



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de un entorno didáctico virtual destinado al aprendizaje
tecnológico mediante realidad mixta*

Grado en Ingeniería Mecánica

ALUMNO: Julián Guillermo Garófano García-Cubillana

DIRECTORES: Francisco Troncoso Pastoriza

Xavier Núñez Nieto

CURSO ACADÉMICO: 2025-2026

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de un entorno didáctico virtual destinado al
aprendizaje tecnológico mediante realidad mixta*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

En la actualidad, las tecnologías inmersivas abren nuevas posibilidades para el aprendizaje y la interacción, al combinar el espacio físico con elementos virtuales de forma natural. En este contexto, este Trabajo Fin de Grado se centra en el diseño y desarrollo de un entorno en realidad mixta orientado al aprendizaje tecnológico. Para ello, se emplea *Unity* como motor de desarrollo y el equipo *Meta Quest 3* como dispositivo principal.

Sobre esta base, se han diseñado distintas pruebas interactivas en las que el usuario puede manipular objetos virtuales, orientarse en el espacio mixto y responder a diferentes estímulos. Las pruebas desarrolladas incluyen actividades centradas en la interacción con el entorno, la coordinación espacial y la precisión de movimiento, incorporando además sistemas de realimentación háptica para reforzar la inmersión.

El resultado es un entorno funcional, adaptable y de carácter didáctico, planteado como una propuesta de iniciación a la realidad mixta y como una base ampliable para futuros desarrollos.

PALABRAS CLAVE

Realidad Mixta, Aprendizaje inmersivo, Retroalimentación háptica, *Meta Quest 3*, *Unity*

AGRADECIMIENTOS

En primer lugar, agradecer a mis tutores de este trabajo. Al Profesor Francisco Troncoso Pastoriza por el interminable asesoramiento técnico y por estar ahí en los malos, y sobre todo, en los peores momentos del desarrollo de este proyecto, que desde luego no han sido fáciles. Y por otro lado al Profesor Xavier Núñez Nieto, que, con su fina ironía y su impecable estilo, mostró desde el primer momento un gran entusiasmo y un alto nivel de exigencia; además, puedo decir con casi total confianza, que le he conseguido caer bien después de todo.

En segundo lugar, quiero agradecer a mis amigos de la promoción 426-156, en particular a mis compañeros de camareta y mis inestimables vecinos de la 117. Si algo bueno he hecho tras cinco años en la ENM ha sido gracias a la gran familia que me ha rodeado desde el primer momento, llegué aquí solo y sin saber nada, y tras todo el tiempo que ha pasado puedo decir con seguridad que me llevo conmigo algo de valor incalculable.

Por último, agradecer a mi familia, los responsables de que haya hecho este trabajo en un primer lugar.

CONTENIDO

Contenido	1
Índice de Figuras	4
Índice de Tablas.....	7
1 Introducción y objetivos.....	9
1.1 Descripción del apartado.....	9
1.2 Antecedentes	9
1.3 Motivación	10
1.4 Objetivos	10
1.5 Estructura del trabajo	11
2 Estado del arte	13
2.1 Descripción del apartado.....	13
2.2 Evolución histórica de los entornos virtuales e inmersivos	13
2.2.1 Orígenes de la simulación y los entornos artificiales	13
2.2.2 Evolución de la computación gráfica y la interacción.....	14
2.3 Tecnologías de realidad extendida.....	15
2.3.1 La Realidad Virtual.....	15
2.3.2 La Realidad Aumentada	15
2.3.3 La Realidad Mixta	16
2.3.4 Aplicaciones de la realidad mixta.....	17
2.4 Entornos virtuales aplicados al aprendizaje tecnológico	18
2.4.1 Aplicaciones educativas y de entrenamiento técnico	18
2.4.2 Limitaciones de los sistemas actuales.....	18
2.5 Entornos de iniciación y familiarización en realidad mixta.....	19
2.6 Plataformas software para la creación de entornos virtuales y mixtos	20
2.6.1 Unity como entorno de desarrollo para aplicaciones inmersivas	20
2.6.2 Integración de Unity con dispositivos de realidad mixta.....	21
2.7 Equipamiento inmersivo para entornos virtuales.....	22
2.7.1 Dispositivos de visualización	22
2.7.2 Sistemas de retroalimentación háptica.....	22
2.7.3 Sistemas de locomoción	24
2.8 Aplicaciones de la realidad mixta en el ámbito naval y de defensa.....	25
3 Desarrollo	27
3.1 Descripción del apartado.....	27

3.2 Introducción a Unity	27
3.2.1 Instalación de la máquina virtual y creación del proyecto.....	28
3.2.2 Sincronización de Unity con el entorno Meta	31
3.3 Adquisición de Conceptos Básicos	32
3.3.1 Camera Rig y las interacciones con los entornos VR.....	32
3.3.2 Passthrough.....	33
3.3.3 Spatial Anchors.....	33
3.3.4 Programación en C#	34
3.3.5 Entorno bHaptics	34
3.4 Construcción de las escenas	36
3.4.1 Elementos comunes	36
3.4.2 Menú Inicial.....	38
3.4.3 Prueba del Aro	42
3.4.4 Prueba de Baloncesto.....	46
3.4.5 Puzzle 3D.....	50
3.4.6 Prueba del lanzador.....	56
3.4.7 Prueba de puntería	60
3.4.8 Instalación del proyecto en las gafas Meta Quest 3.....	64
4 Resultados	67
4.1 Descripción del apartado.....	67
4.2 Descripción general y objetivos de las pruebas	67
4.3 Escenario de Realidad Mixta	70
4.4 Elementos de apoyo a la experiencia	72
4.5 Manipulación, precisión y coordinación espacial	75
4.6 Retroalimentación háptica.....	77
5 Conclusiones y líneas futuras	79
5.1 Descripción del apartado.....	79
5.2 Conclusiones	79
5.2.1 Escenario de realidad mixta.....	79
5.2.2 Interacción con objetos y coordinación espacial	79
5.2.3 Integración de dispositivos y retroalimentación háptica.....	80
5.2.4 Conclusión final.....	80
5.3 Líneas Futuras	80
6 Bibliografía.....	83
Anexo I: Implicaciones Sociales, Económicas y Ambientales	87

Anexo II: Reflexiones Éticas y Sociales	89
Anexo III: Diagramas de flujo de los códigos del proyecto	91
Anexo IV: Declaración del uso de Inteligencia Artificial	111

ÍNDICE DE FIGURAS

Figura 2-1: Piloto adiestrándose en simulador <i>Link Trainer</i> [1].....	14
Figura 2-2: Disposición de entorno CAVE.....	14
Figura 2-3: Usuario de equipos de Realidad Virtual [6].....	15
Figura 2-4: Aplicación de realidad Aumentada en teléfono móvil [6]	16
Figura 2-5: Explicación de las diferencias entre la RV, RA y RM.....	17
Figura 2-7: Primer simulador RV 3D de mantenimiento de Siemens [13].....	18
Figura 2-8: Tutorial de uso en aplicación de realidad extendida PLNAR [14]	19
Figura 2-9: Captura de pantalla de la interfaz de Unity en el tutorial de iniciación.	20
Figura 2-10: Capacidades de <i>OpenXR</i> [17].....	21
Figura 2-11: Equipo de visualización Meta Quest 3 junto con los controladores manuales [18].....	22
Figura 2-12: Dispositivo ProtubeVR Gunstock [20]	23
Figura 2-13: Chaleco <i>Bhaptics Tactsuit x40</i> [20]	23
Figura 2-14: Guantes <i>bHaptics Tactglove DK2</i> [21]	24
Figura 2-15: Plataforma de locomoción KAT VR [22]	24
Figura 2-16: Dotación del USS Michael Murphy empleando equipos VR para adiestramiento [24]	25
Figura 3-1: Interfaz inicial de UnityHub.....	28
Figura 3-2: Selección del tipo de proyecto	28
Figura 3-3: Vista de Escena en Unity	29
Figura 3-4: Barras de herramientas y ejes de orientación espacial	29
Figura 3-5: Ventana de Jerarquía	30
Figura 3-6: Inspector de componentes	30
Figura 3-7: Ventana de Proyecto	31
Figura 3-8: <i>Game View</i> del resultado del tutorial <i>Hello World</i> de <i>Meta For Developers</i> [26]	32
Figura 3-9: [29] Interfaz de la aplicación <i>bHaptics Player</i>	35
Figura 3-10: Gestión de los eventos hápticos en <i>bHaptics Developer</i>	35
Figura 3-11: Configuración de los patrones hápticos en <i>bHaptics Designer</i> [30].....	36
Figura 3-12: Panel introductorio escena inicial e inspector del texto	37
Figura 3-13: Inspector del código <i>HideIntroOnA</i>	37
Figura 3-14: Inspector del código <i>ReturnToMenuOnY</i>	38
Figura 3-15: Inspector del código <i>SpawnEstructuraOnB_NoFloor</i>	38
Figura 3-16: Aplicación del código <i>SceneLoader</i> en el inspector de los botones de la pantalla	39
Figura 3-17: Configuración del sombreado de los botones al ser pulsados.....	39
Figura 3-18: Inspector del código <i>SimpleLaser</i>	40
Figura 3-19: Inspector del código <i>OVRRaycaster</i>	40

Figura 3-20: Inspector del <i>EventSystem</i>	41
Figura 3-21: TV LED 30" del <i>Unity Asset Store</i> [31]	41
Figura 3-22: Cuerpo y jerarquía de componentes de la pantalla del menú inicial	41
Figura 3-23: Inspector del código <i>RightHandSpawner</i>	42
Figura 3-24: Inspector de componentes del aro	43
Figura 3-25: Inspector del código <i>RingCollidersGenerator</i>	43
Figura 3-26: Cuerpo del aro y su jerarquía de componentes	44
Figura 3-27: Inspector de componentes de las barras de la estructura	45
Figura 3-28: Inspector del código <i>StructureCollisionCounter</i>	45
Figura 3-29: Estructura de barras de la primera prueba y sus componentes	46
Figura 3-30: <i>SceneView</i> de la superficie de suelo	47
Figura 3-31: Inspector del código <i>SpawnEnSuelo</i>	47
Figura 3-32: Inspector del código <i>BallThrownManager</i>	48
Figura 3-33: Paquete <i>BasketballRealistic</i> en <i>Unity Asset Store</i> [32]	48
Figura 3-34: Cuerpo de la pelota y su inspector de componentes	49
Figura 3-35: <i>BasketballRing</i> en <i>Unity Asset Store</i> [33]	49
Figura 3-36: Cuerpo de la canasta con <i>colliders</i> y el inspector del código <i>HoopScoreZone</i>	50
Figura 3-37: Inspector del código <i>RightHandSpawner_Free</i>	51
Figura 3-38: Inspector del componente <i>Bhaptics Gloves</i>	52
Figura 3-39: Inspector del código <i>SnapSocket</i>	53
Figura 3-40: Cuerpo y jerarquía de componentes del cubo central	53
Figura 3-41: Inspector del código <i>SnapPiece</i>	54
Figura 3-42: Inspector del <i>Bhaptics Event Player</i>	54
Figura 3-43: Inspector del <i>Unity Event Wrapper</i>	55
Figura 3-44: Configuración de los <i>patterns snap</i> (izquierda) y <i>grab</i> (derecha)	55
Figura 3-45: Cuerpo y jerarquía de los Cubos-Pieza	55
Figura 3-46: Cuerpo, inspector y jerarquía de los <i>colliders</i> del torso	57
Figura 3-47: Inspector del código <i>BallLauncher</i>	58
Figura 3-48: Inspector del código <i>LauncherGripInput</i>	58
Figura 3-49: Cuerpo y Jerarquía de componentes del lanzador	59
Figura 3-50: Cuerpo e inspector de la bola	59
Figura 3-51: Inspector del código <i>BallHitBodyZones</i>	60
Figura 3-52: Inspector del código <i>TargetSpawner</i>	61
Figura 3-53: <i>ForcetubeVR</i> (Izquierda) y pistola háptica <i>Quest3 Provolver</i> (Derecha) [20]	61
Figura 3-54: Rotación del objeto <i>FirePoint</i> del mando izquierdo	62

Figura 3-55: Rotación del objeto FirePoint del mando derecho	62
Figura 3-56: Inspector del código <i>ProtubeShooter</i>	62
Figura 3-57: Configuración de vibración de retroceso en <i>Bhaptics designer</i>	63
Figura 3-58: <i>Archery Target</i> del <i>Unity Asset Store</i>	63
Figura 3-59: Cuerpo e inspector de componentes de la diana	64
Figura 3-60: Acceso a los <i>Build Profiles</i>	64
Figura 3-61: Botón para realizar el <i>Clean Build</i>	65
Figura 3-62: Instalación del programa en las gafas desde Windows PowerShell	65
Figura 4-1: Circuito de barras y aro de la Prueba del Aro	68
Figura 4-2: Pelota y Canasta en la prueba del Baloncesto	68
Figura 4-3: Puzle 3D completado	69
Figura 4-4: Estructura lanzadora de la prueba del Lanzador	69
Figura 4-5: Canasta y pelota virtual sobre suelo físico	70
Figura 4-6: Mismo objeto, la pantalla del menú inicial, situado en distintas partes de la habitación.....	71
Figura 4-7: Cubos piezas y cubo central instanciados para empezar el puzle 3D	71
Figura 4-8: Puntero de la pantalla del menú inicial	72
Figura 4-9: Cambio de color en botones al ser pulsados	73
Figura 4-10: Paneles introductorios de distintas escenas del proyecto	73
Figura 4-11: Contadores de las distintas pruebas.....	74
Figura 4-12: Luz indicadora de colisiones en la Prueba del Aro	74
Figura 4-13: Aro pasando por la estructura de barras	75
Figura 4-14: Dinámica de lanzamiento en la prueba de baloncesto	75
Figura 4-15: Usuario agarrando los cubos del puzle 3D con las manos	76
Figura 4-16: Usuario apuntando con pistola háptica en la prueba de puntería	76
Figura 4-17: Guantes <i>bHaptics</i> en vista de juego	77
Figura 4-18: Usuario con el chaleco <i>bHaptics TactSuit x40</i>	78
Figura 4-19: Usuario empleando el <i>ForcetubeVR</i> (Izquierda) y la pistola <i>ProvolverVR</i> (Derecha)	78

ÍNDICE DE TABLAS

Tabla 1: Aplicaciones de los entornos de realidad mixta.....	17
--	----

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Descripción del apartado.

En este apartado se exponen los antecedentes y el contexto en el que se enmarca el presente Trabajo Fin de Grado. Asimismo, se detalla la motivación que ha dado lugar a su desarrollo y se definen los objetivos principales que se persiguen. Por último, se describe la estructura del documento y la organización de los contenidos que lo componen.

1.2 Antecedentes

En los últimos años, el uso de tecnologías de realidad virtual, aumentada y mixta ha experimentado un notable crecimiento en ámbitos como la formación técnica, el entrenamiento profesional y la simulación de entornos complejos. Estas tecnologías han permitido el desarrollo de sistemas capaces de recrear escenarios virtuales e híbridos en los que el usuario puede interactuar con elementos digitales de forma cada vez más natural, abriendo nuevas posibilidades para el aprendizaje y la preparación previa a entornos reales.

Tradicionalmente, gran parte de los desarrollos orientados al ámbito educativo se han centrado en la creación de entornos virtuales destinados a la enseñanza de contenidos específicos o a la simulación directa de tareas concretas. Sin embargo, la correcta utilización de estos sistemas requiere que el usuario posea previamente una serie de habilidades básicas relacionadas con la interacción en entornos virtuales, como la manipulación de objetos, la orientación espacial o el desplazamiento dentro del espacio virtual. La ausencia de estas competencias iniciales puede dificultar el aprovechamiento de aplicaciones más avanzadas y aumentar la curva de aprendizaje asociada al uso de tecnologías inmersivas.

En este contexto, surge la necesidad de disponer de entornos específicos orientados a la iniciación y familiarización del usuario con la realidad virtual y mixta, que permitan adquirir de forma progresiva las habilidades fundamentales necesarias para desenvolverse con soltura en este tipo de sistemas. Estos entornos no persiguen la enseñanza de contenidos teóricos concretos, sino el entrenamiento del usuario en el uso de la propia tecnología, actuando como una fase previa y complementaria a aplicaciones de formación técnica más complejas.

La evolución hacia tecnologías de realidad mixta refuerza esta necesidad, al introducir una mayor integración entre el entorno físico real y los elementos virtuales. Esta combinación incrementa las posibilidades de interacción, pero también añade nuevos retos relacionados con la percepción espacial, la coordinación de acciones físicas y la adaptación del usuario a entornos híbridos. Por ello, resulta

especialmente relevante el desarrollo de sistemas que faciliten la transición del usuario hacia este tipo de entornos, reduciendo la complejidad inicial y favoreciendo una experiencia de aprendizaje progresiva.

En base a estos antecedentes, este Trabajo Fin de Grado se plantea como una evolución de trabajos previos centrados en entornos virtuales, proponiendo el desarrollo de un sistema orientado a la adquisición de competencias básicas para el trabajo en entornos de realidad mixta, mediante la realización de pruebas prácticas diseñadas para entrenar al usuario en las principales formas de interacción con el entorno virtual.

1.3 Motivación

La realidad virtual se ha consolidado como una herramienta eficaz para la formación y el entrenamiento en entornos donde la práctica real presenta limitaciones de tipo económico, logístico o de seguridad. En este contexto, los entornos inmersivos permiten al usuario interactuar de forma controlada con escenarios virtuales, facilitando el aprendizaje práctico y la adquisición de habilidades básicas necesarias para desenvolverse en este tipo de sistemas.

La evolución hacia tecnologías de realidad mixta representa el siguiente paso natural respecto a la realidad virtual, al integrar elementos virtuales con el entorno físico real del usuario. Esta transición introduce nuevas formas de interacción y mayores posibilidades de aplicación, pero también requiere que el usuario se familiarice previamente con el manejo de objetos, la orientación espacial y el desplazamiento en entornos híbridos. Por ello, este Trabajo Fin de Grado se motiva en el desarrollo de un entorno de iniciación en realidad mixta, orientado a entrenar estas habilidades básicas y servir como base para aplicaciones más complejas en el futuro.

1.4 Objetivos

El objetivo principal de este trabajo es el desarrollo de un entorno base de realidad mixta que permita al usuario interactuar de forma intuitiva con distintos elementos virtuales integrados sobre el espacio real. Para ello, se busca construir una experiencia comprensible, accesible y adaptable a diferentes condiciones de uso, de manera que sirva como punto de partida para la familiarización con este tipo de tecnologías inmersivas.

Para ello, se han definido los siguientes objetivos específicos:

1. Desarrollo de un entorno de realidad mixta adaptable al espacio de uso: se pretende diseñar un sistema capaz de integrarse de forma coherente con el entorno físico del usuario, permitiendo que la experiencia pueda utilizarse en distintas ubicaciones sin perder funcionalidad ni claridad de uso.
2. Diseño de una mecánica sencilla de configuración del escenario: se pretende que el propio usuario pueda participar en la colocación y organización de los elementos principales de la experiencia mediante acciones simples con los mandos, facilitando así la adaptación del entorno al espacio real disponible.
3. Implementación de dinámicas de interacción con objetos virtuales: se busca incorporar distintas acciones que permitan al usuario manipular elementos dentro del entorno, favoreciendo tareas como agarrar, mover, soltar, lanzar o colocar objetos en el espacio.
4. Incorporación de sistemas de retroalimentación háptica: se trabajará en la integración de dispositivos que complementen la experiencia visual mediante estímulos táctiles, reforzando la inmersión y mejorando la percepción de determinadas acciones dentro del sistema.

Como objetivo adicional del trabajo, se plantea que el sistema desarrollado no solo cumpla una función técnica, sino también didáctica, permitiendo al usuario adquirir una primera toma de contacto

con la lógica de funcionamiento propia de la realidad mixta y con las posibilidades de interacción que esta ofrece.

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 general de las tecnologías, conceptos y herramientas relacionados con el proyecto. También se presentan los recursos y equipos empleados durante su desarrollo.
3. Desarrollo: Se expone de manera estructurada y detallada el proceso seguido para el diseño e implementación del proyecto, describiendo las distintas etapas que han permitido construir el sistema final.
4. Resultados: En este apartado se presentan los resultados obtenidos a lo largo del desarrollo del trabajo. Se analiza el funcionamiento del sistema implementado y se valora su adecuación a los objetivos planteados al inicio del proyecto.
5. Conclusiones y líneas futuras: Se ofrece una valoración global del trabajo realizado, examinando en qué medida se han alcanzado los objetivos iniciales. Asimismo, se plantean posibles mejoras y futuras líneas de continuidad del proyecto.
6. Bibliografía: Se recopilan y organizan todas las fuentes de información consultadas y utilizadas como referencia durante la realización del trabajo.
7. Anexos: Se adjunta material complementario relevante para facilitar la comprensión del proyecto, así como información adicional de interés relacionada con su desarrollo.

2 ESTADO DEL ARTE

2.1 Descripción del apartado

En este capítulo se pretende revisar y analizar los principales trabajos y tecnologías existentes relacionados con los entornos de inmersión virtual, centrando el estudio en la realidad mixta y su aplicación en entornos de aprendizaje tecnológico. Se describen los conceptos fundamentales de la realidad virtual y la realidad aumentada, así como su evolución hacia sistemas híbridos que permiten la integración de elementos reales y virtuales, es decir, la realidad mixta.

Asimismo, se analizan las aplicaciones actuales de estos entornos en el ámbito educativo y de entrenamiento técnico, junto con las tecnologías de visualización tridimensional e interacción empleadas. El estudio del estado del arte permite contextualizar el Trabajo Fin de Grado y justificar el desarrollo del entorno didáctico virtual propuesto.

2.2 Evolución histórica de los entornos virtuales e inmersivos

2.2.1 Orígenes de la simulación y los entornos artificiales

Los orígenes de los entornos virtuales e inmersivos se encuentran en el desarrollo de los primeros sistemas de simulación utilizados con fines de entrenamiento y formación técnica. Desde mediados del siglo XX, distintas disciplinas comenzaron a emplear simuladores mecánicos y visuales para reproducir de manera controlada situaciones complejas que resultaban difíciles, costosas o peligrosas de recrear en entornos reales, especialmente en ámbitos como la aviación, la ingeniería y la defensa.

Uno de los primeros campos en adoptar este tipo de sistemas fue el entrenamiento aeronáutico, donde se desarrollaron simuladores de vuelo capaces de reproducir el comportamiento básico de una aeronave y sus instrumentos, como es el caso del *Link Trainer*, el primer simulador de vuelo, creado en 1929 en Nueva York [1] mostrado en la Figura 2-1. Aunque estos sistemas presentaban un grado limitado de realismo gráfico e interacción, permitían al usuario familiarizarse con procedimientos y situaciones operativas, sentando las bases del aprendizaje basado en simulación [2].

Paralelamente, en otros ámbitos técnicos comenzaron a emplearse entornos artificiales para el análisis y la experimentación de sistemas complejos. El uso de modelos físicos y representaciones visuales permitió avanzar en la comprensión de procesos técnicos y mecánicos, introduciendo el concepto de interacción entre el usuario y un entorno simulado como herramienta de apoyo al aprendizaje y a la toma de decisiones [3].

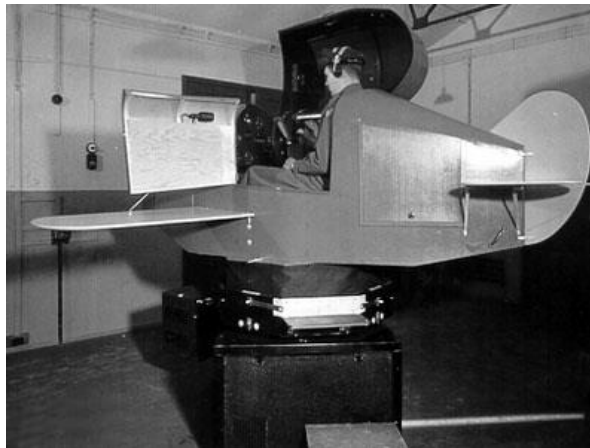


Figura 2-1: Piloto adiestrándose en simulador *Link Trainer* [1]

Estos primeros sistemas de simulación no tenían como objetivo la inmersión sensorial completa, sino la reproducción funcional de escenarios y procedimientos. Sin embargo, establecieron los fundamentos conceptuales sobre los que se desarrollaron posteriormente los entornos inmersivos, al demostrar el valor de la simulación como herramienta eficaz para la formación técnica y el entrenamiento especializado.

2.2.2 Evolución de la computación gráfica y la interacción

La evolución de los entornos virtuales estuvo estrechamente vinculada al desarrollo de la computación gráfica y de los sistemas de interacción hombre-máquina. El aumento progresivo de la capacidad de procesamiento permitió pasar de representaciones bidimensionales y estáticas a entornos tridimensionales interactivos, lo que supuso un avance significativo en la visualización y comprensión de sistemas técnicos complejos.

Durante las décadas de 1980 y 1990 comenzaron a desarrollarse sistemas gráficos capaces de generar escenas tridimensionales en tiempo real, utilizados inicialmente en simuladores y aplicaciones de visualización científica. Un ejemplo destacado de esta etapa es el uso de entornos gráficos tridimensionales en simuladores avanzados y en sistemas de visualización inmersiva como los entornos tipo CAVE, esquematizado en la Figura 2-2: Disposición de entorno CAVE, que permitían al usuario explorar espacios virtuales proyectados en salas envolventes mediante técnicas de proyección estereoscópica [4].

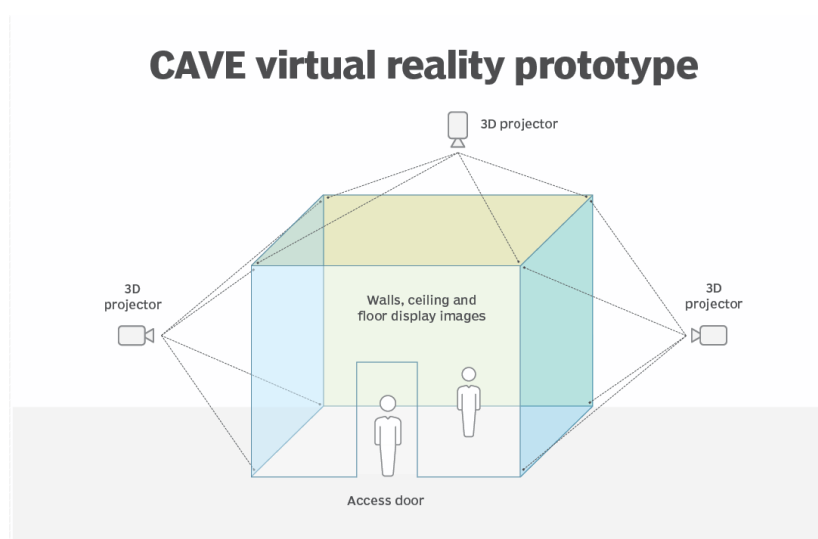


Figura 2-2: Disposición de entorno CAVE

Paralelamente, se produjo una evolución en los dispositivos de interacción, incorporándose sistemas de seguimiento de posición y orientación que permitían al usuario desplazarse e interactuar con los entornos virtuales de forma más natural [5]. Estos avances redujeron la dependencia de interfaces tradicionales como el teclado y el ratón, facilitando una interacción más intuitiva y cercana a la percepción humana, especialmente en aplicaciones de ingeniería y visualización técnica.

La combinación de gráficos tridimensionales avanzados y sistemas de interacción espacial sentó las bases de los entornos inmersivos modernos, al permitir una mayor sensación de presencia y una interacción directa con la información visualizada. Aunque estos sistemas aún presentaban limitaciones en términos de accesibilidad y coste, representaron un paso fundamental en la transición desde la simulación funcional hacia experiencias interactivas más avanzadas, que posteriormente darían lugar a las tecnologías de realidad extendida descritas en el siguiente apartado.

2.3 Tecnologías de realidad extendida

2.3.1 La Realidad Virtual

La realidad virtual (RV) es una tecnología basada en simulación por ordenador que crea entornos tridimensionales interactivos en los que el usuario puede experimentar una sensación de presencia dentro de un mundo completamente generado digitalmente. Esta inmersión se logra mediante dispositivos específicos, como cascos o gafas de realidad virtual como aparece en la Figura 2-3, que sustituyen la percepción del entorno físico por experiencias sensoriales visuales y auditivas propias del entorno simulado, facilitando así la interacción natural con los elementos virtuales. [1]



Figura 2-3: Usuario de equipos de Realidad Virtual [6]

Desde el punto de vista educativo, la RV representa una herramienta con gran potencial, ya que permite la exploración de conceptos y escenarios que, en un contexto tradicional, serían difíciles de experimentar directamente. La interacción en espacios tridimensionales favorece la visualización de estructuras complejas, la práctica segura de procedimientos técnicos y la repetición de actividades de aprendizaje sin riesgos reales ni costes de materiales físicos. Estos aspectos han sido destacados específicamente en estudios que analizan el impacto de la RV en contextos de enseñanza y aprendizaje, donde se observa que entornos inmersivos pueden mejorar la participación del estudiante y facilitar una comprensión más profunda de contenidos técnicos. [7]

2.3.2 La Realidad Aumentada

La realidad aumentada (RA) es una tecnología que permite superponer información digital como imágenes, modelos tridimensionales, textos o datos, sobre el entorno físico real, combinando en tiempo real. A diferencia de la realidad virtual, la RA no sustituye completamente el entorno real, sino que lo complementa mediante la incorporación de elementos virtuales que enriquecen la percepción del usuario

[8]. Esta integración se realiza mediante dispositivos como teléfonos móviles, tabletas, gafas inteligentes o visores específicos, que permiten visualizar la información aumentada manteniendo el contacto con el entorno físico.

Desde un punto de vista tecnológico, la realidad aumentada se basa en sistemas de captura y procesamiento del entorno real, junto con técnicas de posicionamiento y seguimiento espacial que permiten al sistema situar correctamente los elementos virtuales en el espacio físico [8] (Figura 2-4). El avance de estas tecnologías ha facilitado su adopción en múltiples ámbitos, incluyendo la industria, la medicina y, de forma creciente, el ámbito educativo y formativo [9].

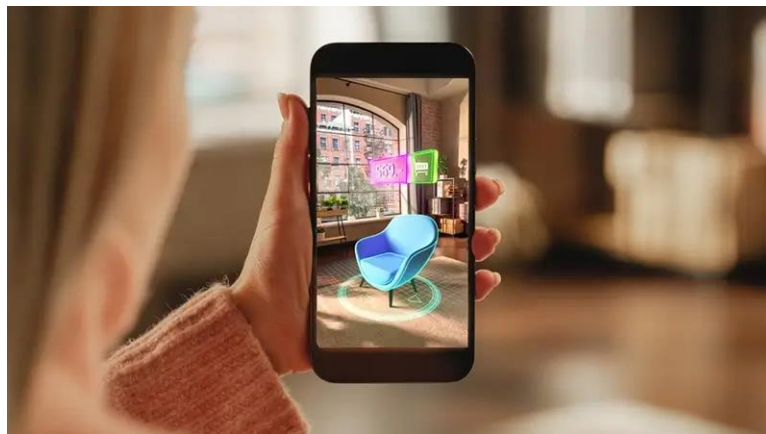


Figura 2-4: Aplicación de realidad Aumentada en teléfono móvil [6]

En el contexto del aprendizaje tecnológico, la RA ofrece ventajas significativas al permitir la visualización interactiva de información técnica directamente sobre objetos reales o escenarios físicos. Esta característica resulta especialmente útil para la comprensión de sistemas complejos, el apoyo a tareas de mantenimiento, la identificación de componentes y la adquisición de conocimientos prácticos sin necesidad de desmontajes físicos o equipamiento adicional [10]. Diversos estudios destacan que la realidad aumentada puede mejorar la motivación del alumno y favorecer un aprendizaje más intuitivo y contextualizado.

2.3.3 La Realidad Mixta

La realidad mixta (RM) es una tecnología que combina elementos de la realidad virtual y de la realidad aumentada, permitiendo la coexistencia e interacción en tiempo real entre objetos físicos y elementos virtuales dentro de un mismo entorno. A diferencia de la realidad aumentada, en la que los elementos virtuales se superponen al mundo real sin una interacción profunda, la realidad mixta permite que los objetos digitales se integren espacialmente en el entorno físico, reaccionando a este y al usuario de forma coherente [8].

Desde el punto de vista tecnológico, la realidad mixta se apoya en sistemas avanzados de mapeado espacial, seguimiento de posición y reconocimiento del entorno, lo que permite situar los elementos virtuales de manera precisa en el espacio físico [6]. Estas capacidades hacen posible que los objetos digitales puedan interactuar con superficies reales, ocluirse correctamente y responder a las acciones del usuario, proporcionando una experiencia más natural e intuitiva que la ofrecida por la realidad aumentada convencional [11]

En cuanto al aprendizaje tecnológico y del entrenamiento técnico, la realidad mixta ofrece ventajas significativas al combinar la inmersión propia de la realidad virtual con la contextualización del entorno real. Esta tecnología permite la visualización y manipulación de modelos tridimensionales directamente sobre escenarios físicos reales, facilitando la comprensión de sistemas complejos, el análisis espacial y la adquisición de habilidades prácticas. Diversos estudios señalan que la realidad mixta puede mejorar

la eficiencia del aprendizaje y reducir errores en procesos formativos técnicos, especialmente en entornos industriales y educativos avanzados [4, 12].

Teniendo en cuenta todo lo mencionado, resumido en la Figura 2-5, se puede concluir que la Realidad Mixta reúne una serie de características que la convierten en una herramienta especialmente adecuada para el desarrollo de entornos didácticos avanzados, como el propuesto en este Trabajo Fin de Grado, al permitir una interacción directa y significativa entre el usuario, el entorno real y los elementos virtuales.



Figura 2-5: Explicación de las diferencias entre la RV, RA y RM

2.3.4 Aplicaciones de la realidad mixta

Ámbito	Aplicación
Formación y educación	Laboratorios virtuales, formación técnica, entornos de iniciación
Industria y mantenimiento	Guiado para montaje y mantenimiento
Medicina	Entrenamiento quirúrgico, visualización anatómica
Defensa	Familiarización con procedimientos, entornos operativos simulados
Entretenimiento	Videojuegos, experiencias interactivas
Arquitectura y construcción	Visualización de proyectos en el entorno real

Tabla 1: Aplicaciones de los entornos de realidad mixta

2.4 Entornos virtuales aplicados al aprendizaje tecnológico

2.4.1 Aplicaciones educativas y de entrenamiento técnico

En los últimos años se ha apreciado un notable crecimiento en el uso de las tecnologías de realidad extendida en el ámbito educativo y del entretenimiento. Estas tecnologías permiten simulaciones reales de situaciones complejas o difíciles de resolver en las que se ponen a prueba los conocimientos y habilidades prácticas difíciles de reproducir mediante los métodos educativos tradicionales.

En el ámbito de la ingeniería y la formación técnica, la realidad virtual se ha utilizado ampliamente en simuladores de entrenamiento, como los empleados para la enseñanza de procedimientos de mantenimiento, montaje y operación de maquinaria. Empresas del sector industrial han desarrollado entornos virtuales que permiten a los usuarios entrenarse en tareas técnicas sin necesidad de acceder físicamente a los equipos reales, reduciendo riesgos y costes asociados [12]. Ejemplos de este enfoque pueden encontrarse en programas de formación industrial basados en simulación desarrollados por Siemens y Bosch (Figura 2-6), donde se emplean entornos virtuales para la capacitación técnica previa a la intervención real [13].

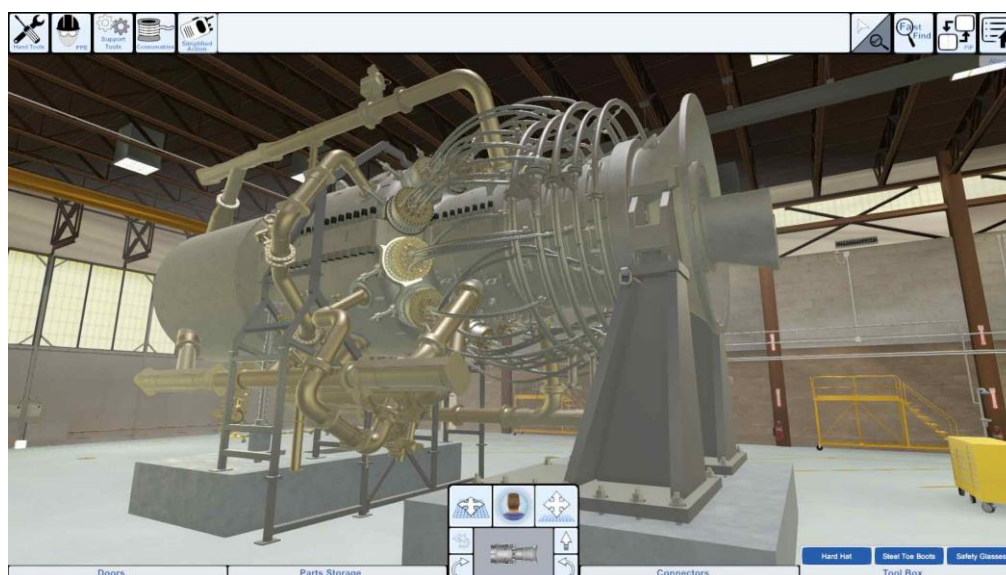


Figura 2-6: Primer simulador RV 3D de mantenimiento de Siemens [13]

Por su parte, la realidad mixta ha comenzado a emplearse en entornos educativos avanzados, combinando la visualización de modelos tridimensionales con la interacción directa en el espacio físico. Dispositivos como *Microsoft HoloLens* se han utilizado en universidades y centros de formación técnica para la enseñanza de ingeniería, anatomía y sistemas mecánicos, permitiendo a los estudiantes manipular modelos virtuales integrados en el entorno real. Estas aplicaciones destacan por mejorar la comprensión espacial y la interacción con sistemas complejos, aspectos clave en el aprendizaje tecnológico.

La incorporación de tecnologías como la realidad virtual, la realidad aumentada y, más recientemente, la realidad mixta, ha ampliado las posibilidades de estos entornos educativos, al combinar la inmersión digital con la contextualización del entorno real. Estas características hacen que los entornos virtuales aplicados al aprendizaje tecnológico constituyan una base sólida para el desarrollo de soluciones didácticas avanzadas.

2.4.2 Limitaciones de los sistemas actuales

A pesar del creciente uso de entornos virtuales en contextos educativos y de entrenamiento, muchos de los sistemas actuales presentan una elevada complejidad de uso, especialmente para usuarios sin experiencia previa en tecnologías inmersivas. En numerosos casos, las interfaces y los esquemas de

interacción no resultan intuitivos, lo que genera una curva de aprendizaje inicial que dificulta el aprovechamiento del entorno.

Asimismo, gran parte de estas aplicaciones carecen de un enfoque didáctico orientado a la iniciación del usuario, centrándose en la simulación de tareas finales sin contemplar un proceso previo de aprendizaje de las habilidades básicas necesarias para interactuar con el entorno virtual. Estas limitaciones evidencian la necesidad de entornos específicos que faciliten la familiarización progresiva del usuario con la realidad mixta, aspecto que se aborda en el apartado siguiente.

2.5 Entornos de iniciación y familiarización en realidad mixta

Los entornos de iniciación y familiarización en realidad mixta surgen como respuesta a la necesidad de preparar al usuario para interactuar de forma eficaz con sistemas inmersivos avanzados. A diferencia de los entornos didácticos tradicionales, orientados a la enseñanza de contenidos específicos o a la simulación directa de tareas finales, este tipo de sistemas tienen como objetivo principal el aprendizaje del uso de la propia tecnología, actuando como una fase previa al empleo de aplicaciones de mayor complejidad.

En muchos contextos formativos y profesionales, se ha observado que la introducción directa de usuarios noveles en entornos de realidad virtual o mixta complejos puede generar dificultades relacionadas con la comprensión de los esquemas de interacción, la orientación espacial o la manipulación de objetos virtuales. Como consecuencia, parte del esfuerzo formativo se dedica a superar la curva de aprendizaje asociada al uso del sistema, en lugar de centrarse en la tarea o contenido que se desea entrenar. Los entornos de iniciación permiten abordar este problema de forma específica, aislando y entrenando las habilidades básicas necesarias para desenvolverse en entornos de realidad mixta.

Este tipo de enfoques ya se ha aplicado de forma implícita en distintos ámbitos tecnológicos. Por ejemplo, en el desarrollo de aplicaciones de realidad virtual y mixta es habitual el uso de escenarios tutoriales o experiencias guiadas iniciales, en las que el usuario aprende acciones básicas como seleccionar objetos, interactuar con interfaces espaciales o desplazarse dentro del entorno antes de acceder a la aplicación principal. De forma similar, plataformas de formación inmersiva incorporan fases de entrenamiento previo destinadas a familiarizar al usuario con los dispositivos y los modos de interacción, especialmente en aplicaciones industriales o de simulación técnica como es el caso de la Figura 2-7.

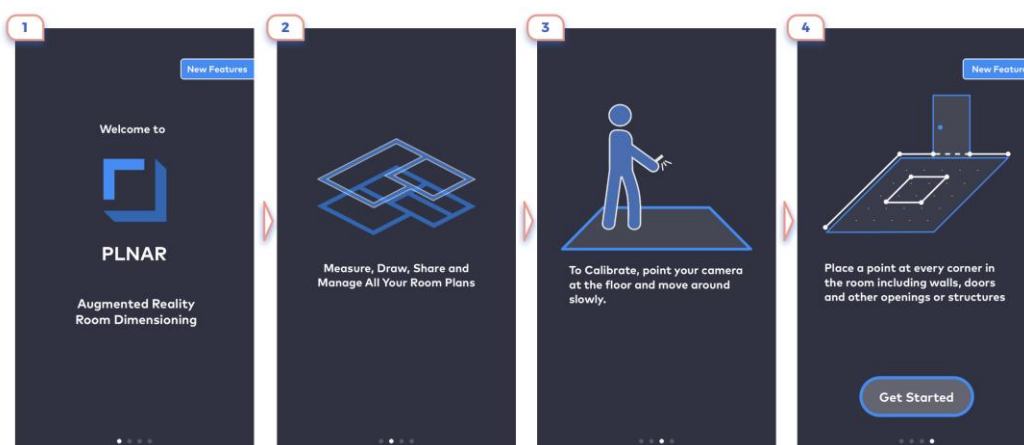


Figura 2-7: Tutorial de uso en aplicación de realidad extendida PLNAR [14]

En el ámbito educativo y de entrenamiento profesional, los entornos de iniciación se utilizan como herramientas de alfabetización en tecnologías inmersivas, permitiendo que el usuario adquiera confianza y destreza antes de enfrentarse a escenarios de trabajo más complejos. Estos entornos pueden estructurarse mediante ejercicios o pruebas progresivas, centradas en acciones fundamentales como la

manipulación y orientación de objetos virtuales, la coordinación entre movimientos físicos y respuestas virtuales, o el desplazamiento controlado dentro de un espacio híbrido.

La aplicación de este enfoque resulta especialmente relevante en la realidad mixta, donde la integración entre el entorno físico real y los elementos virtuales introduce nuevas exigencias en términos de percepción espacial e interacción. La correcta comprensión de la relación entre ambos mundos es un factor determinante para el uso eficaz de estas tecnologías, lo que refuerza la necesidad de entornos específicamente diseñados para la iniciación del usuario. En este sentido, los entornos de familiarización en realidad mixta se presentan como una solución adecuada para reducir la complejidad inicial y facilitar la transición hacia aplicaciones inmersivas más avanzadas.

Un entorno de iniciación bien diseñado debe permitir que el usuario mejore su desempeño a medida que repite las tareas propuestas. Esta mejora suele manifestarse en una reducción del tiempo necesario para completar las acciones, una disminución del número de errores y una mayor fluidez en la interacción con los elementos virtuales. Estos indicadores reflejan que el usuario está interiorizando los mecanismos de control, orientación y manipulación propios del entorno inmersivo.

2.6 Plataformas software para la creación de entornos virtuales y mixtos

2.6.1 Unity como entorno de desarrollo para aplicaciones inmersivas

La interfaz de Unity (Figura 2-8) se organiza en distintos paneles que facilitan el diseño del entorno y la previsualización de la experiencia desde la perspectiva del usuario. A través de la misma es posible configurar y posicionar los elementos del entorno virtual, mientras que la vista de juego permite comprobar el comportamiento del sistema en tiempo real. Este enfoque favorece un proceso de desarrollo iterativo, especialmente adecuado para la creación de entornos de iniciación.

Además, Unity adopta un modelo basado en componentes, mediante el cual el comportamiento de los objetos se define de forma modular. Esta característica resulta especialmente útil en entornos de aprendizaje, donde es habitual diseñar pruebas o ejercicios independientes centrados en habilidades concretas de interacción.

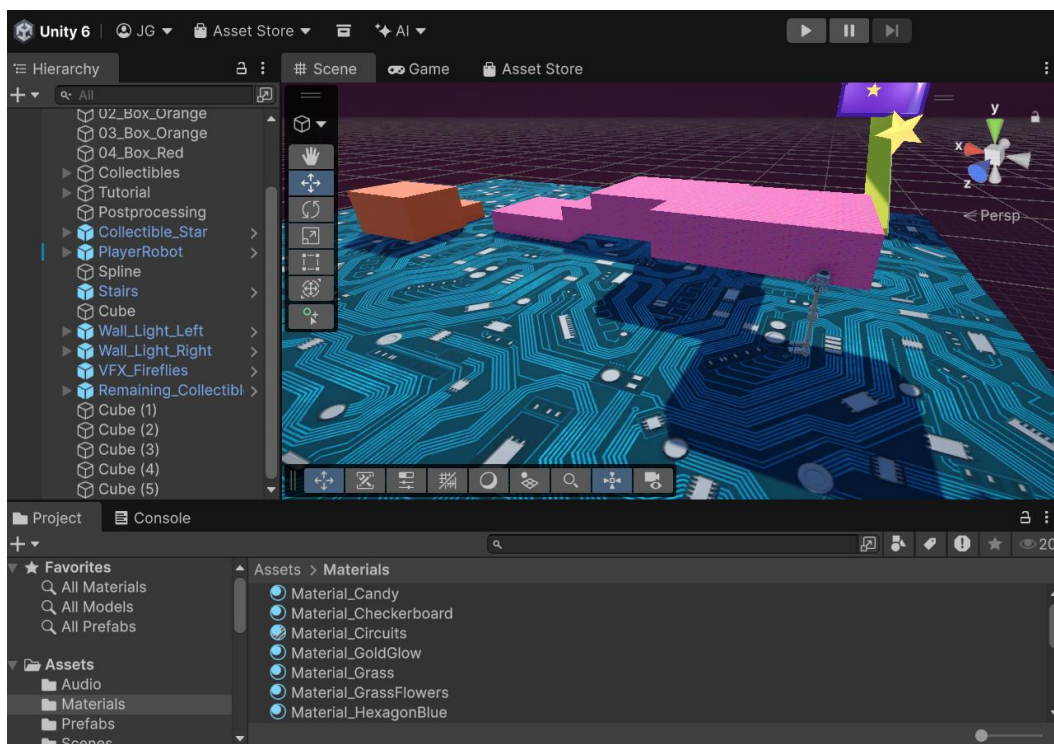


Figura 2-8: Captura de pantalla de la interfaz de Unity en el tutorial de iniciación.

Asimismo, Unity ofrece un amplio soporte para el desarrollo de aplicaciones inmersivas mediante la integración de tecnologías y estándares orientados a la realidad virtual y mixta. La compatibilidad con plataformas XR y su enfoque multiplataforma permiten que los entornos desarrollados puedan adaptarse a distintos dispositivos sin modificar sustancialmente la lógica del sistema [15]. Esta versatilidad convierte a *Unity* en una herramienta especialmente adecuada para proyectos centrados en la experimentación y el aprendizaje progresivo en entornos inmersivos.

Aunque existen otros motores de desarrollo en tiempo real, como el ya mencionado *Unreal Engine*, *Unity* se caracteriza por un aprendizaje más accesible y por un entorno de trabajo orientado a la creación de experiencias interactivas, lo que ha favorecido su adopción en numerosos proyectos de formación y simulación inmersiva.

2.6.2 Integración de Unity con dispositivos de realidad mixta

Uno de los aspectos que ha favorecido la adopción de *Unity* en el desarrollo de aplicaciones inmersivas es su capacidad de integración con dispositivos de realidad virtual y mixta mediante estándares y herramientas específicas orientadas a XR. En este contexto, *OpenXR* actúa como intermediario entre el entorno virtual desarrollado y el hardware de visualización e interacción, permitiendo gestionar de forma unificada el seguimiento del usuario, la interacción y la representación del entorno [16]

Para el presente trabajo, la plataforma Unity se integra con dispositivos de realidad mixta del ecosistema de Meta, lo que permite el desarrollo de aplicaciones compatibles con visores de la familia *Meta Quest*. Esta integración se apoya en el uso de los estándares abiertos de *OpenXR*, que facilitan la comunicación entre el motor de desarrollo y el dispositivo, reduciendo la dependencia de soluciones propietarias y favoreciendo la portabilidad del entorno desarrollado actuando como intermediario como se muestra en el esquema de la Figura 2-9.

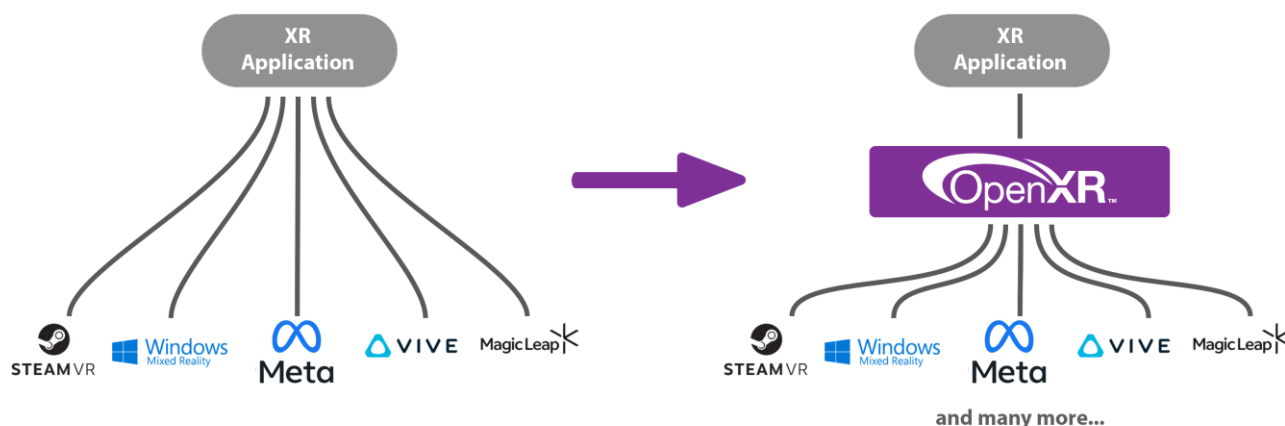


Figura 2-9: Capacidades de *OpenXR* [17]

La utilización de estas tecnologías permite a *Unity* gestionar aspectos fundamentales de la experiencia inmersiva, como el seguimiento de la posición y orientación del usuario, la detección de movimientos de manos o controladores y la correcta superposición de elementos virtuales sobre el entorno físico. Estas capacidades resultan esenciales en aplicaciones de realidad mixta, donde la coherencia entre las acciones físicas del usuario y la respuesta del sistema influye directamente en la calidad de la experiencia y en el proceso de aprendizaje.

En conjunto, la compatibilidad de *Unity* con dispositivos de realidad mixta y su integración mediante estándares abiertos proporciona un marco de desarrollo flexible y robusto, adecuado para la creación de entornos como el propuesto en este Trabajo Fin de Grado.

2.7 Equipamiento inmersivo para entornos virtuales

El desarrollo de entornos virtuales inmersivos aplicados al aprendizaje tecnológico requiere el empleo de un conjunto de dispositivos hardware capaces de proporcionar visualización tridimensional, interacción con el entorno y retroalimentación al usuario. La combinación adecuada de estos elementos permite configurar sistemas inmersivos que favorecen la comprensión espacial, la adquisición de habilidades técnicas y la simulación de procedimientos complejos en contextos formativos.

En este apartado se describen los principales equipos utilizados en sistemas de realidad virtual y mixta orientados a la formación y al entrenamiento técnico. El objetivo es presentar las características y funciones de cada dispositivo dentro de una configuración inmersiva, sin entrar en detalles de integración o implementación, que se abordan en capítulos posteriores.

2.7.1 Dispositivos de visualización

Los dispositivos de visualización constituyen el elemento central de cualquier sistema de realidad virtual o mixta, ya que permiten al usuario percibir el entorno simulado de forma inmersiva. En la actualidad, los visores de realidad virtual autónomos se han consolidado como una solución eficaz para aplicaciones educativas y de entrenamiento técnico, al integrar en un único dispositivo la visualización, el procesamiento y el seguimiento del usuario.

Entre estos dispositivos destacan las gafas Meta Quest 3 (Figura 2-10), que ofrecen visualización estereoscópica, seguimiento de movimiento y funcionamiento autónomo sin necesidad de equipos externos. Este tipo de visores permite al usuario desplazarse y orientarse dentro del entorno virtual de manera natural, facilitando una experiencia inmersiva adecuada para aplicaciones de aprendizaje tecnológico [18].



Figura 2-10: Equipo de visualización Meta Quest 3 junto con los controladores manuales [18]

Además de la visualización, estos dispositivos incorporan sistemas de seguimiento de posición y orientación del usuario, así como controladores manuales que permiten la interacción básica con los elementos del entorno virtual. Estas capacidades hacen que los visores autónomos sean especialmente adecuados para entornos formativos, donde se busca un equilibrio entre facilidad de uso, movilidad y nivel de inmersión.

El empleo de dispositivos de visualización de este tipo constituye la base sobre la que se integran el resto de los elementos del sistema inmersivo, como los dispositivos de interacción física, la retroalimentación háptica y los sistemas de locomoción, que se describen en los apartados siguientes.

2.7.2 Sistemas de retroalimentación háptica

La retroalimentación háptica constituye un elemento clave en los entornos inmersivos avanzados, ya que permite complementar la información visual y auditiva mediante estímulos físicos percibidos por el usuario. En el contexto de la realidad virtual y la realidad mixta, los sistemas hápticos tienen como objetivo mejorar la sensación de presencia y facilitar una interacción más natural con los elementos del entorno virtual, especialmente en aplicaciones de entrenamiento y aprendizaje procedimental [19].

Los sistemas hápticos pueden adoptar distintas formas, desde dispositivos de vibración y resistencia hasta soluciones que simulan el contacto físico o el peso de objetos virtuales. En el ámbito de la interacción manual, existen dispositivos diseñados para proporcionar *feedback* físico durante la manipulación de objetos, permitiendo al usuario percibir fuerzas, impactos o resistencias asociadas a sus acciones. Este tipo de retroalimentación resulta especialmente relevante en fases de iniciación, ya que refuerza la correspondencia entre la acción realizada y la respuesta del sistema.

Un ejemplo representativo de este enfoque son los dispositivos desarrollados por *ProTubeVR* como el *ForcetubeVR* de la Figura 2-11, que combinan soportes físicos y sistemas de resistencia para simular la interacción con herramientas u objetos virtuales. Este tipo de soluciones ha sido ampliamente utilizado en entornos de entrenamiento inmersivo, donde el realismo en la manipulación contribuye a una mayor transferencia de habilidades y a una mejor comprensión del entorno virtual [20].



Figura 2-11: Dispositivo ProtubeVR Gunstock [20]

Además de los dispositivos de interacción manual, existen sistemas hápticos corporales, como chalecos como el de la Figura 2-12, capaz de proporcionar estímulos físicos distribuidos por distintas zonas del cuerpo. Estos sistemas permiten representar eventos del entorno virtual mediante vibraciones o impactos controlados, incrementando la inmersión del usuario y reforzando la percepción de interacción dentro del entorno mixto [21].



Figura 2-12: Chaleco Bhaptics Tactsuit x40 [20]

Por último, como dispositivo corporal, los guantes hápticos son dispositivos diseñados para llevar la retroalimentación táctil directamente a las manos del usuario, que constituyen la principal vía de interacción con los objetos virtuales en entornos inmersivos. Su utilidad radica en que permiten reforzar acciones como agarrar, soltar, tocar o encajar elementos, haciendo que la relación entre el gesto físico y la respuesta del sistema resulte más clara e intuitiva. Dentro de este tipo de soluciones destacan los guantes *bHaptics TactGlove*, capaces de generar vibraciones localizadas en distintas zonas de la mano y especialmente adecuados para aplicaciones de realidad virtual y mixta orientadas a la manipulación directa de objetos, el aprendizaje interactivo y la mejora de la inmersión.



Figura 2-13: Guantes *bHaptics Tactglove DK2* [21]

Por lo tanto, los sistemas de retroalimentación háptica desempeñan un papel fundamental en la mejora de la experiencia del usuario en entornos de realidad mixta. Su integración en sistemas de iniciación y entrenamiento facilita el aprendizaje de las dinámicas de interacción, reduciendo la desconexión entre el entorno físico real y los elementos virtuales, y contribuyendo a una adaptación más progresiva del usuario a este tipo de tecnologías.

2.7.3 Sistemas de locomoción

Los sistemas de locomoción permiten simular el movimiento del usuario dentro del espacio virtual de forma controlada, manteniendo la seguridad y reduciendo la necesidad de grandes áreas físicas reales.

Entre las soluciones existentes destacan las plataformas de locomoción omnidireccional, diseñadas para permitir el desplazamiento continuo del usuario en cualquier dirección mientras permanece en una posición física limitada. Este tipo de sistemas ha sido ampliamente estudiado en el ámbito de la realidad virtual por su capacidad para aumentar la sensación de presencia y mejorar la coherencia entre el movimiento físico y el virtual.

Un ejemplo representativo de estas soluciones son las plataformas desarrolladas por *KAT VR* (Figura 2-14), que permiten simular el desplazamiento del usuario mediante una superficie deslizante combinada con sistemas de sujeción. Estas plataformas han sido empleadas principalmente en aplicaciones de simulación y entrenamiento, donde el movimiento forma parte del proceso de aprendizaje y contribuye a una mayor inmersión del usuario [22].



Figura 2-14: Plataforma de locomoción *KAT VR* [22]

El uso de sistemas de locomoción omnidireccional permite, además, mitigar algunos de los problemas asociados a otros métodos de desplazamiento virtual, como la desorientación espacial o la desconexión entre el movimiento visual y la acción física del usuario. Diversos estudios indican que la integración del desplazamiento real con la navegación virtual puede mejorar la experiencia del usuario y reducir efectos adversos como el mareo por movimiento, especialmente en entornos de entrenamiento prolongados [23].

2.8 Aplicaciones de la realidad mixta en el ámbito naval y de defensa

En el ámbito naval y de defensa, la realidad mixta ha sido aplicada principalmente como herramienta de apoyo al entrenamiento, la formación técnica y el mantenimiento de sistemas complejos. Diversas organizaciones han llevado a cabo prototipos y aplicaciones prácticas que ponen de manifiesto el interés creciente por estas tecnologías en contextos operativos y formativos.

Uno de los casos más conocidos es el uso de dispositivos de realidad mixta por parte de la *U.S. Navy*, donde se han empleado visores de realidad mixta para apoyar tareas de mantenimiento y formación técnica a bordo de buques. Mediante la superposición de información digital sobre el entorno real, los operarios pueden visualizar instrucciones, esquemas y procedimientos directamente sobre los equipos, reduciendo errores y tiempos de intervención [24].

Asimismo, el Departamento de Defensa de Estados Unidos ha impulsado el uso de tecnologías de realidad mixta en programas de entrenamiento y simulación, especialmente mediante el empleo de dispositivos como *HoloLens* para la instrucción técnica y el apoyo a tareas complejas (Figura 2-15). Estos sistemas se han utilizado tanto en entornos de formación como en escenarios controlados de entrenamiento, con resultados positivos en términos de comprensión y eficiencia.



Figura 2-15: Dotación del USS Michael Murphy empleando equipos VR para adiestramiento [24]

En el contexto europeo, organismos como la *NATO Science and Technology Organization* han analizado el potencial de la realidad mixta y la realidad extendida para aplicaciones de entrenamiento militar y formación técnica. A través de estudios y demostradores tecnológicos, se ha destacado la utilidad de estas herramientas para la simulación de procedimientos, la preparación previa a operaciones y la formación en sistemas complejos, manteniendo siempre un enfoque no operativo y experimental [25].

También en el ámbito industrial-militar, empresas como *Lockheed Martin* han incorporado tecnologías de realidad mixta en procesos de fabricación y formación técnica, empleando estas herramientas para la visualización de sistemas complejos y la asistencia a tareas de ensamblaje y mantenimiento. Estas experiencias han servido como referencia para su posible aplicación en entornos de defensa y formación técnica avanzada.

3 DESARROLLO

3.1 Descripción del apartado

Este apartado describe el proceso seguido para llevar a cabo el proyecto, desde las primeras etapas de aprendizaje hasta el refinamiento final de las escenas. El desarrollo se ha estructurado en tres fases principales, aplicadas de manera progresiva e iterativa.

Cada una de estas fases cumple un papel fundamental en la construcción del proyecto y contribuye de forma gradual a su implementación completa. A continuación, se presenta un resumen de cada etapa:

- Familiarización de conceptos básicos: esta primera fase se centró en adquirir una base sólida de trabajo con la plataforma y el entorno de desarrollo. Durante este periodo se realizaron pruebas iniciales para comprender la configuración del proyecto, la lógica de interacción y los componentes principales (seguimiento, colocación de objetos, colisiones y códigos base). El objetivo fue reducir la curva de aprendizaje y establecer una metodología de trabajo estable para el resto del proyecto.
- Diseño preliminar de las escenas: una vez asentados los fundamentos, se elaboró un diseño inicial de todas las escenas que componen el proyecto. En esta etapa se definieron los objetivos de cada prueba y su flujo general de interacción, construyendo de manera progresiva versiones funcionales que permitiesen validar la viabilidad de cada actividad antes de entrar en ajustes funcionales o mejoras estéticas.
- Perfeccionamiento iterativo: la fase final consistió en mejorar progresivamente las escenas mediante un enfoque no lineal. A medida que avanzaba el desarrollo, las habilidades adquiridas permitían introducir refinamientos en distintas partes del proyecto, por lo que se fue alternando entre escenas para aplicar mejoras incrementales (ajustes de físicas y colisiones, *feedback* visual, reorganización de componentes y correcciones derivadas de pruebas repetidas). Este enfoque permitió que todas las escenas evolucionasen en paralelo hasta alcanzar un resultado más sólido y consistente.

3.2 Introducción a Unity

En este apartado se introduce la instalación y puesta a punto del motor de desarrollo Unity, que será la herramienta principal empleada para la realización del presente Trabajo Fin de Grado, para que se adapte a las necesidades y requerimientos de los escenarios que se van a llevar a cabo. A través de este entorno de desarrollo se llevará a cabo tanto el diseño y la construcción de los distintos escenarios de realidad mixta, como la programación de la lógica de funcionamiento e interacción del sistema propuesto.

3.2.1 Instalación de la máquina virtual y creación del proyecto

El motor de desarrollo *Unity* se distribuye y gestiona a través de la aplicación *Unity Hub*. Esta herramienta centraliza la instalación de versiones del motor, la gestión de módulos adicionales y la creación y organización de proyectos.

El proceso comienza con la descarga e instalación de *Unity Hub* desde la página oficial del desarrollador (Figura 3-1). Una vez instalado, *Unity Hub* permite seleccionar la versión del motor más adecuada para el proyecto, así como los módulos necesarios en función de la plataforma objetivo. En el caso de este trabajo, se ha seleccionado la versión 6000.3.6f1, compatible con el desarrollo de aplicaciones de realidad extendida, garantizando la integración con dispositivos de realidad mixta.

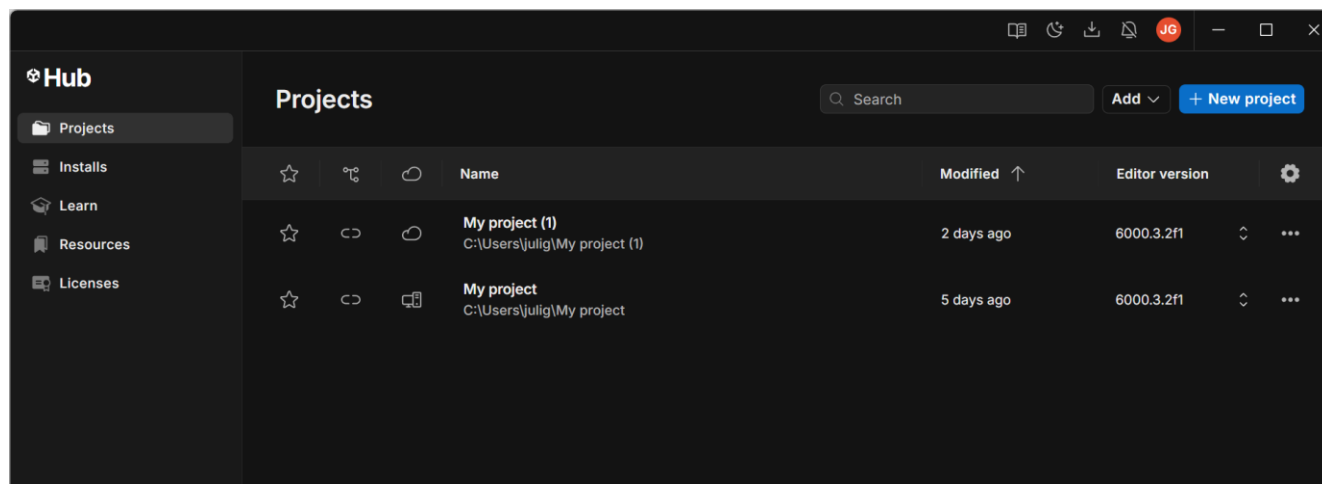


Figura 3-1: Interfaz inicial de UnityHub

Tras completar la instalación del motor, se procede a la creación de un nuevo proyecto desde *Unity Hub* (Figura 3-2). Durante este proceso se definen aspectos fundamentales como el nombre del proyecto, su ubicación en el sistema de archivos y la plantilla inicial. La selección de una plantilla adecuada permite disponer desde el inicio de una configuración básica optimizada para el trabajo con entornos tridimensionales, facilitando el desarrollo posterior de escenarios en realidad mixta.

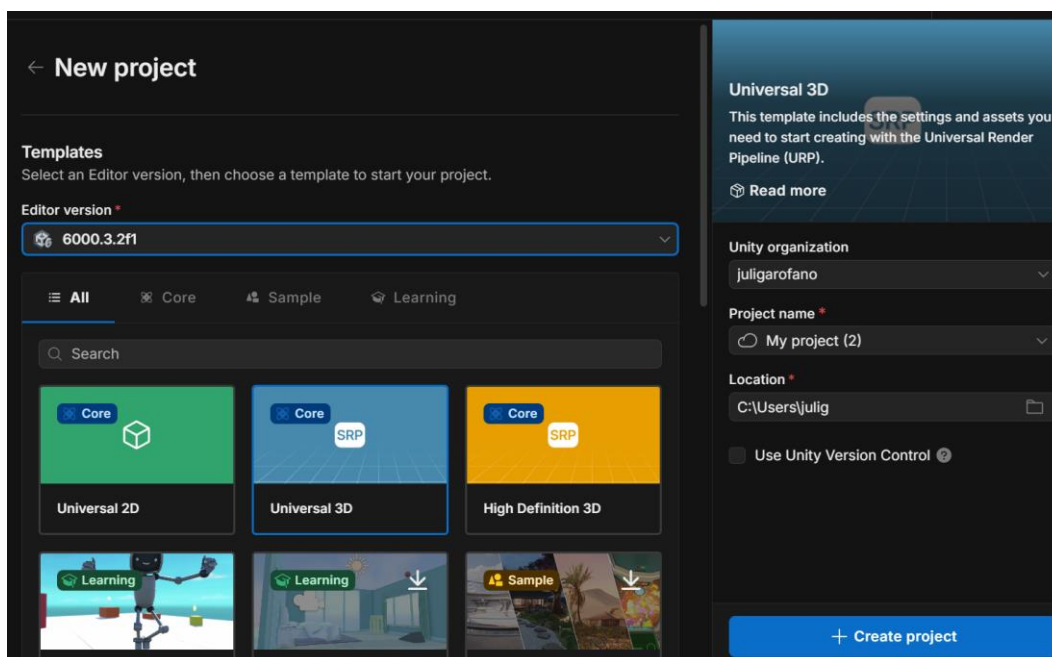


Figura 3-2: Selección del tipo de proyecto

Una vez creado el proyecto, se accede al entorno de trabajo de Unity, cuya interfaz está compuesta por diferentes paneles y herramientas que permiten diseñar, organizar y programar los distintos elementos del sistema. A continuación, se describen los componentes principales de dicha interfaz, los cuales serán utilizados de forma recurrente a lo largo del desarrollo del proyecto, los componentes y escenas que aparecen en las imágenes no forman parte del proyecto final, forman parte de un proyecto de prueba previo a la versión final.

- La vista de escena (*Scene View*) (Figura 3-3) constituye el espacio principal de diseño, donde se representan y manipulan los objetos tridimensionales que conforman el entorno virtual. En esta vista es posible posicionar, rotar y escalar los elementos de la escena, así como configurar su disposición espacial en relación con el entorno físico que se desea simular o complementar.

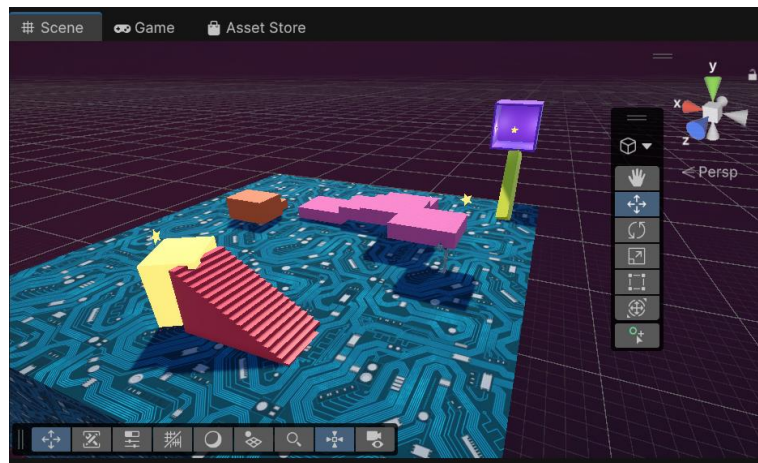


Figura 3-3: Vista de Escena en Unity

- La vista de juego (*Game View*) muestra una representación del resultado final desde el punto de vista del usuario, permitiendo comprobar en tiempo real el comportamiento de la escena durante la ejecución de la aplicación. Esta vista resulta especialmente útil para validar la interacción y la correcta visualización de los elementos virtuales. En el caso de este trabajo, al tratarse de realidad mixta, el *Game View* resulta poco útil debido a que la escena se construye una vez compilada en el equipo de VR por lo que nos mostrará una pantalla en negro.
- La barra de herramientas (Figura 3-4) proporciona acceso rápido a las funciones básicas de edición y control, como las herramientas de redimensión, ubicación, rotación y transformación de objetos, y los controles de ejecución del proyecto.

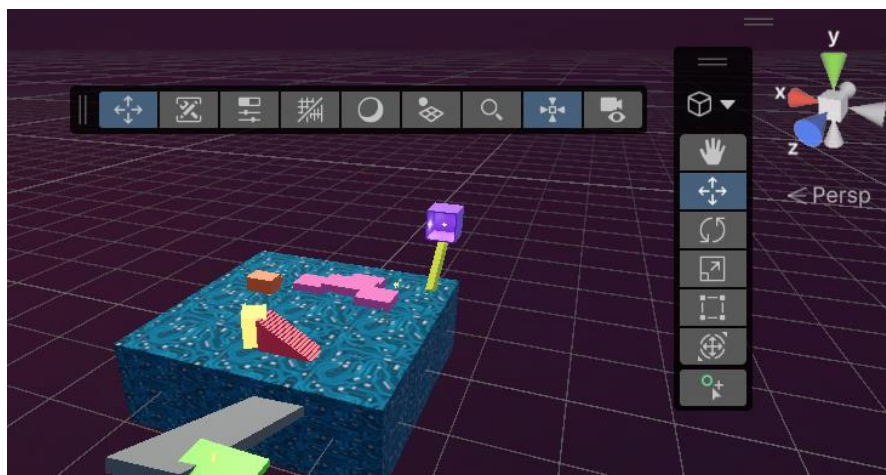


Figura 3-4: Barras de herramientas y ejes de orientación espacial

- El panel de jerarquía (*Hierarchy*) (Figura 3-5) muestra la estructura organizativa de la escena, listando todos los objetos que la componen y su relación jerárquica. Este panel permite gestionar de manera ordenada los distintos elementos del escenario, algo fundamental en proyectos que incluyen múltiples objetos interactivos.

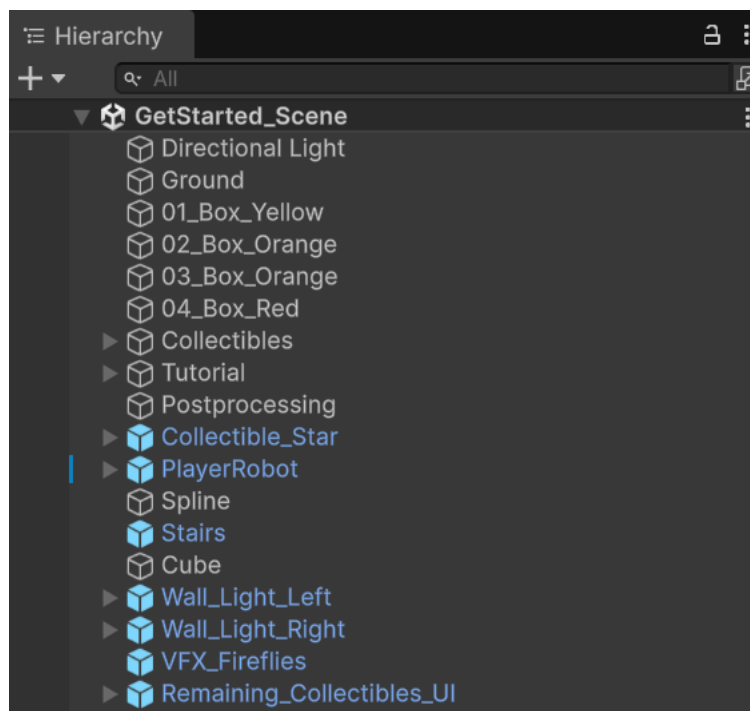


Figura 3-5: Ventana de Jerarquía

- El inspector de componentes (*Inspector*) (Figura 3-6) permite visualizar y modificar las propiedades de cada objeto seleccionado. En este panel se configuran los componentes asociados a los objetos, tales como transformaciones, materiales, códigos o sistemas de interacción, constituyendo una de las herramientas más importantes para la personalización del comportamiento de los elementos del entorno.

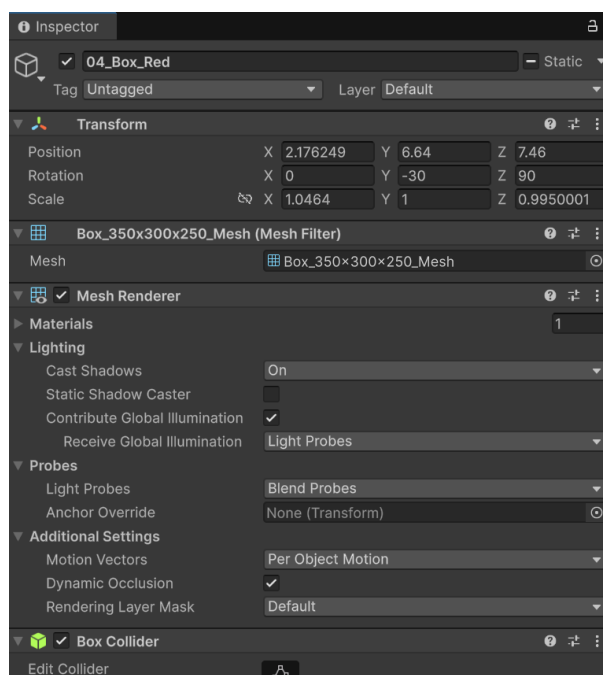


Figura 3-6: Inspector de componentes

- El panel de proyectos o ventana de *Assets (Project)* (Figura 3-7) muestra la estructura de archivos del proyecto y contiene todos los recursos utilizados en el desarrollo, tales como modelos tridimensionales, texturas, materiales, códigos, escenas y otros elementos auxiliares. Esta ventana actúa como el repositorio central de contenidos, permitiendo importar, organizar y gestionar los distintos *assets* que componen la aplicación.

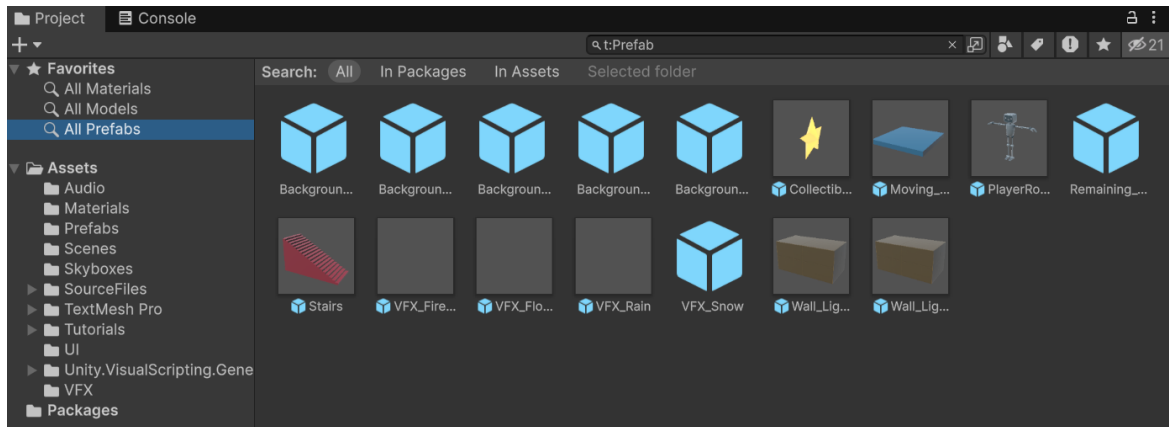


Figura 3-7: Ventana de Proyecto

3.2.2 Sincronización de Unity con el entorno Meta

Para que un proyecto hecho en Unity pueda ejecutarse en el equipo *Meta Quest 3*, primero hay que hacer una preparación básica: instalar los paquetes adecuados y activar el soporte de XR (realidad extendida). Puede verse como ponerle a Unity el idioma de *Quest*, para que el motor sepa cómo comunicarse con el visor y con sus funciones principales (seguimiento de cabeza, controladores, etc.). En este proyecto se trabajó con dos grupos de paquetes.

Por un lado, los paquetes de Meta (*Meta XR SDK*): Meta proporciona su propio conjunto de herramientas para *Unity*. Se distribuyen principalmente mediante el *Unity Package Manager (UPM)*, y una forma habitual de empezar es utilizar *Meta XR All-in-One SDK* (disponible en los canales oficiales de Meta/Unity), ya que incluye o referencia los módulos principales necesarios para *Quest* y evita tener que gestionar dependencias manualmente. A partir de este paquete se incorporan los componentes básicos para trabajar con el visor (por ejemplo, *rig* de cámara y utilidades de interacción). *Unity*, además, documenta qué paquetes de Meta existen para Quest y cómo se identifican dentro de UPM.

Además de lo anterior, Unity necesita sus propios paquetes para habilitar XR de forma general y elegir el estándar de ejecución. En concreto, se utilizaron:

- *XR Plug-in Management*: sirve para activar y gestionar el proveedor XR del proyecto desde los ajustes del editor de *Unity*.
- *OpenXR Plugin*: permite trabajar con *OpenXR*, el estándar más común para ejecutar aplicaciones XR en distintos dispositivos, incluido Quest.
- *Meta OpenXR*: añade extensiones específicas de Meta dentro del flujo *OpenXR* cuando se requieren capacidades concretas del ecosistema Quest.

Una vez instalados los paquetes, la configuración mínima para poder compilar y probar en el visor consistió en: (1) seleccionar *Meta Quest* como destino de compilación, (2) activar XR desde *XR Plug-in Management*, y (3) habilitar el proveedor escogido (en este caso, *OpenXR*). Con estos pasos, el proyecto queda listo para ejecutar una primera escena en el visor y continuar con el desarrollo del escenario.

3.3 Adquisición de Conceptos Básicos

Para iniciar el desarrollo del proyecto, fue necesario realizar una fase previa de familiarización centrada en una serie de conceptos principales, que constituyen la base sobre la que se construye el escenario. Esta etapa se considera parte del desarrollo porque permitió definir el flujo de trabajo, validar la viabilidad técnica de las funcionalidades previstas y reducir errores durante la implementación. Además, al tratarse de la primera toma de contacto con este entorno, se dedicó tiempo a consultar documentación y tutoriales oficiales, asegurando que las decisiones posteriores se apoyasen en prácticas recomendadas.

3.3.1 Camera Rig y las interacciones con los entornos VR

Para comprender la estructura del *OVRCameraRig*, el flujo de interacción básico y el uso de *Building Blocks*, se siguió el tutorial oficial *Hello World* de Meta [26]. Este tutorial se empleó como punto de partida porque permite validar, de forma guiada, que el proyecto está correctamente configurado para Quest y que la escena dispone de los elementos mínimos para comenzar a desarrollar (*rig* de cámara + una interacción funcional).

En primer lugar, el tutorial introduce el concepto de *Camera Rig* como representación del usuario dentro de la escena. A nivel práctico, se reemplaza la cámara por defecto de *Unity* (eliminando *Main Camera*) y se añade el *rig* mediante el menú de herramientas de Meta (*Meta XR Tools* → *Building Blocks*), seleccionando el bloque *Camera Rig*. Con ello queda incorporada al *Hierarchy* la estructura base desde la que se actualiza el tracking del visor y se referencian los puntos de interacción del usuario.

A continuación, el tutorial muestra cómo añadir una interacción real de manera controlada (como la de la Figura 3-8, incorporando el *Grab Interaction Building Block*, permitiendo que el objeto pueda ser agarrado y soltado cuando se emplea el equipo de Meta. Este bloque incluye una escena mínima de prueba (un cubo “*grabbable*”) y guía ajustes sencillos (como recolocar el objeto para situarlo en el campo de visión).

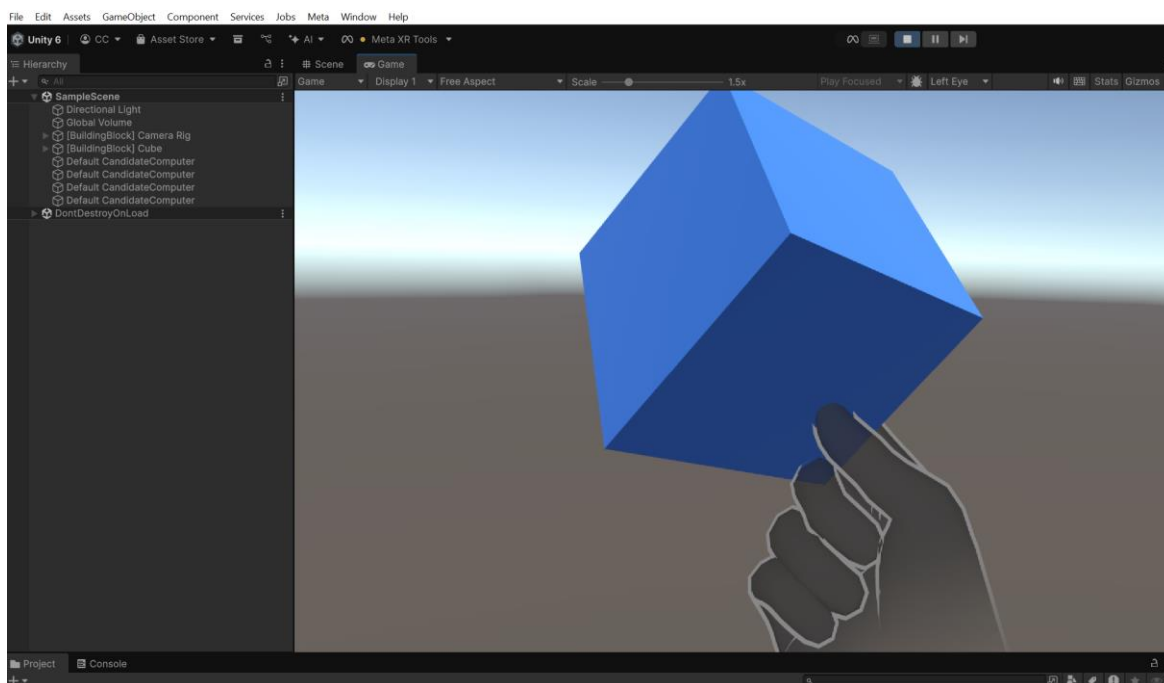


Figura 3-8: *Game View* del resultado del tutorial *Hello World* de *Meta For Developers* [26]

La importancia de realizar este tutorial radica en que actúa como una verificación inicial y, a la vez, como una introducción al patrón de desarrollo recomendado por Meta: construir la escena a partir de componentes modulares (*Building Blocks*) que encapsulan configuraciones correctas y aceleran el

arranque. Esto reduce errores típicos de la fase inicial (por ejemplo, componentes mal configurados o interacciones incompletas), paquetes mal instalados, y permite centrar el esfuerzo desde el principio en la lógica específica del proyecto. Pese a que se trate simplemente de un escenario en el que solo existe un cubo que se puede agarrar y soltar, este paso ha resultado imprescindible para tener una base de partida sobre la que comenzar el desarrollo del proyecto.

El componente *OVRCameraRig* estará presente en todas las escenas del proyecto, ya que actúa como la base del sistema de seguimiento e interacción del usuario, debido a que gracias a este componente se mantienen de forma consistente el posicionamiento de la cámara, el *tracking* de cabeza y manos/controladores, y los elementos necesarios para que la experiencia de realidad mixta funcione correctamente en cada una de las pruebas.

3.3.2 *Passthrough*

El *passthrough* es la funcionalidad que permite visualizar el entorno real a través de las cámaras del visor y, simultáneamente, integrar sobre esa visión elementos virtuales generados en *Unity*. En una aplicación de Realidad Mixta, este recurso permite que la experiencia se desarrolle sobre el espacio físico real, manteniendo referencias visuales (habitación, mobiliario, límites) y favoreciendo una interacción más natural.

Desde el punto de vista técnico, Meta describe el *passthrough* como una capa que se compone a nivel del sistema, lo que implica que la aplicación no trabaja con el vídeo como si fuese una textura convencional, sino que habilita una *passthrough layer* que el *runtime* mezcla con la escena virtual. Esta aproximación está orientada a ofrecer una visualización cómoda y consistente del entorno real dentro del visor.

Para comprender su uso en *Unity*, tras completar el tutorial *Hello World* (empleado como iniciación al *rig* y a las interacciones básicas), se dedicó tiempo a seguir las guías y directrices de Meta sobre *passthrough*. Estas páginas están planteadas precisamente como el siguiente paso lógico: una vez validada la configuración mínima del proyecto, se introduce cómo habilitar y probar el *passthrough* en una escena *Unity* y cómo comenzar a integrarlo en el flujo de desarrollo.

Finalmente, la incorporación de *passthrough* condiciona decisiones de diseño del escenario de realidad mixta: los elementos virtuales deben mantener buen contraste y legibilidad sobre fondos reales variables, respetar escalas y distancias cómodas para el usuario y evitar una sobrecarga visual innecesaria. En conjunto, el *passthrough* constituye la base visual que habilita la Realidad Mixta y permite que el resto de componentes del proyecto se sitúen y perciban en relación directa con el entorno real.

Al igual que el ya mencionado *OVRCameraRig* el componente *OVRpassthroughLayer* estará presente en todas las escenas de este trabajo ya que permitirá mantener la naturaleza de realidad mixta en cada una de ellas.

3.3.3 *Spatial Anchors*

Para completar la adaptación a la lógica de funcionamiento del programa, se abordó el concepto de *spatial anchors*. Un *spatial anchor* puede entenderse como un punto de referencia vinculado al mundo real, que permite fijar la posición y orientación de un objeto virtual respecto al entorno físico. Gracias a ello, es posible situar elementos digitales de forma coherente dentro del espacio real y mantener su alineación incluso cuando el usuario se desplaza o el sistema de tracking realiza pequeñas correcciones.

Para familiarizarse con su funcionamiento se siguió el tutorial oficial *Spatial Anchors Basic Tutorial* de Meta, en el que se recorre el flujo básico de trabajo en *Unity* (creación del ancla, asociación de contenido, y operaciones como persistencia y recuperación). Este tutorial resultó especialmente útil porque no solo explica el concepto, sino que también introduce el estilo de programación habitual en

este tipo de desarrollos: una lógica apoyada en estados y operaciones que no siempre se completan de forma inmediata, al depender del *runtime* del dispositivo.

Por este motivo, el código y la estructura mostrados en el tutorial se emplearon como referencia base para el proyecto. En particular, sirve como ejemplo práctico de cómo organizar códigos que interactúan con funcionalidades XR (creación, verificación, guardado y carga de elementos), y ha sido un punto de partida para desarrollar posteriormente otros componentes del sistema con una arquitectura y un flujo de trabajo coherentes.

Finalmente, una vez comprendido el concepto y su aplicación práctica, se dispone de una base sólida para construir nuevos escenarios. En concreto, aprender a situar *spatial anchors* permite generar estructuras virtuales que quedan integradas en el mundo real de forma estable, facilitando que el contenido se perciba como parte del entorno y no como un elemento sin referencia física.

3.3.4 Programación en C#

En *Unity*, los objetos de la escena no tienen ninguna función específica, se comportan en función de los componentes que se les añaden desde el Inspector (físicas, colisiones, renderizado, audio, UI, etc.). Durante esta fase, una de las ideas clave ha sido entender que, cuando esos componentes no cubren lo que se necesita, el comportamiento se define creando códigos en C#. En la práctica, estos códigos se añaden como un componente más al objeto correspondiente y permiten controlar la lógica de cada escena; detectar entradas del usuario, gestionar estados, activar eventos, actualizar contadores o lanzar acciones en momentos concretos.

A la hora de programar, los comandos y funciones que se utilizan se apoyan principalmente en la documentación oficial de *Unity*, tanto en el *Unity Manual* [27] (para entender cómo funcionan los sistemas del motor) como en la *Unity Scripting API* (para consultar clases y métodos concretos como *GameObject*, *Transform*, *Rigidbody*, *Collider* o *SceneManager*). Además, como este proyecto se desarrolla para *Meta Quest* y contiene interacción VR/MR e integración háptica, también se recurre a la documentación de los SDK específicos utilizados: por un lado, *Meta XR* (por ejemplo, lectura de botones y entradas mediante *OVRInput* y configuración del *rig*), y por otro, el SDK de *bHaptics* para gestionar la reproducción de patrones de vibración en los dispositivos hápticos.

Por último, con el objetivo de mantener el proyecto ordenado y fácilmente consultable, los códigos desarrollados se almacenarán en un repositorio de *GitHub* [28], que habilita desde internet acceso claro al código. En el Anexo III se incluye una descripción sencilla del flujo general de todos los códigos elaborados, apoyada en diagramas y resúmenes, de modo que sirve como referencia rápida sin interrumpir el hilo principal de la memoria.

3.3.5 Entorno bHaptics

Para integrar dispositivos, o periféricos, hápticos en el proyecto, como guantes o el chaleco, fue necesario familiarizarse con el entorno de *bHaptics* y su flujo de trabajo. En *bHaptics* la vibración de los periféricos se trabaja con una estructura basada en una *haptic app*, una serie de *events* (eventos) y sus correspondientes *patterns* (patrones), que después se reproducen desde el proyecto cuando desde la escena se llama a una interacción concreta.

El primer paso consiste en instalar e importar el *plugin* oficial (el SDK) dentro del proyecto de *Unity*. Este *plugin* es el que permite que el proyecto pueda comunicarse con el ecosistema de *bHaptics* y lanzar la háptica de forma controlada. En paralelo, es necesario tener instalada la aplicación *bHaptics Player*, que actúa como centro de control, desde ahí se gestionan los dispositivos, se comprueba que están conectados correctamente y se valida que el sistema está listo para recibir y reproducir vibraciones (Figura 3-9).

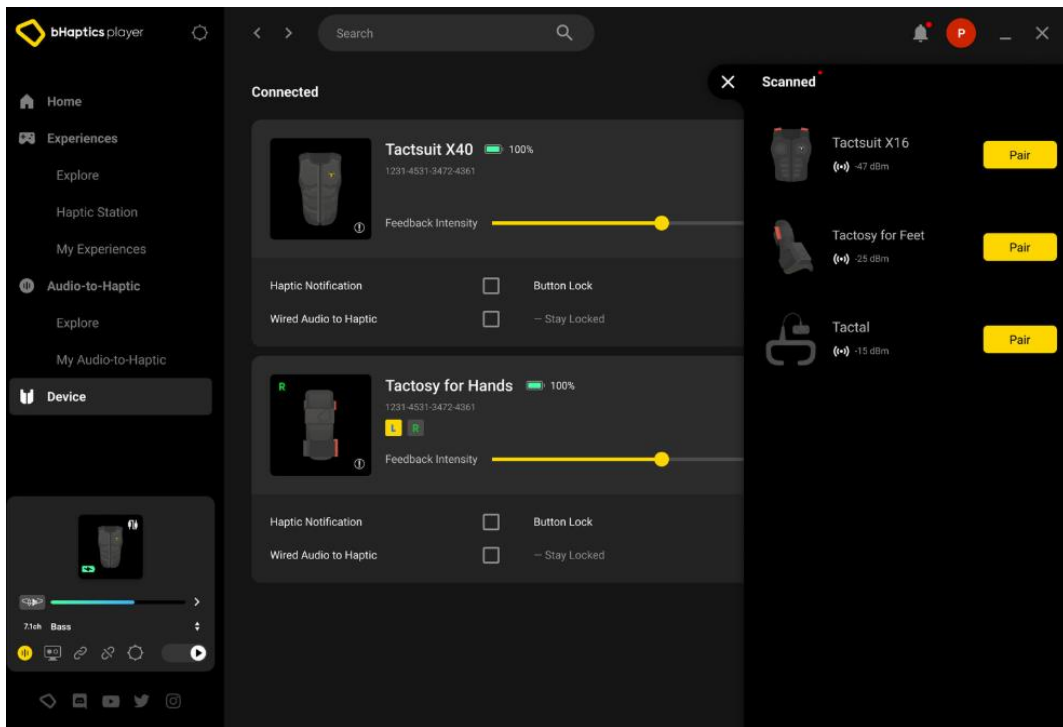


Figura 3-9: [29] Interfaz de la aplicación *bHaptics Player*

A continuación, para vincular el proyecto con *bHaptics*, hay que crear una cuenta en la plataforma y trabajar desde el *Developer Portal*, accesible desde la barra del editor de la interfaz de *unity* (Figura 3-10). En este portal se crea una *haptic app* asociada al proyecto, y antes de poder enlazarla se deben cumplir algunos requisitos: tener al menos un *haptic event* creado, como la vibración de los dedos del guante; generar una *API Key* (no viene por defecto) y publicar la aplicación mediante la opción de *deploy*. Una vez hecho esto, el portal proporciona el *App ID* y la *API Key*, que son los datos necesarios para enlazar el proyecto.

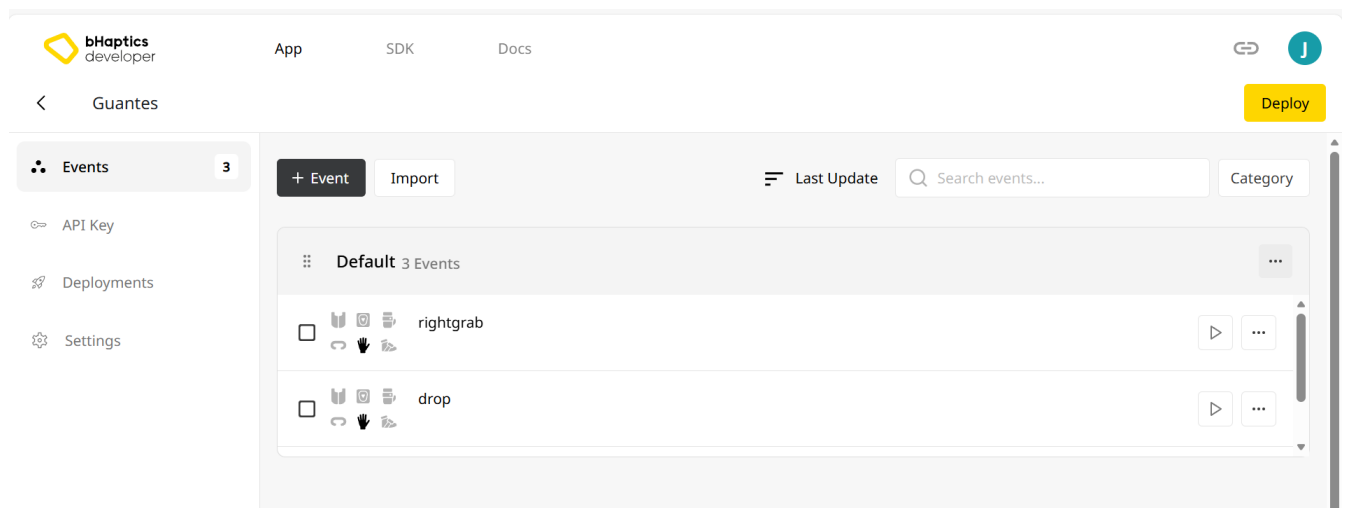


Figura 3-10: Gestión de los eventos hápticos en *bHaptics Developer*

Un detalle importante del flujo de trabajo es la incorporación a la escena de un *prefab* llamado [*bHaptics*], que prepara el entorno háptico antes de ejecutar cualquier función de vibración. Este objeto será común en todas las escenas del proyecto.

En cuanto a la creación de la háptica, el contenido se organiza mediante *events* y *patterns*. Los *patterns* se pueden entender como mensajes táctiles que se asignan a acciones del juego: confirmaciones,

errores, impactos, encajes, etc. La clave es diseñarlos para que sean cortos y reconocibles, de forma que realmente aporten información y no sean solo un adorno. Después, en el proyecto se reproducen esos *events* en el momento exacto en el que ocurre la interacción, manteniendo coherencia entre lo que se ve y lo que se siente. Estos *patterns* se configuran y definen en la página de *bHaptics designer*, en la que se definen las acciones hápticas de los propios dispositivos como la ubicación, duración e intensidad de las vibraciones (Figura 3-11).

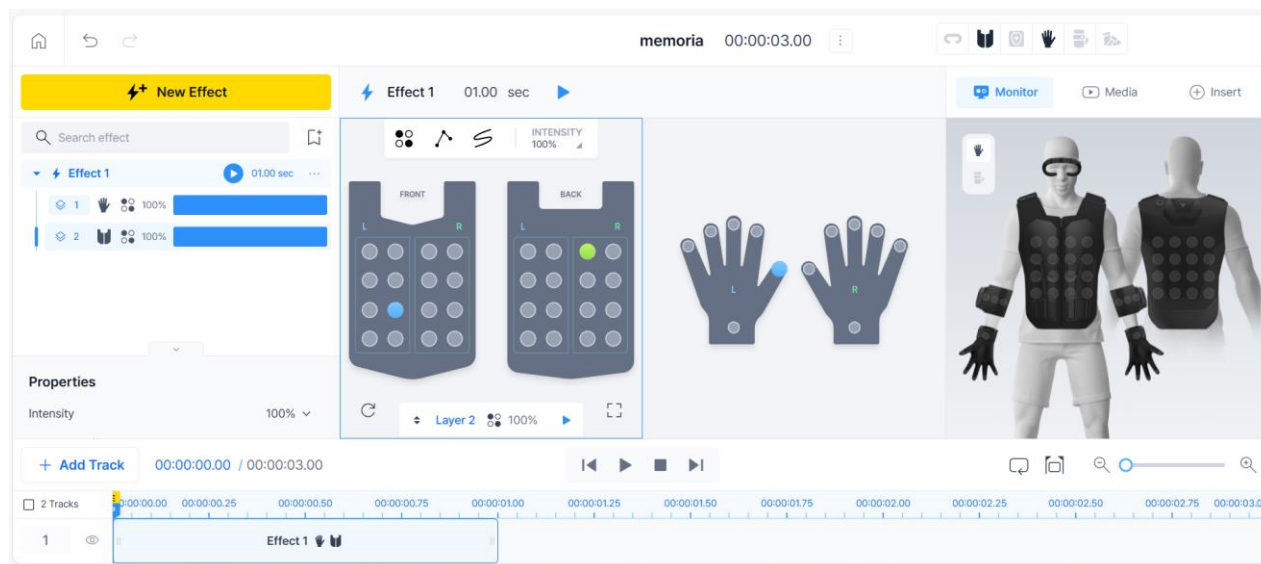


Figura 3-11: Configuración de los patrones hápticos en *bHaptics Designer* [30]

3.4 Construcción de las escenas

La construcción de un proyecto en realidad mixta como el del presente trabajo difiere en varios aspectos respecto a la elaboración de otro en un entorno de realidad virtual. Una de las diferencias principales se centra en el escenario. En realidad virtual, el entorno se diseña y construye dentro de la propia plataforma, como si de un plató de televisión se tratara. En cambio, en realidad mixta el escenario lo marca la ubicación en la que se encuentre el usuario en ese momento; por ello, el diseño de este proyecto busca adaptar su funcionalidad a cualquier tipo de entorno real.

Esto se consigue variando el comportamiento y la funcionalidad de los distintos elementos, aparatos y periféricos según la prueba que se quiera llevar a cabo. *Unity* hace esto posible mediante el entrelazamiento de distintas escenas dentro de una misma experiencia, de manera que, dependiendo de la escena activa, la construcción del escenario, la jugabilidad y el objetivo del entorno serán diferentes.

En este apartado se describe el diseño y la construcción del producto final del TFG a través del desarrollo y la conexión de cada una de las escenas que conforman el proyecto. Cada una contará con un diseño propio y un propósito concreto (por ejemplo, menú, pruebas de interacción o actividades específicas). En conjunto, estas escenas forman la experiencia completa del sistema y definen el recorrido que seguirá el usuario dentro del entorno de realidad mixta.

3.4.1 Elementos comunes

Antes de explicar la construcción de cada escena por separado, merece la pena comentar algunos elementos que se repiten a lo largo de todo el proyecto. De esta forma, se evita repetir varias veces las mismas ideas y queda más clara la base común sobre la que se han desarrollado las distintas pruebas.

Por un lado, todas las escenas parten del uso del *OVRCameraRig* y de los componentes relacionados con la realidad mixta, como el *Passthrough Layer*, que ya han sido explicados anteriormente (Página 32). Aun así, conviene mencionarlos de nuevo porque son la base de toda la experiencia, ya que permiten situar al usuario dentro del entorno y combinar los objetos virtuales con la visión del espacio real.

Además, para todas las pruebas se crea un panel translúcido con un texto introductorio en forma de pantalla al arrancar la escena. Su función es indicar al usuario qué tiene que hacer, cuál es el objetivo de la actividad, como construir el escenario y qué controles debe utilizar. Aunque es un elemento simple, ayuda bastante a que cada prueba se entienda rápido y a que la experiencia resulte más clara desde el primer momento. El texto se construye a partir de los siguientes componentes:

- Cuadro de texto (*TextMeshPro*). que hace posible la elaboración y visibilidad del texto. En ellos se fija el estilo de la letra, tamaño y color (Figura 3-12).
- *Canvas* con panel negro para asegurar la visibilidad del texto independientemente del entorno (Figura 3-12).

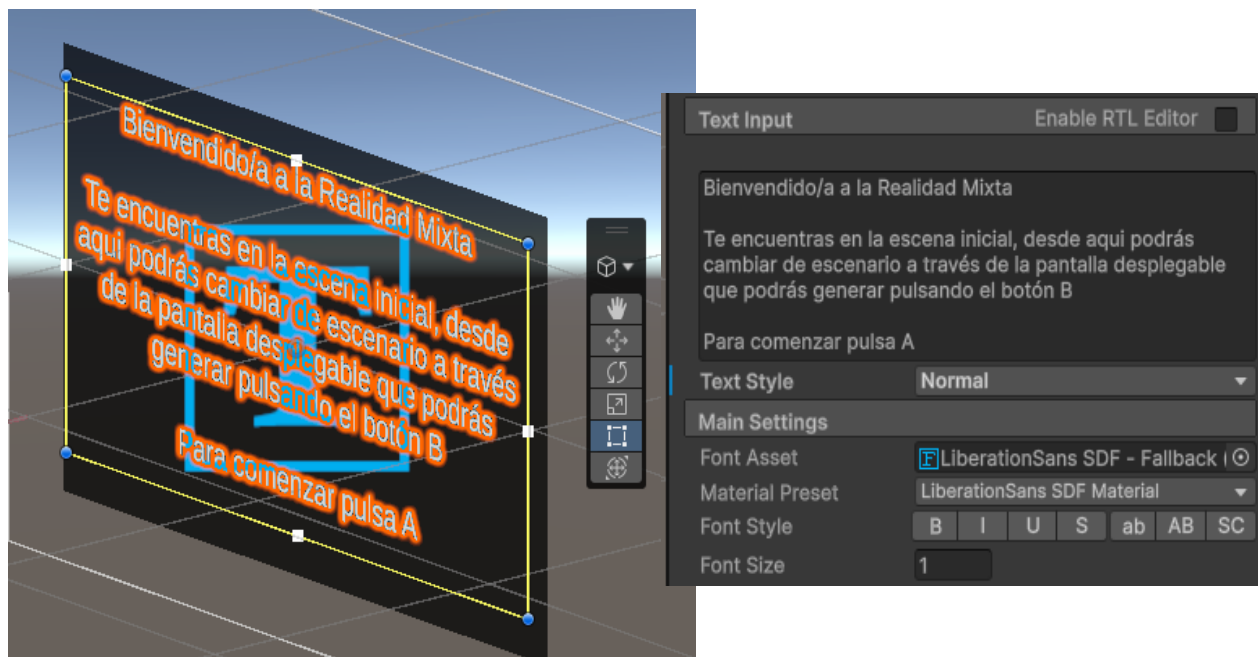


Figura 3-12: Panel introductorio escena inicial e inspector del texto

- Código (*HideIntroOnA*) que habilita que la pantalla desaparezca para empezar a desempeñar la función de la escena pulsando el botón A. Este código funciona destruyendo el *prefab* definido en el inspector (Figura 3-13) cuando recibe el input del mando.

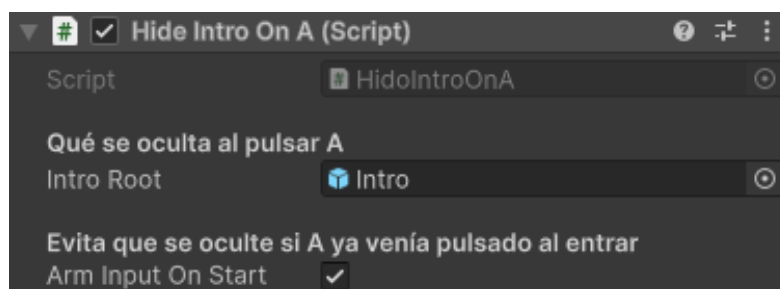
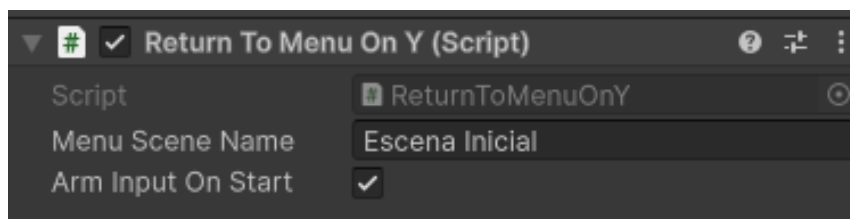


Figura 3-13: Inspector del código HideIntroOnA

Otro elemento común es el código *ReturnToMenuOnY*, que se ha incluido en todas las escenas para que el usuario pueda volver al menú inicial pulsando el botón Y del mando. Esto permite mantener una misma lógica de navegación en toda la aplicación y hace más cómodo cambiar de una prueba a otra. Cabe destacar que lo que particulariza a este código es que identifica la escena a la que transporta al usuario por el nombre concreto de la escena, que se introduce en el inspector de la Figura 3-14.

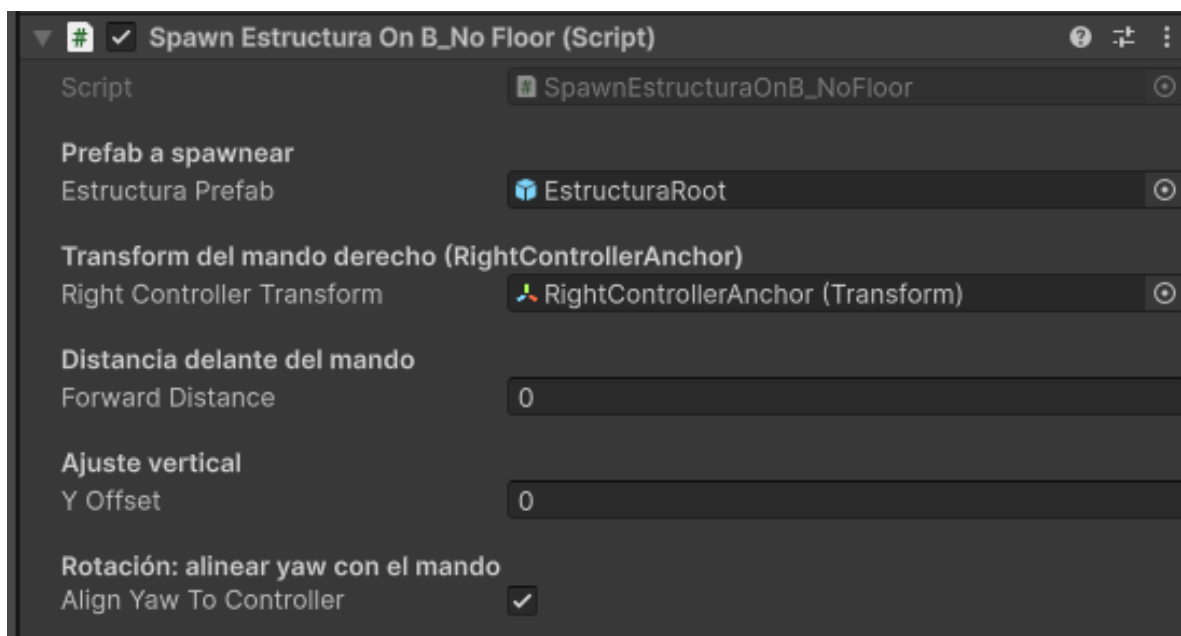
Figura 3-14: Inspector del código *ReturnToMenuOnY*

Por último, durante el desarrollo aparecen también componentes de Unity que se repiten en muchos objetos, como el *Mesh Filter* o el *Mesh Renderer*, de los objetos 3D. Estos componentes forman parte de la estructura básica de los objetos de Unity y hace posible la visualización de los mismos.

3.4.2 Menú Inicial

Esta escena actúa como la pieza central del sistema, ya que organiza el acceso al resto de escenas y sirve como punto de conexión entre todas las pruebas. De este modo, el menú no solo guía al usuario dentro de la experiencia, sino que también estructura el flujo general del proyecto y facilita el cambio entre escenas de forma clara e intuitiva. El menú se presenta como una pantalla con forma de televisión con cinco botones con imágenes de la prueba a la que dirige cada uno, los botones se pulsán usando el mando derecho del equipo de Meta Quest 3 como un mando a distancia con un puntero laser.

En primer lugar, el componente con el código de generación de la pantalla del menú de elección al pulsar el botón B del mando derecho (*SpawnEstructuraOnB_NoFloor*). Para que este funcione, como se observa en la Figura 3-15, se necesita un *prefab* a instanciar, en este caso el de la pantalla, y por otro lado una posición de referencia para situar el *prefab*. Además, el código habilita ajustar los parámetros de orientación y distancia con respecto al mando para ajustar como se va a generar la estructura. Por último, el código tiene una función de destruir al pulsar el botón X del mando izquierdo, permitiendo de esta manera que se pueda instanciar otra estructura en otra posición, si la primera no permitiera el uso correcto del objeto.

Figura 3-15: Inspector del código *SpawnEstructuraOnB_NoFloor*

La creación de la pantalla comienza con la elaboración de un componente de interfaz de usuario (UI) tipo *canvas* en el cual se introducen todos los componentes del menú inicial, que será los siguientes.

- El Panel será el componente que componga el fondo de la UI, sobre él se indica el color del fondo, y se establece la organización del resto de los componentes.
- Cuadros de texto TMP (*TextMeshPro*) en los que se introduce el nombre a los botones y un pequeño texto de guía para el usuario.
- Los botones son la parte clave y más compleja de esta escena. En ellos se introduce un código que les da la funcionalidad de que al ser pulsados con el gatillo del mando derecho del equipo *Meta quest 3* como con un mando a distancia, el usuario es transportado a la escena correspondiente. Este código, denominado *SceneLoader*, se introduce en el inspector de componentes de los propios botones y se configura introduciendo el nombre de la escena a la que se quiere cambiar en la ventana del inspector (Figura 3-16) al igual que el ya mencionado *ReturnToMenuOnY*. (Página 36).

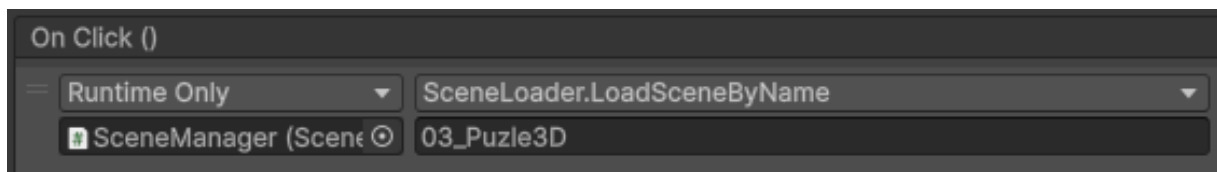


Figura 3-16: Aplicación del código *SceneLoader* en el inspector de los botones de la pantalla

Con el objetivo de mejorar la estética de la pantalla y la experiencia del usuario, los botones aparecen en la pantalla como imágenes de las escenas a las que dirige, por lo que en la jerarquía se introduce un objeto *raw image* con la imagen correspondiente, como se muestra en la Figura 3-22. Posteriormente, en el inspector de los botones se configura que se produzca un cambio de color en la imagen cuando los botones son pulsados y un sombreado cuando el puntero se encuentra apuntándolos.

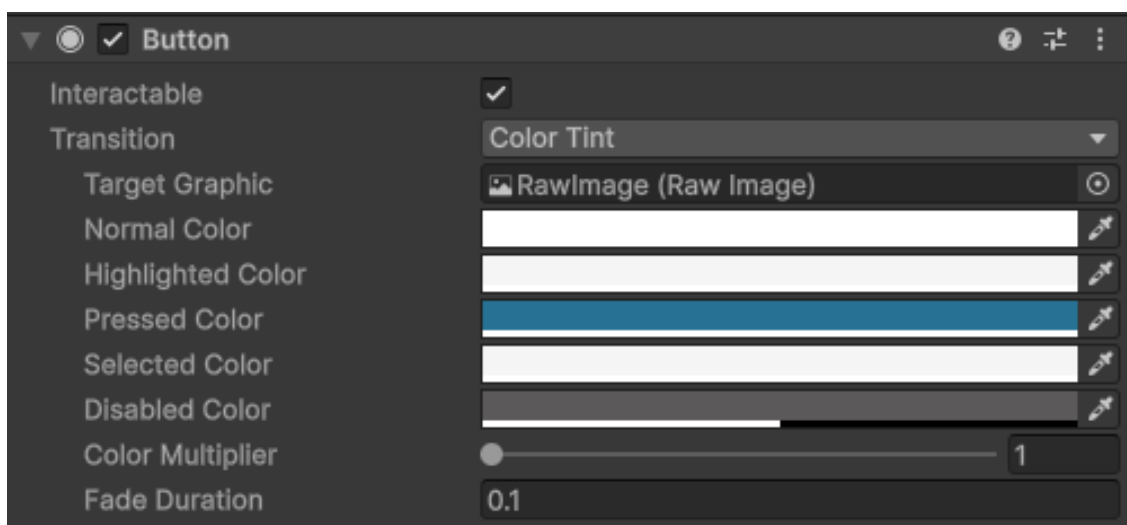
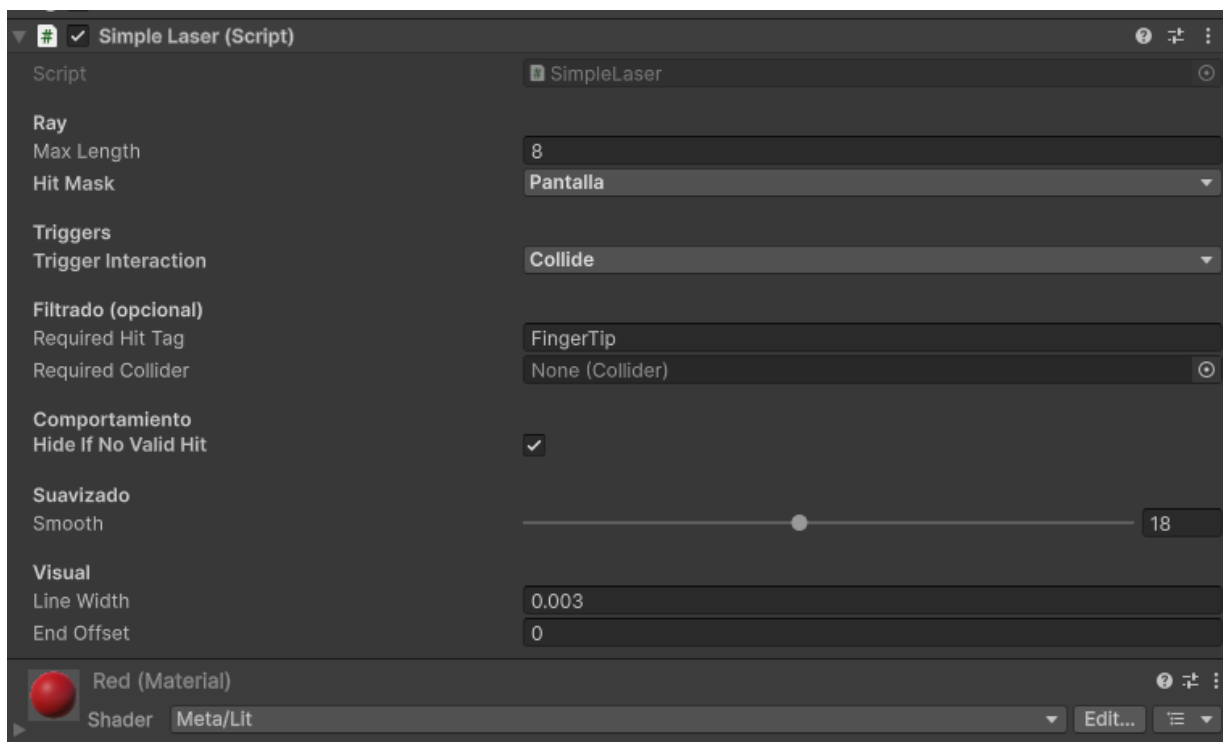
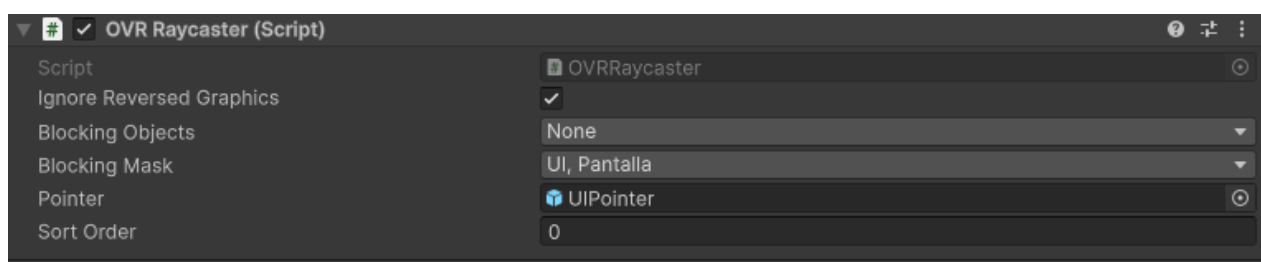


Figura 3-17: Configuración del sombreado de los botones al ser pulsados

Además de la propia pantalla, esta escena incluye otros elementos en la jerarquía que permiten la interacción del usuario con el panel. En concreto, dentro del *RightControllerAnchor* del *OVRCameraRig* se ha añadido el componente *UIPointer*, encargado de habilitar la interacción con la UI mediante un puntero tipo láser configurado con la elaboración del código *SimpleLaser*. Gracias a este código, cuando el usuario apunta con el mando hacia el panel, se genera un rayo visible que actúa como cursor y permite seleccionar los distintos botones del menú. Debido a su complejidad, para una mejor comprensión del inspector de la Figura 3-18, funcionamiento y flujo de este código, se recomienda consultar el Anexo III: Diagramas de flujo de los códigos del proyecto.

Figura 3-18: Inspector del código *SimpleLaser*

Por otro lado, en la configuración básica del *Canvas* se incluye el componente *Graphic Raycaster*, cuya función es permitir que la interfaz detecte apuntings e interacciones sobre los elementos gráficos (como los botones) mediante un sistema de *raycast*. Sin embargo, este componente está pensado principalmente para entradas simples como el cursor de un ratón y no se integra de forma directa con el sistema de puntero de los mandos de *Meta Quest 3*. Por ello, el paquete *Meta All-in-One* incorpora el componente *OVRRaycaster*, que habilita el uso de los mandos como un puntero a distancia sobre la pantalla. Por lo tanto, en este proyecto se sustituye el *Graphic Raycaster* por *OVRRaycaster* para garantizar una interacción correcta en realidad mixta. Para este proyecto se ha configurado para que el *raycaster* siga la posición del objeto *UIPointer* y funcione contra la capa de la pantalla como se muestra en la Figura 3-19.

Figura 3-19: Inspector del código *OVRRaycaster*

También con la creación del *Canvas*, automáticamente se crea un componente en la barra de jerarquía llamado *eventsystem*. Este se encarga de gestionar la interacción con la interfaz de usuario. Su función es detectar las acciones del usuario (como la pulsación de los botones de los mandos) y enviarlas al elemento correspondiente, en este caso, la pantalla del menú inicial. Gracias a este sistema, la pantalla puede responder correctamente a las entradas de los mandos, para este caso, se ha configurado este componente para que los botones de la pantalla se pulsen a través del gatillo del controlador derecho como se muestra en la Figura 3-20.

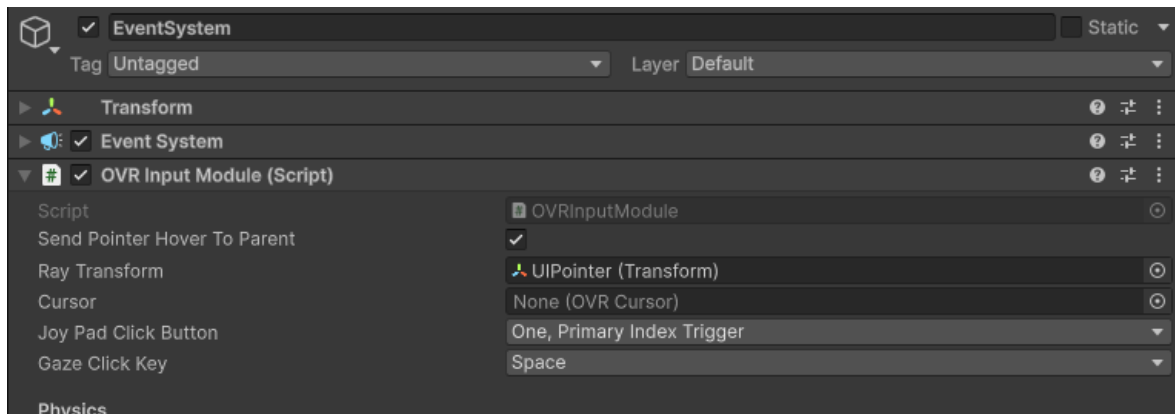


Figura 3-20: Inspector del *EventSystem*

Por último, para aportar un componente estético al menú, se adquirió en *Unity Asset Store* el objeto prefabricado *TV Led 30* (Figura 3-21). Este recurso consiste en un modelo de pantalla de televisión de aspecto realista sobre el que se sitúan los componentes mencionados anteriormente, lo que permite dotar al menú de una apariencia visual más cuidada y coherente con el entorno desarrollado. Además, para suavizar los acabados de la pantalla, a través de *unity* se le añaden prismas rectangulares en los bordes de la pantalla añadiendo y editando *prefabs* de cubos.



Figura 3-21: TV LED 30" del *Unity Asset Store* [31]

Uniendo todo lo mencionado, el resultado es un objeto cargado de una gran variedad de componentes que le dan una función esencial en el desarrollo del proyecto (Figura 3-22).



Figura 3-22: Cuerpo y jerarquía de componentes de la pantalla del menú inicial

3.4.3 Prueba del Aro

La prueba del aro se plantea como un ejercicio introductorio para que el usuario se familiarice con la interacción en realidad mixta dentro del sistema. El objetivo principal es mejorar la precisión en la manipulación de objetos virtuales usando los controladores, trabajando aspectos como la orientación del objeto, la coordinación mano-ojo y el control del movimiento en el espacio. Además, la prueba busca que el usuario aprenda a interpretar señales de retroalimentación del entorno, como cambios de iluminación, como indicaciones claras del estado de la interacción.

El flujo de la prueba se basa en tres acciones principales. En primer lugar, el usuario genera un aro en la mano mediante el *grip* del mando del equipo de *Meta Quest 3*, lo que permite iniciar la interacción de forma directa y natural. En segundo lugar, cuando el usuario pulsa el botón B, el sistema instancia la estructura objetivo en el entorno, creando el elemento con el que se debe interactuar. Finalmente, durante la manipulación, el sistema monitoriza la relación entre ambos elementos; cuando el aro entra en contacto con la estructura, se activa una señal visual en forma de luz roja, que actúa como *feedback* inmediato para indicar que se ha producido la colisión y reforzar la percepción de interacción. Los componentes principales de la prueba son los siguientes:

1.- Escena base

La prueba del aro se desarrolla en una escena de composición sencilla, pensada para que el usuario pueda centrarse en el objetivo principal de la actividad sin distracciones innecesarias. Por ello, la base de la escena incluye únicamente los elementos esenciales para el desarrollo de la prueba.

En primer lugar, el componente con el código de generación de la estructura principal al pulsar el botón B del mando derecho, es el mismo que el empleado para generar la pantalla del Menú Inicial (*SpawnEstructuraOnB_NoFloor*).

Por otro lado, se incluye también el componente con el código de generación del aro en la misma mano al pulsar el *grip* del mando derecho (*RightHandSpawner*) de la Figura 3-23. Al igual que el anterior, a este código se le configura el *prefab* a instanciar, una posición de referencia y la rotación y distancia al mando en la generación. Por otro lado, este código permite que cuando se suelta el aro, pueda volver a ser agarrado tantas veces como haga falta.

El objetivo de esta configuración base es garantizar estabilidad: que el seguimiento del mando sea consistente, que los objetos aparezcan siempre en una posición predecible y que el usuario reciba una respuesta inmediata cuando realiza la acción correcta.

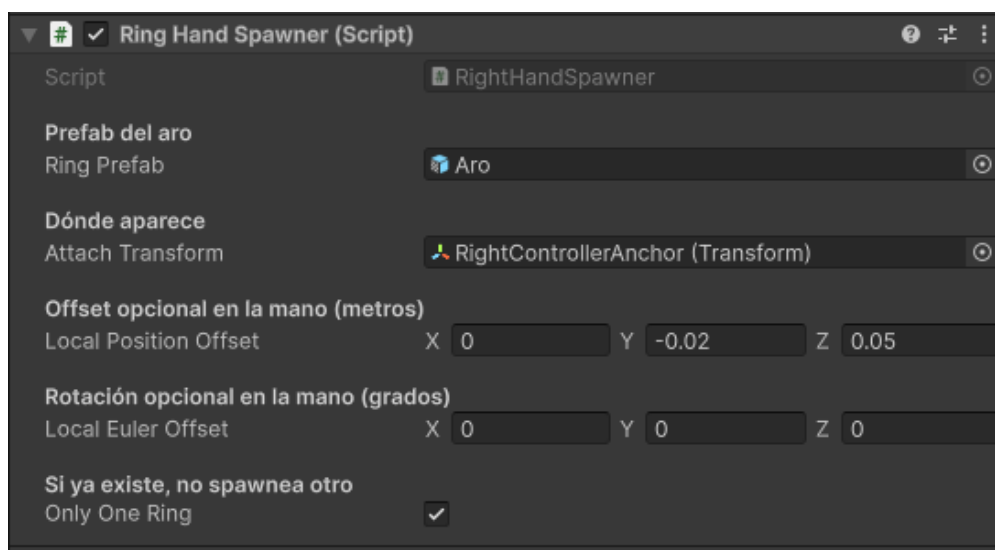


Figura 3-23: Inspector del código *RightHandSpawner*

2.- Aro

El aro se genera directamente en la mano del usuario mediante el *grip* del mando derecho, de modo que la interacción empieza con el objeto ya listo para usar, sin necesidad de que el usuario tenga que buscarlo en el entorno.

En *Unity*, el *prefab* del aro incorpora los siguientes componentes que definen su función:

- Rigidbody en modo Kinematic para que el aro se ancle a la mano por interacción de agarre del mando (Figura 3-24).
- Código detector de las colisiones entre el aro y la estructura de barras (*RingTouchDetectorOnRing*) (Figura 3-24). La detección se hace visible activándose una luz roja en la zona de la colisión.

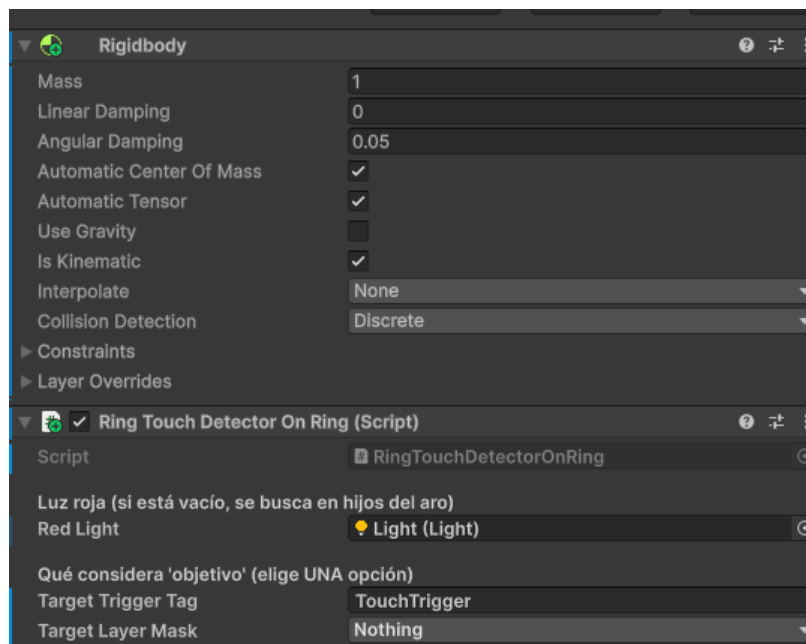


Figura 3-24: Inspector de componentes del aro

En este caso, en lugar de crear un *collider* con una forma compleja y susceptible a dar problemas con la interacción, se ha optado por crear un anillo de *colliders* en forma de capsulas que vienen ya prefabricados en los *assets* de *Unity*. Esto ha sido posible por la elaboración del código *RingCollidersGenerator* que permite ajustar la disposición de los *colliders* y sus tamaños a través del inspector de componentes (Figura 3-25).

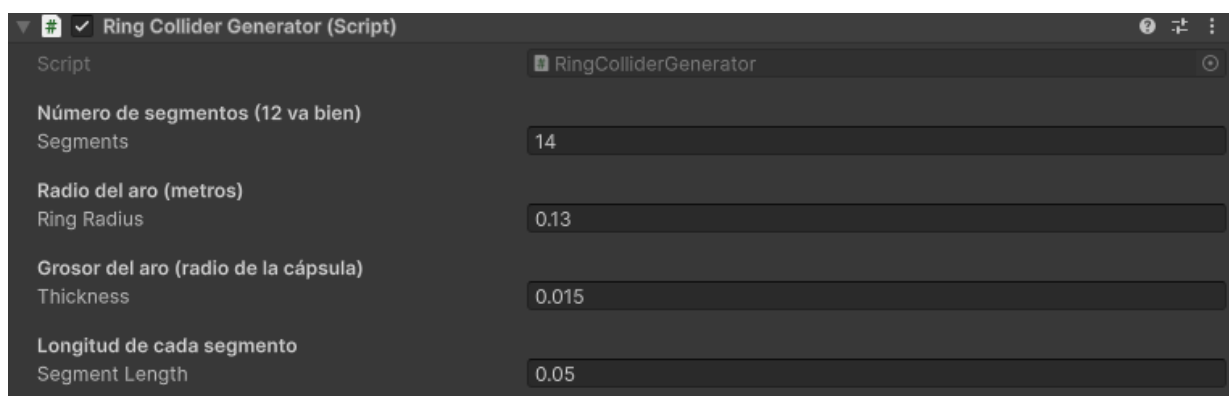


Figura 3-25: Inspector del código *RingCollidersGenerator*

La clave del aro es que su movimiento sea natural y su colisión fiable: si la detección falla o es inconsistente, el usuario no podrá entender qué está haciendo bien o mal y la prueba pierde valor didáctico.

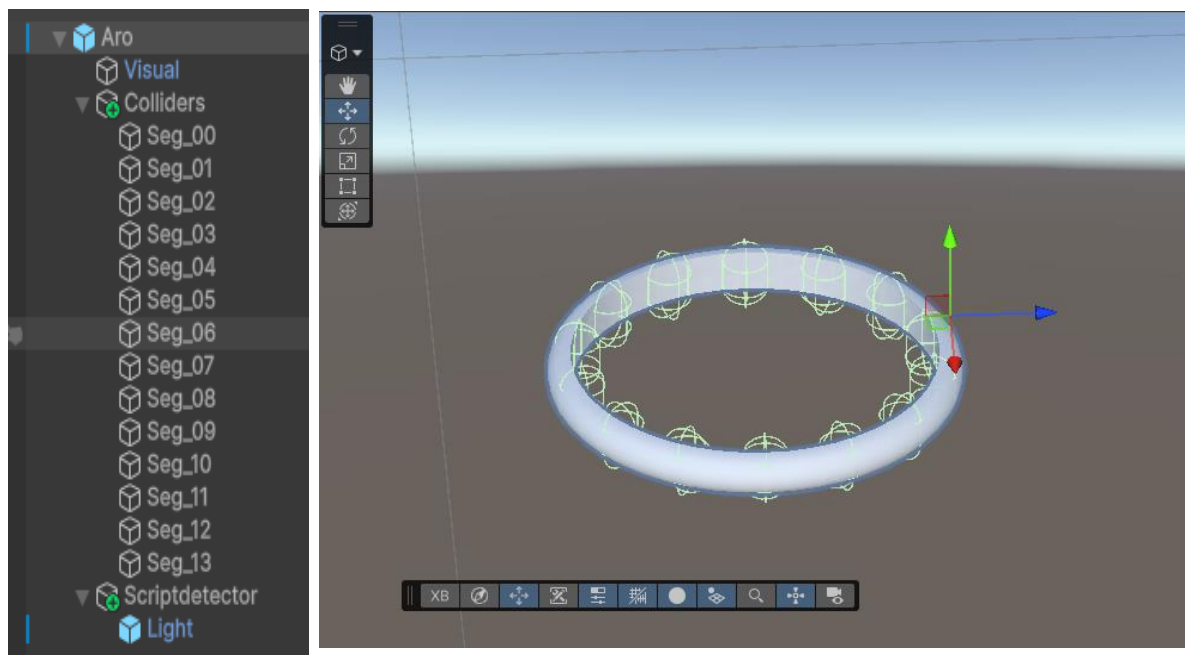


Figura 3-26: Cuerpo del aro y su jerarquía de componentes

Un aspecto importante de esta prueba es que el *feedback* visual no se encuentra en la estructura, sino dentro del propio *prefab* del aro. En concreto, la señal se implementa como una luz roja que se activa cuando el aro entra en contacto con la estructura. Al estar el indicador anclado a la mano del usuario, el *feedback* es inmediato y siempre visible, el usuario no tiene que mirar a un punto fijo del escenario, sino que recibe la confirmación justo donde está interactuando.

Este planteamiento tiene varias ventajas a nivel de desarrollo y de experiencia de usuario:

- Visibilidad garantizada: el indicador está en la mano, que es donde el usuario está prestando atención.
- Modularidad: la lógica de *feedback* queda asociada al objeto que el usuario controla, lo que facilita reutilizarla si en el futuro se cambia la estructura por otra diferente.
- Refuerzo del aprendizaje: el usuario aprende rápidamente qué acciones provocan la respuesta del sistema, ya que la señal aparece exactamente cuando su mano logra el objetivo.

3.- Estructura

La estructura utilizada en esta prueba fue el primer elemento interactivo construido a mano en todo el desarrollo. En lugar de modelarla externamente, se diseñó directamente en *Unity* mediante la combinación de objetos primitivos, principalmente cilindros para representar las barras y esferas para rematar uniones o extremos. Esta elección se tomó como punto de partida para el proyecto porque permitía avanzar rápido, probar colisiones e interacciones desde el primer momento y ajustar proporciones sin depender de una fase previa de modelado.

La estructura, como se observa en la Figura 3-29, se compone de varias barras (cilindros) colocadas de manera que construyen un entramado lineal con distintas formas que definen la dificultad de la prueba. La ventaja de este enfoque es que cada elemento es independiente y permite controlar bien qué parte del objeto interactúa con el aro. Para facilitar la detección del contacto, cada barra incorpora su propio *collider*, ajustado a su geometría. Esto hace que la detección sea más estable que utilizar un único

collider grande que englobase toda la estructura, ya que se detecta el impacto justo en la barra que se toca, se reducen falsos positivos por colisiones vacías, y resulta más sencillo depurar si alguna zona no responde como debería.

A nivel técnico, se han implementado los siguientes componentes:

- Sistema de colisión distribuido por barras (*Box Colliders* por cada cilindro): cada barra incorpora su propio *collider* ajustado a su tamaño y orientación, lo que permite detectar el contacto con el aro de forma más precisa y estable que usando un único *collider* global para toda la estructura (Figura 3-27).
- Sin comportamiento físico activo (sin *Rigidbody*): la estructura actúa como objetivo fijo una vez instanciada, por lo que no se busca que reaccione a fuerzas ni que se desplace al interactuar con el aro.

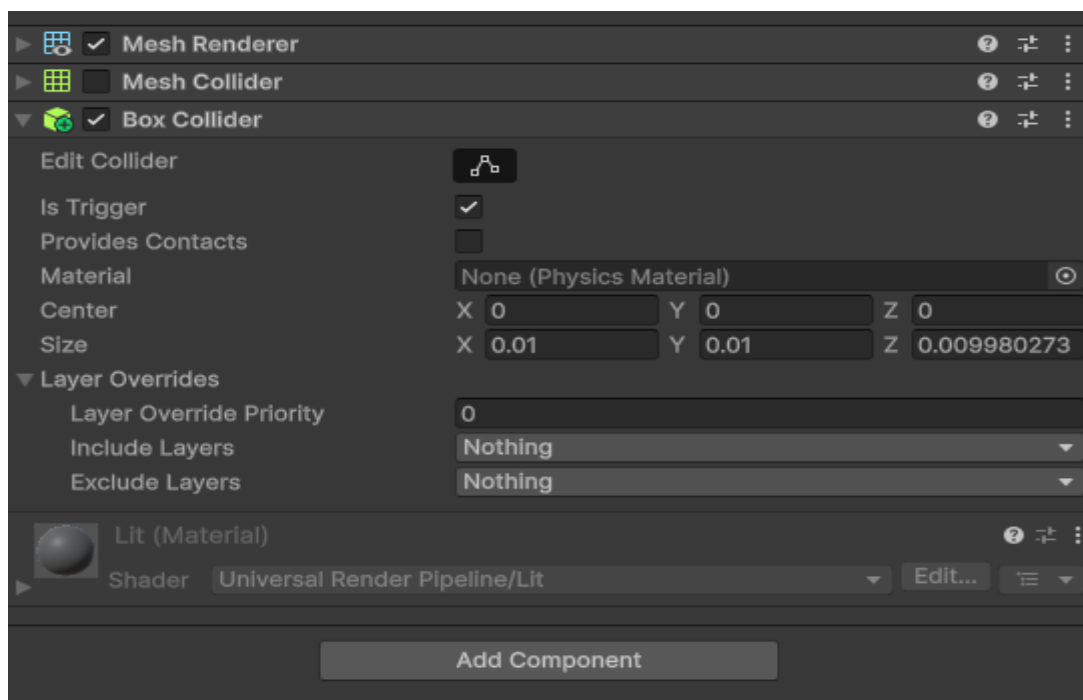


Figura 3-27: Inspector de componentes de las barras de la estructura

Por último, a través del código *StructureColisiónCounter*, se añade un contador de colisiones para que el usuario observe su progreso a medida que va avanzando. En el inspector del código de la Figura 3-28 se define un texto (*TextMeshPro*) que se sitúa previamente sobre la esquina de la estructura de barras. El código se encarga de sumar las colisiones que se produzcan, generando el texto “Colisiones: XX”.

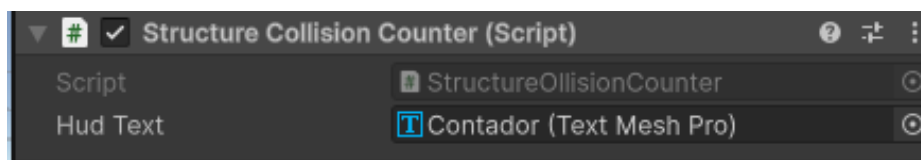


Figura 3-28: Inspector del código *StructureCollisionCounter*

En resumen, la estructura no se entiende como un único bloque, sino como una suma de piezas simples con colisión individual, lo que encaja muy bien con la idea de una primera prueba y un primer prototipo dentro del proyecto.

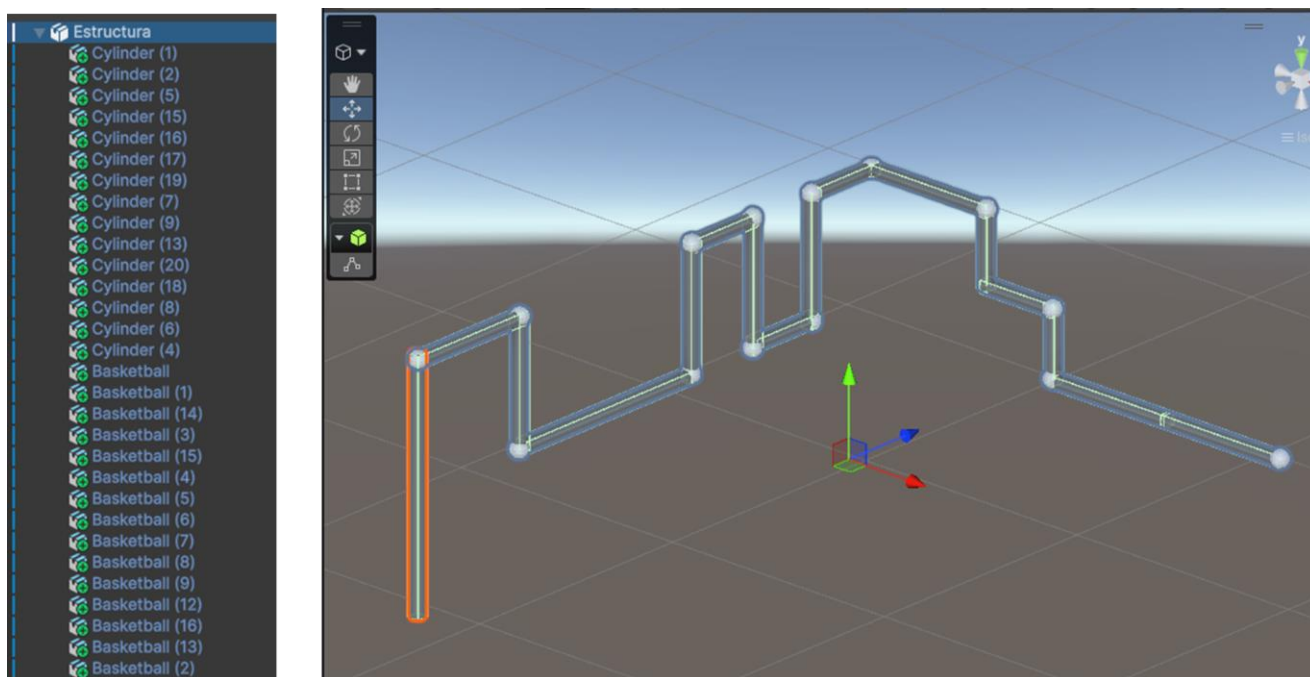


Figura 3-29: Estructura de barras de la primera prueba y sus componentes

3.4.4 Prueba de Baloncesto

La prueba de baloncesto se plantea como un ejercicio orientado a entrenar la coordinación y la precisión en acciones dinámicas dentro del entorno de realidad mixta. El objetivo principal es que el usuario aprenda a generar, manipular y lanzar un objeto virtual (la pelota) de forma controlada, ajustando fuerza, dirección y timing del movimiento. Además, esta prueba refuerza la comprensión de eventos de colisión y validación (cuándo se considera una canasta) y el uso de *feedback* inmediato para que el usuario entienda si la acción realizada ha sido correcta.

El flujo de la prueba se basa en la interacción con una pelota virtual y un aro de baloncesto colocado en el entorno. En primer lugar, el sistema permite instanciar la pelota mediante una acción directa del usuario (Con un *input* del mando), de modo que la pelota aparece lista para ser agarrada o impulsada. A continuación, el usuario realiza el lanzamiento hacia la canasta, y el sistema evalúa el resultado mediante una zona de detección asociada al aro (un detector con contador ubicado justo bajo el aro).

Cuando la pelota atraviesa correctamente dicha zona, se registra la canasta y se muestra una respuesta clara al usuario mediante *feedback* visual (texto flotante con actualización de puntuación). De este modo, el usuario recibe una confirmación inmediata y puede repetir el ciclo de lanzar-validar para mejorar progresivamente la precisión.

1.- Escena base

La escena de baloncesto reutiliza la misma base técnica que el resto de las escenas del proyecto, compuesta por una serie de elementos comunes que garantizan el seguimiento, la estabilidad física y la coherencia en la interacción.

Se sitúa en la escena una superficie de referencia de manera que quede a la altura del suelo de la ubicación real (Figura 3-30), permitiendo que se simule que los componentes de la prueba interaccionan con el suelo real. Concretamente, se usa como soporte físico para que la pelota pueda colisionar y botar de forma controlada, evitando comportamientos poco realistas al caer o rodar, además la canasta también está configurada para que se genere en esta superficie.

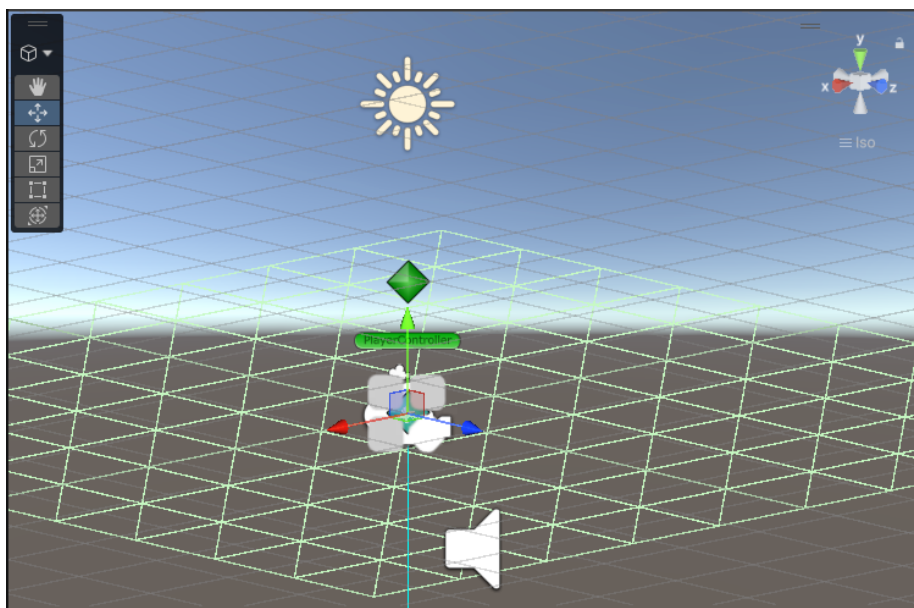


Figura 3-30: *SceneView* de la superficie de suelo

Para instanciar la canasta, se emplea un componente con código generador de la canasta en el suelo al pulsar el botón B (*SpawnEnSuelo*). Este código, a diferencia del *SpawnEstructuraOnB_NoFloor* de escenas anteriores, tiene la particularidad que genera la canasta sobre la superficie de suelo ya mencionada, para comprender las diferencias en el flujo de ambos códigos se recomienda consultar el Anexo III: Diagramas de flujo de los códigos del proyecto.

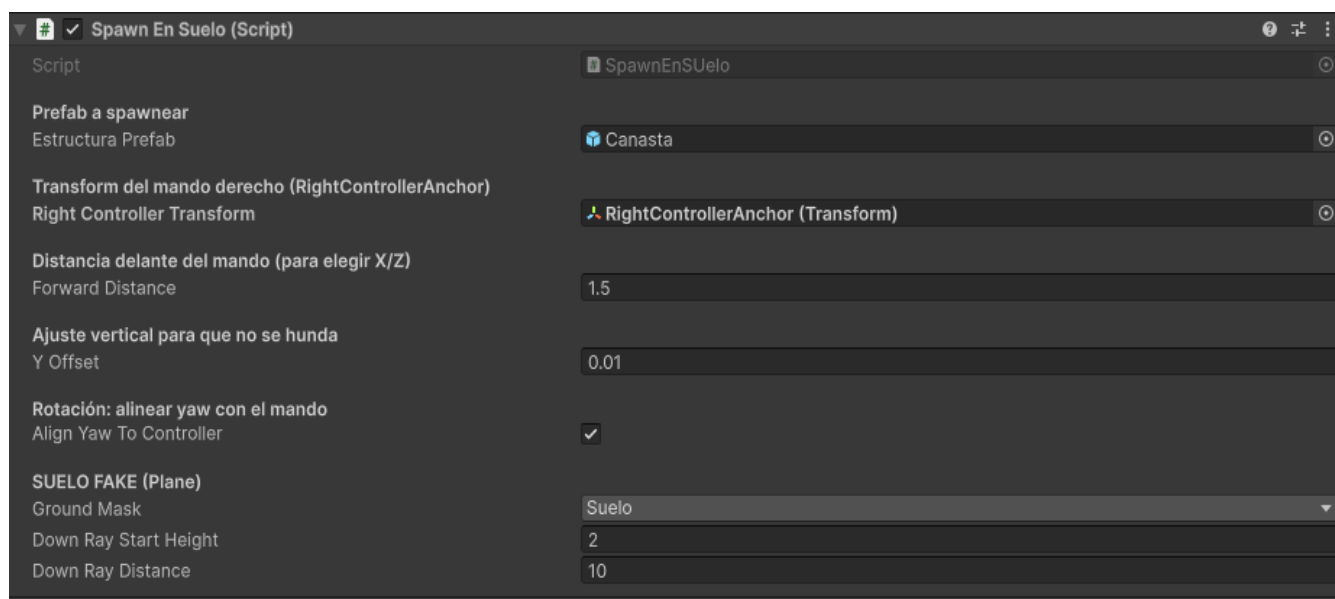
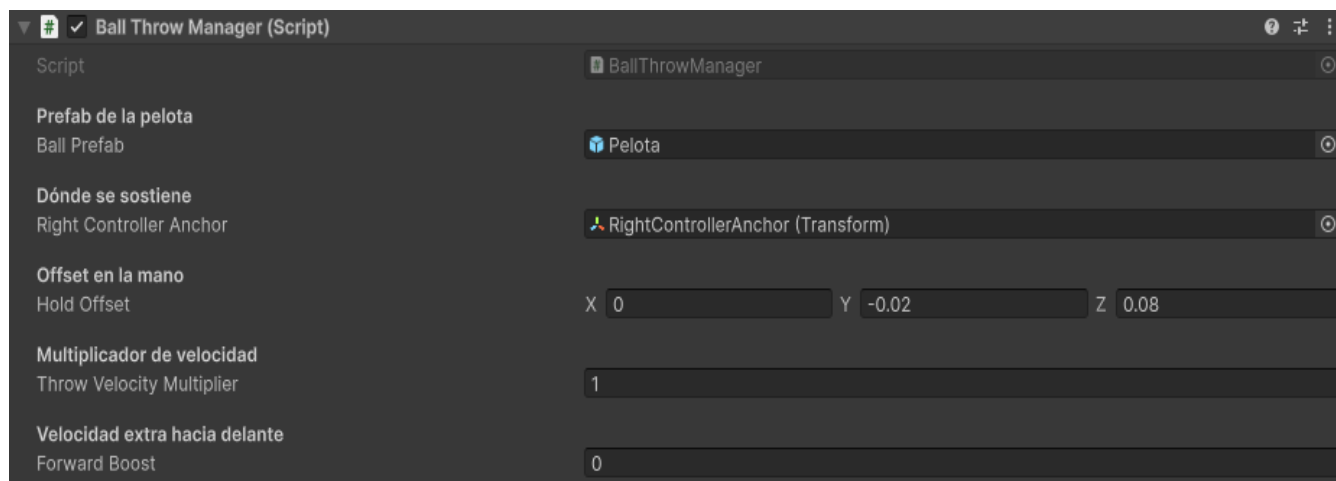


Figura 3-31: Inspector del código *SpawnEnSuelo*

Por otro lado, para instanciar la pelota, se incluye otro componente con código de generación de la pelota en la mano del usuario pulsando el *grip* del mando derecho (*BallThrownManager*). Este componente se diferencia del generador aro de la prueba anterior, debido a que lleva incluido la gestión de la física de la pelota, permitiendo de esta manera que pueda ser lanzada ajustando los parámetros de la cinemática del lanzamiento en el inspector de componentes (Figura 3-32), esta gestión de la física también se encuentra explicada en el Anexo III: Diagramas de flujo de los códigos del proyecto

Figura 3-32: Inspector del código *BallThrownManager*

2.- Pelota

La pelota es el elemento central de la prueba, ya que concentra la parte práctica de generación, manipulación y lanzamiento. Para asegurar un acabado visual realista y evitar dedicar tiempo al modelado, se utiliza un objeto importado desde *Unity Asset Store* (del paquete *Basketball Realistic* [32], Figura 3-33), que proporciona la malla y los materiales listos para su uso. En este paquete viene incluido un objeto prefabricado de una pelota de baloncesto además con materiales y mallas que dan varias opciones para el color la misma.

Figura 3-33: Paquete *BasketballRealistic* en *Unity Asset Store* [32]

Tras su importación, la pelota se convierte en un *prefab* funcional dentro del proyecto, configurado para que su interacción sea consistente en todas las repeticiones de la prueba. Esta configuración se muestra en la Figura 3-34, y se centra en definir correctamente sus componentes físicos (colisión y dinámica) y en integrarlo con el sistema de generación y lanzamiento controlado por el usuario, e incluye los siguientes componentes:

- Sistema de colisión (*Sphere Collider*) ajustado al radio del balón, utilizando un *collider* simple para garantizar impactos estables con el suelo, el tablero y el aro.
- *Rigidbody*, imprescindible para gravedad, bote y respuesta a fuerzas durante el lanzamiento (masa, velocidad lineal y *angular drag* ajustables según el comportamiento buscado).

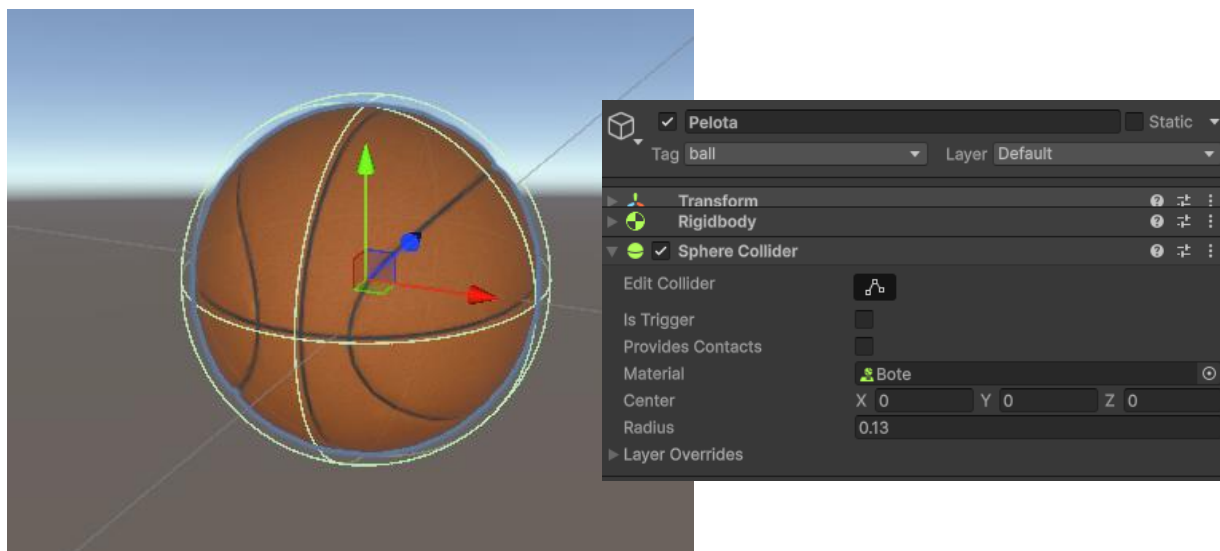


Figura 3-34: Cuerpo de la pelota y su inspector de componentes

3.- Canasta

La canasta se utiliza como referencia de la prueba, ya que marca el punto al que el usuario debe dirigir el lanzamiento y donde se valida el acierto. Para su implementación se emplea un modelo importado desde *Unity Asset Store* (Paquete *BasketballRing* [33], Figura 3-35), que incluye el conjunto completo (tablero, aro, soporte y red) ya preparado a nivel visual.



Figura 3-35: *BasketballRing* en *Unity Asset Store* [33]

A nivel de Unity, la canasta se integra como un *prefab* jerárquico, es decir, un objeto principal con varios *GameObjects* hijos (tablero, aro, red, etc.). Todos estos elementos comparten los componentes visuales básicos propios de cualquier modelo 3D, además, se le añaden los componentes relevantes para la interacción y la validación de la canasta, como se muestra en la Figura 3-36:

Por un lado, *colliders* de interacción física que configuran colisiones en las partes que deben afectar al comportamiento de la pelota. Por una parte, un *Box Collider* en el tablero, para permitir rebotes. Y por otro lado, el mismo código que se empleó en la Prueba del Aro para colocar los *colliders* en el mismo

(*RingColliderGenerator*), se ha empleado en este caso para hacer que la pelota pueda rebotar en el anillo de la canasta y permitiendo que también la pelota se pueda “encestar”.

Además de las colisiones físicas, se añade una zona específica para validar el enceste, situada en el centro del aro, como se muestra en la Figura 3-36. Esta trata de un *Box Collider* con *trigger* activado, situado en el interior de la canasta en una zona por la que la pelota pasa cuando es encestando. Para contar las anotaciones del jugador, se ha elaborado el código *HoopScoreZone* para generar bajo el anillo un texto *TextMeshPro* que va actualizando las anotaciones del jugador a medida que van sucediendo, para asegurar que se trata de una anotación, el código solo cuenta las colisiones que se producen con una velocidad de la pelota negativa en el eje Y, asegurándose así de que la pelota acaba de pasar por el anillo.

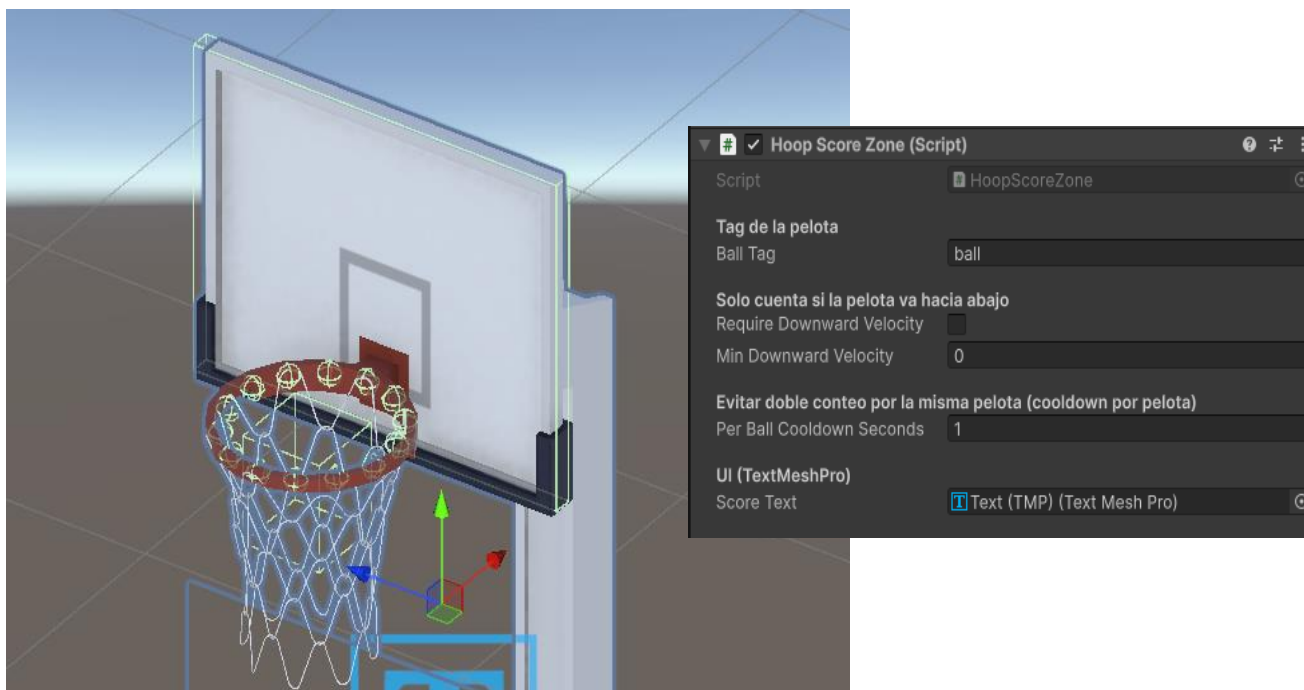


Figura 3-36: Cuerpo de la canasta con *colliders* y el inspector del código *HoopScoreZone*

3.4.5 Puzle 3D

La escena del puzle 3D se diseña como una prueba centrada en la manipulación espacial avanzada de objetos en realidad mixta. Su objetivo principal es que el usuario desarrolle habilidades de orientación y colocación precisa de piezas tridimensionales, trabajando la percepción de profundidad, la alineación de formas y la coordinación manual. Además, en esta prueba se incorpora retroalimentación háptica mediante guantes *bHaptics*, con la intención de reforzar la sensación de contacto y mejorar la comprensión de la interacción durante el ensamblaje. De este modo, se busca que el usuario no solo vea la colocación, sino que también perciba el proceso mediante estímulos táctiles.

El funcionamiento se basa en un conjunto de piezas 3D que forman un objeto completo cuando se colocan en su posición correcta. El usuario puede agarrar, mover y rotar cada pieza dentro del espacio físico hasta encontrar la orientación adecuada. En paralelo, los guantes *bHaptics* aportan vibración como señal de apoyo durante la interacción, por ejemplo, para indicar que una pieza está siendo manipulada o que ha alcanzado su posición final.

La escena incluye una referencia en forma de cubo central que ayuda a identificar dónde debe colocarse cada elemento. A medida que el usuario acerca una pieza a su posición final, el sistema aplica una lógica de validación para comprobar si la pieza está correctamente alineada y situada; cuando se cumple la condición, la pieza queda colocada de forma estable, facilitando el progreso del ensamblaje. Así, el usuario repite un ciclo de selección-orientación-colocación, hasta completar el puzle y formar la figura final.

1.- Escena base

En primer lugar, se emplea el código *SpawnEstructuraOnB_NoFloor*, encargado de generar la estructura principal de la prueba dentro del entorno cuando el usuario pulsa el botón B. Este código también se emplea en otras pruebas del proyecto.

Por otro lado, el código *RightHandPieceSpawner_Free* se encarga de generar de forma secuencial las distintas piezas del puzle delante de la mano derecha del usuario, tomando como referencia el objeto *RightControllerAnchor* del *OVRCameraRig* y activándose mediante la pulsación del *grip* derecho del mando. De este modo, las piezas no aparecen todas desde el inicio, sino que se introducen progresivamente en la escena, lo que permite una interacción más ordenada durante el ensamblaje.

Además, este código no solo instancia los *prefabs*, sino que también prepara automáticamente su configuración básica, asignando a cada pieza un identificador propio, comprobando la presencia del componente *SnapPiece*, necesario para la lógica de encaje, y añadiendo un *Rigidbody* para que pueda manipularse físicamente dentro del entorno. Asimismo, la gravedad se desactiva en el momento de la generación para evitar que las piezas caigan o se desplacen de forma no deseada, facilitando así su uso en la experiencia de realidad mixta. En el inspector de componentes (Figura 3-37) se definen los *prefabs* de los cubos a instanciar, el tag identificador de cada uno y la posición de aparición con respecto al mando.

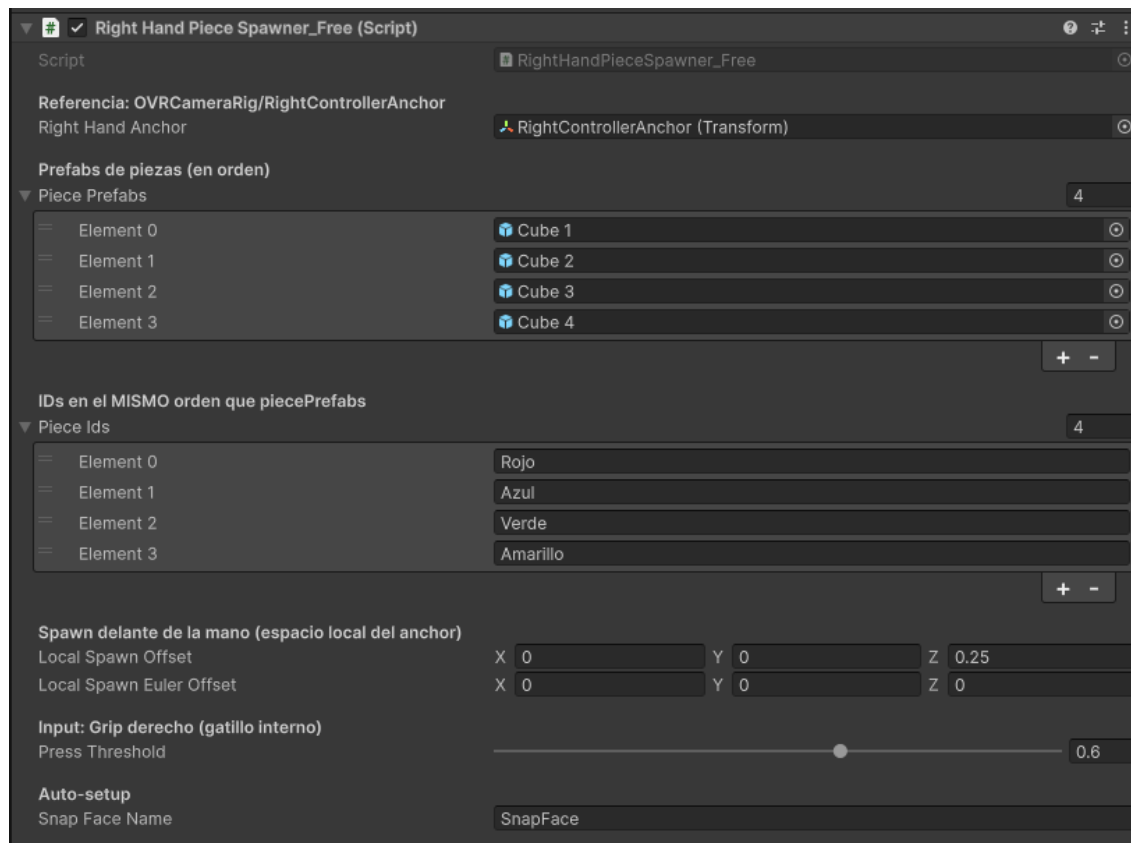


Figura 3-37: Inspector del código *RightHandSpawner_Free*

En esta escena se incorporan además los componentes *bHaptics* y *bHaptics Glove*, ambos procedentes del SDK oficial de *bHaptics* para Unity, ya que son los elementos que permiten integrar el funcionamiento de los guantes hápticos dentro del proyecto. El primero de ellos se encarga de inicializar el sistema háptico en la escena, haciendo posible la comunicación con el entorno de *bHaptics* y la reproducción de eventos de vibración, mientras que el segundo está orientado específicamente al uso de los *TactGlove* (Guantes hápticos), permitiendo asociar respuestas táctiles a la interacción física de las manos virtuales con los objetos y ajustando los parámetros básicos de las vibraciones como se muestra

en la Figura 3-38. Su inclusión resulta necesaria en esta prueba porque el puzzle 3D no busca únicamente que el usuario coloque correctamente las piezas en el espacio, sino también que perciba mediante el tacto acciones como el contacto, la manipulación y el encaje de cada una de ellas, reforzando así la sensación de inmersión durante la experiencia.

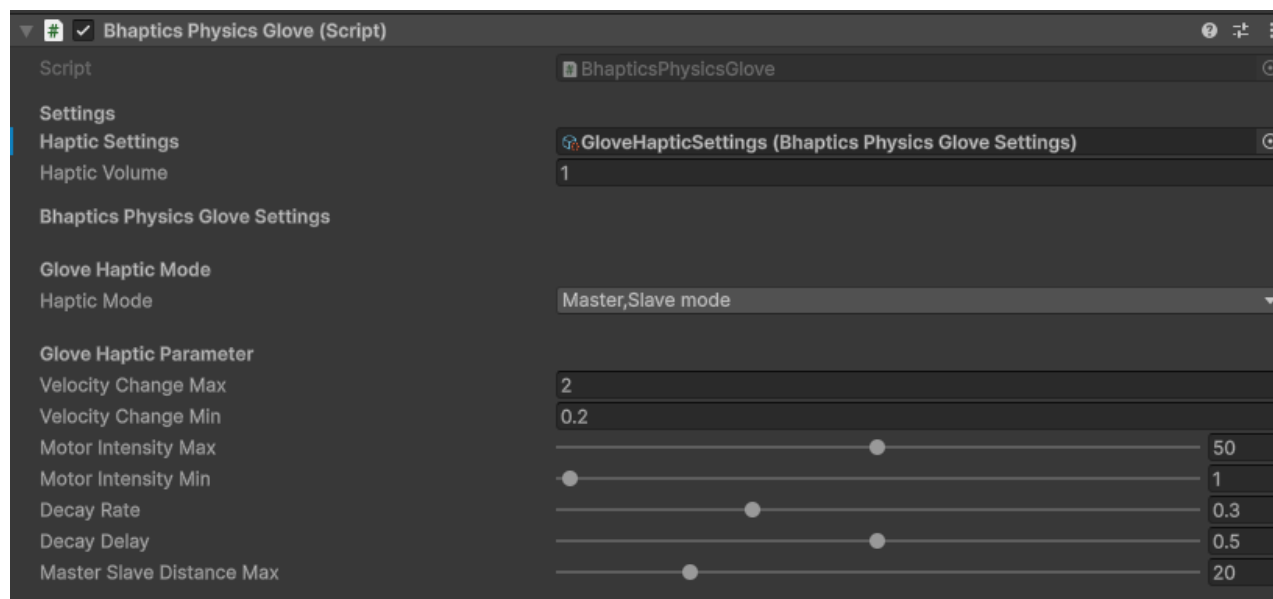


Figura 3-38: Inspector del componente *Bhaptics Gloves*

2.- Cubo central

El cubo central constituye la estructura principal de referencia del puzzle 3D. Su función es servir de base para el ensamblaje de las distintas piezas, ayudando al usuario a identificar de forma visual dónde debe colocarse cada una de ellas. Para ello, se diseña como un cubo con cuatro caras de diferente color, asociadas a las piezas que completan la figura final.

Los *colliders* situados en cada una de las caras permiten detectar cuándo una pieza entra en la zona de encaje correspondiente. Al estar configurados como *trigger*, no bloquean físicamente el movimiento, sino que actúan como áreas de detección que permiten iniciar la comprobación del encaje.

El código *SnapSocket* es el componente encargado de controlar la lógica principal de colocación de las piezas sobre el cubo central. Su función consiste en comprobar, en primer lugar, que la pieza que entra en contacto con una cara sea la correcta mediante la comparación de identificadores; es decir, que cada pieza solo pueda encajarse en la cara que le corresponde.

Una vez validado esto, el código analiza si la pieza se encuentra a una distancia suficientemente próxima al punto de encaje y si presenta la orientación adecuada con respecto a la cara del cubo. Solo cuando se cumplen todas estas condiciones, el sistema da por válida la colocación y fija automáticamente la pieza en su posición final. De este modo, el *SnapSocket* evita colocaciones incorrectas y asegura que el montaje del puzzle se realice de manera controlada, precisa y coherente con la estructura final deseada.

En el inspector, mostrado en la Figura 3-39, se definen la posición de referencia y el vector de orientación, que actúan conjuntamente como los elementos que definen el encaje correcto de cada pieza en el cubo central. Por un lado, la posición marca el punto exacto en el que la pieza debe situarse para quedar colocada en su lugar correspondiente. Por otro, el vector establece la dirección con la que esa pieza debe alinearse al aproximarse a la cara del cubo. Gracias a la combinación de ambos elementos, el sistema puede comprobar no solo que la pieza esté suficientemente cerca de su destino, sino también que lo haga con la orientación adecuada, garantizando así un ensamblaje preciso y coherente.

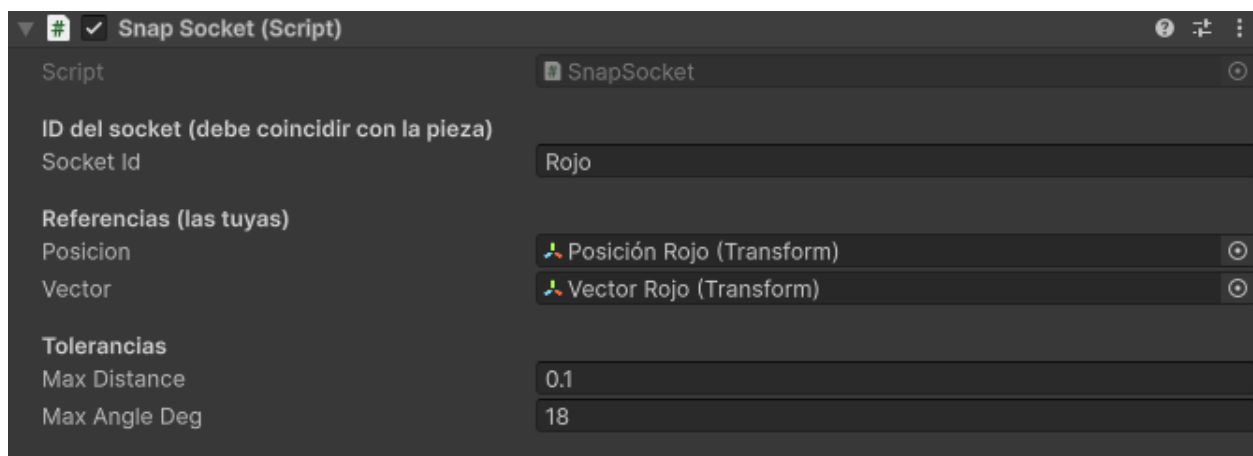


Figura 3-39: Inspector del código SnapSocket

En conjunto, el cubo central concentra la referencia visual y la lógica principal de ensamblaje del puzle, ya que no solo orienta al usuario mediante la diferenciación por colores de sus caras, sino que también integra los elementos necesarios para validar la colocación de cada pieza (Figura 3-40). Gracias a esta combinación de guía visual, detección y comprobación de posición y orientación, este componente se convierte en la base que hace posible un montaje preciso, ordenado y coherente dentro de la escena.

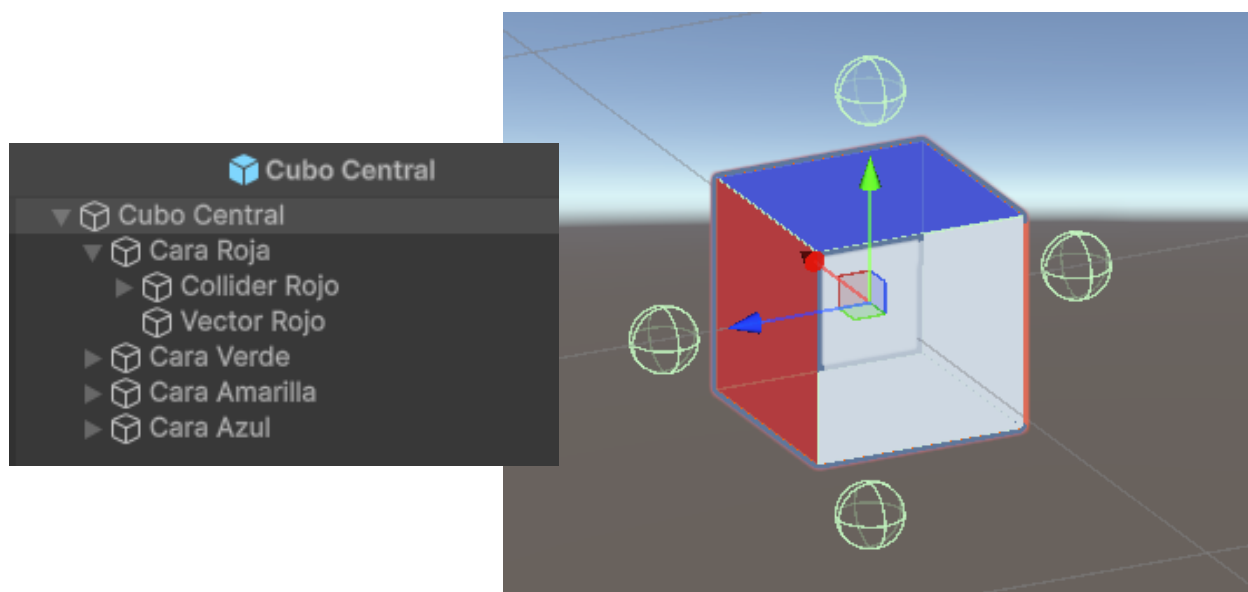


Figura 3-40: Cuerpo y jerarquía de componentes del cubo central

3. Cubos (piezas)

Los cubos-pieza son los elementos que el usuario debe manipular para completar el puzle 3D. Cada uno de ellos representa una parte independiente de la figura final y está asociado a uno de los colores del cubo central, de manera que el usuario debe cogerlo, orientarlo correctamente y aproximarlos a la cara que le corresponde para que pueda encajarse. Su función dentro de la prueba es materializar la interacción principal de la escena, ya que son los objetos sobre los que recae el ejercicio de manipulación, orientación espacial y colocación precisa.

El componente `[BuildingBlock] HandGrabInstallationRoutine` permite que cada pieza pueda ser agarrada e interactuada con la mano dentro del entorno de realidad mixta. Gracias a este elemento, el usuario puede coger el cubo, moverlo libremente por el espacio y colocarlo en la posición deseada, haciendo posible la interacción directa que requiere la prueba.

El código *SnapPiece* es el componente encargado de gestionar el comportamiento de cada cubo-pieza dentro de la lógica de ensamblaje del puzzle. En primer lugar, asigna a cada pieza un identificador propio, que después será comparado con el del *SnapSocket* del cubo central correspondiente para comprobar si el encaje se realiza en la cara adecuada. Además, incorpora una referencia denominada *snapFace*, que actúa como punto de comparación espacial durante la validación de la colocación. Cuando el sistema determina que la pieza se encuentra en la posición y orientación correctas, este código se encarga de fijarla automáticamente en su destino final, marcarla como ya colocada y detener su comportamiento físico para evitar movimientos no deseados.

Junto a ello, también llama al evento háptico asociado al encaje correcto mediante los guantes de *bHaptics*, de forma que el usuario recibe la vibración definida en el inspector, como se muestra en la Figura 3-41) al completar la acción. Por último, el código puede desactivar los componentes de agarre y, si se desea, también los *colliders*, con el objetivo de impedir que una pieza ya colocada vuelva a manipularse, asegurando así la estabilidad y coherencia del montaje. Para comprender el flujo de la validación de identificación y posición del puzzle, se recomienda consultar el Anexo III: Diagramas de flujo de los códigos del proyecto tanto para este código como para el anterior.

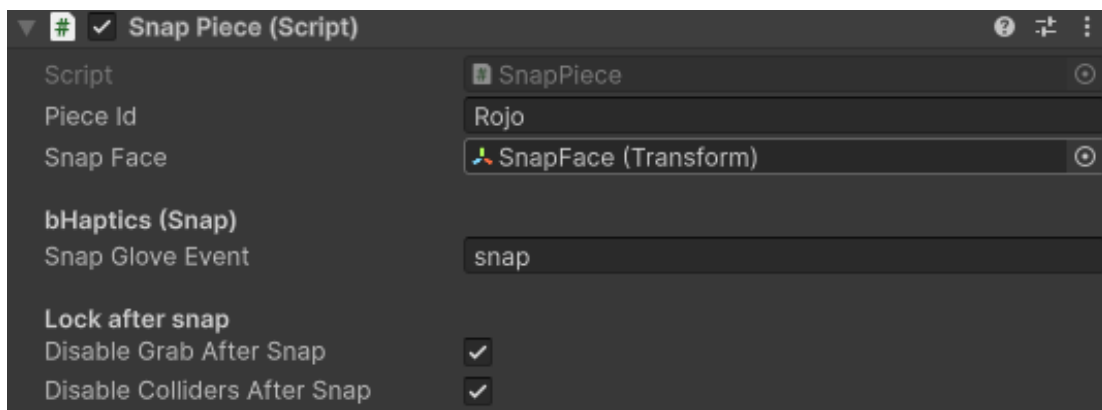


Figura 3-41: Inspector del código *SnapPiece*

El código *BhapticsEventListener* es el componente encargado de reproducir los eventos hápticos asociados a la interacción básica con cada cubo-pieza. En concreto, permite lanzar una vibración diferente cuando el usuario agarra la pieza y cuando la suelta, utilizando para ello los eventos previamente definidos en el *bHaptics developer*. De este modo, acciones sencillas como coger o liberar un cubo no se perciben únicamente a nivel visual, sino que también van acompañadas de una respuesta táctil en los guantes, reforzando la sensación de contacto y mejorando la inmersión durante la manipulación de las piezas.

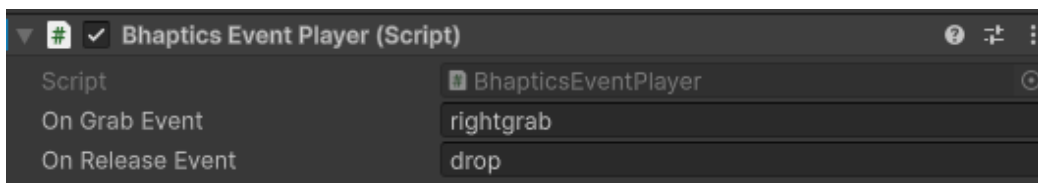


Figura 3-42: Inspector del *Bhaptics Event Player*

El componente *Interactable Unity Event Wrapper*, del *Meta XR All-in-One SDK*, sirve como enlace entre la interacción del usuario con la pieza y la reproducción de los eventos hápticos. Dentro de su inspector (Figura 3-43), se encuentra una lista de acciones posibles, las relevantes para esta prueba son las que definen cuando el usuario selecciona o suelta un cubo, en ellas se introduce el código *BhapticsEventListener* de tal manera que en esas acciones, se reproduce la vibración en los guantes definidas en el código.

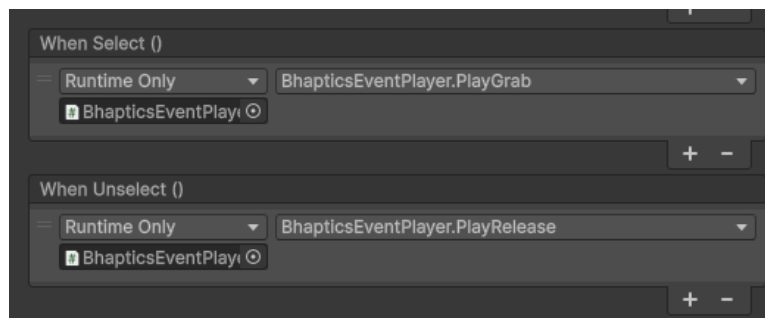


Figura 3-43: Inspector del *Unity Event Wrapper*

Como se muestra en la Figura 3-44, empleando el *Bhaptics Designer*, se han configurado tres eventos hápticos distintos para esta escena en los guantes. Cuando se agarra el objeto (*grab*) se produce una vibración en las puntas de los dedos, cuando se suelta el objeto (*drop*), vibra la muñeca, y por último cuando se produce el *snap*, se produce una vibración en todo el guante a la vez (*Snap*).

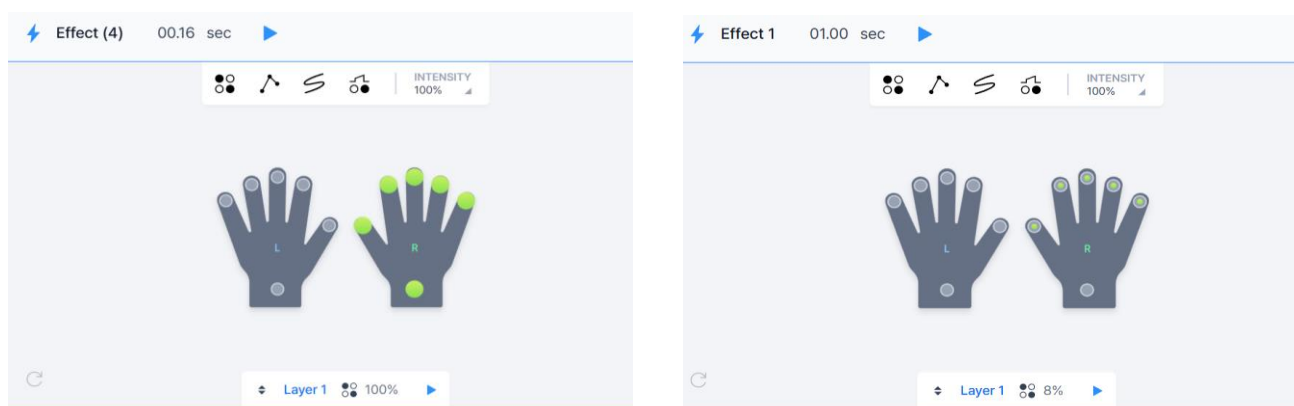


Figura 3-44: Configuración de los *patterns snap* (izquierda) y *grab* (derecha)

En conjunto, los cubos-pieza constituyen el elemento activo principal de la prueba, ya que concentran la manipulación directa por parte del usuario y enlazan tanto la lógica de encaje como la retroalimentación háptica. Gracias a la combinación de componentes de interacción, validación espacial y respuesta táctil, que se muestran en la Figura 3-45, estas piezas permiten que el ensamblaje del puzzle se desarrolle de manera precisa, inmersiva y coherente con los objetivos de la escena.

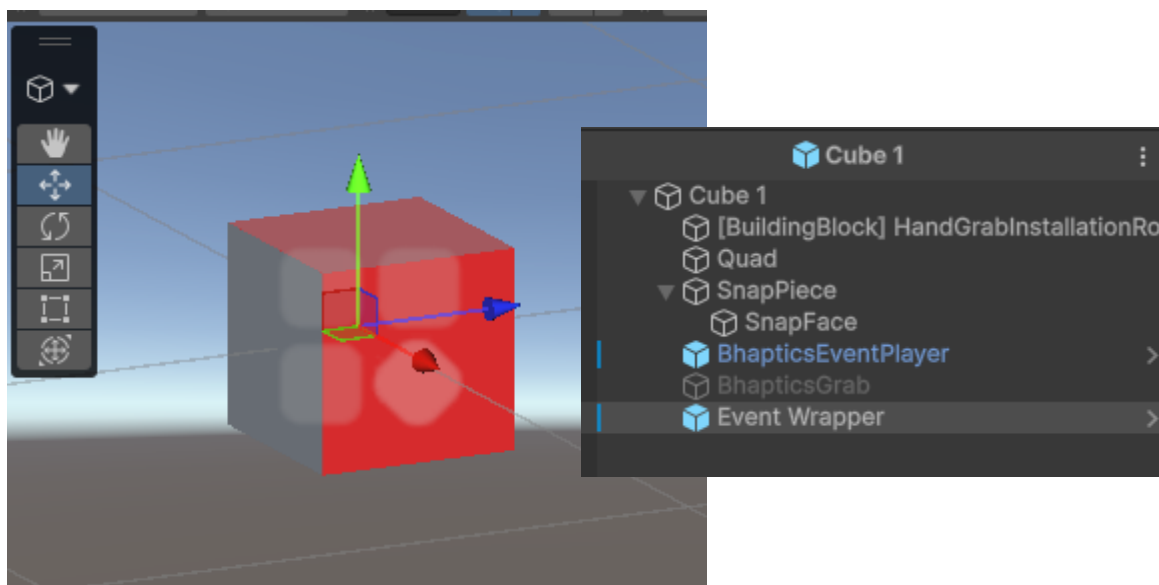


Figura 3-45: Cuerpo y jerarquía de los Cubos-Pieza

3.4.6 Prueba del lanzador

La prueba del lanzador se plantea como un ejercicio orientado a trabajar la interacción dinámica con objetos en movimiento. El objetivo principal es que el usuario aprenda a reaccionar y actuar con precisión ante estímulos generados por el sistema, entrenando la coordinación, el tiempo de respuesta y el control del espacio.

Además, esta escena está pensada para incorporar retroalimentación háptica corporal mediante el chaleco *bHaptics TactSuit X40*, con el fin de aumentar la inmersión y permitir que el usuario perciba impactos o eventos relevantes de forma más intuitiva. Esto refuerza la idea de entrenamiento sensorial: no solo ver el estímulo, sino también sentirlo.

El funcionamiento se basa en un sistema que genera y lanza pelotas desde una estructura situada en el entorno. Una vez activada la prueba, el lanzador instancia pelotas de forma controlada y las impulsa en una dirección definida, de modo que el usuario debe seguirlas y reaccionar (por ejemplo, esquivándolas o interceptándolas).

Las pelotas se comportan mediante físicas, simulando trayectorias, caídas y rebotes al colisionar con elementos del escenario. Como parte del diseño, se contempla que los eventos de impacto o contacto se acompañen de señales hápticas en el chaleco *bHaptics*, de manera que el usuario reciba una confirmación sensorial clara cuando se produzcan impactos o situaciones relevantes durante la ejecución.

1.- Escena Base

En primer lugar, se utiliza el código, *SpawnEstructuraOnB_NoFloor* (Empleado en pruebas anteriores), encargado de generar en la escena la estructura principal desde la que se lanzarán las pelotas.

Junto a ello, la escena incorpora también el componente *bHaptics*, necesario para integrar el sistema de retroalimentación háptica corporal dentro del proyecto. Su presencia resulta especialmente importante en esta prueba, ya que el objetivo no es únicamente que el usuario vea las pelotas aproximarse y reaccione ante ellas, sino también que pueda llegar a percibir físicamente los impactos mediante el chaleco háptico. La inclusión de este componente desde la base de la escena deja preparado el entorno para asociar futuras vibraciones a los contactos detectados en distintas zonas del cuerpo.

Por último, uno de los elementos más importantes de esta escena es el sistema de *colliders* corporales asociado al jugador, cuya finalidad es detectar las colisiones entre el cuerpo del usuario y las pelotas lanzadas por la estructura. En este caso, estos *colliders* no permanecen estáticos, sino que deben acompañar en todo momento la posición y orientación general del jugador para que la detección de impactos resulte coherente con su movimiento dentro del entorno.

Para ello, el objeto corporal incorpora un *Box Collider* para cada zona del torso (Dividido en cuatro partes como aparece en la Figura 3-46) y, además, se le añade el código *BodyColliderFollow*, que se encarga de actualizar continuamente su posición tomando como referencia el objeto *CenterEyeAnchor* del *OVRCameraRig* en su inspector (Figura 3-46). A partir de esa referencia, el código desplaza el *collider* verticalmente mediante un offset para situarlo aproximadamente a la altura del pecho y, además, ajusta su rotación siguiendo el giro horizontal del usuario, de forma que el volumen de colisión acompaña de manera natural la orientación del cuerpo.

Gracias a esta lógica, el sistema puede detectar con mayor fiabilidad cuándo una pelota impacta sobre la zona corporal definida, lo que resulta fundamental tanto para registrar el contacto como para poder asociar posteriormente una respuesta háptica en el chaleco.

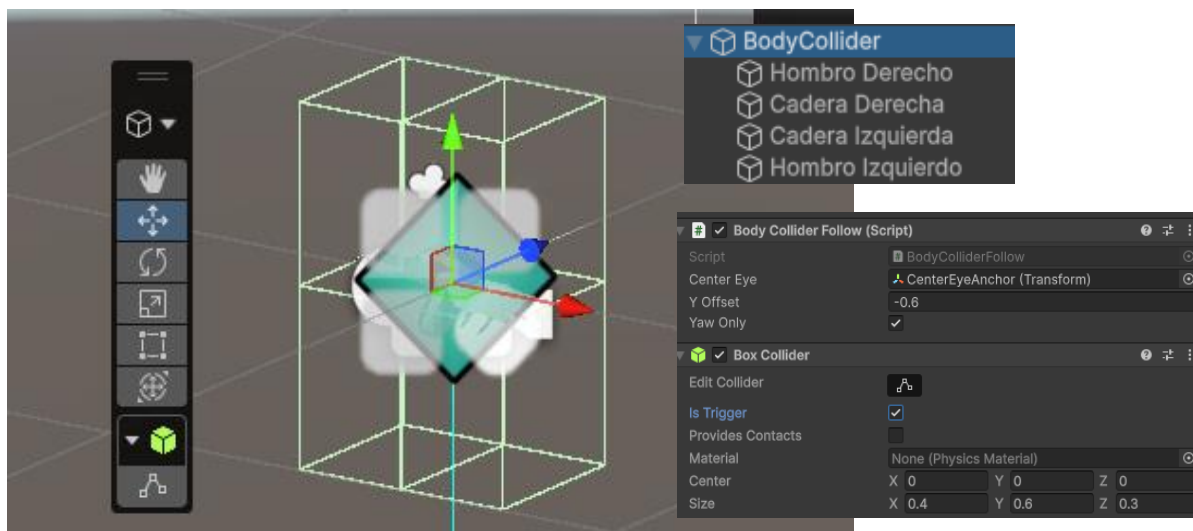


Figura 3-46: Cuerpo, inspector y jerarquía de los *colliders* del torso

2.- Lanzador

El Lanzador se constituye a partir la base física principal del mismo ya que sobre ella se sitúan el resto de los elementos que componen el dispositivo, como los cañones, las salidas, el panel informativo y los controles de activación como se muestra en la Figura 3-49. Visualmente, este elemento se corresponde con la parte gris de la estructura y se ha construido a partir de un objeto *Cube* de *Unity* redimensionado, adaptando sus proporciones para que funcione como soporte del conjunto. Además, sobre este cuerpo se añaden dos paneles de distintos colores en el que aparecen instrucciones para el usuario sobre como arrancar y parar el lanzador.

El elemento Cañones agrupa los componentes desde los que se representan visualmente los lanzamientos de la prueba, constituyendo una de las partes más características del dispositivo. En este caso, los cañones no se han construido a partir de prefabricados incluidos en *Unity*, sino que han sido modelados externamente en *Autodesk Inventor*, ya que ninguno de los objetos disponibles por defecto se ajustaba de forma satisfactoria a los requisitos geométricos y visuales de la escena.

Una vez importados al proyecto, estos modelos se integran en la estructura del lanzador para aportar una apariencia más coherente con el dispositivo diseñado. Además, en el interior de cada uno de ellos se sitúan *prefabs* de pelotas iguales a las que posteriormente se lanzan durante la prueba, con una finalidad principalmente decorativa, evitando que los cañones se perciban como elementos vacíos y reforzando visualmente la idea de que se trata de conductos preparados para disparar proyectiles.

Para situar los puntos de origen de los lanzamientos de las bolas, el elemento Salidas agrupa una serie de puntos de referencia colocados en los cañones cuya función es definir las posiciones desde las que se originan las pelotas durante la prueba. Estos objetos no están pensados como elementos visibles para el usuario, sino como referencias internas empleadas por el código encargado de gestionar los lanzamientos, ya que permiten indicar con precisión en qué lugar de la estructura debe generarse cada proyectil. Gracias a ello, el sistema puede asociar cada disparo a una salida concreta, organizando de forma clara y controlada los distintos puntos desde los que las pelotas pueden aparecer y comenzar su trayectoria dentro de la escena.

El código *BallLauncher* constituye el núcleo funcional del sistema de disparo de la escena, ya que es el encargado de gestionar la generación continua de pelotas desde la estructura del lanzador. Para ello, en el inspector, mostrado en la Figura 3-47, se toma como referencia el conjunto de salidas definido previamente y selecciona de forma aleatoria una de ellas cada vez que debe producirse un disparo, instanciando en esa posición una pelota del *prefab* correspondiente. Una vez creada, la pelota recibe una velocidad inicial en la dirección marcada por la salida elegida, lo que permite reproducir trayectorias

coherentes con la orientación de cada cañón. Además, este código controla también aspectos temporales del sistema, como el intervalo entre lanzamientos, que puede mantenerse fijo o variar aleatoriamente dentro de un rango determinado, y limita el número máximo de pelotas activas al mismo tiempo para evitar una sobrecarga innecesaria de la escena. Junto a ello, incorpora una lógica auxiliar que elimina automáticamente las pelotas transcurrido un tiempo determinado y actualiza el recuento de proyectiles activos, contribuyendo así a mantener un funcionamiento estable y ordenado durante la prueba.

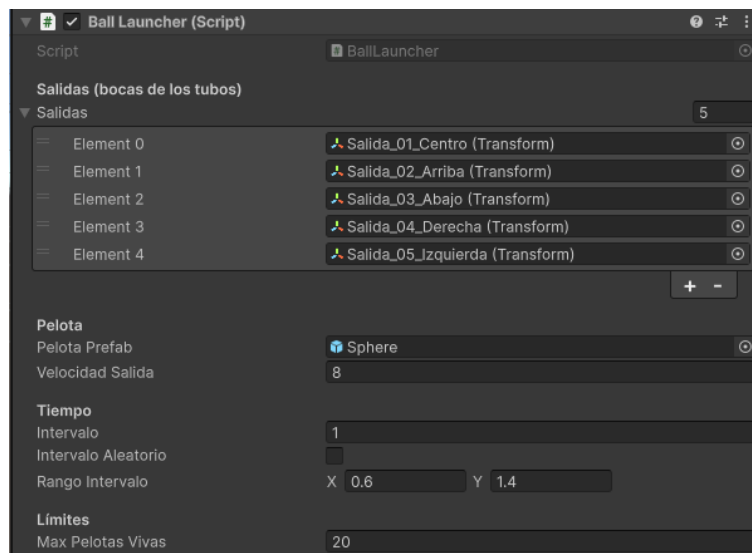


Figura 3-47: Inspector del código *BallLauncher*

Por su parte, el código *LauncherGripInput* se encarga de vincular el funcionamiento del lanzador con la acción directa del usuario, permitiendo iniciar o detener la prueba mediante los *grips* de los mandos. En concreto, el *grip* activa el sistema de lanzamiento, mientras que el izquierdo lo detiene, estableciendo un control sencillo e intuitivo sobre la escena. Además, este componente incorpora una lógica que evita activaciones no deseadas al comenzar la interacción, asegurando que el sistema solo responda cuando se produce una pulsación válida.

Por otro lado, el código permite asociar eventos de *bHaptics* al comienzo y al fin de la actividad, de modo que el usuario recibe una vibración en los guantes cuando pone en marcha o detiene el lanzador, estos eventos son llamados por el código por el nombre que se le da en el inspector de la Figura 3-48, por lo que es importante que el nombre escrito sea el mismo que aparece en el *Bhaptics Developer Hub*. Así, este componente no solo actúa como enlace entre el usuario y la lógica del sistema, sino que también refuerza sensorialmente las acciones de control sobre la prueba.

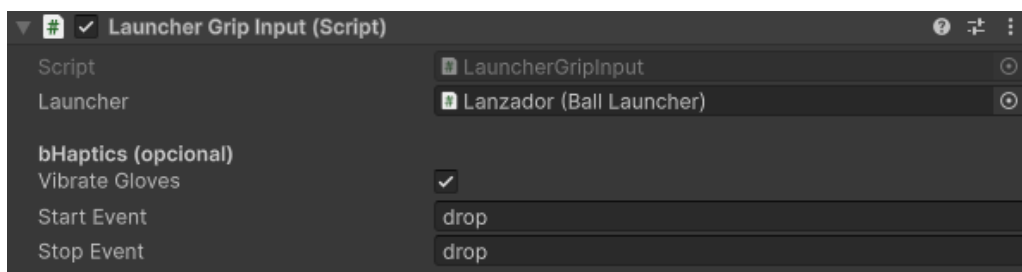


Figura 3-48: Inspector del código *LauncherGripInput*

Por último, el objeto *ImpactCounter* actúa como el elemento visual encargado de mostrar al usuario el número de impactos detectados durante la prueba. Para hacer posible esta función, el contador incorpora un texto gestionado mediante *TextMeshPro* y el script *ImpactCounterTMP*, encargado de actualizar su contenido conforme se van registrando los impactos. Gracias a ello, la escena no solo

detecta las colisiones a nivel interno, sino que también las refleja de forma visible para el usuario, reforzando el seguimiento de la prueba.

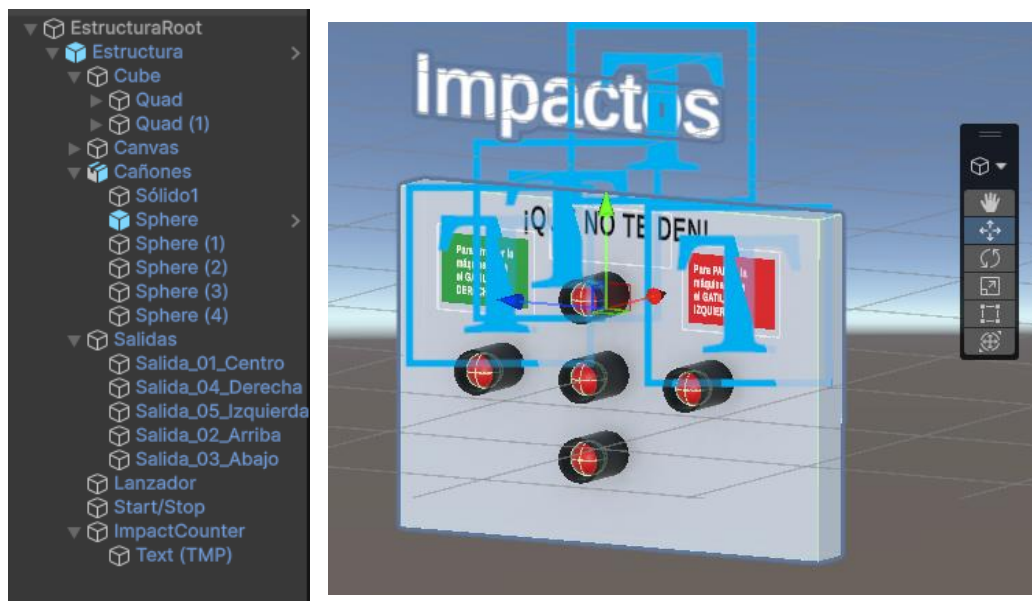


Figura 3-49: Cuerpo y Jerarquía de componentes del lanzador

3.- Bola

La propia bola se ha construido a partir de una esfera básica de *Unity*, a la que se le han añadido los componentes necesarios para que pueda participar en la prueba de forma física y reactiva, como se muestra en la Figura 3-50.

- *Sphere Collider*: define la superficie de colisión de la pelota y permite detectar el contacto con las zonas objetivo de la escena. En su configuración, debe ajustarse al tamaño real de la esfera para que el impacto se registre de forma correcta además del Trigger desactivado ya que los colliders del torso ya lo tienen.
- *Rigidbody* con *kinematic* desactivado hace que la pelota quede sometida al motor de físicas de *Unity*, permitiendo su lanzamiento y un movimiento natural dentro del entorno. Además, se desactiva también el factor gravedad para que la trayectoria de lanzamiento de las bolas sea rectilínea.

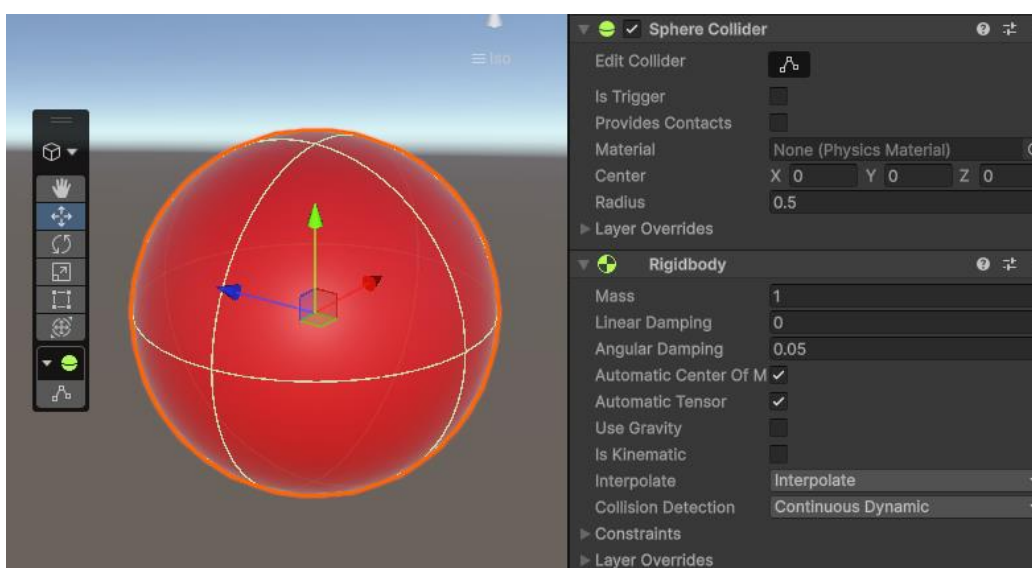


Figura 3-50: Cuerpo e inspector de la bola

Además, la pelota incorpora el código *BallHitBodyZones*, encargado de gestionar la interacción de las bolas con las distintas zonas corporales configuradas en la prueba. Su funcionamiento se basa en la detección de entrada en los *colliders* de dichas zonas, identificando a partir de su nombre si el impacto se ha producido en el hombro derecho, hombro izquierdo, cadera derecha o cadera izquierda, como se introduce en la Figura 3-51. Una vez reconocida la zona alcanzada, el código reproduce el evento háptico correspondiente mediante *bHaptics*, de forma que el usuario recibe la vibración en la parte adecuada del chaleco. Estas vibraciones son cuatro eventos hápticos configurados de manera independiente en el *Bhaptics Designer*, y se trata de vibraciones de alta intensidad, pero puntuales en la zona del impacto, cada evento se llama como la parte que tiene asignada, tal y como se muestra en la Figura 3-51.

De manera complementaria, este código también incluye un pequeño tiempo de espera entre impactos consecutivos para evitar dobles detecciones no deseadas cuando la pelota entra en contacto de forma muy rápida con una misma zona. Asimismo, actualiza el contador de impactos mostrado en pantalla, permitiendo llevar un seguimiento visual de los aciertos registrados durante la prueba. Por último, el código contempla de forma opcional la destrucción de la pelota tras el impacto, lo que facilita controlar su comportamiento una vez ha cumplido su función dentro de la escena.

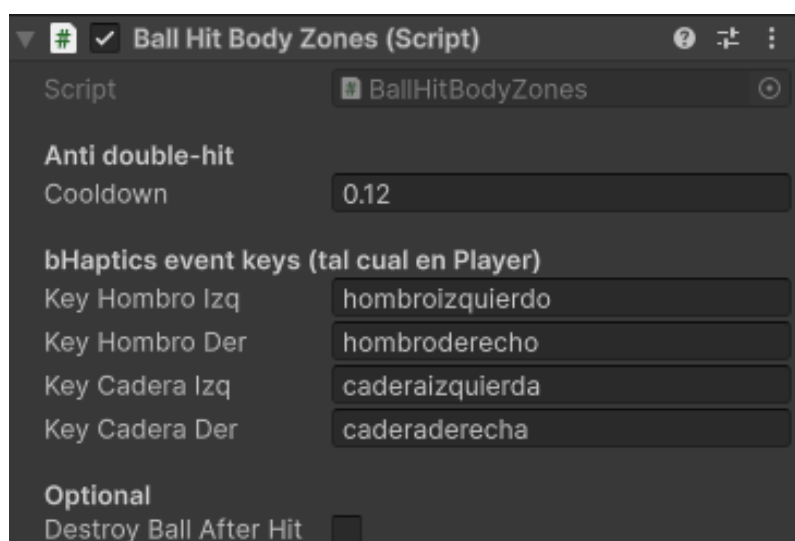


Figura 3-51: Inspector del código *BallHitBodyZones*

3.4.7 Prueba de puntería

La prueba de puntería se plantea como un ejercicio orientado a trabajar la precisión espacial, la estabilidad del apuntado y la coordinación dentro del entorno de realidad mixta. A diferencia de otras pruebas del proyecto, en este caso el usuario no debe manipular directamente un objeto virtual, sino que empleando un elemento físico real, dirigir el disparo hacia distintos objetivos virtuales distribuidos alrededor de su posición. De este modo, la escena introduce una forma de interacción distinta, más centrada en el control postural, la orientación en el espacio y la rapidez de reacción.

El funcionamiento general de la prueba se basa en la aparición sucesiva de dianas alrededor del usuario. Una vez iniciada la prueba, el usuario emplea el accesorio físico *ForcetubeVR* y la pistola háptica *provolverVR* como soporte para los mandos de Meta Quest 3, de forma que la experiencia de apuntado resulte más estable y natural. Al pulsar el gatillo derecho, el sistema lanza un disparo en la dirección establecida por el punto de salida configurado sobre el mando izquierdo o derecho, según el periférico que se esté empleando. Si el disparo impacta sobre la diana activa, esta desaparece, se incrementa el contador de aciertos y se genera automáticamente una nueva diana en otra posición del espacio. Además, cada disparo activa una respuesta háptica de retroceso, reforzando la sensación física de la interacción.

1.- Escena Base

El componente que define la construcción y dinámica de esta escena es el objeto que incorpora el código *TargetSpawner*, encargado de gestionar la generación de las dianas. Su cometido consiste en instanciar sucesivamente los objetivos en posiciones variables alrededor del usuario, evitando que la prueba se limite a un único punto fijo de disparo. Gracias a este sistema, la actividad gana dinamismo y obliga al usuario a reorientar continuamente el apuntado, reforzando así la sensación de variedad y el componente de precisión espacial. Para la configuración del inspector del código, como se muestra en la Figura 3-52, se definen puntos de referencia para la posición y orientación de la diana al generarse, en este caso el *CenterEyeAnchor* del *OVRCameraRig* para que la dianas se generen siempre alrededor de la posición del usuario y orientadas al mismo. También se configura el rango de altura, distancia de separación y ángulo de aparición para aumentar o disminuir la zona de aparición de las dianas.

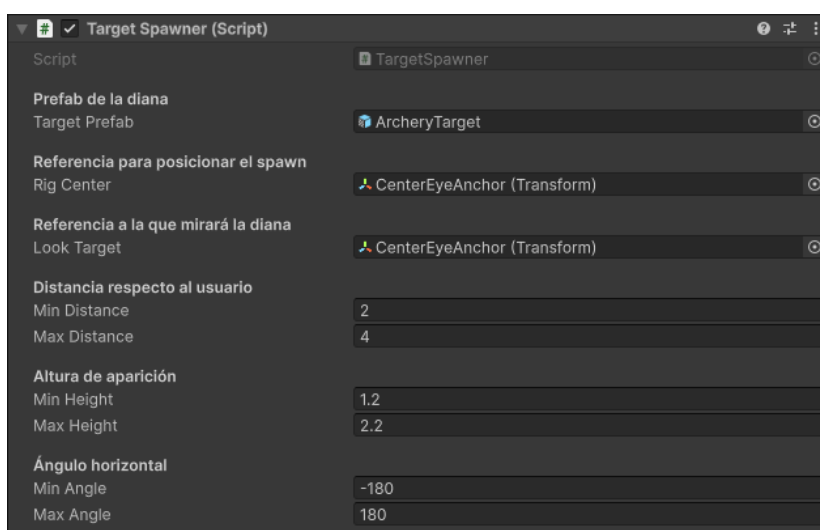


Figura 3-52: Inspector del código *TargetSpawner*

El elemento central de la mecánica de disparo se encuentra en el objeto *FirePoint*, sobre el que se incorpora el código *ProtubeShooter*. Este componente se encarga de gestionar el sistema de disparo mediante el lanzamiento de un *raycast* desde cada mando cada vez que el usuario acciona el gatillo, permitiendo comprobar de forma inmediata si se ha alcanzado una diana válida. A partir de que el *raycast* se lance de ambos controladores, la escena puede emplearse con distintas configuraciones de interacción, ya que el disparo puede apoyarse tanto en el uso de la pistola háptica de *ProTube VR* como en la configuración tipo fusil con *ForceTube* de la Figura 3-53.



Figura 3-53: *ForcetubeVR* (Izquierda) y pistola háptica *Quest3 Provolver* (Derecha) [20]

Para que la lógica de apuntamiento sea coherente, como referencia de punto de salida para el *raycast* en el inspector mostrado en la Figura 3-56 se introducen dos objetos hijos de los objetos *rightcontrolleranchor* y *leftcontrolleranchor* del *OVRCameraRig*. Estos objetos, además de seguir las

posiciones de cada mando, como se muestra en la Figura 3-54 y en la Figura 3-55, se les introduce un pequeño desvío en la rotación para que de esa manera el apuntado se ajuste a las posiciones de los mandos en los dispositivos físicos de la Figura 3-53, para los que está diseñada la prueba.

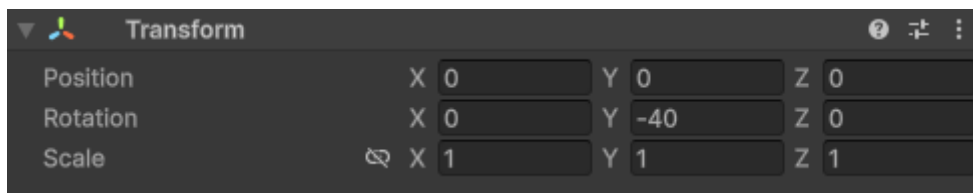


Figura 3-54: Rotación del objeto *FirePoint* del mando izquierdo

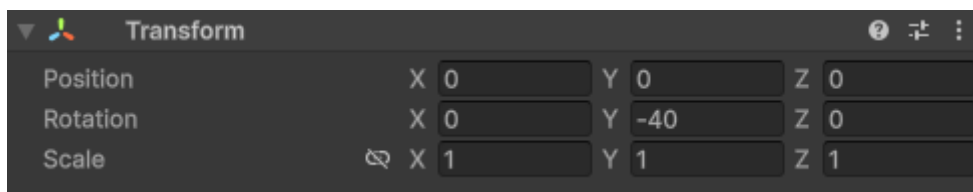


Figura 3-55: Rotación del objeto *FirePoint* del mando derecho

Los dispositivos de *ProtubeVR*, no solo introducen una referencia física para el apuntado, simulando una pistola y un fusil en este caso, sino que también introducen un evento háptico que simula el retroceso de disparo en cada arma. Esta interacción se hace posible mediante la inclusión en el proyecto del *plugin* del *ForceTubeVR Unity Package*, disponible como repositorio de archivos en *GitHub* [34]. En este se introducen una serie de funciones de código, *ForceTubeVRInterface.InitAsync* y *ForceTubeVRInterface.Kick*, que permiten llamar a la acción de retroceso de cada dispositivo en el caso que este se encuentre conectado por *bluetooth* a las gafas. Para consultar la implementación de esta interacción, se puede acceder al código *ProtubeShooter* en el repositorio en *Github* donde se almacenan todos los códigos elaborados para este proyecto [28], además su implementación en flujo está representada en el Anexo III: Diagramas de flujo de los códigos del proyecto.

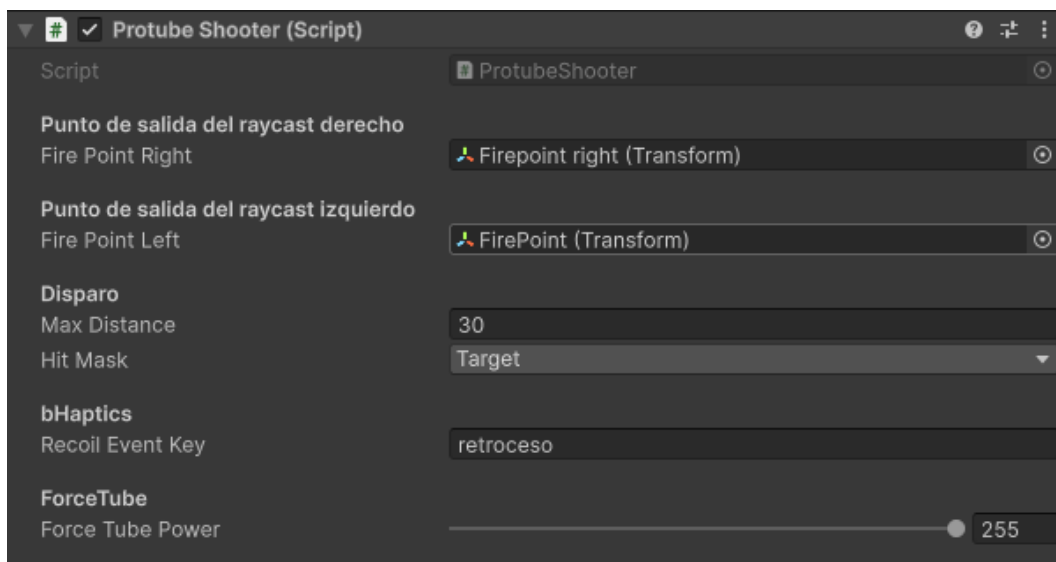


Figura 3-56: Inspector del código *ProtubeShooter*

Para que el usuario lleve un recuento de los disparos acertados se incluye un contador *TextMeshPro* como hijo del componente *CentreEyeAnchor* para que de esta manera el texto siga la cabeza del usuario y se sitúe en la parte alta del campo de visión, sin interferir en el apuntado a las dianas. Este contador es configurado por el código *TargetCounter*, que sigue una estructura muy similar a otros contadores

también presentes en este proyecto (consultar Anexo III: Diagramas de flujo de los códigos del proyecto).

Por último, la escena incorpora el componente necesario para enlazar la aplicación con el chaleco *bHaptics*. Este elemento permite que cada disparo vaya acompañado del efecto háptico de retroceso que viene predefinido en el paquete básico de *Bhaptics Designer* [30], de igual manera, se modificó ligeramente para que se adaptara a las características del proyecto como se muestra en la Figura 3-57.

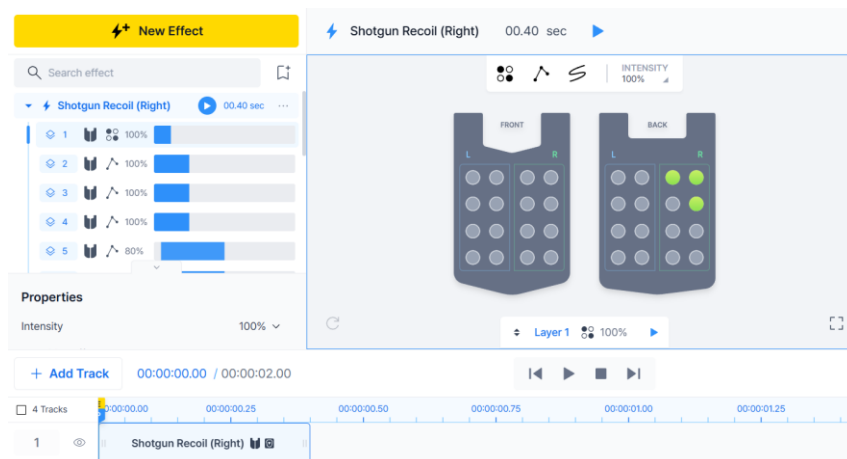


Figura 3-57: Configuración de vibración de retroceso en *Bhaptics designer*

2.- Diana

La diana constituye el elemento objetivo de la prueba y, por tanto, el objeto central de toda la escena. No obstante, su construcción se ha planteado de manera razonablemente sencilla, ya que en este caso no era necesario desarrollar un objeto complejo, sino disponer de una referencia visual clara y reconocible sobre la que el usuario pudiera dirigir el disparo. Para ello se ha recurrido al un modelo *Archery Target* del *Unity Asset Store* [35], mostrado en la Figura 3-58. lo que ha permitido incorporar un objeto visualmente adecuado y centrar el trabajo de desarrollo en la lógica de interacción de la escena, en lugar de invertir tiempo adicional en el modelado de una diana propia.



Figura 3-58: *Archery Target* del *Unity Asset Store*

A partir de ese objeto base, se ha añadido un *Box Collider*, cuya función es definir la zona de impacto sobre la que actúa el sistema de disparo. Gracias a este componente, el *raycast* lanzado desde los mandos puede detectar cuándo el disparo alcanza la diana, estableciendo así la condición necesaria para que el sistema la considere un objetivo válido. Aunque la geometría visual del modelo no sea rectangular, el uso de este tipo de *collider* resulta adecuado por su sencillez de ajuste y por ofrecer una solución estable y eficaz para la detección de impactos.

Junto a ello, como se muestra en la Figura 3-59, también se incorpora el código *TargetBehaviour*, cuya función es hacer que, una vez alcanzada, la diana desaparezca y se registre correctamente el impacto dentro de la prueba. De este modo, aunque se trate de un objeto simple, cumple de forma eficaz con todo lo necesario para sostener la dinámica básica de la escena.

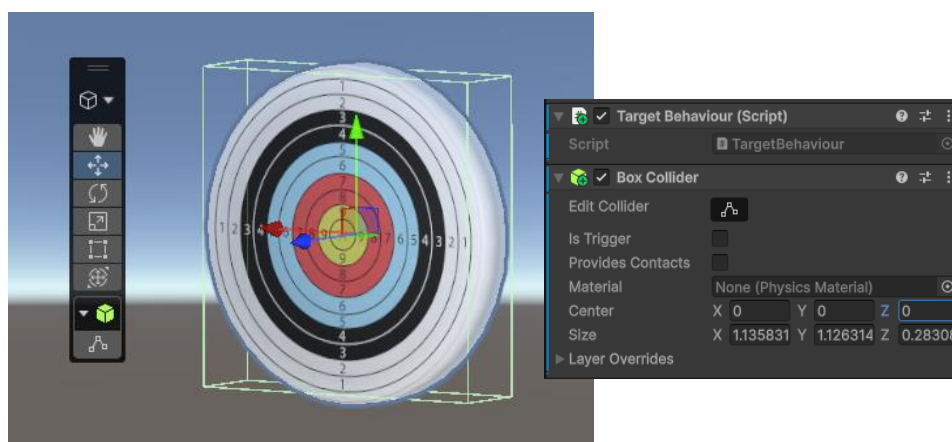


Figura 3-59: Cuerpo e inspector de componentes de la diana

3.4.8 Instalación del proyecto en las gafas Meta Quest 3

Durante la fase de desarrollo de las distintas escenas del proyecto, la comprobación de funcionalidades, objetos, interacciones y modificaciones realizadas en *Unity* se llevó a cabo de forma periódica mediante la conexión del equipo *Meta Quest 3* al ordenador a través de *AirLink*. Este sistema permite ejecutar la aplicación directamente desde el editor de *Unity*, de manera que, al iniciar la reproducción con el botón *Play*, la experiencia se visualiza inmediatamente en las gafas.

Sin embargo, en este modo de funcionamiento el proyecto no se está ejecutando realmente de manera autónoma en las gafas, sino en el propio ordenador. Las gafas actúan, en este caso, como dispositivo de visualización e interacción, mientras que el procesamiento de la aplicación, la carga de la escena y la gestión de los recursos recaen sobre el equipo conectado, en este caso, el PC.

Por este motivo, una vez finalizada o suficientemente avanzada la fase de desarrollo, se realizará la instalación del proyecto directamente en las gafas en modo *standalone*. Este procedimiento permite comprobar el comportamiento real de la aplicación en las condiciones finales de uso, es decir, ejecutándose de forma nativa en el propio dispositivo, sin dependencia de ningún ordenador.

Para realizar esta instalación fue necesario configurar el proyecto de *Unity* para la plataforma *Meta Quest* en los *build profiles*, accesible desde la ventana del editor como se muestra en la Figura 3-60.

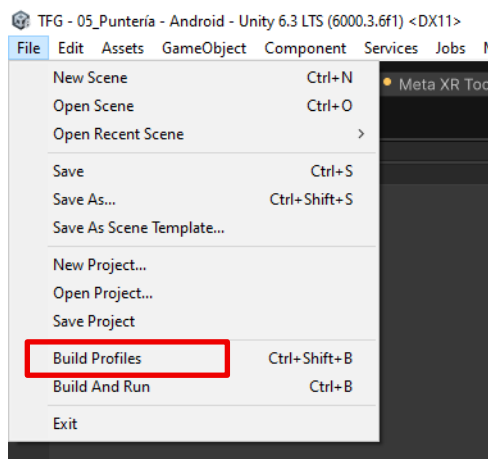


Figura 3-60: Acceso a los *Build Profiles*

A continuación, se procede a generar una versión compilada pulsando *clean build* en la ventana de los *Build Profiles* como se muestra en la Figura 3-61.

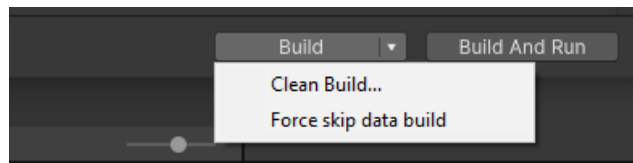


Figura 3-61: Botón para realizar el *Clean Build*

Una vez realizada la compilación, se genera un documento APK en el ordenador que a través de la ventana de Windows PowerShell se instala el proyecto en las gafas siguiendo la secuencia de la Figura 3-62.

```
Windows PowerShell
PS C:\Users\Garofano\Desktop> adb devices
List of devices attached
2G0YC5ZF9w0277    device

PS C:\Users\Garofano\Desktop> adb install '.\prueba final.apk'
Performing Streamed Install
Success
PS C:\Users\Garofano\Desktop> _
```

Figura 3-62: Instalación del programa en las gafas desde Windows PowerShell

Una vez completado este proceso, el proyecto queda instalado en el equipo Meta Quest 3, dando lugar a la versión final sobre la que se evaluarán los resultados obtenidos.

4 RESULTADOS

4.1 Descripción del apartado

En este apartado se presentan los resultados obtenidos tras el desarrollo del proyecto, centrándose en comprobar el correcto funcionamiento de las distintas pruebas implementadas y en valorar su utilidad desde un punto de vista didáctico. No se trata únicamente de mostrar el resultado visual final de las escenas, sino de analizar qué capacidades se han trabajado en las mismas y de qué manera contribuyen al aprendizaje del usuario dentro del entorno inmersivo.

Con este objetivo, los resultados se organizan en función de las competencias y metas alcanzadas a través del sistema desarrollado. De este modo, se estudiarán aspectos como la construcción de un escenario de realidad mixta, la manipulación de objetos virtuales, la mejora de la coordinación espacial y la incorporación de retroalimentación háptica como apoyo a la interacción. Esta estructura permite relacionar directamente cada resultado con los objetivos planteados al inicio del trabajo y valorar de forma más clara el alcance didáctico del entorno desarrollado.

4.2 Descripción general y objetivos de las pruebas

Las pruebas desarrolladas en el proyecto se han planteado como actividades interactivas con una finalidad concreta dentro del entorno de realidad mixta. En todas ellas, el usuario debe realizar una acción principal orientada a un objetivo definido, lo que aporta sentido a la interacción y hace que la experiencia resulte más dinámica, comprensible y motivadora. De este modo, cada escena no se limita a ofrecer una interacción aislada con objetos virtuales, sino que propone un pequeño reto que da dirección a la actuación del usuario.

En la prueba del aro, el usuario debe desplazar el aro a lo largo de la estructura siguiendo el recorrido establecido, mostrado en la **¡Error! No se encuentra el origen de la referencia.**, y tratando de evitar el mayor número posible de contactos con los elementos del circuito. De esta manera, la prueba exige un control preciso del movimiento y una atención continua durante toda la prueba.

En la prueba de baloncesto, la interacción se centra en el lanzamiento de un balón hacia la canasta dentro del entorno virtual. En este caso, el objetivo es encestar el mayor número posible de canastas, lo que convierte la prueba en una actividad directa y fácilmente comprensible, en la que la coordinación del gesto y la precisión en el lanzamiento adquieren un papel principal.



Figura 4-1: Circuito de barras y aro de la Prueba del Aro



Figura 4-2: Pelota y Canasta en la prueba del Baloncesto

En el puzle 3D, la tarea del usuario consiste en manipular las distintas piezas disponibles y colocarlas correctamente hasta completar la forma mostrada en la Figura 4-3. El objetivo de esta prueba es reconstruir la estructura final mediante la colocación adecuada de cada pieza en su posición correspondiente, lo que requiere orientación espacial, control en la manipulación y comprensión de la forma global del conjunto.

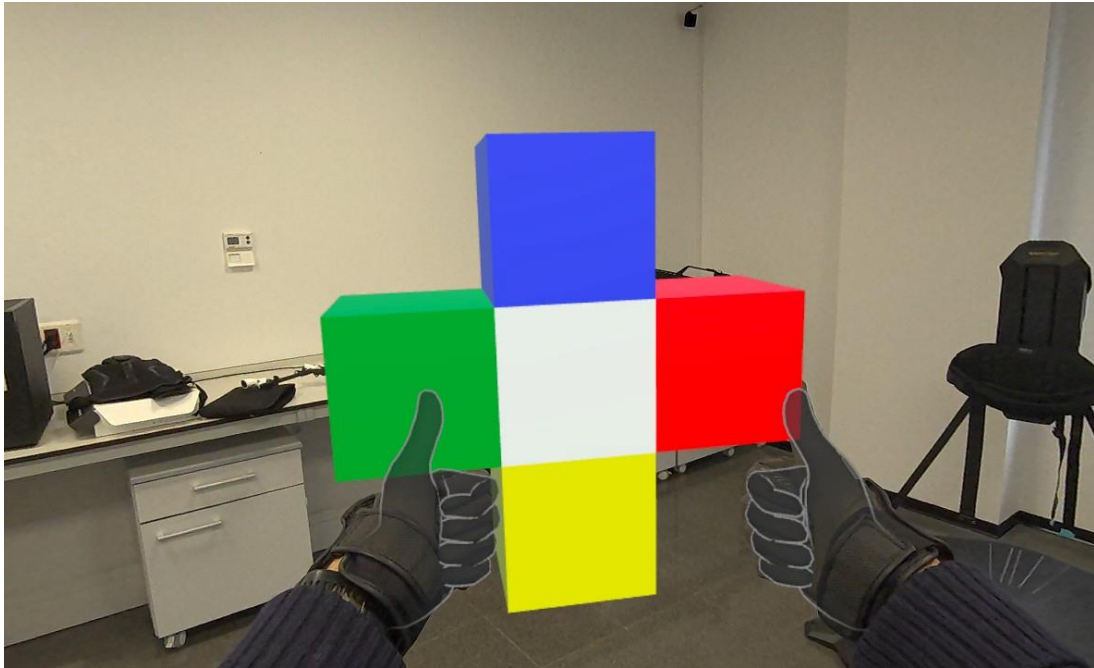


Figura 4-3: Puzzle 3D completado

En la prueba del lanzador, la dinámica cambia hacia una actividad basada en la respuesta corporal dentro del espacio de juego. En este caso, el usuario debe esquivar las bolas que son lanzadas por la estructura de la Figura 4-4 en su dirección, de manera que el objetivo principal pasa por evitar los impactos. Aunque no existe manipulación directa de objetos, la prueba exige atención, anticipación y coordinación espacial dentro del entorno de realidad mixta.



Figura 4-4: Estructura lanzadora de la prueba del Lanzador

Por último, en la prueba de puntería, el usuario debe utilizar el sistema de disparo implementado para apuntar y acertar al mayor número posible de dianas. El objetivo es, por tanto, alcanzar la máxima

cantidad de blancos, planteando una actividad en la que la precisión, la estabilidad del apuntado y la coordinación del disparo resultan fundamentales.

En conjunto, la presencia de un objetivo definido en cada una de las pruebas refuerza el carácter interactivo del proyecto y contribuye a que la experiencia resulte más completa y atractiva. Cada escena propone una forma distinta de participación, lo que ayuda a mantener el interés del usuario y a dar sentido a las acciones realizadas dentro del entorno.

4.3 Escenario de Realidad Mixta

Uno de los principales resultados alcanzados en este proyecto ha sido la construcción de un escenario de realidad mixta funcional, estable y adaptable al espacio real de uso. A diferencia de lo que ocurre en un entorno de realidad virtual convencional, en este caso el escenario no queda completamente prefijado desde el diseño, sino que depende del lugar en el que el usuario se encuentre en el momento de la experiencia.

Este resultado se apoya, en primer lugar, en el uso del *passthrough* y de los componentes base de realidad mixta, que permiten mantener visible el entorno real mientras se superponen sobre él los objetos virtuales. Gracias a esta base, los elementos del proyecto no aparecen aislados en un espacio artificial, sino integrados sobre referencias físicas reales, lo que favorece una percepción más natural de las distancias, la orientación y la posición relativa de los objetos. Además, en el caso de la prueba del baloncesto, se añaden componentes que simulan superficies del entorno real como el suelo, en la Figura 4-5, se muestran la canasta y la pelota posadas sobre el suelo físico.



Figura 4-5: Canasta y pelota virtual sobre suelo físico

Sin embargo, el aspecto más destacable de este apartado no es solo esa integración visual, sino la forma en la que se construyen las escenas. En las distintas pruebas del proyecto, el usuario participa activamente en el montaje del escenario mediante acciones sencillas realizadas con los mandos, estas acciones se explican al usuario en paneles introductorios que se explican más adelante (Página 70). Esta característica aporta una ventaja clara: el sistema no obliga al usuario a jugar en un escenario exacto previamente preparado, sino que le permite colocar los elementos donde el uso sea más cómodo, visible y seguro. En la Figura 4-6, se muestra como un objeto puede ser situado en distintas partes del entorno real.



Figura 4-6: Mismo objeto, la pantalla del menú inicial, situado en distintas partes de la habitación

Además, esta construcción del escenario sigue una dinámica de control simple e intuitiva. El usuario emplea los botones del mando para instanciar, recolocar o eliminar los distintos elementos del juego, mientras que otros controles permiten comenzar la actividad, manipular los objetos o regresar al menú principal. Por ejemplo, como se muestra en la Figura 4-7, en la escena del Puzle 3D, antes de empezar el puzle, el usuario instancia los cubos donde le sea más cómodo para luego manipularlos.

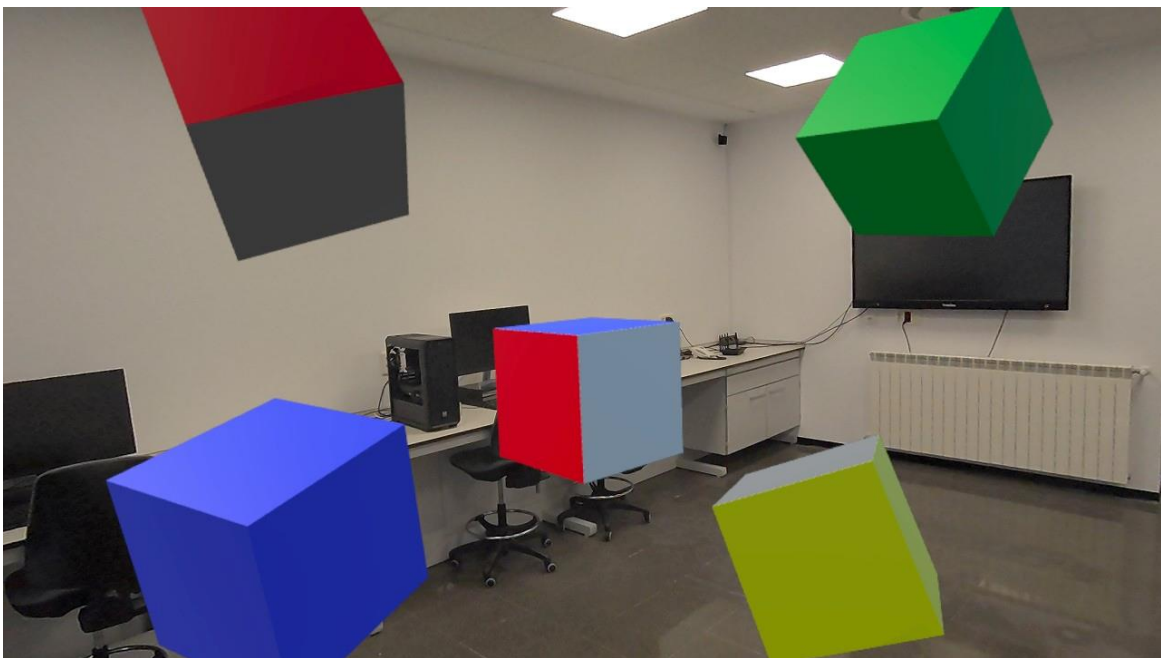


Figura 4-7: Cubos piezas y cubo central instanciados para empezar el puzle 3D

En definitiva, esta manera de construir las escenas refuerza el valor didáctico del proyecto. El usuario no solo interactúa con objetos virtuales ya presentes, sino que comprende desde el inicio que, en un entorno de realidad mixta, el espacio de juego se configura en relación directa con el mundo real. Así, antes incluso de comenzar cada actividad, se familiariza con una idea fundamental de este tipo de tecnologías: los contenidos digitales no sustituyen el entorno físico, sino que se apoyan en él, se colocan sobre él y adaptan su disposición a sus características.

4.4 Elementos de apoyo a la experiencia

Además de las mecánicas principales de cada una de las pruebas, el proyecto incorpora una serie de elementos que definen la experiencia de uso y facilitan la interacción del usuario con el entorno. Aunque estos componentes no constituyen por sí mismos el núcleo de las actividades desarrolladas, su presencia resulta fundamental para que el sistema sea más claro, accesible y fácil de utilizar desde el primer momento.

Uno de los elementos más importantes en este sentido es la pantalla de menú, diseñada como punto de acceso a las distintas escenas del proyecto. A través de ella, el usuario puede desplazarse entre pruebas de forma sencilla, seleccionando cada actividad mediante un sistema de interacción adaptado a los mandos. Para ello, se incorpora un puntero láser, que actúa como medio de selección sobre los botones de la interfaz y permite una navegación más intuitiva dentro del entorno inmersivo. Gracias a este recurso, el cambio entre escenas se realiza de forma ordenada y comprensible, sin necesidad de recurrir a menús externos a la aplicación.



Figura 4-8: Puntero de la pantalla del menú inicial

También se incorporó en la pantalla, concretamente en los botones, una lógica de cambio de color al ser pulsados. Este recurso refuerza visualmente la acción realizada con el puntero láser, facilitando la interacción y haciendo la experiencia más clara y atractiva para el usuario. Asimismo, como se muestra en la Figura 4-9 cada botón presenta una respuesta diferente, para introducir una mayor variedad visual a la pantalla.

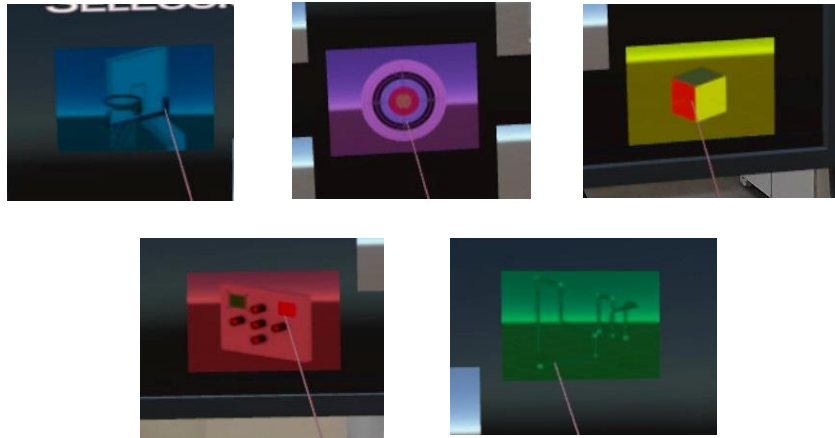


Figura 4-9: Cambio de color en botones al ser pulsados

Junto a ello, en las distintas pruebas se han incluido los paneles introductorios de la Figura 4-10 que aparecen al inicio de cada escena y ofrecen al usuario una explicación breve sobre su funcionamiento. Estos paneles siguen la mirada del usuario y permiten comprender de antemano qué debe hacerse en cada actividad, qué controles se utilizan y cuál es el objetivo de la prueba antes de comenzar la interacción. De este modo, facilitan la adaptación del usuario al sistema,

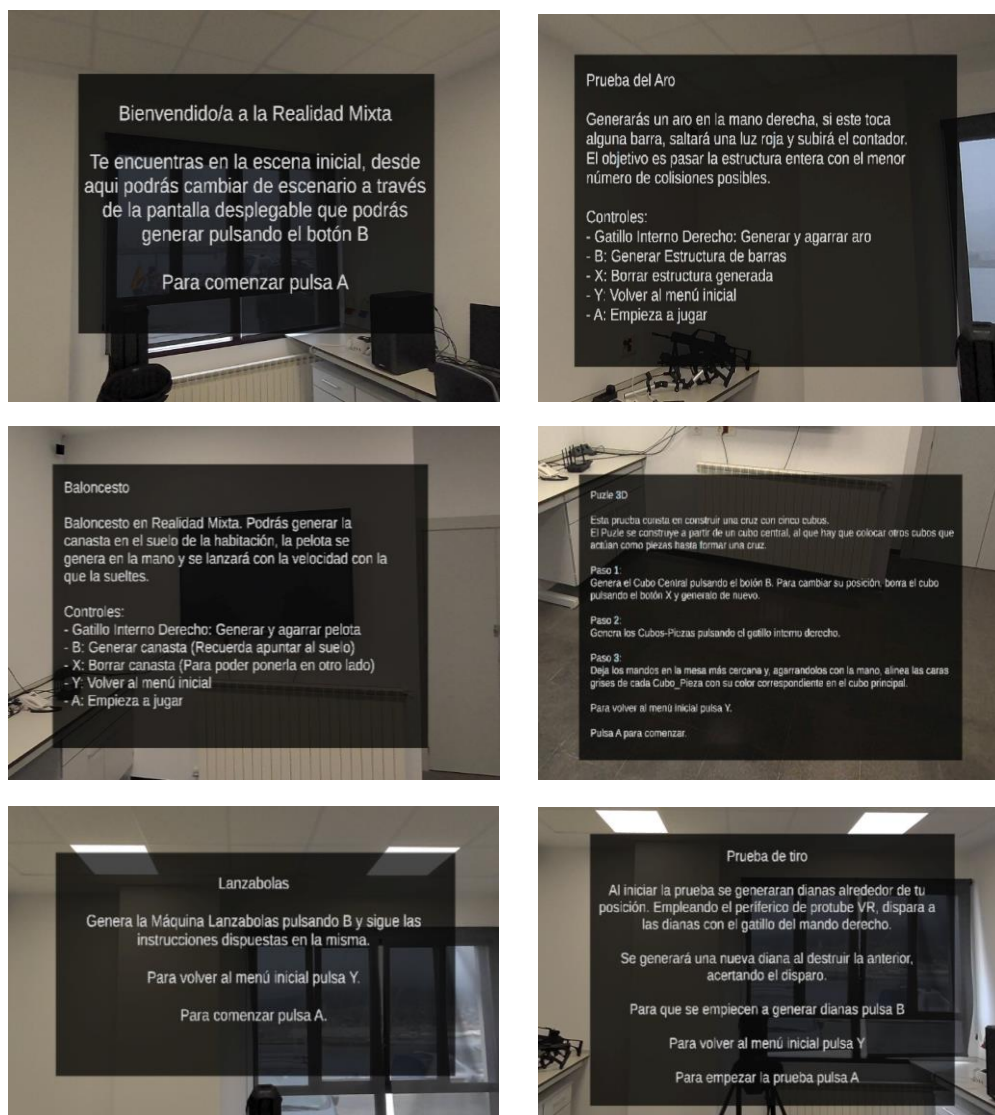


Figura 4-10: Paneles introductorios de distintas escenas del proyecto

Por otro lado, también se han incorporado elementos de seguimiento, como los textos de contador presentes en algunas pruebas mostrados en la Figura 4-12, que permiten visualizar de forma directa el número de colisiones, impactos o acciones realizadas. Su función principal es ofrecer una referencia cuantitativa del progreso del usuario durante la actividad, haciendo más sencillo interpretar el desarrollo de la prueba y valorar el resultado obtenido.



Figura 4-11: Contadores de las distintas pruebas

Junto a estos elementos de seguimiento, el sistema incluye también indicadores visuales inmediatos que actúan como confirmación instantánea de una acción concreta. Un ejemplo de ello es la luz roja utilizada en la prueba del aro, que se activa cuando se produce el contacto con la estructura, como aparece en la Figura 4-12. A diferencia de los contadores, este tipo de recurso no busca registrar la evolución de la actividad, sino reforzar visualmente un evento puntual, permitiendo al usuario identificar de forma clara y rápida que la interacción ha tenido lugar.



Figura 4-12: Luz indicadora de colisiones en la Prueba del Aro

En conjunto, estos componentes adicionales refuerzan la utilidad del sistema desarrollado, ya que no solo acompañan a las mecánicas principales, sino que contribuyen a hacer la experiencia más organizada, comprensible y completa. Gracias a ellos, el proyecto presenta un entorno más cuidado, en el que la navegación, la orientación inicial, el seguimiento del progreso y el *feedback* visual forman parte activa de la experiencia.

4.5 Manipulación, precisión y coordinación espacial

Se ha conseguido el desarrollo de un conjunto de pruebas orientadas a trabajar la manipulación de objetos virtuales, la precisión de movimiento y la coordinación espacial dentro de un entorno de realidad mixta. A través de las distintas escenas implementadas, el usuario no se limita a observar elementos digitales, sino que interactúa con ellos de forma activa, aprendiendo a controlar su posición, su trayectoria y su relación con el espacio real en el que se desarrolla la experiencia.

En la prueba del aro, esta capacidad se trabaja mediante la manipulación directa del objeto con el mando. El usuario, como se muestra en la Figura 4-13, debe sostener el aro, orientarlo correctamente y desplazarlo y evitar que colisione con la estructura objetivo, lo que exige controlar con precisión el movimiento de la mano y su correspondencia con el elemento virtual. De este modo, la escena permite desarrollar una primera toma de contacto con la interacción espacial, centrada en la coordinación entre la acción física del usuario y la respuesta del objeto en el entorno de realidad mixta.

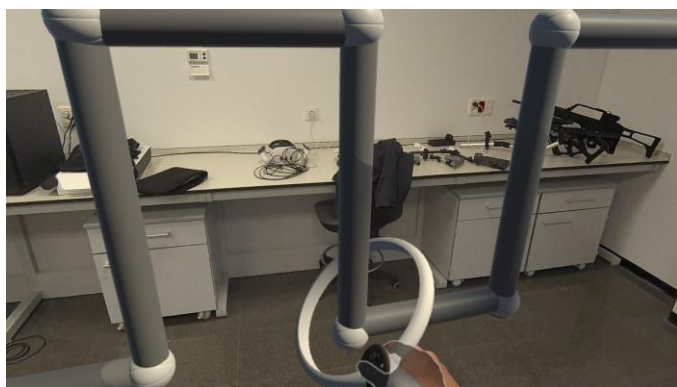


Figura 4-13: Aro pasando por la estructura de barras

En la prueba del baloncesto, además de la manipulación inicial de la pelota, se incorpora la comprensión de la física del lanzamiento como parte del aprendizaje, como se muestra en la Figura 4-14. El usuario no solo debe sujetar y soltar el objeto, sino también entender cómo influyen la dirección, la fuerza y la trayectoria en el resultado final del tiro. Esto añade un nivel mayor de complejidad a la interacción, ya que obliga a relacionar el gesto realizado con el comportamiento físico de la pelota.

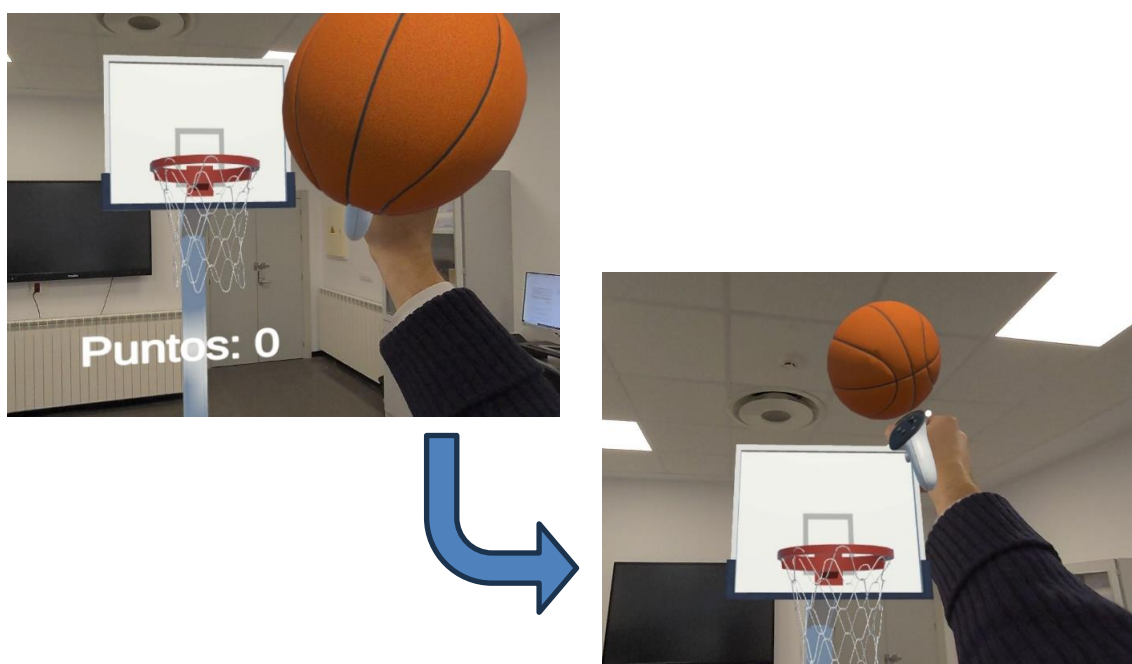


Figura 4-14: Dinámica de lanzamiento en la prueba de baloncesto

En el puzzle 3D, la interacción se centra en la manipulación de piezas con una misma mano, lo que exige un control más fino y continuado del objeto. En este caso, el usuario debe agarrar con la mano, desplazar, orientar y colocar cada pieza en su posición correcta, prestando atención tanto a su forma como a su encaje dentro de la estructura. Esta escena refuerza especialmente la coordinación espacial y la precisión en movimientos de ajuste, al requerir una interacción más detallada y sostenida que en las pruebas anteriores.

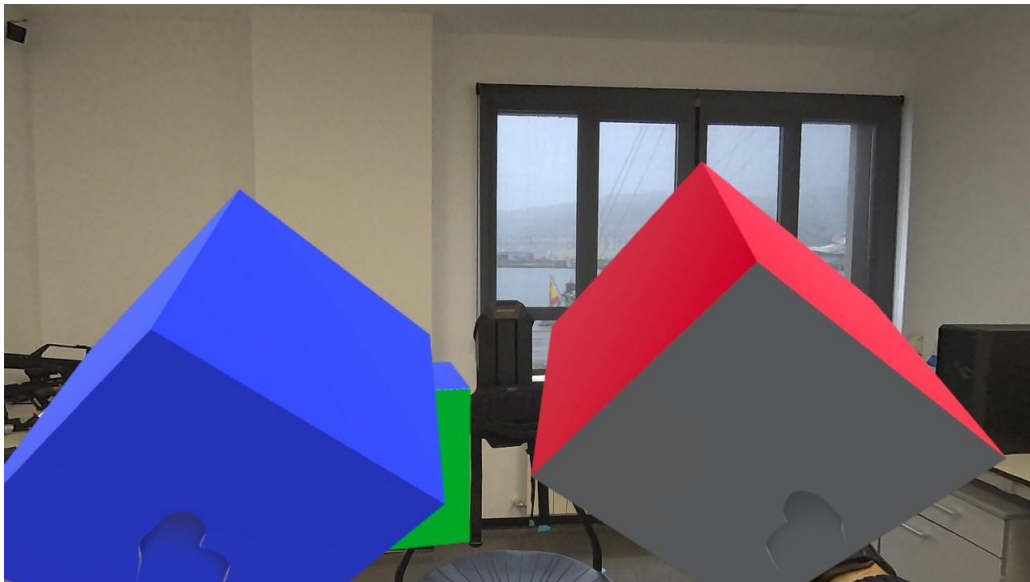


Figura 4-15: Usuario agarrando los cubos del puzzle 3D con las manos

En la prueba de puntería, el aprendizaje se orienta hacia el control de la dirección del movimiento y la precisión sobre un objetivo concreto. En este caso, el usuario debe apuntar y ejecutar la acción de disparo de manera que el impacto se produzca en el blanco, lo que obliga a ajustar la posición de la mano, la orientación del elemento de interacción y la coordinación entre percepción visual y respuesta motora. A diferencia de las escenas centradas en el agarre o la colocación de objetos, esta prueba pone el foco en la exactitud del gesto y en la capacidad de dirigir la acción hacia un punto determinado del espacio, reforzando así la precisión y el control espacial dentro del entorno de realidad mixta.



Figura 4-16: Usuario apuntando con pistola háptica en la prueba de puntería

Por último, aunque en la prueba del lanzador no se produce una manipulación directa de objetos por parte del usuario, sí se trabaja la coordinación espacial dentro del entorno de realidad mixta. En este caso, la interacción se basa en la capacidad de percibir la trayectoria de las bolas, anticipar su recorrido y ajustar la posición del cuerpo para esquivarlas correctamente. De este modo, la escena incorpora una forma diferente de relación con el espacio, centrada en la respuesta del usuario ante estímulos en movimiento.

4.6 Retroalimentación háptica

Otro de los resultados más significativos del proyecto ha sido la integración de retroalimentación háptica como apoyo a la interacción dentro del entorno de realidad mixta. Este aspecto permite que determinadas acciones no se perciban únicamente a través de la vista, sino también mediante estímulos táctiles asociados a eventos concretos del sistema.

En el caso del puzle 3D, la retroalimentación háptica se incorpora mediante los guantes *bHaptics tactglove DK2* de la Figura 4-17 acompaña directamente a la manipulación de las piezas. Durante esta prueba, el usuario recibe una respuesta táctil en momentos concretos de la interacción, como al agarrar una pieza, soltarla o completar correctamente su encaje. Esto hace que acciones sencillas como coger, orientar o colocar un elemento resulten más claras desde el punto de vista perceptivo, reforzando la sensación de contacto con el objeto virtual. Además, esta respuesta táctil actúa como una confirmación inmediata del progreso de la tarea, ya que ayuda a distinguir cuándo la colocación de las piezas ha sido válida y cuándo no lo ha sido.



Figura 4-17: Guantes *bHaptics* en vista de juego

Por su parte, en la prueba del lanzador, la retroalimentación háptica pasa a un nivel más amplio. Por un lado, si el usuario dispone de los guantes, la escena ya contempla vibraciones en los mismos, asociadas al inicio y al final de la actividad, de modo que las acciones de control sobre la prueba quedan reforzadas también desde el tacto. Por otro lado, el diseño de esta escena se ha planteado específicamente para incorporar respuesta corporal mediante el chaleco *bHaptics tactsuit x40*, que cubre el torso del usuario, como se muestra en la Figura 4-18, de forma que los impactos de las pelotas puedan percibirse en distintas zonas del torso según el punto alcanzado.



Figura 4-18: Usuario con el chaleco *bHaptics TactSuit x40*

En la prueba de puntería también se ha logrado incorporar dos lógicas de disparo diferentes, con su respectiva respuesta háptica. Además de emplear el chaleco háptico con una vibración simulando el retroceso del arma se ha introducido la posibilidad de emplear dos periféricos hápticos distintos para simular dos tipos de disparo cometido por diferentes armas según el periférico que se encuentre conectado a las gafas. Por un lado, se ha implementado un modo de disparo con pistola, utilizando la pistola háptica de *ProTube VR*, que ofrece un efecto de retroceso como el de una pistola semiautomática. Y por otro, se ha desarrollado una segunda lógica de disparo orientada al uso como fusil, empleando el dispositivo *ForceTubeVR*, lo que introduce una sensación de manejo distinta y un componente físico simulando el retroceso en la culata de un fusil al disparar. Este resultado es especialmente interesante porque muestra la capacidad del sistema para adaptarse a diferentes periféricos, haciendo que la prueba sea más completa, flexible y enriquecedora para el usuario.



Figura 4-19: Usuario empleando el *ForcetubeVR* (Izquierda) y la pistola *ProvolverVR* (Derecha)

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, valorando su grado de cumplimiento a partir del sistema desarrollado. Asimismo, se presentan las principales conclusiones derivadas del trabajo realizado y se plantean posibles líneas futuras orientadas a ampliar las capacidades del entorno, perfeccionar la experiencia de uso e incorporar nuevas funcionalidades que refuercen su utilidad didáctica a corto y a largo plazo.

5.2 Conclusiones

5.2.1 *Escenario de realidad mixta*

El desarrollo de este proyecto ha permitido cumplir de forma satisfactoria el objetivo de crear un escenario de realidad mixta comprensible y adaptable al entorno de uso. El sistema desarrollado no depende de una disposición fija, sino que permite colocar y organizar las distintas pruebas sobre el espacio real disponible, haciendo que la experiencia pueda trasladarse a diferentes ubicaciones sin perder coherencia.

A través de acciones simples con los mandos, el usuario puede instanciar, recolocar o eliminar elementos principales de las distintas pruebas, de manera que la preparación de cada escena pasa a formar parte de la propia experiencia.

Esto resulta especialmente importante porque una de las claves de la realidad mixta es precisamente esa relación directa entre el entorno físico y los elementos virtuales. En este trabajo se ha conseguido que esa integración no sea solo visual, sino que forme parte real del funcionamiento de las distintas escenas. Por tanto, se puede decir que se ha desarrollado un entorno funcional, claro y adecuado como base para introducir al usuario en este tipo de tecnologías.

5.2.2 *Interacción con objetos y coordinación espacial*

Otro de los objetivos que puede darse por cumplido es el relacionado con la interacción con objetos virtuales. A lo largo del proyecto se han desarrollado distintas pruebas en las que el usuario debe agarrar, mover, orientar, lanzar, encestar, colocar o apuntar, lo que da lugar a una experiencia variada y activa.

Gracias a estas dinámicas, el sistema obliga al usuario a interactuar con los objetos de forma continua. Esto permite trabajar la coordinación espacial, la precisión de movimiento y la comprensión de cómo se comportan los objetos dentro del entorno mixto. En este sentido, pruebas como la del aro, el

baloncesto, el puzle 3D o la de puntería muestran que el proyecto cumple bien su función como entorno de aprendizaje y familiarización con este tipo de interacción.

5.2.3 Integración de dispositivos y retroalimentación háptica

También se ha alcanzado de forma satisfactoria el objetivo de integrar dispositivos hápticos y periféricos físicos dentro de la experiencia. La incorporación de los guantes *bHaptics* en el puzle 3D, del chaleco háptico en la prueba del lanzador y de periféricos como la pistola *ProTube VR* o el *ForceTubeVR* en la prueba de puntería demuestra que el proyecto no se apoya solo en el apartado visual, sino que añade también una respuesta táctil y física que hace la interacción más completa.

Esta parte tiene bastante peso dentro del trabajo, ya que refuerza la sensación de inmersión y hace más clara la relación entre lo que el usuario hace y lo que el sistema devuelve. Las vibraciones en agarres, impactos, encajes o disparos ayudan a que cada prueba resulte más realista e intuitiva. Además, el hecho de haber integrado distintos dispositivos muestra que el entorno desarrollado tiene bastante margen de ampliación de cara a futuras mejoras.

5.2.4 Conclusión final

Puede decirse que este Trabajo Fin de Grado ha permitido cumplir de forma satisfactoria los objetivos planteados al inicio. A lo largo del desarrollo se ha trabajado con un entorno nuevo basado en *Unity*, así como con las gafas *Meta Quest 3* y las distintas herramientas, configuraciones y paquetes necesarios para su correcto funcionamiento, lo que ha supuesto un proceso de aprendizaje constante desde las primeras fases del proyecto. Todo ello ha implicado no solo comprender el funcionamiento general de este tipo de tecnologías, sino también aprender a integrarlas dentro de una experiencia interactiva real y coherente.

A pesar de la curva de aprendizaje inicial, los objetivos del trabajo se han alcanzado de manera eficaz. Se ha conseguido construir un escenario de realidad mixta funcional y adaptable al espacio real, con distintas mecánicas de interacción con objetos virtuales y con la inclusión de retroalimentación háptica mediante varios dispositivos distintos; todo ello adaptado de manera sencilla para la comprensión y utilización del usuario. De esta forma, el resultado final no se limita a una prueba aislada, sino que constituye un conjunto de escenas y actividades con sentido dentro de una propuesta común.

Otro aspecto importante del proyecto ha sido el trabajo realizado a nivel de programación. A lo largo del desarrollo se han creado y ajustado una gran variedad de códigos y funciones que han permitido dar respuesta a las necesidades concretas de cada escena, desde la aparición y manipulación de objetos hasta la detección de acciones, la gestión de eventos o la integración de respuestas hápticas. Esto ha hecho que el proyecto no solo tenga valor por el resultado visible final, sino también por todo el trabajo técnico que hay detrás de su funcionamiento.

En definitiva, el Trabajo Fin de Grado ha servido tanto para alcanzar los objetivos planteados como para adquirir una base sólida de conocimientos sobre el desarrollo de experiencias inmersivas en *Unity* y *Meta Quest 3*. Además, el sistema obtenido deja abierta la posibilidad de seguir incorporando mejoras, nuevas escenas y formas de interacción en el futuro, por lo que puede considerarse una base útil y bien construida para continuar trabajando dentro del campo de la realidad mixta.

5.3 Líneas Futuras

Para continuar con el estudio de la tecnología empleada para este proyecto, pueden plantearse distintas líneas futuras de desarrollo a partir de la base construida.

En primer lugar, a corto plazo, una posible evolución del proyecto sería profundizar en el uso de herramientas como el *Mixed Reality Utility Kit* de Meta para desarrollar escenarios de realidad mixta vinculados a un entorno físico concreto. Mientras que en este trabajo se ha buscado crear un entorno adaptable a distintos espacios reales, una línea futura interesante consistiría en diseñar una experiencia

fija asociada a un escenario específico, como podría ser, por ejemplo, el puente de un barco. Esto permitiría aprovechar con mayor precisión las características del espacio real, integrando los elementos virtuales de manera más directa y coherente con un contexto determinado.

Por otro lado, como se explica en el desarrollo, la funcionalidad de este proyecto se basa en códigos elaborados en lenguaje C#, y se ha demostrado que esta herramienta resulta eficaz para este tipo de trabajos. No obstante, este tipo de programación en ocasiones puede resultar poco intuitiva y difícil de abordar de primeras para todo aquel que no esté familiarizado con el mundo de la programación. Una buena línea a tener en cuenta sería la elaboración de trabajos de este tipo en esta misma plataforma, pero empleando la programación visual, es decir, por cajas. Para ello, *Unity* ofrece la herramienta *Visual Scripting*, a través de esta se pueden elaborar funciones de manera más visual y fácil de modificar. En este trabajo no se emplea esta herramienta debido a que todavía no se encuentran integradas las funciones de compatibilización con el entorno del equipo de *Meta Quest 3*, si en un futuro se implementase esta capacidad en *Unity* sería la mejor opción para este tipo de proyectos.

Como última línea a corto plazo, se propone el desarrollo de un entorno que combinase las principales ventajas de la realidad virtual y de la realidad mixta dentro de una misma experiencia. Esta evolución permitiría aprovechar, por un lado, la capacidad de la realidad mixta para integrar el entorno físico real en la interacción y, por otro, el potencial de la realidad virtual para construir escenarios totalmente controlados e independientes del espacio del usuario. Además, esta línea abriría la puerta a la incorporación de nuevos periféricos, como plataformas de locomoción u otros dispositivos complementarios, ampliando todavía más las posibilidades de inmersión e interacción del sistema. Esto se podría desarrollar de la misma manera en la que lo hace este proyecto, mediante el desarrollo de distintas escenas en la misma máquina virtual; cada escena se podría construir en base al escenario que más potencie el objetivo de cada una.

Por último, a más largo plazo, una línea futura especialmente interesante sería el desarrollo de un entorno compartido que pudiera ser utilizado por varios usuarios al mismo tiempo. Esto permitiría crear un escenario de aprendizaje grupal en un mismo entorno de realidad mixta, de forma que varios participantes pudieran interactuar entre sí y con los elementos virtuales de manera simultánea. De esta manera, se podría potenciar el adiestramiento de todo grupo de trabajo que trabaje en un mismo entorno de manera coordinada, como puede ser, nuevamente, el puente de un barco. No obstante, una propuesta de este tipo supondría un salto importante con respecto al trabajo actual, ya que introduciría componentes de sincronización, colaboración y experiencia colectiva, ampliando notablemente el alcance del proyecto y sus posibles aplicaciones formativas.

6 BIBLIOGRAFÍA

- [1] «Link Trainer: El primer simulador de vuelo,» 14 Enero 2021. [En línea]. Available: <https://flightdemy.com/blog/primer-simulador-vuelo>. [Último acceso: 2 Enero 2026].
- [2] UNIR, «¿Qué es la realidad virtual y para qué sirve?,» 2023. [En línea]. Available: <https://www.unir.net/revista/ingenieria/realidad-virtual-que-es/>. [Último acceso: 3 Enero 2026].
- [3] «CEI Escuela de Diseño y Marketing,» 2022. [En línea]. Available: <https://cei.es/realidad-virtual>. [Último acceso: 2 Enero 2026].
- [4] G. Makransky y G. Petersen, The Cognitive Affective Model of Immersive Learning (CAMIL): a Theoretical Research-Based Model of Learning in Immersive Virtual Reality., 2021.
- [5] Barcelona Reality VR, «La evolución de la realidad virtual: desde sus inicios hasta hoy,» [En línea]. Available: <https://www.barcelonarealityvr.com>. [Último acceso: 10 Enero 2026].
- [6] Microsoft Learn, «¿Qué es la realidad mixta?,» Microsoft, [En línea]. Available: <https://learn.microsoft.com/es-es/windows/mixed-reality/discover/mixed-reality>. [Último acceso: 3 Enero 2026].
- [7] J. Gallego García, *Desarrollo de un módulo interactivo para la simulación de entornos inmersivos mediante realidad virtual*, Marín: Centro Universitario de la Defensa, 2025.
- [8] M. Hayes y A. Downie, «What is augmented reality? IBM Cloud education,» IBM, 2022. [En línea]. Available: <https://www.ibm.com/think/topics/augmented-reality>. [Último acceso: 13 Enero 2026].
- [9] F. Navarro Pulido, A. Martínez y J. M. Martínez, Realidad virtual y realidad aumentada. Desarrollo de aplicaciones., Grupo Editorial RA-MA, 2018.
- [10] J. Cabero Almenara, I. de la Horra Villacé y J. Sánchez Bolado, La realidad aumentada como herramienta educativa. Aplicación a Educación Infantil, Primaria, Secundaria y Bachillerato., Ediciones Parafino, 2018.
- [11] S. García-Cubillana López, *Modelado gráfico de un dron aéreo de combate para visualización inmersiva en realidad mixta*, Marín: Centro Universitario de la Defensa, 2025.

- [12] J. Radianti, T. A. Majchrzak, J. Fromm y I. Wohlgenannt, «A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda,» 2020.
- [13] PRLOG, «DiSTI Delivers Siemens Energy's First 3D Virtual Maintenance Trainer,» [En línea]. Available: <https://www.prlog.org/12691463-dist-delivers-siemens-energys-first-3d-virtual-maintenance-trainer.html>. [Último acceso: 8 Enero 2026].
- [14] N. Babich, «UX Design Principles for Augmented Reality, Adobe,» 13 Enero 2020. [En línea]. Available: <https://xd.adobe.com/ideas/principles/emerging-technology/ux-design-principles-for-augmented-reality/>. [Último acceso: 15 Enero 2026].
- [15] Unity Learn, «Extended Reality,» 2023. [En línea]. Available: <https://learn.unity.com/search?k=%5B%22q%3AXR%22%5D>. [Último acceso: 10 Enero 2026].
- [16] Unity 3D, «Unity Documentation,» [En línea]. Available: <https://docs.unity3d.com/6000.3/Documentation/Manual/>. [Último acceso: 15 Marzo 2026].
- [17] OpenXR, «OpenXR Tutorial,» [En línea]. Available: <https://www.openxr-tutorial.com/windows/opengl/1-introduction.html>. [Último acceso: 26 Enero 2026].
- [18] Meta, «Meta Quest 3: Product Overview,» 2023. [En línea]. Available: <https://www.meta.com/es/quest/quest-3/>. [Último acceso: 20 Enero 2026].
- [19] F. Sabry, Tecnología háptica: Mejorar la interacción mediante retroalimentación sensorial., Mil Millones de Conocimientos [Spanish]., 2025.
- [20] «Protube VR - Haptics Accessories,» [En línea]. Available: <https://www.protubevr.com/en/>. [Último acceso: 30 Enero 2026].
- [21] «bHaptics - Shop,» [En línea]. Available: <https://www.bhaptics.com/tactsuit/tactsuit-pro/>. [Último acceso: 20 Febrero 2025].
- [22] KAT VR, «Kat VR C2 Plus Enhanced,» [En línea]. Available: <https://www.kat-vr.com/products/kat-walk-c-2-plus>. [Último acceso: 10 Enero 2026].
- [23] K. S. Hale y K. M. Stanney, Handbook of Virtual Environments: Design, Implementation, and Applications (2.^a edición)., CRC Press, 2014.
- [24] US Naval Institute, «Fleet Commanders View Innovation as a Challenge to Operate Smarter, 2017.,» [En línea]. Available: <https://news.usni.org/>. [Último acceso: 15 Enero 2026].
- [25] Nato Science And Technology Organisation, «What Is Essential for Virtual Reality Systems to Meet Military Human Performance Goals?,» 8 Julio 2023. [En línea]. Available: <https://www.sto.nato.int>. [Último acceso: 25 Enero 2026].
- [26] Meta for Developers, «Hello World,» [En línea]. Available: <https://developers.meta.com/horizon/documentation/unity/unity-tutorial-hello-vr/>. [Último acceso: 25 Enero 2026].
- [27] Unity Manual, «Unity 3D,» [En línea]. Available: <https://docs.unity3d.com/6000.0/Documentation/Manual/UnityManual.html>. [Último acceso: 16 Enero 2026].
- [28] GitHub, «TFG-Garófano, Repositorio de códigos,» [En línea]. Available: <https://github.com/juligarofano/TFG-Garofano>.

- [29] «bHaptics Developer,» [En línea]. Available: <https://developer.bhaptics.com/applications>. [Último acceso: 20 Febrero 2026].
- [30] «bHaptics Designer,» [En línea]. Available: <https://designer.bhaptics.com/pattern/>. [Último acceso: 20 Febrero 2025].
- [31] Unity Asset Store, «TV LED 30",» [En línea]. Available: <https://assetstore.unity.com/packages/3d/props/electronics/tv-led-30-336056>. [Último acceso: 20 Febrero 2026].
- [32] Unity Asset Store, «Paquete Basketball Realistic,» [En línea]. Available: <https://assetstore.unity.com/packages/3d/props/basketball-realistic-313677>. [Último acceso: 5 Febrero 2026].
- [33] Unity Asset Store, «Basketball Ring,» [En línea]. Available: <https://assetstore.unity.com/packages/3d/props/basketballring-297800#description>. [Último acceso: 3 Febrero 2026].
- [34] GitHub, «ForceTubeVR Unity Package,» [En línea]. Available: <https://github.com/ProTubeVR/ForceTubeVR-Unity-Package/tree/master/new/Assets>. [Último acceso: 6 Marzo 2026].
- [35] Unity Asset Store, «Archery Target,» [En línea]. Available: <https://assetstore.unity.com/packages/3d/props/archery-target-319592>. [Último acceso: 9 Marzo 2026].
- [36] «Mermaid.ai,» [En línea]. Available: <https://mermaid.ai/web/>. [Último acceso: 15 Marzo 2026].

ANEXO I: IMPLICACIONES SOCIALES, ECONÓMICAS Y AMBIENTALES

El desarrollo de este proyecto no solo tiene interés desde el punto de vista técnico, sino que también plantea una serie de implicaciones sociales, económicas y ambientales que conviene tener en cuenta. Aunque se trata de un entorno didáctico de realidad mixta desarrollado en un contexto académico, su planteamiento permite ver cómo este tipo de tecnologías pueden tener aplicaciones útiles en ámbitos de aprendizaje, formación y familiarización con entornos inmersivos.

Desde el punto de vista social, el proyecto muestra el potencial de la realidad mixta como herramienta para acercar al usuario a nuevas formas de interacción con elementos digitales. A través de las distintas pruebas desarrolladas, el usuario no se limita a observar objetos virtuales, sino que participa de forma activa en tareas de manipulación, coordinación espacial, precisión de movimiento y respuesta a estímulos. Esto hace que la experiencia resulte más intuitiva, dinámica y motivadora que otros métodos más tradicionales. Además, la incorporación de dispositivos hápticos refuerza la sensación de inmersión y mejora la relación entre la acción realizada y la respuesta percibida. Aun así, también debe tenerse en cuenta que el acceso a este tipo de experiencias depende de disponer de hardware específico, lo que puede dificultar su uso generalizado.

En el plano económico, una de las principales ventajas de este tipo de desarrollos es la posibilidad de crear entornos reutilizables y ampliables sin necesidad de construir un montaje físico distinto para cada nueva actividad. Una vez creada la base del sistema, es posible añadir nuevas escenas, mecánicas o dispositivos con un coste menor que si hubiera que desarrollar cada experiencia desde cero. Esto hace que la realidad mixta sea una opción interesante de cara a futuros usos didácticos o de entrenamiento. Sin embargo, también hay que considerar que el uso de gafas de realidad mixta, periféricos hápticos y otros accesorios supone una inversión económica que puede limitar su implantación en determinados contextos.

En cuanto al aspecto ambiental, el uso de entornos virtuales y mixtos puede reducir la necesidad de fabricar grandes instalaciones para la creación de escenarios virtuales, como es el caso de los simuladores de vuelo o de navegación, con todo el impacto que supone la construcción y el empleo del mismo; con esta tecnología, desaparece la necesidad de construir un escenario de simulación, el mismo escenario real se puede emplear como zona de uso. No obstante, este beneficio no elimina completamente el impacto ambiental, ya que el funcionamiento del sistema depende de dispositivos electrónicos y consumo energético. Por ello, resulta importante que este tipo de proyectos se planteen de forma reutilizable y reduccionista en el número y dimensión de los recursos materiales que se empleen para la utilización de esta tecnología.

Por lo tanto, puede decirse que este proyecto presenta unas implicaciones mayoritariamente positivas. Desde el punto de vista social, aporta una nueva forma atractiva e interactiva de aprendizaje tecnológico; desde el económico, demuestra la utilidad de construir una base flexible sobre la que seguir desarrollando nuevas experiencias; y desde el ambiental, ofrece ventajas derivadas de la reducción de ciertos recursos físicos gracias al uso del entorno digital. Todo ello refuerza el valor del proyecto más allá de su dimensión puramente técnica.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Aunque el objetivo principal del trabajo ha sido la creación de un entorno interactivo orientado al aprendizaje y la familiarización con tecnologías inmersivas, el uso de este tipo de sistemas lleva asociado un impacto que va más allá de su funcionamiento técnico.

Desde el punto de vista social es importante tener en cuenta que el acceso a estas tecnologías no es igual para todo el mundo. El uso de dispositivos como gafas de realidad mixta, periféricos físicos o sistemas hápticos depende de contar con unos recursos materiales concretos, lo que puede generar diferencias en su disponibilidad. Por tanto, aunque se trate de una tecnología con muchas posibilidades, su aplicación real puede verse condicionada por factores económicos y de acceso.

Otro aspecto relacionado con este tipo de desarrollos es el papel que ocupa la tecnología dentro de la actividad humana. La realidad mixta no debe entenderse como un sustituto del aprendizaje convencional o de la experiencia real, sino como una herramienta complementaria que puede enriquecer determinados procesos. En este proyecto, esa idea está presente en el propio planteamiento del sistema, ya que se busca ofrecer una base de familiarización con entornos inmersivos y con nuevas formas de interacción, pero sin perder de vista que se trata de una herramienta de apoyo.

En conclusión, El proyecto propone un uso responsable de la realidad mixta, orientado al aprendizaje, la experimentación y la mejora de la interacción con entornos digitales. Al mismo tiempo, pone de manifiesto la necesidad de seguir desarrollando estas tecnologías de forma accesible, controlada y centrada en el usuario, para que su evolución futura resulte beneficiosa tanto desde el punto de vista técnico como desde el humano.

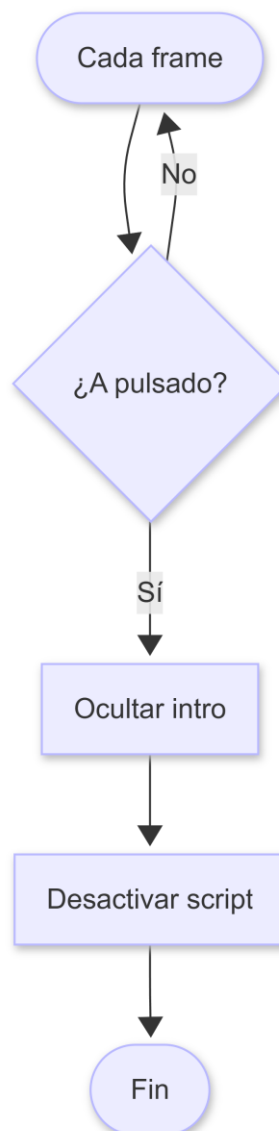
ANEXO III: DIAGRAMAS DE FLUJO DE LOS CÓDIGOS DEL PROYECTO

En este anexo se presentan los diagramas de flujo correspondientes a los scripts desarrollados para el proyecto. El objetivo es ofrecer una visión clara y rápida de la lógica de funcionamiento de cada componente sin necesidad de entrar directamente en el código fuente. Para cada código se muestra el flujo principal de ejecución, indicando cómo se gestionan eventos como la interacción del usuario, la detección de colisiones o el cambio de estados dentro de la escena. Estos diagramas complementan la explicación del desarrollo descrita en los apartados anteriores y sirven como referencia visual para comprender la estructura general del sistema y la relación entre sus distintas funcionalidades.

Todos los códigos presentados se encuentran almacenados y accesibles desde internet en un repositorio de códigos en *GitHub* [28]. Para la realización de los diagramas, se ha recurrido al programa de inteligencia artificial Mermaid.ai [36].

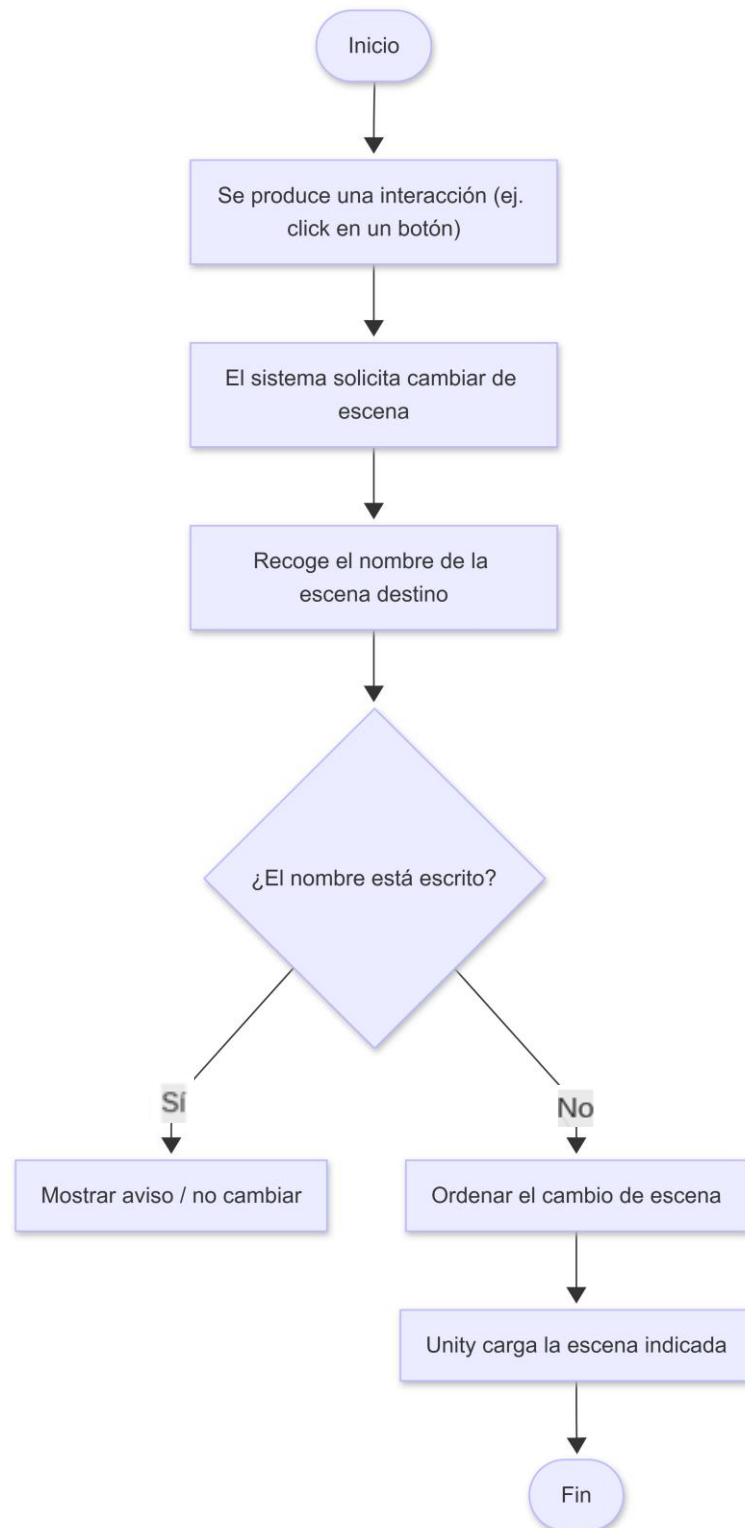
- *HideIntroOnA*

Este flujo se encarga de que el panel de introducción desaparezca cuando el usuario pulsa A. Mientras no se pulse, no ocurre nada y el sistema sigue esperando. En cuanto se pulsa el botón, se oculta la introducción y la comprobación deja de hacerse porque ya no es necesaria.



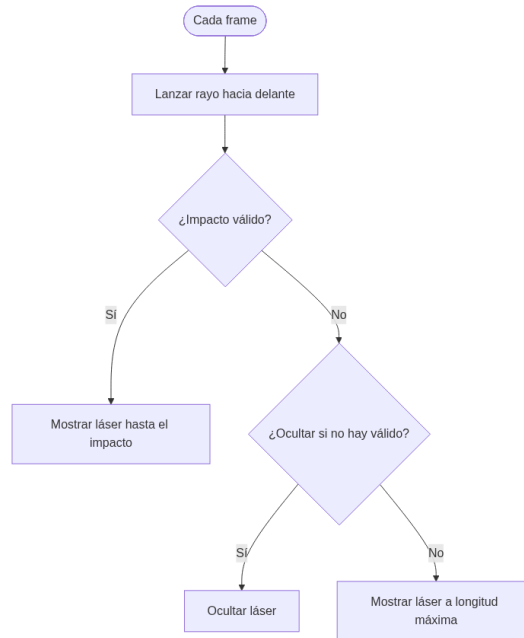
- *SceneLoader*

Permite cambiar de escena llamando a un único método público (*LoadSceneByName*). Se emplea en los botones de la escena inicial para saltar entre escenas.



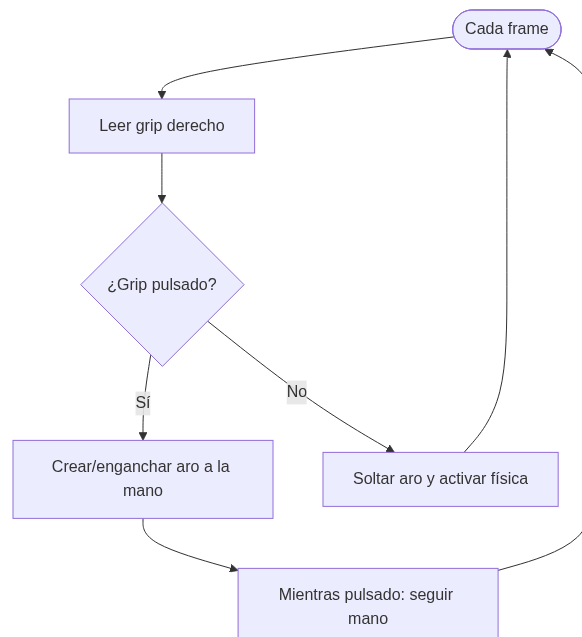
- *SimpleLaser*

Este código genera un láser visual que solo se muestra cuando el rayo detecta un impacto considerado válido. La validez puede filtrarse por *collider* específico, por *tag*, o desactivando filtros para aceptar cualquier impacto dentro de la máscara de capas. Además, la posición final del láser se suaviza para evitar “saltos” bruscos al mover el mando o apuntar a la interfaz.



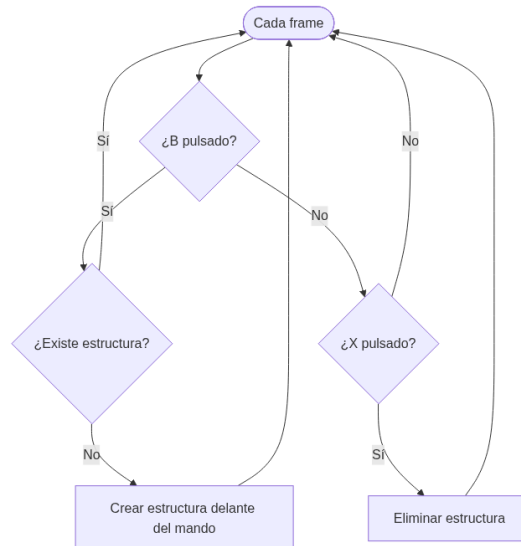
- *RightHandSpawner*

Este código permite generar y sostener un *prefab* en la mano derecha usando el *grip*. Para evitar que el *prefab* aparezca nada más iniciar la escena si el usuario ya tiene el botón pulsado, se incluye un sistema de armado de entrada, que no habilita la interacción hasta detectar el *grip* suelto al menos una vez. Al pulsar *grip*, el aro se crea (o se reutiliza si está activado solo uno), se coloca en la mano y se desactiva la física para que no interfiera con el seguimiento. Al soltar *grip*, el *prefab* se libera y recupera la física para que pueda caer.



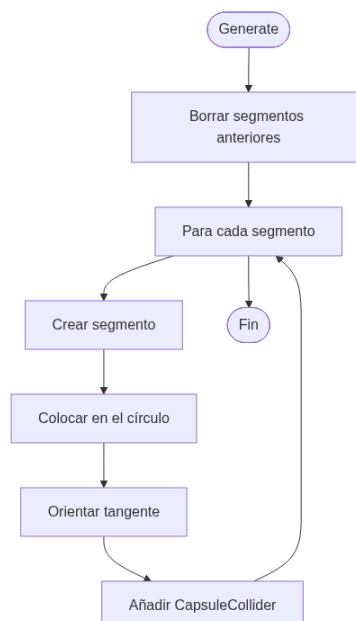
- *SpawnEstructuraOnB_NoFloor*

Este código permite generar y eliminar la estructura desde el mando derecho. Cuando el usuario pulsa B, y no estaba pulsado ya, se crea una única instancia de la estructura delante del mando (si no existe ya). Cuando pulsa X, la estructura se elimina. La posición se calcula a una distancia fija hacia delante y, opcionalmente, la rotación se alinea con la orientación horizontal (*yaw*) del mando para que la estructura aparezca correctamente orientada en la escena.



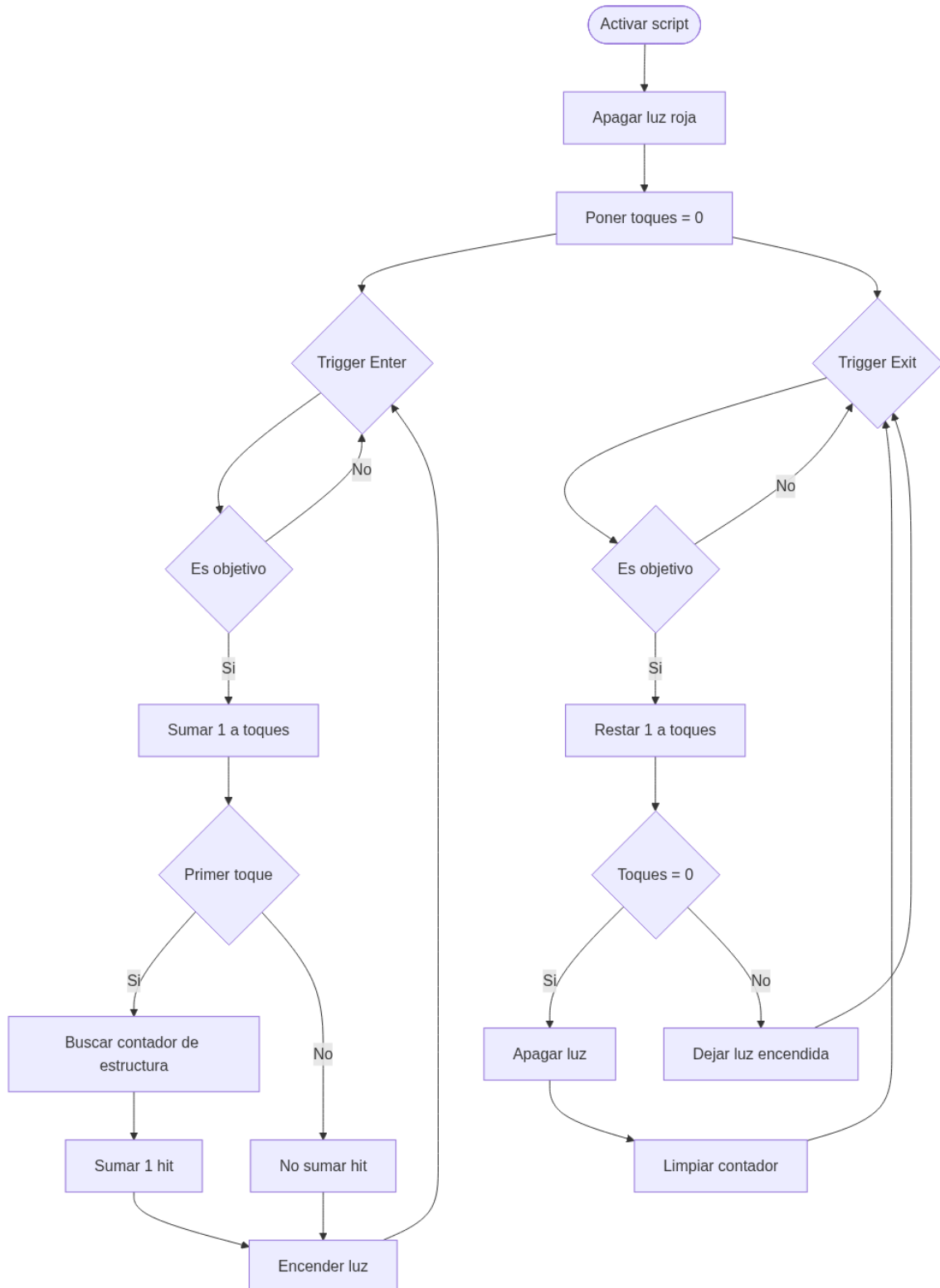
- *RingColliderGenerator*

Este código genera automáticamente los *colliders* del aro creando varios segmentos alrededor de un círculo. Primero elimina los segmentos anteriores (hijos del objeto) y después, para cada segmento, crea un *GameObject*, lo coloca en la posición correspondiente del aro, lo orienta de forma tangente y le añade un *CapsuleCollider* con el grosor y la longitud configurados. El número de segmentos, al igual que las dimensiones del círculo, se definen en el inspector. De esta forma el aro queda con una colisión uniforme sin tener que ajustar *colliders* a mano.



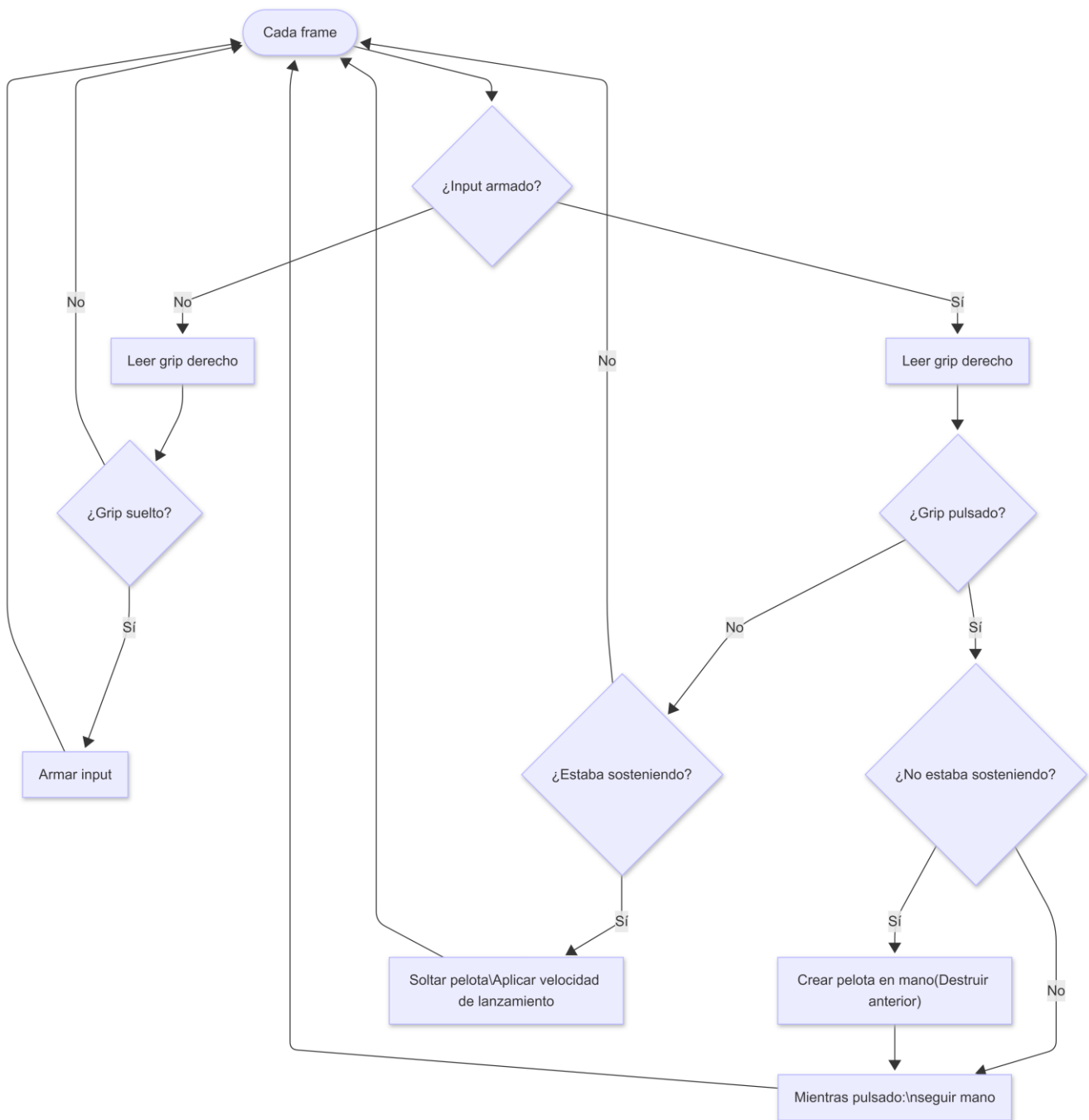
- *RingCollisionDetector*

Este código detecta cuándo el aro entra o sale de contacto con los triggers de la estructura. Mientras haya contacto, se enciende una luz roja en el aro; cuando ya no hay ningún contacto, se apaga. Además, solo cuando empieza el contacto (pasa de 0 a 1), se busca el contador de la estructura y se incrementa una colisión para evitar doble conteo.



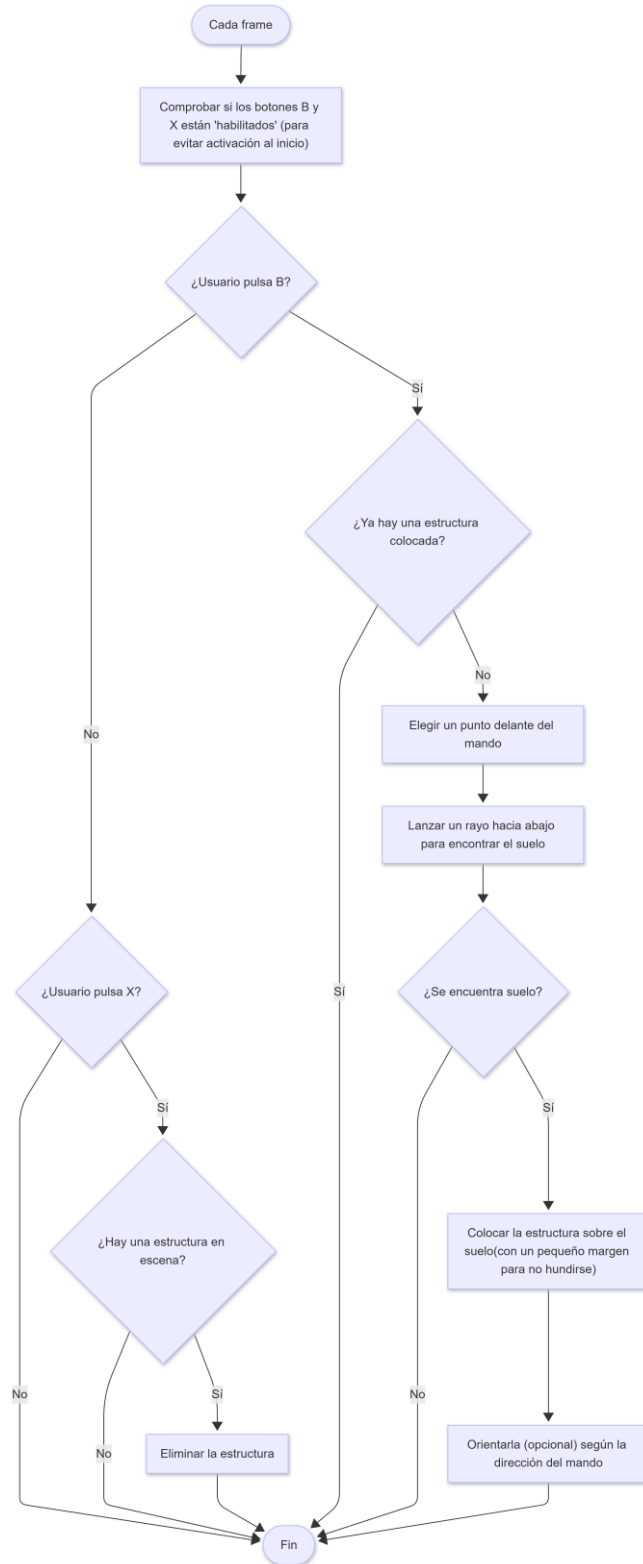
- *BallThrownManager*

Este código permite generar y lanzar una pelota desde la mano derecha usando el *grip* del mando derecho. Para evitar que se cree una pelota nada más iniciar la escena si el usuario ya tenía el botón pulsado, se incluye un sistema de “armado” de entrada que no habilita la interacción hasta detectar el *grip* suelto al menos una vez. Al pulsar el *grip*, se instancia una pelota en la mano (y se destruye la anterior, incluso si estaba en vuelo) y se desactiva la física para que siga correctamente el movimiento del mando. Al soltar el *grip*, la pelota se libera, recupera la física y se lanza aplicando la velocidad real del controlador, ajustada por un multiplicador y un empuje extra opcional hacia delante.



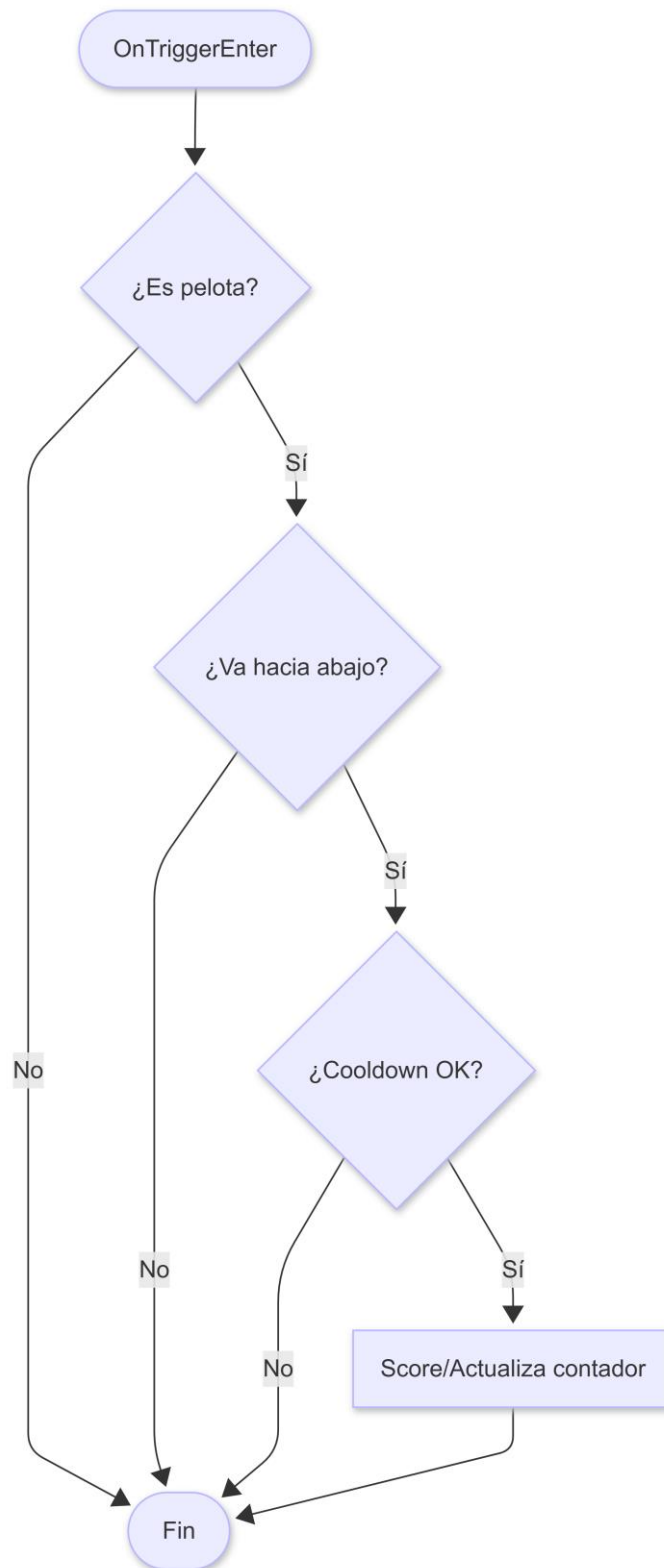
- *SpawnEnSuelo*

Este código permite colocar una estructura en el suelo usando el mando derecho como referencia. Al pulsar B se genera la estructura delante del mando y se sitúa en el suelo mediante un rayo hacia abajo para encontrar el plano de suelo; al pulsar X se elimina. Incluye un armado de botones para evitar que se ejecute nada al entrar en la escena si los botones ya estaban pulsados.



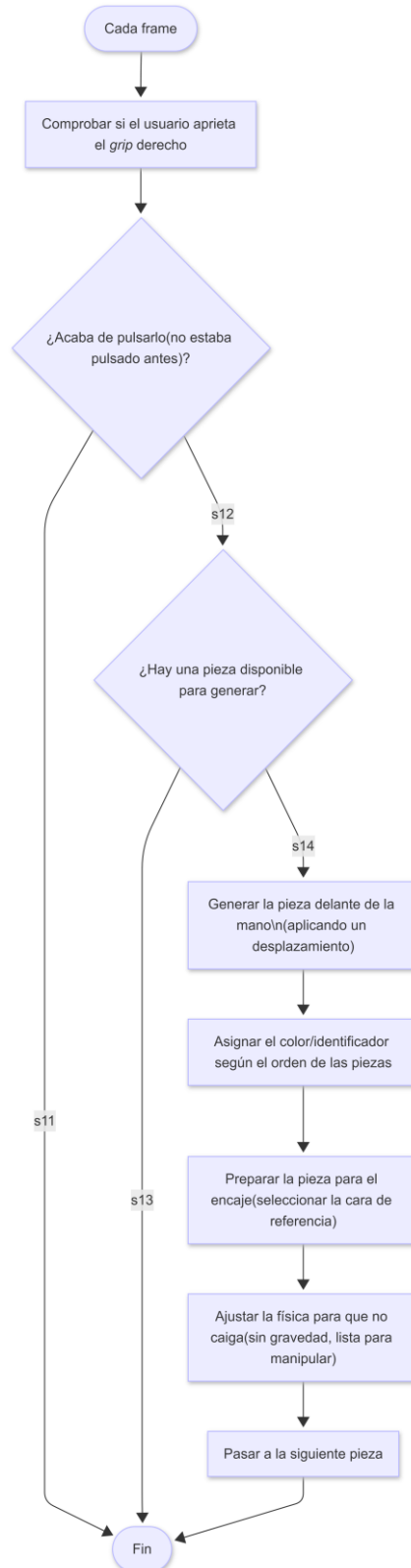
- HoopScoreZone

Este código detecta una canasta cuando la pelota entra en una zona *trigger*. Opcionalmente solo cuenta si la pelota va hacia abajo y evita el doble conteo con un *cooldown* por pelota. Al sumar, actualiza el texto de puntos.



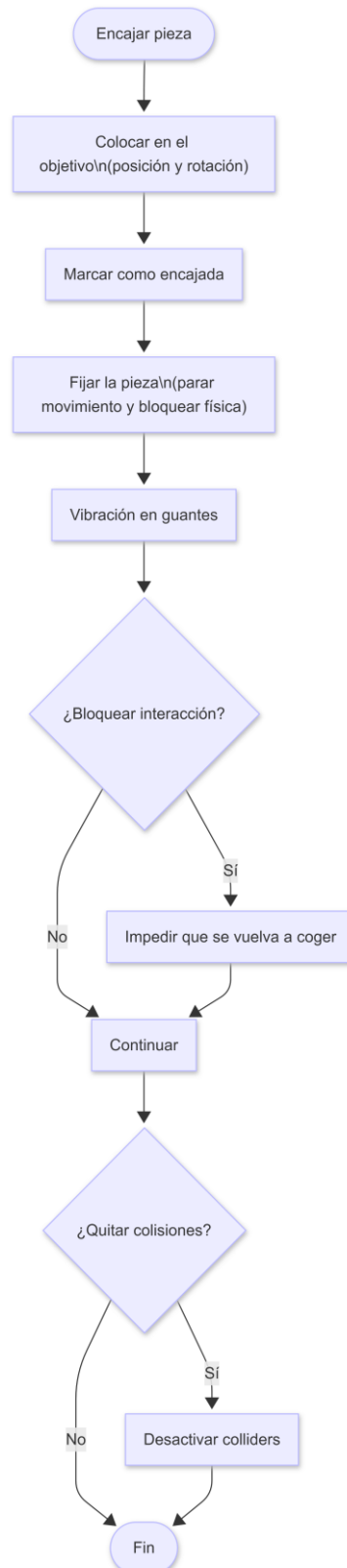
- *RightHandPieceSpawner_Free*

Este código genera piezas delante de la mano derecha al pulsar el *grip*, de forma similar al *RightHandSpawner*, detecta la transición de “no pulsado a pulsado” para crear el siguiente *prefab* (pieza del puzle) en orden. Cada pieza se instancia en el mundo con un *offset*, se le asigna un *pieceld*, se asegura que tenga *SnapPiece* y se prepara la física para que no se caiga (sin gravedad).



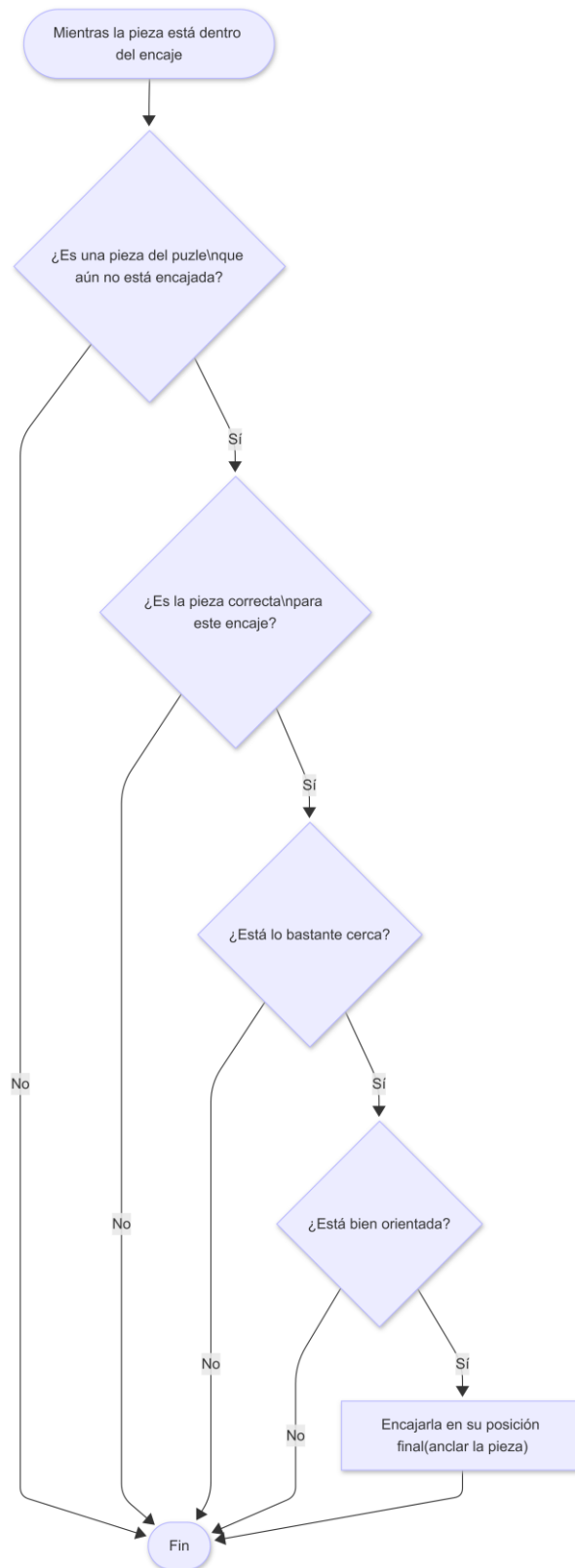
- *SnapPiece*

Este código marca una pieza como “encajada” cuando llega a su objetivo. Al encajarla, la coloca exactamente en la posición y orientación del destino, detiene su movimiento y la bloquea para que no se pueda volver a coger. Además, reproduce una vibración en *bHaptics* y lanza un evento para llamar a los demás códigos.



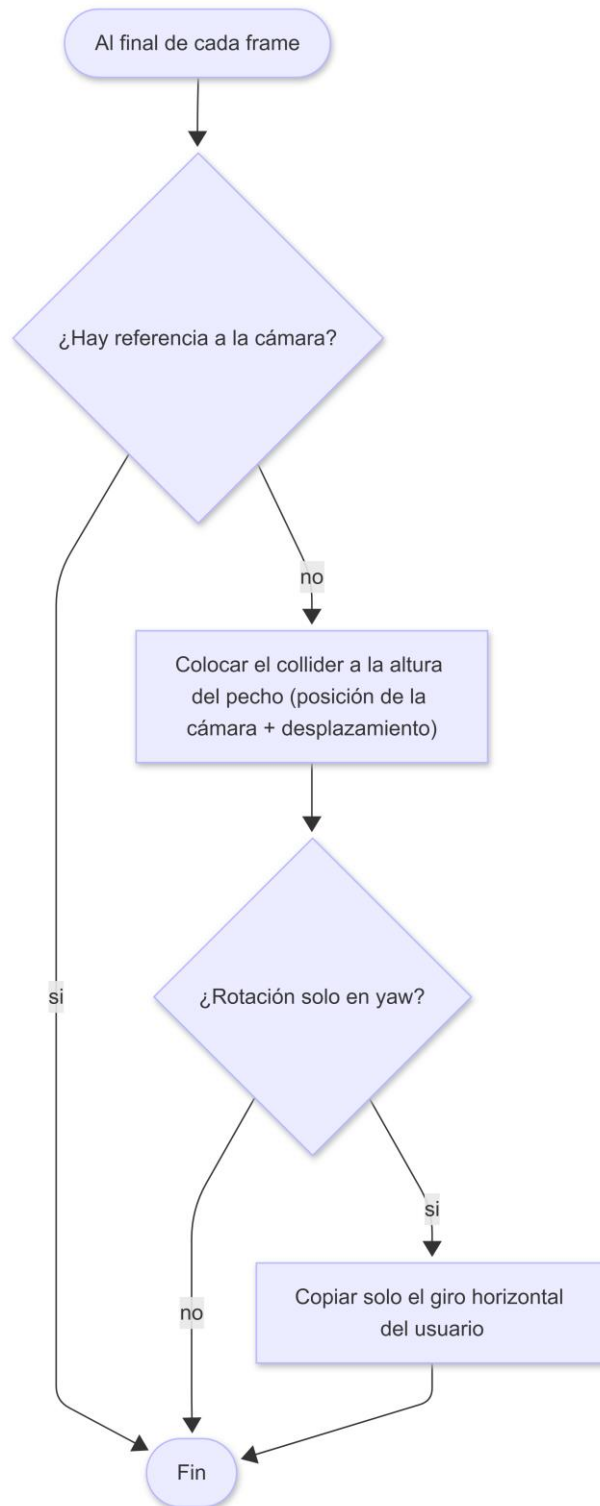
- *SnapSocket*

Este código define una zona de encaje para cada pieza del puzzle. Mientras la pieza esté dentro del *trigger*, comprueba que sea la pieza correcta (mismo identificador) y que esté lo bastante cerca y bien orientada. Si cumple esas condiciones, la ancla en su posición final (*snap*).



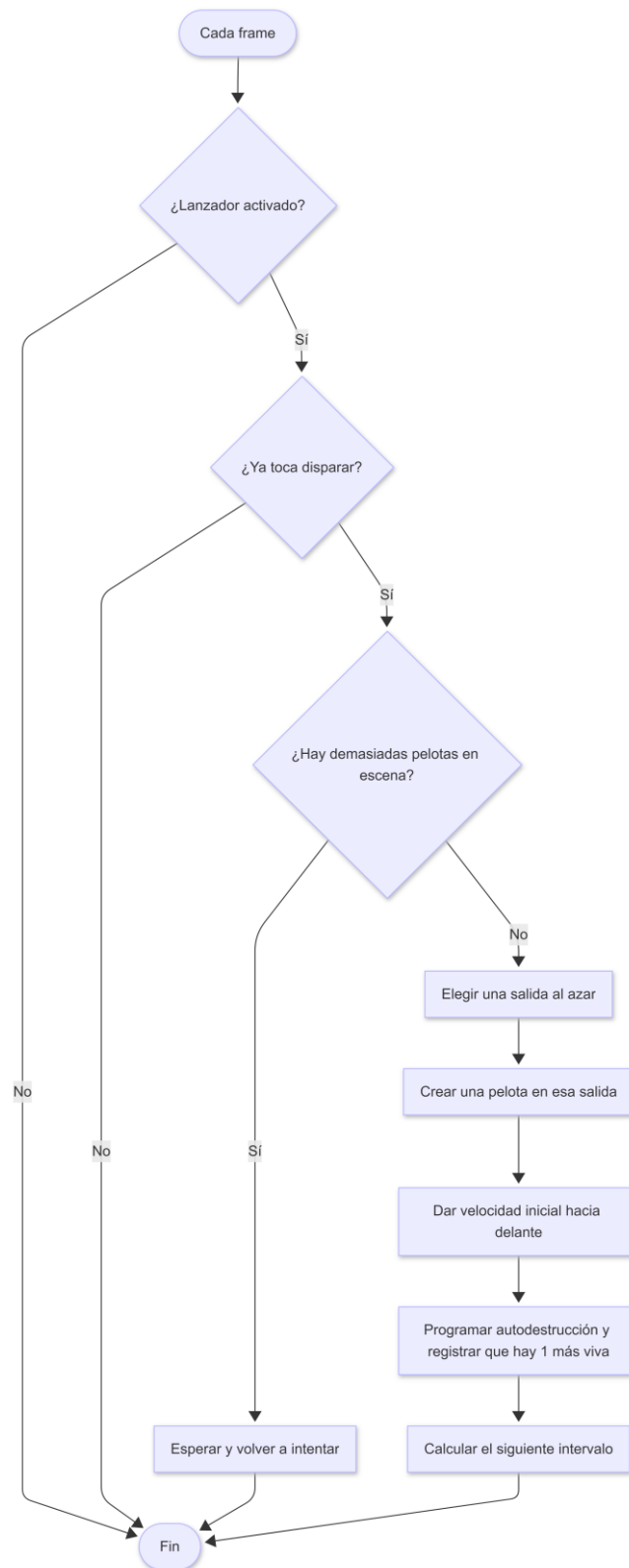
- *BodyColliderFollow*

Este código mueve el *collider* del cuerpo para que siga la posición del usuario. Toma como referencia la cámara (*CenterEyeAnchor*), aplica un desplazamiento vertical para colocarlo a la altura del pecho y, si se activa, solo copia la rotación en el eje vertical para que el *collider* gire con el usuario sin inclinarse.



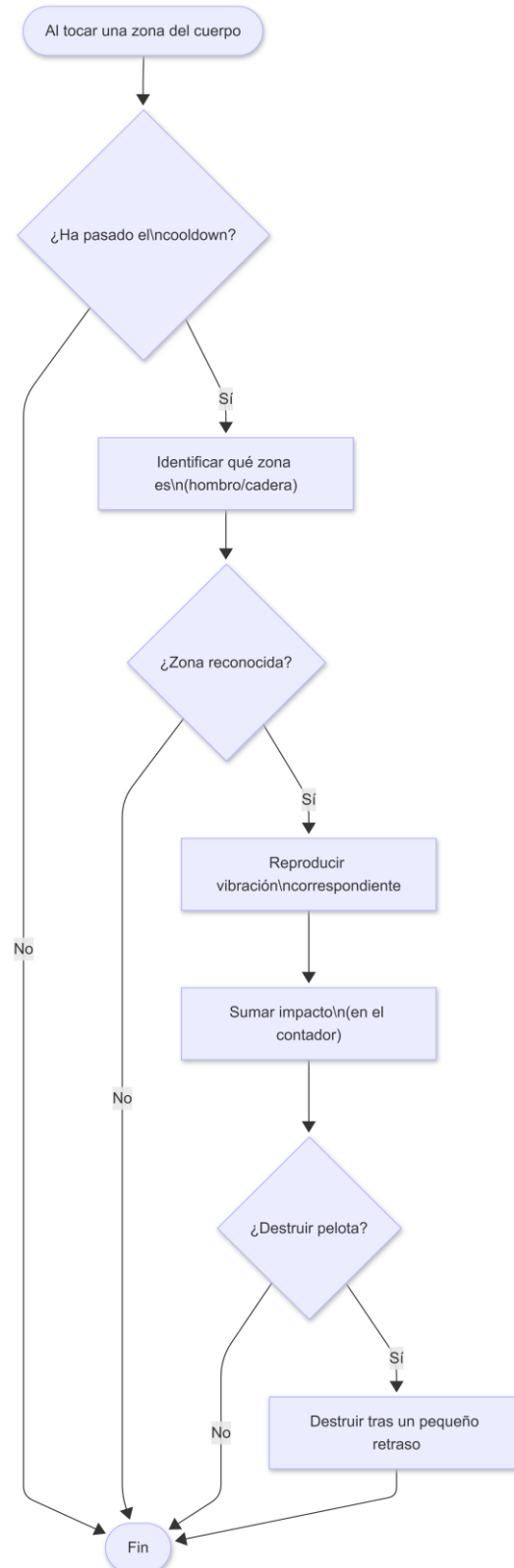
- *BallLauncher*

Este código controla el objeto que dispara pelotas desde varias salidas en la escena del Lanzador. Cuando está activa, va generando pelotas a intervalos fijos o aleatorios y limita cuántas pueden estar vivas a la vez. Cada pelota sale con una velocidad inicial y se destruye sola tras un tiempo; al destruirse, avisa al lanzador para actualizar el contador



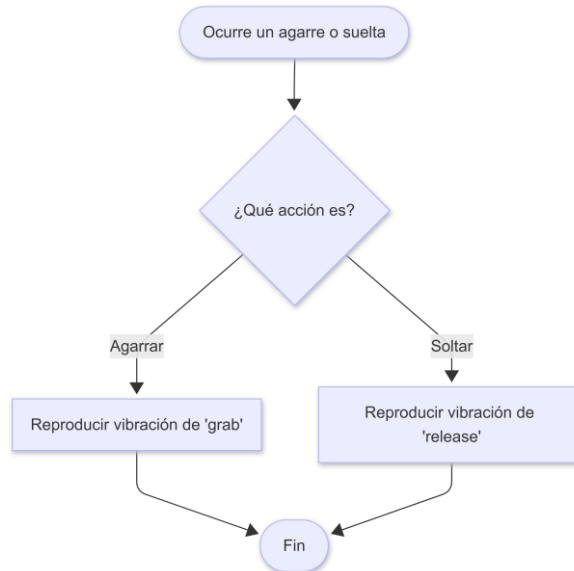
- *BallHitBodyZones*

Este código detecta cuando una pelota impacta con zonas del cuerpo (hombros y caderas) y reproduce la vibración correspondiente en chaleco háptico. Para evitar múltiples impactos seguidos, aplica un pequeño *cooldown*. Opcionalmente, puede destruir la pelota tras el golpe y suma el impacto en un contador sobre el objeto.



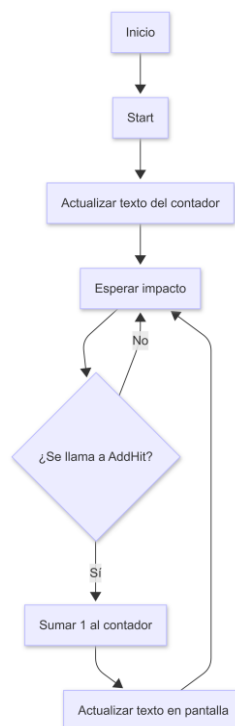
- *BhapticsEventPlayer*

Este script sirve para lanzar una vibración cuando el usuario agarra un objeto y otra cuando lo suelta. Si la acción es agarrar, reproduce el efecto asignado a “*grab*”; si la acción es soltar, reproduce el efecto asignado a “*release*”. Así se refuerza la interacción con una respuesta táctil inmediata.



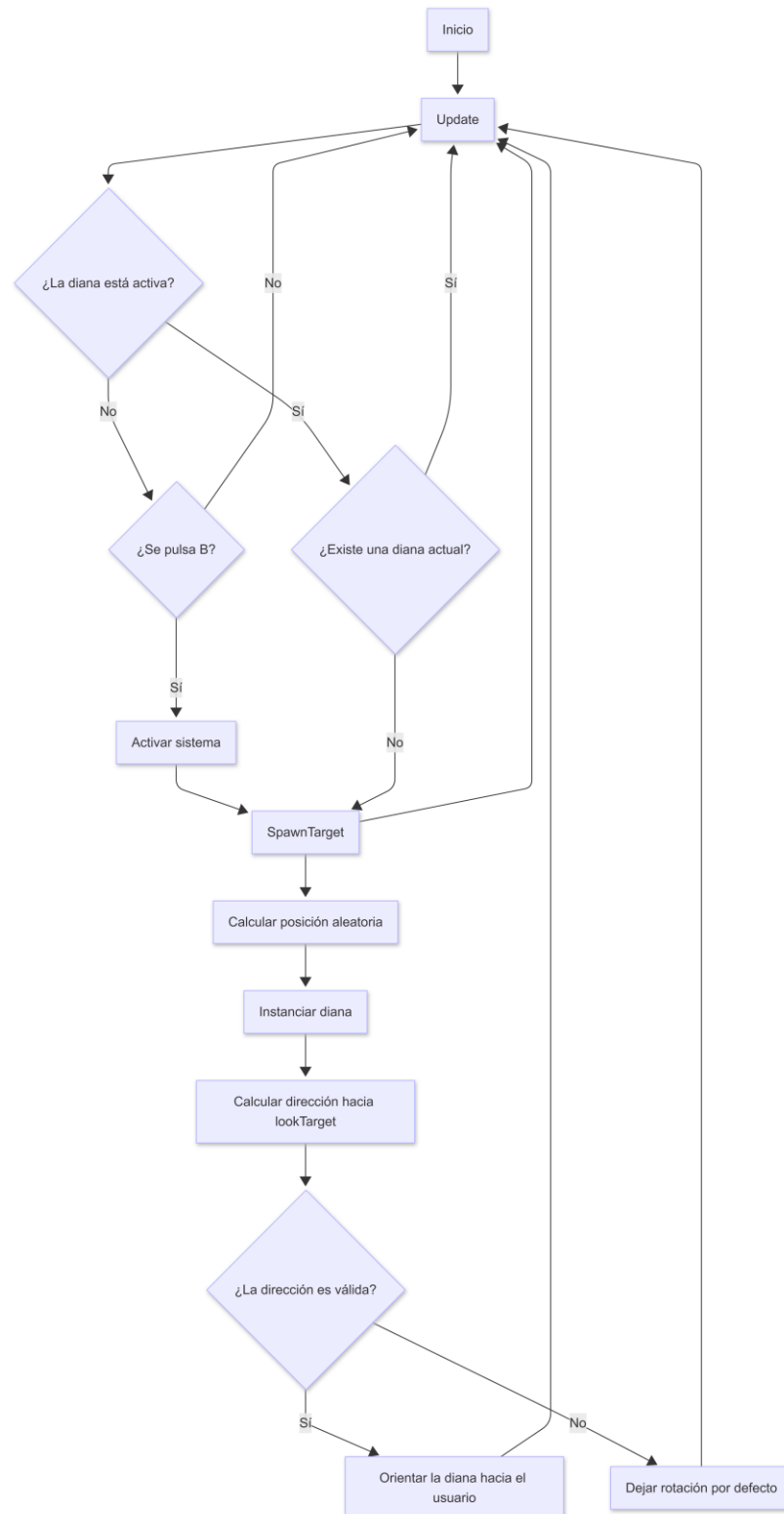
- *TargetCounter*

Este script se encarga de llevar la cuenta del número de dianas alcanzadas y mostrar ese valor en pantalla. Al iniciarse la escena, actualiza el texto para que el contador comience en cero. Cada vez que otro elemento del sistema llama al método AddHit, el valor almacenado aumenta en una unidad y el texto se vuelve a actualizar. De esta forma, el usuario puede ver en todo momento cuántas dianas ha conseguido acertar.



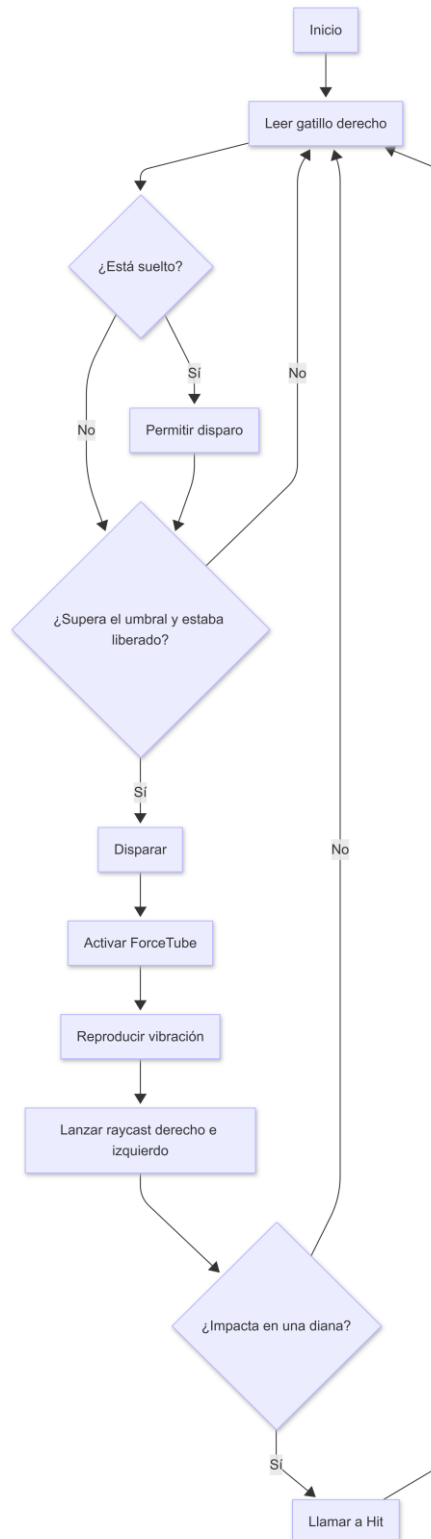
- *TargetSpawner*

Este código controla la aparición automática de una diana en la escena. Al principio, el sistema permanece inactivo hasta que el usuario pulsa el botón B. En ese momento, se genera una primera diana en una posición aleatoria dentro de unos límites de distancia, altura y ángulo definidos previamente. Una vez activado, comprueba constantemente si sigue habiendo una diana en la escena. Si la diana desaparece, se genera otra nueva de forma automática. Además, cada nueva diana se orienta hacia el usuario para que quede correctamente colocada de frente y resulte más natural durante la interacción.



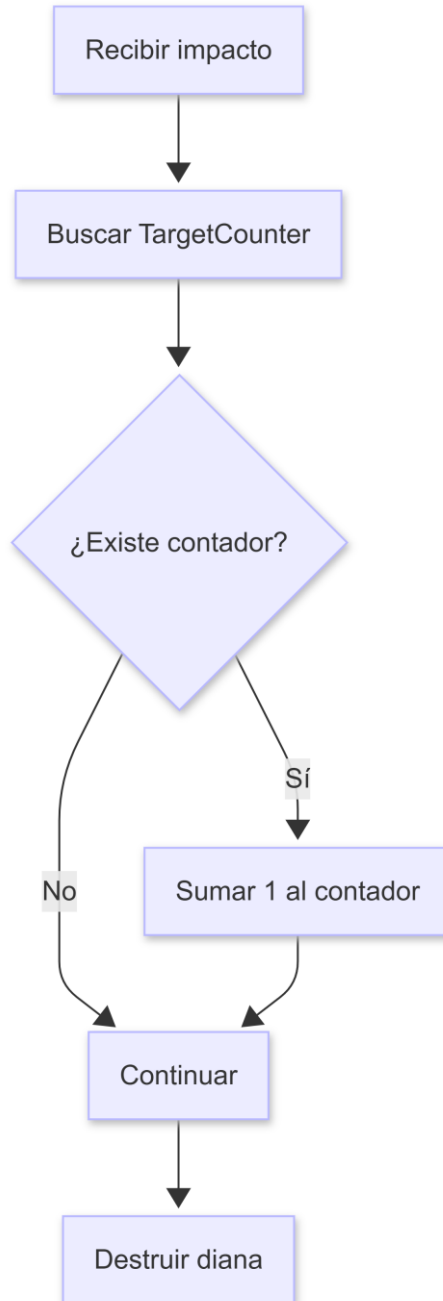
- *ProtubeShooter*

Este código detecta la pulsación del gatillo derecho y, cuando se supera el umbral marcado, ejecuta un disparo. Antes de permitir otro, obliga a soltar el gatillo, evitando así varios disparos seguidos con una sola pulsación. Al disparar, se activa primero el retroceso del periférico de *ProTubeVR* que esté conectado y, además, se reproduce la vibración háptica asociada en el chaleco. A continuación, se lanzan dos *raycast* hacia delante, uno desde el punto de disparo derecho y otro desde el izquierdo. Si alguno de ellos impacta sobre una diana válida, se llama al método Hit del código *TargetBehaviour*-



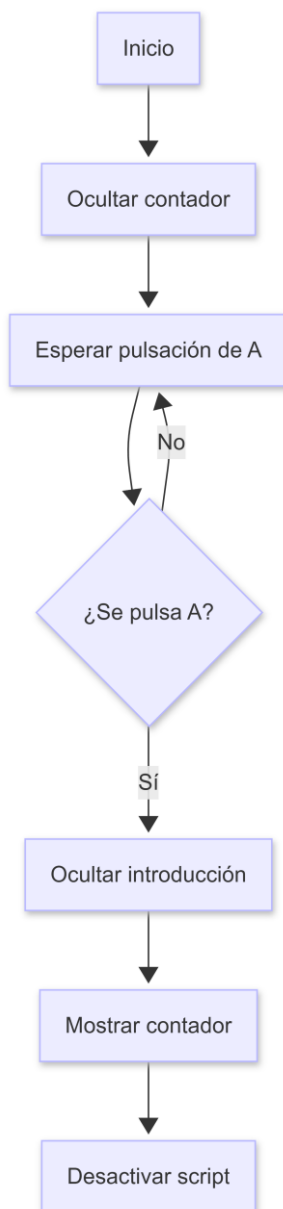
- *TargetBehaviour*

Este código define el comportamiento de la diana cuando recibe un disparo. Al producirse el impacto, busca el contador de dianas de la escena y, si lo encuentra, incrementa su valor en una unidad. Después de actualizar el contador, la diana se destruye para desaparecer de la escena. Esto permite que el sistema genere una nueva automáticamente si así está configurado.



- *HideIntroOnA2*

Este script espera a que el usuario pulse el botón A para iniciar la prueba. Cuando eso ocurre, oculta el panel introductorio, muestra el contador de dianas y deja de ejecutarse, porque ya no necesita seguir comprobando nada.



ANEXO IV: DECLARACIÓN DEL USO DE INTELIGENCIA ARTIFICIAL

Durante la realización de este Trabajo Fin de Grado se ha hecho un uso puntual de herramientas de inteligencia artificial como apoyo en determinadas tareas. En concreto, se han utilizado principalmente en aspectos relacionados con la programación, como ayuda para la elaboración, revisión o ajuste de algunos fragmentos de código, y de forma ocasional para mejorar la redacción de ciertos apartados del documento.

En cualquier caso, este uso ha sido siempre complementario y no ha sustituido el trabajo personal realizado en el proyecto. Tanto el planteamiento general como el desarrollo de las escenas, la implementación de las pruebas, la interpretación de los resultados y la redacción final del trabajo han sido llevados a cabo por el autor. Las aportaciones obtenidas mediante estas herramientas han sido revisadas, adaptadas y utilizadas únicamente cuando contribuían a mejorar la calidad final del trabajo.