UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una aplicación de conducción inmersiva usando chaleco táctil y guantes con retroalimentación háptica

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: García Tomás, Mario

Tutor/a: Gómez Barquero, David

Director/a Experimental: Hernandez Gobertti, Fernando Agustin

CURSO ACADÉMICO: 2023/2024

Resumen

El presente proyecto tiene como objetivo desarrollar una experiencia inmersiva en realidad virtual (VR) que combine estímulos auditivos, visuales y hápticos para crear una experiencia de conducción realista y atractiva. La experiencia se implementará utilizando la plataforma Unity en conjunto con los SDKs disponibles de bHaptics para interacción con chalecos y guantes hápticos. La experiencia desarrollada servirá de modelo para futuras aplicaciones en diferentes campos, como la formación, el entretenimiento y la simulación.

Principalmente, se creará una pista genérica en VR donde el usuario podrá moverse en un vehículo virtual por un circuito preestablecido. La pista incluirá obstáculos hápticos que se detectarán mediante un chaleco háptico, proporcionando al usuario una sensación de impacto físico al colisionar con ellos.

Por último, se integrarán guantes hápticos para nuevos obstáculos y permitir al usuario sentir el interior del vehículo virtual. El seguimiento de las manos permitirá detectar la interacción del usuario con objetos virtuales dentro y fuera del vehículo, proporcionando una experiencia más realista e inmersiva.

Palabras clave: Realidad virtual; Retroalimentación háptica; Unity; Guantes hápticos; Chaleco háptico.

Resum

El present projecte té com a objectiu desenrotllar una experiència immersiva en realitat virtual (VR) que combine estímuls auditius, visuals i hàptiques per a crear una experiència de conducció realista i atractiva. L'experiència s'implementarà utilitzant la plataforma Unity en conjunt amb els SDKs disponibles de bHaptics per a interacció amb jupetins i guants hàptics . L'experiència desenrotllada servirà de model per a futures aplicacions en diferents camps, com la formació, l'entreteniment i la simulació.

Principalment, es crearà una pista genèrica en VR on l'usuari podrà moure's en un vehicle virtual per un circuit preestablit. La pista inclourà obstacles hàptics que es detectaran mitjançant un jupetí hàptic, proporcionant a l'usuari una sensació d'impacte físic en col·lidir amb ells.

Finalment, s'integraran guants hàptics per a nous obstacles i permetre a l'usuari sentir l'interior del vehicle virtual. El seguiment de les mans permetrà detectar la interacció de l'usuari amb objectes virtuals dins i fora del vehicle, proporcionant una experiència més realista i immersiva.

Paraules clau: Realitat virtual; Retroalimentació hàptica; Unity; Guants hàptics; Jupetí hàptic.

Abstract

This project aims to develop an immersive virtual reality (VR) experience that combines auditory, visual and haptic stimulation to create a realistic and engaging driving experience. The experience will be implemented using the Unity platform in conjunction with the available bHaptics SDKs for interaction with haptic vests and gloves. The experience developed will serve as a model for future applications in different fields, such as training, entertainment and simulation.

Mainly, a generic track will be created in VR where the user will be able to move in a virtual vehicle along a pre-established circuit. The track will include haptic obstacles that will be detected by a haptic vest, providing the user with a sensation of physical impact when colliding with them.

Lastly, haptic gloves will be integrated for new obstacles and allow the user to feel the interior of the virtual vehicle. Hand tracking will allow detecting the user's interaction with virtual objects inside and outside the vehicle, providing a more realistic and immersive experience.

Key Words: Virtual reality; Haptic feedback; Unity; Haptic gloves; Haptic vest.

Resumen ejecutivo

La memoria del TFG del "Desarrollo de una aplicación de conducción inmersiva usando chaleco táctil y guantes con retroalimentación háptica" debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la Realidad Virtual.

CONCEPT (ABET)	**CONCEPTO (ABET)**	**¿Cumple? (S/N)**	**¿Dónde? (página/s)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Restricciones (normas, códigos, necesidades, requisitos y especificaciones)	S	19 - 23
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	2
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	30
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	46 - 48
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Cumplimiento de objetivos	S	50 - 51
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Impacto global y alcance (contribuciones y recomendaciones prácticas)	S	53 - 54

Índice

Capítulo 1. Introducción 1

1.1 Contexto y motivación 1

1.2 Objetivos 2

1.3 Estado del Arte 2

1.3.1 Historia VR 2

1.3.2 Historia tecnología háptica 4

Capítulo 2. Marco Teórico 8

2.1 Motor de videojuego 8

2.2 Dispositivos hápticos 11

2.3 Realidad virtual 15

Capítulo 3. Tecnologías utilizadas 19

3.1 Entorno de desarrollo 19

3.2 Meta Quest Pro 20

3.3 Dispositivos hápticos 21

3.3.1 BHapticsPlayer [32] 23

3.3.2 BHaptics Designer [33] 25

3.3.3 BHaptics Developer [34] 26

3.4 5G Robot Race 28

Capítulo 4. Desarrollo 30

4.1 Integración dispositivos hápticos 30

4.2 Circuito 31

4.3 Integración VR 34

4.4 Movimiento 35

4.4.1 Primera forma 35

4.4.2 Segunda forma 37

4.5 Colisiones 38

4.6 Obstáculos 40

4.6.1 Barreras 40

4.6.2 Torre de conos 41

4.6.3 Grava 42

4.6.4 Badén de velocidad 43

4.6.5 Lluvia 43

4.7 Interacción de las manos 44
4.7.1 Colisión de manos 45
4.7.2 Agarre de objetos 46
4.8 Escena inicial 48
4.9 Sonido 49
Capítulo 5. Resultados 50
Capítulo 6. Trabajo futuro 52
Capítulo 7. Conclusiones 53
Capítulo 8. Anexo 55
8.1 Enlace al proyecto 55
8.2 Índice de figuras 56
8.3 Bibliografía 58

Capítulo 1. Introducción

1.1 Contexto y motivación

La Realidad Virtual (VR) ha emergido como una de las tecnologías más prometedoras e innovadoras en el campo de la interacción hombre-máquina, ofreciendo experiencias en entornos tridimensionales con los que los usuarios interactúan de manera natural, simulando tanto escenarios ficticios como situaciones que podrían darse en el mundo real.

Esta tecnología permite a los usuarios alcanzar un nuevo nivel de inmersión que puede mejorar numerosos aspectos de la vida cotidiana. El campo del ocio y el entretenimiento es uno de los que más han notado el impacto de la VR. Pero hay otros ámbitos como la medicina, la educación o el turismo que también aprovechan este recurso para sus intereses, aunque estos sean radicalmente distintos.

Sin embargo, una de las limitaciones más significativas de la VR tradicional ha sido la falta de retroalimentación más allá de la visual y auditiva. Para superar esta barrera, la incorporación de tecnología háptica se presenta como una solución que enriquece la experiencia inmersiva al permitir que los usuarios perciban nuevas sensaciones, ampliando así el abanico de posibilidades en la interacción virtual.

Este Trabajo de Fin de Grado (TFG) se centra en el diseño y desarrollo de una aplicación de Realidad Virtual que integra elementos hápticos, con el objetivo de explorar y demostrar cómo la retroalimentación táctil puede mejorar la experiencia del usuario en entornos virtuales. La aplicación desarrollada permite al usuario interactuar con objetos virtuales, no solo a través de la visión y el sonido, sino también mediante la percepción táctil, gracias al uso de dispositivos hápticos avanzados.

La motivación principal de este proyecto es investigar y documentar cómo la integración de tecnología háptica en aplicaciones de VR puede influir en la percepción del realismo, la inmersión y la satisfacción del usuario, así como conocer los límites actuales de estas tecnologías.

Este trabajo se realizó en colaboración con el iTEAM, más en concreto el laboratorio inmersivo (IMM-Lab). Tanto los dispositivos VR como los dispositivos hápticos usados fueron proporcionados por ellos.

En esta memoria se describirá el proceso de planteamiento, diseño y desarrollo del proyecto. Se mostrarán las diferentes ideas propuestas, así como los distintos problemas encontrados y cómo han sido abordados y resueltos.

1.2 Objetivos

El objetivo general es crear una experiencia inmersiva en la que el usuario pueda interactuar con el entorno haciendo uso de la realidad virtual, así como de los dispositivos hápticos para ensalzar la interactividad con el entorno. Para conseguir esto se deben cumplir varios requisitos:

- Crear un escenario virtual por el que se mueva el usuario con objetos con los que interactuar. Este entorno consta de un recorrido predefinido en el que varios obstáculos se disponen frente al usuario.

- Sumergir al usuario creando una experiencia VR con movimientos fluidos, que no mareen ni saquen de la experiencia.

- Conseguir un seguimiento de manos preciso que permita al usuario ver sus manos y cómo estas influyen en el mundo a su alrededor.

- Integrar tanto el chaleco háptico como los guantes para conseguir una retroalimentación respecto a los eventos que ocurran en el mundo virtual.

Para lograr estos objetivos, se ha llevado a cabo un análisis detallado tanto de las tecnologías VR como de las tecnologías hápticas disponibles, así como su integración técnica en un entorno VR, y se han realizado pruebas para evaluar la efectividad de la retroalimentación háptica.

1.3 Estado del Arte

1.3.1 Historia VR

La Realidad Virtual (VR) ha evolucionado rápidamente desde sus primeros días, donde las experiencias se limitaban a estímulos visuales y auditivos, hacia sistemas cada vez más inmersivos que buscan involucrar cada vez más al usuario.

El concepto de Realidad Virtual no es nuevo. Algunos dispositivos pioneros como el Sensorama datan de mediados de los años 50. [Figura 1] Sin embargo, no fue hasta 1987 que Jaron Lanier, junto a Tom Zimmerman, acuñarían el término “realidad virtual” al desarrollar el “guante de datos”, dispositivo que supuso un gran avance en la realidad virtual con dispositivos hápticos. [1]

Figura 1: Sensorama de Morton Heilig (1950). [1]

En 1963 Hugo Gernsbac desarrolló *The Teleyeglasses* una televisión portátil sujeta a la cabeza como si se tratase de unas gafas. [Figura 2] Aunque no respondía a los movimientos de cabeza, era una forma nueva de ver la televisión. Este aparato nunca llegó al mercado, pero puede considerarse el antecedente de las actuales gafas de VR. Aunque hubo varios intentos de dispositivos VR de grandes empresas como Nintendo o Sega, ninguno fue un gran éxito.

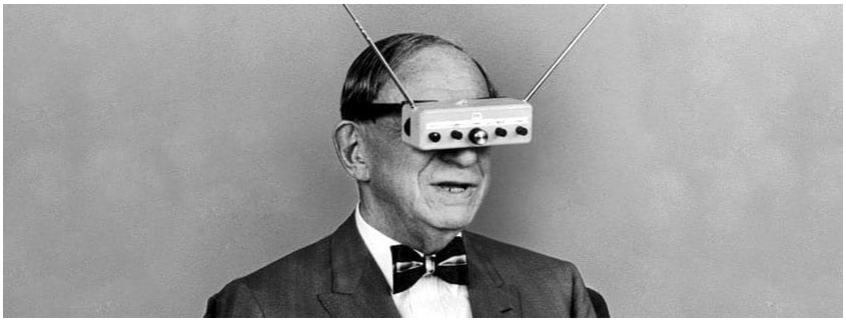


Figura 2: *The Teleyeglasses* de Hugo Gernsbac (1963). [1]

La revolución llegó cuando Palmer Luckey creó el prototipo de gafas *Oculus* en 2012, financiadas a través de la plataforma de microfinanciación Kickstarter. [Figura 3] Estas fueron compradas por Facebook dos años después por 2 billones de dólares. A partir de este momento se disparó la fiebre de la realidad virtual.

Figura 3: Prototipo *Oculus* de Palmer Luckey (2012). [1]

Hoy en día, la VR se utiliza en una amplia variedad de campos. En el entretenimiento, videojuegos como Half-Life: Alyx han demostrado el potencial de la VR para ofrecer experiencias inmersivas sin precedentes. [2] [Figura 4] En medicina, la VR se emplea para la formación de cirujanos, la rehabilitación de pacientes y el tratamiento de fobias. En educación, permite a los estudiantes explorar entornos históricos o científicos de manera interactiva. En el turismo, la VR ofrece la posibilidad de realizar visitas virtuales a destinos turísticos. Estas aplicaciones destacan la versatilidad de la VR, pero también revelan una limitación constante: la falta de retroalimentación táctil.[3]

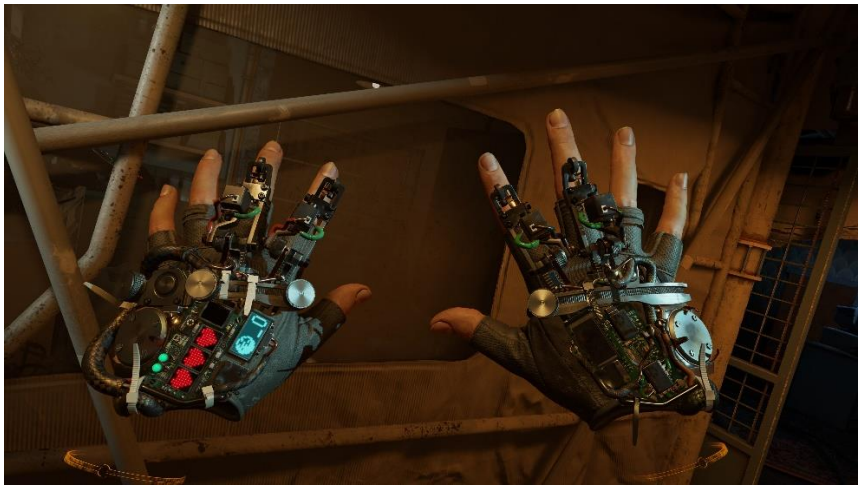


Figura 4: Guantes magnéticos (Russels) de Half-Life: Alyx. [2]

1.3.2 Historia tecnología háptica

La tecnología háptica se refiere al conjunto de interfaces tecnológicos que interaccionan con el ser humano mediante el sentido del tacto, permitiendo crear experiencias al aplicar fuerzas,

vibraciones o movimientos en el usuario. Estas tecnologías se pueden usar para controlar objetos virtuales en simulaciones o para realzar el control de máquinas y dispositivos.[4]

Una de las formas más tempranas de tecnología háptica fue usada en los sistemas servo de ciertos aviones para operarlos remotamente. En un primer momento, estos sistemas solo transmitían la fuerza aplicada a través de un mando al avión, mientras que el piloto no obtenía ninguna respuesta por parte de la aeronave. Para resolver este problema se instaló un sistema que proporcionaba resistencia a la palanca de manejo, transmitiendo las fuerzas a las que se sometía el avión.

Podemos encontrar varios ejemplos de uso en el cine como *The Tingler,* [Figura 5] película estrenada en 1959 que hacía uso de un sistema llamado "Percepto!", un dispositivo vibratorio instalado en las sillas del teatro, que respondía a la acción en pantalla. Otro ejemplo es la película *Honey, I Shrunk the Audience!* de 1994, que usaba pequeños tubos de plástico por los que se bombeaba aire para que se agitasen violentamente, imitando la sensación de las colas de los ratones rozando las piernas de los espectadores.

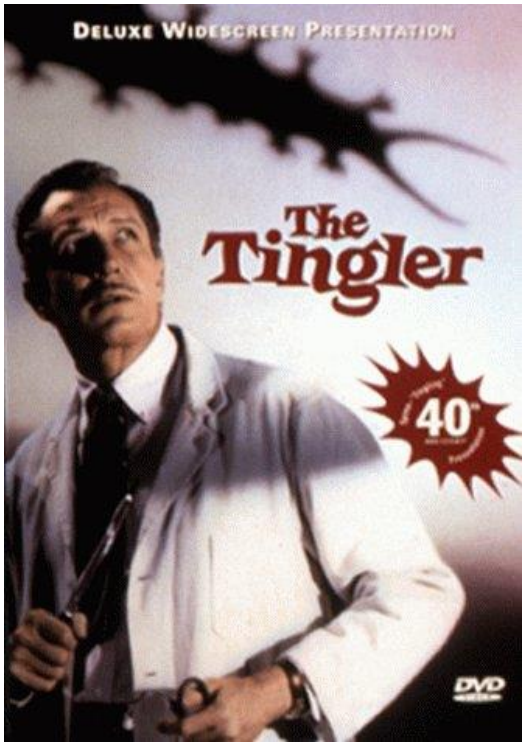


Figura 5: Póster de The Tingler (1959). [4]

En la actualidad existen cada vez más dispositivos hápticos. Una de las formas más sencillas es la vibración que se puede encontrar en muchos controladores diseñados para videojuegos. Es el caso del *DualSense,* [Figura 6] mando desarrollado por Sony para la consola PlayStation 5 que cuenta con dos activadores que producen vibraciones dinámicas que simulan todo tipo de sensaciones. [5]

Figura 6: Mando DualSense de Sony. [5]

Otro ejemplo comúnmente usado en simuladores y videojuegos de conducción es la respuesta de fuerza. En estos casos, el volante usado para controlar el vehículo dispone de motores que dificultan el giro, simulando las fuerzas que se experimentarían en un vehículo real.

Hoy en día están apareciendo nuevas tecnologías que aumentan la sensación que transmite tocar una pantalla. La más común es la vibración al pulsar un botón, pero están en desarrollo tecnologías que cambian la fricción de la pantalla, así como pantallas capaces de inflarse, resaltando ciertas partes de la pantalla.

La combinación de VR con tecnología háptica ha abierto nuevas fronteras en la investigación y desarrollo de aplicaciones interactivas. Los usuarios experimentan una mayor sensación de presencia cuando pueden sentir los objetos con los que interactúan, en comparación con escenarios donde solo están presentes estímulos visuales y auditivos.

Varios estudios demuestran el impacto positivo que significa emplear la tecnología háptica en la realización de tareas complejas en entornos simulados. Un ejemplo de esto es la simulación quirúrgica basada en VR. Los resultados demuestran que la retroalimentación háptica mejora la fidelidad y el realismo, ensalzando el aprendizaje con estos simuladores. [6]

Figura 7: Simulación quirúrgica basada en VR. [7]

Sin embargo, la integración de tecnología háptica en la VR no está exenta de desafíos. Uno de los problemas actuales es que la mayoría de dispositivos hápticos suelen ser costosos y, en algunos casos, voluminosos, lo que limita su adopción masiva. Aun así, el desarrollo de tecnologías hápticas más accesibles y precisas continúa, con el objetivo de ofrecer experiencias más inmersivas y realistas al alcance de todos.

El desarrollo de software y herramientas para la creación de experiencias hápticas en VR también ha avanzado, permitiendo a los desarrolladores diseñar interacciones más precisas. Estas herramientas facilitan la creación de simulaciones realistas que pueden ser utilizadas en una amplia gama de aplicaciones, desde el entretenimiento hasta la formación profesional.

Capítulo 2. Marco Teórico

2.1 Motor de videojuego

Un motor de videojuego hace referencia al conjunto de herramientas que permiten el diseño, la creación y la representación de un videojuego de manera eficiente. Estas herramientas son proporcionadas en un entorno de desarrollo integrado, facilitando la creación de videojuegos de manera rápida y sencilla mediante el uso de una base de datos.[8]

Uno de los componentes clave es el motor de físicas, que simula de manera realista interacciones como colisiones, gravedad, lanzamientos y peso. Además, estos motores deben encargarse del renderizado, gestionando los gráficos del juego, tanto en 2D como en 3D, y permitiendo al programador añadir o modificar elementos gráficos en las escenas o niveles. [9]

Es fundamental que estos motores integren herramientas para la gestión de animaciones, scripts, sonidos, conexiones a internet, gestión de memoria y otros sistemas esenciales del videojuego. Deben contar con una interfaz intuitiva que permita a los programadores acceder rápidamente a todas las funciones, optimizando el flujo de trabajo.

Actualmente, se pueden encontrar una gran variedad de motores gráficos, cada uno con sus propias fortalezas. Entre ellos podemos encontrar motores privados utilizados por grandes compañías, como el RAGE (Rockstar Advanced Game Engine) de Rockstar Games [Figura 8], famoso por la saga GTA, y el Frostbite de EA, empleado en sus populares juegos de deportes. [10]

Figura 8: Logo de Rockstar Games. [10]

También existen muchas posibilidades gratuitas o de código abierto, como pueden ser: Godot, AppGameKit, Game Maker Studio o RPGMaker. Estos suelen ser motores más sencillos, centrados para un tipo de juego en específico o para ciertas plataformas. Aun así, usados correctamente pueden ser útiles para crear grandes juegos. Entre las opciones gratuitas, y en general, los dos motores más destacados son Unreal Engine y Unity.

Unreal Engine [11]

Ampliamente conocido entre los desarrolladores, Unreal Engine fue creado por la empresa de videojuegos Epic Games en 1998 y ha sido actualizado regularmente hasta su versión más reciente: Unreal Engine 5, del que se están empezando a ver los resultados en los primeros videojuegos desarrollados con este motor.

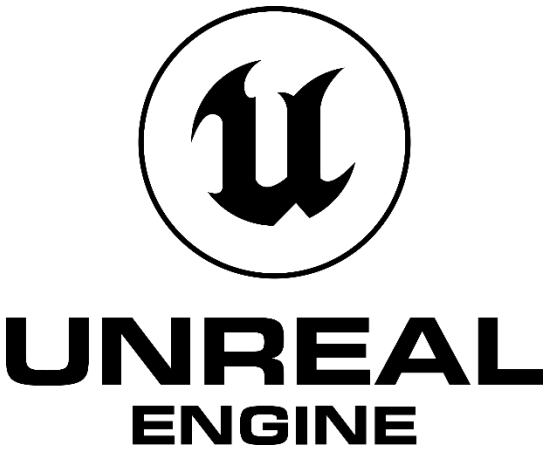


Figura 9: Logo de Unreal Engine. [11]

Está altamente enfocado en el desarrollo en 3D, ofreciendo una potencia gráfica excepcional, lo que lo convierte en una de las mejores opciones del mercado. Siendo usado por muchas desarrolladoras para juegos con los mayores presupuestos en la industria. Uno de los ejemplos más recientes es Black Myth: Wukong que en pocas horas desde su salida se convirtió en uno de los juegos más jugados en Steam, la plataforma de distribución de videojuegos de Valve. [12]

Otra ventaja importante es la amplia difusión de este motor en la industria, lo que ha generado una gran cantidad de tutoriales y guías creadas por la comunidad. Además, Unreal Engine utiliza el lenguaje C++, uno de los más empleados a nivel global en diversos campos, lo que representa una ventaja significativa para quienes han estudiado este lenguaje.

Eso sí, el uso de este motor requiere tener un equipo de alto rendimiento, ya que ocupa mucho espacio y consume mucha memoria.

Unity [13]

Unity es, sin duda, uno de los motores de desarrollo más populares y ampliamente utilizados por desarrolladores a nivel global. Creado por Unity Technologies, este motor se ha consolidado como una herramienta fundamental no solo para la creación de videojuegos, sino también para el desarrollo de aplicaciones en realidad virtual (RV), simulaciones, y experiencias interactivas en una amplia variedad de plataformas.

Figura 10: Logo de Unity Technologies. [13]

El éxito de Unity no solo radica en su versatilidad, sino también en su comunidad activa y en la enorme cantidad de recursos disponibles. Al ser el motor más popular en la industria, la cantidad de tutoriales, documentación, guías y assets disponibles en su tienda proporciona a los desarrolladores, tanto novatos como experimentados, una amplia gama de herramientas y recursos para aprender y mejorar sus habilidades.

Este motor es conocido por su compatibilidad multiplataforma, lo que significa que los desarrolladores pueden crear un juego o una aplicación y desplegarla en una amplia variedad de plataformas, como PC, consolas o dispositivos móviles sin necesidad de rehacer el proyecto desde cero para cada plataforma.

Unity utiliza el lenguaje de programación C#, derivado de C++. Este lenguaje es muy usado hoy en día, esto significa que muchas aplicaciones son compatibles, además de simplificar la búsqueda de información por la cantidad de gente en internet que lo conoce. Su editor es intuitivo y facilita la creación de prototipos rápidos y la iteración, lo que resulta crucial en el desarrollo de videojuegos.

Otro aspecto destacado de este motor es la existencia de la Unity Asset Store, [Figura 11] una plataforma oficial donde los usuarios pueden subir sus creaciones y ponerlas a disposición de la comunidad al precio que consideren adecuado. En esta tienda se pueden encontrar desde modelos de objetos gratuitos hasta paquetes más avanzados que añaden nuevas funcionalidades al motor. Esta posibilidad reduce significativamente el tiempo de desarrollo, facilitando la creación de proyectos más complejos en menos tiempo.[14]

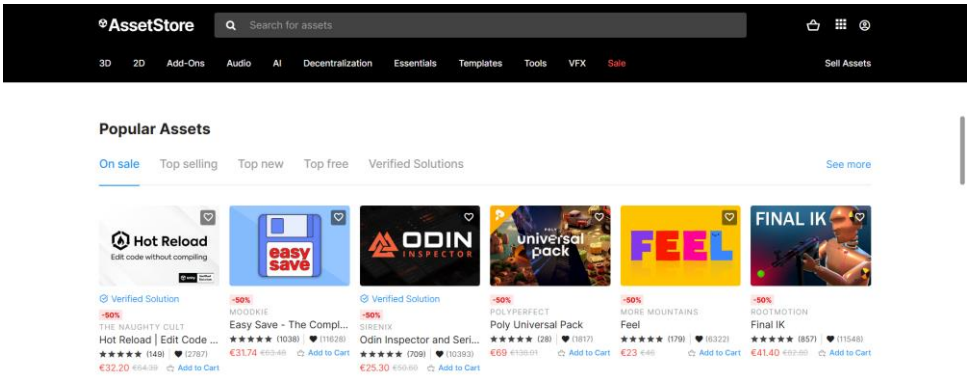


Figura 11: Tienda de Assets de Unity. [14]

2.2 Dispositivos hápticos

Se puede afirmar que la tecnología háptica está en auge, aunque no es algo que se suele experimentar en el uso cotidiano, más allá de la ligera vibración que sentimos al tocar la pantalla de nuestro móvil. Aun así, existen muchas empresas que se dedican al desarrollo de nuevas tecnologías que puedan aplicarse en varios ámbitos. Desde el entretenimiento, hasta la salud o al entrenamiento militar, estas tecnologías demuestran ser muy útiles, aumentando cada vez más su uso.

BHaptics [15]

BHaptics es una empresa surcoreana especializada en tecnología háptica avanzada diseñada para experiencias VR, AR, videojuegos y otros medios interactivos. La compañía es conocida por su gama de dispositivos que permiten a los usuarios sentir físicamente las interacciones en entornos virtuales, como golpes, disparos, vibraciones y más, lo que incrementa el nivel de inmersión en los mundos digitales.

Disponen de un traje completo que incluye chaleco, guantes, y dispositivos hápticos para cabeza, brazos y piernas. [Figura 12] Uno de los productos más complejos es el Tactsuit X40. Un chaleco háptico inalámbrico que cuenta con 40 actuadores distribuidos en el torso. Estos actuadores pueden simular una variedad de sensaciones, desde una simple vibración hasta impactos más complejos

Figura 12: ***Tactosy for Arms*** **de bHaptics.** [15]

Los dispositivos bHaptics son inalámbricos, lo que ofrece mayor libertad de movimiento. Además, son compatibles con varias plataformas como PC, consolas de videojuegos y dispositivos de realidad virtual. La empresa también proporciona herramientas de software, como bHaptics Designer, que permiten a los desarrolladores integrar y personalizar las sensaciones hápticas en juegos y aplicaciones.

HaptX [16]

Esta empresa se especializa en guantes hápticos que permiten al usuario sentir texturas, formas y movimientos en entornos virtuales. Su dispositivo más reciente, los guantes G1, [Figura 13] son dispositivos de última generación diseñados para ofrecer una experiencia táctil inmersiva en entornos de realidad virtual (VR). Utilizan tecnología de microfluidos que permite simular de manera extremadamente realista una amplia gama de sensaciones táctiles, como la textura, la forma y la presión al interactuar con objetos virtuales.

Una de las características más destacadas de los G1 es su capacidad para proporcionar retroalimentación de fuerza. Esto significa que los usuarios no solo pueden sentir la superficie de los objetos, sino también la resistencia cuando interactúan con ellos, lo que añade una capa adicional de realismo a la experiencia virtual.

Figura 13: Ejemplo de uso de los guantes G1 de HaptX. [16]

Además, los guantes G1 cuentan con un sistema avanzado de seguimiento de movimiento que captura con gran precisión los movimientos de las manos y los dedos. Esto asegura que cada gesto y movimiento se refleje de manera exacta en el entorno virtual, mejorando la inmersión y la precisión en aplicaciones de simulación y formación.

Estos guantes están dirigidos principalmente a usos industriales, de simulación y entrenamiento, donde la precisión y el realismo en la interacción táctil son fundamentales. Gracias a su tecnología avanzada permiten a los usuarios experimentar un nivel de inmersión táctil que hasta ahora era difícil de alcanzar en la realidad virtual, haciendo que sean una herramienta valiosa para profesionales en diversas industrias.

Ultraleap [17]

Ultraleap es una empresa líder en la tecnología de seguimiento de manos y retroalimentación háptica, centrada en mejorar la interacción entre humanos y máquinas sin necesidad de controladores físicos. Utilizando su avanzada tecnología de seguimiento de manos, permite a los usuarios controlar dispositivos y aplicaciones con simples gestos, capturando con precisión los movimientos de las manos y dedos en tiempo real.

Figura 14: Leap Motion Controller de Ultraleap. [17]

Una de las innovaciones clave de Ultraleap es su tecnología de retroalimentación háptica basada en ultrasonido, que crea sensaciones táctiles en el aire sin necesidad de contacto físico. Esto permite a los usuarios sentir objetos virtuales y recibir retroalimentación háptica en entornos virtuales o aumentados sin la necesidad de llevar ningún dispositivo adicional, lo que mejora significativamente la inmersión y la interacción.

La empresa está enfocada en redefinir cómo las personas interactúan con la tecnología, eliminando la barrera física y creando experiencias más naturales e intuitivas. Su tecnología se está adoptando en diversas industrias, incluyendo entretenimiento, educación, y atención médica, ofreciendo nuevas formas de interactuar con el mundo digital.

Teslasuit [18]

Teslasuit es la empresa desarrolladora de una innovadora prenda háptica con el mismo nombre que integra tecnología avanzada para proporcionar una experiencia inmersiva total en entornos de realidad virtual. [Figura 15] El traje es capaz de simular sensaciones físicas a través de electroestimulación, permitiendo a los usuarios sentir contacto, presión, y temperatura, lo que lo convierte en una herramienta muy útil para entrenamientos y simulaciones.

Además de sus capacidades hápticas, incluye un sistema de captura de movimiento que registra con precisión los movimientos corporales del usuario en tiempo real, mejorando la interacción en mundos virtuales. También cuenta con sensores biométricos que monitorean las señales vitales,

como la frecuencia cardíaca y los niveles de estrés, lo que permite ajustar las experiencias en función del estado fisiológico del usuario.

Figura 15: Traje háptico Teslasuit. [18]

El traje ha sido adoptado en diversas industrias, incluyendo la formación militar y de primeros auxilios, donde se utiliza para crear simulaciones realistas y seguras. Teslasuit está diseñado para integrarse con plataformas de realidad virtual y aumentada, ofreciendo una combinación única de tecnología háptica, captura de movimiento y biometría que redefine la inmersión y la interacción en el entorno digital.

Tanvas [19]

Tanvas es una empresa innovadora que se especializa en el desarrollo de tecnología háptica para mejorar la interacción táctil en dispositivos digitales. Fundada por un equipo de ingenieros y diseñadores, busca transformar la forma en que las personas experimentan el tacto a través de pantallas táctiles. Su principal producto, TanvasTouch, [Figura 16] permite a los usuarios sentir texturas, formas y patrones a través de superficies planas, brindando una experiencia sensorial más rica y envolvente.

Figura 16: Tanvas Touch Dev Kit. [19]

La tecnología de Tanvas se basa en un enfoque único que combina hardware y software para crear sensaciones táctiles realistas. Utilizando electroestática y una interfaz de usuario intuitiva, sus dispositivos pueden simular una variedad de texturas, desde superficies rugosas hasta suaves. Esto abre un amplio abanico de aplicaciones, desde el diseño industrial y la educación hasta la atención médica y el entretenimiento.

2.3 Realidad virtual

El concepto realidad virtual hace referencia a un entorno de escenas y objetos simulados con apariencia real. Este entorno generado crea en el usuario la sensación de estar inmerso en él. La realidad virtual consiste en la comunicación entre el mundo real y el sistema, consiguiendo que las acciones de entrada del usuario sean procesadas e interpretadas, cambiando el escenario virtual en consecuencia. [20]

De la misma forma, el usuario percibe los cambios en el entorno digital haciendo uso de varias tecnologías. La base de la gran mayoría de aplicaciones inmersivas son las gafas VR. [Figura 17] Estos dispositivos permiten reproducir las imágenes creadas por ordenador en una pantalla muy cercana a los ojos o proyectando directamente estas imágenes sobre la retina de los ojos.

Figura 17: Gafas o casco de realidad virtual. [21]

Existen varios conceptos claves comunes a la mayoría de cascos de realidad virtual. Uno de los más importantes es la resolución de pantalla, de ella depende la definición de las imágenes percibidas por los usuarios. Al igual que el campo de visión, aunque los seres humanos tenemos un campo de visión horizontal de entre 180º a 220º, la mayoría de gafas VR disponen de un campo de visión de entre 110º y 120º, ya que es el rango percibido por ambos ojos. [21]

La latencia es algo muy importante a tener en cuenta. La diferencia de tiempo entre el movimiento del usuario y la reacción de la imagen puede provocar mareos y una menor sensación de realismo si es demasiado alta. Ocurre lo mismo con la tasa de refresco, es necesario un mínimo de 60 FPS para que el ojo perciba las imágenes de manera natural sin provocar mareos.

Estos dispositivos cuentan con un seguimiento de orientación posible mediante sensores internos, como pueden ser giroscopios, acelerómetros, etc. Con ellos es posible detectar la orientación de la cabeza del usuario. También pueden contar con sensores que permiten un seguimiento de posición, logrando situar exactamente al usuario.

Oculus [22]

Una empresa pionera en el desarrollo de tecnología de realidad virtual (VR) es Oculus. Fundada en 2012 por Palmer Luckey, creció rápidamente gracias a su dispositivo Oculus Rift, un visor de realidad virtual que ofrecía una experiencia inmersiva en videojuegos y otras aplicaciones. En 2014, fue adquirida por Facebook, lo que impulsó su desarrollo y expansión en el mercado de la realidad virtual.

Figura 18: Logo de Oculus. [22]

Desde entonces, Oculus ha lanzado una serie de dispositivos, incluidos Oculus Go, la serie Oculus Quest, o las Meta Quest Pro, mejorando la accesibilidad y calidad de la experiencia VR. Estos dispositivos pueden ser usados sin un ordenador que soporte el software, lo que permite a más usuarios acceder a la realidad virtual.

Oculus recibió un cambio de nombre, pasando a llamarse Reality Labs en 2020. [23] Aun así, sigue siendo una de las principales marcas en el mercado de la realidad virtual, creando nuevos dispositivos e impulsando innovaciones bajo el nombre de Meta, ayudando a definir el futuro de esta tecnología.

Valve Index [24]

Valve Index es un sistema de realidad virtual (VR) de alta gama desarrollado por Valve. [Figura 19] Lanzado en 2019, se destaca por su enfoque en ofrecer una experiencia VR premium con características avanzadas, como pantallas LCD con una alta frecuencia de actualización de hasta 144 Hz, lo que proporciona una imagen más fluida y clara. También cuenta con un amplio campo de visión y un ajuste ergonómico para mayor comodidad.

Figura 19: Kit de VR de Valve Index. [24]

Anteriormente, Valve había colaborado con HTC para fabricar las HTC Vive, [Figura 20] estas gafas están diseñadas para ser usadas en una habitación, entrando a un mundo virtual que permite al usuario caminar y utilizar los mandos para interactuar con él. Es un dispositivo diseñado principalmente para SteamVR, siendo compatible con cientos de juegos disponibles en la plataforma. [25]

Figura 20: Gafas VR HTC Vive. [25]

Valve Index incluye controladores *Knuckles* que permiten un seguimiento preciso de los movimientos de la mano y los dedos, ofreciendo una interacción más natural y envolvente en los juegos y aplicaciones VR. Además, el sistema utiliza estaciones base para un seguimiento preciso en 360 grados, lo que mejora la inmersión y la precisión en el espacio virtual.

Este kit es compatible con la plataforma SteamVR y una amplia gama de contenidos, y es considerado uno de los mejores sistemas VR disponibles en el mercado por su calidad de construcción, rendimiento y soporte para experiencias de realidad virtual avanzadas.

HoloLens [26]

HoloLens es un dispositivo de realidad mixta desarrollado por Microsoft que superpone hologramas interactivos en el entorno real del usuario. Lanzado inicialmente en 2016, combina elementos de realidad aumentada (AR) y realidad virtual (VR), permitiendo a los usuarios ver e interactuar con objetos digitales como si estuvieran presentes en su entorno físico.

Cuenta con sensores avanzados, cámaras, y un procesador dedicado que permite el seguimiento preciso del entorno y las manos del usuario sin necesidad de cables ni conexión a un PC. HoloLens se ha utilizado en muchos ámbitos, desde la educación y la medicina hasta la arquitectura y la fabricación, ofreciendo aplicaciones prácticas como diseño 3D, simulaciones, y colaboración remota.

Figura 21: Gafas HoloLens de Microsoft. [26]

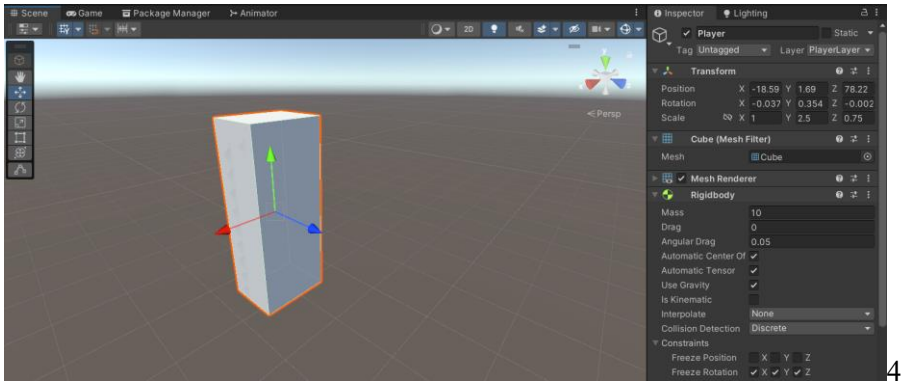
Microsoft lanzó una versión mejorada, HoloLens 2, en 2019, con un campo de visión más amplio, mayor comodidad, y mejor interacción táctil, consolidando su posición como una herramienta muy útil tanto para aplicaciones profesionales como empresariales de realidad mixta.

Capítulo 3. Tecnologías utilizadas

3.1 Entorno de desarrollo

Viendo los motores disponibles, se eligió Unity ya que se tenía experiencia previa y la cantidad de documentación disponible simplifica el desarrollo del entorno virtual. Unity organiza su funcionamiento a través de escenas en las que se encuentran los objetos que componen el espacio virtual de cada una. Por ejemplo, un juego puede utilizar una escena para el menú principal y una para cada nivel, permitiendo que el programa final alterne entre ellas durante la ejecución.

Cada escena de Unity se compone por *GameObjects,* es decir, por objetos. Cada uno contiene información sobre sí mismo y sobre cómo interactúa con su entorno. Esta información viene definida por las componentes del objeto, que se muestran en el lado derecho de la interfaz y definen las características del objeto. [Figura 22] Incluyen desde la posición, rotación y el tamaño del objeto, hasta la textura, masa y como este objeto interactúa con las físicas de la aplicación y del entorno a su alrededor.



[4]

Figura 22: Ventana de componentes de Unity.

Los objetos que agregamos a las escenas se muestran, ordenados según su creación, en la ventana *Hierarchy*, ubicada a la izquierda de la pantalla. En esta ventana, podemos utilizar el *Parenting*, que permite hacer que un objeto sea hijo de otro. Al hacerlo, el objeto hijo hereda cualquier cambio que realicemos en la posición y rotación del objeto padre. Además, esta funcionalidad facilita la organización del proyecto, ya que los objetos hijos pueden ocultarse bajo su padre, lo que ayuda a mantener el proyecto más ordenado y menos confuso. [27]

Figura 23: Pestaña de jerarquías. (Hierarchy)

La principal herramienta para hacer una aplicación en Unity son los scripts. Estos scripts, escritos en C#, son conjuntos de instrucciones que permiten a los objetos ejecutar funciones específicas. Mediante los scripts, se puede modificar cómo los objetos interactúan entre sí, cambiando la información de sus propios componentes o de los componentes de otros objetos. Entre sus funciones, permiten que los objetos respondan a las entradas del usuario, como las teclas presionadas. También se puede generar fuerzas que influyan en el entorno, lo que resulta esencial para programar movimientos.

3.2 Meta Quest Pro

Los dispositivos de realidad virtual disponibles eran las gafas Meta Quest 2, las Meta Quest 3 y las Meta Quest Pro. La diferencia de especificaciones no era un problema puesto que la aplicación no es demasiado compleja. La elección de las Meta Quest Pro fue debido a su mayor disponibilidad en el entorno de trabajo, además de contar con un sistema de agarre más cómodo.

Este casco de realidad virtual salió al mercado el 25 de octubre de 2022, desarrollado por Reality Labs, una división de Meta Platforms. Cuenta con un diseño más delgado y una pantalla que no bloquea completamente la visión periférica del usuario. Permite ajustar la distancia entre las pupilas de las pantallas, al igual que se pueden mover adelante y atrás. [28]

Figura 24: Gafas VR Meta Quest Pro. [29]

Utiliza pantallas LCD con una resolución de 1800×1920 por ojo que cuentan con un rango de color más amplio y un mejor contraste gracias la atenuación local, pudiendo controlar más de 500 bloques LED de forma independiente. La batería está integrada en la parte trasera de la correa de la cabeza para una mejor distribución del peso, con una duración alrededor de las 2 horas por carga. [29]

Para funciones de realidad mixta, Quest Pro utiliza cámaras de alta resolución en color. También incluye sensores internos para el seguimiento de ojos y cara, principalmente para el uso con avatares. Estas gafas incluyen los controladores Touch Pro, que cuentan con un diseño más compacto respecto a anteriores versiones. Disponen de tecnología háptica mejorada y cuentan con nuevos sensores para controlar los movimientos de la mano de una forma más precisa.

3.3 Dispositivos hápticos

Los dispositivos hápticos usados para este proyecto son el chaleco háptico TactSuit X40 y los guantes TactGlove DK2, ambos de la empresa bHaptics. Funcionan mediante actuadores que producen vibración, creando la respuesta háptica que percibe el usuario.

Estos dispositivos son versátiles en términos de compatibilidad. Pueden ser utilizado con una amplia gama de dispositivos, incluyendo sistemas de realidad virtual como Oculus Rift, Oculus Quest, HTC Vive, y Valve Index, además de ser compatible con consolas de videojuegos y ordenadores personales.

Más allá del ámbito de los videojuegos, estos dispositivos tienen aplicaciones potenciales en otros campos como el entrenamiento militar, la simulación médica, y la educación. Por ejemplo, en un entorno de entrenamiento militar, podrían simular las sensaciones de estar en combate, ayudando a entrenar a los soldados en condiciones más cercanas a la realidad.

TactSuit X40

El TactSuit X40 es un chaleco háptico avanzado diseñado específicamente para mejorar la experiencia en realidad virtual (VR), videojuegos, y aplicaciones interactivas. Este dispositivo permite a los usuarios sentir una variedad de sensaciones físicas en el cuerpo, incrementando el realismo y la inmersión en los entornos virtuales.

Cuenta con un total de 40 actuadores de vibración, repartidos equitativamente en la parte frontal y en la espalda del torso. Estos actuadores son capaces de generar patrones de vibración complejos que simulan diferentes tipos de interacciones físicas, como el impacto de balas, golpes, el roce del viento, o la sensación de recibir una descarga eléctrica en un entorno de juego. [30]

Figura 25: Parte frontal del TactSuit X40. [30]

En la parte trasera cuenta con el botón de encendido junto al puerto de carga. Este botón cuenta con una luz LED que muestra si el chaleco está conectado a algún dispositivo o está siendo emparejado. También cuenta con un conector jack de audio en la parte trasera y la batería dura alrededor de 12 horas. El chaleco cuesta 529 $ y viene solo en una talla, aunque puede ajustarse con las tiras que encontramos en los laterales.

Figura 26: Parte trasera del TactSuit X40. [30]

TactGlove DK2

El TactGlove DK2 se enfoca en proporcionar retroalimentación táctil realista en cada dedo, lo que hace que la manipulación de objetos virtuales se sienta más auténtica. La tecnología de bHaptics se centra en la comodidad y la precisión, permitiendo que los usuarios experimenten sensaciones detalladas y responsivas.

Cada guante cuenta con seis actuadores, uno en la punta de cada dedo, y uno en la muñeca, sobre el botón de encendido, esto permite generar sensaciones más precisas en cada dedo, permitiendo una mayor libertad para las sensaciones que creamos. [31]

Figura 27: TactGlove DK2. [31]

En la parte superior cuentan con el botón de encendido y el puerto de carga, que funcionan de la misma forma que en el chaleco. La parte exterior de guante es intercambiable, lo que permite reusarlos en caso de deterioro. La batería en este caso solo dura 3.5 horas, siendo bastante inferior al chaleco. Los guantes cuestan 249 $ y disponen de 4 tallas distintas.

3.3.1 BHapticsPlayer [32]

Para conectar estos dispositivos al ordenador, es necesario instalar la aplicación bHapticsPlayer. Esta aplicación permite conectar todos los dispositivos de bHaptics de manera sencilla usando la conexión bluetooth. No solo sirve para conectar los dispositivos, sino que también es una plataforma de distribución que cuenta con experiencias propias que se pueden descargar y jugar.

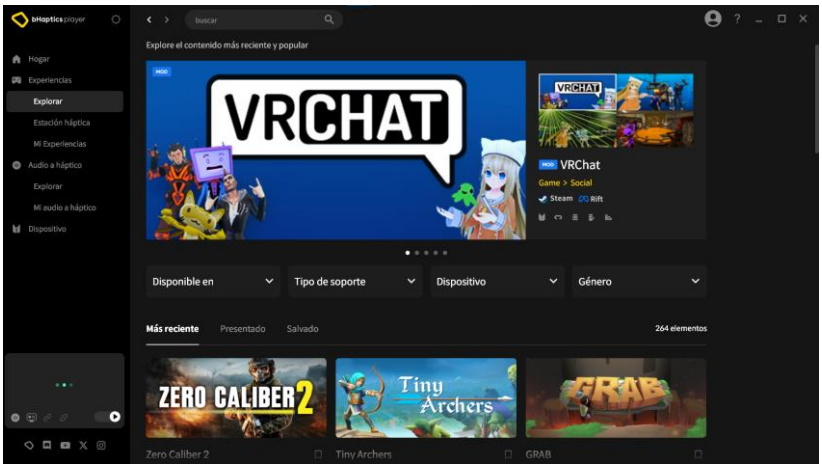


Figura 28: Pestaña "Explorar" de bHapticsPlayer.

Para conectar un dispositivo tenemos que entrar a la pestaña *Dispositivo*, la última opción de la columna izquierda. En caso de tener dispositivos ya emparejados nos aparecerán ahí. Si queremos realizar la conexión por primera vez tendremos que mantener pulsado el botón de encendido del dispositivo que queramos conectar y debería aparecer en la ventana desplegable *Escaneado* que podemos ver en la esquina superior derecha.

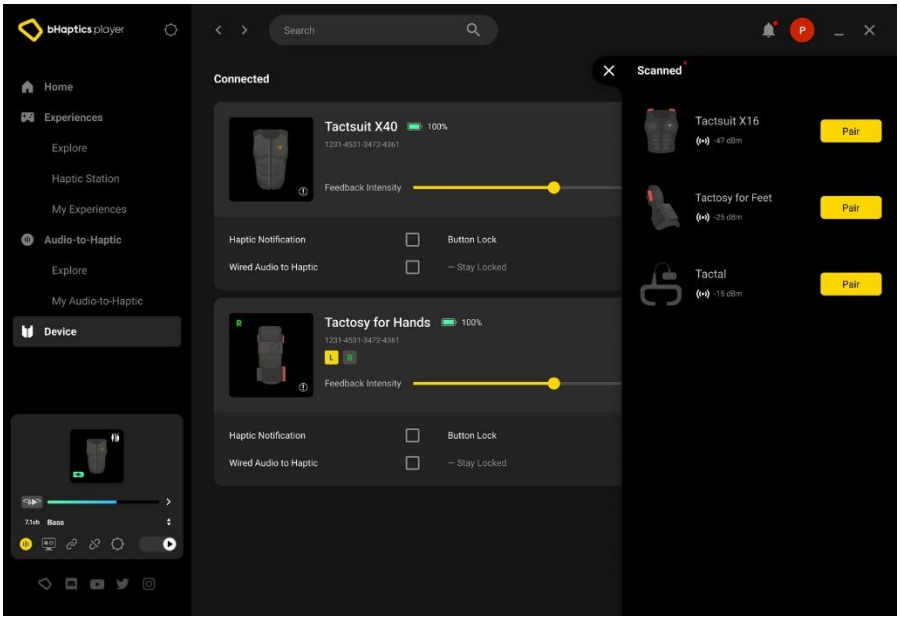


Figura 29: Pestaña "Device" de bHapticsPlayer.

Una vez conectado podemos realizar ajustes en la intensidad con la que queremos que vibre el dispositivo. También podemos realizar pruebas de retroalimentación para comprobar que funcione correctamente, activando cada uno de los actuadores por separado para comprobar que todos están disponibles.

3.3.2 BHaptics Designer [33]

Los dispositivos de bHaptics no ofrecen solo experiencias predefinidas, sino que también permite a los desarrolladores crear sus propias configuraciones de retroalimentación háptica. Esto es posible a través del bHaptics Designer proporcionado por bHaptics, que facilita la integración y personalización de las respuestas hápticas en aplicaciones y juegos.

Esta es una página web que proporciona las herramientas para crear una serie de instrucciones que ejecutar en los dispositivos. Al entrar, se puede observar una lista de proyectos. A la izquierda es posible cambiar entre los creados por uno mismo y los que proporciona bHaptics.

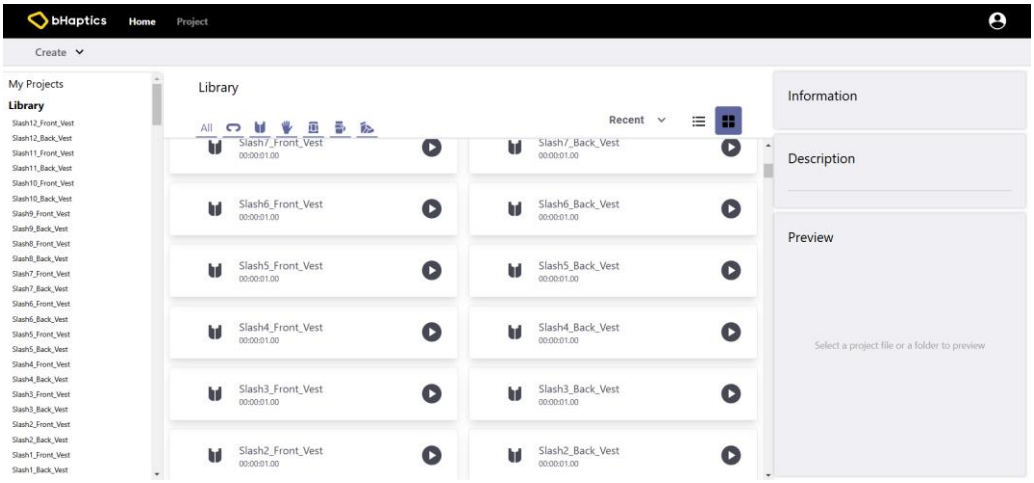


Figura 30: Lista de proyectos de bHaptics Designer.

La página ofrece varias opciones a parte de crear un proyecto, como hacer carpetas para ordenar las sensaciones creadas, o importar experiencias previamente descargadas. Para crear un proyecto nuevo hay que pulsar en el botón *Create* en la esquina superior izquierda y en la opción *Project.* Es necesario elegir el nombre de la experiencia y el dispositivo específico.

Tras esto, nos lleva a la ventana en la que podemos crear la experiencia. Se pueden ver dos matrices de puntos, que cambian dependiendo del dispositivo seleccionado, así como dos pistas abajo. Cada proyecto está dividido en efectos que, a su vez, se pueden dividir en varias capas con distintos patrones cada una.

Para crear los patrones se ponen puntos a lo largo de la matriz. Hay dos modos de crear los efectos, modo por puntos y modo por caminos. [Figura 31] El modo por puntos crea vibraciones estáticas en los puntos seleccionados durante un cierto tiempo, mientras que el modo por caminos crea varios puntos en orden. La vibración empezará en el primer punto e irá recorriendo los demás hasta llegar al final.

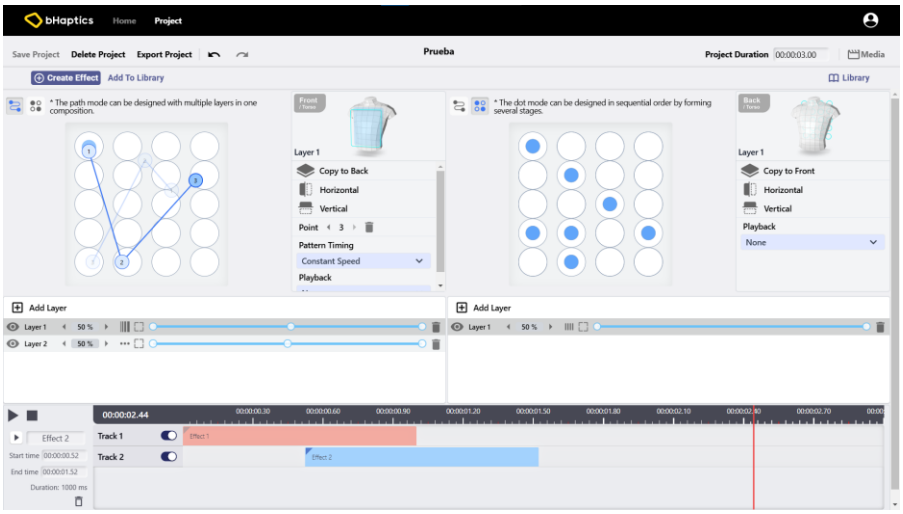


Figura 31: Pestaña de creación de proyecto de bHaptics Designer.

Este editor ofrece muchas más opciones, como cambiar la intensidad de los puntos, o de las capas, cambiar la duración del proyecto, copiar el patrón de la parte delantera a la parte de atrás, etc. Esto invita a jugar con el editor para crear las experiencias que el desarrollador quiera.

3.3.3 BHaptics Developer [34]

Para implementar las experiencias creadas en Unity es necesario hacer uso de la página web bHaptics Developer. [Figura 32] Lo primero que hay que hacer es pulsar el botón *Create* y asignar un nombre a la aplicación. Esta aplicación tendrá dentro los patrones creados en el designer.

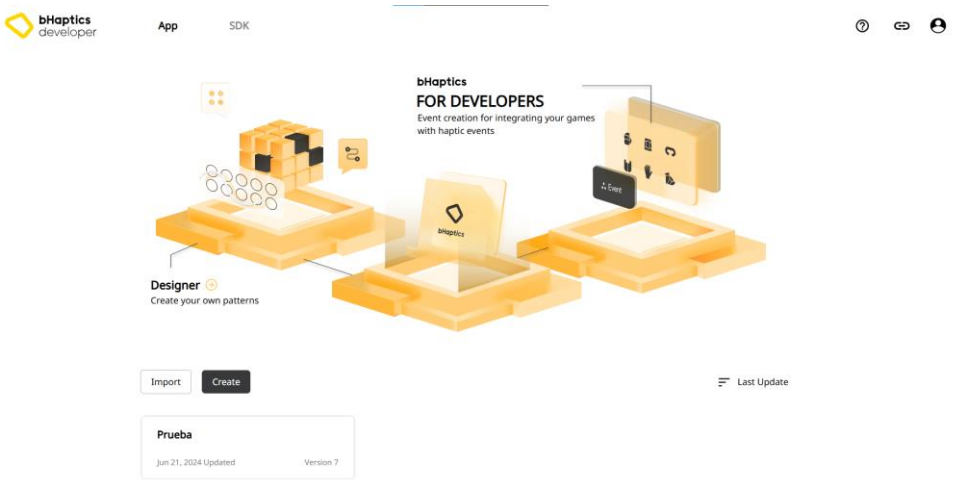


Figura 32: Pestaña inicial de bHaptics Developer.

Estando dentro de la siguiente pestaña, es posible empezar a añadir las experiencias. [Figura 33] Pero antes hay que instalar el paquete de bHaptics en Unity. Una vez instalado, hay que crear una *API Key* desde el Developer para poder conectar la aplicación creada con Unity. Esto se hace desde la pestaña *API Key* de la columna izquierda.

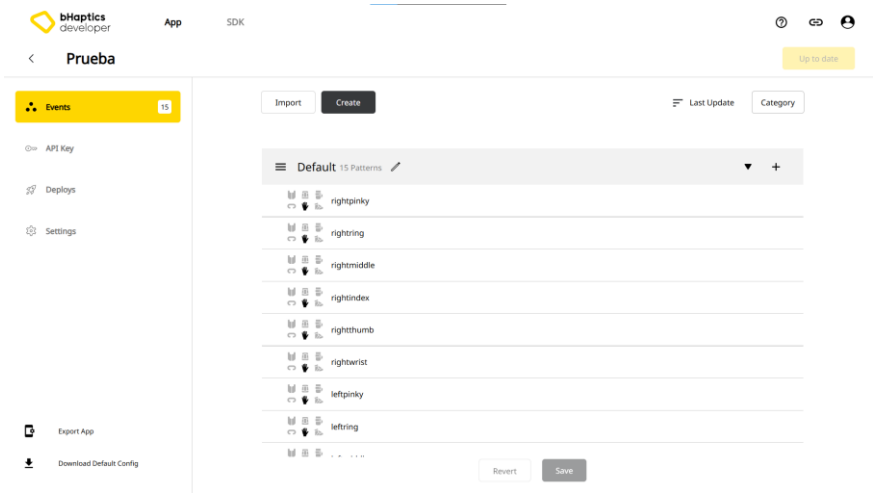


Figura 33: Pestaña Events de bHaptics Developer.

Una vez creada, se configurará Unity para que pueda acceder a la aplicación creada. Para ello hay que ir a la opción *Developer Window* en la pestaña de *bHaptics* en la parte superior del editor. Ahí será necesario indicar la ID y la Key de la aplicación. Hecho esto será posible acceder a los eventos previamente creados. [Figura 34] Además, desde esa misma ventana es posible probar cada patrón si los dispositivos están conectados, comprobando su funcionamiento.

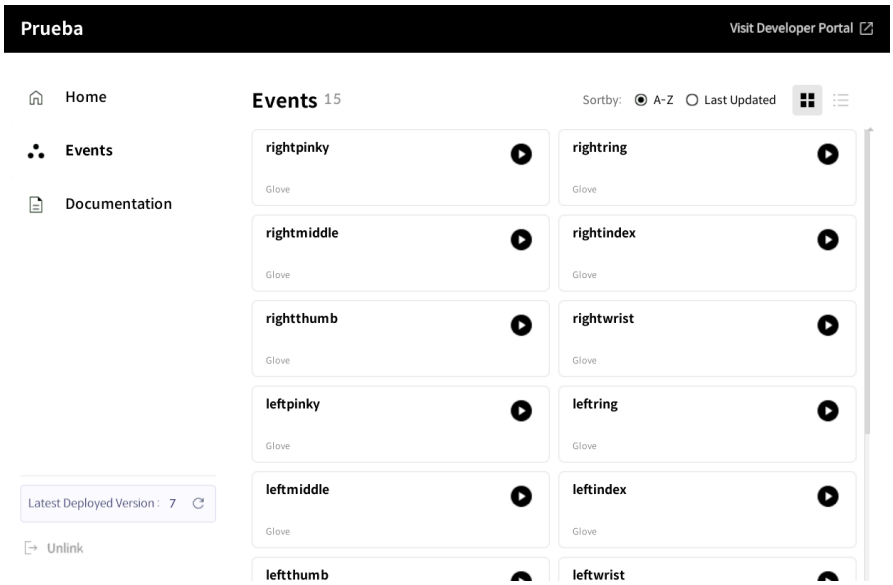


Figura 34: Developer Window de Unity.

Volviendo al Developer, para añadir los efectos creados, hay que pulsar en *Create* y seleccionar la categoría y el nombre en la pestaña que se abre. Se pueden elegir varios patrones creados, de esta forma es posible combinar efectos para llamarlos al mismo tiempo. Una vez asignadas las sensaciones se pulsa el botón *Save.* [Figura 35]

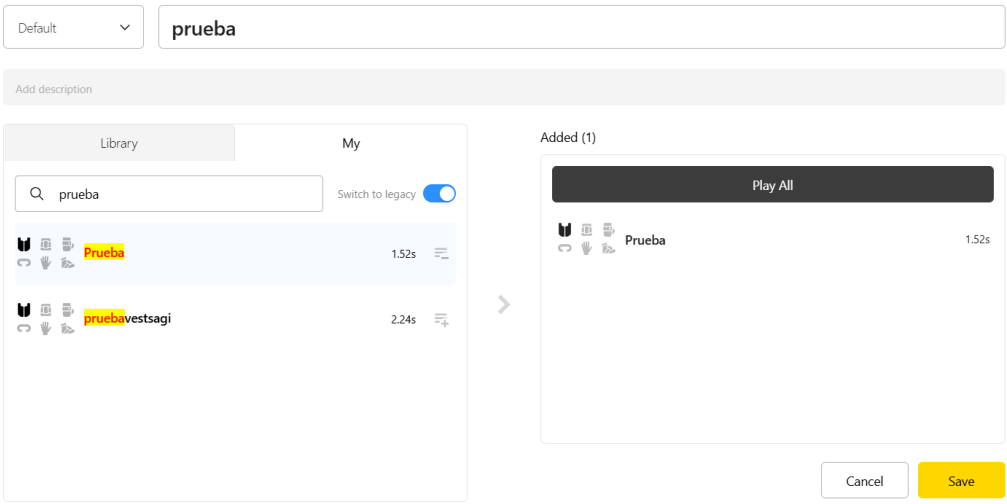


Figura 35: Añadiendo un evento en el Developer.

Con esto el efecto ya se habrá guardado en la aplicación. Lo siguiente es pulsar el botón *Deploy* que encontramos en la esquina superior derecha para desplegar la nueva versión. Cada vez que se cree una nueva versión será necesario actualizar la *Developer Window* para encontrar los efectos nuevos. Habiendo sido añadidos será posible llamar desde el código a las sensaciones haciendo uso del nombre que se le haya asignado.

3.4 5G Robot Race

Cómo bien se ha especificado antes, este proyecto se realizó en conjunto con el Laboratorio inmersivo (IMM-Lab) del iTEAM. El entorno usado finalmente para mi aplicación fue el velódromo de la UPV, realizado por fotogrametría por la empresa de Nokia. Este entorno fue creado para una aplicación en desarrollo del laboratorio llamado 5G Robot Race.

Este proyecto consiste en una carrera entre dos robots AGVs (Vehículos de Guiado Automático), que tiene como circuito el velódromo de la UPV. El control de estos robots se realiza mediante unos *cockpits* situados en el Palacio de Congresos de Valencia, comunicándose en tiempo real con los robots del velódromo.

El entorno virtual es un gemelo digital del velódromo. Los usuarios podrán visualizar lo que está viendo el robot en todo momento. Este gemelo digital cuenta con objetos de realidad aumentada (AR) que afectarán al robot, aumentando su velocidad en caso de pasar por ellos.

Figura 36: Poster 5G Robot Race del iTEAM.

Los vehículos disponen de sensores y cámaras que permiten su control remoto, además envían datos telemétricos en tiempo real para crear un gemelo virtual que especifica la posición del robot de forma que el usuario tenga visualización en tiempo real de los cambios que sufra.

Este proyecto cuenta con retroalimentación háptica a través del chaleco. Este activará ciertos patrones cuando ocurran varios eventos, como al completar una vuelta o al recoger un objeto potenciador, dando una sensación de aumento.

Capítulo 4. Desarrollo

El desarrollo de este proyecto se puede dividir en varios apartados como pueden ser el entorno, los obstáculos o el movimiento del usuario. Estas secciones se han ido desarrollando poco a poco, sin una clara separación de tiempo entre ellas debido a que los problemas que han surgido a lo largo del proyecto han requerido realizar cambios en partes supuestamente terminadas.

4.1 Integración dispositivos hápticos

El primer paso fue configurar e incorporar los dispositivos hápticos en la aplicación, ya que estos podían suponer la mayor complicación por ser la primera vez que se trabajaba con ellos. Al ser una tecnología relativamente nueva, no existe mucha información por parte de otros usuarios en la red.

Mirando la documentación oficial, podemos encontrar dos formas principales de conseguir la vibración de los dispositivos en Unity. [35] La primera de ellas es usando uno de los patrones previamente creados en BhapticsDesigner e importados usando BhapticsDeveloper. [Figura 36]

```
BhapticsLibrary.Play(BhapticsEvent.SHOOTPISTOL);
or
BhapticsLibrary.Play("shootpistol");


// If you want to customize the event, call PlayParam()
// For example, when you attcked

BhapticsLibrary.PlayParam(BhapticsEvent.SHOOTPISTOL,
    intensity: 1f,   // The value multiplied by the original value
    duration: 1f,    // The value multiplied by the original value
    angleX: 20f,     // The value that rotates around global Vector3.up(0~360f)
    offsetY: 0.3f);  // The value to move up and down(-0.5~0.5)
```

Figura 36: Añadir evento de bhaptics usando el nombre. [35]

En este caso es posible elegir entre ejecutar el evento tal y como ha sido creado, o cambiar algunos parámetros como la intensidad o la duración. De esta forma se pueden realizar pequeños cambios directamente desde Unity, sin tener que pasar por todo el proceso otra vez.

La segunda forma es llamando directamente a los motores del dispositivo desde el código. [Figura 37]. Para ello es necesario especificar en el parámetro *position* el dispositivo de bhaptics que estamos usando, ya sea el chaleco, los guantes o cualquier otro. Debemos crear un array con el

número de motores, especificando la intensidad de cada uno por separado usando un número entero entre 0 y 100 y por último, elegir la duración de la vibración en milisegundos.

```
BhapticsLibrary.PlayMotors(
    position      : (int)Bhaptics.SDK2.PositionType.Head,
    motors        : new int[6] { 60, 80, ... },
    durationMillis: 1000
);
```

Figura 37: Añadir eventos de forma directa. [35]

Teniendo esto en cuenta, se creó un script básico que pusiese lo aprendido a prueba usando el chaleco. Este código se encargaba de realizar un patrón creado en el Designer al pulsar una tecla, y de ejecutar uno distinto usando la función PlayMotors al pulsar otra distinta. Al funcionar de la manera esperada se hizo lo mismo con los guantes. De esta forma fue posible comprobar que los dispositivos hápticos funcionaban correctamente y era posible implementarlos en el proyecto.

4.2 Circuito

Con la seguridad de que los dispositivos hápticos funcionaban, se empezó con el desarrollo del entorno, para ello era necesario conseguir los *assets* con los que crear el paisaje por el que se movería el usuario. En un principio la estética del proyecto iba a ser *low poly* [Figura 38]. Traducido al castellano como “bajo poligonaje”, es una forma de trabajar usando elementos geométricos con pocas caras y vértices. De esta forma se simplifican los detalles, consiguiendo un mejor rendimiento.

Figura 38: Creación de entorno *low poly*.

Para conseguir esta estética se descargaron ciertos paquetes de la tienda de assets de Unity, los paquetes descargados son:

- Low-Poly Simple Nature Pack [36]
- Low Poly Nature - FREE Vegetation [37]

Estos paquetes incluyen elementos tales como árboles, rocas y plantas que permiten decorar haciendo una experiencia más agradable y simplificando el desarrollo. Con esta base para el entorno se empezó a diseñar el recorrido que seguiría el usuario. Se diseñó una carretera sencilla con una rampa y una curva, para poder hacer pruebas de movimiento [Figura 39].

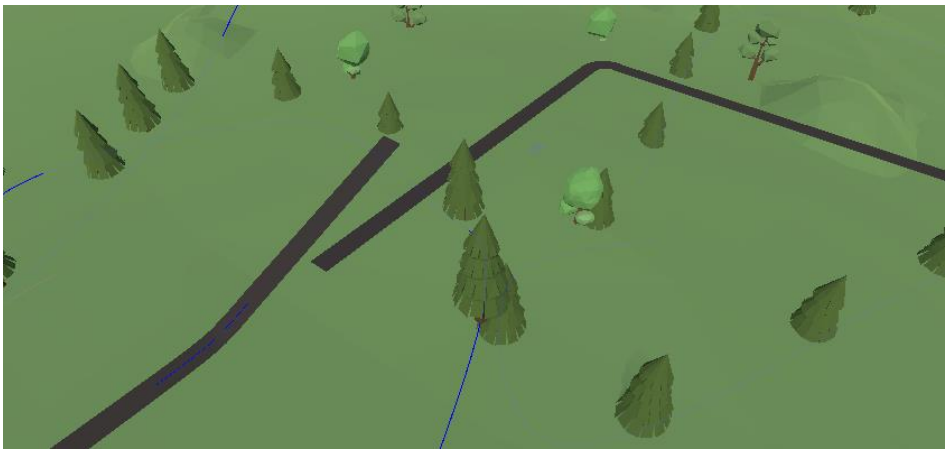


Figura 39: Prototipo de carretera.

Al poco se llegó a la conclusión de que era necesario cambiar la forma de diseñar el recorrido ya que la forma actual no era eficiente, especialmente si se quería añadir verticalidad. Para cambiar esto se empezó a hacer uso del objeto *Terrain*. Este elemento de Unity permite cambiar de forma sencilla el terreno, pudiendo realizar modificaciones en el relieve de forma sencilla con las herramientas por defecto [Figura 40].

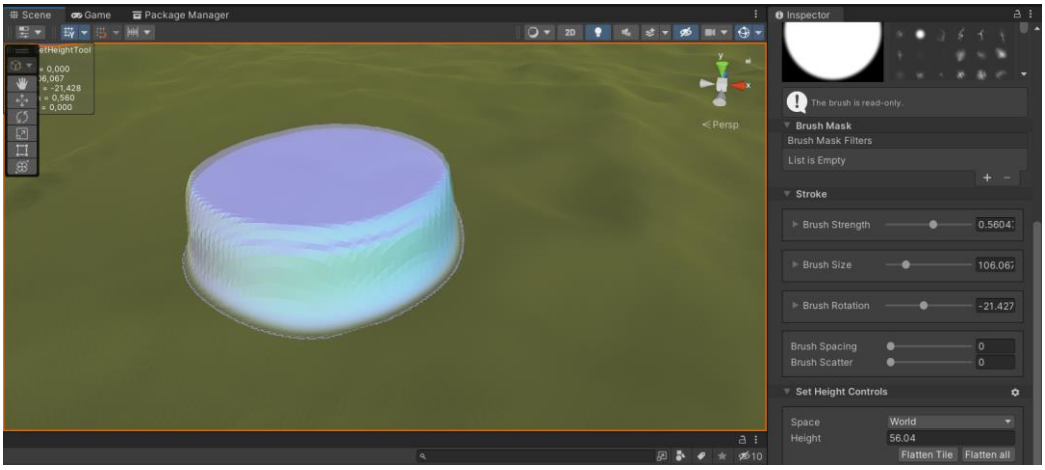


Figura 40: Edición del elemento *Terrain*.

Además, esta herramienta permite añadir árboles y otros elementos fácilmente en forma de detalles [Figura 41]. Estos elementos se pueden configurar para que tengan mejor o peor calidad dependiendo de la distancia, al igual que cambiar si son afectados por el viento o no, lo que permite personalizar el entorno más a fondo de una forma mucho más sencilla.

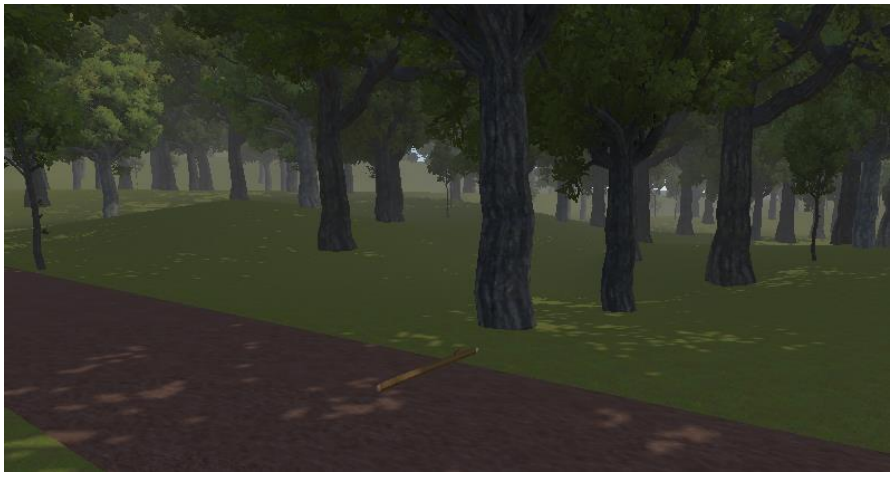


Figura 41: Visualización del terreno con árboles.

A pesar de ser una forma más rápida de diseñar el entorno, seguía habiendo mucho trabajo por delante. Por ello, hablando con el tutor del proyecto, se llegó a la conclusión de que sería más adecuado usar el modelo del velódromo del que disponía el iTEAM [Figura 42].

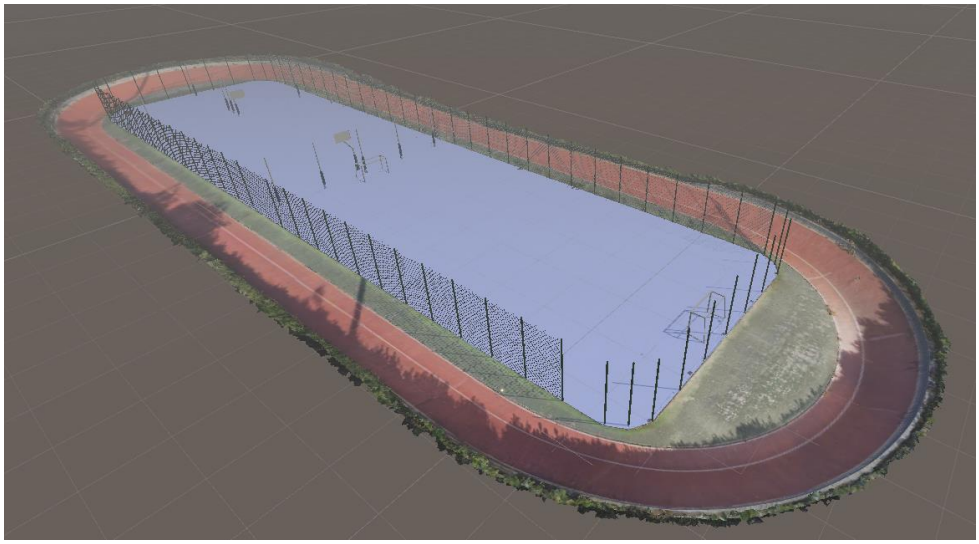


Figura 42: Modelo del velódromo del iTEAM

Usar este circuito significaba una preocupación menos ya que el mapa estaba desarrollado. Esto implicaba sacrificar parte creativa del proyecto, cambiando el enfoque del proyecto a algo más cercano a una demo técnica centrada en los dispositivos hápticos. El nuevo planteamiento hacía que este trabajo estuviera más relacionado con una de las aplicaciones en desarrollo del equipo del laboratorio inmersivo, por lo que sería más fácil unir partes en caso de que fuera necesario.

Fue necesario hacer algunos cambios meramente técnicos en el mapa, añadiendo ciertos *colliders* que hiciesen que tanto el jugador como los distintos objetos con los que interactúa no pudieran salir del mapa. El cambio de perspectiva también supuso volver a pensar los obstáculos que aparecerían en el recorrido.

Este mapa también incluye un modelado de varios barrios de Nueva York que se usarán de fondo para dar la sensación de estar cerca de una ciudad. Al igual que un terreno que incluye unas colinas que se pueden ver de fondo.

4.3 Integración VR

Para integrar las gafas VR se hizo uso de los ejemplos incluidos en el paquete "Meta XR Interaction SDK". [38] Con estos ejemplos se pudieron configurar tanto Unity como las gafas para que funcionasen correctamente. Uno de los problemas encontrados fue que la versión de Unity que se estaba usando hasta ese momento no era compatible con el paquete, por lo que hubo que migrar el proyecto a una versión más nueva.

Fue necesario realizar varios cambios en las configuraciones del proyecto. Se cambió la plataforma de la aplicación a Android y se seleccionaron varias opciones del paquete de integración de Meta, [Figura 43] consiguiendo que la imagen transmitida a las gafas cambiase al girar la cabeza y no quedase congelada.

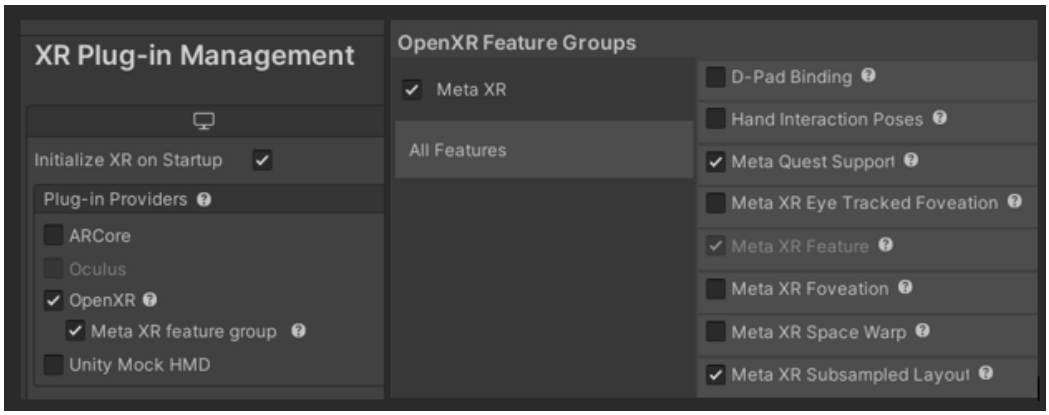


Figura 43: Configuración del paquete de integración VR.

Al hacer estos cambios se consiguió el correcto funcionamiento del ejemplo. El siguiente paso era implementarlo en la aplicación. Para ello hay que usar el *prefab* llamado *OVRCameraRig*. [Figura 44] Este objeto sustituye a la cámara predeterminada y crea todos los elementos necesarios para hacer un tracking tanto de la cabeza como de las manos de forma automática.

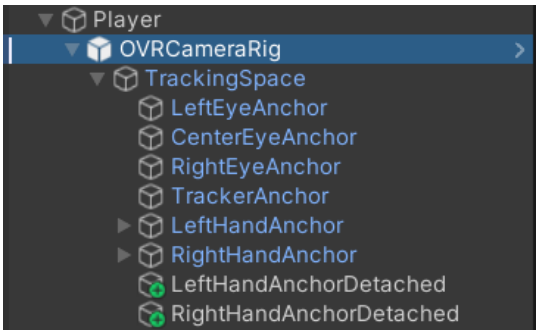


Figura 44: Objeto *OVRCameraRig* en la pestaña Hierarchy.

Al hacer que este objeto sea hijo de *Player*, la cámara seguirá directamente al usuario, por lo que no es necesario añadir ningún script de seguimiento. Con esto se consiguió el correcto funcionamiento de las gafas VR, pudiendo visualizar el entorno virtual siguiendo el movimiento del jugador en todo momento.

4.4 Movimiento

La manera en la que se movería el usuario por el escenario ha ido cambiando a lo largo del proyecto. Desde un principio se decidió que se recorrería un circuito predeterminado, pero el diseño de este ha ido cambiando a lo largo del desarrollo. La posibilidad de hacerlo usando una animación quedó descartada enseguida ya que era importante que la interacción con los objetos se sintiese real, y que estos pudiesen cambiar la velocidad del usuario.

4.4.1 Primera forma

La primera manera con la que se intentó conseguir el movimiento del usuario fue usando un sistema de fuerzas aplicadas al objeto *Player*. Estas fuerzas se aplicaban dependiendo de la superficie sobre la que se encontrase el jugador en cada momento, comprobando que este objeto formase parte de la carretera y aplicando las fuerzas necesarias en consecuencia.

Para poder discernir entre los distintos objetos con los que colisiona el jugador se hace uso del sistema de *tags,* o etiquetas, de Unity. Las etiquetas son una herramienta que permite separar los objetos en distintos tipos. Esto nos permite crear distintas interacciones en función del tipo de objeto que sea de forma sencilla a través de nuestros *scripts*.

Un problema que se encontró era que, como la carretera estaba formada por varias partes, al cambiar entre objetos el jugador dejaba de moverse por un momento, dando así pequeños tirones molestos que sacaban de la experiencia. La solución fue realizar un contador que sumaba uno cuando se entraba a una carretera nueva y restaba uno cuando se salía de ella. Para saber si se debía aplicar la fuerza se comprobaba que este número fuese mayor de cero.

Otra cosa a tener en cuenta era la orientación del objeto, tanto para dar curvas, como para subir y bajar rampas, era importante que el jugador mirase en la dirección correcta. Se creó un código que calculaba la orientación de la carretera y se la asignaba al usuario. [Figura 46] Este cambio de dirección se aplica progresivamente en función de un parámetro que controla la velocidad a la que cambia la cámara.

```
RaycastHit hit;
if (Physics.Raycast(transform.position, -Vector3.up, out hit))
{
    if (hit.collider.CompareTag("Road"))
    {
        Quaternion surfaceRotation = hit.transform.rotation;

        Quaternion targetRotation = surfaceRotation;

        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.deltaTime * cameraSpeed);

        Vector3 newVelocity = transform.TransformDirection(localVelocity);

        rb.velocity = newVelocity;
    }
}
```

Figura 46: Código encargado de la orientación del jugador.

Para conseguir que el usuario diese las curvas correctamente, se hacía uso de *triggers* que cambiaban los parámetros como la velocidad del usuario y la velocidad de la cámara, evitando que el jugador se saliera de la curva. [Figura 47] Un *collider* con el parámetro *isTrigger* activado implica que el objeto que choque con él no sufrirá una colisión directa, pero podrá saber cuándo está dentro de este *trigger* para poder actuar en consecuencia a través del código.

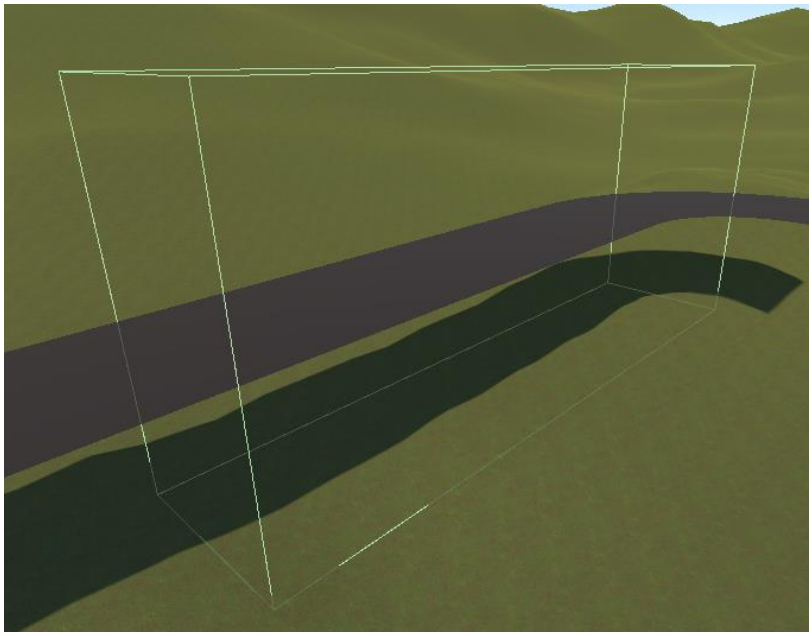


Figura 47: *Trigger* que controla la velocidad del objeto.

Para poder hacer que el objeto pudiese saltar rampas, hubo que tener en cuenta a la hora de aplicar las fuerzas en el código, la fuerza de la gravedad. Esto supuso añadir una fuerza continua hacia abajo que estaría aplicándose en todo momento.

Esta forma de hacer el movimiento permitía al objeto viajar por el entorno, pero era una forma muy tediosa, ya que cada curva tenía que configurarse a mano para que el objeto no se saliera, además que en muchas ocasiones el objeto no quedaba centrado en la carretera. Al cambiar el entorno del proyecto por el velódromo, este problema disminuía ya que solo eran dos curvas. Se probó a crear una carretera escondida, por la que pasaría el objeto.

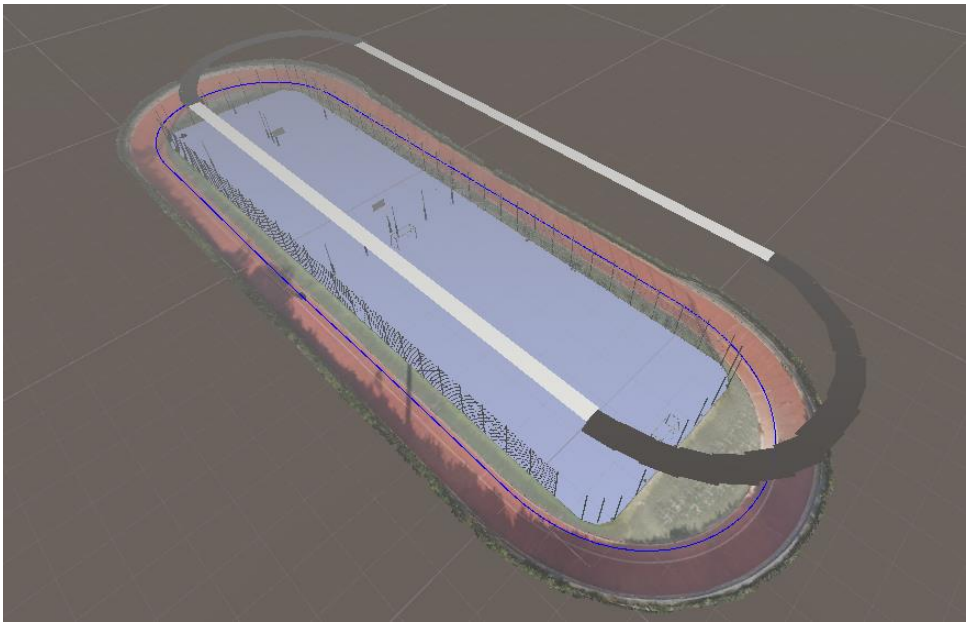


Figura 48: Carretera creada para el velódromo.

Aun así, el jugador no tomaba las curvas de forma suave, sino que se notaba que iba a golpes. Por lo que se decidió cambiar la manera de programar el movimiento por algo que hiciese un movimiento más fluido y sencillo.

4.4.2 Segunda forma

Viendo que era necesario un cambio completo en el movimiento, era hora de parar de hacer cosas e investigar formas de mover un objeto siguiendo una línea. Explorando formas de hacerlo, se encontró el paquete *Spline,* junto a un tutorial que explicaba su funcionamiento. [39] Este es un paquete gratuito oficial de Unity. La principal función del paquete es crear caminos que pueden usarse tanto para generar objetos encima de ellos, como para moverlos a lo largo de la línea.

Este paquete viene con algunos scripts por defecto. Uno de ellos permite animar un objeto que se mueva por el camino creado. En esta animación se puede personalizar el tiempo que tarda en completarse, o la velocidad a la que lo hace, pero no permite que el objeto cambie la velocidad a mitad de camino o que colisione con otros objetos. Buscando la forma de hacerlo se encontró otro vídeo que explicaba la forma de aplicar fuerzas mientras se seguía el camino definido por la *spline*.[40]

En este vídeo lo que se hace es coger el script original y buscar las partes que interesan para poder hacer que los objetos se queden en la *spline*. Con la ayuda de este vídeo se ha realizado un *script* que combina la forma de aplicar fuerzas y velocidades del sistema de movimiento anterior, con la posibilidad de hacerlo sobre un camino preciso.

La principal diferencia con el sistema anterior es que en este se busca todo el rato el punto de la *spline* más cercano al jugador, y se le asigna esta posición. Tras esto se le aplican las fuerzas que moverán al jugador hasta el próximo fotograma, momento en el que se volverá a buscar el punto más cercano. Con esto se consigue que el objeto avance sin salirse del camino definido.

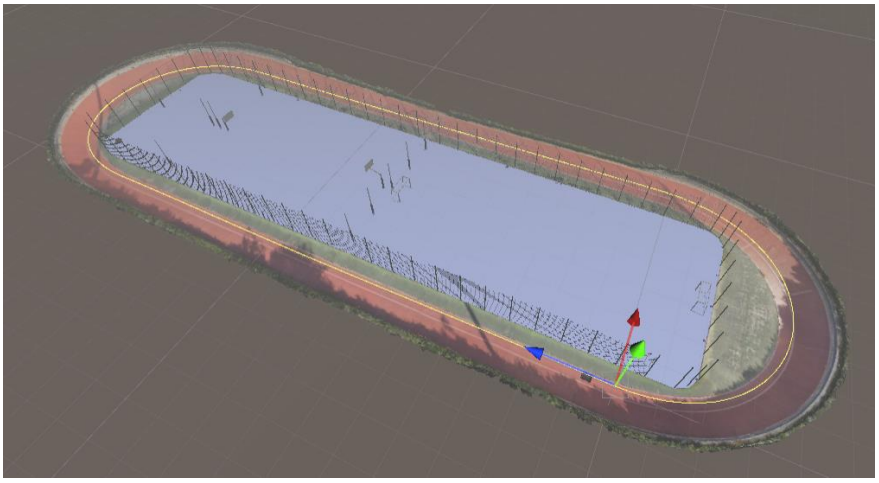


Figura 49: Objeto *spline* que recorre el velódromo.

Al igual que antes teníamos en cuenta la orientación de la carretera, en este caso podemos usar la dirección en la que mira el punto en el que nos encontremos para girar al jugador en la dirección correcta. Con este nuevo sistema de movimientos es posible mover al jugador a lo largo del recorrido de forma precisa, consiguiendo interactuar fácilmente con los objetos del entorno.

4.5 Colisiones

La colisión del cuerpo del usuario y la retroalimentación háptica del chaleco se gestionan desde un único archivo de código llamado *VestCollision*. Este archivo es responsable de controlar las interacciones entre el usuario y los diferentes objetos del escenario. En este proyecto, se implementan tanto colisiones normales como colisiones que utilizan la función *trigger*.

Como se mencionó anteriormente, los objetos que funcionan como *triggers* no colisionan directamente con otros objetos, pero es posible detectar cuándo se está invadiendo un *collider* con este parámetro activado. Para ello, se utilizan las funciones *OnTriggerEnter* y *OnTriggerExit*. Por otro lado, la detección de colisiones con objetos normales se realiza mediante la función *OnCollisionEnter*.

Para saber el tipo de objeto con el que colisiona se hace uso de los *tags* previamente asignados a cada uno de los objetos. De esta forma el código ejecutará distintas instrucciones dependiendo del obstáculo.

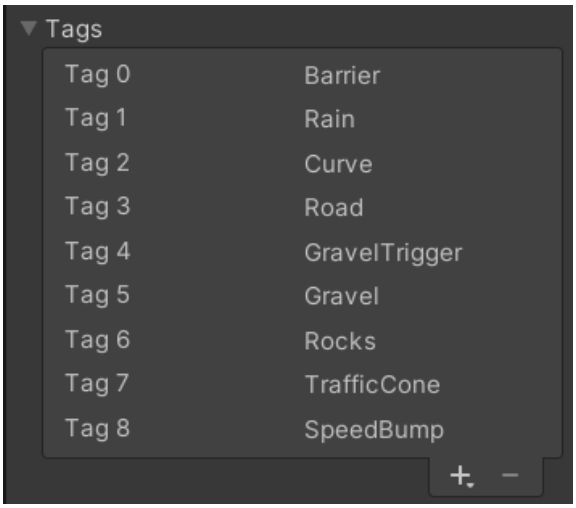


Figura 50: Lista de etiquetas del proyecto.

Algo a tener en cuenta es la matriz de físicas de Unity. [Figura 51] Con esta matriz se controlan las colisiones que pueden tener los elementos del entorno con otros. Usando un sistema de capas, nos permite seleccionar si un tipo de elemento puede chocar con otro o no. Estas capas se le asignan a cada objeto en el inspector de objetos, pero no son las mismas que los tags.

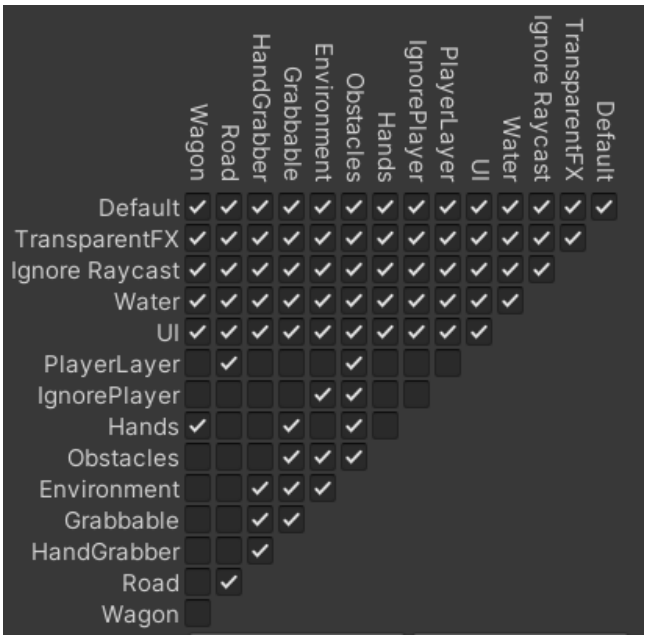


Figura 51: Matriz de físicas del proyecto.

Algunas de estas capas vienen por defecto, mientras que las otras son completamente personalizables. Con esto se puede observar los objetos que pueden interactuar entre ellos. Un

ejemplo es la capa *Wagon* que corresponde al vehículo. Es interesante que solo las manos puedan interactuar con este objeto, más específicamente con la palanca, de esta forma también evitamos que se produzcan colisiones no deseadas con otras partes del entorno.

4.6 Obstáculos

Se pueden encontrar varios obstáculos a lo largo del recorrido, cada uno de ellos interactuará de forma distinta con el jugador. Estos casos están contemplados dentro del script *VestCollision*, donde cada objeto devuelve respuestas distintas.

4.6.1 Barreras

Uno de los obstáculos más sencillos que podemos encontrar son las barreras. Estas barreras constan de un modelo sencillo que reacciona al chocar tanto con el usuario como con las manos de este. Cuando se detecta un choque con una, estas se giran mediante una simple animación a la que se llama desde el código para dejar paso al jugador. [Figura 52] Esta animación consiste en un giro de 90 grados respecto al eje de la barrera.

Figura 52: Visualización de las barreras en el circuito.

Cuando el usuario choca con la barrera, el chaleco reproduce un patrón que consiste en una vibración fuerte en el lado del impacto que se va atenuando conforme se aleja de este lugar. También se desactiva el *collider* de la barrera, de forma que no genere ningún problema.

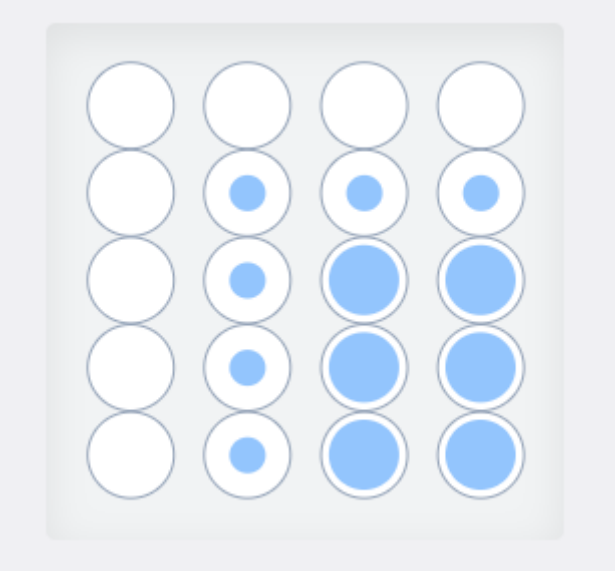


Figura 53: Patrón de vibración de las barreras.

4.6.2 Torre de conos

El segundo obstáculo que encontramos son las torres de conos de tráfico. [Figura 54] Estas torres están hechas para que el usuario las tire con la mano, observando la forma en la que los conos chocan entre ellos, el entorno y cómo reaccionan al impacto de la mano. En caso tirarlos con el cuerpo, los conos no producen ningún tipo de retroalimentación en el chaleco ya que no suponen un obstáculo tan grande.

Figura 54: Torres de conos.

El modelo de estos conos ha sido sacado de un paquete de la asset store de Unity que cuenta con varios objetos que se podrían encontrar en la carretera, en zonas de obras. [41]

4.6.3 Grava

Este obstáculo consiste en un suelo rugoso al que se le ha intentado dar una textura de grava. [Figura 55] Cuando el jugador pasa por encima, genera un movimiento aleatorio que combinado a la vibración del chaleco intenta simular el circular por un terreno accidentado.

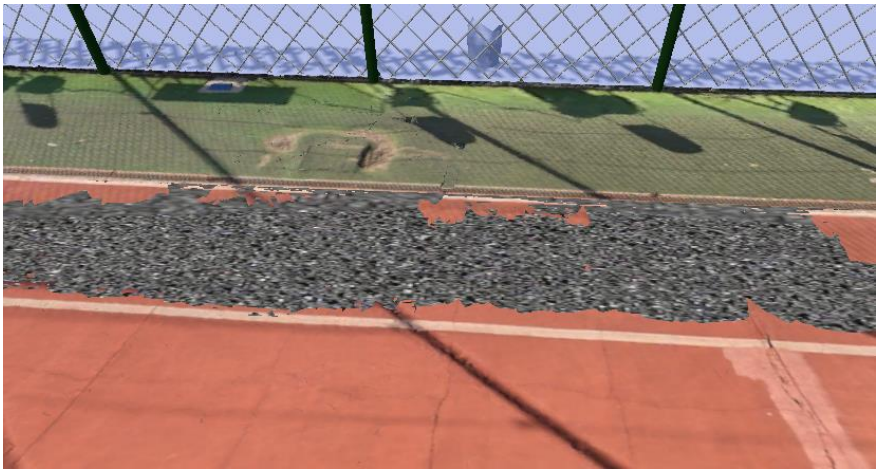


Figura 55: Obstáculo grava.

Para ello se ha creado un código que genera un movimiento aleatorio en el vehículo y la cámara el mismo tiempo. Este código genera una diferencia de posición de forma aleatoria delimitado por un valor previamente especificado y que puede variar en la velocidad a la que lo hace. Esta diferencia se suma a la posición original, cambiando la posición por un breve periodo de tiempo hasta que vuelve a la posición original.

En este caso, para la vibración del chaleco se ha creado una experiencia en el designer que corresponde a un camino que se mueve de forma aleatoria de arriba a abajo cambiando la intensidad en cada instante. De esta forma se da una sensación de aleatoriedad.

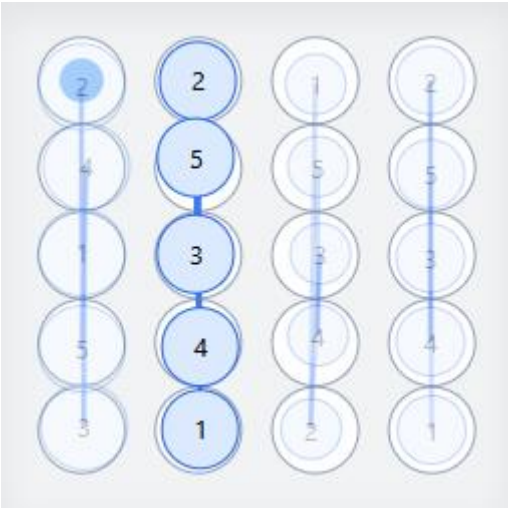


Figura 56: Patrón de vibración de la grava.

4.6.4 Badén de velocidad

Estos objetos intentan simular los badenes que sirven para indicar que hay que reducir la velocidad. El modelo se ha sacado del mismo paquete que los conos de tráfico. Este objeto funciona de forma que al pasar por encima la cámara hace un movimiento hacia arriba en un corto periodo de tiempo, y luego hacia abajo a la misma velocidad.

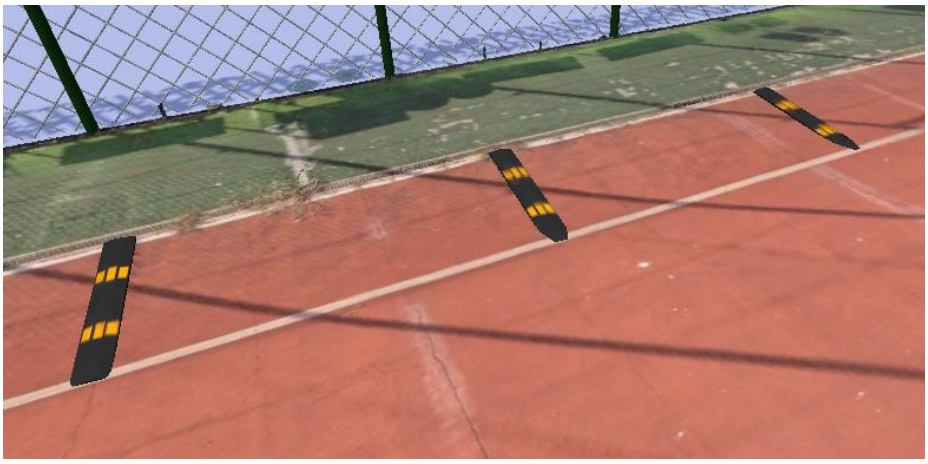


Figura 57: Obstáculo badenes.

La sensación háptica creada intenta imitar un golpe en la parte baja que disminuye de intensidad y vuelve a aumentar al llegar a la posición inicial. Para ello ha sido importante tener en cuenta el tiempo que tardaba la cámara y adecuar el patrón a el mismo tiempo.

4.6.5 Lluvia

El último obstáculo es el de lluvia. Ha sido creado con un sistema de partículas que simula de forma adecuada las gotas de agua cayendo. También se ha usado un modelo de nubes que sirven como origen de la lluvia, intentando crear una mayor continuidad. Al igual que los conos, este modelo se ha descargado de la Asset Store.

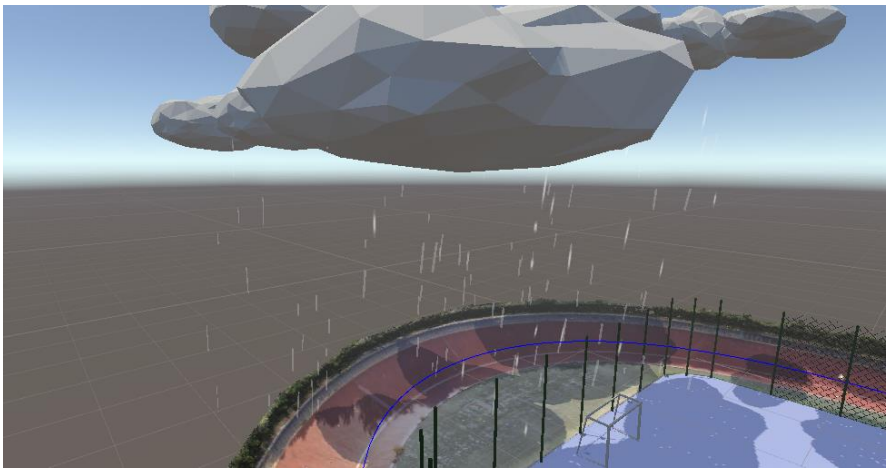


Figura 58: Obstáculo lluvia.

Para esta sensación se ha usado la función *PlayMotors* que permite crear la sensación directamente desde el código. De esta forma se ha intentado recrear las gotas de lluvia golpeando tanto en la parte superior del chaleco como en los guantes, usando una intensidad muy baja calculada de forma aleatoria.

```
BhapticsLibrary.PlayMotors(
    position      : (int)Bhaptics.SDK2.PositionType.Vest,
    motors        : new int[40] { Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5),
                                  Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5),
                                  0, 0, 0, 0,
                                  0, 0, 0, 0,
                                  0, 0, 0, 0,

                                  Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5), Random.Range(0, 5),
                                  0, 0, 0, 0,
                                  0, 0, 0, 0,
                                  0, 0, 0, 0,
                                 },
    durationMillis
);
```

Figura 59: Código que genera la sensación de lluvia.

4.7 Interacción de las manos

La integración de las manos se hace con el mismo paquete que usamos para el VR. En el *OVRCameraRig* usado anteriormente para la cámara, se encuentra seleccionada por defecto la opción de usar los controladores. En este objeto se encuentra los anclajes para la mano izquierda y derecha. En estos anclajes podemos añadir el *prefab* de las manos, lo que nos permitiría verlas en todo momento.

El siguiente paso era conseguir que las manos chocasen con los obstáculos del entorno. Para que un objeto pueda ser afectado por el sistema de físicas, es necesario que disponga tanto de un *collider* como de un *rigidbody.* Con esto era posible que la mano chocase con otros objetos, sin embargo, para poder integrar el guante correctamente, es interesante separar la mano en partes, de forma que podamos discernir la parte afectada.

Para ello fue necesario crear una mano personalizada, ya que la que venía por defecto no permitía interactuar de la forma querida. Los objetos predefinidos *OculusHand_L* y *OculusHand_R* creaban un esqueleto de forma automática, pero fue necesario añadir 3 conjuntos de *collider* y *rigidbody* en cada dedo para que fuese afectado por las físicas. [Figura 60] También hubo que mapear los huesos manualmente para la correcta representación de la mano.

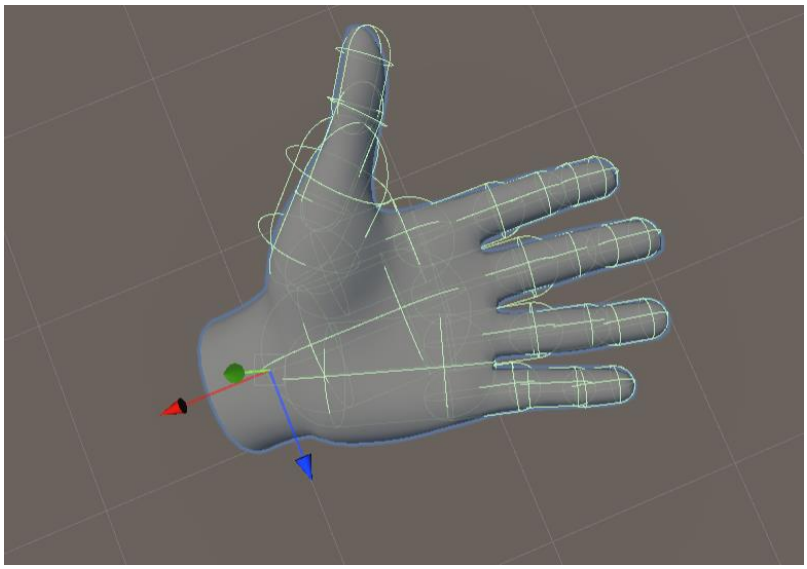


Figura 60: Mano izquierda con sus *colliders*.

Con esto se había conseguido el funcionamiento del *tracking* de la mano, así como la interacción con los objetos del entorno. El siguiente paso era hacer que los objetos respondieran a las manos y configurar las sensaciones del guante háptico.

4.7.1 Colisión de manos

Al contrario que para el chaleco y el jugador, en este caso el código se asignaría a cada uno de los obstáculos con los que interactuase la mano. De esta forma sería posible reconocer que dedo estaba tocando y cuales no, configurando la respuesta háptica para cada dedo.

Este script se asigna tanto a los conos de tráfico como a las barreras. En caso de que sea un cono de tráfico, la colisión aplicará una fuerza en la dirección contraria a la que se detectó la mano. Esta fuerza dependerá de la velocidad de la mano, y se calcula usando un *script* que tienen las manos que calcula la velocidad de la mano comprobando la distancia recorrida en la última décima de segundo.

En caso de ser una barrera, ocurre lo mismo que al chocar con el cuerpo y se activa la animación que la abre. En ambos casos, se comprueba la parte de la mano que ha chocado y se llama al evento que corresponde con esa parte, habiéndose creado previamente estos eventos en el *designer*. [Figura 61]

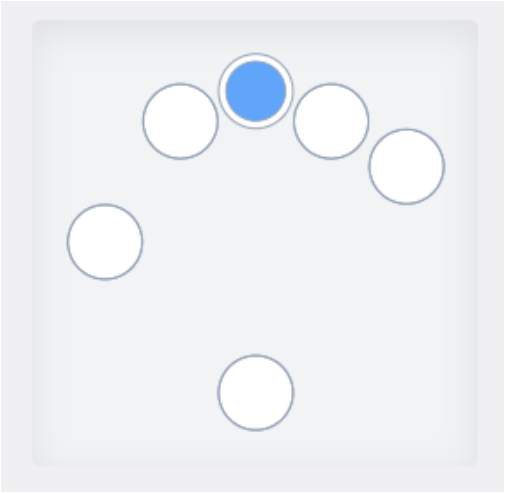


Figura 61: Patrón de vibración dedo corazón mano derecha.

4.7.2 Agarre de objetos

Para lograr que la experiencia fuese más interactiva, se propuso crear una interacción con la que el usuario fuese capaz de agarrar algún objeto del entorno. En un principio se pensó en que el usuario pudiese recoger algún objeto del suelo o del alrededor y pudiese lanzarlo.

Esto se llevó a cabo partiendo de los *scripts* disponibles en el paquete de Meta VR. *OVRGrabber* y *OVRGrabbable* permiten crear la interacción entre dos objetos, consiguiendo acoplar el objeto agarrado a la mano. [Figura 62] Fue necesario crear un script que fuese una extensión de *OVRGrabber* ya que había que configurar algunas cosas.

Para poder agarrar los objetos usando gestos de la mano, se usó la función *GetFingerPinchStrength* que devuelve la fuerza de agarre dependiendo de la distancia entre el dedo pulgar y el dedo escogido. El código tiene en cuenta la posibilidad de agarrarlo usando cada uno de los dedos, adaptando así la respuesta háptica del guante en consecuencia.

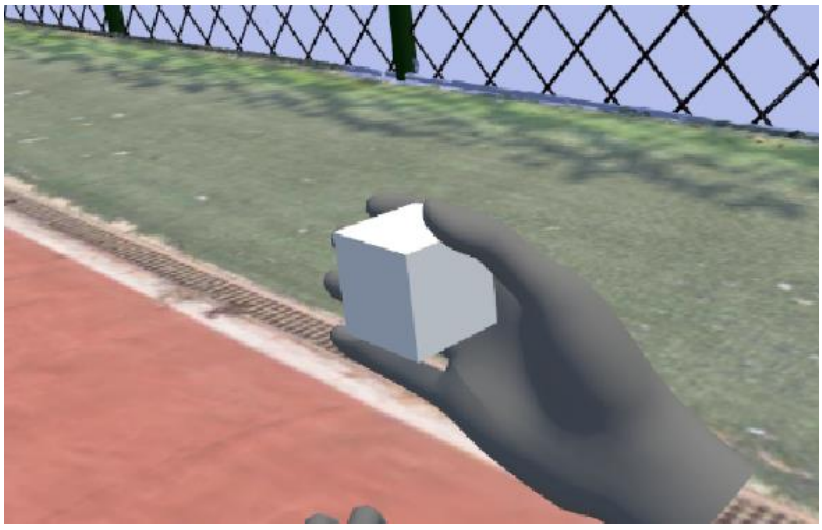


Figura 62: Mano agarrando un cubo.

Esta forma de interactuar con los objetos era incómoda ya que no es sencillo adaptar la velocidad de forma que puedas agarrar objetos en movimiento. Por lo tanto, se propuso otra interacción distinta, que consistía en una palanca que permita al usuario aumentar o disminuir la velocidad del jugador.

Fue necesario crear un vehículo al que poder acoplar la palanca, ya que hasta este momento el usuario no tenía vehículo visible. [Figura 63] Este objeto consiste en una vagoneta muy sencilla, ya que solo se va a ver desde dentro, por lo que el aspecto exterior no es importante. El vehículo contiene una palanca que puede moverse adelante a atrás. Dependiendo de la posición de esta, la velocidad aumentará o disminuirá.

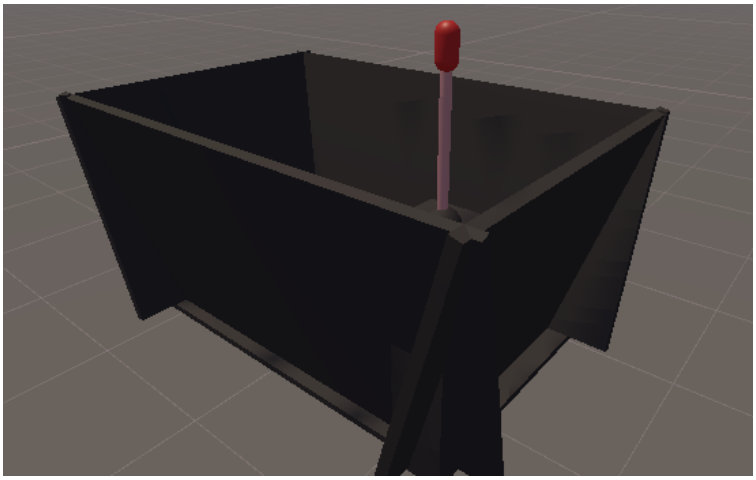


Figura 63: Vehículo del usuario.

La lógica tras esta palanca se creó usando el paquete Tilia.[42] Este paquete permite crear varias interacciones de forma sencilla, sin apenas tener que programar código. Los objetos de Tilia se dividen en objetos con los que se puede interactuar y objetos que interactúan. Para poder hacer que la palanca sea un objeto con el que interactuar, es decir, que se mueva con el movimiento de la mano, hay que ir a la pestaña *Window → Tilia → Interactions → Controllable Creator.* Al pulsar se abrirá una ventana en la que se elegirá *Angular Transform Drive.*

De esta forma la palanca se convierte en un objeto con el que interactuar que puede girar en el eje que se seleccione, pudiendo cambiar varios parámetros como el ángulo de giro, número de saltos que tiene, o si queremos que se mueva a la posición más cercana. Lo siguiente es crear el objeto que va a interactuar con la palanca. Para ello se usará un *prefab* de tilia llamado *Interactions.Interactor* que se asignará a ambas manos.

Este objeto crea lo necesario para poder interactuar con otros objetos de Tilia, entre lo que se incluye un cubo al que le asigna el *tag* “GrabZone” que marca la zona en la actúa el objeto. Es necesario ajustar esta zona para que funcione en la palma de la mano. También hay que indicar la

acción que activará la interacción. Estas acciones vienen dadas por un *prefab* de Unity y están mapeadas para activarse con los controladores de las gafas VR. [Figura 64]

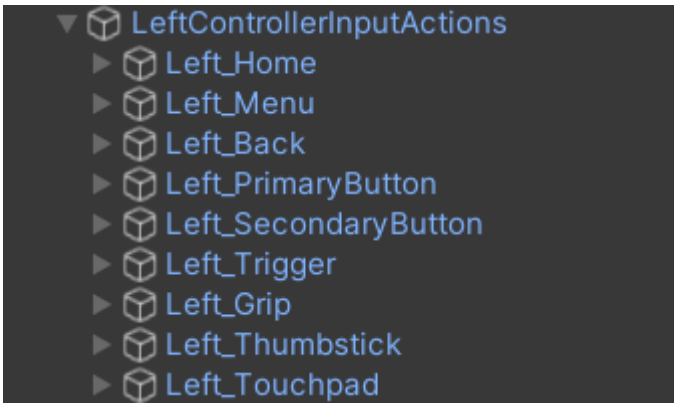


Figura 64: Acciones predeterminadas del controlador

En este caso la interacción tiene que activarse al cerrar los dedos. Para ello hay que editar el código de antes. Habrá que elegir una acción que se asignará en el inspector de Unity. En este caso el *LeftGrip*. Luego esta acción se invocará desde el código, consiguiendo así poder mover la palanca. Por último, se modificará el código del movimiento para añadir una parte que, dependiendo de la inclinación de la palanca, haga que la velocidad del objeto aumente o disminuya.

4.8 Escena inicial

Se ha creado una escena inicial muy sencilla con dos botones, uno que permite iniciar el juego y otro para salir de este. [Figura 65] Esta escena se puede controlar tanto con el toque de los dedos como con un puntero que sale de la mano. La implementación de estas funciones es automática gracias al paquete de interacción VR de meta.

Figura 65: Escena inicial.

Para poder cambiar entre escenas, es necesario crear un script que tenga una función se encargue de cargar la escena a la que se llamará cuando se pulse el botón. Al igual que con el botón de salir, que dispondrá de una función para cerrar la ejecución.

4.9 Sonido

Al estar centrado en las experiencias hápticas que puede tener el usuario, la inmersión ha pasado a un segundo plano. Es por ello por lo que el sonido no ha sido algo que se haya tenido muy en cuenta para el desarrollo del proyecto. A pesar de eso se añadieron dos pistas de audio al proyecto, una que suena durante todo el rato, que simula el ambiente de la ciudad. Y otra para la lluvia. Estos sonidos se descargaron de la página Pixabay que ofrece audios gratuitos de uso libre, por lo que no supone ningún problema de Copyright. [43]

Para añadir estos sonidos es necesario añadir un objeto *Audio Source* que se encarga de reproducir el sonido. Estos pueden configurarse para que los sonidos comiencen en función del código, aunque en este caso suenan en todo momento. Para la lluvia, se ha configurado el sonido de forma que suene únicamente cuando el jugador se encuentre cerca de la fuente. [Figura 66]

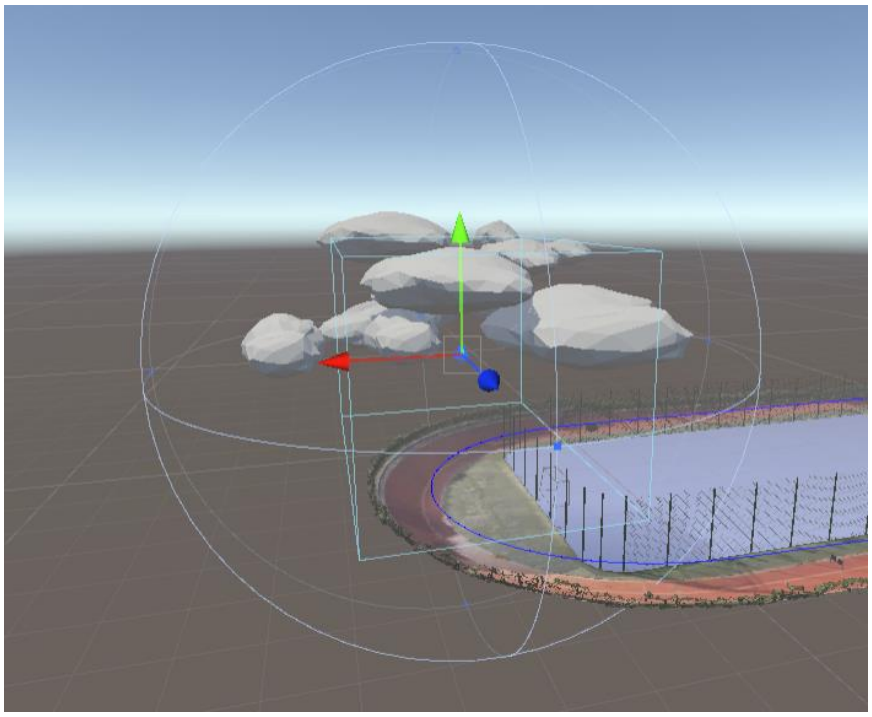


Figura 66: Área de efecto del sonido de lluvia.

Con esto podemos configurar la distancia mínima y máxima a la que el jugador escuchará el sonido, haciendo que solo lo escuche cuando se encuentre bajo la lluvia.

Capítulo 5. Resultados

Una vez vistos todos los puntos del desarrollo, se puede ver el resultado final. Siguiendo el recorrido, el usuario empieza en una parte sin obstáculos para poder acostumbrarse al entorno y poder probar el funcionamiento de la palanca.

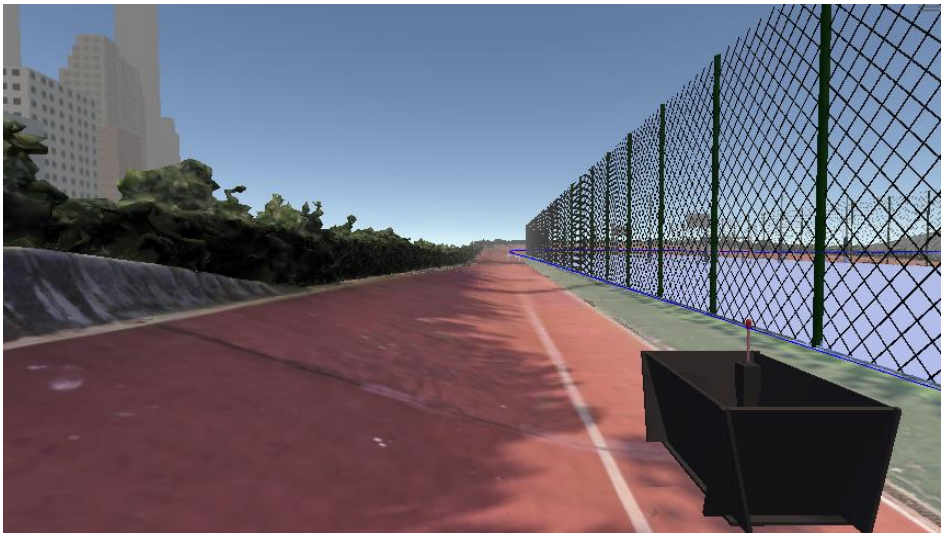


Figura 67: Parte 1 del circuito.

Al avanzar el usuario llega hasta las dos primeras barreras, que frenarán la velocidad del usuario y le darán respuesta háptica, ya sea en los guantes o en el chaleco. Más adelante se puede ver una torre de conos que derribará el vehículo, justo antes de la primera curva del velódromo.

Figura 68: Parte 2 del circuito

Después de la curva el vehículo pasará por la grava, donde se verá afectado por el terreno, sintiéndolo el usuario en el chaleco. Más adelante llegará a los badenes, que harán botar al

vehículo, junto a las torres de conos a los lados, colocadas para que el usuario las derribe con las manos.

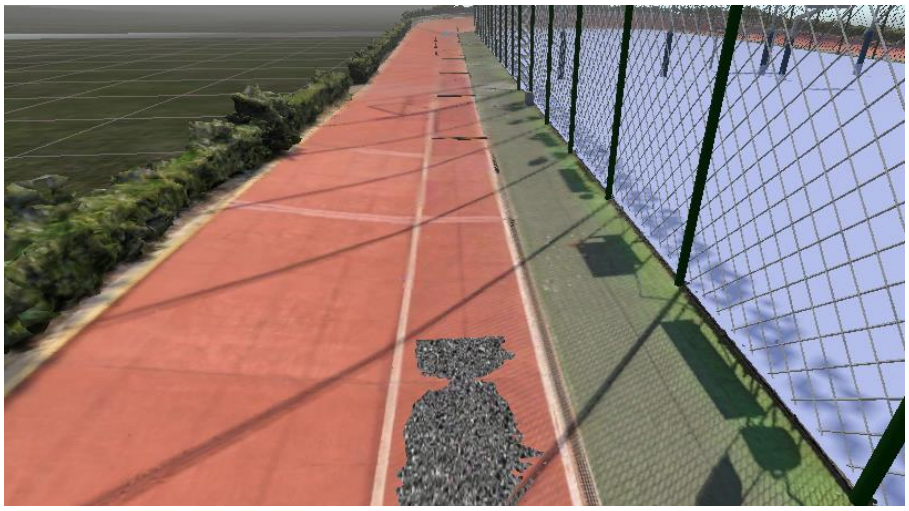


Figura 69: Parte 3 del circuito.

Para terminar el recorrido llegaremos a la segunda curva, no sin antes encontrarnos con una barrera doble que frenará al jugador antes de pasar por la lluvia, que sentirá tanto en la parte superior del chaleco como en las manos con una vibración suave.

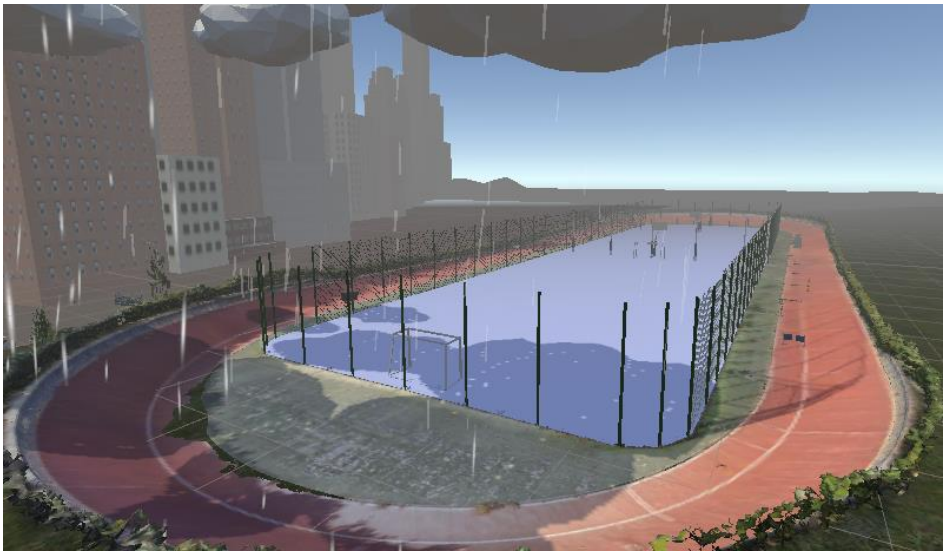


Figura 70: Parte 4 del circuito.

Capítulo 6. Trabajo futuro

Este proyecto es una demostración técnica enfocada en añadir funcionalidades en un entorno cerrado, lo que implica que los obstáculos no siempre guardan coherencia entre sí. Una posible dirección de trabajo futura sería crear un recorrido con una temática específica. Por ejemplo, se podría desarrollar una ruta que simule un paraje natural, como se había propuesto al inicio del proyecto, lo cual no solo mejoraría la inmersión, sino también la experiencia general del usuario.

Al observar el proyecto, surgen varias oportunidades de mejora. Uno de los aspectos a perfeccionar es el funcionamiento de las manos virtuales. Aunque actualmente cumplen su propósito, con más trabajo se podría lograr un manejo mucho más refinado y realista, mejorando la interacción del usuario con el entorno.

Como se ha dicho antes, tanto el sonido como el menú son aspectos que podrían haber sido ampliados y mejorados. Aunque el objetivo principal del proyecto fue probar las funcionalidades de los dispositivos hápticos, es indiscutible que el sonido juega un papel crucial en aplicaciones inmersivas por lo que podría ser algo a tener en cuenta. Mientras que al menú se le podrían añadir varias configuraciones, siendo una de ellas el movimiento manual, en caso de haber sido desarrollado, o deshabilitar y habilitar dispositivos hápticos.

En cuanto al movimiento dentro del entorno, una opción que se consideró fue implementar un sistema de movimiento libre. Aunque esto podría haberse logrado dentro del tiempo estipulado, se llegó a la conclusión de que no era necesario dado el enfoque del proyecto. En este caso, se optó por un movimiento automático, lo que libera las manos del usuario para que puedan interactuar libremente con el espacio.

Capítulo 7. Conclusiones

Realizar este trabajo ha servido para demostrar cómo la integración de la tecnología háptica en entornos de Realidad Virtual puede elevar significativamente el nivel de inmersión y realismo percibido por los usuarios. A través del desarrollo y la implementación de una aplicación con retroalimentación háptica, se ha comprobado que la inclusión de sensaciones físicas en la experiencia virtual aumenta la satisfacción del usuario al crear una conexión más profunda con el entorno simulado.

Este proyecto también ha permitido identificar los límites actuales de la tecnología háptica. Dispositivos como el chaleco o los guantes hápticos no pueden ofrecer una inmersión completa por sí solos, en cambio, funcionan como complementos de otras aplicaciones. Por esta razón, su uso se destaca particularmente en el ámbito de los videojuegos, donde pueden sacar a relucir su potencial al transmitir sensaciones que acompañan a las acciones realizadas por el usuario.

Estas tecnologías aún tienen que avanzar bastante en el desarrollo si quieren convertirse en algo más común para todos los usuarios. El precio actual es muy alto para lo que pueden llegar a ofrecer. Aún así, está claro que en unos años veremos un claro aumento tanto de la realidad virtual como de las tecnologías hápticas.

Una aplicación inmersiva como esta supone muchas capas de desarrollo. Aunque se han dejado de lado algunas cosas, en esta memoria se resume el proyecto, destacando las cosas más importantes y que más trabajo han supuesto.

A nivel personal, el desarrollo de este proyecto ha representado un desafío considerable. Aunque ya contaba con cierta experiencia en Unity, la realidad virtual era un ámbito en el que apenas había trabajado, más allá de lo aprendido en las clases. Del mismo modo, las tecnologías hápticas eran completamente nuevas para mí, teniendo que aprender desde cero su funcionamiento.

Este proyecto no habría sido posible sin la colaboración del iTEAM, que no solo facilitó el acceso a los dispositivos necesarios, sino que también brindó su apoyo continuo, resolviendo dudas y ofreciendo orientación, especialmente durante la fase de ideación del proyecto.

Si bien considero que el resultado final podría haber sido mejor, estoy satisfecho con lo que se logró, a pesar de las limitaciones de tiempo y los problemas que surgieron durante su desarrollo. Cabe destacar que el proyecto solo podía ser trabajado en el laboratorio inmersivo debido a la disponibilidad de los dispositivos, al que solo podía acudir por las mañanas.

En resumen, este trabajo ha sido una experiencia enriquecedora que me ha permitido profundizar en un área de gran interés para mí, como es Unity, además de potenciar las habilidades de programación adquiridas durante la carrera. También me brindó la oportunidad de experimentar con tecnologías a las que, de otro modo, no habría tenido acceso. El desarrollo de este proyecto

ha contribuido significativamente a mi crecimiento profesional y técnico, especialmente a la hora de resolver los problemas encontrados, buscando información de la que poder sacar las partes que me interesaban, adaptando las partes necesarias a mi proyecto.

Capítulo 8. Anexo

8.1 Enlace al proyecto

Enlace al zip con el proyecto de Unity:

- ImmersiveHaptics.zip

8.2 Índice de figuras

Figura 1: Sensorama de Morton Heilig (1950). [1] 3
Figura 2: *The Teleyeglasses* de Hugo Gernsbac (1963). [1] 3
Figura 3: Prototipo *Oculus* de Palmer Luckey (2012). [1] 4
Figura 4: Guantes magnéticos (Russels) de Half-Life: Alyx. [2] 4
Figura 5: Póster de The Tingler (1959). [4] 5
Figura 6: Mando DualSense de Sony. [5] 6
Figura 8: Logo de Rockstar Games. [10] 8
Figura 9: Logo de Unreal Engine. [11] 9
Figura 10: Logo de Unity Technologies. [13] 10
Figura 11: Tienda de Assets de Unity. [14] 11
Figura 12: *Tactosy for Arms* de bHaptics. [15] 11
Figura 13: Ejemplo de uso de los guantes G1 de HaptX. [16] 12
Figura 14: Leap Motion Controller de Ultraleap. [17] 13
Figura 15: Traje háptico Teslasuit. [18] 14
Figura 16: Tanvas Touch Dev Kit. [19] 15
Figura 17: Gafas o casco de realidad virtual. [21] 15
Figura 18: Logo de Oculus. [22] 16
Figura 19: Kit de VR de Valve Index. [24] 17
Figura 20: Gafas VR HTC Vive. [25] 17
Figura 21: Gafas HoloLens de Microsoft. [26] 18
Figura 22: Ventana de componentes de Unity. 19
Figura 23: Pestaña de jerarquías. (Hierarchy) 20
Figura 24: Gafas VR Meta Quest Pro. [29] 20
Figura 25: Parte frontal del TactSuit X40. [30] 22
Figura 26: Parte trasera del TactSuit X40. [30] 22
Figura 27: TactGlove DK2. [31] 23
Figura 28: Pestaña “Explorar” de bHapticsPlayer. 24
Figura 29: Pestaña “Device” de bHapticsPlayer. 24
Figura 30: Lista de proyectos de bHaptics Designer. 25
Figura 31: Pestaña de creación de proyecto de bHaptics Designer. 26
Figura 32: Pestaña inicial de bHaptics Developer. 26
Figura 33: Pestaña Events de bHaptics Developer. 27
Figura 34: Developer Window de Unity. 27
Figura 35: Añadiendo un evento en el Developer. 28
Figura 36: Poster 5G Robot Race del iTEAM. 29

Figura 36: Añadir evento de bhaptics usando el nombre. [35] 30
Figura 37: Añadir eventos de forma directa. [35] 31
Figura 38: Creación de entorno *low poly.* 31
Figura 39: Prototipo de carretera. 32
Figura 40: Edición del elemento *Terrain* 32
Figura 41: Visualización del terreno con árboles. 33
Figura 42: Modelo del velódromo del iTEAM 33
Figura 43: Configuración del paquete de integración VR. 34
Figura 44: Objeto *OVRCameraRig* en la pestaña Hierarchy. 35
Figura 46: Código encargado de la orientación del jugador. 36
Figura 47: *Trigger* que controla la velocidad del objeto. 36
Figura 48: Carretera creada para el velódromo. 37
Figura 49: Objeto *spline* que recorre el velódromo. 38
Figura 50: Lista de etiquetas del proyecto 39
Figura 51: Matriz de físicas del proyecto. 39
Figura 52: Visualización de las barreras en el circuito. 40
Figura 53: Patrón de vibración de las barreras. 41
Figura 54: Torres de conos. 41
Figura 55: Obstáculo grava. 42
Figura 56: Patrón de vibración de la grava 42
Figura 57: Obstáculo badenes. 43
Figura 58: Obstáculo lluvia. 43
Figura 59: Código que genera la sensación de lluvia. 44
Figura 60: Mano izquierda con sus *colliders*. 45
Figura 61: Patrón de vibración dedo corazón mano derecha. 46
Figura 62: Mano agarrando un cubo. 46
Figura 63: Vehículo del usuario 47
Figura 64: Acciones predeterminadas del controlador 48
Figura 65: Escena inicial. 48
Figura 66: Área de efecto del sonido de lluvia. 49
Figura 67: Parte 1 del circuito. 50
Figura 68: Parte 2 del circuito 50
Figura 69: Parte 3 del circuito. 51
Figura 70: Parte 4 del circuito. 51

8.3 Bibliografía

[1] 'Xperimenta Cultura Historia de la Realidad Virtual'. [Online]. Available: https://xperimentacultura.com/historia-de-la-realidad-virtual/

[2] E. equipo de 3DJuegos, 'Análisis de Half-Life Alyx. Revolución confirmada', Mar. 2020. [Online]. Available: https://www.3djuegos.com/juegos/half-life-alyx/analisis/revolucion-confirmada-200329-8759

[3] 'Aplicaciones y usos de la Realidad Virtual Teseo Noticias'. [Online]. Available: https://teseo.es/noticias/aplicaciones-y-usos-de-la-realidad-virtual/

[4] 'Háptica', Jan. 2024. [Online]. Available: https://es.wikipedia.org/w/index.php?title=H%C3%A1ptica&oldid=157752601#Tecnolog%C3%ADa_h%C3%A1ptica

[5] 'Mando inalámbrico DualSense El innovador mando de PS5'. [Online]. Available: https://www.playstation.com/es-es/accessories/dualsense-wireless-controller/

[6] K. Rangarajan, H. Davis, and P. H. Pucher, 'Systematic Review of Virtual Haptics in Surgical Simulation: A Valid Educational Tool?', *J Surg Educ*, vol. 77, no. 2, 2020, doi: 10.1016/j.jsurg.2019.09.006.

[7] A. Wade, 'UK firm claims haptic VR surgery simulation can cost less than a cadaver', Aug. 2018. [Online]. Available: https://www.theengineer.co.uk/content/news/uk-firm-claims-haptic-vr-surgery-simulation-can-cost-less-than-a-cadaver/

[8] 'Motor de videojuego', Aug. 2024. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Motor_de_videojuego&oldid=162164948#cite_note-11

[9] '¿Qué es un motor de videojuegos? – Observatorio del Gabinete de Tele-Educación'. [Online]. Available: https://blogs.upm.es/observatoriogate/2018/07/04/que-es-un-motor-de-videojuegos/

[10] U. Ruelas, '¿Qué es un motor de videojuegos (game engine)?', Jul. 2017. [Online]. Available: https://codingornot.com/que-es-un-motor-de-videojuegos-game-engine

[11] 'Unreal Engine 5'. [Online]. Available: https://www.unrealengine.com/es-ES/unreal-engine-5

[12] J. García, '"Black Myth Wukong" se lanzó hace apenas unas horas. El nuevo juego chino acaba de hacer historia', Aug. 2024. [Online]. Available: https://www.xataka.com/videojuegos/black-myth-wukong-se-lanzo-hace-apenas-unas-horas-nuevo-juego-chino-acaba-hacer-historia

[13] 'Plataforma de desarrollo en tiempo real de Unity Motor de 3D, 2D, VR y AR'. [Online]. Available: https://unity.com

[14] 'Unity Asset Store'. [Online]. Available: https://assetstore.unity.com

[15] 'Buy next generation full body haptic suit - bHaptics TactSuit'. [Online]. Available: https://www.bhaptics.com

[16] 'Home'. [Online]. Available: https://haptx.com/

[17] 'Digital worlds that feel human Ultraleap'. [Online]. Available: https://www.ultraleap.com/

[18] 'Teslasuit Meet our Haptic VR Suit and Glove with Force Feedback', Mar. 2022. [Online]. Available: https://teslasuit.io/

[19] 'Tanvas - Surface haptic technology and products'. [Online]. Available: https://tanvas.co/

[20] 'Realidad virtual', Aug. 2024. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Realidad_virtual&oldid=162046487

[21] 'Casco de realidad virtual', Mar. 2024. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Casco_de_realidad_virtual&oldid=159089911

[22] 'Oculus VR', Feb. 2024. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Oculus_VR&oldid=158310242

[23] 'Reality Labs', Aug. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Reality_Labs&oldid=1238591525

[24] 'Valve Index'. [Online]. Available: https://store.steampowered.com/valveindex

[25] 'VIVE European Union Discover Virtual Reality Beyond Imagination'. [Online]. Available: https://www.vive.com/eu/

[26] 'Microsoft HoloLens Tecnología de realidad mixta para empresas'. [Online]. Available: https://www.microsoft.com/es-es/hololens

[27] 'Unity - Manual: La ventana de Jerarquía (Hierarchy)'. [Online]. Available: https://docs.unity3d.com/es/530/Manual/Hierarchy.html

[28] 'Meta Quest Pro', Sep. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Meta_Quest_Pro&oldid=1243763983

[29] 'Meta Quest Pro: realidad mixta premium Meta Store'. [Online]. Available: https://www.meta.com/es/quest/quest-pro/#overview

[30] 'TactSuit X40 - bHaptics TactSuit'. Accessed: Sep. 04, 2024. [Online]. Available: https://www.bhaptics.com/shop/tactsuit-x40/

[31] 'TactGlove DK2 - bHaptics TactSuit'. Accessed: Sep. 04, 2024. [Online]. Available: https://www.bhaptics.com/shop/tactglove/

[32] 'bHapticsPlayer'. [Online]. Available: https://www.bhaptics.com

[33] 'bHaptics Designer'. [Online]. Available: https://designer1.bhaptics.com/

[34] 'bHaptics developer portal'. [Online]. Available: https://developer.bhaptics.com/

[35] 'Plug in deployed events to Unity'. [Online]. Available: https://bhaptics.notion.site/Plug-in-deployed-events-to-Unity-33cc33dcfa44426899a3f21c62adf66d

[36] 'Low-Poly Simple Nature Pack 3D Landscapes Unity Asset Store'. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153

[37] 'Low Poly Nature - FREE Vegetation 3D Vegetation Unity Asset Store'. [Online]. Available: https://assetstore.unity.com/packages/3d/vegetation/low-poly-nature-free-vegetation-134006

[38] 'Meta XR Interaction SDK Integration Unity Asset Store'. [Online]. Available: https://assetstore.unity.com/packages/tools/integration/meta-xr-interaction-sdk-265014

[39] Octodemy, 'Create more with splines Unity 2022 splines', Aug. 2023. [Online]. Available: https://www.youtube.com/watch?v=n-o2e4KxbW4

[40] Paridot, 'Making a Spline Train Using Rigidbodies to Move Unity C#', Oct. 2022. [Online]. Available: https://www.youtube.com/watch?v=fNIrXVBuMqI

[41] 'Road props for games, diffuse map, atlas, lp. 3D Roadways Unity Asset Store'. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/roadways/road-props-for-games-diffuse-map-atlas-lp-238835

[42] 'VRTK v4 Tilia Package Importer Utilities Tools Unity Asset Store'. [Online]. Available: https://assetstore.unity.com/packages/tools/utilities/vrtk-v4-tilia-package-importer-214936

[43] '90,000+ Free Sound Effects for Download - Pixabay - Pixabay'. [Online]. Available: https://pixabay.com/sound-effects/