Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Diseño de escenario bélico para simulación táctica de combate mediante realidad virtual

Grado en Ingeniería Mecánica

ALUMNO: Álvarez Díaz, Carlos

DIRECTORES: Núñez Nieto, Xavier

Rodelgo Lacruz, Miguel

CURSO ACADÉMICO: 2021-2022

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Diseño de escenario bélico para simulación táctica de combate mediante realidad virtual

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

La llegada de la Industria de Cuarta Generación (i4.0) a nuestras Fuerzas Armadas ha consolidado un dominio híbrido, que entremezcla cada vez más el mundo real con el digital. Dentro de este marco operativo, se cuenta con la presencia de técnicas y metodologías orientadas a la simulación virtual para el adiestramiento militar, que se ve reflejada en múltiples proyectos de gamificación. Estas nuevas metodologías pueden suponer una revolución en los procesos de enseñanza de los tres ejércitos: Armada, Tierra y Aire. Este trabajo hace uso de dicha tecnología para desarrollar una plataforma operativa de adiestramiento táctico. Sirviéndose de diferentes técnicas de cuarta generación, se ha modelado un escenario bélico fotorrealista, cuyo objetivo busca poner en práctica una situación de combate. A través de la programación de Inteligencia Artificial y del uso de la Realidad Virtual, el simulador plantea una experiencia realista/inmersiva de la que se puede extraer un análisis individualizado de cada usuario, para su posterior seguimiento y evaluación.

PALABRAS CLAVE

Realidad Virtual, Blueprints, Inteligencia Artificial, Simulación de combate, Gamificación.

AGRADECIMIENTOS

.

Antes de nada, ofrezco mi agradecimiento al profesor Don Xavier Núñez Nieto, tutor del presente Trabajo de Fin de Grado, por contagiarme su característica e insaciable sed de mejora y al profesor Don Miguel Rodelgo Lacruz por mantener en todo momento una férrea confianza en mí.

También agradezco a mis compañeros de estudio, camareta y batallas, los Alféreces de Fragata Alegre, Aycart y Arcila, por ser siempre una excelente vía de escape ante las dificultades. Lo hemos logrado juntos.

Por último, no puedo dejar escapar la oportunidad de mostrar mi profundo agradecimiento a toda mi familia. En especial a mis padres, los auténticos conocedores de mis ambiciones y anhelos por convertirme en un Oficial de la Armada Española. A mi padre, por ser mi ejemplo y referente de esfuerzo y sacrificio diario. A mi madre por su leal ternura y su apoyo incondicional desde que mi sueño empezó a resonar por las paredes de nuestra casa.

CONTENIDO

Contenido 1
Índice de Figuras 5
1 Introducción 9
1.1 Descripción del apartado 9
1.2 Antecedentes 9
1.3 Justificación 11
1.4 Objetivos 12
1.5 Estructura 13
2 Estado del arte 15
2.1 Descripción del apartado 15
2.2 Del ámbito militar 15
2.2.1 La Industria 4.0 en Defensa 15
2.2.2 Alta tecnología en las FAS 17
2.2.3 Simuladores, la herramienta definitiva para el adiestramiento 19
2.2.4 Combate en población 21
2.2.4.1 Definición 21
2.2.4.2 Reglas de enfrentamiento 21
2.2.4.3 Tipos de acciones tácticas/ofensivas 22
2.2.5 El ataque 22
2.2.5.1 Ataque inmediato 22
2.2.5.2 Ataque premeditado 23
2.2.6 Fases del ataque 24
2.2.6.1 Asalto a los objetivos intermedios 24
2.2.6.2 Progresión y limpieza 24
2.3 Del ámbito civil 25
2.3.1 Experiencias inmersivas 25
2.3.2 La tecnología de la Realidad Virtual 26
2.3.2.1 Evolución de la RV 27
2.3.2.2 Introducción al concepto de Realidad Aumentada 28
2.3.2.3 Causas y tratamientos del “motionsickness” 29
2.3.2.4 Aplicaciones de la RV 29
2.3.3 Inteligencia artificial 29
2.3.4 Software 30
2.3.4.1 Epic Store 31

2.3.4.2 Unreal Engine v4.27.2 31
2.3.4.3 Codificación mediante Blueprints 32
2.3.4.4 Steam VR 32
2.3.4.5 Autodesk Inventor 33
2.3.4.6 Autodesk 3DS Max 33
2.3.5 Equipos 34
2.3.5.1 NVIDIA 2080 Super 34
2.3.5.2 HTC VIVE PRO 35
2.3.5.3 Adaptador Beswin VR M4 38
2.3.6 Recorrido histórico: juegos de temática militar 38
3 Desarrollo 43
3.1 Descripción del apartado 43
3.2 Diseño y modelado del escenario 44
3.2.1 Creación del proyecto 44
3.2.1.1 Selección de Categoría y Template inicial 44
3.2.1.2 Configuración inicial del proyecto 45
3.2.1.3 Interfaz del editor de niveles de Unreal Engine 45
3.2.2 Estudio previo del terreno 47
3.2.3 Diseño del terreno 48
3.2.3.1 Diseño y construcción de la casa objetivo 52
3.2.4 Diseño del aspecto sonoro 54
3.2.5 Diseño del fusil HK G36 55
3.2.6 Diseño e implementación de la mira telescópica 56
3.3 Programación en Unreal Engine 59
3.3.1 Programación del sistema de movimiento 59
3.3.1.1 Planificación de los sistemas de movimiento 59
3.3.1.2 Movimiento mediante headtracking 61
3.3.1.3 Movimiento mediante trackpad 62
3.3.1.4 Solución al problema de colisiones 64
3.3.2 Programación del fusil 65
3.3.2.1 Integración del fusil con el entorno 65
3.3.2.2 Interacción del fusil con el controlador 65
3.3.2.3 Comportamiento del fusil 71
3.3.3 Programación de la Inteligencia Artificial 73
3.3.3.1 Árbol de comportamiento 74
3.3.3.2 Controlador de la IA 78
3.3.3.3 Mecánicas del enemigo 79

3.3.4 Diseño del nivel 81
3.3.4.1 Widgets en RV 81
3.3.4.2 Temporizador: su uso como game manager 84
3.3.4.3 Análisis de datos 85
3.3.4.1 Leaderboard o tabla de clasificación 87
3.3.4.2 Nivel con modo visión nocturna 90
3.4 Realidad Virtual 91
3.4.1 Preparación del headset 91
3.4.2 Simulaciones en RV desde Unreal Engine 92
3.4.3 Inclusión del modo con adaptador Beswin VR M4 92
4 Resultados 95
4.1 Descripción del apartado 95
4.2 Escenario bélico generado 95
4.2.1 Personajes 98
4.2.1.1 Avatar principal 98
4.2.1.2 Enemigos 98
4.2.1.3 Rehenes 99
4.3 Resultados de la programación 100
4.4 Simulación mediante el equipo de RV 103
4.5 VR CAD Simulator: ejecutable definitivo 104
4.6 Vídeo demostración 105
5 Conclusiones y líneas futuras 107
5.1 Descripción del apartado 107
5.2 Conclusiones 107
5.2.1 Conclusiones previas 107
5.2.1.1 Diseño y modelado del escenario bélico 107
5.2.1.2 Programación 107
5.2.1.3 La Realidad Virtual como herramienta 108
5.2.2 Conclusión final 108
5.3 Líneas futuras 109
5.3.1 A corto plazo 109
5.3.2 A medio-largo plazo 109
6 Bibliografía 111
Anexo I: Bloques de código: Tareas del árbol de comportamiento de la IA 115
Anexo II: Bloques de código: Inteligencia Artificial 119

Anexo III: Bloque de código: Disparo 128
Anexo IV: Adaptador Beswin VR M4 HTC VIVE 129

ÍNDICE DE FIGURAS

Figura 1-1 Herramientas de la Armada 4.0 10
Figura 1-2 Infantes de marina adiestrándose (@Armada_esp) 11
Figura 2-1 Evolución histórica de la Industria y las tecnologías de cada etapa [3] 15
Figura 2-2 Las 11 áreas de I+D+i contempladas por la ETID [4]....16
Figura 2-3 Sistema de radio SDR para vehículos E-Lynx de Elbit systems [5] 17
Figura 2-4 Avenger, el robot del Ejército de Tierra frente a minas trampa [6]....17
Figura 2-5 Poniendo a prueba la aeronave no tripulada desarrollada por INDRA [7]....18
Figura 2-6 Exoesqueleto Lockheed Martin FORTIS para aliviar el estrés de la rodilla [9] 18
Figura 2-7 Simulador de vuelo Eurofighter desarrollado por Indra [11] 19
Figura 2-8 Simulador de tiro Victrix y simulador de navegación de la Escuela Naval Militar 20
Figura 2-9 Militares españoles practican un ataque inmediato [20] 23
Figura 2-10 El Primer Batallón se adiestra en combate en población (@Armada_esp) 23
Figura 2-11 Comparación entre eL HKG36 y su versión corta [17]....24
Figura 2-12 Captura extraída de la publicación de Defensa PD4021 25
Figura 2-13 Uno de los simuladores de conducción de la escudería de Ferrari [24] 26
Figura 2-14 La modalidad que practica la simulación militar es conocida como *milsim* [25]....26
Figura 2-15 Subroc, la primera recreativa que usaba gafas de Realidad Virtual [26]....27
Figura 2-16 Oculus Rift, las primeras gafas de RV accesibles para todo el mundo [26]....28
Figura 2-17 La Realidad Aumentada, cambia el mundo tal y como lo vemos [28]....28
Figura 2-18 Uno de los síntomas más comunes de la RV: el “motionsickness”....29
Figura 2-19 Los NPCs del GTA considerados los más complejos[32]....30
Figura 2-20 Especificaciones del PC utilizado....31
Figura 2-21 El sistema de scripting visual Blueprints facilita la codificación en C++ [36] 32
Figura 2-22 Autodesk Inventor, programa utilizado para el diseño del fusil HKG36 33
Figura 2-23 Autodesk 3DS MAX, programa utilizado para el modelado del fusil HKG36....33
Figura 2-24 Geforce RTX 2080 SUPER, tarjeta gráfica instalada en el equipo [39] 34
Figura 2-25 Relación entre la experiencia de juego esperada y del número de FPS [38]....35
Figura 2-26 Diferencia entre un procesado con Raytracing (RTX) y sin él [39]....35
Figura 2-27 Elementos periféricos del HTC VIVE....36
Figura 2-28 Mandos HTC VIVE Controller 2.0 37
Figura 2-29 Adaptador Beswin VR M4 utilizado [43]....38
Figura 2-30 Kriegspiel, el juego de mesa [44] 39
Figura 2-31 Firefight, el primer juego táctico encargado por el ejército de EEUU [47] 40

Figura 2-32 Transformación del videojuego comercial DOOM en su versión militar [44].....40
Figura 2-33 El videojuego America´s Army lanzado en distintas plataformas [51].....41
Figura 3-1 Etapas del desarrollo.....43
Figura 3-2 Selección del template Virtual Reality dentro de la categoría Games.....44
Figura 3-3 Interfaz del Level Editor de Unreal Engine y sus partes principales.....46
Figura 3-4 Militares españoles durante una misión en Abzi-Khuda, de Afganistán [55].....47
Figura 3-5 Chaki Wardak, poblado afgano situado a 97 kilómetros al suroeste de Kabul [56].....48
Figura 3-6 Opciones de creación de assets básicos desde la carpeta de contenido.....48
Figura 3-7 Configuración de nivel diurno y nocturno en UE4.....50
Figura 3-8 Aplanando el terreno con la herramienta esculpir.....51
Figura 3-9 Colocación de assets sobre el escenario.....52
Figura 3-10 Construcción de la casa objetivo.....53
Figura 3-11 Colocación del techado de la casa objetivo.....53
Figura 3-12 Elección de material para suelos y paredes de la casa objetivo.....54
Figura 3-13 Importación de assets.....54
Figura 3-14 Rediseño del modelo inicial del HK G36.....55
Figura 3-15 Partes movibles del HKG36 en la simulación.....56
Figura 3-16 Colocación del cilindro y la cámara en el fusil.....57
Figura 3-17 Creación del material de la mira telescópica.....58
Figura 3-18 Posibilidades del sistema de movimiento.....59
Figura 3-19 Ventana gráfica del motion controller pawn.....60
Figura 3-20 Blueprint que asocia la rotación en el eje Z de la cámara y el personaje.....61
Figura 3-21 Configuración del Axis Mapping.....62
Figura 3-22 Blueprint que controla el desplazamiento con trackpad.....63
Figura 3-23 Blueprint que soluciona el problema de las colisiones.....64
Figura 3-24 Animación de las manos en 3DS Max.....66
Figura 3-25 Trabajando en la Blendspace.....67
Figura 3-26 Configuración de las cajas de colisión del arma.....68
Figura 3-27 Colocación de puntos de acople o sockets del arma.....69
Figura 3-28 Colocación de la munición 5,56x45mm NATO en el cargador.....69
Figura 3-29 Fusil y cargador acoplados.....70
Figura 3-30 Colocación de sockets de agarre en la mano.....71
Figura 3-31 Funciones de disparo por defecto del arma.....72
Figura 3-32 Configurando variables del arma.....73
Figura 3-33 Definiendo las claves de la IA desde el Blackboard.....75
Figura 3-34 Árbol de comportamiento de la IA programada.....75

Figura 3-35 Secuencias de la 1º y 2º rama del árbol de comportamiento 76
Figura 3-36 Secuencias de la 3º y 4º rama del árbol de comportamiento 77
Figura 3-37 Blueprint de la percepción de la IA 78
Figura 3-38 Determinando el área de patrulla de la IA mediante el navmesh 79
Figura 3-39 Configuración de los sentidos de la IA desde el AI Perception 80
Figura 3-40 Acople del arma a la malla esquelética del enemigo 80
Figura 3-41 Asociando el Widget de información a un actor para situarlo sobre el nivel 82
Figura 3-42 Blueprint para orientar el Widget según nuestra visión 83
Figura 3-43 Blueprint que inicia el temporizador al entrar en la casa 83
Figura 3-44 Blueprint del temporizador 84
Figura 3-45 Blueprints de los eventos de victoria y derrota 85
Figura 3-46 Blueprint de cálculo de precisión 86
Figura 3-47 Blueprint que genera el archivo de texto con la información almacenada 87
Figura 3-48 Diseñador que controla el aspecto visual de la leaderboard 88
Figura 3-49 Gráfico que controla el comportamiento de la leaderboard 88
Figura 3-50 Asociando la leaderboard a un actor para situarlo sobre el nivel 89
Figura 3-51 Blueprint que extrae y ordena los datos de la leaderboard 89
Figura 3-52 Material generado para la visión nocturna 90
Figura 3-53 Comprobación de equipo de RV listo para usar desde Steam VR 91
Figura 3-54 Activación de la simulación mediante el equipo de RV 92
Figura 3-55 Blueprint que controla la recarga automática del modo con fusil real 93
Figura 3-56 Compensación del arma virtual 94
Figura 4-1 Parte del escenario generado 95
Figura 4-2 Poblado generado para aumentar la sensación de inmersión 96
Figura 4-3 Resultado de la casa objetivo 96
Figura 4-4 Interior de la casa objetivo 97
Figura 4-5 Fusil HKG36 Modelado 97
Figura 4-6 Avatar principal controlable por el jugador 98
Figura 4-7 Enemigos 99
Figura 4-8 Rehenes 99
Figura 4-9 Situación inicial de la simulación 100
Figura 4-10 Enfrentamiento con enemigo 101
Figura 4-11 Uso de la mira telescópica 101
Figura 4-12 Evento de victoria 102
Figura 4-13 Modo visión nocturna 102

Figura 4-14 Registro generado con los datos de cada jugador.....103
Figura 4-15 Simulación con controladores acoplados al dispositivo.....104
Figura 4-16 Logo diseñado para el simulador.....104
Figura 4-17 Video demostración.....105
Figura 4-18 Código QR con enlace al vídeo demostración.....105

1 INTRODUCCIÓN

1.1 Descripción del apartado

El desarrollo de este primer apartado de carácter introductorio está destinado a la interiorización del porqué de este proyecto, así como a la comprensión de los objetivos que plantea. Tras una revisión de los antecedentes que han cosido la línea de investigación de proyectos de esta índole se pasará a realizar una justificación de este de tal forma que toda persona con acceso a la presente memoria pueda hacerse una idea clara de las necesidades y los problemas que se han tratado de dar solución. Por último, se detallará la estructura que organizará los contenidos recogidos en la memoria.

1.2 Antecedentes

Para entender la motivación y las necesidades que han llevado a comenzar la presente línea de investigación es necesario primero conocer los antecedentes que nos han llevado hasta este punto. Cuando nace la necesidad de entrar en un tema de investigación, especialmente si uno no es experto en la materia, no solo es importante adquirir el conocimiento de la metodología empleada, es importante también primero limitar con exactitud el concepto que estamos tratando. De esta manera el lector será capaz de seleccionar después la información y adquirir la perspectiva necesaria con la que personalmente prefiera abordar la lectura de la memoria.

Sabiendo que el presente trabajo de fin de grado está enmarcado dentro del plan de estudios que plantea el Centro Universitario de la Defensa, tanto la temática como el objeto de los proyectos que de estos trabajos pueden llegar a surgir, en su inmensa mayoría, pertenecen al campo de la investigación militar. Este campo, como el resto de los campos de investigación, no es más que un proceso de estudio de una necesidad o de una idea que tras ser desarrollada plantea nuevas soluciones y nuevos resultados hasta ahora nunca concebidos.

Para dar los primeros pasos en una investigación se debe plantear en primer lugar cuáles son las necesidades reales a las que nos enfrentamos y a las que se podría dar respuesta. ¿Existe una necesidad? ¿Cómo de importante es esta necesidad? ¿Es factible una solución para la necesidad que me planteo?

Tras cuestionarse las necesidades, comienza el proceso de esbozo de una idea. Este proceso ya debe tener en mente lo que se necesita conseguir y para ahondar aún más y asentarse como idea definitiva a desarrollar deben estructurarse sus bases en la convicción de que ya no solo la idea es factible, sino que además se disponen de los medios y recursos necesarios para su desarrollo. Es aquí donde la investigación y la lectura previa, donde los trabajos que con anterioridad ya han abordado necesidades de carácter similar y la examinación de los recursos dotarán a la idea esbozada el cariz de concepto factible, realista y definitivo.

Concretamente, tal y como justificaremos en el siguiente apartado, la necesidad de equipar a nuestras Fuerzas Armadas de una tecnología adecuada, práctica, económica y eficiente desemboca en la idea de implementar el uso de la Realidad Virtual (RV) como un elemento más de entre todos los recursos con los que hoy en día cuenta nuestro Ejército.

La realidad virtual es en primer lugar, una tecnología imprescindible dentro del proceso de transición que estamos actualmente viviendo hacia la cuarta revolución industrial y su concepto de Industria 4.0. Más adelante, en el apartado del estado del arte, se tratará esto más a fondo. Por lo tanto, si la realidad virtual es un elemento inseparable de esta nueva forma de concebir la tecnología podemos afirmar que su uso será también relevante para la Armada. Esta asociación entre Industria 4.0 y Armada es algo sobre lo que trabaja en el ámbito logístico la Dirección General de Armamento y Material. Durante la segunda jornada del ciclo de conferencias Symdex, que tuvo lugar en Madrid en el año 2019, los jefes de sección de la gestión de la información y el apoyo al sostenimiento verbalizaron el deseo de dotar a la Armada de la correcta coordinación en el ámbito de la transformación digital. Se nombraron las principales herramientas con las que cuenta la Armada 4.0, incluidas en la Figura 1-1, También se recalcó el fundamental papel que la Realidad Virtual juega en este proceso.

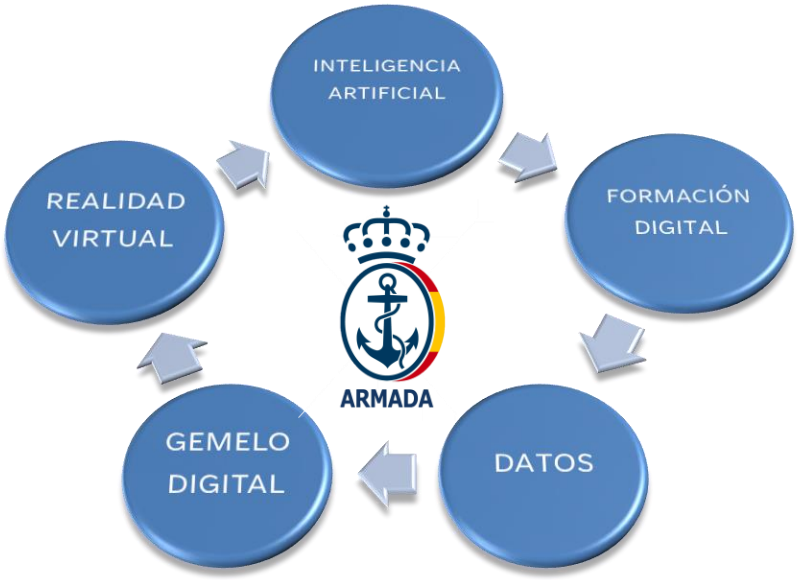


Figura 1-1 Herramientas de la Armada 4.0 [1]

El mundo de la simulación y el amplio abanico de posibilidades que presenta no puede pasarse por alto y menos siendo conscientes de las ventajas que plantea concretamente en el ámbito de la enseñanza que en nuestro caso está concebida como instrucción y adiestramiento.

La Armada Española, conocedora de la importancia de esto mismo, enfoca desde hace tiempo sus investigaciones en aras de una mejora continua dotando siempre a la tecnología del rol que se merece. Para ello se han tenido que dejar atrás conceptos equívocos que por ejemplo asociaban el uso de la RV exclusivamente al mundo de los videojuegos. Por medio de la gamificación y la visualización 3D la Enseñanza Naval poco a poco está labrando un interesante y sobre todo provechoso camino que sin lugar a duda supone una mejora el rendimiento en todos los aspectos.

El reto es, por tanto, siendo conscientes de lo que la Armada y en concreto nuestros combatientes necesitan, aportar un nuevo escalón hacia la consecución de los objetivos de modernización de los recursos disponibles en las escuelas y centros docenes de formación, alcanzar una nueva forma de entender el adiestramiento mediante la simulación y reforzar el concepto de aprovechamiento de la RV como herramienta indispensable en nuestras Fuerzas Armadas.

1.3 Justificación

"La guerra es un hecho social que se extiende hasta allí donde llega el hombre. Si el hombre llega hasta internet, a las redes sociales, la guerra también". Esta interesante reflexión tratada por Federico Aznar Montesinos en su ensayo sobre los militares y la tecnología explica bien la conexión existente entre los Ejércitos y el desarrollo tecnológico. La guerra es mutación, cambio, superación. No son pocos los avances acaecidos en la historia de la humanidad que han sido alcanzados gracias a la estimulación de un estado de necesidad. [1]

Pero estos mismos hechos históricos no sólo valen para ejemplificar todo lo que el terreno de lo militar ha alcanzado en cuanto a materia tecnológica, además estos hitos refuerzan, validan, en esencia, justifican la necesidad de estos desarrollos en base a la evidente superioridad que aportan. Esta superioridad tecnológica ha sido tratada, sobre todo desde la segunda mitad del siglo XIX, por parte de las grandes potencias militares como un elemento decisivo para las guerras. Las diferencias, que en este ámbito se comprende como un gap tecnológico, limita mucho las capacidades de quien se encuentra en inferioridad tecnológica y casi siempre resulta fatal.

Aparece por tanto la primera necesidad, la de mantenerse a la vanguardia en cuanto a tecnología como elemento determinante y decisivo para nuestro propio beneficio militarmente hablando. El presente trabajo de fin de grado podría quedar más que justificado tras este planteamiento, sin embargo, se puede concretar más aún y enfocar este concepto hacia el plano de la enseñanza.

El adiestramiento (ver Figura 1-2), es la educación en materia de destrezas y habilidades que aporta al personal la capacidad de desenvolverse dentro de un puesto de nuestras Fuerzas Armadas. Hoy en día, ya es difícil separar la tecnología de la educación y, por ende, del adiestramiento. La digitalización y las herramientas virtuales han cambiado radicalmente el concepto hasta ahora un tanto inamovible que tenían ya no sólo los centros docentes militares, sino también los centros y escuelas civiles.

Figura 1-2 Adiestramiento en combate urbano (Twitter Oficial Armada Española, @Armada_esp)

El adiestramiento militar es especialmente complejo, su desarrollo exige de una correcta coordinación logística y de personal que no siempre resulta eficiente en términos de rentabilidad. La planificación, la ejecución y la evaluación de un plan de enseñanza militar actualmente contempla una

red de propósitos poco acorde con los recursos de los que se dispone, por lo que desemboca muchas veces en la limitación y en el recorte por necesidad. Esto por supuesto, tiene un impacto negativo en la enseñanza del alumnado. La simulación plantea una solución directa a esto, enterrando las complicaciones logísticas y de organización de personal que como ya hemos explicado restringen las buenas intenciones de la enseñanza militar. A continuación, se listan algunas de las principales ventajas que un simulador de combate puede aportar:

- La recreación del escenario de manera virtual ahorra la necesidad de desplazamiento de los alumnos.
- Al encontrarse en un entorno al cien por cien controlado, los instructores pueden obtener datos paramétricos y precisos de cada uno de los alumnos. Se alcanza, por tanto, una enseñanza individualizada muy positiva.
- La simulación de distintas situaciones sirve para medir las reacciones, los comportamientos y la conducta del alumnado además de ayudarlo a familiarizarse con el estrés propio de las situaciones reales a las que en un futuro tendrá que hacer frente. Se alcanza así, una educación clave para el liderazgo y se mejora la formación en toma de decisiones.
- Evidente ahorro de coste y menor tiempo de ejecución.
- La disposición, la inclusión de ejercicios basados en los conceptos que se pretenden enseñar permite al alumnado poner en práctica la teoría previamente adquirida. Con esto se logra ejecutar la doctrina y dotar al simulador de una importante capacidad táctica.

1.4 Objetivos

Los objetivos del presente Trabajo de Fin de Grado permitirían dar solución a las necesidades anteriormente contempladas. Pero para ello, primero se deben trazar las expectativas de lo que se pretende lograr teniendo siempre en mente las limitaciones de tiempo que presenta la exigencia de una fecha de entrega como es el caso. A continuación, se presentan los objetivos iniciales del proyecto:

- Generar un escenario bélico, fotorrealista y detallado cuya finalidad sea su exploración mediante el uso de la tecnología de Realidad Virtual
- Alcanzar un alto nivel de interacción con el entorno desde la perspectiva de una persona real. La posibilidad de desplazamiento debe ser lo más realista posible al igual que las mecánicas de interacción con el armamento para alcanzar una buena sensación de inmersión.
- En concreto se recreará el fusil de asalto HKG36 y se diseñará la mira de aumento característica del fusil buscando su uso funcional. El fusil debe de contar con una secuencia lógica de agarre, carga y disparo.
- Desarrollar una inteligencia artificial de nivel que controle el comportamiento de enemigos que opongan resistencia. Esta inteligencia debe responder a los estímulos del entorno, ser capaz de reconocer visualmente la presencia del personaje controlado y contar con un sistema de salud.

Tras la consecución de estos objetivos previos se pasa a la descripción del último de todos, el que engloba la verdadera razón de ser del presente trabajo de fin de grado:

- Generar un simulador concebido para mejorar el adiestramiento individual y que se valga de la Realidad Virtual para controlar la experiencia. El nivel deberá basarse en el asalto a un objetivo, dónde se puedan aplicar los conocimientos doctrinales y se pongan en práctica los movimientos y actuaciones tácticas esperadas por el alumno/usuario. Estos niveles deben contar con dos fases principales: una primera donde se permita la familiarización con el entorno y el armamento virtual y una segunda destinada al asalto de la posición con un tiempo limitado para ello.

1.5 Estructura

El orden de escritura y el desarrollo de la presente memoria sigue las pautas marcadas por el Centro Universitario de la Defensa. En concreto el trabajo queda estructurado en los cinco apartados principales (desglosados en detalle en subapartados) que a continuación se listan:

1. *Introducción*: apartado al que pertenece el presente subapartado, donde como se ha podido comprobar, quedan establecidos los antecedentes, la justificación y los objetivos a cumplir por el presente trabajo de fin de grado. Tras su lectura el lector podrá hacerse una idea de qué es este proyecto, qué es lo que se busca y cuáles son las necesidades que han llevado hasta el mismo.
2. *Estado del arte*: apartado destinado a la contextualización histórica y actual que atañe la tecnología empleada. También se presentan los diferentes recursos utilizados tales como el software y los equipos implicados. También se esclarece la instalación de estos últimos. Tras su lectura el lector adquirirá las nociones básicas con las que introducirse en la Realidad Virtual y comprenderá la importante relación que existe entre la simulación y el adiestramiento militar.
3. *Desarrollo*: apartado con el contenido grueso, de carácter más técnico, donde se explicarán los pasos realizados para alcanzar los objetivos del proyecto. Seguirá una estructura que facilitará su comprensión, partiendo desde la concepción del mapa inicial hasta el diseño final del nivel alcanzado. Tras esto se pretende que el lector sea capaz de comprender el proceso de creación del proyecto, buscando ser lo más claro y entendible posible para la mayoría de las personas que accedan a la memoria.
4. *Resultados*: apartado que presenta los frutos del desarrollo previamente explicado. Es el producto final de todo el trabajo hecho. Se presentará el simulador y el nivel en su versión definitiva de tal manera que el lector tras leer este apartado sepa a ciencia cierta qué se ha logrado.
5. Conclusiones y líneas futuras: se examinarán los resultados y se valorará su correspondencia con los objetivos iniciales del proyecto. Además, se indagará en la observación de nuevas posibilidades de desarrollo y se plantearán algunas mejoras que por falta de tiempo no han podido ser incluidas como parte del presente trabajo de fin de grado. El lector, una vez concluya este apartado sabrá valorar la utilidad real de este simulador y podrá, en el mejor de los casos, continuar con la línea de investigación que se plantea ya que esta se deja aún abierta con la esperanza de encontrar continuación.

2 ESTADO DEL ARTE

2.1 Descripción del apartado

El presente apartado se procede a contextualizar y a definir el marco histórico que ha desembocado en la necesidad de creación de un proyecto como este. Se comenzará con un repaso aclaratorio desde un punto de vista militar de los conceptos básicos y de necesario conocimiento para la mejor comprensión por parte del sector civil y finalmente se abordará la contextualización desde el punto de vista del ámbito civil donde se discutirán las tecnologías empleadas.

Todas las figuras incluidas en estado del arte a excepción de las debidamente referenciadas son de elaboración propia.

2.2 Del ámbito militar

2.2.1 La Industria 4.0 en Defensa

La cuarta revolución industrial también conocida como Industria 4.0 es una realidad y el mundo militar no es ajeno a esto. Determinar cuándo comenzó exactamente esta nueva era, en la que la interconexión, la globalización y la fusión de las altas tecnologías han cambiado nuestra forma de entender la industria por completo, no es tarea sencilla. Sin embargo, podemos remontarnos a la primera mención al término la cual tuvo lugar el año 2016. El economista y empresario alemán Klaus Schwab en el contexto del Foro Económico Mundial del citado año entonces se refirió a esta revolución como el nuevo escalón alcanzado por la humanidad. La Figura 2-1 muestra la evolución histórica del concepto de Industria. [2]

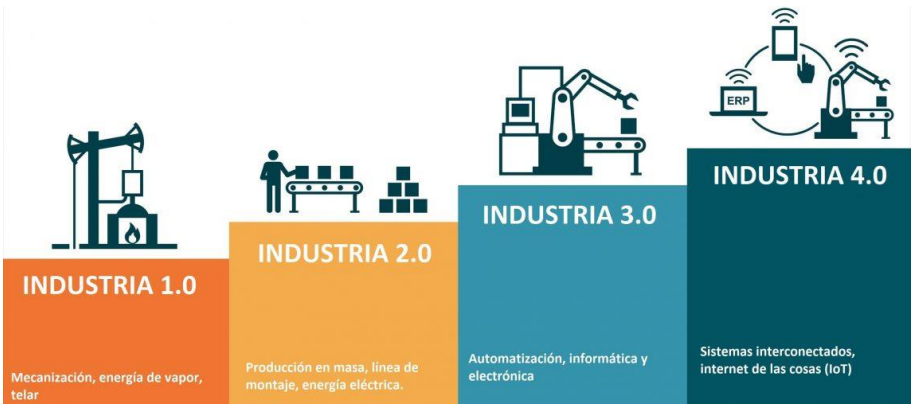


Figura 2-1 Evolución histórica de la Industria y las tecnologías características de cada etapa [3]

A pesar de la incertidumbre que generan estas etapas transitorias, estamos empezando a destapar las ventajas de la nueva era industrial y como consecuencia la guerra en el mundo entero está en pleno proceso de adaptación también. La guerra ahora puede llegar a ganarse sin librarla, a través de una victoria virtual, otorgada por una diferencia de tecnología. En este marco se explicaría la carrera espacial abandonada poco después del fin de la Guerra Fría. Las carreras armamentísticas y ahora tecnológicas corresponden a la puesta en marcha de dialécticas de superación fruto del principio clauswitziano de acción recíproca. Otro ejemplo es la doctrina británica conocida como Two Powers Standard según la cual la armada británica debería ser superior a la suma de las dos siguientes marinas. Aunque si bien estos objetivos son en muchas ocasiones prácticamente irrealizables, debido a las limitaciones económicas y sociales de cada país, el afán de superación constante en cualquier materia resultará siempre beneficioso. [1]

Este cambio de mentalidad, al que en España se está sumando con eficacia, debe ser estudiado minuciosamente. En concreto, la industria de Defensa avanza en pos de modernizarse de manera progresiva. La evolución del modelo de las Fuerzas Armadas (FAS) se comprende como el resultado de la aplicación de la lógica de adaptación-superación con el fin último de mantenerse a la vanguardia en cuanto a tecnología se refiere. Sin olvidar sus capacidades, los planes a corto y largo plazo de nuestra política de Defensa han entendido el valor de la Industria 4.0 y el papel que esta ahora juega en los procesos de modernización.

En la Estrategia de Tecnología e Innovación para la Defensa (ETID) publicada en el año 2020 por la Secretaría de Estado de Defensa, a través de la Dirección General de Armamento y Material se marcan los tres pilares que deben guiar cualquier actividad I+D+i promovida en el ámbito del Ministerio de Defensa: los objetivos tecnológicos, la cooperación y la mejora continua. Dentro del Anexo A. Líneas de I+D+i de interés para defensa de la ETID se describen hasta un total de 11 áreas que organizan el conjunto de líneas de I+D+i relevantes para el futuro de nuestras Fuerzas Armadas (Figura 2-2). La última de estas áreas es la denominada Tecnologías de la Información, Comunicaciones y Simulación, la cual describe el adiestramiento avanzado mediante simuladores como una solución innovadora y efectiva para la preparación del personal. Además, destaca lo beneficioso que resulta una recreación fiel a las condiciones de las operaciones y pone en valor esta herramienta como un potencial adiestrador para la toma de decisiones. [4]

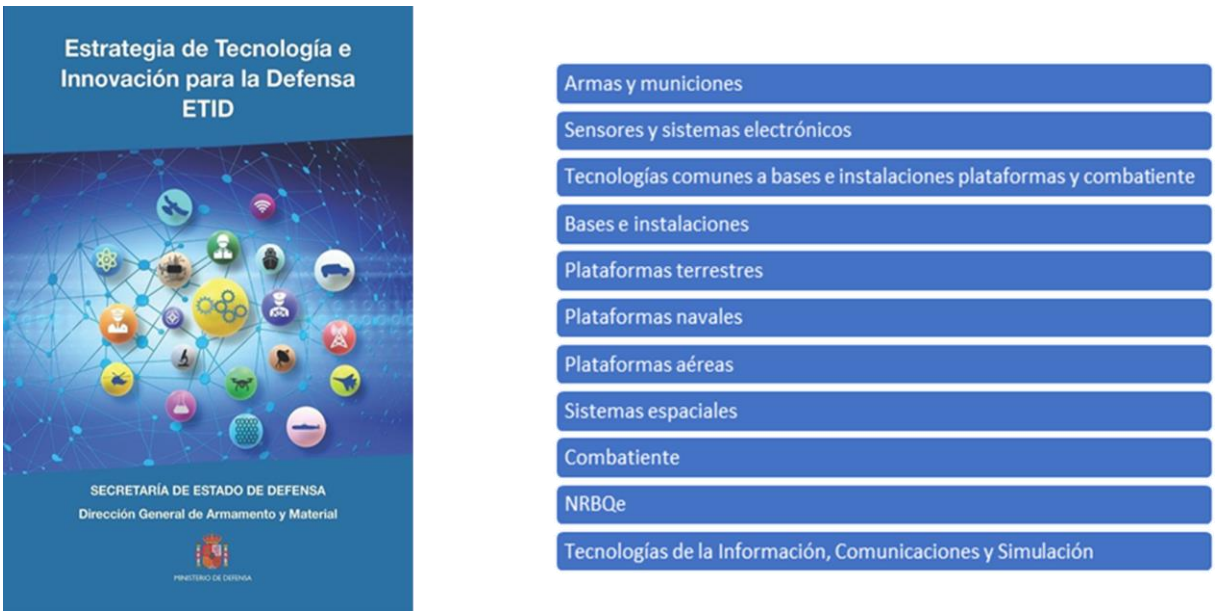


Figura 2-2 Las 11 áreas de I+D+i contempladas por la ETID [4]

2.2.2 Alta tecnología en las FAS

En el marco de la Industria 4.0 descrita en el apartado anterior, las Fuerzas Armadas han explotado muchas de las tecnologías innovadoras que desde su implementación han facilitado el desarrollo de las misiones y han aumentado el nivel de seguridad de nuestros sistemas.

Estas mejoras tecnológicas alcanzan casi cualquier ámbito de nuestro Ejército. Sin ir más lejos, las comunicaciones en entornos complejos han evolucionado dando paso a nuevos tipos de plataformas y sensores. Muchos de los problemas de interoperabilidad entre distintos niveles y las limitaciones de ancho de banda generadas por las nuevas necesidades han sido solucionados basándose en la tecnología SDR (Software Defined Radio, Figura 2-3) que es capaz de realizar diferentes operaciones modificando su configuración desde el software o firmware sin necesidad de alterar su parte física de fábrica o hardware. Además, la problemática generada por la coexistencia de radios actuales con los inhibidores también ha sido solventada mediante el uso de plataformas multicanal o multibanda. [5]

Figura 2-3 Sistema de radio definido por software SDR para vehículos E-Lynx de Elbit systems [5]

El empleo de las tecnologías más avanzadas ha supuesto una mejora en nuestros sistemas de detección remota de artefactos explosivos improvisados (IED, Figura 2-4). Estos ahora son capaces de localizar una sustancia explosiva o un artefacto gracias a sus emisiones de vapor o por sus residuos sólidos y líquidos. También tienen la habilidad de detectar anomalías en la densidad del medio y movimientos sospechosos de tierras, algo que hasta hace poco no podría ni llegar a imaginarse. [6]

Figura 2-4 Avenger, el robot del Ejército de Tierra frente a minas trampa [6]

Otra de las tecnologías con mayor potencial, es la relacionada con el mundo de los vehículos no tripulados. Además de contar con las capacidades de los drones (Figura 2-5), ya muy comunes también

en el ámbito civil, las FAS trabajan por la implementación y el desarrollo de nuevas alternativas que reduzcan al mínimo el empleo de personal en las operaciones y, por tanto, el riesgo humano durante las mismas. Entre estos proyectos encontramos submarinos, embarcaciones de superficie y plataformas terrestres no tripuladas. [7]

Figura 2-5 Pelotón de Observación del BIPR "Las Navas" II/6 perteneciente al Ejército de Tierra poniendo a prueba la aeronave no tripulada RPAS MANTIS EQF desarrollada por INDRA [7]

El uso de exoesqueletos para misiones de Defensa a pesar de encontrarse en un nivel de madurez tecnológico aún bajo debido a los complicados retos que plantea, es otro de los campos a desarrollar según la ETID y, por lo tanto, es una nueva línea de trabajo de nuestras FAS a tener muy en cuenta. Con la implantación de los exoesqueletos se podría dar un salto enorme en cuanto a potenciación de las capacidades de nuestro personal, especialmente de aquel personal inmerso en operaciones de alta demanda física. La marcha, el transporte y la manipulación de cargas, cada vez más pesadas, debido al aumento de sistemas que un combatiente debe portar consigo actualmente, se verían altamente beneficiadas, ya que se reduciría casi al mínimo el esfuerzo físico y el riesgo de lesiones. La Figura 2-6 muestra uno de los exoesqueletos más avanzados hasta la fecha. [8]

Figura 2-6 Exoesqueleto Lockheed Martin FORTIS para aliviar el estrés de la rodilla [9]

En definitiva, lo que se trata es de mejorar el rendimiento humano a través de la tecnología. En la Agencia Europea de Defensa, el concepto se conoce como 'human enhancement' (perfeccionamiento humano) y en la OTAN hablan de 'human augmentation' (aumentación humana). Numerosos países del entorno OTAN ya incluyen sistemas de realidad virtual y tecnologías que facilitan la interacción hombre-máquina.

Todo estos son ejemplos de una extensa red de avances y líneas de investigación que manifiestan la importancia y la situación actual de las altas tecnologías en nuestras Fuerzas Armadas.

2.2.3 Simuladores, la herramienta definitiva para el adiestramiento

Uno de los grandes logros de la tecnología actual es sin duda su facultad de recrear virtualmente la realidad tal y como la conocemos casi a la perfección. Gracias a esto, el mundo de la Defensa ha sido capaz de probar y refinar sus propias teorías sobre la guerra, así como llevar a cabo simulaciones de especial utilidad para la optimización y mejora de la formación de sus fuerzas militares en un amplio espectro de operaciones.

La empresa multinacional española Indra Sistemas, S.A ha desarrollado numerosos proyectos relacionados con la simulación para nuestras Fuerzas Armadas (Figura 2-7), desde sistemas de entrenamiento para conductores, operadores de torres y jefes de carro para vehículos como el Leopardo 2E o RGP-31, hasta otros tipos de simuladores más relacionados con el ámbito naval, como los simuladores tácticos y de navegación de los nuevos submarinos S-80. [10]

El día 7 de Julio de 2021 Indra anunció el lanzamiento de un sistema de simulación de vuelo basado en realidad virtual y aseguran que con este nuevo simulador se reducirá a la mitad el tiempo para formar pilotos. Este simulador incluye el sistema de entrenamiento de pilotos de aeronaves remotamente pilotadas fabricado en España más avanzado hasta la fecha. El proyecto ha sido desarrollado en colaboración con la Xunta de Galicia dentro de la Civil UAVs initiative. Uno de los grandes logros de estos simuladores es que además cuentan con una valiosa función de interconexión entre varios simuladores a través de una red de área local específica, lo que permite ampliar el espectro de operaciones y entrenamientos a nuestro alcance. [10] [11]

Figura 2-7 Simulador de vuelo Eurofighter desarrollado por Indra [11]

La posibilidad de probar diferentes plataformas, elegir los entornos y condiciones en los que desarrollar la simulación, el beneficio económico que supone el no precisar de apoyos logísticos de ningún tipo más allá de los mantenimientos y operadores del propio simulador y la reducción al mínimo del riesgo de personal convierten a esta herramienta en la solución prácticamente definitiva para el adiestramiento.

Es por ello por lo que los centros docentes militares en los últimos años han incluido las horas de simulación como una asignatura más a superar en sus planes de estudio. Sólo en la Escuela Naval Militar podemos encontrar un total de tres simuladores de distintos tipos: un simulador de navegación, un simulador táctico y un simulador de tiro virtual. [12] [13]

Como alumno de la Escuela Naval y desde la perspectiva adoptada por la experiencia propia he podido comprobar como el simulador de navegación es un medio excelente para poner en práctica los conocimientos previamente adquiridos en las clases teóricas. Las cinemáticas de movimiento, el empleo de los diferentes tipos de armamento o los sensores del barco son mucho más sencillos de comprender tras pasar por el simulador y haber practicado sus numerosos procedimientos. Además, otorga una visión mucho más amplia del concepto de navegación gracias a que este permite simular maniobras tales como

atracadas y desatracadas en diferentes puertos de toda la geografía española. Practicar esta clase de maniobras sería mucho más complejo y costoso en la vida real.

En cuanto al simulador táctico, su uso resulta también muy provechoso. Aporta, al igual que el simulador de navegación, una valiosa formación práctica en cuanto a utilización de equipos, pero presenta la ventaja de poder trabajar simultáneamente desde varias consolas operadas por un grupo de alumnos de forma independiente. Así se simula de manera muy fiel a la realidad todo el trabajo de planificación que requiere la coordinación previa a una operación. En una consola aparte, el instructor puede variar las condiciones y las amenazas a las que los alumnos deben hacer frente, por lo que la improvisación y la comunicación entre ellos también es algo que se pone en práctica durante los ejercicios. Al finalizar la simulación se puede extraer la información necesaria para la evaluación de los alumnos.

Los testimonios de los alumnos del cuerpo de Infantería de Marina de la Escuela Naval Militar respecto al uso del simulador de tiro Victrix arrojan luz sobre su satisfactoria aplicación, sin embargo, todos parecen estar de acuerdo en que la experiencia podría mejorarse si esta fuese algo más dinámica. Los alumnos consideran que sería más provechoso adiestrarse en el tiro instintivo si este se desarrollase en un ambiente en tres dimensiones. Actualmente el simulador de tiro está limitado al empleo de una pantalla que proyecta el entorno, pero ésta no ofrece más que una visión parcial y limitada del complejo ejercicio que se pretende llevar a cabo. En el combate en tiempo real los estímulos provienen desde cualquier dirección e influyen de manera directa en la forma de actuar del combatiente de ahí que este proyecto pretenda abordar esta cuestión.

Figura 2-8 Simulador de tiro Victrix y simulador de navegación de la Escuela Naval Militar

2.2.4 Combate en población

El presente trabajo de fin de grado tiene como objetivo la simulación de un entorno bélico, sin embargo, el concepto "entorno bélico" es un término que abarca demasiado. Es necesario antes de comenzar, hacer ciertas puntualizaciones en aras de una mejor comprensión de la ambientación escogida para este proyecto en concreto.

Como entorno bélico se entiende cualquier posición geográfica afectada por las hostilidades derivadas de un enfrentamiento directo o indirecto entre dos fuerzas que están en conflicto por una lucha de intereses. Hay tantos entornos bélicos como guerras. La topología bélica a lo largo de los años ha ido evolucionando gracias, principalmente, a los avances tecnológicos, sociales y económicos hasta abarcar cualquier escenario posible. [1]

De primeras, puede parecer evidente que las capacidades actuales permiten el desarrollo de la guerra tanto por mar como por tierra y aire sin embargo un entorno bélico también puede ser, por ejemplo, el ciberespacio. De hecho, éste es ya considerado por el Centro Superior de Estudios de la Defensa Nacional (CESEDEN) como un nuevo escenario de confrontación. Además de la guerra terrestre, naval, aérea y la ya mencionada cibernética, el mismo CESEDEN reconoce la existencia de una quinta guerra: la guerra espacial. [14] [15]

En este proyecto se tratará un tipo específico de escenario bélico, concretamente el terrestre y dentro de este se abarcarán las operaciones desarrolladas en ambiente urbano. Este tipo de operaciones durante los últimos años se han vuelto cada vez más frecuentes debido a fenómenos como la expansión urbana y cabe recalcar que no sólo tienen un importante valor estratégico; estas operaciones, además cuentan con un marcado componente político, diplomático, económico y social. Por eso, las operaciones urbanas se consideran de especial complejidad. Son muchas las consideraciones que hay que tener en cuenta antes de iniciar una operación de este tipo, por lo que para dotarlo de un mayor realismo y de un componente doctrinal es conveniente primero dejar claro a todo el personal, tanto civil como militar, la definición exacta de las reglas o directrices de obligatorio conocimiento que marcan su desarrollo. También es importante conocer los tipos de acciones tácticas de carácter ofensivo que existen. Para su explicación se tomarán las publicaciones de referencia pertinentes tanto a nivel OTAN (Organización del Tratado del Atlántico Norte) como a nivel nacional que regulan este tipo de operaciones. [16] [17]

2.2.4.1 Definición

Según la publicación de Defensa PD4-021 el combate en ambiente urbano o como se le conoce comúnmente entre nuestras unidades el "combate en población" solo puede considerarse como tal cuando converjan tres elementos esenciales y estrechamente interrelacionados entre sí: el terreno urbano, el elemento humano y las infraestructuras. [17]

Por *terreno urbano* se entiende el entramado de estructuras realizadas por el hombre y asentadas sobre terreno natural. El *elemento humano* hace referencia a la población habitante del terreno urbano y que influenciará de manera acentuada en el desarrollo de estas operaciones. Las *infraestructuras* son los elementos de los que depende el funcionamiento de la zona urbanizada. [17]

2.2.4.2 Reglas de enfrentamiento

Además de los conceptos tácticos, vitales para el éxito de una operación, el aspecto legal también es de necesario conocimiento para una vez desplegados ser capaces de discernir entre lo que se puede hacer y lo que no a fin de evitar las conocidas escaladas de tensión innecesarias, el fuego amigo o un enfrentamiento derivado de un malentendido por alguna de las partes. Actualmente, el ámbito legal contempla cuatro conceptos básicos que regulan las operaciones militares: las Reglas de Enfrentamiento (REN o ROE, Rules Of Engagement), la Defensa Personal, la Defensa Personal Extensible y el Derecho Internacional Humanitario. Las Reglas de Enfrentamiento surgen como consecuencia de las dificultades a las que debía hacer frente el Derecho Internacional Humanitario para dirigir cada operación en

específico. Esto se debía a la enorme cantidad de variables a tener en cuenta durante los conflictos. De ahí por tanto nace la necesidad de definir detalladamente la legalidad de cada una de las acciones tomadas durante los conflictos armados. Las reglas de enfrentamiento plantean órdenes y códigos de actuación en forma de catálogos de autorizaciones y prohibiciones sobre el uso de armas. A su vez las ROES sustentan y marcan las responsabilidades de cada acción tomada. A través de la cadena de mando se puede otorgar, retener o delegar el respaldo legal de las decisiones contempladas en las ROEs. [18] [19]

2.2.4.3 Tipos de acciones tácticas/ofensivas

Las publicaciones nacionales que discuten los procedimientos a seguir durante las operaciones tácticas de carácter ofensivo en zonas urbanizadas incluyen las siguientes acciones que podrían llevarse a cabo:

- Reconocimiento en fuerza: no se tiene que realizar de manera obligada, aunque se recomienda cuando exista necesidad de adquirir inteligencia precisa sobre el enemigo o su posible localización. De los reconocimientos se esperan obtener los puntos más fuertes y los menos sólidos de la defensa del enemigo. También se estudia la conveniencia de ejecutar un ataque inmediato o un ataque premeditado. [17]
- Ataque inmediato o premeditado: sobre esto se versa más en profundidad en el siguiente apartado El ataque2.2.5 titulado *El Ataque* debido a su relevancia. [17]
- Ataque de desarticulación: se busca interrumpir las acciones preparatorias de un ataque por parte del enemigo. [17]

Llegados a este punto es conveniente introducir otro concepto vinculado al presente trabajo de fin de grado, y es el concepto de *incursión*, quedando este definido por las acciones llevadas a cabo durante la penetración en una zona con el fin de realizar un ataque. La incursión se caracteriza por su rapidez y su movilidad sobre objetivos clave. La incursión será posible mientras el área urbanizada lo permita y no se haya perdido el factor sorpresa sobre la resistencia enemiga. Al acabar una incursión se puede consolidar la zona o se pueden replegar las fuerzas para estar en condiciones de llevar a cabo una nueva misión. [17]

2.2.5 El ataque

De entre las tres acciones tácticas ofensivas posibles contempladas por nuestros procedimientos, mencionadas en el apartado anterior, se cree conveniente detenerse y hacer mención específica y detallada de la acción del *ataque* debido a que ésta es la acción más decisiva en las operaciones de combate en ambiente urbano. El ataque, busca el centro de gravedad de la defensa del enemigo, y actúa contra él desde direcciones que este no espera. Cobran entonces especial relevancia las dimensiones del terreno urbano, la infiltración y la sorpresa. Dependiendo de la situación y la defensa presentada por el enemigo, el ataque puede llevarse a cabo de dos formas: ataque inmediato o ataque premeditado.

2.2.5.1 Ataque inmediato

Es la forma de ejecución consecuente tras un combate de encuentro. También este tipo de ataques está pensado para aprovechar la sorpresa ante defensas poco preparadas. Los ejes de progresión de nuestras fuerzas deben buscar siempre huir de las zonas en las que el enemigo pueda tener previstos fuegos de destrucción (normalmente se evitarán grandes avenidas y áreas despejadas). Se puede usar *la infiltración*, para ocupar puntos clave de la localidad. La secuencia de actuación de un ataque inmediato es difícil de marcar con rigidez, debido a las posibilidades que pueden ir surgiendo durante el desarrollo de la operación. Se puede alterar su orden, pero en cualquier caso las acciones esenciales durante un ataque inmediato serán:

- Avanzar a la zona urbana para establecer contacto. [17]
- Fijar los elementos avanzados enemigos. [17]
- Identificar las zonas o puntos más débiles del sistema defensivo. [17]

- Desplazarse rápidamente, a través de los puntos débiles de la defensa o las vías de penetración urbanas, evitando un combate de desgaste. [17]

Un ejemplo de ataque inmediato puede ser la acción iniciada como consecuencia de una emboscada o un ataque sorpresa por parte del enemigo (ver Figura 2-9)

Figura 2-9 Militares españoles practican un ataque inmediato [20]

2.2.5.2 Ataque premeditado

Esta clase de ataques se fundamenta en la preparación y el planeamiento previo detallado a través de un aporte de inteligencia preciso. Se llevará a cabo cuando el enemigo presente una defensa organizada, cuando no podamos contar con el factor sorpresa o tras un ataque inmediato sin éxito. El planeamiento debe prevenir la zona más probable de encuentro con el enemigo, y arrojar luz sobre sus condiciones para el combate, es decir, si es posible, se debe primero estudiar el armamento o el emplazamiento en que el enemigo ha establecido su defensa. El desarrollo de la operación conducirá a situaciones y cambios difíciles de prever por lo que el plan de ataque debe ser sencillo y flexible, pero no debe dar de lado nunca las consideraciones de inteligencia.

Un ejemplo de ataque premeditado puede ser una operación destinada al rescate de un rehén o a la neutralización de un objetivo en específico del que hemos obtenido suficiente inteligencia. Esta clase de ejercicios son muy comunes en el adiestramiento de nuestras unidades de Infantería de Marina tal y como muestra la Figura 2-10. [17]

Figura 2-10 El Primer Batallón de Desembarco del Tercio de Armada se adiestra en combate en población (Twitter oficial de la Armada Española, @Armada_esp)

2.2.6 Fases del ataque

La ejecución de un ataque incluye las siguientes fases:

1. Aislamiento
2. Asalto a los objetivos intermedios
3. Progresión y limpieza en el interior
4. Consolidación y reorganización

A ser posible estas fases se ejecutarán de forma metódica y secuencial. En cualquier caso, es importante saber que no es conveniente la pausa entre ellas, para evitar que el enemigo tenga tiempo para su reorganización y reacción. De entre estas cuatro fases se considera de especial conveniencia la aclaración y la exposición en detalle de dos fases en específico que serán las que se abarcarán en los niveles de simulación del presente trabajo de fin de grado: el asalto a los objetivos y la progresión y limpieza en el interior de la población. [17]

2.2.6.1 Asalto a los objetivos intermedios

Durante el desarrollo de esta fase, posterior al aislamiento, se deben ejecutar todas las acciones posibles que faciliten la sorpresa. Se conquistarán las posiciones que puedan servir de apoyo a nuestro objetivo principal. Se puede proceder a la neutralización de defensas exteriores por nuestros propios medios o solicitando apoyo (del tipo artillería o aviación). El objeto de la fase debe tener como consecuencia el minar la voluntad de resistencia del defensor.

El asalto consistirá en acciones de fuego y movimiento dependiendo de la zona urbana. En concreto, *la incursión*, considerada como un posible procedimiento del asalto, debe ser una acción audaz y rápida, por medio de la infiltración y posterior exfiltración. Tal y como en la simulación, un ejemplo de asalto puede ser la incursión cuya finalidad sea la recuperación de prisioneros. [17]

2.2.6.2 Progresión y limpieza

Una vez resuelta la fase de asalto, se procede a la progresión y limpieza de las zonas interiores. Durante la progresión se debe buscar la rapidez en el avance hacia los objetivos asignados. La limpieza es un conjunto de procedimientos, metódicos, rigurosos y coordinados. En esta fase en concreto, cobra especial relevancia la ausencia de errores o ambigüedades. Se ejecutarán preferiblemente de día. La unidad puede dividirse en fracciones, con áreas de responsabilidad y acción en específico para cada una.

Los equipos y sistemas para utilizar deben considerar sobre todo la utilización de armas de corto y muy corto alcance. En concreto, el cuerpo de Infantería de Marina, para la realización de esta clase de operaciones equipa el subfusil de asalto HK G36KV, que es la versión de exportación del G36K (Kurz -Corta). Esta a su vez es la variante corta del HK G36 (fusil de reglamento de las Fuerzas Armadas). La Figura 2-11muestra la diferencia entre los dos modelos mencionados. [21]

Figura 2-11 Comparación entre el subfusil de asalto HKG36 en su versión de exportación y su versión corta [17]

Es muy importante buscar siempre la superioridad de fuerzas respecto al enemigo, dada la efectividad que este puede alcanzar en su defensa de zonas urbanas con escasos medios. La Figura 2-12 muestra la relación de efectivos a emplear en función del tipo de edificio donde se desarrolle el asalto y la defensa del enemigo.

PROPORCIÓN RELATIVA DE FUERZAS ATAQUE A ZZUU	
Tipo edificio	**Ataque / Defensa**
+ 6 plantas	+12/1
4-6 plantas	10/1
3 plantas	8/1
2 plantas	7/1
1 planta	6/1

Figura 2-12 Captura extraída de la publicación de Defensa PD4021 "Empleo de pequeñas unidades en ambiente urbano"

En concreto, durante nuestra simulación se ejecutará el asalto a un edificio de una sola planta por lo que la situación ideal implicaría la disponibilidad de seis operativos por cada adversario. Debido a las limitaciones de espacio y equipos, el presente trabajo de fin de grado se centra en desarrollar la simulación de un solo componente de la unidad, sin embargo, al final de esta memoria se propone como futura línea de investigación el completar el proyecto con la integración de varios usuarios en línea, con el fin de poder simular una unidad completa y mejorar el adiestramiento en un ámbito más realista.

Como apunte final al procedimiento, hay que destacar la importancia de la consolidación de la posición. Siempre posterior a la limpieza y una vez alcanzado el objetivo principal del asalto se procederá a la reorganización de la unidad y se adoptará el dispositivo conveniente para permitir la defensa frente a un posible contrataque. [17]

Hasta aquí se cree conveniente abordar las cuestiones teóricas de ámbito militar. Las explicaciones anteriores suponen sólo un acercamiento muy somero e incompleto de toda la doctrina de difusión pública que rige los procedimientos que se pretenden ejercitar en el presente Trabajo de fin de grado, sin embargo, sirven como acercamiento a las nociones primarias con las que adentrarse en la simulación.

2.3 Del ámbito civil

Hasta ahora, los conceptos abarcados estaban orientados al aspecto militar. En el presente subapartado se pliegan las nociones propias del ámbito general o civil que se consideran oportunas antes de dar comienzo al desarrollo

2.3.1 Experiencias inmersivas

Entre los objetivos a alcanzar está el lograr que la experiencia se sienta lo más realista posible. Recrear una localización fielmente, hacer que el jugador reproduzca los movimientos esperados, conseguir la máxima fluidez posible en la interacción jugador-visor y además generar sensaciones parecidas a las que durante una situación real pueden experimentarse no es algo sencillo. Todo es un conjunto, donde hay que cuidar los detalles con tal de que nuestro cerebro al procesar la información que le vamos a transmitir durante la simulación sea capaz de asimilarlo sin sobresaturarlo ni llevarlo a equívocos. En definitiva, el primer aspecto a conseguir debe ser lograr la máxima *inmersión* posible.

La experiencia inmersiva consiste en recrear un ambiente y un entorno donde el espectador recibe de manera constante un cúmulo de sensaciones a través de varios sentidos simultáneamente (el sentido de la vista y el oído son los más importantes). De lo que se trata es de conseguir transportar al público a un estado de percepción similar al de su vida corriente. [22]

Las experiencias inmersivas en la actualidad han alcanzado tal grado de realismo que su empleo empieza a formar parte de nuestras vidas. Son realmente útiles y en algunos ámbitos han llegado incluso pasado a ser indispensables. Como lo es en el caso de los pilotos de Fórmula 1, los cuales hoy en día antes de cada carrera pasan horas adiestrándose en el simulador (Figura 2-13), potenciando sus capacidades y memorizando los circuitos en los que posteriormente correrán en la vida real. [23]

Figura 2-13 Uno de los simuladores de conducción de la escudería de Ferrari [24]

Actualmente existen numerosas competiciones a nivel profesional que enfrentan a jugadores de todo el mundo. Estos competidores en muchos casos son jugadores expertos en la simulación de deportes. El fenómeno reúne ya un amplio abanico de disciplinas que van más allá de los deportes. También existe una modalidad que practica la simulación militar (*milsim*, *militar simulator*, Figura 2-14). [25]

En este proyecto, se realiza un acercamiento a las experiencias inmersivas haciendo uso de la realidad virtual.

Figura 2-14 La modalidad que practica la simulación militar es conocida como *milsim* [25]

2.3.2 La tecnología de la Realidad Virtual

Dentro del marco de las experiencias inmersivas, desde hace una década aproximadamente, una tecnología ha advertido un rápido progreso. Se trata de la Realidad Virtual (RV).

La RV es un entorno de apariencia real, generado mediante tecnología informática, que provoca en el usuario la sensación de estar inmerso en él. Esta definición asienta esta tecnología como una experiencia inmersiva, pero es importante apuntar que en la actualidad para poder hacer referencia al término de Realidad Virtual es necesario contemplar el entorno generado a través de dispositivos conocidos como gafas o cascos de Realidad Virtual.

2.3.2.1 Evolución de la RV

Esta tecnología a pesar de su carácter llamativo y futurista no tiene un origen tan reciente como podríamos pensar. Uno de los primeros dispositivos de realidad virtual fue la denominada Sensorama, la máquina con asiento que en los años 50 ya era capaz de reproducir películas en 3 dimensiones, además emanaba olores y generaba vibraciones en el propio asiento. A partir de ahí el desarrollo tecnológico y sobre todo la llegada del software han traído importantes evoluciones hasta el día de hoy. [26]

A principios de los 60 empezaron a desarrollarse gafas más parecidas a las actuales. Estos dispositivos mucho más limitados y menos complejos consistían en gafas que utilizaban una pantalla y un sistema magnético básico para obtener la posición de la cabeza del jugador (headtracking), todo conectado a un CCTV (circuito cerrado de televisión). Si bien tuvieron ciertas aplicaciones militares como la inspección de zonas peligrosas que no se podían visitar de cerca ni personalmente, no fue hasta el año 1982 que las gafas no entraron en el terreno de los videojuegos. Entonces la compañía SEGA, lanzó lo que por aquel entonces era lo más popular: una máquina recreativa llamada SubRoc-3D (Figura 2-15). Con esta recreativa los jugadores se ponían al mando de un submarino que podía enfrentarse a barcos enemigos. [27]

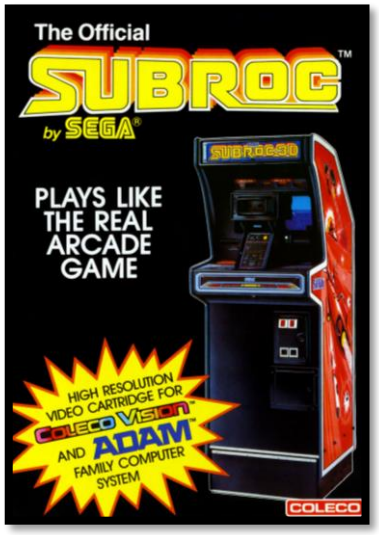


Figura 2-15 Subroc, la primera recreativa que usaba gafas de Realidad Virtual [26]

En 1984 se lanzaron al mercado las gafas llamadas RB2, aunque su elevado precio de 100.000$ limitaba mucho su comercialización. Con este objeto se podía interactuar con objetos virtuales por medio de unos guantes que estaban conectados a las gafas. El año 1991 supuso toda una revolución para el mundo de las experiencias inmersivas. La llegada de CAVE (Cave Assisted Virtual Enviroment), el entorno virtual creado por los científicos de la Universidad de Illinois en Chicago introducía a los usuarios en una sala en forma de cubo en la que diferentes proyectores junto con un juego de espejos de localizaciones y orientaciones perfectamente diseñadas dotaban al entorno de un realismo bastante logrado. Además, se usaban unas gafas especiales para visualizar los objetos virtuales en 3 dimensiones, se podía pasear y ver estos objetos desde cualquier perspectiva. [28]

El siguiente gran paso y el último hasta la fecha fue en 2009, cuando un Kickstarter que recaudó 2.437.429 $ permitió a su creador desarrollar las Oculus Rift (Figura 2-16). Estas gafas inspiraron y

fueron el detonante para que las grandes compañías pusieran el ojo de nuevo en la realidad virtual. Sony o HTC son dos de las grandes compañías tecnológicas que apuestan por el desarrollo de gafas de RV, sorprendiendo año tras años con novedades que nos acercan al futuro de estas experiencias. [27]

Figura 2-16 Oculus Rift, las primeras gafas de RV accesibles para todo el mundo [26]

2.3.2.2 Introducción al concepto de Realidad Aumentada

Este concepto de Realidad Virtual sin embargo puede no ser del todo cercano para todo el mundo, y por ello es bastante común confundirlo con un concepto de nombre parecido pero diferente significado. La Realidad Aumentada (RA) es distinta a la Virtual, pues la Aumentada es la tecnología que logra convertir nuestro mundo en el soporte para colocar objetos o imágenes virtuales mientras que la Realidad Virtual como ya hemos explicado es la que construye el mundo artificial en el que nosotros nos sumergimos. Además, en la RA no serían estrictamente necesarias las gafas o los cascos para poder ver los objetos generados. Como ejemplo, desde un teléfono móvil (Figura 2-17) se podrían visualizar sin problema estos objetos. También existe una combinación de ambas tecnologías, la denominada Realidad Mixta, que permite ver objetos virtuales en el mundo real, a la vez que estamos inmersos dentro de una experiencia en realidad virtual. [29]

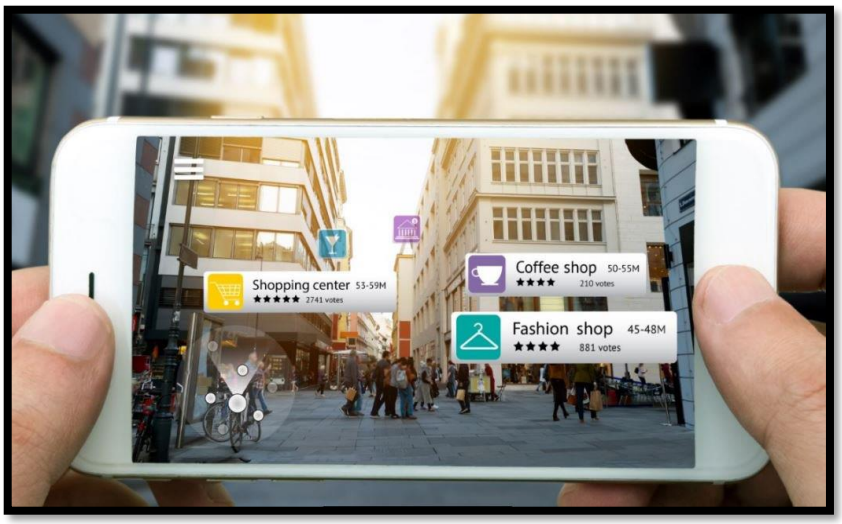


Figura 2-17 La Realidad Aumentada, una tecnología que cambia el mundo tal y como lo vemos [28]

2.3.2.3 Causas y tratamientos del "motionsickness"

La cantidad de información que nuestro cerebro procesa durante una simulación es muy elevada. Al someterlo a una sensación novedosa, este trata de adaptarse y hay veces que los efectos pueden desajustar su percepción de la situación. Es entonces cuando podemos experimentar un aspecto negativo de la realidad virtual y que se cree conveniente explicar en la presente memoria. El motionsickness o cinetosis es la sensación de mareo, angustia o náuseas que podemos sufrir tras un tiempo de uso de los dispositivos de realidad virtual (Figura 2-18). Este estado está provocado en esencia, por la sensación de movimiento que generamos cuando nos desplazamos virtualmente, pero sin embargo nos encontramos parados en el mundo real. La aceleración y desaceleración lineal y angular provocan que nuestro aparato vestibular sufra alteraciones que el cerebro no llega a entender, tal y como podemos experimentar por ejemplo durante un viaje en coche o en un barco. [30]

Figura 2-18 Uno de los síntomas más comunes de la RV: el "motionsickness" [30]

Para mitigar los efectos del motionsickness es conveniente dar unos pasos alrededor de nuestro espacio libre, para que nuestro cerebro perciba que nos movemos en el mundo real. Sin embargo, si la sensación es persistente, se recomienda realizar descansos en el uso de las gafas de RV e incluso finalizar la simulación. Al ser conscientes de que estos efectos son frecuentes, el sistema de movimiento desarrollado en el proyecto trata de minimizar los posibles detonantes del motionsickness, haciendo que el mismo jugador se desplace casi constantemente por la habitación o el espacio en que se encuentre. [31]

2.3.2.4 Aplicaciones de la RV

Los negocios, la industria, el gobierno, los museos, las instituciones didácticas e incluso el mundo de la medicina han empezado ahora a destapar las ventajas y las capacidades de esta tecnología que augura ser el futuro de la investigación, la exploración, la ingeniería, el diseño y el marketing.

2.3.3 Inteligencia artificial

Un aspecto sustancial por tratar en el simulador es el desarrollo de la Inteligencia Artificial.

La naturaleza misma presenta muchos casos y tipos de inteligencia, siendo ésta no sólo una función propia del ser humano. Ni siquiera es exclusiva de los seres vivos. La inteligencia, al ser la capacidad de un sistema de adaptarse y ofrecer respuestas oportunas a su entorno, podría incluirse en casi cualquier cosa que cumpla con esta condición. En este caso, la inteligencia artificial, término acuñado por John

McCarthy en 1956, abarca un dominio tan basto y cubre tantas técnicas que existen debates acerca de lo qué es en sí y sobre su propia definición. Lo que parece común a la mayoría de los razonamientos es que para que un sistema pueda llegar a considerarse como dotado de inteligencia artificial es necesario precisamente, y, sobre todo, que dé la impresión de ser inteligente, ya sea mediante la resolución de problemas o la propia imitación del comportamiento humano. [32]

En este proyecto se puede comprobar cómo somos capaces de desarrollar sistemas dotados de Inteligencia Artificial. Mediante el uso de técnicas de inteligencia artificial, se ha dotado a un sistema (en este caso a una representación virtual) la capacidad de tomar decisiones y reaccionar correctamente a las interacciones o estímulos que le rodean desde su propio entorno (en este caso el entorno es otra representación virtual).

Los NPC (Non-Player Character, Figura 2-19) son los personajes manejados en todo momento por la máquina y, por lo tanto, dotados en mayor o menor medida de la ya comentada inteligencia artificial. Gracias a herramientas de implementación como RNA (Redes Neuronales Artificiales), máquinas de estados, agentes softwares inteligentes, la minería de datos y el aprendizaje junto a los importantes árboles de decisión logramos que estos personajes a los que no podemos controlar tengan comportamientos lógicos y racionales en cuanto a movimiento, objetivos y estrategia. La finalidad, de nuevo, es obtener una representación lo más fiel posible a la realidad. [33]

Figura 2-19 Los NPCs del videojuego Grand Theft Auto considerados como los más complejos a nivel comercial [32]

2.3.4 Software

En este bloque se analizan los distintos programas utilizados para lograr el objetivo del presente trabajo de fin de grado. Todos son de fácil instalación, aunque su exigencia de procesado puede no resultar del todo asequible para el común de los ordenadores. Por esto mismo a continuación se listan las especificaciones del ordenador con el que se ha trabajado y cuyo rendimiento ha resultado suficiente a modo de referencia. La Figura 2-20 contiene la lista de especificaciones del ordenador proporcionado

por el Centro Universitario de Defensa, del que cabe destacar el procesador AMD Ryzen 7 2700X que ha resultado ideal para sacarle el máximo partido a las aplicaciones de diseño y renderizado :

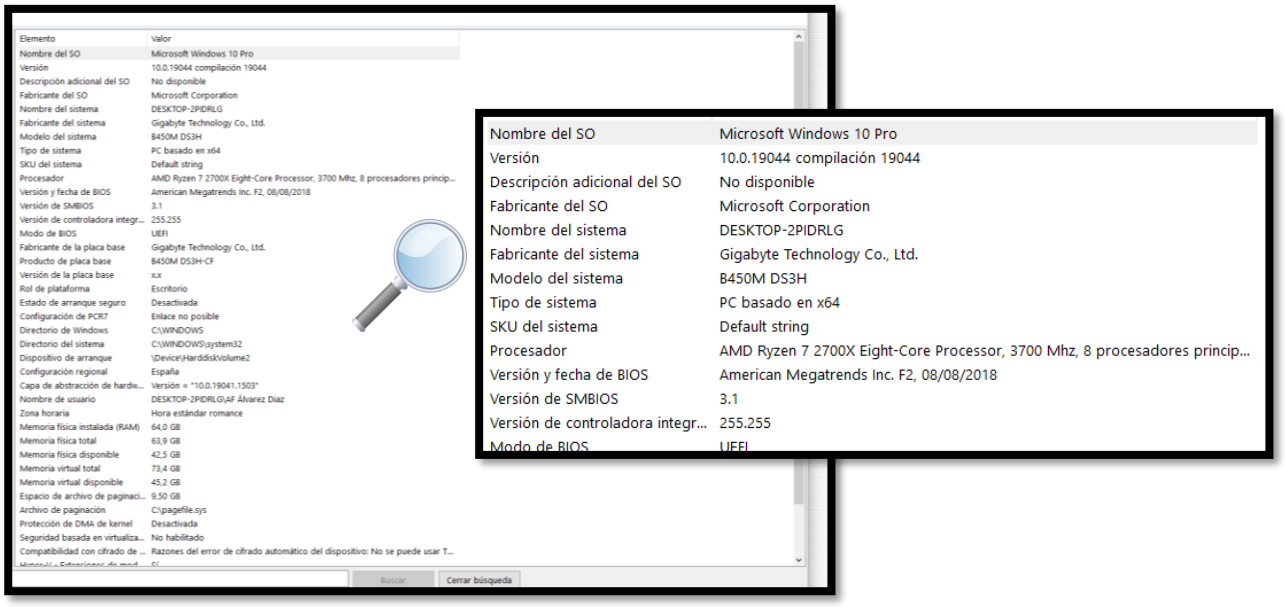


Figura 2-20 Especificaciones del PC utilizado

2.3.4.1 Epic Store

La principal herramienta utilizada durante el desarrollo será un motor gráfico. Actualmente, el motor gráfico que mejores prestaciones presenta está disponible para su descarga de forma gratuita desde la Epic Store. Esta "tienda" virtual, pertenece a la compañía Epic Games, la desarrolladora de videojuegos fundada en 1991 por Tim Sweeney, un joven ingeniero mecánico que soñaba con hacerse un hueco en la industria. Actualmente, Epic Games es la creadora de franquicias de juegos tan conocidos como Fortnite o Gears of War. Además, ha lanzado al público su propio motor gráfico de nombre Unreal Engine y que utilizaremos en este proyecto. [34]

2.3.4.2 Unreal Engine v4.27.2

El motor gráfico es un concepto que nace a mediados de los años noventa, en pleno auge de los juegos en primera persona de disparos (FPS, First Person Shooter). El juego que revolucionó la industria de videojuegos "DOOM" fue diseñado con una arquitectura perfectamente definida. El código de este videojuego separaba claramente todos los componentes del núcleo del motor, los recursos artísticos del juego, los mapas diseñados, así como las mecánicas y las reglas. La valoración de esta separación fue bien recibida por los desarrolladores que por fin encontraron una forma de crear juegos partiendo de una estructura básica, y a la que solo debían de incorporar sus propios recursos e innovaciones. En el año 1998 Epic Games decidió lanza su motor gráfico Unreal Engine, pero no sería la única compañía, Valve otra compañía de desarrollo de videojuegos lanzó el motor Source en el 2004. Más tarde vendrían nuevos lanzamientos, hasta la actualidad donde los tres motores más famosos son Unity de Unity Technologies, CryEngine de la empresa Crytek y el propio Unreal Engine. [34]

Una de las ventajas de Unreal Engine es que es de código abierto, es decir, cualquiera, desde un usuario hasta una compañía, puede mejorarlo y adaptarlo a sus propias necesidades. Además, destaca por poseer los sistemas de iluminación más avanzados y de disponer de una versión para su descarga gratuita. Como inconveniente quizás resulte un programa complejo a ojos del público inexperto en la materia, su curva de aprendizaje es pronunciada y es necesario tener conocimientos de C++ para sacarle el máximo partido. Aun así, sigue siendo una herramienta más que recomendable para desarrollar desde

videojuegos o simulaciones hasta renders de arquitectura. El cine también ha utilizado este motor para realizar todo tipo de animaciones y efectos especiales. En mayo de 2020 Epic Games anunció la quinta versión de Unreal Engine, incorporando importantes novedades como la tecnología Lumen (iluminación global en tiempo real) y Nanite (sistema de geometría micro poligonal que permite diseñar objetos virtuales con miles de detalles geométricos sin mermar la carga ni la calidad). Sin embargo, para el presente trabajo de fin de grado se ha utilizado la versión 4.27.2 la cual es considerada la versión más estable hasta la fecha. [35]

2.3.4.3 Codificación mediante Blueprints

Antes de la llegada de los motores gráficos el desarrollo de videojuegos era una ardua tarea que requería de amplios conocimientos de programación. Hoy en día las cosas son distintas, gracias a las herramientas que los motores otorgan a los usuarios y que facilitan ampliamente la tarea del desarrollo. Si bien antes era necesaria la programación de códigos en bruto, ahora una interfaz muestra los elementos que el usuario tiene a su disposición para modificar el código internamente sin tener siquiera que acceder al mismo.

Unreal Engine pone a disposición un sistema de scripting visual comúnmente llamado Blueprints que permite crear y modificar el código de una forma sencilla y visual, sin llegar a sustituir del todo la programación en C++, ya que ambos sistemas están perfectamente integrados y coordinados entre sí (Figura 2-21). Mediante la distinción de distintos tipos de Blueprints conectados entre ellos con nodos podemos obtener en poco tiempo los mismos resultados que con una codificación compleja. [36]

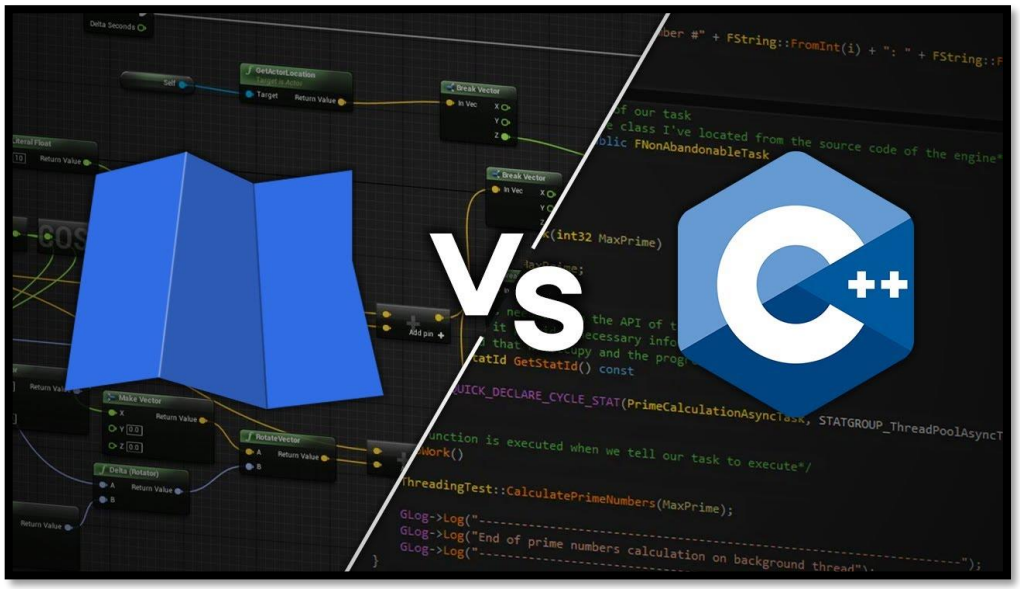


Figura 2-21 El sistema de scripting visual Blueprints facilita la codificación en C++ [36]

2.3.4.4 Steam VR

Para poder adaptar el uso de las gafas de realidad virtual a un equipo personal es necesario contar con la plataforma Steam VR. Esta plataforma lanzada en 2014 por Steam, propiedad de Valve, se trata de una potente herramienta que permite a los usuarios experimentar el contenido de RV en el hardware que estos elijan. Por ejemplo, en nuestro caso, las gafas utilizadas son las HTC Vive, sin embargo, si quisiéramos cambiarla por otra cualquiera como pueden ser las Oculus Rift gracias a esta plataforma no tendríamos problema. Además, facilita el acceso a los juegos de RV que ofrece el mercado e permite interactuar con el escritorio de nuestro PC en cualquier momento. [37]

2.3.4.5 Autodesk Inventor

En los videojuegos o en las simulaciones el diseño de los elementos con los que se interactúa es sin ninguna duda, una parte fundamental y más si entre los objetivos está lograr alcanzar el mayor grado de realismo posible, como lo es en el caso del presente trabajo de fin de grado. La base del modelado se fundamenta en el control milimétrico de todos los elementos que afectan a la geometría (texturas, topología, normales, grupos de suavizado, etc.) para obtener la mayor calidad visual con el menor costo posible de procesamiento del equipo.

Autodesk Inventor es un software CAD 3D (Computer-Aided Design o diseño asistido por ordenador) que se utiliza para el diseño, la renderización y la simulación de productos. Tiene una gran cantidad de funciones como el diseño paramétrico de piezas y ensamblajes, simulación, automatización, bibliotecas de elementos normalizados y bocetaje.

En nuestro caso, se ha utilizado puntualmente dicho software en su versión más actual (Autodesk Inventor Professional 2022, Figura 2-22) para el diseño del fusil de asalto HK G36 buscando el máximo grado de semejanza respecto al real.

Figura 2-22 Autodesk Inventor, programa utilizado para el diseño del fusil HKG36

2.3.4.6 Autodesk 3DS Max

El modelado 3D es el proceso de desarrollo de una representación matemática de cualquier objeto tridimensional a través de un software 3D. En concreto, nuestro motor Unreal Engine solo admite modelos en formato .fbx por lo que para convertir el diseño del fusil y su posterior renderizado se ha hecho uso del software 3DS Max 2022 de Autodesk (Figura 2-23). Además, este Software permite crear animaciones a través de un conjunto de herramientas flexibles y eficaces que también se han utilizado en el presente trabajo de fin de grado. [38]

AUTODESK®
3DS MAX®

Figura 2-23 Autodesk 3DS MAX, programa utilizado para el modelado del fusil HKG36

2.3.5 Equipos

El hardware (conjunto de elementos físicos o materiales que constituyen un sistema) requerido para la realización del simulador está constituido principalmente por dos equipos: un ordenador de uso personal (PC) y un sistema de periféricos que conectados al mismo ordenador ofrecen la experiencia de la RV. Del ordenador es conveniente especificar la tarjeta gráfica, como elemento a destacar debido a su relevancia en cuanto al procesamiento de proyectos de esta índole. Para el equipo de RV, se ha contado con las gafas de RV HTC Vive.

2.3.5.1 NVIDIA 2080 Super

Una tarjeta gráfica o tarjeta de vídeo a grandes rasgos es un componente electrónico integrado en la placa base que se encarga de procesar la información que llega al dispositivo para después mostrarla generalmente a través de un monitor haciendo que el usuario pueda ver la interacción con el equipo en tiempo real. Los videojuegos y la edición de vídeo son las actividades que más requieren de las tarjetas gráficas debido a que es donde más potencia de cálculo se pone en funcionamiento. Por ello, para obtener la mayor tasa de rendimiento posible, en el presente trabajo de fin de grado se ha optado por la utilización de una tarjeta gráfica de suficiente capacidad. La Nvida 2080 Super (Figura 2-24) es una tarjeta gráfica de alta calidad, con una de las memorias más rápidas del mercado (15,5 Gbps). [39]

Figura 2-24 Geforce RTX 2080 SUPER, tarjeta gráfica instalada en el equipo [39]

Lo ideal en cuanto a jugabilidad, siempre es obtener el mayor refresco de imágenes en el menor tiempo posible. A este refresco de imágenes por unidad de tiempo se le conoce como fotogramas por segundo (FPS). A mayor número de FPS más rápido será el procesado y mayor será la fluidez de nuestra interacción con lo virtual. En la Figura 2-25 se muestra la relación entre los FPS y el nivel de experiencia en cuanto a jugabilidad que ofrecen según su valor:

FRAMES POR SEGUNDO	
Frames por Segundo (FPS)	Jugabilidad
Menos de 30 FPS	Limitada
30 ~ 40 FPS	Jugable
40 ~ 60 FPS	Buena
Mayor de 60 FPS	Bastante Buena o Excelente

Figura 2-25 Relación entre la experiencia de juego esperada y del número de FPS [38]

Para tener una referencia, la Nvidia 2080 Super utilizada no ha presentado ningún inconveniente en el procesamiento. La tasa de FPS durante las pruebas y testeos de la simulación siempre se ha mantenido entre los 40 o más FPS. Ha mostrado en todo momento unas prestaciones estables y fluidas que han permitido el manejo y la interoperabilidad entre los distintos softwares de diseño y el renderizado de calidad exigido por el motor gráfico. Además, como todas las GPU de su clase al tratarse de una RTX, cuenta con el soporte para el ray tracing en tiempo real. El ray tracing o trazado de rayos es una técnica de iluminación de máxima calidad. Con esta tecnología se pueden obtener luces, reflejos y sombras ultrarrealistas. En la se pueden apreciar los sobresalientes resultados que ofrecen las tarjetas gráficas que cuentan con la tecnología ray tracing. [40]

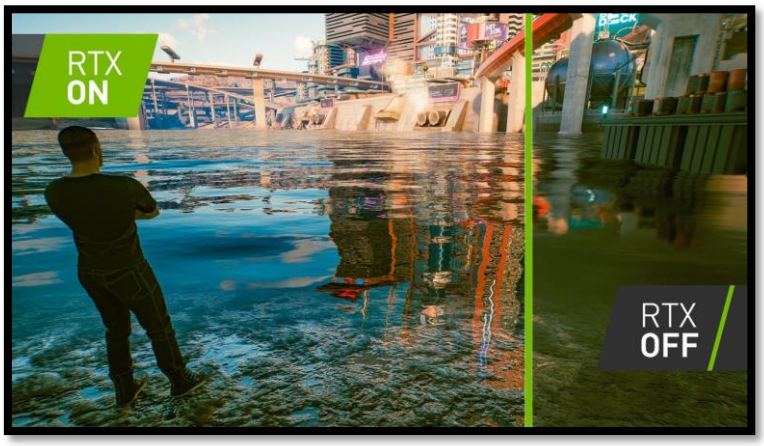


Figura 2-26 Diferencia entre un procesado con Raytracing (RTX) y sin él [39]

2.3.5.2 HTC VIVE PRO

2.3.5.2.1 Justificación de empleo

Para alcanzar la experiencia en RV resulta necesario contar con un equipo adecuado para ello. Dentro de las alternativas actuales del mercado encontramos dos tipos bien diferenciadas. Una ofrece la experiencia inmersiva, pero desde un punto de vista físico inmóvil, como es el caso de las gafas GearVR de Samsung o las Oculus Rift. Con estos equipos podemos admirar el entorno virtual desde cualquier dirección, pero presenta la limitación de que carece de reconocimiento de movimiento, o seguimiento de manos. La segunda opción, ofrece una experiencia distinta y aporta la novedad de integrar el seguimiento continuo del usuario, así como la validación de los movimientos y gestos que este realiza con sus manos sobre los mandos. Gracias a esto, se logra reflejar el movimiento tanto de nuestro cuerpo,

en cuanto a dirección y movimiento se refiere, así como el de nuestras manos y las acciones de estas mismas, como el agarre o el disparo de un arma.

Las HTC Vive utilizadas, pertenecen a esta segunda clase de gafas de RV. Para muchos estas gafas son las mejores del mercado a nivel usuario debido a sus altas prestaciones, a su versatilidad y la comodidad de uso que ofrece. [41]

2.3.5.2.2 Elementos periféricos

A continuación, se describen los elementos periféricos y las conexiones de las que dispone (Figura 2-27). Posterior a ello, es conveniente también realizar una breve puntualización sobre la disposición de los sensores para su máximo aprovechamiento.

1. Gafas HTC Vive con cable de alimentación y conexión al PC mediante HDMI y USB de gran longitud, aproximadamente 5 metros.

2. 2 unidades de sensores de posición o base station con cables de alimentación. Para colocarlos se deben situar de tal manera que tengan visión directa entre ellos a una altura mínima por encima de la cabeza del jugador y en esquinas opuestas del espacio disponible al que llamaremos “zona de juego”. La zona de juego mínima debe de ser de dimensiones 2x1,5 m y la máxima tendrá una diagonal (distancia en línea recta desde un sensor de posición a otro) de 5 metros. También existe la posibilidad de usar un cable de conexión directa entre ambos sensores para mejorar la transmisión de datos.

3. 2 mandos inalámbricos con cargadores vía microUSB. Estos elementos serán descritos en el siguiente apartado de forma detallada.

4. Link box o caja de conexiones. Se encarga de conectar las gafas al ordenador. Viene con enchufe de alimentación y su adaptador de luz, un cable HDMI/Display y un cable USB para PC .

Figura 2-27 Elementos periféricos del HTC VIVE

2.3.5.2.3 Button mapping HTC VIVE controller 2.0

Este apartado está orientado a familiarizarse con la nomenclatura de los diferentes botones, sensores y elementos de los mandos. Esta nomenclatura es considerada de necesario conocimiento para la lectura de la presente memoria. A este proceso de aprendizaje, asimilación y reconocimiento del funcionamiento de un mando se le conoce como button mapping. [42]

En nuestro caso contamos con la versión 2.0 del mando HTC VIVE Controller (Figura 2-28). Esta versión totalmente inalámbrica lo convierte en un dispositivo innovador y fácil de usar. Cuenta con una batería recargable de 960 mAh. A continuación, se muestra el button mapping de los mandos:

1. *Sensores integrados*. Junto a las bases station del equipo realizan la función de detectar en todo momento la posición y orientación de las manos del jugador. Su seguimiento del movimiento es sumamente preciso.

2. *Trackpad multifunción*. Ofrece mucha precisión sin esfuerzo gracias a la retroalimentación táctil de alta definición. En nuestro caso servirá para controlar ciertos aspectos del movimiento del pesonaje virtual.

3. *Botón del sistema*. Su función es abrir el menú virtual del HTC VIVE desde el que podemos controlar y configurar diferentes aspectos del mando como la sensibilidad, el reconocimiento de los movimientos o cambiar la configuración de los botones mismos.

4. *Botón de menú*. Su función en la mayoría de las aplicaciones y juegos es la de abrir menús de interacción, sin embargo, su función es configurable desde el motor gráfico.

5. *Disparadores duales o triggers*. Su función en la mayoría de los casos es la de simular el disparador de un arma, aunque también es utilizado para conseguir agarrar y sostener objetos dentro del entorno virtual. Cuenta con seguimiento continuo de la posición del dedo en función de la presión que este ejerza sobre el trigger.

6. *Botón de agarre*. Conocido por varios nombres en función del cometido en específico que este tenga asignado dentro de la simulación (grip,handle o grab button). En nuestro caso, será utilizado principalmente para sostener el arma en todo momento.

7. *Correa ajustable*. De uso recomendable para evitar la caída de los mandos al suelo ya que estos son sensibles y pueden verse dañados con facilidad.

Figura 2-28 Mandos HTC VIVE Controller 2.0

2.3.5.3 Adaptador Beswin VR M4

El laboratorio de investigación del Centro Universitario de la Defensa cuenta con un dispositivo al que se le pueden acoplar los controladores duales del HTC Vive para simular el uso de un fusil real. El acople de los controladores al fusil es manual y no requiere de ningún tipo de configuración electrónica para su utilización, ya que solo es una plataforma mecánica destinada a disminuir los efectos de alta sensibilidad que presentan los controladores. La Figura 2-29 muestra como quedarían instalados los mandos sobre el dispositivo. [43]

Figura 2-29 Adaptador Beswin VR M4 utilizado [43]

2.3.6 Recorrido histórico: juegos de temática militar

Por último, antes de comenzar con el desarrollo propiamente dicho de este simulador de combate, es importante repasar el contexto histórico que nos ha llevado hasta aquí, hasta este momento en el que los videojuegos y las simulaciones han tomado protagonismo y han demostrado ser una excelente herramienta en múltiples ámbitos. Si bien el presente trabajo de fin de grado tiene como objetivo realizar un simulador de carácter táctico y orientado al adiestramiento no cabe olvidar que este concepto surge tras comprobar las múltiples posibilidades y ventajas que los videojuegos poco a poco han ido desarrollando y poniendo a nuestra disposición. Podemos afirmar por tanto que la línea histórica recorrida por los videojuegos de temática militar está estrictamente relacionada con el campo de la simulación. Los videojuegos se enmarcarían dentro de las aplicaciones interactivas orientadas al entretenimiento mientras que las simulaciones se conciben para mejorar ciertos aspectos del usuario o la simple adaptación visual de un entorno simulado e interactivo.

Desde sus primeros pasos la industria del videojuego ha considerado el campo de lo militar como un potencial atractivo ya no solo a nivel de ventas (los conocidos comúnmente como “juegos de guerra” o “disparos” año tras año se colocan como líderes del mercado) sino también a nivel desarrollo. Los propios desarrolladores de videojuegos desde el principio entendieron que los juegos de esta clase eran fuente de inspiración para la investigación y la implementación de nuevas mecánicas jugables. De ahí que estos juegos se hayan mantenido siempre en la cúspide de una industria tan compleja y aún hoy sigan sorprendiendo al mundo entero con importantes logros. La industria militar, nunca ha sido ajena a esto, aunque ahora haya encontrado en la simulación el aliado perfecto, desde el principio también contó con los videojuegos y se sirvió de estos para dar rienda suelta a sus propias investigaciones.

Antes del comienzo de la era de los videojuegos, la guerra ya había sido representada como juego de varias maneras distintas. En 1824 el teniente Von Reisswitz del Ejército prusiano diseñó un juego de guerra para el entrenamiento de la planificación estratégica. Este juego llamado Kriegspiel (que en alemán precisamente significa “juego de guerra”, Figura 2-30) usaba mapas militares y tablas de

probabilidad. Además, se designaba un árbitro que se encargaba de velar por el cumplimiento de las reglas con el fin de simular la forma en la que resolver conflictos como si de un comité bélico se tratase. Sin embargo, el Kriegspiel no fue bien recibido por los oficiales del ejército prusiano. A pesar de su poca aceptación a nivel militar la publicación de la primera revista de juegos de guerra de nombre Kriegspieler Verein ayudó a la popularización del juego y los métodos que este proponía. A partir de ahí, muchas naciones atribuyeron al Kriegspiel el éxito en sus misiones, como el caso de la victoria de Ejército Alemán en la guerra Franco-Prusiana. Estados Unidos también se interesó por este concepto y llegaron a desarrollar su propia versión del juego que pasó a llamarse war game ("juego de guerra" en inglés). Ahora el interés generado abarcaba desde el público militar hasta el civil, por lo que los war games pasaron de ser además de herramientas para el entrenamiento militar una forma de entretenimiento lúdica y comercial. [44]

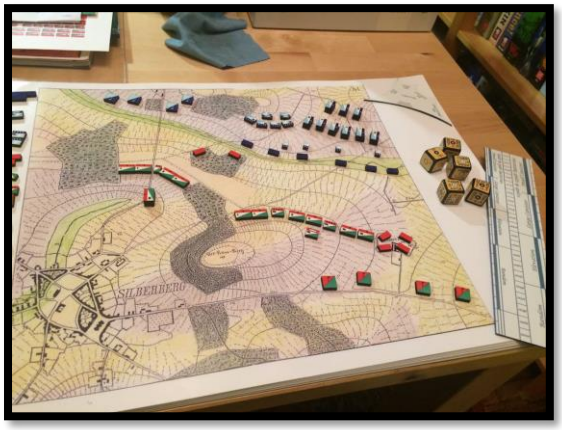


Figura 2-30 Kriegspiel el juego de mesa que entrenó a los oficiales prusianos para ganar la guerra contra Francia [44]

Durante la Segunda Guerra mundial tuvo lugar un curioso caso que ejemplifica esta relación entre juego y realidad. Los oficiales japoneses en un elaboradísimo juego de comité simularon lo que más tarde sería la conocida batalla de Midway. En el juego, el bando americano lanzó un ataque a flota de portaviones japoneses infligiendo un daño devastador, pero en mitad del juego este ataque fue cancelado, ya que el árbitro del ejército japonés consideró inviable un ataque de tal magnitud por parte de su enemigo cuando en realidad, la batalla más tarde se desarrolló exactamente como había predicho el juego. El juego La Batalla, de José Perelló Pérez fue publicado en 1944 y es posiblemente el primer juego de guerra en español diseñado por un autor nacional. [46]

En 1976 el Ejército estadounidense encargó a la empresa SPI un juego de guerra de nivel táctico que también fue vendido en el mercado civil en forma de tablero llamado Firefight (Figura 2-31). [47]

Pero para llegar al primer lanzamiento de un juego de guerra informático y accesible por todo el mundo habría que avanzar hasta el mes de febrero de 1980, cuando la compañía Strategic Simulations Inc. (SSI) lanzó al mercado su videojuego llamado Computer Bismarck. Este contaba con una jugabilidad por turnos y gráficos bidimensionales muy avanzados para la época y estaba basado en la persecución de las fuerzas armadas británicas al acorazado alemán Bismarck en el año 1941. El juego resultó todo un éxito de ventas, lo que dio pie a la aparición de nuevas compañías y desarrolladores a la carrera por lograr algo similar. [47]

En la misma década de los 80 Atari lanzó un videojuego que resultó muy popular: Battlezone. Con este videojuego podías manejar carros de combate e hizo que el Army Training Doctrine and Command de las fuerzas militares estadounidenses solicitase a Atari una nueva versión del juego, pero adaptada para simular el entrenamiento de su último vehículo de combate de infantería, el M2 Bradley. [48]

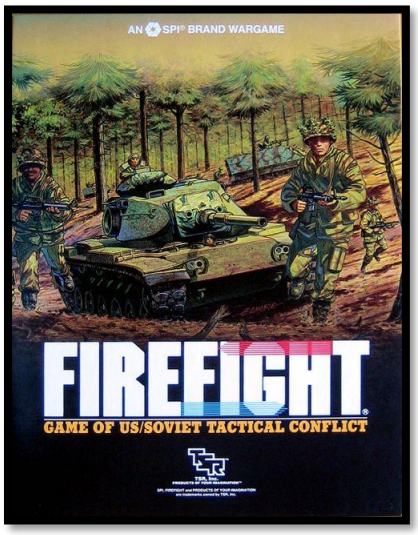


Figura 2-31 Firefight, el primer juego táctico encargado por el ejército de EEUU [47]

El siguiente paso llegó una década después, en 1993, cuando una pequeña desarrolladora de Texas, ID Software, lanzó DOOM. Este juego fue el verdadero pionero en el género de los shooters en primera persona (FPS). De nuevo el ejército estadounidense encargó un producto comercial que pudiera ser modificado a las necesidades del entrenamiento militar. Al teniente Scott Barnett se le asignó la tarea de buscar juegos de PC en el mercado que pudieran encajar, y finalmente seleccionó DOOM II, la secuela del ya mencionado DOOM. Este juego al que llamaron "Marine DOOM" pasó de ser un juego de ciencia ficción ambientado en el planeta Marte a situarse un pequeño pueblo del desierto, y sustituyó sus demoníacos enemigos por adversarios humanos (Figura 2-32). El resultado a pesar de estar lejos de ser fiel a la realidad sí que resultó ser una buena simulación de lo que supone la intensidad de un combate. Además, promovía el trabajo en equipo constante propio de las operaciones militares. Marine DOOM llegó a instalarse por defecto en todos los ordenadores de los cuarteles americanos y se animó a todos los soldados a jugarlo. En 1997, el General Charles C. Krulak, que era el comandante del Cuerpo de Marines en ese momento, emitió una directiva que apoyaba el uso de juegos de PC para "ejercicios de pensamiento y decisión militar". [49] [50]

Figura 2-32 Transformación del videojuego comercial DOOM en su versión militar [44]

Llegado el nuevo milenio, Estados Unidos volvió a implicarse en un proyecto para el entrenamiento de equipos de fuego. En colaboración con Destineer Studios, desarrollaron el videojuego Close-Combat: First to fight, que años más tarde, en 2005 concretamente, fue comercializado para todos los públicos.

Sin embargo, el proyecto militar más ambicioso y controvertido es el videojuego America's Army (Figura 2-23). Lanzado en 2002, este no fue un instrumento de entrenamiento, sino una herramienta de reclutamiento. Era un juego de disparos en primera persona multijugador en línea en el que los jugadores asumían el papel de diferentes puestos de trabajo relacionados con la infantería en el Ejército. Financiado y desarrollado por el Ejército, el juego rivalizaba con muchos de sus contemporáneos comerciales en términos de calidad. El juego también se diseñó para reforzar los valores y el entrenamiento; los jugadores tenían que completar un campamento de entrenamiento virtual y una prueba de puntería antes de entrar en línea, y los roles especializados, como el de médico o francotirador, estaban encerrados tras otros niveles de entrenamiento. El juego destacaba por el hecho de que los jugadores sólo podían jugar como el Ejército de EE. UU; el equipo enemigo siempre aparecía como una facción genérica de la "fuerza contraria".

America's Army no tardó en ser criticado por su estrategia de reclutamiento dirigida a los adolescentes; el juego pretendía que los estudiantes de secundaria pensaran en una carrera en el ejército mucho antes de cumplir los 18 años. Esta controversia no afectó a la enorme popularidad del juego y el proyecto ha continuado recibiendo un total de 41 actualizaciones desde enero de 2014. [51] [52]

Figura 2-33 El videojuego America´s Army lanzado en distintas plataformas [51]

Hoy en día, algunos de los videojuegos más populares incluyen imágenes y jugabilidad de inspiración militar también. La serie de videojuegos de disparos en primera persona Call of Duty lleva vendidas casi 400 millones de copias de sus entregas. La base de datos de juegos de Giant Bomb revela que actualmente existen casi 1.000 títulos con una ambientación "militar moderna". [53] [54]

Onward VR es un juego en Realidad Virtual lanzado en 2016, pensado para simular de manera fiel una situación de combate de manera táctica. Desde un punto de vista jugable, podría parecer un shooter multijugador más en el que los jugadores utilizan la coordinación, la comunicación y la puntería para completar los objetivos sin embargo también ofrece la posibilidad de planificar junto a todo tu equipo el ataque previamente a la operación. Cuenta con efectos meteorológicos, múltiples entornos y escenarios, así como una amplia variedad de armas a elegir. En este juego no se dispone de la ayuda del aviso de número de balas restantes que le quedan al jugador en el cargador, algo muy común en el resto de shooters. Esto está así diseñado para convertir la experiencia en algo más parecido a la vida real, donde el combatiente debe prestar atención hasta el número de cargadores disponibles que tiene y las balas que

le quedan en el mismo. Es por ello por lo que el juego es considerado como uno de los mejores simuladores militares en la actualidad. El presente trabajo de fin de grado tomará muchas de las ideas planteadas por este videojuego para su propia implementación.

3 DESARROLLO

3.1 Descripción del apartado

En este apartado se detallará el proceso de creación del presente proyecto, el cual tiene como fin último la creación de un simulador táctico de combate específicamente diseñado para los alumnos. Para lograrlo se ha aplicado la metodología propia de la industria 4.0, generando un entorno en Realidad Virtual con un alto grado de realismo.

Para empezar, se partirá desde el diseño y la elaboración del escenario en sí. En esta primera etapa del desarrollo se explicará el proceso de creación de toda la localización, el terreno simulado, las infraestructuras involucradas, así como el arma que utilizaremos. Una vez generados los elementos virtuales que necesitamos, se procederá a la programación mediante Blueprints. Esta etapa será esencial para dotar a estos elementos de toda su lógica de comportamiento. Por último, se procederá a explicar la transición realizada entre el código y su posterior visionado a través de las distintas formas de interacción con la Realidad Virtual. La Figura 3-1 muestra las distintas etapas que han conformado el desarrollo.

Todas las figuras incluidas en el desarrollo a excepción de las debidamente referenciadas son de elaboración propia.

Figura 3-1 Etapas del desarrollo

3.2 Diseño y modelado del escenario

3.2.1 Creación del proyecto

Este subapartado está destinado a entender los pasos iniciales del proyecto generado en Unreal Engine, de necesario conocimiento para el entendimiento de las explicaciones posteriores.

3.2.1.1 Selección de Categoría y Template inicial

Tras la instalación de la última versión del Unreal Engine (UE4) se ha creado el proyecto sobre el que trabajar. Este apartado inicial es muy importante ya que una correcta elección en la configuración de inicio resulta determinante y allana el terreno sobre el que empezaremos a construir el simulador. En función del tipo de proyecto que se tenga en mente escogeremos una categoría y un template de entre todas las opciones que UE4 nos ofrece.

Al iniciar Unreal Engine (UE4), aparece automáticamente el navegador de proyectos de Unreal En categorías de proyectos nuevos, se selecciona la categoría de desarrollo que mejor se adapte a nuestro sector. En nuestro caso, la categoría de desarrollo Games (Juegos) se ha considerado como la más conveniente ya que ésta está pensada para proyectos destinados a la creación de simulaciones y videojuegos controlados por una persona en tiempo real. Sin embargo, también se puede seleccionar otros tipos de categorías como Film (Cine), Television (Televisión) y Live events (Eventos en directo), Architecture (Arquitectura), Engineering, and Construction (AEC, Ingeniería y Construcción), o Automotive, Product Design, and Manufacturing (APM, Automoción, Dieño de Productos y Fabricación).

Una vez seleccionada la categoría Games se nos ofrecerá la posibilidad de escoger un template inicial. Estos templates no son más que plantillas que ofrecen un punto de partida muy básico para cada tipo de proyecto. En concreto, nuestro proyecto, como ya hemos adelantado en anteriores apartados, se basará en la utilización de la Realidad Virtual (RV). Existen un gran número de templates especialmente diseñados para la categoría Juegos, por lo que hay que especificar con cual queremos empezar. El motor gráfico ofrece un template inicial centrado en el desarrollo de Juegos y experiencias en RV. Esta plantilla aparece con el nombre Virtual Reality y sirve de gran ayuda para iniciar el proyecto con algunas de las acciones de entrada más comunes de la RV como agarrar y fijar objetos con la mano, por lo que finalmente este ha sido el template escogido para comenzar el proyecto (ver Figura 3-2).

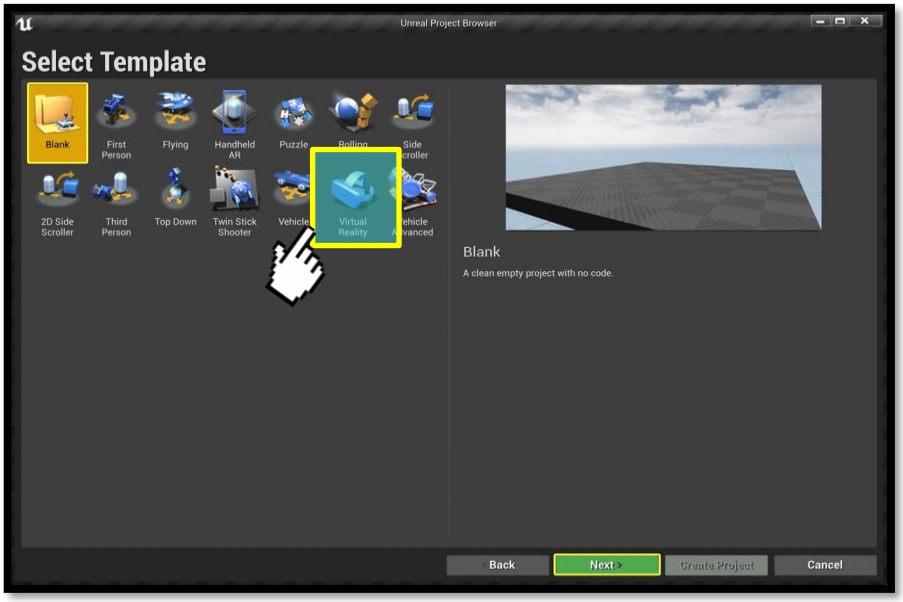


Figura 3-2 Selección del template Virtual Reality dentro de la categoría Games

3.2.1.2 Configuración inicial del proyecto

En este apartado, conocido como *project setting*, deberemos configurar algunos aspectos del proyecto relacionados con el nivel de calidad/rendimiento, la plataforma de destino, si desea incluir el contenido de inicio, etc.

En primer lugar, el menú desplegable Blueprint nos ofrece dos opciones: Blueprint o C++. Como en nuestro caso se quiere construir el proyecto en el Editor de Unreal, y utilizar el sistema Blueprint Visual Scripting para crear interacciones y comportamientos se ha escogido la opción de Blueprint. Si quisiéramos construir el proyecto programando con C++ en Visual Studio deberíamos haber escogido la opción C++.

A continuación, se ha configurado el menú desplegable que regulará el nivel de calidad con el que nuestro proyecto será procesado. Si se está desarrollando un proyecto para ser utilizado en un ordenador o consola de juegos, se debe elegir Calidad Máxima (como es nuestro caso), en cambio sí se está desarrollando un proyecto para ser visto en un dispositivo móvil, habría que elegir la opción de 3D o 2D escalable.

El siguiente desplegable marcado es el de Escritorio/Consola el cual no es necesario cambiar a Móvil/Tableta debido a que el simulador está pensado para ser lanzado desde un ordenador.

El menú Contenido inicial ofrece la opción de partir con recursos iniciales propios del motor. Este contenido inicial presenta muchas características y funciones de interés para nuestro proyecto por lo que lo dejaremos marcado con Contenido Inicial.

El último desplegable maneja la implementación del ray tracing. Como ya hemos visto, esta tecnología aporta un gran realismo a la simulación en cuanto a sistemas de iluminación se refiere, además de que el equipo disponible cuenta con la suficiente potencia gráfica para soportarlo, por lo que también se ha decidido marcar como enabled (activado).

Por último, se ha escogido donde se almacenará el proyecto y se le ha dado un nombre. El nombre del proyecto escogido será: VR_CAD_SIM haciendo referencia al simulador (SIM) en realidad virtual (RV) desarrollado por el Alférez de Fragata, D. Carlos Álvarez Díaz (CAD) y autor del presente Trabajo de Fin de Grado. Tras la elección del nombre y hacer click sobre la opción Create Project, la carpeta con todo el proyecto es automáticamente generada en la carpeta de destino previamente seleccionada.

3.2.1.3 Interfaz del editor de niveles de Unreal Engine

Tras la creación de nuestro proyecto, este se inicia automáticamente, lanzándose desde el Level Editor (editor de nivel) de Unreal Engine. Durante el transcurso del desarrollo se usarán muchas referencias visuales directamente tomadas de este editor por lo que es conveniente realizar un breve repaso de la interfaz que presenta el mismo. La distribución del level editor es el núcleo central del Unreal Editor, desde donde podemos construir los distintos niveles del juego, colocar y modificar los diferentes actores, probar la simulación, etc. Esta distribución puede ser alterada en cualquier momento o reiniciada desde el propio menú de Windows.

Además del Level Editor, existen otros editores que resuelven tareas más específicas y que se discutirán en los próximos apartados. Gracias a esta amplia variedad de editores y herramientas, UE4 contempla los diferentes escenarios que surgen en el desarrollo de un videojuego o una simulación.

La captura más presentada en la página siguiente (ver Figura 3-3) muestra lo primero que observamos al iniciar cualquier proyecto en Unreal Engine por primera vez. Aunque el motor nos permita seleccionar una versión en Castellano, ésta no afecta a los elementos de la interfaz por lo que a continuación y de ahora en adelante siempre se seguirá su terminología en inglés:

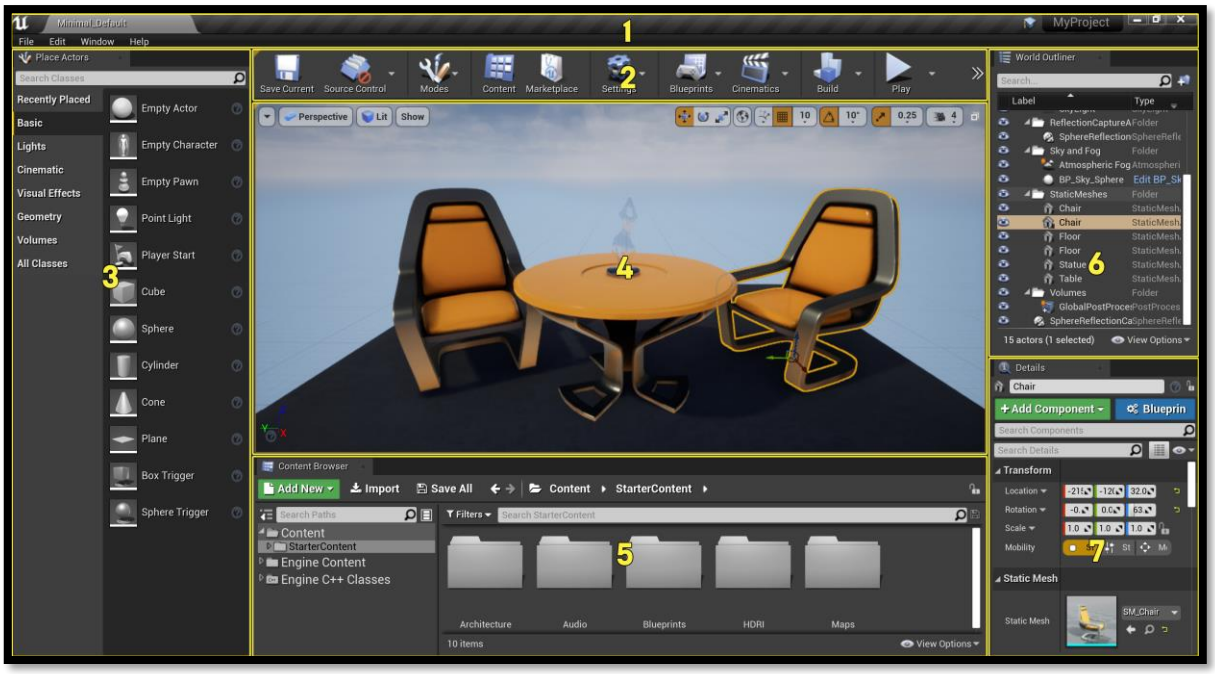


Figura 3-3 Interfaz del Level Editor de Unreal Engine y sus partes principales

1. *Tab Bar and Menu Bar*: El Level Editor tiene una pestaña en la parte superior con el nombre del nivel actual. Las pestañas de otras ventanas del editor pueden anclarse junto a esta pestaña para una navegación rápida y sencilla, similar a la de un navegador web. El nombre de la pestaña refleja el nivel que se está editando. Este es un patrón constante en todo el editor: las pestañas tendrán el nombre del activo de edición. La forma más común de navegar por este panel es mediante el botón derecho del ratón para rotar la cámara, junto a las teclas w, a, s, d para desplazarse.
2. *Toolbar*: El panel de la barra de herramientas muestra un grupo de comandos que proporcionan un acceso rápido a las herramientas y operaciones más utilizadas.
3. *Place Actor / Modes:* Lista de todos los actores y elementos que podemos incluir en nuestro proyecto. Basta con arrastrarlos desde el mismo menú hasta el viewport para que estos pasen a formar parte de nuestro nivel.
4. *Viewports:* El panel Viewport es tu ventana a los mundos que creas en Unreal Engine. Este panel contiene un conjunto de vistas, cada una de las cuales puede maximizarse para llenar todo el panel y ofrecer la posibilidad de mostrar el mundo desde una de las tres vistas ortográficas (Superior, Lateral, Frontal) o desde una vista en perspectiva, dándote un control total sobre lo que ves y cómo lo ves.
5. *Content Browser:* Es el navegador de contenidos de Unreal Engine. Desde el mismo se puede acceder a cualquier elemento que se introduzca en la simulación. Se recomienda una correcta organización de este para no presentar dificultades de búsqueda.
6. *World Outliner*: Lista de todos los elementos que en ese momento se encuentran activos sobre el nivel. Pinchando sobre cualquiera de ellos, el viewport automáticamente nos llevará hasta su posición dentro del mundo generado.
7. *Details panel:* El panel de Detalles contiene información, utilidades y funciones específicas para la selección activa del viewport. Contiene cajas de edición y de transformación para mover, rotar y escalar los actores, muestra todas las propiedades editables y proporciona acceso rápido a funcionalidades de edición adicionales dependiendo del tipo de actor seleccionado en el viewport. Este panel permite ver los materiales utilizados por los actores seleccionados, si los hay, y abrirlos rápidamente para su edición.

3.2.2 Estudio previo del terreno

El primer paso será determinar la localización que se pretende simular. Este apartado es importante no solo para lograr la máxima inmersión posible, si no como elemento crucial en el planeamiento previo al enfrentamiento y en su posterior ejecución. Las operaciones en zonas de combate urbanas a veces resultan difíciles de prever; como se ha explicado anteriormente el factor sorpresa juega un rol importante y siempre se debe contar con él cuando nos encontramos desplegados.

Un conocimiento profundo del terreno disminuirá notablemente los efectos negativos del factor sorpresa cuando este se ejerza sobre nosotros. En este proyecto también se considera este estudio del terreno y se han analizado a fondo los distintos emplazamientos donde suelen darse las situaciones hostiles que pretendemos simular. La búsqueda de entornos, el estudio de la topografía y el clima de las zona más comunes de despliegue, así como la disposición de las casas, la arquitectura y los materiales de construcción de estas junto a testimonios de personal con experiencia en este tipo de operaciones del cuerpo de Infantería de Marina han sido la fuente de información determinante para escoger el tipo de terreno a diseñar.

Como conclusión, estadísticamente, el mayor número de enfrentamientos hostiles tienden a darse en la zona occidental del globo, en poblados de pequeña extensión donde el calor, la sequedad y la topografía característica de los biomas desérticos servirán de inspiración para un diseño de nivel lo más ajustado a la realidad posible.

Las siguientes imágenes han sido tomadas como referencias visuales para comenzar con el diseño del terreno y la colocación de las infraestructuras sobre éste. Cabe destacar que el estudio previo del terreno resultará determinante a la hora de diseñar el nivel y todas las decisiones tomadas en este aspecto han tenido muy en cuenta la ambientación escogida

Figura 3-4 Militares españoles durante una misión cívico-militar en la localidad de Abzi-Khuda, en el oeste de Afganistán [55]

Figura 3-5 Chaki Wardak, poblado afgano situado a 97 kilómetros al suroeste de Kabul [56]

3.2.3 Diseño del terreno

Unreal Engine es capaz de crear mundos enormes basados en el diseño del terreno utilizando el conjunto de potentes herramientas de edición de las que dispone. La herramienta Landscape permite crear terrenos directamente desde el viewport y además los optimiza para que puedan mantener una velocidad de fotogramas jugable en una multitud de dispositivos diferentes.

Para empezar a diseñar el terreno, primero se ha creado un nuevo diseño de nivel dentro de la carpeta Content (carpeta de contenido general) del proyecto. Al hacer click derecho con el cursor dentro de la carpeta Content se pueden generar nuevos elementos como materiales, sistemas de partículas o clases de Blueprints. En nuestro caso se ha seleccionado la opción de Nivel para generar el mapa. Al nivel se le ha llamado con el nombre: Mapa_CAD.

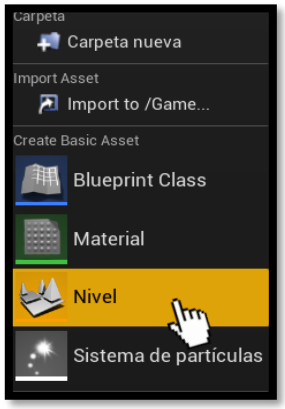


Figura 3-6 Opciones de creación de assets básicos desde la carpeta de contenido

Al generar un nuevo nivel desde cero siempre se deben añadir una serie de elementos básicos que generarán la primera sensación de estar inmersos en un mundo parecido al real. Estos elementos aportarán la simulación de la luz solar, la atmósfera, la nubosidad, el color del cielo o la cantidad de estrellas (cuando se active el modo noche).

Para añadir estos elementos bastaría con arrastrarlos desde el panel de *place actors* al *viewport*. En nuestro caso, se han añadido añadido los siguientes elementos:

Una luz direccional (actor *directional light*): simula la luz que se emite desde una fuente que está infinitamente lejos. Esto significa que todas las sombras proyectadas por esta luz serán paralelas, lo que la convierte en la opción ideal para simular la luz del sol. La luz direccional, una vez colocada, puede ajustarse a nuestro gusto en la ventana opciones.

Una luz de cielo (actor *light sky*): captura las partes distantes del nivel y las aplica a la escena como una luz. Esto significa que la apariencia del cielo y su iluminación/reflejos serán uniformes sobre todos los elementos del mapa sea donde sea que hayamos dispuesto nuestra *directional light* anterior. Sirve para que la luz direccional, cuando vaya a ser usada como luz solar, ilumine la escena tal y como lo haría en la realidad.

Niebla atmosférica (actor *atmospheric fog*): proporciona una aproximación a la dispersión de la luz a través de una atmósfera planetaria. Esto da a los niveles exteriores un aspecto mucho más realista.

Niebla de altura exponencial (actor *exponential fog*): crea más densidad en los lugares bajos de un mapa y menos densidad en los lugares altos. La transición es suave, por lo que nunca se produce un corte brusco al aumentar la altitud. También proporciona dos colores de niebla: uno para el hemisferio orientado a la luz direccional dominante (o directamente hacia arriba si no existe), y otro color para el hemisferio opuesto.

Un "*BP_Sky_Sphere*": es un Blueprint que contiene varios elementos ya programados de manera conjunta para simular rápidamente lo que sería el comportamiento de la esfera terrestre. Dentro del Blueprint podemos variar todos los parámetros, sin embargo, desde los detalles del actor BP_Sky_Sphere, en la esquina inferior derecha variaremos los que más nos interesan. Para que se nos muestre el sol en la escena solo hay que asociar este actor a la luz direccional que hemos creado con el primer elemento de la lista y cambiar su altura. Por defecto el sol aparecerá en su posición cenital, por lo que para crear un nivel nocturno solo habría que hacerlo descender en altura. Para ello hay primero que quitar la relación existente con la luz direccional. La imagen muestra un ejemplo de configuración para un nivel diurno frente a uno nocturno.

Tras aplicar todos los elementos anteriormente comentados a nuestro nivel Mapa_CAD lo dejamos configurado para que sea un nivel diurno. Más adelante, cuando acabemos de diseñar todo el nivel, tan solo habría que duplicarlo y cambiar los parámetros de la altura del sol para obtener el mismo nivel, pero de noche.

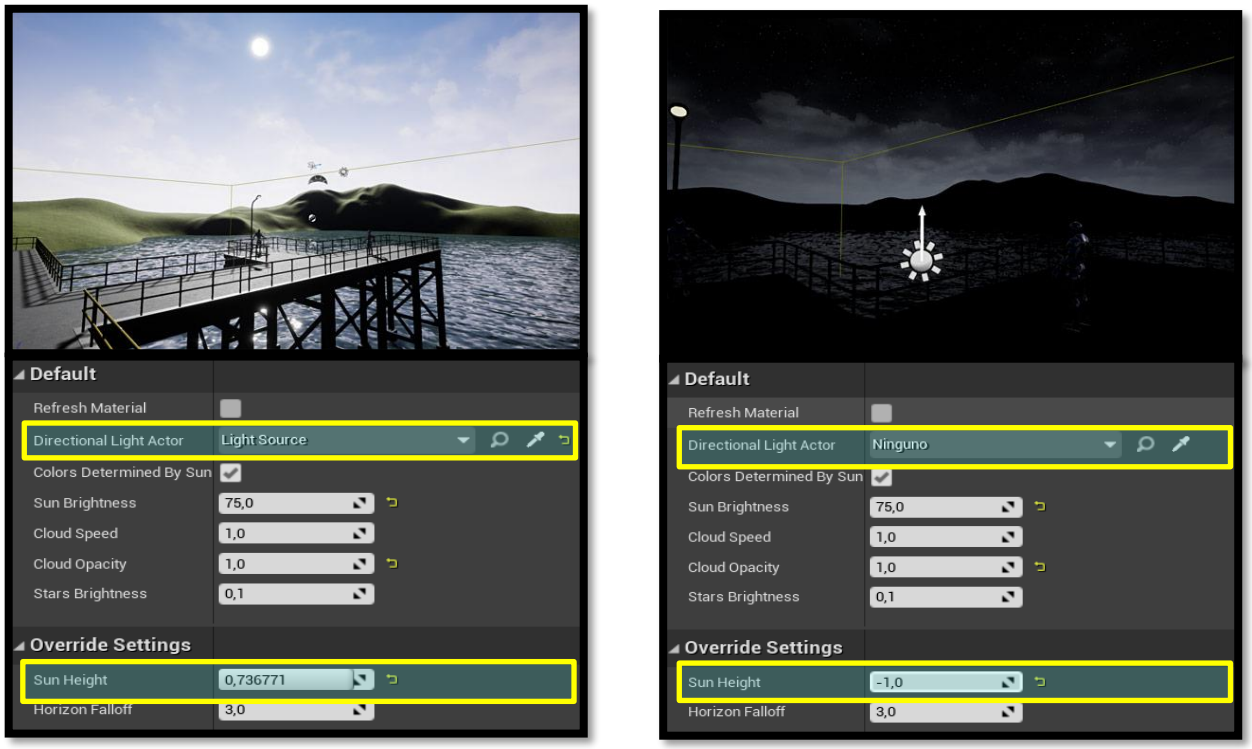


Figura 3-7 Configuración de nivel diurno y nocturno en UE4

El siguiente paso sería conseguir el terreno sobre el que queremos que se ubique toda la simulación. Para ello UE4 cuenta con herramientas de esculpido y modelado de tierras. Partiendo de una base de terreno adquirida directamente desde la Unreal Engine Marketplace, concretamente del pack gratuito Mout_Arabian, podemos aplicar un landscape que se ajusta exactamente a lo buscado. Para su implementación, sólo hay que añadir el pack al proyecto directamente desde la pestaña Biblioteca dentro de la aplicación de escritorio de la Epic Store.

Sin embargo, este terreno inicial, como vemos en la imagen superior de la Figura 3-8, podría resultar demasiado montañoso. Para nuestra simulación sería conveniente un terreno algo más plano donde poder asentar las casas y las infraestructuras. Para conseguir aplanar el terreno y hacer que siga pareciendo realista se han utilizado las herramientas de modelado del propio UE 4. Estas herramientas son muy útiles a la hora de esculpir y diseñar exteriores ya que permiten escoger el tipo de operación que queremos realizar sobre el entorno a tiempo real.

Con la ayuda de un pincel o brush que ofrece la herramienta Esculpir, dentro del modo de edición de terreno, se define el tamaño y la forma del área del paisaje que se verá afectada al esculpir o pintar. Los pinceles pueden tener diferentes formas, tamaños y caídas. Los pinceles deberían ser un concepto familiar para cualquiera que tenga experiencia en el uso de Photoshop o una aplicación de edición de imágenes similar.

En nuestro caso se han usado los siguientes tipos de pincel para lograr aplanar el terreno:

El sculpt brush para subir y bajar la altura del paisaje, el smooth brush para modificar los valores pintados del mapa y dar al paisaje una apariencia más suave deshaciéndose de las partes dentadas, el flatten brush para aplanar el nivel que esté bajo el ratón cuando se activa y aumentar o disminuir los valores de los mapas de altura circundantes y hacer que tengan el mismo valor. Además de estos pinceles, existen otros muy útiles que permiten generar incluso la apariencia propia de la erosión por lluvia.

En la Figura 3-8 se muestra cómo es el proceso de aplanado del terreno con la herramienta esculpir.

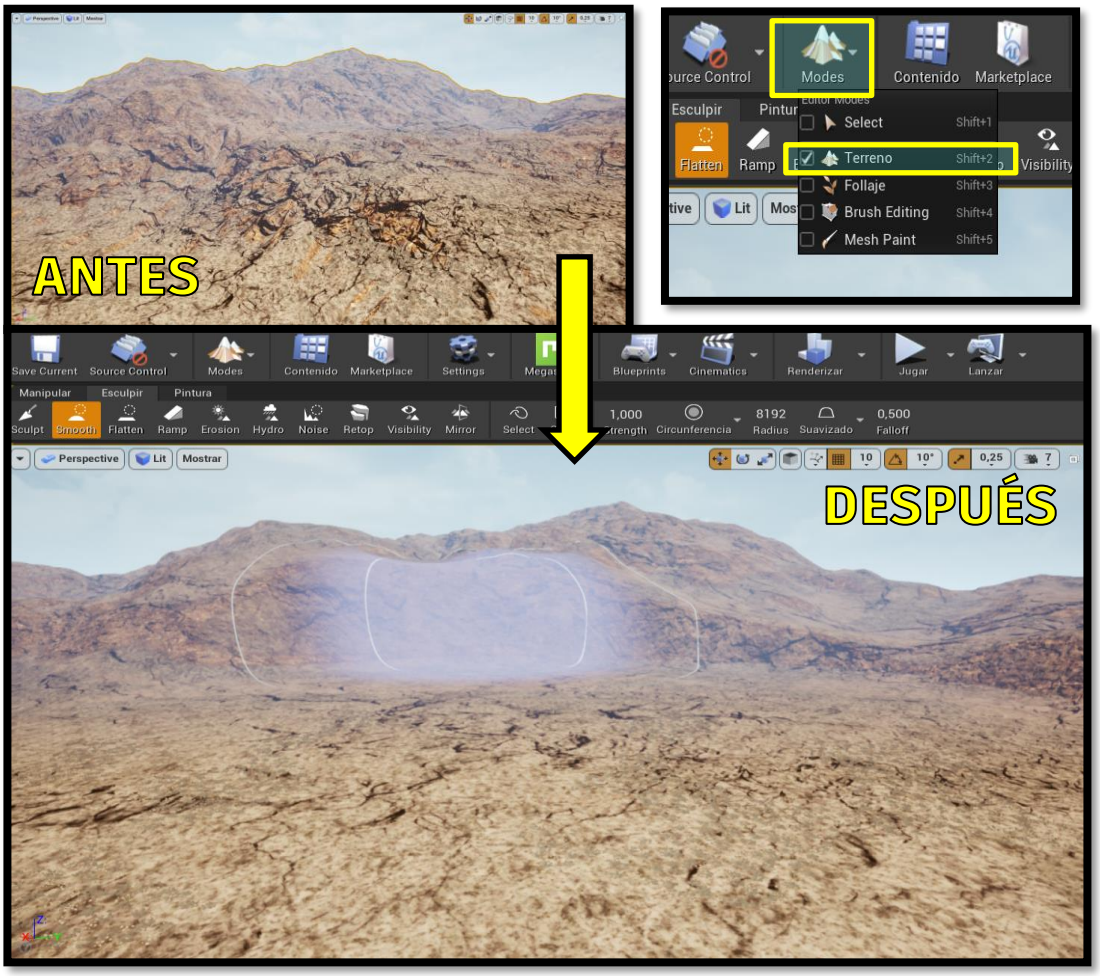


Figura 3-8 Aplanando el terreno con la herramienta esculpir

Tras dar la forma deseada al terreno se ha procedido con la construcción del poblado y la ambientación en general. Para esto, se han utilizado una considerable cantidad de assets disponibles en el mismo pack del que se obtuvo el landscape. Estos assets son actores ya diseñados y preparados para ser ya incluidos sobre el nivel. Concretamente los assets escogidos para nuestra simulación cuentan con una gran resolución y una calidad suficiente por lo que aportan el realismo buscado. La localización y la disposición de las casas han sido diseñadas de tal manera que se asemeje lo máximo posible a las referencias visuales mostradas en el subapartado 3.3.1 estudio previo del terreno. Además, se han planificado dos zonas principales: una donde se situará la casa objetivo en la que se centra el desarrollo del presente Trabajo de Fin de Grado y otra, destinada al posible diseño de un escenario bélico de mayor extensión y totalmente exterior. La implementación de los diferentes assets se ha hecho de manera directa desde la carpeta Mout_Arabian en la carpeta Content, arrastrándolos sobre el viewport. Cuando tenemos activo un actor ya colocado sobre el mapa (este aparecerá con la silueta marcada en naranja y un eje de coordenadas en el centro) podemos moverlo con la herramienta transform en la pestaña de detalles del actor o utilizando el mismo sistema de coordenadas que aparece sobre el viewport. Aquí, podemos seleccionar la posición en coordenadas del actor o arrastrar cualquier eje de forma que las coordenadas varíen rápidamente. Los assets escogidos, han servido para generar la ambientación correcta. Se ha extremado al detalle la colocación de todos los actores sobre el terreno, de tal forma que, a vista de cualquiera, este disponga de la lógica topográfica y arquitectónica propia de un poblado de estas características. Además de casas, se han añadido actores que dotan al entorno de un marcado componente bélico, como torres de vigilancia, sacos terreros, neumáticos a modo de barrera, barricadas, líneas de carretera y un tendido eléctrico.

En la Figura 3-9 se muestra cómo se colocarían los distintos actores sobre el terreno.

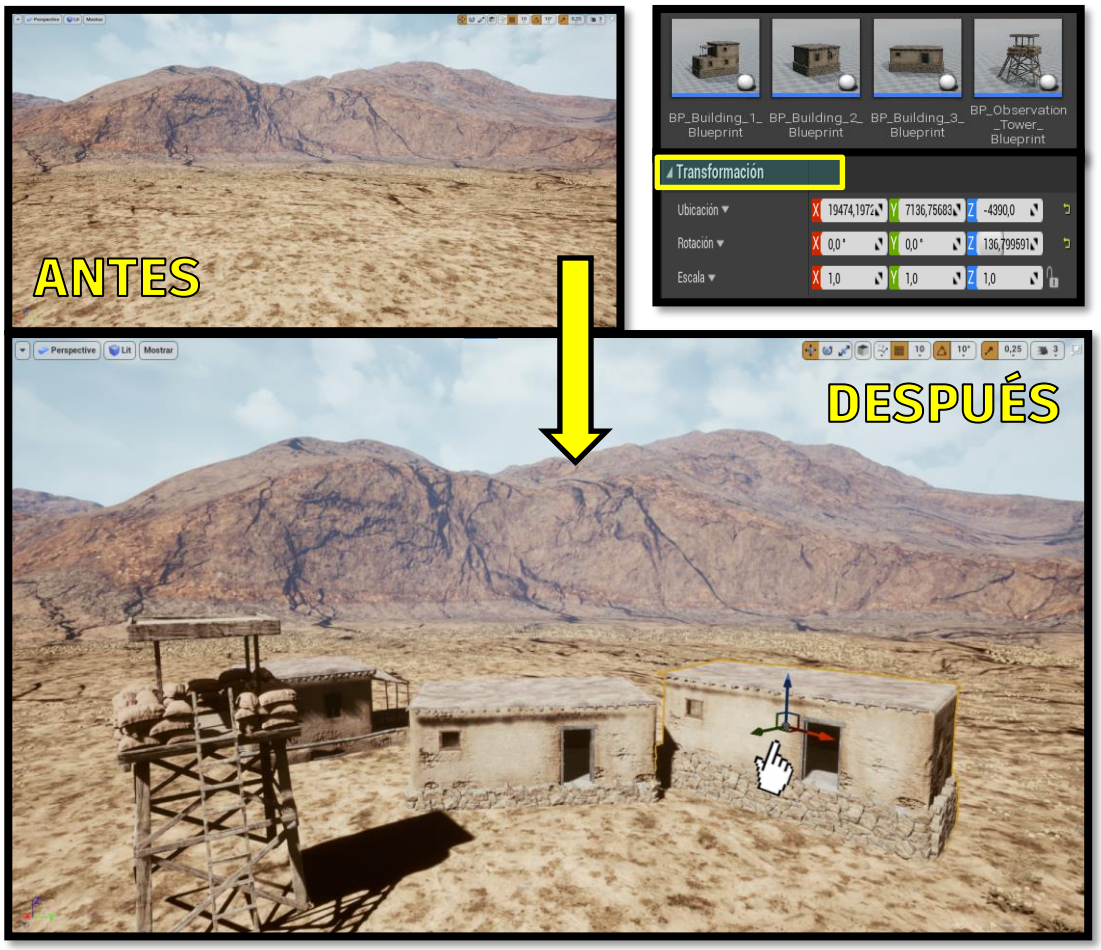


Figura 3-9 Colocación de assets sobre el escenario

3.2.3.1 Diseño y construcción de la casa objetivo

Una vez dispuestos los elementos sobre el terreno y lograda la ambientación deseada, comienza el proceso de diseño de la casa principal donde se desarrollará el nivel. Esta casa, a diferencia del resto, sí será diseñada desde cero para obtener algo aún más ajustado a los requerimientos tácticos. La casa debe de ser lo suficientemente grande como para que un asalto tenga una duración de tiempo considerable y además permita presentarse como una buena plataforma de adiestramiento para este tipo de operaciones tal y como lo hacen en la actualidad nuestras infraestructuras de entrenamiento real. Su diseño, ha seguido la línea estética y arquitectónica propia de la zona. Además, ha sido generada de tal forma que de la sensación de ser la base o el centro de operaciones de la "organización" a la que se pretende atacar.

Partiendo desde el suelo, delimitando la zona y escogiendo las dimensiones de cada habitación, la casa se ha levantado de forma progresiva hasta el techado (Figura 3-10 y Figura 3-11). La elección de los materiales y los elementos decorativos ha sido cuidada al detalle buscando el máximo realismo posible. Colocados uno a uno sobre el nivel, los assets utilizados han sido directamente extraídos desde la biblioteca de contenido MegaScans, una biblioteca masiva de escaneos en línea de alta resolución. Reúne más de 10.000 recursos de fotogrametría, además de diversas aplicaciones de escritorio para gestionar, mezclar y exportar los datos de escaneo descargados. Estos escaneos se traducen en assets fotorrealistas, con calidades de hasta 8K directamente importables al nivel, por lo que su uso es altamente recomendable. Además, ofrece un amplio catálogo de recursos en cuanto a texturas y materiales. Estos han sido los utilizados para darle el aspecto deseado a la casa, desde el suelo hasta las paredes. El uso

de esta librería es totalmente gratuito con la cuenta de Epic Store. Una vez descargada la aplicación Quixel Bridge se puede realizar una importación automática descomprimiendo los materiales de MegaScans descargados desde múltiples ubicaciones hasta un repositorio configurable (Figura 3-12 y Figura 3-13). La aplicación permite ver las imágenes con vista previa, exportar la información de los activos como datos JSON a un puerto específico o navegar, buscar y descargar activos de una manera ligera, rápida y con múltiples funciones.

La casa cuenta con un total de siete espacios navegables: Una primera sala a modo de recibidor, un cuarto con camas, un baño, un salón principal y un almacén (donde se encuentran los rehenes). Además, existe un primer pasillo que conecta el recibidor con el baño y un segundo pasillo que une el salón con el almacén.

Figura 3-10 Construcción de la casa objetivo

Figura 3-11 Colocación del techado de la casa objetivo

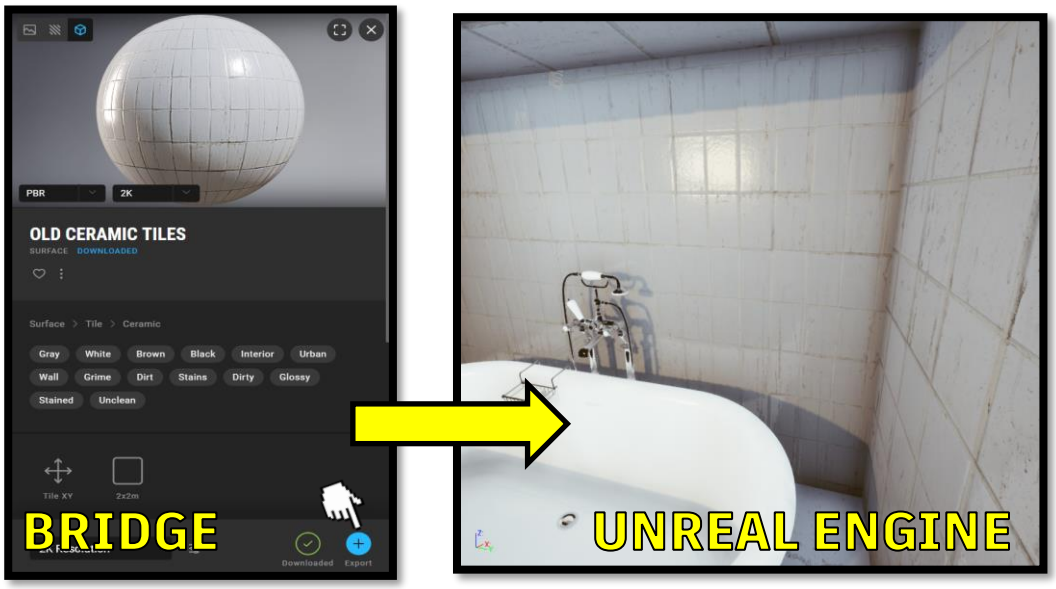


Figura 3-12 Elección de material para suelos y paredes de la casa objetivo

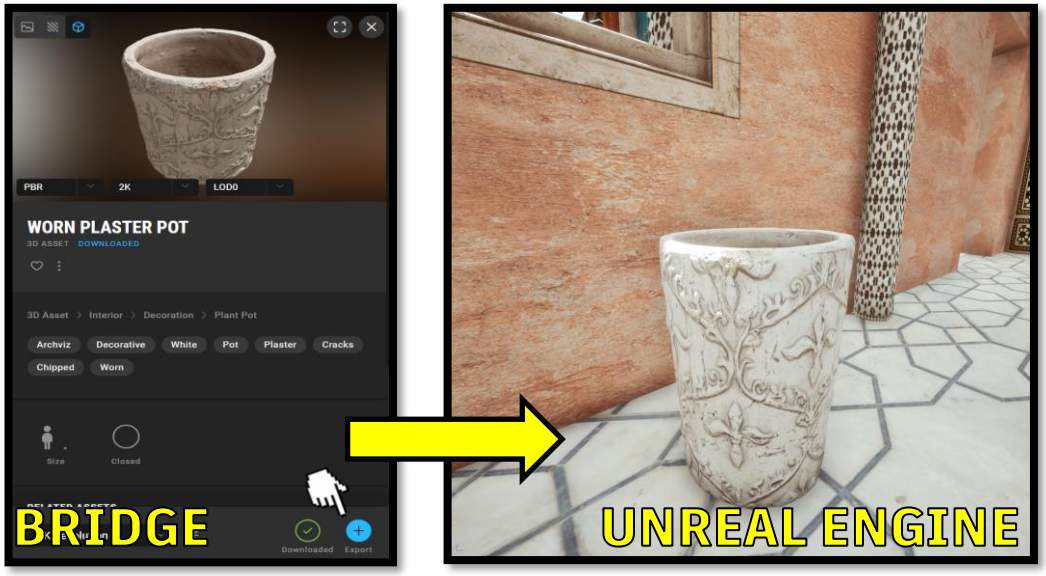


Figura 3-13 Importación de assets

3.2.4 Diseño del aspecto sonoro

Todo uso hiperrealista del sonido, especialmente por parte de las compañías de videojuegos de alto presupuesto, se hace en un intento de asegurar que el jugador no experimente esquizofonía, el intento automático del cerebro de distinguir los sonidos reales de los falsos/reproducidos. Esto hay que tratar de evitarlo ya que puede provocar una ruptura de la inmersión. Tampoco quiere decir que los únicos sonidos necesarios sean los hiperrealistas, pero la versatilidad del sonido es un aspecto importante en la creación del ambiente. [57]

Por eso en este proyecto también se tiene en cuenta que para aumentar la sensación de inmersión es conveniente incluir un sonido ambiente de acorde al emplazamiento que se está simulando. Con pistas

de audio directamente grabadas del desierto tanto de día como de noche y extraídas de internet, se han logrado implementar de manera que el jugador sienta en todo momento el viento y los sonidos característicos de todo lo que está viendo a su alrededor.

3.2.5 Diseño del fusil HK G36

El diseño del fusil de asalto a controlar durante la simulación parte de un modelo en formato. ipt ya existente. Sin embargo, antes de realizar su implementación en el nivel, el fusil ha sufrido un proceso de rediseño y modelado con el fin de mejorar su aspecto y adaptarlo a nuestras necesidades.

Este modelo inicial cuenta con algunas partes que no son necesarias, como la mira digital en la parte superior o la bayoneta calada incluida. También se espera diseñar la mira telescópica del fusil por lo que se ha modificado la zona adyacente a la misma para permitir incluir la lente en una posición correcta y funcional. Para la eliminación de los elementos innecesarios y el rediseño de la mira se ha utilizado el programa Autodesk Inventor. El resto de las modificaciones del arma, así como la transformación en formato. fbx (formato de archivo que Unreal Engine utiliza) ha sido mediante el programa Autodesk 3DS Max. La Figura 3-14 muestra el proceso de modelado de fusil desde su diseño inicial hasta el definitivo.

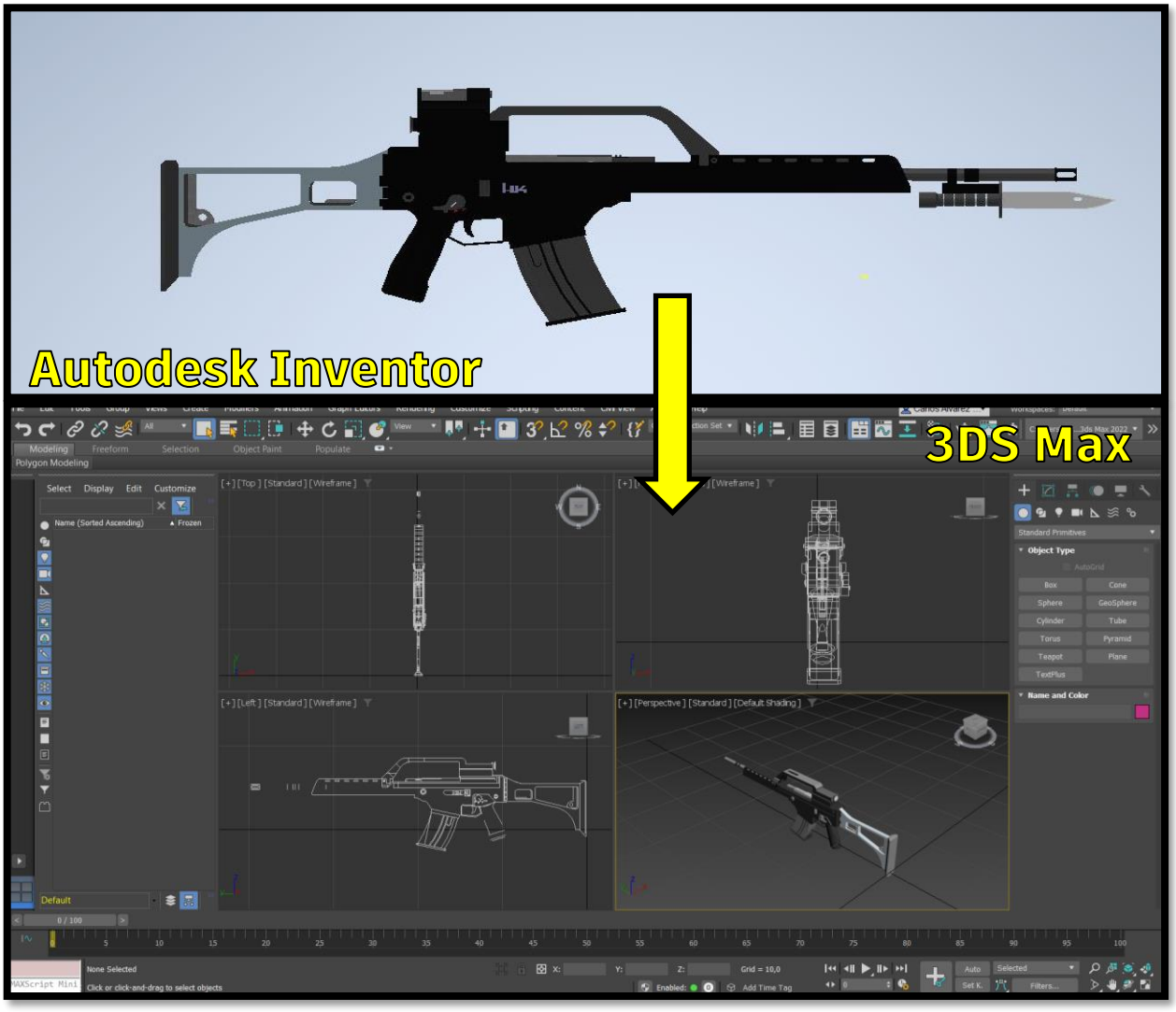


Figura 3-14 Rediseño del modelo inicial del HK G36

Pensando ya en la implementación y en la interacción con las manos se han separado los elementos que posteriormente serán las partes movibles del arma. En total, tal y como muestra la Figura 3-15, se ha dividido en 4 partes:

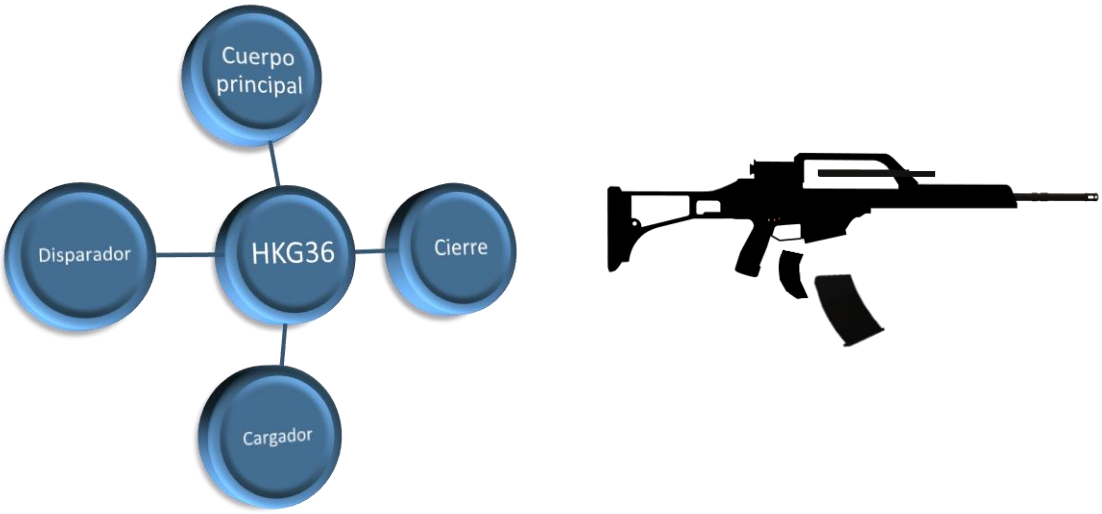


Figura 3-15 Partes movibles del HKG36 en la simulación

El arma, con las 4 partes por separado, ha sido importada a la carpeta Content, dentro de una subcarpeta de nombre HKG36_CAD. Una vez importadas a Unreal Engine podemos trabajar sobre las partes para dotarles de un aspecto y de un color lo más parecido a la realidad posible. Unreal Engine cuenta con varias opciones de personalización para esto mismo.

3.2.6 Diseño e implementación de la mira telescópica

La mira telescópica de 3 aumentos del HK G36 es otro de los objetivos de diseño del simulador. Este diseño, debido a las características de la RV y su profundidad real, dista del diseño de miras telescópicas para fusiles en simulaciones de un FPS (First Person Shooter). En estos juegos, las miras telescópicas suelen activarse con un botón que automáticamente presenta una pantalla con el campo de visión aumentado a través de una lente y una cruceta de apuntado. Dicho de otra manera, en la mayoría de los juegos el arma no dispone de una mira telescópica, solo se posee la opción de pasar a una cámara distinta que simula el apuntado con la mira. En nuestro caso, el arma debe de incorporar la mira como si fuera real de tal forma que cada vez que uno acerque su campo de visión a la mira pueda ver el aumento a través de ella sin necesidad de pulsar un botón ni cambiar de cámara.

Aunque para el diseño de la mira también se necesita programar una parte mediante Blueprints este se ha incluido en el subapartado de diseño y modelado del HKG36 ya que es un componente sustancial del mismo.

Para ello empezar con el diseño, en primer lugar, hay que crear un *Blueprint* propio para el cuerpo del fusil. Desde dentro de la ventana del Blueprint, ponemos la vista de la *ventana gráfica* y en la parte izquierda podremos observar una pestaña con todos los componentes que son inherentes o dependientes de nuestro actor (en este caso el cuerpo del fusil). Como queremos generar una lente que siempre acompañe al fusil, como si esta parte fuera una más, podemos añadir un nuevo *static mesh* desde la pestaña componentes. Este nuevo componente ya tendrá la característica de ser inherente al actor desde el que se ha creado, por lo que siempre se desplazará con él. La forma más aproximada de entre todas las disponibles para simular una lente es el cilindro. Una vez incorporado el cilindro al fusil, este aparecerá en el *viewport* de la ventana gráfica del mismo y podremos recolocarlo y reescalarlo de tal manera que lo dejemos situado justo donde queremos que se sitúe la mira telescópica.

Posterior a esto, se ha incorporado el fusil un nuevo componente de igual manera que el cilindro, pero siendo ahora un componente de captura de la escena 2D. Este componente es una cámara que en todo momento genera un render (representación gráfica) a tiempo real de lo que está capturando. Este render después será lo que el jugador vea a través de la mira telescópica. Como lo que se busca es que el disparo vaya en la dirección en la que la mira está apuntando y el centro de la cruceta de apuntado sea el lugar de impacto de las balas, debemos colocar esta cámara centrada en la boca del cañón. Una vez colocada la cámara en el sitio correcto, podemos disminuir su tamaño a fin de que moleste lo menos posible en la ventana gráfica. Además, se ha configurado su *FOV* (*Field Of View* o campo de visión) desde el apartado *projection* en los detalles de la cámara para lograr un aumento de hasta x3. La Figura 3-16 muestra la disposición de los componentes generados sobre el fusil.

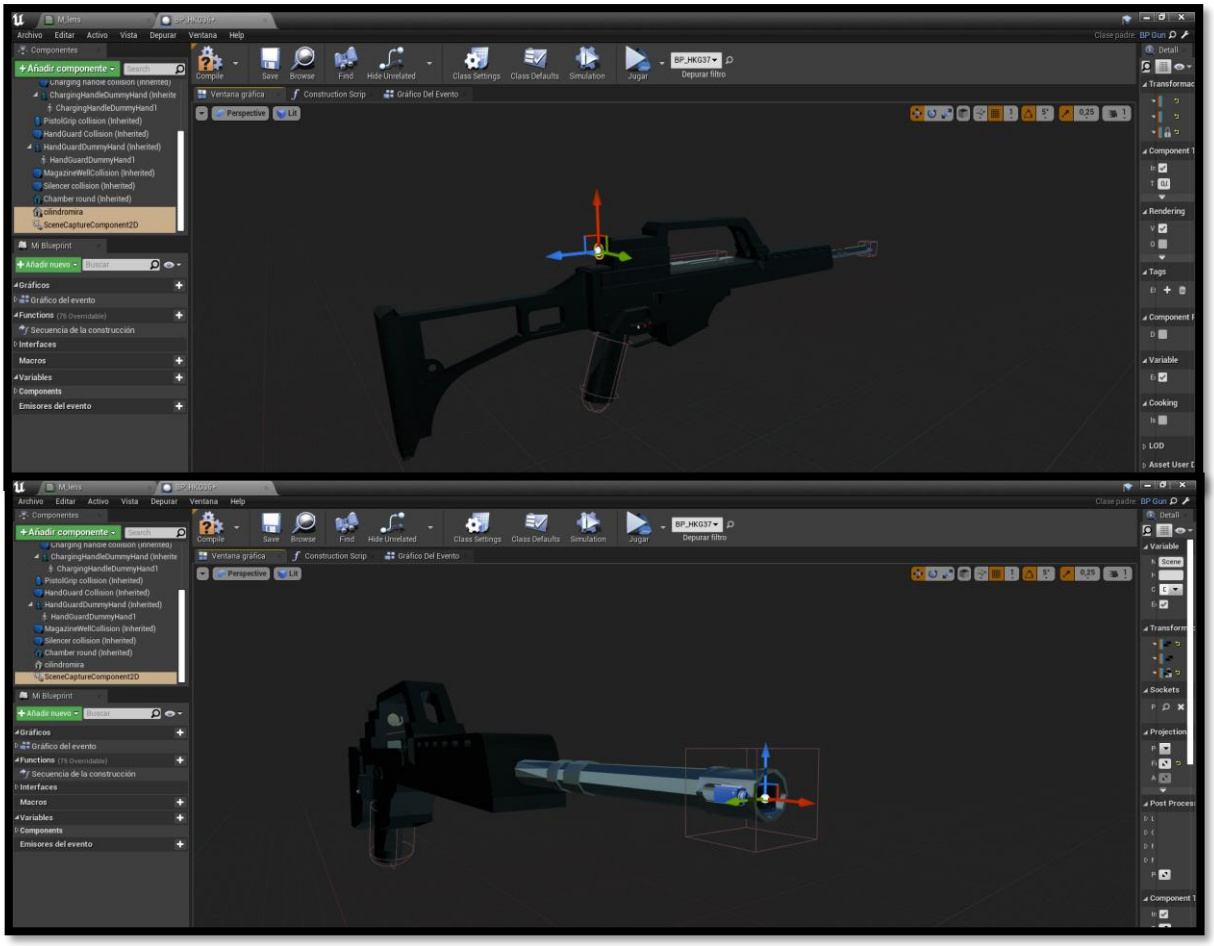


Figura 3-16 Colocación del cilindro y la cámara en el fusil

Lo siguiente ha sido generar un nuevo material que será el que le aplicaremos al cilindro. Un material es un asset que se puede aplicar sobre una malla para definir el aspecto visual de una escena. A modo de ejemplo, un material representa la pintura que se aplica sobre un objeto para obtener un resultado más realista. Sin embargo, un material puede estar asociado con otras propiedades, como por ejemplo el brillo o la capacidad de ver a través del objeto que lo usa. Otra alternativa podría ser el uso de una textura, que es una imagen 2D que envuelve a un objeto para dotarlo de un aspecto visual determinado.

La creación de un nuevo material puede ser directamente llevada a cabo desde una carpeta cualquiera. Dentro de una nueva carpeta llamada Mira_CAD se ha generado un nuevo material pulsando sobre el botón derecho. A este material se le ha nombrado material_mira.

También desde dentro de la misma carpeta se ha generado un *render target*, igualmente seleccionable haciendo click derecho dentro de la carpeta. Este *render target*, convierte todo lo que la cámara está grabando en una textura aplicable a una superficie. Para asociar la cámara al *target* debemos volver a la ventana gráfica y seleccionarla. Desde los detalles, en el apartado de *scene capture*, en *texture target* podemos abrir un desplegable en el que aparece el *render target* al que queremos asociar la cámara con el mismo nombre con el que lo creamos. En nuestro caso el *target* tiene el nombre de render_mira.

Además, debemos de incorporar a la mira la cruceta de apuntado del fusil HKG36. Partiendo de una imagen en .png y con ayuda del editor fotográfico gratuito GIMP 2.10.30 se ha modificado para eliminar el fondo y sustituirlo por un fondo totalmente blanco. Así se dispone solamente de la silueta de los elementos propios de la cruceta. La imagen también ha sido importada a la carpeta Mira_CAD.

Por lo tanto, lo que haremos será generar un material que mezcle el *render target* y una imagen de la cruceta de apuntado y se lo aplicaremos al cilindro para que este genere la impresión de ser la lente de aumento del fusil (Figura 3-17).

Al hacer doble click sobre el material creado en la carpeta se accede al editor de materiales cuya programación también es de manera visual mediante Blueprints. De inicio, aparece un nodo con una serie de pines de entrada a los que se puede conectar otros nodos para definir las propiedades del material. Arrastrando sobre la ventana de edición del material el *render target* y la imagen importada se subirán como dos nodos independientes con pines de salida. Estos nodos tendrán la expresión *texture sample*, lo que indica que su salida dará los valores de color de una textura. Para mezclar ambas texturas debemos usar la función *lerp* que situaremos entre los nodos de salida de las texturas y los de entrada del material. Esta función matemática realiza una interpolación lineal entre los valores de ambas texturas en función de la que se le asigne el nodo de entrada Alpha (la función devuelve un 100% de A cuando Alpha=0 y 100% de B cuando Alpha=1). En nuestro caso asignamos los valores RGB de la *texture sample* de la cruceta al nodo Alpha y al nodo A y los del render target al nodo B. Por último, la salida de la interpolación es directa con la entrada emissive color del material. Además, a fin de mejorar el aspecto se han ajustado parámetros como la rugosidad o el brillo.

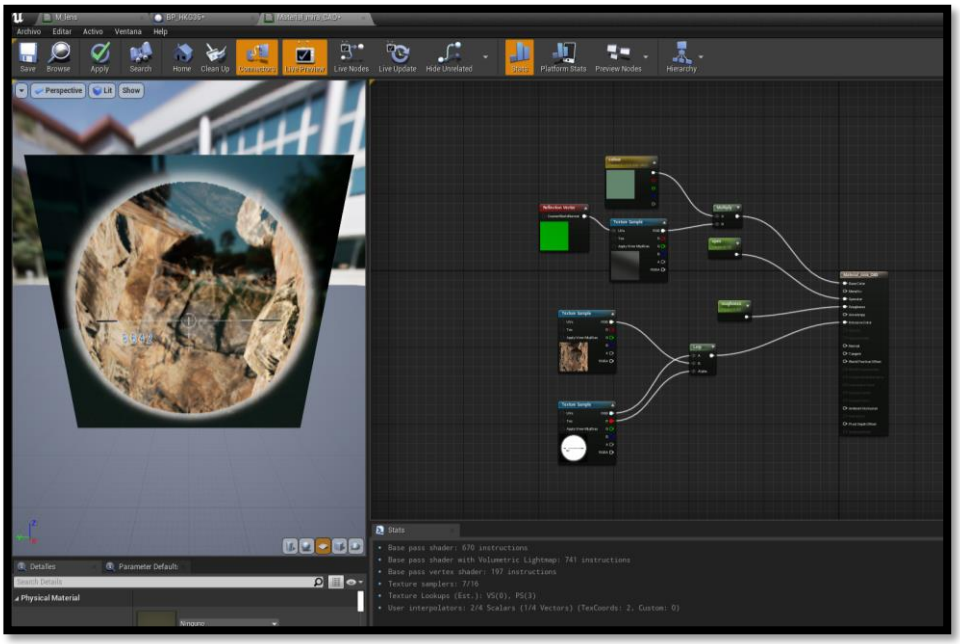


Figura 3-17 Creación del material de la mira telescópica

Tras esto, habría que guardar el material y aplicárselo al cilindro directamente desde la ventana de detalles de este. Ahora el cilindro muestra en todo momento lo que está renderizando la cámara en la boca del cañón con la cruceta de apuntado del fusil HKG36 superpuesta tal y como en la vida real.

3.3 Programación en Unreal Engine

El sistema de scripting visual Blueprints es el corazón de la programación en UE4 y está concebido para facilitar enormemente el desarrollo de un proyecto como este. Para ello, se basa en un enfoque basado en nodos visuales para definir los elementos de juego e interacción.

En este apartado se definirá el comportamiento de todos los elementos que forman parte de la simulación y se programará la lógica del nivel. La estructura del apartado sigue a grandes rasgos un orden cronológico de desarrollo. De forma progresiva se irán alcanzando los objetivos programación hasta terminar con la integración de todos los elementos en el diseño del nivel al completo.

3.3.1 Programación del sistema de movimiento

Los sistemas de movimiento son el medio por el que el jugador controla y maniobra el mundo virtual que lo rodea. Son la base de casi todas las simulaciones y dictan las opciones que tiene el jugador para atravesar los obstáculos del entorno.

3.3.1.1 Planificación de los sistemas de movimiento

El siguiente paso lógico en el desarrollo de nuestra simulación pasa por tanto por diseñar los distintos tipos de desplazamientos, conocidos como *locomotions*, con los que se manejarán los movimientos del personaje dentro de la simulación.

Hasta aquí y dado que utilizamos la plantilla RV que suministra UE, el desplazamiento sólo es posible mediante el denominado teletransporte o *teleport*, el cual es buena manera de introducirse en la realidad virtual sin sufrir de alguno de los efectos no deseados como el motionsickness ya comentado en el subapartado 2.3.2.3. Sin embargo, no es la única manera de moverse por una experiencia en RV, de hecho, la lista de métodos de desplazamiento está en constante cambio ya que la industria nos tiene acostumbrados a refinar sucesivamente aquello que aparentemente funciona. Este es el caso de las nuevas variantes que hacen uso del seguimiento de manos o hand tracking en donde no hay mandos que podamos utilizar para desplazarnos. Idealmente nuestras experiencias deberían contar con diferentes métodos de desplazamiento, es por esto, que nos centraremos en desarrollar y mejorar la plantilla con un movimiento libre o *free locomotion* con nuestro propio desplazamiento gracias al *headtracking* (seguimiento continuo de la posición del headset de RV) y la posibilidad de utilizar el trackpad como método alternativo de movimiento (Figura 3-18).

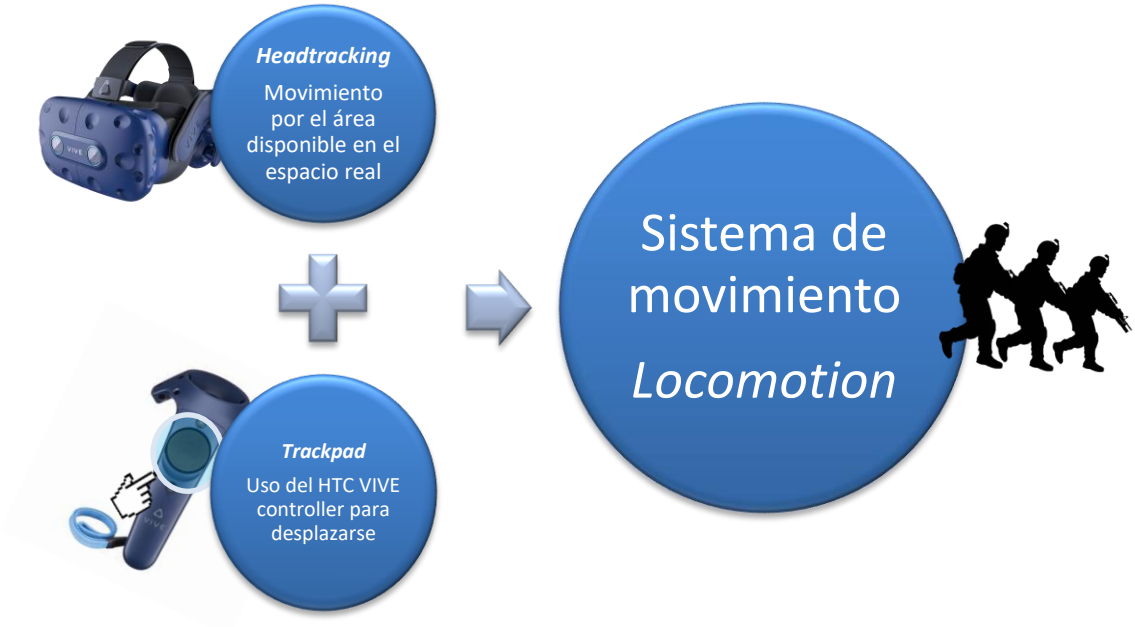


Figura 3-18 Posibilidades del sistema de movimiento

El controlador de movimiento o motion controller pawn es el encargado de generar el personaje virtual que queremos manejar. Este motion controller pawn alberga toda la información con respecto a los sistemas de movimientos, físicas, animaciones o el sistema de vida del personaje por lo que la programación de los sistemas de movimiento será editable desde el Blueprint (BP) del motion controller pawn.

La plantilla RV inicial con la que se inició el proyecto utiliza un motion controller pawn que como ya hemos comentado solo permite el movimiento con teletransporte. Además, debido a que queremos alcanzar el máximo grado de realismo, es conveniente utilizar un controlador que cuente con un IK (Inverse Kinematic) locomotion. Esto quiere decir que el personaje a controlar sea capaz de realizar animaciones reactivas, como la colocación de los pies en terrenos no planos. Gracias a esto los pies se irán adaptando a la altura de todo lo que tenga debajo y elevarán o bajarán la posición del cuerpo en función de esto mismo (por ejemplo, al subir una escalera). Como el motion controller de la plantilla no dispone de las características IK ni tampoco de un maniquí de cuerpo completo (sólo muestra en pantalla la representación virtual de las manos) se ha partido del motion controller pawn del Kit disponible en el Marketplace de la EpicStore VR Weapons Tactical Assault.

Este motion controller pawn se incluye en el nivel tras ser arrastrado desde la carpeta de contenido del VR Weapons Kit directamente sobre el viewport. Una vez situado en el nivel, al hacer click en el maniquí podemos acceder al BP que lo controla desde el World Outliner a la derecha de la pantalla para observar la composición del controlador y la programación por defecto con la que cuenta. Al situarnos sobre la ventana gráfica del BP (Figura 3-19) se observan el controlador y los componentes atribuidos al mismo. Este motion controller pawn consta de dos partes diferenciadas y a las que haremos constante referencia en el desarrollo debido a su implicación directa con la jugabilidad y el desplazamiento. En primer lugar, el controlador cuenta con una malla o mesh que aporta el componente físico al controlador. En otras palabras, es el maniquí con animaciones y movimientos ya predefinidos. Por otro lado, situada encima de la malla, nos encontramos con la cámara, la cual aporta el campo de visión desde la cual el jugador ve en todo momento lo que está sucediendo. Estos dos elementos al ser independientes deberán de ser tenidos en cuenta a la hora de programar el movimiento para que funcionen de manera correcta y pareja. Desde el apartado class defaults del motion controller es importante marcar como verdaderas las opciones de Locomotion (indica que usaremos un método de desplazamiento diferente al teletransporte) y IK (indica que el controlador cuenta el Inverse Kinematic Locomotion comentado anteriormente).

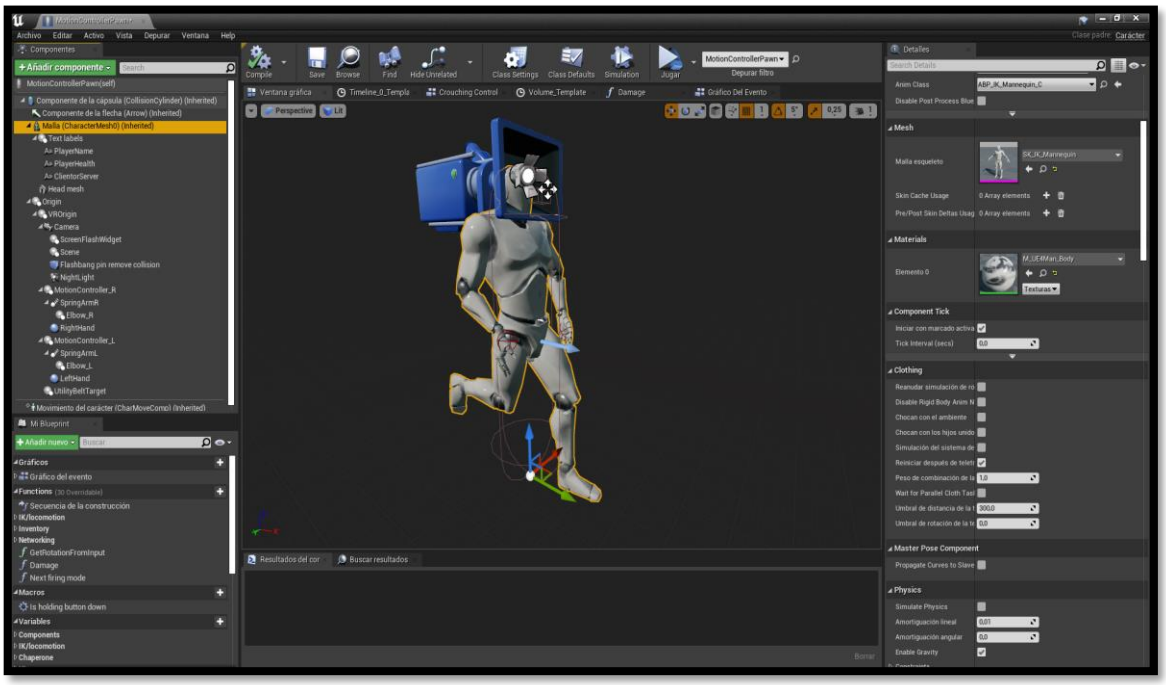


Figura 3-19 Ventana gráfica del motion controller pawn

3.3.1.2 Movimiento mediante headtracking

Esta forma de desplazamiento es de especial interés sobre todo cara al adiestramiento en la movilidad que exige una operación de este tipo. Si bien el movimiento por el espacio disponible puede resultar algo complicado de asimilar, ya que el uso de las gafas de RV genera siempre una sensación a priori de "indefensión" y desconcierto frente al nuevo "mundo" que el cerebro trata de procesar, con un buen adiestramiento del personal se pueden sacar provecho de las numerosas ventajas que implica su inclusión como sistema alternativo de movimiento.

Para comprender como se genera este desplazamiento, hay que tener claro los conceptos explicados anteriormente. La cámara (lo que vemos a través de las gafas) y el cuerpo (la malla del personaje virtual) son dos actores independientes, por eso necesitamos enlazarlos de forma correcta. El cuerpo será el que hará que nos desplacemos, y la orientación de la cámara determinará la rotación del cuerpo para ello. Dentro del BP del motion controller pawn se ha generado un nuevo evento llamado TickAutoMoveByHead.

Este evento se inicia con un nodo Set World Rotation que rotará sobre el nivel la malla del personaje en función de los ejes X e Y (Roll and Pitch) y adecuará la rotación del eje Z (Yaw) en función de la rotación de la cámara en todo momento. De esta manera, si en la vida real nos giramos, el cuerpo y la visión del personaje virtual también lo harán. La cámara por defecto tiene un offset o desfase respecto al cuerpo de 90º en el eje Z, por lo que mediante una operación de resta se ha solventado esta diferencia.

Una vez configurada la malla para seguir la rotación en el eje Z de la cámara, se ha añadido el noto Set World Location que colocará sobre el nivel el componente de la malla en función de un sistema de coordenadas cartesianas espacial. El equipo de RV mediante los sensores obtiene en todo momento los valores de las coordenadas para situar al jugador en el espacio, por lo que estos valores serán los que acoplaremos a los pines de entrada del Set World Location asociado al componente de la malla. Los valores cartesianos de la cámara se obtienen directamente con un nodo GetActorLocation asociado al componente de la cámara. La programación del Blueprint queda expuesta en la Figura 3-20.

Así, el cuerpo del personaje se desplaza en función del movimiento por el espacio de nuestro equipo de RV colocado sobre nuestra cabeza en todo momento. Sin embargo, tras los testeos se ha apreciado una diferencia notable de velocidad entre el movimiento real y el movimiento del personaje. Para solventar esta diferencia y adecuar un desplazamiento virtual más parejo al real se ha utilizado una operación de multiplicación de valor variable (float). En este caso se ha dejado en x 1000 debido a que es el resultado que mejores resultados ha ofrecido durante las pruebas realizadas.

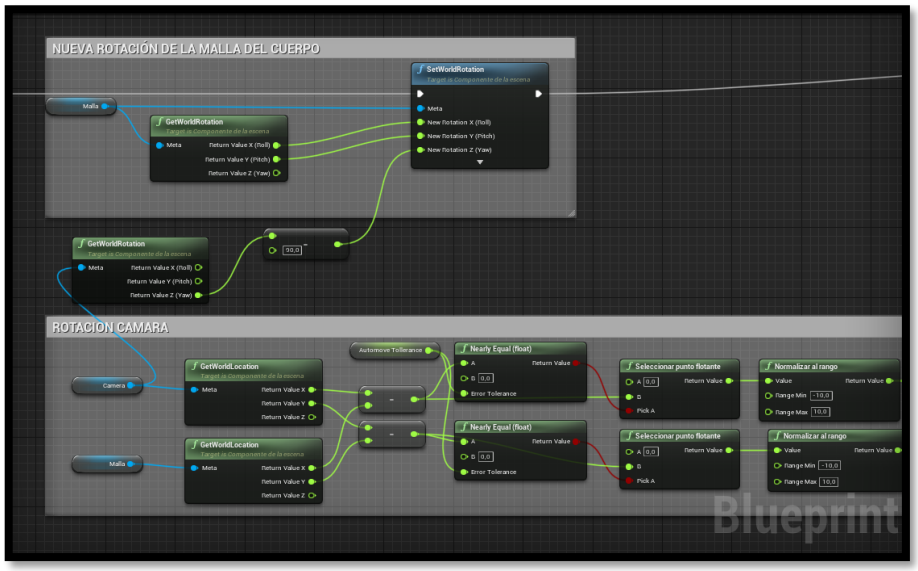


Figura 3-20 Blueprint que asocia la rotación en el eje Z de la cámara y el personaje

3.3.1.3 Movimiento mediante trackpad

Este sistema de movimiento es conocido comúnmente como thumbstick locomotion ya que hace uso del dedo pulgar de nuestra mano para lograr el desplazamiento. Todo lo que el dedo pulgar hace sobre el controlador es una señal o Input que este recibe y que podemos procesar con la finalidad de acabar forzando una respuesta por parte del personaje. Este juego de señales que el jugador hace sobre los controladores es conocido como PlayerInput.

Para que el actor (en nuestro caso el motion controller pawn) pueda entender y utilizar los player inputs debemos transformar la entrada del jugador en datos. Este proceso es parte de un flujo de procesamiento que traduce la entrada de hardware en eventos de juego y movimiento.

Unreal Engine ofrece la posibilidad de modificar el PlayerInput directamente desde la opción ajustes del proyecto en el apartado Entrada dentro de la opción Motor (ver Figura 3-21). Aquí podemos manejar las opciones de entrada de hardware y redefinir las acciones que toman cada una. Se pueden diferenciar dos tipos de configuraciones dentro de este proceso. El action mapping y el axis mapping. Ambos sirven para asignar un botón o la pulsación de una tecla a un nombre que más tarde se vinculará a un comportamiento del juego o bien de manera discreta o bien de manera continua. Las entradas del action mapping solo se activan cuando se hace una acción concreta mientras que las acciones del axis mapping son continuamente sondeadas, incluso si sólo informa que su valor de entrada es igual a cero (lo que permite transiciones suaves en el movimiento).

En nuestro caso, para hacer que el controlador reconozca y nombre sus entradas de movimiento debemos ir a la pestaña axis mapping y generar una nueva entrada que llamaremos MoveForward_Y (entrada del movimiento hacia delante y hacia detrás) y otra llamada MoveRight (entrada del desplazamiento lateral). Una vez creadas las entradas podremos asignarle tantos inputs como queramos. El motor gráfico Unreal Engine es capaz de reconocer una amplia variedad de inputs de distintos tipos de controladores. Con esto podemos tener la misma respuesta, es decir generar la misma entrada, para diferentes tipos de equipos de Realidad Virtual y aumentar la cantidad de dispositivos que podrían manejar el simulador. En nuestro caso a la entrada MoveForward_Y le asignaremos el Input VIVE (L) Trackpad Y con escala 1 (esto es el eje vertical del trackpad izquierdo) y a la entrada MoveRight el Input VIVE (L) trackpad X también con escala 1 (esto es el eje horizontal del trackpad izquierdo).

Figura 3-21 Configuración del Axis Mapping

Finalmente solo se ha atribuido un comportamiento a cada uno de los dos eventos que generan las entradas que se acaban de crear para indicarles que el personaje debe desplazarse cuando recibe alguno de los inputs. Para ello, se ha accedido al BP del motioncontroller pawn y se han generado dos nuevos eventos InputAxis Moveforward_Y y InputAxis Move Right directamente desde el desplegable que se abre al hacer click derecho en el viewport del BP. Estos eventos se inician cuando tocamos el trackpad izquierdo. Las salidas de estos dos eventos deben ir cada uno directamente hasta el gráfico colapsado con el sistema de movimiento por defecto del motion controller pawn. Un gráfico colapsado no es más que un conjunto de funciones que por ser de uso común han sido contraídos hasta convertirse en un solo bloque para mejorar su manejo y reproducción. Si hacemos doble click sobre un gráfico colapsado podemos observar la arquitectura del Blueprint y realizar variaciones en el mismo. La salida de los eventos InputAxis irán el vertical hacia el MoveForward del gráfico colapsado y el horizontal al MoveRight del mismo. Los valores axis que salen de los eventos se suman para activar el spawn sound at location que disparará un footstep cue (biblioteca con sonidos almacenados) con el sonido de pisadas aleatorias al moverse. La programación del Blueprint queda expuesta en la Figura 3-22.

Dentro del gráfico colapsado locomotion controller podemos observar cómo el BP provoca el movimiento del peón en función del valor que reciba de los eventos de entrada (vertical+horizontal) mediante la función añadir movimiento. Hay que recordar que esta función provoca el movimiento en sentido contrario cuando detecta datos de entrada negativos por lo que directamente sirve para provocar el movimiento hacia detrás y hacia la izquierda también. Además, de cada dirección que toma el desplazamiento obtiene un vector que asocia a la cámara para posteriormente hacer que esta tenga siempre la misma orientación que la dirección de movimiento mediante las funciones Set y Get World Rotation.

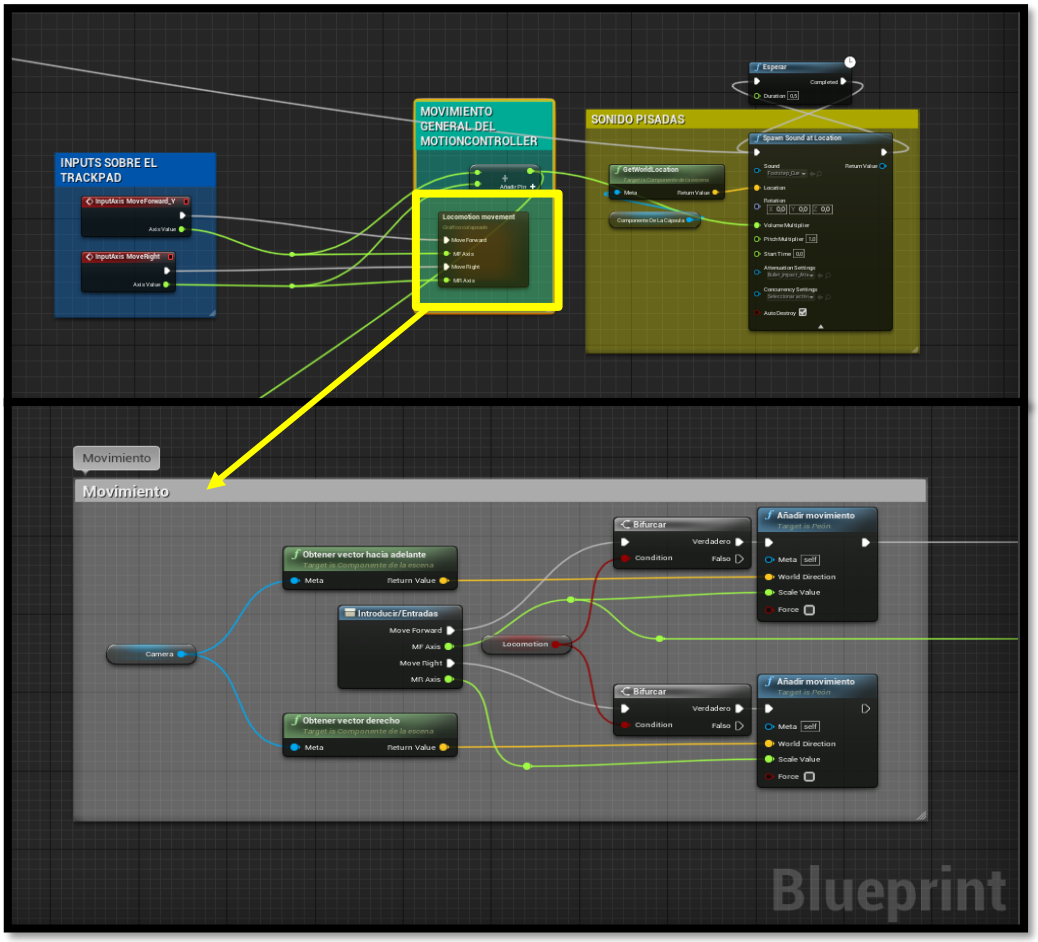


Figura 3-22 Blueprint que controla el desplazamiento con trackpad

3.3.1.4 Solución al problema de colisiones

Un comportamiento característico y anómalo de las experiencias en RV y que en el resto de los juegos de monitor prácticamente no existe es el de poder atravesar con la cabeza objetos del nivel, lo que hace que podamos ver a través de estos, aun teniendo las colisiones bien configuradas. Este comportamiento hay que evitarlo siempre que sea posible ya que en ocasiones puede sacarnos de la inmersión y hacer que una experiencia rica en detalles se difumine. Actualmente este problema lo sufren la mayoría de los juegos que hay en RV y tan solo se gestionan en mayor o menor medida por los estudios de cierto nivel.

Si el jugador cada vez que introduzca la cabeza en la pared, en vez de ver a través de ella, observa como su visión pasa a ser nula de tal forma que solo vea en negro y además el sonido ambiente se pare de reproducir entenderá tras muy pocas repeticiones que no puede meter la cámara (su campo de visión) dentro de ninguna pared con colisión.

Para conseguir este efecto se ha procedido a la creación de un nuevo evento dentro del *BP del motion controller pawn* al cual llamaremos TickCameraFade (en español desvanecimiento de la cámara) que inicia una secuencia de tres acciones. La primera que actualiza la variable booleana Head in object Check (variable que controla si el jugador tiene introducida o no la cabeza en algún objeto con colisión programada) a través la función set o ajustar en español. Después la secuencia mediante un multisphere trace by channel es capaz de determinar si hay que objetos que están en colisión directa con la cámara y activar la variable booleana, que por defecto estaba en false, en caso de que esto ocurra. Por último, bajo la condición de que la variable booleana siga activa, la secuencia continúa con un start camera fade (función que realiza el fundido a negro de la pantalla). Dentro del start camera fade al dejar activa la opción booleana de should fade audio conseguimos que todo el sonido desaparezca de la misma manera que lo hace la imagen de la pantalla. Como la operación de comprobación de colisión de la cámara está metida dentro de un loop o bucle de repetición que está en continua revisión el evento acaba cuando la variable boolenana de Head in object check vuelva a false que solo sucederá cuando el jugador vuelva a sacar la cabeza de la colisión. La programación del Blueprint queda expuesta en la Figura 3-23.

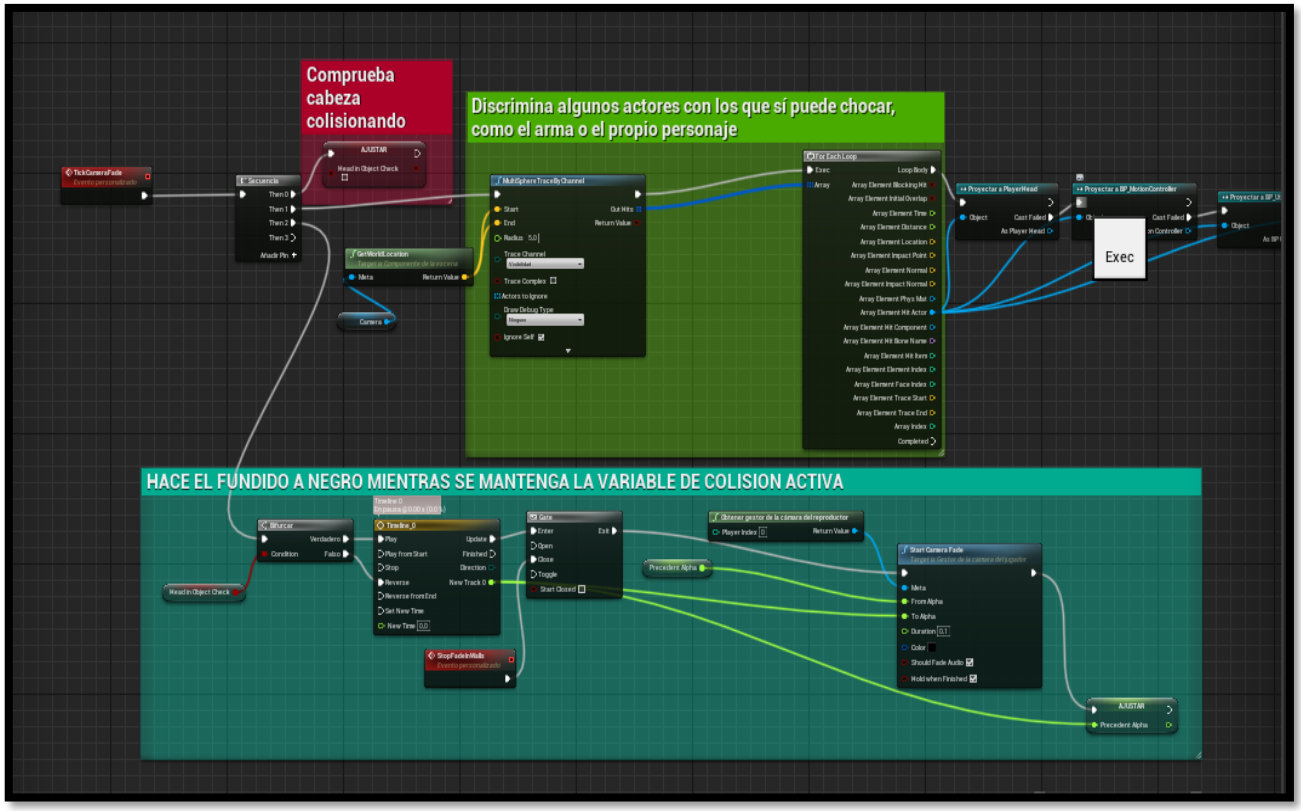


Figura 3-23 Blueprint que soluciona el problema de las colisiones

3.3.2 Programación del fusil

3.3.2.1 Integración del fusil con el entorno

Para lograr que nuestra arma se comporte como tal en el entorno y sea además manejable por el jugador se ha partido de la base de programación de un arma extraída del VR Weapons Tactical Assault. Dentro de la Content/VR_Weapons/Global/Blueprints accedemos al BP_Gun el cual es un Blueprint que contiene las funciones de agarre y funcionamiento que queremos implementar en nuestra arma. Unreal Engine ofrece la posibilidad de generar un children Blueprint a partir de un Blueprint inicial. Este children Blueprint heredará automáticamente la funcionalidad de su padre (en este caso será el BP_Gun), que puede ser aumentada, añadida o alterada. Haciendo click sobre el botón derecho del BP_Gun obtenemos la opción de create Child Blueprint Class. El Child BP ha sido nombrado con el nombre BP_HKG36. Además, también incluiremos el arma dentro de la lista E_Weapon_Type incluida en la carpeta Content/VR Weapons/Pawn/Blueprints con el nombre HKG36 para que esta ya aparezca como un arma más dentro de las funciones y pueda ser llamada cuando se requiera.

3.3.2.2 Interacción del fusil con el controlador

La comunicación entre el motion controller pawn y el arma se produce a través de la interfaz que encontramos dentro de la carpeta Content/VR_Weapons /Pawn/Blueprints que se llama PickupActorInterface. Una interfaz de Blueprint es una colección de una o más funciones (sólo el nombre, sin implementación) que pueden añadirse a otros Blueprints. Cualquier Blueprint al que se le añada la interfaz tiene garantizado que tendrá esas funciones por lo que en nuestro caso se la añadimos al motion controller pawn y al BP_HKG36. Muchas de estas funciones son llamadas desde el BP del motion controller como resultado de los inputs que realiza el jugador (por ejemplo, el pulsar el gatillo o presionar el botón de agarre lateral) y son ejecutadas en el BP_Gun como eventos en el evento gráfico general. Estas funciones son:

- Pickup (agarre del arma).
- Drop (soltar el arma).
- PullTrigger (pulsar el disparador).
- ReleaseTrigger (liberar disparador).
- GripHandGuard (agarre con la segunda mano).
- RealeaseGripHandGuard (liberar el segundo agarre).
- InsertMagazine (introducir el cargador).

Los inputs han sido previamente programados para nuestros controladores HTC VIVE directamente desde el action mapping de los ajustes proyecto tal y cómo se hizo con el axis mapping del movimiento por lo que ahora podríamos asociar las funciones a los inputs.

De momento el arma sigue siendo el arma por defecto del BP_Gun (recordemos que de momento es un hijo idéntico) por lo que se ha tenido que rehacer el proceso de animación y comportamiento completo del arma para adaptarlo al modelo del HKG36 previamente diseñado.

El maniquí del motion controller pawn posee una mano que utiliza una *blend animation* para mezclar las animaciones de distintas armas. Como queremos que las animaciones de nuestras manos respondan a los inputs y realicen de forma correcta y realista las funciones asociadas a nuestra arma la *blend animation* usará diferentes poses de la mano y las mezclará en un orden secuencial de tal forma que la transición de una pose a otra sea lo más natural posible.

Pero para ello, en primer lugar, hay que generar los .fbx para cada una de las posiciones que la mano vaya a adoptar. Estos .fbx después la blend animation se encargará de mezclarlos.

Extrayendo el .fbx de la mano derecha del maniquí desde la carpeta Content/ VR_Weapons/ Pawn/ Mannequin/ Character/ Mesh llamado MannequinHand_Right e importándolo en 3DS Max junto al

modelo del HKG36 adecuamos las diferentes posturas de la mano y los dedos para ajustarlas todo lo posible a la realidad. Una vez adoptadas las posiciones correctas, se puede eliminar el fusil ya que lo único que usamos para la blendspace animation es la mano. En total se han generado 5 meshes (componentes de malla) en formato .fbx directamente importables en Unreal Engine con las posturas mostradas en la Figura 3-24:

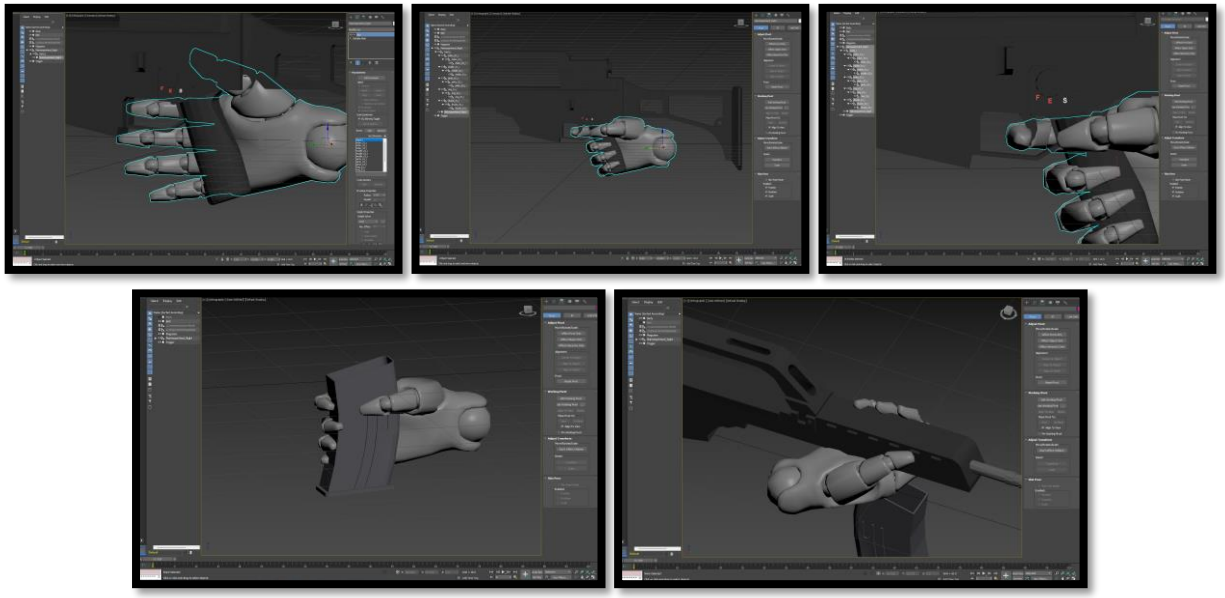


Figura 3-24 Animación de las manos en 3DS Max

Tras crear los meshes de las manos estos han sido guardados en la carpeta Content/HKG36 junto a las partes que componían el fusil diseñado en el apartado. Es importante recalcar que en la importación de los .fbx se ha tenido que especificar que estos eran importados como mallas con el esqueleto propio de la mano derecha de nuestro motion controller pawn llamado MannequinHand_Right_Skeleton.

Para iniciar la mezcla de las nuevas animaciones diseñadas se ha procedido a localizar la blend animation que maneja la mano derecha. Este viene dentro de VR_Weapons/ Pawn/ Mannequin/ Animations bajo el nombre Grips_Blend Space. Al hacer doble click sobre esta se abre la ventana de trabajo. El viewport ahora nos muestra la mano que queremos animar. En la esquina inferior derecha, se puedes ver la lista de animaciones que hemos migrado anteriormente. Además, también muestra cualquier animación en los directorios de contenido que esté asignada al esqueleto de la mano. En la parte inferior central, bajo la vista previa de la mano, podemos ver el espacio de trabajo donde vamos a construir la mezcla de animaciones.

La mano pertenece al motion controller pawn y ya trae por defecto varias animaciones para otros tipos de armas incluidas en el kit. Para incluir las nuevas sobre el espacio de mezcla solo hay que arrastrarlas directamente desde el buscador de activos. Para que la animación tenga el orden secuencial correcto se colocan de izquierda a derecha las siguientes posturas de la mano sobre la línea de animación tal y como muestra la Figura 3-25:

- Open (animación de mano abierta)
- CanGrab (animación de puede agarrar)
- HKG36_hand_grip (animación de agarre) y HKG36_magazine_grip (animación de agarre de cargador)

- HKG36_hand_grip (animación de agarre) Y HKG36_hand_grip_triggerPulled (animación de disparador pulsado)
- HK G36_handguard_grip (animación de segundo agarre

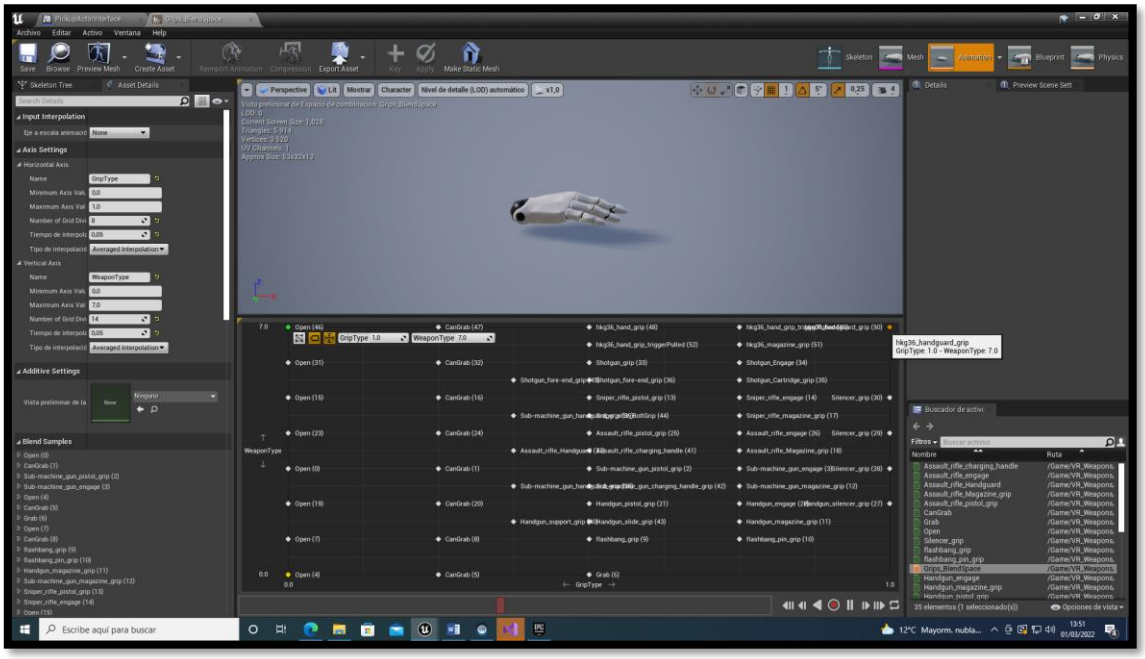


Figura 3-25 Trabajando en la Blendspace

Lo siguiente ha sido sustituir el modelo del BP_Gun por los componentes del HKG36 diseñados**.** Al abrir la ventana gráfica del BP_HK36 se pueden observar los distintos componentes del arma por defecto. En concreto, el Gun mesh será el equivalente al cuerpo de nuestro fusil, el trigger mesh el disparador y el charging handle mesh el cierre. Para cambiar el modelo de cada una de estas partes se ha accedido a los detalles de cada una y han sido reemplazadas por la malla estática que ya teníamos creadas en la carpeta HKG36_CAD en el apartado Static Mesh.

La disposición, así como las dimensiones de las partes del HKG36 difieren de las del arma por defecto, por lo que se han recolocado en su posición correcta. Para ello se han ajustado las coordenadas relativas en la ventalla de detalles de cada parte y se han adaptado hasta adoptar las mismas coordenadas que tenían en el espacio de diseño de 3DS Max.

Una vez cambiados los modelos de las partes movibles del arma, se han readaptado las distintas cajas de colisión inherentes a cada una. En la ventana de componentes podemos ver como debajo de cada malla existe una colisión del mismo nombre seguido de la palabra (inherited). Una caja de colisión es un área determinada, que puede adoptar distintas formas y que normalmente se utiliza como trigger o disparador de eventos al entrar en contacto con cualquier otro actor del nivel. Este disparador puede servir para iniciar una colisión que provoque un bloqueo de movimiento (por ejemplo, cuando la caja de colisión de una pared entra en contacto con la caja de colisión del personaje, el movimiento de este automáticamente se bloqueará) o bien para activar o desactivar cosas (por ejemplo, cuando la caja de colisión del personaje entra en contacto con la caja de colisón de un interruptor de luz este desencadena en el encendido de una luz). En nuestro caso, las cajas de colisión de las partes movibles del arma tienen como función el desencadenar la animación de agarre cuando entra en contacto con la colisión de las manos del motion controller pawn. Sabiendo las dificultades que muchas veces presenta un primer acercamiento a la RV, conviene hacerlas lo más grande posible, para que toda persona que acceda a la simulación pueda agarrar las cosas con facilidad (ver Figura 3-26).

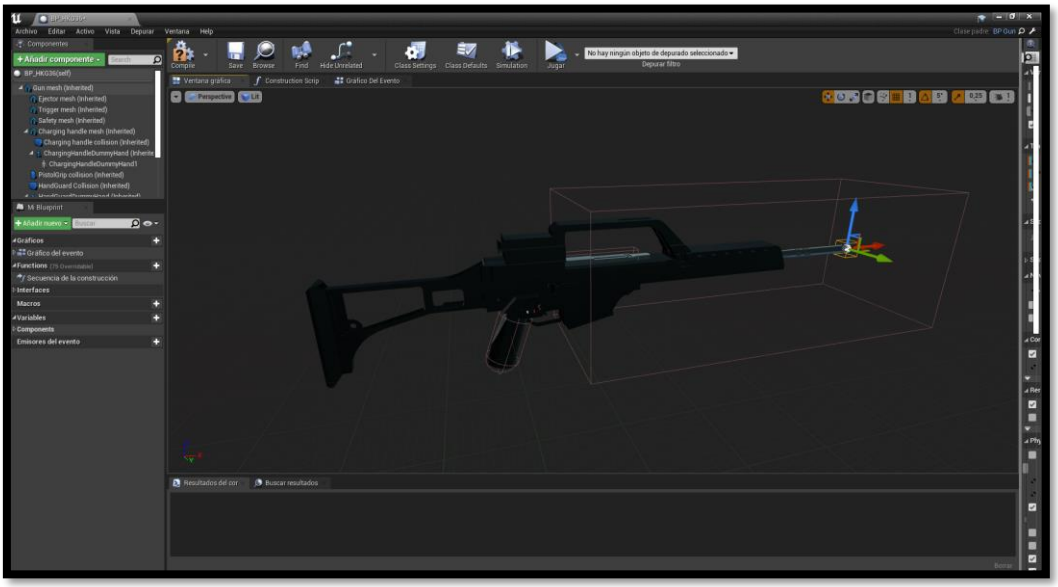


Figura 3-26 Configuración de las cajas de colisión del arma

Para configurar el segundo agarre se ha accedido a los detalles de la malla HandGuardDummyHand y se ha activado la opción de visible. Una vez la mano aparezca en la ventana gráfica cambiamos en el apartado Animation el Anim to Play e introducimos la animación HK G36_handguard_grip creada anteriormente en el blendspace. Al tener ya la mano con la posición de agarre específica del arma se ha procedido a su colocación de tal forma que quede de igual manera que en la vida real. Una vez tengamos la mano en la posición de agarre deseada, podemos volver a desactivar la opción de visible. Ahora la segunda mano, mientras la otra esté agarrando el fusil, agarrará el arma tal y como lo hemos configurado cuando la caja de colisión del HandGuard y las manos entren en contacto.

El siguiente paso ha sido ajustar el cargador al fusil. Recordemos que el fusil es una malla estática, y UE4 permite emparentar un objeto con la malla estática en el World Outliner, pero esto resulta en cierta manera algo tedioso ya que habría que colocar el cargador en el lugar exacto en relación con la malla. Sin embargo, al igual que se pueden adjuntar objetos a una malla esquelética, también se pueden adjuntar objetos a una malla estática utilizando sockets. Los sockets son comúnmente utilizados para adjuntar un objeto a otro mediante puntos de acople. Para crear un socket es necesario acceder al Editor de malla estática, colocarlo donde se quiera que esté el objeto en relación con la malla, y luego adjuntar el objeto, ya sea una luz, un efecto de partículas, o incluso otra malla estática (como es el caso del cargador). A continuación, se han añadido en el arma los sockets o puntos de acople pertinentes para adjuntar el cargador, el cierre y determinar el punto desde el que emitir los efectos de fuego al disparar.

Estos sockets serán la posición fija respecto al fusil que adoptará el cargador cuando entre en contacto con la caja de colisión del cargador del fusil. Así se logra determinar donde tiene que quedar acoplado el cargador cuando este es introducido. Para añadir un nuevo socket al fusil se ha accedido directamente al mesh del cuerpo del HKG36 que tenemos en la carpeta HKG36_CAD. En la pestaña socket manager se crea un nuevo socket llamado Magazine (cargador). Para determinar la posición del socket debemos de pensar en cómo queremos que quede el cargador una vez introducido. Para ello, visualmente podemos colocarlo directamente moviendo sus coordenadas con el eje que aparece sobre el socket. Además del cargador, se ha generado otro socket llamado Barrel que se ha colocado en la boca de fuego del cañón para determinar el lugar desde donde saldrá el vector que simulará la traza de disparo y otro socket llamado Ejector para determinar la máxima posición de retroceso del cierre al montarse.

La Figura 3-27 muestra la disposición de los sockets sobre el arma. Es conveniente recordar que la disposición de los mismos se hace de manera visual directamente desde el administrador de puntos de acople.

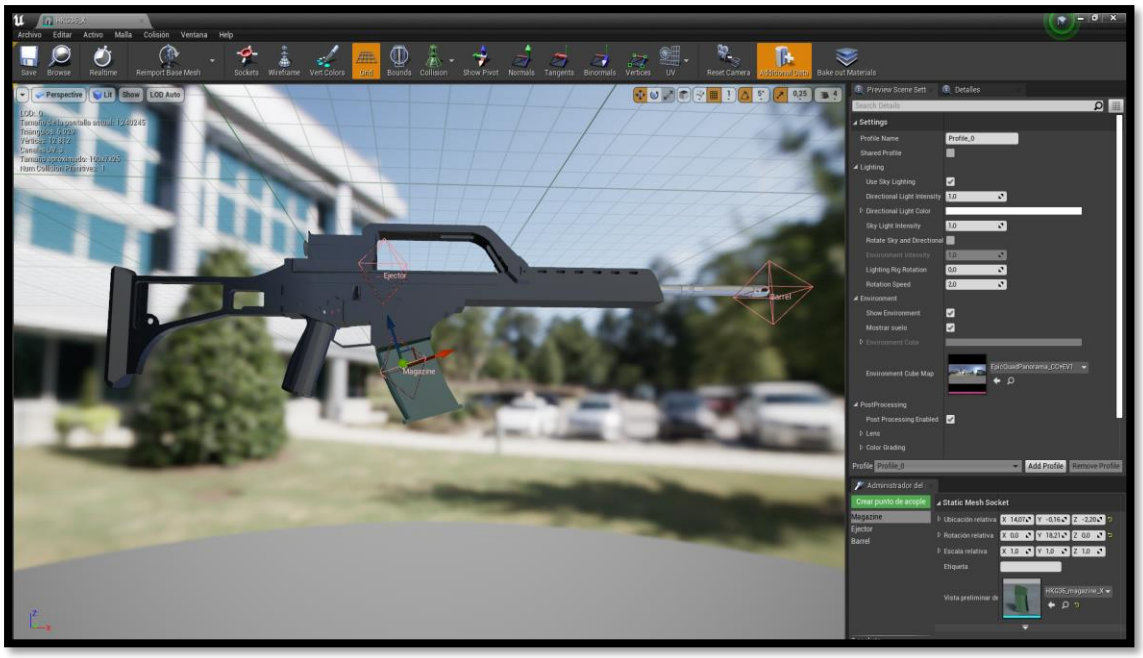


Figura 3-27 Colocación de puntos de acople o sockets del arma

Además de configurar el fusil también se han realizado una serie de modificaciones en el cargador. De igual manera que con el fusil, creamos un Child Blueprint Class a partir del BP_Magazine que se encuentra en la carpeta Content/VR_Weapons/Global/Blueprint. Creado este nuevo generado con las mismas características que el cargador por defecto lo renombramos como BP_Magazine_HKG36 y lo movemos a la carpeta HKG36_CAD. Igual que antes, al abrir este nuevo BP cambiamos el statich mesh por defecto por el cargador que necesitamos. Para el proceso de ajuste de cajas de colisión sólo hay que ocuparse de dos áreas en específico: el área de colisión para el agarre y el área de colisión que al entrar en contacto con el del arma fije el cargador en el socket creado anteriormente.

Se han añadido uno a uno los 30 cartuchos de bala de los que dispone el HKG36. En concreto, el fusil utiliza la munición 5,56x45mm NATO. Para hacer que el modelo de esta clase de munición extraído de internet vaya siempre junto al cargador y esté dentro del mismo se han añadido como componentes independientes dentro de la lista de componentes del BP_Magazine_HKG36 y se han colocado visualmente de la forma más realista posible (ver Figura 3-28).

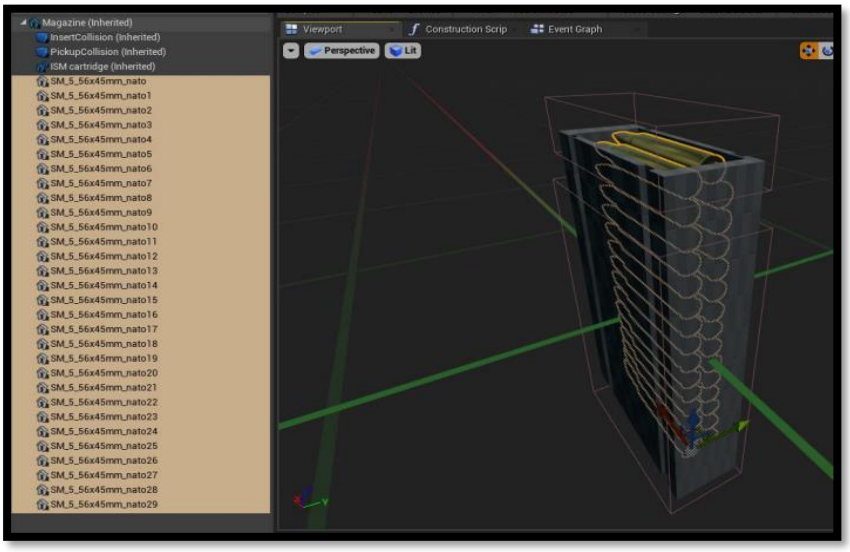


Figura 3-28 Colocación de la munición 5,56x45mm NATO en el cargador

Durante el proceso de disparo estas mismas balas serán llamadas en la función disparar. Para mejorar el rendimiento durante la simulación es conveniente procesar estas balas y hacer que pasen de ser mallas estáticas independientes a una sola malla estática instanciada. Esto es algo que utilizan numerosos desarrolladores para optimizar la utilización de un mismo actor cuando este aparece en grandes cantidades dentro de un mismo nivel.

Para lograrlo se han arrastrado los componentes de los 30 cartuchos directamente sobre el Construction Script del BP_HKG36. Para convertir todas las mallas en un array o matriz hay que conectar todos los pines de salida de las mallas con un nodo make array. Esto junta todas las mallas en una sola matriz a la que podemos llamar Cartidge Meshes (cartuchos) mediante el nodo set. Después mediante un bucle en continua repetición hacemos que los meshes dentro del array generado se conviertan en meshes instanciados mediante la función add instance. La función destroy component finalmente será la encargada de eliminar un componente del array cada vez que se dispare.

Para que el cargador reconozca el tipo de arma en el que debe ser introducido, hay que añadirle un Magazine Type con el nombre del arma que añadimos a la Weapons_Type (la lista de armas disponibles) al principio de este apartado. En concreto especificamos en el apartado default del cargador el Magazine Type HKG36.

Para comprobar que funciona correctamente llegados a este punto se han añadido los BP del arma y del cargador al nivel y se ha hecho click en simulate. Al acerca el cargador al arma se ha podido comprobar como este queda fijado en su posición.

Figura 3-29 Fusil y cargador acoplados

Por último, hay que defininir los sockets de las manos del maniquí para determinar los puntos donde queremos que queden fijos el arma y el cargador una vez agarrados. Para definir los nuevos sockets vamos directamente al MannequinHand_Rifht_Skeleton que se encuentra en la carpeta Content/VR_Weapons/Pawn/Mannequin/Character/Mesh. Al abrir la malla esquelética de la mano podemos añadir dos nuevos sockets haciendo click derecho sobre el hand_right (malla esqueleto de la mano completa). Los dos nuevos sockets creados se llamarán HKG36_GripSocket y HKG36_Magazine_GripSocket.

Para incluir estos sockets en la lógica de funcionamiento del arma y del cargador hay que irse a sus Blueprints y en la ventana de la izquierda buscar la función PickupWeapon. Dentro de la función select debemos de introducir el nombre del socket de nuestro nuevo socket creado. En el caso del cargador la función que maneja la lógica de agarre se llama PickupMagazine.

Lo siguiente ha sido añadir al nivel la mano derecha. Esta se encuentra en la carpeta Content/ VR_Weapon/ Pawn/ Mannequins/ Character/Mesh. Al soltarla sobre el nivel se cambia su Anim to play por la animación previamente creada HKG36_hand_grip. Haciendo click derecho sobre el arma en attach to se puede asignar al arma el socket creado. Ahora el arma se desplazará siempre con la mano ya que le hemos indicado que sus coordenadas ahora son relativas al socket.

Por último, se ha reorientado el arma de tal forma que ha quedado perfectamente ajustada a la mano. Una vez se ha conseguido la posición deseada, se han copiado las coordenadas del arma y han sido pegadas en las coordenadas de parámetros del socket para que este se reposicione justo donde queremos.

El mismo proceso serviría para recolocar el cargador en su posición de agarre.

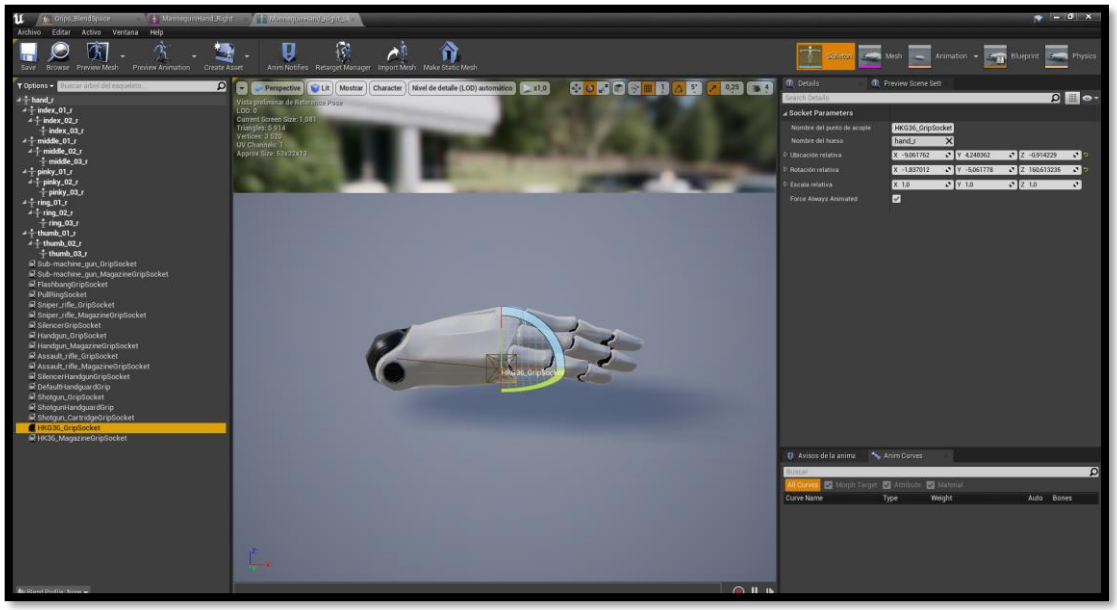


Figura 3-30 Colocación de sockets de agarre en la mano

3.3.2.3 Comportamiento del fusil

El arma al ser una Childblueprint class (actor hijo) de un arma ya generada cuenta con numerosas funciones. Estas funciones son directamente accesibles desde el BP_Gun y están ordenadas o enmarcadas dentro de tres funciones principales. A continuación, se listan las funciones más importantes de las que dispone la programación del fusil.

- ***Función Weapon* Firing** (ver Figura 3-31)***:*** incluye las funciones que manejan el disparo, así como sus efectos.

  - *Función LineTraceFromBarrel*: El disparo en el arma se simula mediante un vector de dirección que sale directamente desde el punto de acople que generamos previamente llamado Barrel mediante una función line trace by channel. El vector trazado tiene una longitud de 2 km y cuenta con el efecto ricochet característico de todo lanzamiento balístico que genera un ángulo de rebote cuando detecta el impacto.
  - *Función Particle Effects*: maneja los efectos de partícula que salen de la bocacha de fuego.
  - *Single Shot Sound Effects*: se encarga de generar los efectos de sonido al disparar. El sonido de disparo.
  - *Report noise event to IA*: en caso de programar una inteligencia artificial con capacidad de escucha esta función genera el evento necesario para que nos detecten.

- *Bullet impact sound and particles*: en función del material donde colisione por vez primera el vector generado este reaccionará emitiendo un sonido desde el punto de impacto mediante la función Spawn Sound at Location.
- *Bullet Decals*: cuando el vector impacta, además de generar un sonido, deja una marca que simula la traza que dejaría un impacto de bala mediante la función Generar Textura Decorativa Adjunta. También es función del material sobre el que impacte y está programada para desaparecer a los 15 segundos tras el impacto.
- *Apply physics and damage*: Añade el impacto que genera el vector al entrar en colisión con algo. Además, si la superficie de impacto cuenta con un sistema de vida este se ve comunica directamente con esta función a través de la función Bullet Damage.
- *Randomise Recoil*: ajusta el valor del retroceso al disparar en función de si se está agarrando el arma con una o con dos manos. Esto lo hace mediante un rango de números decimales (float) de los que extrae un valor aleatorio.

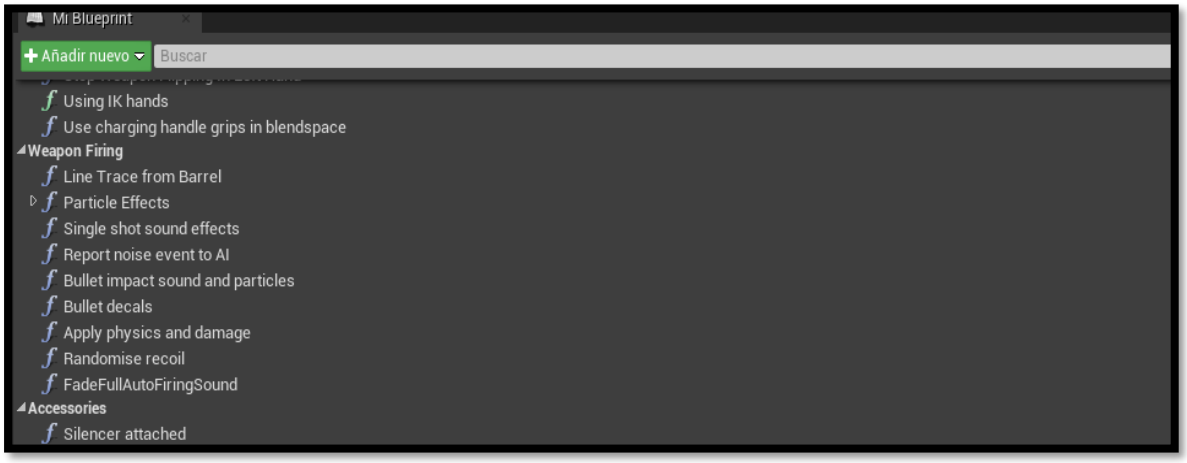


Figura 3-31 Funciones de disparo por defecto del arma

- ***Función Motion Controllers:*** son todas las funciones que comunican el arma de manera directa con los controladores.

  - *Haptics:* sirve para llamar a las funciones hápticas. Estas funciones generan las vibraciones que después el jugador siente sobre los controladores al disparar o recibir impacto.
  - *Stop Haptics:* que sirve para parar las vibraciones sobre los controladore.
  - *Using IK Hands:* Comprueba si el motion controller está usando IK Locomotion.
  - *Use charging Handle grips in Blendspace:* consigue hacer que la mano virtual use la mezcla de animaciones previamente creada en el Blendspace de acorde a los movimientos y acciones sobre los controladores.

- ***Función Reloading:*** con todas las funciones del proceso de recarga del arma.

  - *Ready to fire:* maneja las condiciones mediante las cuales el fusil está listo para disparar, mediante un nodo AND, que devuelve una variable booleana en verdadero cuando se cumple que están en verdadero las variables booleanas de One in the chamber (indica que hay una bala aún en la recámara) y Weapon Cocked (variable que se activa al cargar el arma con el cierre).
  - *Load next round into chamber:* con la condición de que aún tenga balas y el arma esté cargada esta función simula la introducción de una bala en la recámara automáticamente cuando el arma dispara. Recordemos que esto solo tiene valor para perfeccionar las condiciones que

deben tener las variables ya que el fusil al disparar no lanza un proyectil, sino un vector de dirección.

- *Release Magazine:* detecta cuando el arma tiene introducido el cargador o no. Para que la variable booleana Weapon Cocked esté activa debe tener el cargador introducido como condición.

Estas son muchas de las funciones con las que el arma cuenta para desarrollar la lógica de disparo. Sin embargo, a la hora de adecuar el arma a nuestras necesidades existen ciertos parámetros que han sido modificados. Estos parámetros por lo general son fácilmente accesibles por el usuario.

En la class default del BP_HKG36 se pueden cambiar muchas variables (ver Figura 3-32), por ejemplo, podemos dictaminar cuanta distancia queremos que recorra el cierre cuando este amartilla el arma directamente desde el apartado charging handle max distance. En el apartado Recoil podemos retocar los retrocesos del arma al disparar en función de cómo esté siendo agarrada (una o dos manos). Para añadir los efectos de sonido y partículas, se han usado los que podemos encontrar dentro de la carpeta de assault rifle.

Figura 3-32 Configurando variables del arma

3.3.3 Programación de la Inteligencia Artificial

Hay tantas maneras de programar inteligencias como programadores, sin embargo y por ser estructurados se ha procedido con las herramientas que nos proporciona UE, en donde la creación de Inteligencia Artificial (IA) para personajes u otras entidades se logra a través de múltiples sistemas que trabajan en sintonía, desde un árbol de comportamiento que se ramifica entre diferentes decisiones, acciones o tareas, como ejecutar una consulta para obtener información sobre el entorno a través del sistema de consulta del entorno (EQS) hasta utilizar el sistema de percepción de inteligencia artificial para recuperar información sensorial como la vista, el sonido o la información de daños. Cuando se crea la IA en UE4 y se utiliza cada uno de estos sistemas, una buena forma de pensar es que el proceso de toma de decisiones lo manejan los Árboles de Comportamiento, los estímulos del entorno (como la

información sensorial) se envían a los Árboles de Comportamiento desde el sistema de Percepción de la IA, y las consultas sobre el propio entorno se manejan a través del EQS. Todos estos sistemas juegan un papel clave en la creación de inteligencias artificiales creíbles.

Toda la programación del árbol de comportamiento se hace desde el Behavior Tree Editor; el editor que permite definir la IA a través de un sistema visual basado en nodos (similar a los Blueprints) para los actores del nivel, por ejemplo, diferentes enemigos, personajes no jugadores, vehículos, etc.

En concreto, se busca una inteligencia artificial que sea capaz de reaccionar a nuestra presencia y que además trate de eliminar al usuario de manera hostil. Recordemos que el uso y la implementación de una IA que funcione correctamente aumentará en gran medida la sensación de inmersión buscada. Por eso la programación de los NPC,s primero otorgará a la inteligencia artificial una lógica de comportamiento avanzada capaz de patrullar una zona navegable que también definiremos.

Toda IA programada en Unreal Engine nace de una serie de activos que manejan los sistemas previamente explicados. En nuestro proyecto se ha creado una carpeta en Content/VR_Weapons/IA donde se generarán los activos necesarios. En primer lugar, se crea un árbol de comportamiento al que llamamos BT_Enemy, un AI Controller llamado Enemy_controller, añadimos un Character (extraído de la MarketPlace y gratuito) al que llamamos Enemy_character, un BP_PerceptionBlocker y por último una Blackboard que nombramos como BB_Enemy.

3.3.3.1 Árbol de comportamiento

El Árbol de Comportamiento o Behaviour Tree es un activo que ofrece Unreal Engine que se utiliza para ejecutar ramas que contienen lógica y para determinar qué ramas deben ser ejecutadas. El Árbol de Comportamiento se basa en otros dos activos, el llamado Pizarra o Blackboard que almacena toda la información necesaria definiendo las variables que después el árbol leerá, y el AI Controller que actúa como el "cerebro" de un Árbol de Comportamiento.

La Pizarra o *Blackboard* contiene varias claves definidas por el usuario que contienen información utilizada y que desencadenarán las acciones futuras del NPC. Por lo que un primer paso para generar la IA debe de ser definir con qué claves será capaz de tomar todas las decisiones. Estas claves al ser variables también pueden ser de distintas clases: booleana, enteros, strings, floats… Habrá tantas clases de claves como tipos de variables existen. En nuestro caso se han definido las siguientes claves dentro de la BB_Enemy (ver Figura 3-33):

Claves de tipo *booleano*

- Has line of sight (el enemigo tiene o no al jugador en su campo de visión)
- Stunned (el enemigo está o no aturdido)
- Overlapping (el enemigo se superpone en su patrulla con otro o no)
- Can patrol (el enemigo puede patrullar o no)
- Is armed (el enemigo está o no armado)

Claves de tipo *actor*

- SelfActor (define el actor del personaje del propio enemigo)
- EnemyActor (define el actor del personaje que controlamos, el motion controller pawn)

Claves de tipo *vector*

- PatrolLocation (define la posición que ocupa el enemigo en su patrulla)
- Last known enemy location (define nuestra última posición registrada por el enemigo)

Definidas las claves, podemos dar forma al Árbol de Comportamiento BT_Enemy (ver Figura 2-1). La estructura del árbol debe de ser familiar para todo el que conozca el funcionamiento de un diagrama de flujo ya que éste basa su arquitectura en el mismo concepto de mostrar los resultados dependiendo de la elección tomada.

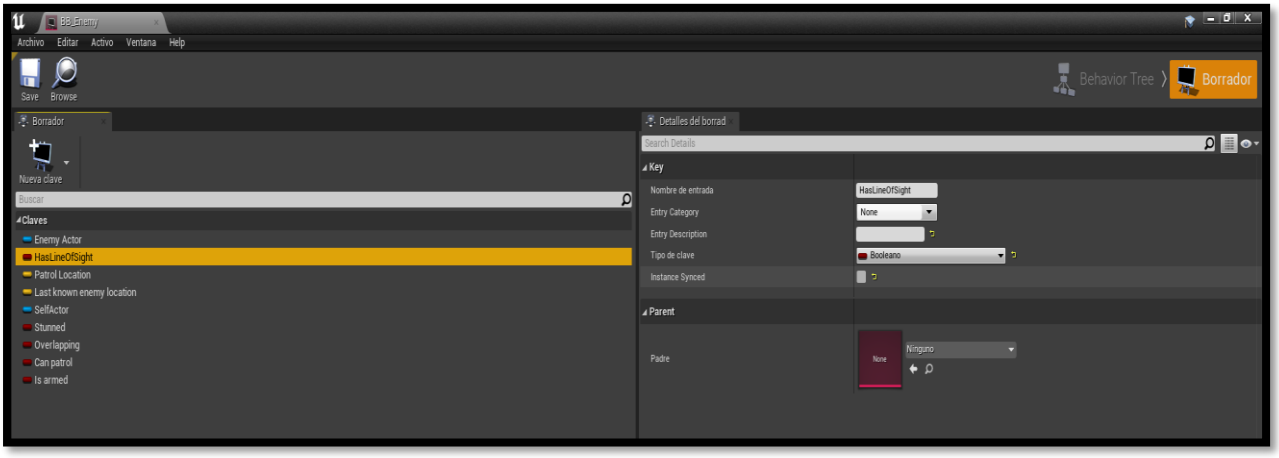


Figura 3-33 Definiendo las claves de la IA desde el Blackboard

Nuestro árbol de comportamiento cuenta con 4 ramas principales. Como veremos dentro de estas ramas las decisiones implican unas acciones que son ejecutadas como tareas a las que llamamos BTT seguido del nombre de la tarea en inglés. Dentro de cada tarea se encuentra la programación que controla cada una en específico. Adjuntas en el Anexo I: Bloques de código: Tareas del árbol de comportamiento de la IA se encuentran detalladas en forma de Blueprint cada una de las tareas que serán nombradas en la explicación. A continuación, se explican todas y cada una de las ramas con la lógica que encierran.

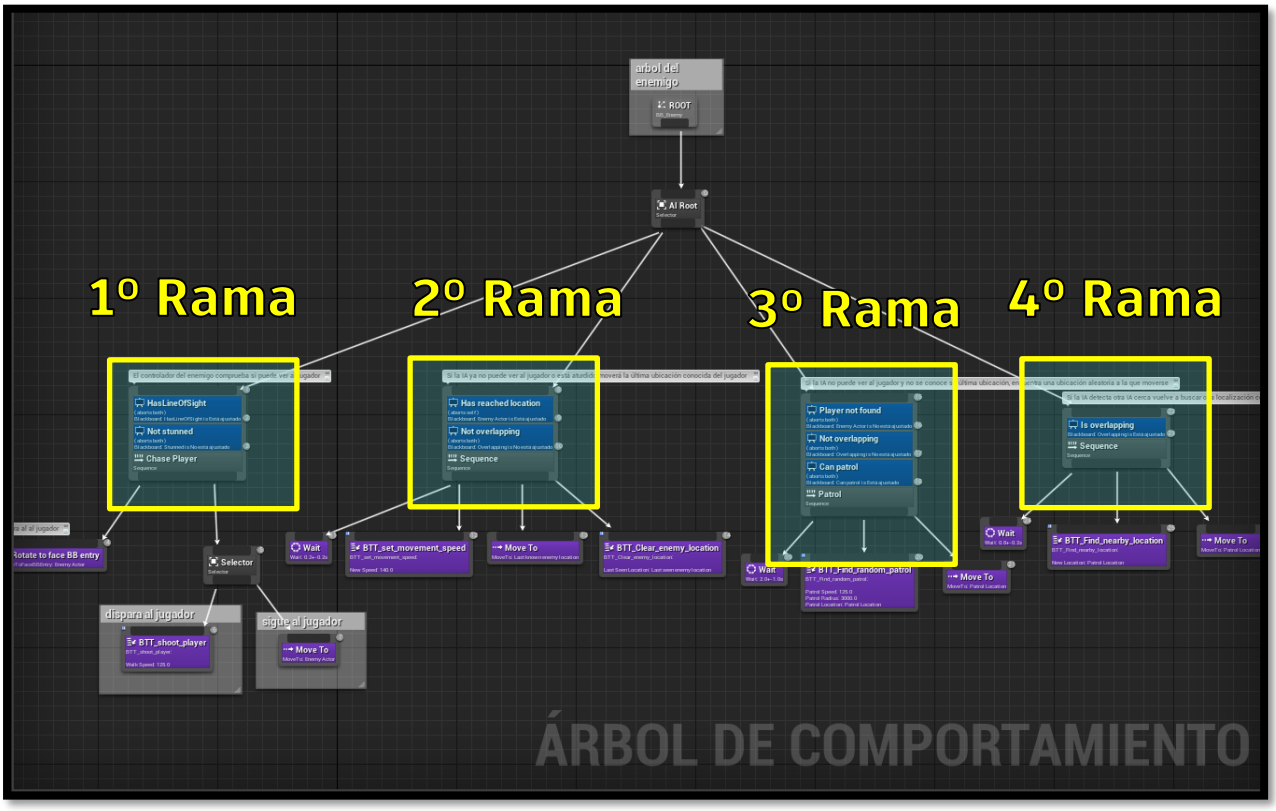


Figura 3-34 Árbol de comportamiento de la IA programada

Primera Rama (ver Figura 3-35): Decide en función de si el enemigo puede ver al jugador o no.

Dentro de la lógica de esta rama, por orden de revisión, se establecen las siguientes condiciones para continuar bajando en la secuencia: clave HasLineOfSight debe estar activa y la clave Stunned desactivada. Esto significa que el enemigo nos tiene en su campo de visión y no se encuentra aturdido. Si esto es así, entonces la secuencia sigue con la ejecucion de Rotate to face BB Entry, que implica la rotación del enemigo hasta encararnos. Después de esto, la rama se divide en dos acciones simultáneas. Entonces el enemigo ejecutará la tarea BTT_Shoot_player (comienza a dispararnos) y la tarea Move to (nos sigue mientras dispara).

Segunda rama (ver Figura 3-35): Controla las acciones del enemigo cuando pierde de vista al objetivo.

Por orden de revisión, se comprueban las variables de tipo vector Enemy Actor y la booleana Overlapping. Si el enemigo tiene registrada nuestra última posición y no está superponiendo su espacio al de otro enemigo se generan cuatro acciones seguidas. Primero espera 2 segundos (función Wait), después aumenta su velocidad con la BTT_set_movement_speed, se mueve hasta la última posición registrada (función Move to) y por último olvida la última localización registrada en la que ya se encuentra con la BTT_Clear_enemy_location.

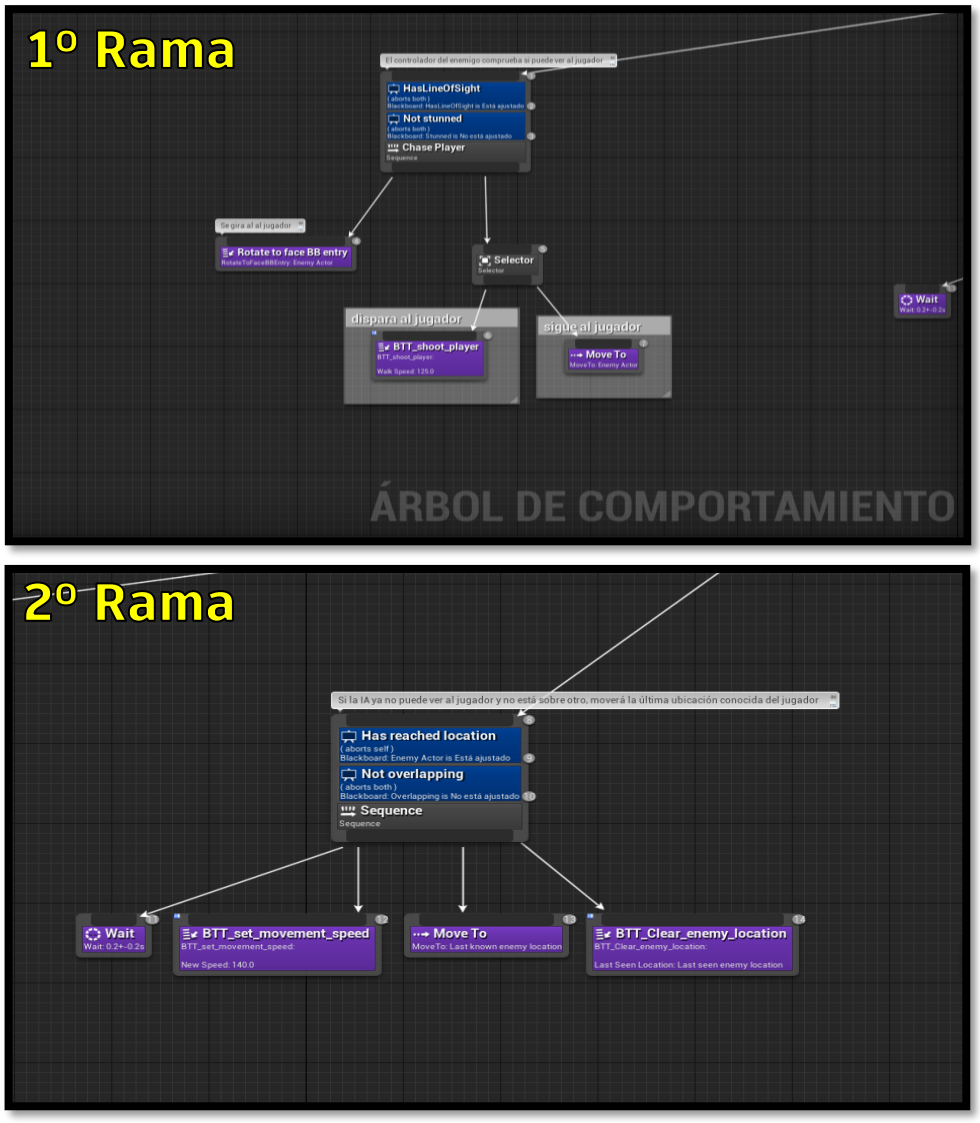


Figura 3-35 Secuencias de la 1º y 2º rama del árbol de comportamiento

Tercera rama (ver Figura 3-36): Cuando la IA no puede ver al jugador y no tiene una última posición registrada se desplaza a una nueva posición aleatoria.

Primero se revisa la variable vector Enemy Actor y las variables booleanas Not overlapping y Can patrol. En caso de que no vea al jugador, y no tenga última posición registrada y además no esté en posición interfiriendo contra IA y pueda patrullar de nuevo se suceden tres tareas. Primero espera 2 segundos, después busca una nueva posición para patrullar con la BTT_Find_random_Patrol y por último ejecuta un move to para acudir a esa última posición.

Cuarta rama (ver Figura 3-36): en esta última rama la IA actúa frente al posible sobre posicionamiento respecto a otra IA.

Si la variable booleana Overlapping está activada, automáticamente se inicia una secuencia de tres acciones en el siguiente orden: espera dos segundos, busca una nueva posición cerca donde patrullar y por último se dirige a ella con un move to.

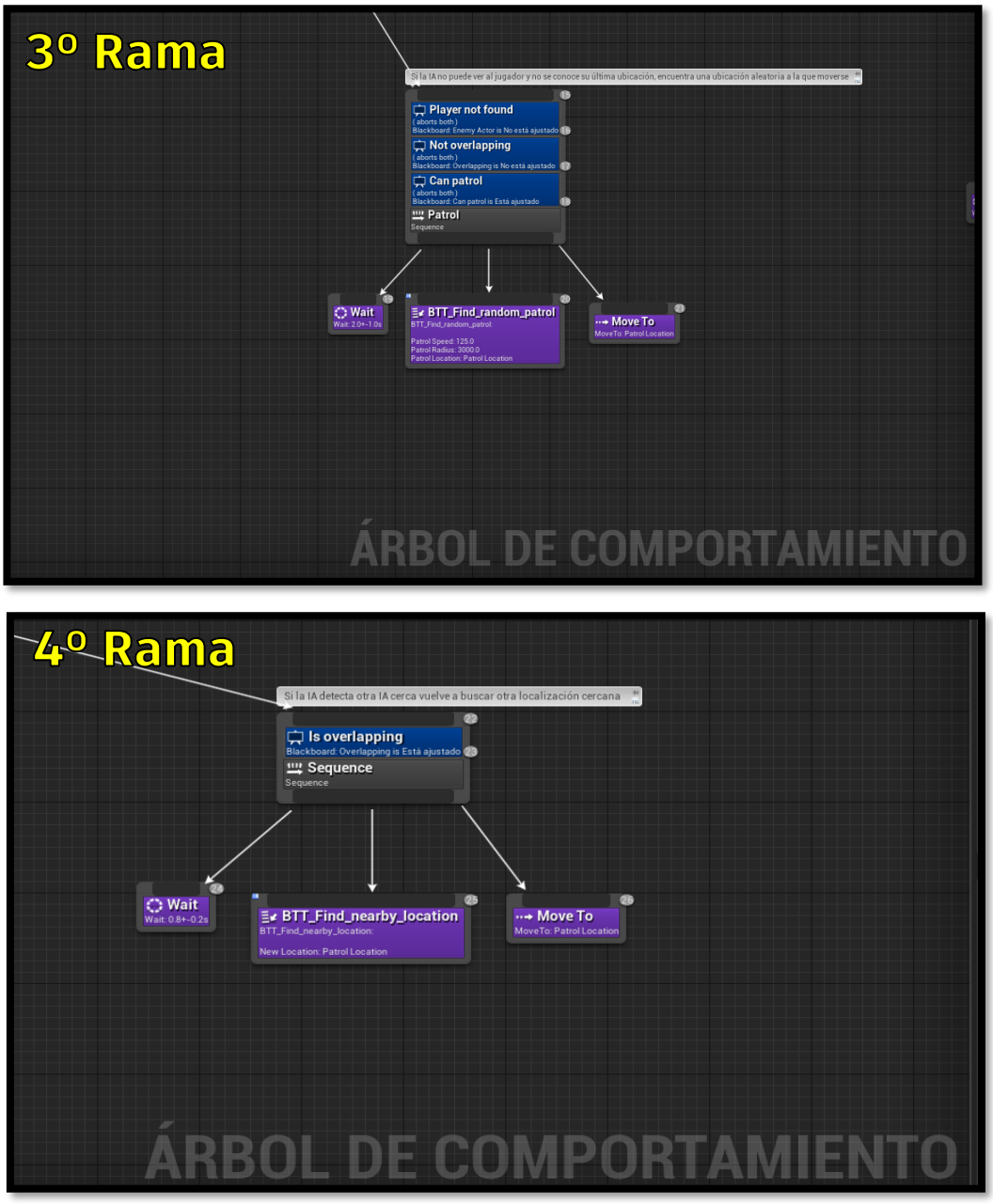


Figura 3-36 Secuencias de la 3º y 4º rama del árbol de comportamiento

3.3.3.2 Controlador de la IA

El AI Controller se centra más en responder a la información del entorno. El trabajo del AIController es observar el mundo que le rodea y tomar decisiones y reaccionar en consecuencia sin la intervención explícita de un jugador humano. Mientras que en el árbol de comportamiento se establecía un orden y una lógica para tomar acciones, en el controlador de la IA se ejecutan acciones inmediatamente si se dan las condiciones necesarias sin un proceso previo de toma de decisión.

Además, el controlador de la IA al manejar la interacción con el entorno se encargará de establecer el estado de las variables. En concreto, en nuestro controlador de IA se establecen los estados de las claves Is armed, Can patrol, Overlapping, Has line of sight, Enemy actor, last known enemy location a través de la codificación mediante Blueprints.

Como ejemplo se explicará la lógica seguida en la programación del Blueprint que activa o desactiva las variables has line of sight, enemy actor y last known enemy location.

En primer lugar, el BP se inicia con el evento On target Perception Updated el cual tiene lugar cuando algo entra dentro del campo de visión del enemigo (previamente definido mediante el bloque de percepción). El enemigo comprueba con las funciones Target is actor si el actor detectado es un arma o el jugador. Recordemos que el bloque de percepción detecta tanto sonidos asociados a nuestra arma como a nuestro jugador en sí si lo tiene en visual. Como queremos que el enemigo entienda que somos nosotros independientemente de la forma de detección, usamos un nodo OR como condición al que unir los dos pines de salida del Target is actor weapon y player. Al salir del nodo OR con cualquiera de esas dos condiciones, llegamos a un nodo AND que pone la condición de que también se compruebe que el enemigo no se encuentra muerto. Esto lo hace comprobando la variable booleana dead que generamos cuando se programa el BP del enemigo y que se explicará más adelante. Si alguna de las dos variables exigidas no está activa se pone en marcha el contador de línea de visión del enemigo que tiene 4 segundos para decidir que definitivamente ha perdido de vista al jugador. En cambio, si tiene algo en el campo de percepción y no está muerto (salida en verdadero después del AND) se inicia un proceso secuencial que activa la variable booleana has line of sight mediante una función Set Value as Bool, la variable actor Enemy Actor con una función Set Value as Object y la variable tipo vector Last Known Enemy Location con la función Set Value as Vector. La programación del Blueprint queda expuesta Figura 3-37.

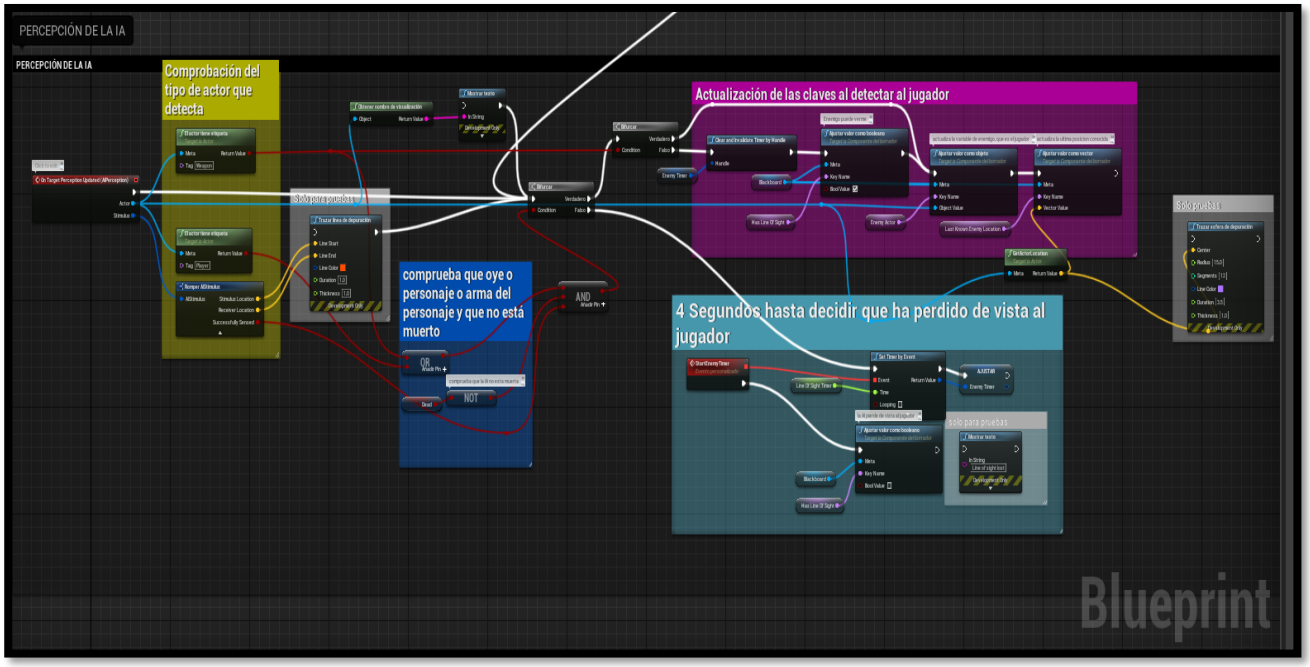


Figura 3-37 Blueprint de la percepción de la IA

3.3.3.3 Mecánicas del enemigo

Las mecánicas definen los límites de percepción y actuación del enemigo. Además, incluye la programación de los movimientos que realiza la Inteligencia Artificial (ver Figura 3-40).

Como hemos visto, el enemigo es capaz de patrullar. Sin embargo, ¿cómo sabe por dónde puede patrullar? Esto es lo primero que define la mecánica del enemigo. Sus movimientos quedan limitados a lo que queremos que haga en todo momento, por lo que lo primero es definirle su propia área navegable.

Esto se hace arrastrando desde el panel Place Actors a la izquierda del viewport del nivel el elemento Nav Mesh Bounds Volume. Con el Nav Mesh Bounds Volume seleccionado al presionar R este se reescala a todo el área que encuentre disponible a su alrededor (ver Figura 3-38). En nuestro caso al soltar el nav mesh en medio de la casa ha detectado todo el espacio ocupado por el suelo de la casa . Situando un Nav Modifier Volume (también seleccionable desde el place actor) definimos un volumen que superpuesto al nav mesh elimina este último. Esto lo usamos para evitar que los enemigos salgan de sus habitaciones. Desde el menú de detalles del enemigo le asignamos el área de patrulla creado con el nav mesh.

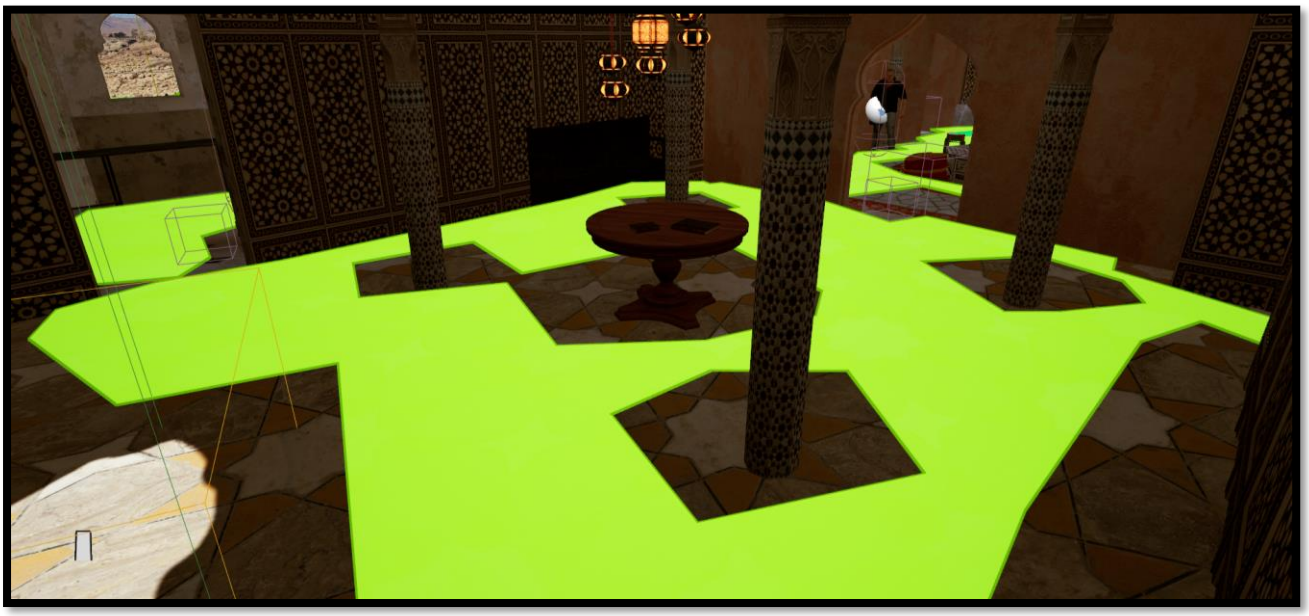


Figura 3-38 Determinando el área de patrulla de la IA mediante el navmesh

La siguiente mecánica ha sido configurar la percepción del enemigo. En el apartado anterior, hemos visto qué decisiones toma el enemigo en función de lo que detecte, pero ¿cómo es capaz de detectar cosas? El componente de Percepción AI es un tipo de componente que puede añadirse al AI Controller directamente desde la ventana de Componentes y se utiliza para definir qué sentidos tiene activos la IA. Con esto conseguimos hacer que el enemigo vea, escuche cosas y tenga sentido del daño recibido. Desde aquí se ajustan los parámetros que queramos cambiar de los sentidos.

En nuestro caso tras añadir el AI Perception (ver Figura 3-39) al AI Controller le asignamos tres tipos de sentidos:

- El sentido de la vista (AISense_sight): podemos cambiar su radio de visión. En nuestro caso configuramos un radio de 25 m.
- El sentido del daño (AISense_damage): es el sentido por el cual el enemigo detecta que ha recibido algún tipo de daño.

- El sentido de la escucha (AISense_hearing): se establece la capacidad de percepción sonora del enemigo, la cual dejamos establecida en 30 m de radio.

Para que el AI Perception funcione es importante que el actor que controla la Inteligencia Artificial (el Enemy_character) cuente con un bloque de percepción (el BP_Perception_Blocker) desde el que se trazarán los radios de visión y de escucha.

El resto de las mecánicas del enemigo serán directamente programadas en el Blueprint del Enemigo (BP_Enemy_character). Estos Blueprints establecen el comportamiento del enemigo en cuanto se inicia la simulación, la forma en la que es capaz de disparar, el sistema de vida y daño del enemigo y cómo ejecuta las acciones tras estar superponiéndose en el espacio con otra IA. Todos los Blueprints se encuentran en el Anexo II: Bloques de código: Inteligencia Artificial, explicados visualmente.

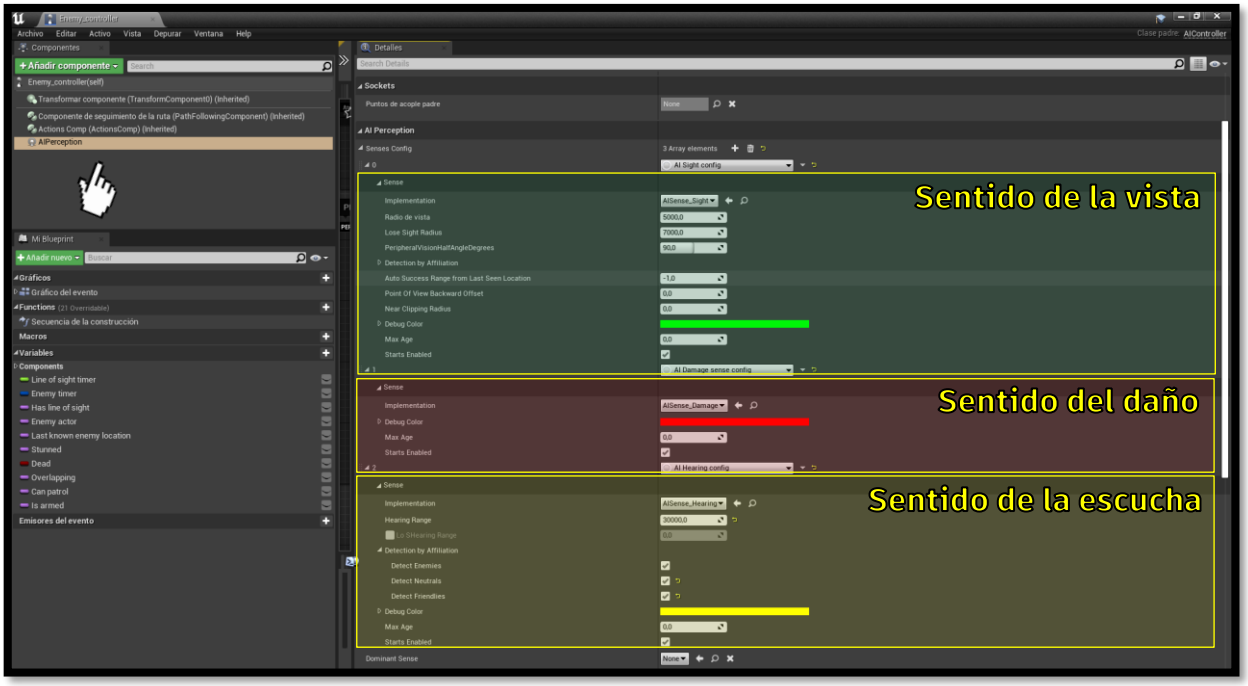


Figura 3-39 Configuración de los sentidos de la IA desde el AI Perception

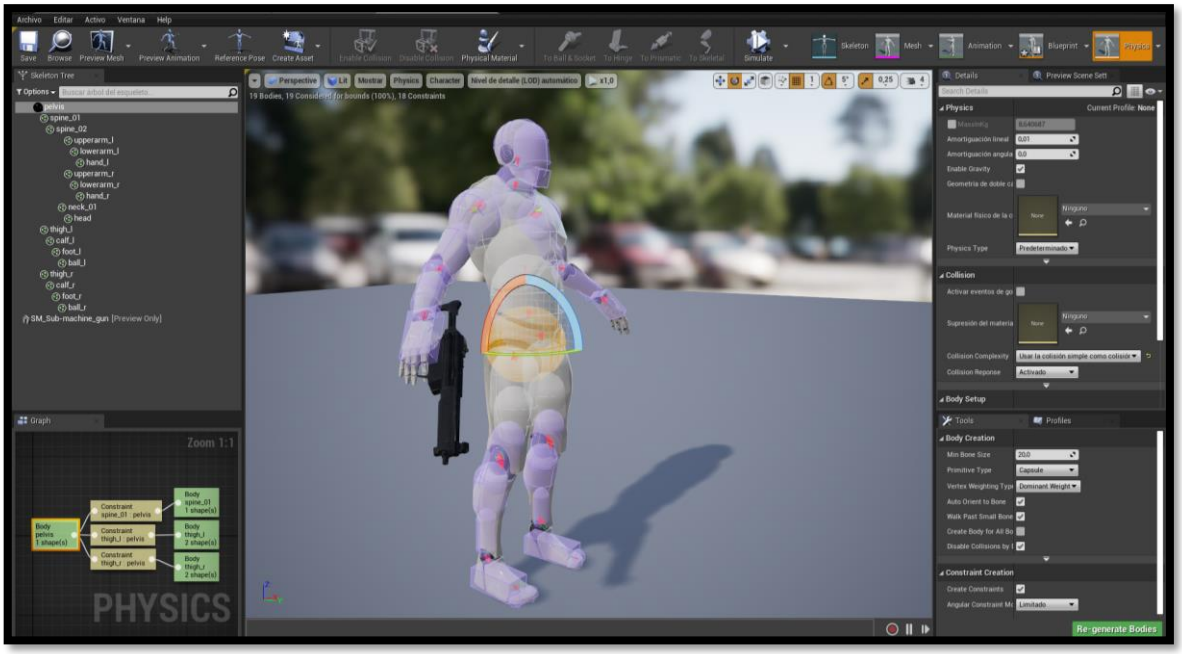


Figura 3-40 Acople del arma a la malla esquelética del enemigo

3.3.4 Diseño del nivel

El diseño de nivel es la fase del proceso de desarrollo cuyo objetivo general es crear situaciones o eventos interactivos dentro del entorno para desafiar al jugador y mantenerlo interesado. El diseño de nivel reúne todos los elementos previamente diseñados para dar forma a la experiencia del jugador: mecánica del juego, jugabilidad, obstáculos, historia, etc.

Como dice Cliff Bleszinski, antiguo diseñador jefe de niveles de Epic Games:

"El desarrollo de un juego puede compararse con la construcción de un coche. Tienes todas estas partes diferentes creadas por gente con talento -programación, modelado, sonido y arte- y en algún momento, el duro trabajo de todos en un coche se une, y los neumáticos salen a la carretera. En un juego, el trabajo de todos se mantiene unido por los niveles que utilizan todo eso, y más vale que sean excepcionales o el juego se tambalea". [58]

Por eso antes de comenzar con el diseño del nivel se ha planificado cómo queremos enfocar la experiencia; qué queremos transmitir al jugador cuando inicie la inmersión.

En primer lugar, hay que tener claro el propósito. En nuestro caso es la instrucción de los alumnos de la Escuela Naval Militar en un ambiente hostil donde puedan poner en práctica la táctica previamente adquirida en las aulas.

El planteamiento de nivel propuesto es el siguiente:

El alumno deberá realizar una limpieza completa de la casa objetivo (aplicando la táctica necesaria) con un tiempo limitado para ello (añade un componente de estrés además de obligar al alumno a controlar en todo momento la situación de la incursión). El fin de la misión es realizar el rescate de un grupo de rehenes sin que estos sufran ningún tipo de daño.

3.3.4.1 Widgets en RV

Para conseguir una experiencia bien lograda, en la que el jugador solo tenga que hacer exactamente lo que queremos, debemos de guiarlo en sus primeros pasos. Con unas simples indicaciones de texto conseguiremos realizar la conexión necesaria con el jugador para transmitirle lo que queramos. En nuestro caso, es necesario que el jugador al empezar la simulación sepa lo que hacer, por lo que incluiremos dos indicaciones: una para indicarle que debe de agarrar el fusil y otra antes de entrar en la casa donde se le advertirá de que dispone de un tiempo limitado para llevar a cabo su misión.

Los textos y menús interactivos que normalmente ayudan al jugador en la mayoría de las simulaciones se llaman Widgets.

En un juego de escritorio, los widgets están en lo que se denomina Screen Space o espacio de pantalla, lo que significa que, por un lado, el menú está bloqueado y fijado a la vista del jugador y por otro que está proyectado ortogonalmente, es decir, no aplica una perspectiva 3D. Sin embargo, estos factores beneficiosos para el desarrollo de simulaciones de escritorio presentan problemas importantes para la RV ya que, entre otros, un menú proyectado ortogonalmente en RV actúa como si estuviera infinitamente lejos, por lo que es complicado enfocar en un entorno estereoscópico como es el de la RV.

La solución más simple es no utilizar la proyección ortográfica y adecuar el widget para que tenga una representación en el mundo 3D. De esta manera se resuelve por un lado el problema de la ortogonalidad ya que el widget se verá afectado por todas las transformaciones utilizadas en el juego y por otro el problema de la periferia, dando opción al jugador de girar y fijar el punto en el que enfocarse.

La creación de widgets se realiza mediante el Unreal Motion Graphics (UMG). Los Widgets se editan en un Widget Blueprint especializado, que utiliza dos pestañas para su construcción: la pestaña Designer permite la disposición visual de la interfaz y las funciones básicas, mientras que la pestaña Graph proporciona la funcionalidad detrás de los Widgets utilizados.

Lo primero ha sido generar una nueva carpeta llamada Widgets dentro de la carpeta Content. Los widgets del que ayudarán a guiar al jugador se han creado dentro de otra carpeta llamada TutorialInfo. Se han generado en primer lugar una lista donde introducimos el número de widgets que queremos generar con las mismas características. En total enumeramos 2 áreas. Una para el primer widget y otra para el segundo. Después haciendo click derecho, generamos un nuevo Blueprint Widget al que llamamos TutorialInfo. Dentro del Blueprint en Diseñador (designer) se han generado desde la paleta en la parte izquierda por orden de jerarquía un panel de lienzo y un bloque de texto. Este bloque de texto debemos activarlo como variable desde la ventana de detalles del Widget. En el apartado Gráfico (graph) del mismo Widget generamos los dos textos que queremos que muestren.

El texto del área 1 debe queda definido por una variable string: “Agarre su fusil y prepárese para asaltar el objetivo.” El texto del área 2 también queda definido por una variable string: “Según informes de inteligencia sólo dispone de 7 minutos para completar su misión antes de que el aparezcan refuerzos del enemigo.”

En función del actor al que asociemos el widget se mostrará un texto u otro. Esto se hace con la función Set Text que actualiza el bloque de texto variable que hemos generado anteriormente.

Lo siguiente ha sido generar un Blueprint class de padre actor en la carpeta de los TutorialInfo que llamaremos BP_TutorialInfo. En la ventana gráfica de este BP añadiremos el componente Widget. Una vez añadido el componente widget, le decimos cuál de los widgets queremos que se asocie al actor en el Widget Class desde el apartado User interface de los detalles (ver Figura 3-41). Aquí se ha seleccionado el widget TutorialInfo previamente generado. Ahora el actor muestra el texto en función del que elijamos desde los detalles del componente propio del BP_TutorialInfo en Tutorial Info que se encuentra en la opción Default. Para generar los dos textos solo hay que duplicar sobre el nivel el mismo BP_TutorialInfo y asignar al primero el área 1 y al segundo el área 2.

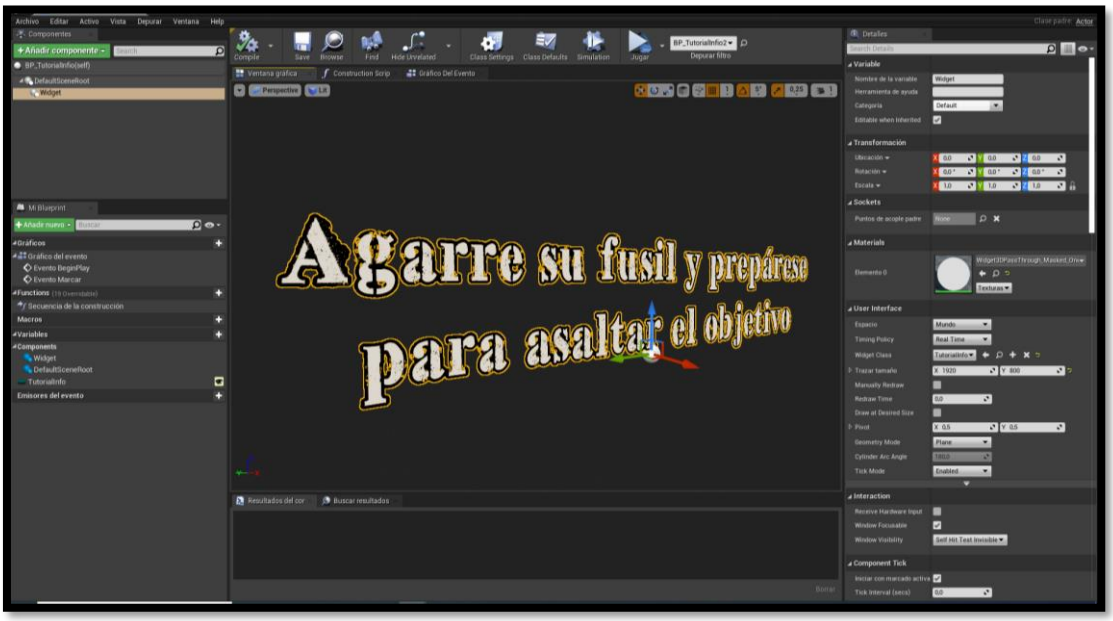


Figura 3-41 Asociando el Widget de información a un actor para situarlo sobre el nivel

Como queremos que el widget sea leíble por el jugador en todo momento, debemos de forzar unas transformaciones del actor de tal manera que este siempre muestre el texto independientemente de la orientación desde donde sea observado. Esto se ha hecho desde el gráfico de evento del BP_Tutorial Info. Mediante un bucle de infinita revisión, se define una nueva rotación en función de la cámara que gestiona la visión del jugador y se asigna al actor que alberga el widget (DefaultSceneRoot) con un nodo Set World Rotation. La programación del Blueprint queda expuesta en la Figura 3-42.

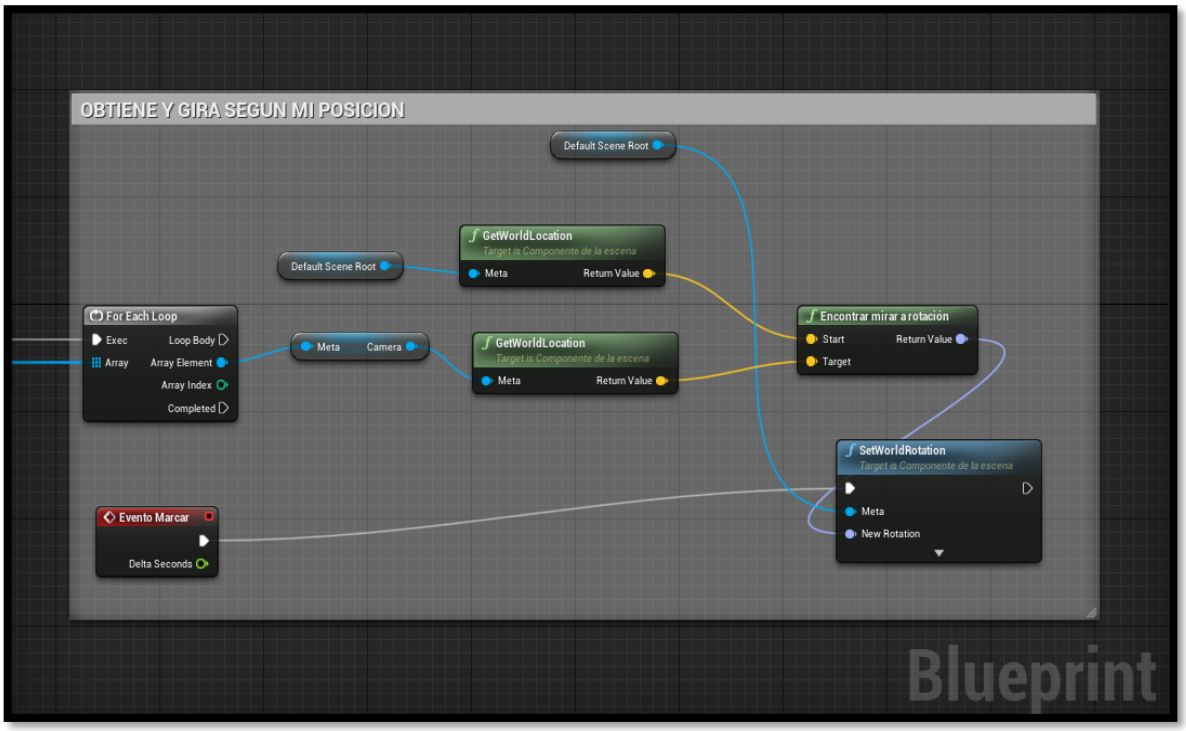


Figura 3-42 Blueprint para orientar el Widget según nuestra visión

Además del Blueprint que controla el widget, la lista de widgets y el widget general, en la carpeta Widget se han generado también dos accionadores cúbicos o trigger box. El trigger box Enter house sirve para lanzar el evento Start Timer (explicado en el siguiente subapartado) y acabar con el sonido ambiente al entrar en la casa. El trigger box Exit house sirve para volver a activar el sonido ambiente una vez salimos de la casa. Ambos han sido colocados en las inmediaciones de la puerta de entrada. La programación del Blueprint queda expuesta en la Figura 3-43.

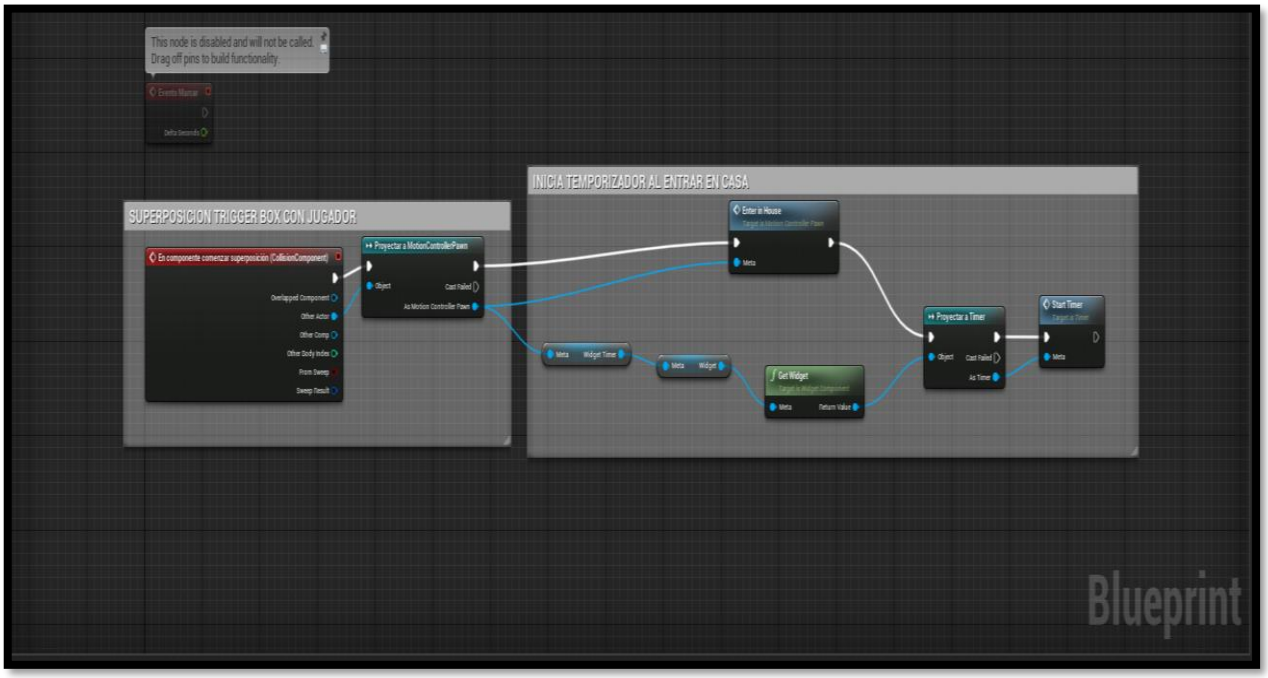


Figura 3-43 Blueprint que inicia el temporizador al entrar en la casa

3.3.4.2 Temporizador: su uso como game manager

La inclusión del temporizador debe de ser lo más natural posible y debe de acompañar siempre al jugador una vez entre en la casa objetivo. Por eso, podemos referirnos al temporizador como un elemento del HUD (heads-up display) o barra de estado. Esto es un método por el cual la información se transmite visualmente al jugador como parte de la interfaz de usuario. Además, el temporizador será lo que determine si se ha logrado el objetivo o no. En caso de que el jugador elimine todos los enemigos antes de que el temporizador se vuelva 0 la misión se dará por cumplida. Si en cambio, no se logra acabar con los enemigos en tiempo, la misión se dará por fallida.

Para generar el temporizador primero se ha creado un widget tal y como generamos los dos widgets en el apartado anterior. El widget para el temporizador se ha llamado Timer y se ha guardado dentro de Content/Widgets/HUD.

Desde la paleta se han añadido por jerarquía un panel de lienzo, una caja horizontal que situamos en la esquina inferior derecha del screen space y dentro de la caja incluimos: una caja de texto variable llamada MinutesTimer enlazada con una variable Integer llamada (Minute), una caja de texto estática con el texto (:) y una caja de texto variable llamada SecondsTimer enlazada con una variable Integer llamada (Seconds).

Para programar el temporizador primero se ha generado el evento StartTimer. Este evento más tarde será el evento que disparará el trigger box (caja de activación explicada anteriormente) que situaremos en la puerta de la casa objetivo para que inicie el temporizador en ese punto. Este evento primero muestra en pantalla el timer con un nodo play animation asociado a la animación appear (animación de aparecer en pantalla). Después la secuencia sigue con la resta de uno a la variable de segundos (antes comprueba que no está en 0). Cuando los segundos son 0, se inicia la resta de uno a la variable minutos y se ajustan los segundos a 59. Esto se repite en bucle hasta que los segundos y los minutos son 0. La programación del Blueprint queda expuesta en la Figura 3-44.

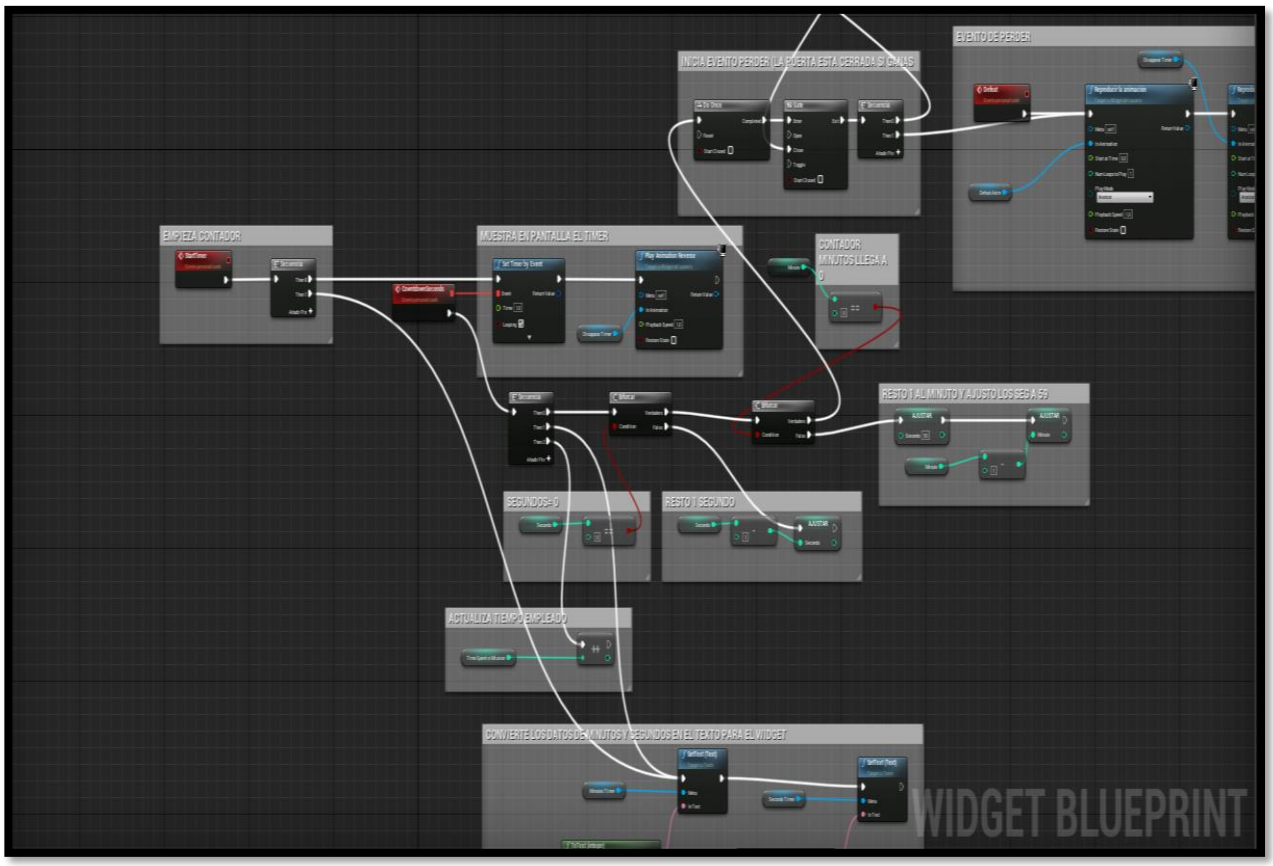


Figura 3-44 Blueprint del temporizador

Que el contador llegue a 0 supone que el jugador no ha eliminado antes de tiempo a los enemigos, por lo que se iniciará el evento derrota (defeat) que dará pie a la animación que hace desaparecer el timer y mostrará en pantalla el texto “Misión Fallida”.

Para programar el evento victoria (victory), se debe de entender éste como algo independiente del temporizador pero que compartirá con el evento perder (este sí que está relacionado con el temporizador) las funciones de acabar la simulación, reiniciarla y guardar los datos para el registro. Por eso desde el mismo Blueprint del Timer podemos programar el evento de victoria para acabar enlazándolo con esas funciones comunes a los dos eventos.

Se inicia el evento victoria cuando se haga la comprobación efectiva de enemigos restantes desde el BP Killshot dentro del BP que gestiona la salud y el daño de la IA (a su vez dentro del Enemy_character) desarrollado en el Anexo II: Bloques de código: Inteligencia Artificial. El evento victoria cierra mediante una puerta (gate) la opción de que se produzca el evento derrota e inicia una secuencia que primero elimina el timer de la pantalla, después muestra en pantalla el texto “Misión Cumplida BZ” en verde y por último marca como activa una variable booleana Victory Bool que más tarde usaremos en el registro. La programación de los Blueprints queda expuesta en la Figura 3-45.

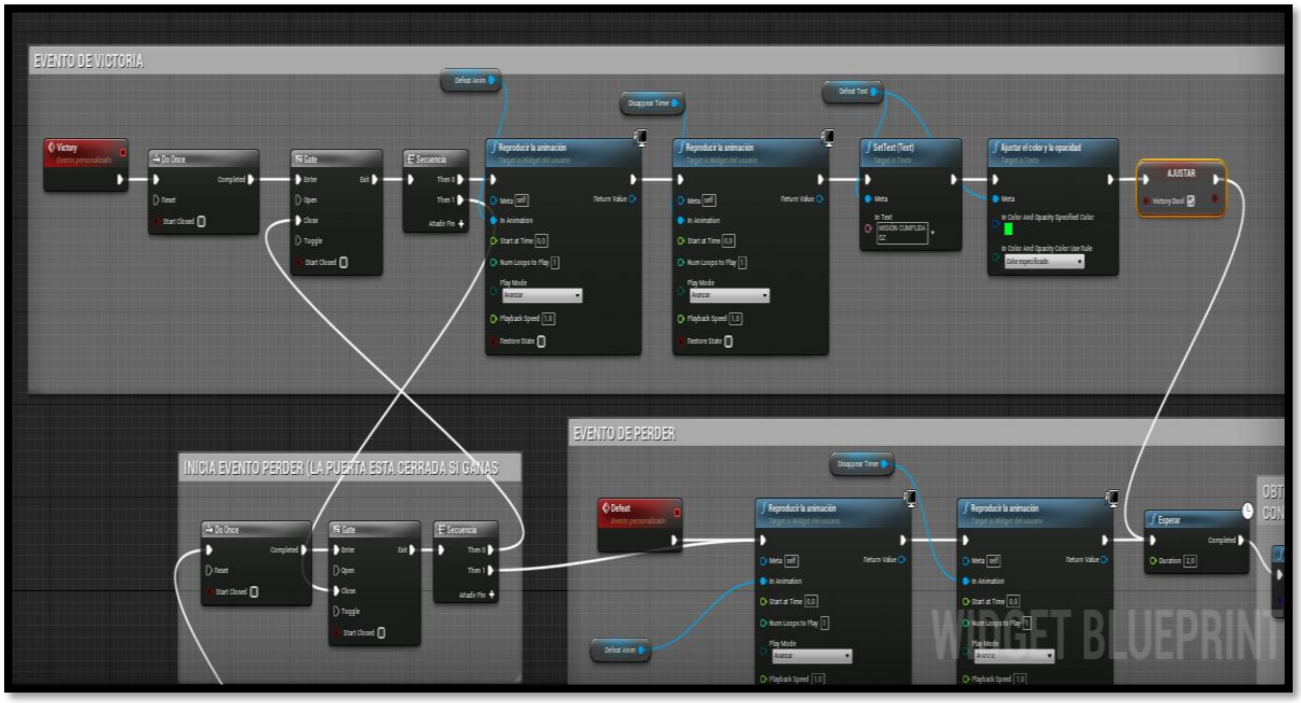


Figura 3-45 Blueprints de los eventos de victoria y derrota

El final de los nodos de los eventos de victoria y derrota acaban en una secuencia que primero guarda los datos de la partida, después inicia un fundido a negro para finalizar la simulación y por último reinicia la simulación en el punto inicial.

3.3.4.3 Análisis de datos

Una de las ventajas de las simulaciones es que todos los parámetros pueden ser registrados o cuantificados. El carácter instructivo del simulador vuelve muy interesante esta característica. Esto podemos aprovecharlo para tener una información detallada e individualizada de cada alumno.

La intención es que tras un intento de asalto se genere un documento que almacene a modo de registro toda la información que interesa de cada jugador.

Todos los cálculos necesarios para obtener cada dato, así como la programación para generar el documento de texto están incluidos dentro del BP del Timer. Primero veremos cómo se genera cada dato para finalmente ver cómo se genera el documento con todos estos concatenados.

- El primer dato que veremos en pantalla será el número de intento. Esto sirve para ordenar desde el primer alumno que comenzó intentándolo hasta el último. Para generar este dato se genera una variable Integer llamada attemp (intento) de valor inicial 0 que unida a un nodo de operación suma aumentará en 1 su valor cada vez que se inicia el proceso de guardado de información.
- El siguiente dato, el nombre del alumno, es una variable string que generaremos al introducir el nombre del alumno con el teclado al principio de la simulación (explicado en el siguiente apartado).
- Para acceder al dato de objetivo conseguido se ha recurrido a la variable booleana Victory Bool que marcábamos como activa cuando se daba el evento victory. Recordemos que ésta está desactivada por defecto.
- El tiempo empleado es una variable integer compuesto por dos variables integer guardadas al acabar cada partida y generadas al final de la secuencia que inicia el StartTimer.
- Por último, para el cálculo de la precisión se ha recurrido a dos variables integer previamente registradas (ver Figura 3-46): Total damage Amount Recived (es el número de impactos de bala registradas por el enemigo) y Bullet Shooted (es el número de balas que dispara el jugador). Dividiendo la primera variable entre la segunda con la operación Division y después multiplicándolo por 100 con la operación Multiplicación obtenemos una variable float que será el % de precisión.

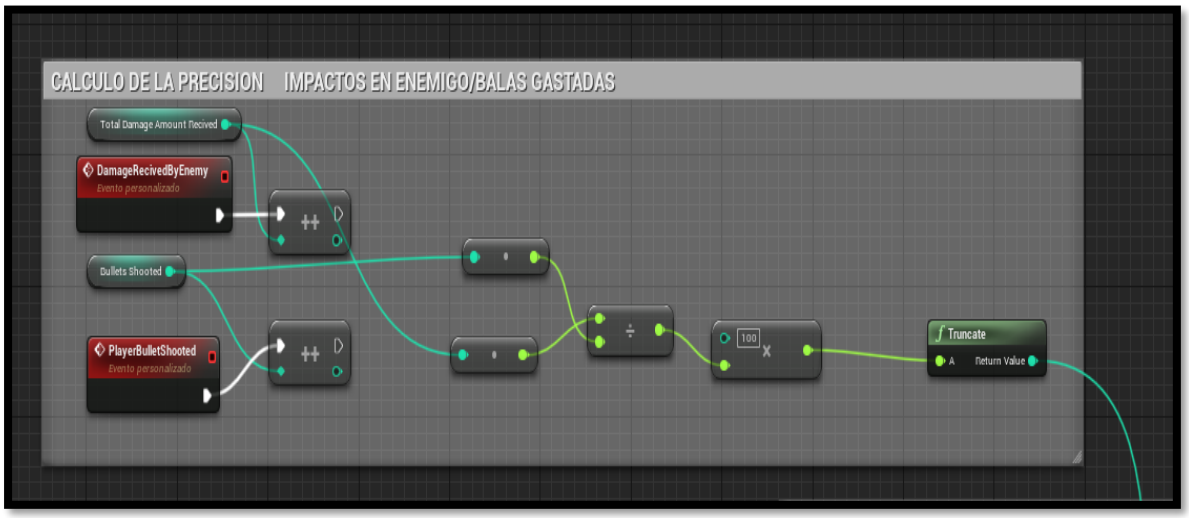


Figura 3-46 Blueprint de cálculo de precisión

Para generar el documento .txt con toda la información primero se ha generado un array que agrupa en una matriz las variables previamente explicadas. Esto se hace siempre al acabar cada partida (ya sea ganando o perdiendo) y se genera con un get array. Este array se configura para que guarde los datos en orden con un nodo Saves Structure. El array con toda la información es nombrado como “Saves”.

Al final de cada partida se vuelve a actualizar todos los datos del array y se inicia el evento SavesLog. Este evento inicia el proceso en cada partida que acaba y transforma los datos almacenados en el array “Saves” en un solo string (cadena de texto). Todos los datos han sido concatenados con la función Append. Cuando todos los datos ya están concatenados en forma de texto, se asocia a una nueva variable string llamada Log Data. Esta variable finalmente se guarda en una carpeta mediante el nodo Save String to File. Los pines de salida de este nodo sirven para concatenar el formato (.txt) y el nombre de la carpeta generada (/playerLogs). La programación del Blueprint queda expuesta en la Figura 3-47.

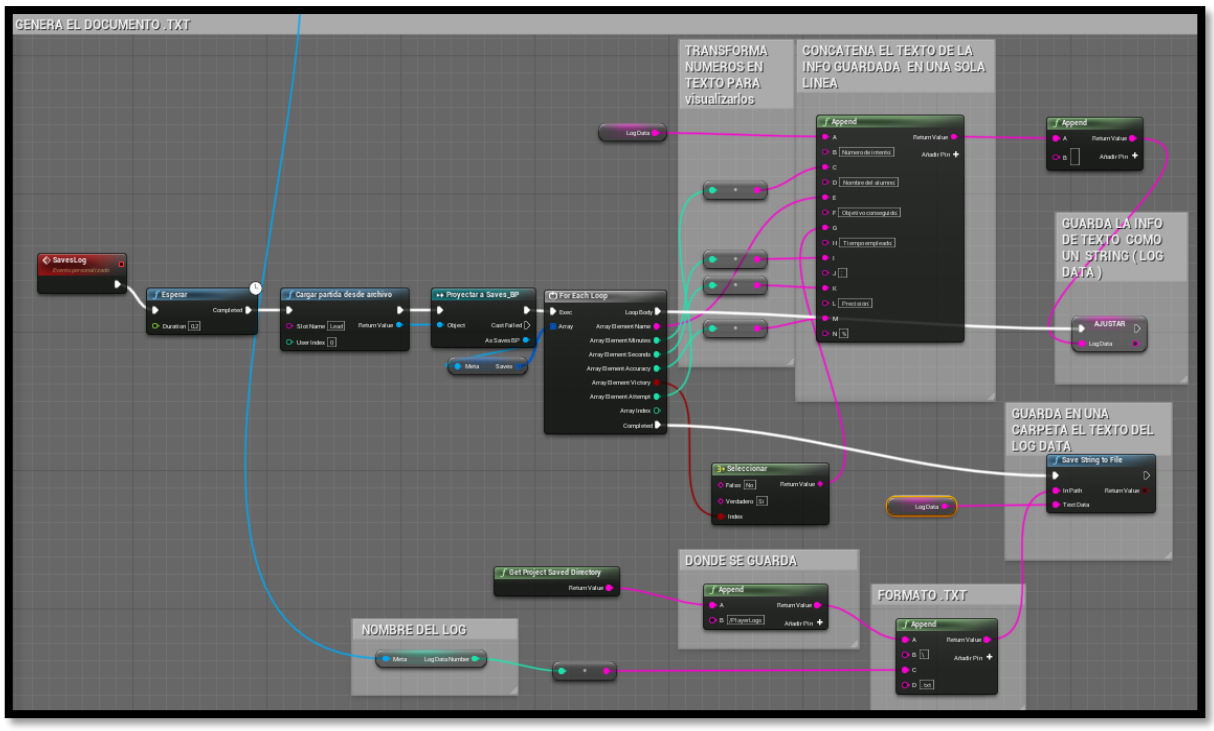


Figura 3-47 Blueprint que genera el archivo de texto con la información almacenada

3.3.4.1 Leaderboard o tabla de clasificación

El término Leaderboard, leaderbox o “tabla de clasificación" se utiliza a menudo en la industria de los videojuegos para mostrar los resultados de cada intento. Los jugadores pueden compararse con otros en función de su número de muertes (las más comunes), de los objetos recogidos o de alguna otra métrica. En general, las tablas de clasificación pueden ser un incentivo para que los jugadores mejoren, ya que dan a muchos una sensación de superioridad o logro. En el presente trabajo de fin de grado se ha generado también nuestra particular tabla de clasificación de tal manera que el alumno antes de empezar pueda observar las estadísticas de sus predecesores y así incentivar su propósito de mejora.

El UMG de Unreal permite generar estas leaderbox directamente desde el diseñador de un Blueprint Widget.

Se ha generado un nuevo Blueprint Widget llamado MainMenuEndLeaderboard dentro de una carpeta llamada Mainmenu a su vez dentro de la carpeta de Widgets.

Dentro del Diseñador del widget (ver Figura 3-48) se han añadido los siguientes elementos: en primer lugar, un panel de lienzo con una leaderboardcanvas (que contendrá la información que queremos mostrar) y un panel de lienzo (que servirá para introducir el nombre del jugador). El leaderboardcanvas está compuesto por una imagen translúcida de color verde para dar fondo y consistencia a la tabla y una caja horizontal con las cajas de texto de las columnas que identifican los datos: Nombre del alumno, Tiempo empleado, Precisión, Objetivo Conseguido. El panel de lienzo para introducir el nombre está compuesto en primer lugar también por una imagen translúcida de color verde y una caja horizontal con un Editabletext que configuramos con un texto indicativo (pestaña content) para que el jugador sepa que ahí es donde debe de introducir su nombre con el teclado.

La secuencia de acción del leaderboard ha sido programada directamente desde el Gráfico del BP (ver Figura 3-49). Esta secuencia inicia con la aparición en pantalla de la leaderboard con los datos que tenga almacenados. Posterior a esto, se programa para que el jugador no sea capaz de ejecutar la simulación hasta que no tenga introducido un nombre. El modo “bloqueado” en el que sólo se puede

introducir el nombre se configura con los nodos Set Input Media UI only y Set User Focus asociados a la caja de texto editable que antes hemos generado para escribir el nombre. Hasta que no se detecte un nombre introducido con tecla Enter no se cerrará el widget y no se pasará a la situación de game ready (preparado para jugar).

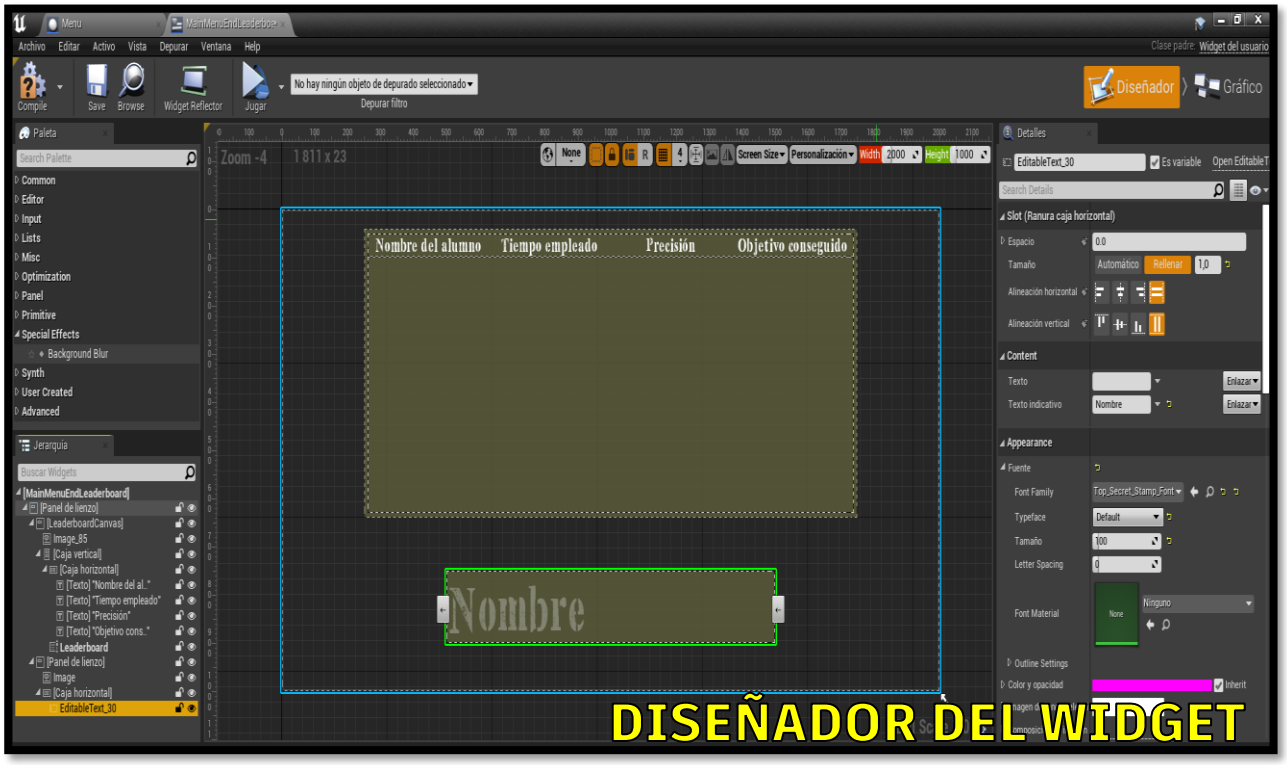


Figura 3-48 Diseñador que controla el aspecto visual de la leaderboard

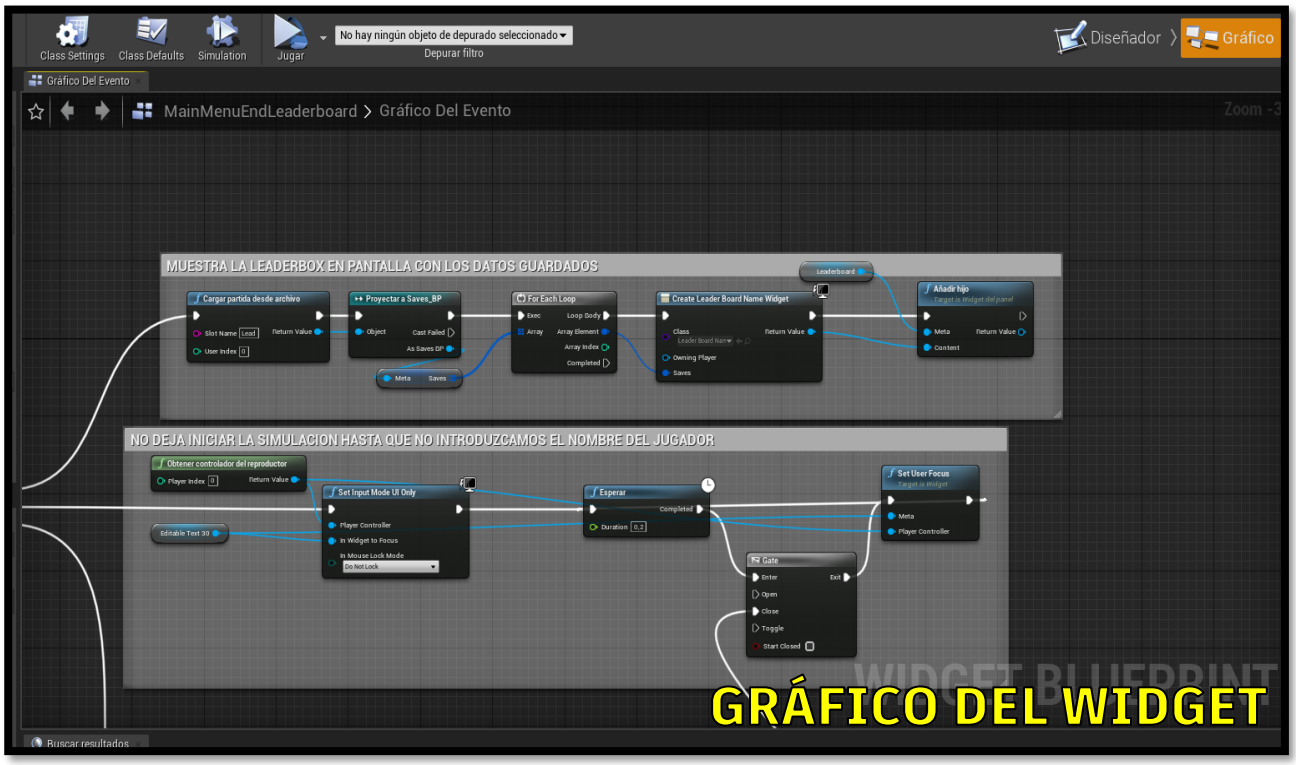


Figura 3-49 Gráfico que controla el comportamiento de la leaderboard

Tal y como se hizo antes, para colocar el widget en la escena primero ha sido necesario asociarlo a una Blueprint Class de padre actor (ver Figura 3-50). En este caso se ha nombrado como Menu. En la pestaña de user interface en el Widget class ha sido asociado al widget MainMenuEndLeaderboard. Finalmente, se ha colocado sobre el nivel para que sea lo primero que vea el usuario al iniciar.

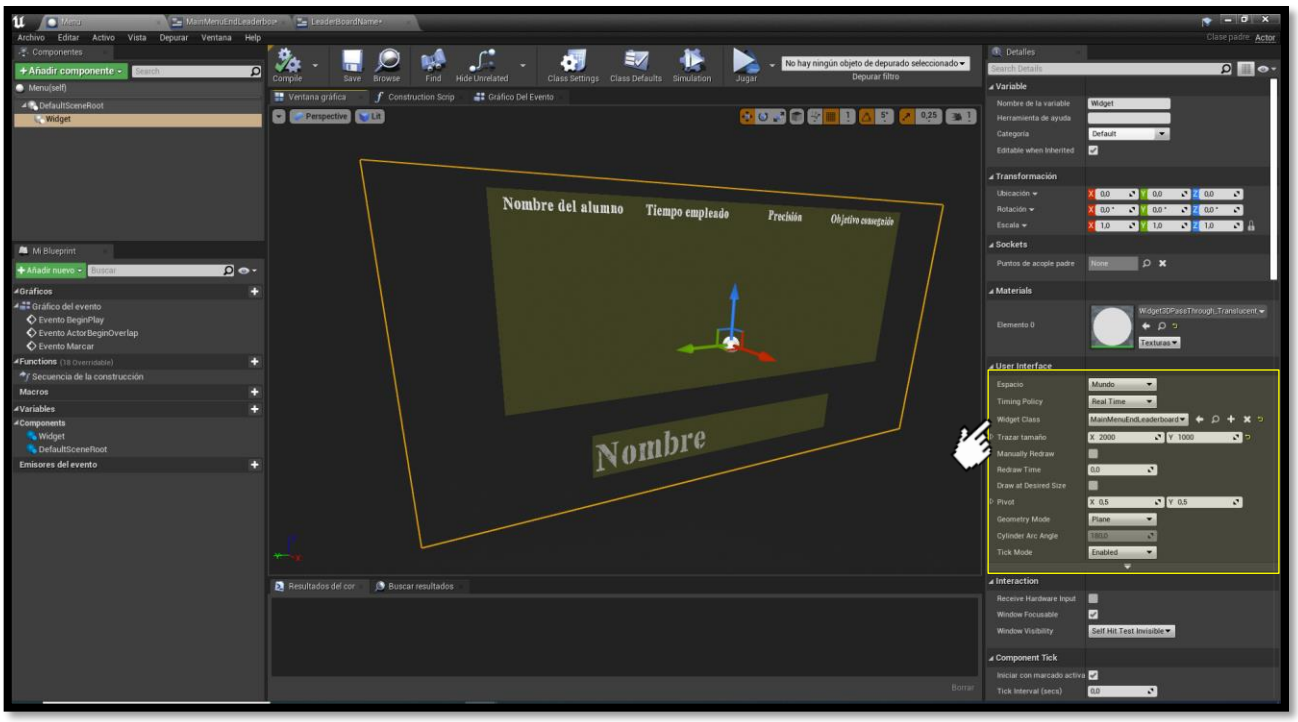


Figura 3-50 Asociando la leaderboard a un actor para situarlo sobre el nivel

Posteriormente se ha generado un segundo Blueprint Widget que servirá para ordenar la forma en la que se presentarán los datos sobre el Widget anteriormente generado. Este widget de nombre LeaderBoardName contiene un panel de lienzo con una sola caja horizontal que a su vez está compuesta por una serie de cajas de texto modificables y enlazadas con las variables almacenadas. La construcción del gráfico (ver Figura 3-51) que maneja qué datos muestra en pantalla el widget se basa en el uso de nodos SetText(Text) que transforma las variables enteras de número de intento, nombre, minutos, segundos, precisión y la variable booleana de victoria o derrota en strings (cadenas de texto). Todas estas variables son las mismas que las anteriormente explicadas en el subapartado 3.3.4.3 y que sirvieron para guardar el registro con todo el análisis de datos.

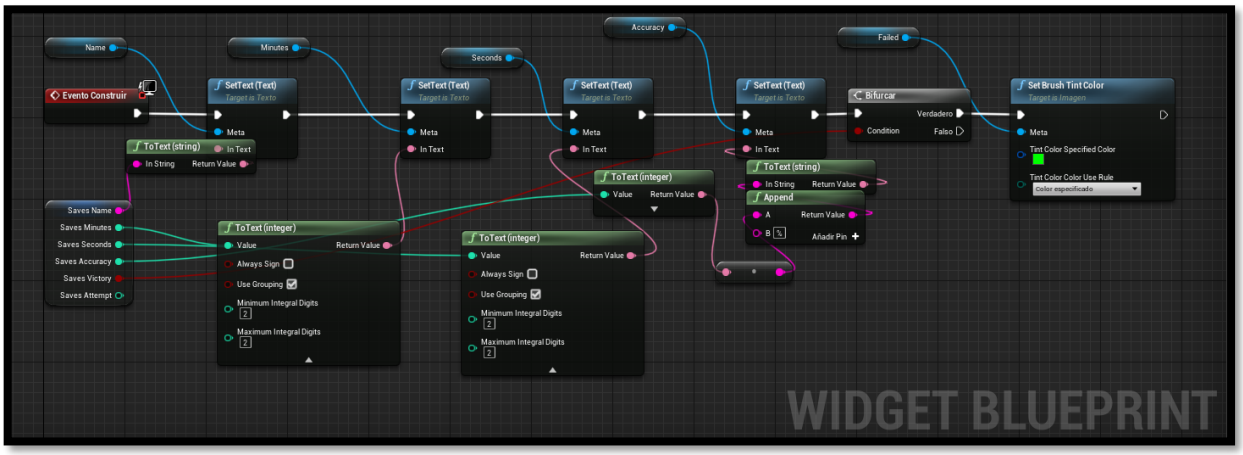


Figura 3-51 Blueprint que extrae y ordena los datos de la leaderboard

3.3.4.2 Nivel con modo visión nocturna

Las misiones nocturnas son también de interés en las operaciones en zonas urbanas y su entrenamiento depende mucho del sigilo y la capacidad de visión que aportan los medios de visión nocturna actuales.

Se ha planteado que además del nivel diurno ya programado la simulación también ofrezca la posibilidad del mismo nivel, pero en modo noche y con la opción de utilizar un visor nocturno.

Para conseguir adaptar el nivel a un nivel nocturno se ha procedido tal y como ya se explicó en el subapartado 3.2.3. Una vez duplicado y variado el nivel para que aparezca en modo noche solo queda generar la opción de activar el modo visión nocturna.

Para conseguir el modo visión nocturna se ha creado un nuevo material de nombre NightVision (ver Figura 3-52) y almacenado en una carpeta del mismo nombre. Al abrir el editor de material lo primero que se ha hecho es cambiar el material domain (dominio del material) a post process y añadir un nodo Scene Texture Post Process Input 0. Con esto estamos modificando el aspecto de visión que procesamos constantemente. Lo siguiente ha sido unir los nodos del SceneTexture con el nodo Emissive Color con una configuración intermedia de otros nodos que añaden la tonalidad verde buscada (mediante un color de base verde) y un nodo Radial Gradient Exponential (con los pines de entrada radio y densidad configurados en 0,6 y 3 respectivamente). Después de configurar el material se ha accedido a las propiedades la cámara del motion controller pawn desde la pestaña Post Process, en Rendering Features y ahí se ha añadido el material creado como un Post Process Material más (por defecto no viene ninguno). Lo último que habría que añadir a la cámara como un componente inherente a la misma es una luz de foco para que simule la proyección de la visión nocturna.

Por último, se ha programado desde el Blueprint que controla el motioncontroller pawn la activación y la desactivación del modo visión nocturna a través de una función Flip Flop enlazada al botón de menú del controlador a su vez unido al nodo Set Post Process Weight.

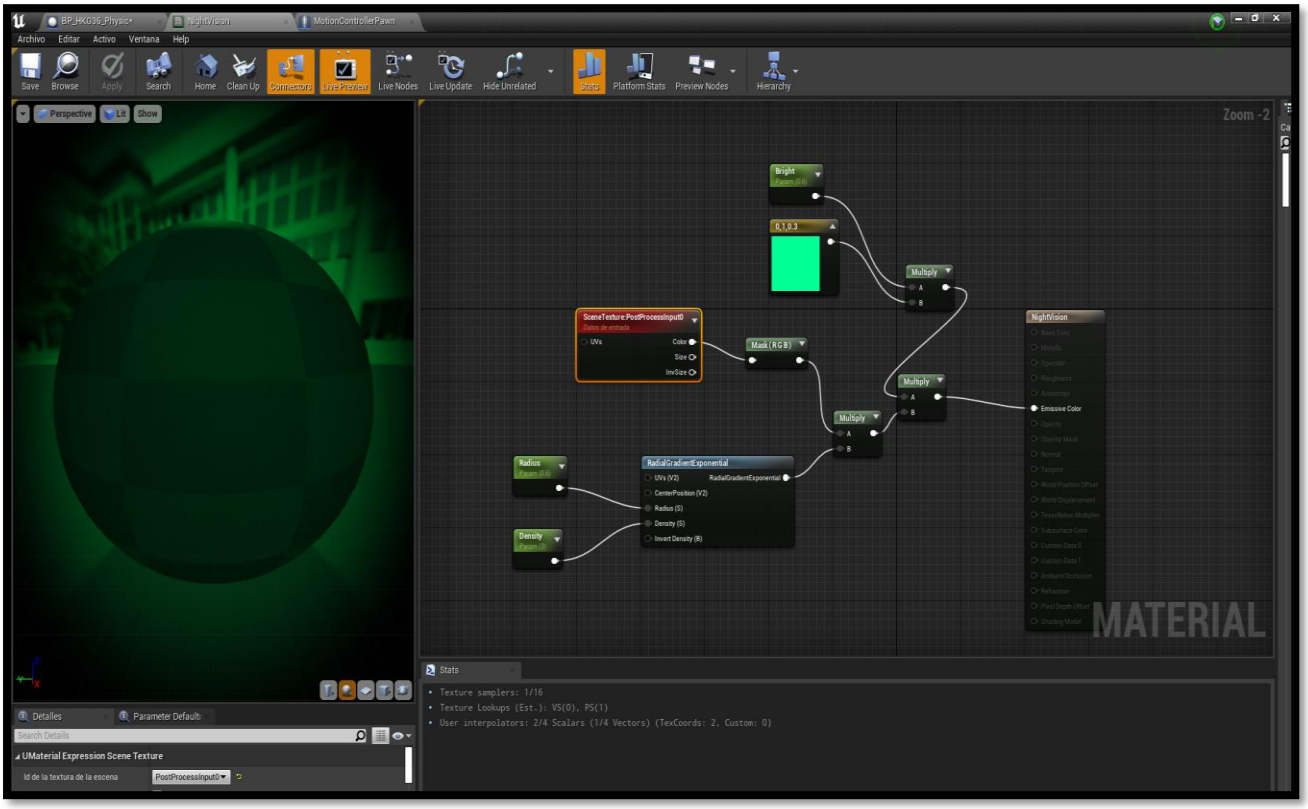


Figura 3-52 Material generado para la visión nocturna

3.4 Realidad Virtual

Uno de los principales errores que cometen muchos desarrolladores al acercarse por primera vez a la RV es que intentan aplicar al espacio de la RV los diseños tradicionales que están acostumbrados a crear en el espacio 2D y, en la mayoría de los casos, esto no funciona. La RV es su propio medio y no sigue las mismas reglas que los medios que le precedieron. El casco de RV es (dependiendo de si incluye o no audio integrado) principalmente un dispositivo de visualización, pero la experiencia que crea para el usuario es muy diferente a la que crea una pantalla plana tradicional. La diferencia está principalmente en la inmersión. Los medios no inmersivos, por muy grandes o detallados que sean, siguen dejando al espectador rodeado de recordatorios de que la escena no es real. Los medios inmersivos, en cambio, parecen rodear al usuario por completo. [59]

Esta mezcla de inmersión con percepción de la profundidad provoca la respuesta natural a tus movimientos que se unen para convencer a tu cuerpo de que lo que estás percibiendo es real. A este fenómeno lo llamamos sensación de presencia o presence. La presencia en la RV se refiere a la sensación del usuario de que está físicamente en el mundo virtual, respondiendo al entorno como si estuviera realmente allí y experimentara estas cosas. [59]

En nuestro caso, necesitamos que la sensación de presencia sea lo más lograda posible, por lo que la transición y la integración de la Realidad Virtual en nuestro proyecto debe de ser cuidadosamente tratada.

3.4.1 Preparación del headset

Como se ha visto en el apartado del headset utilizado será el propio del HTC VIVE 2.0 con sus distintos periféricos. Partiendo de la base de que ya hemos instalado el software de control, así como los drivers necesarios para que el PC pueda utilizar el equipo es importante saber que para poder probar los cascos de RV en Unreal Engine es necesario primero que estos estén encendidos. De ninguna manera el proyecto detectará los cascos si estos no han sido encendidos de manera previa al lanzamiento del programa. Además, para que UE4 sea capaz de usar nuestro headset (u otro cualquiera) debemos de tener instalado el plugin Steam VR. Recordemos que esta es la plataforma que enlaza el headset con el equipo. Con el plugin lo que hacemos es enlazar Steam VR con UE4.

El lanzamiento de SteamVR es automático al arrancar el ordenador, por lo que analizará automáticamente la presencia de las conexiones necesarias del equipo de RV cada vez que iniciamos sesión. Una buena manera de comprobar que todos los elementos del headset están bien conectados y listos para usarse es en la ventana de configuración de Steam VR dónde con una simple vista (ver Figura 3-53) podemos saber el estado en que se encuentran cada uno de los periféricos.

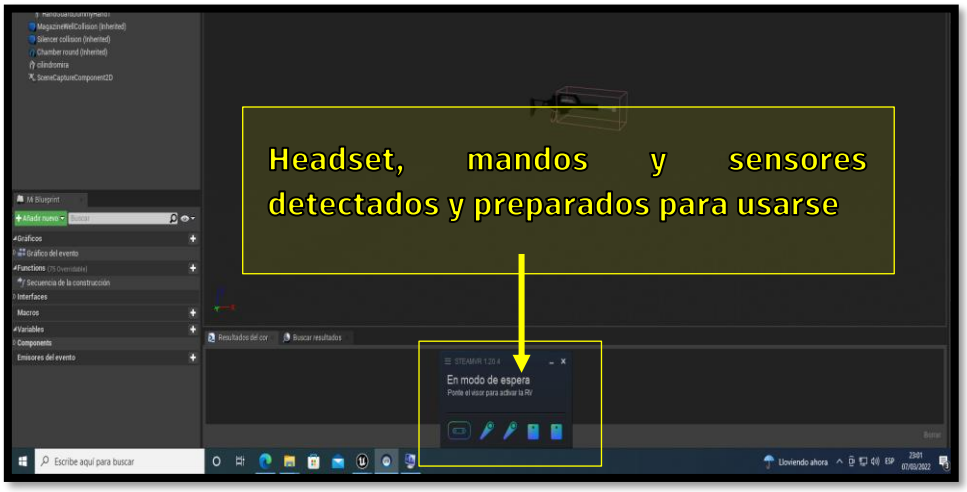


Figura 3-53 Comprobación de equipo de RV listo para usar desde Steam VR

3.4.2 Simulaciones en RV desde Unreal Engine

Una vez encendido el equipo de RV y lanzado el proyecto de UE4, para su prueba sólo hay que irse al apartado Jugar de la toolbar. Este apartado incluye diferentes formas de lanzamiento del proyecto para probar su funcionamiento. En concreto, si el programa detecta que tenemos el equipo conectado en el desplegable incluirá la opción VR Preview. Al seleccionar esta opción entraremos en la simulación previamente programada.

La opción seleccionada se guardará automáticamente como opción por defecto por lo que las siguientes veces que queramos probar el funcionamiento del nivel sólo hay que hacer click en Jugar (ver Figura 3-54).

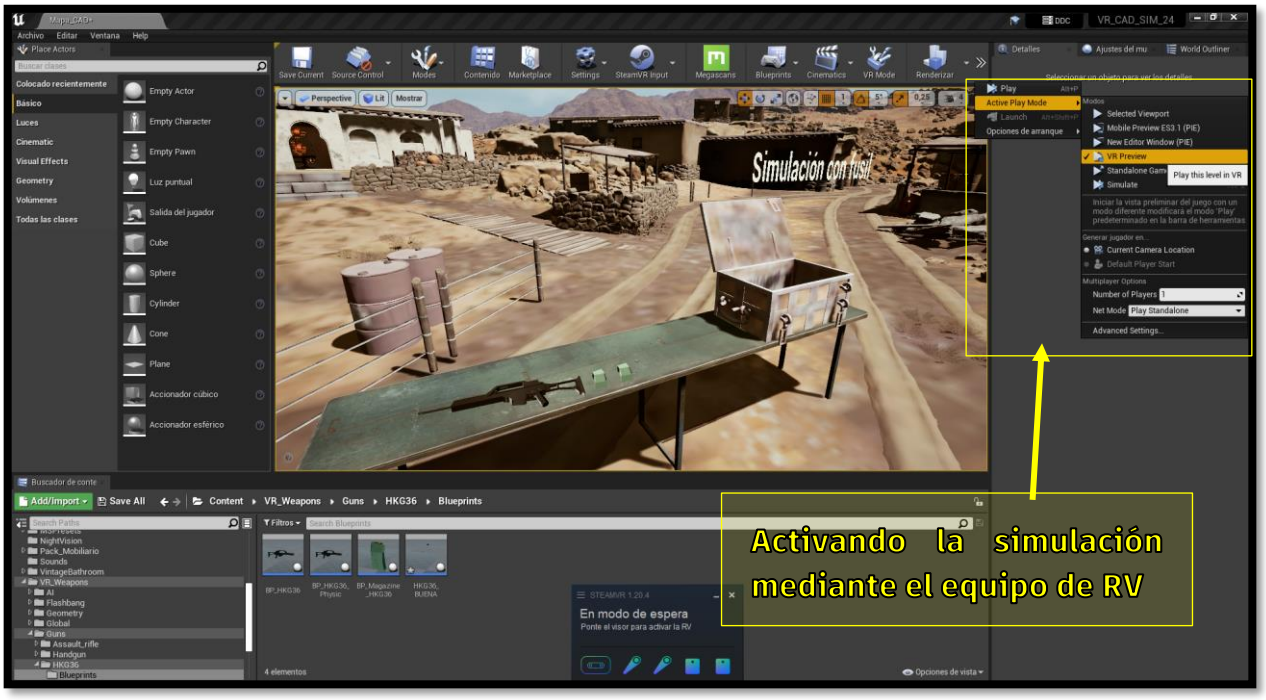


Figura 3-54 Activación de la simulación mediante el equipo de RV

3.4.3 Inclusión del modo con adaptador Beswin VR M4

Para aumentar la sensación de presencia en el entono generado, podemos utilizar un elemento más que aporte un mayor grado de realismo. El laboratorio de diseño del Centro Universitario de Defensa cuenta con un fusil dispositivo diseñado específicamente para utilizar como soporte de los controladores HTC VIVE. Este dispositivo cuando tiene los controladores integrados funciona como réplica de un arma real por lo que resulta muy interesante como elemento a incluir en el proyecto.

Sin embargo, su uso requiere de una planificación previa ya que el concepto de controladores con movimiento totalmente liberado queda sustituido por el uso de controladores con movimiento restringido. Este movimiento restringido es debido a que los controladores sobre el fusil real siempre se encuentran en la misma posición relativa el uno respecto al otro. Por eso si se trata de iniciar la simulación con los controladores integrados en el fusil real encontramos muchas complicaciones de manejo. Se pierden entonces funcionalidades como la recogida de cargador o el proceso de carga del arma. Además, presenta la problemática de que existe una descoordinación entre el apuntado virtual y el que hacemos con el fusil real, debido a una diferencia de altura constante entre ambos controladores.

Para solventar estos problemas se ha planteado lo siguiente:

El jugador podrá elegir el modo de juego en el que se encuentra; existe el modo normal con controladores y el modo de simulación con fusil. El primero tendrá activas todas las funciones previamente programadas de agarre, carga y disparo con cargadores extraíbles y el segundo modo estará destinado al uso del dispositivo y limitará la simulación al agarre del fusil con las dos manos de manera constante. Además, el segundo modo recargará automáticamente las balas del cargador una vez este se agote ya que ya no disponemos del movimiento de manos liberadas que teníamos antes.

El cambio de un modo a otro se ha integrado dentro de un trigger box que cambia de modo cuando lo tocamos con las manos. Este trigger box cuenta con un texto indicativo del modo en que se encuentra el nivel.

Por defecto empezará el nivel en modo con controladores. Si el jugador decide usar el fusil tendrá que introducir las manos (sin tener un fusil agarrado previamente) en el trigger box. Una vez pase al modo con fusil automáticamente un fusil con cargador ya incluido será spawneado (aparición repentina de un elemento en el nivel) y agarrado con ambas manos.

Debido al problema de descoordinación entre la orientación del dispositivo y el fusil virtual debemos de generar un BP idéntico del fusil programado, pero al que debemos de incluirle una serie de cambios para mejorarlo. Por eso, se ha generado un segundo fusil como Childblueprint Class directamente desde el BP_HKG36. El nuevo fusil, que será el spawneado automáticamente en el modo con fusil real, se ha llamado BP_HKG36_Physics.

Dentro de este Blueprint se ha configurado el arma para que spawnee un nuevo cargador cada vez que el contador de balas llegue a 0 mediante la función SpawnActor BP Magazine HKG36. El spawn del cargador se configura para que se haga justo en la posición relativa del socket respecto al fusil y así forzar que siempre se introduzca automáticamente. Además, en la secuencia que sigue el BP de este nuevo modo incluimos el evento Force Grab que lleva directamente a la secuencia de agarre del fusil por defecto. Así, cada vez que introducimos las manos en el trigger box, conseguimos que spawnee el fusil ya agarrado y con el cargador ya introducido. La programación del Blueprint queda expuesta en la Figura 3-55.

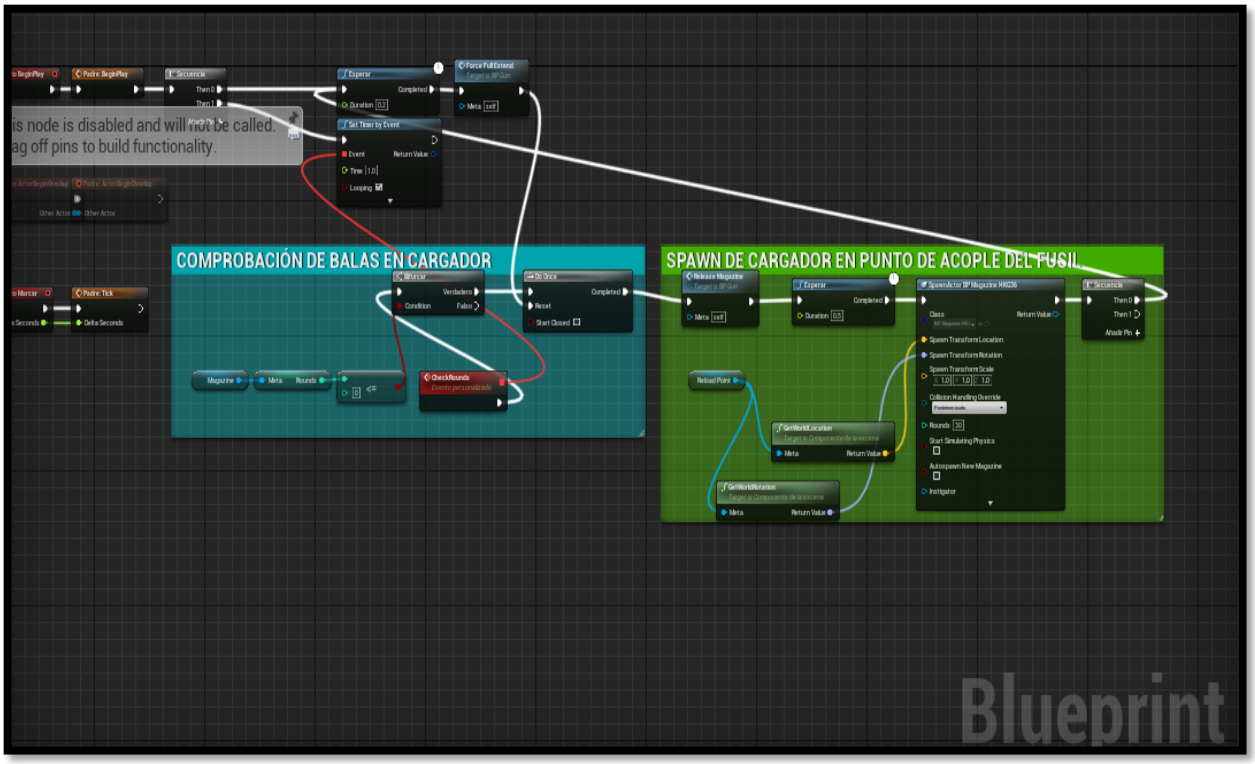


Figura 3-55 Blueprint que controla la recarga automática del modo con fusil real

Para resolver el problema que presenta el dispositivo a la hora de apuntar con los controladores acoplados al dispositivo se ha realizado una compensación que modifica la variable Height offset, variable de tipo float, que se encuentra disponible en la class default del Blueprint que controla el arma.

Con esta nueva variable ya cambiada a un valor de -12 ha quedado compensada la diferencia de altura entre controladores. Tras las pruebas realizadas, la experiencia y la sensación de inmersión ha quedado aumentada notablemente, ya que el dispositivo solventa la sensación de inestabilidad o temblor que pueden generar los controladores debido a su alto grado de sensibilidad. Con este modo, el jugador puede hacer uso de la mira telescópica y poner en práctica los movimientos de forma mucho más realista. La Figura 3-56 muestra de manera visual el problema planteado y la solución adoptada para solventarlo.

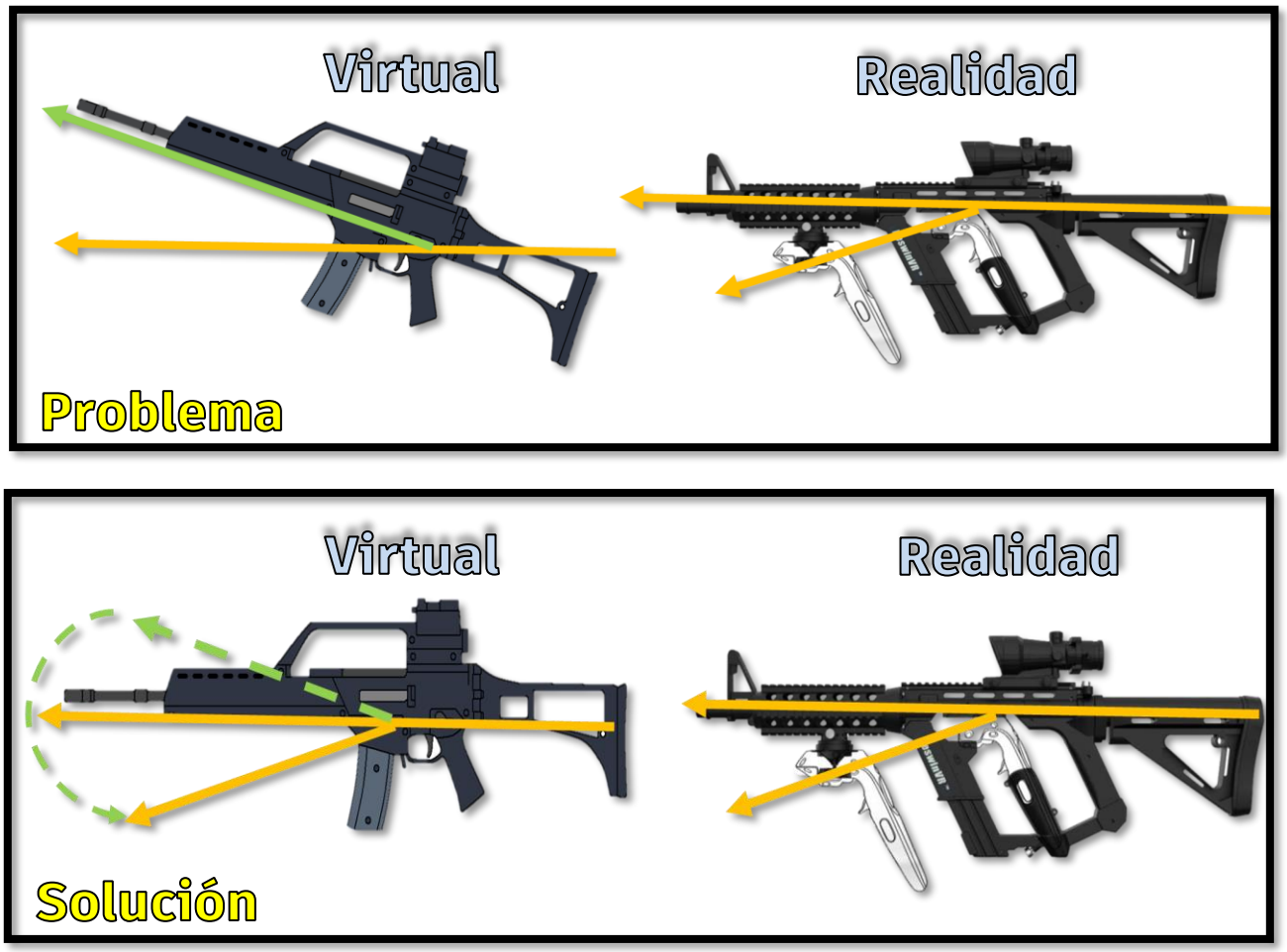


Figura 3-56 Compensación del arma virtual

4 RESULTADOS

4.1 Descripción del apartado

En el presente apartado se mostrarán los resultados obtenidos tras el desarrollo. Se listarán uno a uno cada uno de los elementos que han sido diseñados y programados de tal manera que tras la lectura del apartado el lector tenga claro qué objetivos se han alcanzado satisfactoriamente y cuál ha sido el desenlace del proyecto.

Al tratarse de un proyecto marcado por lo visual, la mayoría de los resultados serán mostrados mediante imágenes de autoría propia.

4.2 Escenario bélico generado

El terreno generado sobre el que se desplaza el jugador durante la simulación ha resultado ser fielmente recreado. La topografía y la extensión disponible han acabado por ofrecer un escenario tal y como se planteó al principio del proyecto y cuenta con suficiente espacio navegable como para continuar incluyendo niveles de simulación (ver Figura 4-1).

Las diferentes pruebas a las que ha sido sometida la simulación corroboran la sensación de inmersión que genera la ambientación escogida (ver Figura 4-2).

Figura 4-1 Parte del escenario generado

Figura 4-2 Poblado generado para aumentar la sensación de inmersión

Además del diseño del escenario ficticio se ha generado una casa objetivo dónde se centra el desarrollo del nivel y que permite el entrenamiento táctico de los alumnos (ver Figura 4-3). La casa ha sido diseñada para que el usuario pueda adiestrarse en diferentes tipos de situaciones que pueden darse en la vida real e incluye elementos de alta resolución para aumentar la sensación de realismo (ver Figura 4-4). Así mismo, el funcionamiento de la ambientación sonora funciona correctamente ya que este se activa o se desactiva en función de si el jugador se encuentra dentro de la casa objetivo o no.

Figura 4-3 Resultado de la casa objetivo

Figura 4-4 Interior de la casa objetivo

El arma a utilizar durante la simulación escogida ha sido la propia del Ejército. El fusil de asalto HKG36 ha sido diseñado y modelado de tal forma que se asemeja en gran medida al real. Mediante técnicas de texturización se ha podido aumentar aún más el grado de realismo. Adicionalmente, se ha diseñado la mira telescópica de 3 aumentos y se ha integrado en el cuerpo del fusil.

Figura 4-5 Fusil HKG36 Modelado

4.2.1 Personajes

El conjunto de avatares y personajes no controlables generados durante la simulación son los siguientes: el avatar principal, el NPC enemigo tipo 1, el NPC enemigo tipo 2 y el grupo de rehenes formado por 3 personajes.

4.2.1.1 Avatar principal

Personaje controlado por el usuario cuyo aspecto es por defecto el que ofrece Unreal Engine (ver Figura 4-6). Cuenta con un sistema de salud regenerable y es considerado como muerto tras recibir 4 impactos de bala seguidos. Porta en el pecho hasta 4 cargadores que sirven para simular el chaleco de combate. Tiene la habilidad de interactuar con todos los objetos físicos del mundo virtual gracias a los controladores. Además, posee la capacidad de moverse tanto de pie como agachado.

Figura 4-6 Avatar principal controlable por el jugador

4.2.1.2 Enemigos

Son los personajes no controlables por el usuario. Cuentan con la Inteligencia Artificial programada y su finalidad es representar la amenaza enemiga. Los enemigos tienen la habilidad de patrullar dentro de la casa objetivo y de disparar tras detectar (bien mediante el sentido de la vista o bien mediante el sentido del oído) la presencia del avatar principal. Su sistema de salud sirve para abatirlos tras dos disparos efectivos. Existen dos tipos de enemigos cuya única diferencia reside en el aspecto visual (ver Figura 4-7).

Figura 4-7 Enemigos

4.2.1.3 Rehenes

Son también personajes no controlables por el usuario. Su finalidad es meramente narrativa ya que no disponen de Inteligencia Artificial. Se encuentran al final de la casa objetivo, y cuentan con el mismo sistema de salud que los enemigos. El usuario debe de tratar de evitar que reciban daños derivados del enfrentamiento. En total, existen 3 rehenes claramente diferenciados visualmente: un hombre y una mujer de edad adulta y un niño (ver Figura 4-8).

Figura 4-8 Rehenes

4.3 Resultados de la programación

Tras configurar y programar los comportamientos de todos los elementos virtuales, se han generado dos niveles, uno diurno y otro nocturno. Ambos son exactamente iguales estando el segundo destinado al adiestramiento de asaltos con medios de visión nocturna.

Los niveles comienzan con la aparición del usuario sobre el terreno. El instructor o el mismo usuario deben introducir su nombre en la tabla de clasificación que se encuentra sobre el nivel. En esta tabla (ver Figura 4-9) puede observarse en análisis de los datos extraídos de los jugadores que le han precedido

Figura 4-9 Situación inicial de la simulación

Una vez introducido el nombre, comienza la simulación, donde el usuario deberá de agarrar o bien el fusil que dispone sobre la mesa con sus propias manos o bien el fusil que se encuentra dentro del baúl añadido para cambiar al modo simulación con fusil.

Independientemente del modo de simulación, el usuario se acerca a las proximidades de la casa objetivo. Al entrar dispone de 4 minutos en los que deberá de acabar con un total de 7 enemigos localizados en diferentes habitaciones del interior de la casa.

Los enemigos, cuentan con la capacidad de patrullar, escuchar y disparar cuando detectan la presencia del usuario. Cuentan con un sistema de vida mediante el cual son abatidos al recibir dos impactos del usuario siempre que estos sean certeros. Si el disparo se produce sobre la cabeza los enemigos son abatidos directamente.

El usuario cuenta con cuatro cargadores dispuestos en el pecho como si los portase en el chaleco de combate.

Si el temporizador llega a 0 se considerará misión fallida y se reinicia el nivel.

Si el usuario acaba con todos los enemigos antes de tiempo, los rehenes se considerarán liberados y la misión se da por cumplida. En este caso también se vuelve a reiniciar el nivel.

La Figura 4-10 muestra una situación de combate directo con uno de los enemigos situado en la sala principal de la casa objetivo. Como se puede observar el fusil al disparar cuenta con un efecto de fuego en la boca del cañón y el enemigo está respondiendo a nuestra presencia abriendo fuego sobre nosotros.

Figura 4-10 Enfrentamiento con enemigo

Aunque el uso de la mira telescópica es desaconsejable durante las operaciones de este tipo debido al poco tiempo de reacción del que se dispone, se puede comprobar como esta ha sido implementada con éxito. El aumento es el correcto y los disparos van dirigidos hacia donde apunta la cruceta (ver Figura 4-11).

Figura 4-11 Uso de la mira telescópica

Las secuencias que inician los eventos de victoria y derrota funcionan correctamente y determinan cómo de efectiva ha resultado la misión. Tal y como muestra la Figura 4-12 el Widget aparece sobre nuestro campo de visión en cuanto eliminamos a todos los enemigos presentes en la casa objetivo. Tras su aparición se reinicia correctamente el nivel dando pie a que un nuevo usuario pueda tratar de completar el ejercicio.

Figura 4-12 Evento de victoria

El nivel de noche cuenta con las mismas funciones que el nivel de día, pero con la opción de activar un dispositivo de visión nocturna que habilita al jugador completar la misión en condiciones de visión muy parejas a las que se encontraría en la realidad (ver Figura 4-13).

Figura 4-13 Modo visión nocturna

Cuando la simulación se da por finalizada y todos los usuarios han realizado el ejercicio se genera un documento de texto con los datos de número de intento, nombre del jugador y si ha completado la misión con éxito. Adicionalmente muestra el tiempo empleado por cada jugador y el % de precisión demostrado durante el ejercicio. La Figura 4-14 es una captura del documento de texto generado tras finalizar una simulación en la que han participado varios usuarios.

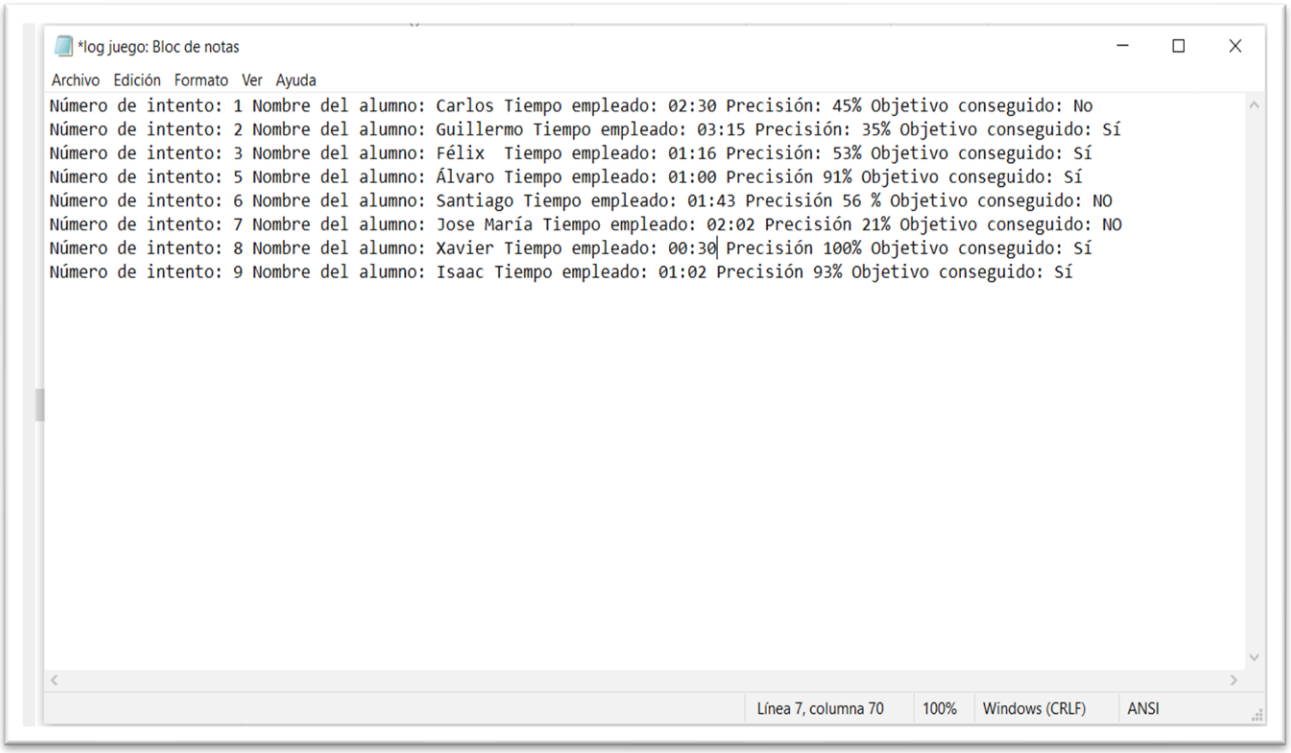


Figura 4-14 Registro generado con los datos de cada jugador

4.4 Simulación mediante el equipo de RV

Se ha logrado, además de la creación de un entorno virtual para el simulador de combate táctico y el diseño de los niveles de adiestramiento, la adaptación del equipo de Realidad Virtual para su uso como controlador.

En el simulador se puede por tanto emplear el dispositivo de Realidad Virtual para visualizar el escenario, controlar el movimiento del personaje que representa al usuario, realizar todas las acciones implementadas y, además, llevar a cabo el ejercicio planteado en el nivel.

Cabe reseñar que dada la limitación del cable de conexión de los cascos y la diferencia de dimensiones y disposición entre el área disponible en el laboratorio y el entorno virtual generado pueden experimentarse limitaciones de desplazamiento. Esta limitación es común a todas las experiencias en Realidad Virtual ya que casi en ningún caso es posible disponer de un espacio similar al que representamos. De hecho, la Realidad Virtual está concebido para esto mismo. Por eso, para un correcto uso del simulador, sería conveniente primero adiestrar al usuario en el uso combinado de los sistemas de movimiento programados.

La Figura 4-15 muestra como el usuario puede apuntar con los controladores acoplados al dispositivo Beswin VR M4. La sensación de inmersión así como la estabilidad que este dispositivo aporta se ha considerado todo un acierto de implementación.

Figura 4-15 Simulación con controladores acoplados al dispositivo

4.5 VR CAD Simulator: ejecutable definitivo

Finalmente se ha compilado todo el proyecto para tener un ejecutable que pueda lanzar el simulador directamente desde el escritorio de cualquier PC.

También se ha diseñado un logo de identidad para el propio simulador mediante la aplicación de navegador Canvas y se ha incluido como icono del proyecto (ver Figura 4-16).

Figura 4-16 Logo diseñado para el simulador

4.6 Vídeo demostración

De forma adicional, para presentar las características principales del simulador se ha producido un vídeo de demostración. El enlace al vídeo es el siguiente: https://youtu.be/cMoOZXlW4pA

La Figura 4-18 contiene un enlace QR con acceso directo al vídeo.

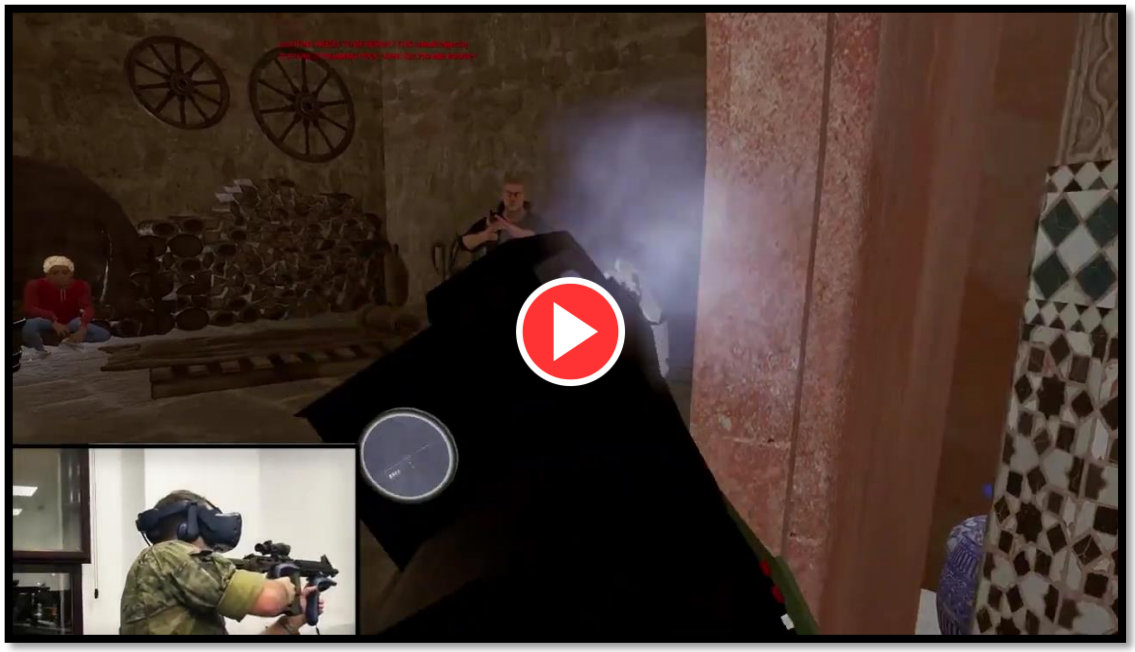


Figura 4-17 Video demostración

Figura 4-18 Código QR con enlace al vídeo demostración

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Descripción del apartado

El presente apartado versa sobre la interpretación definitiva del Trabajo de Fin de Grado y las conclusiones obtenidas. También se explicará si se han alcanzado o no los objetivos iniciales. Finalmente se listarán una serie de ideas a modo de propuesta para futuras líneas de investigación y desarrollo que servirían para ampliar el alcance y la capacidad del presente proyecto.

5.2 Conclusiones

5.2.1 Conclusiones previas

Se da paso al comentario de las diferentes fases del proyecto, desglosadas por orden: diseño y modelado del escenario bélico, programación y uso de la Realidad Virtual.

5.2.1.1 Diseño y modelado del escenario bélico

El escenario de combate para el adiestramiento táctico seleccionado y sus objetivos de cumplir con unos requisitos de inmersión y fotorrealismo mínimos han sido alcanzados con éxito. La utilización de modelos de alta resolución y las herramientas de edición del terreno han dado muy buenos resultados. Cabe reseñar el gran acierto de focalizar el diseño en una sola infraestructura (haciendo referencia a la construcción de la casa objetivo), ya que gracias a esto se ha podido dotar al escenario de una credibilidad y de un nivel de detallismo suficiente.

Por otro lado, la extensión inicial planteada para el escenario ha resultado quizás un tanto “excesiva” ya que se han quedado partes del escenario excluidas totalmente de la experiencia que el nivel plantea. Sin embargo, esta sobreextensión puede suponer un buen material de inicio para continuar con el desarrollo del presente proyecto.

En términos generales, el diseño y el modelado del escenario ha cumplido satisfactoriamente con las necesidades que le dieron inicio y los objetivos que se plantearon en un principio.

5.2.1.2 Programación

En primer lugar y como conclusión primera, se debe de mencionar la utilidad y la facilidad planteada por la programación multi nodal del motor gráfico escogido. La programación mediante el visual scripting de Blueprints de Unreal Engine supone una gran ayuda frente a la programación en C++ sin perder por ello cierto componente de complejidad.

La programación ha supuesto el elemento crucial para el desarrollo del simulador, debido a que como consecuencia de la misma se ha podido diseñar toda la lógica ya no solo del entorno generado, si no del comportamiento del personaje a controlar y de toda la Inteligencia Artificial.

En relación con la programación de la IA, se han alcanzado los objetivos iniciales; los enemigos suponen un desafío para el usuario y cuentan con un árbol de comportamiento que, si bien aún está lejos de ser perfecto, supone un acercamiento a la consecución de enemigos que cuenten con una forma de actuar y proceder cada vez más realistas.

Tal y como se planteó inicialmente, la interacción con el fusil, tanto su agarre como su funcionalidad de disparar y provocar daño en el enemigo ha sido alcanzado con éxito también.

La programación de los dos niveles ha sido resultado de una planificación minuciosa basada en las necesidades y los elementos que serían de provecho para un simulador de las características requeridas. Por lo tanto, ninguna de las decisiones tomadas en el diseño del nivel son causa del azar, a excepción de las propias consecuentes de la falta de tiempo; desde la inclusión de un temporizador para aumentar la sensación de estrés en el jugador, hasta la decisión de no incluir una barra de salud o contador de balas han sido razonadas para asemejar lo máximo posible la simulación a una situación de combate real.

5.2.1.3 La Realidad Virtual como herramienta

La implementación de esta tecnología tal y como era de esperar ha aportado la posibilidad de "meter" al usuario en el escenario con las ventajas que conlleva toda experiencia tridimensional.

La visualización, la interacción con los objetos y la sensación de inmersión de la RV han elevado la experiencia a un concepto que debe de seguir explorándose en aras de un mayor aprovechamiento. El potencial de la Realidad Virtual como herramienta, por tanto, no puede pasarse por alto en la recapitulación de las conclusiones obtenidas, sin embargo, también hay que mencionar las dificultades que esta puede plantear.

La falta de información, debido a que aún se trata de una tecnología poco "madura" sobre todo en términos de gamificación, supone a veces una dificultad añadida. Además, aún cuenta con ciertas limitaciones como el motionsickness o los problemas de las colisiones, que si bien como ya hemos visto en el desarrollo pueden ser solventados en mayor o menor medida gracias al ingenio, pueden desembocar a veces en una ruptura inmediata de la inmersión.

El presente proyecto ha tratado y evaluado estos problemas planteados de una manera correcta. Para hacerles frente se propusieron hasta dos sistemas de movimientos para evitar los mareos durante la simulación y se generó una respuesta que reaccionaba al problema de las colisiones con las paredes del nivel.

En contraposición de los problemas menores que pueda plantear en momentos puntuales, la Realidad Virtual ha servido para demostrar las ventajas de la simulación frente al adiestramiento real. Todo lo que una simulación no puede calcar de la realidad, lo compensa con la capacidad de procesamiento de datos en tiempo real que ofrece. Poder generar un documento detallado, con datos tales como la precisión de cada alumno, es un enorme paso que la enseñanza puede dar si empieza a hacer uso de tecnologías como esta de manera más asidua.

5.2.2 Conclusión final

La Armada 4.0 no es el mañana, debe ser el hoy. Entendiendo el progreso y la evolución como un vector que debemos dirigir hacia un dominio híbrido, en el que el «mundo físico» y el «entorno digital» interactúen de manera unánime, la Armada y las Fuerzas Armadas en general pueden verse en pocos años envueltas en una auténtica revolución en todos los niveles. Todo se reduce a identificar y aplicar las nuevas herramientas de las que dispone la Industria 4.0 para cumplir con los mismos objetivos y/o misiones que ejecutamos hoy en día, pero, probablemente, de una manera más eficiente. El uso de tecnologías como la Realidad Virtual o la Inteligencia Artificial no buscan en ningún caso sustituir al

ser humano, si no apoyarlo en los procesos que este ejecuta. Las ventajas que se presentan, como ya hemos podido comprobar con los resultados obtenidos de este proyecto, son y serán una de las armas más valiosas con las que podremos contar en los años venideros.

En concreto, el campo de la enseñanza y el adiestramiento militar pueden ser uno de los grandes beneficiados de esta revolución tecnológica. La llegada de la Realidad Virtual a las aulas y a los centros docentes militares, así como el acercamiento al futurista concepto de metaverso académico militar, solo pueden suponer una ampliación efectiva del abanico de posibilidades y recursos de unos planes de estudio más acordes con la sociedad tecnológica y avanzada en la que vivimos.

Se ha conseguido el objetivo principal, el simulador es una plataforma de adiestramiento virtual para la puesta en práctica de los movimientos tácticos necesarios durante una operación como la planteada. El análisis de datos posterior al asalto es una herramienta potente para la evaluación y el seguimiento personalizado de cada alumno. Además de los objetivos planteados inicialmente, también se han alcanzado otros nuevos que surgieron durante el desarrollo del proyecto.

Por eso, y como punto final a las conclusiones, el presente trabajo de fin de grado queda en términos totales finalizado de manera satisfactoria.

5.3 Líneas futuras

Para finalizar, se exponen las posibles líneas de desarrollo que se dejarán planteadas para mejorar el y ampliar el campo de investigación abierto por el proyecto.

5.3.1 A corto plazo

En primer lugar, como posible mejora, el simulador podría ser aún más realista si incluyese la opción de un modo multijugador en el que varios usuarios con equipos de Realidad Virtual pudiesen realizar la preparación previa a la operación y la posterior ejecución del nivel interactuando entre ellos durante el ejercicio, tal y como sucedería en la realidad.

El sistema de movimiento, así como las mecánicas del avatar principal podrían verse altamente beneficiadas con la inclusión de dispositivos que permitan caminar, correr y girar todo lo que el usuario requiera por el espacio real sin problemas de espacio. Estos dispositivos pueden ser desde cintas mecánicas hasta dispositivos de seguimiento acoplables en las botas del usuario.

Además, existen numerosas posibles mejoras de carácter mucho más evidente como la mejora del código, la inteligencia artificial, las animaciones o el aspecto del avatar principal.

5.3.2 A medio-largo plazo

Con la vista puesta en el futuro, sería interesante contemplar la posibilidad de ampliar el abanico de disciplinas abarcables por la simulación en Realidad Virtual. Se podría plantear el aumentar el número de espectros, incluyendo otros ámbitos indispensables para la enseñanza cómo lo son el ámbito naval o el aéreo.

Además, tras comprobar las ventajas de la interacción directa con los objetos que proporciona esta tecnología se podrían diseñar simuladores de adiestramiento más específicos que sirviesen para instruir al personal en la utilización de la maquinaria de un buque. Esto facilitaría la comprensión y el aprendizaje de numerosas tareas, como el control y el manejo del sistema de la propulsión o el reconocimiento de todas las opciones disponibles en una consola táctica.

6 BIBLIOGRAFÍA

[1] E. S. Gómez, «La Guerra Difusa: La Sociedad Meta-Bélica», p. 11 Disponible: http://serbal.pntic.mec.es/~cmunoz11/esanchez39.pdf.

[2] «La cuarta revolución industrial- Klaus Schwab. pdf». Disponible: https://www.fundaciobit.org/es/la-cuarta-revolucion-industrial-klaus-schwab-2/

[3] «¿Qué es la Industria 4.0?», *LIS Data Solutions*. Disponible: https://www.lisdatasolutions.com/blog/que-es-la-industria-4-0/ (accedido 16 de febrero de 2022).

[4] «Estrategia de Tecnología e Innovación para la Defensa ETID - 2020», p. 160. Disponible: https://www.tecnologiaeinnovacion.defensa.gob.es/es-es/Estrategia/Paginas/Defensa.aspx

[5] «E-LynX^{TM} Mobile Tactical SDR Solutions», *Elbit Systems*. Disponible: https://elbitsystems.com/products/communication-systems/software-defined-radios/ (accedido 16 de febrero de 2022).

[6] «Avenger UGV», *Med-Eng*. Disponible: https://www.med-eng.com/product/avenger/ (accedido 16 de febrero de 2022).

[7] «Mantis_Esp.pdf». Disponible: https://www.indracompany.com/sites/default/files/Mantis_Esp.pdf Accedido: 16 de febrero de 2022.

[8] M. A. C. Cardona, F. R. Spitia, y A. B. López, «Exoesqueletos para potenciar las capacidades humanas y apoyar la rehabilitación», *Rev. Ing. Bioméd.*, p. 11.

[9] «University of Michigan Study Suggests Soldiers Could Cover Inclined Terrain More Easily Using Lockheed Martin's FORTIS K-SRD Exoskeleton», Disponible: https://news.lockheedmartin.com (accedido 16 de febrero de 2022).

[10] «ENTAC Simuladores de altas prestaciones INDRA.pdf». Disponible: https://www.indracompany.com/sites/default/files/triptico_entac_simulador_conduccion_vehiculos_pesados_v01_baja_esp_.pdf

[11] «indra_fighters_flight_simulators_en_2019_2.pdf». Disponible: https://www.indracompany.com (accedido: 27 de enero de 2022).

[12] «La Escuela Naval Militar estrena dos simuladores de navegación», *Faro de Vigo*, 19 de enero de 2006. Disponible: https://www.farodevigo.es/pontevedra/2006/01/19/escuela-naval-militar-estrena-simuladores-18262890.html (accedido 27 de enero de 2022).

[13] J. I. V. Cancela, «SIMULADORES DE NAVEGACIÓN EN LA ENSEÑANZA NAVAL», Disponible: https://armada.defensa.gob.es/ArmadaPortal/page/Portal/ArmadaEspannola/mardigitalrevistas/prefLang-es/02revistaGenMarina--02catalogoRGM--2021--202106-es?_pageAction=selectItem&_selectedNodeID=4441197&paramNo=000000

[14] «Publicación combate en zonas urbanizadas del Ejército de Tierra.pdf». Disponible: https://publicaciones.defensa.gob.es/media/downloadable/files/links/m/o/monografia_126.pdf (accedido: 27 de enero de 2022)

[15] «LOS ÁMBITOS NO TERRESTRES EN LA GUERRA FUTURA: ESPACIO», p. 350. Disponible: https://publicaciones.defensa.gob.es/media/downloadable/files/links/m/o/monografia_128.pdf

[16] Publicación NATO «ATP-99 (A) Urban Tactics».

[17] Publicación de Defensa «PD4-021_Empleo de pequeñas unidades en ambiente urbano».

[18] España y Ministerio de Defensa, *Doctrina para el empleo de las FAS: PDC-01(A).* 2018. Disponible: https://publicaciones.defensa.gob.es/pdc-01-a-doctrina-para-el-empleo-de-las-fas-libros-papel.html

[19] «Adiestramiento en las Fuerzas Armadas.pdf» Disponible: https://www.defensa.gob.es/portaldecultura/Galerias/actividades/fichero/2013_QUERO L_03_A_14.pdf (accedido: 27 de enero de 2022).

[20] «RCZM66 Combate en poblacion - Ejército de tierra». Disponible: https://ejercito.defensa.gob.es/multimedia/videos_2016/2016_rczm66_combate_poblacion.html (accedido 16 de febrero de 2022).

[21] «FUSA HK - Ejército de tierra». Disponible: https://ejercito.defensa.gob.es/materiales/armamento_ligero/FUSIL-G36E.html (accedido: 2 de febrero de 2022).

[22] «CREATIVIDAD INMERSIVA, INMERSIVIDAD CREATIVA». Disponible: https://www.accioncultural.es/media/2018/ebook/Anuario/2JMMene%CC%81ndezDJBermejo.pdf (accedido: 2 de febrero de 2022)

[23] «POTENTIALITIES OF DRIVING SIMULATOR FOR ENGINEERING APPLICATIONS TO FORMULA 1». Disponible: https://cdn.archilovers.com/projects/39c6eb1a-29fb-4e19-8112-736fcb14ab84.pdf (accedido: 2 de febrero de 2022).

[24] «F1 Driving Simulators Ferrari Museums - Ferrari.com». Disponible: https://www.ferrari.com/en-EN/museums/f1-driving-simulation (accedido: 18 de febrero de 2022).

[25] M. Wagner, *On the Scientific Relevance of eSports.* 2006, p. 442. Disponible: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.82&rep=rep1&type=pdf

[26] S. Mandal, «Brief Introduction of Virtual Reality & its Challenges». Disponible: https://www.ijser.org/researchpaper/Brief-Introduction-of-Virtual-Reality-its-Challenges.pdf

[27] T. Mazuryk y M. Gervautz, «Virtual Reality - History, Applications, Technology and Future». Disponible: https://www.researchgate.net/publication/2617390_Virtual_Reality_-_History_Applications_Technology_and_Future

[28] «History, application and use of Steam VR». Disponible: https://store.steampowered.com/app/250820/SteamVR/?l=spanish (accedido: 8 de febrero de 2022.

[29] J. L. Ordóñez, «Realidad Virtual y Realidad Aumentada», p. 21.Disponible: https://www.acta.es/medios/articulos/ciencias_y_tecnologia/063001.pdf

[30] E. Chang, H.-T. Kim, y B. Yoo, «Virtual Reality Sickness: A Review of Causes and Measurements», *Int. J. Hum.-Comput. Interact.*, vol. 36, pp. 1-25, jul. 2020,

[31] L. J. Hettinger y G. E. Riccio, «Visually Induced Motion Sickness in Virtual Environments», *Presence Teleoperators Virtual Environ.*, vol. 1, n.º 3, pp. 306-310, ene. 1992

[32] V. Mathivet, *Inteligencia artificial para desarrolladores: conceptos e implementación en C#*. Ediciones ENI, 2018.

[33] I. Sagredo-Olivenza, G. Florez-Puga, M. A. Gomez-Martın, y P. A. Gonzalez-Calero, «Implementacio´n de nodos consulta en ´arboles de comportamiento», p. 12. Disponible: http://ceur-ws.org/Vol-1394/paper_14.pdf

[34] «Epic Games | About», *Epic Games*. https://www.epicgames.com/site/es-MX/about (accedido: 8 de febrero de 2022).

[35] A. Sanders, *An Introduction to Unreal Engine 4*. New York: A K Peters/CRC Press, 2016.

[36] B. Sewell, *Blueprints visual scripting for unreal engine: build professional 3D games with Unreal Engine 4's Visual scripting system*. Birmingham: Packt Publishing, 2015.

[37] «SteamVR en Steam». https://store.steampowered.com/app/250820/SteamVR/ (accedido 8 de febrero de 2022).

[38] «Funciones de 3ds Max | Funciones de 2022, 2021, 2020 | Autodesk». https://www.autodesk.es/products/3ds-max/features (accedido 8 de febrero de 2022).

[39] В. Санжаров *et al.*, *Examination of the Nvidia RTX*. 2019, p. 12. doi: 10.30987/graphicon-2019-2-7-12.

[40] A. S. Glassner, Ed., *An introduction to ray tracing*, Transferred to digital print. San Francisco, Calif: Morgan Kaufmann, 2007.

[41] «Comparison of Oculus Rift and HTC Vive: Feasibility for Virtual Reality-Based Exploration, Navigation, Exergaming, and Rehabilitation | Games for Health Journal». Disponible: https://www.liebertpub.com/doi/full/10.1089/g4h.2017.0114 (accedido: 9 de febrero de 2022).

[42] D. M. Shafer, C. P. Carbonara, y L. Popova, «Controller Required? The Impact of Natural Mapping on Interactivity, Realism, Presence, and Enjoyment in Motion-Based Video Games», *Presence Teleoperators Virtual Environ.*, vol. 23, n.º 3, pp. 267-286, oct. 2014, doi: 10.1162/PRES_a_00193.

[43] «M4VR Rifle Adapter for Vive» Disponible: *https://www.beswinvr.com*. https://www.beswinvr.com/M4VR-Rifle-Adapter-for-Vive-

p985884.html?utm_source=sns_share&utm_medium=open_graph (accedido 14 de marzo de 2022).

[44] «Sims vs. Games: The Difference Defined», *Edutopia*. Disponible: https://www.edutopia.org/sims-vs-games (accedido 9 de febrero de 2022).

[45] J. Wintjes, «"Not an Ordinary Game, But a School of War": Notes on the Early History of the Prusso-German Kriegsspiel», *Vulcan*, vol. 4, n.º 1, pp. 52-75, ago. 2016.

[46] T. Cornell y T. B. Allen, *War and Games*. Boydell Press, 2002.

[47] D. A. Clearwater, «Living in a Militarized Culture: War, Games and the Experience of U.S. Empire», *TOPIA Can. J. Cult. Stud.*, vol. 23-24, pp. 260-285, oct. 2010,

[48] «Atari's Army Battlezone Project | ROMchip». Disponible: https://www.romchip.org/index.php/romchip-journal/article/view/108

[49] S. Belli y C. L. Raventós, «A brief history of videogame», n.º 14, pp. 159-179, sep. 2008.

[50] L. Grossman y J. Ressner, «The Age of Doom.», *Time*, vol. 164, n.º 6, 2004, Disponible: https://www.elibrary.ru/item.asp?id=8461461

[51] Z. Li, «The potential of America's Army, the video game as civilian-military public sphere», Thesis, Massachusetts Institute of Technology, 2003. Disponible: https://dspace.mit.edu/handle/1721.1/39162

[52] «Rethinking Military Gaming: America's Army and Its Critics - Marcus Schulzke, 2013». https://journals.sagepub.com/doi/abs/10.1177/1555412013478686 (accedido 9 de febrero de 2022).

[53] K. M. Kovatch, «Call To Duty: Video Game Effects on the Military», Naval Postgraduate School, sep. 2018. Disponible: https://apps.dtic.mil/sti/citations/AD1065398

[54] «Military Games», *Giant Bomb*. https://www.giantbomb.com/military/3015-1009/games/ (accedido 9 de febrero de 2022).

[55] F. Cancio@ferpott, «Récord de misiones en el exterior», *La Razón*, 4 de enero de 2015. Disponible: https://www.larazon.es/espana/record-de-misiones-en-el-exterior-KE8327682/ (accedido 22 de febrero de 2022).

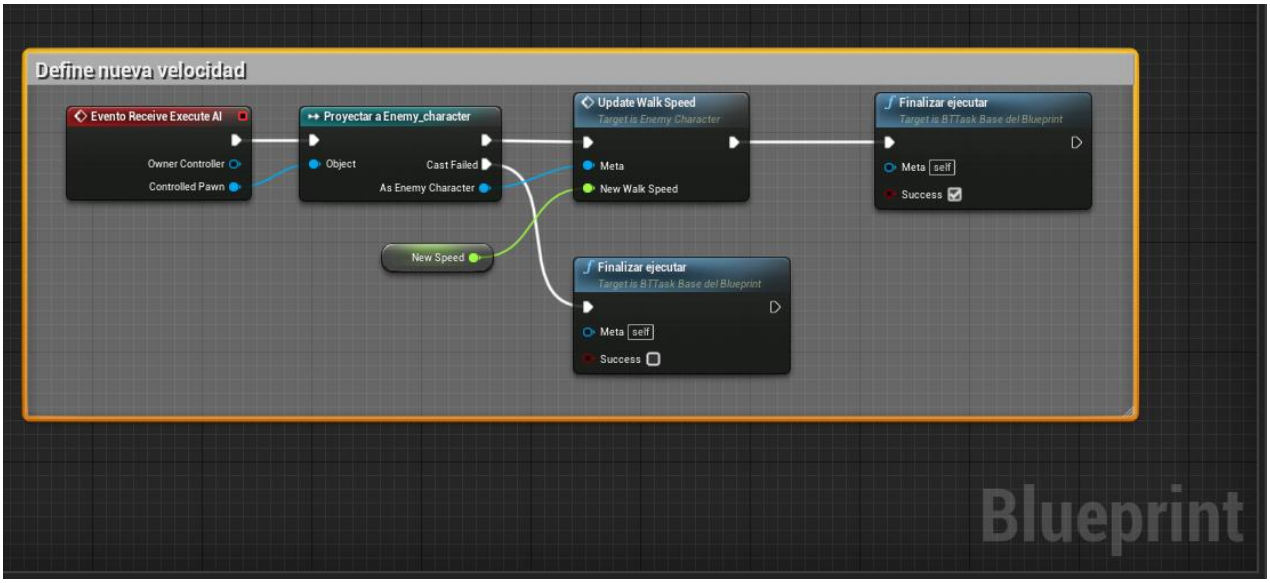
[56] «This Is Life in Rural Afghanistan After the Taliban Takeover - The New York Times». https://www.nytimes.com/2021/09/15/world/Afghanistan-rural-life-under-Taliban.html (accedido 22 de febrero de 2022).

[57] J. Boen, «Sound in Video Games: How Sound Is an Important Aspect of the Virtual Experience», p. 7.

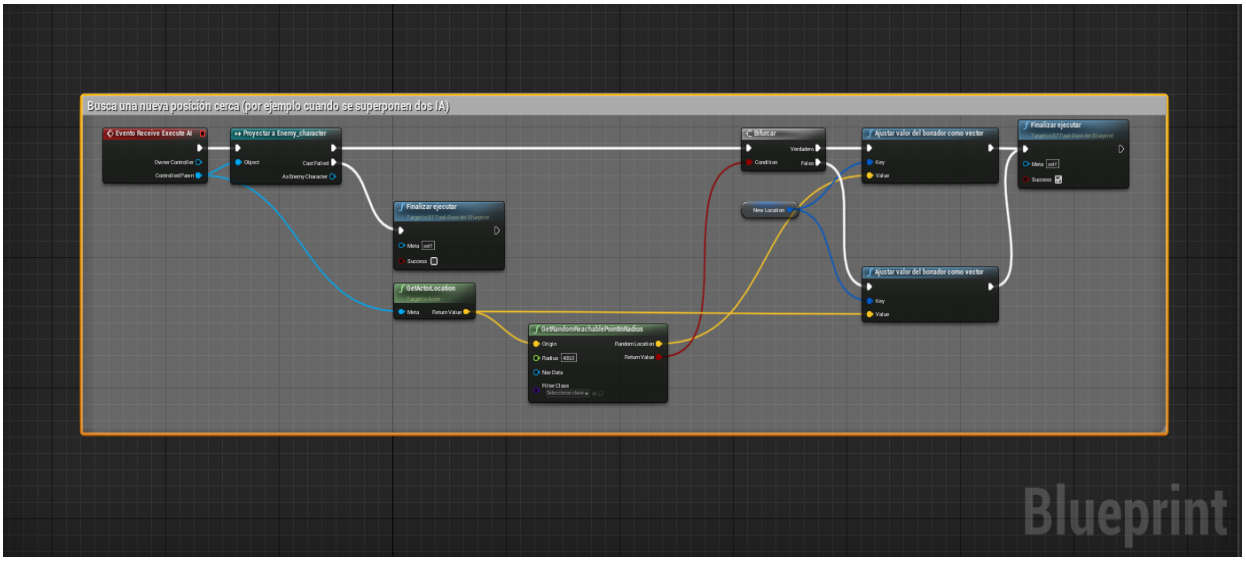
[58] «Video Game Level Design». Disponible: https://www.nuclino.com/articles/level-design

[59] K. Mack y R. Ruud, *Unreal Engine 4 Virtual Reality Projects: Build immersive, real-world VR applications using UE4, C++, and Unreal Blueprints*. Packt Publishing Ltd, 2019.

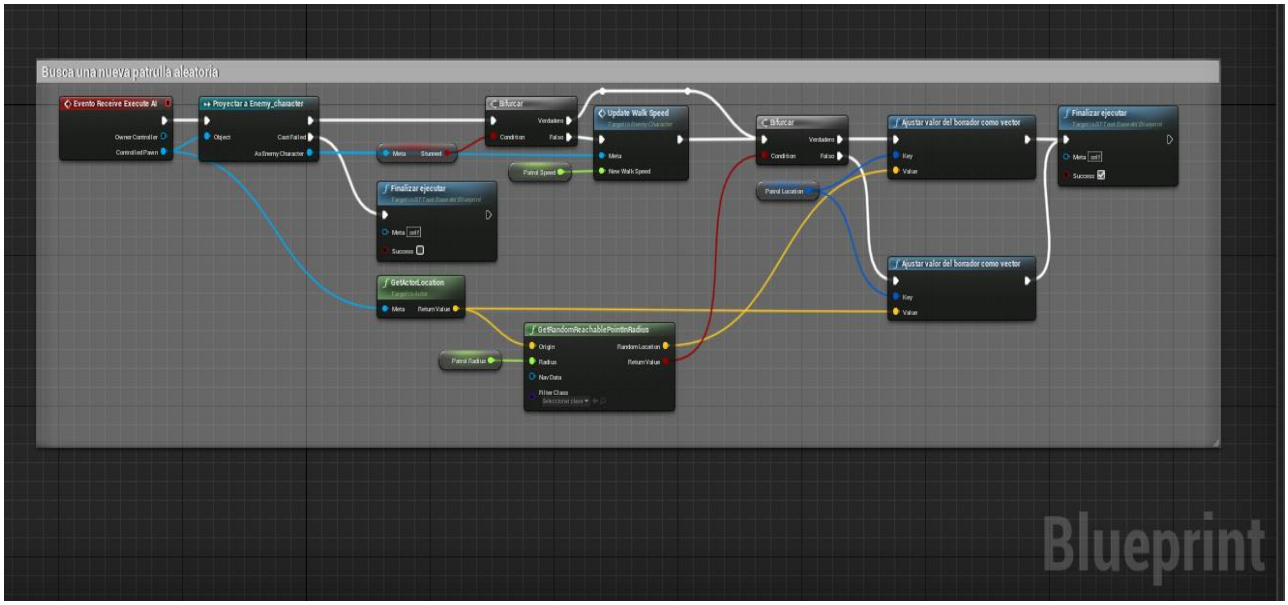
ANEXO I: BLOQUES DE CÓDIGO: TAREAS DEL ÁRBOL DE COMPORTAMIENTO DE LA IA



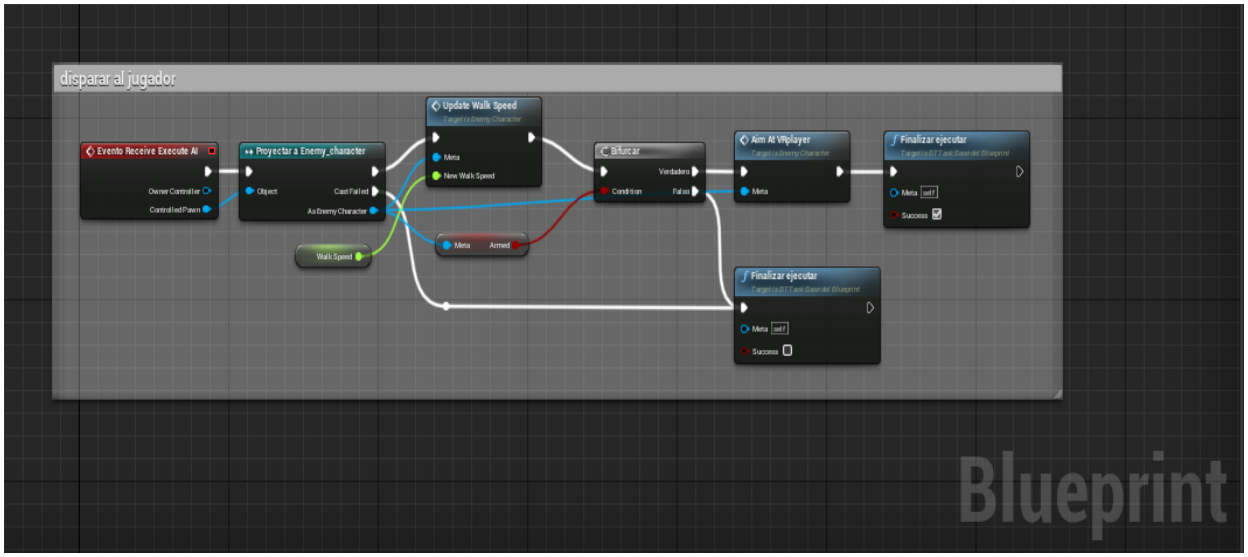
BTT CLEAR ENEMY LOCATION



BTT FIND NEARBY LOCATION

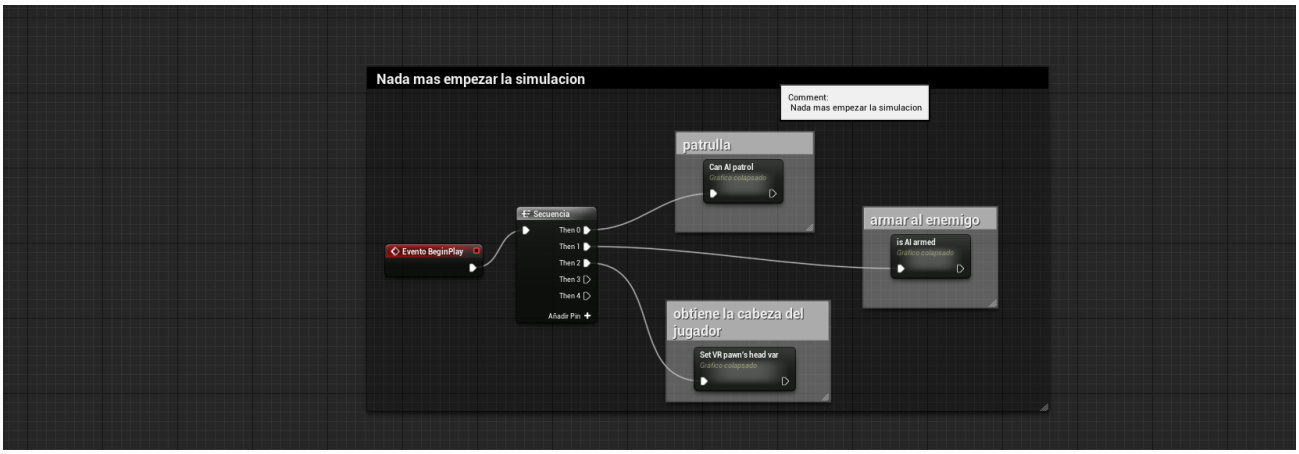


BTT FIND RANDOM PATROL

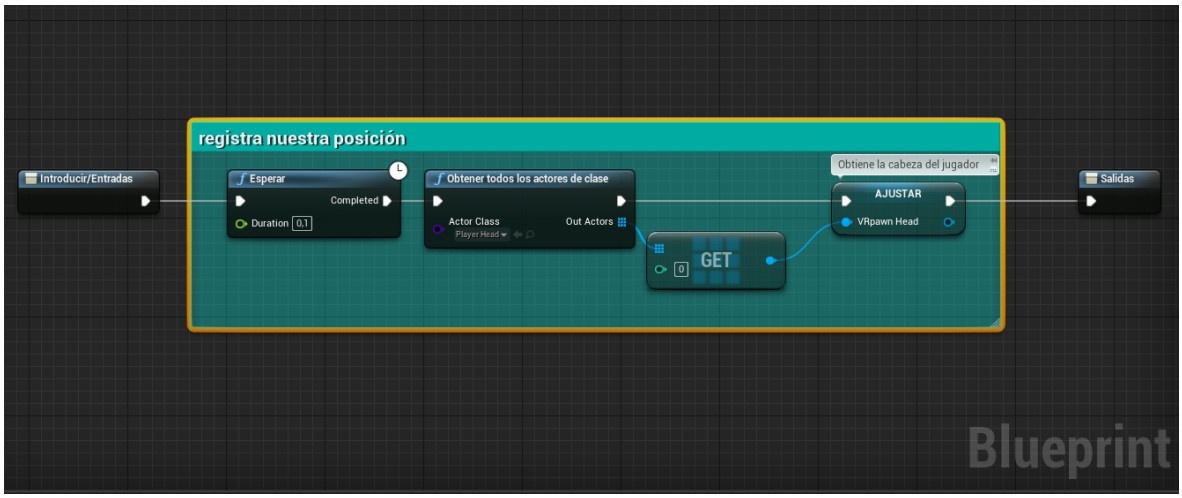


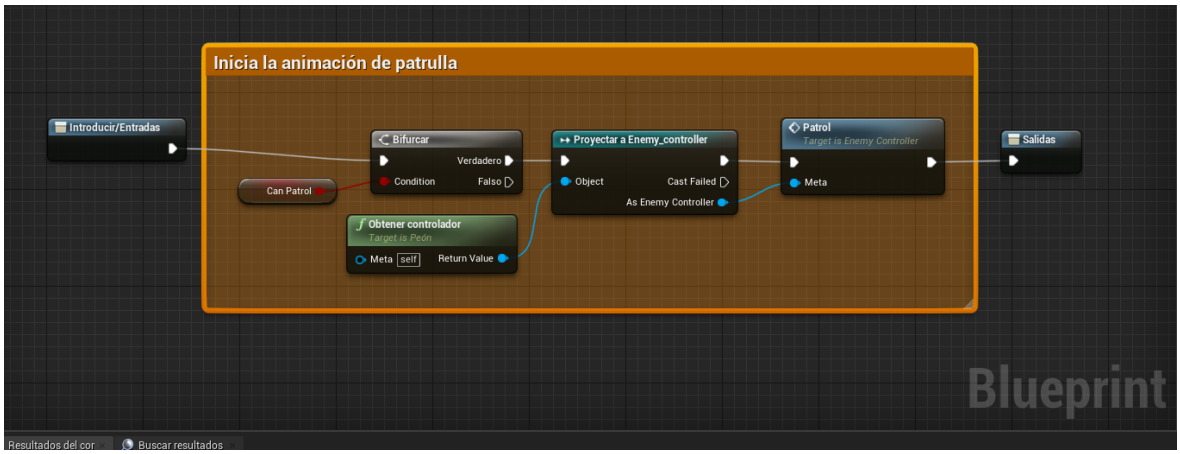
BTT SHOOT ENEMY

ANEXO II: BLOQUES DE CÓDIGO: INTELIGENCIA ARTIFICIAL

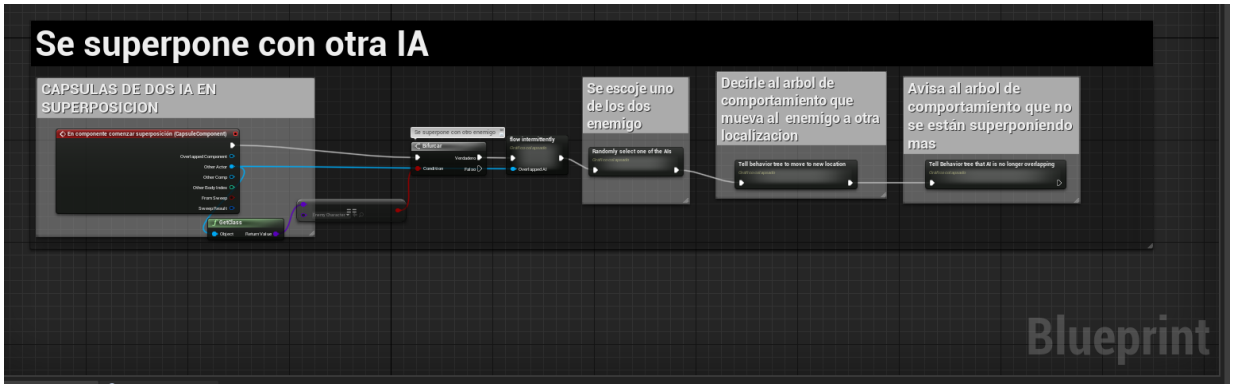


Blueprint del Setup inicial del enemigo completo

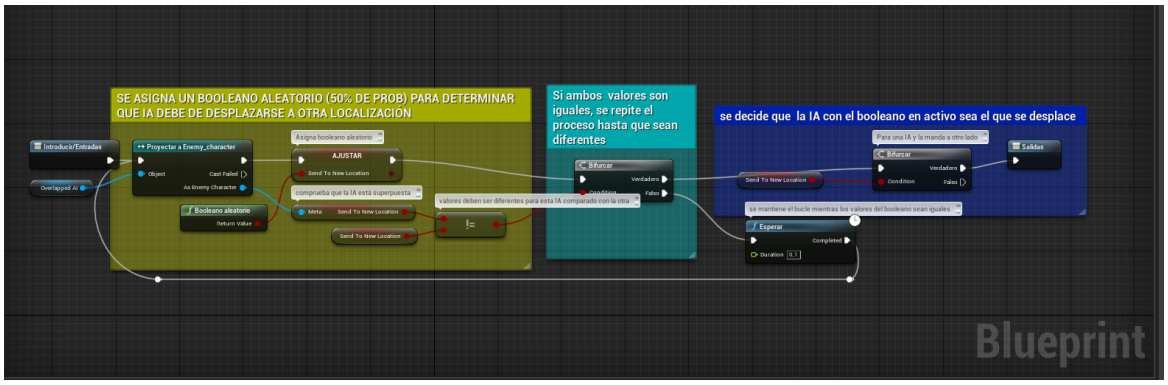


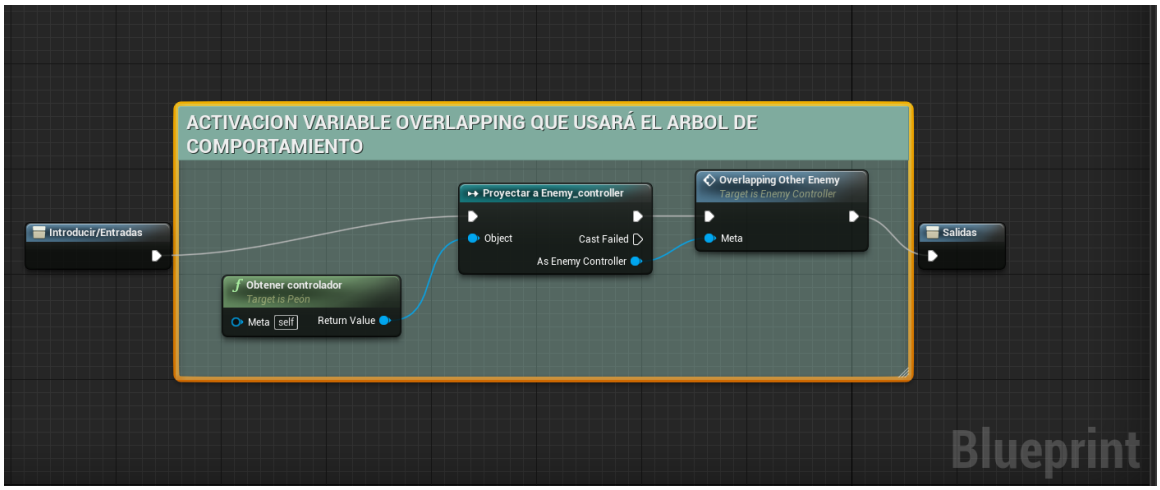


Blueprints del Setup inicial del enemigo desglosado

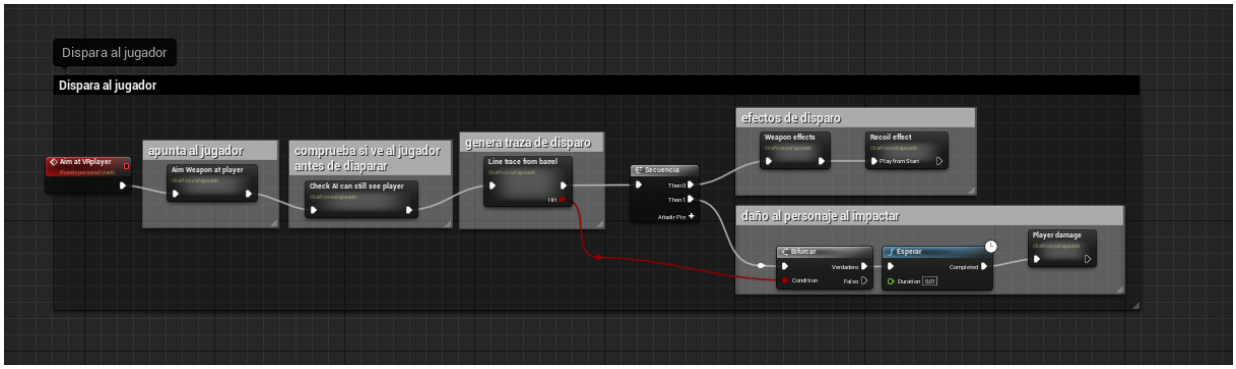


Blueprint de control del enemigo cuando se superpone a otro enemigo completo

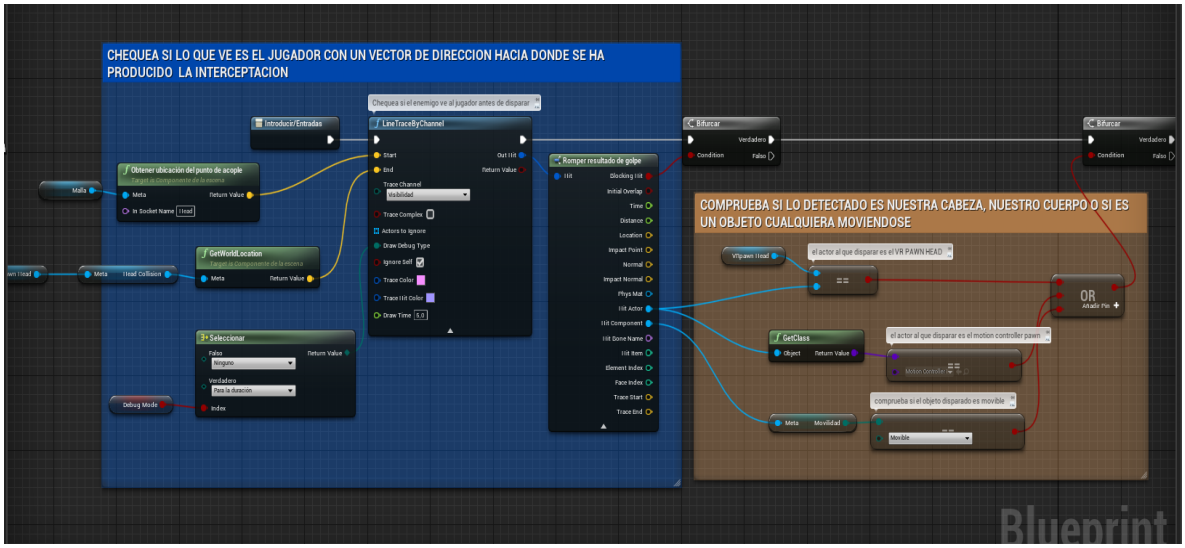


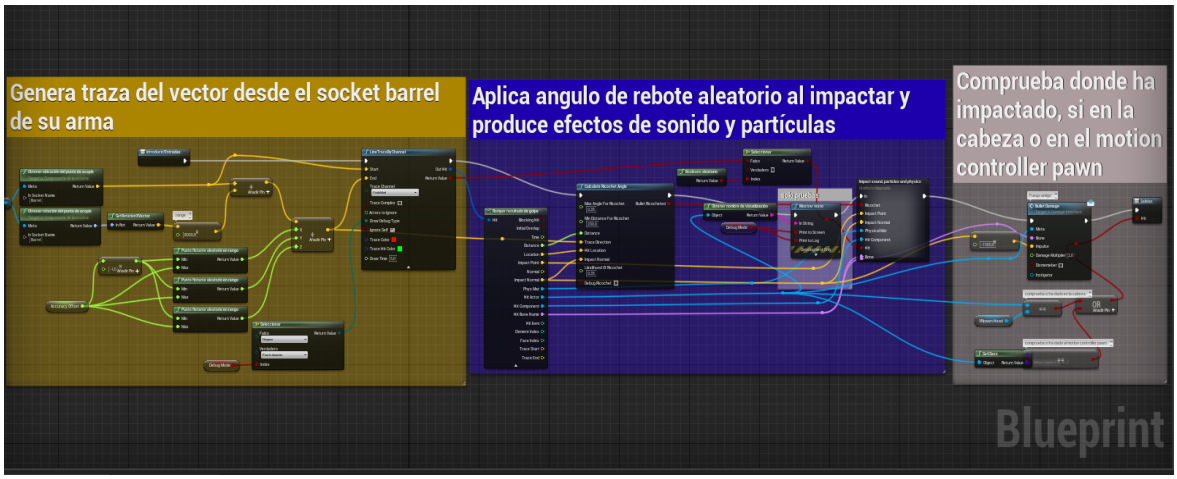


Blueprints de control del enemigo cuando se superpone a otro enemigo desglosado

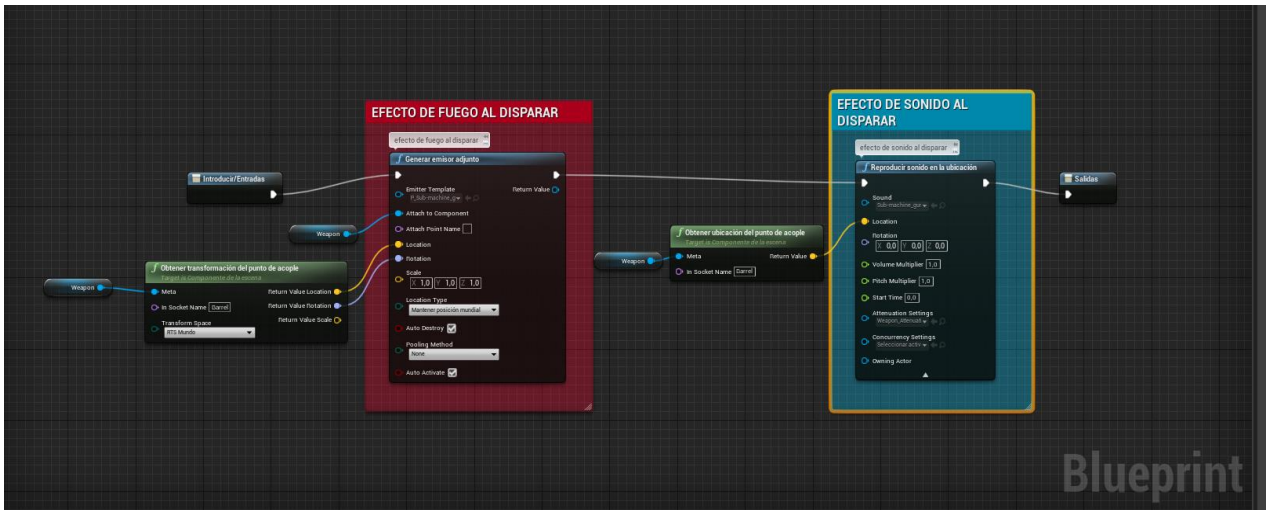


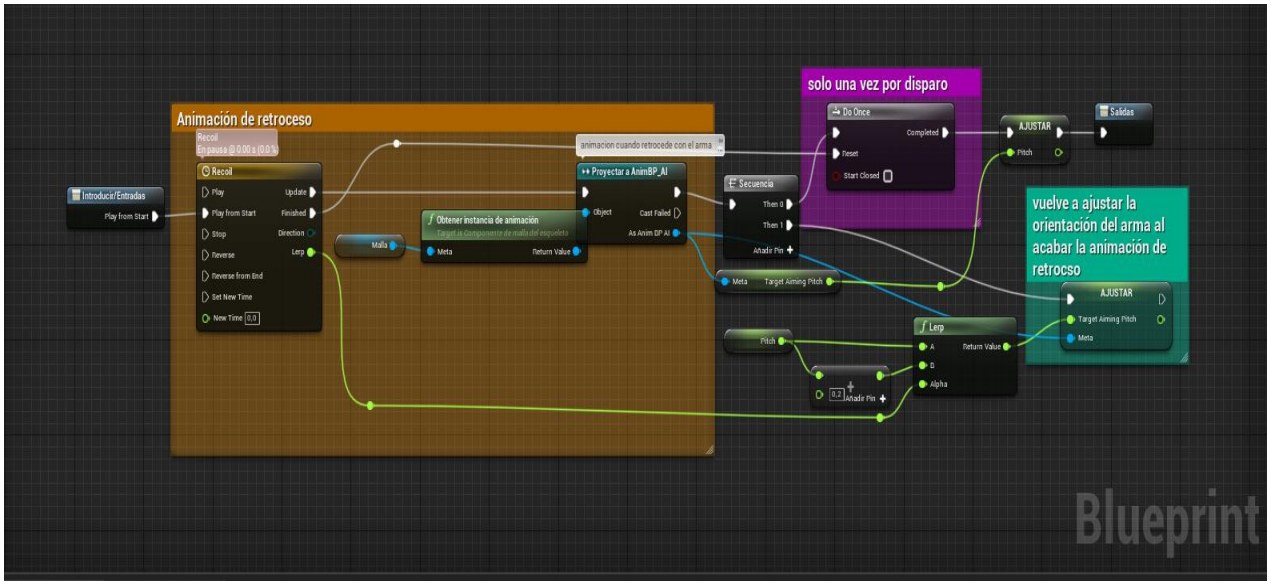
Blueprint de disparo al jugador completo





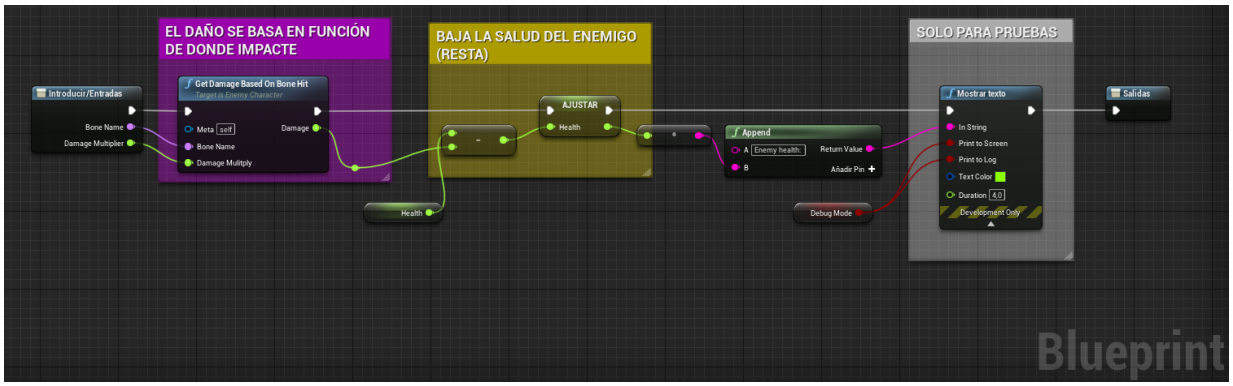
Blueprints del disparo al jugador desglosado 1º parte

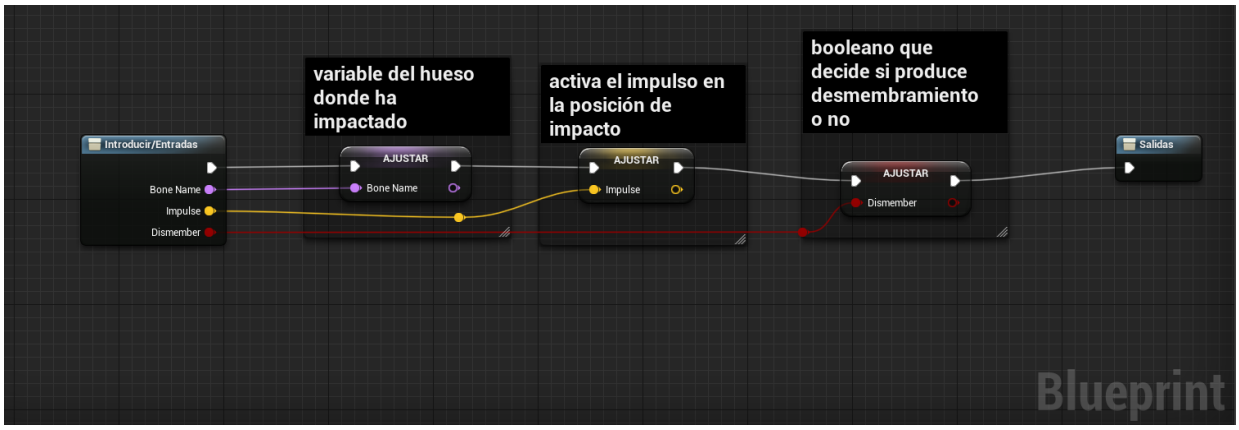




Blueprints del disparo al jugador desglosado 2º parte

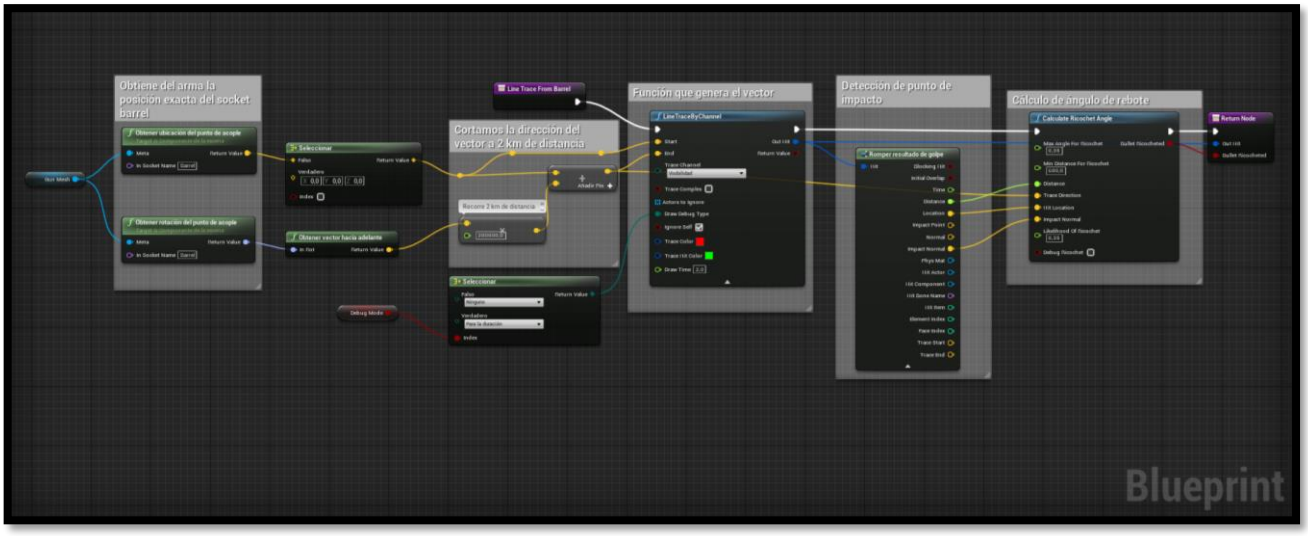
Blueprint de control de salud y daño de la IA completo





Blueprints de control de salud y daño de la IA completo

ANEXO III: BLOQUE DE CÓDIGO: DISPARO



Blueprint que controla el disparo

ANEXO IV: ADAPTADOR BESWIN VR M4 HTC VIVE