



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de un módulo base interactivo para la simulación
de entornos inmersivos mediante realidad virtual*

Grado en Ingeniería Mecánica

ALUMNO: Jorge Gallego García

DIRECTORES: Xavier Núñez Nieto

Francisco Troncoso Pastoriza

CURSO ACADÉMICO: 2024-2025

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de un módulo base interactivo para la simulación
de entornos inmersivos mediante realidad virtual*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

La realidad virtual (RV) ha avanzado significativamente en los últimos años, convirtiéndose en una herramienta clave en diversos ámbitos, desde la educación hasta la simulación profesional. Es por ello que las Fuerzas Armadas muestran un creciente interés por esta innovadora tecnología.

Los entornos de RV requieren una integración eficiente de dispositivos y periféricos para potenciar la sensación de realismo. La conexión de múltiples dispositivos entre sí y con el entorno virtual permite una experiencia más completa y envolvente para el usuario.

En este trabajo se desarrolla un módulo base de interacción en realidad virtual, en el que se puedan realizar interacciones en el mundo virtual de manera intuitiva para cualquier usuario, sin necesidad de conocimientos avanzados. Además, el proyecto se caracteriza por la integración de múltiples dispositivos, tanto hápticos como una plataforma de movilidad, para maximizar la interactividad. El estudio abarca desde el diseño del módulo hasta la programación de los distintos escenarios mediante programación gráfica.

De este modo, se establecen bases sólidas para el desarrollo de futuras aplicaciones en ámbitos como la formación académica y profesional, el entretenimiento interactivo y la simulación avanzada para entornos laborales y especializados.

PALABRAS CLAVE

Interacción virtual, Blueprints, Inmersividad, Dispositivos hápticos, Conectividad de periféricos.

AGRADECIMIENTOS

En primer lugar, quiero expresar mi agradecimiento a Don Xavier Núñez Nieto y Don Francisco Manuel Troncoso Pastoriza, quienes han sido mis tutores en este Trabajo Fin de Grado. Su dedicación y constante disposición para orientarme han sido claves durante este último periodo. Gracias a su guía, he podido afrontar los retos de este proyecto con mayor confianza y claridad.

También me gustaría agradecer a mis compañeros del laboratorio de Realidad Virtual, con quienes he compartido momentos que no solo han enriquecido esta experiencia académica, sino que han hecho mucho más amena y llevadera la estancia en el laboratorio.

Por último, me gustaría agradecer de corazón a las personas de mi círculo más cercano, por su compañía constante y por estar presentes, incluso en los momentos más exigentes.

CONTENIDO

Contenido	1
Índice de Figuras	4
1 Introducción.....	9
1.1 Descripción del apartado.....	9
1.2 Antecedentes	9
1.3 Motivación	10
1.4 Objetivos	10
1.5 Estructura del Trabajo	10
2 Estado del arte	12
2.1 Descripción del apartado.....	12
2.2 Industria 4.0	12
2.2.1 Realidad Extendida	13
2.2.1.1 Realidad Virtual.....	14
2.3 Simulación en las FAS	17
2.3.1 Armada	17
2.3.2 Ejército de Tierra	19
2.3.3 Ejército del Aire.....	20
2.4 Software y Periféricos.....	21
2.4.1 Epic Games.....	22
2.4.2 Unreal engine.....	22
2.4.3 Gafas VR Meta Quest 3.....	24
2.4.4 ProTubeVR	25
2.4.5 BHaptics	27
2.4.6 KAT VR.....	27
3 Desarrollo	29
3.1 Descripción del apartado.....	29
3.2 Diseño y modelado.....	30
3.2.1 Introducción a Unreal Engine 5	30
3.2.1 Modelado de entorno base	34
3.2.2 Entorno de Interacción.....	36

3.2.2.1 Sección de Colisión	36
3.2.2.2 Diseño Botón	38
3.2.2.3 Sección de Puzzle	41
3.2.3 Entorno de Retroalimentación Háptica.....	44
3.2.3.1 Sección de Pistola	44
3.2.3.2 Sección de Chaleco Háptico	45
3.2.4 Pizarra Digital	46
3.2.5 Pirámide Central	52
3.2.6 Entorno de Plataforma VR.....	56
3.3 Realidad Virtual	58
3.4 Integración de Periféricos	60
3.4.1 ProtubeVR	60
3.4.2 BHaptics	62
3.4.3 Plataforma VR	67
4 Resultados	71
4.1 Descripción del apartado.....	71
4.2 Escenario Final.....	71
4.3 Entorno de Interacción.	72
4.3.1 Sección de Colision	72
4.3.2 Sección de Puzzle	74
4.4 Entorno de Retroalimentación Háptica.	76
4.4.1 Sección de Pistola	77
4.4.1 Sección de Chaleco Háptico	79
4.5 Pirámide Central	79
4.6 Pizarra Digital	80
4.7 Entorno de plataforma VR	82
5 Conclusiones y líneas futuras	85
5.1 Descripción del apartado.....	85
5.2 Conclusiones	85
5.2.1 Entorno de interacción.....	85
5.2.2 Integración de periféricos	85

5.2.3 Pizarra digital.....	85
5.2.4 Pirámide central	86
5.2.5 Plataforma VR	86
5.2.6 Conclusión final.....	86
5.3 Líneas futuras	86
5.3.1 Integración de Plataforma KATVR	86
5.3.2 Evolución del proyecto	87
6 Bibliografía.....	88
Anexo I: Implicaciones Sociales, Económicas y Ambientales	90
Anexo II: Reflexiones Éticas y Sociales	93
Anexo III: Bloque de Código para Botón de Pelotas	95
Anexo IV: Bloque de Código para Anclaje de Pieza	97
Anexo V: Bloque de Código para Interacción con Botones de la Pizarra	99

ÍNDICE DE FIGURAS

Figura 2-1 Industria 4.0 [1]	12
Figura 2-2 Realidad Extendida (XR) [3]	13
Figura 2-3 Videojuego de realidad aumentada <i>Pokémon Go</i> [4]	13
Figura 2-4 Videojuego de realidad virtual [6]	14
Figura 2-5 Realidad mixta en industria [8]	14
Figura 2-6 Simulador de Navegación de la ENM	15
Figura 2-7 Sistema inmersivo de RV [10]	15
Figura 2-8 Simulador de cirugía para adiestramiento de cirujanos [12]	16
Figura 2-9 Simulador de esquí [13]	16
Figura 2-10 Simulador TALOS [14]	18
Figura 2-11 Simulador de plataforma SIMPLAT [14]	18
Figura 2-12 Simulador de artillería SIMACA [15]	19
Figura 2-13 Simulador de duelo SIB [15]	20
Figura 2-14 Videojuego de simulación de Jefe de Pelotón [15]	20
Figura 2-15 Simulador de F-18 [16]	21
Figura 2-16 Logo e interfaz inicial de Epic Games [17] [19]	22
Figura 2-17 Gráficos en Unreal Engine 5 [19]	23
Figura 2-18 Interfaz Unreal Engine 5.5 [20]	24
Figura 2-19 Gafas VR Meta Quest 3 y controladores Touch Plus [21]	25
Figura 2-20 Ejercicios Fitness de FitXR [21]	25
Figura 2-21 ForceTube Haptic Gunstock [22]	26
Figura 2-22 ProVolver Haptic Pistol [22]	26
Figura 2-23 TacSuit de BHaptics [23]	27
Figura 2-24 Plataforma KAT Walk C2 Core [24]	27
Figura 3-1 Esquema de ciclo de desarrollo del proyecto	29
Figura 3-2 Interfaz Epic Games Launcher	30
Figura 3-3 Navegador de Proyectos Unreal Engine	30
Figura 3-4 Interfaz de Unreal Engine 5	31
Figura 3-5 Menú superior de Unreal Engine 5	31
Figura 3-6 Barra de herramientas principal	32

Figura 3-7 Menús desplegables de Herramientas.....	32
Figura 3-8 Barra de herramientas del <i>Viewport</i>	33
Figura 3-9 Panel <i>Details</i>	33
Figura 3-10 Panel <i>Outliner</i>	34
Figura 3-11 Panel del <i>Content Browser</i>	34
Figura 3-12 Módulo base inicial	35
Figura 3-13 <i>Material Graph</i>	35
Figura 3-14 Pirámide central y <i>Player Start</i>	36
Figura 3-15 Componentes y <i>viewport</i> de <i>BP_Pelota</i>	37
Figura 3-16 Componentes y <i>viewport</i> de <i>Grabbable_SmallCube</i>	37
Figura 3-17 Prismas de Soporte de pelotas y cubos	38
Figura 3-18 <i>Viewport</i> de <i>BP_Boton_Pelota</i> y <i>Hand VR</i>	38
Figura 3-19 Código para bajar botón	39
Figura 3-20 Código para subir botón	39
Figura 3-21 Código para instanciar pelota	40
Figura 3-22 Sección de colisión	40
Figura 3-23 Componentes y <i>viewport</i> de <i>BP_PiezaPuzzle</i>	41
Figura 3-24 Componentes y <i>viewport</i> de <i>BP_AnclajePuzzle</i>	42
Figura 3-25 Código para anclaje de piezas.....	42
Figura 3-26 Código para comprobar la orientación de las piezas	43
Figura 3-27 Sección de Puzzle	44
Figura 3-28 Componentes y <i>viewport</i> de <i>BP_Diana</i>	45
Figura 3-29 Sección de Pistola.....	45
Figura 3-30 Componentes y <i>viewport</i> de <i>BP_Cartel</i>	46
Figura 3-31 Sección de chaleco háptico	46
Figura 3-32 Jerarquía y <i>viewport</i> de <i>W_Pantalla</i>	47
Figura 3-33 Código para cambiar de imagen con el teclado	48
Figura 3-34 Componentes y <i>viewport</i> de <i>BP_Pizarra</i>	48
Figura 3-35 Código de Función <i>ActivarBoton1</i>	49
Figura 3-36 <i>Sphere Collision</i> de <i>HandVR</i>	49
Figura 3-37 Código para interactuar con el widget.....	50

Figura 3-38 Código para añadir vibración al pulsar pantalla	50
Figura 3-39 Componentes y viewport de <i>BP_PuntoAnclaje</i>	51
Figura 3-40 Código para anclar <i>BP_Pizarra</i>	51
Figura 3-41 Componentes y viewport de <i>BP_Spawnear</i>	52
Figura 3-42 Código para hacer aparecer objetos mediante el ratón	53
Figura 3-43 Jerarquía y variables de <i>Widget Menu</i>	53
Figura 3-44 Jerarquía, interfaces y contenido de <i>W_Boton</i>	54
Figura 3-45 Código para obtener asociar texto y clase a cada botón	54
Figura 3-46 Código para crear botones y añadirlos al <i>Widget Menu</i>	55
Figura 3-47 Código de la función <i>Spawn Actor</i>	55
Figura 3-48 Código para activar la función <i>Spawn Actor</i>	55
Figura 3-49 Lista de elementos de la variable <i>Object List</i>	56
Figura 3-50 Corte en pared del Entorno de Plataforma VR	56
Figura 3-51 Componentes y viewport de <i>BP_Door</i>	57
Figura 3-52 Código para hacer desaparecer la puerta	57
Figura 3-53 Materiales del pasillo del entorno de plataforma	58
Figura 3-54 Gafas Meta Quest 3	58
Figura 3-55 Steam VR	59
Figura 3-56 Funciones de cada botón en controladores VR	59
Figura 3-57 ProVolver Haptic Pistol e interfaz de ProtubeVR Companion-App	60
Figura 3-58 Plugin ForceTubeVR-Unreal Engine	61
Figura 3-59 Código para activar el disparo	61
Figura 3-60 Código para introducir efecto háptico de pistola	62
Figura 3-61 ChalecoTactSuitX40 e interfaz de bHaptics Player	62
Figura 3-62 Plugin de BHaptics compatible con Unreal Engine	63
Figura 3-63 Plugins en Unreal Engine	63
Figura 3-64 Portal de Bhaptics Developer	63
Figura 3-65 Project Settings Unreal Engine	64
Figura 3-66 Blueprint GameInstance y código para inicio y fin de comunicación háptica	64
Figura 3-67 BHaptics Designer	65
Figura 3-68 Efectos importados en BHaptics Developer	66

Figura 3-69 Código para añadir retroceso al disparar con la pistola.....	67
Figura 3-70 Código para activar la retroalimentación háptica de explosión.....	67
Figura 3-71 Plataforma KATVR Walk C2 Core y KATVR Shoes.....	68
Figura 3-72 Descarga de KAT Gateway.	68
Figura 3-73 Interfaz de KATVR Gateway	69
Figura 3-74 Plugin de KATVR compatible con Unreal Engine.....	69
Figura 3-75 Panel de <i>World Settings</i> de Unreal Engine	70
Figura 4-1 Resultado final del escenario	71
Figura 4-2 Entorno de Interacción.....	72
Figura 4-3 Vista VR de sección de colisión	72
Figura 4-4 Vista VR de agarrar pelotas	73
Figura 4-5 Vista VR de lanzamiento e impacto de pelotas	73
Figura 4-6 Vista VR de reaparición de pelotas	74
Figura 4-7 Vista VR de comportamiento de los cubos	74
Figura 4-8 Vista VR de sección del puzzle	74
Figura 4-9 Vista VR de orientación incorrecta de la pieza del puzzle	75
Figura 4-10 Vista VR de anclado de pieza del puzzle	75
Figura 4-11 Vista VR de anclado de pieza en la posición correcta.....	76
Figura 4-12 Puzzle completo	76
Figura 4-13 Entorno de retroalimentación háptica.....	76
Figura 4-14 Usuario equipado correctamente para sección de retroalimentación háptica.....	77
Figura 4-15 Vista VR de pistolas	77
Figura 4-16 Vista VR de disparo con pistola	78
Figura 4-17 Vista VR de disparo con pistola a objetos del escenario	78
Figura 4-18 Vista VR de botón para sentir efecto háptico	79
Figura 4-19 Vista VR de menú desplegable para la visualización de objetos	79
Figura 4-20 Vista VR de selección de objetos y vaciar pirámide	80
Figura 4-21 Vista VR de cambio de transparencia.....	80
Figura 4-22 Vista VR de coger pizarra	81
Figura 4-23 Vista VR de pizarra movida	81
Figura 4-24 Vista VR de anclaje de pizarra	81

Figura 4-25 Entorno de Plataforma VR	82
Figura 4-26 Usuario sobre plataforma VR	83
Figura 4-27 Panel de <i>World Settings</i> en Unreal Engine	84
Figura 4-28 Vista VR de apertura del pasillo	84

1 INTRODUCCIÓN

1.1 Descripción del apartado

En este apartado se presentan los antecedentes y el contexto en el que se desarrolla este Trabajo de Fin de Grado (TFG). También se detallan los principales objetivos que se busca alcanzar, justificando su importancia dentro del ámbito de aplicación. Finalmente, se describe la estructura del trabajo y la manera en que se ha organizado su contenido.

1.2 Antecedentes

La Cuarta Revolución Industrial, o Industria 4.0, está transformando los procesos productivos y la manera en que interactuamos con la tecnología en nuestra vida cotidiana. Gracias a la digitalización y la interconectividad, han surgido herramientas avanzadas como la inteligencia artificial, el Big Data o la robótica, que están revolucionando sectores clave como la manufactura, la logística y la atención sanitaria. Dentro de este contexto, la realidad virtual (RV) ha adquirido un papel fundamental, permitiendo la creación de entornos inmersivos que mejoran la formación, la simulación y la optimización de procesos de manera más efectiva.

Este Trabajo de Fin de Grado se centrará en la realidad virtual, con el objetivo de diseñar y desarrollar un escenario inmersivo que simule situaciones específicas en un entorno digital dinámico y realista. La RV ofrece una experiencia interactiva y realista, brindando un espacio seguro para la experimentación sin los riesgos ni costos asociados al mundo real y físico. Su implementación está en auge en diversos ámbitos, como la educación, la medicina y la industria, debido a su capacidad para mejorar la capacitación, la toma de decisiones y la eficiencia operativa.

Uno de los principales beneficios de la realidad virtual es su capacidad para sumergir al usuario en entornos tridimensionales envolventes, facilitando el aprendizaje experiencial y la retención del conocimiento adquirido de forma más natural. En el sector industrial, por ejemplo, se utiliza para entrenar a trabajadores en el manejo de maquinaria o protocolos de seguridad sin exponerlos a situaciones peligrosas o costosas. En la medicina, los simuladores de RV permiten practicar procedimientos quirúrgicos en un ambiente controlado, mejorando notablemente la preparación de los profesionales de la salud.

Dado su creciente impacto en la transformación digital, la realidad virtual se presenta como una herramienta con un gran potencial para mejorar la eficiencia, la productividad y la seguridad en distintos sectores clave. A lo largo de este trabajo, se analizarán las aplicaciones y ventajas de la realidad virtual, explorando su implementación en diferentes contextos y contribuyendo al desarrollo de soluciones innovadoras en entornos digitales cada vez más complejos y exigentes.

1.3 Motivación

La realidad virtual se ha convertido en una herramienta clave para la formación y el entrenamiento en entornos que requieren precisión y toma de decisiones en tiempo real. En sectores como la defensa y la seguridad, la capacitación práctica es esencial para garantizar una respuesta efectiva en situaciones críticas. Sin embargo, los entrenamientos convencionales pueden verse limitados por restricciones económicas, logísticas e incluso de seguridad. En este contexto, los simuladores de realidad virtual ofrecen una solución innovadora, proporcionando entornos interactivos que mejoran el aprendizaje y la toma de decisiones en condiciones controladas.

Este proyecto tiene como objetivo el desarrollo de un escenario de realidad virtual, que permitirá a los usuarios familiarizarse con el manejo de objetos dentro de un entorno inmersivo. Además de facilitar la adaptación a la interacción en realidad virtual, este escenario básico servirá como base para la creación de simulaciones más específicas orientadas a la instrucción en distintas áreas, permitiendo su expansión y adaptación según las necesidades formativas.

1.4 Objetivos

El objetivo principal de este trabajo es el desarrollo de un escenario de realidad virtual en el que se puedan realizar distintas funciones interactivas. Además, se pretende lograr la integración de periféricos con el propósito de incrementar la sensación de realismo en el entorno virtual. De esta manera, el escenario permitirá una interacción natural e intuitiva, de modo que aquellos usuarios sin experiencia previa en realidad virtual puedan desenvolverse con facilidad y aprovechar al máximo la experiencia.

Para ello, se han marcado los siguientes objetivos específicos:

- Interacción con objetos: se pretende implementar un sistema que permita la manipulación de objetos dentro del entorno virtual, incluyendo acciones como cogerlos, soltarlos y lanzarlos, así como la detección de colisiones entre ellos.
- Integración de dispositivos hápticos: se trabajará en la integración del chaleco háptico y el accesorio háptico de la pistola, permitiendo al usuario sentir la retroalimentación de los dispositivos dentro de la inmersión virtual.
- Visualizador interactivo de objetos: se implementará una pirámide en el centro del escenario, que servirá como punto de referencia para la visualización de objetos. Sobre ella, se podrá hacer aparecer cualquier elemento para su visualización, facilitando su observación desde distintos ángulos.
- Pizarra digital: se creará una pantalla interactiva que el usuario podrá manipular dentro del escenario, permitiendo su desplazamiento y la posibilidad de anclarla en cualquier momento a su posición original.

Como objetivo adicional planteado para el trabajo, y aprovechando la reciente adquisición de una nueva plataforma de realidad virtual, se plantea su integración en el escenario desarrollado. El propósito es permitir el desplazamiento completo del usuario dentro del entorno virtual, mientras permanece estático en el mundo real. Con ello, se busca potenciar la sensación de inmersión.

1.5 Estructura del Trabajo

La estructura del trabajo se organiza en:

- 1) Introducción: En este apartado se presentan los antecedentes y el contexto en el que se enmarca el proyecto. Además, se detallan los principales objetivos que se buscan alcanzar, así como la motivación que impulsó su desarrollo.

- 2) Estado del Arte: Se realiza un análisis detallado sobre los ámbitos de aplicación del proyecto y las tecnologías que lo sustentan. También se incluyen las herramientas, software y equipos utilizados durante el desarrollo.
- 3) Desarrollo: Se expone de manera estructurada y detallada cada una de las etapas que conforman el diseño y la implementación del proyecto, describiendo los pasos seguidos desde su concepción hasta su culminación.
- 4) Resultados: En este apartado se analizan en profundidad los hallazgos obtenidos durante el desarrollo del trabajo. Se presentan las decisiones tomadas y se expone el producto final alcanzado, asegurando su coherencia con los objetivos planteados.
- 5) Conclusiones y líneas futuras: Se ofrece un resumen de los resultados obtenidos, evaluando en qué medida se han logrado los objetivos iniciales. Además, se sugieren posibles mejoras y futuras líneas de investigación que podrían expandir el trabajo desarrollado.
- 6) Bibliografía: Se recopilan y listan todas las fuentes de información consultadas y utilizadas como referencia en el desarrollo del proyecto.
- 7) Anexos: Se adjunta material complementario relevante para el proyecto. Se incluyen análisis sobre las implicaciones sociales, económicas y ambientales del trabajo, así como una reflexión ética. También se proporciona información adicional que pueden facilitar la comprensión del proyecto.

2 ESTADO DEL ARTE

2.1 Descripción del apartado

En este apartado se busca proporcionar una base que facilite la comprensión del resto del proyecto y lo contextualice para entender el proceso que ha llevado a su desarrollo. Para ello, se comenzará con un subapartado en el que se describirán las tecnologías 4.0 relacionadas con el trabajo. A continuación, se abordará la simulación virtual, incluyendo su aplicación desde un enfoque militar. Finalmente, se presentarán el software y el equipo empleados.

2.2 Industria 4.0

La Industria 4.0 representa una nueva etapa en la evolución de la producción, impulsada por tecnologías disruptivas como la inteligencia artificial, la realidad extendida, el Big Data y la impresión 3D, que actúan de manera colaborativa, como se muestra en la Figura 2-1. Además de integrar estas innovaciones entre sí, esta industria busca combinarlas con el ser humano para fomentar un desarrollo tecnológico continuo. Estas tecnologías están transformando las reglas del juego, fusionando productos físicos con servicios digitales y obligando a las empresas a adaptarse con rapidez. Por ello, es fundamental que las compañías enfoquen sus esfuerzos en la creatividad, el conocimiento y la tecnología para destacar en un entorno altamente competitivo.

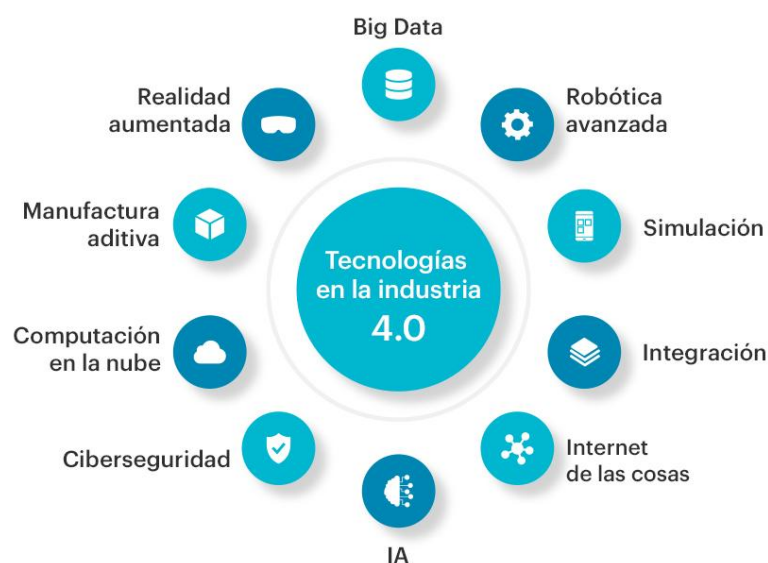


Figura 2-1 Industria 4.0 [1]

Asimismo, esta revolución tecnológica ha logrado un mundo más interconectado que nunca, generando nuevas oportunidades de crecimiento a través del comercio electrónico y las plataformas digitales. Sin embargo, también plantea desafíos, como la superación de barreras entre mercados, la adaptación a nuevas normativas y el aprendizaje continuo para manejar tecnologías en constante evolución. En este contexto, la Industria 4.0 no solo impulsa avances, sino que también nos desafía a redefinir la manera en que trabajamos y colaboramos en un mundo cada vez más globalizado [2].

2.2.1 Realidad Extendida

La realidad extendida (XR) es un término que engloba un conjunto de tecnologías inmersivas que fusionan el mundo físico con el digital. A través de dispositivos como gafas y auriculares, la XR permite a los usuarios interactuar con entornos virtuales de manera más natural e intuitiva. Como se muestra en la Figura 2-2, esta tecnología abarca desde la realidad virtual (RV), que sumerge al usuario por completo en un mundo digital, hasta la realidad aumentada (RA), que superpone elementos digitales sobre el mundo real, y la realidad mixta (RM), que combina aspectos de ambas.

Gracias a la XR, es posible explorar nuevos universos, visualizar información de forma más intuitiva y colaborar con otros de maneras antes unimaginables, transformando así la manera en que vivimos, trabajamos y nos relacionamos.

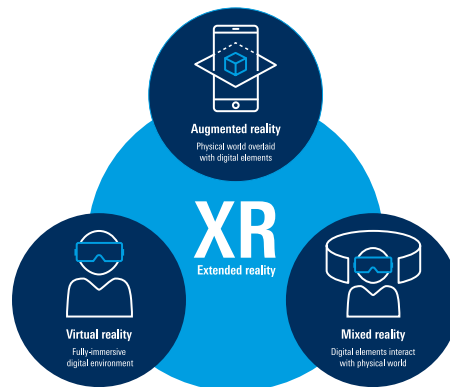


Figura 2-2 Realidad Extendida (XR) [3]

La realidad aumentada (RA) es una tecnología que combina el mundo real con elementos digitales, creando experiencias inmersivas y enriquecedoras al permitir la visualización de información adicional sobre un entorno físico. Gracias a la RA, podemos interactuar con nuestro entorno de una manera completamente nueva, superponiendo capas de información digital en tiempo real.

Esta tecnología no solo ofrece entretenimiento, sino que también tiene aplicaciones prácticas en diversos campos, como la educación, la medicina y el comercio [4]. Un ejemplo de realidad aumentada es el popular juego *Pokémon Go*, representado en la Figura 2-3.



Figura 2-3 Videojuego de realidad aumentada *Pokémon Go* [4]

La realidad virtual (RV) es como una ventana a un mundo completamente nuevo creado por una computadora. Al utilizar dispositivos como cascos o gafas especiales que se aprecian en la Figura 2-4, podemos sumergirnos en entornos virtuales que parecen increíblemente reales. Estos mundos virtuales pueden ser lugares imaginarios o recreaciones de lugares reales, y nos permiten explorar, interactuar y experimentar como si estuviéramos realmente allí. Gracias a la RV, podemos visitar lugares lejanos, probar nuevas experiencias y hasta aprender habilidades de una manera completamente inmersiva [5].



Figura 2-4 Videojuego de realidad virtual [6]

La realidad mixta (RM) es una tecnología que fusiona lo mejor de la realidad virtual y la aumentada, creando un espacio híbrido donde objetos digitales y físicos coexisten e interactúan en tiempo real. La RM permite a los usuarios manipular y explorar entornos virtuales superpuestos a su mundo real, ofreciendo experiencias inmersivas y colaborativas con aplicaciones en diversas industrias, desde el diseño y la arquitectura hasta la medicina y el entretenimiento [7]. En la Figura 2-5 se muestra el uso de elementos de RV, como pueden ser las gafas que portan los 3 individuos, en combinación con el mundo real.



Figura 2-5 Realidad mixta en industria [8]

2.2.1.1 Realidad Virtual

Como se ha dicho anteriormente, para que un sistema sea considerado de realidad virtual, debe sumergir al usuario en un entorno tridimensional generado por computadora, donde pueda interactuar de manera natural y en tiempo real con objetos virtuales que poseen propiedades físicas realistas. La RV debe generar una sensación de presencia, que se logra al estimular múltiples sentidos y al permitir al usuario moverse libremente dentro del entorno virtual.

El realismo en la realidad virtual va más allá de la simple representación visual. Implica crear una experiencia sensorial multidimensional que engañe a nuestros sentidos y nos haga creer que estamos presentes en un entorno diferente. Además de la alta calidad visual, factores como el sonido

especializado y la simulación de movimientos corporales contribuyen a una sensación de presencia más profunda. La psicología juega un papel crucial en la experiencia de inmersión, ya que nuestras expectativas, experiencias previas y creencias influyen en nuestra percepción de la realidad virtual.

La realidad virtual es un campo diverso y en constante evolución. No existe un modelo único, ya que las tecnologías y aplicaciones varían ampliamente. Sin embargo, podemos clasificar los sistemas de realidad virtual en tres categorías principales [9]:

- Sistemas de sobremesa o no inmersivos: Estos sistemas ofrecen una experiencia virtual a través de la pantalla de un ordenador. El usuario interactúa con el entorno digital, pero sin la inmersión completa que ofrecen otros sistemas. Un ejemplo de este tipo de sistemas es un PC apto para realidad virtual, mediante el cual podríamos jugar a un videojuego.
- Sistemas proyectivos o semi-inmersivos: Estos sistemas crean un entorno virtual proyectando imágenes en las paredes de una habitación. El usuario puede moverse e interactuar con el entorno. Son utilizados en simuladores, presentaciones y aplicaciones educativas. Un claro ejemplo es el Simulador de Navegación disponible en las Instalaciones de la ENM, reflejado en la Figura 2-6.



Figura 2-6 Simulador de Navegación de la ENM

- Sistemas inmersivos: Estos sistemas buscan sumergir completamente al usuario en el entorno virtual. Se utilizan cascos de realidad virtual que aíslan al usuario del mundo real. En la Figura 2-7 se muestra un sistema inmersivo en el cual el usuario está completamente inmerso en un entorno virtual mediante el uso de los distintos periféricos. Tiene aplicaciones prometedoras en diversos campos, como el entrenamiento militar o los videojuegos de RV avanzada.



Figura 2-7 Sistema inmersivo de RV [10]

La elección del sistema de realidad virtual depende de la aplicación específica y de los objetivos que se persiguen. Cada sistema tiene sus propias ventajas y desventajas en términos de inmersión, interactividad y coste.

Entre las principales áreas de aplicación de estos sistemas de realidad virtual, las más destacadas son [11]:

- Medicina: En este ámbito, la RV proporciona a los profesionales nuevas herramientas. Ofrece simulaciones quirúrgicas realistas que permiten practicar procedimientos complejos sin riesgos para los pacientes, mejorando su precisión y reduciendo errores. Además, facilita el aprendizaje de anatomía y fisiología a través de modelos 3D interactivos, lo que optimiza la comprensión de estructuras y procesos biológicos. De esta manera, la RV mejora la formación y las habilidades de los médicos, lo que se traduce en una atención más segura y efectiva para los pacientes. La Figura 2-8 muestra un ejemplo de intervención quirúrgica simulada.



Figura 2-8 Simulador de cirugía para adiestramiento de cirujanos [12]

- Deportes: La RV ofrece nuevas oportunidades tanto para atletas como para aficionados. Los atletas pueden utilizarla para simular entornos de competencia, mejorar habilidades técnicas mediante retroalimentación en tiempo real y analizar su rendimiento en detalle. Además, facilita la rehabilitación de lesiones y reduce considerablemente el riesgo de lesión. Para los aficionados, ofrece experiencias inmersivas al brindar acceso a contenido exclusivo y permitir juegos y simulaciones deportivas. En la Figura 2-9 se muestra un simulador de esquí.



Figura 2-9 Simulador de esquí [13]

- Adiestramiento militar: Estos entornos inmersivos permiten a los militares practicar escenarios complejos y peligrosos sin los riesgos que se puedan dar en un entrenamiento en el mundo real. Además, facilitan la repetición de ejercicios y la evaluación del rendimiento de manera objetiva. La realidad virtual también fomenta el trabajo en equipo y la toma de

decisiones bajo presión, habilidades cruciales en el campo de batalla. A medida que la tecnología avanza, se espera que los simuladores de realidad virtual sean aún más realistas y efectivos en la preparación de los militares para diversas situaciones.

- Educación: La RV es capaz de transformar la educación mediante la inmersión de los estudiantes en experiencias interactivas y atractivas. Imagina explorar el cuerpo humano en 3D, viajar en el tiempo para presenciar eventos históricos o practicar habilidades complejas en entornos virtuales seguros. Este nuevo método, permite a los estudiantes comprender conceptos abstractos, desarrollar empatía y aumentar su motivación. A pesar de la existencia de desafíos como los costos y la necesidad de capacitación, la RV tiene el potencial de revolucionar la educación, haciéndola más accesible, personalizada y efectiva para todos los estudiantes.

2.3 Simulación en las FAS

La simulación es fundamental en las Fuerzas Armadas españolas, abarcando desde simuladores de vuelo hasta entornos virtuales de combate. Las Fuerzas Armadas apuestan por estos sistemas para alcanzar la eficacia en el adiestramiento y la planificación de misiones, siendo una herramienta fundamental en la preparación para el combate y la conducción de operaciones.

España cuenta con una amplia gama de simuladores para los tres ejércitos, permitiendo un seguimiento personalizado de la formación. El futuro de la simulación se enfoca en tecnologías inmersivas como la realidad virtual y aumentada, que crearán entornos más realistas e interactivos, junto con una mayor integración de la inteligencia artificial para simulaciones más dinámicas y adaptativas.

2.3.1 Armada

Como resultado de un análisis exhaustivo, la Armada ha implementado un programa de modernización que incorpora medios de simulación avanzados para la enseñanza, instrucción y entrenamiento. Estos recursos permiten a los usuarios experimentar entornos bélicos navales y situaciones de emergencia de alto riesgo que podrían ocurrir durante una operación real. El enfoque principal de la simulación se centra en las escuelas de enseñanza, como la Escuela Naval Militar (ENM), además de las escuelas de submarinos, buceo, o aeronavales.

Los simuladores con los que la Armada cuenta hoy en día son los siguientes [14]:

- Simuladores de inmersión:
 - Simulador de inmersión a gran profundidad con aire (50 m)
 - Simulador de inmersión a gran profundidad con mezclas respirables (90 m)
 - Simulador de ventilación y entrega de contenedores estancos para submarinos accidentados, a bordo del buque Neptuno.
- Adiestramiento del combatiente:
 - VBS3: Simulador para modelado de escenarios de combate.
 - SCATT: Simulador de tiro.
 - VICTRIX: Simulador de tiro con implementación de movimiento del usuario.
- Navegación y Táctica:
 - SIMNAV. Simulador de maniobra y navegación de la ENM.
 - GALEÓN: Simulador Táctico de la ENM. Adiestramiento práctico en táctica naval.
 - SCOMBA-T: Simulador de los sistemas de combate de los barcos de la Armada.
 - SCE: Sistema de Control de Embarcaciones.
 - SACAZ: Simulador para adiestramiento en cazaminas.
 - SATS: Simulador táctico para submarinos de la clase s-70.
 - SIMTAC: Simulador táctico para submarinos de la clase s-80.

- TALOS (Figura 2-10): Simulador de táctica en mando y control de Infantería de Marina.



Figura 2-10 Simulador TALOS [14]

- Simulador de Plataforma:
 - SISMA: Simulador para adiestramiento en emergencias para submarinos clase s-70.
 - SIMPRA Simulador de propulsión en submarinos de la clase s-70.
 - SIFAS: Simulador para adiestramiento en fugas y vías de agua.
 - Simulador SICP: Consolas individuales, con capacidad para asimilar el control de los sistemas SICP instalados a bordo de los buques de la armada.
 - SOTV-EAV8B Plus: Simulador de vuelo de avión de ala fija.
 - SOTV-ESH60B LAMPS: Simulador de vuelo de helicóptero.
 - SIMPLAT (Figura 2-11): Simulador de Plataforma. Submarinos clase s-80.



Figura 2-11 Simulador de plataforma SIMPLAT [14]

- Sistemas de Comunicaciones:
 - DIANA: Simula el funcionamiento del sistema de distribución avanzada de datos de navegación.
 - GMDSS (Sistema de salvamento y seguridad marítima)
 - ICCS-5 (Sistema integrado de control de comunicaciones)
- Simuladores para análisis y toma de decisiones:
 - EXODUS: Análisis para la evacuación en buques de nueva construcción
 - SIMDIS: Análisis lanzamiento misiles
 - ALSA: Simulación para la defensa antimisil

La Armada está apostando fuertemente por la digitalización. Esto se refleja en iniciativas como el Apoyo Logístico 4.0, que busca optimizar la gestión de recursos a través de plataformas digitales y modelos de simulación. El desarrollo del CESADAR (Centro de Simulación Avanzada de la Armada) es otro pilar fundamental, donde se crearán entornos virtuales inmersivos para el adiestramiento. Además, la Maqueta Digital, que integra datos de los proyectos F-110, S-80 y BAM, permitirá simular escenarios operativos complejos y evaluar el rendimiento de los sistemas de armas. En conjunto, estas iniciativas buscan mejorar la eficiencia, la seguridad y la preparación de la Armada a través de la simulación avanzada.

2.3.2 Ejército de Tierra

La simulación es fundamental para el Ejército de Tierra, optimizando la formación y preparación de sus miembros. Permite el aprendizaje eficiente del uso de armamento y material, mejora la planificación y ejecución de operaciones, y contribuye a la sostenibilidad económica y operativa. Además, fomenta un aprendizaje dinámico, complementa ejercicios reales y permite la repetición de prácticas.

El Ejército de Tierra cuenta con un amplia gama de simuladores [15]:

- Simuladores Virtuales: Los participantes operan en sistemas simulados que recrean entornos reales o ficticios, permitiendo practicar tareas complejas sin riesgos.
 - Helicópteros: FBMS, FMS.
 - Carros y vehículos de combate: EAO, LEOPARD, LEOPARDO.
 - Conducción: Simuladores de conducción y manejo de vehículos.
 - Artillería: CAÑON 35/90, SIMACA (Figura 2-12).
 - Misiles: C-100/C-90, EUMP, MISTRAL SALA, ROLAND, SPIKE.
 - UAV: PASI.



Figura 2-12 Simulador de artillería SIMACA [15]

- Simuladores constructivos: Se basan en el empleo de unidades simuladas en entornos simulados. Orientados principalmente al mando y control en las altas esferas.
- Simuladores reales o en vivo: Los participantes simulan operando en sistemas reales.
 - Simuladores de blancos: SAAB, LOMAH.
 - Simuladores de tiro: NOPTel, VITRIX, STS.
 - Simuladores de duelo: ADGUS, PIZARRO, SIB (Figura 2-13).
 - Simuladores NBQ: CAMSIM, AN/PDR-77.



Figura 2-13 Simulador de duelo SIB [15]

- Serious Games: Juegos comerciales basados en escenarios adaptados a las necesidades de enseñanza del Ejército de Tierra. Disponen de numerosos modos de juego, escenarios y tiene distintas aplicaciones, entre las que destacan:
 - Mando y control (Figura 2-14).
 - Táctica.
 - Transporte.
 - Topografía.
 - Reconocimiento.



Figura 2-14 Videojuego de simulación de Jefe de Pelotón [15]

Los simuladores son una herramienta esencial en el Ejército de Tierra, la cual forma parte de su adiestramiento, y contribuye a la mejora de la formación de los militares. Las líneas futuras se centran en desarrollar y mejorar los simuladores ya presentes, además de adquirir nuevos, con los que se consiga abarcar una mayor área.

2.3.3 Ejército del Aire

La simulación se ha convertido en un aliado estratégico para optimizar el entrenamiento de los pilotos del Ejército del Aire. Si bien la hora de vuelo es un recurso costoso, los simuladores ofrecen una alternativa rentable para complementar el entrenamiento real, en un entorno seguro. Sin embargo, es fundamental destacar que la simulación no puede reemplazar por completo la experiencia del vuelo real, y los pilotos deberán continuar su formación con vuelos reales. Por lo tanto, en este ejército, la

simulación se concibe como un complemento esencial, pero no como un sustituto de vuelo real. Actualmente, cuenta con los siguientes simuladores [16]:

- Simulador Básico de caza (Figura 2-15): Entrenamiento básico de tácticas y emergencias del 'F-18'.
- Simulador FMS (Full Mission Simulators) de caza: Entrenamiento en escenario de batalla.
- Simulador de C-101.
- Simulador de C-295.
- Simulador de F5M.
- Simulador de C-130.
- Simulador de RPA.
- Simulador de A-400M.
- Simulador de NH-90.
- Simulador de Apoyo aéreo.
- Simulador de torre de control.



Figura 2-15 Simulador de F-18 [16]

Aunque los simuladores representan una inversión considerable, su función principal es complementar el entrenamiento real, brindando un entorno seguro y rentable para que pilotos y personal técnico perfeccionen sus habilidades.

A mediano plazo, la capacidad de conectar simuladores en tiempo real para crear escenarios tácticos complejos promete transformar la formación de las tripulaciones. A largo plazo, la integración de tecnologías de cuarta y quinta generación ofrece un abanico de posibilidades aún por explorar, lo que sugiere que la simulación seguirá desempeñando un papel fundamental en el futuro del Ejército del Aire.

2.4 Software y Periféricos

Este apartado se centra en el software y los componentes que se utilizarán en el desarrollo del proyecto. Se incluye una descripción de cada componente y software, así como su función específica dentro del proyecto.

2.4.1 Epic Games

Fundada en 1991 por Tim Sweeney [17], Epic Games ha revolucionado la industria del videojuego con innovaciones tecnológicas y una visión de futuro. Desde sus inicios, ha evolucionado hasta convertirse en un referente en el desarrollo de motores de juego, juegos de alta calidad y plataformas de distribución digital. En la Figura 2-16 se muestra su logo e interfaz inicial.

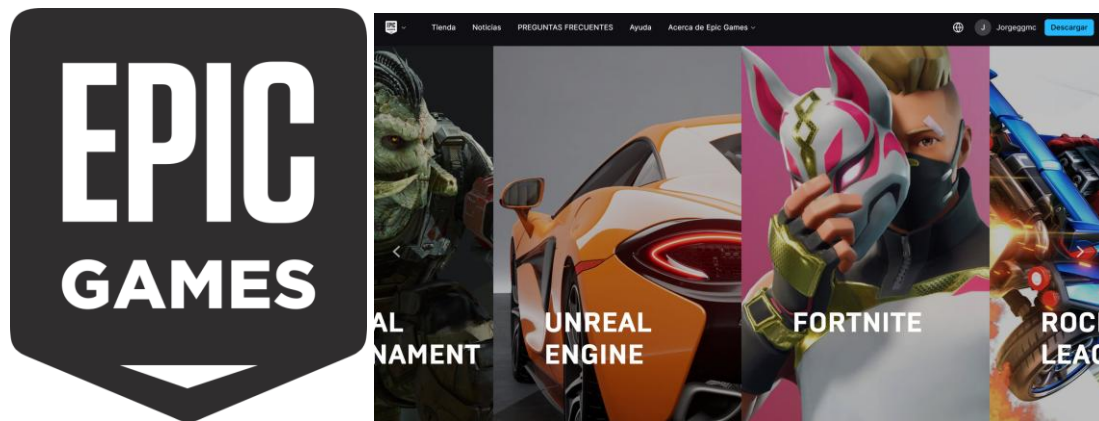


Figura 2-16 Logo e interfaz inicial de Epic Games [17] [19]

Uno de los pilares de su éxito es Unreal Engine, un motor gráfico utilizado en numerosos juegos que ha sido clave para el desarrollo de gráficos 3D realistas y experiencias inmersivas. Su versatilidad y potencia lo han convertido en una herramienta esencial para ingenieros y desarrolladores. Además de su motor gráfico, Epic Games ha destacado por el desarrollo de juegos icónicos como el fenómeno mundial Fortnite. Este último, un juego Battle Royale gratuito, ha alcanzado una popularidad masiva y ha generado un impacto significativo en la cultura popular y la industria.

La visión de Epic Games va más allá del desarrollo de juegos. La empresa ha incursionado en la creación de su propia tienda digital, Epic Games Store, que compite con otras plataformas y ofrece a los desarrolladores una alternativa para publicar sus juegos, impulsando la competencia en el mercado. El impacto de Epic Games en la ingeniería es innegable. La empresa ha sido pionera en el desarrollo de tecnologías y herramientas que han facilitado la creación de videojuegos más complejos y realistas. Su motor Unreal Engine, en particular, ha sido fundamental para la formación de ingenieros y desarrolladores, que han encontrado en esta herramienta un aliado para llevar a cabo sus proyectos.

En conclusión, Epic Games ha dejado una huella imborrable en la industria del videojuego y en la ingeniería. Su enfoque en la innovación, su visión de futuro y su compromiso con la calidad la han convertido en un referente para ingenieros y desarrolladores de videojuegos en todo el mundo.

2.4.2 Unreal Engine

En 1998, la primera versión de Unreal Engine (UE) [18] hizo posible la modificación de un juego de disparos en primera persona. Esta versión permitía modificar el comportamiento de los personajes no jugadores (NPC), también conocidos como bots (jugadores controlados por la computadora a través de inteligencia artificial) usando UnrealScript. Posteriormente, se agregaron características multijugador en línea a través del desarrollo de Unreal Tournament.

Para 2002, UE había mejorado a pasos agigantados, llevándolo a la siguiente generación con el desarrollo de herramientas de malla estática para manipular objetos, un motor de física que permite la interacción entre objetos y una nueva herramienta para crear escenas de corte. Esta mejora impulsó el desarrollo de Unreal Championship y Unreal Tournament 2003.

El desarrollo del juego ‘The Awakening’, impulsó a UE 2 con un sistema de animación y una mejora general con su motor existente. El desarrollo de velocidades de Internet más rápidas a principios de la década de 2000 también aumentó la demanda de juegos multijugador en línea. Unreal Tournament 2004 permitió a los jugadores participar en batallas en línea entre ellos. En 2005, el lanzamiento de Unreal Championship 2 en la consola de juegos Xbox reforzó las capacidades de UE en la consola. También vio la creación de una característica muy importante de una nueva cámara en tercera persona. Esto abrió mayores posibilidades en los tipos de juegos que se podían crear usando el motor.

Gears of War, una de las franquicias más conocidas en la industria de los videojuegos, empujó a Epic Games a crear y lanzar la tercera versión de su motor de juego, UE 3, en 2006. La mejora del motor de gráficos utilizó DirectX 9/10 para permitir la creación de personajes y objetos más realistas. La introducción de Kismet, que es un sistema de scripting visual, permitió a los diseñadores de juegos y niveles crear lógica de juego para un juego de combate más atractivo sin tener que profundizar en la escritura de códigos.

La creciente demanda de juegos móviles también llevó a UE 3 a ser empujado en la dirección de aumentar su compatibilidad con varias plataformas móviles. Todos estos avances y capacidades tecnológicas han hecho de UE3 la versión más popular de Unreal Engine y todavía se usa ampliamente en la actualidad.

UE 3 dominó el mercado durante 8 años hasta la llegada en 2014 de UE 4, que introdujo el mayor cambio al reemplazar Kismet con el nuevo concepto de Blueprint. El éxito de UE 4 se debe en gran parte a su innovador modelo de negocio. Al ofrecer el motor de forma gratuita y cobrar regalías solo a aquellos juegos que alcanzan ingresos significativos, Epic Games logró atraer a una gran comunidad de desarrolladores. La popularidad de UE 4 no se limitó a los videojuegos, ya que su potencia y flexibilidad lo convirtieron en una herramienta valiosa en la producción audiovisual, la arquitectura y la realidad virtual.

La versión de UE más reciente es la 5 [19]. Presentada en 2020 y lanzada en 2022, ha revolucionado la industria con tecnologías innovadoras como Nanite y Lumen. Estas herramientas permiten a los desarrolladores crear juegos con un nivel de detalle visual sin precedentes y una iluminación dinámica en tiempo real que rivaliza con el cine. Además de su impacto en los videojuegos, UE 5 está siendo utilizado en la producción cinematográfica, la visualización arquitectónica y el diseño de experiencias inmersivas. En la Figura 2-17 se muestra la calidad de imagen que se puede obtener con la última versión de UE.



Figura 2-17 Gráficos en Unreal Engine 5 [19]

Con Unreal Engine [20], la creación de experiencias de Realidad Extendida se vuelve accesible y versátil. Este motor gráfico líder en la industria no solo facilita la creación de una experiencia XR, sino que también permite su despliegue en una amplia gama de plataformas. Su compatibilidad con OpenXR y las API de los principales proveedores de hardware, como HoloLens, ARCore y Oculus, garantiza una adaptabilidad excepcional. Además, UE se destaca por su capacidad de optimización, lo que permite alcanzar un realismo visual impresionante en tiempo real sin comprometer el rendimiento en diversos dispositivos, desde HMD de alta resolución hasta dispositivos móviles. La personalización es otro punto fuerte, ya que ofrece la posibilidad de crear aplicaciones XR a medida utilizando scripting visual Blueprint o codificación en C++. Su marco extensible y el acceso al código fuente completo brindan una libertad creativa ilimitada. Finalmente, UE acelera el proceso de desarrollo gracias a su amplia biblioteca de plantillas, muestras y recursos gratuitos, lo que permite a los desarrolladores materializar sus ideas de forma rápida y eficiente.

La reciente actualización de UE a la versión 5.5 trae consigo mejoras sustanciales para los creadores de contenido. Los avances en animación permiten flujos de trabajo más eficientes y resultados más realistas, mientras que las herramientas de producción virtual se benefician de un mayor refinamiento. Además, el renderizado y los efectos visuales están listos para su uso en entornos de producción profesional. La introducción de MegaLights es un ejemplo del compromiso continuo con la innovación. En la Figura 2-18 se muestra la interfaz de diseño de UE 5.5.

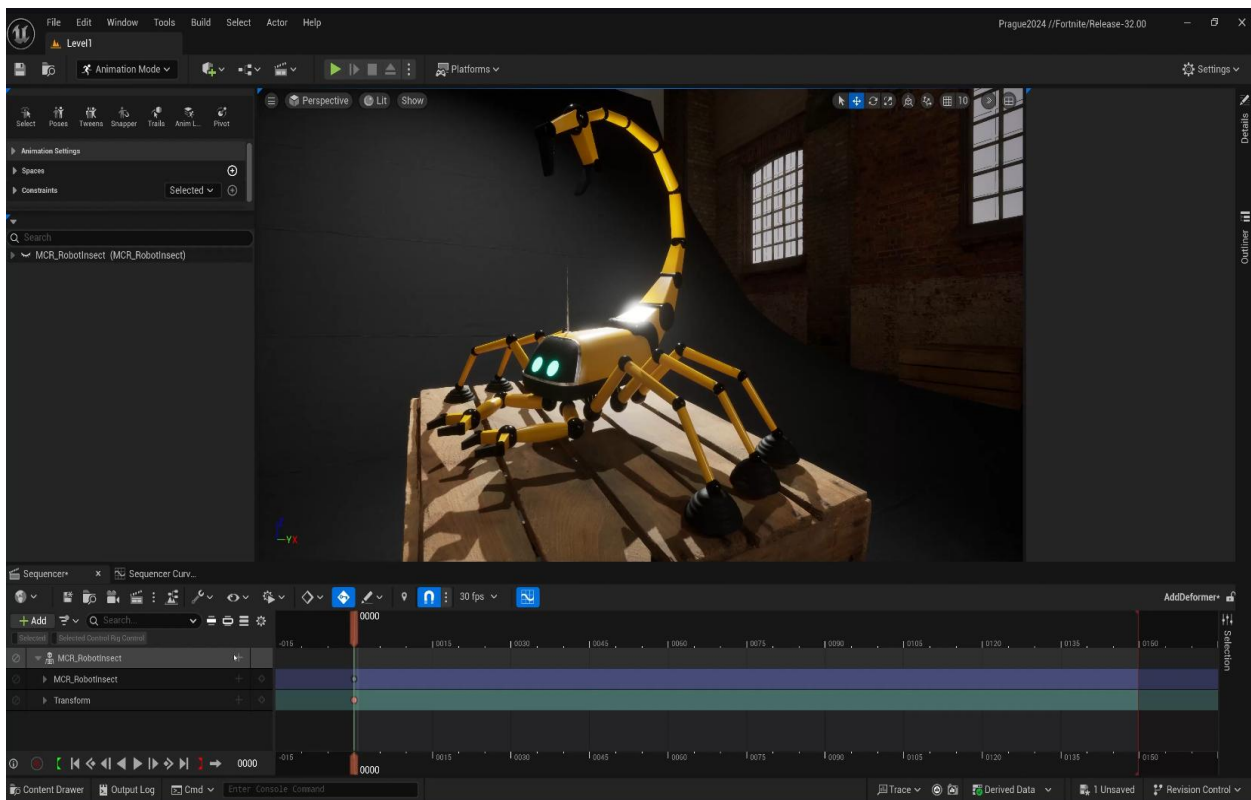


Figura 2-18 Interfaz Unreal Engine 5.5 [20]

2.4.3 Gafas VR Meta Quest 3

Las Meta Quest 3 [21] son las gafas de realidad virtual más avanzadas de Meta, diseñadas para ofrecer una experiencia inmersiva de alta calidad. Equipadas con el procesador Snapdragon XR2 Gen 2, duplican la potencia de procesamiento gráfico en comparación con la Meta Quest 2, garantizando tiempos de carga más rápidos y una jugabilidad más fluida. Además, cuentan con una resolución 4K y un diseño premium y estilizado, proporcionando imágenes más nítidas y realistas. Su diseño autónomo elimina la necesidad de una PC o consola externa, permitiendo una mayor libertad de movimiento.



Figura 2-19 Gafas VR Meta Quest 3 y controladores Touch Plus [21]

Estas gafas, mostradas en la Figura 2-19, incorporan una experiencia de realidad mixta, permitiendo a los usuarios interactuar con elementos virtuales superpuestos en el mundo real. Tienen un campo de visión amplio y una pantalla infinita que ofrece más píxeles, lo que brinda una experiencia visual envolvente. Además, cuentan con un diseño más ligero y cómodo, facilitando sesiones de uso prolongadas sin molestias. Con una amplia biblioteca de juegos y aplicaciones disponibles en la tienda de Meta Quest, estas gafas se posicionan como una opción destacada para los entusiastas de la realidad virtual.

Las Meta Quest 3 ofrecen una amplia variedad de experiencias inmersivas que van más allá de lo convencional. Las principales áreas de uso son:

- Entretenimiento: Visualización de videos en plataformas como YouTube o Prime Video.
- Videojuegos: Inmersión en diversos videojuegos.
- Deportes: Práctica de diversos deportes.
- Fitness: Para realizar ejercicios desde el hogar como se muestra en la Figura 2-20.



Figura 2-20 Ejercicios Fitness de FitXR [21]

2.4.4 ProTubeVR

ProTubeVR [22] es un dispositivo diseñado para mejorar la experiencia de VR al proporcionar una interfaz física que simula la manipulación de armas en entornos virtuales. Consiste en una estructura modular que permite a los usuarios acoplar sus controladores de VR, ofreciendo una sensación más realista y precisa durante el juego o simulaciones.

Este dispositivo no solo enriquece la experiencia del usuario, sino que también abre nuevas posibilidades en campos como la formación militar, la simulación médica y la educación, donde la interacción precisa con entornos virtuales es crucial.

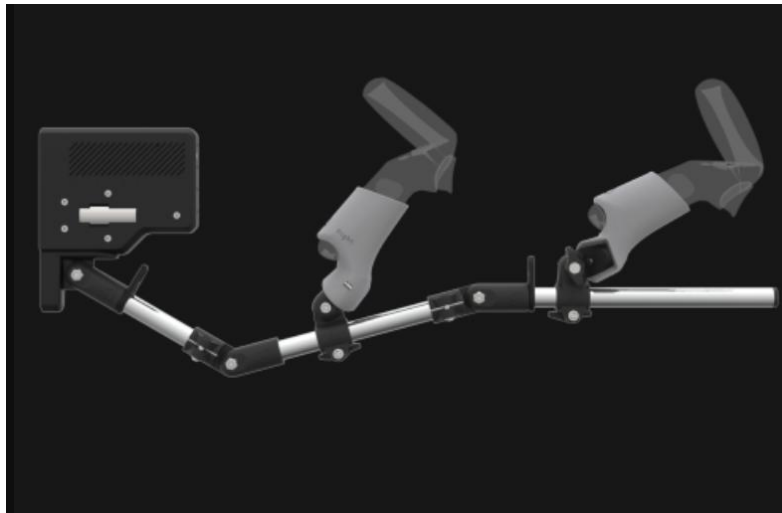


Figura 2-21 ForceTube Haptic Gunstock [22]

ProTubeVR ofrece varios dispositivos que simulan el manejo de armas y herramientas. Entre sus productos principales se encuentra el *ForceTube Haptic Gunstock*, apreciado en la Figura 2-21 formado por una estructura modular en la que se pueden acoplar los controladores de realidad virtual, proporcionando una sensación más realista al sujetar y apuntar. Este accesorio es ampliamente utilizado en juegos de disparos y simulaciones tácticas, ya que mejora la precisión del movimiento y reduce la fatiga del usuario al distribuir mejor el peso del dispositivo. Su diseño ajustable permite adaptarlo a diferentes tipos de armas virtuales y preferencias del jugador. Para mejorar aún más la experiencia, la estructura dispone de un sistema de vibración y retroceso que agrega retroalimentación háptica. La tecnología háptica avanzada mejora la inmersión al proporcionar una respuesta física más realista en cada interacción. Además, es compatible con múltiples plataformas y juegos de VR. Esto lo convierte en una opción versátil tanto para jugadores como para profesionales que requieren simulaciones de alta fidelidad, como cuerpos de seguridad y entrenamientos militares.

Otro accesorio innovador es el *ProVolver Haptic Pistol*, mostrado en la Figura 2-22, un controlador háptico diseñado específicamente para simular el uso de pistolas. Incorpora un sistema de retroceso motorizado que emula la sensación de disparo de un arma real. Además, cuenta con sensores que detectan la presión en el gatillo y la posición del dispositivo, lo que mejora la precisión y la respuesta en el juego. Su diseño ergonómico y su compatibilidad con múltiples plataformas de VR lo convierten en una herramienta ideal tanto para jugadores que buscan una experiencia más realista como para aplicaciones profesionales en entrenamientos tácticos y simulaciones de seguridad.



Figura 2-22 ProVolver Haptic Pistol [22]

2.4.5 BHaptics

Entre las innovaciones más destacadas en el campo de la VR se encuentra bHaptics [23], una empresa líder en el desarrollo de tecnologías hápticas que permiten una inmersión más realista. Con dispositivos como el TactSuit, mostrado en la Figura 2-23, que ofrece una experiencia multisensorial mediante el uso de actuadores hápticos distribuidos estratégicamente. Además, son capaces de generar vibraciones de distinta intensidad y duración según la interacción en el entorno virtual. Estos dispositivos son inalámbricos y compatibles con diversas plataformas, incluyendo visores de realidad virtual como Meta Quest y HTC Vive, lo que los hace accesibles a una amplia gama de usuarios.



Figura 2-23 TacSuit de BHaptics [23]

Estas soluciones mejoran la percepción táctil en videojuegos y simulaciones VR al replicar sensaciones como impactos, pulsaciones o efectos ambientales. Además, encuentran aplicaciones en campos como la medicina, la formación profesional y la rehabilitación. Estos dispositivos facilitan simulaciones realistas para sectores como las fuerzas armadas, donde la experiencia práctica en un entorno seguro es crucial. Además, su integración con software de terceros permite personalizar las respuestas hápticas según necesidades específicas, ampliando aún más sus posibilidades de aplicación.

2.4.6 KAT VR

KAT VR [24] es una empresa destacada en el ámbito de la RV, reconocida por desarrollar plataformas de locomoción que permiten a los usuarios moverse físicamente en entornos virtuales de manera natural y segura. Estas plataformas, conocidas como caminadoras omnidireccionales, están diseñadas para mejorar la inmersión en experiencias de RV al permitir movimientos en 360 grados.



Figura 2-24 Plataforma KAT Walk C2 Core [24]

Uno de los productos más representativos es el *KAT Walk C2 Core* [24], que se muestra en la Figura 2-24. Este dispositivo utiliza la tecnología KAT Optical Foot Tracking, basada en sensores ópticos que detectan movimientos hacia adelante, atrás, en zigzag y en curva. Además, permite realizar acciones como saltar, agacharse o arrodillarse. El *KAT Walk C2 Core* es compatible con el visor de RV Meta Quest. Para desarrolladores interesados en integrar las plataformas de KAT VR en sus proyectos, la empresa ofrece un SDK compatible con los motores de juego Unreal Engine. Este SDK facilita la incorporación de las funcionalidades de las caminadoras de KAT VR en los entornos virtuales.

En resumen, KAT VR ofrece soluciones avanzadas para la locomoción en entornos de realidad virtual, proporcionando dispositivos que mejoran la inmersión del usuario. Su compatibilidad con motores de desarrollo como Unreal Engine facilita a los desarrolladores la creación de experiencias de RV más envolventes y realistas.

3 DESARROLLO

3.1 Descripción del apartado

Este apartado detalla el proceso seguido para llevar a cabo el proyecto, estructurado en tres fases principales. En la Figura 3-1, se representa visualmente la secuencia de estas etapas.

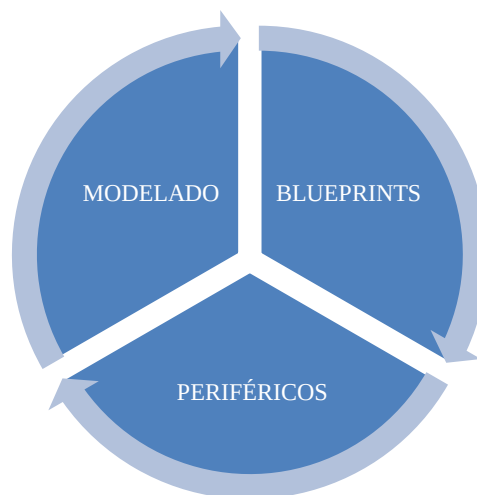


Figura 3-1 Esquema de ciclo de desarrollo del proyecto

Cada una de estas fases cumple un rol fundamental en la construcción del entorno de realidad virtual, contribuyendo de manera progresiva a su implementación completa. A continuación, se presenta un resumen de cada etapa, que se desarrolla con mayor detalle en los siguientes apartados:

1. Modelado: esta fase se centra en la creación de la estructura base y elementos principales del entorno virtual. Para facilitar su organización visual, cada entorno se ha diferenciado mediante el uso de distinto color, logrando un diseño simple pero atractivo.
2. Blueprints: las mecánicas de interacción y el comportamiento del entorno se han desarrollado mediante Blueprints, el sistema visual de programación de Unreal Engine, permitiendo definir respuestas dinámicas y eventos sin necesidad de escritura de código.
3. Periféricos: se basa en la integración de dispositivos en el entorno virtual. Su conexión con Unreal Engine se realiza mediante la instalación de plugins, que son complementos de software que amplían las funcionalidades del motor gráfico y permiten la compatibilidad con hardware externo.

3.2 Diseño y modelado

3.2.1 Introducción a Unreal Engine 5

El primer paso en el desarrollo del proyecto es la instalación de Unreal Engine 5, el motor de creación que servirá como base para la construcción del entorno virtual. Para ello, se utilizará Epic Games Launcher que aparece en la Figura 3-2, la plataforma oficial desde donde se descarga y gestiona Unreal Engine. Una vez instalado, se procederá a la configuración inicial del proyecto y la preparación del entorno de trabajo.

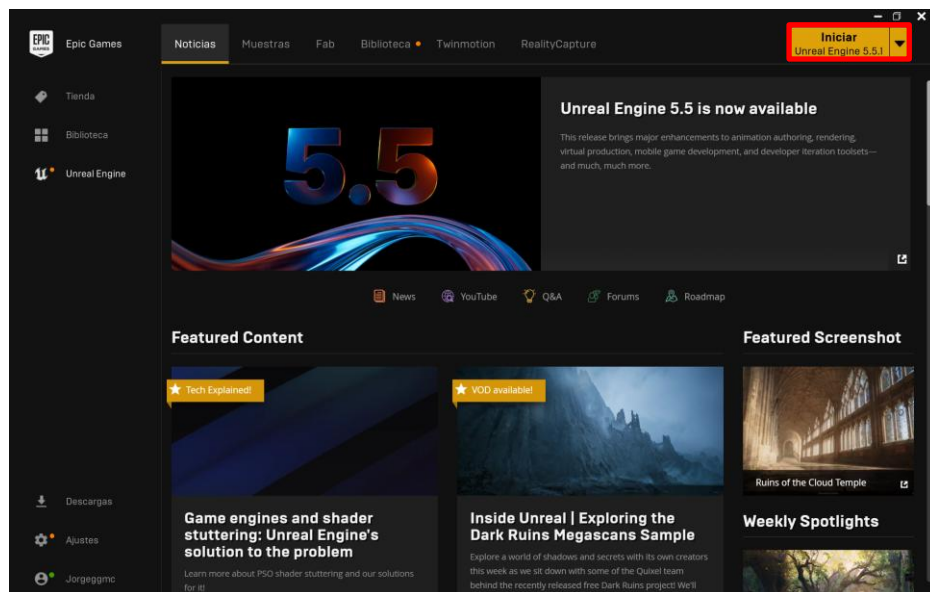


Figura 3-2 Interfaz Epic Games Launcher

Una vez instalado Unreal Engine 5, al abrir el programa se muestra la pantalla de inicio de proyectos, como se puede observar en la Figura 3-3. Desde este menú, se puede acceder a proyectos recientes o crear uno nuevo. Para este proyecto, se ha seleccionado la opción *Virtual Reality* dentro de la categoría *Simulation*, ya que proporciona una plantilla base optimizada para experiencias en realidad virtual. Tras definir la ubicación del proyecto y asignarle un nombre, se procede a su creación, lo que permite acceder al entorno inicial de desarrollo en Unreal Engine 5.

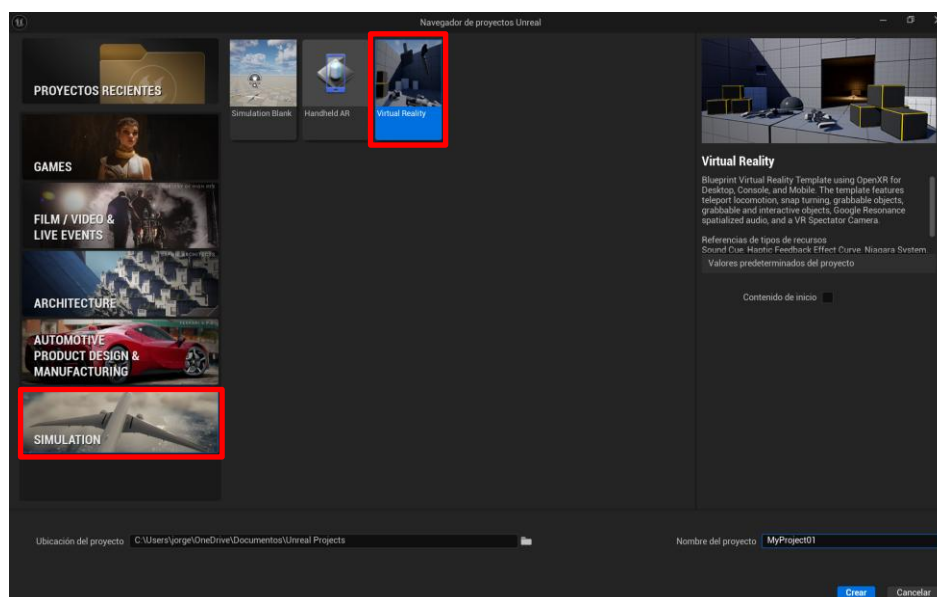


Figura 3-3 Navegador de Proyectos Unreal Engine

Una vez creado el proyecto utilizando la plantilla *Virtual Reality*, se accede a la interfaz de desarrollo de *Unreal Engine 5*. En esta vista inicial, se pueden observar los elementos predeterminados que conforman el entorno base para experiencias de realidad virtual, incluyendo el espacio de trabajo, las herramientas principales y la configuración inicial del proyecto. La siguiente imagen muestra la disposición de la interfaz al comenzar con esta plantilla.

La interfaz del editor de *Unreal Engine 5* se compone de diferentes paneles que permiten gestionar la escena y los elementos del proyecto de manera eficiente. En la Figura 3-4 se pueden identificar algunos de los más importantes, como el *Viewport*, donde se visualiza el nivel en tiempo real, el *Outliner*, que organiza los objetos de la escena, y el *Content Browser*, donde se almacenan los recursos del proyecto. Además, el Panel de *Details* ofrece opciones avanzadas para modificar las propiedades de los elementos seleccionados. En conjunto, estas herramientas permiten construir, ajustar y optimizar el entorno de realidad virtual, facilitando un flujo de trabajo estructurado y eficiente.

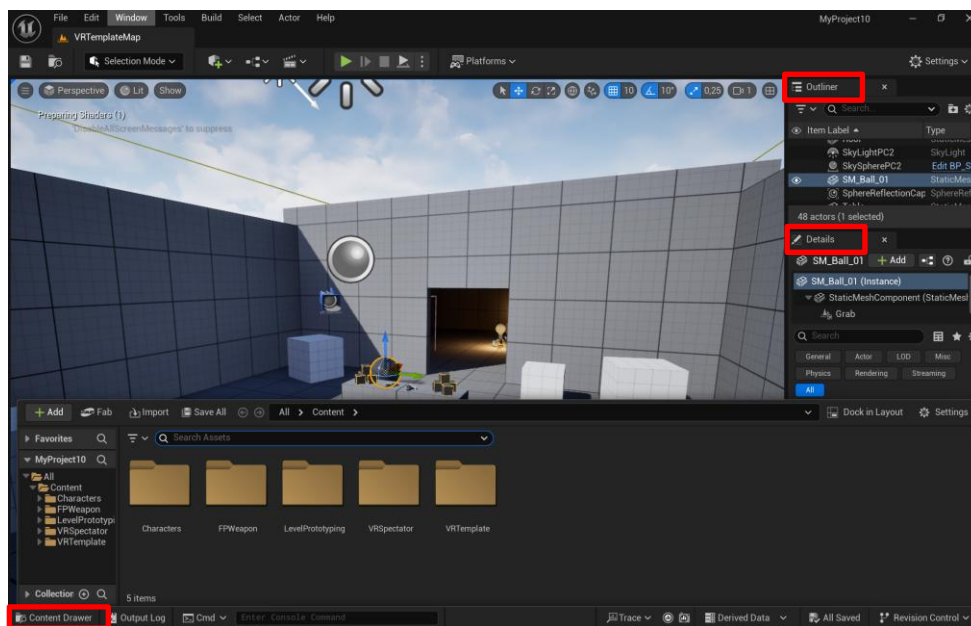


Figura 3-4 Interfaz de Unreal Engine 5

El menú superior de *Unreal Engine 5.5*, ampliado en la Figura 3-5, proporciona acceso a herramientas clave para la gestión del proyecto y la configuración del entorno de desarrollo. La pestaña *File* permite administrar archivos, crear, abrir y guardar niveles, mientras que *Edit* incluye opciones para modificar configuraciones y realizar acciones como deshacer y rehacer. *Window* facilita la organización de la interfaz, permitiendo personalizar la disposición de los paneles, y *Tools* ofrece herramientas adicionales para optimizar el desarrollo. *Build* es esencial para compilar la escena y aplicar cambios en iluminación y navegación, mientras que *Select* y *Actor* ayudan a gestionar y manipular los elementos dentro del nivel. Finalmente, *Help* proporciona acceso a documentación y asistencia técnica, facilitando el aprendizaje y la resolución de problemas dentro del editor.

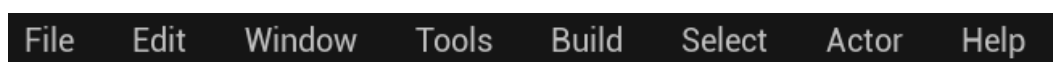


Figura 3-5 Menú superior de Unreal Engine 5

La barra de herramientas ubicada debajo del menú superior y mostrada en la Figura 3-6 contiene accesos directos esenciales para la gestión y simulación del proyecto en *Unreal Engine 5.5*. Incluye opciones para guardar el progreso, buscar activos dentro del contenido y cambiar el *Selection Mode*, lo que permite modificar la forma en que se interactúa con los objetos dentro del nivel. También ofrece herramientas para añadir nuevos actores a la escena y gestionar la disposición de los elementos. En la

parte derecha se encuentran los controles de simulación, donde el botón *Play* permite ejecutar la escena y probar interacciones en tiempo real, mientras que las opciones de pausa y detener controlan la ejecución. Finalmente, el menú *Platforms* permite configurar el destino de la compilación, asegurando la compatibilidad del proyecto con diferentes dispositivos y sistemas, incluyendo visores de realidad virtual.



Figura 3-6 Barra de herramientas principal

Esta barra de herramientas posee menús desplegables como los de la Figura 3-7. El submenú de la izquierda permite importar contenido y agregar actores a la escena. Se observan distintas funcionalidades como *Get Content*, donde se encuentran opciones para importar activos, o *Content Browser*, desde donde poder gestionar recursos. En *Place Actors*, se pueden añadir elementos esenciales como actores básicos, luces, formas geométricas, efectos visuales y herramientas cinematográficas. También ofrece acceso a la lista completa de actores y un panel dedicado para su colocación. Es una herramienta clave para construir y personalizar el entorno del proyecto.

Además, en el apartado *Lights*, se pueden añadir diferentes tipos de luces para establecer la iluminación del escenario según las necesidades del proyecto. En nuestro caso Hemos configurado una *Directional Light* con parámetros realistas para lograr profundidad, sombras naturales y reflejos precisos, asegurando una iluminación equilibrada y envolvente. Es una solución eficiente para entornos abiertos o donde se requiere una iluminación global sin necesidad de múltiples fuentes de luz. Además, combinada con una *Sky Light*, se pueden suavizar las sombras y mejorar la iluminación indirecta en el entorno.

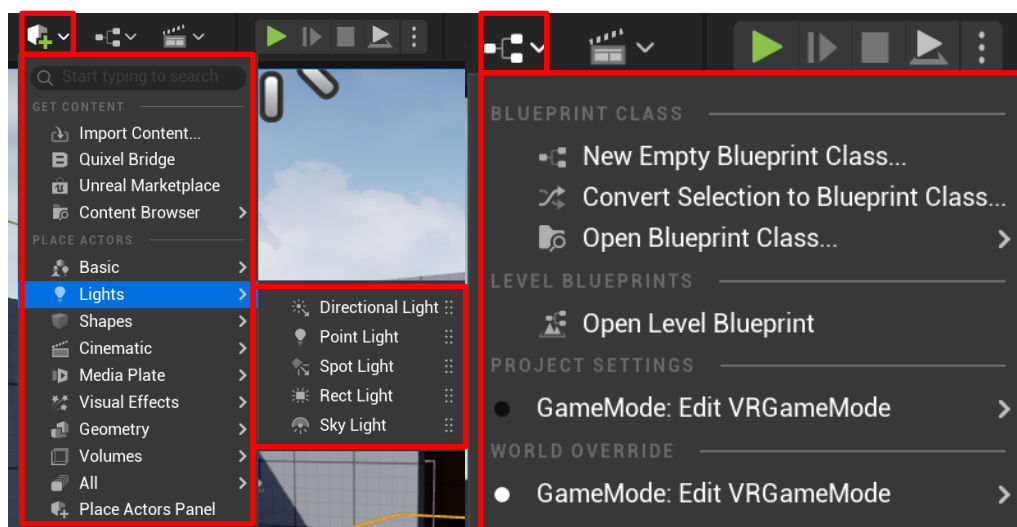


Figura 3-7 Menús desplegables de Herramientas

El desplegable de la derecha de la Figura 3-7 proporciona acceso a la gestión y ajustes de Blueprints y del *GameMode*. En *Blueprint Class*, se pueden crear nuevos Blueprints vacíos, convertir actores en Blueprints o abrir los existentes. La sección *Level Blueprints* permite editar el Blueprint del nivel, donde se pueden definir lógicas específicas para la escena actual. En *Project Settings* y *World Override*, se configuran las reglas del *GameMode*, determinando el comportamiento del jugador y la interacción en el entorno VR. Es una herramienta clave para personalizar la jugabilidad y la lógica del proyecto.

La barra de herramientas situada en la parte superior derecha del Viewport y mostrada en la Figura 3-8 permite controlar la navegación y manipulación de objetos dentro de este. Aquí se encuentran opciones para seleccionar y mover actores, rotarlos, escalarlos y ajustar su alineación en la escena.

También incluye configuraciones para activar o desactivar restricciones como el *snap*, lo que facilita la colocación precisa de los elementos.

En nuestro caso, hemos mantenido los valores predeterminados para la rotación y la escala, lo que garantiza que, al modificar estos parámetros, los objetos roten en incrementos fijos de 10° o se escalen en proporciones definidas. Esta configuración facilita una distribución más ordenada y uniforme dentro del escenario, asegurando coherencia en la disposición y evitando ajustes manuales imprecisos.



Figura 3-8 Barra de herramientas del Viewport

El Panel *Details* de la Figura 3-9 permite modificar las propiedades de los actores seleccionados en la escena, proporcionando un control detallado sobre su comportamiento y apariencia. A través de este panel, es posible mover, rotar y escalar objetos, ajustar su interacción con el entorno, modificar su comportamiento físico y cambiar su aspecto visual. Además, permite configurar propiedades avanzadas como la visibilidad, la iluminación o la interacción con otros elementos dentro del nivel.

Su estructura organizada facilita el acceso a estos parámetros sin necesidad de editar directamente en el *Viewport*, lo que mejora la precisión y eficiencia en el desarrollo.

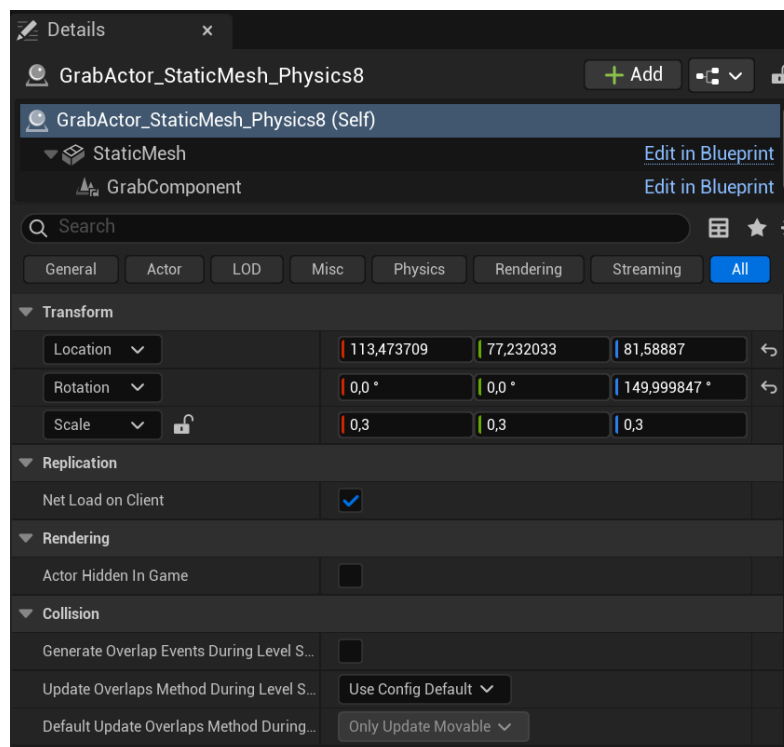


Figura 3-9 Panel Details

Por otro lado, el *Outliner* es una herramienta esencial para la organización de la escena, ya que muestra una lista estructurada de todos los actores presentes en el nivel. Permite buscar, seleccionar y gestionar elementos de manera eficiente, facilitando la navegación dentro del proyecto.

En la Figura 3-10 se pueden ver distintos objetos como estructuras arquitectónicas, fuentes de luz y actores de tipo malla estática. Al seleccionar un actor, se resalta en la lista y en el *Viewport*, lo que facilita su identificación y manipulación dentro del nivel. Mantener un orden es fundamental, ya que a medida que el escenario crece, encontrar y gestionar los elementos sin una organización adecuada puede volverse tedioso y poco eficiente.

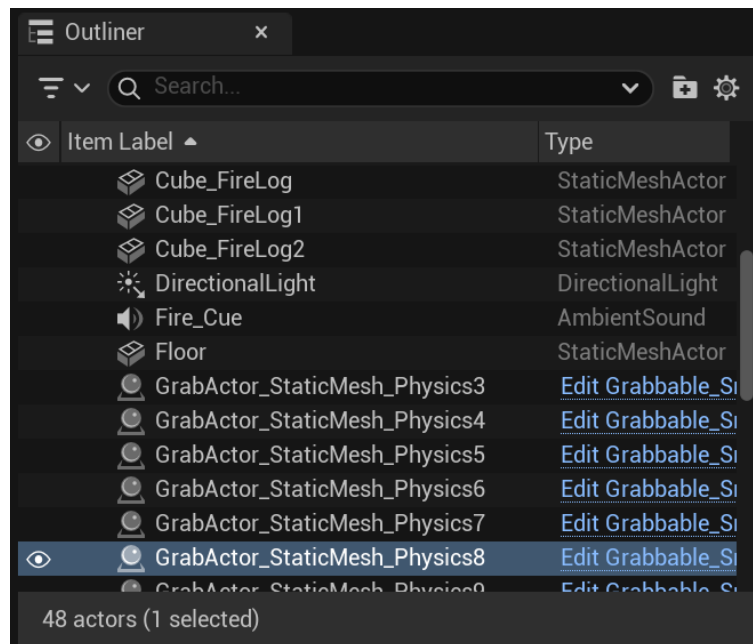


Figura 3-10 Panel Outliner

Por último, el *Content Browser* es la herramienta principal para gestionar los activos del proyecto, permitiendo organizar, buscar e importar recursos como modelos 3D, materiales, texturas, sonidos y Blueprints. En la Figura 3-11 se observa la estructura del contenido dentro del proyecto, con carpetas organizadas por categorías, lo que facilita el acceso rápido a los elementos necesarios para el desarrollo. Además, desde esta interfaz es posible agregar nuevos activos, importar archivos externos y guardar todos los cambios realizados en el proyecto, optimizando la eficiencia en la gestión de recursos.

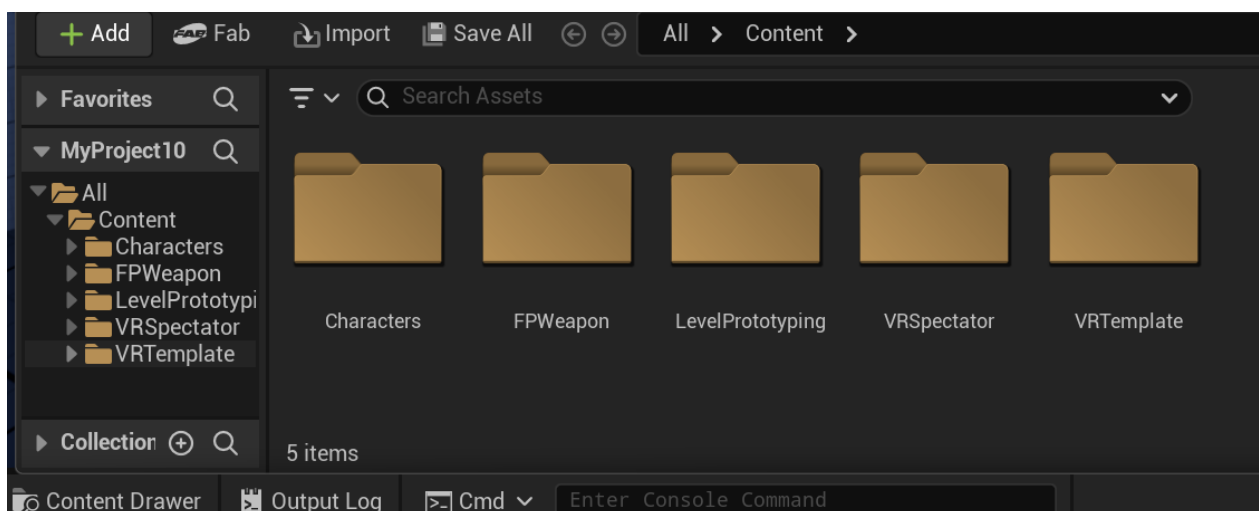


Figura 3-11 Panel del Content Browser

3.2.1 Modelado de entorno base

Para la creación del entorno base, que se muestra en la Figura 3-12, se ha diseñado un habitáculo cuadrado dividido en cuatro secciones diferenciadas por colores, cada una destinada a una funcionalidad específica dentro del escenario virtual. Este modelo se ha desarrollado en Unreal Engine, utilizando geometría simple para optimizar el rendimiento y asegurar una estructura clara y funcional. La segmentación por colores no solo facilita la identificación de cada área, sino que también contribuye a la organización del espacio, permitiendo una mejor distribución de las interacciones y elementos dentro del entorno.

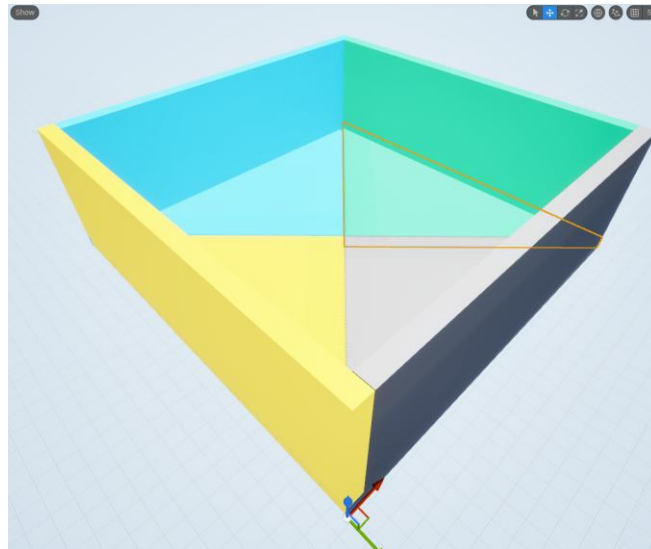


Figura 3-12 Módulo base inicial

Para definir los colores de cada entorno, se ha utilizado el sistema de materiales de Unreal Engine 5. El proceso comienza haciendo clic derecho en el *Content Browser*, seleccionando la opción *Material* y asignándole un nombre. Al hacer doble clic sobre el material creado, accedemos al *Material Graph*, como se muestra en la Figura 3-13. Dado que en este caso solo necesito un color sólido y uniforme, he utilizado un nodo *Constant3Vector*, al cual le asigno un código de color específico en formato RGB. Este valor se conecta directamente a la propiedad *Base Color* del material, consiguiendo así un tono uniforme sin necesidad de texturas o efectos adicionales. De esta manera, cada área del habitáculo cuenta con su propio color distintivo de forma sencilla y eficiente.

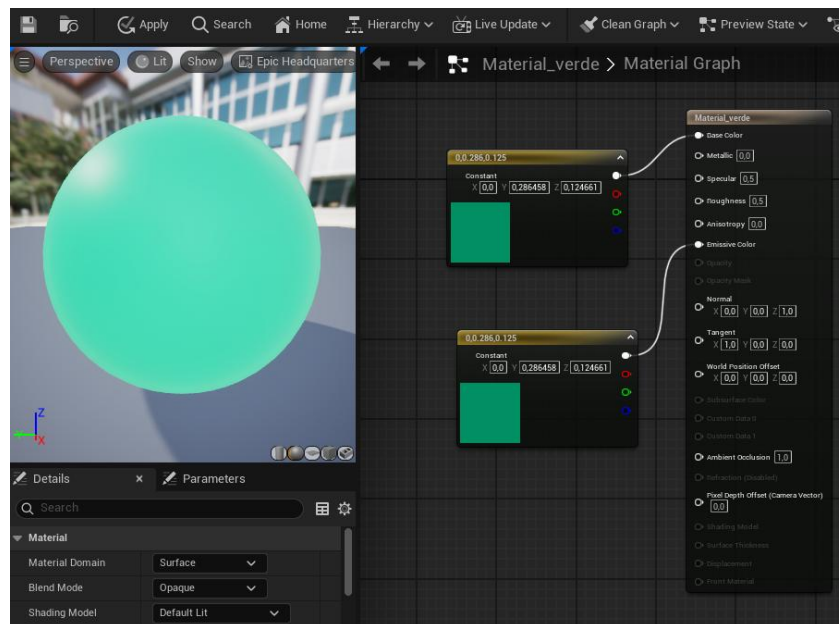


Figura 3-13 Material Graph

Para completar el diseño inicial del entorno, se ha colocado una pirámide en el centro del escenario mostrado en la Figura 3-14, el cual servirá posteriormente como base para visualizar objetos dentro del entorno virtual. Esta estructura permitirá que los elementos que sean llamados para su visualización aparezcan sobre ella, facilitando su observación e interacción.

Además, se ha ubicado el *Player Start* en la sección de color gris, donde se encontrará la pizarra digital. De esta manera, al iniciar la inmersión, el jugador comenzará en un punto estratégico desde el

cual podrá orientarse y dirigirse libremente hacia cualquiera de las secciones del escenario, según la funcionalidad que desee explorar. Esta distribución está diseñada para optimizar la navegación dentro del entorno de realidad virtual.

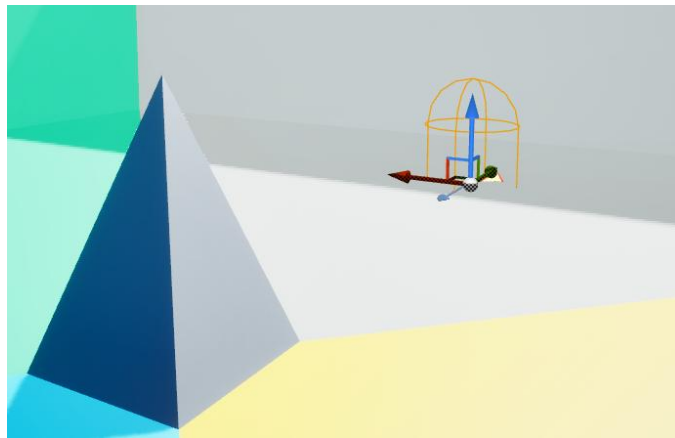


Figura 3-14 Pirámide central y *Player Start*

3.2.2 Entorno de Interacción

La sección de interacción está diseñada para ofrecer una experiencia variada, combinando manipulación libre de objetos y desafíos interactivos. Se divide en dos áreas principales, cada una con diferentes elementos para explorar la física y la interactividad dentro del entorno virtual.

En la primera parte, el usuario podrá manipular objetos, haciéndolas interactuar con otros colisionando entre sí. Además, se ha incorporado un botón interactivo, que al presionarse genera un efecto visual y podrá programarse para diferentes funciones.

La segunda parte incluye un puzzle simple, donde el usuario deberá colocar piezas en su ubicación correcta. Para que una pieza quede anclada, deberá estar en la posición y orientación adecuadas en los tres ejes, añadiendo un componente lógico a la interacción.

3.2.2.1 Sección de Colisión

El desarrollo del entorno de interacción comenzó con la implementación de un objeto físico que sirviera como base para probar las mecánicas de interacción. Como se muestra en la Figura 3-15, se creó un Blueprint Actor denominado *BP_Pelota*, un componente programable que permite definir tanto la apariencia como el comportamiento del objeto.

Se le asignó una *Static Mesh* con forma esférica, garantizando una geometría óptima para la simulación física. Para mejorar su integración visual en el entorno virtual, se aplicó un material específico, otorgándole una estética diferenciada y resaltando su presencia dentro de la escena.

Con el objetivo de dotar a la pelota de un comportamiento realista, se activó la opción *Simulate Physics*, permitiéndole responder a la gravedad y a las fuerzas externas. Además, se configuraron correctamente las colisiones, asegurando que el objeto interactúe con otros elementos del entorno sin atravesarlos ni comportarse de manera errática. Finalmente, se estableció su movilidad como *Movable*, lo que permite su desplazamiento y manipulación dentro del espacio virtual.

Para facilitar la interacción directa del usuario con la pelota, se incorporó un *Grab Component*, un sistema que permite que el objeto pueda ser agarrado, movido y soltado dentro del entorno de realidad virtual. Esto posibilita la manipulación mediante controladores de VR o comandos de interacción, asegurando una experiencia más natural e inmersiva en la simulación.

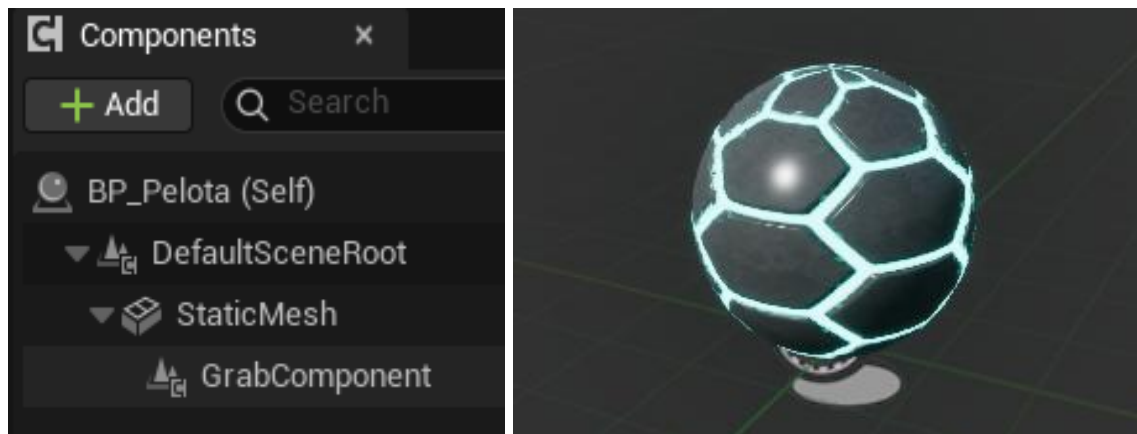


Figura 3-15 Componentes y viewport de BP_Pelota

Para continuar con el desarrollo del entorno de interacción, se añadirán al escenario cubos del tipo *Grabbable_SmallCube* que se muestran en la Figura 3-16, disponibles en la biblioteca de la plantilla de realidad virtual que proporciona Unreal Engine 5. Estos objetos serán integrados con el propósito de evaluar la interacción entre las pelotas ya creadas y estos nuevos elementos dentro del entorno virtual.

Los cubos, al igual que las pelotas, cuentan con un *Grab Component*, lo que significa que podrán ser manipulados directamente por el usuario. Esto permitirá experimentos más avanzados de interacción, ampliando las posibilidades del entorno, y acercándolo cada vez más a una experiencia inmersiva donde los objetos reaccionan de manera realista a las acciones del usuario.

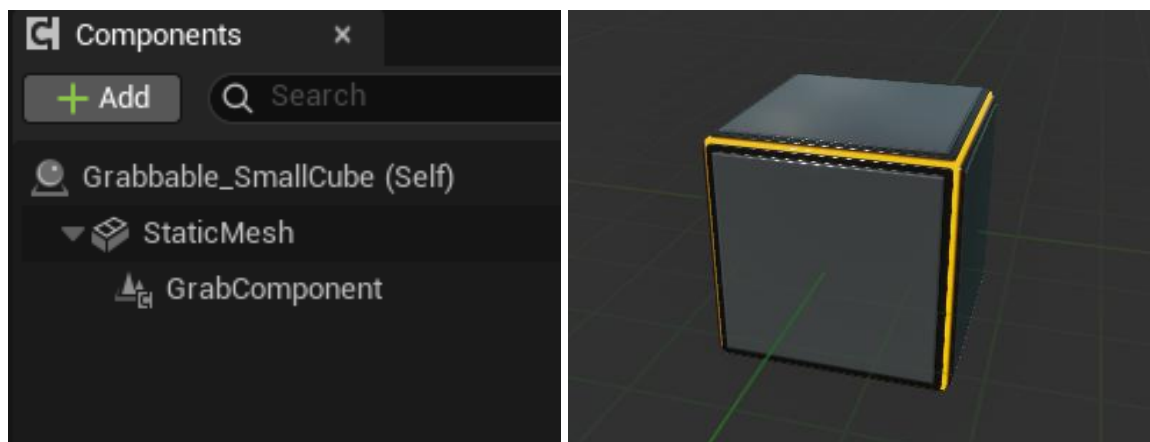


Figura 3-16 Componentes y viewport de Grabbable_SmallCube

Para garantizar que tanto las pelotas como los cubos estén correctamente posicionados al inicio de la interacción, se procederá a la creación de soportes adecuados. La solución más sencilla y eficaz, manteniendo la premisa de un escenario visualmente simple, consiste en la implementación de prismas rectangulares como bases de apoyo. Estos prismas estarán fabricados con el mismo material que el suelo de la zona de interacción, asegurando coherencia estética y evitando distracciones visuales innecesarias.

Para organizar de manera óptima los elementos interactivos, se han dispuesto dos prismas rectangulares. Como se muestra en la Figura 3-17, el primero sirve como base para las pelotas, asegurando que se mantengan accesibles desde una altura cómoda para el usuario, facilitando su manipulación y lanzamiento sin necesidad de realizar movimientos incómodos. El segundo prisma actúa como soporte para los cubos, los cuales han sido organizados en una disposición piramidal. Esta configuración ha sido pensada para que el usuario experimente con la interacción física del entorno, intentando derribar la pirámide lanzando las pelotas contra los cubos.

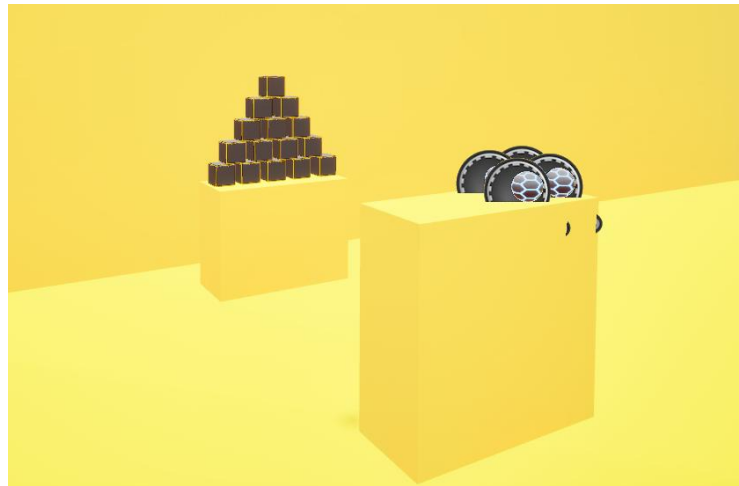


Figura 3-17 Prismas de Soporte de pelotas y cubos

3.2.2.2 Diseño Botón

Durante la etapa de diseño, se concibió una idea versátil para mejorar la interactividad del escenario. Se trata de la implementación de un botón interactivo, un elemento reutilizable que puede integrarse en distintas partes del entorno y adaptarse a diversas funcionalidades.

El objeto, representado en la imagen, consiste en un botón de diseño simple y llamativo, que al ser presionado puede activar diferentes acciones dentro del entorno virtual. En el contexto de este escenario, su función podría ser hacer aparecer nuevas pelotas cuando el usuario lo pulse. De esta manera, si en algún momento se agotan las pelotas disponibles, el usuario podrá generar más y continuar con la interacción sin interrupciones.

Para la creación del botón interactivo, se crea un Blueprint Actor denominado *BP_Boton_Pelota*. En su estructura, se añadieron dos *Static Mesh* que conforman su diseño. La primera es la base del botón, de color blanco, que sirve como soporte y proporciona estabilidad visual y estructural al objeto. La segunda corresponde al botón en sí, de color rojo, que será el elemento interactivo principal.

Además, al botón se le incorpora un *Box Collision*, que se puede apreciar en la Figura 3-18, encargado de detectar la interacción del usuario con el botón, mediante la detección del *Box Collision* que se colocará en las manos del usuario en realidad virtual. De esta manera, cuando ambos colisionen, se podrá programar la lógica del botón en función del impacto, permitiendo que se activen distintas funcionalidades dentro del entorno.

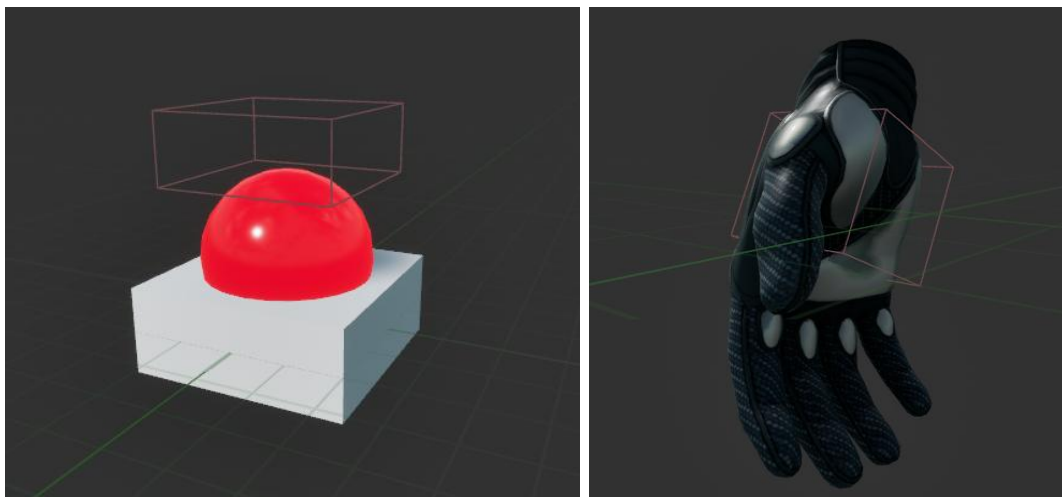


Figura 3-18 Viewport de *BP_Boton_Pelota* y *Hand VR*

Para programar el comportamiento del botón, cuyo bloque de código se refleja en el Anexo III: Bloque de Código para Botón de Pelotas, nos dirigimos al *Event Graph* de *BP_Boton_Pelota*. La programación, mostrada en la Figura 3-19, comienza con el evento *On Component Begin Overlap (Box)*, que detecta cuando un objeto entra en el área de colisión del botón. Luego, se usa *Cast To VRPawn* para comprobar si el objeto detectado es la mano del usuario en VR. Si la verificación es correcta, se activa la acción programada.

Para simular la presión del botón, se usa *Move Component To*, que desplaza la parte roja del botón hasta la posición deseada. Esto crea el efecto visual de que el botón se hunde al ser presionado. El movimiento se suaviza con un tiempo de transición de 0.1 segundos, y la opción *Force Shortest Rotation Path* asegura que el botón se mueva de la forma más eficiente. Así, se consigue una interacción intuitiva y realista dentro del entorno virtual.

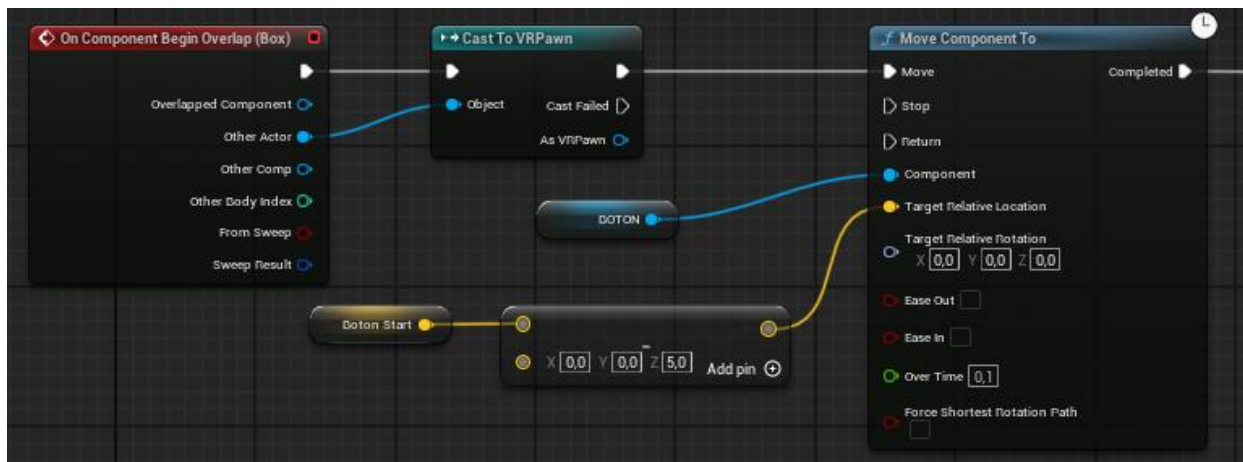


Figura 3-19 Código para bajar botón

Para que el botón vuelva a su posición original, aplicaremos el código de la Figura 3-20, que hace que el botón vuelva a su posición original cuando la mano del usuario deja de tocarlo.

El evento *On Component End Overlap (Box)* detecta cuando el objeto que estaba en contacto con el botón se retira. Luego, se usa *Cast To VRPawn* para comprobar si el objeto que se alejó era la mano del usuario en VR. Si la verificación es correcta, se activa la acción programada.

Se utiliza *Move Component To* para devolver la parte roja del botón a su posición inicial, usando *Boton Start* como referencia. El movimiento se suaviza con un tiempo de transición de 0.1 segundos, asegurando un efecto visual natural. Así, el botón funciona como en el mundo real, hundiéndose al presionarlo y volviendo a su lugar cuando se deja de tocar.

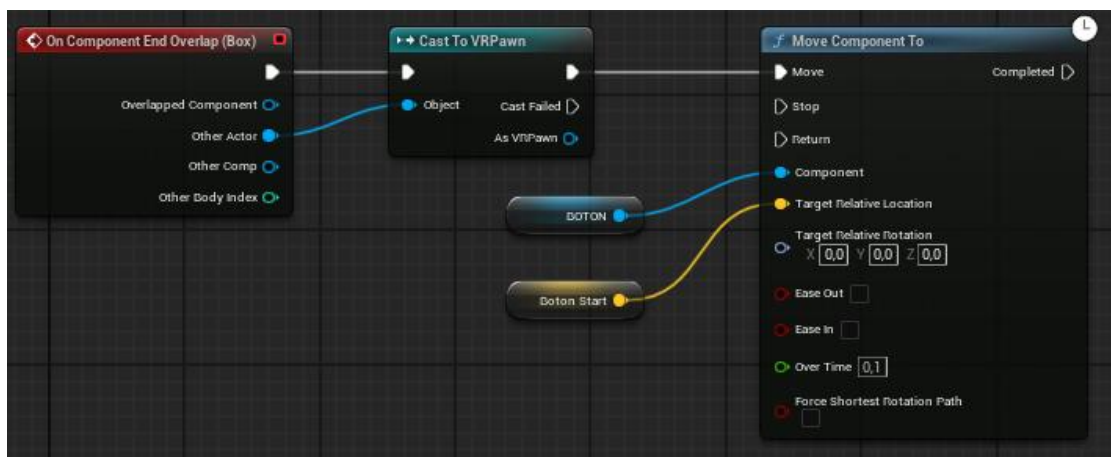


Figura 3-20 Código para subir botón

Para lograr que aparezca la pelota al pulsar el botón, conectaremos el código de la Figura 3-21 al nodo *Completed* que mueve el botón hacia abajo.

Se comienza creando una variable llamada *ultimapelota*, que almacena la última pelota generada. Antes de crear una nueva, se verifica si esta variable contiene un objeto válido mediante el nodo *Is Valid*. Si nunca se ha presionado el botón, la variable estará vacía y la lógica pasará directamente a la creación de la primera pelota.

Si la variable Última Pelota ya tiene un objeto asignado, se activa un *Branch* que dirige la ejecución por la salida *True*, lo que activa el nodo *Destroy Actor*. Este paso es crucial para evitar que el escenario se llene de pelotas innecesarias. Después de eliminar la pelota anterior, se ejecuta el nodo *SpawnActor BP Pelota*, que genera una nueva pelota en la escena y la guarda en la variable *Última Pelota*.

Para garantizar que la nueva pelota no aparezca encima del botón o en una posición incómoda, se usa *Make Transform*, que define su ubicación con un pequeño desplazamiento en el eje Y.

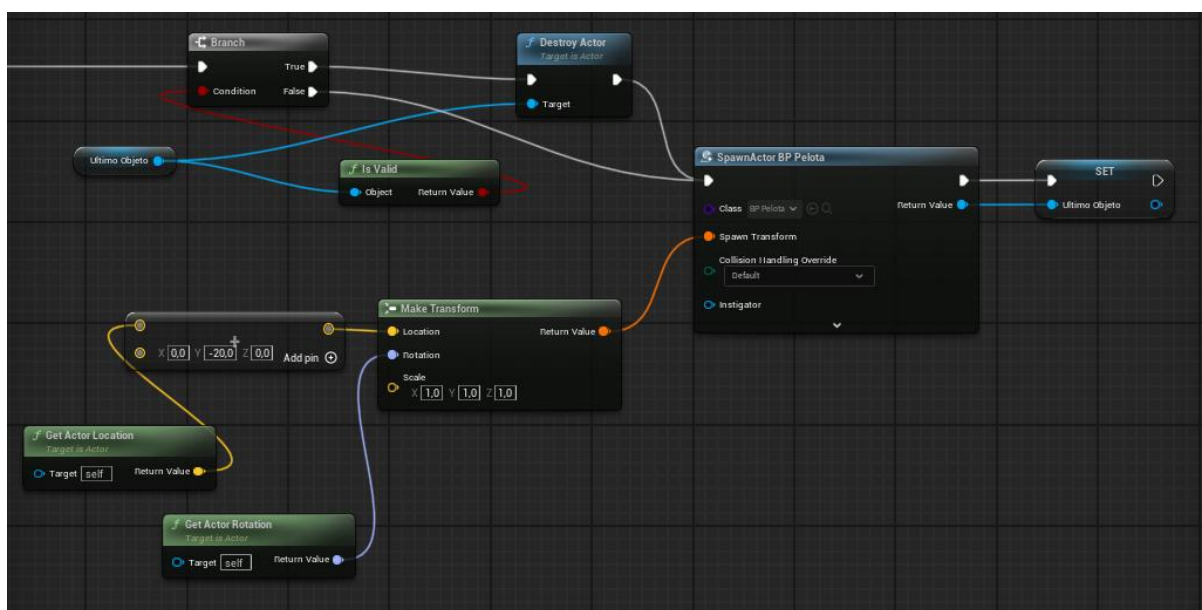


Figura 3-21 Código para instanciar pelota

Finalmente, se incorpora el *BP_Boton_Pelota* en el escenario, como se muestra en la Figura 3-22.

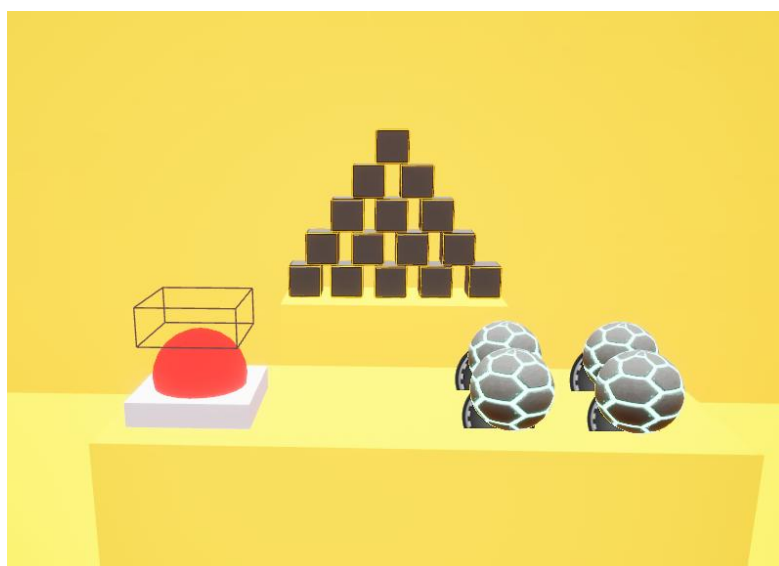


Figura 3-22 Sección de colisión

3.2.2.3 Sección de Puzzle

Con el objetivo de mejorar la interacción del usuario dentro del entorno de realidad virtual, se ha desarrollado un sistema de ensamblaje basado en un puzzle de cuatro piezas geométricas simples. Esta mecánica permite al usuario familiarizarse con la manipulación de objetos en un espacio tridimensional, fomentando la precisión en la colocación y orientación de elementos dentro del entorno virtual.

El funcionamiento del puzzle se basa en la detección de la posición y orientación de cada pieza en los tres ejes espaciales. Si la pieza es colocada correctamente en su posición predeterminada y con la orientación adecuada, quedará anclada de forma automática. En caso contrario, la pieza permanecerá libre, permitiendo que el usuario ajuste su colocación hasta lograr la alineación correcta.

Para el diseño del puzzle, se decidió utilizar el logotipo del Centro Universitario de la Defensa (CUD) como imagen de referencia. Para ello, en primer lugar, se descargó el logo y se dividió en cuatro partes iguales, asegurando que cada segmento correspondiera a una pieza individual del puzzle.

A continuación, se crearon cuatro Widget Blueprints, uno para cada pieza del puzzle. En cada widget se insertó la parte correspondiente de la imagen, de manera que, al ensamblar todas las piezas en su posición correcta, se reconstituya la imagen completa del logotipo.

Para representar físicamente cada pieza dentro del entorno de realidad virtual, se generó un Blueprint Actor como el que aparece en la Figura 3-23 por cada una de las piezas. Dentro de cada Blueprint, se añadió una *Static Mesh* rectangular que serviría como base para la pieza del puzzle. Posteriormente, se insertó el widget con la imagen en la cara superior de la malla, asegurando un ajuste preciso en su posición y tamaño para que coincidiera con el diseño original.

Finalmente, se incorporó un *Grab Component*, permitiendo que el usuario pueda manipularlas libremente dentro del entorno virtual. De esta manera, el sistema facilita la interacción y el ensamblaje del puzzle, manteniendo una mecánica intuitiva y accesible para el usuario.

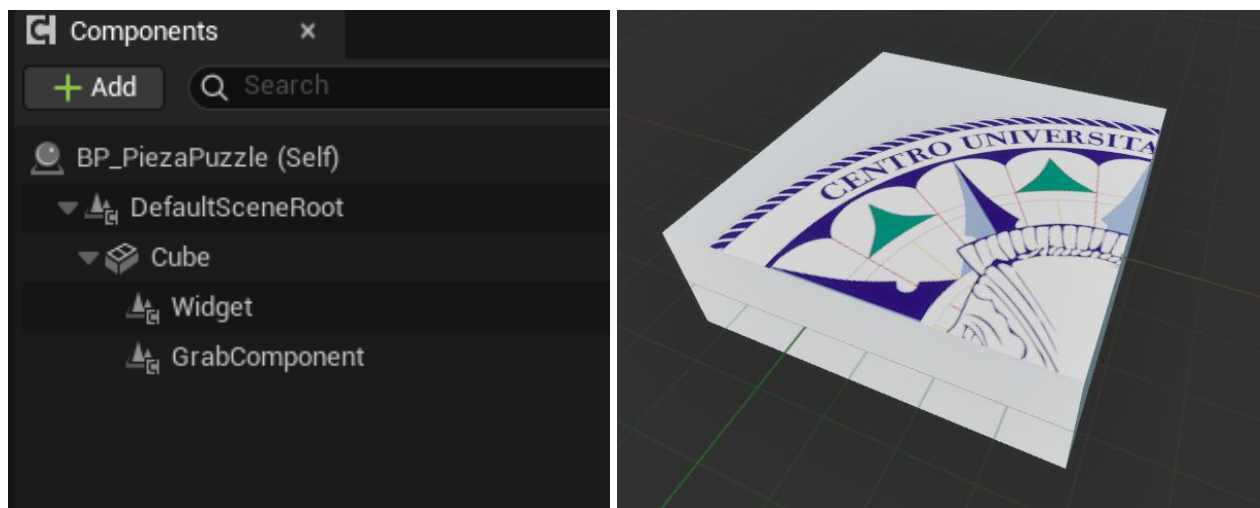


Figura 3-23 Componentes y viewport de BP_PiezaPuzzle

Una vez definidas las cuatro piezas del puzzle, es necesario establecer los puntos de anclaje donde cada una deberá ser colocada correctamente. Para ello, se crea un Blueprint Actor por cada pieza como el mostrado en la Figura 3-24.

Cada punto de anclaje está compuesto por un plano estático al que se le ha asignado un color similar al del escenario, de modo que se integre visualmente con el entorno y no distraiga al usuario. Además, se ha incorporado un *Box Collision*, el cual será el encargado de detectar la presencia de la pieza correcta y validar su posición y orientación en los tres ejes espaciales.

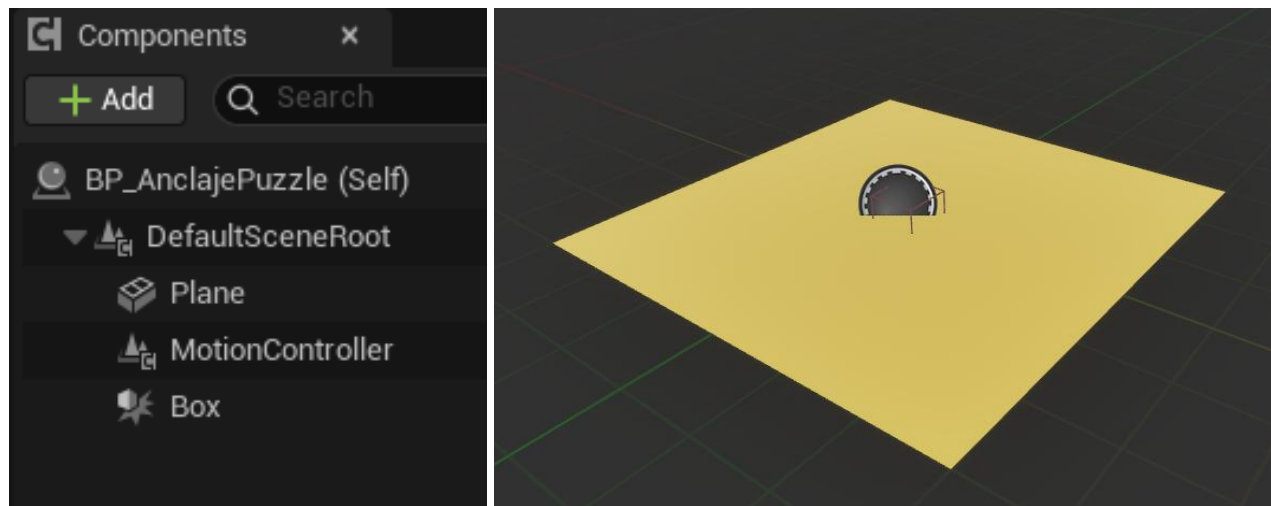


Figura 3-24 Componentes y viewport de BP_AnclajePuzzle

Para programar el anclaje, cuyo bloque de código al completo se refleja en el Anexo IV: Bloque de Código para Anclaje de Pieza, accedemos al Event Graph del BP_AnclajePuzzle. Como se muestra en la Figura 3-25, la programación comienza con el evento *On Component Begin Overlap (Box)* que se activa cuando un objeto entra en el área de colisión del *Box Collision* asociado al punto de anclaje. En este punto, se obtiene el objeto que ha generado la colisión con el fin de verificar si se trata de una pieza válida del puzle.

Para ello, se realiza un *Cast To BP_PiezaPuzzle*, comprobando si el objeto detectado corresponde efectivamente a una pieza del puzle. Si la conversión es exitosa, el sistema procede a configurar el tipo de agarre mediante el nodo *SET - Grab Type*, estableciéndolo en modo Snap. Esto permite que la pieza se ajuste automáticamente en su posición predeterminada dentro del punto de anclaje, asegurando una colocación precisa.

Seguidamente, se ejecuta el nodo *Try Grab*, el cual intenta realizar un agarre automático de la pieza mediante su *Grab Component*, utilizando el *Motion Controller* del usuario para asignarle la acción de agarre. Una vez que la pieza ha sido correctamente colocada en el anclaje, el tipo de agarre se restablece a *Free*, lo que significa que la pieza queda fija en su sitio y el usuario podrá volver a manipularla, evitando desplazamientos involuntarios o reajustes erróneos.

Para completar la lógica, el sistema obtiene una referencia del *Player Pawn* mediante *Get Player Pawn*, lo que permite acceder a las propiedades del personaje en VR. A continuación, se realiza un *Cast To VRPawn* para comprobar si el objeto pertenece al jugador y garantizar que la lógica se aplique correctamente.

Si el Cast es exitoso, se establecen a nulo los valores de *Held Component Left* y *Held Component Right*, asegurando que la pieza se suelte de cualquiera de las dos manos, quedando fijada en el punto de anclaje.

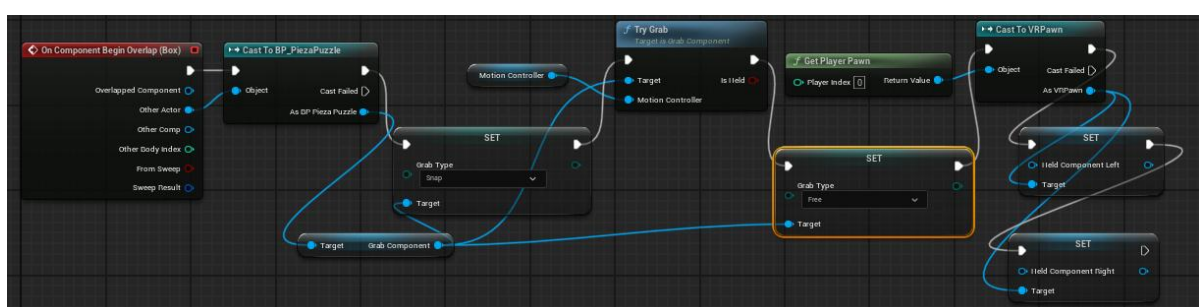


Figura 3-25 Código para anclaje de piezas

Para garantizar que la pieza del puzzle solo se ancle si se aproxima al punto de anclaje con la orientación correcta en los tres ejes, se ha implementado un sistema de validación de rotación antes de activar la lógica de anclaje, reflejado en la Figura 3-26.

Cuando la pieza entra en contacto con el área de colisión del punto de anclaje, se ejecuta un *Cast To BP_PiezaPuzzle* para verificar que el objeto detectado es una pieza del puzzle. A continuación, se introduce un *Branch* que evalúa si la orientación de la pieza coincide con la del anclaje dentro de un margen de tolerancia. Para ello, se obtiene la rotación de ambos elementos mediante el nodo *Get World Rotation*.

Las rotaciones se convierten a cuaterniones, una forma matemática de representar rotaciones en el espacio tridimensional. Los cuaterniones son preferidos en motores gráficos como Unreal Engine porque evitan problemas como el ‘*gimbal lock*’ [25], una limitación de los ángulos de Euler que puede generar bloqueos en ciertos ejes de rotación. Ocurre cuando dos ejes se alinean, provocando la pérdida de un grado de libertad y dificultando representar ciertos movimientos tridimensionales con precisión. Este problema puede afectar la estabilidad y el control en simulaciones. Por ello, los cuaterniones permiten interpolaciones más suaves y cálculos más eficientes en comparación con otros sistemas de rotación.

Una vez obtenidos los cuaterniones de ambas rotaciones, se calcula la distancia angular entre ellos. Este valor, expresado en radianes, se convierte a grados para facilitar su comparación. Mediante el nodo *In Range (Float)*, se verifica si la diferencia de orientación está dentro del rango permitido, estableciendo un margen de 0° a 15°. Si la condición se cumple, el *Branch* continúa por la salida True, activando la lógica previa de anclaje de la pieza. En caso contrario, la pieza permanecerá libre hasta que el usuario la oriente correctamente.

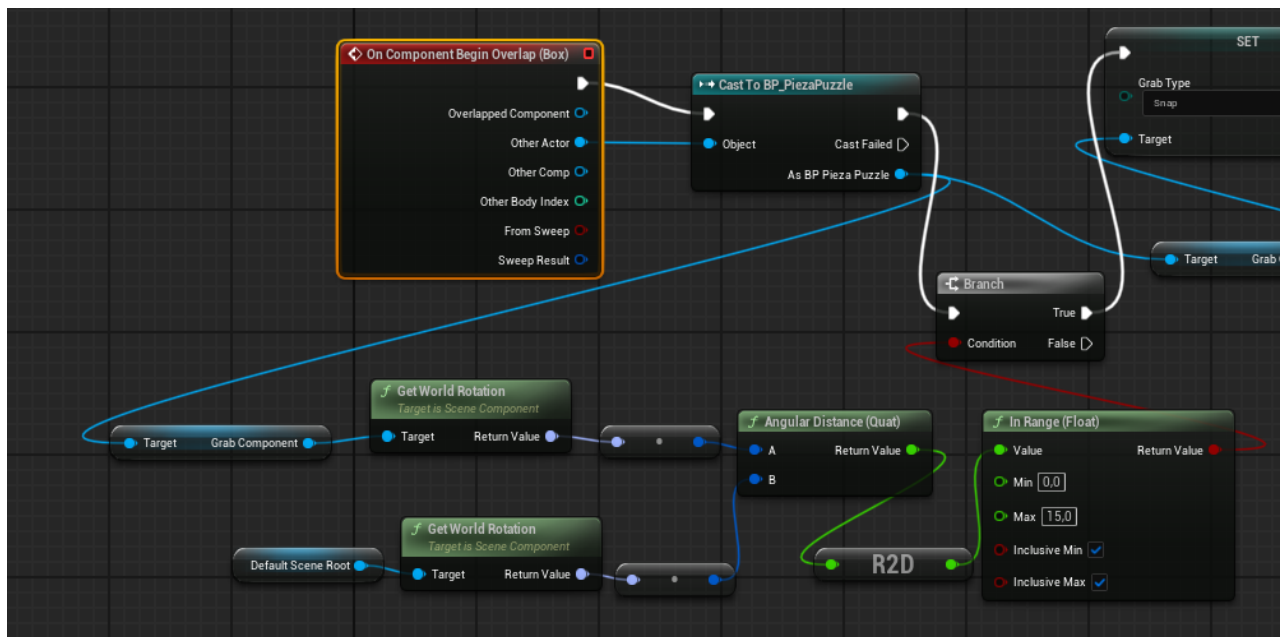


Figura 3-26 Código para comprobar la orientación de las piezas

Por último, se ha creado un prisma de soporte similar al utilizado en la sección de colisiones, sobre el cual se han colocado las cuatro piezas del puzzle. Este soporte permite que las piezas se encuentren organizadas y accesibles para el usuario, facilitando su manipulación dentro del entorno virtual.

Además, se han colocado los cuatro puntos de anclaje en la pared situada frente a las piezas, asegurando que se encuentren en una posición adecuada y cómoda para la interacción en realidad virtual. Cada punto de anclaje ha sido ubicado en su posición correspondiente, de manera que, una vez que el usuario encaje correctamente todas las piezas en sus respectivos lugares, se reconstruya la imagen completa del logo del CUD.

En la Figura 3-27 se muestra el resultado final del área del puzle.



Figura 3-27 Sección de Puzle

3.2.3 Entorno de Retroalimentación Háptica

Este apartado aborda el diseño del escenario de integración de dispositivos hápticos dentro del entorno de realidad virtual. Para ello, se han definido dos zonas específicas.

La primera está destinada a la simulación de disparo con retroalimentación háptica, donde el usuario podrá experimentar el retroceso de un arma virtual sincronizada con un dispositivo háptico. Se ha diseñado un espacio con una diana para garantizar una interacción fluida y realista.

La segunda zona está enfocada en la recepción de estímulos a través de un chaleco háptico, simulando impactos de proyectiles u otros efectos. Aquí, el usuario podrá experimentar distintas respuestas táctiles en función de los eventos del entorno.

3.2.3.1 Sección de Pistola

Para el desarrollo de la sección de pistolas dentro del entorno de retroalimentación háptica, lo primero fue la creación de un prisma de soporte similar a los utilizados en el entorno de interacción. Sin embargo, en esta ocasión, se ha adaptado su color al del nuevo entorno para mantener coherencia visual y facilitar la diferenciación entre las distintas áreas del escenario.

En esta sección se han utilizado las pistolas incluidas en la biblioteca de la plantilla de realidad virtual de Unreal Engine, ya que el objetivo principal de este espacio no es el desarrollo de un modelo de arma propio, sino la integración y prueba del efecto háptico asociado al disparo. Al emplear estos modelos predefinidos, se optimiza el proceso de implementación y se garantiza una compatibilidad inmediata con el sistema de interacción VR.

Para complementar la sección de pistolas y proporcionar un objetivo visual al usuario, se desarrolló una diana dentro del entorno de realidad virtual. La creación de este elemento se llevó a cabo mediante la creación de un Widget Blueprint en el que se insertó una imagen de una diana de tiro. Este widget serviría como interfaz visual para proyectar la imagen en el entorno 3D.

Posteriormente, se generó un Blueprint Actor denominado *BP_Diana*, mostrado en la Figura 3-28. A este actor se le añadió un plano estático, que serviría como superficie base sobre la que se mostraría la imagen de la diana. Finalmente, el widget con la imagen de la diana se colocó sobre la superficie del plano y se ajustaron sus dimensiones para asegurar una representación clara y proporcionada.

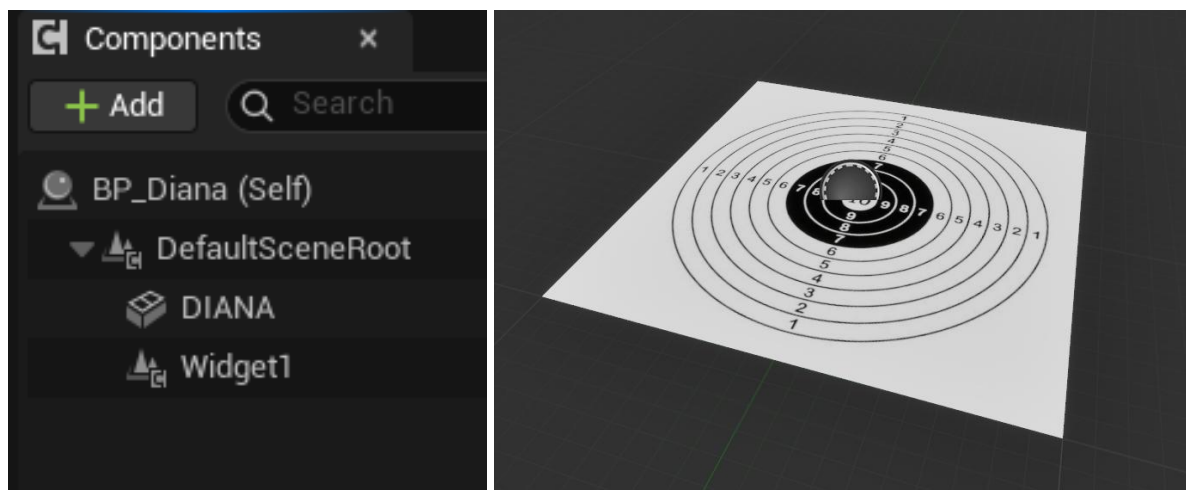


Figura 3-28 Componentes y viewport de BP_Diana

Finalmente, se incorporaron al escenario los elementos necesarios para la práctica de tiro con retroalimentación háptica, como aparecen en la Figura 3-29. Se posicionó el prisma de soporte, que sirve como base para las pistolas, asegurando su accesibilidad y alineación con la zona de interacción. Sobre este soporte, se colocaron las pistolas de la plantilla VR, listas para ser utilizadas por el usuario en la simulación de disparo.

Asimismo, la diana se fijó en la pared frente al soporte, en una ubicación óptima para la práctica de tiro. Su colocación garantiza una referencia visual clara para el usuario, facilitando la evaluación de precisión y mejorando la experiencia inmersiva dentro del entorno virtual.

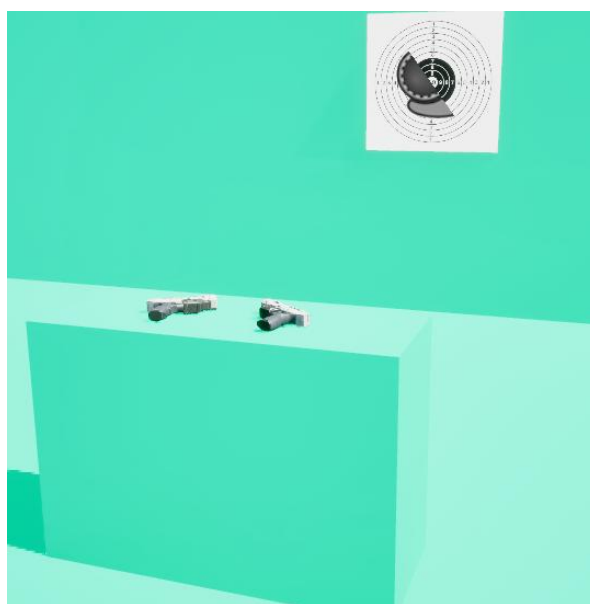


Figura 3-29 Sección de Pistola

3.2.3.2 Sección de Chaleco Háptico

Para la implementación de los efectos hápticos en el entorno de realidad virtual se utilizará el *TacSuit* de bHaptics [23], un chaleco háptico que proporcionará retroalimentación táctil en función de los eventos simulados. Este dispositivo permitirá que el usuario experimente sensaciones físicas como impactos o vibraciones, aumentando la inmersión y realismo de la simulación.

Esta sección se compone de cuatro soportes, cada uno equipado con un botón interactivo idéntico a los de la sección de interacción. Al presionar cada uno de ellos, se activarán diferentes efectos en el

chaleco, permitiendo al usuario experimentar diversas respuestas táctiles. Para facilitar la identificación, cada soporte incluirá un cartel señalizador indicando el tipo de efecto que se activará. Los carteles consisten en un Blueprint actor como el de la Figura 3-30, compuestos por un widget con el texto para cada efecto.

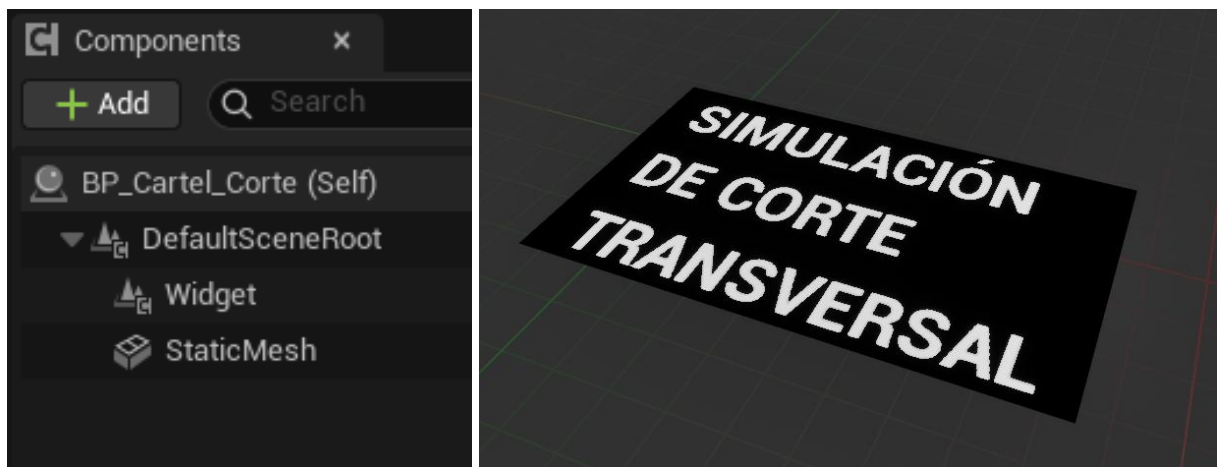


Figura 3-30 Componentes y viewport de BP_Cartel

Para los botones de la sección de chaleco háptico se reutilizará el mismo sistema implementado en el escenario de interacción, manteniendo su estructura y comportamiento básico. Sin embargo, se eliminará el código responsable de la generación de la pelota, de la Figura 3-31, ya que en este caso los botones no estarán vinculados a la aparición de objetos en el entorno.

En su lugar, cada botón será conectado posteriormente al efecto háptico correspondiente, permitiendo que, al ser presionado, se active la respuesta específica.

Se incorporan al escenario cuatro soportes del mismo color que el entorno, cada uno equipado con un botón interactivo en la parte superior y un cartel indicador que señala el efecto háptico correspondiente. Estos elementos están distribuidos de manera organizada para facilitar la interacción del usuario, permitiendo la activación de distintas simulaciones táctiles. La disposición final de los soportes, botones y carteles puede apreciarse en la Figura 3-31.

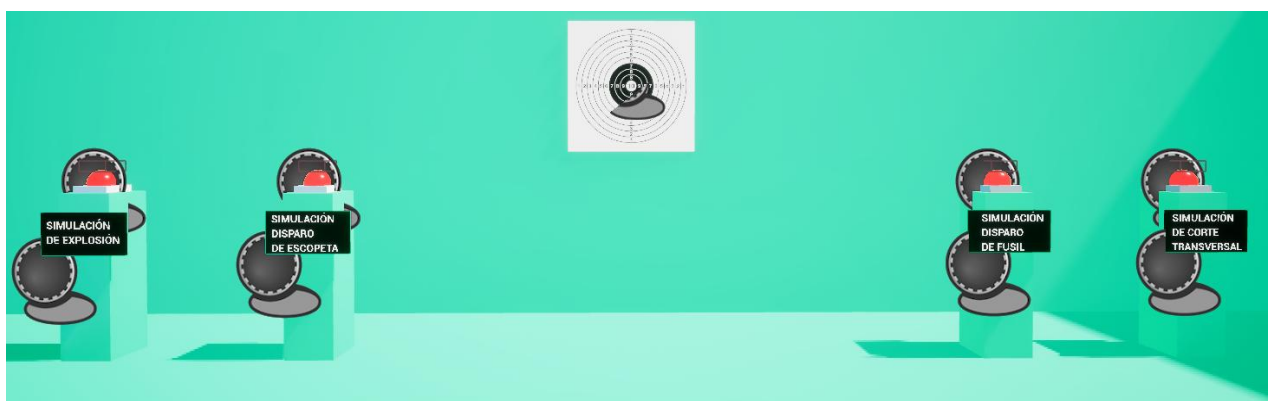


Figura 3-31 Sección de chaleco háptico

3.2.4 Pizarra Digital

Para mejorar la interactividad del entorno virtual, se desarrollará una pizarra digital que imite la utilizada en el CUD. La intención es que pueda moverse libremente dentro del entorno de realidad virtual, permitiendo al usuario posicionarla según sus necesidades. Su integración facilitará la

visualización y manipulación de información de forma interactiva, mejorando la experiencia educativa y de simulación.

El primer paso para la creación de la pantalla interactiva será el desarrollo de un Widget Blueprint, como el mostrado en la Figura 3-32. En la parte superior del widget se insertará una imagen, que representará el contenido visual de la pizarra. En la parte inferior, se añadirán dos botones, organizando su disposición para que el diseño resulte vistoso y realista dentro del entorno de realidad virtual.

Para mejorar la experiencia de usuario y facilitar la navegación, se asignará al botón izquierdo una imagen de una flecha apuntando a la izquierda y al botón derecho su respectiva flecha apuntando a la derecha. Estos elementos permitirán la interacción con la pantalla, según la funcionalidad que se implemente posteriormente.

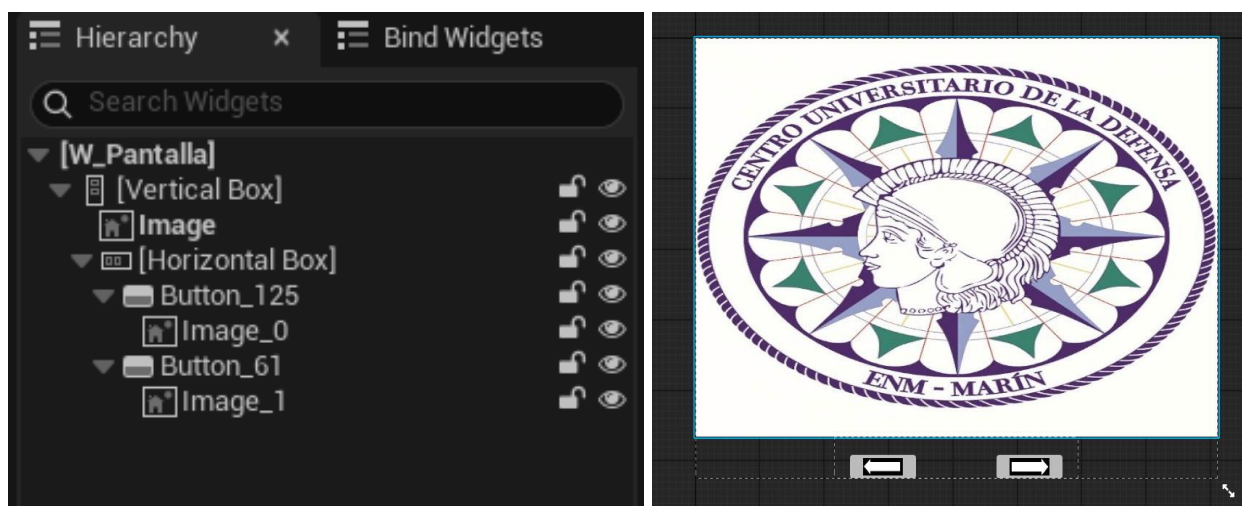


Figura 3-32 Jerarquía y viewport de W_Pantalla

Para programar el comportamiento de la pizarra digital, se implementará el sistema mostrado en la Figura 3-33. Inicialmente, la interacción se realizará mediante las teclas de flecha izquierda y derecha del teclado para cambiar las imágenes de la pantalla. Posteriormente, este control se vinculará a los botones interactivos, pero se comenzará con esta configuración para asegurar el correcto funcionamiento del código antes de integrarlo completamente en el entorno de realidad virtual.

El primer paso consiste en la creación de una variable llamada *textura*, de tipo *array*, en la que se almacenarán cinco imágenes diferentes. Estas imágenes serán las que se mostrarán en la pantalla de la pizarra cuando el usuario interactúe con los controles. Además, se creará una variable entera llamada *indice*, que permitirá llevar un seguimiento de la imagen que se está mostrando en cada momento.

El funcionamiento del sistema se basa en la modificación de la variable *indice* al presionar las teclas de flecha. Cuando se presiona la flecha derecha, el valor de *indice* se incrementa en 1, mientras que, al presionar la flecha izquierda, se reduce en 1. De este modo, el cambio en el valor de *indice* actualizará automáticamente la imagen mostrada en la pantalla de la pizarra.

Con el fin de evitar errores en la ejecución del código, se establecerá un límite en la variable *indice*. Se utilizará la función *Clamp (Integer)*, que restringirá su valor dentro del rango válido, desde 0 hasta la longitud total del array *textura*. Esto asegurará que el valor de *indice* nunca salga de los límites del array, evitando errores que podrían producirse si se intentara acceder a una posición inexistente en la lista de imágenes.

Finalmente, cada vez que se modifique el valor de *indice*, se ejecutará un *Set Brush* en la imagen objetivo, lo que permitirá actualizar la pantalla de la pizarra con la imagen correspondiente al nuevo

valor de la variable. Esto garantiza que el contenido visual de la pizarra cambie dinámicamente en respuesta a la interacción del usuario.

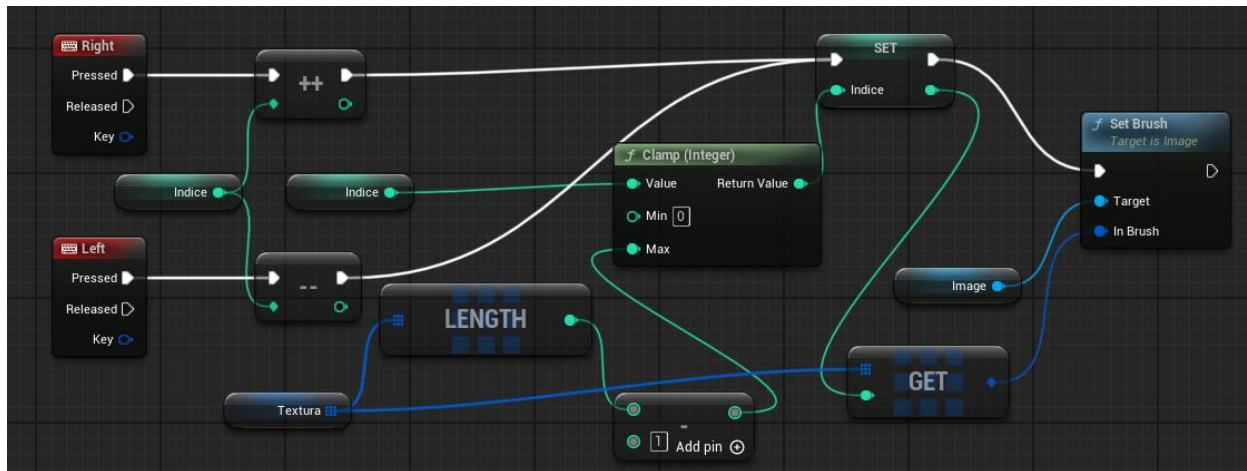


Figura 3-33 Código para cambiar de imagen con el teclado

Para dar una tercera dimensión al widget creado y permitir su integración en el entorno de realidad virtual, se desarrolló un Actor Blueprint denominado *BP_Pizarra*, como el que aparece en la Figura 3-34. Dentro de este actor, se insertó un plano sobre el cual se aplicó el widget previamente diseñado, ajustando sus dimensiones para garantizar una correcta alineación y visualización del contenido.

Se incorporó un marco alrededor de la pantalla para proporcionar mayor realismo y volumen a la pizarra, asegurando una apariencia más sólida y realista dentro del entorno virtual. Este marco deja libres los huecos de los botones, los cuales se diseñan conforme a las dimensiones adecuadas que se incorporarán a la pizarra.

Finalmente, se añadió un *Grab Component*, lo que permitirá al usuario manipular y mover la pizarra libremente dentro del escenario de realidad virtual.

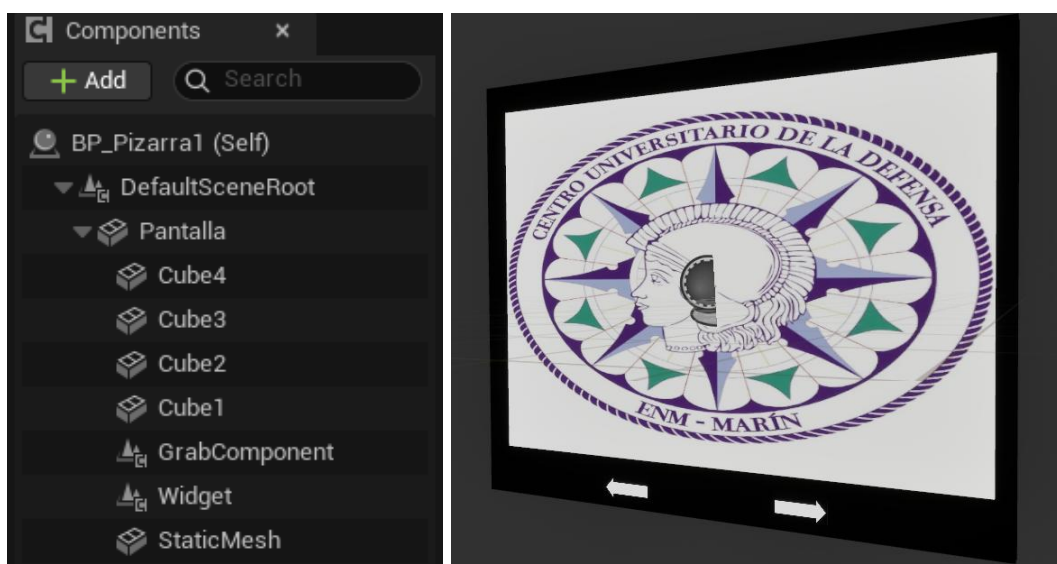


Figura 3-34 Componentes y viewport de *BP_Pizarra*

Para permitir el cambio de imagen al tocar los botones con el dedo en VR, se crearon dos funciones dentro del Widget, replicando la lógica previamente implementada con las teclas del teclado.

La primera función, *ActivarBoton1*, mostrada en la Figura 3-35, incrementa el valor de la variable *Indice*, permitiendo avanzar en la lista de imágenes almacenadas en el *array textura*. La segunda función,

ActivarBoton2, tiene la misma lógica, pero en este caso decreuenta el valor de *indice*, permitiendo retroceder en la secuencia de imágenes.

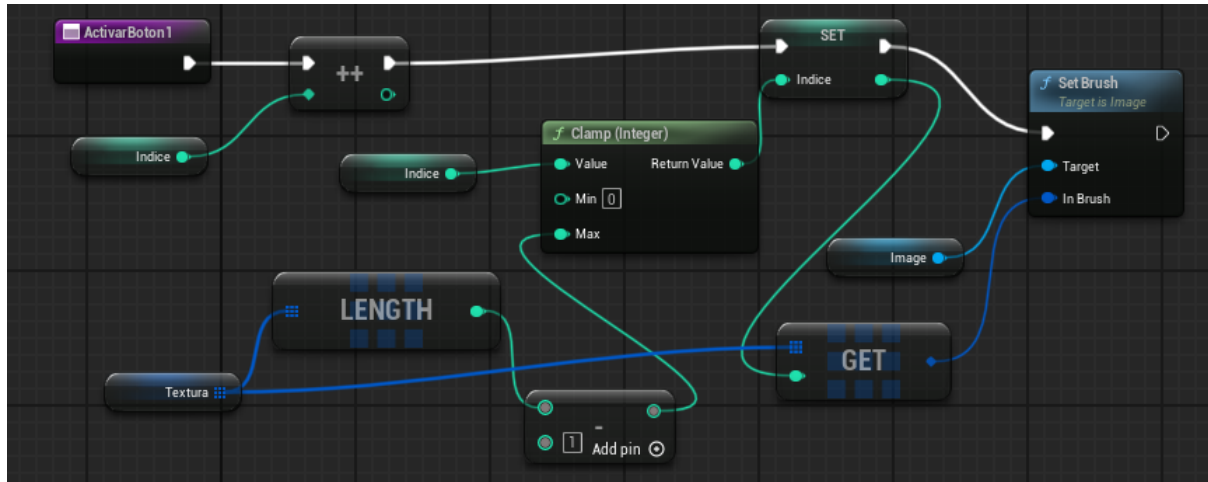


Figura 3-35 Código de Función *ActivarBoton1*

Para permitir la interacción con la pizarra digital en realidad virtual, se añadieron varios elementos clave. Primero, se colocó una *Sphere Collision* en el dedo índice de ambas manos de RV, lo que servirá como punto de contacto con los botones del widget. Esta esfera, mostrada en la Figura 3-36, detectará cuándo el dedo entra en contacto con los elementos interactivos de la pizarra.

Además, se integró la función *Widget Interaction*, que permite la interacción con widgets en 3D dentro del entorno virtual. Esta función asegurará que los botones del widget respondan correctamente cuando el usuario los toque con el dedo en VR.

Por último, dentro del Blueprint *BP_Pizarra*, se añadió un *Box Collision* ubicado estratégicamente sobre los dos botones interactivos de la pantalla. Este *Box Collision* servirá para detectar cuándo el dedo del usuario entra en la zona de interacción y activar la función correspondiente dentro del widget, permitiendo que el cambio de imagen se realice de manera fluida y natural.



Figura 3-36 *Sphere Collision* de *HandVR*

Para programar la interacción con los botones, cuyo bloque de código al completo se refleja en el Anexo V: Bloque de Código para Interacción con Botones de la Pizarra, nos dirigimos al *Event Graph* de BP_Pizarra y establecemos como evento de inicio el *On Component Begin Overlap* de la *Box Collision* asociada al botón derecho, como aparece en la Figura 3-37. A continuación, realizamos un *Cast* a *VRPawn* para comprobar que el objeto que ha activado la colisión es el peón del jugador en VR. Si esta condición se cumple, procedemos a verificar si existe interacción con el widget de la pizarra. Para ello, utilizamos la función *Get Widget*, lo que nos permitirá acceder a los elementos interactivos del widget.

Si la interacción es válida, llamamos a la función *ActivarBoton1* en el caso del botón derecho, lo que permitirá avanzar en la secuencia de imágenes de la pizarra. Para el botón izquierdo, seguimos exactamente el mismo procedimiento, pero en este caso llamamos a la función *ActivarBoton2*, lo que permitirá retroceder en la secuencia de imágenes. Con esta implementación, se completa el sistema de interacción de la pizarra digital, permitiendo cambiar las imágenes simplemente tocando los botones con el dedo dentro del entorno de realidad virtual.

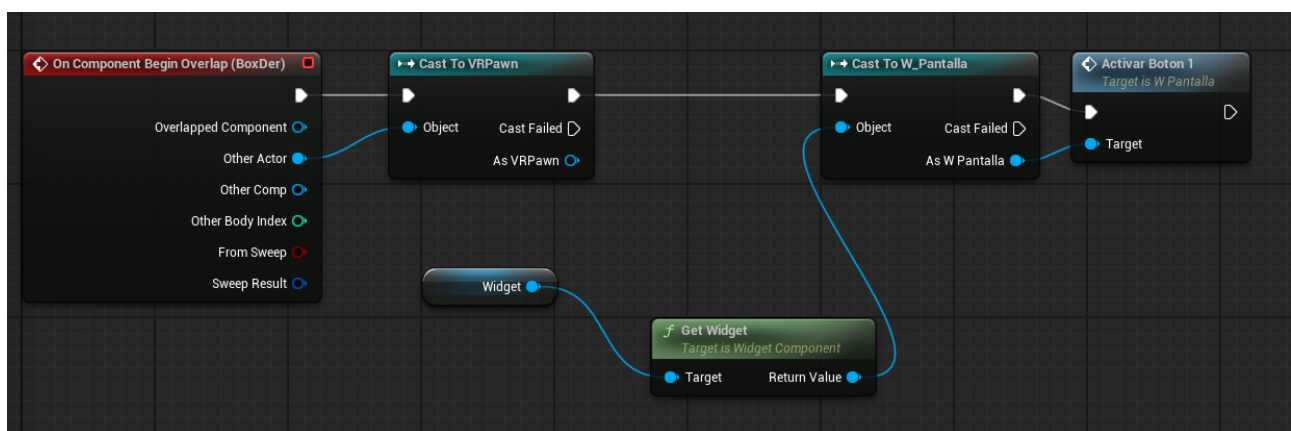


Figura 3-37 Código para interactuar con el widget

Para dotar de mayor calidad y realismo al efecto táctil, se ha configurado una ligera vibración en el mando de la mano con la que se pulsa. De este modo, al presionar un botón, el jugador sentirá una pequeña vibración, similar a la que incorporan algunos dispositivos en la vida real. Como se muestra en la Figura 3-38, utilizamos el nodo *Play Haptic Effect*, asignando el controlador correspondiente mediante *Get Player Controller*, asegurando que la vibración ocurra en la mano correcta.

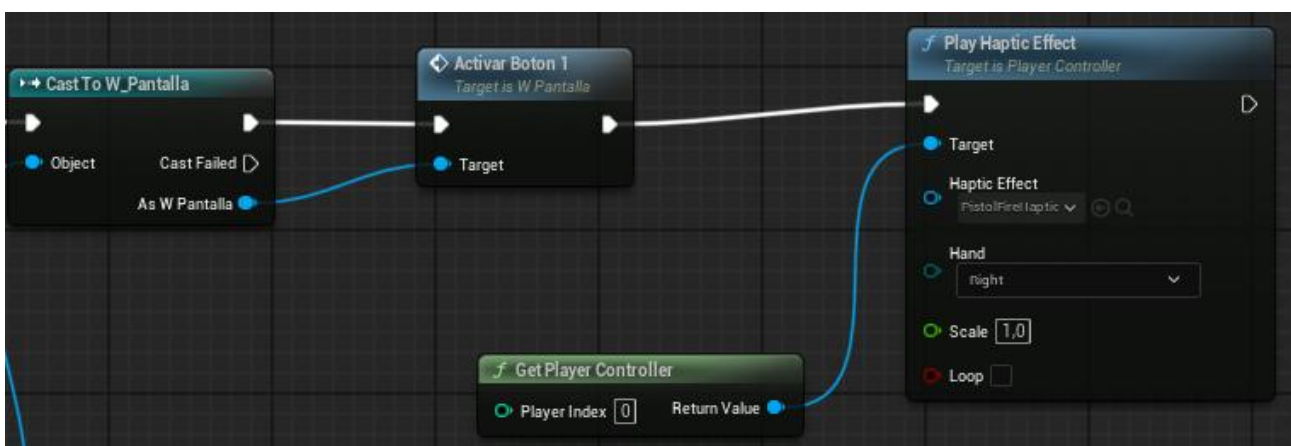


Figura 3-38 Código para añadir vibración al pulsar pantalla

Para permitir que la pizarra pueda ser anclada nuevamente a su posición original, se creó un Blueprint Actor que servirá como punto de anclaje. Este actor, mostrado en la Figura 3-39, simula un

tablón blanco, que servirá de base de la pantalla. Para detectar la presencia de la pizarra, se incorporó una *Sphere Collision* dentro del Blueprint. Esta colisión será la encargada de identificar cuando esta se acerque al punto de anclaje, permitiendo que el sistema ejecute la lógica necesaria para volver a fijarla en su sitio.



Figura 3-39 Componentes y viewport de BP_PuntoAnclaje

El código de anclaje de la pizarra, mostrado en la Figura 3-40, es prácticamente el mismo que el utilizado en el sistema de anclaje del puzzle. Se detecta la colisión con el punto de anclaje mediante una *Sphere Collision*, verificando que el objeto que entra en la zona es la pizarra a través de un *Cast* a BP_Pizarra.

Si la verificación es correcta, se configura el tipo de agarre en *Snap*, ajustando automáticamente la pizarra en su posición. Luego, se restablece el agarre a *Free* para permitir que pueda volver a moverse cuando el usuario lo desee.

Finalmente, se obtiene una referencia al *Player Pawn* y se realiza un *Cast* a *VRPawn* para garantizar que la lógica se aplique correctamente y se establezcan a nulo los valores de *Held Component Left* y *Held Component Right*, para que la pizarra se suelte de cualquiera de las manos, y quede anclada.

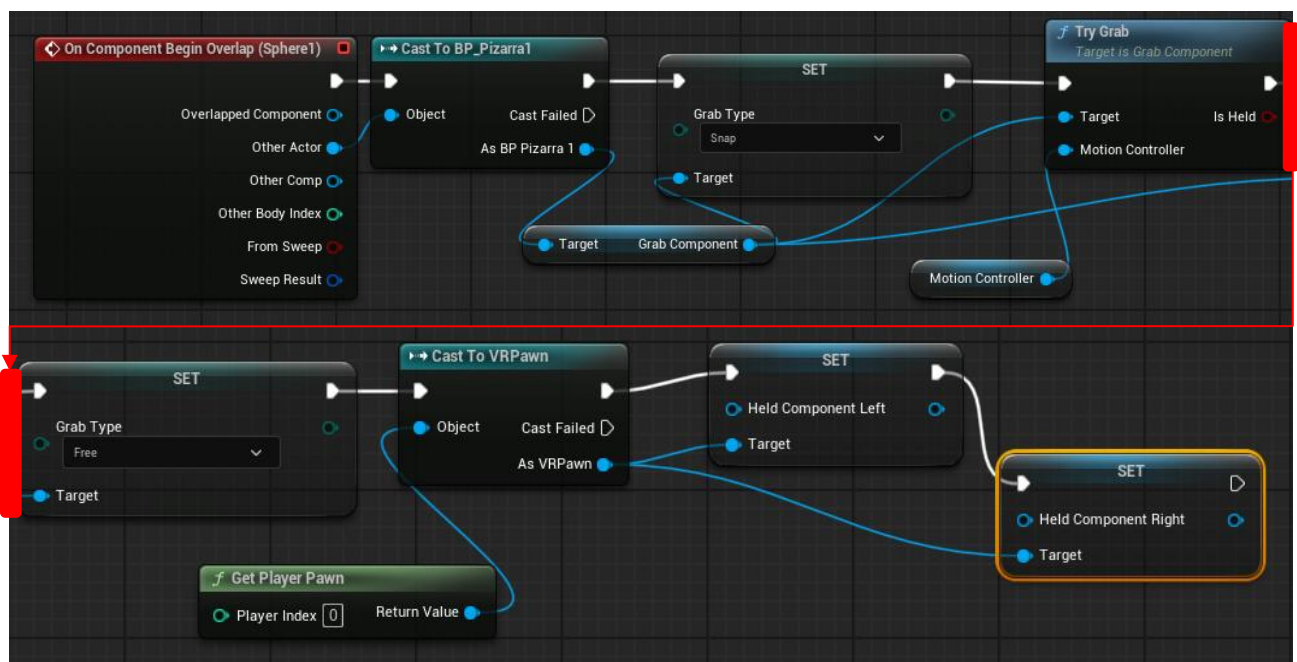


Figura 3-40 Código para anclar BP_Pizarra

3.2.5 Pirámide Central

Para la creación de la pirámide central del escenario, se procedió al diseño de un Blueprint Actor denominado *BP_Spawnear*, cuya función principal será la de generar diferentes objetos dentro del entorno virtual. Tal como se muestra en la Figura 3-41, se le asignó una geometría de tipo pirámide, la cual fue ajustada en dimensiones para adaptarse al diseño general del módulo base. Asimismo, se le aplicó un material con una tonalidad gris, asegurando así una integración visual coherente con el resto del entorno.

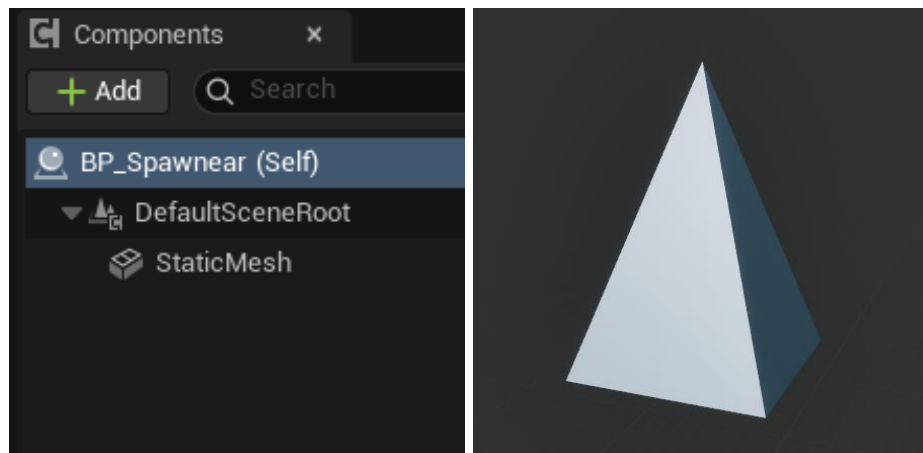


Figura 3-41 Componentes y viewport de *BP_Spawnear*

A continuación, se procedió a la programación de la lógica que permite hacer aparecer distintos objetos sobre la pirámide central. Para validar el correcto funcionamiento del sistema, se optó inicialmente por un diseño simple y funcional. Para ello, se accedió al *Event Graph* del Blueprint *BP_Spawnear*, mostrado en la Figura 3-42.

En primer lugar, se creó una variable de tipo *Array*, en la que se introdujeron los distintos objetos que se desean visualizar. Asimismo, se añadió una variable de tipo entero denominada *Nobjeto*, que representa el objeto visualizado en cada momento. Por último, se definió una variable adicional, *ÚltimoObjeto*, destinada a almacenar al objeto actualmente visualizado. Esta última permite destruir dicho objeto antes de generar uno nuevo, evitando así acumulaciones innecesarias en el escenario.

Como eventos de inicio se emplearon las pulsaciones del botón derecho e izquierdo del ratón. Al presionar el botón derecho, se incrementa en una unidad el valor de *Nobjeto*, mientras que al presionar el izquierdo, este valor se reduce. Para evitar errores por desbordamiento del *array*, el valor de esta variable se limita mediante el nodo *Clamp (Integer)*, fijando un mínimo de 0 y un máximo igual a la longitud del *array*.

Una vez determinado el índice del objeto a mostrar, se utiliza el nodo *SpawnActor* para hacer aparecer el objeto correspondiente. Para posicionarlo correctamente, se toma la ubicación actual de la pirámide en los ejes X e Y, y se modifica el eje Z para que el objeto aparezca justo encima de ella. Este desplazamiento vertical se configura mediante el nodo *Make Transform*.

Antes de realizar un nuevo *spawn*, se emplea un nodo *Branch* junto con *IsValid* para comprobar si ya existe un objeto visualizado. En caso afirmativo, se procede a destruirlo mediante el nodo *Destroy Actor*, garantizando que solo haya un objeto activo sobre la pirámide en todo momento. Si no hay ninguno, como sucede en la primera ejecución, el sistema continúa directamente con la generación del nuevo objeto.

Finalmente, se actualiza la variable *ÚltimoObjeto* con la referencia del nuevo actor instanciado, permitiendo así su correcta gestión en futuras interacciones.

A continuación, se procedió a la creación de un Blueprint object denominado *ButtonName*, cuya finalidad es servir como estructura de datos para cada entrada del menú. En su interior, se definieron únicamente dos variables: una de tipo *String*, destinada a almacenar el nombre identificativo del objeto, y otra de tipo *Actor Class*, en la cual se guarda la clase correspondiente al actor que será instanciado posteriormente.

Una vez establecido este Blueprint, se creó un nuevo widget llamado *W_Boton*, que actuará como plantilla para los botones que se añadirán dinámicamente al componente *ListView* del menú principal. Como se muestra en la Figura 3-44, dentro del content de *W_Boton*, se incluyó el *ButtonName*. De esta forma, cada botón del menú no solo mostrará el nombre del objeto, sino que también llevará asociada su clase. Además, dentro de las interfaces del widget *W_Boton*, se implementó la interfaz *User Object List Entry*, para habilitar la comunicación entre el widget y la *List View*.

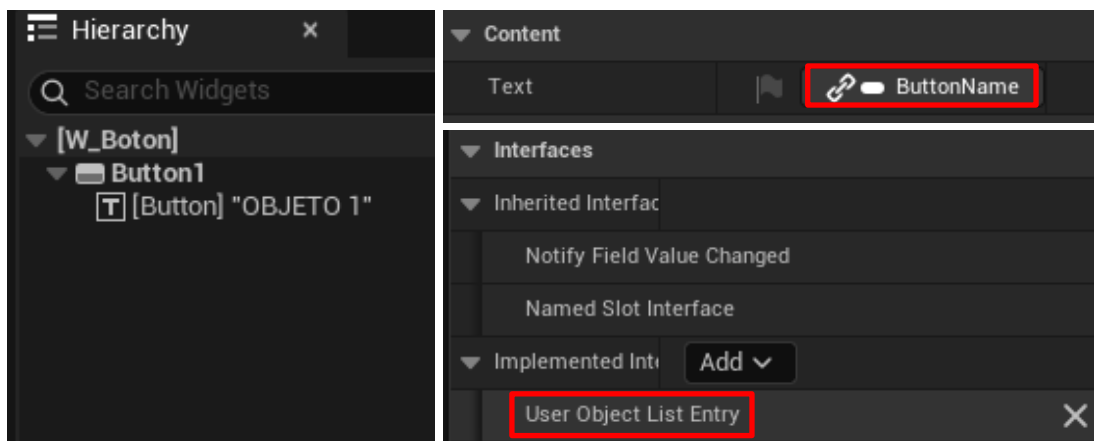


Figura 3-44 Jerarquía, interfaces y contenido de *W_Boton*

Posteriormente, nos dirigimos al *Graph* de *W_Boton* mostrado en la Figura 3-45, añadimos el evento *On List Item Object Set* se ejecuta automáticamente cada vez que el *ListView* asigna un nuevo objeto de datos a una instancia del widget. Se realiza un *cast* al Blueprint *ButtonName* para acceder a sus variables. A continuación, se extraen el nombre y la clase del objeto recibido, y se asignan a las variables internas del botón *ButtonName* y *ButtonClass*. Esto permite que cada botón del menú tenga su texto y funcionalidad correctamente configurados en función del objeto correspondiente.

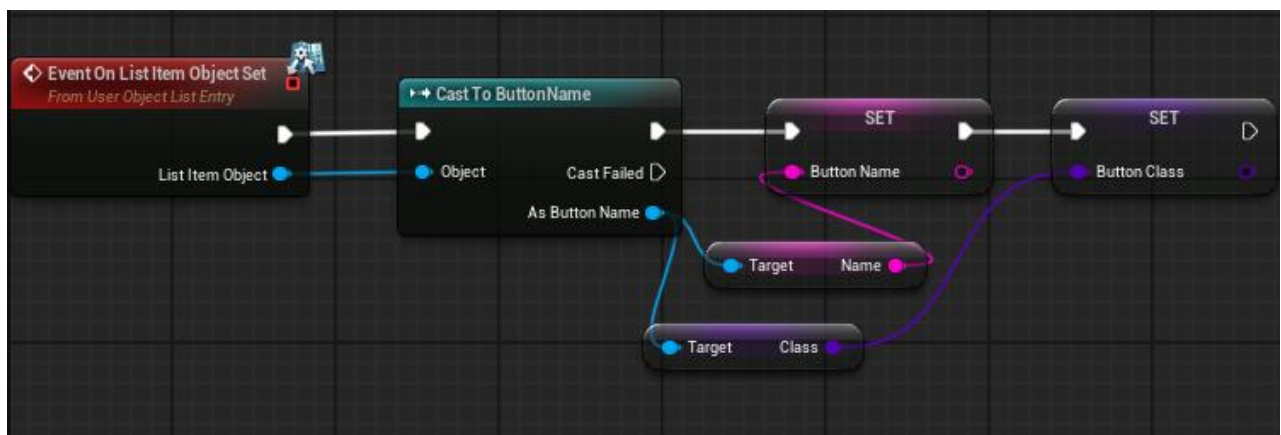


Figura 3-45 Código para obtener asociar texto y clase a cada botón

Con el fin de generar de forma dinámica los botones que se añadirán al menú, se desarrolló el código mostrado en la Figura 3-46 dentro del *Graph* del widget *WidgetMenu*. A partir de la variable *ObjectList*, de tipo *Map*, que almacena pares de claves tipo *String* asociadas a clases de tipo *Actor*, se obtienen únicamente las claves de tipo *String* mediante el nodo *Keys*. Estas claves representan los nombres de los objetos y serán utilizadas como etiquetas visibles en los botones del menú.

Mediante un bucle *For Each Loop*, se recorre cada clave y, por cada una, se construye un nuevo botón utilizando el nodo *Construct Button Name*, con la clase previamente creada *ButtonName*. A continuación, se establece el valor del nombre del botón con la clave actual, y se recupera su clase correspondiente del mapa mediante el nodo *Find*, asignándola a la variable de clase del botón.

Finalmente, cada botón configurado se añade al componente *ListView* a través del nodo *Add Item*, lo que permite que aparezca automáticamente en el menú. Esta lógica asegura que el número y contenido de los botones del menú se adapte de forma automática a los elementos definidos en el mapa *ObjectList*.

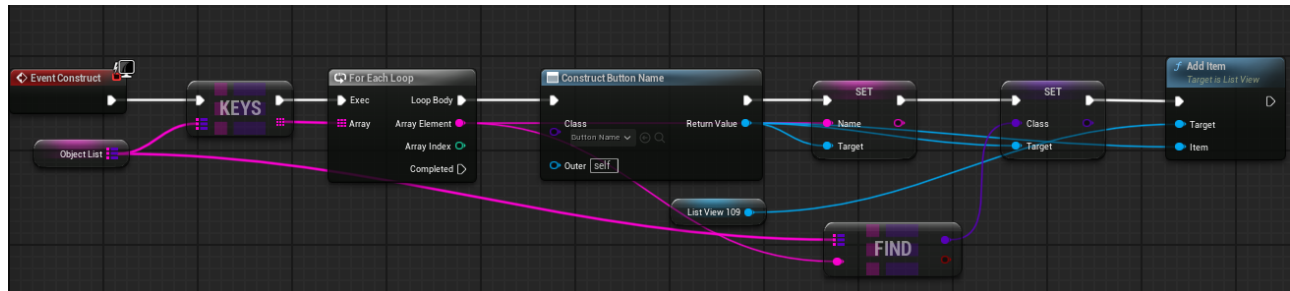


Figura 3-46 Código para crear botones y añadirlos al *Widget Menu*

Una vez finalizado el sistema de menú, se creó una función dentro del *BP_Spawnnear* denominada *Spawn Actor*. Para ello, se procedió a reutilizar la lógica de aparición de objetos previamente desarrollada, pero esta vez la lógica estaría dentro de la nueva función, tal y como se muestra en la Figura 3-47.

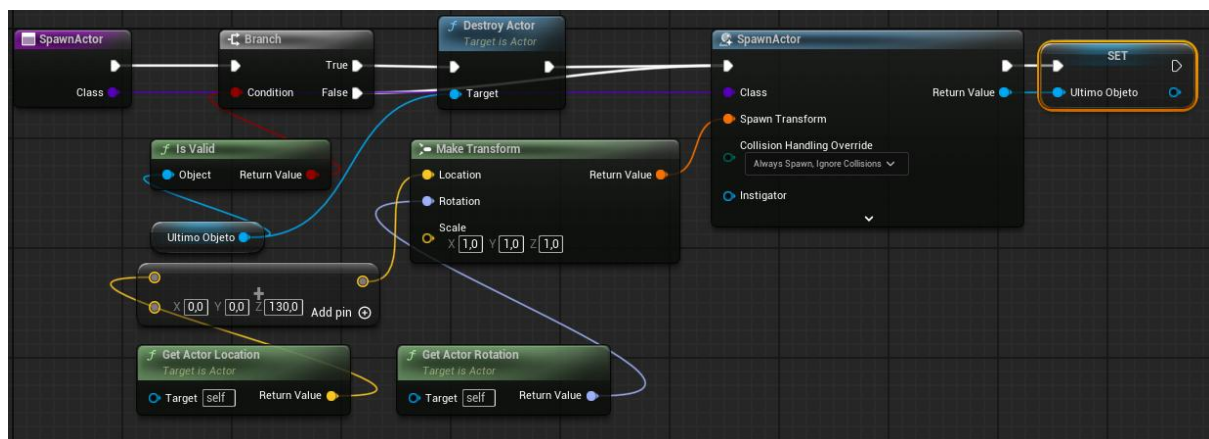


Figura 3-47 Código de la función *Spawn Actor*

Para finalizar la lógica, se añadió al widget *W_Botón* un código utilizando el evento *OnClicked*, como se muestra en la Figura 3-48. Al pulsar un botón del menú, se extrae la clase del actor asociada al botón mediante *Get Actor of Class*. Finalmente, se ejecutará la función *Spawn Actor* con el parámetro de clase de actor correspondiente, que será el que se visualizará sobre la pirámide.

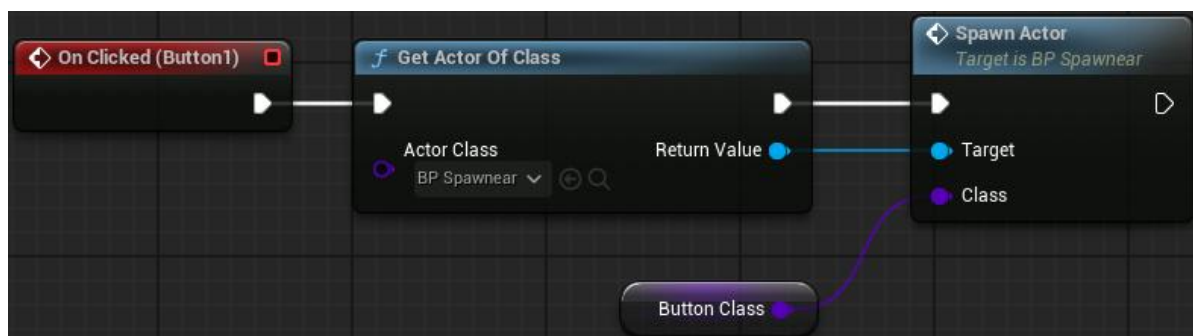


Figura 3-48 Código para activar la función *Spawn Actor*

Por último, se procedió a añadir diversos elementos a la variable *Object List*, de tipo *Map*, asociando a cada uno un nombre identificativo. Como aparece en la Figura 3-49, se incluyeron dos elementos con geometrías simples, una pelota y un cubo, además de la pistola utilizada en el escenario, a la cual se le desactivaron las propiedades físicas y la gravedad para evitar comportamientos no deseados. Adicionalmente, se incorporó una entrada denominada *Vacio* sin ningún actor asociado, que permite dejar la pirámide sin objeto tras haber visualizado alguno, si así se desea.

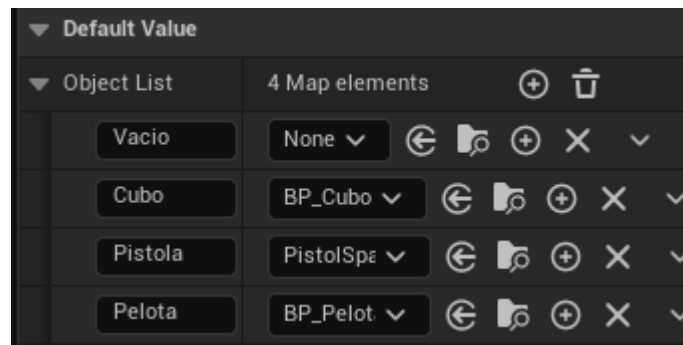


Figura 3-49 Lista de elementos de la variable *Object List*

3.2.6 Entorno de Plataforma VR

Este apartado se centra en el diseño del área destinada al uso e interacción con la plataforma *KAT VR* [24] dentro del escenario. Para implementar su funcionamiento, surgió la idea de ampliar el escenario añadiendo un pasillo adyacente a la pared de esta sección, permitiendo probar la plataforma sin las limitaciones de espacio del entorno principal.

Para ello, el primer paso consiste en modificar la pared de la sección, que se muestra en la Figura 3-50, realizando un corte en la parte central con las dimensiones de la puerta que se colocará posteriormente. De este modo, se permite cerrar el acceso cuando sea necesario, manteniendo la esencia del escenario original y facilitando la integración del pasillo.

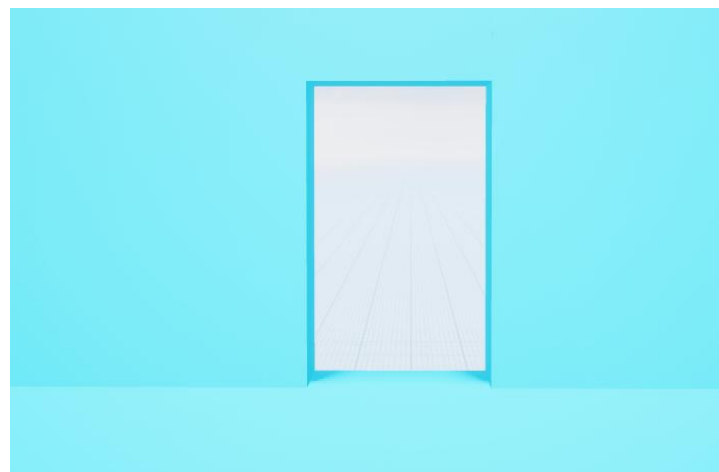


Figura 3-50 Corte en pared del Entorno de Plataforma VR

A continuación, se crea la puerta, la cual consiste en un Blueprint Actor con forma rectangular, adaptado a las dimensiones del hueco en la pared y manteniendo el color de la sección, como se muestra en la Figura 3-51. La intención es que el usuario inmerso en la simulación pueda acceder al pasillo adyacente de manera intuitiva. La solución más simple dentro de la inmersión será que al tocar la puerta, esta desaparezca, permitiendo el acceso. Para facilitar aún más la interacción se creará un cartel que indique el punto exacto donde el usuario debe tocar para abrir el paso al pasillo. Dicho cartel, consiste en un Blueprint actor de color similar al de la sección al que se le incorpora un widget con el texto.

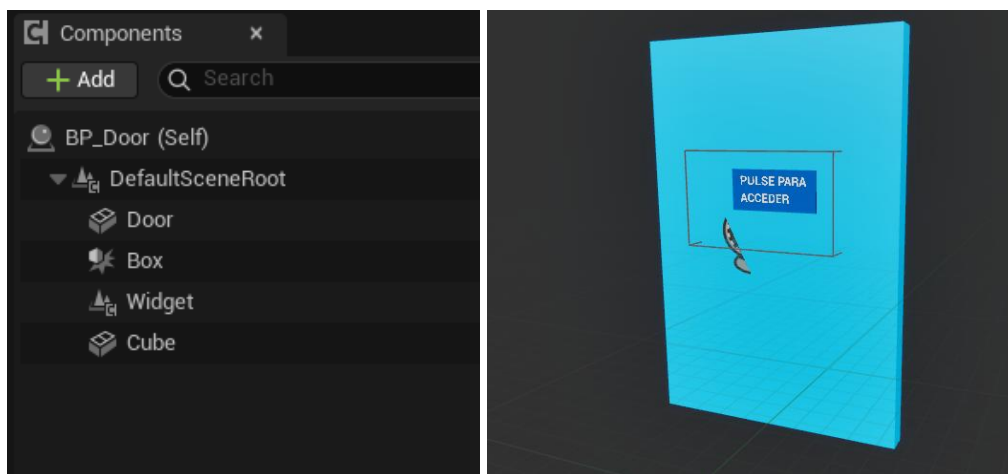


Figura 3-51 Componentes y viewport de BP_Door

El código programado en Blueprints que aparece en la Figura 3-52 establece la lógica para hacer desaparecer la puerta cuando el usuario la toca. Para ello, se ha añadido una *Box Collision* a la puerta, centrado con la posición del cartel. Cuando el usuario entra en contacto con esta área de colisión, se activa el evento *On Component Begin Overlap (Box)*, que detecta la interacción. Una vez activado el evento, se ejecuta un *Cast To VRPawn* para verificar si el objeto que ha colisionado es el usuario en realidad virtual. Si la verificación es exitosa, se procede a la siguiente acción.

Cuando se confirma que el usuario ha tocado la puerta, se ejecuta el nodo *Destroy Actor*, eliminando la puerta del escenario y permitiendo el acceso al pasillo adyacente. Para mejorar la respuesta sensorial y hacer la interacción más realista, se incorporan efectos hápticos en los controladores de realidad virtual. Mediante el nodo *Get Player Controller*, se obtiene el controlador del jugador y se activan los nodos *Play Haptic Effect*, generando una vibración en ambas manos al tocar la puerta. Esto refuerza la sensación de interacción física dentro de la simulación.

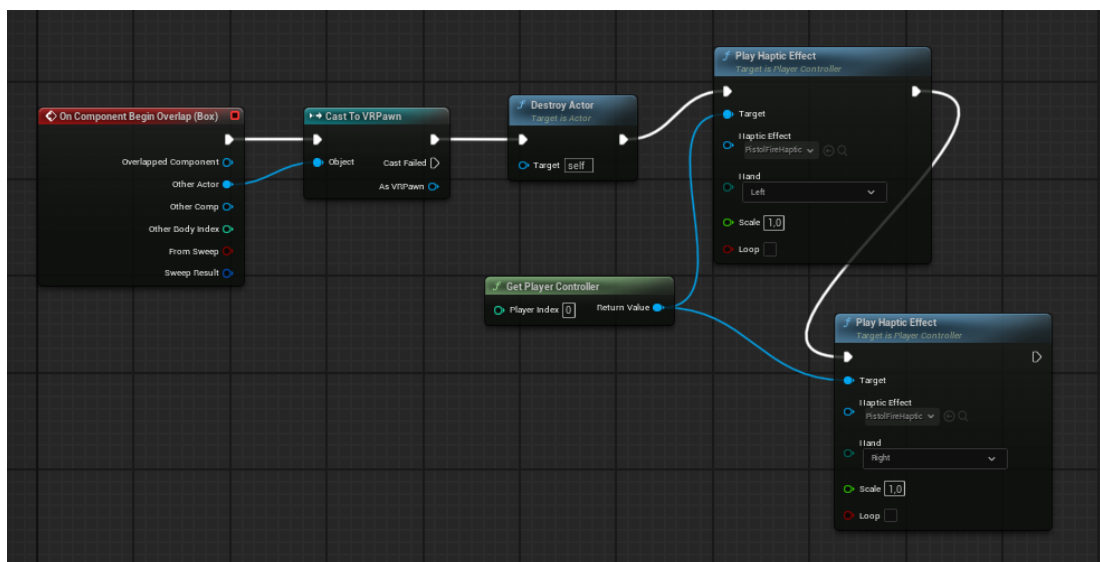


Figura 3-52 Código para hacer desaparecer la puerta

Para la creación del pasillo, se ha diseñado un entorno que evoca la apariencia de una calle empedrada. Este diseño busca ofrecer una sensación de realismo y familiaridad, asemejándose a los senderos de exteriores urbanos o parques. Además, se ha incorporado vegetación cuidadosamente perfilada a lo largo de los laterales, evocando la estética de jardines bien cuidados. Este elemento natural no solo mejora la ambientación del espacio, sino que también contribuye a generar una atmósfera más agradable y acogedora para el usuario. La combinación de estos factores favorece un entorno idóneo

tanto para la exploración como para actividades de desplazamiento, como caminar o correr, fomentando la inmersión del usuario en el mundo virtual.

Se comenzó definiendo la forma y dimensiones del pasillo, para luego aplicar los materiales adecuados que le otorgaran realismo. Como se muestra en la Figura 3-53.



Figura 3-53 Materiales del pasillo del entorno de plataforma

3.3 Realidad Virtual

Las gafas de realidad virtual Meta Quest 3 [21], mostradas en la Figura 3-54 constituyen el dispositivo fundamental para sumergirse en una experiencia inmersiva, actuando como nexo principal entre usuario y entorno virtual. Su importancia durante la fase de desarrollo radica en la posibilidad de detectar y corregir errores en la programación de manera más eficiente. Al permitir la interacción directa con el escenario en tiempo real, facilitan la identificación visual del comportamiento de los distintos componentes, lo que permite analizar y depurar la codificación de forma más intuitiva.

Además, la posibilidad de visualizar continuamente el escenario permite realizar ajustes en cada detalle hasta alcanzar un nivel óptimo de realismo. Esta capacidad de iteración constante facilita la mejora progresiva del entorno, asegurando que cada elemento esté correctamente integrado y optimizado. Como resultado, se obtiene un escenario final completamente pulido, listo para ofrecer una experiencia inmersiva de alta calidad.



Figura 3-54 Gafas Meta Quest 3

El proceso seguido para conectar las gafas al ordenador comienza con la configuración de la conexión Bluetooth a través de la plataforma Steam. Para ello, accedemos a la barra superior de Steam, abrimos el menú Ver y seleccionamos la opción Iniciar SteamVR como se muestra en la Figura 3-55.

A continuación, habrá que colocarse la gafas, ajustarlas, y encenderlas pulsando el botón blanco circular ubicado en la parte izquierda del dispositivo. Una vez encendidas, accedemos al menú de SteamVR pinchando el logo de Steam, ya sea con el mando o de forma táctil con el dedo. Desde este menú, seleccionamos nuestro ordenador y presionamos *Iniciar*. Tras la conexión, se procede a establecer los límites de VR, asegurando que el área de juego esté despejada para evitar obstáculos durante la experiencia inmersiva. Una vez definidos los límites, las gafas quedan correctamente conectadas al ordenador, permitiendo su uso dentro del entorno virtual. Finalmente, para previsualizar nuestro escenario de realidad virtual, accedemos al menú de Unreal Engine y seleccionamos los tres puntos ubicados en la barra superior de la interfaz y escogemos la opción *VR Preview*, tal como se muestra en la parte superior derecha de la Figura 3-55. Este proceso nos permite visualizar el entorno en tiempo real.

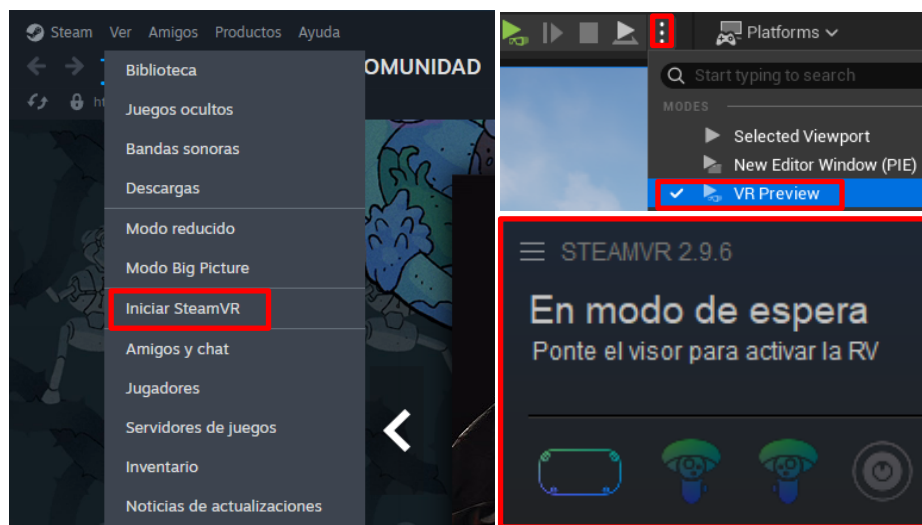


Figura 3-55 Steam VR

En la Figura 3-56 se representa la disposición de los controladores utilizados durante la simulación, detallando las funcionalidades asignadas a cada uno de los botones. Estos mandos, diseñados específicamente para entornos de realidad virtual, permiten al usuario interactuar de forma natural e intuitiva con los elementos del entorno.

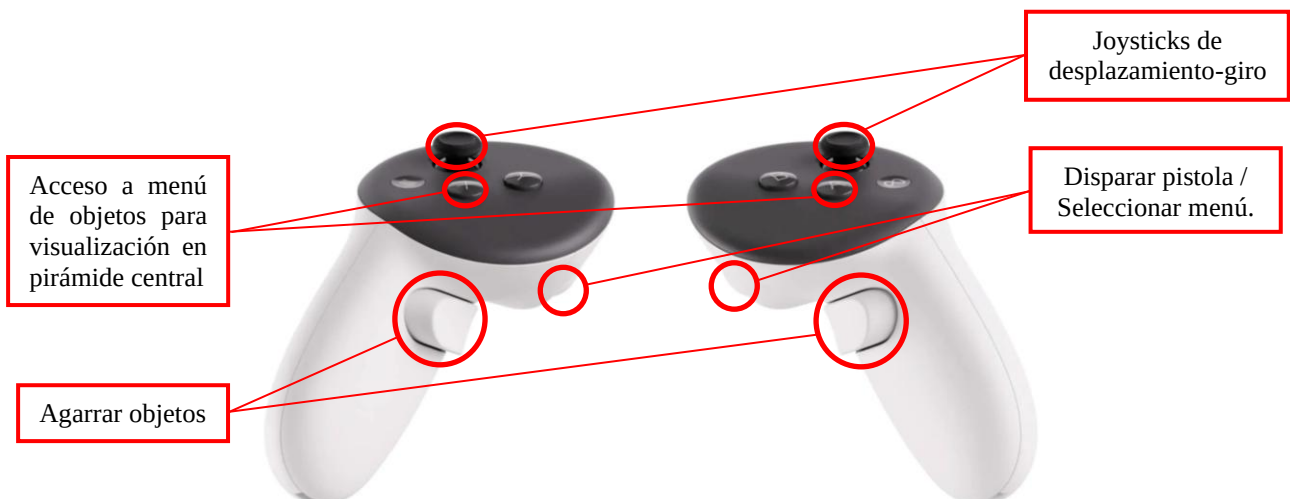


Figura 3-56 Funciones de cada botón en controladores VR

Aunque es posible desplazarse en el entorno virtual mediante movimiento físico en el mundo real, el método principal de locomoción utilizado en la simulación será el teletransporte con los joysticks. Para ello, basta con orientar el joystick en la dirección deseada. En ese momento, se proyecta un rayo con forma de parábola desde el controlador virtual, el cual apunta hacia la zona del suelo a la que se realizará el teletransporte una vez se suelte el joystick.

La posición de destino puede ajustarse fácilmente variando la dirección y la distancia mediante el propio joystick, lo que permite un control preciso y adaptado al entorno. Esta técnica resulta especialmente cómoda e intuitiva cuando no se dispone de un amplio espacio físico, ya que permite al usuario desplazarse con fluidez sin necesidad de caminar realmente.

Adicionalmente, al realizar un movimiento corto del joystick hacia la derecha o la izquierda, se genera un leve giro en la orientación del usuario dentro del entorno virtual. Esta función es especialmente útil para hacer ajustes finos de orientación, o para alinear al avatar en la dirección que ofrezca más espacio libre en el entorno real, maximizando así el área utilizable.

3.4 Integración de Periféricos

3.4.1 *ProtubeVR*

Para comenzar a utilizar el ProVolver Haptic Pistol [22], en la parte izquierda de la Figura 3-57, lo primero que debemos hacer es asegurarnos de que el dispositivo tiene suficiente carga. Es recomendable cargar la batería por completo antes de su uso. Una vez que el dispositivo esté cargado, procederemos a conectarlo al ordenador mediante Bluetooth. Para verificar que la conexión es correcta y que el sistema háptico funciona adecuadamente, abriremos la aplicación ProTubeVR Companion-App, que se muestra en la Figura 3-57. Dentro de la aplicación, encontraremos diversas opciones de simulación que nos permitirán probar el funcionamiento del sistema háptico. Estas pruebas nos ayudarán a asegurarnos de que la respuesta del dispositivo es la adecuada y está correctamente sincronizada.



Figura 3-57 ProVolver Haptic Pistol e interfaz de ProtubeVR Companion-App

Una vez verificado que el dispositivo háptico de ProTubeVR funciona correctamente, procederemos a instalar el plugin correspondiente a nuestra versión de Unreal Engine, como se muestra en la Figura 3-58. Para ello, debemos dirigirnos al repositorio oficial de ProTubeVR en GitHub [26], donde encontraremos distintas versiones del plugin. Es importante descargar la versión compatible con nuestra instalación de Unreal Engine para garantizar su correcto funcionamiento.

Tras descargar el archivo comprimido, lo extraeremos y ubicaremos los archivos en la carpeta correspondiente dentro del directorio de nuestro proyecto en Unreal Engine.

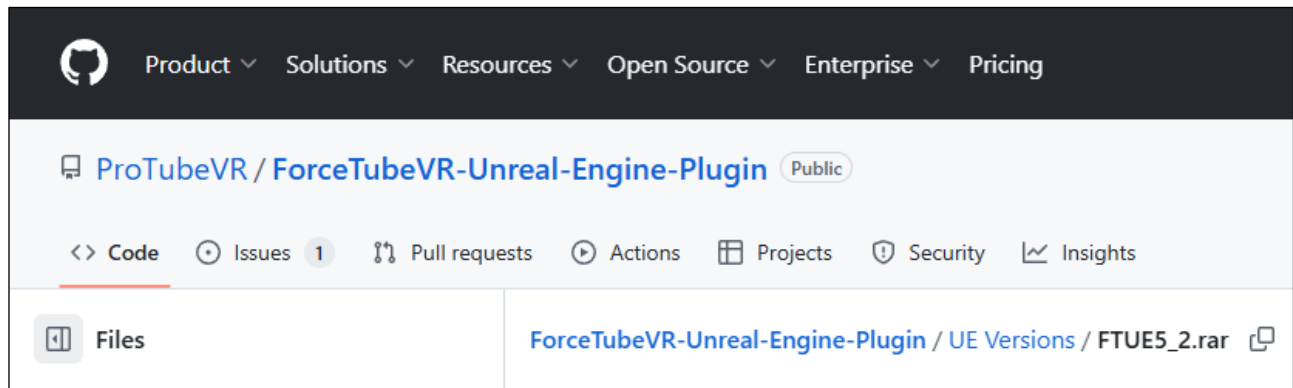


Figura 3-58 Plugin ForceTubeVR-Unreal Engine

Tras instalar el paquete y ubicarlo correctamente dentro del proyecto, visualizamos el nuevo código que incorpora. Posteriormente accederíamos al *Event Graph* de la pistola, lo analizaremos para localizar donde añadiremos el nuevo código de retroalimentación háptica.

En la Figura 3-59, se observa que el código maneja dos eventos distintos uno para la mano izquierda y otro para la derecha. Cada uno se activa al presionar el botón de disparo correspondiente para cada mano y primero verifica si la pistola está correctamente agarrada en la mano. De esta manera, si se presiona el botón de disparo con la misma mano con la que se esté agarrando la pistola, el flujo de la programación continuará, para posteriormente hacer que salga disparado el proyectil, mediante un nodo *SpawnActor*.

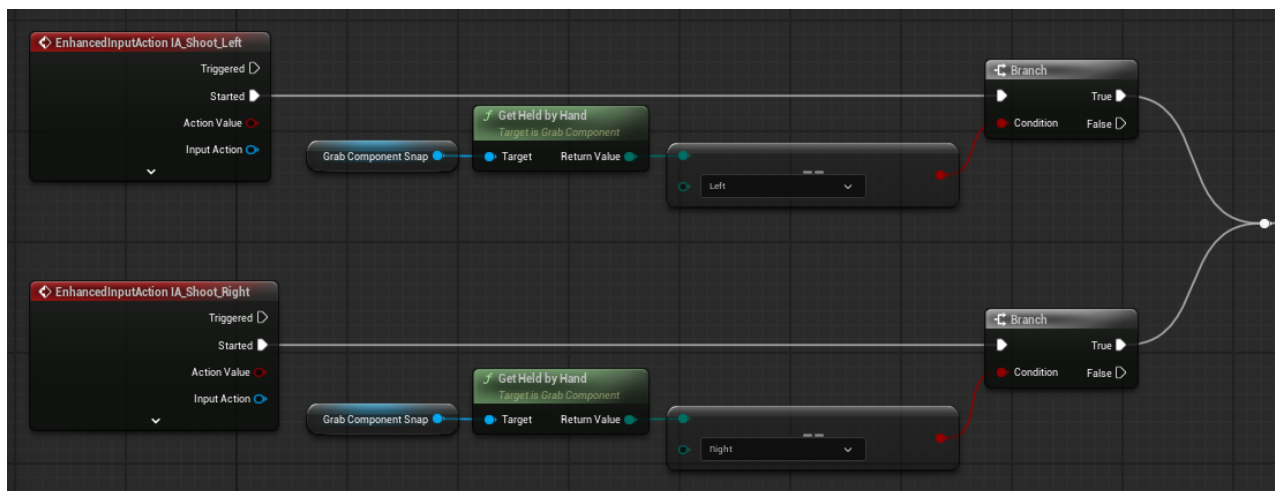


Figura 3-59 Código para activar el disparo

Una vez localizada la parte del código encargada de activar el disparo, procedemos a introducir la lógica necesaria para generar el efecto háptico justo antes de que se dispare el proyectil, asegurando así que el usuario reciba una respuesta táctil en el momento exacto en que ejecuta el disparo.

En la Figura 3-60 se muestra la implementación, donde primero se ajusta la potencia del retroceso mediante una función que convierte un valor de tiempo en una fuerza de impacto. Seguidamente, se configura el canal de vibración, estableciendo una intensidad de 200 y una duración de 0,3 segundos, lo que simula el impacto del disparo en la pistola. Esta vibración se asocia al canal correspondiente, asegurando que se active únicamente cuando se realice un disparo.

Finalmente, se ejecuta la función que genera el proyectil en el entorno de realidad virtual. Para ello, se define la clase del proyectil, su posición inicial en el escenario y los parámetros de colisión, asegurando que el objeto se cree con las propiedades adecuadas.

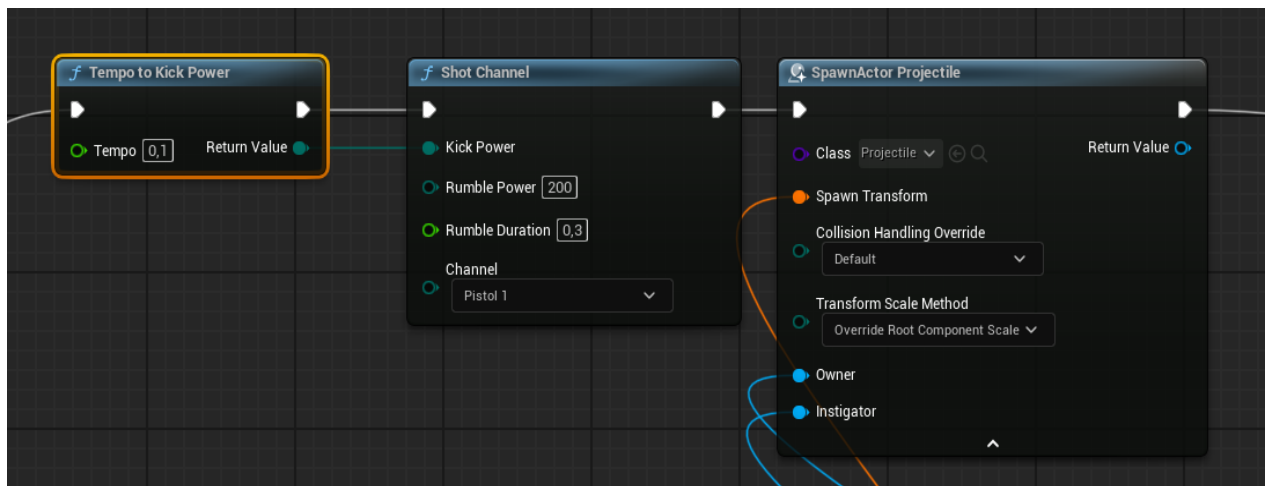


Figura 3-60 Código para introducir efecto háptico de pistola

3.4.2 BHaptics

Para integrar el TactSuitX40 de bHaptics, que aparece en la Figura 3-61, el primer paso es conectarlo al ordenador mediante Bluetooth. Al encender el chaleco, debemos asegurarnos de que este en modo emparejamiento. Luego, se selecciona desde la lista de dispositivos disponibles y se completa la conexión.

Después de la conexión, es necesario instalar bHaptics Player, mostrado en la Figura 3-61, descargándolo desde la página oficial. Una vez instalado y ejecutado, se verifica que el chaleco aparece como conectado en la interfaz del software. Para probar su funcionamiento, se accede a la opción *Prueba de retroalimentación*, donde se puede seleccionar diferentes posiciones e intensidades de vibración. Este programa permite controlar y ajustar la retroalimentación háptica, asegurando que todas las respuestas del chaleco sean correctas antes de su integración en entornos de realidad virtual.

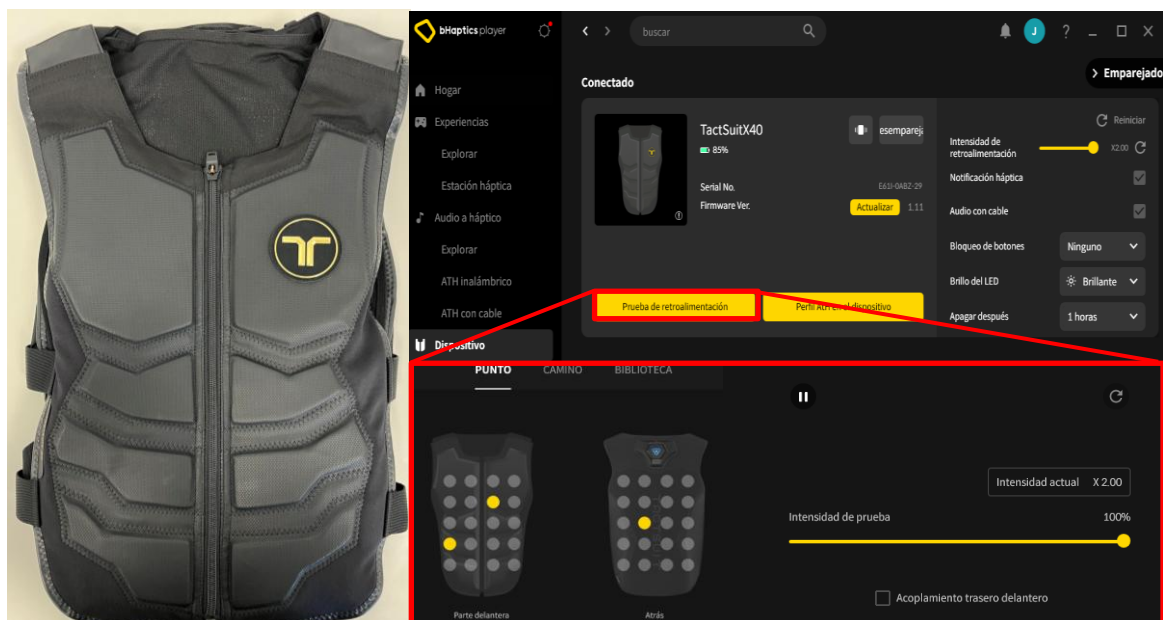


Figura 3-61 ChalecoTactSuitX40 e interfaz de bHaptics Player

Lo siguiente será descargar el plugin de bHaptics para Unreal Engine, como se muestra en la Figura 3-62. Este SDK2, compatible con Unreal Engine 4.26 - 5.5, nos permitirá integrar el TactSuitX40 en el proyecto. Para ello, basta con hacer clic en *Download* y seguir la *Integration Guide* para configurarlo correctamente en el entorno de desarrollo.

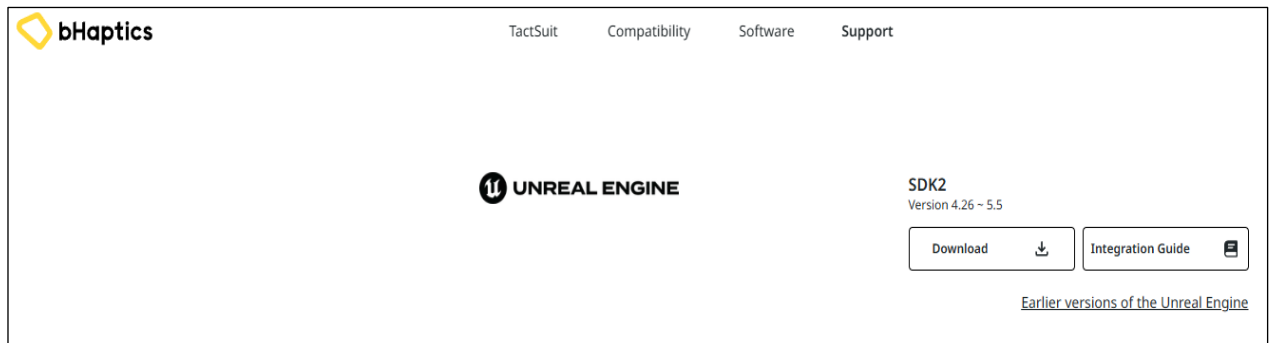


Figura 3-62 Plugin de BHaptics compatible con Unreal Engine

En Unreal Engine, accedemos al menú superior que se muestra en la Figura 3-63 hacemos clic en *Edit* y seleccionamos *Plugins*. Luego, en la lista de plugins instalados, buscamos la categoría *Haptic* y activamos *bHapticsPlugin* marcando la casilla correspondiente. Con esto, el plugin estará listo para su configuración en nuestro proyecto de realidad virtual.

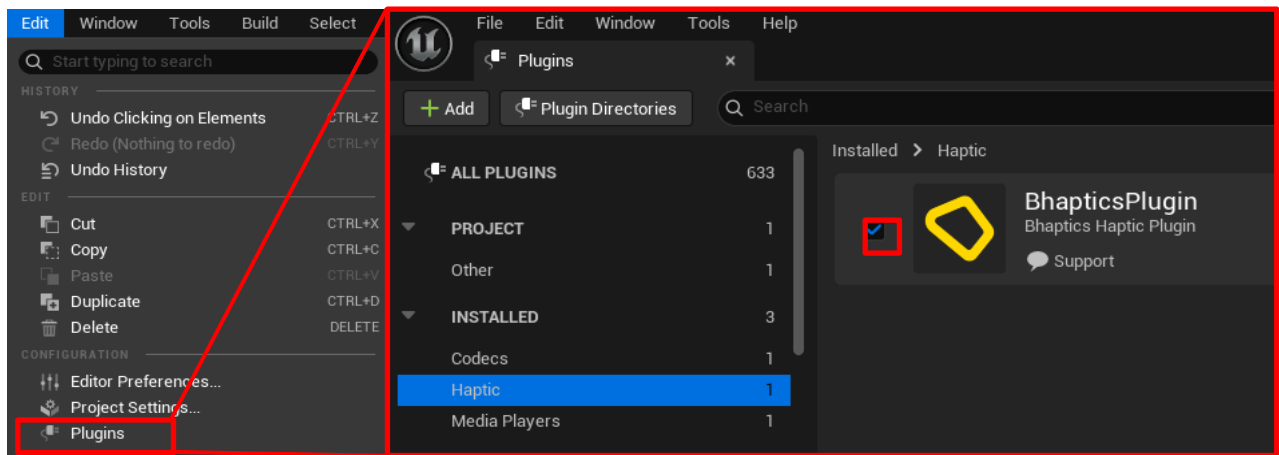


Figura 3-63 Plugins en Unreal Engine

Seguidamente, nos dirigimos al portal de bHaptics Developer que aparece en la Figura 3-64, donde crearemos un nuevo proyecto llamado *PROJECT* como se muestra en la imagen. Para ello, hacemos clic en *Create* y le damos nombre.

Abrimos el proyecto en el portal de bHaptics Developer y configuramos nuestra cuenta siguiendo la guía. Una vez completada la configuración, nos dirigimos a la sección *Settings*, donde se nos mostrarán dos códigos clave: *Application ID* y *Latest API Key*. Estos serán esenciales más adelante para enlazar Unreal Engine con nuestro proyecto de bHaptics, permitiendo la integración de la retroalimentación háptica en nuestro entorno de realidad virtual.

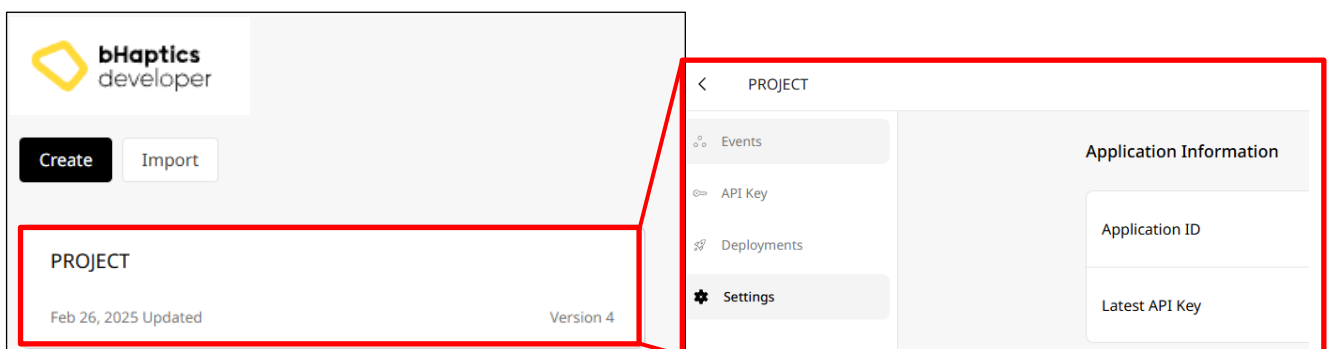


Figura 3-64 Portal de Bhaptics Developer

Para completar la integración del sistema háptico en Unreal Engine, accedemos a *Edit*, nos dirigimos a *Project Settings*, y dentro de la sección *Plugins*, seleccionamos *bHaptics*, como se muestra en la Figura 3-65.

A continuación, introducimos los códigos previamente generados en el portal de *bHaptics Developer*. Si los datos son correctos, aparecerá la opción *Update bHaptics*, la cual debemos pulsar para validar la configuración. Al hacer clic, se genera automáticamente una carpeta con los archivos necesarios para la integración háptica, que se ubicará en el *Content Browser* de nuestro proyecto.

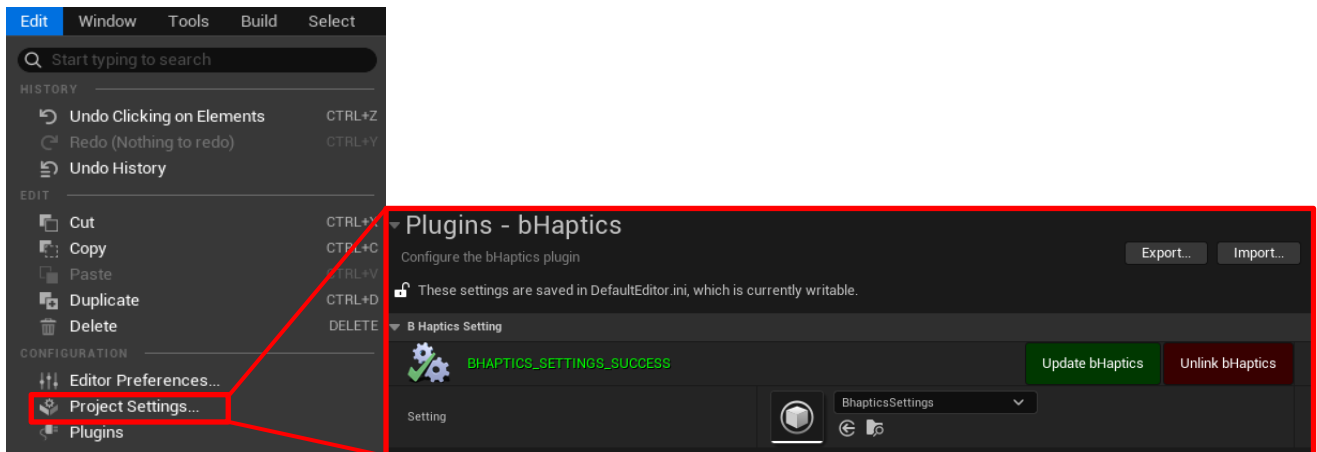


Figura 3-65 Project Settings Unreal Engine

Para continuar con la integración, creamos un nuevo Blueprint basado en la clase *GameInstance*, como se muestra en la Figura 3-66. Este Blueprint será fundamental para gestionar la conexión persistente con el sistema háptico a lo largo de toda la ejecución del proyecto. Una vez creado, lo guardamos en el *Content Browser*.

A continuación, abrimos el Blueprint recién creado y accedemos a la sección *Event Graph*, donde añadimos el código para establecer la comunicación con el sistema.

El nodo *Event Init* se ejecuta de manera automática al iniciar el juego, estableciendo la comunicación con los dispositivos hápticos mediante la función *Initialize bHaptics*. Esto permite que la retroalimentación háptica esté disponible desde el comienzo de la simulación.

Por otro lado, el nodo *Event Shutdown* se activa al cerrar el juego y ejecuta la función *Destroy bHaptics*, la cual desconecta los dispositivos y libera los recursos utilizados. Esto es crucial para evitar conflictos en futuras ejecuciones, asegurando un funcionamiento estable y optimizado del sistema háptico en el entorno de realidad virtual.

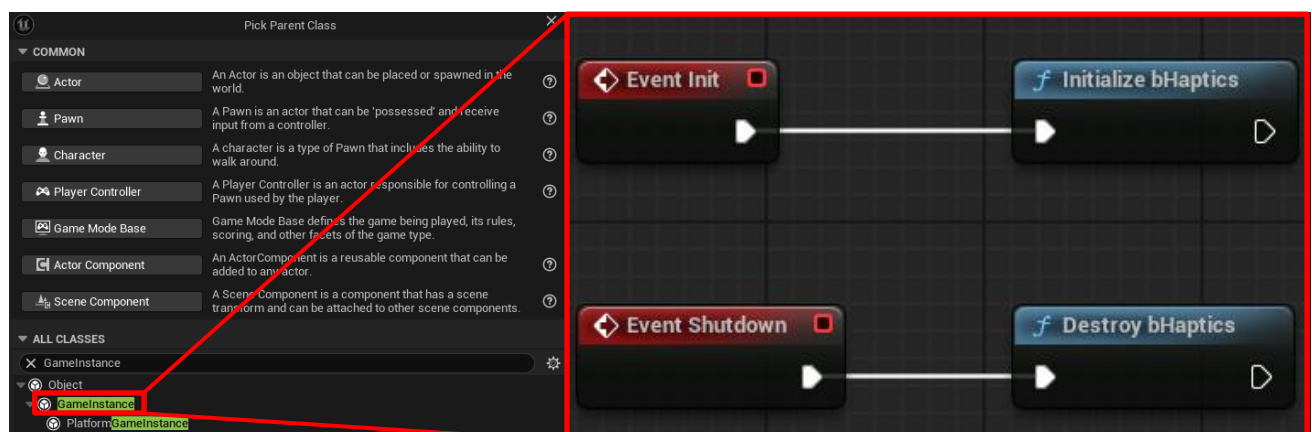


Figura 3-66 Blueprint GameInstance y código para inicio y fin de comunicación háptica

Para finalizar el proceso de integración, nos dirigimos nuevamente a *Project Settings* y accedemos a la sección *Maps & Modes*. Dentro de este apartado, buscamos la opción *Game Instance Class* y seleccionamos el Blueprint que acabamos de crear.

A continuación, accedemos a bHaptics Designer, como se muestra en la Figura 3-67. Este diseñador nos permite crear y personalizar efectos hápticos con las características que deseemos, adaptándolos a las necesidades de nuestro proyecto en realidad virtual.

Dentro del editor, podemos diseñar distintos efectos hápticos, ya sea combinándolos o separándolos en la línea de tiempo, ajustando su inicio y duración según lo requiera la simulación. Además, es posible controlar la intensidad de la retroalimentación, permitiendo que sea constante o fluctuante según el tipo de efecto que queramos generar.

También podemos definir en qué puntos del chaleco háptico se aplicará la retroalimentación, ya sea de forma simultánea en varias zonas o siguiendo un orden específico, lo que permite recrear sensaciones más naturales y precisas.

Gracias a todas estas opciones, bHaptics Designer nos proporciona un control detallado sobre la experiencia háptica, asegurando que los efectos sean lo más realistas e inmersivos posible dentro del entorno virtual.

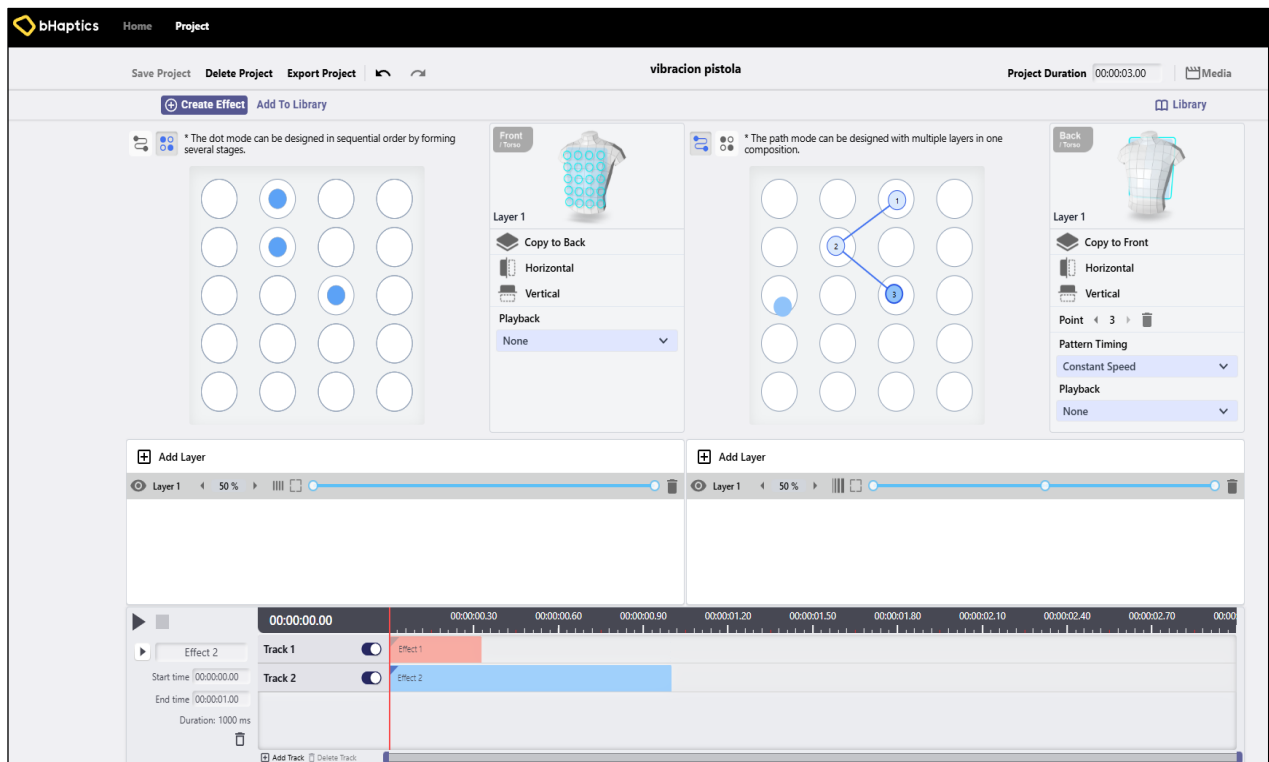


Figura 3-67 BHaptics Designer

Una vez que hayamos diseñado el efecto háptico con las características deseadas en *bHaptics Designer*, el siguiente paso será exportarlo. Para ello, crearemos una carpeta en nuestro ordenador, donde guardaremos todos los efectos hápticos que hayamos diseñado.

Este paso es importante porque el Portal de *bHaptics Developer*, donde importaremos los efectos personalizados, únicamente permite importar carpetas, en lugar de archivos individuales, por lo que facilitamos su correcta importación al sistema.

Una vez importados al Portal de bHaptics Developer, los efectos se almacenarán dentro del apartado *PROJECT*, como se muestra en la Figura 3-68. Por último, hacemos clic en *Save* para guardar los cambios y luego en *Deploy* para activarlos y sincronizarlos con nuestro proyecto en Unreal Engine.

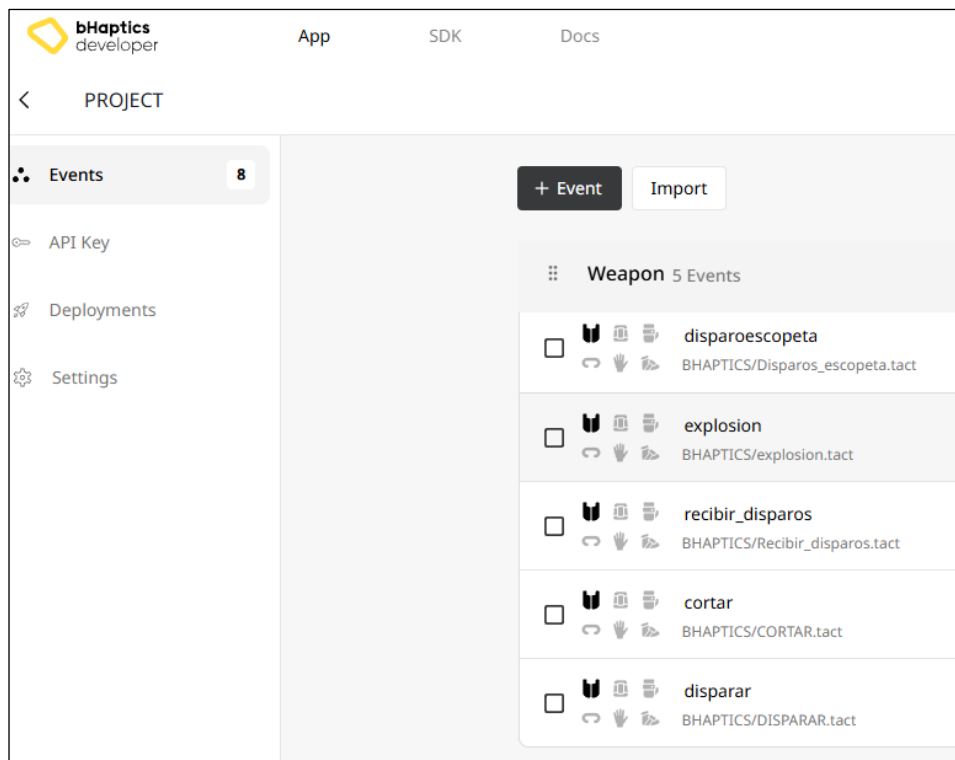


Figura 3-68 Efectos importados en BHaptics Developer

Para este proyecto, se han creado los siguientes efectos:

- Disparar pistola: Este efecto, a diferencia de los otros, ira incorporado en el evento de disparo de la pistola. Consiste en un ligero impacto háptico que se extiende por el tren superior del usuario en el momento en que la pistola es disparada.
- Recibir disparo con escopeta: Este efecto simula el golpe de los perdigones en el torso mediante tres impactos sucesivos. Entre ellos, se añade un efecto de empuje, recreando la sensación de retroceso corporal tras el disparo.
- Explosión: Este efecto comienza con un cosquilleo ligero por todo el chaleco, aumentando rápidamente en intensidad. A continuación, se aplica un impacto simultáneo en todos los puntos de retroalimentación al máximo nivel, simulando la onda expansiva de la explosión. Finalmente, la intensidad disminuye rápidamente, recreando la disipación del impacto hasta su completa desaparición.
- Corte: Este efecto simula un corte transversal a través del torso, aplicando una retroalimentación intensa tanto en la parte delantera como trasera del chaleco. La vibración comienza en un lado del torso y avanza en línea recta hasta el otro extremo, ajustando la velocidad del desplazamiento para mejorar la sensación de realismo y simular con precisión el avance de la motosierra.
- Recibir ráfaga de Disparos: Este efecto consiste en una serie de impactos rápidos y sucesivos distribuidos por el torso, simulando el impacto de una ráfaga de disparos.

Para incorporar los efectos hápticos en nuestro proyecto de Unreal Engine, comenzaremos conectando el retroceso al disparar la pistola.

Nos dirigimos al *Event Graph* del arma y agregamos el nodo *Play Haptic*, que se encargará de activar la vibración háptica al disparar. Este nodo debe colocarse inmediatamente después del *Shot Channel* del sistema háptico de ProTubeVR, como se muestra en la Figura 3-69. En el nodo *Play Haptic*, asignamos el nombre del efecto previamente guardado en *bHaptics Developer*, asegurándonos de que coincida exactamente con el nombre con el que fue importado.

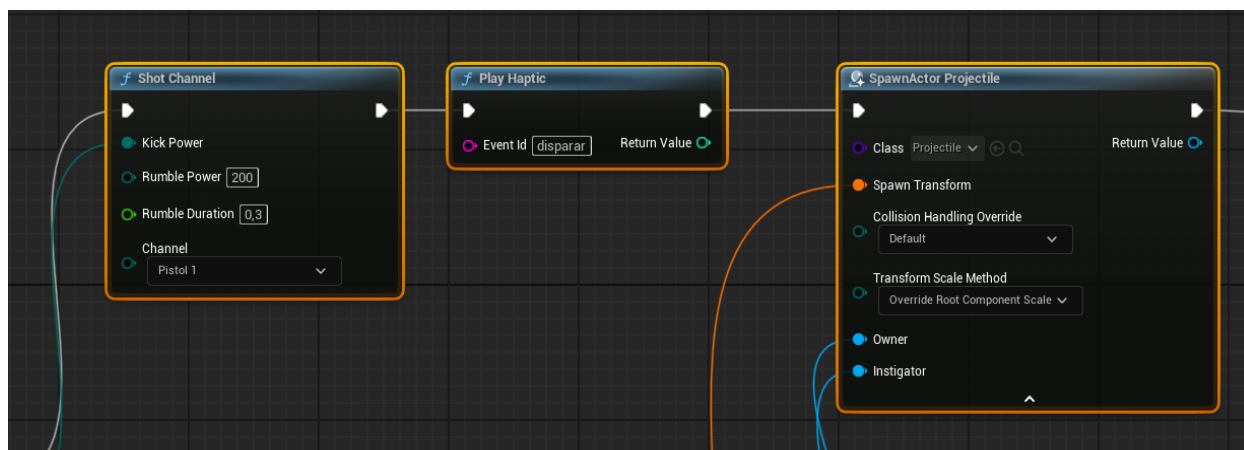


Figura 3-69 Código para añadir retroceso al disparar con la pistola.

Ahora continuaremos con la sección del chaleco háptico, donde previamente se habían colocado cuatro botones, cada uno asignado a un efecto específico. Para completar la integración, conectaremos el efecto háptico correspondiente a cada botón, asegurando que al presionarlos se active la vibración programada. Como se muestra en la Figura 3-70, el nodo *Play Haptic* se acopla después de que el movimiento del botón se haya completado, garantizando una transición fluida en la activación del efecto.

En este caso, hemos vinculado el botón al efecto de explosión, utilizando el nombre exacto del efecto guardado en *bHaptics Developer*. Este mismo proceso se repetirá para los demás botones, asignando a cada uno el nombre del efecto correspondiente.

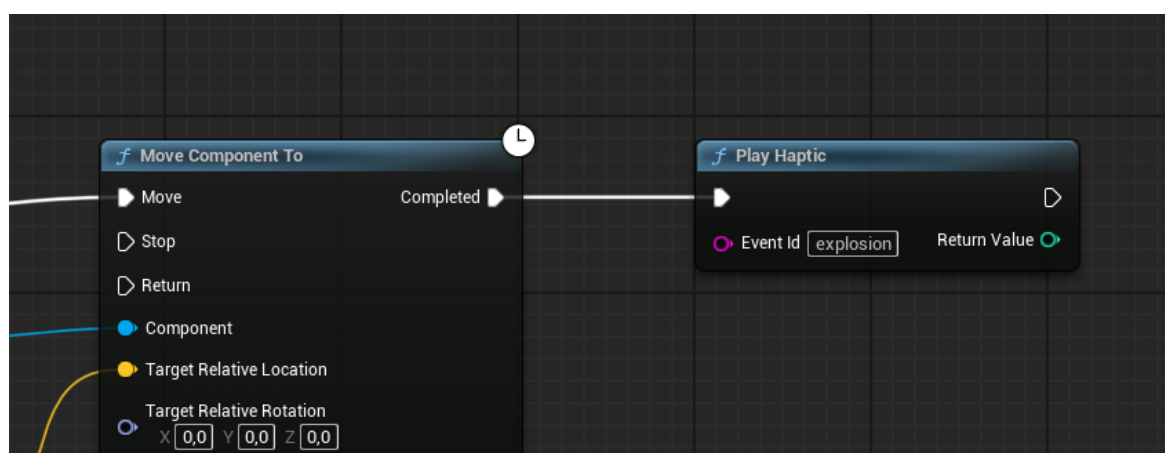


Figura 3-70 Código para activar la retroalimentación háptica de explosión

3.4.3 Plataforma VR

Para conectar la plataforma *KATVR*, el primer paso es asegurarse de que esté correctamente conectada a la fuente de alimentación. Además de la plataforma, se requiere el uso de los zapatos específicos de *KATVR*, los cuales están diseñados con ruedas para permitir el desplazamiento sobre la base sin necesidad de moverse físicamente en el mundo real.

Junto con los zapatos, es fundamental el sensor de movimiento integrado en cada uno de ellos, los cuales deben ser conectados a la plataforma junto con el sensor de dirección. Ambos componentes necesitan ser cargados cuando la plataforma está enchufada a la corriente, garantizando su correcto funcionamiento. Como se muestra en la Figura 3-71, estos sensores permiten un seguimiento preciso del movimiento del usuario dentro del entorno virtual.

Finalmente, para completar la configuración, es necesario conectar la plataforma al ordenador mediante un cable USB, lo que permitirá la comunicación entre el sistema y el software de realidad virtual.

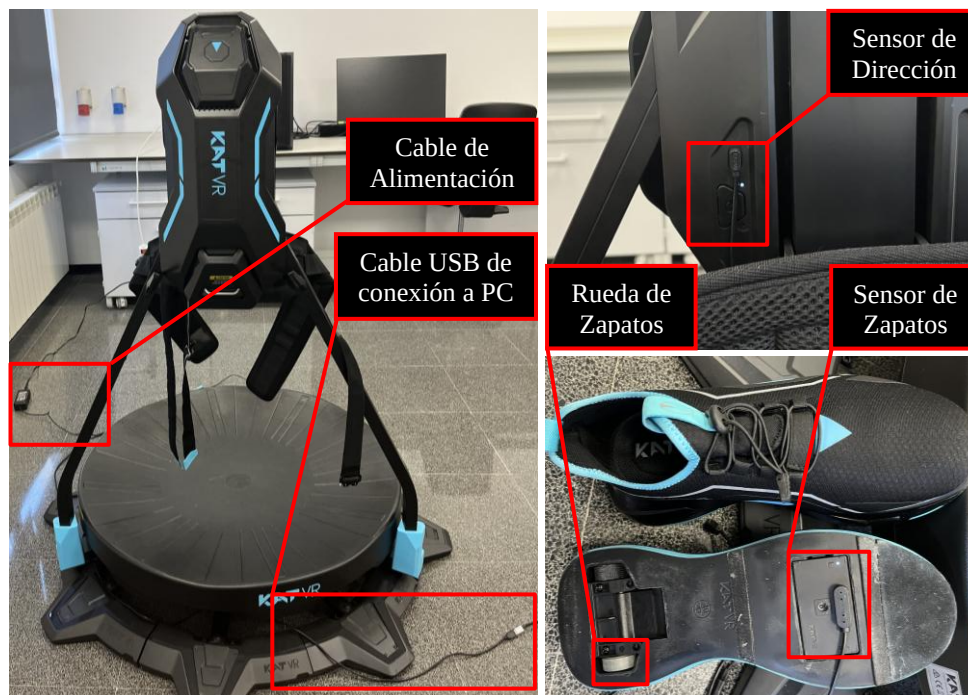


Figura 3-71 Plataforma KATVR Walk C2 Core y KATVR Shoes

Ahora se continua con la descarga del *KAT Gateway* desde la página oficial de KATVR, como se muestra en la Figura 3-72. Este software es fundamental para comprobar si la plataforma funciona correctamente. Además, se dispondrá de un manual de uso del Gateway, el cual deberá ser leído antes de la instalación para garantizar una configuración adecuada. Este manual servirá como referencia para futuras consultas en caso de dudas o necesidad de ajustes durante el uso de la plataforma.

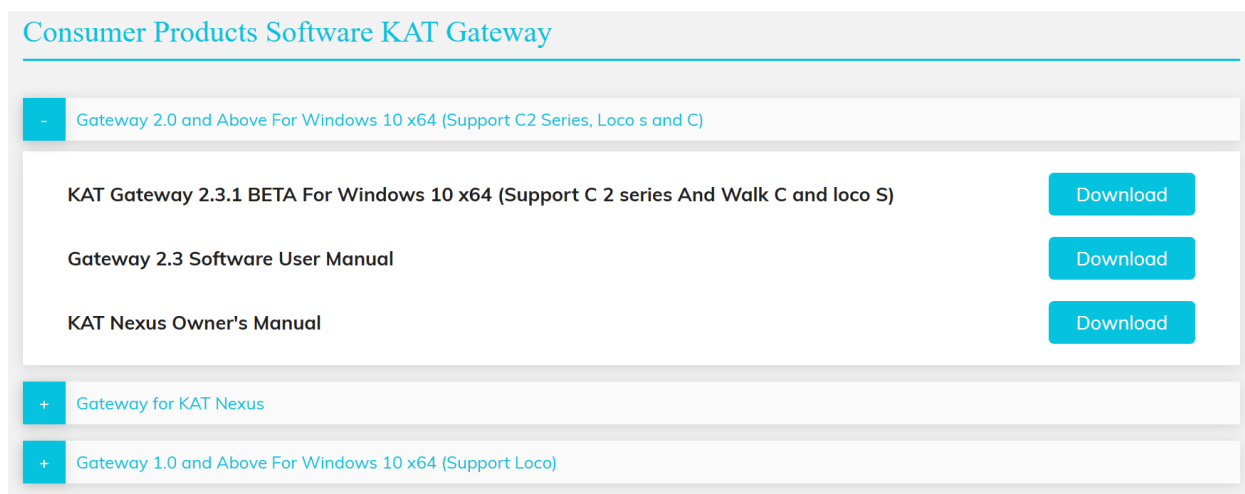


Figura 3-72 Descarga de KAT Gateway.

Tras la instalación, será necesario conectar la plataforma al PC, calibrar los sensores y utilizar las herramientas de prueba incluidas en el software para verificar su correcto funcionamiento. En la Figura 3-73 se muestra la interfaz del KATVR Gateway, en la que aparecen los sensores en verde, ya que están correctamente calibrados. Además, aportan todo tipo de información entre la que destaca el ángulo de orientación, y la posición de los zapatos.

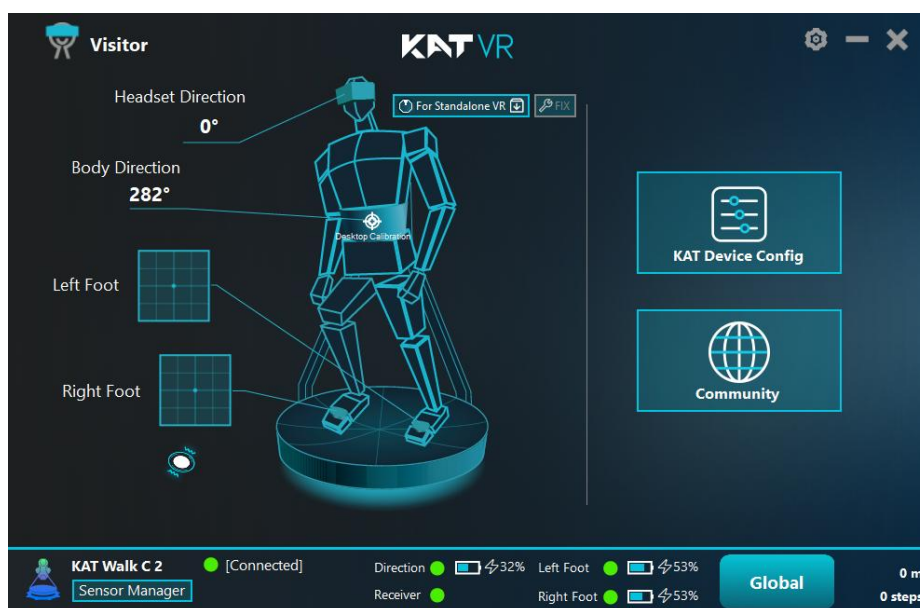


Figura 3-73 Interfaz de KATVR Gateway

Una vez verificada la funcionalidad de la plataforma y su correcta conectividad con el ordenador, procederemos a la instalación del plugin compatible con nuestra versión de Unreal Engine, como se muestra en la Figura 3-74. Este paso es esencial para garantizar la integración de la plataforma con el motor gráfico y permitir su correcto funcionamiento dentro del entorno virtual.

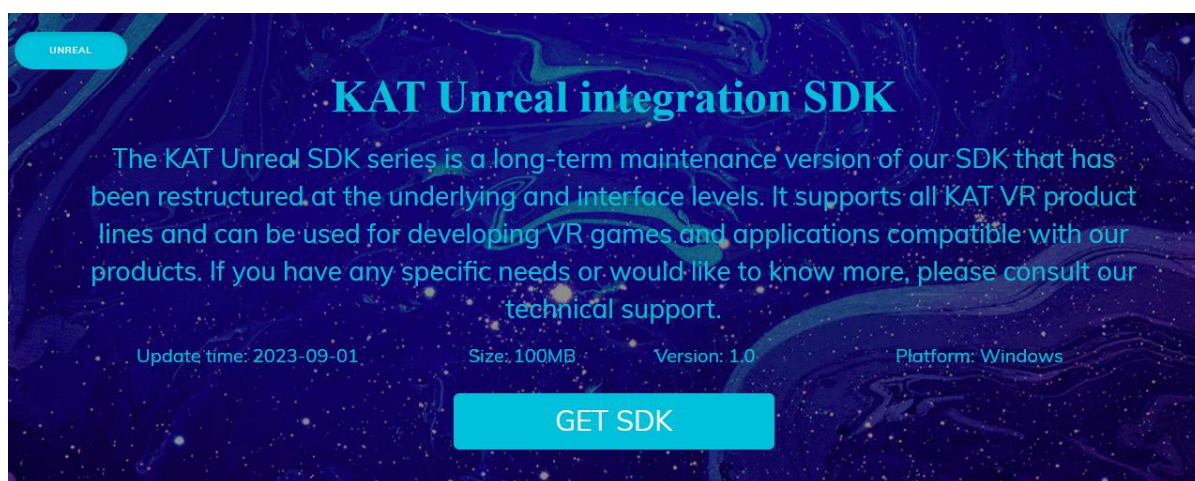


Figura 3-74 Plugin de KATVR compatible con Unreal Engine

Tras descargar la versión del plugin que, en teoría, era compatible con nuestra versión de Unreal Engine, se detectó un error en el código que impedía su correcta compilación. El problema se encontraba específicamente en el *KATDemoCharacterController*, dentro del código de calibración. Concretamente, el nodo *Get Orientation and Position* no compilaba correctamente, lo que generaba fallos en la integración del plugin con el motor.

Para solucionar este inconveniente, recompilamos el plugin, adaptándolo a nuestra versión de Unreal Engine. Tras realizar esta modificación, volvimos a abrir el editor y verificamos que los errores de compilación habían desaparecido, permitiendo así la correcta ejecución del sistema.

El plugin incluía una guía de integración de plataformas KAT VR en Unreal Engine. Sin embargo, lo primero que llamó la atención fue que la documentación solo hacía referencia a los modelos *KAT Walk Mini S* y *KAT Walk 3DT*, sin mencionar en ningún momento el modelo de nuestra plataforma. Curiosamente, al revisar el plugin de integración para Unity, que es otra plataforma de desarrollo,

encontramos que sí incluía soporte explícito para nuestro modelo. A pesar de esta discrepancia, decidimos continuar siguiendo los pasos de la guía para Unreal Engine. Al avanzar en la documentación, observamos que el proceso de integración estaba diseñado para las plantillas *First Person* o *Third Person*, lo que resultaba poco adecuado para nuestro caso, ya que trabajamos con una plantilla de VR. No obstante, optamos por continuar con la integración, adaptando los pasos según fuera necesario.

Por último, siguiendo las indicaciones de la guía, accedimos a la configuración de *World Settings* en Unreal Engine y realizamos los ajustes necesarios para la integración de la plataforma *KAT VR*. En *GameMode Override*, establecimos *KATDemoGameMode*, mientras que en *Default Pawn Class* asignamos *KATDemoCharacter*. Asimismo, en *Player Controller Class*, configuramos *KATDemoCharacterController*, tal como se muestra en la Figura 3-75.

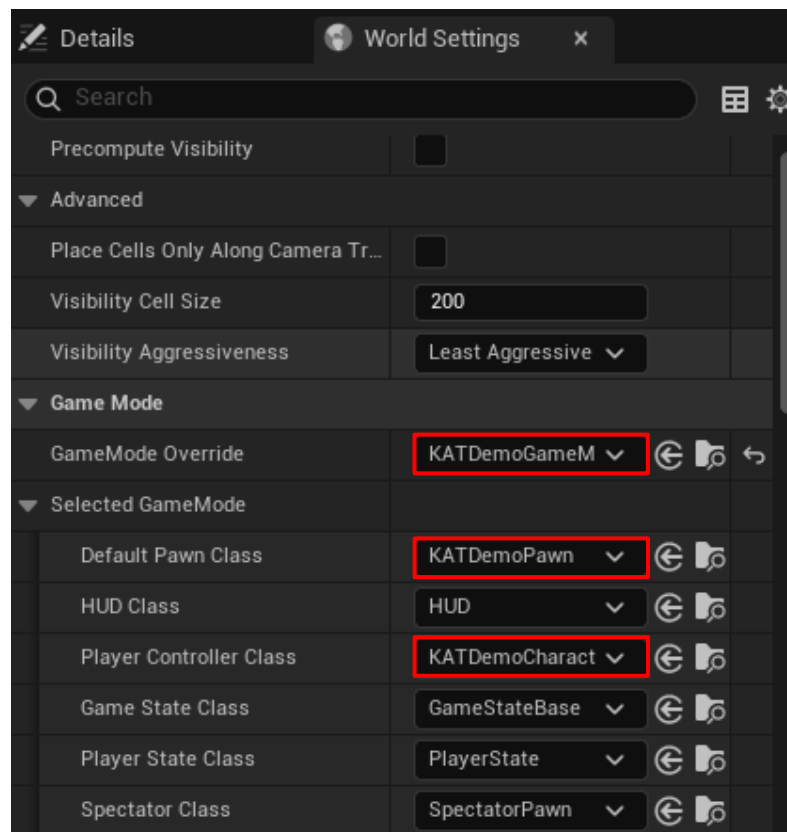


Figura 3-75 Panel de *World Settings* de Unreal Engine

4 RESULTADOS

4.1 Descripción del apartado.

En este apartado se expondrán los resultados obtenidos tras el desarrollo del proyecto, centrándose en el diseño final del entorno virtual y la interacción del usuario con sus elementos. Se analizarán las sensaciones percibidas, la respuesta del sistema y el impacto de la integración de dispositivos en la experiencia inmersiva.

4.2 Escenario Final

El resultado final del escenario puede observarse en la Figura 4-1. Se aprecia una distribución clara y estructurada de las distintas zonas, con una combinación de colores vivos que aportan un fuerte componente visual y facilitan la orientación dentro del entorno. La iluminación aplicada contribuye a generar una atmósfera realista, añadiendo profundidad y mejorando la percepción espacial. En conjunto, el diseño logrado resulta vistoso, intuitivo y en línea con el objetivo principal del proyecto: ofrecer un entorno virtual claro y funcional para la interacción del usuario.

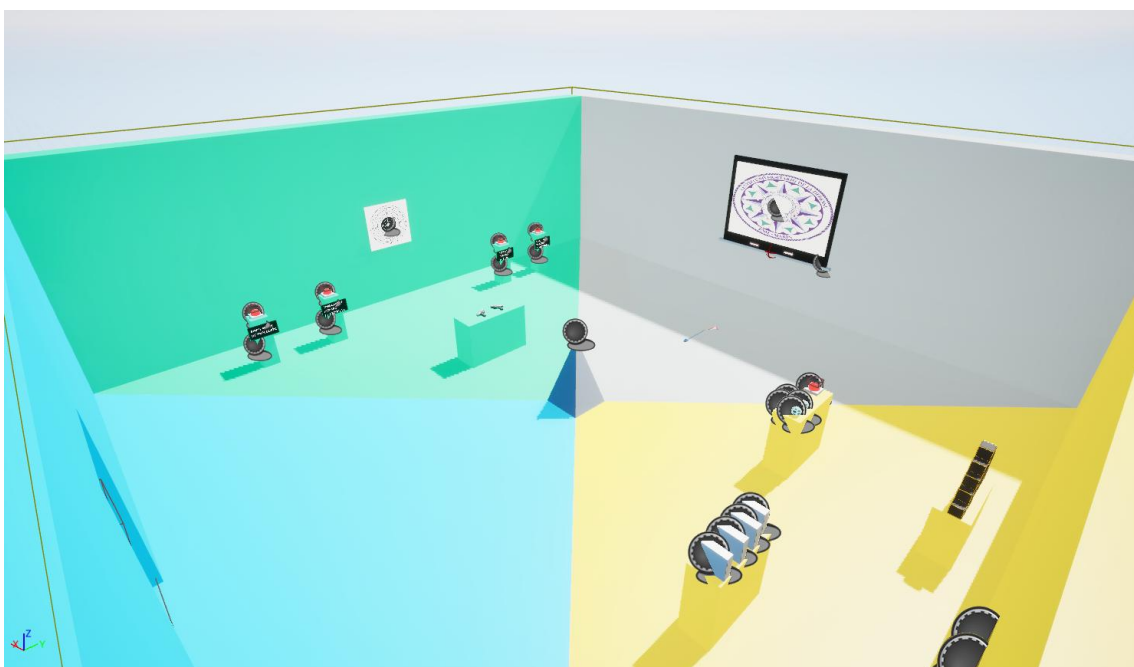


Figura 4-1 Resultado final del escenario

4.3 Entorno de Interacción.

En la Figura 4-2 se muestra el escenario de interacción, compuesto por dos secciones diferenciadas. La de interacción, permite manipular objetos y comprobar su comportamiento en el entorno virtual. La del puzle, requiere colocar las piezas en la posición y orientación correctas para que queden ancladas.



Figura 4-2 Entorno de Interacción

4.3.1 Sección de Colision

Pasando a la simulación en la sección de interacción, en la Figura 4-3 se muestra la vista inicial desde el entorno de realidad virtual. En ella pueden apreciarse las manos virtuales del usuario, lo que indica que la inmersión ha sido correctamente establecida. También se observan los elementos principales de esta sección: el botón interactivo, varias esferas y una estructura de cubos dispuesta en forma piramidal.

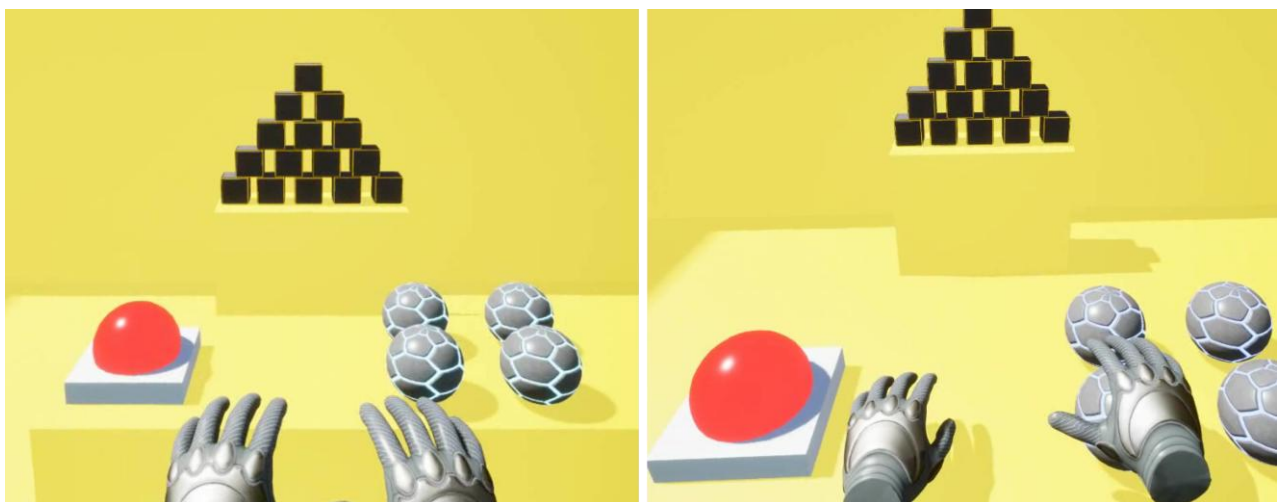


Figura 4-3 Vista VR de sección de colisión

Tal como se muestra en la Figura 4-4, es posible agarrar las pelotas del entorno, incluso una con cada mano simultáneamente. Para ello, basta con acercar cualquiera de las manos virtuales a una de las bolas y presionar el botón a la altura del dedo de medio en el controlador de realidad virtual, lo que activa el sistema de agarre.

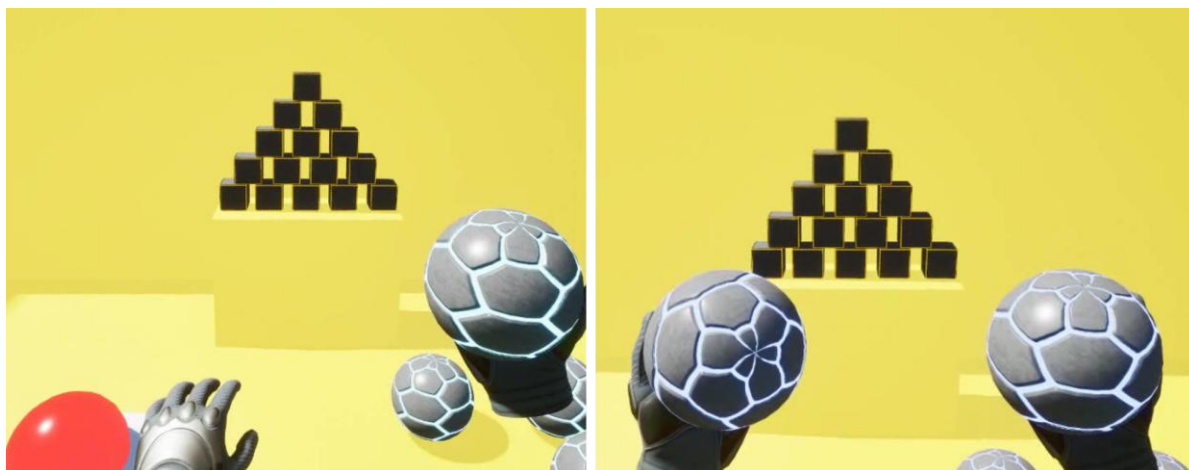


Figura 4-4 Vista VR de agarrar pelotas

En la Figura 4-5 se observa el resultado de lanzar una de las pelotas contra la estructura de cubos, provocando su colapso parcial tras el impacto. Para realizar esta acción, el usuario debe ejecutar el gesto físico de lanzamiento mientras mantiene presionado el botón de agarre del controlador. En el momento justo previo a soltar la pelota, es necesario liberar dicho botón para que el sistema interprete la acción como un lanzamiento.

Cabe destacar que este movimiento requiere cierta práctica, ya que, de forma instintiva, el usuario tiende a soltar el botón con retraso, lo que reduce la velocidad del proyectil. Como consecuencia, la esfera puede no alcanzar la trayectoria deseada y caer rápidamente por efecto de la gravedad.

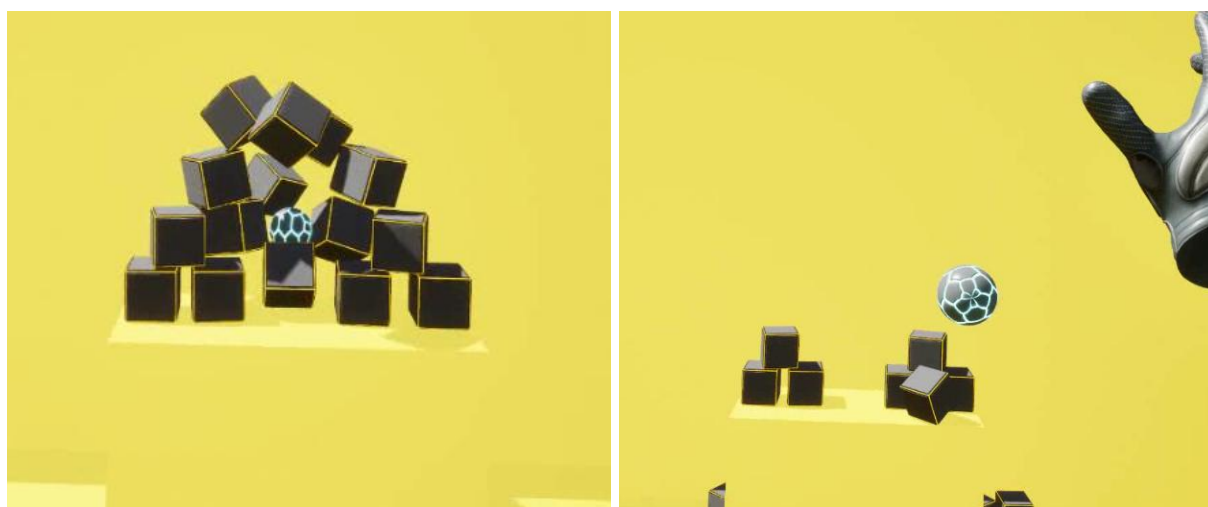


Figura 4-5 Vista VR de lanzamiento e impacto de pelotas

Una vez efectuados los primeros lanzamientos con las cuatro pelotas disponibles al inicio de la simulación, es habitual que el usuario no logre derribar la totalidad de los cubos en sus primeros intentos. Esto se debe, principalmente, a la necesidad de familiarizarse con la mecánica de lanzamiento en realidad virtual.

Para mantener la continuidad de la experiencia interactiva, el usuario puede pulsar el botón rojo situado que aparece en la Figura 4-6, lo que genera automáticamente una nueva pelota en el escenario, lista para ser utilizada. Esta funcionalidad permite repetir el proceso de lanzamiento tantas veces como se desee.

Gracias a este sistema, el usuario puede continuar experimentando y perfeccionando la técnica de lanzamiento hasta conseguir derribar completamente la estructura de cubos.

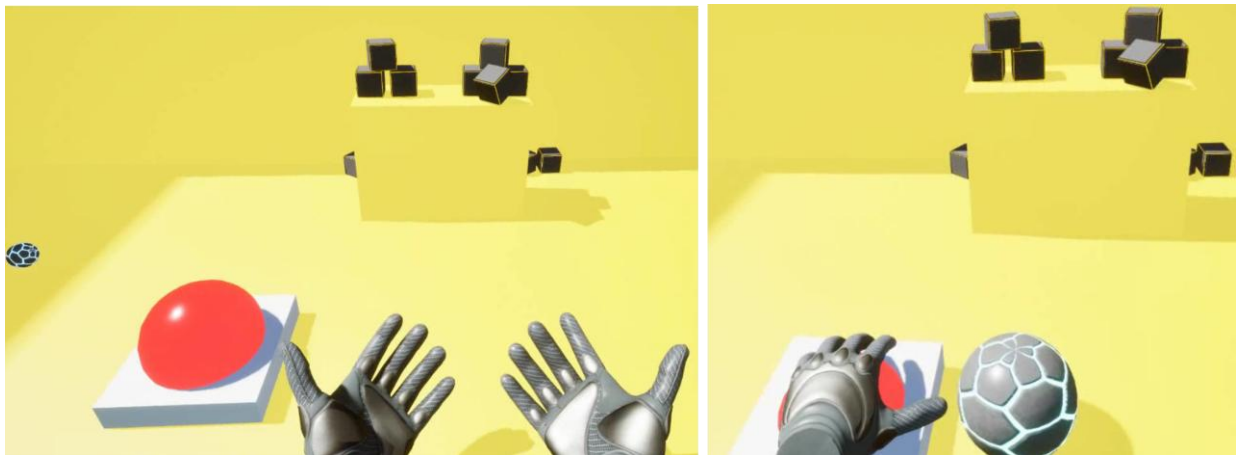


Figura 4-6 Vista VR de reaparición de pelotas

Finalmente, en la figura se muestra la interacción con los cubos, ya que estos actúan como lo harían en el mundo real.

Finalmente, en la Figura 4-7, se muestra la interacción con los cubos, los cuales se comportan de forma coherente, como lo harían en el mundo real

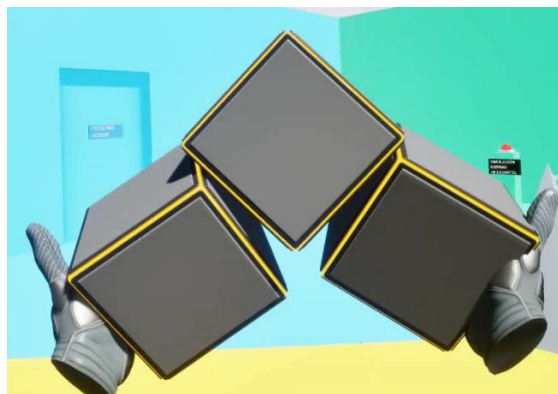


Figura 4-7 Vista VR de comportamiento de los cubos

4.3.2 Sección de Puzle

Comenzando con la sección del puzle, en la Figura 4-8 se muestra la vista VR inicial. En esta, se observan las cuatro piezas que conforman el puzle, además del marco que actuará como zona de anclaje.

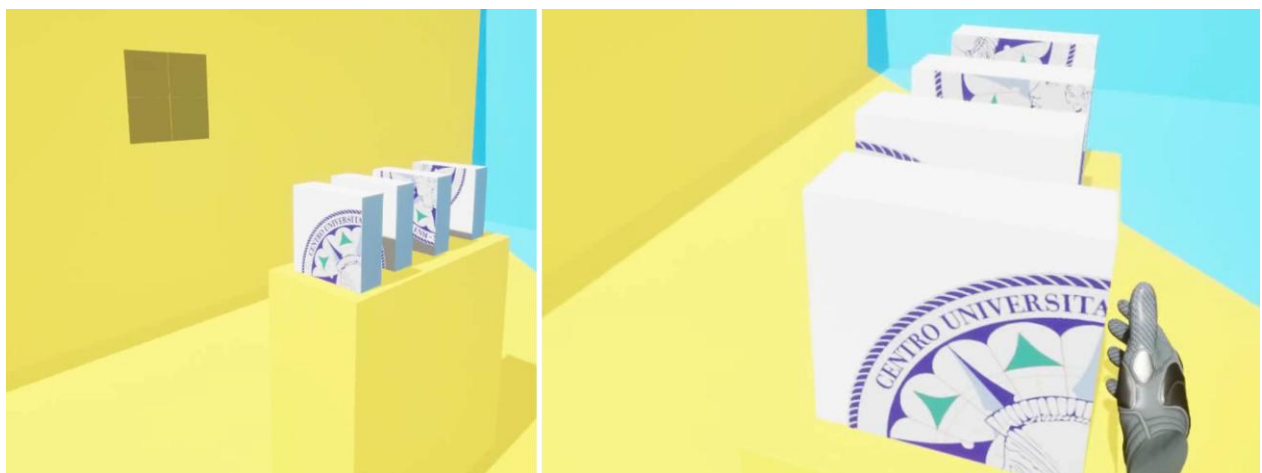


Figura 4-8 Vista VR de sección del puzle

A continuación, se procede a coger una de las piezas del puzle, de la misma manera que cogíamos las pelotas, para intentar anclarla en el marco. Tal como se muestra en la Figura 4-9, la pieza se ha colocado en la posición correcta, pero con una orientación incorrecta. Como resultado, el sistema no reconoce una coincidencia válida y la pieza no queda anclada en su lugar. Esta condición obliga al usuario a ajustar tanto la posición como la rotación de cada pieza para lograr una colocación precisa.

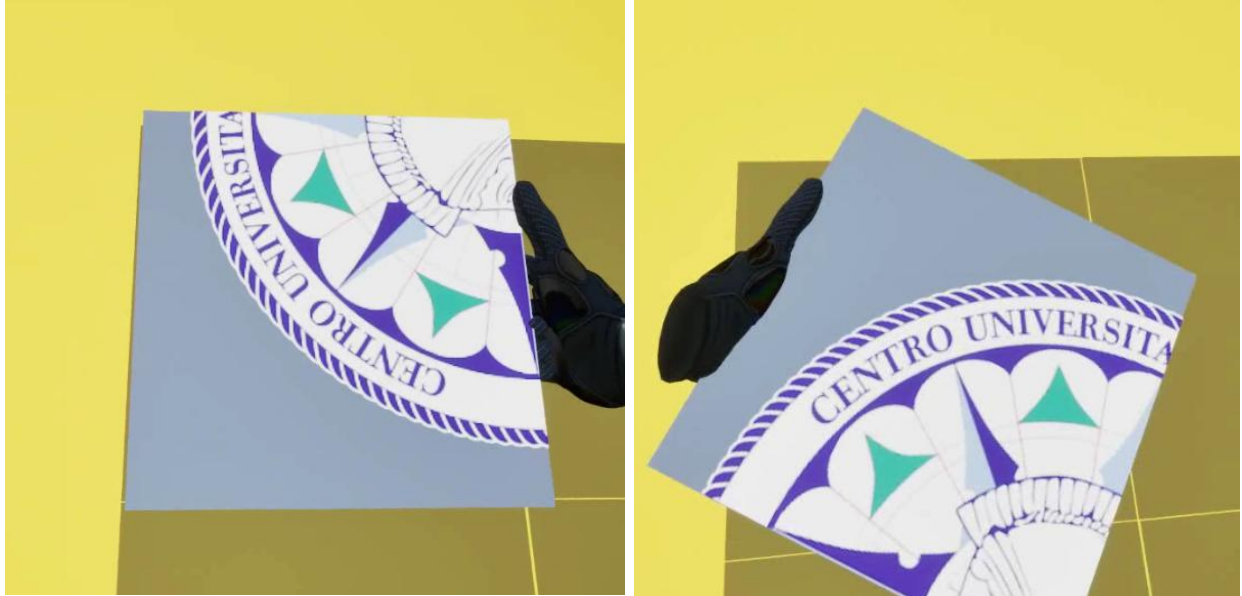


Figura 4-9 Vista VR de orientación incorrecta de la pieza del puzle

En la Figura 4-10 se muestra cómo, una vez ajustada correctamente la orientación de la pieza, esta queda anclada en su ubicación dentro del marco. El sistema valida tanto la posición como la rotación del objeto, permitiendo su fijación únicamente cuando ambas condiciones se cumplen con precisión. Esta funcionalidad garantiza una interacción más realista y fomenta la atención al detalle por parte del usuario.

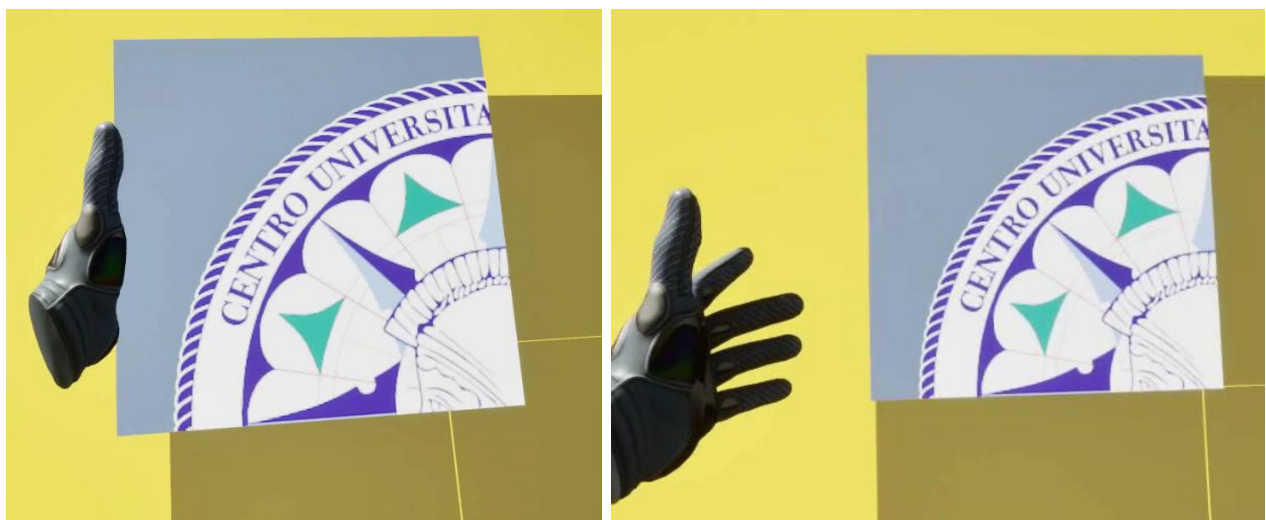


Figura 4-10 Vista VR de anclado de pieza del puzle

En la Figura 4-11 se observa que, aunque una pieza del puzle tenga la orientación correcta, si no se encuentra en la posición asignada dentro del marco, no quedará anclada

Una vez ubicada correctamente en ambos parámetros, la pieza se ancla automáticamente, contribuyendo a la reconstrucción progresiva de la imagen completa.

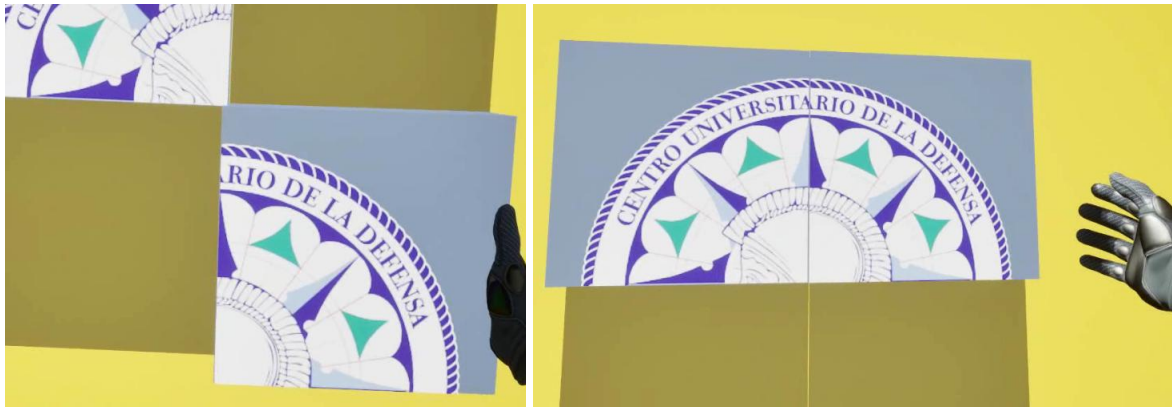


Figura 4-11 Vista VR de anclado de pieza en la posición correcta

Finalmente, en la Figura 4-12 se muestra el resultado obtenido tras anclar correctamente las cuatro piezas del puzle.



Figura 4-12 Puzle completo

4.4 Entorno de Retroalimentación Háptica.

En la Figura 4-13 se muestra el entorno de retroalimentación háptica, compuesto por la sección de disparo con pistolas y la zona de efectos asociados al chaleco háptico.

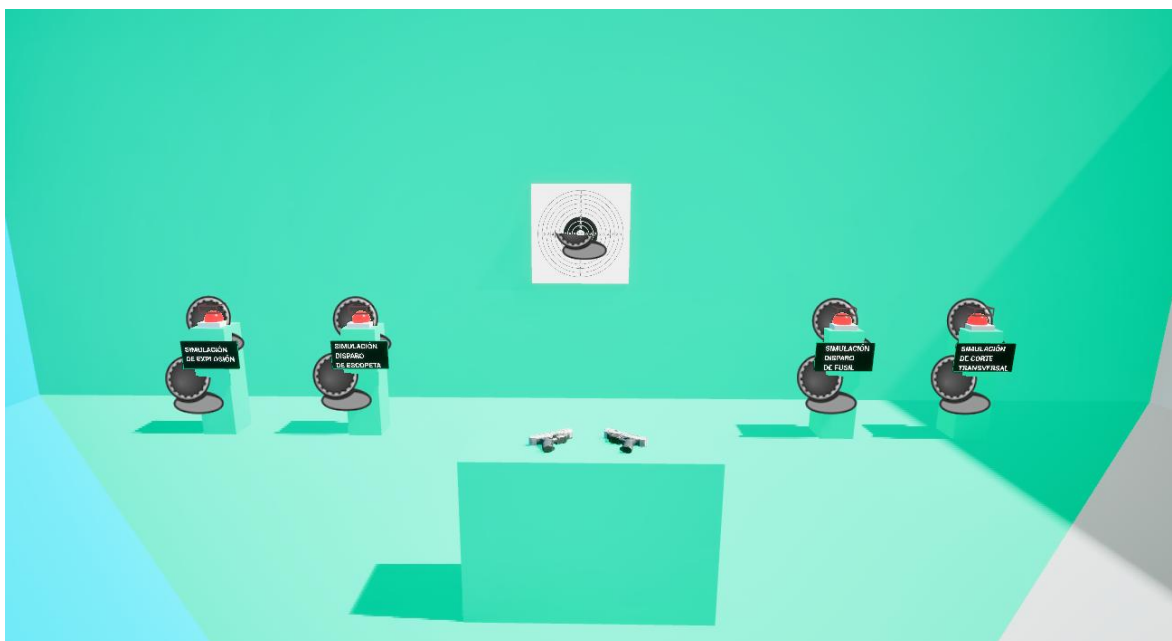


Figura 4-13 Entorno de retroalimentación háptica

Antes de iniciar la inmersión en esta sección, es necesario equiparse adecuadamente con los dispositivos hápticos, como se muestra en la Figura 4-14. Para ello, se introduce el mando de realidad virtual en el soporte de la pistola háptica de ProTubeVR, y se coloca el chaleco háptico de bHaptics sobre el torso. Ambos dispositivos deben haber sido encendidos y correctamente conectados con anterioridad, garantizando así su funcionalidad durante la simulación.



Figura 4-14 Usuario equipado correctamente para sección de retroalimentación háptica

4.4.1 Sección de Pistola

Comenzando con la sección de pistolas, en la Figura 4-15 se muestran las dos armas disponibles en el escenario virtual. El usuario puede coger cualquiera de ellas, de la misma manera que en el entorno de interacción, ya sea con una mano, o incluso utilizar ambas simultáneamente, una en cada mano.



Figura 4-15 Vista VR de pistolas

A continuación, tras coger una de las pistolas, se procede a apuntar hacia la diana situada en el fondo del escenario, tal como se muestra en la Figura 4-16. El disparo se realiza presionando el botón del controlador correspondiente al dedo índice, replicando la acción natural de un arma real.

Al efectuar el disparo, se percibe el retroceso háptico programado en la pistola, acompañado de una vibración sincronizada en el chaleco bHaptics, que intensifica la sensación de inmersión. Además, los proyectiles impactan sobre la diana y permanecen temporalmente visibles en el punto de impacto, lo que permite al usuario comprobar la precisión de sus disparos en tiempo real.



Figura 4-16 Vista VR de disparo con pistola

Además de la diana, la pistola puede utilizarse para disparar a cualquier elemento del escenario, como se aprecia en la Figura 4-17. Los proyectiles interactuarán con otros objetos, como las pelotas o los cubos, generando impactos visibles que pueden derribarlos. Esta funcionalidad amplía las posibilidades de interacción dentro del entorno virtual y refuerza la sensación de realismo durante la simulación.



Figura 4-17 Vista VR de disparo con pistola a objetos del escenario

4.4.1 Sección de Chaleco Háptico

Siguiendo con la sección dedicada a los efectos del chaleco háptico, en la Figura 4-18 se muestra uno de los soportes con botón, que permite activar una simulación concreta. Cada soporte está identificado mediante un cartel que indica el tipo de efecto háptico que el usuario experimentará al pulsarlo.

Al presionar el botón, el chaleco reproduce el patrón de vibración correspondiente al efecto indicado, entre los que se incluyen el impacto de fusil, disparo de escopeta, explosión y corte transversal. Todos los efectos han sido configurados para proporcionar una experiencia inmersiva, transmitiendo sensaciones realistas sin resultar molestas ni intrusivas para el usuario.



Figura 4-18 Vista VR de botón para sentir efecto háptico

4.5 Pirámide Central

El objetivo principal de la pirámide central era el de servir como superficie de visualización de objetos. Tal y como se muestra en la Figura 4-19, se despliega un menú flotante mediante los botones asociados al dedo pulgar. Una vez abierto el menú, la selección del objeto deseado se realiza utilizando la otra mano y presionando el botón correspondiente al dedo índice. Esta mecánica facilita una interacción natural y fluida dentro del entorno virtual.



Figura 4-19 Vista VR de menú desplegable para la visualización de objetos

Tal y como se muestra en la Figura 4-20, el usuario puede seleccionar cualquiera de los objetos disponibles en el menú para visualizarlos sobre la pirámide central. En caso de querer dejar la superficie libre, simplemente se selecciona la opción *Vacío*, lo que elimina cualquier objeto previamente mostrado. Esta funcionalidad proporciona un control sencillo y eficaz sobre el contenido visualizado.

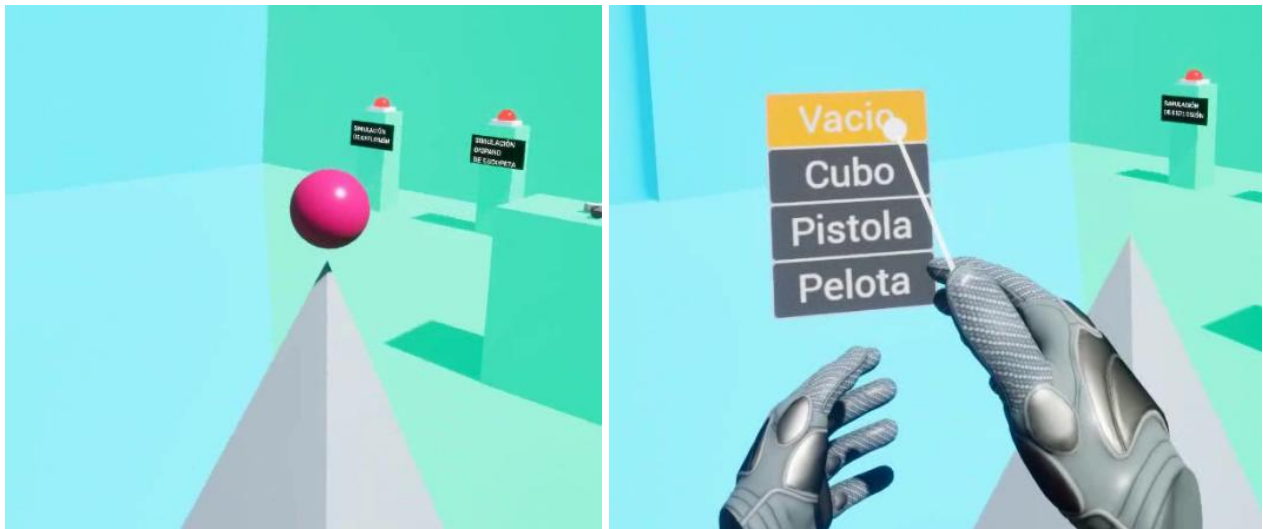


Figura 4-20 Vista VR de selección de objetos y vaciar pirámide

4.6 Pizarra Digital

En la Figura 4-21 se puede apreciar la funcionalidad interactiva de la pizarra digital. A través de los dedos de realidad virtual, es posible interactuar de forma táctil con los botones situados en la parte inferior de la pantalla. Al tocar cualquiera de las flechas con el dedo virtual, se produce un cambio de diapositiva, permitiendo avanzar o retroceder entre las diferentes transparencias disponibles.

Además, esta interacción se ve reforzada por una ligera vibración generada por los controladores VR, la cual proporciona una respuesta háptica que mejora la sensación de realismo durante el uso del sistema.

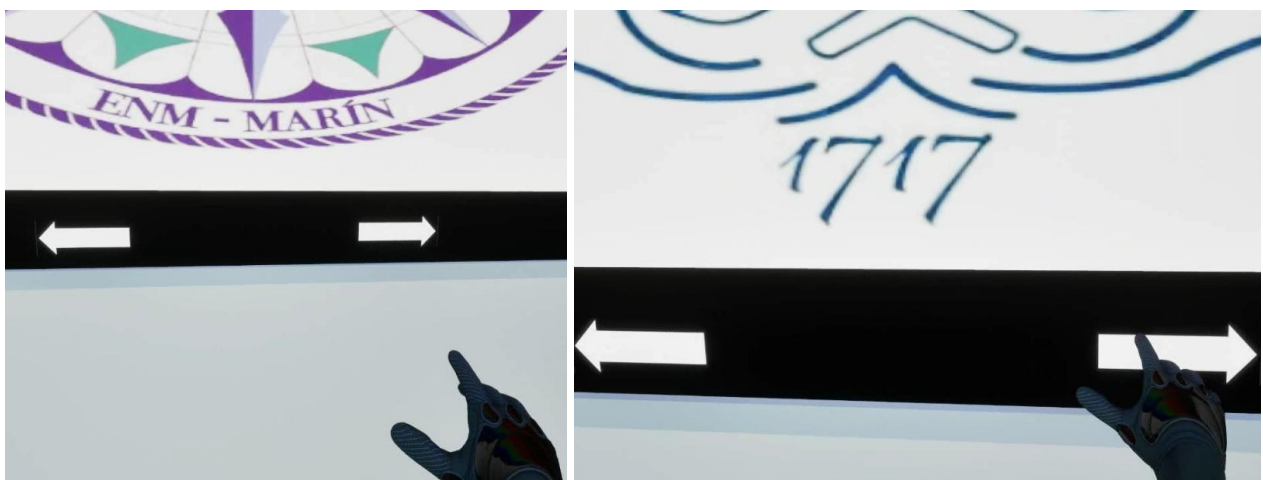


Figura 4-21 Vista VR de cambio de transparencia

Tal y como se muestra en la Figura 4-22, la pizarra digital no solo permite la interacción táctil con sus elementos, sino que también puede ser desplazada libremente por el entorno virtual. De forma similar a otros objetos descritos anteriormente, es posible agarrarla utilizando el mismo sistema de interacción. Una vez sujeta, el usuario tiene la posibilidad de moverla por todo el escenario, ajustando su ubicación en función de las necesidades del momento o del área de trabajo que desee utilizar.



Figura 4-22 Vista VR de coger pizarra

Gracias a esta funcionalidad, es posible desplazar la pizarra digital por el entorno y soltarla en cualquier ubicación del escenario, tal y como se observa en la Figura 4-23. Esto permite al usuario colocar la pantalla en la posición que considere más cómoda o accesible para continuar la interacción, adaptando así el entorno virtual a sus preferencias o necesidades específicas.



Figura 4-23 Vista VR de pizarra movida

Posteriormente, si se desea devolver la pizarra a su ubicación original para evitar que interfiera en otras partes del escenario, basta con agarrarla y acercarla a su soporte. Una vez posicionada correctamente, el sistema detecta la proximidad y activa automáticamente el anclaje, lo que permite que la pantalla quede fijada de forma precisa en su lugar, tal y como se muestra en la Figura 4-24.



Figura 4-24 Vista VR de anclaje de pizarra

4.7 Entorno de plataforma VR

En la Figura 4-25 se presenta el entorno diseñado específicamente para el uso de la plataforma de realidad virtual. Este espacio cuenta con una puerta de acceso, a través de la cual el usuario podrá ingresar a un pasillo alargado. La disposición de este pasillo ha sido concebida para maximizar el aprovechamiento de las capacidades de movilidad ofrecidas por la plataforma, permitiendo al usuario desplazarse de forma natural dentro del entorno virtual.

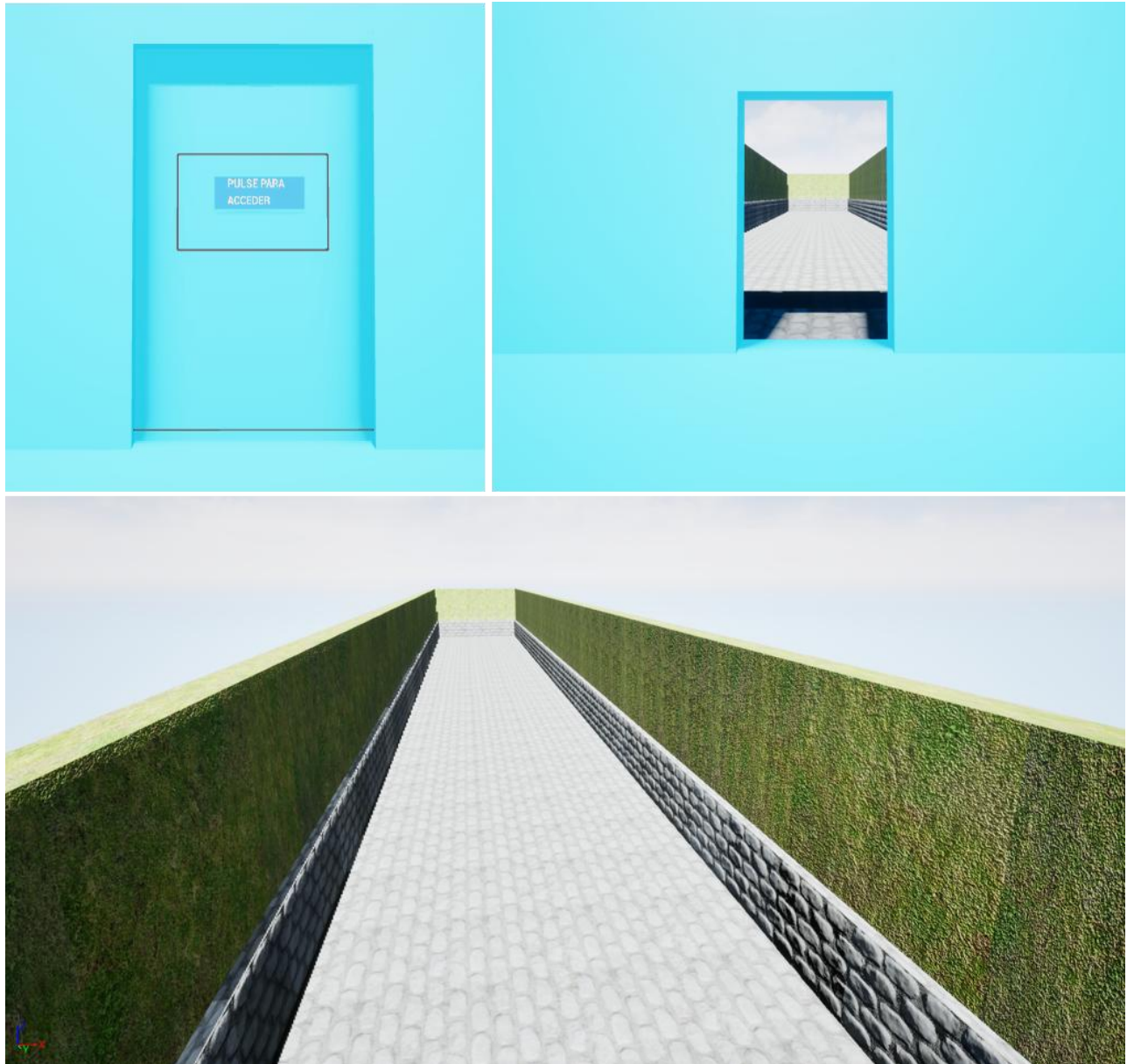


Figura 4-25 Entorno de Plataforma VR

Antes de iniciar la inmersión en esta sección del entorno, es imprescindible que el usuario se equipe de forma adecuada para garantizar la seguridad y el correcto funcionamiento de la plataforma de realidad virtual, como se muestra en la Figura 4-26.

Para ello, debe calzarse correctamente el calzado específico de KAT VR, diseñado para su uso sobre la superficie de la plataforma. A continuación, debe subirse a la plataforma y asegurarse mediante las correas ajustables situadas a la altura del abdomen, las cuales proporcionan estabilidad y permiten un desplazamiento controlado dentro del entorno simulado.



Figura 4-26 Usuario sobre plataforma VR

Tras la instalación del plugin, su configuración siguiendo la guía proporcionada, la conexión de la plataforma y una comprobación previa en KAT Gateway, se procedió a evaluar su funcionamiento dentro del escenario de simulación. Al iniciar la simulación, únicamente se visualizaba el entorno desde un punto fijo que no coincidía con el *Player Start*, y no se detectaba ningún tipo de respuesta a estímulos de movimiento. Esto evidenciaba un fallo en la integración, lo cual resultaba esperable considerando las discrepancias ya mencionadas previamente en el desarrollo.

Con el objetivo de confirmar si la guía era errónea, se creó un nuevo proyecto basado en la plantilla *First Person*, al que también se le instaló el plugin siguiendo los pasos de integración indicados. Sin embargo, al ejecutar la simulación, se observó nuevamente que el entorno permanecía inmóvil, sin reacción alguna. Este resultado sugería que, o bien la guía contenía información incorrecta, o bien el modelo de la plataforma utilizada no era compatible con Unreal Engine.

Pese a ello, se continuó investigando y realizando pruebas. Uno de los avances clave se obtuvo al modificar la clase asignada como *Default Pawn Class*, estableciendo el valor *VRPawn*, como se muestra en la Figura 4-27. Esta configuración, aunque no indicada explícitamente en la guía, resultó lógica en el contexto del proyecto. Tras este ajuste, se reinició la simulación, y esta vez sí se observó una respuesta del sistema.

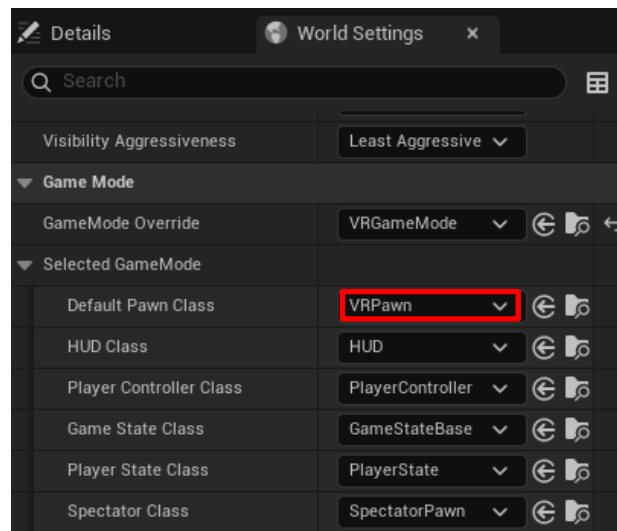


Figura 4-27 Panel de *World Settings* en Unreal Engine

No obstante, al probar el cambio de orientación sobre la plataforma, se detectó un comportamiento anómalo: no solo se modificaba la orientación dentro del entorno virtual, sino que también se producía un desplazamiento no deseado. Ante esta situación, se decidió inspeccionar el código del *KATVRCharacter*, centrándose en el apartado relativo a la calibración, ya que se sospechaba que el origen del problema podría estar allí.

Para verificarlo, se introdujeron nodos de *Print String* en distintas variables, especialmente aquellas relacionadas con la orientación, con el fin de observar sus variaciones durante el uso. Gracias a esta estrategia, se confirmó que, aunque la calibración parecía correcta en el KAT Gateway, no lo era dentro del entorno de Unreal Engine. Se observó que, al girar una vuelta completa sobre la plataforma real, el entorno virtual registraba aproximadamente dos vueltas completas.

A partir de este punto, se introdujeron modificaciones progresivas en el código y se reinició la simulación en repetidas ocasiones. A pesar del esfuerzo dedicado, el resultado final fue insatisfactorio, y no se logró establecer una conexión funcional y precisa entre la plataforma y el entorno de simulación.

A pesar de las limitaciones derivadas del funcionamiento incorrecto de la plataforma de realidad virtual, se procedió a verificar la operatividad del escenario diseñado. Para ello, nos acercamos a la puerta virtual con las manos, tal y como se muestra en la Figura 4-28, con el objetivo de activar el mecanismo programado que permite hacerla desaparecer y así acceder al pasillo interior del entorno inmersivo.

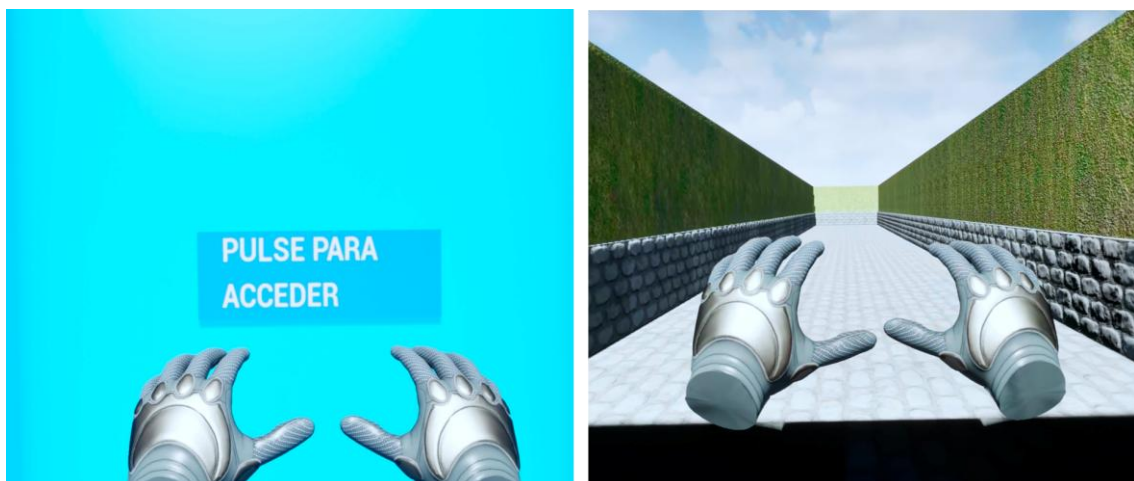


Figura 4-28 Vista VR de apertura del pasillo

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Descripción del apartado

En este apartado se evalúa si se han alcanzado los objetivos definidos al inicio del proyecto, determinando su grado de cumplimiento. Además, se plantean posibles líneas de desarrollo futuro orientadas a ampliar las capacidades del entorno de simulación, mejorar la experiencia del usuario e integrar nuevas funcionalidades que potencien su utilidad.

5.2 Conclusiones

5.2.1 Entorno de interacción

El entorno de interacción ha logrado cumplir con éxito los objetivos propuestos. La sección de colisión permite una manipulación precisa y fluida de objetos como las pelotas, reproduciendo de forma realista las acciones de coger, lanzar y soltar. Además, la incorporación del botón interactivo ha resultado clave, no solo en esta sección, sino como recurso reutilizable en otras partes del escenario, mejorando la interactividad global.

Por su parte, la sección del puzzle introduce un componente lógico que exige al usuario razonar mínimamente durante la interacción, aportando un valor añadido en términos de atención y comprensión espacial. En conjunto, esta área ofrece una experiencia completa, intuitiva y fácilmente ampliable.

5.2.2 Integración de periféricos

Se ha conseguido integrar de forma satisfactoria tanto el chaleco como la pistola háptica, permitiendo su uso conjunto o por separado dentro del entorno virtual. Este resultado confirma el cumplimiento del objetivo de integración de estos dispositivos. A pesar de ser productos de distintos fabricantes, el proceso de implementación ha seguido una lógica similar en ambos casos, lo que demuestra la viabilidad de incorporar otros sistemas hápticos en el futuro. La inclusión de esta tecnología supone un incremento notable en el realismo de la simulación, mejorando la sensación de inmersión y la percepción del usuario durante la experiencia virtual.

5.2.3 Pizarra digital

Tal y como se establecía en los objetivos, se ha implementado una pizarra interactiva que puede ser manipulada libremente dentro del entorno virtual. Su tamaño ha sido ajustado para corresponder a proporciones reales, y permite ser desplazada en cualquier parte del escenario, pudiéndose anclar de nuevo en su posición inicial. Además, se ha incorporado la funcionalidad táctil para modificar su transparencia, junto con un efecto háptico que refuerza la sensación de realismo durante la interacción.

5.2.4 Pirámide central

La pirámide central implementada en el escenario cumple con el objetivo de servir como visualizador interactivo de objetos. Su forma geométrica permite colocar elementos en su vértice de forma destacada, facilitando su observación desde distintos ángulos dentro del entorno virtual. Además, se ha desarrollado un menú que permite seleccionar y mostrar objetos de forma intuitiva, manteniendo un diseño claro, funcional y fácilmente ampliable para futuras integraciones.

5.2.5 Plataforma VR

Aunque como objetivo adicional se encontraba la integración de la plataforma de realidad virtual, este no ha podido alcanzarse. Las pruebas realizadas indican que el modelo específico de plataforma disponible no parece ser totalmente compatible con Unreal Engine, tal y como refleja la documentación del plugin. Este problema de integración podría deberse a errores en la programación del propio plugin, lo que ha dificultado su correcta implementación en el entorno. En cambio, la documentación correspondiente a otros motores de desarrollo, como Unity, confirma la compatibilidad con dicho modelo.

A pesar de esta limitación, el entorno está preparado para incorporar esta funcionalidad en cuanto se asegure la compatibilidad de la plataforma, lo que permitiría una integración inmediata sin necesidad de realizar modificaciones significativas.

5.2.6 Conclusión final

Este Trabajo de Fin de Grado ha cumplido ampliamente los objetivos inicialmente propuestos. No solo se han alcanzado las funcionalidades previstas, sino que, gracias a un proceso continuo de pruebas, se han implementado mejoras orientadas a optimizar la experiencia inmersiva. El resultado es un escenario pulido y realista, en el que cada detalle ha sido afinado para ofrecer una simulación convincente.

Uno de los objetivos, la integración de la plataforma KAT VR, no pudo completarse debido a problemas de compatibilidad con Unreal Engine. Aun así, esto abre una posible línea futura de desarrollo si se supera dicha limitación técnica.

Un aspecto especialmente valioso del proyecto ha sido el aprendizaje del sistema de programación visual *Blueprints*. Esta herramienta ha resultado mucho más intuitiva que el uso tradicional de C++, permitiendo desarrollar funcionalidades de forma visual, comprensible y accesible para quienes no provienen de un perfil puramente informático.

Además, una práctica fundamental durante el desarrollo ha sido la realización de simulaciones de prueba tras cada cambio implementado. Esta dinámica de comprobación continua ha permitido detectar errores, ajustar detalles y perfeccionar el entorno hasta lograr una experiencia inmersiva sólida y bien optimizada.

En conjunto, el proyecto ha servido como base sólida para futuras simulaciones inmersivas, aportando experiencia técnica y una visión clara del potencial de la realidad virtual en ámbitos como la formación y el entrenamiento profesional.

5.3 Líneas futuras

5.3.1 Integración de Plataforma KATVR

En cuanto a las líneas futuras, uno de los principales objetivos pendientes es la integración completa de la plataforma de locomoción KATVR en el entorno virtual. Una de las vías planteadas para lograr esta integración es la actualización oficial del plugin de KATVR. Esta opción requeriría esperar a una nueva versión que asegure la compatibilidad con el modelo de plataforma utilizado y con Unreal Engine.

De este modo, se mantendría la continuidad del trabajo ya desarrollado, evitando ajustes profundos en el entorno actual.

Otra posibilidad a considerar es la incorporación de la plataforma mediante un motor de desarrollo distinto, como Unity. Esta opción ha sido propuesta anteriormente y no se descarta para el futuro, ya que Unity ofrece compatibilidad directa con la plataforma KATVR según su documentación técnica. Sin embargo, no se plantea como una solución a corto plazo, dado que el laboratorio de RV actualmente está completamente adaptado al uso de Unreal Engine.

5.3.2 Evolución del proyecto

Una posible evolución del módulo base desarrollado consiste en incorporar la funcionalidad multiusuario, permitiendo la conexión simultánea de varios participantes en el mismo entorno virtual. Esto habilitaría escenarios colaborativos, especialmente útiles en entrenamientos de tipo militar o profesional donde la coordinación en equipo es fundamental. La inclusión de este enfoque no solo amplía el realismo de la simulación, sino que también potencia el aprendizaje mediante la cooperación, la comunicación y la toma de decisiones conjunta en tiempo real.

Otra línea prometedora es la ampliación del ecosistema háptico, incorporando nuevos dispositivos como guantes hápticos o trajes de cuerpo completo. Estos periféricos permitirían al usuario recibir retroalimentación táctil detallada en más puntos del cuerpo, lo que incrementaría significativamente el nivel de inmersión. Además, podrían integrarse sensores biométricos para registrar parámetros fisiológicos como el ritmo cardíaco o el nivel de estrés, generando simulaciones adaptativas y evaluaciones más precisas del desempeño bajo presión.

Finalmente, el módulo podría orientarse hacia la creación de escenarios específicos adaptados a distintos contextos de formación, lo que permitiría una simulación más precisa y útil según el perfil del usuario. Entre las posibles aplicaciones destacan las prácticas de tiro en galería, donde el usuario podría entrenar puntería, manejo de armas y tiempos de reacción en un entorno controlado. Asimismo, se podría desarrollar un escenario simulado del puente de gobierno de un buque, ideal para instruir en navegación, control de sistemas y toma de decisiones en situaciones operativas. Otra opción de gran interés sería la impartición de clases teórico-prácticas interactivas, donde los alumnos puedan interactuar con el entorno virtual, manipular elementos y colaborar entre sí, potenciando la comprensión de conceptos mediante la experimentación activa.

6 BIBLIOGRAFÍA

- [1] «Tecnologías de la industria 4.0», AuraQuantic. Accedido: 20 de enero de 2025. [En línea]. Disponible en: <https://www.auraquantic.com/es/tecnologias-industria-inteligente/>
- [2] J. Peralta Abarca, B. Martínez Baena y J. Enríquez Urbano, «Industria 4.0», Universidad Autónoma del Estado de Morelos. Accedido: 8 de enero de 2025. Disponible en: <https://inventio.uaem.mx/index.php/inventio/article/view/45/4>
- [3] «Pruebas de realidad extendida (XR)». Accedido: 20 de enero de 2025. [En línea]. Disponible en: https://www.rohde-schwarz.com/lat/soluciones/wireless-communications-testing/wireless-standards/5g-nr/extended-reality-xr-testing/pruebas-de-realidad-extendida-xr_258007.html
- [4] C. C. Morales, «Pokémon Go y el fenómeno que desató la realidad aumentada», Tvñ Panamá. Accedido: 21 de enero de 2025. [En línea]. Disponible en: https://www.tvn-2.com/contenido-exclusivo/pokemon-go-fenomeno-realidad-aumentada_1_1091773.html
- [5] «Realidad virtual». Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://www.fib.upc.edu/retro-informatica/avui/realitatvirtual.html>
- [6] «Videojuegos de realidad virtual », ADSLZone. Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://www.adslzone.net/noticias/videojuegos/videojuegos-realidad-virtual-top-famosos/>
- [7] «Realidad Extendida: El Futuro de las Tecnologías Inmersivas». Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://www.onirix.com/es/realidad-extendida/>
- [8] «Realidad mixta - Problemas y beneficios», Arsoft. Accedido: 21 de enero de 2025. [En línea]. Disponible en: <https://www.arsoft-company.com/noticia/realidad-mixta-problemas-y-beneficios>
- [9] D. Levis, «La tecnología que hace posible la Realidad Virtual», *Art. Rev. 1*, vol. 1, n.º 1, p 6-9, 2006.
- [10] J. A. P. Estapé, «La cinta de correr para realidad virtual », Computer Hoy. Accedido: 22 de enero de 2025. [En línea]. Disponible en: <https://computerhoy.20minutos.es/noticias/gaming/kat-vr-walk-c-2-plus-cinta-correr-realidad-virtual-video-1160845>
- [11] J. S. M. Rodríguez, Q. J. R. Aspiazu, Á. M. C. Magallón, y M. R. L. García, «Simulación y realidad virtual aplicada a la educación», *RECIAMUC*, vol. 5, n.º 2, Art. n.º 2, jun. 2021
- [12] «Simulador de realidad virtual e Inteligencia Artificial para entrenar a cirujanos», Red de simulacion en salud. Accedido: 4 de febrero de 2025. [En línea]. Disponible en:

<https://reddesimulacionensalud.com/tecnologia/simulador-de-realidad-virtual-e-inteligencia-artificial-para-entrenar-a-cirujanos/>

- [13] E. Sandu, «La realidad virtual en el deporte - metaverso.pro». Accedido: 4 de febrero de 2025. [En línea]. Disponible en: <https://metaverso.pro/blog/la-realidad-virtual-en-el-deporte/>
- [14] E. Gómez Quijano, «La simulación en el ámbito de la Armada - Innovación y Defensa », Ministerio de Defensa de España. Accedido: 10 de febrero de 2025. [En línea]. Disponible en: <https://www.tecnologiaeinnovacion.defensa.gob.es/Lists/Publicaciones/Attachments>
- [15] J. Martín Llamas, «La simulación en el ámbito del Ejército de Tierra - Innovación y Defensa», Ministerio de Defensa de España. Accedido: 12 de febrero de 2025. [En línea]. Disponible en: <https://www.tecnologiaeinnovacion.defensa.gob.es/Lists/Publicaciones/Attachments>
- [16] E. Sánchez Pintado, «La simulación en el ámbito del Ejército del Aire - Innovación y Defensa », Ministerio de Defensa de España. Accedido: 13 de febrero de 2025. [En línea]. Disponible en: <https://www.tecnologiaeinnovacion.defensa.gob.es/Lists/Publicaciones/Attachments>
- [17] «Learn About Epic Games», Epic Games. Accedido: 14 de febrero de 2025. [En línea]. Disponible en: <https://www.epicgames.com/site/es-ES/about>
- [18] J. Lee, «Learning Unreal Engine Game Development», Packt Publishing Ltd, vol. 1, n.º 1, p 2-3 February 2016.
- [19] «Esta es la historia de Unreal Engine | The Factory School». Accedido: 16 de febrero de 2025. [En línea]. Disponible en: <https://thefactoryschool.com/blog/la-historia-de-unreal-engine/>
- [20] «Unreal Engine para la realidad extendida (XR): RA, RV y MR», Unreal Engine. Accedido: 16 de febrero de 2025. [En línea]. Disponible en: <https://www.unrealengine.com/es-ES/xr>
- [21] «Gafas de realidad mixta y virtual Meta Quest 3». Accedido: 17 de febrero de 2025. [En línea]. Disponible en: <https://www.meta.com/es/quest/quest-3/>
- [22] «VR Gunstock and Accessories by ProTubeVR», ProTubeVR. Accedido: 18 de febrero de 2025. [En línea]. Disponible en: <https://www.protubevr.com/en/>
- [23] «BHaptics TactSuit», bHaptics. Accedido: 18 de febrero de 2025. [En línea]. Disponible en: <https://www.bhaptics.com>
- [24] «KAT Walk C 2 Core», KATVR. Accedido: 19 de febrero de 2025. [En línea]. Disponible en: <https://www.kat-vr.com/products/kat-walk-c-2-core>
- [25] H. D. Mouton, «Comparison of body rotations using Euler angles and quaternions», 2021, Accedido: 8 de marzo de 2025. [En línea]. Disponible en: <http://hdl.handle.net/11427/33222>
- [26] «ForceTubeVR Unreal Engine Plugin», GitHub. Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://github.com/ProTubeVR/ForceTubeVR-Unreal-Engine-Plugin>

ANEXO I: IMPLICACIONES SOCIALES, ECONÓMICAS Y AMBIENTALES

La realidad virtual ha generado un impacto social significativo al transformar la forma en la que las personas interactúan con la tecnología y entre ellas mismas. En el contexto del presente trabajo, que se centra en el desarrollo de un módulo base interactivo para simulación de entornos inmersivos, la realidad virtual no solo ofrece nuevas formas de aprendizaje y entrenamiento, sino que también introduce mejoras en la interacción usuario-máquina. Este módulo establece una base versátil sobre la cual se pueden desarrollar aplicaciones más específicas, incluyendo aquellas orientadas a la educación avanzada, la simulación industrial y otros sectores especializados. Gracias a la implementación de dispositivos hápticos, la experiencia inmersiva alcanza un nivel más realista, lo que mejora la respuesta del usuario en situaciones interactivas. En ámbitos como la educación y la formación profesional, este tipo de simulaciones ofrecen entornos controlados donde los usuarios pueden practicar habilidades técnicas de manera repetitiva y sin riesgos. Además, la posibilidad de personalizar y adaptar los escenarios de realidad virtual según las necesidades específicas de cada usuario o institución facilita su integración en distintos programas de enseñanza y capacitación. Sin embargo, el uso prolongado de estas tecnologías puede generar fatiga visual y desconexión con el entorno físico, lo que hace necesario desarrollar pautas de uso responsable y recomendaciones ergonómicas adaptadas al usuario.

Desde una perspectiva económica, el desarrollo de un módulo base interactivo en realidad virtual tiene implicaciones clave. Al tratarse de una tecnología en expansión, su integración en la formación y entrenamiento representa una oportunidad para reducir costes operativos, evitando el desgaste de materiales físicos y optimizando el tiempo de instrucción. Si bien este trabajo no se orienta exclusivamente a un sector en particular, su modularidad y escalabilidad permiten que otros desarrollos puedan adaptarlo a necesidades más específicas, como el entrenamiento en seguridad, simulaciones técnicas o formación industrial. En el ámbito empresarial, la realidad virtual aplicada a la simulación facilita la optimización de procesos de diseño y fabricación, minimizando errores y reduciendo tiempos de desarrollo de productos. Además, permite realizar pruebas y evaluaciones en entornos simulados antes de pasar a la fase de producción real, reduciendo costos y riesgos asociados a fallos en el diseño. No obstante, la adopción de estas tecnologías sigue estando limitada por los costes iniciales de adquisición y mantenimiento, así como por la necesidad de formación del personal que las utiliza. En este sentido, la flexibilidad del sistema desarrollado en este trabajo busca ofrecer una solución eficiente y escalable que permita su implementación en distintos entornos sin generar barreras de entrada económicas excesivas. Con una correcta planificación y estrategias de adopción, la realidad virtual puede convertirse en una inversión rentable para las empresas y centros de formación, optimizando sus recursos y mejorando la calidad del aprendizaje.

Las implicaciones ambientales de la realidad virtual están directamente relacionadas con la producción de hardware y el consumo energético de los sistemas empleados en su implementación. En el contexto de este trabajo, la selección de materiales y procesos de fabricación ha sido un factor clave para reducir el impacto ecológico del producto final. La gestión de residuos electrónicos derivados de los equipos de realidad virtual sigue siendo uno de los principales desafíos, ya que muchos de estos dispositivos tienen una vida útil relativamente corta y requieren programas específicos de reciclaje y reutilización. Además, el alto consumo energético de los servidores y estaciones de trabajo empleados en la simulación es otro aspecto a considerar. A medida que las simulaciones en realidad virtual se vuelven más complejas y detalladas, la demanda de procesamiento y almacenamiento de datos aumenta significativamente, lo que puede generar un impacto ambiental considerable. La optimización de los algoritmos de procesamiento y el uso de tecnologías eficientes en términos de consumo eléctrico pueden mitigar este impacto, pero sigue siendo fundamental desarrollar estrategias para reducir la huella de carbono generada por estos sistemas. Implementar prácticas de eficiencia energética en los centros de datos y fomentar el desarrollo de hardware de bajo consumo puede contribuir a disminuir el impacto ambiental de esta tecnología a largo plazo.

En conclusión, el desarrollo de un módulo base interactivo para la simulación de entornos inmersivos en realidad virtual presenta múltiples implicaciones sociales, económicas y ambientales.

Aunque este trabajo no está específicamente diseñado para un sector concreto, sienta las bases para que otros entornos más específicos puedan desarrollarse a partir de él. Su aplicación en la formación profesional permite entrenamientos más eficientes, seguros y accesibles, optimizando recursos y mejorando la experiencia del usuario. A nivel económico, su implementación representa una reducción de costes operativos a largo plazo, aunque su adopción inicial puede requerir una inversión significativa en infraestructura y capacitación. Sin embargo, la posibilidad de personalización y adaptación a distintos escenarios hace que la inversión sea viable en múltiples sectores. En términos ambientales, sigue siendo necesario abordar la gestión de residuos electrónicos y el consumo energético de estos sistemas. En definitiva, la realidad virtual ofrece un enorme potencial para transformar la manera en que se desarrollan los procesos de aprendizaje y entrenamiento, siempre que su implementación se realice de manera ética, sostenible y accesible.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El desarrollo de un módulo base interactivo para la simulación de entornos inmersivos en realidad virtual conlleva una serie de reflexiones éticas y sociales que deben ser consideradas para garantizar su implementación responsable y su impacto positivo en la sociedad. La realidad virtual está transformando múltiples sectores, desde la educación hasta la formación profesional, proporcionando experiencias inmersivas que mejoran la comprensión y el desarrollo de habilidades. Sin embargo, su uso también plantea desafíos en términos de accesibilidad, privacidad, bienestar del usuario y posibles efectos sobre la percepción de la realidad.

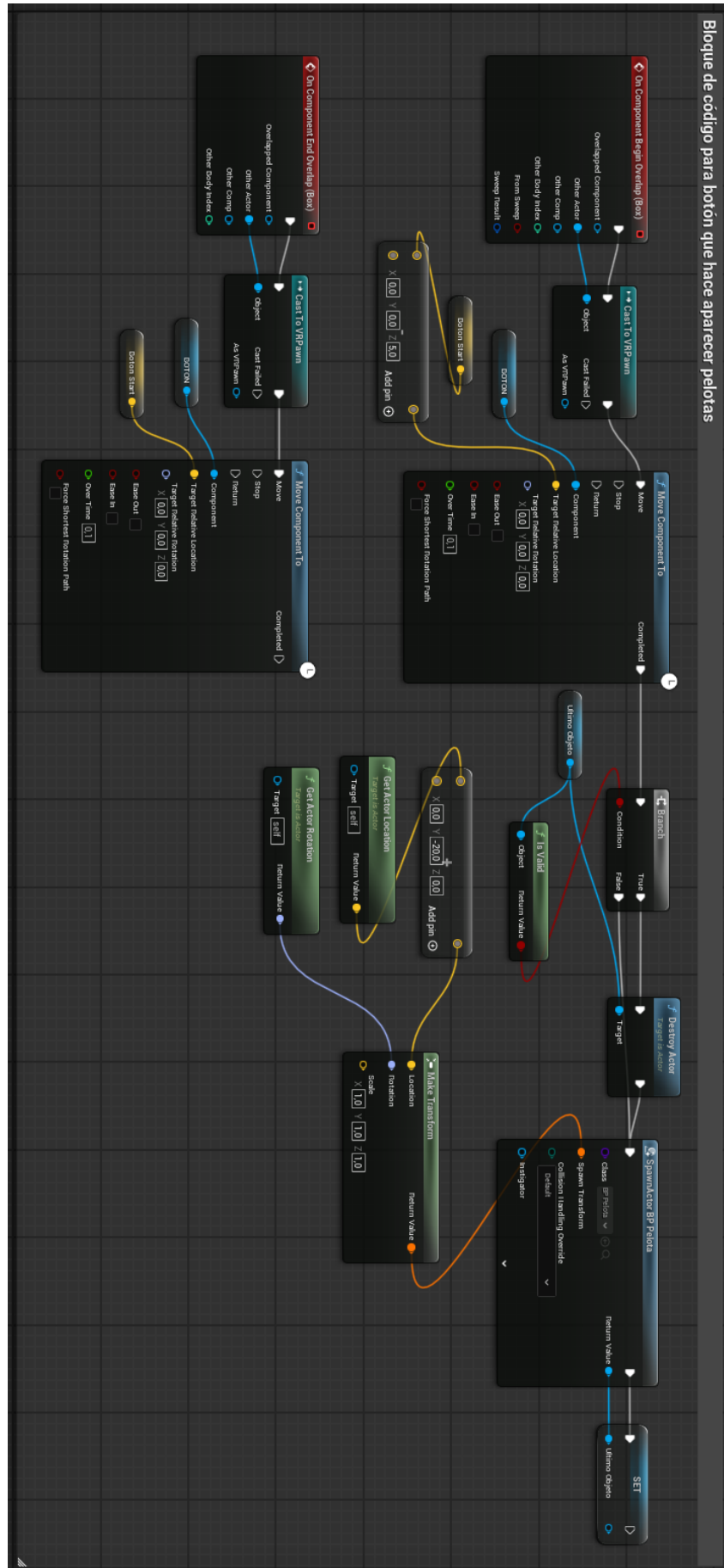
Desde una perspectiva ética, la accesibilidad a la tecnología de realidad virtual es un aspecto crucial. Aunque su potencial para la educación y la capacitación es enorme, no todas las instituciones o individuos tienen los recursos para acceder a ella. Es fundamental promover el desarrollo de soluciones más asequibles y accesibles para evitar la brecha digital y garantizar que un mayor número de personas pueda beneficiarse de estas herramientas. En este sentido, el módulo desarrollado en este trabajo busca ser una base flexible y adaptable que permita su aplicación en distintos entornos sin requerir una inversión prohibitiva.

En cuanto a las implicaciones sociales, la realidad virtual tiene el potencial de mejorar la educación y la formación al proporcionar experiencias de aprendizaje más dinámicas e interactivas. Sin embargo, su uso debe complementarse con estrategias que eviten la desconexión del mundo real. El riesgo de aislamiento social es un desafío importante, ya que un uso excesivo o descontrolado de la realidad virtual puede llevar a una disminución de la interacción con el entorno físico y las relaciones interpersonales. Para mitigar este efecto, es recomendable que las aplicaciones de realidad virtual fomenten un equilibrio entre el mundo digital y el contacto con la realidad tangible.

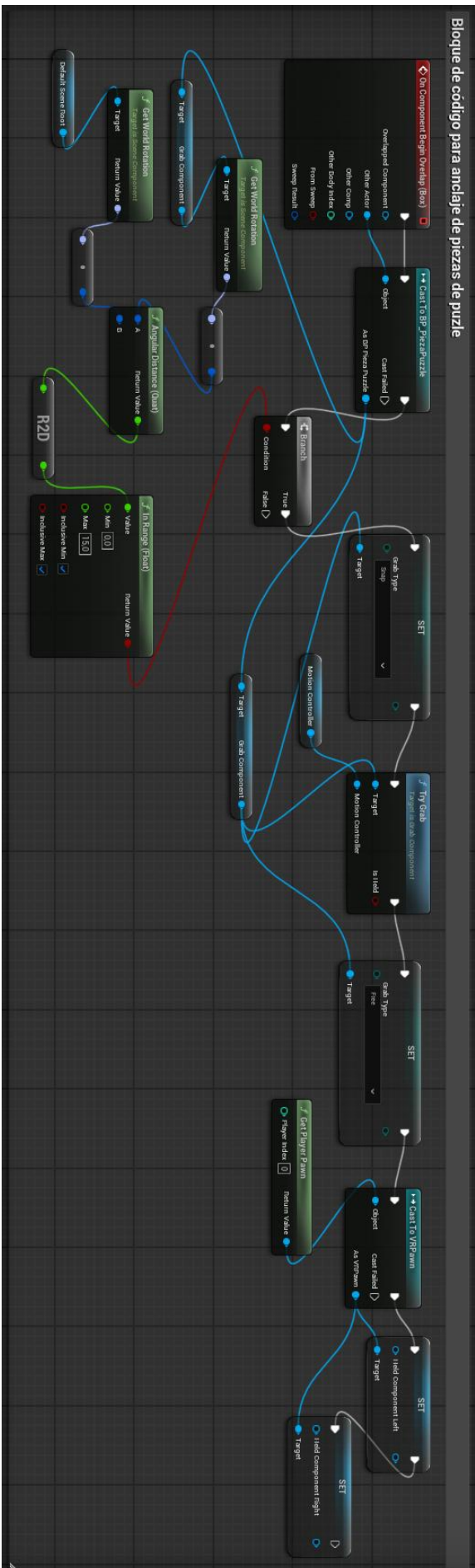
Asimismo, el impacto de la realidad virtual en la percepción de la realidad es un tema de debate. La exposición prolongada a entornos simulados puede generar confusión en algunos usuarios, especialmente en aquellos que participan en simulaciones de alta intensidad. En este sentido, es fundamental establecer directrices que regulen el uso responsable de estas tecnologías y evitar que su abuso pueda derivar en efectos psicológicos adversos. La inclusión de pausas, la supervisión en entornos educativos y la adaptación de los escenarios a las necesidades de los usuarios son estrategias clave para minimizar estos riesgos.

En conclusión, el desarrollo y aplicación de un módulo base interactivo para simulaciones en realidad virtual conlleva importantes consideraciones éticas y sociales. La accesibilidad, el impacto en la percepción de la realidad y el equilibrio entre lo digital y lo físico son aspectos fundamentales que deben abordarse para asegurar un uso responsable de esta tecnología. Con un enfoque adecuado, la realidad virtual puede ser una herramienta poderosa para la educación, la formación y el desarrollo profesional, siempre que su implementación se realice de manera ética, inclusiva y socialmente responsable.

ANEXO III: BLOQUE DE CÓDIGO PARA BOTÓN DE PELOTAS



ANEXO IV: BLOQUE DE CÓDIGO PARA ANCLAJE DE PIEZA



ANEXO V: BLOQUE DE CÓDIGO PARA INTERACCIÓN CON BOTONES DE LA PIZARRA

