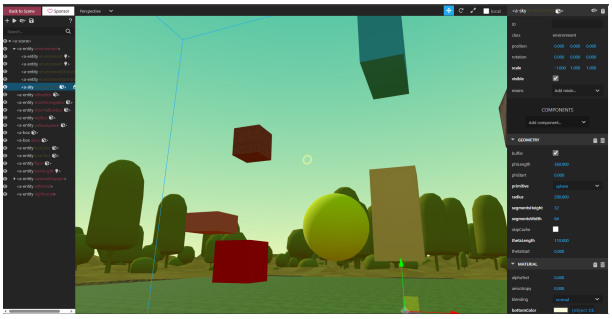


Figura 2.7: Ejemplo A-frame Inspector

El trabajo aquí desarrollado se diferencia de estas herramientas principalmente en el modo de interacción: en lugar de utilizar el ratón o paneles de control tradicionales, el usuario se comunica directamente con el sistema mediante la voz, utilizando lenguaje natural. Esta forma de interacción busca reducir la barrera de entrada para usuarios no técnicos.

2.2.3. Interfaces naturales basadas en lenguaje

Los sistemas de interacción basados en lenguaje natural, ya sea hablado o escrito, están cobrando relevancia en campos como el diseño asistido, la programación generativa y la creación de contenidos digitales. Herramientas como *RunwayML*[20] o *Spline AI*[21] permiten generar modelos 3D o escenas a partir de descripciones textuales. En general, se apoyan en modelos de lenguaje grandes (LLM) y técnicas de generación condicional.

Figura 2.8: Modelo 3D generado con un prompt en Spline

Estos sistemas suelen estar limitados a entrada de texto o requieren una interfaz más tradicional. El proyecto desarrollado en este trabajo aporta una propuesta distinta: se basa en voz, directamente en la web, sin dependencias complejas ni procesos de entrenamiento personalizados. Además, el flujo está centrado en modificar modelos genéricos ya cargados, lo cual simplifica la carga cognitiva del usuario y acota el dominio del problema, facilitando la interacción.

2.2.4. Proyectos previos basados en A-Frame

A-Frame ha sido ampliamente utilizado en la creación de experiencias inmersivas web, desde recorridos virtuales hasta juegos sencillos. Algunos proyectos de código abierto incluyen:

- **A-Painter**[16]: una herramienta de pintura en 3D, construida con A-Frame y WebVR, que permite a los usuarios dibujar en el espacio tridimensional usando dispositivos de VR.

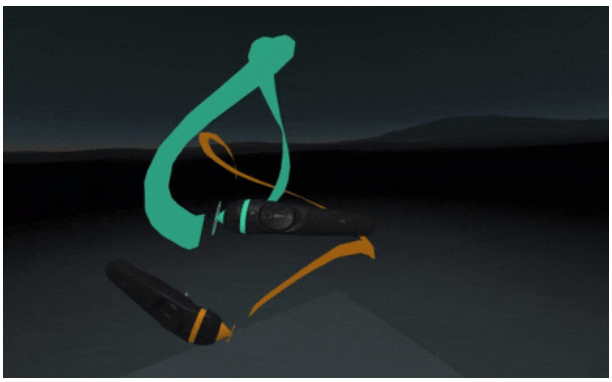


Figura 2.9: Demo de A-painter

- **JanusWeb**[9]: un navegador social tridimensional basado en tecnologías web que permite a los usuarios construir y compartir mundos 3D utilizando una sintaxis HTML extendida.

Estos proyectos han servido como inspiración en el uso de componentes personalizados, control de escena en tiempo real y manipulación directa de entidades A-Frame. Sin embargo, el enfoque propuesto en este TFG añade un nuevo modo de interacción centrado en el uso de lenguaje natural hablado, así como una arquitectura que separa el procesamiento de lenguaje mediante modelos alojados externamente, facilitando la escalabilidad del sistema.

2.3. Resumen del capítulo

En este capítulo se han descrito las tecnologías empleadas para el desarrollo del proyecto, incluyendo A-Frame, JavaScript, la API de MediaStream, Groq y el modelo Gemma 2 9B, entre otras. También se han presentado diversos trabajos relacionados, tanto en el ámbito de la edición 3D web como en el uso de voz y lenguaje natural como interfaz de control. El sistema propuesto se sitúa en la intersección de estas áreas, ofreciendo una experiencia innovadora basada en tecnologías abiertas y accesibles.

Capítulo 3

Desarrollo del Proyecto

El desarrollo del sistema se ha llevado a cabo mediante una implementación incremental, inspirada en los principios de las metodologías ágiles, en particular SCRUM. Aunque no se ha seguido este marco de trabajo de manera estricta, se ha adoptado su estructura basada en sprints para organizar el proceso de desarrollo. Esta estrategia permitió avanzar de forma progresiva, desde la exploración tecnológica inicial hasta la implementación de un entorno inmersivo controlado por voz, facilitando la corrección de errores a lo largo del proyecto y la adaptación de este en función de los hallazgos técnicos y los requisitos de usabilidad. Este capítulo detalla las etapas de desarrollo, las decisiones técnicas adoptadas y los desafíos superados durante la implementación.

3.1. Sprints

3.1.1. Sprint 1: Configuración inicial y escena base en A-Frame

Objetivo

El propósito de este primer sprint fue doble: por un lado, adquirir un conocimiento práctico del framework **A-Frame**[15], y por otro, sentar las bases de la escena inmersiva sobre la que

se construiría el sistema. Se buscaba obtener una primera versión funcional de una escena 3D accesible desde el navegador, capaz de renderizar elementos básicos y servir como punto de partida para las interacciones posteriores. Un ejemplo del resultado obtenido puede observarse en la Figura 3.1.

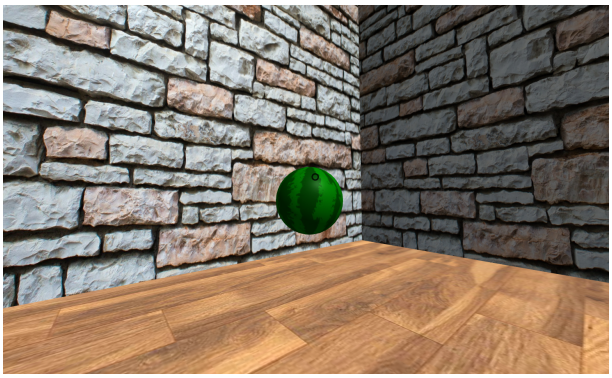


Figura 3.1: Primera escena generada

Actividades

- **Revisión superficial de la documentación oficial de A-Frame y sus ejemplos**: Se consultó de forma general la documentación disponible en la web oficial del framework, prestando atención a los aspectos esenciales para montar una escena básica. También se exploraron algunos ejemplos destacados para identificar las entidades más comunes y entender la estructura general del código en A-Frame.

- **Pruebas con entidades primitivas**: Se realizaron pruebas con algunas entidades primitivas de A-Frame (`a-box`, `a-sphere` y `a-plane`) para evaluar su comportamiento al modificar parámetros fundamentales como posición, rotación, escala y propiedades visuales (color, materiales y texturas básicas), lo que permitió verificar las capacidades básicas de renderizado y sentar las bases para desarrollos posteriores del entorno 3D.

- **Configuración inicial del proyecto**: Se definió la estructura básica de archivos del sistema, incluyendo los documentos HTML, el script JavaScript asociado y la carpeta de recursos. Se habilitó el entorno de desarrollo en *Visual Studio Code*, configurando extensiones útiles para depurar A-Frame en tiempo real.

- **Construcción de una escena mínima**: Se añadió una cámara con control orbital limitado, una luz direccional y unos planos que simulan las paredes y el suelo. Además, se incluyó un objeto interactivo (una esfera), que al pulsarse cambia de color y que se renderiza correctamente tanto en PC como en un visor de realidad virtual compatible con WebXR.
- **Pruebas de compatibilidad en diferentes navegadores**: Se verificó que la escena funcionara correctamente en *Google Chrome*, *Microsoft Edge* y *Firefox*

Prototipo generado

El resultado del sprint fue una escena 3D completamente funcional que se puede visualizar en cualquier navegador moderno sin necesidad de instalación. Contiene:

- Cinco planos que actúan como las paredes y suelo de una sala.
- Una cámara inicial con punto de vista fijo.
- Un objeto 3D interactivo (esfera) ubicado en el centro de la escena.
- Iluminación básica mediante luz direccional.

Este entorno constituye el esqueleto sobre el que se construirán las siguientes funcionalidades, especialmente las interacciones controladas por voz.

Lecciones aprendidas

- A-Frame ofrece una curva de aprendizaje suave gracias a su estructura basada en HTML, lo que permite realizar cambios rápidamente y visualizar resultados en tiempo real sin procesos de compilación complejos.
- La modularidad de A-Frame permite ampliar sus funcionalidades a través de componentes personalizados escritos en JavaScript, lo cual será clave para las fases siguientes del desarrollo.

- Es fundamental tener en cuenta la experiencia del usuario desde el principio: la altura de la cámara, el tamaño relativo de los objetos y la iluminación afectan significativamente a la percepción de inmersión.

3.1.2. Sprint 2: Captura de voz y transcripción con Whisper

Objetivo

Diseñar e implementar un sistema robusto para la captura y transcripción de voz, funcional tanto en navegadores de escritorio como en dispositivos de realidad virtual. Este sistema constituye la base de interacción natural por voz con el modelo de lenguaje que se implementará posteriormente.

Actividades

- **Evaluación preliminar de alternativas tecnológicas**: Se compararon dos enfoques principales para la transcripción de voz: el uso de *Web Speech API* (disponible en algunos navegadores) frente a la combinación de *MediaStream Recording API* con el modelo *Whisper*. Se concluyó que, aunque *Web Speech API* es más sencilla de integrar, su falta de soporte en entornos VR y su dependencia del navegador la hacen poco fiable para este proyecto.

- **Pruebas de compatibilidad en navegadores con soporte para VR**: Se realizaron tests en *Microsoft Edge* y *Google Chrome*, tanto en modo escritorio como en modo inmersivo (WebXR), comprobando que la API de grabación de audio mediante `MediaStream` funcionaba de manera consistente en ambos entornos.

- **Captura y empaquetado de audio**: Se implementó un sistema básico para capturar la voz del usuario mediante el micrófono, usando `MediaRecorder`, y almacenar los datos en un buffer que posteriormente se convierte a formato `Blob`. Esta transformación es necesaria para enviar los datos por red.

- **Conexión con Whisper vía API**: Se estableció un flujo de comunicación con el modelo *Whisper* a través de un proveedor de inferencia externa que permite enviar audio grabado y recibir como respuesta una transcripción en texto plano.
- **Validación funcional**: Se probó el sistema con distintas frases y entonaciones para evaluar la respuesta de *Whisper*, incluyendo pruebas con acentos, ruido ambiental moderado y variaciones en el ritmo del habla.

Prototipo generado

Se logró una primera integración funcional de la grabación de voz desde el navegador y su transcripción automática usando inteligencia artificial. Aunque aún no se incorporaba detección de intención, ya era posible obtener frases completas con alta fiabilidad a partir del audio del usuario.

Lecciones aprendidas

- La **compatibilidad multiplataforma** debe ser una prioridad desde el diseño: tecnologías como *Web Speech API* pueden parecer atractivas inicialmente, pero presentan limitaciones importantes en VR.
- **Whisper** ofrece una tasa de acierto muy superior a alternativas como Web Speech Api, especialmente en entornos ruidosos o con pronunciaciones no estándar, lo que lo hace ideal para sistemas de voz que permiten una interacción libre y natural.
- Aunque la grabación de voz en navegador conlleva ciertos retos técnicos —como gestionar la latencia, el inicio y fin de las grabaciones, y la conversión de formatos de audio compatibles con el modelo Whisper—, esta vía ofrece mayor control y compatibilidad que soluciones embebidas como Web Speech API, especialmente en entornos VR.

3.1.3. Sprint 3: Análisis semántico y generación de instrucciones con LLMs

Objetivo

Traducir comandos hablados en lenguaje natural a instrucciones estructuradas que puedan ser interpretadas por el motor 3D para crear o modificar objetos dentro de la escena.

Actividades

- **Desarrollo de un script de benchmarking:** Se creó un script que envía el mismo conjunto de comandos a distintos modelos LLM a través de la API de Groq. Se evaluaron tiempos de inferencia y coherencia semántica de la respuesta.
- **Comparación de modelos:** Se probaron modelos como **Gemma 2 9B**, **Mixtral 8x7B**, **LLaMA 3** y **Deepseek r1**. Se analizaron tiempos de respuesta (latencia en ms), precisión en la interpretación de órdenes y adecuación a la tarea. Los resultados obtenidos pueden observarse en la Figura 3.2.
- **Diseño de un esquema JSON:** Se estableció un formato estándar para estructurar la salida de los LLMs. Este JSON contiene campos como `model` (tipo de objeto), `scale`, `rotation`, `position` permitiendo una traducción directa entre intención del usuario y parámetros en A-Frame.
- **Sistema de inferencia:** Se implementó un módulo de procesamiento que recibe como entrada el texto transcrito mediante Whisper, lo envía a un modelo de lenguaje (LLM) para su interpretación semántica, y recibe como salida una respuesta estructurada en formato JSON. Este esquema permite la transformación programática de las instrucciones en lenguaje natural en modificaciones específicas de los atributos de los componentes A-Frame dentro del entorno virtual.

Modelo	Tiempo (ms)	Precisión
llama-3.1-8b-instant	4727	90%
llama-3.3-70b-versatile	5441.70	100%
llama-3.2-3b-preview	2663.30	80%
qwen-2.5-32b	5094.90	100%
mixtral-8x7b-32768	2321.90	80%
deepseek-r1-distill-llama-70b	5016.5	0%
gemma2-9b-it	2202.89	100%

Figura 3.2: Resultados de la comparación de los modelos

Prototipo generado

Un módulo funcional que, dado un comando como *"Quiero un coche grande y alto"*, produce una estructura JSON como:

```
{
  "model": "car",
  "scale": {"x":2, "y":3, "z":2},
  "position": {"x":0, "y":0, "z":-3},
  "rotation": {"x":0, "y":0, "z":0}
}
```

Este JSON será procesado más adelante para obtener y asignar los valores de cada atributo al modelo que se está editando en la escena.

Lecciones aprendidas

- **Gemma 2 9B** demostró un rendimiento óptimo, con tiempos de inferencia por debajo de 300ms y una semántica robusta, especialmente en comandos ambiguos o con estructura conversacional.

- Estandarizar la salida en un esquema estructurado fue fundamental para asegurar la modularidad del sistema y facilitar el tratamiento posterior por parte de los componentes de visualización y edición.
- El diseño de prompts adecuados y consistentes para cada modelo es determinante para obtener respuestas fiables. Los modelos LLM no siempre son deterministas, por lo que pequeñas variaciones pueden implicar salidas distintas.

3.1.4. Sprint 4: Generación y edición de objetos por voz

Objetivo

Permitir la creación y edición de objetos en la escena utilizando comandos en lenguaje natural, logrando un flujo continuo desde la captura de voz hasta la manipulación directa de los modelos 3D.

Actividades

- Implementación completa del flujo de procesamiento de comandos: desde la captura de audio, su transcripción a texto, la conversión del texto en un formato JSON estructurado y finalmente la aplicación de los atributos extraídos sobre el objeto 3D correspondiente.
- Desarrollo de componentes A-frame para la creación dinámica de entidades A-Frame (botones) que constituirán la interfaz base de la escena.
- Desarrollo del método *editarObjeto* en Javascript para la transformación de las propiedades principales de los modelos como escala, rotación y posición, basándose en los parámetros proporcionados por el JSON.
- Integración y carga de modelos 3D complejos en formato GLB/GLTF mediante el uso del componente `gltf-model`, ampliando el catálogo visual disponible para el usuario.
- Diseño del sistema para que la acción de "crear" se interprete como una edición inicial sobre un modelo base predeterminado, simplificando el manejo interno al reutilizar la

lógica de edición para la configuración inicial del objeto.

Prototipo generado

Se implementó un sistema funcional capaz de interpretar comandos de voz complejos para modificar atributos de entidades A-Frame, incluyendo: (1) generación de modelos (*Genera un dinosaurio"*, utilizando archivos `.glb`), (2) ajuste de escala (*Haz el objeto más grande/alto/ancho"*), (3) rotación (*Gira el objeto 180º"*), y (4) posicionamiento espacial (*Pon el objeto más arriba/abajo"* o con coordenadas específicas *"Ponlo en la posición x=3, y=2, z=0"*). Este sistema traduce instrucciones en lenguaje natural a acciones programáticas sobre los modelos 3D, garantizando una interacción intuitiva con la escena virtual.

Lecciones aprendidas

- Centralizar la lógica de transformación en un método único como *editarObjeto* optimiza el código y unifica la gestión tanto de la creación como de la edición de objetos.
- Interpretar la creación como una edición inicial sobre un modelo base simplifica el flujo de trabajo y reduce la duplicidad de funciones.
- La carga de modelos GLB/GLTF amplía las posibilidades visuales y de complejidad, aunque implica gestionar correctamente la carga y posibles impactos en el rendimiento.
- Mantener un control estricto sobre el objeto activo es esencial para una experiencia fluida y para resolver ambigüedades en los comandos por voz.

3.1.5. Sprint 5: Interacción fluida por voz y mejoras en la usabilidad

Objetivo

Optimizar la experiencia de entrada por voz para permitir un control más natural e inmersivo en entornos VR, eliminando fricciones derivadas de la interacción limitada por tiempo y

mejorando la detección de intenciones del usuario.

Actividades

- Sustitución del sistema anterior de grabación temporal (fijo a 3 segundos) por un sistema interactivo que activa la grabación de voz mientras se mantiene pulsado el clic izquierdo (en dispositivos de escritorio) o se realiza el gesto *pinch* (en visores de realidad virtual). Esta modificación permite una interacción más precisa y natural, adaptándose al tiempo requerido por el usuario para emitir los comandos vocales.
- Integración con el sistema de selección de objetos ("coger") para asociar la entrada de voz a un modelo 3D concreto, permitiendo que las instrucciones se apliquen directamente al objeto activo.
- Incorporación de una palabra clave (`"Gema"`) necesaria al enviar el comando para validar que la intención del usuario está dirigida al sistema y así evitar llamadas a la api accidentales o no deseadas.
- Ajuste de la lógica de control para permitir múltiples transformaciones (como escala y rotación) dentro de una única frase, facilitando comandos más complejos y naturales como "Gema, hazlo más grande y gíralo 45 grados".

Prototipo generado

Se obtiene una interfaz mucho más fluida y cómoda, especialmente pensada para la interacción inmersiva en realidad virtual. El usuario puede ahora "coger" un objeto y, mientras mantiene pulsada la acción, emitir comandos por voz sin límite de tiempo artificial. Esto permite combinar múltiples transformaciones en una única interacción, reduciendo pasos y aumentando la naturalidad en el proceso de edición de escenas.

Lecciones aprendidas

- **Persistencia de grabación basada en acción física:** asociar la escucha a una acción mantenida como el clic o el gesto *pinch* ofrece mayor control y reduce la frustración por cortes prematuros o grabaciones accidentales.
- **Uso de palabra clave (”Gema”):** filtrar las llamadas a la api por palabra clave resulta efectivo para descartar entradas irrelevantes, aumentar la precisión del sistema y ahorrar costes.
- **Compatibilidad con VR sin teclado:** adaptar la interacción al contexto inmersivo obliga a replantear el modelo clásico de entrada, pero ofrece a cambio una experiencia mucho más intuitiva y centrada en la voz como medio principal.
- **Comandos compuestos:** permitir la expresión de múltiples intenciones en una sola frase mejora la eficiencia y reduce la carga cognitiva del usuario.

3.1.6. Sprint 6: Interacción directa y edición manual

Objetivo

Complementar la interacción por voz con mecanismos de manipulación directa de objetos en la escena 3D, facilitando una experiencia más tangible e intuitiva para el usuario.

Actividades

- Implementación de un componente personalizado de A-Frame llamado `draggable`, que permite seleccionar, arrastrar y soltar modelos 3D directamente con el ratón en PC, pulsación en dispositivos táctiles o *pinch* en VR.
- Sincronización de las acciones manuales con el sistema de edición por voz, de modo que un objeto que ha sido “cogido” también pueda recibir comandos de transformación verbal (escala, rotación, duplicado, etc.).

- Añadido de retroalimentación auditiva al sistema de manipulación manual: se reproducen sonidos específicos al “coger” y “soltar” un objeto, mejorando la percepción de respuesta del sistema.
- Mejora del control de cámara en dispositivos de escritorio, permitiendo navegar libremente con solo mover el ratón (sin necesidad de hacer clic), lo que aporta mayor fluidez y realismo a la exploración del entorno.

Prototipo generado

Una escena 3D más rica en posibilidades interactivas, en la que el usuario puede manipular objetos de manera directa a través de mecanismos “drag and drop” y continuar editándolos mediante comandos por voz. La navegación se vuelve más natural y continua, con una cámara que responde al movimiento del cursor sin acciones adicionales.

Lecciones aprendidas

- **Combinación sin fricción:** Las interacciones por voz y manuales no se excluyen, sino que se integran de manera complementaria, permitiendo transiciones naturales entre ambas según el contexto de uso.
- **Feedback multisensorial:** El uso de sonido como confirmación de acciones mejora la comprensión de la interfaz y refuerza la sensación de control.
- **Navegación fluida:** El movimiento libre de cámara, sin necesidad de clics, contribuye a una mayor sensación de presencia e inmersión en la experiencia.

3.1.7. Sprint 7: Guardado y reutilización de escenas

Objetivo

Dotar al sistema de persistencia, permitiendo guardar el estado actual de la escena y recuperarlo posteriormente sin necesidad de rehacer manualmente el trabajo realizado.

Actividades

- Desarrollo de una funcionalidad de exportación de la escena actual en formato `.html`, que incluye todas las entidades presentes y sus transformaciones (posición, escala, rotación, modelo asociado).
- Creación de un método `cargarEscena`, ejecutado automáticamente al inicio de la aplicación, que detecta si existe una escena previamente guardada en la raíz del proyecto y la inserta dinámicamente dentro del entorno base.
- Inclusión de un botón accesible desde la interfaz que permite reiniciar completamente la escena, eliminando todos los objetos creados y dejando únicamente el entorno inicial intacto.

Prototipo generado

Un sistema funcional de persistencia que permite guardar la escena activa como un archivo HTML independiente y volver a cargarla automáticamente al acceder de nuevo a la aplicación. También se incluye la opción de limpiar por completo la escena con un solo clic.

Lecciones aprendidas

- **Compatibilidad y simplicidad:** Exportar escenas como archivos HTML asegura una alta compatibilidad con entornos web y permite reutilizar fácilmente el contenido sin depender de formatos propietarios o bases de datos.

- **Separación de responsabilidades:** Mantener el entorno base separado del contenido dinámico facilita la carga condicional de escenas y mejora la mantenibilidad del código.
- **Productividad del usuario:** El guardado automático y la recuperación de escenas anteriores mejora significativamente la continuidad del trabajo, especialmente en procesos creativos iterativos.

3.1.8. Sprint 8: Comandos avanzados y replicación de objetos

Objetivo

Extender las capacidades creativas del usuario mediante comandos de alto nivel, como la replicación múltiple de objetos (Figura 3.3) y la incorporación de ayudas iniciales (Figura 3.4) para facilitar la experiencia de usuario.

Actividades

- Refactorización del *prompt* enviado al modelo de lenguaje para incluir instrucciones explícitas sobre la detección de cantidades y generación de una variable `multiplicador` en el esquema JSON de respuesta.
- Implementación de una función de replicación en JavaScript, que instancia un número determinado de copias del objeto activo en función del valor de `multiplicador`.
- Desarrollo de un algoritmo de distribución espacial automática que posiciona las copias de manera ordenada y en fila en un lateral del objeto original, evitando solapamientos y mejorando la legibilidad de la escena.

Figura 3.3: Resultado de solicitar 3 copias más de un objeto

- Inclusión de una ventana flotante de bienvenida que se muestra automáticamente al inicio de la escena. Esta ventana contiene instrucciones breves sobre cómo interactuar con el sistema por voz y qué tipos de comandos están disponibles, con un botón para cerrarla manualmente.

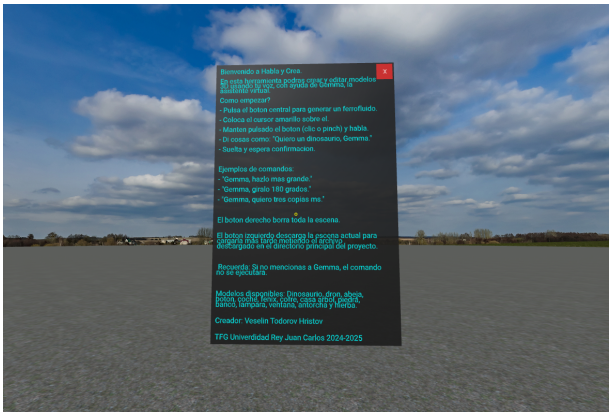


Figura 3.4: Ventana flotante para introducir la herramienta al usuario

Prototipo generado

Sistema capaz de interpretar comandos como “Haz cinco copias de este objeto” y generar múltiples instancias en la escena distribuidas de forma armónica. Además, al acceder a la escena, el usuario es guiado mediante una ventana de introducción flotante que mejora la accesibilidad y reduce la curva de aprendizaje.

Lecciones aprendidas

- **Lógica espacial adaptativa:** La replicación de objetos requiere lógica adicional para asegurar una distribución visualmente clara y evitar interferencias entre entidades.
- **Interfaz pasiva con valor añadido:** El uso de ayudas contextuales, como una ventana informativa al inicio, mejora la experiencia del usuario sin sobrecargar la interfaz principal.
- **Replicación eficiente:** La incorporación del sistema de copias ahorra tiempo y esfuerzo al permitir duplicar modelos ya configurados mejorando considerablemente la fluidez en el diseño de escenarios.

3.2. Conclusión del desarrollo del proyecto

Dada la incertidumbre técnica asociada a la integración de tecnologías como A-Frame, reconocimiento de voz y modelos de lenguaje natural, el enfoque ágil resultó especialmente adecuado. La flexibilidad del proceso permitió redefinir objetivos, experimentar con distintas soluciones y optimizar el diseño sin comprometer la coherencia global del sistema.

La organización por fases favoreció la integración progresiva de funcionalidades esenciales, tales como:

- Detección de comandos de voz.
- Comunicación con el modelo de lenguaje.
- Creación y modificación de modelos 3D a través del habla.
- Guardado de escenas

Como resultado, se ha obtenido una herramienta intuitiva y multiplataforma, especialmente útil en entornos donde el uso de teclado o controladores físicos no es viable.

Decisiones técnicas clave

- **Framework A-Frame**: Seleccionado por su capacidad para construir escenas 3D de forma declarativa en HTML, con buena integración de componentes externos, ideal para un entorno WebXR.
- **Reconocimiento de voz**: Se descartó `WebSpeechAPI` por falta de compatibilidad con gafas. Se optó por una solución híbrida con `MediaStream Recording API` para la captura de audio local y Whisper desde Groq como servicio de transcripción.
- **Modelo de lenguaje**: Se realizó una prueba comparativa entre distintos modelos de Groq para optimizar la velocidad y precisión. Se definió un formato de respuesta estructurado que garantiza que las órdenes puedan ser procesadas de forma robusta.
- **Modularización**: El paso a componentes A-Frame personalizados permitió reutilizar y mantener más fácilmente la lógica de creación y edición de objetos, aumentando la escalabilidad del sistema.
- **Unificación de creación y edición**: Se decidió unificar el flujo de **creación y edición** de objetos en un mismo sistema semántico, ya que conceptualmente *crear un objeto* puede entenderse como una edición inicial sobre un modelo base (por ejemplo, una entidad vacía con transformaciones por defecto). Esto permitió reutilizar el mismo esquema de procesamiento e instrucciones JSON, simplificando el pipeline lógico y reduciendo duplicidad en el código. Además, mejora la escalabilidad del sistema, ya que nuevos comandos o atributos se aplican por igual a objetos nuevos o existentes.

En definitiva, el desarrollo del proyecto ha demostrado la capacidad de integrar tecnologías avanzadas de IA y 3D en una aplicación accesible desde navegador, con control completo mediante lenguaje natural y una experiencia intuitiva y fluida.

Capítulo 4

Descripción del resultado

Este capítulo presenta el resultado funcional del sistema desarrollado: una herramienta inmersiva de construcción y edición de escenas 3D mediante lenguaje natural. La aplicación está disponible públicamente a través de una URL[1] accesible desde cualquier navegador compatible con WebXR.

El objetivo principal del sistema es facilitar la creación de entornos tridimensionales sin necesidad de conocimientos técnicos ni de herramientas complejas, mediante una interfaz sencilla basada en voz e interacción directa con elementos 3D. El usuario puede construir escenas, añadir modelos, redimensionarlos, rotarlos, moverlos o replicarlos únicamente usando comandos hablados, apoyado por una interfaz física accesible dentro del propio entorno.

La experiencia se adapta a tres modalidades de uso:

- **En escritorio (PC):** el usuario emplea el ratón para seleccionar y mover modelos con *drag and drop*, y el micrófono del equipo para dictar comandos de voz.
- **En realidad virtual (VR):** la interacción se realiza mediante controladores o gestos de "pinch" con las manos, mientras que el micrófono del visor permite la entrada de voz. El entorno está optimizado para ofrecer una experiencia totalmente inmersiva, considerando

[1] URL del Trabajo de Fin Grado: https://vescoth.github.io/tfg-site/

las limitaciones y capacidades específicas de la VR (como la puntería con el cursor central o la navegación por el escenario).

- **En dispositivos móviles:** el usuario puede observar la escena y generar comandos por voz, pero la movilidad en el entorno está limitada y la interacción por contacto es menos precisa. La experiencia móvil está pensada como un modo de consumo o edición ligera.

En las siguientes secciones se describen las funcionalidades principales del sistema, la organización de la interfaz, las capacidades de edición mediante lenguaje natural, la gestión de la escena y las formas de navegación. Todo ello se ha diseñado con el propósito de ofrecer una experiencia accesible, flexible y multiplataforma para cualquier tipo de usuario.

4.1. Descripción funcional

El sistema desarrollado es una aplicación web inmersiva que permite crear, editar y guardar escenas tridimensionales mediante comandos de voz. La aplicación funciona tanto en plataformas de escritorio como en dispositivos de realidad virtual, y está diseñada para que el usuario pueda construir entornos 3D de forma intuitiva, sin necesidad de conocimientos técnicos ni interfaz compleja.

El resultado final es un entorno interactivo donde el usuario puede:

- Crear modelos 3D en la escena.
- Seleccionarlos y arrastrarlos por la escena.
- Modificarlos mediante órdenes habladas.
- Guardar y recuperar el estado completo de la escena.
- Interactuar en un entorno 3D accesible desde navegador o visor VR.

Figura 4.1: Aplicación general

Nota sobre las figuras

Todas las capturas de pantalla de la escena VR presentadas en esta memoria utilizan una imagen de fondo común creada por Ivan Tsyrkunovich, disponible en la plataforma Vecteezy[2]. Se ha utilizado con fines educativos, respetando la atribución exigida por su licencia.

4.1.1. Interfaz inicial

Al acceder a la escena, el usuario se encuentra con una interfaz minimalista compuesta por un panel flotante con unas breves instrucciones de uso y tres botones situados sobre el suelo que permanecen siempre en la escena: Guardar, Crear y Borrar (Figura 4.2). La disposición de los botones surgió durante las primeras fases de desarrollo, en las que se priorizó la inmersión sobre los menús flotantes tradicionales, ya que se requería un disparador claro para eventos clave (como la generación de modelos). Los botones anclados al entorno —en lugar de interfaces superpuestas— resultaron más intuitivos y menos disruptivos para la experiencia del usuario. Estos botones no solo son el punto de partida para construir la escena, sino también un reflejo del enfoque de diseño centrado en la simplicidad y la interacción orgánica con el espacio 3D.

[2] https://www.vecteezy.com/free-photos/equirectangular?license-free=true

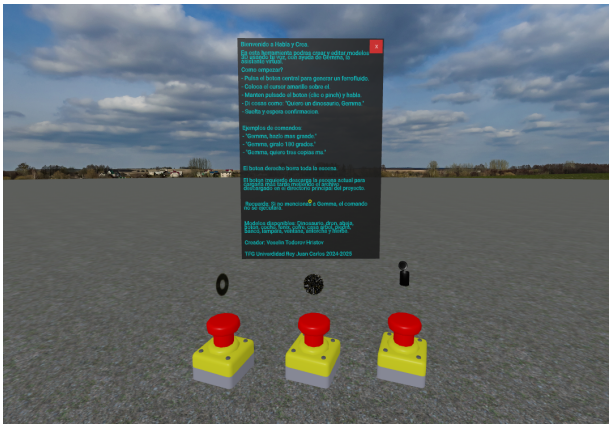


Figura 4.2: Interfaz inicial

4.1.2. Creación y edición de modelos

Durante el desarrollo de la aplicación surgió un desafío clave: *¿cómo identificar de manera intuitiva cuándo un usuario desea editar un objeto en la escena?* Se exploraron diversas alternativas, como el uso de menús contextuales o comandos de voz directos, pero estas opciones rompían la inmersión o requerían una curva de aprendizaje más pronunciada.

La solución adoptada fue vincular la edición a un **evento de agarre** (*mouse down* o *pinch* en VR), combinado con un sistema de grabación de voz activado durante la interacción. Este enfoque logra dos objetivos fundamentales:

1. **Señal clara de intención**: El gesto físico de ”coger” un objeto indica explícitamente que el usuario quiere modificarlo.

2. **Interacción multimodal**: Permite ajustar la posición (*drag and drop*) mientras se dictan órdenes complejas (ej: cambiar a un banco y duplicar su tamaño”), todo en un flujo continuo.

El flujo resultante, detallado a continuación, optimiza la usabilidad en entornos inmersivos:

Al pulsar el botón **Crear**, el sistema genera y añade a la escena un modelo 3D genérico (Figura 4.3) que actúa como elemento base para la interacción. Cada nueva pulsación del botón

produce una instancia independiente del modelo, permitiendo así la incorporación múltiple de objetos en el entorno virtual.

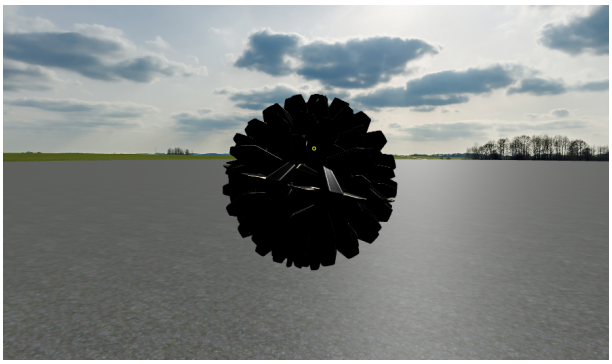


Figura 4.3: Elemento base que aparece al presionar el botón ”Crear”

Para editar un modelo, el usuario debe primero seleccionarlo. Esto se hace colocando el cursor sobre el modelo 3D y manteniendo pulsado el clic izquierdo del ratón (en escritorio), realizando un gesto de “pinch” (si se está en VR) o haciendo ”tap” (si nos encontramos en un dispositivo táctil). Mientras se mantiene pulsado, el objeto queda ”cogido”:

- Mientras está cogido, el usuario puede moverlo libremente por la escena mediante *drag and drop*.
- Al mismo tiempo, el sistema comienza a grabar audio para interpretar órdenes del usuario por voz en lenguaje natural.
- Cuando se suelta el clic (o se deja de hacer pinch o tap), se detiene la grabación, y la orden es enviada para su interpretación.

Este diseño permite realizar ajustes físicos (posición) mientras se dicta una orden de edición semántica, como cambiar el modelo o su tamaño.

4.1.3. Acciones disponibles por voz

Las acciones disponibles por voz han sido cuidadosamente diseñadas a través de la construcción de un *prompt* adaptado, que se transmite al modelo de lenguaje (LLM) con el objetivo

de interpretar correctamente la intención del usuario a partir de sus expresiones en lenguaje natural. Más que definir comandos rígidos, el sistema se apoya en un *prompt* cuidadosamente estructurado que orienta al modelo en la identificación precisa de la acción deseada entre todas las posibles opciones de transformación.

Para optimizar las llamadas a la API del modelo y evitar interpretaciones innecesarias, el sistema requiere que el usuario utilice la palabra clave ”Gema” en cada instrucción por voz. Esta palabra actúa como disparador, permitiendo que solo las expresiones relevantes sean procesadas por el modelo de lenguaje, lo cual mejora el rendimiento y reduce el consumo de recursos.

Estas opciones de edición han sido definidas en función de las capacidades nativas que ofrece A-Frame, e incluyen principalmente operaciones de reposicionamiento, escalado, rotación y sustitución del modelo 3D mediante archivos en formato `glb`. Esto permite al usuario, por ejemplo, modificar la posición del objeto en el espacio con comandos como Gema, muévelo un poco hacia abajo”, cambiar su forma diciendo Gema, hazlo un coche”, o ajustar su tamaño mediante expresiones como “Gema, hazlo el doble de grande”.

Gracias a esta arquitectura basada en prompts, el sistema permite al modelo de lenguaje adaptarse a una amplia variedad de formulaciones naturales, manteniendo al mismo tiempo la precisión necesaria para ejecutar transformaciones técnicas sobre los objetos de la escena. Esto proporciona una experiencia fluida y accesible tanto para usuarios expertos como para usuarios sin conocimientos técnicos.

El sistema reconoce instrucciones naturales como las siguientes:

- **Cambiar el modelo:** ”Gema, convierte esto en un coche”, ”Hazlo un dinosaurio, Gema”.
- **Moverlo:** ”Gema, muévelo un poco hacia arriba”, ”Colócalo en la coordenada 3, 0, -2, Gema”.
- **Rotarlo:** ”Gíralo 90 grados en el eje Y, Gema”, ”Gema, dale la vuelta”.
- **Escalarlo:** ”Hazlo el doble de grande, Gema”, ”Gema, tres veces más ancho”.
- **Replicarlo:** ”Quiero cuatro copias más, Gema”.

- **Eliminarlo:** ”Gema, borra este objeto”, “Elimina esto, Gema”.

Este enfoque permite que usuarios sin formación técnica puedan interactuar fluidamente con la escena, mientras el sistema mantiene un alto nivel de precisión en la ejecución de las modificaciones solicitadas.

4.1.4. Gestión de la escena

La escena se gestiona a través de los botones de la interfaz inicial, presentes en la Figura 4.2

- **Botón guardar:** Genera un archivo `HTML` con el estado actual de la escena. Este archivo puede descargarse y guardarse junto al archivo principal. Si el archivo está presente al iniciar la aplicación, la escena se cargará automáticamente.
- **Botón crear:** Instancia un objeto base a partir del cual se puede generar cualquier modelo glb.
- **Botón borrar:** Elimina todos los modelos de la escena y la deja vacía.

4.1.5. Navegación

El usuario puede moverse por el entorno de distintas formas dependiendo del dispositivo que esté utilizando:

- En ordenador: con las teclas `WASD` para desplazarse, `espacio` para elevarse y `shift` para descender.
- En VR: el usuario se puede desplazar por la escena caminando físicamente.
- En móvil: el usuario puede mover la cámara pero no se puede desplazar.

4.1.6. Ejemplo de uso: “coches huyendo de un dinosaurio”

A continuación, se describe un ejemplo concreto de uso que ilustra el flujo típico de interacción:

1. El usuario accede a la escena base.

Figura 4.4: Escena base

2. Pulsa el botón **Crear** una vez para generar un modelo base (ferrofluido).

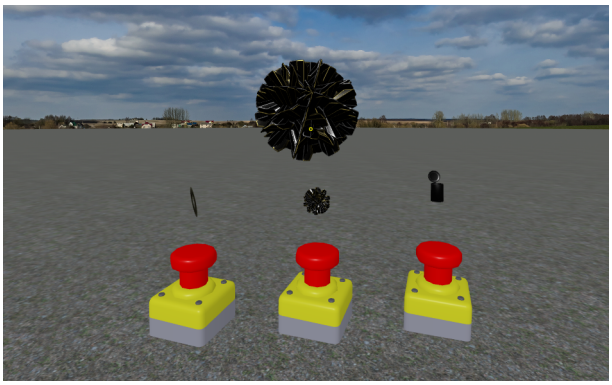


Figura 4.5: Creación del ferrofluido

3. Coge el ferrofluido manteniendo ”click”, ”pinch” o ”tap” sobre este, pronuncia el comando ”Gema, necesito generar un dinosaurio gigante”

4. Coge el modelo del dinosaurio generado y lo posiciona en un punto fuera del espacio de creación.

Figura 4.6: Creación y desplazamiento del dinosaurio

5. Pulsa el botón **Crear** de nuevo para generar otro ferrofluido.

6. Coge el nuevo ferrofluido y pronuncia ”Gema, genera un coche”

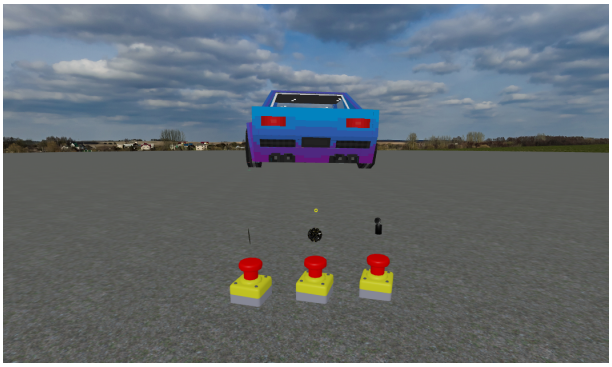


Figura 4.7: Creación del coche

7. Coge el coche y dice ”Oye Gema, gíralo 180 grados y genera 5 copias más”

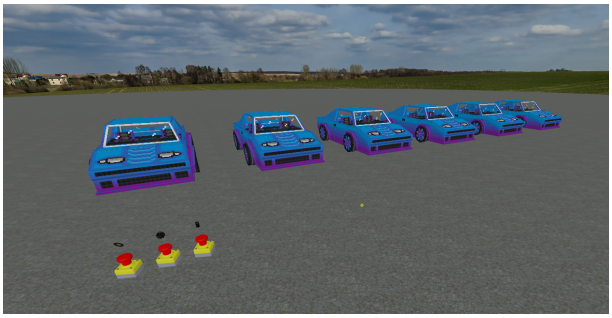


Figura 4.8: Giro y réplica del coche

8. El usuario coge y reposiciona con la funcionalidad de *drag & drop* los coches delante del dinosaurio como mejor le parezca.

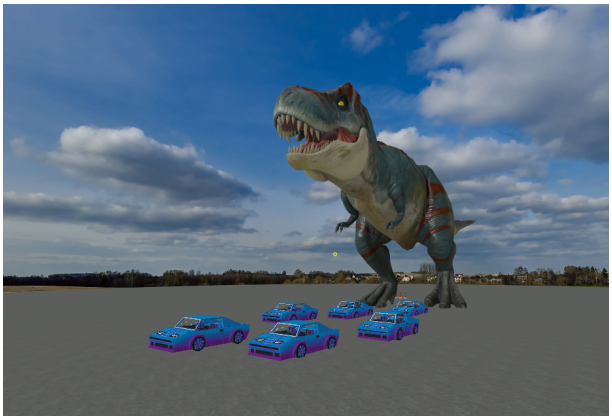


Figura 4.9: Escena Final

9. Pulsa **Guardar** para descargar la escena final.

Este ejemplo demuestra cómo es posible construir escenas narrativas completas con apenas unos clics y comandos de voz.

4.2. Descripción de la implementación

En este capítulo se describe la arquitectura interna de la aplicación, desglosando los principales componentes que permiten su funcionamiento. La implementación se basa en el framework **A-Frame** que permite construir experiencias inmersivas en WebXR mediante una estructura declarativa basada en HTML.

A-Frame se organiza en torno al concepto de **componentes**, pequeñas unidades reutilizables de funcionalidad que pueden añadirse a entidades HTML para modificar su comportamiento o apariencia. Estos componentes pueden ser nativos del framework (como `position`, `geometry` o `material`) o definidos por el desarrollador para implementar lógicas personalizadas.

Durante el desarrollo del proyecto se han creado e integrado múltiples componentes personalizados que encapsulan funcionalidades clave del sistema, desde la creación y edición de objetos 3D hasta la captura de voz, el procesamiento semántico o la gestión de escenas. Esta

aproximación modular ha facilitado tanto la escalabilidad como el mantenimiento del proyecto.

A continuación, se detallan los elementos estructurales más relevantes de la aplicación y cómo se integran entre sí para ofrecer una experiencia interactiva y fluida.

4.2.1. Estructura general del sistema

El sistema desarrollado está estructurado como una arquitectura modular en torno al motor A-Frame (Figura 4.10), aprovechando su modelo basado en componentes para encapsular comportamientos visuales e interactivos. Esta arquitectura permite una clara separación entre la entrada del usuario, el procesamiento semántico y la manipulación de la escena 3D, facilitando la escalabilidad y el mantenimiento del código.

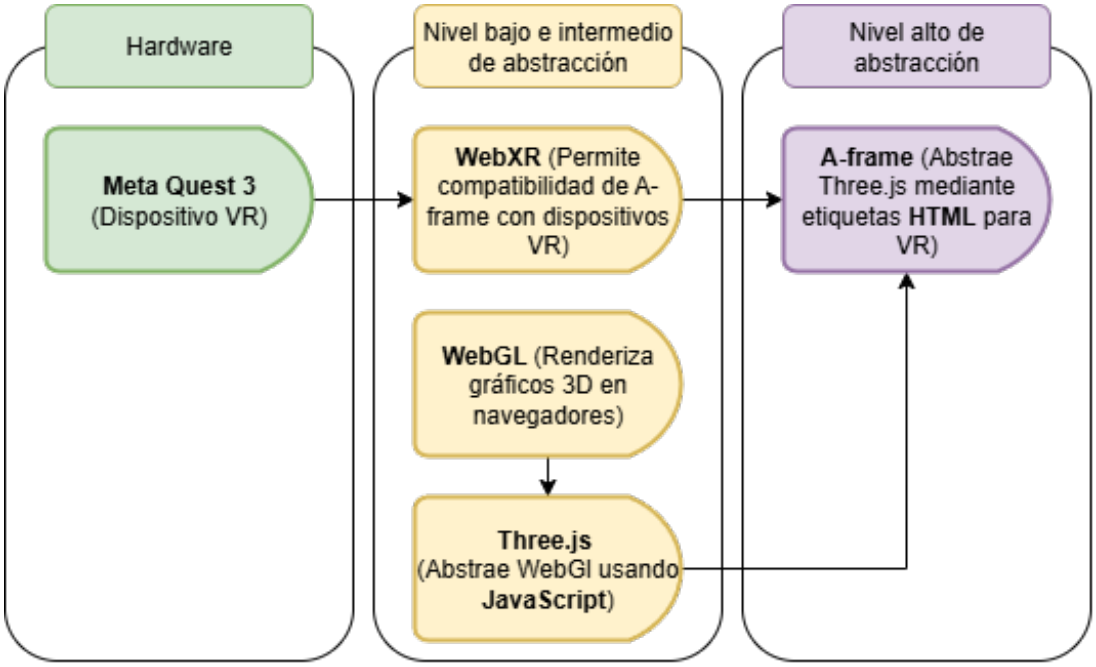


Figura 4.10: Diagrama que relaciona las tecnologías visuales del sistema

La interacción principal se basa en la voz del usuario, y sigue un flujo de procesamiento que transforma el audio en acciones visuales (Figura 4.11). Este flujo puede describirse en varias etapas:

1. **Captura de audio:** El sistema emplea la `MediaStream Recording API` para re-

gistrar la voz del usuario mientras mantiene çogidoün objeto editable. El audio se procesa en fragmentos como *blobs* temporales y se envía directamente a la API para su transcripción, sin necesidad de almacenamiento local.

2. **Transcripción del habla:** El *blob* de audio generado durante la grabación es enviado directamente a la API de Groq, donde se utiliza el modelo **Whisper** para transcribir la voz en texto natural. Este modelo permite una transcripción precisa incluso en entornos con ruido o pronunciación variable.

3. **Interpretación semántica:** El texto transcrito se envía como entrada a un modelo de lenguaje grande, en este caso **Gemma 2**, también alojado en Groq. Este modelo ha sido afinado con un *prompt* específico para devolver como salida un `JSON` estructurado que describe las acciones que el sistema debe ejecutar (tipo de objeto, color, número de copias, transformaciones, etc.).

4. **Procesamiento del JSON:** Este JSON es recibido en el cliente y analizado mediante funciones JavaScript personalizadas. A partir de su contenido, se instancian o modifican entidades A-Frame en la escena.

5. **Renderizado en A-Frame:** Finalmente, los cambios se reflejan directamente en el entorno WebXR, donde los objetos se crean, transforman o eliminan según las instrucciones interpretadas. A-Frame se encarga de renderizar los modelos 3D en tiempo real tanto en navegador como en dispositivos de realidad virtual.

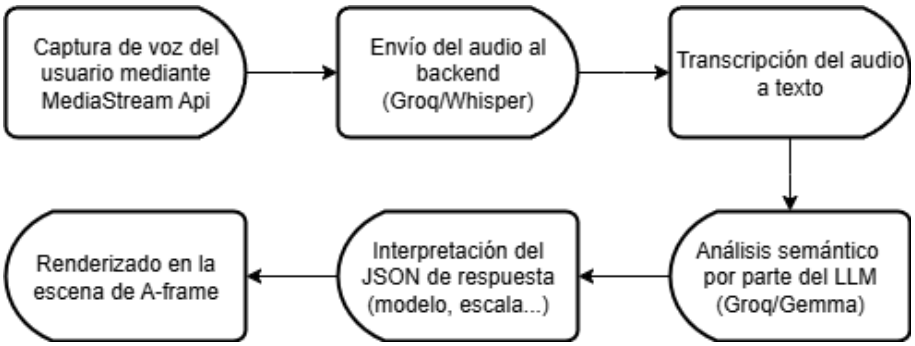


Figura 4.11: Diagrama de flujo del sistema de control por voz

Este flujo voz → JSON → escena permite que el usuario interactúe con el entorno 3D usando

principalmente lenguaje natural, sin necesidad de interfaces gráficas tradicionales. Además, el sistema está diseñado para ser reactivo: cada nuevo comando puede modificar o complementar lo anterior, lo que habilita una edición progresiva e intuitiva de la escena.

En las siguientes secciones se describen en detalle los componentes principales que conforman esta arquitectura y cómo se relacionan entre sí para dar soporte a las funcionalidades implementadas.

4.2.2. Componentes principales

- **Componente `creador`:** Este componente permite que cualquier entidad a la que se le asigne escuche eventos del tipo *mousedown* para iniciar un proceso de edición. Su lógica se define dentro del método init, que es ejecutado cuando el componente se inicializa.

  Dentro del init, se añaden dos listeners: uno para el evento ”click” y otro para ”mousedown”. Ambos eventos invocan event.stopPropagation() para evitar que el evento se propague a otros elementos de la escena. Sin embargo, la lógica principal está en el evento ”mousedown”, que además de detener la propagación, obtiene el atributo id de la entidad (this.el.getAttribute('id')). Posteriormente, llama a una función asincrónica externa llamada startEdit, pasándole dicho id como argumento. Esta función se encarga de iniciar el proceso de edición del objeto que tiene asignado el componente.

- **Componente `eraser`:** Este componente permite asociar una funcionalidad de borrado a la entidad sobre la que se aplica. Al inicializarse, este componente registra un listener para el evento `mousedown`, que se activa cuando el usuario presiona el botón del ratón o ”pinch”(su equivalente en VR).

  Al producirse el evento, se ejecuta la función `borrarTodo()` que se encarga de eliminar los elementos de la escena editables que se encuentran dentro del elemento con el identificador `lanzador` en formato HTML.

- **Componente `saver`:** Este componente permite asociar una funcionalidad de guardado a la entidad sobre la que se aplica. Al inicializarse, registra un listener para el evento `mousedown`, que se activa cuando el usuario presiona el botón del ratón o su equivalente

en VR.

Al producirse el evento, se ejecuta la función `actualizarAtributos()`, que provoca que los objetos de la escena actualicen sus atributos en el DOM; a continuación, se obtiene el contenido HTML del elemento con identificador `lanzador`. Este contenido, que corresponde a los hijos de la entidad que contiene el componente `creador-lanzador`, se convierte en un archivo de texto (`.html`) mediante la creación de un `blob`, el cual se transforma en una URL temporal. A continuación, se crea dinámicamente un enlace (`<a>`) con esa URL y se simula un clic para descargar el archivo con el nombre `mi-escena.html`. Finalmente, se libera la URL generada para evitar pérdidas de memoria.

- **Componente `draggable`:** Este componente permite que los objetos a los que se aplica puedan ser arrastrados y reposicionados en la escena utilizando el cursor central, tanto en dispositivos tradicionales como en entornos de realidad virtual.

  Al inicializarse, el componente define varios atributos internos: un flag `grabbed` para indicar si el objeto está siendo arrastrado, la profundidad inicial del objeto respecto a la cámara (`initialDepth`), un vector de desplazamiento (`offset`) para mantener la distancia relativa con respecto al punto de intersección, y referencias al cursor (`gaze-cursor`) y la cámara.

  El arrastre comienza cuando el usuario activa un evento `mousedown` o `touchstart` sobre el objeto. Si el cursor (implementado como un raycaster) detecta una intersección válida con el objeto, se marca como ”grabbedz se calcula la diferencia entre la posición actual del objeto y el punto de intersección, así como la distancia desde la cámara.

  Durante cada ciclo de renderizado (`tick`), si el objeto está en estado ”grabbed”, se recalcula su nueva posición manteniendo su profundidad original y el desplazamiento inicial, moviéndolo hacia el punto que apunta el cursor.

  El arrastre finaliza con un evento `mouseup` o `touchend`, tras lo cual se actualiza el atributo `position` del objeto en A-Frame y se llama a la función `actualizarAtributos()` para reflejar los cambios en el DOM.

- **Componente `creador-lanzador`:** Este componente se encarga de generar dinámi-

camente nuevas entidades en la escena cada vez que se produce un evento `mousedown` sobre el elemento al que está asignado. Estas entidades creadas reciben automáticamente un conjunto de atributos predefinidos.

Al activarse el evento, se incrementa un identificador global (`state.creation_id`) y se crea un nuevo elemento `a-entity` al que se le asignan múltiples propiedades: un modelo 3D definido por el atributo `gltf-model`, una posición inicial centrada frente al usuario (`0 2 -5`), una escala uniforme, rotación nula y animación mediante el componente `animation-mixer`.

Además, se le asignan los componentes `creador` y `draggable`, lo que le permite ser editado y arrastrado posteriormente. También se incluye una clase CSS `clickable` —que se utiliza para identificar qué entidades de la escena pueden ser detectadas e interactuadas mediante el cursor— y se añaden efectos sonoros para los eventos `mousedown` y `mouseup`, proporcionando retroalimentación auditiva al usuario. Finalmente, la nueva entidad se añade como hijo del elemento original que contiene el componente.

- **Componente `creador-ui`:** Este componente genera una interfaz de usuario tridimensional al ser asignado a una entidad de la escena. Durante su inicialización, crea y posiciona tres botones funcionales: uno para lanzar nuevos objetos (`lanzador`), otro para guardar el estado de la escena actual (`saver`), y uno adicional para eliminar todos los elementos creados (`borrar`). Cada botón se implementa como una entidad con un modelo 3D asignado y una clase `clickable` que permite la interacción mediante el cursor del usuario.

  A cada botón se le asocia su respectivo componente funcional (`creador-lanzador`, `saver`, o `eraser`), así como sonidos de confirmación al hacer clic. Además, junto a cada botón se genera una señal visual flotante —una entidad 3D adicional— que ayuda a identificar la funcionalidad del botón mediante un modelo ilustrativo (por ejemplo, un icono de CD para `saver` o de papelera para `borrar`). Todos los elementos generados son agregados como hijos de la entidad que contiene este componente.

4.3. Reutilización de componentes en otras aplicaciones

Los componentes desarrollados para esta herramienta en A-Frame son altamente reutilizables y pueden integrarse en otras aplicaciones inmersivas con necesidades similares de interacción, manipulación de objetos o persistencia de escenas. A continuación, se explica cómo podría reutilizarse cada uno de los componentes descritos anteriormente en distintos contextos, así como un ejemplo concreto de adaptación a una nueva aplicación.

4.3.1. Potencial de reutilización por componente

- **Componente `creador-ui`:** Puede ser reutilizado en cualquier aplicación que requiera una interfaz física dentro de la escena 3D. Dado que los botones no flotan sino que se colocan sobre el suelo, son especialmente útiles en entornos inmersivos donde se espera que el usuario explore e interactúe con elementos físicos en el espacio. Los botones pueden ser fácilmente renombrados o reubicados para adaptarse a otras funcionalidades, como cambiar el modo de edición, alternar entre escenas o seleccionar herramientas.

- **Componente `creador-lanzador`:** Este componente puede ser adaptado a cualquier sistema que necesite instanciar entidades base en respuesta a acciones del usuario. Por ejemplo, en un entorno de diseño arquitectónico, se podría usar para colocar módulos constructivos predefinidos (paredes, ventanas, columnas, etc.) que luego podrán editarse con comandos de voz.

- **Componente `creador`:** Es el núcleo del sistema, ya que permite editar los modelos 3D usando lenguaje natural. Puede ser reutilizado en cualquier aplicación donde los objetos 3D deban ser modificables mediante voz, ya sea para cambiar su forma, tamaño, posición o propiedades visuales. Su capacidad de reconocer comandos naturales y reflejar los cambios en tiempo real lo hace ideal para aplicaciones educativas, de diseño o entretenimiento.

- **Componente `eraser`:** Se puede utilizar en cualquier entorno donde se requiera una opción rápida de reinicio o limpieza de la escena, eliminando todos los elementos tempo-

rales sin afectar los controles principales. También puede adaptarse para eliminar únicamente objetos seleccionados, extendiendo su funcionalidad.

- **Componente `saver`:** Es útil para cualquier aplicación donde el usuario desee guardar el estado actual de la escena para continuar editando posteriormente o para exportar su trabajo. Por ejemplo, en un simulador de escenografía teatral, este componente permitiría exportar el diseño 3D generado por el usuario.

- **Componente `draggable`:** Permite una manipulación intuitiva y física de los objetos en la escena. Su reutilización es directa en cualquier aplicación que requiera la reubicación libre de entidades, como juegos, simuladores de montaje, o herramientas de prototipado rápido.

4.3.2. Ejemplo de reutilización: Simulador de decoración de interiores

Un caso concreto donde se pueden reutilizar los componentes desarrollados es en una aplicación de simulación de decoración de interiores. En este contexto, el usuario podría diseñar y modificar la disposición de una habitación virtual utilizando comandos de voz naturales, sin necesidad de menús complejos o interacciones tradicionales.

- **Interfaz mediante `creador-ui`:** La escena puede presentar botones físicos como "Añadir mueble", "Guardar diseño" o "Vaciar habitación", colocados en algún punto en el interior o exterior de la habitación, accesibles mediante desplazamiento dentro de la propia escena. Esto facilita la interacción sin necesidad de abandonar la vista inmersiva.

- **Instanciación con `creador-lanzador`:** Al pulsar el botón "Añadir mueble", el componente genera una entidad base (por ejemplo, una silla genérica) que se colocará en la escena. Esta entidad incluirá el componente `creador`, permitiendo su posterior personalización.

- **Edición con lenguaje natural mediante `creador`:** El usuario podrá seleccionar cualquier objeto en la habitación y emitir comandos como "hazlo más grande", "gira la mesa

45 grados" o "cambia el color del sofá a azul". Estas órdenes serán interpretadas mediante un sistema de reconocimiento de voz conectado a modelos de lenguaje natural, y aplicadas automáticamente sobre el modelo 3D seleccionado.

- **Eliminación con `eraser`:** La opción "Vaciar habitación" permite eliminar todos los muebles con un solo comando, reiniciando la escena sin afectar a los controles ni la estructura base.
- **Exportación con `saver`:** Una vez satisfecho con el diseño, el usuario puede guardar su configuración actual mediante el botón "Guardar diseño", que descargará un archivo HTML con la disposición completa, permitiendo compartir o cargar ese diseño posteriormente.
- **Interacción física con `draggable`:** Además de comandos de voz, el sistema permite al usuario mover libremente los muebles arrastrándolos por la habitación, facilitando un flujo de trabajo híbrido que combina interacción física y lenguaje natural.

Este caso demuestra cómo los componentes diseñados para el sistema original pueden integrarse eficazmente en nuevas aplicaciones, manteniendo el valor diferenciador de la plataforma: la edición natural e intuitiva de modelos 3D mediante lenguaje hablado.

Capítulo 5

Pruebas y validación

5.1. Objetivo del experimento

El objetivo principal de este experimento es validar el sistema de edición de modelos 3D mediante lenguaje natural en distintos dispositivos y modalidades de uso (VR, PC y móvil). Se busca evaluar tanto el rendimiento funcional como la experiencia de usuario durante la creación, edición, replicación, eliminación y guardado de entidades en una escena A-Frame.

5.2. Diseño del experimento

Se han diseñado siete tareas básicas, cubriendo el flujo principal del sistema:

1. Borrar todos los objetos de la escena.
2. Crear un modelo concreto (por ejemplo, un coche).
3. Escalar el modelo mediante un comando de voz.
4. Eliminar un modelo concreto.
5. Rotar y replicar un modelo.

6. Construir una estructura con varios bloques apilados.

7. Guardar la escena generada.

Los participantes han sido observados mientras realizaban estas tareas. Se midieron los tiempos de ejecución, se registraron errores y se recopilaron opiniones finales. También se evaluó la tasa de éxito y el grado de satisfacción general con el sistema.

5.3. Metodología

Las pruebas se llevaron a cabo en sesiones individuales. A cada usuario se le explicó brevemente el funcionamiento general del sistema y luego se le pidió que realizara las tareas por sí mismo, utilizando comandos por voz, la funcionalidad de *drag & drop* y el sistema de interacción integrado en la escena (botones físicos, cursor, etc...). Las pruebas se realizaron en tres modalidades:

- Tres usuarios utilizando visores de realidad virtual.
- Un usuario usando navegador web en PC.
- Un usuario usando un dispositivo móvil con navegador compatible.

5.4. Descripción de los participantes

- **Usuario A (original, VR)**: Usuario con conocimientos básicos en VR, sin experiencia previa con editores A-Frame.
- **Usuario B (VR)**: Perfil técnico, desarrollador de experiencias inmersivas. Alta familiaridad con VR.
- **Usuario C (VR)**: Usuario casual, aficionado a la realidad virtual, sin conocimientos técnicos.

- **Usuario D (PC)**: Diseñador gráfico acostumbrado a editores 3D tradicionales (Blender, SketchUp).
- **Usuario E (Móvil)**: Usuario general, con perfil no técnico, experiencia limitada en apps de edición.

5.5. Resultados cuantitativos

A continuación se presentan los resultados obtenidos por los cinco participantes en cada una de las tareas planteadas. Se midió el tiempo en segundos desde que el usuario iniciaba la acción hasta que se completaba correctamente.

Tarea	**Usuario A (VR)**	**B (VR)**	**C (VR)**	**D (PC)**	**E (Móvil)**
Borrar escena	7s	4s	9s	5s	8s
Crear coche	10s	6s	14s	8s	12s
Escalar modelo	8s	5s	10s	7s	11s
Borrar modelo	16s	8s	20s	10s	17s
Rotar y replicar	10s	7s	12s	6s	13s
3 bloques apilados	23s	18s	30s	20s	27s
Guardar escena	8s	6s	9s	5s	10s

Cuadro 5.1: Tiempos de ejecución por usuario y tarea

5.5.1. Porcentajes de éxito por tarea

Se considera éxito cuando la tarea se completa sin errores ni repeticiones del comando.

- **Borrar escena: 100 %** Todos los usuarios completaron esta tarea sin dificultad, demostrando que la interacción básica de eliminación es intuitiva y funciona correctamente en todas las plataformas probadas (escritorio, móvil y VR).

- **Crear coche: 80 % (fallos de reconocimiento en Usuario A)** La mayoría de los usuarios lograron crear el modelo de coche sin problemas, aunque el Usuario A experimentó dificultades con el reconocimiento de voz en entornos ruidosos, sugiriendo la necesidad de mejorar la robustez del sistema en condiciones acústicas adversas.

- **Editar modelo: 60 % (problemas de interpretación de comandos como ”largo”)** Solo tres de cada cinco usuarios pudieron editar correctamente los parámetros del modelo. Los fallos se concentraron en la interpretación de comandos dimensionales, indicando que se necesita mejorar el prompt para contemplar todos los casos.

- **Borrar modelo: 60 % (dificultades con el posicionamiento del cursor en VR y móvil)** La precisión requerida para seleccionar modelos pequeños resultó problemática en dispositivos móviles y gafas VR, donde los usuarios reportaron frustración con el sistema de apuntado. Esto sugiere la necesidad de implementar ayudas de selección o zonas interactivas más amplias.

- **Rotar y replicar: 80 %** El 80 % de los usuarios logró replicar modelos correctamente mediante el comando de voz. Sin embargo, se identificó un comportamiento inesperado: al decir ”duplicar”, en lugar de crear una sola copia del modelo original (duplicarlo), el sistema generó dos copias adicionales (totalizando tres objetos).

- **Apilar bloques: 100 %** Todos los participantes consiguieron apilar correctamente los tres bloques asignados en la prueba, demostrando una excelente usabilidad del sistema de manipulación de objetos. Los usuarios destacaron especialmente la precisión del sistema de agarre en VR, aunque algunos sugirieron añadir guías visuales para facilitar el alineamiento.

- **Guardar escena: 100 %** El proceso de exportación de la escena fue completado exitosamente por todos los participantes, confirmando que el flujo de trabajo para guardar r creaciones está correctamente implementado y es accesible desde todas las plataformas.

5.5.2. Media de tiempo por tarea

- Borrar escena: 6.6s

- Crear coche: 10s
- Escalar modelo: 8.2s
- Borrar modelo: 14.2s
- Editar y replicar: 9.6s
- Apilar bloques: 23.6s
- Guardar escena: 7.6s

5.6. Resultados cualitativos y opiniones

Usuario A (VR, original): Dificultades con las instrucciones iniciales. Problemas con comandos ambiguos (“largo”), frustración al repetir comandos que no se ejecutaban. Considera que el cursor central en VR puede resultar incómodo. Preferiría una interfaz sin suelo blanco (solucionado posteriormente).

Usuario B (VR): Experiencia fluida. Valora positivamente el control por voz. Sugiere implementar retroalimentación visual tras cada comando. Considera que el botón “Guardar” podría estar también disponible como comando de voz.

Usuario C (VR): Le gusta la idea general, pero tuvo problemas con la precisión al seleccionar objetos. Opinó que mover objetos con la cabeza es poco intuitivo. Le gustaría tener una vista previa del modelo antes de colocarlo.

Usuario D (PC): Buena precisión con el ratón, experiencia rápida gracias a la libertad de movimiento que se proporciona frente a móvil y VR. Sugiere una mejora estética en los botones físicos. Considera muy útil la edición por voz como alternativa a los menús tradicionales.

Usuario E (Móvil): El cursor es difícil de controlar. Requiere varios intentos para activar los botones. Considera interesante la propuesta, pero frustrante en su dispositivo. Sugiere una guía inicial más clara.

5.7. Limitaciones detectadas

- El reconocimiento de voz presenta limitaciones semánticas, especialmente en casos donde palabras similares fonéticamente pueden ser malinterpretadas (por ejemplo, confunde "largo" con "large"), lo cual afecta directamente a la precisión de los comandos ejecutados.
- Se han registrado fallos en la ejecución de comandos válidos, que pueden deberse a problemas de latencia de red o interrupciones temporales en la conexión con el modelo de lenguaje en el backend. Esto genera frustración en el usuario al requerir repetir la orden.
- La interacción espacial mediante cursor es imprecisa en entornos amplios, dificultando el enfoque sobre botones u objetos pequeños, especialmente cuando el usuario está en movimiento o el objeto está demasiado lejos del centro de visión.
- No existe una confirmación visual que indique que un comando ha sido recibido y está siendo procesado, lo que genera ambigüedad y reduce la confianza en el sistema.
- Las tareas que implican edición precisa de objetos (cambiar ancho, alto o rotación) resultaron especialmente difíciles, dado que la posición del usuario no se tiene en cuenta al interpretar los comandos de transformación, lo que puede llevar a resultados inesperados en la manipulación del modelo.
- Durante las pruebas se produjeron errores en la edición y creación de entidades debido a los límites de uso impuestos por proveedores como Groq, cuyo acceso gratuito restringe el número de peticiones concurrentes al modelo.
- En dispositivos móviles o en VR, la experiencia es menos fluida debido a la ausencia de métodos eficaces para navegar por la escena.

5.8. Satisfacción general

Se preguntó a los usuarios su grado de satisfacción en una escala de 1 a 5 respecto a los siguientes aspectos:

Aspecto	**A**	**B**	**C**	**D**	**E**	**Media por Aspecto**
Facilidad de uso	3	5	3	4	2	3.4
Reconocimiento de voz	2	4	3	4	3	3.2
Interfaz visual	2	4	3	3	2	2.8
Precisión de edición	3	4	3	4	3	3.4
Experiencia general	3	5	3	4	3	3.6
Media por usuario	2.6	4.4	3	3.8	3.2	**Media Usuarios = 3.4**

Cuadro 5.2: Grado de satisfacción (1=muy baja, 5=muy alta)

Del análisis de la tabla de grado de satisfacción (Cuadro 5.2), se observa que los aspectos mejor valorados por los usuarios son la **experiencia general** (media de 3.6) y la **precisión de edición** (media de 3.4), seguidos muy de cerca por la **facilidad de uso** (también con 3.4). Estos resultados sugieren que el sistema ha logrado proporcionar una experiencia satisfactoria, especialmente en términos de interacción intuitiva y la capacidad para editar modelos 3D de manera eficaz.

Por otro lado, los aspectos con menor puntuación son la **interfaz visual**, con una media de 2.8, y el **reconocimiento de voz**, con 3.2. Aunque esta última puntuación no es especialmente baja, sí indica cierto margen de mejora en la fiabilidad o naturalidad del reconocimiento. Además, si bien la puntuación global por usuario es en promedio de 3.4, se aprecia una ligera dispersión, con valores que oscilan entre 2.6 y 4.4. Esto refleja diferencias en la percepción individual de la experiencia, posiblemente relacionadas con la familiaridad con la tecnología o las expectativas de cada usuario.

5.9. Visualización de resultados

5.9.1. Tiempo medio por tarea

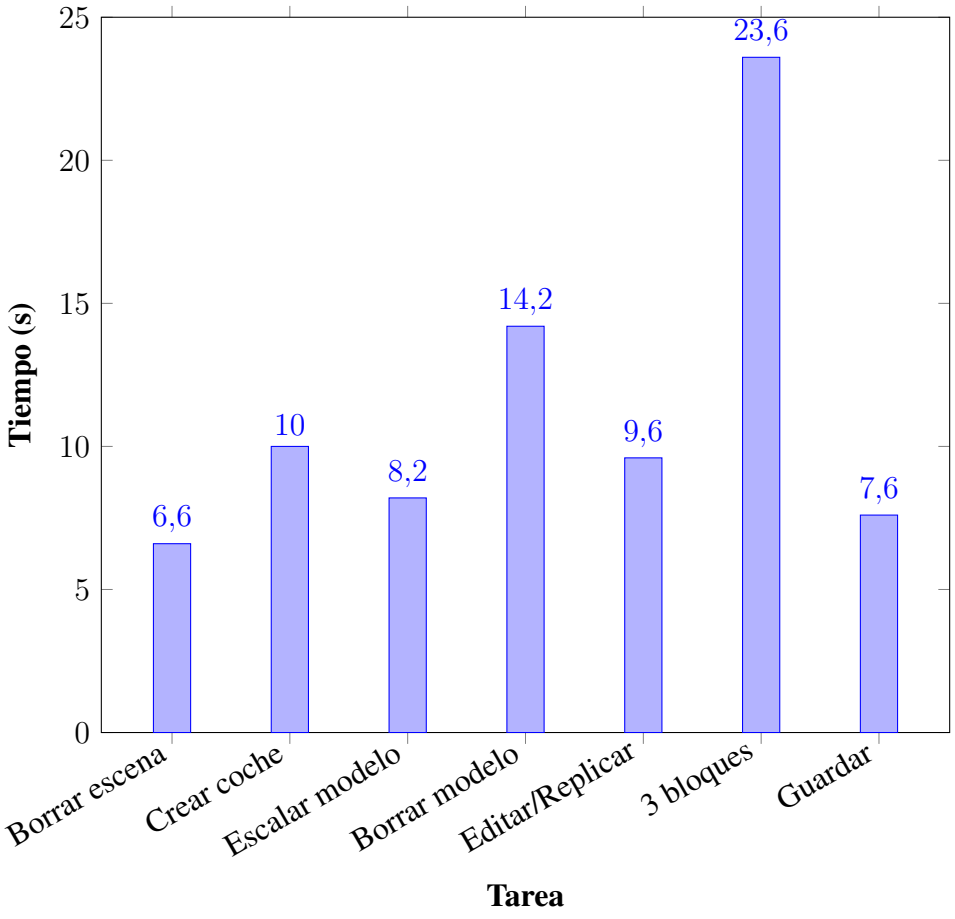


Figura 5.1: Tiempo medio de ejecución por tarea

5.9.2. Grado de satisfacción por aspecto

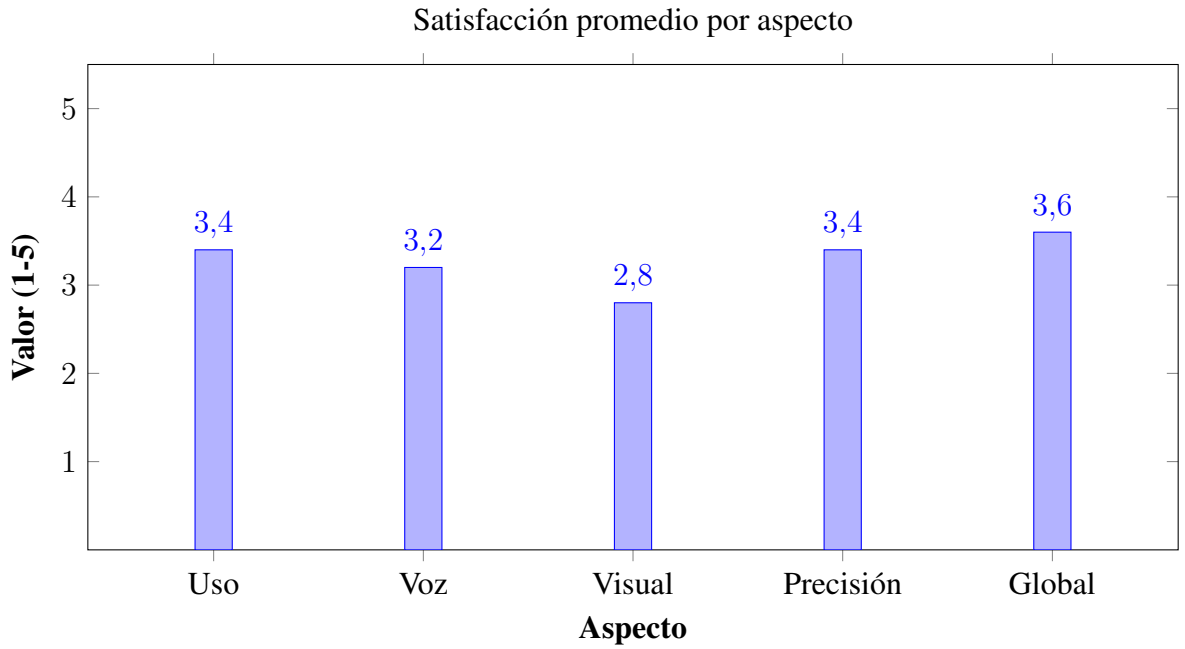


Figura 5.2: Promedio de satisfacción por aspecto

5.10. Conclusiones del experimento

El sistema ha demostrado ser funcional en diferentes dispositivos, con un tiempo de aprendizaje corto y gran potencial para tareas creativas. El control mediante lenguaje natural se percibe como innovador y útil, aunque con margen de mejora en la interpretación de comandos y la retroalimentación visual/auditiva. La experiencia es más fluida en PC, mientras que en móvil y VR presenta limitaciones claras.

Las tareas que implicaban mayor dificultad fueron aquellas relacionadas con la modificación del ancho, largo o rotación de los objetos. Esto se debe, en gran parte, a que actualmente el sistema no tiene en cuenta la posición del usuario respecto al objeto en edición, lo que dificulta interpretar correctamente transformaciones espaciales relativas.

Además, durante el experimento se registraron algunos fallos de edición atribuibles a los límites de uso impuestos por la plataforma Groq en cuentas gratuitas de desarrollador, lo que

ocasionó respuestas incompletas o tiempos de espera excesivos en momentos puntuales.

Para ser un primer prototipo funcional, desarrollado con recursos limitados y sin acceso a modelos comerciales de última generación, los resultados del experimento indican una buena acogida por parte de los usuarios y un alto potencial de evolución. Cabe destacar que la latencia en la comunicación con el servidor de transcripción y procesamiento de lenguaje natural ha influido de forma considerable en la fluidez de la interacción. La adopción de modelos más potentes o dedicados, junto con una infraestructura de baja latencia, podría incrementar notablemente la satisfacción del usuario final.

Estos hallazgos refuerzan la viabilidad de este tipo de interfaces para futuras aplicaciones interactivas, y sientan las bases para iteraciones posteriores que mejoren tanto la capacidad de respuesta del sistema como su precisión en contextos inmersivos.

Capítulo 6

Conclusiones

6.1. Resumen general

El presente Trabajo de Fin de Grado[1] ha tenido como objetivo el desarrollo de una interfaz interactiva para la creación y edición de objetos tridimensionales en un entorno de realidad virtual, mediante comandos de voz. El resultado ha sido una solución funcional, inmersiva y multiplataforma que integra tecnologías emergentes como la Realidad Virtual (VR) y la Inteligencia Artificial (IA), permitiendo al usuario manipular modelos 3D de manera natural, intuitiva y expresiva.

Más allá de la funcionalidad específica implementada, este proyecto representa un primer paso hacia la construcción de una arquitectura base adaptable, desde la cual es posible diseñar nuevas aplicaciones en VR donde la interacción mediante lenguaje natural cobre un papel protagonista. La combinación de sistemas de reconocimiento de voz con capacidades de interpretación semántica habilita formas de interacción mucho más cercanas al comportamiento humano, eliminando barreras técnicas tradicionales y facilitando experiencias inmersivas accesibles a usuarios sin formación técnica.

El sistema implementado no solo permite crear, escalar, rotar o eliminar modelos 3D, sino

[1] URL del Trabajo de Fin Grado: https://vescoth.github.io/tfg-site/

que también ofrece una experiencia adaptable a distintos dispositivos (PC, móviles y visores VR), con especial enfoque en la accesibilidad y la naturalidad de uso. El trabajo demuestra que la integración de VR e IA puede derivar en herramientas versátiles, con potencial para su aplicación en campos tan diversos como la educación, el diseño, la simulación o la visualización de datos.

En definitiva, este proyecto sienta las bases para el desarrollo de entornos interactivos inteligentes, donde la voz se convierte en un medio principal de comunicación con el sistema, y donde el usuario puede construir y modificar escenarios virtuales de forma fluida, expresiva y directa.

6.2. Esfuerzo y recursos dedicados

El desarrollo de este proyecto ha requerido una inversión significativa de tiempo, esfuerzo y recursos tanto técnicos como personales. A continuación, se presenta una estimación detallada del tiempo invertido por fases y tareas:

6.2.1. Distribución temporal por sprint

Sprint	Tareas principales	Horas estimadas
Sprint 1	Familiarización con A-Frame, primeras pruebas con entidades primitivas y creación de una escena inicial básica.	12 h
Sprint 2	Evaluación de tecnologías, pruebas de compatibilidad e integración de tecnologías de voz y transcripción.	18 h
Sprint 3	Benchmarking de modelos, diseño de esquema JSON y generación de instrucciones con LLM	12 h
Sprint 4	Implementación y desarrollo del flujo para la creación y transformación de entidades 3D e incorporación de modelos GLB/GLTF.	28 h
Sprint 5	Optimización de las llamadas a api y eliminación de limites de grabación	16 h
Sprint 6	Desarrollo de *drag & drop* y mejora en el control de la cámara.	14 h
Sprint 7	Implementación de carga, guardado y borrado de escenas	14 h
Sprint 8	Implementación de una función para replicar modelos	16 h
Total Desarrollo		**130 h**

Cuadro 6.1: Tiempo aproximado dedicado a cada sprint

6.2.2. Tiempo de aprendizaje previo y formación autodidacta

Antes de comenzar el desarrollo activo del proyecto, se dedicaron aproximadamente **30 horas** al estudio y experimentación con tecnologías clave como A-Frame, MediaStream Recording API, integración con Groq y uso de formatos glTF/glb.

6.2.3. Preparación de la memoria

La redacción, edición y organización de la memoria del proyecto requirió aproximadamente **40 horas**, incluyendo la recopilación de referencias, generación de diagramas y revisión del formato académico.

6.2.4. Estimación total de dedicación

- Tiempo de desarrollo técnico: **130 h**
- Aprendizaje y pruebas previas: **30 h**
- Redacción de la memoria: **40 h**

Total estimado de horas dedicadas al TFG: 200 horas.

6.3. Lecciones aprendidas y principales problemas resueltos

Durante el desarrollo del proyecto se han extraído una serie de aprendizajes clave que no sólo han permitido alcanzar los objetivos propuestos, sino también mejorar las competencias técnicas y organizativas del autor:

6.3.1. Experiencias documentadas

- **Diseño iterativo y refactorización constante:** La transición hacia componentes reutilizables mediante `A-Componentes` fue clave para escalar y mantener el proyecto de forma limpia y modular.
- **Gestión de errores y tolerancia a fallos:** Muchos errores fueron detectados durante la integración entre reconocimiento de voz, inferencia con Groq y renderizado 3D. La implementación de flujos asíncronos, validaciones y pruebas incrementales permitió mejorar la robustez del sistema.

- **Interacción natural en VR:** Adaptar la interacción a dispositivos como Meta Quest 3 exigió entender sus limitaciones (como la falta de soporte para Web Speech API) y trabajar con alternativas viables como MediaStream + Whisper.
- **Optimización de recursos:** Se implementó un sistema de activación por palabra clave ("Gema") que evita llamadas innecesarias al modelo durante interacciones no intencionadas, reduciendo el coste computacional y mejorando la eficiencia.
- **Claridad en los formatos de entrada y salida:** Definir con precisión el esquema JSON para la comunicación con los modelos de lenguaje fue crucial para interpretar correctamente las instrucciones del usuario y garantizar una ejecución consistente.
- **Equilibrio entre funcionalidad y experiencia de usuario:** Se aprendió que una buena experiencia no se basa únicamente en ofrecer muchas capacidades, sino en garantizar una interacción fluida, natural y predecible, especialmente en entornos sin interfaz textual.
- **Integración de componentes externos:** Aunque herramientas como Whisper, Groq y A-Frame aceleran el desarrollo, su integración exige atención a compatibilidades, formatos y flujos asincrónicos para evitar conflictos y cuellos de botella.
- **Importancia del testeo continuo:** Realizar pruebas frecuentes y resolver errores en fases tempranas facilitó el mantenimiento, redujo problemas acumulativos y permitió iterar de forma ágil en cada sprint.

6.3.2. Valoración de funcionalidades implementadas

El resultado es un sistema funcional capaz de crear, editar, mover y replicar objetos 3D dentro de un entorno WebXR, empleando únicamente la voz del usuario o interacciones básicas mediante ratón o controladores VR.

El proyecto ha demostrado que es viable controlar un entorno 3D utilizando lenguaje natural sin depender de interfaces gráficas tradicionales, lo cual representa un paso hacia sistemas más accesibles y usables, especialmente en plataformas emergentes como gafas de realidad aumentada o virtual. La combinación de manipulación directa y control por voz ha resultado

especialmente efectiva para mantener una experiencia inmersiva, sin distracciones ni barreras técnicas para el usuario final.

6.4. Impacto de asignaturas cursadas en la carrera

El desarrollo de este TFG ha sido posible gracias a la base de conocimientos adquirida en múltiples asignaturas, aunque también ha revelado algunas carencias en el plan formativo actual:

6.4.1. Conocimientos útiles aplicados

- **Lenguajes y programación web:** HTML, JavaScript y conceptos de desarrollo frontend, aprendidos en asignaturas como *Servicios Telemáticos* y *Aplicaciones Telemáticas*, fueron esenciales.
- **Estructuras de datos y arquitectura software:** Aunque el proyecto no requiere estructuras de datos complejas, los principios de diseño modular y patrones de organización han sido fundamentales.
- **Metodologías ágiles:** El enfoque basado en sprints con revisión iterativa fue facilitado por la experiencia obtenida en prácticas y asignaturas de ingeniería telemática.

6.4.2. Conocimientos que se han tenido que adquirir

- **Desarrollo en realidad virtual:** El uso de A-Frame y entornos inmersivos no formaba parte del conjunto de conocimientos adquiridos en el grado, y requirió una fase de autoformación específica.
- **Reconocimiento de voz e integración con modelos de lenguaje:** El uso de Whisper y Groq, así como el diseño de prompts y el procesado de sus salidas, fueron aprendizajes realizados completamente de forma autodidacta.

- **Interacción y control en entornos 3D en tiempo real:** Se adquirieron conocimientos sobre técnicas para la selección, manipulación y transformación dinámica de objetos 3D dentro de la escena, incluyendo la sincronización de eventos de usuario con actualizaciones visuales inmediatas.

6.5. Resumen de lo conseguido y comparación con otros sistemas

El sistema desarrollado es una herramienta funcional e innovadora para la creación y edición de escenas 3D mediante lenguaje natural en un entorno de realidad virtual. A través del uso de reconocimiento de voz, IA generativa y tecnologías web como A-Frame, se ha conseguido un flujo de trabajo accesible y flexible que permite a usuarios sin conocimientos técnicos manipular objetos 3D de forma intuitiva.

6.5.1. Funcionalidades logradas

- Creación de objetos 3D por comandos de voz.
- Edición en tiempo real de propiedades como posición, escala, rotación y apariencia usando lenguaje natural.
- Reposicionamiento preciso de objetos 3D mediante *drag and drop*.
- Persistencia de escenas a través de archivos HTML descargables.
- Flujo de control optimizado por activación de comandos mediante palabra clave.

6.5.2. Comparación con otros sistemas

Aunque existen entornos de edición 3D profesionales (Blender, Unity), estos requieren altos conocimientos técnicos y no están diseñados para control por lenguaje natural. Algunos proyec-

tos recientes permiten generación 3D por texto, pero no suelen estar integrados en entornos de VR interactiva. La principal ventaja del sistema desarrollado es su accesibilidad y la interacción directa desde un navegador web sin instalación de software adicional.

6.6. Estimación de presupuesto y planificación temporal

Aunque el proyecto no ha requerido un presupuesto económico explícito, se estiman las siguientes necesidades si se deseara escalar o mantener de forma profesional:

- **Servicios de IA (Groq/Whisper):** uso gratuito en fase de desarrollo, pero sujeto a costes en entorno productivo.
- **Dispositivo de prueba (Meta Quest 3):** en torno a 550€.
- **Hosting y almacenamiento web (opcional):** a partir de 5€/mes.

6.6.1. Diagrama de GANTT resumido

Tarea	Feb	Mar	Abr	May	Jun	Jul
Estudio de tecnologías						
Desarrollo por sprints						
Pruebas en VR						
Documentación						
Revisión y mejoras						

6.7. Trabajos futuros

- **Integración multimodal:** Incluir entrada por gestos o comandos táctiles además de voz.

- **Mejora en la precisión de edición:** Implementar mecanismos de corrección de órdenes y validación visual de acciones.
- **Soporte para múltiples usuarios:** Posibilitar colaboración en tiempo real dentro de la escena.
- **Ampliación de comandos:** Incorporar funciones más avanzadas como animaciones, propiedades físicas o interacciones entre objetos.
- **Persistencia en la nube:** Guardado y carga de escenas directamente desde almacenamiento remoto.
- **Eliminación de botones:** Transicionar hacia un sistema de voz íntegro sin necesidad de tener que pulsar ningún botón.
- **Mejorar feedback del sistema:** Integrar un sistema de retroalimentación visual que indique cuándo la orden ha sido correctamente entendida.
- **Ampliar versatilidad del sistema:** Añadir más opciones de edición con la voz (textura, color, materiales).
- **Mejorar sistema de almacenamiento** Implementar una interfaz visual para gestionar múltiples escenas guardadas.
- **Reposicionamiento adaptativo:** Identificar la posición del usuario con respecto al modelo a editar para mejorar la precisión de los comandos de reposicionamiento. Por ejemplo: ”Muévelo a la derecha”, ”Gíralo a la izquierda”, etc...

Bibliografía

[1] Autodesk. Tinkercad. https://www.tinkercad.com/, 2025. Herramienta online para modelado 3D.

[2] Carlos González González. Web XR: Explorando la Realidad Extendida en la Web. https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API, 2023. Realidad extendida en navegadores.

[3] Clara.io. Full-featured cloud-based 3D modeling, animation and rendering software. https://clara.io/, 2023. Editor 3D online.

[4] Google DeepMind. Gemma: Open models for responsible AI. https://deepmind.google/models/gemma/, 2024.

[5] DeepSeek. DeepSeek AI: Large Language Models for Open Innovation. https://deepseek.com/, 2024. Proveedor de modelos de lenguaje abiertos.

[6] Estudiantes del Laboratorio de Colaboración en XR de Cornell Tech. Voice2Action – VR Multimodal Interaction with LLM Agents. https://yang-su2000.github.io/Voice2Action/, 2024. Proyecto académico que explora la interacción multimodal en entornos de realidad virtual mediante agentes controlados por modelos de lenguaje.

[7] Genonymous. WebGL Public Wiki. https://www.khronos.org/webgl/wiki/Main_Page, 2022. Gráficos 3D en navegadores.

[8] Groq. AI inference at the speed of thought. https://console.groq.com/docs/overview, 2024.

[9] JanusXR. JanusWeb: An open source 3D social platform for the web. https://janusxr.org/, 2023. Plataforma de interacción social en 3D.

[10] MDN Web Docs. MediaStream Recording API. https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API, 2024.

[11] MDN Web Docs. WebXR Device API. https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API, 2024.

[12] MDN Web Docs. HTML: HyperText Markup Language. https://developer.mozilla.org/en-US/docs/Web/HTML, 2025.

[13] MDN Web Docs. WebGL Api. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API, 2025.

[14] Microsoft. Visual Studio Code. https://code.visualstudio.com/, 2023.

[15] Mozilla. A-frame: A web framework for building virtual reality experiences. https://aframe.io/, 2023.

[16] Mozilla. A-Painter: Paint in VR with A-Frame. https://aframe.io/a-painter/, 2023. Ejemplo de creación artística en VR.

[17] OpenAI. ChatGPT. https://chat.openai.com/, 2023. Asistente conversacional basado en IA.

[18] OpenAI. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision. https://github.com/openai/whisper, 2025.

[19] Ricardo Cabello. Three.js: Javascript 3d library. https://threejs.org/, 2023.

[20] Runway. Creative tools powered by artificial intelligence. https://runwayml.com/, 2024. Herramientas de generación creativa con IA.

[21] Spline. Design in 3D with Spline and AI. https://spline.design/ai, 2024. Plataforma para diseño 3D en tiempo real.

[22] W3Schools. JavaScript Tutorial. https://www.w3schools.com/js/, 2023.

[23] Wikipedia contributors. Large language model — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Large_language_model, 2024.