Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo de Fin de Grado

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Desarrollo de videojuego para la promoción del castillo de la Mota utilizando tecnología de realidad virtual

Autor:
D. Alfredo Fernández Ramos

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo de Fin de Grado

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Desarrollo de videojuego para la promoción del castillo de la Mota utilizando tecnología de realidad virtual

Autor:
D. Alfredo Fernández Ramos
Tutor:
Dr. Carlos Enrique Vivaracho Pascual
D. Pablo Sánchez Gatón

Agradecimientos

"A mis padres, a mis hermanos y mis amigos, hacia quienes sólo puedo expresar mi más sincero agradecimiento por apoyarme durante la etapa académica que hoy culmina."

Resumen

El mundo de los juegos está en un periodo de avance constante, causado por la cada vez mayor demanda de estos. El auge de tecnologías como la realidad virtual, está ampliando los horizontes de los videojuegos aportando un nuevo enfoque, que los diseñadores de videojuegos pueden explotar para crear nuevos sistemas de mecánicas. Hay que destacar que la realidad virtual es una tecnología que se encuentra en una fase muy temprana del ciclo de vida de producto, y que todavía son muy pocas las empresas distribuidoras y desarrolladoras de videojuegos que estén centrando sus esfuerzos en el uso de estas tecnologías. La búsqueda del realismo y la sensación de inmersión dentro del juego, son las principales motivaciones para utilizar esta tecnología. En este proyecto se pretende aprovechar estas ventajas, para desarrollar un videojuego moderno, que muestre el potencial de estas tecnologías en el uso de videojuegos de grandes dimensiones.

La utilización de la realidad virtual ya no solo se limita a la producción de videojuegos, sino que se apoya en motores gráficos para la creación de herramientas útiles en muchos ámbitos, como por ejemplo, el diseño gráfico en 3D, promoción turística, espeleología, medicina, etcétera.

En este trabajo convergen la necesidad de aprovechar el modelo en 3D de un castillo histórico, con los videojuegos para promover el conocimiento de esta edificación histórica y el turismo, así como para dar a conocer las posibilidades que puede ofrecer, utilizando tecnología de realidad virtual, la empresa *Irzón*, empresa creadora del modelo del castillo.

Abstract

The world of games is in a period of constant progress, caused by the increasing demand for these. The rise of technologies such as virtual reality is broadening the horizons of videogames, providing a new approach that video game designers can exploit to create new systems of mechanics. It should be noted that virtual reality is a technology that is in a very early phase of the product life cycle, and that there are still very few video game distribution and development companies that are focusing their efforts on the use of these technologies. The search for realism and the feeling of immersion within the game are the main causes of the success of motivations to use this technology. This project aims to take advantage of these advantages to develop a modern video game that shows the potential of these technologies in the use of large video games.

The use of virtual reality is not only limited to the production of video games, but is supported by graphic engines for the creation of useful tools in many areas, such as 3D graphic design, tourism marketing, caving, medicine, etc.

In this work converge the need to take advantage of the 3D model of a historic castle, with video games to promote knowledge of this historic building and tourism, as well as to make known the possibilities it can offer, using virtual reality technology, the Irzón company, creator of the castle model.

Índice General

Contenido

Índice de figuras

Índice de tablas

Capítulo I

Introducción

1.1. Juego y videojuego. Conceptos clave

El juego es una actividad que el ser humano practica desde el año 3000 a.c., año en el que se referencia a esta actividad por primera vez. Los primeros juegos podrían ser análogos a la comunicación, permitiendo esta antes de la existencia de un lenguaje. Forma parte de nosotros, nos permite desarrollarnos y socializarnos en muchos contextos.

"Juego es toda actividad que realizan uno o más jugadores, empleando su imaginación o herramientas para crear una situación con un número determinado de reglas, con el fin de proporcionar entretenimiento y diversión" [1].

Jugar no es exclusivo de la raza humana, ya que algunos animales utilizan estas actividades como vía de aprendizaje.

Un videojuego es un tipo específico de juego, en el que mediante el uso de tecnologías informáticas, se simula el entorno, dinámicas y mecánicas para un juego. Mediante unos controladores de entrada, el jugador puede comunicarse directamente con el entorno virtualizado.

Este proyecto es una continuación de otro proyecto: "Uso de Realidad Virtual para recorridos sobre edificios históricos" [2] de la Ingeniera Industrial Cristina Ribate del Álamo.

1.2. Motivación

El mundo de los videojuegos está en plena revolución y conceptos como "gamificación" [3] y "juego serio" [4] empiezan a sonar con fuerza en diferentes ámbitos empresariales.

En nuestro caso, a raíz de la necesidad de aprovechar el modelo en 3D de un castillo importante, como es el castillo de la Mota, surgen posibilidades prácticas de utilización del modelo.

Desde los años 80, se empiezan a utilizar los videojuegos con fines publicitarios. El "advergaming" (del inglés, advertising y game), es la práctica de crear videojuegos para estos fines, y forma parte de una subcategoría dentro de juego serio [5]. Nuestra finalidad será poder tener un producto software, el videojuego, que se pueda ofrecer a instituciones de turismo, instituciones públicas, e incluso, sector privado, para que puedan utilizarlo para promocionar el castillo y fomentar el turismo en Medina del Campo, pueblo al que pertenece el castillo.

En este proyecto, el público al que se quiere llegar y la manera de publicitar el castillo es diferente. Primero se intenta llegar a un sector de la población más restrictivo, más juvenil, público al que le interese el mundo de los videojuegos y la realidad virtual (VR). Segundo se busca que la gente que juegue al videojuego, vea la calidad del modelo del castillo y quiera comprobar el estado real del castillo. También se busca crear una sensación de familiaridad, cuando las personas visiten la zona y encuentren semejanzas con el videojuego.

Dado que la creación de un videojuego que pueda cumplir las exigencias del mercado, implicaría un equipo muy completo con diferentes perfiles profesionales, se busca entrar en el nicho de mercado

de los videojuegos independientes (más conocidos como “videojuegos indie”), ofreciendo una experiencia visual muy buena, dentro de los marcos de la tecnología de realidad virtual, y ofrecer una jugabilidad innovadora en el mundo de la realidad virtual, creando un videojuego para VR que mezcle dos géneros de videojuegos, acción FPS (First Person Shooter) y RTS (Real Time Strategy).

Dentro de este objetivo, hay que concluir, que también se buscará el impulso de la promoción de las tecnologías y los productos que puede ofrecer la empresa fabricante del modelo, Irzón.

1.3. Objetivos

Objetivo general: Realizar un videojuego usando como escenario el castillo de la Mota, que, además de uso lúdico, permita la promoción del castillo, utilizando tecnologías de realidad virtual

Este objetivo general se concreta en los siguientes objetivos específicos:

- El juego se realizará en 3D aprovechando un modelo ya existente del castillo.
- El escenario deberá ser lo suficientemente detallado para que el jugador pueda apreciar el castillo, pero sin restar fluidez al juego.
- El juego se realizará para gafas de realidad virtual, para lograr una sensación más inmersiva en el jugador.
- Para una mejor promoción del castillo, el juego debe favorecer que el jugador explore tanto interior y como su exterior.
- El juego debe ser divertido y “enganchar” al jugador

Capítulo II

Teoría de videojuegos

No hay muchos trabajos de tipo científico en torno al estudio de los videojuegos. Uno de las más importantes es el realizado por Hunicke, LeBlanc y Zubek [6], proponiendo el modelo MDA, que se compone de mecánicas, dinámicas y estéticas para generar en el jugador una experiencia o respuesta emocional.

La integración de las mecánicas y las dinámicas se realiza mediante los componentes del videojuego. Responsables de generar esta integración y también de definir la lógica del videojuego.

2.1. Modelo MDA

El modelo MDA (Mechanics, Dynamics, Aesthetics) (Figura 1) es un enfoque formal para el diseño de videojuegos. Está basado en capas y fue propuesto por Hunicke, LeBlanc y Zubek en el taller “Challenges in Games AI” del congreso estadounidense “National Conference of Artificial Intelligence” [6].

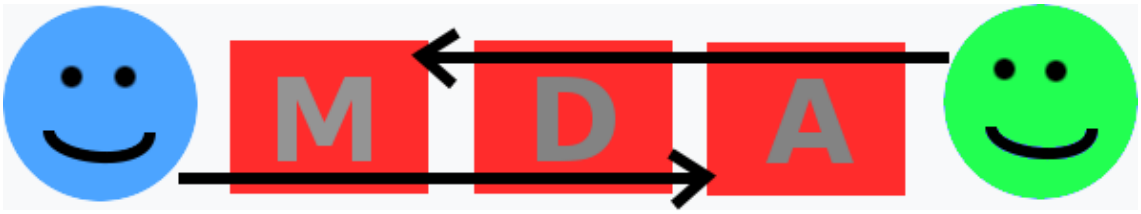


Figura 1: Diagrama modelo MDA

Una de las características del modelo MDA, es que permite ordenar las capas en función del rol que las visualice: El diseñador del videojuego o el jugador (ver Figura 1, diseñador en azul, jugador en verde). Desde la perspectiva del diseñador, las mecánicas generan dinámicas y estas a su vez generan estéticas, entendidas como las respuestas emocionales del jugador. El diseñador solo puede crear las mecánicas. Estas, por sí solas deben ser suficientemente capaces de generar las dinámicas, que generarán las estéticas y que serán las responsables de proporcionar la experiencia de juego al usuario. El usuario por su parte, percibe el videojuego con un flujo en orden inverso. A través de lo que él experimenta por medio de las estéticas, asimila las dinámicas, y estas están regidas por reglas definidas en las mecánicas.

Usando este modelo se puede extraer la funcionalidad de un videojuego, permitiendo clasificar los elementos de este para explotar la capacidad de generar emociones en la persona que lo juega.

2.1.1. Elementos del modelo MDA

Los elementos que conforman este modelo de diseño ya se habían utilizado con anterioridad para analizar videojuegos existentes, pero la utilización de estos como marco para el diseño, es algo novedoso.

Estos elementos y sus características se describen a continuación:

Mecánicas

Las mecánicas son los componentes básicos del juego, permiten que el jugador avance dentro del juego. Las mecánicas definen las reglas del juego, las acciones del jugador, las respuestas de estas acciones dentro del juego, el modelo de datos del juego, la comunicación, etc.

Hay muchos elementos y diferentes clasificaciones para estos dentro de las mecánicas. Podemos definirlos siguiendo la lista de Jesse Schell definida en su libro “The Art of Game Design” [7]:

- Espacio: Se trata de un escenario abstracto, no entra dentro de este elemento la parte artística, tecnológica o multimedia. Es un marco físico de contexto para el videojuego, una escena donde se desarrolla este.

- Objetos: Todos los elementos que existen dentro de un espacio. Estos tienen atributos, estados y relaciones.

- Acciones: Son todas las interacciones que el jugador puede tener con estos objetos, cabe destacar que nos referimos a jugador en este contexto, como el objeto jugador, cuyas relaciones con otros objetos definen las acciones.

- Reglas: Estas son las limitaciones y las normas básicas que definen la jugabilidad del videojuego. Y gracias a estas se permite el avance del jugador dentro del juego.

- Habilidades: Se caracterizan por el tipo de juego o la experiencia que se quiera crear en el jugador.

- Casualidad: Tanto las reglas como las relaciones entre los objetos pueden, no siempre estar perfectamente definidas, depende del tipo de juego que se desee diseñar agregar un componente aleatorio puede mejorar la experiencia del usuario. Reducen la predictibilidad del videojuego.

Dinámicas

Las dinámicas son resultado directo de la interacción entre el jugador y las mecánicas definidas por el diseñador. Según el tipo de dinámicas generadas, se puede hacer una clasificación del estilo de juego, clasificados en función de la jugabilidad que el videojuego aporta al jugador. Diferentes usuarios establecen tipos de jugador diferentes, que buscan a su vez, diferentes tipos de experiencias. Existen varios estilos de juego clasificados por las dinámicas que los hacen posibles:

- Acción: Es un tipo de juego en el que sus dinámicas principales marcan como objetivo al jugador avanzar en el videojuego en base a las mecánicas de movimiento y combate.

- Rol: Las dinámicas específicas de este tipo de juegos se basan en un avance lineal a través de una historia mejorando las características del objeto personaje, con el que interactúa directamente el jugador. Las mecánicas principales de este estilo de juego son la mejora de atributos del objeto personaje, en función del avance y de la elección de unas opciones u otras por parte del jugador.

- Simulación: En este género de videojuego, las dinámicas generan una sensación de similitud con la realidad. Se crean mecánicas con una lógica muy similar a la de la realidad en función del tipo de simulación.

- Estrategia: El jugador toma decisiones sobre las relaciones de los objetos entre sí para que el juego evolucione siguiendo un patrón que el jugador intenta controlar y predecir por medio de estas decisiones y algo de azar.

- Party: Juegos centrados en las dinámicas de grupo y competitividad, se usan mecánicas dedicadas a la competitividad entre diferentes jugadores.

Estéticas

Las estéticas definen las respuestas emocionales evocadas en los jugadores. En función de estas se puede realizar otra clasificación de videojuegos:

- Sensación: las principales estéticas son los efectos audiovisuales

- Fantasía: Evasión de la realidad por medio de la fantasía y la narrativa.

- Desafío: Cumplir retos para fomentar al jugador una competitividad intrínseca.

- Comunidad: Fomentan la relación con otras personas/jugadores

- Descubrimiento: El juego insta a los jugadores a explorar.

2.2. Evolución de los videojuegos.

Los videojuegos han formado parte de la sociedad mucho tiempo, razón por la cual han ido evolucionando. El primer videojuego que puede considerarse como tal, “Tennis For Two”, desarrollado por el físico William Higinbotham [8], utilizaba un ordenador analógico conectado a un osciloscopio, el cual hacía de pantalla. Este juego fue el precursor del pong, pero con la diferencia de que este presentaba el campo de juego desde un punto de vista lateral. En la actualidad se utilizan unos periféricos ya preparados, como el caso de monitores para mostrar el juego, o tecnologías más modernas como la realidad virtual de la que se hablará más adelante.

Las limitaciones del hardware limitan a los desarrolladores, por esta razón el estilo del videojuego ha cambiado tanto en relación a los años. Durante las décadas de los 70 y 80 al no tener suficiente capacidad de procesamiento para renderizar escenarios, los juegos solían ser en dos dimensiones, con una vista cenital o lateral en su gran mayoría. Para conseguir visualizar la profundidad como

tercera componente en un sistema de referencia cartesiano, tenían que desarrollar videojuegos limitados a la vista en primera persona, en la que se dibujaban las líneas con un grado de inclinación usando técnicas de perspectiva isométrica para dar sensación de profundidad(Figura 2). Otras técnicas como la basada en capas de visualización y vista lateral supuso el origen de un nuevo género de videojuegos, como los “arcade”, “beat’em up”, etc.

En la actualidad gracias, en parte, a la explotación de las tarjetas gráficas como hardware motor de los videojuegos, los videojuegos tienen pocos límites aparte de la creatividad de los desarrolladores. Surgen nuevas tecnologías para cambiar el sistema de visionado del videojuego, como la realidad virtual y la realidad aumentada. A raíz de estas nuevas tecnologías los desarrolladores empiezan a desarrollar videojuegos centrados en estéticas de sensación de realismo e inmersión dentro del juego.

Figura 2: Videojuego Doom 1993

2.2.1. Roles de Jugadores

Con estas nuevas generaciones de videojuegos y el auge creciente que estos tienen en la sociedad, se empiezan a estudiar perfiles de jugadores para lograr satisfacer las exigencias de estéticas de los usuarios. Dando lugar a los roles de jugador.

Estos clasifican a los jugadores en función de qué tipo de estéticas prefieren en los videojuegos. Un ejemplo de estos roles sería el del jugador explorador, una persona que consigue su máximo divertimento descubriendo todas las dinámicas y las mecánicas del juego. Crear contenido para este perfil de jugador, implicaría hacer hincapié en el diseño del escenario, intentar ofrecer libertad de movimiento (videojuegos de mundo abierto [9]), cuidar los detalles, y evitar restringir con demasiadas mecánicas las posibilidades del jugador. Sin embargo estas decisiones valdrían para satisfacer a este perfil de jugador, si el rol del jugador fuera otro, es probable que estas medidas no fueran suficientes.

En este contexto, entra el llamado test de Bartle [10]. Esta es una clasificación de los jugadores basado en un escrito de 1996 hecho por Richard Bartle, que se basa en la teoría de los personajes. Existen cuatro modelos de personaje (Figura 3):

- Triunfadores: Son jugadores que prefieren ganar logros, puntos, niveles dentro del juego. Se centran en alcanzar metas que no reportan un beneficio directo en el juego ni en su experiencia, solo buscan el prestigio de ser poseedores de estos “logros”

- Exploradores: Como ya explicamos en el ejemplo, son jugadores que prefieren descubrir todo el escenario del juego, a menudo se sienten restringidos cuando el juego les limita el movimiento, o el tiempo para realizar ese movimiento. Este perfil tiene una peculiaridad y es que son unos probadores de videojuegos excelentes, porque disfrutan descubriendo errores o “glitches” [11], ya sea en la generación del escenario, que les conceda acceso a zonas restringidas que no forman parte del juego, u otro tipo de errores que no afecten directamente al rendimiento del juego.

- Sociables: Para este perfil de jugador, lo más importante es la interacción con otros jugadores y a veces con personajes no jugables (NPC del inglés “non-player character). Utilizan el videojuego únicamente como herramienta para socializarse.

- Asesinos: Este perfil, es muy competitivo, buscan ser mejores en cualquier faceta que otros jugadores, suelen preferir juegos de carácter multijugador ya que les da acceso a una competitividad directa.

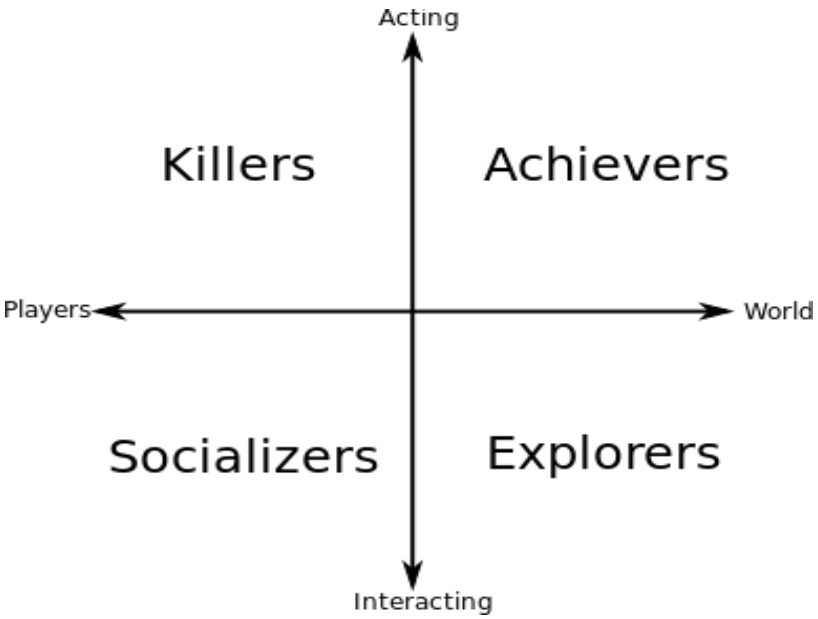


Figura 3: Gráfica de clasificación de la taxonomía de Bartle

En 2018, según Twitch, la mayoría de los jugadores se engloban en el perfil de asesino, según esta plataforma de “streaming” la mayoría de los videos y directos visionados fueron sobre juegos de competición directa [12]. Videojuegos como Fortnite, un juego de acción multijugador en primera persona, son el género preferido de este tipo de rol de jugador.

No obstante, es difícil encuadrar a un jugador únicamente en un perfil, por esta razón se añadió a la clasificación un nuevo eje de clasificación común a todas las categorías: implícito/explícito, generando 8 nuevos perfiles de jugador.

Lo normal es que un jugador tenga preferencias dentro de estos roles. Generando una tabla de frecuencias se puede clasificar al jugador según sus preferencias y diseñar la clase de videojuego que busca. (Véase Figura 4).

	B	C	D	E	F
1	Name	Achiever	Explorer	Socializer	Killer
2	Malcolm	87%	80%	67%	7%
3	Matt	73%	40%	40%	47%
4	Jack	73%	27%	33%	67%
5	Nathan	67%	27%	40%	67%
6	Tyler	60%	13%	60%	67%
7	Chris	60%	60%	47%	33%
8	Nathan	60%	93%	40%	13%
9	Nolan	60%	80%	33%	33%
10	David	60%	47%	33%	80%
11	Logan	53%	33%	73%	40%
12	Jordan	53%	87%	47%	13%
13	Isaiah	53%	80%	47%	27%
14	Kapri	47%	73%	47%	33%
15	Jonathan	47%	53%	40%	60%
16	Rafe	40%	47%	47%	67%
17	Hayley	40%	80%	20%	60%
18	Caitlin	33%	80%	73%	20%
19	Kira	33%	53%	60%	53%
20	Connor	33%	33%	47%	93%
21	Kirk	27%	53%	73%	53%
22	Crystal	27%	80%	33%	67%

Figura 4: Ejemplo de tabla de frecuencias según perfiles de jugador

2.3. Realidad Virtual

La realidad virtual es una tecnología, que por medio de unas gafas especiales y unos controladores permite reproducir visualmente una escena en 3D e interactuar con un entorno sensorial tridimensional. Según la RAE: “Representación de escenas o imágenes de objetos producida por un sistema informático, que da la sensación de su existencia real.” [13].

Esta tecnología se ha popularizado en la última década, gracias a una mejora de las herramientas que permiten simular estos entornos tridimensionales. Actualmente existe mucho contenido en videojuegos apto para estas tecnologías. También ha ayudado la creación de API’s como openVR, que permiten a los desarrolladores integrar la funcionalidad de sus videojuegos con el hardware de la realidad virtual.

Ejemplos de productos hardware VR son: Oculus rift o HTC vive (Figura 5).

Figura 5: Gafas de VR HTC Vive

2.3.1. Diferentes tecnologías de realidad virtual

Como hemos explicado anteriormente, el concepto de realidad virtual puede ser muy amplio, podríamos estar hablando de un juego de simulación de conducción como un ejemplo de realidad virtual, sin embargo en nuestro caso, estamos centrándonos en aquellas tecnologías de realidad virtual, que por medio de unos visores especiales y detectores de posición nos permiten ubicarnos dentro de un escenario virtual.

Existen otro tipo de tecnologías como las siguientes:

- Avatares: Los usuarios se introducen en un escenario virtual utilizando un avatar que ha sido previamente generado por ordenador, o utilizando fotos reales.
- Proyección de imágenes reales: Se basa en la proyección de imágenes reales que serán aplicadas en la realidad.
- Por ordenador: Este tipo conlleva mostrar un mundo en tres dimensiones directamente en el ordenador, sin utilizar ningún periférico adicional. Podría decirse que la mayoría de juegos en 3D se engloban dentro de esta categoría
- Inmersión en entornos virtuales: Esta proporciona la mejor sensación inmersiva, consiste en vivir la realidad virtual a través de la interfaz cerebro-máquina, para una comunicación directa. Aunque en esta categoría ya hay interfaces lo suficientemente sofisticadas como para mantener una comunicación directa cerebro-máquina sin utilizar nuestros sentidos como intermediarios [14], la realidad es que nos encontramos todavía en un paso intermedio, en el que tenemos que utilizar unos periféricos especiales y nuestros sentidos para interactuar con esta tecnología. Este es el tipo de tecnología que vamos a usar.

Dentro de este género de la realidad virtual, existen muchas tecnologías, dependientes del hardware desarrollado por diferentes fabricantes. Ejemplos de este tipo de tecnología son las ofrecidas por HTC, Oculus, PlayStation VR, Gear VR o Virtual Boy, y serán estas en las que nos basamos para nuestro proyecto.

La principal diferencia entre las tecnologías anteriormente mencionadas son el tipo de periféricos que utilizan y la tecnología de visualización de las imágenes.

En nuestro caso utilizaremos las gafas de HTC Vive. Podríamos haber utilizado gafas como las Oculus Rift, que tienen características muy similares y suelen tener SDK's compatibles entre estas. El resto de dispositivos de VR tienen unas especificaciones menores, generando altas latencias entre renderizado de imagen y visualizado, lo que se traduce en una sensación de desincronización movimiento-visualización, que puede provocar mareos en los usuarios. Otras directamente utilizan una pantalla para ambos ojos lo que reduce en gran medida la sensación de inmersión, debido a que no tenemos la sensación de estar viendo directamente el mundo virtual, sino que cambiamos la visualización en un monitor, por una visualización en una pantalla más pequeña y más cerca de nuestros ojos, que sigue nuestros movimientos.

2.3.2. Especificaciones de HTC vive

Las gafas de realidad virtual de HTC (HTC vive), tienen los siguientes componentes:

- Gafas: se trata de un casco que incorpora unas lentes que permiten la visualización independiente de imágenes en cada ojo, generando una sensación de visionado en 3D.

- Controladores: Tienen forma de mando y son representados dentro de la escena del videojuego, permiten la detección de movimiento y disponen de varios botones accesibles mediante un mapeo de estos.

- Estaciones satélite: Estas permiten localizar la posición del jugador dentro del espacio real donde use las gafas, y restringir su movimiento en la realidad, evitando posibles accidentes por no poder visualizar obstáculos de la realidad estando dentro del juego.

2.3.3. Ventajas de usar tecnologías de VR

La principal ventaja que puede tener usar esta tecnología está relacionada con satisfacer a los jugadores que necesitan de estéticas inmersivas. Uno de los principales inconvenientes de los juegos es la limitación de estos para poder ofrecer experiencias de juego inmersivas. Esto es debido a que no podemos engañar con pantallas a nuestro cerebro, y aunque los desarrolladores utilicen herramientas para mejorar esta inmersión, como uso de bandas sonoras, efectos de sonido, gráficos hiperrealistas, visión en primera persona, etc., lo cierto es que las experiencias de jugadores que buscan la inmersión total en el videojuego es algo precaria.

La realidad virtual que actualmente existe en el mercado aún es de segunda generación, pero a pesar de ello los jugadores han notado una sensación de inmersión mucho más completa, y muchos optan por esta tecnología a pesar de que la calidad gráfica que ofrece, por ejemplo, en los detalles visuales, no es comparable a la que ofrecen las tecnologías tradicionales que usan monitores. Esto es debido a que es mucho más fácil engañar a nuestro cerebro si hacemos que la vista crea que está observando un entorno tridimensional. Si a esto le añadimos un sonido envolvente usando auriculares y la capacidad de interactuar con nuestras propias manos con el mundo, obtenemos una verdadera sensación de inmersión que puede satisfacer las necesidades de estas estéticas.

Capítulo III

Contexto

3.1. Precedentes. Modelo del castillo

Para la realización de este proyecto se partirá, como ya se ha comentado anteriormente, de un modelo del castillo proporcionado por la empresa Irzón, que ya ha sido utilizado y presentado para una demo de un recorrido virtual del castillo, como proyecto de fin de grado de Ingeniería en diseño industrial [15].

El modelo ha sido mejorado y optimizado para cumplir con las exigencias del videojuego. Se ha reducido notablemente el número de polígonos que lo conforman. Esto permitirá cargar en memoria el modelo con mucha más eficiencia, permitiendo que el grueso del coste computacional se dedique íntegramente al cálculo de las mecánicas del videojuego.

3.2. Descripción del videojuego.

El videojuego se desarrolla en un ambiente postapocaliptico, en el que buscando la supervivencia de la protagonista, nos ocultaremos en un Castillo para evitar las oleadas de zombis que nos intentarán dar caza. Utilizando un arco y flechas podremos defender el castillo de oleadas de zombis infinitas.

En el Apéndice E, podemos encontrar el “Game Design Document” (documento de diseño del juego) [16] en el que se detallan todos los aspectos del videojuego.

En resumen, el juego se basa en la utilización de las mecánicas del tiro con arco de SteamVR, un kit de desarrollo software para realidad virtual que nos proporciona interfaces e implementaciones, para la integración de la realidad virtual. En concreto hay una implementación que nos permite simular el tiro con arco en realidad virtual. [17].

El juego consistirá en dos fases:

1 Fase de preparación: En esta fase, el jugador podrá prepararse para el combate de las oleadas, fabricar flechas, reparar la puerta, explorar el castillo para encontrar puntos flacos de la defensa, etcétera.

2 Fase de oleada: En ésta, el jugador deberá eliminar a todos los enemigos que le correspondan en esa oleada, con lo que conseguirá puntuación que podrá usar para conseguir flechas especiales, o acumularla para conseguir alcanzar los mejores puestos en la clasificación.

3.2.1. Elementos del videojuego

Tal y cómo se definió el diseño del videojuego (epígrafe 1.1.3), éste se divide en 3 partes principales: Mecánicas, dinámicas y estéticas. La descripción de cada una de estas partes para nuestro videojuego es la siguiente.

Mecánicas:

- **Tiro con arco:** La principal mecánica se basa en el manejo del arco, que usará para derrotar a todos los enemigos de cada fase de oleada.

- **Competición:** Los jugadores compiten por obtener la mejor puntuación y escalar puestos en una clasificación global.

- **Mejora estratégica:** A medida que avanzan en el videojuego, los jugadores aprender a optimizar los recursos que utilizan, así como su puntería para conseguir acabar con las oleadas más rápido.

- **Sistema de oleadas:** Los enemigos tienen que ser atacados en la fase correspondiente, mientras que en la fase de preparación, el jugador puede explotar otras posibilidades del juego.

- **Historia mediante "Easter egg"** [18]**:** Se añaden determinados objetos dispersos por todo el castillo, con los que el jugador puede interactuar para descubrir parte de la historia del juego.

- **Fin de partida:** Dado que el juego no tiene un final práctico, por ser un juego basado en oleadas "infinitas", se define la regla principal de salida del videojuego. Cuando los puntos de vida del jugador descienden de 100 el juego ha terminado. Se calcula la última puntuación y se sube a la clasificación global si es superior a la máxima alcanzada por el jugador.

Dinámicas:

- **Explotación del rol "Asesino":** Se busca alentar este perfil de jugador por medio de las mecánicas descritas anteriormente, impulsar su afán de superación de los rivales anónimos mediante un sistema de clasificación global.

- **Explotación del rol "Explorador":** Se impulsa este perfil complementario al anterior, mediante la posibilidad de exploración del castillo y redescubrir la historia del mismo mediante las mecánicas de *"easter egg"*.

Estéticas:

- **Inmersión:** Se genera una sensación de inmersión fruto del escenario utilizado y de las mecánicas del tiro con arco combinado con la tecnología de RV.

- **Evasión de la realidad:** Se consigue al enmarcar al jugador en un mundo post-apocalíptico [19] por medio de los efectos audiovisuales generados por el entorno/modelos, la mecánica del tiro con arco y el escenario.

- **Sentimiento de mejora:** Se busca que el jugador sienta que va progresando en el juego adquiriendo habilidades cada vez mejores, ya sea con el manejo del arco, como con el sistema de teletransporte rápido para posicionarse mejor antes de las oleadas.

- **Diversión:** Uno de los objetivos es que el jugador se divierta con el videojuego.

3.2.2. Versiones del videojuego

Debido a la complejidad del videojuego se opta por separar la funcionalidad descrita en el GDD en un sistema de versiones finales separadas. En la tabla 1 se detallan las funcionalidades que se incluyen en cada versión.

ID VERSIÓN	FUNCIONALIDAD AÑADIDA
V1.0.0	Primera versión jugable, se podrá navegar por todo el castillo, saldrán oleadas en un "spawn" concreto y todas del mismo punto. Solo se podrá defender el castillo con el arco. Se implementará el sistema de rondas, el combate, y el evento fin de partida.
V2.0.0	Las oleadas saldrán de diferentes puntos del mapa, se implementará el sistema de supervivientes (solo para la creación de flechas). Se incluirá música a las escenas.
V3.0.0	Se implementará el sistema de defensa. Se incluirán objetos con los que interactuar para la historia, Audios y diálogos de la historia. Se añadirá la funcionalidad para almacenar la puntuación de los jugadores en una base de datos alojada en un servidor externo, también se incluirá un "dashboard" para mostrar las puntuaciones de los jugadores en una página web externa.
V4.0.0	Se utilizará esta versión para mejorar todas las cuestiones anteriores, conseguir mejores efectos visuales, corregir problemas de interacción, añadir más "easter eggs" para relatar la historia del juego.

Tabla 1 Versiones del videojuego

3.2.3. Evaluación del videojuego

Para la fase de evaluación del videojuego, se realizará un sistema de pruebas de aceptación basado en pruebas alfa y beta [20]. Este tipo de pruebas son típicas de los entornos de desarrollo de videojuegos, se establecen dos fases, típicamente llamadas fases alfa y beta. Ambas son pruebas sobre productos finales que hacen normalmente un grupo de usuarios finales, previamente seleccionados, si son fases cerradas.

Las pruebas alfa se realizarán en dos fases:

- **Pruebas alfa sobre funcionalidad específica:** En este caso se realizarán baterías de prueba sobre funcionalidades concretas del videojuego aisladas del resto de funcionalidades, con el fin de encontrar fallos concretos de implementación.
- **Pruebas alfa de sistema global:** Para esta última fase de pruebas alfa, se realizarán las pruebas sobre un entorno jugable con toda la funcionalidad de versión disponible. La intención es comprobar la consistencia de las funcionalidades expuestas en conjunto, para evitar que se produzcan condiciones no esperadas, a raíz de la interacción cruzada entre dichas funcionalidades.

Tras terminar con las pruebas alfa, se realizarán las pruebas beta. Estas serán cerradas, un grupo de personas limitado tendrá acceso a una versión jugable. La duración de cada prueba individual no superará la media hora. Una vez los “beta testers” hayan concluido con el periodo de prueba, se les realizará una encuesta, en la que tendrán que responder a preguntas relacionadas con la jugabilidad y aspecto visual del videojuego. La batería de preguntas se encuentra en el Apéndice D de esta documentación.

Aunque el objetivo de la fase de pruebas beta no es determinar fallos programáticos en la implementación, eso se determinará en las pruebas alfa, sí que se valorará si los testers han tenido algún fallo al jugar. No obstante, la finalidad de esta fase de pruebas es determinar entre otros, si la experiencia de juego es satisfactoria, problemas de jugabilidad, experiencia visual, inmersión y dificultad. Todas estas características serán las principales a evaluar, el análisis de las respuestas de la encuesta puede revelar carencias del juego que necesiten ser corregidas antes de su puesta en producción.

3.3. Estado del arte

3.3.1. Panorama actual de los videojuegos

El mundo del videojuego bien puede ser considerado como el 8º arte, no solo por la cantidad de ingresos que genera esta industria, sino por la cultura que se ha generado a raíz de este.

En la actualidad existen grandes producciones de videojuegos cuyo coste es superior al de grandes producciones cinematográficas. Hacer esta comparativa tiene sentido, dado que algunos videojuegos buscan generar las mismas sensaciones que una película (inmersión, narrado de historia, etc.), añadiéndoles un componente que genera nuevas experiencias, hablamos de la jugabilidad. Este nuevo componente permite que el usuario pueda interactuar dentro de la historia que se narra aportando sensaciones de inmersión aumentadas, y generando un vínculo con nuestro personaje.

Este tipo de cultura del juego es común en un tipo concreto de juego, el juego de aventura gráfica [21]. Sin embargo, este tipo de juegos no son los más explotados por la comunidad, siendo los juegos orientados al multijugador online los más explotados.

Si nos centramos en estos últimos, cabe destacar que el éxito que están teniendo se ve reflejado en la cantidad de nuevos juegos de este género que se han producido en los últimos 10 años. Todo motivado por un auge de los deportes electrónicos (E-Sports) en los que se explota este tipo de género y el afán de competitividad que generan.

Según datos de Newzoo, una consultora dedicada al mercado de los videojuegos, actualmente se estima que haya más de 1,200 millones de videojugadores en el mundo [22]. Si nos centramos en España, según un estudio realizado por el portal de anuncios clasificados “Clasf España” en 2016, se estima que hay cerca de 14,7 millones de jugadores, cerca del 40% de la población. Los cuales pasan una media de 5,8 horas semanales. Se trata de un sector juvenil en su gran mayoría, pero sorprende el aumento de jugadores con edades comprendidas entre los 35-44 años que han tenido. Es explicable si tenemos en cuenta que estos jugadores de media edad pueden considerarse como la primera generación de videojugadores.

Cabe destacar el aumento del número de videojugadoras en 2019, un 19,3% más de mujeres con respecto a 2017. Llegando a la totalidad del 43% de todos los jugadores según el estudio "Descubriendo los Esports en España" de WINK TTD, agencia de transformación digital del grupo Dentsu Aegis Network [23].

3.3.2. Videojuegos más vendidos.

En la tabla 2 se presenta una lista de los juegos más vendidos de todos los tiempos. Esto nos ayudará a saber que juegos tienden a perdurar más a lo largo del tiempo, en contraposición a los más eventuales que analizaremos más adelante.

TITULO	VENTAS	PLATAFORMA	LANZAMIENTO	DESARROLLADOR
Minecraft	176 000 000	Multiplataforma	2011	Mojang
Tetris	170 000 000	Multiplataforma	1984	Elektronorgtechnica
Grand Theft Auto V	115 000 000	Multiplataforma	2013	Rockstar North
Wii Sports	82 870 000	Wii	2006	Nintendo EAD
PlayerUnknown's Battlegrounds	50 000 000	Multiplataforma	2017	PUBG Corporation

Tabla 2 videojuegos con más copias vendidas

De la tabla anterior podemos ver una clara relación entre el número de ventas y la plataforma para la que se distribuye el juego. Es lógico pensar que los juegos que se distribuyen para múltiples plataformas tienen un mayor número de ventas, dado que aplican a todos los usuarios independientemente del hardware que tengan, llegando a todos los públicos.

Cabe también destacar, que la irrupción en el mercado de los juegos con el nuevo subgénero "battle royale" [24], está siendo muy significativa. En este género de videojuego, subgénero de los FPS, los jugadores aterrizan en un mismo escenario, mediante una mecánica, el escenario se irá reduciendo, actuando como descalificador para aquellos jugadores que no estén dentro de un área. La meta de cada partida reside en ser el último jugador vivo. El videojuego PlayerUnknown's Battlegrounds es uno de los primeros de este género, y a pesar de llevar poco tiempo en el mercado ya está dentro del top de videojuegos más vendidos. A pesar de esto y debido a la competencia de otros videojuegos del mismo género en la actualidad no es el más jugado.

Hay que anotar que estos datos, aunque pueden proporcionarnos una información relativa a la comercialización de los videojuegos muy importante, no nos aportan demasiada información de cara al uso y progresión del videojuego a lo largo de los años. Esto es debido a que en la anterior tabla solo se tienen en cuenta los juegos más comprados, excluyendo por razones obvias, a todos los videojuegos "free2play", es decir, videojuegos que son gratuitos para el público y cuya fuente de ingresos se basa principalmente, y en la actualidad, en lo que denominamos sistema de microtransacciones.

Este sistema se basa en paquetes de expansión, paquetes visuales, DLC's. En estos el usuario no paga por el juego completo, se ofrece una funcionalidad particionada, la base del juego es gratis, pero hay contenido bloqueado que se tiene que comprar para jugarlo.

Como ejemplo, podemos poner el caso de un juego que creó mucha controversia entre la comunidad de jugadores, debido al sistema de monetización que se aplicó al juego: Star wars Battlefront II; es un juego que generó muchas expectativas antes de su lanzamiento, entre la comunidad, gran parte de estas expectativas originadas por la buena acogida que tuvo la primera entrega de este videojuego (Star Wars: Battlefront). Se trata de un juego multijugador "shooter" en tercera y primera persona. El objetivo principal es combatir contra otros jugadores en un mundo basado en el éxito del cine, Star Wars. La crítica fue muy dura con este juego debido al sistema de monetización, se aplicó un subtipo concreto de micropago, denominado "loot box". En éste, el jugador compraba la versión base del juego, pero no desbloqueaba todo el contenido, es más en este juego en concreto se desbloqueaba un contenido que la comunidad consideró muy escaso. El resto de contenido se desbloqueaba mediante un sistema de pequeños pagos que daban acceso a unas "cajas virtuales" que contenían nuevos personajes jugables, complementos estéticos, etcétera. El contenido concreto de cada caja es pseudoaleatorio. El principal problema no fue tanto el sistema de monetización en sí, que ya había generado enormes beneficios en otros videojuegos como "Counter-Strike: Global Offensive" o el mismo PlayerUnknown's Battlegrounds que mencionábamos antes, sino que estos desbloqueables aportaban una ventaja para el jugador que los tuviera, generando un sistema en el que los jugadores que pagaran grandes cantidades de dinero, tenían una ventaja muy notable con respecto al resto de jugadores. Finalmente, EA empresa desarrolladora y comercializadora del videojuego, retiro temporalmente las microtransacciones por la cantidad de críticas que estaban recibiendo desde la comunidad.

El ejemplo anterior es una clara señal del poder de persuasión y sentido crítico que tiene la comunidad de videojugadores, pudiendo hacer cambiar el sistema de monetización de un videojuego, una vez este haya sido lanzado.

3.3.3. Videojuegos más jugados

En esta sección analizaremos los videojuegos más jugados de los últimos años, centrados en los videojuegos para la plataforma PC, de acuerdo con un estudio realizado por la compañía NZ, encargada de realizar estudios de mercado de los videojuegos [25]. La muestra utilizada para el estudio es de 12 millones de usuarios distribuidos en 42 países de la plataforma Overwolf's. También se analizará el perfil de jugador que más encaja con cada juego.

1 League of Legends (LOL) / Riot Games

Se trata del juego más popular del mundo, no solo por el número de jugadores que juega, sino por la cantidad de eventos y el dinero que genera este título en la industria de los deportes electrónicos. El juego entra dentro de un género denominado MOBA (Multiplayer Online Battle Arena) [26]. En este, dos equipos de cinco jugadores, se enfrentan. El objetivo principal es destruir el nexo del equipo enemigo que se encuentra en la base de cada equipo, para cumplir el objetivo los jugadores intentan cooperar y adaptarse para mantener el control estratégico del mapa para atacar y defender.

Este videojuego está orientado a jugadores con perfil asesino. Se busca fomentar la competitividad entre sus jugadores. Esto se consigue con un sistema de juego basado en batallas individuales.

League of Legends es jugado por 3.140.400 de jugadores.

Figura 6: League of legends

2 Hearthstone: Heroes of Warcraft / Blizzard Entertainment

Hearthstone es un juego multijugador de cartas. En éste, dos jugadores compiten sobre un tablero virtual en el que posicionan las cartas, el sistema de juego es similar al de otros juegos de mesa como "Magic The Gathering" o "Yu Gi Oh" pero este está adaptado para jugar en PC o móvil. Pese a que los otros similares anteriormente mencionados también disponen de plataformas virtuales para sus juegos, estos no han tenido tanto auge. El éxito de Hearthstone supera las expectativas, nadie se esperaba que un juego de este estilo, que ya contaba con un competidor "offline" con mucho público, como Magic, pudiera tener una comunidad de jugadores tan grande. La razón de este éxito, es en parte, por la sencillez de las mecánicas que definen las reglas del juego y la atracción del público apasionado del mundo de "World of Warcraft" (videojuego MMORPG de 2004, que hoy en día sigue generando beneficios mediante sus actualizaciones de contenido). Actualmente, Hearthstone, es el juego más popular de Blizzard.

Este juego, a pesar de ser de cartas, también busca fomentar la competitividad entre sus jugadores, por lo que el perfil de sus jugadores también es asesino.

Hearthstone es jugado por 2.664.000 de jugadores.

Figura 7: Hearthstone

3 Minecraft - Mojang / Microsoft

Ya mencionamos este juego en la sección de videojuegos más vendidos. Este juego, desarrollado en java, aparentemente puede parecer simple, sin énfasis en la calidad gráfica, pero la verdad es que es un juego cuyo desarrollo ha supuesto todo un reto en cuando a algoritmia y programación.

Figura 8: Minecraft

Se trata de un juego de mundo abierto, que usando como unidad básica el bloque (cubo), puedes ir generando estructuras mediante herramientas con el fin de explorar, crear y sobrevivir.

El éxito de este juego reside en la sencillez de sus mecánicas, junto con la complejidad de dinámicas generadas.

Mediante estas mecánicas, se busca complacer al perfil de jugador, explorador y triunfador. El hecho de que el mundo se vaya generando proceduralmente alienta a los jugadores con estos perfiles.

Minecraft es jugado por 2.278.800 de jugadores.

4 Counter Strike: Global Offensive (CSGO) / Valve

Se trata de un juego FPS, evolución del clásico de PC Counter-Strike, desarrollado por Valve en 1999. Nació como una modificación del juego Half-Life del mismo desarrollador. Es uno de los primeros videojuegos en entrar en los E-Sports. Dos equipos de 5 jugadores cada uno combaten en diferentes escenarios por rondas. Cuando termina la ronda cada jugador puede comprar nuevas armas en función del dinero ganado en la ronda anterior y el acumulado.

Los jugadores asesinos son el público preferido, y casi exclusivo de este videojuego. Igual que en el videojuego League of legends, los jugadores solo tienen como objetivo la victoria, para progresar en un ranking global de jugadores.

CSGO es jugado por 1.973.990 de jugadores.

Figura 9: Counter-Strike Global Offensive

5 PlayerUnknow´s Battlegrounds (PUBG) / BlueHole

Ya mencionamos el éxito que fue en ventas, y el motivo, ha superado el millón de jugadores simultáneos. PlayerUnkown´s Battlegrounds, pertenece al género de los action shooters, es un juego que nos hace vivir una auténtica Battle Royale en su extenso mapa, con una capacidad de hasta 100 jugadores por partida.

En PUBG tenemos total libertad para movernos por el mapa del juego, el factor de la suerte también juega un papel importante a la hora de encontrar el equipamiento y armas necesarias para cumplir nuestro objetivo, sobrevivir y ser el último jugador en pie.

Aunque este juego podría parecer ser exclusivo del rol asesino, lo cierto es que también busca, mediante la mecánica del saqueo de equipo, incentivar a los jugadores exploradores.

PUBG es jugado por 1.296.000 de jugadores.

Figura 10: PlayerUnknown's Battlegrounds

Aunque estos son los juegos más jugados según este estudio que data del 2018, el mercado de los juegos cambia año a año, y no podemos terminar esta sección sin mencionar otro videojuego como Fornite, que actualmente tiene picos de jugadores concurrentes de hasta 10,8 millones de jugadores según Tim Sweeney, CEO de Epic Games [27].

El gran éxito de este videojuego se debe a la absorción de una gran cantidad de jugadores de PUBG, mezclado con una mecánica de juego basado en la construcción defensiva y ofensiva de estructuras, algo bastante innovador que no se había visto hasta el momento en ningún juego del genero battle royale. Además, este juego cuenta con una de las tasas de visualización más altas por parte de los espectadores de plataformas de streaming en directo como Twitch.

Podemos darnos cuenta, analizando este top 5, que la mayoría de videojugadores de la actualidad se clasifican dentro de los perfiles asesino y explorador. Siendo el primero el más predominante de los dos.

Capítulo IV

Videojuegos en realidad virtual

4.1. Contexto

El mercado de los videojuegos en realidad virtual está empezando a crecer, si bien no lo está haciendo al ritmo que se esperaba, sigue siendo un sector que avanza tecnológicamente muy rápido, y es un mercado por el que los fabricantes están apostando muy fuerte, creando su propio hardware, como el caso de PlayStation con PS VR.

El principal motivo por el que no se está llegando a las cifras esperadas, radica en la necesidad de un Hardware específico para poder jugar a estos juegos, que además no es en la actualidad, barato. A pesar de esto, el crecimiento económico de este sector está en auge, gracias a las múltiples aplicaciones que la realidad virtual ofrece.

De todas las aplicaciones de la VR en concreto es el sector del entretenimiento y los videojuegos donde el crecimiento está siendo más pronunciado.

4.2. Videojuegos más jugados.

Dentro de los videojuegos hay varios títulos exclusivos de VR que están teniendo gran impacto entre la comunidad “Gamer” [28]. Entre ellos se encuentran los siguientes:

- **Beat Saber**

  Aprovechando el tirón producido por la industria cinematográfica con el lanzamiento de la nueva saga del universo Star Wars, se lanza este juego, que si bien no tiene mucho que ver con Star Wars, utiliza como gancho una de sus icónicas armas para utilizar en un juego musical de coordinación.

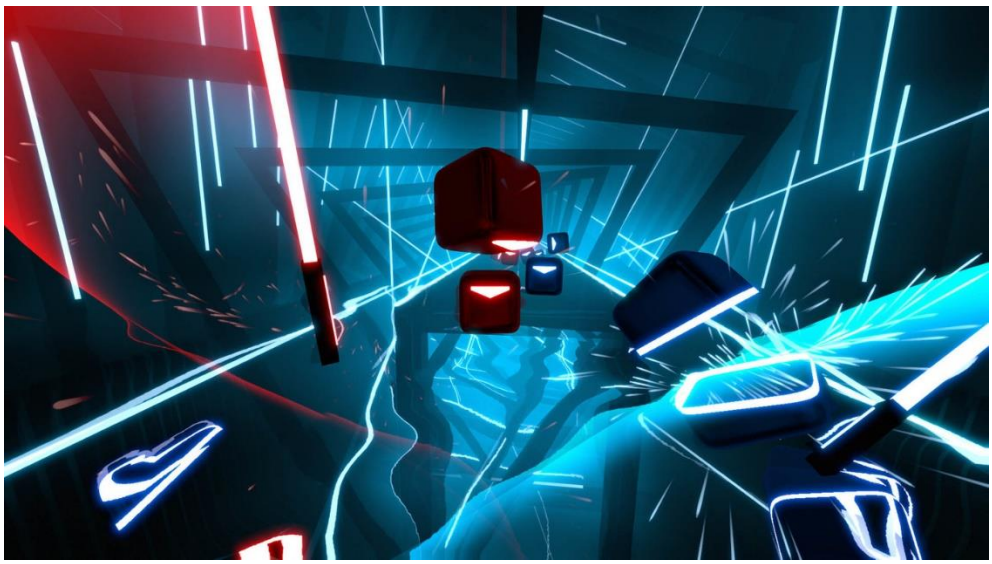


Figura 11: Beat Saber VR

Al ritmo de una música el jugador tiene que ir rompiendo unos bloques que se acercan al punto de vista del jugador, mientras esquiva otros elementos con dos sables laser. Aunque parezca un detalle sin importancia, debido a que no es el primer juego de coordinación musical, tampoco el primero en VR, ni con este estilo de mecánicas, el detalle de añadir a un juego de realidad virtual un elemento de una saga tan exitosa como Star Wars, es uno de los puntos fuertes que más ha atraído a la comunidad.

- **Superhot VR**

  En este juego el jugador se irá enfrentando a puzles de acción en primera persona con varios objetivos que abatir, la acción transcurre en lo que podría definirse como un sistema por turnos. El juego es extremadamente innovador al mezclar dos mecánicas de juego tan diferentes. La precisión de los controladores para detectar el movimiento de las manos, proporciona un gran sentido de inmersión dentro del juego, convirtiendo cada puzle en situaciones con muchas posibles soluciones. Aunque este juego no se centre en el hiperrealismo visual, cabe destacar que las texturas poligonales, y los colores de alto contraste como el blanco y el rojo, unido a unas animaciones muy bien construidas, generan un grado de satisfacción muy alto.

Figura 12: SuperHot VR

- **Elder scrolls V: Skyrim VR**

  Como otros muchos juegos que han triunfado en la comunidad “gamer”, Elder scrolls V, o como se le conoce dentro de la comunidad: Skyrim, ha creado una versión adaptada a la realidad virtual de mano de la empresa desarrolladora, Bethesda, y nuevamente en esta categoría vuelve a estar en el top de ventas y es uno de los juegos más jugados de realidad virtual, El éxito en este juego radica en la jugabilidad FPS, unido a una lógica de decisiones tan compleja como el videojuego original.

Figura 13: Skyrim VR

Matar dragones se convierte en una tarea divertida, cargada de emoción e inmersiva gracias a la gran capacidad de adaptación de este juego al mundo VR. En el videojuego para las plataformas convencionales el modo primera persona, nos mostraba nuestras manos en una posición fija, estas podían cargar armas, escudos, arcos, o conjurar hechizos y proyectarlos en la dirección marcada por el ratón. Este sistema de jugabilidad y visualización hacen que este juego sea un candidato óptimo a una adaptación a VR, dado que solo tenemos que substituir cada mano con un controlador de RV y disfrutar del movimiento libre de estos.

- **Minecraft VR**

  Otro juego que ya vimos en la sección videojuegos más jugados, que dispone de una versión en realidad virtual, es Minecraft.

  La mecánica del juego es exactamente la misma que en el juego para PC y consolas, pero en este caso se intenta explotar una estética a mayores. Hablamos de la inmersión, ésta mediante un sistema de mundo abierto y creado proceduralmente, lleva a nuevos niveles de satisfacción a los jugadores con perfil explorador, sin embargo, dado que el resto de mecánicas es igual, y el sistema de interfaz se adapta mediante canvas anclados a los controladores, el acceso a los menús y gran parte de la lógica del juego empeoran bastante en relación a los videojuegos de las plataformas clásicas.

Como hemos podido observar el mercado de los videojuegos VR, está bastante copado de adaptaciones de videojuegos que ya han tenido una repercusión en el mercado tradicional. Sin embargo, hay que destacar especialmente los juegos con unas mecánicas bastante originales que son perfectos para su uso en realidad virtual, y que ofrecen una experiencia de usuario muy placentera. Algunos de estos juegos son los anteriormente mencionados Super hot VR y Beat Sabre. Otros que, aunque suelen ser adaptaciones y sus mecánicas son muy conocidas, ofrecen una experiencia única en plataformas de realidad virtual. Estos son los videojuegos de simulación de conducción, como Assetto Corsa, o Project CARS 1 y 2. Y también los

videojuegos de simulación de pilotaje como Ace Combat 7 (Figura 14): Skies Unknown y Star Wars Battlefront Rogue One: Misión RV.

Figura 14: Ace Combat 7

4.3. Videojuegos similares

Debido a la mezcla de dos estilos de juego tan diferentes, existen pocos videojuegos en el mercado conocido con características similares al que estamos desarrollando. Hablaremos de ellos más adelante.

Encontrar un videojuego de estrategia que mezcle las dinámicas de FPS no es algo que se vea a menudo en plataformas tradicionales y mucho menos en plataformas de realidad virtual. No obstante, sí que hay un videojuego con características similares en cuanto a la opción de construcción de defensas mezclado con un componente shooter, este juego se llama XLR. Este juego está disponible en Steam VR, pero está todavía en fase beta y sus componentes se orientan más a un FPS con características diferentes a las de nuestra propuesta.

Si separamos el juego en sus dos géneros principales, sí que podemos encontrar una gran variedad de juegos que cuadran con uno de ellos. Centrándonos en las características FPS del videojuego basadas en defensa de posición, ejemplos de juegos con características similares a éstas son:

- **Tower's Guard:**

  Este juego tiene un diseño artístico de tipo cartoon [29], el jugador se haya en el interior de una pequeña torre y debe defenderla de oleadas de criaturas que intentarán asaltarla. El jugador dispone de arco y flechas, trampas y bombas para evitar que los enemigos asalten la torre. [30]

- **Medieval Archer**

  Dispone de un modo de juego que se basa en los tower defense, este juego se utiliza el sistema de arco y flechas clásico que tiene SteamVR y la base que utilizamos en nuestro videojuego. [31]

Si nos centramos en la dinámica estratégica, posicionamiento de defensas, control de la economía, en esencia los elementos clásicos del tipo de videojuego Tower Defense, tenemos una variedad de títulos como los siguientes:

- **Defense Grid 2: Enhanced VR Edition:**

  Mediante un sistema basado en misiones podemos ir defendiendo una estructura en varios modos de juego, en todos ellos la gestión económica y posicionamiento de defensas son las características fundamentales igual que un tower defense clásico. Este videojuego es una versión adaptada para VR de releases anteriores que usan el mismo sistema de mecánicas, la realidad virtual aporta un sistema de visionado en un entorno 3D muy inmersivo. [32]

- **The Last Day Defense:**

  Basado en un mundo de conflicto permanente entre dos imperios galácticos, the last day defense proporciona todas las mecánicas clásicas de los tower defense, en un sistema cerrado. [33]

- **Castle Defense:**

  Este juego puede que sea el que más contenido típico del genero tower defense tenga de los anteriores listados. El objetivo, posicionar una serie de torres en un mapa por el que discurren unas rutas de paso de enemigos. El juego se desarrolló con fines educativos, y también está hecho en Unity 3D pero a diferencia de nuestro juego, este está diseñado para la plataforma de VR Oculus. [34]

Videojuegos cuyos diseñadores buscaron las mismas ideas innovadoras de mezcla de mecánicas entre géneros que se buscan en nuestro videojuego. Habiendo diferencias claras entre estos y el videojuego expuesto, casi todas estas diferencias se basan más en la estética y el diseño gráfico que en la dinámica o mecánica. Ejemplos de estos videojuegos son los siguientes:

- **SurviVR - Castle Defender:**

  Este juego en concreto se basa en un sistema de defensa de oleadas. Basado en un mundo medieval con un castillo deberemos evitar que estas oleadas nos maten. Para ello dispondremos de un variado arsenal de ataque directo y magias, además de un sistema estratégico de posición de defensas, que no está demasiado explotado. Debido a que el objetivo de las oleadas es el propio jugador, la mecánica estratégica de defensa juega un papel más secundario dado que no nos podremos defender exclusivamente con este sistema, y deberemos recurrir a las mecánicas FPS para ello.

  El juego cuenta con un apartado artístico basado en modelos de baja poligonización con escenarios creados virtualmente, que lo alejan mucho de la inmersión ofrecida por los entornos hiperrealistas, aunque busque una aestética inmersiva realista. [35]

Figura 15: SurviVR

- **Wrack: Exoverse**

  De manera similar al videojuego comentado anteriormente, Wrack combina las mecánicas de FPS, con las propias de un tower defense, aunque con algunas modificaciones de estas últimas. El sistema de spawn de enemigos también se basa en las oleadas, que convergen en un punto central del mapa a través de varias rutas que el jugador deberá defender. Sin embargo, Los objetivos no solo irán hacia el punto de convergencia si no que podrán atacar al jugador. Debido a esta combinación de posibilidades de perder la partida, se realiza un equilibrio entre mecánicas mucho mayor que en el caso de SurviVR, aunque nuevamente, las mecánicas FPS son prioritarias.

Figura 16: Wrack Exoverse

En cuanto al apartado artístico, es un juego que utiliza el estilo Comic para el diseño e iluminación de sus modelos. Es algo que no se suele hacer en la industria y en plataformas VR es mucho más difícil de encontrar, pero debido al alto coste computacional que conlleva en una plataforma VR, utilizar modelos hiperrealistas, es una solución muy elegante y que ofrece un alto grado de satisfacción visual. [36]

Capítulo V

Análisis de tecnologías

Para la implementación de un videojuego, que como hemos visto en el capítulo anterior, sea innovador respecto al género del mismo, necesitaremos componentes especializados para el diseño y desarrollo del videojuego.

En este capítulo analizaremos las herramientas de las que disponemos para la realización de esta tarea y las compararemos, a fin de poder seleccionar la herramienta adecuada en función de las variables.

Separaremos las herramientas a comparar, por tipo de funcionalidad, en los siguientes grupos: Motores de videojuego y entornos de desarrollo integrado. La más importante y que desarrollaremos en mayor profundidad en futuros capítulos, es el motor del videojuego elegido.

5.1. Motores de videojuego

Un motor de videojuego, del inglés game engine [37], es una herramienta computacional, que proporciona una serie de procedimientos algorítmicos que permiten el diseño, construcción y funcionamiento de videojuegos.

Normalmente el motor de un videojuego, dispone de tres componentes principales:

- **Motor Gráfico:** Se encarga de realizar el renderizado [38] de los gráficos ya sean en 2 dimensiones o en 3 dimensiones. Esto es la construcción de todos los componentes que definen una representación visual: forma, textura, iluminación.

- **Motor Físico:** El objetivo de este componente es de cálculo e integración. Su función es la de aplicar todos los cálculos físicos que se precisen a los objetos que tengamos en la escena de nuestro videojuego. Entre estos cálculos se hayan, la detección de colisiones, simulación de las leyes físicas, cálculo mecánico y dinámico, etcétera.

- **Motor de Scripting:** Para que podamos generar nuestras mecánicas, es necesario un sistema que interprete un lenguaje de programación para poder definir las reglas y relaciones que haya entre nuestros objetos. Este es el objetivo del motor de scripting. Normalmente los motores gráficos poseen varios de estos motores para aumentar la flexibilidad de lenguajes de programación soportados por el motor de videojuego. Los lenguajes más usados son los siguientes: C++, C#, C, javascript, Java, LUA.
  Algunos motores de videojuego proporcionan herramientas de generación automática de código, esto es el caso de Unreal Engine, que proporciona la herramienta Blueprints [39], una herramienta para diseñar algoritmos de manera visual para que luego se pueda generar código en C++ a partir de estos diseños.

Aparte de estos componentes, la mayoría de motores de videojuego disponen de otros sistemas que proporcionan herramientas útiles para nuestro videojuego, sistemas de animación, gestión de relaciones, gestión de sonido, inteligencia artificial, etcétera.

5.1.1. Unity

Unity es un motor de videojuegos creado por la compañía Unity Technologies, lanzado en 2005. Con unity se pueden crear videojuegos multiplataforma, entre las que se incluyen: WebGL para su uso en plataformas web, Windows, OS X, SteamOS y GNU/Linux para las plataformas pc, iOS, Android y Windows Phone para las plataformas móviles, Playstation 4, PS vita, Xbox One, Xbox 360 Wii U, Nintendo Switch y Nintendo 3DS para videoconsolas y Oculus Rift, Google Cardboard, HTC Vive, PS RV, y Microsoft Hololens para plataformas de realidad virtual.

La última versión disponible para su descarga desde la página oficial es Unity 2019.3 [40].

El motor de scripting del que dispone Unity puede interpretar código escrito en los lenguajes C# y javascript.

Otra ventaja que dispone este motor, es la disponibilidad de una gran variedad de contenido gratuito a través de su tienda oficial, que simplifican el proceso de creación de videojuegos, en caso de que el desarrollador o desarrolladores deseen centrarse en las mecánicas y en la lógica del videojuego.

Unity dispone de diferentes licencias separados por dos categorías principales, Individual y Business. Mientras que la categoría individual se limita a su uso académico, la categoría Business está enfocada a estudios de desarrollo de videojuegos profesionales.

Dentro de la categoría Individual disponemos de dos planes diferentes:

- **Personal**:

  Esta versión es totalmente gratuita pero limita su uso en función de las ganancias obtenidas con los productos creados a partir de esta licencia, en concreto no puedes obtener más de 100000 $ a lo largo de un año. A cambio se proporciona la última versión del core de Unity y un pack de recursos para iniciarte en el uso de la herramienta. Este pack contiene documentación, algunas herramientas y la posibilidad de descargar algunos assets de manera gratuita desde la tienda oficial de Unity.

- **Learn Premium:**

  En este caso, la licencia cuesta 15 $ mensuales, se trata de una licencia académica ampliada, incluye sesiones en directo con instructores de certificados de Unity, contenido bajo demanda actualizado, y contenido específico para cada fase del aprendizaje.

Dentro de la categoría Business disponemos de 3 planes diferentes:

- **Plus:**

  A cambio de una subscripción mensual de 40 $, con esta licencia se puede acceder a todos los recursos de la categoría “individual” además de cursos especializados más avanzados, estilos para la interfaz y herramientas para creadores más complejas, entre las cuales se incluyen un sistema de métricas de rendimiento en vivo y sistemas de diagnóstico en cloud. Las ganancias por los productos creados bajo este plan, no pueden superar los 200000 $ a lo largo del año.

- **Pro:**

  Este es el plan, según la propia página de Unity, que más se vende, con un coste de suscripción de 150 $ mensuales. La elección de este plan es obligatoria si las ganancias son superiores a los 200000 $. A parte de contener todo lo que contiene el plan “plus”, dispone de

herramientas de gestión de código en equipos específicas para entornos Unity, totalmente integradas con el motor de videojuego. También se añade un servicio de atención al cliente especializado para este plan, incluido la posibilidad de acceso al código de las herramientas de Unity.

- **Enterprise:**

  Este plan es más personalizable y ofrece soluciones concretas para situaciones concretas de estudios de desarrollo, lo que permite elegir las herramientas y componentes que se incluyen en el plan de una manera más flexible con un coste también flexible.

5.1.2. Unreal Engine

El motor de videojuego Unreal engine, al igual que Unity, también permite crear videojuegos multiplataforma. Creado en 1998 por la empresa Epic Games, es uno de los motores de videojuego más utilizados en la actualidad.

Una de las principales ventajas de este motor y que lo convierte en la opción favorita de la mayoría de desarrolladores, es que su código es totalmente libre y accesible por cualquier persona vía Github.

Para desarrollar videojuegos en este motor necesitamos hacer uso del lenguaje de programación C++, aunque dispone de un sistema para diseñar algoritmos de manera visual, llamado Blueprints. Esta funcionalidad permite a los desarrolladores generar esqueletos básicos en C++ para la gestión de la lógica del videojuego de manera más intuitiva y rápida, aunque obviamente está muy limitado, y puede que solo nos sirva como base para la creación de las mecánicas, y que tengamos que modificar el código generado en C++ para conseguir una funcionalidad acorde con el diseño del videojuego.

Dispone de dos tipos de licencia:

- **Publishing License:**

  No tiene ningún coste inicial, pero debes proporcionar el 5% de las ganancias con productos creados bajo licencia. Se trata de una versión orientada a fines académicos y estudios de desarrollo profesionales, con ingresos superiores a los 3000 $ por trimestre.

- **Creators License:**

  Esta licencia es totalmente gratuita y no tiene tasas adicionales en los beneficios obtenidos en los productos creados a partir de esta licencia. En cambio no se pueden obtener beneficios superiores a los 3000 $ trimestrales. Esta es la opción perfecta para pequeños estudios que se inician en el desarrollo de videojuegos y con fines plenamente académicos.

5.1.3. Source Engine

Este motor fue lanzado en 2004 y desarrollado por Valve Corporation, empresa administradora de la plataforma de distribución de videojuegos más reconocida a nivel mundial, Steam [41].

Su uso es gratuito, pero requiere de una cuenta en Steam, y las condiciones de uso son muy específicas y restrictivas en función del tipo de producto que vendes, por ejemplo, si quieres vender un videojuego creado con el motor source, deberás:

- Aceptar un acuerdo para la distribución de productos creados con Source
- Si utilizas algunas herramientas, como RAD, que están incluidas en el SDK de Source tendrás que ponerte en contacto con RAD para que te informen de los costes asociados por las licencias que utiliza.
- Havok cobra 25000 $ por el uso del motor de físicas que utilice cualquier juego de pago creado con Source.
- Salvo que adquieras una licencia completa del motor Source, los juegos que crees con este motor solo podrán ser distribuidos a través de Steam.

5.1.4. CryEngine

El motor de videojuego CryEngine, creado por la empresa CryTek, surgió originalmente como motor de demostración para la empresa Nvidia, con el videojuego de la misma empresa creadora del motor, Far cry.

En la actualidad todos los derechos de CryEngine pertenecen a la empresa desarrolladora y productora Ubisoft.

La última versión se denomina CryEngine V, soporta lenguajes de programación como C++, LUA y C#, y permite la creación de videojuegos para las principales plataformas, Windows, Linux, Xbox y PS4. Dispone también de un SDK para desarrollo de juegos de realidad virtual, OSVR [42].

Debido a que tiene una licencia propietaria, se restringe su uso bajo discreción de la empresa CryTek para toda monetización del producto.

5.1.5. GameMaker Studio

Esta plataforma está construida en el lenguaje de programación Delphi, y se basa en un lenguaje de programación interpretado y en un SDK para el desarrollo de videojuegos. Originalmente se creó por el profesor Mark Overmars para que desarrolladores con pocas nociones sobre programación, pudieran desarrollar videojuegos simples.

Hace uso de un lenguaje de programación de scripts llamado Game Maker Language (GML) [43], para ampliar la posibilidades de implementación lógica.

La interfaz principal para el desarrollo de videojuegos, no obstante, se basa en un sistema de arrastrar y soltar, para permitir crear el producto, simplemente organizando iconos en la pantalla.

Los usos prioritarios de esta herramienta, se centran principalmente en videojuegos en 2D.

Permite el desarrollo para plataformas móviles, pc y consolas.

La herramienta es totalmente gratuita, aunque posee una versión comercial con componentes adicionales.

5.2. Entornos de desarrollo integrado

Los entornos de desarrollo integrado, o por sus siglas en inglés IDE’s [44], son aplicaciones informáticas que proporcionan servicios integrales para facilitar el desarrollo software.

Un IDE, consiste en un editor de código fuente, herramientas de construcción y despliegue automático y un depurador de código, para corrección de errores. Existen otras funcionalidades que dependen del tipo de IDE, estas pueden ser soporte para pluggins, conexión a repositorios de código, autocorrección de errores sintácticos, etcétera.

En este proyecto la elección de un IDE apropiado es fundamental, dado que toda la lógica del videojuego se gestionará mediante scripts para determinar el comportamiento de los componentes del videojuego. La mayoría de motores de videojuegos, disponen de algún IDE especializado para el motor, estos pueden o no, estar integrados en el propio motor. Pero no es indispensable utilizarlos, dado que el código fuente se puede interpretar indistintamente del editor de código que se utilice, siempre que se utilice uno de los lenguajes que soporte el motor del videojuego.

5.2.1. MonoDevelop

Este entorno de desarrollo es libre y gratuito, está diseñado esencialmente para los lenguajes de programación .NET, entre los que se incluyen C#, Java, Python, Boo y Nemerle. Originalmente se desarrolló para satisfacer la necesidad de los desarrolladores .NET de trabajar bajo plataformas Linux, sin embargo, hoy en día dispone de un soporte completo multiplataforma.

El instalador del motor de videojuegos Unity permite instalar este software para utilizar como IDE integrado para Unity y que esté sea el entorno por defecto para los scripts que se creen en este motor.

5.2.2. Visual Studio Code

Originalmente esta herramienta era un editor de código fuente simple, sin embargo gracias a la capacidad de soportar pluggins, en la actualidad dispone de todas las capacidades de un IDE completo, con la posibilidad de compilar código en múltiples lenguajes si se instala el pluggin adecuado.

Otro de los aspectos destacados de este ide es el control integrado con repositorios como Git y la capacidad de personalización que tiene gracias al uso de las extensiones.

Para gestionar los complementos y extensiones dispone de un repositorio centralizado.

Este IDE es gratuito y de código abierto, pero la descarga oficial está bajo software propietario, requiriendo tus datos de uso del programa legalmente [45].

5.2.3. Microsoft Visual Studio

Este IDE es la elección por defecto para integrar, como editor de código fuente, del motor de videojuegos Unity.

Permite la programación en múltiples lenguajes de programación, principalmente C++, C#, Visual Basic y .NET, aunque también dispone de compiladores y Lints [46] para Java, Python, Ruby y PHP [47].

Dispone de capacidades de integración y desarrollo “en línea” mediante Windows Azure [48].

Creado por la empresa de software Microsoft, tiene una versión gratuita, Community, orientada a estudiantes y programadores aficionados y está disponible con la versión de Unity orientada a la educación.

También dispone de dos versiones comerciales, Professional y Enterprise. Las principales diferencias entre estas versiones son principalmente relativas a la integración y el desarrollo conjunto, añaden más herramientas de diagnóstico y depuración y mayores capacidades para su uso junto con Windows Azure.

5.3. Comparativa entre tecnologías y conclusión

Ya hemos visto las tecnologías que podemos utilizar para el desarrollo de este proyecto, ahora toca comparar las diferentes soluciones que hemos proporcionado y elegir las que cuadren mejor con nuestros requisitos.

5.3.1 Comparativa. Motores de videojuego

El motor del videojuego es la parte más importante y la herramienta principal de nuestro proyecto. No solo definirá el alcance del videojuego, si no que eligiendo el más adecuado nos va a permitir desarrollar nuestro videojuego de la manera más eficiente para cumplir con las expectativas.

En el apartado 4.1, hemos descrito un conjunto de motores de videojuego, los más importantes en el panorama actual del desarrollo de videojuegos. En la tabla 3 se describen las principales características de cada uno, para tener una visión simplificada de cuál es el mejor en nuestro caso.

Motor	**2D/3D/ RV**	**Motor físico**	**Documentación**	**Licencia**	**Coste**	**Lenguajes soportados**	**Relevancia**
Unity	Todos	Potente	Abundante	Privativa	Gratuito / Royality	C# Javascript	Alta
Unreal Engine	3D y VR	Potente	No muy abundante	Privativa	Gratuito / Royality	C++ UnrealScript	Alta
Source	Todos	Potente	Baja	Privativa	Gratuito / Royality	C++	Media
CryEngine	3D y VR	Potente	Baja	Privativa	Gratuito / Royality	C++	Media
GameMaker	2D	Limitado	Abundante	Privativa	Gratuito	GML	Baja

Tabla 3: Comparativa resumen entre motores de videojuegos

Los valores que se han tenido en cuanta para la comparación son los siguientes:

- **2D/3D/RV:** Se valorará solo aquellos motores que dispongan al menos de soporte para 3D y RV, dado que nuestro videojuego necesita de estas características.

- **Motor físico:** En nuestro videojuego el cálculo de físicas es muy importante para conseguir un efecto satisfactorio en el manejo del arma principal, el arco. Pero también es necesario un motor rápido y eficiente para el cálculo de todas las colisiones y efectos gravitatorios sobre los proyectiles. Contaremos solo con motores que posean sistemas potentes de cálculo de físicas.

- **Documentación:** La documentación es una parte muy importante, esta nos permitirá implementar de manera más eficiente todas las funcionalidades del juego, si disponemos de una amplia documentación para guiarnos nos resultará más fácil evitar los riesgos asociados al desarrollo del videojuego y buscar las mejores soluciones para cumplir con los requisitos.

- **Licencia y coste:** Prácticamente todos los motores de videojuegos poseen licencias privativas para su uso, pero la mayoría disponen también de versiones educativas o para proyectos no lucrativos, y son estas versiones que sean gratuitas, las que utilizaremos.

- **Lenguajes soportados:** La mayoría de los motores funcionan bajo C++, que es un lenguaje orientado a objetos, eficiente y consolidado en la comunidad de desarrollo, sin embargo la curva de aprendizaje de este lenguaje es muy elevada al principio, y hay opciones que son también muy potentes, pero que tienen una sintaxis y principios más sencillos, como C#.

- **Relevancia:** Este atributo hace referencia a la cantidad de desarrolladores y juegos serios que se realizan bajo un motor. Por razones obvias si la comunidad de desarrolladores utilizan con mayor frecuencia un motor en concreto, este estará más consolidado, y que se realicen juegos importantes bajo este motor, es una prueba de campo perfecta para verificar la potencia del motor en cuanto a la capacidad de crear videojuegos importantes.

Teniendo en cuenta estos valores, podemos ir descartando motores por no cumplir con los requisitos previos como GameMaker, Source y CryEngine.

De los dos motores que nos quedan, tanto Unreal Engine como Unity son soluciones muy válidas para el desarrollo del videojuego, sin embargo, hay dos características que nos harán decantarnos por Unity, estas son: La abundante y buena documentación que dispone, y que se pueda desarrollar en un lenguaje más cómodo como C#.

Es cierto que Unreal Engine es un motor que proporciona un mayor grado de potencia de renderizado creando escenarios hiperrealistas muy inmersivos, pero no existe una diferencia lo suficientemente grande que compense el coste de aprendizaje del lenguaje y la escasa documentación para desarrolladores.

5.3.2. Comparativa. Entornos de desarrollo integrado

La elección de un IDE es una cuestión poco objetiva que depende mucho del gusto de cada desarrollador, esto es debido a la cantidad de entornos que ofrecen las mismas, o muy parecidas funcionalidades, para un conjunto de lenguajes de programación.

Para nuestro proyecto utilizaremos el entorno de desarrollo que dispone Microsoft en su versión community, Visual studio.

Aunque a primera vista, lo mejor sería utilizar un IDE que ofrezca más flexibilidad, como Visual Code, o que sea totalmente gratuito y libre de uso como MonoDevelop, nos decantamos por Visual

Studio por la cómoda integración que dispone con el motor de videojuegos Unity, su potente depurador de código para C# y la facilidad de desarrollo que ofrece, integrado con Unity, para su asistente de código, refactorización, generación de código y análisis, todo ello mediante los componentes intelliSense y CodeLens.

En la actualidad, los instructores oficiales de Unity, recomiendan el uso de Visual Studio como IDE para los proyectos de desarrollo de videojuegos, haciendo hincapié en la completa integración del depurador de Unity y el propio de Visual Studio.

Capítulo VI

Unity Conceptos básicos

A lo largo del capítulo anterior analizamos las diferentes tecnologías que hemos considerado para el desarrollo de este proyecto. Tras realizar una comparativa de estas, decidimos que para el desarrollo de nuestro juego, el uso del motor de videojuegos Unity era la mejor opción, y que además, dispone por defecto de un IDE como visual Studio que es la mejor opción para la realización del código fuente y la lógica de nuestro videojuego.

En este capítulo vamos a explicar los fundamentos de este motor en concreto, pero solo vamos a ver los elementos de funcionamiento que influyan directamente sobre la ingeniería del software, dado que esto nos ayudará a comprender como funciona la herramienta de cara a modelar correctamente la lógica del proyecto.

6.1. Introducción. Qué es Unity

Como ya hemos mencionado anteriormente, Unity es un motor de videojuegos creado por la empresa Unity Technologies. El objetivo principal de esta herramienta es mantener un grado de abstracción que facilite el desarrollo de un videojuego. De esta forma los desarrolladores solo tienen que preocuparse de saber manejar esta herramienta para implementar la lógica, visualización y modelo del videojuego, y así poder generar los 3 componentes presentes en el modelo MDA.

Este motor dispone de un conjunto de rutinas, funciones de software que permiten la implementación, gestión y control de los principales elementos de un videojuego como las físicas, iluminación, renderizado.

Al mismo tiempo nos proporciona un lenguaje interpretable por el motor de scripting de la herramienta, con el que podemos generar la lógica y las relaciones entre los objetos de nuestro videojuego. Este lenguaje puede ser C# o Javascript. Nosotros utilizaremos únicamente C#.

La interfaz de la herramienta es muy intuitiva y dispone de un sistema de personalización de vistas basado en ventanas que podemos mostrar/ocultar, desplazar y redimensionar a nuestro antojo.

Existe una documentación oficial muy extensa y completa de todas las posibilidades de las que dispone la interfaz de Unity [49].

6.2. Jerarquía de clases

La librería principal en la que se basa todo el modelo orientado a objetos que podemos usar en Unity, se llama UnityEngine. Esta librería, contiene la clase principal de organización, y de la que heredan el resto de clases que necesitaremos para definir el modelo de un videojuego, Object. Al igual que cualquier lenguaje de programación orientado a objetos, esta clase object, ofrece un marco de referencia para el resto de clases, y se trata de la unidad organizativa de mayor nivel dentro de esta estructura.

En unity sin embargo no vamos a trabajar directamente con la clase Object, sino que, la unidad de referencia que vamos a usar, y que hereda directamente de Object, es la clase GameObject.

Un GameObject es cualquier elemento que se encuentre en una escena de Unity, y por lo tanto supone la unidad de representación básica de cualquier elemento dentro de un videojuego.

Los GameObject tienen asociados a ellos, mediante una agregación, la clase component, con una multiplicidad “uno a varias”, lo que significa que podemos tener uno o más componentes dentro de un GameObject.

Hay un componente, que siempre es del mismo tipo, y que todo GameObject tiene, se llama transform, y define un sistema de representación en coordenadas cartesianas para ubicar a un objeto del juego. Este componente define la posición en las tres componentes de un sistema cartesiano, la rotación en los 3 ejes coordenados y su tamaño, también basado en proyecciones sobre ejes cartesianos.

La Figura 17 muestra un diagrama de clases con las principales relaciones existentes entre las principales clases que conforman el modelo de Unity.

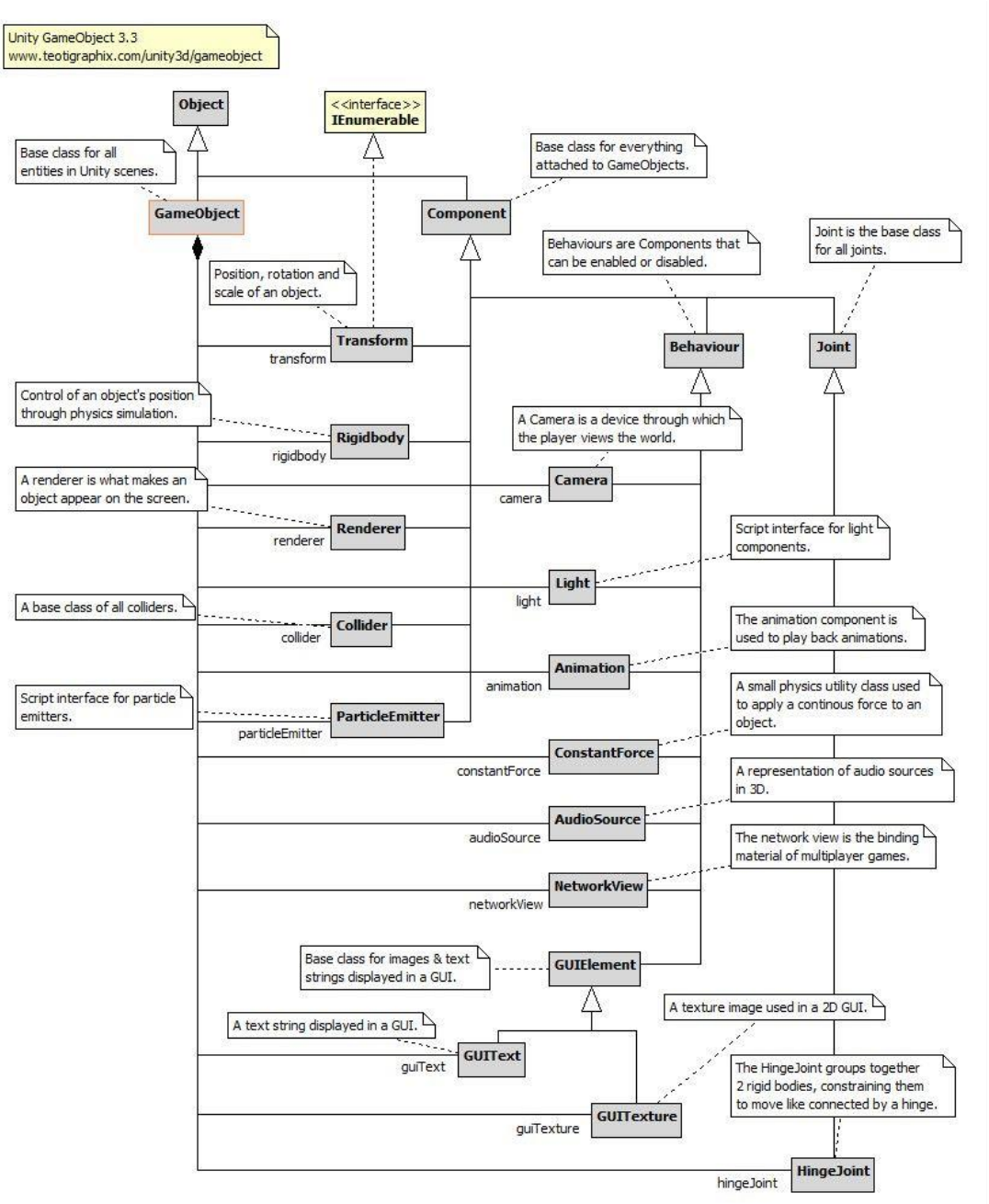


Figura 17: Jerarquía de clases Unity

Hay algunos objetos especiales que ya vienen prediseñados con una serie de componentes como son las cámaras, que sirven para renderizar todos los objetos de la escena, o el objeto light, que puede ser de diferentes tipos, en función del tipo de iluminación que se desea aplicar a la escena.

Un tipo especial de componente que es vital, es el componente script. Este componente nos permite asociar un fichero con código fuente a un objeto y poder de esta manera controlar el comportamiento del objeto en tiempo de ejecución o directamente almacenar información de nuestro modelo en forma de variables y estructuras de datos.

6.3. Unity Scripting

Como hemos mencionado anteriormente, un script dentro de la jerarquía de clases de Unity, se trata como un componente, un componente personalizado, al cual le podemos añadir variables de uso público para que puedan ser retocadas en el propio editor de Unity, o definir rutinas de comportamiento en diferentes fases de ejecución.

Todo script de Unity debe tener un nombre de clase y debe heredar de la clase MonoBehaviour, dado que esta clase nos proporciona las funciones principales de control e integración con Unity, y permiten gestionar el flujo de ejecución del script.

6.4. Flujo de ejecución scripts Unity

Un script de Unity ejecuta unas funciones de evento en un orden predeterminado. Esto nos permite controlar la secuencia de ejecución de un script como una máquina de estados.

La Figura 18 muestra un diagrama de todas las funciones de eventos, su orden de ejecución y sus condiciones de entrada y salida.

Este diagrama nos muestra todo el flujo de ejecución con todas las funciones que posee. En la práctica, sin embargo, salvo circunstancias muy especiales, lo normal es trabajar con no más de 4 o 5 funciones de eventos en el mismo script. Estas funciones son las siguientes [50]:

- Awake: Esta función siempre se ejecuta la primera. Es muy útil para instanciar y realizar las declaraciones de variables y relaciones iniciales.

- Start: Esta función solo se llama una vez por script y tiene una ejecución temprana.

- FixedUpdate: Es la primera función que se ejecuta dentro del ciclo de funciones físicas, estas son funciones dedicadas al tratamiento de físicas, se ejecutan de manera constante, una vez por frame, pero si la tasa de fps (frames per second) es mayor que la tasa de actualización del tiempo del sistema, esta función se llamará más veces, esto puede hacer que se calculen físicas de una manera más fluida, independientemente del rendimiento de la plataforma que lo ejecuta.

- OnTriggerXXX, OnCollisionXXX: Son funciones especiales para tratamiento de eventos lanzados por colisiones o interaccion entre colliders. Establecen la lógica principal de interacción entre objetos de la escena en tiempo de ejecución.

- YieldStartCorrutine: Función específica para trabajar con corrutinas. Estas son una manera de realizar paralelismo dentro del mismo script, permitiendo la ejecución del código por diferentes hilos de computación.

- OnDestroy: Última función de la pila, después de esta llamada el script terminará su ejecución. Se llamará a este método cuando destruyamos un objeto con un script asociado a este que disponga de esta función.

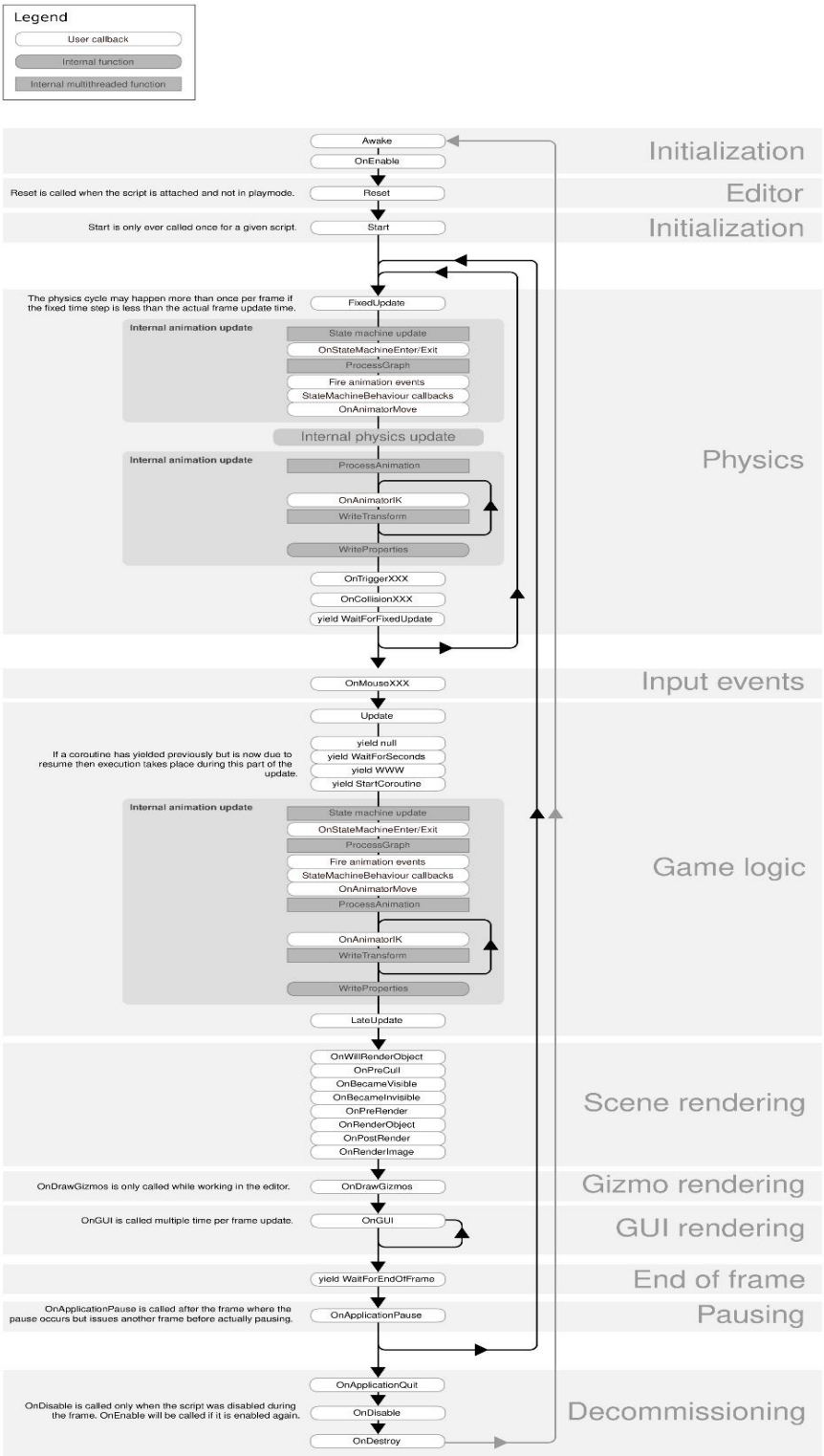


Figura 18: Diagrama de flujo de ejecución de funciones en Unity

6.5. Sistema de ficheros en Unity

Como ya hemos visto anteriormente, la clase básica de unity es el GameObject. Sin embargo, en la lógica de almacenamiento persistente, el elemento principal es el “asset”.

Los assets son ficheros de unity que contienen una colección jerárquica interpretada como objetos, y componen los elementos básicos del videojuego. Ejemplos de estos son: materiales, clips de audio, animaciones, etc.

Algunos assets poseen datos en formatos nativos de unity, otros, en cambio deben procesarse en formatos nativos [51].

Hay que mencionar 3 tipos de asset especiales:

- La escena: La escena es un asset que contiene todos los Objects de un videojuego. Normalmente conforman los diferentes niveles de un videojuego, es importante organizar un videojuego basándolo en niveles, en cada nivel debería de cargarse una escena independiente. La razón principal para tomar esta medida de diseño es la optimización de los recursos, pero en proyectos muy grandes, la organización estructurada en escenas cobra mucha importancia, para que sea cómodo trabajar con todos los objetos de cada escena independientemente.

- Los prefabs: Estos assets tienen una función vital en la implementación de videojuegos, ya que son “Objetos prefabricados”, esto es, GameObjects con unos componentes y unas propiedades predefinidas, que se guardan como un único objeto predefinido. La principal ventaja de usar estos assets es la de poder poner en escena objetos personalizados, de manera rápida tanto en tiempo de compilación como en tiempo de ejecución. De esta forma podemos instanciar objetos prefabricados, con las características que queramos, dinámicamente, sin duplicar código, únicamente configurando uno previamente. Un ejemplo de estos assets podría ser el de un proyectil, este tiene unos componentes que lo definen, una malla, un material, un script, un rigidBody, y estos tendrán unas propiedades, pero en todos los objetos serán las mismas, teniendo esto en cuenta, podemos instanciar en tiempo de ejecución cada prefab de proyectil cada vez que se accione un disparador y destruirlo de la escena cuando termine su función para optimizar recursos, y todos los proyectiles que instanciemos se comportaran de acuerdo a las mismas reglas, definiendo una lógica de interacción homogénea.

- La cámara: Como ya vimos en el apartado 5.2 las cámaras renderizan todos los objetos de una escena. También se almacenan como un asset de unity.

Capitulo VII

Proyecto. Mota Tower Defense

Ya hemos analizado mediante un estudio del arte los videojuegos punteros del mercado, y los relacionados con las nuevas tecnologías. En este capítulo describiremos los pasos previos de planificación, diseño e implementación de la primera versión de nuestro videojuego.

Como mencionamos en capítulos anteriores, el proyecto se desglosa en una serie de versiones (*Tabla 1 Versiones del videojuego*) que irán ampliando funcionalidad para que el juego resulte lo más atractivo posible. La última versión deberá alcanzar los requisitos funcionales y no funcionales disponibles en el documento de diseño de juego (Apéndice E)

Debido a la complejidad del proyecto abordado y las limitaciones del tiempo a dedicar al TFG (300 horas) es imposible abordar todas las versiones indicadas en la tabla 1. Se estima que la primera versión es la única que se podrá completar dentro de los plazos marcados para la realización del TFG.

7.1. Metodología

Para poder realizar el proyecto cumpliendo con los requisitos y la normativa, se opta por utilizar una metodología de desarrollo ágil, en concreto la metodología Scrum.

Sin embargo, debido a las características de este proyecto y del propio TFG, se realizarán algunas modificaciones a la metodología original, para que esta se adapte perfectamente al proceso de diseño e implementación del proyecto.

En la metodología Scrum, los requisitos del proyecto se describen a través de historias de usuario. Estas deben ser independientes unas de otras, lo que permite que equipos de desarrollo independientes puedan dedicarse a estas sin tener que depender del trabajo de otros equipos de desarrollo.

Es importante que estas historias de usuario sean lo más atómicas posible, y que sean verificables. Esto permitirá descomponer el proyecto modularmente en las mínimas unidades funcionales, y que estas puedan verificarse de manera independiente.

El “Product Backlog” colecciona todas las historias de usuario del proyecto, proporcionando una retroalimentación proactiva sobre el estado actual de desarrollo del proyecto. Esto permite verificar que el proyecto se realiza en los plazos marcados, detectar retrasos y consumo extra de recursos.

Normalmente en esta metodología de trabajo ágil, las historias de usuario deben estar completadas y deben ser presentadas dentro de unos marcos temporales llamados Sprints, que suelen tener como duración una o dos semanas.

En nuestro proyecto debido a que solo se podrá destinar un número de horas muy limitadas cada día, se optará por Sprints de 60 horas.

Al inicio de cada Sprint se contabilizan las historias pendientes de implementación e integración y se muestra en un gráfico, este gráfico se llama Burn Down Chart. También se concretan las historias de usuario que se van a implementar durante el Sprint.

A continuación se muestran los roles de Scrum para este proyecto:

- Desarrolladores: El equipo encargado de la implementación de la funcionalidad de cada historia de usuario, para este proyecto el equipo será unipersonal, Alfredo Fernández.

- Product Owner: Se trata de un intermediario entre el cliente y el equipo de desarrollo. Este rol lo ejercerá el tutor, Carlos Enrique Vivaracho y Pablo Gatón.

- Scrum Master: Este rol se encarga de dar soporte y asistencia al equipo de desarrollo, así como de asegurarse de que se cumplen las metodologías. El tutor también actuará como Scrum Master, Carlos Enrique Vivaracho.

Utilizar metodologías ágiles, permite reducir los riesgos asociados a la implementación de un proyecto aportando un nivel muy alto de planificación y flexibilidad en la parte de gestión, facilitando que el proyecto cumpla los requisitos en tiempo y forma.

7.2. Planificación inicial

7.2.1. Distribución inicial

La situación del desarrollador imposibilita su dedicación a tiempo completo. Por lo que se establece una jornada de trabajo de lunes a viernes de 3 horas diarias.

La distribución de los Sprints se detalla a continuación:

- Sprint 1: Empieza el lunes, 13 de Enero de 2020 y termina el viernes 7 de Febrero de 2020.
- Sprint 2: Empieza el lunes, 10 de Febrero de 2020 y termina el viernes 6 de Marzo de 2020.
- Sprint 3: Empieza el lunes, 9 de Marzo de 2020 y termina el viernes 3 de Abril de 2020.
- Sprint 4: Empieza el lunes, 6 de Abril de 2020 y termina el viernes 1 de Mayo de 2020.
- Sprint 5: Empieza el lunes, 4 de Mayo de 2020 y termina el viernes 29 de Mayo de 2020.

7.2.2. Entorno de desarrollo

Para la implementación del proyecto se utilizarán unos determinados recursos software/hardware. Se detallan a continuación.

Recursos Software

La siguiente lista describe todos los productos software que se utilizarán para el desarrollo del videojuego:

- **MS Outlook:** Gestor de correo electrónico para comunicación con tutor y cliente.

- **Lifesize (versión web):** Plataforma de videoconferencia para comunicación con tutor y cliente.
- **MS Word 2013:** Programa de procesamiento de textos destinado a la redacción de la memoria, GDD y el resto de la documentación del proyecto.
- **Google Chrome:** Navegador Web para las consultas bibliográficas y webinars.
- **Modelio 4.0.1:** Herramienta UML2.0 para la creación de los diagramas de modelado y diseño.
- **Unity 2018:** Motor de videojuego, se empleará para la implementación del videojuego.
- **Visual Studio 2019:** IDE para la programación de los scripts en C# que utilizará Unity.
- **SteamVR:** Plataforma y SDK para la utilización de VR en equipos personales, integración con HTC Vive y desarrollo de videojuegos en VR.

Recursos Hardware

Para la realización de este proyecto es necesario unos recursos hardware concretos, que se listan a continuación.

- **Ordenador de Sobremesa con tarjeta gráfica dedicada:**

  Es necesario tener un equipo potente para trabajar con motores de videojuego. Pero todavía hace falta más potencia para trabajar con soltura en productos destinados a la realidad virtual. Es la razón por la que se necesita un ordenador con Gráfica dedicada. Esto descarta la mayoría de ordenadores portátiles del mercado, que disponen de tarjetas integradas en CPU, y los que poseen tarjetas gráficas dedicadas, son versiones mucho menos potentes, y con menor capacidad de refrigeración que las tarjetas normales que se montan en equipos de sobremesa.

Especificaciones técnicas ordenador desarrollo	
Procesador	Intel Core i7 7th Gen
Memoria RAM	8GB DDR4
Tarjeta Gráfica	NVidia GTX GeForce 980 ti
Disco duro	HHD 2TB
Placa base	Asus Maximus X
SO	Windows 10

Tabla 4: Especificaciones técnicas ordenador sobremesa

- **Pantallas:**

  Para este tipo de proyectos es esencial trabajar en un entorno multipantalla, de esta manera podemos tener en una ventana el motor del videojuego constantemente para hacer los retoques necesarios, mientras que en el otro podemos tener el IDE para la programación de la lógica de

negocio del proyecto. Las especificaciones de la pantalla no son tan importantes y bastaría con que un monitor tenga una relación de aspecto estándar moderno (16:10, 16:9). Con estas relaciones se permite visualizar correctamente todas las ventanas necesarias de Unity, sin perder detalles importantes.

- **Gafas de realidad virtual:**

  Obviamente, necesitamos un sistema de visualización en realidad virtual, no solo para el producto final, si no para las pruebas en desarrollo que se tengan que hacer. Para este caso utilizaremos las gafas que proporciona HTC Vive. Podríamos utilizar cualquier versión de estas gafas dado que cualquiera es compatible con SteamVR y valdrán para la perfecta reproducción del videojuego. En nuestro caso utilizamos las HTC Vive, versión normal ampliada, que se compone de las gafas, dos controladores, el hub de conexiones, y dos estaciones satélite.

7.2.4. Estimación de costes

En el siguiente apartado se estiman detalladamente los costes previstos del proyecto en su fase inicial. Los salarios y costes asociados se calcularán incluyendo los impuestos pertinentes en el momento de su planificación del gobierno de España.

- **Salario de los miembros del equipo de desarrollo:**

  En el sector del desarrollo de software, existe un sistema de clasificación de sus profesionales en función de la experiencia laboral y las competencias técnicas demostradas a lo largo de su carrera. Estas categorías definen, de media, el salario. Son las siguientes clasificadas de menor categoría a mayor:

  a) **Desarrollador Junior:** Categoría inicial bajo o nulo nivel de responsabilidades, su trabajo debe ser supervisado y experiencia inferior a 3 años, el salario esta entre los 16000 y los 20000 euros brutos anuales.

  b) Desarrollador Intermediate: El desarrollador empieza a tener responsabilidades y un nivel técnico avanzado. La experiencia suele ser inferior a 6 años y el salario oscila entre los 20000 y los 24000 euros brutos anuales.

  c) Desarrollador Advanced: En este caso el profesional dispone de un alto grado técnico, pero no tiene el nivel de responsabilidades de un desarrollador senior, suelen tener experiencias superiores a los 6 años y su salario está entre los 24000 y los 30000 euros.

  d) Desarrollador Senior: El desarrollador senior es la última categoría puramente técnica dentro del desarrollo de software, dispone de más de 10 años de experiencia, su nivel técnico es muy alto y tiene el grado más alto de responsabilidades dentro de la categoría técnica. Suelen tener salarios superiores a los 30000 euros e inferiores a los 40000 euros.

Dentro del desarrollo de videojuegos se aceptan las mismas categorías pero los salarios suelen ser algo inferiores, la razón de esto es, que la mayoría de empresas del sector son pequeños estudios independientes que no disponen de la capacidad financiera para asumir los salarios del sector. Las empresas “triple A” [52] pueden asumir salarios incluso, más altos que

los descritos anteriormente, pero en España no hay demasiados estudios que entren en esta categoría empresarial.

Hay que destacar, también, que las categorías descritas anteriormente no son categorías profesionales oficiales reconocidas por el estatuto general de los trabajadores, forman parte de una clasificación interna consolidada dentro de las empresas del sector, que se utilizan para definir las franjas salariales de sus profesionales. Estas categorías solo referencias al sector específico del desarrollo de software, en su vertiente más técnica, existen puestos de mayor responsabilidad a nivel de gestión, como por ejemplo el jefe de proyecto.

En este caso, el único miembro del equipo de desarrollo es Alfredo Fernández, ejercerá todos los roles implicados dentro del proceso de planificación, gestión e implementación. Por esta razón y dado que el desarrollador acumula más de dos años de experiencia profesional en el sector del desarrollo software, el coste que se tendría que remunerar a este, asciende a los 27500 euros brutos anuales, que se traduce en 13.22 euros la hora.

- **Licencias de software:**

  Para este proyecto todo el software que se utiliza tiene licencias GPL o son licencias Community, lo que significa que no habrá costes asociados por el uso de estas herramientas.

- **Recursos hardware:**

  Se tienen en cuenta el coste de los equipos hardware necesarios para la implementación del proyecto. Esto incluye el ordenador de sobremesa, los monitores y las gafas de realidad virtual.

- **Recursos de Unity para añadir al videojuego:**

  Se utilizarán unos activos de Unity para los modelos de los zombis del videojuego. Estos se adquirirán de la tienda de assets oficial de Unity. Su coste será de 60 euros. El resto de assets adquiridos de la tienda serán gratuitos y de libre uso.

- **Espacio de trabajo y otros:**

  Para el desarrollo del proyecto es necesario tener un espacio de trabajo acondicionado, esto significa, un espacio cerrado, privado, que disponga de una mesa y una silla, y los servicios indispensables para el desempeño del proyecto, entre los que se incluyen luz, conexión a internet, etcétera.

  Normalmente existen espacios disponibles para su alquiler ya acondicionados para entornos laborales, que cumplen con las disposiciones del reglamento de prevención de riesgos laborales. [53]

  Se tendrán en cuenta en la estimación el coste del alquiler de este espacio durante el periodo total que lleve el desarrollo del proyecto.

  Consideramos opciones en alquiler que disponen de periodos de facturación por horas. En este caso nos costaría 4 euros la hora.

7.2.5. Estimación de costes totales. Desglose

En tabla 5 se muestra un desglose de todos los costes estimados. El coste total del proyecto será de 6586 euros.

Concepto	Coste unitario	Unidades	Subtotal
Desarrollador salario	13.22 €	300 horas	3966 €
Ordenador sobremesa	1000 €	1	1000 €
Gafas HTC Vive	650€	1	600 €
Monitores	180 €	2	360 €
Espacio de trabajo	4 €	300 horas	1200 €
Recursos de Unity	60 €	1	60 €
Total	**6586 €**		

Tabla 5: Costes totales estimados del proyecto. Desglose

7.2.6. Pruebas de validación

Para que una historia de usuario en Scrum pueda pasar al estado “completada”, es necesario que pasen por unas pruebas de validación. Por esta razón todas las historias de usuario que se completen deben de pasar una prueba de validación final. Hay que tener en cuenta que son pruebas unitarias, y que no forman parte de la fase de pruebas del proyecto que explicamos en el apartado “evaluación del videojuego”.

El proceso de validación de las historias será completamente manual, no se implementará ningún mecanismo de integración y evaluación continua, ni se utilizarán herramientas específicas de testing. Se utilizarán las propias condiciones de satisfacción de las historias de usuario, como output de las baterías de pruebas de caja negra.

7.2.7. Análisis de riesgos

En este apartado se realizará un análisis de riesgos únicamente sobre producto, estos pueden producir retrasos en el desarrollo del proyecto. Se realizará en varias fases, estas son: Identificación de los riesgos, evaluar los riesgos, cálculo de impacto y plan de actuación.

Identificación de riesgos

La siguiente tabla describe los riesgos asociados al proceso de desarrollo del proyecto. Están identificados con un id, asociados a un nombre y una descripción.

ID	Riesgo	Descripción
1	Enfermedad	El desarrollador contrae una enfermedad que le incapacita para trabajar temporalmente
2	Planificación inadecuada	La gestión de riesgo y recursos no está bien definida y genera retrasos
3	Inexperiencia con Unity	La falta de conocimientos con el motor de videojuego ocasiona retrasos
4	Inexperiencia con C#	La falta de conocimiento del lenguaje de scripting genera errores imprevistos que ralentizan el desarrollo
5	Fallo del hardware	Algunas de las herramientas, como las gafas de VR tienen un malfuncionamiento, lo que ocasiona que no se pueda trabajar con ellas.
6	Perdida de datos por fallo	Los dispositivos que almacenan datos sobre el proyecto dejan de funcionar provocando una situación de perdida de información necesaria en el desarrollo
7	Fallo en el suministro	Cortes en el suministro tanto de luz como en la conexión a internet provocan perdida de continuidad de desarrollo
8	Fallos por cambios en librerías VR	Actualizaciones en librerías y SDK's pueden generar cambios que provoquen un malfuncionamiento de partes del código ya validadas

Tabla 6: Riesgos asociados al desarrollo del proyecto

Evaluación de riesgos

Ya tenemos los riesgos identificados, ahora hay que evaluarlos, clasificarlos y ordenarlos en una lista, para posteriormente calcular cuales son los riesgos que pueden tener un impacto mayor.

ID	Riesgo	Probabilidad	Retraso estimado en días	Exposición al riesgo (en días)
2	Planificación inadecuada	0.4	15	6
3	Inexperiencia con Unity	0.25	10	2.5
8	Fallos por cambios en librerías VR	0.22	5	1.1
4	Inexperiencia con C#	0.15	7	1.05
1	Enfermedad	0.2	2	0.4

6	Perdida de datos por fallo	0.02	8	0.16
5	Fallo del hardware	0.1	1	0.1
7	Fallo en el suministro	0.05	1	0.05

Tabla 7: Evaluación y clasificación de riesgos

Análisis de riesgos. Matriz y mapa de riesgo

En el apartado anterior hemos estimado la probabilidad de cada riesgo y su posible impacto, y los hemos presentado en una tabla ordenada en función de la mayor probabilidad de riesgo. Sin embargo, clasificar los riesgos en función de la estimación de su probabilidad, no proporciona una información útil. Para tener datos que nos puedan ayudar a priorizar los riesgos, conviene relacionar la probabilidad con su impacto.

Para ello, realizaremos primero una tabla de ponderaciones para clasificar la probabilidad y el impacto, en unas categorías homogéneas que permitan su rápida interpretación y relación.

Probabilidad de ocurrencia			**Grado de impacto**		
0.9 **0.6**	Recurrente	Probabilidad de ocurrencia muy alta	**60** **31**	Catastrófico	Influye directamente en la viabilidad del proyecto
0.5 **0.26**	Probable	Probabilidad de ocurrencia alta	**30** **16**	Grave	Generaría la perdida de funcionabilidad importante
0.25 **0.21**	Posible	Probabilidad de ocurrencia media	**15** **11**	Serio	Ocasionaría perdidas de funcionabilidad secundaria
0.2 **0.06**	Inusual	Probabilidad de ocurrencia baja	**10** **6**	Moderado	No afectaría al cumplimiento de los requisitos
0.05 **0.01**	Remota	Probabilidad de ocurrencia muy baja	**5** **1**	Insignificante	No tiene ningún efecto en el desarrollo del proyecto

Tabla 8: Tabla de ponderación para la valoración de riesgos

Catastrófico					
Grave				2	
Serio			3		
Moderado	6	4			
Insignificante	5 7	1	8		
Impacto / Probabilidad	**Remota**	**Inusual**	**Posible**	**Probable**	**Recurrente**

Tabla 9: Matriz de riesgo asociada al proyecto con los ID de cada Riesgo

En la tabla anterior se han marcado los identificadores correspondientes a los riesgos dentro de una matriz de riesgos.

El objetivo de esta matriz es determinar para que riesgos es necesario tomar medidas inmediatas, cautelares y de prevención, o desestimarlas.

Las celdas de color negro definen los riesgos más críticos que causarían una inviabilidad del proyecto. En rojo se marcan los riesgos que requieren tomar medidas inmediatas, para así evitar el aumento del impacto causado por estas. Para los riesgos en amarillo hay que establecer medidas cautelares preventivas. Y los riesgos que se encuentren en el área verde se pueden desestimar porque no causarían perjuicio en el proyecto.

7.2.8. Plan de actuación

Ahora que ya hemos catalogado y ordenado los riesgos, podemos establecer un plan de actuación para todos los riesgos que estén fuera del área verde, dentro de la matriz de riesgos.

ID	Riesgo	Exposición al riesgo (en días)	Plan de acción
2	Planificación inadecuada	6	Se tendrá que realizar un replanteamiento de la planificación, estableciendo una prioridad más adecuada a la situación, para los requisitos secundarios del proyecto.
3	Inexperiencia con Unity	2.5	Se considerará la compra de cursos oficiales ofrecidos por los desarrolladores de Unity, fuera de horario. También se abrirán peticiones en los foros de la comunidad de Unity para buscar aplicaciones técnicas rápidas en el desarrollo.

8	Fallos por cambios en librerías VR	1.1	Se mantendrán, siempre que sea posible, las versiones antiguas de los SDK's y librerías. De no ser posible se estudiarán los cambios de las nuevas versiones, previo a la actualización, con el fin de anticipar posibles errores de integración.
4	Inexperiencia con C#	1.05	Se buscará ayuda en la comunidad de desarrolladores De .NET oficial de Microsoft. Considerar formación oficial en horas extra.

Tabla 10: Plan de acción para los riesgos previamente catalogados

7.3. Versiones

- v1.0.0: Primera versión jugable, se podrá navegar por todo el castillo, saldrán oleadas en un spawn concreto y todas del mismo punto. Solo se podrá defender el castillo con el arco. Se implementará el sistema de rondas, el combate, y el evento fin de partida.

- v2.0.0: Las oleadas saldrán de diferentes puntos del mapa, se implementará el sistema de supervivientes (solo para la creación de flechas). Se incluirá música a las escenas.

- v3.0.0: Se implementará el sistema de defensa. Se incluirán objetos con los que interactuar para la historia, Audios y diálogos de la historia.

- v4.0.0: Se utilizará esta versión para mejorar todas las cuestiones anteriores, conseguir mejores efectos visuales, corregir problemas de interacción, etcétera.

7.4. Historias de usuario

En este apartado, analizaremos todos los requisitos funcionales del proyecto a través de las historias de usuario. Estas nos proporcionarán la información sobre los módulos funcionales, como ya explicamos anteriormente. Estas tarjetas, nos permitirán, mediante un vistazo rápido, conocer los datos más relevantes sobre una funcionalidad en concreto a desarrollar.

Las historias se componen de un código identificativo de la historia, un título, un valor (para el cliente), una estimación en horas, una descripción y unas condiciones de satisfacción que se deben cumplir para poder validar la historia, una vez desarrollada.

A continuación se exponen todas las historias de usuario que componen el product backlog.

US-01	
Integración del modelo del castillo	
Valor: 1000	**Estimación:** 6h
Descripción: Como cliente, quiero que la aplicación tenga el modelo del castillo de la Mota que hemos proporcionado, para que el jugador pueda ver el castillo.	
Condiciones de satisfacción: • El modelo del castillo debe ser el elemento principal de diseño. • Debe de estar bien acoplado a la escena • Dispondrá de un elemento independiente para la parte interna caminable, y un elemento para la superficie externa que podrán recorrer los zombis	

US-02	
Integración de la cámara para VR	
Valor: 1000	**Estimación:** 15h
Descripción: Como cliente, quiero que la aplicación pueda ser observable usando las últimas técnicas de realidad virtual, para que el jugador pueda utilizar el equipo HTC Vive y aumentar la sensación de inmersión.	
Condiciones de satisfacción: • El escenario debe ser renderizado con las gafas de realidad virtual HTC Vive • Los controladores de realidad aumentada deben de simular las manos del jugador	

US-03	
Sistema de teletransporte	
Valor: 1000	**Estimación:** 15h
Descripción: Como cliente, quiero que la aplicación permita moverse a través de la escena usando los controladores que ofrece SteamVR, para que el jugador pueda desplazarse para recorrer el castillo.	
Condiciones de satisfacción: • El jugador debe poder desplazarse cortas distancias dentro de los límites del castillo • Se usará un marcador indicativo para informar el punto final del teletransporte • El control del teletransporte debe ser fluido para agilizar el desplazamiento del jugador	

US-04	
Lógica de zombis I (instanciación, recorrido y parada)	
Valor: 900	**Estimación:** 20h
Descripción: Como cliente, quiero que los zombis aparezcan en un punto concreto del exterior del castillo, y que recorran la ruta más corta hasta la plaza interior del castillo, para que el jugador pueda ver objetos siguiendo un recorrido	
Condiciones de satisfacción: • El área de aparición no puede ser un punto • La aparición debe ser en una posición aleatoria dentro de unos parámetros que le mantengan dentro del escenario • El objeto debe recorrer siempre el camino más corto • El objeto no puede chocarse con el modelo del castillo, ni con elementos externos	

US-05	
Integración del sistema de arco de SteamVR	
Valor: 900	**Estimación:** 9h
Descripción: Como cliente, quiero que el juego disponga de un sistema de arco y flechas, y que usando los controladores de HTC Vive, pueda disparar flechas que colisionen con el modelo del castillo	
Condiciones de satisfacción: • El jugador debe poder sostener un arco con un controlador y la flecha con el otro • El jugador debe poder realizar el movimiento de tensado de cuerda como si fuera un arco real para poder disparar la flecha, y darle un impulso proporcional a la fuerza de tensado • Las flechas seguirán un sistema de físicas que sea lo más parecido a la realidad posible • Las flechas no podrán atravesar el modelo del castillo, ni elementos externos	

US-06	
Modelos de zombis	
Valor: 800	**Estimación:** 10h
Descripción: Como cliente, quiero que los zombis tengan un modelo asociado, con diferentes variantes, para que el jugador pueda diferenciar tipos de zombis	
Condiciones de satisfacción: • Habrá 4 modelos principales diferentes de zombis • Los zánganos tendrán que tener cambios sutiles para generar una mezcla más heterogénea	

US-07	
Lógica de zombis II (control de estadísticas)	
Valor: 900	**Estimación:** 12h
Descripción: Como cliente, quiero que los zombis posean unas propiedades para que interactúen con el jugador y el entorno	
Condiciones de satisfacción: • Los zombis tendrán unos puntos de vida, puntos de ataque, velocidad y velocidad de ataque predefinidos • Los zombis podrán sufrir daños a los puntos de vida cuando sean impactados por una flecha • Cuando su vida llega a cero deben ser eliminados de la escena	

US-08	
Barrera de acceso al castillo	
Valor: 800	**Estimación:** 5h
Descripción: Como cliente, quiero que el castillo posea una barrera en su puerta principal con la que puedan interactuar los zombis, para frenar su acceso al castillo durante un tiempo, que dependerá del número de zombis que estén en contacto con la puerta, y su tipo	
Condiciones de satisfacción: • La puerta tendrá un modelo que se asemeje a una barricada • La puerta tendrá una cantidad finita de puntos de vida • Los zombis que entren en contacto con la puerta irán bajando sus puntos de vida linealmente en función de su velocidad de ataque y su daño de ataque. • Cuando la puerta baje a los cero puntos de vida, se destruirá	

US-09	
Entorno	
Valor: 500	**Estimación:** 8h
Descripción: Como cliente, quiero que el castillo esté rodeado por un entorno realista, para aumentar la inmersión del jugador	
Condiciones de satisfacción: • Se incorporará un sistema de “skybox” para encerrar a toda la escena en un escenario con cielo y nubes • Las zonas exteriores del castillo tendrán bosque de atrezo	

US-10	
Flechas de hielo	
Valor: 450	**Estimación:** 20h
Descripción: Como cliente, quiero que exista un tipo de flecha diferente que pueda interactuar con los zombis, para ofrecer al jugador más posibilidades de eliminar a los zombis	
Condiciones de satisfacción: • La flecha tendrá cambios en su modelo sutiles que permitan diferenciarla • Cuando impacte con un zombi, aparte de aplicarle el daño correspondiente, se reducirá la velocidad de movimiento del zombi de manera permanente • El número de flechas serán limitadas • El jugador podrá seleccionar rápidamente entre los dos tipos de flecha	

US-11	
Teletransporte rápido de larga distancia	
Valor: 800	**Estimación:** 18h
Descripción: Como cliente, quiero que el jugador disponga de una manera rápida de desplazarse por puntos clave del castillo, para facilitar al jugador el acceso rápido, a una zona desde la que pueda disparar flechas a los zombis	
Condiciones de satisfacción: • El jugador podrá usar un botón para desplegar una réplica del castillo en miniatura en la que se marcarán puntos a los que puede desplazarse • Podrá iterar una lista circular de puntos a los que desplazarse • El jugador tendrá que confirmar dándole a un botón que quiere desplazarse a la posición marcada	

US-12	
Sistema de puntuaciones y persistencia	
Valor: 600	**Estimación:** 12h
Descripción: Como cliente, quiero que al ir avanzando en el juego se vayan almacenando las puntuaciones que consigue un jugador, para que exista una persistencia del progreso del jugador	
Condiciones de satisfacción: • Los zombis deben de proporcionar puntos al jugador • Los puntos del jugador se guardarán persistentemente durante el transcurso de su sesión de juego • Cada tipo de zombi dará diferentes puntos	

US-13	
Sistema de control de oleadas	
Valor: 900	**Estimación:** 20h
Descripción: Como cliente, quiero que los zombis vayan apareciendo controlados por un sistema de oleadas para evitar una generación masiva de zombis, y proporcionar al jugador un tiempo de descanso entre oleadas	
Condiciones de satisfacción: • El número de oleadas será infinito • En cada ronda se generaran unos zombis de uno o varios tipos • Se incrementará el número de zombis generados en cada ronda de cada tipo	

US-14	
Lógica del juego	
Valor: 850	**Estimación:** 8h
Descripción: Como cliente, quiero que exista un sistema de control del juego para definir las reglas del juego	
Condiciones de satisfacción: • El castillo tendrá un número de vidas antes de que el juego termine • Zombis de diferentes tipos que entren al castillo reducirán el número de vidas del castillo, en diferentes cantidades • Al final del juego se mostrará la puntuación alcanzada por el jugador	

US-15	
Escenas introductorias y formulario	
Valor: 600	**Estimación:** 12h
Descripción: Como cliente, quiero que existan al menos 2 escenas en el juego, para que el jugador pueda introducir su “Nick de jugador” y aprender los controles básicos.	
Condiciones de satisfacción: • El jugador podrá introducir mediante una interfaz en un formulario su Nick • Si no es la primera partida se mostrará la puntuación anterior obtenida • Cuando termine una partida se tiene que cargar esta escena con el formulario para que pueda ver su puntuación y empezar una nueva partida	

US-16	
Animaciones de los zombis	
Valor: 700	**Estimación:** 20h
Descripción: Como cliente, quiero los zombis tengan unas animaciones asociadas a actividades, para aumentar la sensación inmersiva del jugador	
Condiciones de satisfacción: • Los zombis tendrán una animación para caminar • Los zombis tendrán una animación para atacar a la barricada • Los zombis tendrán una animación cuando su vida baje a los 0 puntos antes de desaparecer	

US-17	
Sistema de combate	
Valor: 900	**Estimación:** 18h
Descripción: Como cliente, quiero que el juego disponga de un sistema de evaluación de daños de combate por flechas, para que el jugador pueda utilizar el arco para defenderse de las oleadas	
Condiciones de satisfacción: • El jugador aplicará daño por cada flecha que impacte en el modelo de algún tipo de zombi • Las flechas normales que entren en contacto con una fuente de fuego adquirirán fuego en su punta • Si una flecha con fuego entra en contacto con un zombi se le aplicará ticks de daño en el tiempo a parte del daño de la flecha	

US-18	
Banda sonora	
Valor: 250	**Estimación:** 8h
Descripción: Como cliente, quiero que el juego disponga de una música constante para aumentar el nivel de inmersión del jugador	
Condiciones de satisfacción: • La música será continua • No podrá estar a un volumen muy alto, se tiene que poder escuchar el resto de sfx • Tiene que provocar una sensación de tensión constante	

US-19	
Efectos de sonido	
Valor: 300	**Estimación:** 13h
Descripción: Como cliente, quiero que el arco tenga efectos de sonido al ser disparado, así mismo los zombis también deberían tener efectos de sonido que acompañen a las animaciones, para mejorar el realismo del juego	
Condiciones de satisfacción: • El arco tendrá sonido al tensar el arco y al disparar una flecha • Los zombis tendrán que reproducir un grito al morir	

US-20	
Tabla de puntuaciones online	
Valor: 200	**Estimación:** 25h
Descripción: Como cliente, quiero que los jugadores que jueguen y obtengan una puntuación quede esta, registrada y asignada al jugador en una tabla online que se pueda consultar, para fomentar el espíritu competitivo y de superación entre jugadores	
Condiciones de satisfacción: • Cada juego terminará enviando una petición para almacenar los datos del jugador y su clasificación en un servidor online • Se podrá recuperar la tabla con puntuaciones en el propio juego	

7.5. Implementación

En este apartado detallaremos en cada sprint, el sprint backlog, los detalles de la implementación y las pruebas de validación, así como el Burn Down Chart en los sprints correspondientes.

7.5.1. Sprint 1

Las historias de usuario que se implementarán se describen a continuación en el sprint Backlog.

Sprint Backlog

- **US-01**
- **US-02**
- **US-03**
- **US-05**

Detalles de implementación

Se importa como asset el modelo del castillo, proporcionado por Irzón.

Hay 3 elementos a importar:

- El modelo del castillo en su totalidad, incluyendo terrenos exteriores.
- Un mapa de las zonas internas por las que el jugador podrá desplazarse
- Una plataforma exterior para el desplazamiento por la misma de los zombis como un único elemento independiente.

Estos elementos, hay que colocarlos en la escena y acoplarlos en unas posiciones fijas para que encajen. Se establecerá un sistema de herencia con un objeto padre, para que estos se mantengan en la misma posición relativa, independientemente de que modifiquemos la posición del conjunto de modelos.

Se procede a la integración de la cámara utilizando el SDK SteamVR.

Para ello se crea el gameobject Player, este tendrá como objetos en jerarquía la cámara vinculada a los scripts y gameobjects de SteamVR. Dentro de estos se incluyen objetos para los dos controladores de HTC y la cámara.

Figura 19: Jerarquia de objetos "Player"

Para la historia *US-03* incorporamos el sistema de teletransporte, SteamVR dispone de funcionalidad para ese fin utilizando los controladores de las gafas de VR, mediante el prefab Teleport y el elemento físico Raycast. El prefab Teleport, es un objeto prediseñado incluido en la librería de steamVR. Nos permite, mediante el uso de los controladores de VR, marcar una posición del escenario para que el jugador pueda desplazarse instantáneamente hacia la ubicación marcada. Para marcar la posición se hace uso del elemento de Unity Raycast, que proyecta un láser desde el controlador hasta un objeto de la escena con el que colisiona. Este objeto nos facilitará la implementación del sistema de movimiento del jugador, pero hay que definir las partes de la escena hacia las que se puede lanzar el raycast.

Para esto tenemos una malla asociada al modelo de la superficie interna del castillo. Sobre esta añadiremos el script de teleportArea, que nos proporcionará la lógica para desplazarnos usando los controladores.

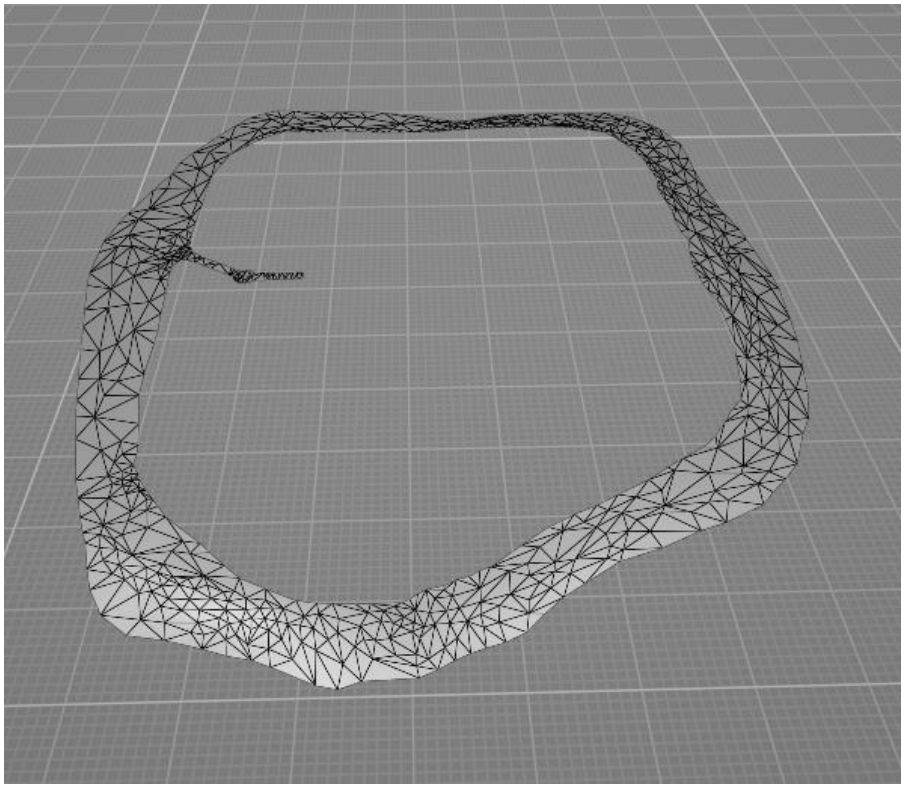


Figura 20: Malla de camino exterior del castillo

De la misma manera que con el teletransporte, el arco ya está disponible como prefab. Lo añadimos a la escena y modificamos las propiedades para evitar que el número de flechas sea finito.

Para cumplir con las condiciones de validación de la historia de usuario, hay que añadir a los elementos del modelo del castillo, un componente collider, estos definen la forma de un objeto en la escena cuando se aplican interacciones físicas, para que pueda usar un material que interactúe con los prefabs de las flechas del sistema de arco. De esta manera las flechas colisionarán con los modelos del castillo, e incluso se podrán clavar si se desplazan lo suficientemente rápido y el ángulo de entrada está dentro de lo aceptable por el material.

Pruebas de validación

US-01-T01	
Título	Prueba de visualización del castillo con cámara por defecto
Comportamiento esperado	El castillo se ve correctamente, no hay partes mal acopladas, los puntos de referencia coinciden
Resultado de la prueba	OK

US-02-T01	
Título	Prueba de visualización del castillo con cámara SteamVR y las HTC Vive
Comportamiento esperado	Al mover la cabeza se simula el movimiento de la cámara que renderiza la escena, los controladores aparecen dentro del renderizado.
Resultado de la prueba	FALLO: Existe un fallo en la configuración de las gafas con Unity que genera un error en tiempo de ejecución

US-02-T02	
Título	Prueba de visualización del castillo con cámara SteamVR y las HTC Vive
Comportamiento esperado	Al mover la cabeza se simula el movimiento de la cámara que renderiza la escena, los controladores aparecen dentro del renderizado
Resultado de la prueba	OK

US-03-T01	
Título	Prueba de teletransporte
Comportamiento esperado	Al usar los controladores se permite el desplazamiento del jugador siempre que sea dentro de los límites del castillo y sobre una superficie plana. En caso contrario se cancela
Resultado de la prueba	OK

US-05-T01	
Título	Prueba de Integración con el arco
Comportamiento esperado	El arco se debe poder coger con ambos controladores. Al cogerlo aparecerá una flecha en la otra mano. Al acercar esta flecha al arco se posicionará para su tensado. Al soltar el botón de tensado se lanzará la flecha con la fuerza proporcionar al tensado. Debe chocar con el modelo del castillo
Resultado de la prueba	FALLO: Las flechas atraviesan el modelo del castillo, en vez de chocar con este.

US-05-T02	
Título	Prueba de Integración con el arco
Comportamiento esperado	Se incorpora el material para que las flechas colisionen con el modelo del castillo
Resultado de la prueba	OK

7.5.2. Sprint 2

Gráfico burn down

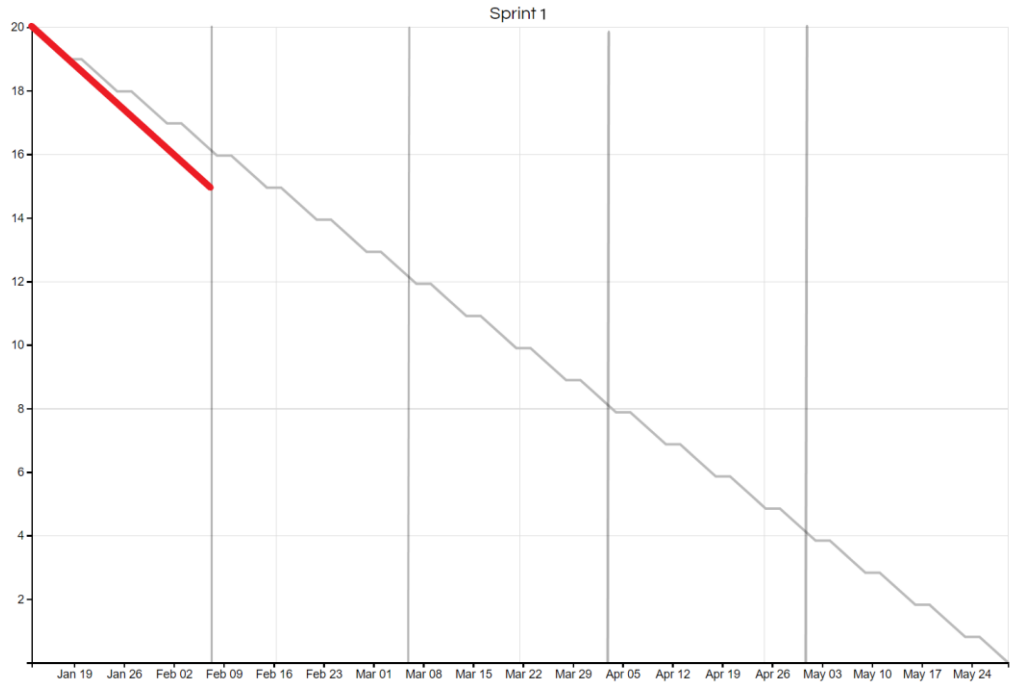


Figura 21: Burn Down Chart correspondiente al sprint 1

En la Figura 21 podemos observar las tareas correspondientes al sprint 1, se han completado todas dentro de los plazos estipulados en la planificación inicial.

Las historias de usuario que se implementarán durante el segundo sprint se describen a continuación en el sprint Backlog.

Sprint Backlog

- **US-04**
- **US-06**
- **US-07**
- **US-08**

Detalles de implementación

Para desarrollar la lógica de los zombis, los aspectos relacionados con la aparición, recorrido y llegada al punto de encuentro, se gestionarán mediante 2 objetos y la herramienta navigator. Un objeto actuará de “Spawn”, donde aparecerán los zombis. Mediante un script se gestiona que los zombis aparezcan en una ubicación aleatoria, delimitada por números máximos y mínimos calculados para que conformen un área rectangular.

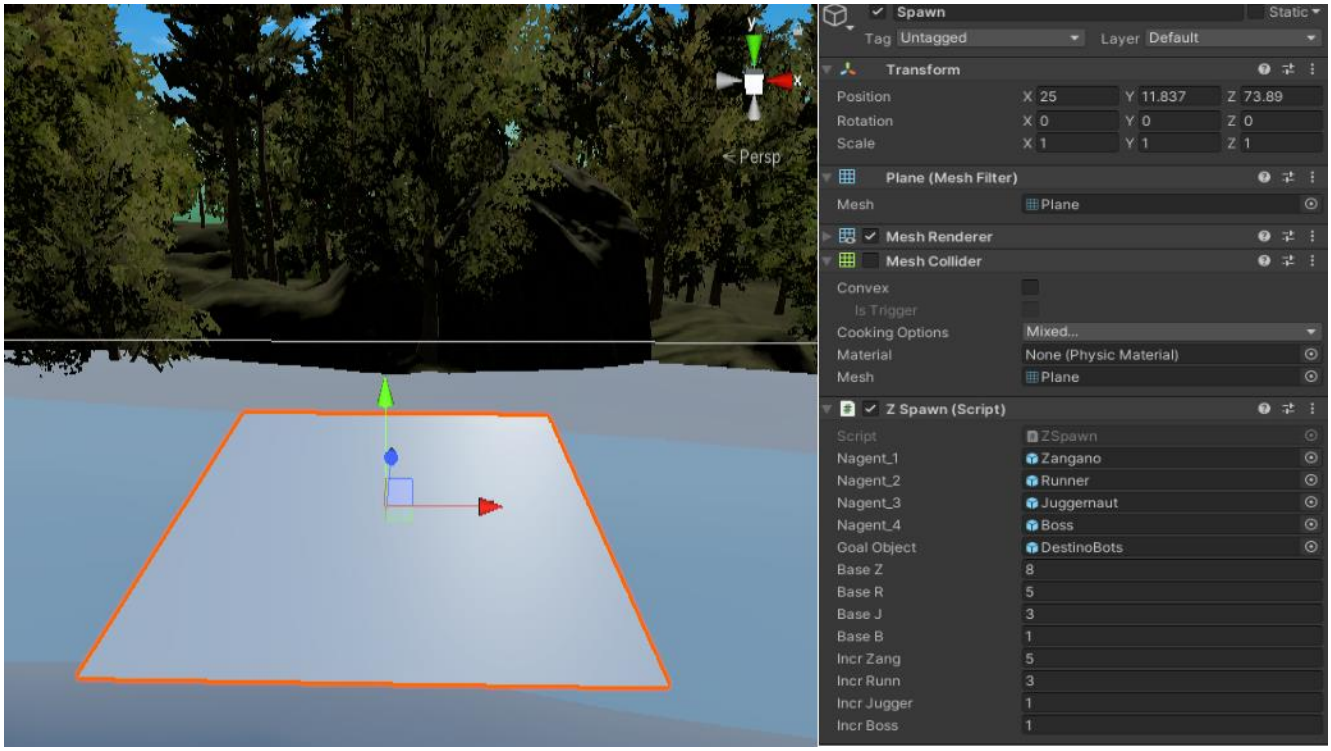


Figura 22: Objeto Spawn y sus componentes

El objeto Spawn será el encargado de instanciar los prefabs de los zombis en cada oleada. Para ello se tienen que definir como variables del script “ZSpawn”, los prefabs de los zombis que tiene que instanciar en tiempo de ejecución, y un objeto destino que se vincula con los prefabs de los zombis para que todos tengan el mismo punto de encuentro.

Aparte de estos objetos hay que configurar en los prefabs de los zombis, un componente “navigation agent”, para establecer las configuraciones del pathfinding. Estas configuraciones permitirán al zombi saber cuál es su destino y definir las propiedades físicas de su movimiento, tales como aceleración, velocidad, frenada, evitar obstáculos, rutas. La mayoría de estas configuraciones serán modificadas dinámicamente, según los eventos que ocurran en el juego que puedan influir en el cálculo de las rutas del zombi.

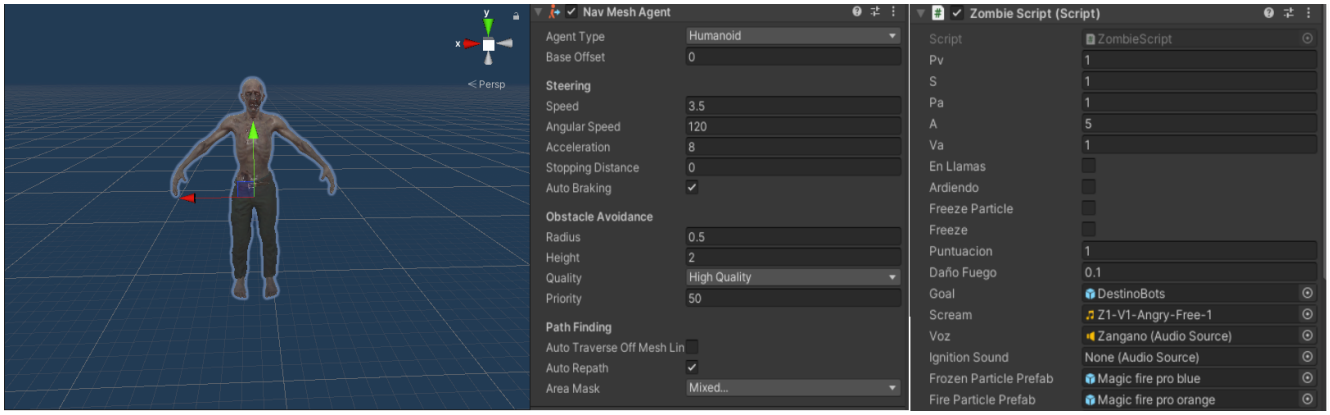


Figura 23: Componente Navigation Agent y configuración en objeto zombi

La herramienta navigator, interactúa con los componentes navigation agent de la escena, y calcula rutas de manera estática o dinámica para estos agentes, y de esta manera generar “caminos” que puedan ayudar a los zombis a recorrer la mínima distancia hacia el punto de encuentro. Se tienen que definir las mallas del castillo por las que los zombis podrán navegar. Se establece siguiendo estos criterios una superficie exterior al castillo que haga de camino. Para que los zombis puedan encontrar un nexo entre esta superficie y el interior del castillo, se tiene que establecer también como navegable la superficie interior del castillo y un plano que generamos entre la superficie exterior y la interior.

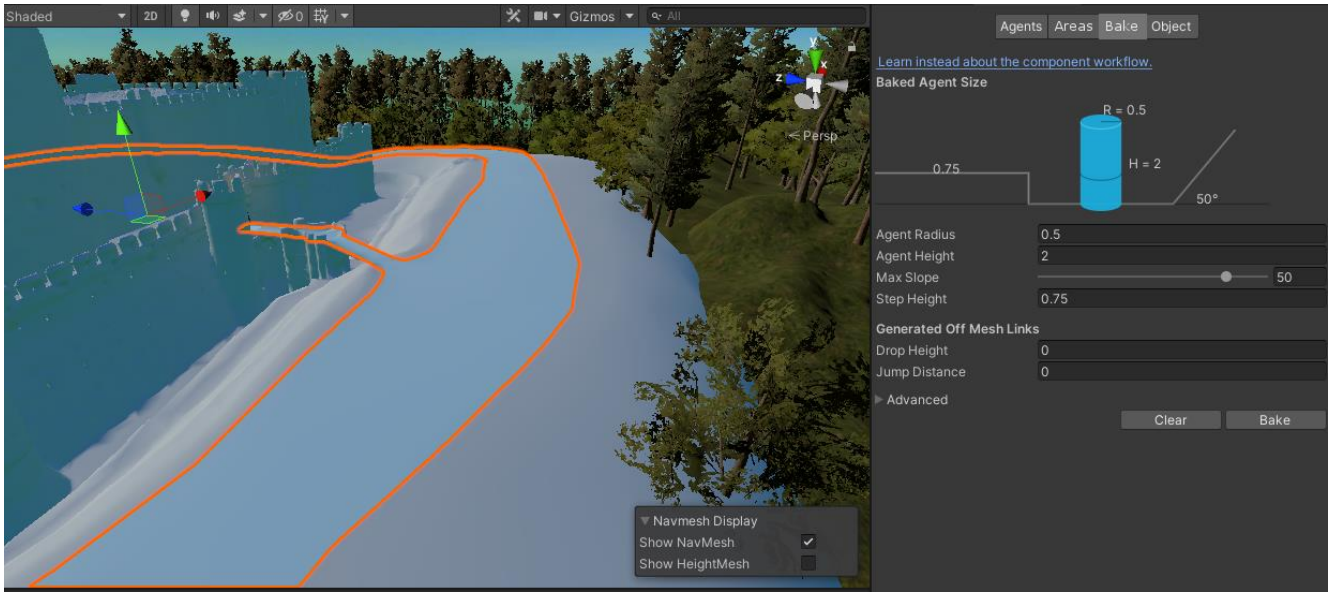


Figura 24: Herramienta Navigator y configuración para zombi

Los prefabs de los zombis inicialmente consistirán en un objeto cilindro de diferentes colores para diferenciar tipos de zombis. A estos objetos hay que añadirles los componentes necesarios para gestionar las propiedades del zombi, así como las futuras interacciones que vayan a tener. A parte del Navigation Agent, necesitarán un componente Collider y RigidBody, este último es un tipo específico de Collider, también sirve para que el objeto que lo contenga se le aplique las interacciones físicas, pero de una forma más básica. Simula la masa del objeto y como interactúan el resto de físicas con este. La idea es simular un entorno real para el cálculo de físicas e interacciones, y tener un script para controlar la lógica y almacenamiento de los datos de cada zombi.

La aplicación de los modelos de los zombis se realizará mediante la importación de las mallas que hemos obtenido de la Asset Store de Unity. Para el control de las propiedades del modelo, este asset necesita de un script adicional que incorporamos, y con el que podremos instanciar zombis con diferente ropa y modelo a partir de los prefabs, de esta forma generamos una masa de zombis más heterogénea y realista.

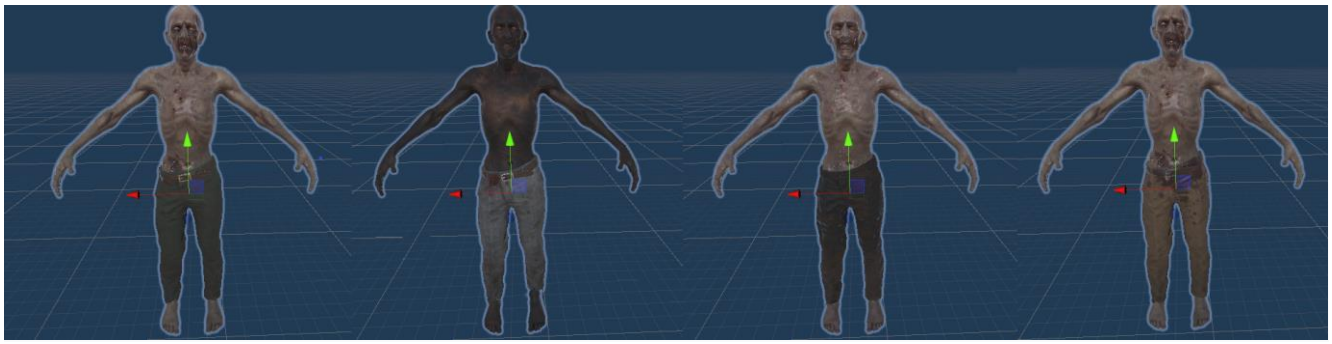


Figura 25: Ejemplos de variantes de Zanganos y Runners

Añadimos al script de los zombis como variables públicas, para poder modificarlas en el editor de Unity, todas las propiedades de los zombis como sus puntos de vida, velocidad, daño de ataque, etcétera.

También tenemos que añadir el evento onColliderEnter, que será el encargado de evaluar primero si el prefab ha sido impactado por una flecha que lancemos, y de hacer el reajuste en tiempo de ejecución para retocar sus propiedades, como aplicar el daño al objeto, o destruir el objeto cuando sus puntos de vida lleguen a cero.

Incorporamos a la escena un modelo de objeto que se parezca a una barricada, para simular la puerta al castillo, lo sacamos de la Unity Store. Le añadimos un RigidBody para que los zombis no puedan atravesarlo, y un Collider para que los zombis sepan cuando están en contacto con la puerta. Mientras los zombis permanezcan en contacto con la puerta se le aplicará, en el script de la puerta, un daño proporcional a la velocidad de ataque, y puntos de ataque del zombi. Si la puerta llega a cero puntos de vida se elimina de la escena permitiendo el paso a los zombis.

Pruebas de validación

US-04-T01	
Título	Prueba lógica de zombis I (instanciación)
Comportamiento esperado	Los zombis aparecen de manera aleatoria dentro de un área delimitada por un rectángulo.
Resultado de la prueba	OK

US-04-T02	
Título	Prueba lógica de zombis I (recorrido y parada)
Comportamiento esperado	Los zombis se dirigen a un destino, usando el camino más corto.
Resultado de la prueba	FALLO: Los zombis atraviesan objetos que no deberían poder atravesar, generando trayectorias incorrectas

US-04-T03	
Título	Prueba lógica de zombis I (recorrido y parada)
Comportamiento esperado	Se retoca la configuración de la herramienta “navigator” para generar trayectorias correctas
Resultado de la prueba	OK

US-06-T01	
Título	Prueba modelos de zombis
Comportamiento esperado	Se incorpora un modelo realista a los zombis.
Resultado de la prueba	OK

US-07-T01	
Título	Prueba lógica de zombis II (control de estadísticas)
Comportamiento esperado	Los zombis tienen en un script las propiedades que definen sus interacciones, poseen eventos asociados a colisionador, para gestionar su interacción.
Resultado de la prueba	OK

US-08-T01	
Título	Prueba barrera de acceso al castillo
Comportamiento esperado	Se incorpora un objeto para obstaculizar el paso directo de los zombis al castillo en la entrada principal.
Resultado de la prueba	OK

7.5.3. Sprint 3

Gráfico burn down

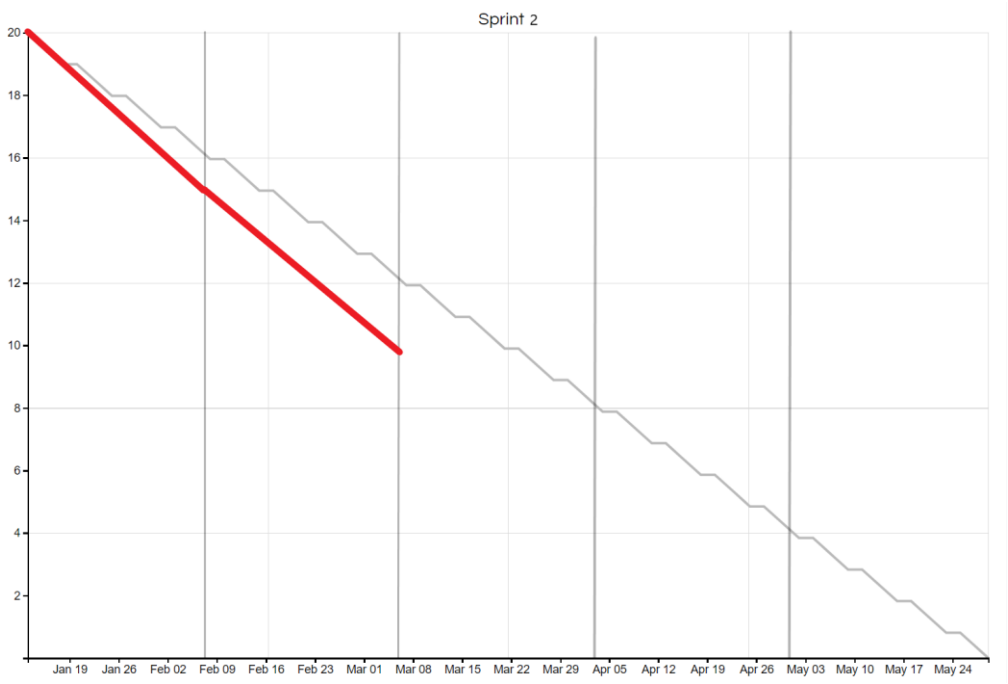


Figura 26: Burn Down chart correspondiente al sprint 2

En la Figura 26 se puede observar que se sigue cumpliendo la planificación inicial, aunque todavía quedan muchas tareas pendientes que se tendrán que tratar en los siguientes sprints para no desviarnos demasiado del tiempo estimado para completar el proyecto.

Durante el seguimiento del tercer sprint, se produce una situación excepcional de cuarentena decretada por el gobierno, a raíz de la crisis sanitaria a nivel mundial provocada por la enfermedad infecciosa COVID-19.

Debido a las restricciones de movilidad, y a la imposibilidad de trabajar desde casa, se decide durante este sprint, realizar una parada no planificada de la implementación.

Los desarrolladores podrán seguir avanzando en las historias de usuario que no necesiten de equipamiento especial desde su casa. Se añadirá una historia de usuario al final para integrar todo el

trabajo realizado y realizar modificaciones, hasta que las medidas de seguridad decretadas por el gobierno posibiliten el correcto desarrollo del proyecto.

Las historias de usuario que se implementarán se describen a continuación en el sprint Backlog.

Sprint Backlog

- **US-09**
- **US-10**
- **US-11**
- **US-12**

Detalles de implementación

Para el entorno podemos utilizar unos elementos de terreno con modelos de árboles que ya vienen preconfigurados de la Asset Store. De igual manera hacemos con el objeto “skybox”, que nos encerrará en un cubo con materiales que se asemejen a un entorno abierto con cielo.

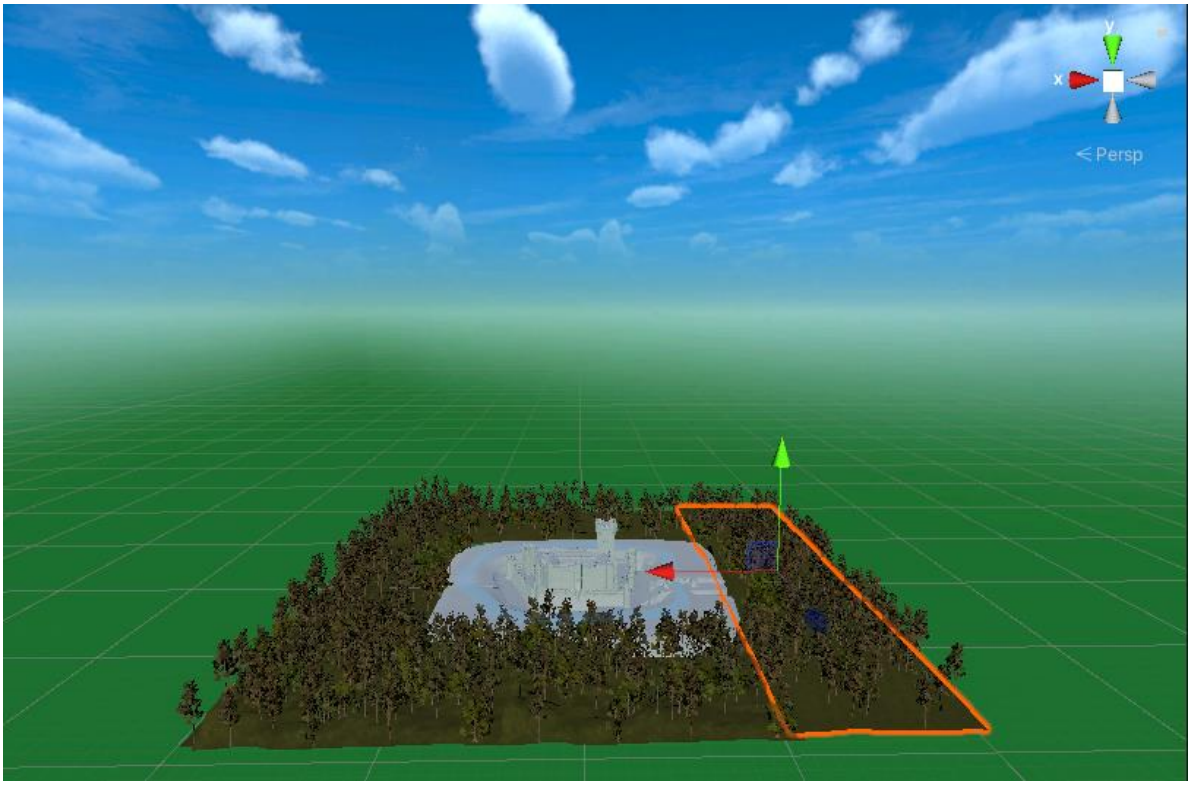


Figura 27: Escena con skybox y objeto árboles para el entorno seleccionado

Las flechas de hielo deberán actuar de la misma forma que las flechas normales con algunas interacciones adicionales, por lo que vamos a copiar los prefabs de las flechas y vamos a aplicarle los cambios pertinentes.

Modificamos el material básico de la nueva flecha con tonos más azulados para diferenciarla de la flecha básica. Tenemos que añadir la funcionalidad para limitar el número de flechas de hielo que podemos instanciar, evitar que se prendan al entrar en contacto con fuentes de fuego, y que se reduzca en una unidad cada vez que se dispara una flecha.

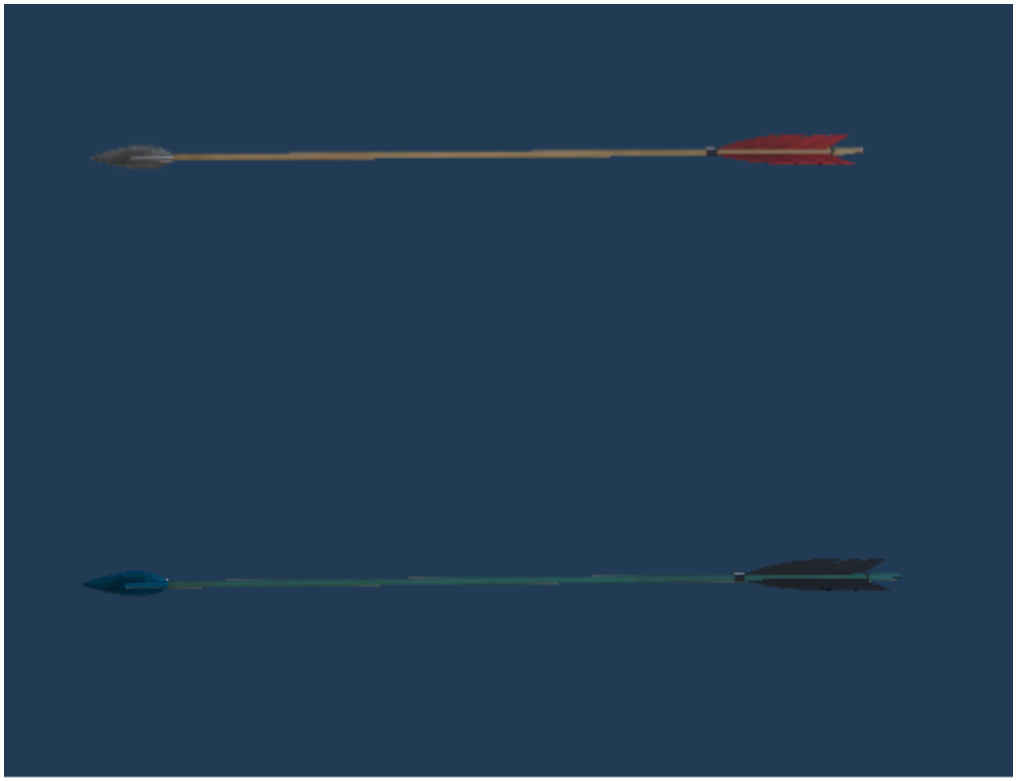


Figura 28: Flecha normal y flecha de hielo

El script de los zombis se debe modificar para que contemple los cambios en la velocidad del objeto que recibe un impacto de una flecha de hielo. También hay que añadir un sistema de compra de flechas basadas en puntuación, lo implementamos usando un panel, que de momento, nos permita añadir flechas de hielo al inventario del jugador.

Los controladores deben de tener la capacidad de permitir al jugador cambiar de tipo de flecha, añadimos esta funcionalidad mediante la posición de contacto del touchpad. Si tocamos la parte derecha del touchpad usaremos una flecha de hielo, mientras que si tocamos la parte izquierda aparecerá una flecha normal. Hay que destacar que los controladores no disponen de demasiados botones, pero sí que hay diferencia entre tocar el touchpad, que registra la posición exacta en coordenadas cartesianas de toque, y presionar el touchpad, que se comporta como un botón independientemente de la posición exacta en la que presionemos.

El sistema de teletransporte rápido es esencial para aportar fluidez a los juegos que utilicen realidad virtual. De esta manera podemos evitar la pérdida de mecánicas de movimiento que suele estar relacionada con controladores de realidad virtual. Al no disponer de espacio físico real para podernos desplazar por escenarios muy grandes, los teletransportes de larga distancia se convierten en una mecánica muy práctica. Desplazarnos de manera manual o utilizando el sistema de teletransporte normal podría generar una sensación de pesadez a la hora de jugar.

Utilizamos el mismo modelo del castillo para miniaturizarlo, y le introducimos un script para controlar los puntos de control dentro de una lista circular, utilizando la propiedad “localposition”, podemos mapear la posición real del castillo en el contexto de la escena, dentro de la miniatura del castillo. Añadimos marcadores a esta miniatura que irán marcando las posiciones del castillo. Al cerrar el mapa si se ha elegido una posición diferente el jugador es transportado a esta.

El mapa, en el momento de aparecer, debe marcar la posición del jugador y con los gatillos de los controladores, recorrerá la lista de puntos de teletransporte.

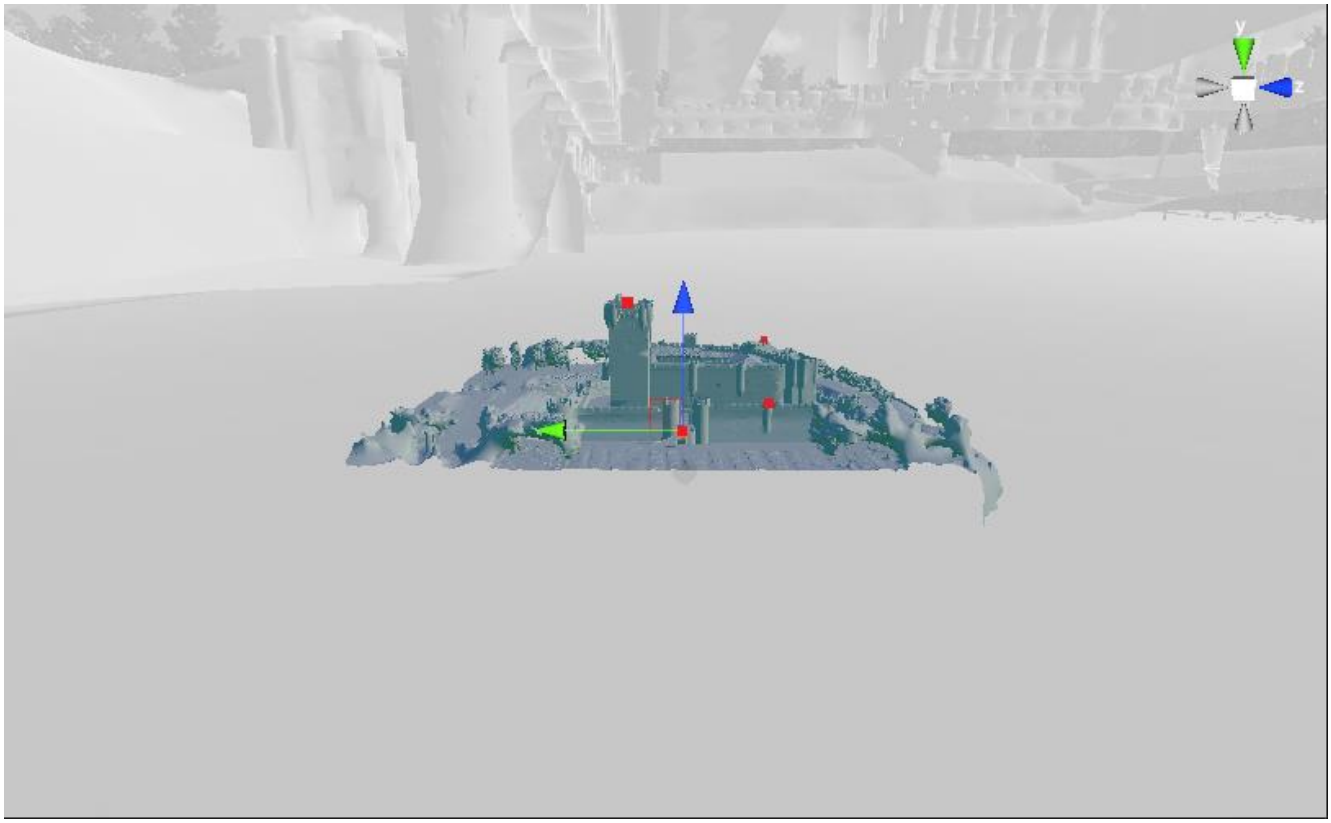


Figura 29: Minimapa y marcadores de teletransporte

También se controla la posición del mapa a la hora de aparecer, para que siempre aparezca enfrente del jugador y a una altura que le permita verlo en su totalidad, sin problemas.

Generamos un objeto que solo va a contener el script que se encargará de la persistencia de los datos de la partida. Antes de mantener estos datos en un sistema persistente más consolidado, como una base de datos, utilizaremos un script para poder mantener los datos de cada partida y que estos sean persistentes entre todas las escenas del juego.

Pruebas de validación

US-09-T01	
Título	Prueba de entorno
Comportamiento esperado	Los alrededores del castillo contienen terrenos con elementos típicos de bosque, solo decorativos. La escena está encerrada en un skybox para simular el cielo de día.
Resultado de la prueba	OK

US-10-T01	
Título	Prueba de flechas de hielo
Comportamiento esperado	El jugador puede hacer aparecer un tipo de flecha especial cuyo comportamiento en físicas es similar a las flechas normales, pero tendrán un efecto en las estadísticas del zombi que golpeen.
Resultado de la prueba	FALLO: Las flechas no activan el evento específico en los zombis.

US-10-T02	
Título	Prueba de flechas de hielo
Comportamiento esperado	Las flechas tienen que reducir la velocidad del zombi y aplicar el daño de una flecha normal
Resultado de la prueba	OK

US-11-T01	
Título	Prueba teletransporte rápido de larga distancia
Comportamiento esperado	El jugador podrá desplegar una reproducción del castillo en miniatura con una lista circular de puntos que pueda iterar para elegir el punto de teletransporte
Resultado de la prueba	FALLO: los marcadores no aparecen en el mapa

US-11-T02	
Título	Prueba teletransporte rápido de larga distancia
Comportamiento esperado	Los marcadores aparecen ajustados en los puntos correctos del mapa
Resultado de la prueba	OK

US-12-T01	
Título	Prueba sistema de puntuaciones y persistencia
Comportamiento esperado	Un objeto del juego contiene un script que no es destruido al cambiar de escena en donde se almacena los datos de la partida
Resultado de la prueba	FALLO: El objeto es destruido al cambiar de escena

US-12-T02	
Título	Prueba sistema de puntuaciones y persistencia
Comportamiento esperado	El objeto debe ser persistente entre escenas de las misma partida
Resultado de la prueba	OK

7.5.4. Sprint 4

Gráfico burn down

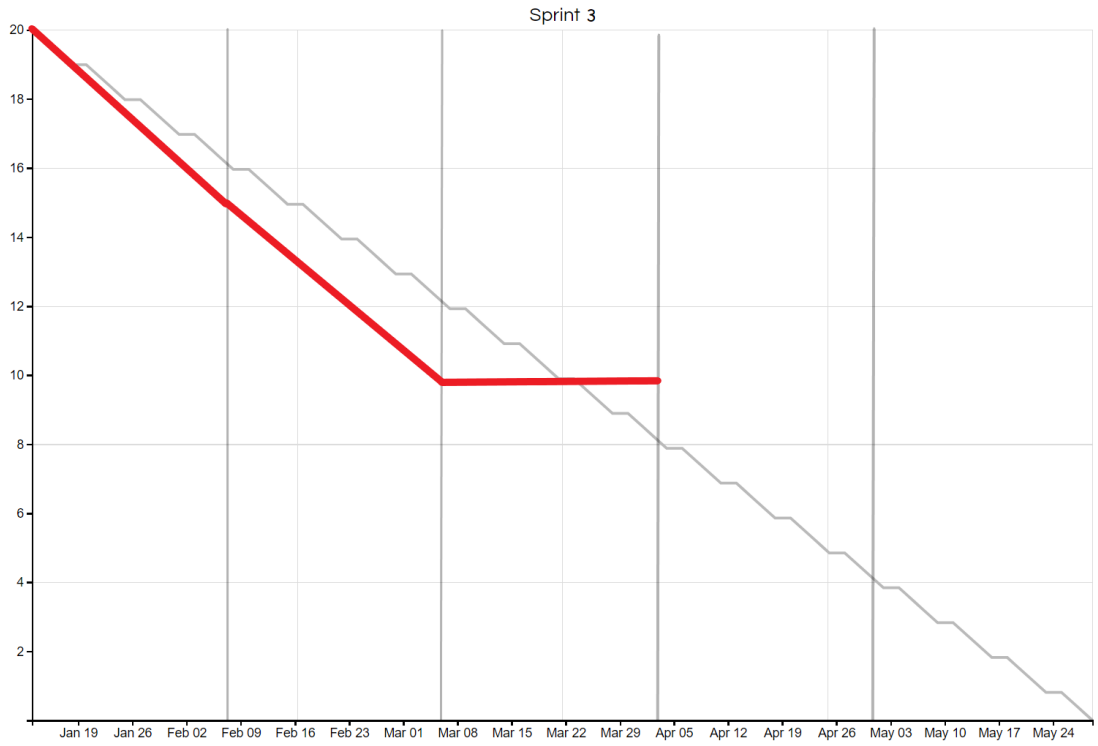


Figura 30: Burn Down chart correspondiente al sprint 3

Debido a las restricciones de movilidad no se ha podido hacer uso del equipamiento de realidad virtual para el desarrollo de las historias de usuario correspondientes al sprint 3. Se decide seguir avanzando con el desarrollo de historias que correspondan, siguiendo la planificación inicial. La integración de todas las historias, pruebas e implementaciones adicionales se realizarán cuando se levanten las restricciones de movilidad.

Sprint Backlog

- **US-13**
- **US-14**
- **US-15**
- **US-16**

Detalles de implementación

Siguiendo las especificaciones del GDD, tenemos que mantener un equilibrio entre rondas de manera que la dificultad del juego sea incremental oleada tras oleada.

Usaremos un panel para gestionar el control de oleadas, el panel dispondrá de un botón asociado a una función, que evaluará si todavía hay zombis en la escena y no es el inicio de la primera ronda. Si todavía quedan zombis, significa que la oleada no ha terminado, por lo que el botón estará

deshabilitado. Cuando se pueda comenzar una nueva oleada se evalúa que oleada corresponde y se comunica con el script de Spawn de zombis para mandar una orden de inicio de oleada, en el script de este objeto se verifica que oleada corresponde, y el número y tipo de zombis que generar.

Se diseña una escena específica introductoria. Esta tendrá un formulario en el que le aparecerá la última puntuación obtenida, si no es su primera partida, un campo para poner su nick y botones para acceder al juego o a la escena tutorial.

En la escena tutorial, se introduce mediante paneles informativos, los controles básicos del juego.

Para las animaciones de los zombis se optará por descargar animaciones de fuentes externas, que se importarán en Unity como objetos “fbx”, formato estándar para assets importados de Unity. La lógica de las animaciones se basará en una máquina de estados que alternará tres tipos de animación diferente, correr, atacar y morir.

La gestión de las transiciones de estas animaciones, se gestionará mediante gatillos activables en los scripts de los zombis.

Pruebas de validación

US-13-T01	
Título	Prueba sistema de control de oleadas
Comportamiento esperado	El jugador es capaz de iniciar una nueva oleada una vez haya terminado la anterior. Está debe hacer aparecer un número y tipo específico de zombis incremental
Resultado de la prueba	FALLO: Se generan los zombis demasiado rápido generando colisiones entre ellos. Se puede acceder a una oleada superior sin haber terminado la actual.

US-13-T02	
Título	Prueba sistema de control de oleadas
Comportamiento esperado	Los zombis aparecerán con un delay para evitar colisiones entre ellos al aparecer
Resultado de la prueba	OK

US-13-T03	
Título	Prueba sistema de control de oleadas
Comportamiento esperado	El botón para empezar una oleada estará deshabilitado mientras la oleada anterior siga activa.
Resultado de la prueba	OK

US-14-T01	
Título	Prueba de lógica del juego
Comportamiento esperado	Cuando los zombis lleguen a su destino se destruirán y se perderán puntos de vida proporcionalmente al tipo de zombi que llegue. Todos los zombis que sean eliminados por el jugador añadirán una puntuación al jugador en función del tipo de zombi.
Resultado de la prueba	OK

US-15-T01	
Título	Prueba escenas introductorias y formulario
Comportamiento esperado	El jugador debe ser capaz de poner su nick en un formulario, y acceder al tutorial o al juego directamente. Si no es la primera partida en la escena inicial se mostrará la puntuación correspondiente a la última partida.
Resultado de la prueba	OK

US-16-T01	
Título	Prueba de animaciones de los zombis
Comportamiento esperado	Los zombis tendrán 3 diferentes animaciones: Una para su movimiento, otra para el ataque y una última para su muerte. Estas animaciones variaran en función del tipo de zombi.
Resultado de la prueba	FALLO: las animaciones no cuadran con el movimiento del modelo.

US-16-T02	
Título	Prueba de animaciones de los zombis
Comportamiento esperado	Ajustando los parámetros de las animaciones, estas deben cuadrar con los modelos para aumentar su realismo.
Resultado de la prueba	OK

7.5.5. Sprint 5

Gráfico burn down

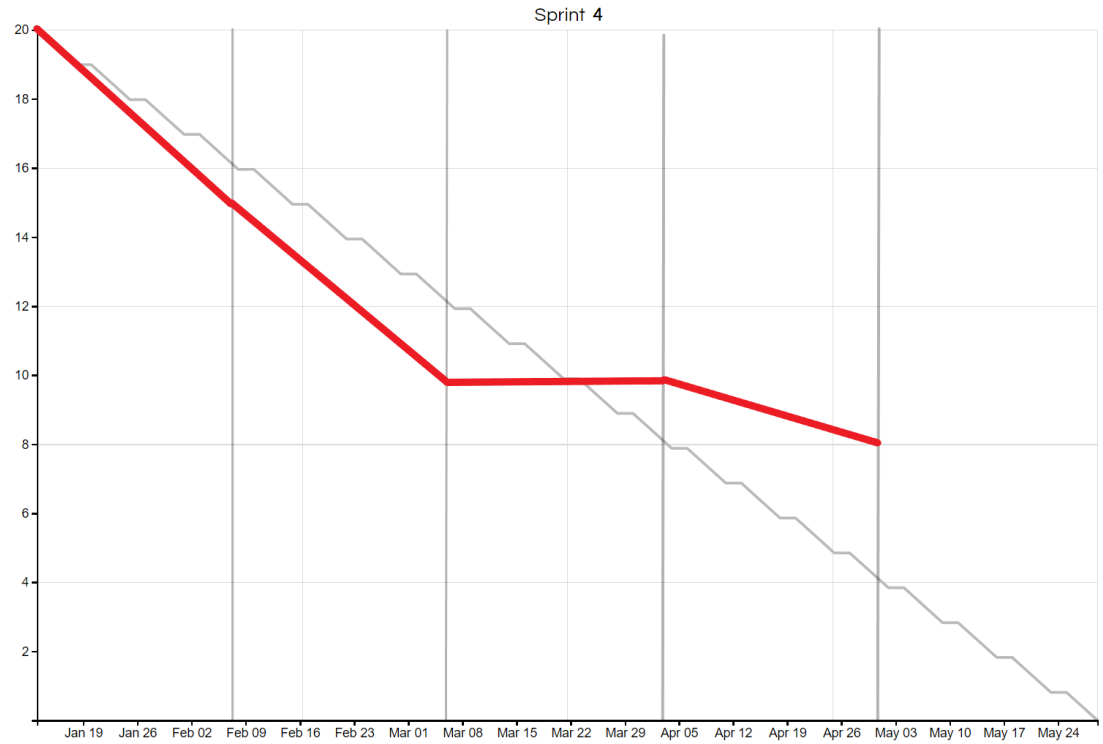


Figura 31: Burn Down chart correspondiente al sprint 4

Se han podido completar 2 historias de usuario, el resto de las que corresponden con el 4º sprint no se han podido realizar por completo. De nuevo, se implementarán cuando se disponga del equipamiento necesario.

Sprint Backlog

- **US-17**
- **US-18**
- **US-19**
- **US-20**

Detalles de implementación

Para el sistema de combate, se tienen que tener en cuenta varios aspectos: primero hay que diferenciar en el script de los zombis, en los eventos que controlan el collider del zombi, el tipo de objeto que impacta con el objetivo. Si es una flecha, aplicar daño, pero si la flecha es del hielo, hay que aplicar también una reducción permanente de la velocidad. Esta reducción de velocidad hay que aplicársela al componente “navigation agent”, para que los movimientos alterados del zombi no parezcan demasiado forzados y que no se desajuste la ruta calculada por la herramienta “navigator”. Igualmente si la flecha está incendiada, debe aplicar un efecto de daño en el tiempo permanente. Para calcular correctamente el daño aplicado en el tiempo del fuego, utilizaremos una corrutina para que vaya incrementando un contador de tiempo, a estos contadores les llamaremos “ticks”. Cada uno de estos ticks aplicará una reducción en la vida del objeto impactado, en función del tiempo del sistema.

Se aplica una banda sonora al juego para que reproduzca la misma música de fondo. Para ello se utilizará un componente audio source dentro de un objeto vacío cuya área de escucha sea uniforme, y albergue toda la escena.

Se incorporará de la misma manera que con la banda sonora, clips de audio, que sean generados de componentes audiosource dentro de los propios prefabs de los zombis para que reproduzcan un sonido al atacar. Se considera mantener un área global de escucha en toda la escena, para que el jugador tenga un feedback adecuado mediante estos sonidos, de las interacciones con el zombi, independientemente de lo lejos que se encuentre de este.

Pruebas de validación

Se realizarán en el último sprint junto con el resto de pruebas.

7.6. Evaluación del producto final

Tras la finalización del 5 y último sprint, en este epígrafe se describe el estado actual del proyecto.

Se han finalizado todas las historias de usuario descritas en el apartado 6.3 por lo que el proyecto se puede considerar concluido.

En el siguiente gráfico burn down se muestra el estado final del proyecto según la planificación inicial.

Gráfico burn down

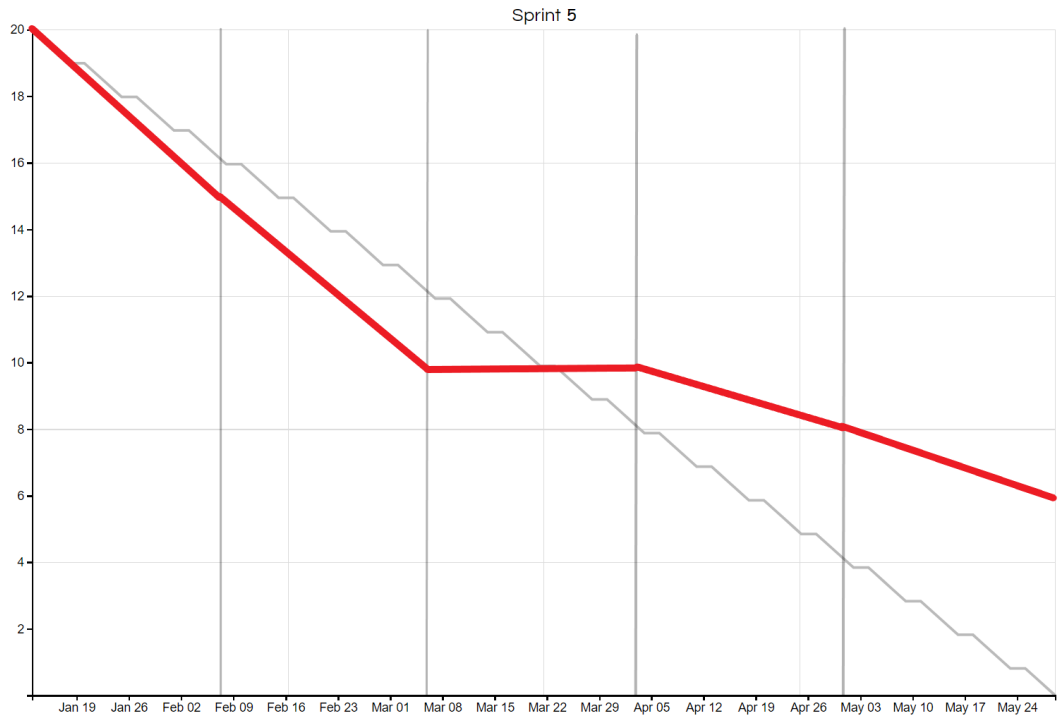


Figura 32: Burn Down chart correspondiente al sprint 5

Como puede observarse en la Figura 32, no se ha podido cumplir con la planificación inicial.

Quedan todavía más de 6 historias de usuario que implementar, y faltan pruebas unitarias y de integración de otras 4 historias de usuario.

No se ha podido cumplir con la planificación inicial a causa de la situación excepcional de cuarentena, provocada por la enfermedad COVID-19. Al no poder hacer uso del equipamiento especial

necesario para el desarrollo de este videojuego en realidad virtual, se ha producido una desviación de 7 meses de los periodos previstos en la planificación inicial.

Esta situación no se recoge en el análisis de riesgos por lo que no existía ningún plan de contingencia para evitar el estancamiento del proyecto. Sin embargo, se trata de una situación excepcional que no es posible predecir. Se ha improvisado un plan de contingencia durante el tercer sprint como se indica en el punto

US-08-T01	
Título	Prueba barrera de acceso al castillo
Comportamiento esperado	Se incorpora un objeto para obstaculizar el paso directo de los zombis al castillo en la entrada principal.
Resultado de la prueba	OK

7.5.3. Sprint 3, en el cual se permitía avanzar en el domicilio, en las tareas que no requirieran de equipamiento especial. A pesar de esto, dado que prácticamente cualquier historia de usuario necesita una validación, usando las gafas de realidad virtual, no se ha podido avanzar lo suficiente como para que la desviación de la planificación inicial fuera pequeña.

7.6.1. Sprint final

Sprint Backlog

- **US-17**
- **US-18**
- **US-19**

Se decide realizar un Sprint extra para todas las historias de usuario que faltan, corregir algunos errores detectados en estas historias y en la integración, y por último para realizar la fase de pruebas. Para este último sprint se realizará una planificación adicional, que cuente con todas las historias que queden por implementar y se añadirán las tareas de pruebas.

Este sprint recoge todas las historias de usuario que no se han podido realizar en los sprints anteriores, así como la validación de las pruebas unitarias y un apartado extra con la fase de “beta testing”.

La fecha prevista de inicio de este sprint es el 1 de Octubre de 2020. Y la fecha de finalización, el 29 de Diciembre de 2020.

Debido a la situación general de la pandemia, no es posible realizar un “beta testing”, pero se explicará en este último sprint los requisitos y la técnica necesaria para su realización.

Pruebas de validación

US-17-T01	
Título	Prueba sistema de combate
Comportamiento esperado	Las flechas que entren en contacto con los colisionadores de los zombis aplicarán un daño en sus puntos de vida. Si la flecha está incendiada aplicarán daño en el tiempo permanente.
Resultado de la prueba	FALLO: El daño de fuego no aplica daño en el tiempo.

US-17-T02	
Título	Prueba sistema de combate
Comportamiento esperado	El fuego debe aplicar ticks de daño en función de delta time para que sea daño en tiempo constante
Resultado de la prueba	OK

US-18-T01	
Título	Prueba banda sonora
Comportamiento esperado	Se añade una fuente de audio que el jugador pueda escuchar en bucle a un volumen que no le distraiga.
Resultado de la prueba	OK

US-19-T01	
Título	Prueba efectos de sonido
Comportamiento esperado	Los zombis reproducen sonidos en determinadas ocasiones como cuando son destruidos, o cuando están atacando.
Resultado de la prueba	FALLO: Los zombis generan gritos de manera constante

US-19-T02	
Título	Prueba efectos de sonido
Comportamiento esperado	Los sonidos deben de reproducirse al cumplirse una condición o entrar en un evento específico.
Resultado de la prueba	OK

7.6.2. Diagrama de clases y Objetos principales

En la Figura 33, se muestra un diagrama de clases, que incluye, los objetos más relevantes de la escena principal, y la comunicación de los scripts entre los diferentes objetos de la escena. Esta comunicación define las mecánicas principales del videojuego.

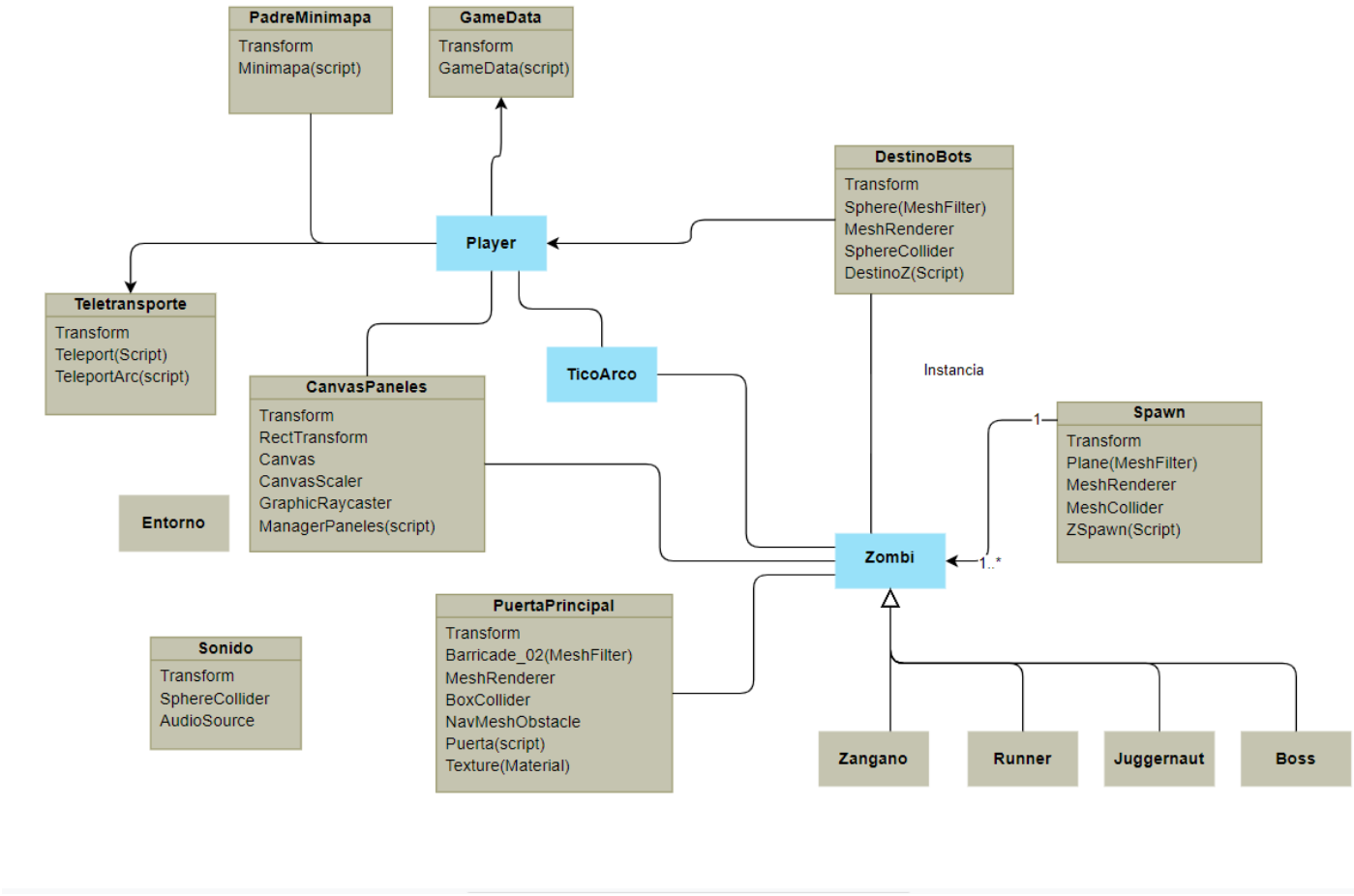


Figura 33: Diagrama de clases

Los objetos principales del videojuego y sus componentes se detallan en las siguientes figuras.

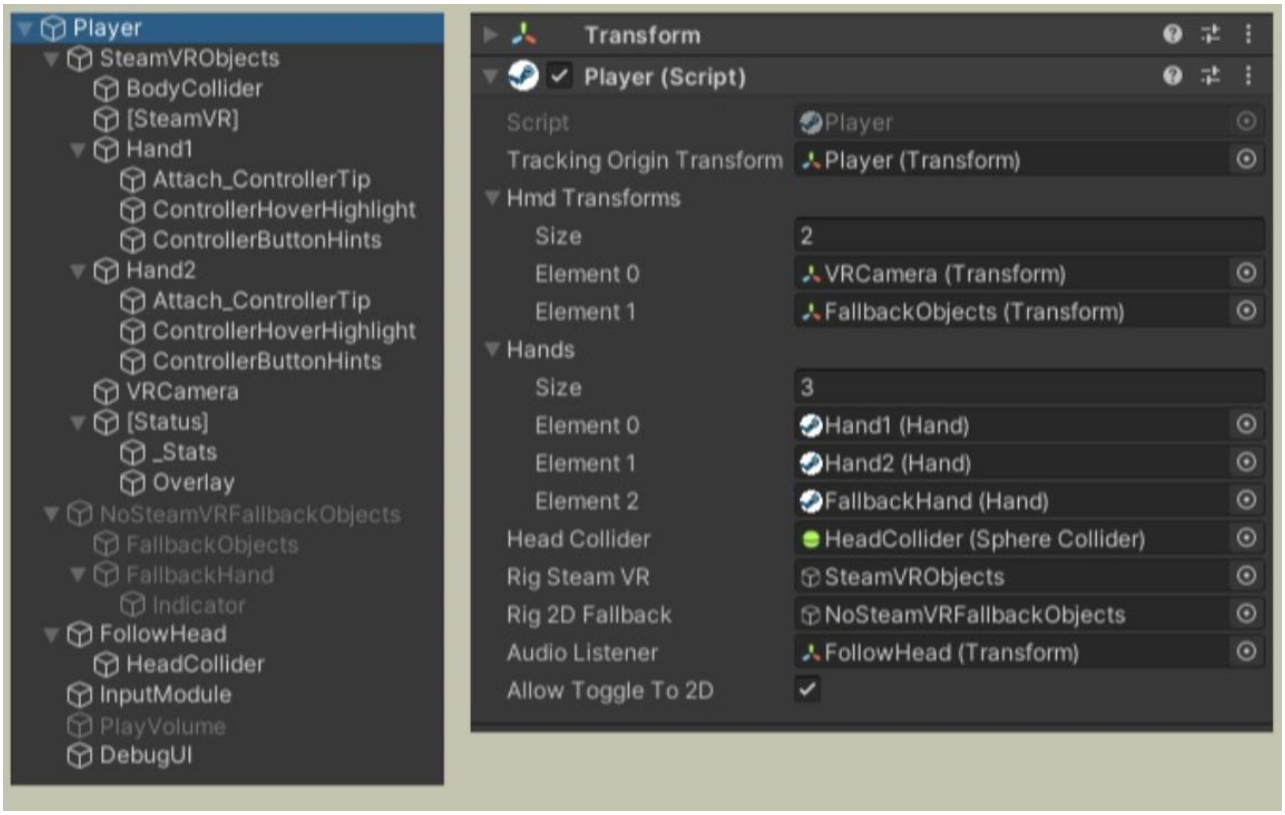


Figura 34: Componentes de los objetos de la jerarquía de Player

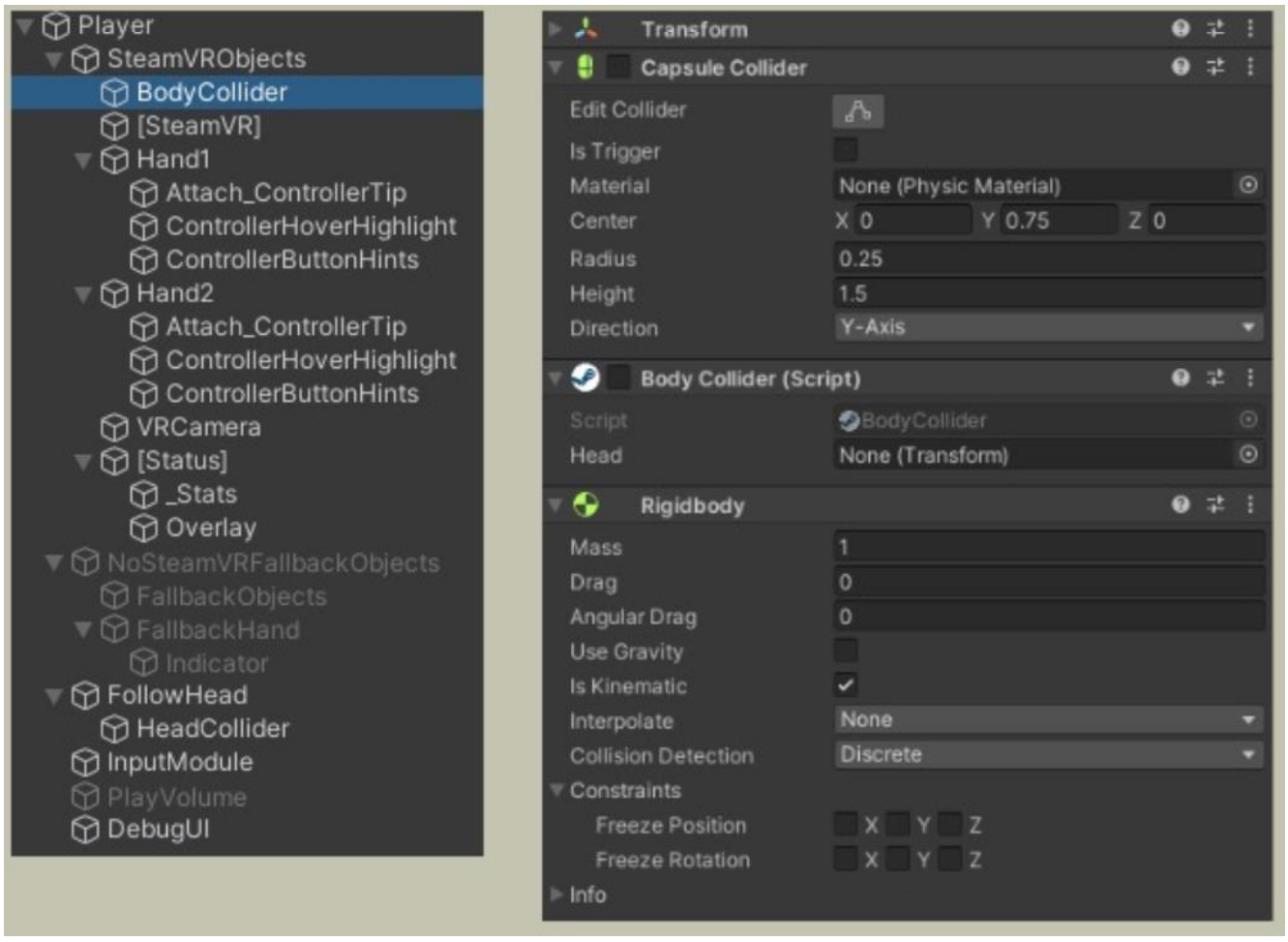


Figura 35: Componentes del objeto BodyCollider

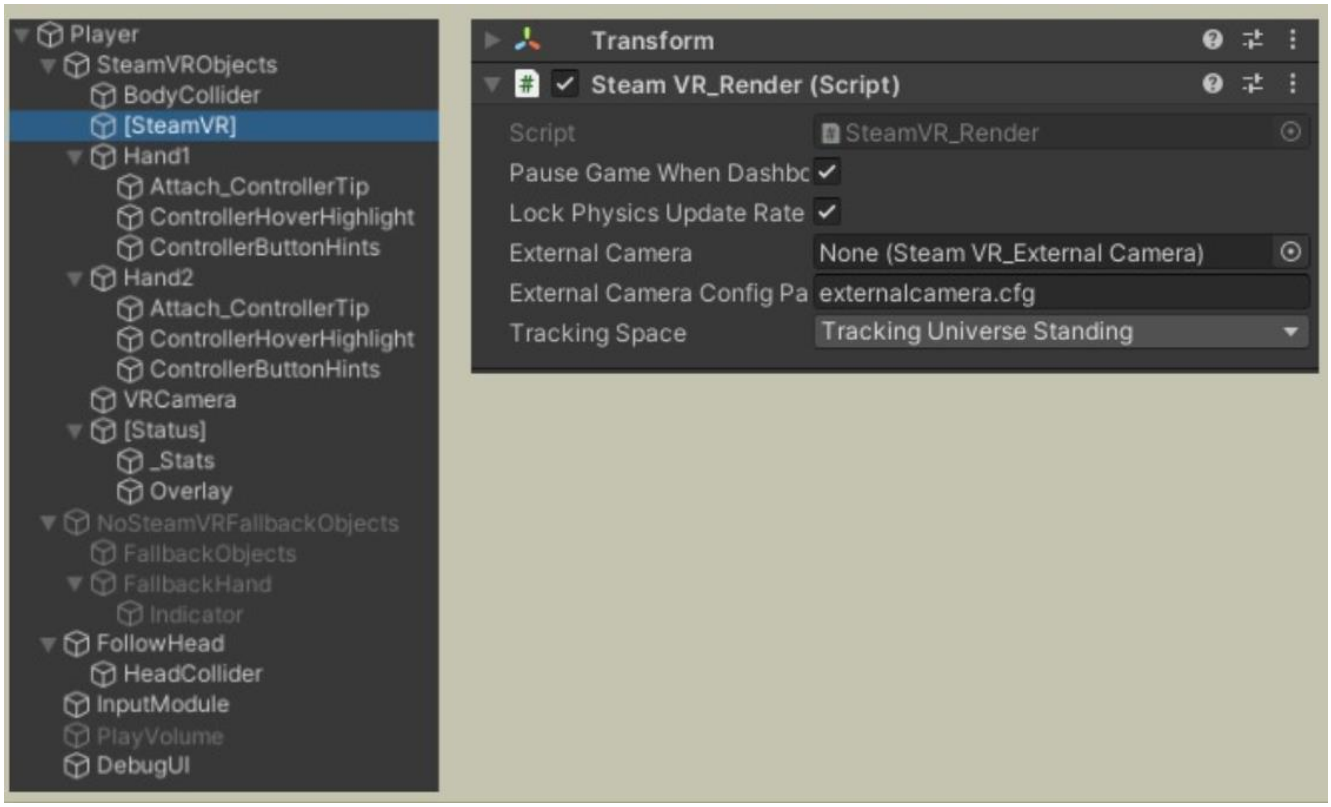


Figura 36: Componentes del objeto SteamVR

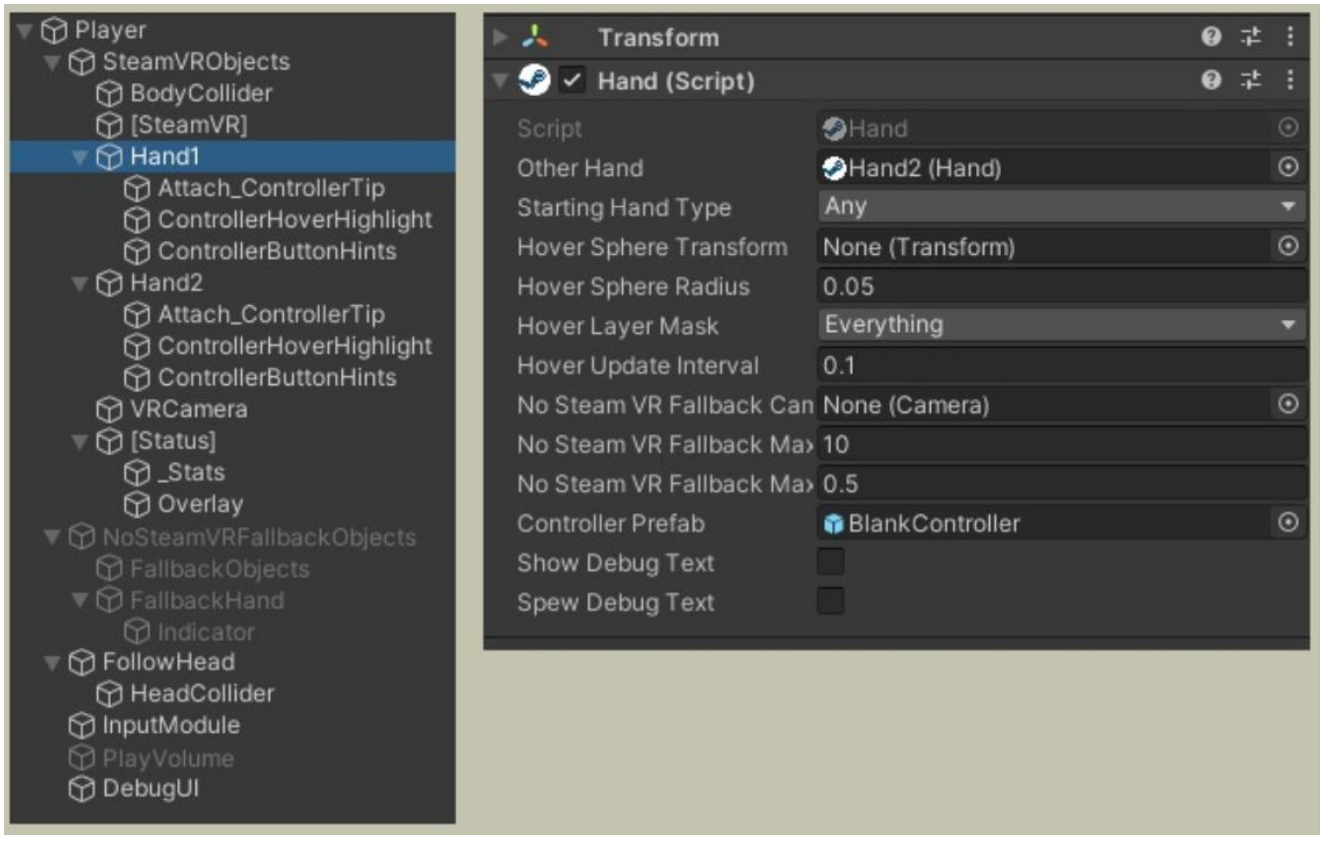


Figura 37: Componentes del objeto Hand1

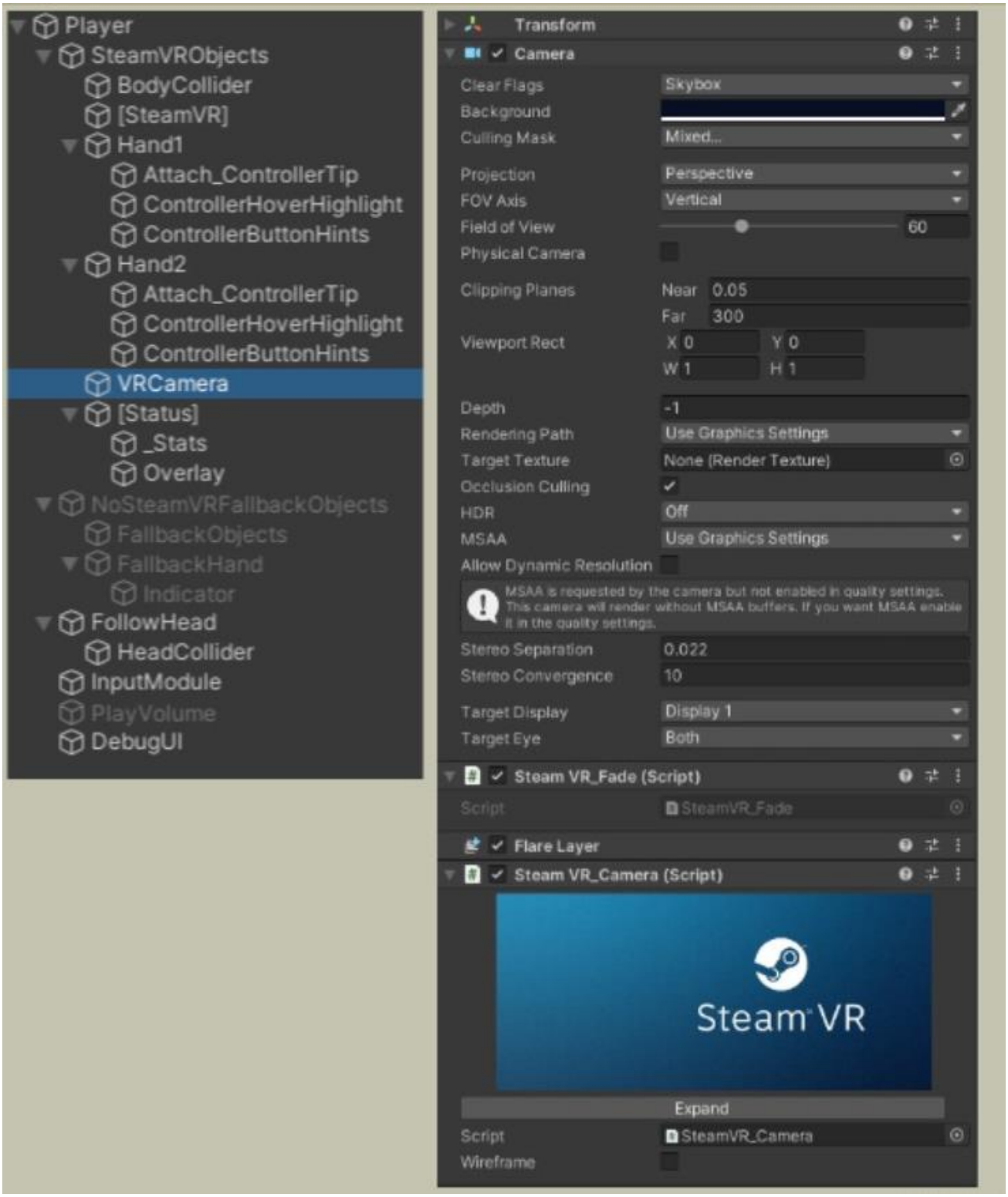


Figura 38: Componentes del objeto VRCamera

En las siguientes figuras se describen los componentes y la jerarquía de objetos TiroArco, que agrupa todas las partes del arco y define su funcionamiento.

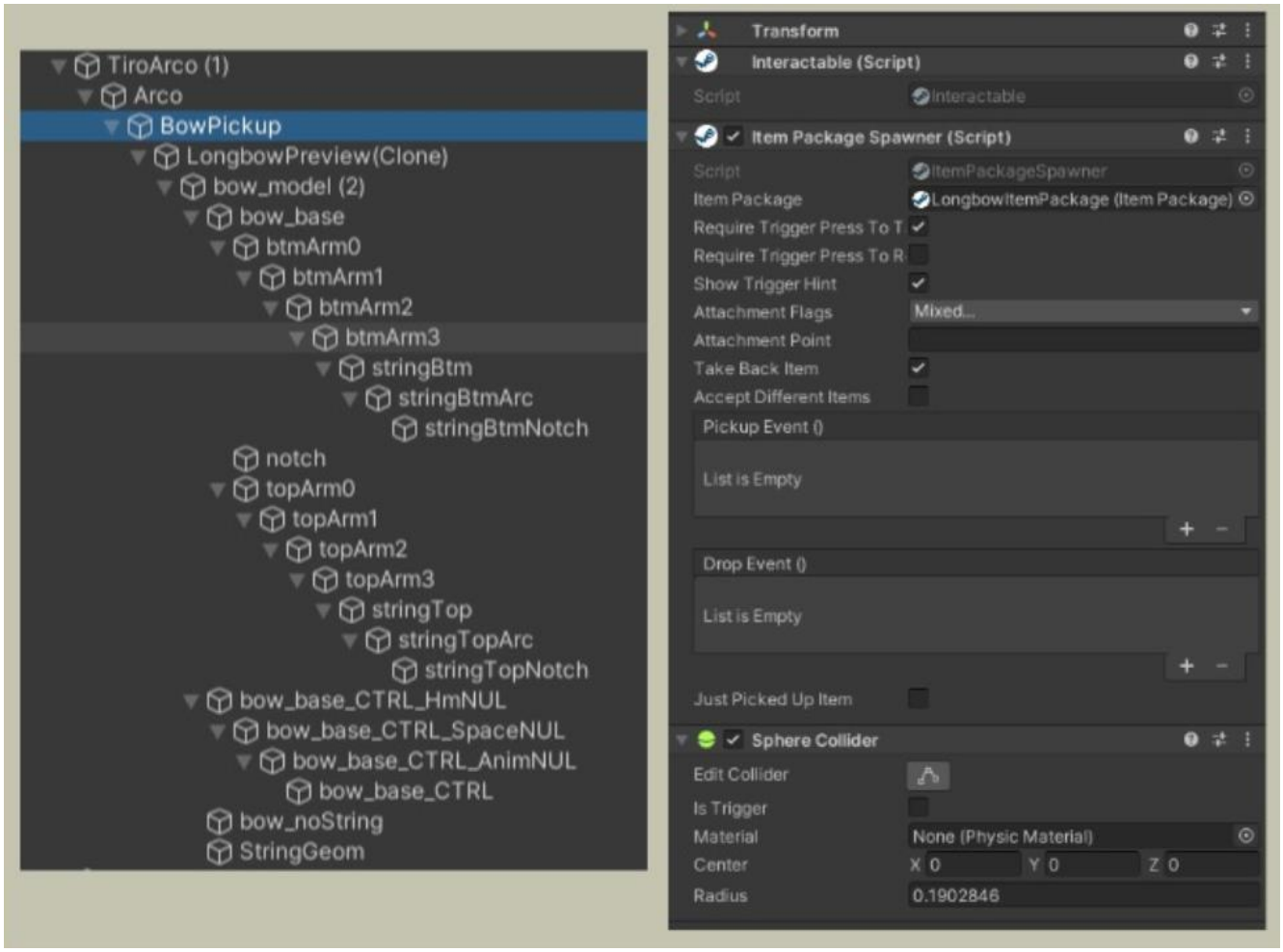


Figura 39: Componentes del objeto BowPickup, objeto principal del arco

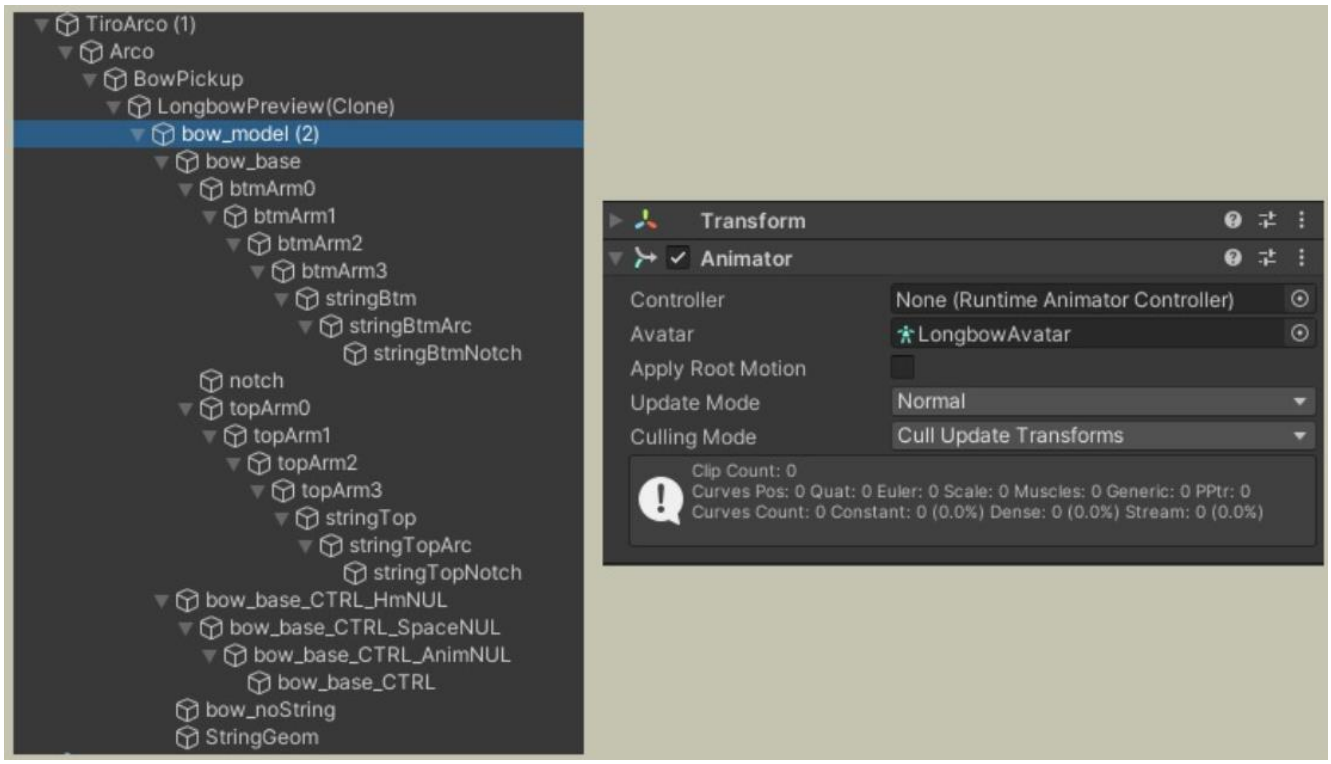


Figura 40: Componentes del objeto bow_model

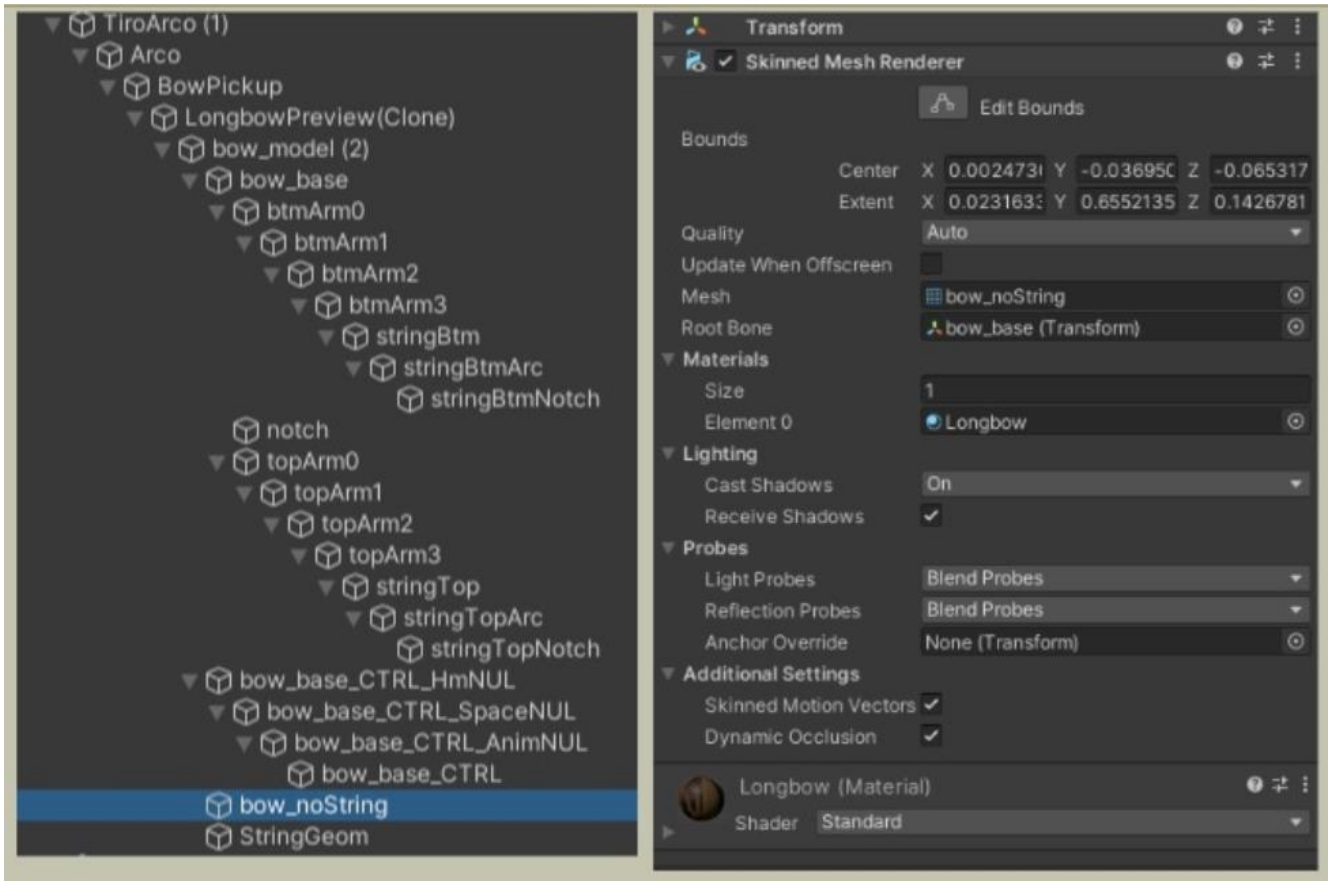


Figura 41: Componentes del objeto bow_noString

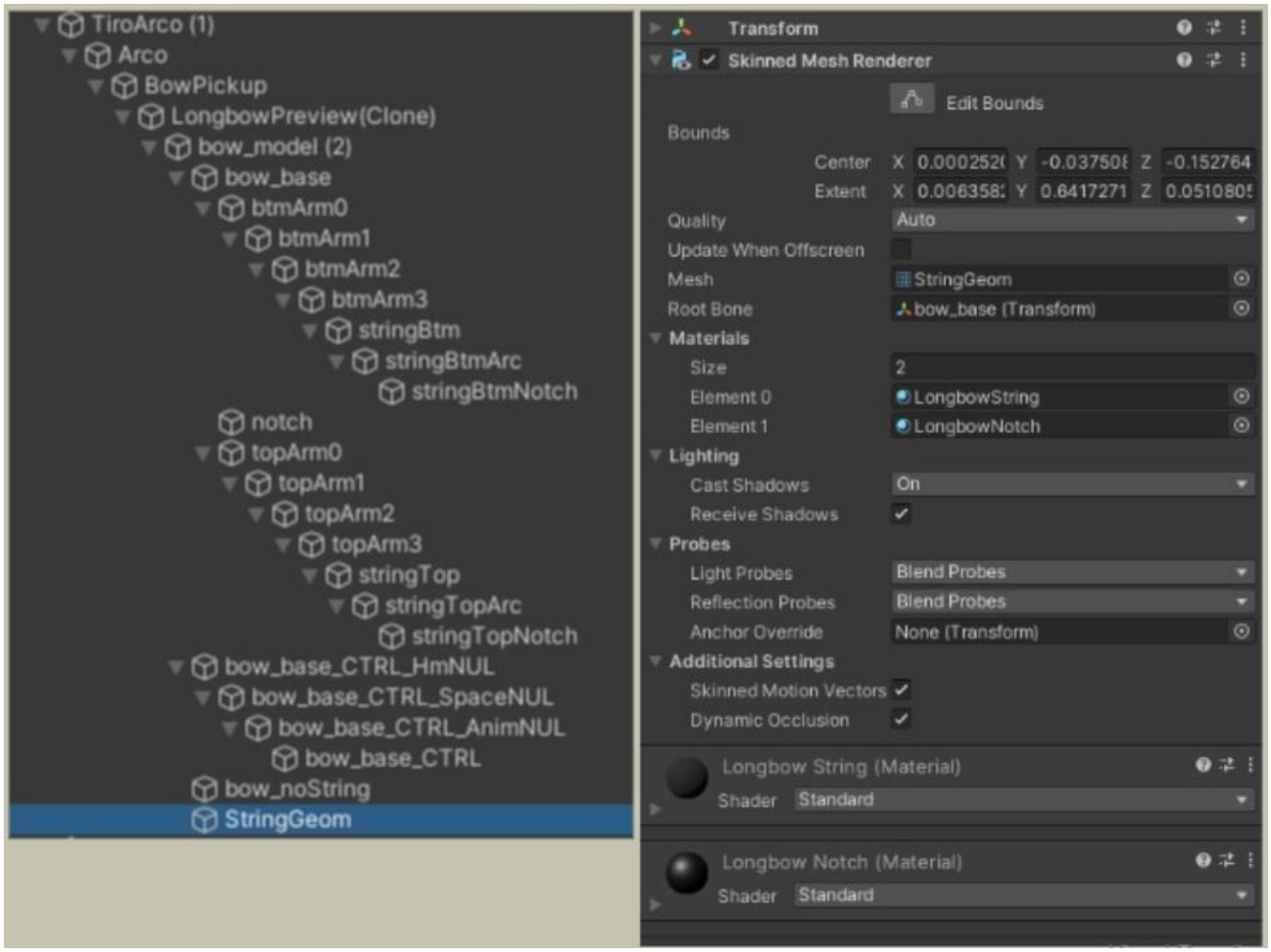


Figura 42: Componentes del objeto StringGeom

Para el objeto zombi, dado que cada tipo de zombi es un objeto prefabricado con los mismos componentes, pero diferentes atributos, usaremos como ejemplo para las figuras, al zombi tipo zángano.

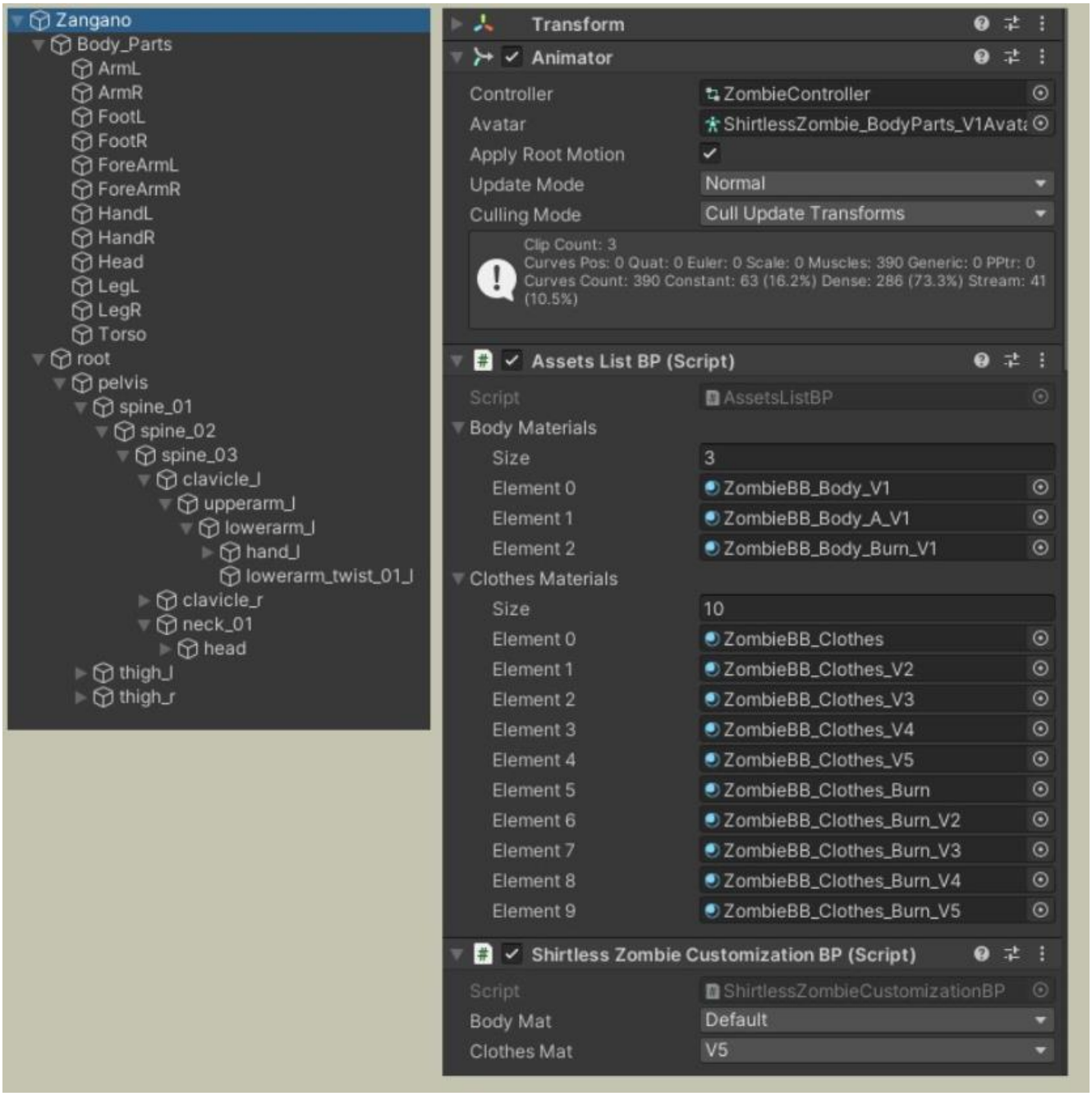


Figura 43: Jerarquía de objeto Zombi y componentes, Animator y Customization, que permite las variantes de modelos

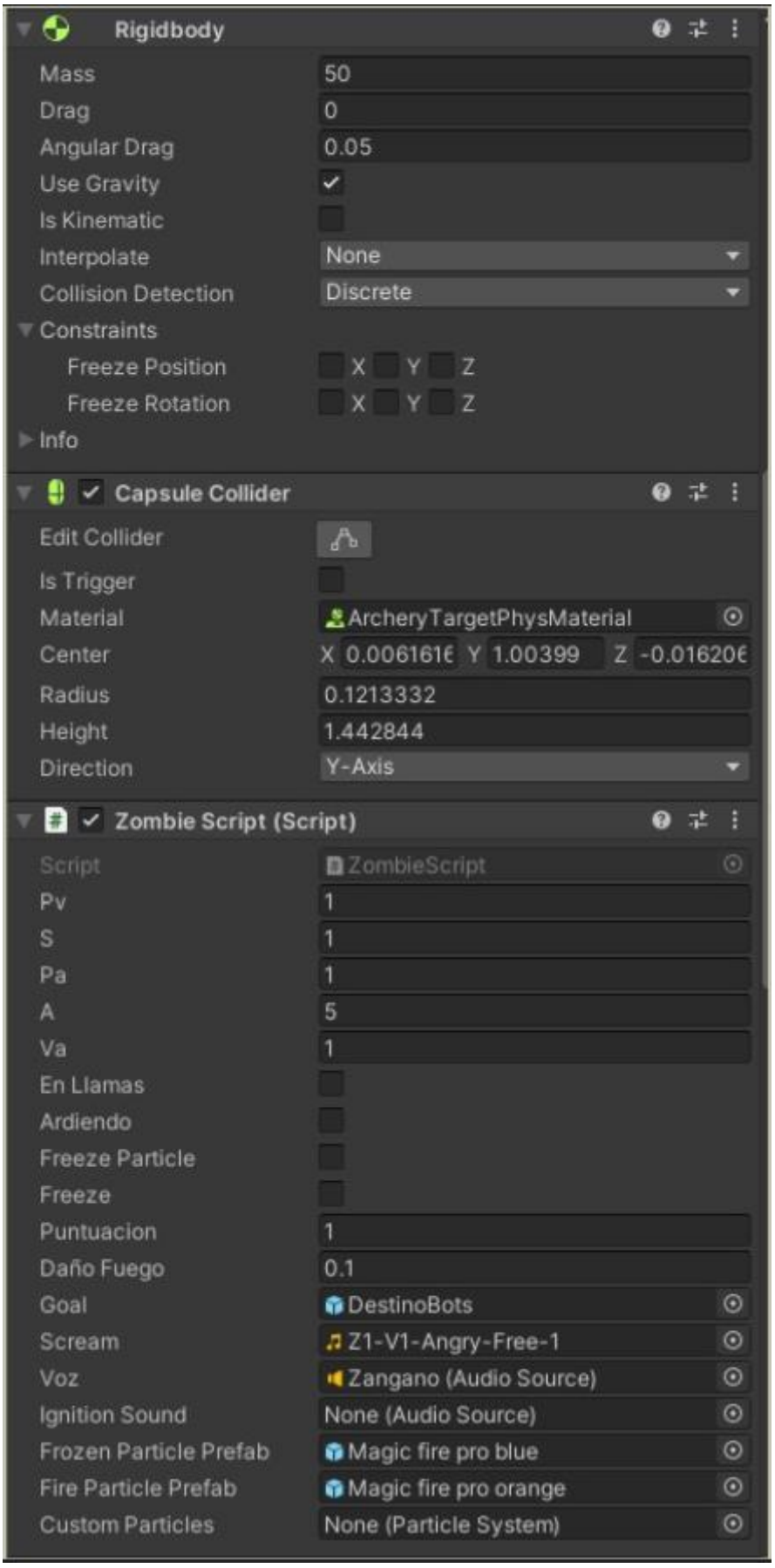


Figura 44: Componentes RigidBody, CapsuleCollider y ZombieScript

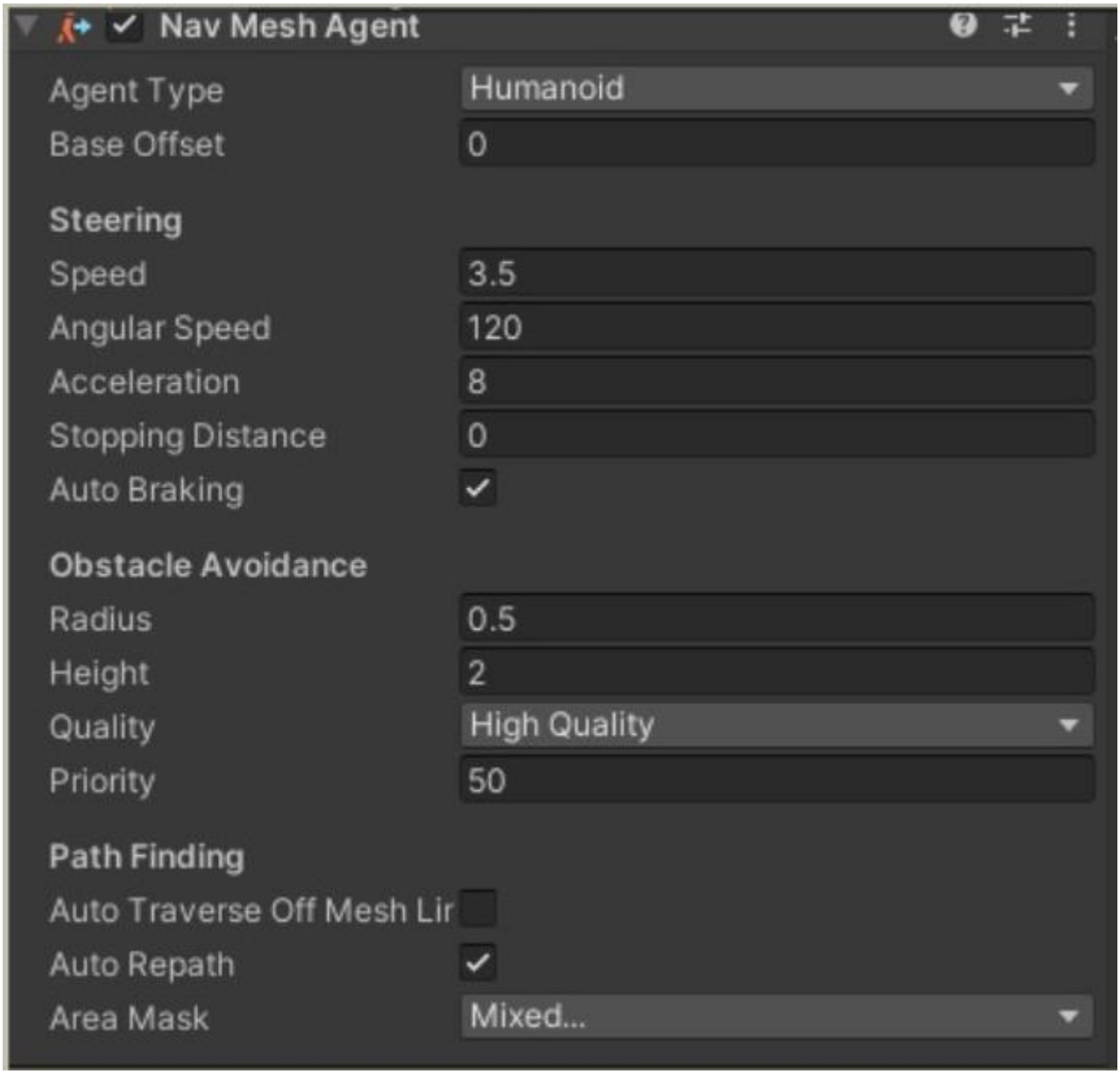


Figura 45: Componente NavMeshAgent, pathfinding y navegación

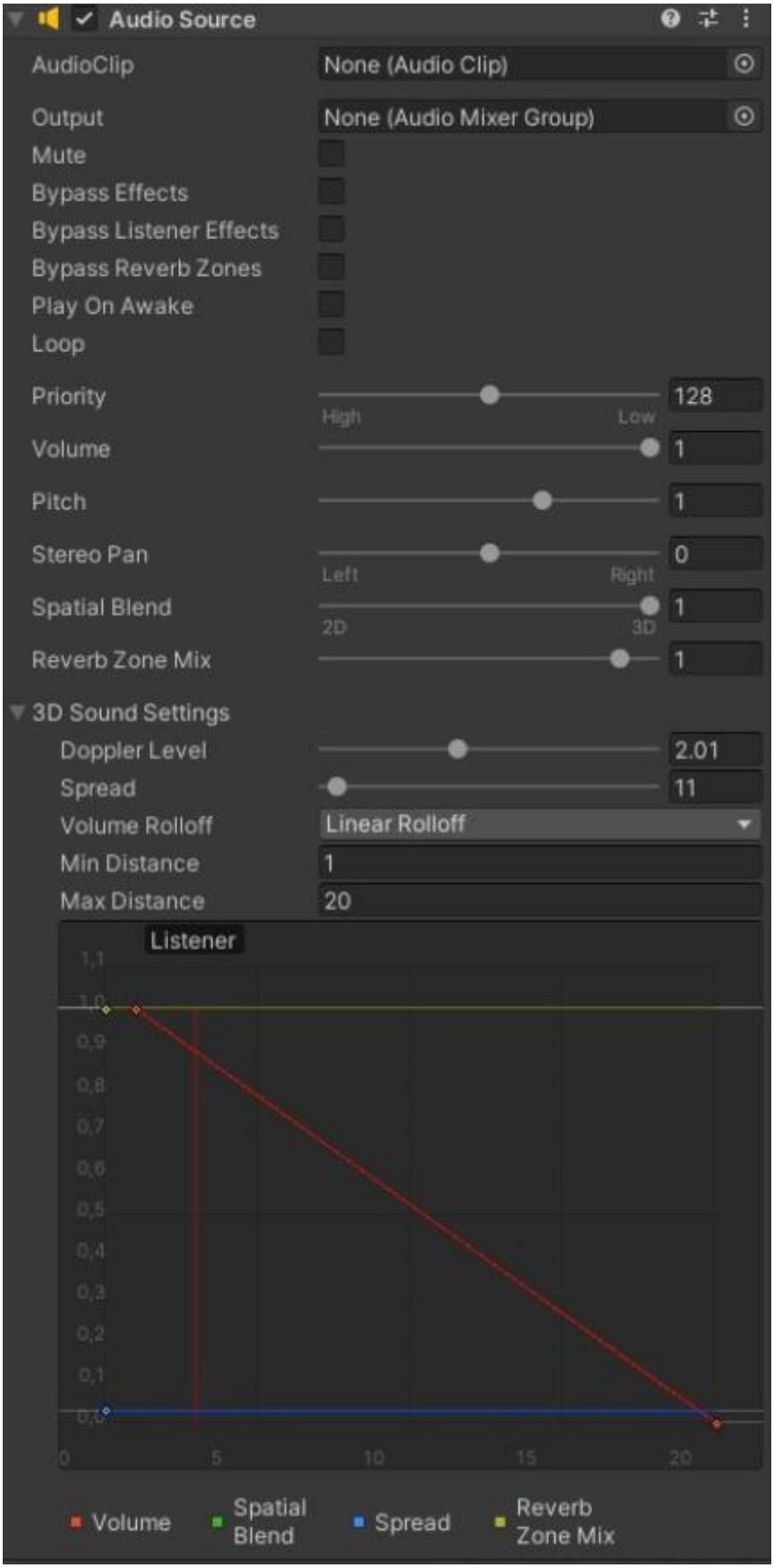


Figura 46: Componente AudioSource, que genera el sonido de los Zombis

7.7. Pruebas Beta

Como en la mayoría de videojuegos, al estar formados por grandes subsistemas de mecánicas que se relacionan entre sí para generar dinámicas, es muy complicado para un grupo de desarrolladores encontrar todos los posibles “bugs”, fallos de implementación y “glitches”. Se describirán a continuación:

- **Bugs**: Un bug, en diseño de videojuegos, es un error en la programación del videojuego que afecta negativamente a su funcionamiento. Se trata de un error grave, ya que dificulta la progresión dentro del juego o directamente impide su ejecución. [54]
- **Glitch**: Los glitches también son errores de programación del videojuego, pero a diferencia de los bugs, estos no suponen un problema ni de rendimiento, ni de jugabilidad, ni de estabilidad del propio videojuego. A menudo estos están relacionados con la explotación de beneficios por parte de los jugadores [55]. Muchas empresas de videojuegos, directamente no corrigen glitches, siempre y cuando estos no supongan un inconveniente para la jugabilidad del videojuego o generen una situación de desigualdad entre jugadores.

Para identificar el mayor número de errores posibles, las empresas de desarrollo de videojuegos, permiten que los jugadores puedan jugar al juego antes de lanzar al mercado la versión final abierta del videojuego.

En nuestro caso las pruebas de integración recopiladas en las pruebas alfa ya se han realizado en el último sprint, como pruebas de integración. A continuación describiremos con más detalle que necesitamos para la fase de “beta testing” del videojuego que introdujimos en el apartado 3.2.3. Evaluación del videojuego.

7.7.1. Implementación especifica de la fase Beta

Para nuestra beta sería conveniente, reunir entre 10 y 20 personas dentro de unas instalaciones que dispongan del material necesario para que todas las personas puedan jugar al videojuego. Esto incluye: Salas o puestos independientes para cada jugador, una persona encargada de guiar, en el transcurso del videojuego, a los jugadores aclarándoles todas las dudas que pudieran tener sobre la jugabilidad. Es competencia de esta persona, apuntar todas las dudas que hayan tenido los jugadores durante el transcurso de la prueba, para sacar conclusiones sobre la curva de aprendizaje del videojuego [56]. Se ha estimado que la duración de la sesión de juego no debe ser inferior a 30 minutos, ni superior a una hora. Esto es así para evitar que el jugador tenga un acceso completo al videojuego y, que le dé el tiempo suficiente para experimentar todas las mecánicas del juego.

7.7.2. Elementos necesarios

Para obtener la mayor cantidad de detalles de los jugadores es conveniente que, después de jugar al videojuego, se le realice una encuesta sobre aspectos del juego. La encuesta en cuestión, tendrá diferentes puntos que nos permitirán evaluar, tanto la viabilidad del videojuego dentro del mercado, en base al entretenimiento que genera, como los posibles glitches, que pudiera contener esta primera versión.

Los bugs encontrados por los jugadores sean notificados al personal de evaluación en el momento de ser encontrados, para que el evaluador pueda apuntar el punto concreto donde sucedió el bug, así

como las circunstancias de su aparición. De esta manera los desarrolladores podrán, posteriormente, reproducir el mismo bug y encontrar una solución rápida para que no se vuelva a producir.

Es importante que la encuesta sea anónima, tener datos de carácter personal o sensible de los jugadores implicados en esta fase de pruebas no aportaría ninguna mejora para esta fase, y estos son datos que hay que tratar de una manera específica y para su recolección es necesario de una autorización expresa de los implicados, tal y como se recoge en el artículo 9 de la LOPD [57].

En el anexo D se muestra un ejemplo de encuesta que podría utilizarse para los propósitos de esta fase de pruebas. Hay que destacar que durante el proceso de evaluación el evaluador deberá resolver todas las dudas que los jugadores pudieran tener, durante la realización de la encuesta.

7.7.3. Interpretación de los resultados

Tras la recopilación de las encuestas, se deben interpretar los datos obtenidos durante el proceso de pruebas.

Se deberán recopilar todos los posibles bugs y glitches registrados por los jugadores, y añadirlos a una lista de hotfixes si el evaluador considera que tienen una solución sencilla. En caso de que su solución implique cambios en el diseño del videojuego se tendrá que plantear otra versión del producto.

Otro punto importante que hay que analizar, es la calidad del videojuego como mecanismo de entretenimiento y publicitario.

El equipo de evaluación deberá sacar las conclusiones en función de las notas medias obtenidas del total de encuestas, e intentar hacer una lista de opiniones, intentando homogeneizar las respuestas de los jugadores.

De estas listas de resultados se recogerán las conclusiones sobre la viabilidad del producto en el mercado.

Capitulo VIII

Conclusiones

Tras lo mostrado en esta memoria se puede concluir que se ha completado una primera versión totalmente funcional del juego, que satisface todos los objetivos propuestos.

Las restricciones a la movilidad y las medidas de higiene impuestas por el gobierno para frenar la crisis sanitaria, ocasionada por la enfermedad COVID-19, han sido las principales causas del fallo en planificación. Es imposible prever en el análisis de riesgos las implicaciones de una pandemia a nivel mundial. No se han podido usar ni las instalaciones ni el equipamiento necesario para completar el videojuego, lo que ha generado largos periodos de inactividad, o de actividad reducida que han aumentado los costes y retrasado la mayoría de las entregas.

Esta primera versión del videojuego se puede presentar a las instituciones públicas y empresa privada, como muestra del potencial que tiene invertir en videojuegos serios. Este es el principal objetivo del trabajo realizado, servir como propuesta de lo que pueden aportar los juegos a la promoción turística del patrimonio. Esta primera versión no se puede considerar preparada para ser lanzada al mercado de los videojuegos, tampoco era el objetivo. Ahora bien, es la semilla para que en siguientes versiones se logre lo planteado en el GDD de este mismo proyecto y lograr un producto adecuado para el sector. De esta forma se podría aprovechar el potencial del proyecto, no solo como videojuego serio, si no como videojuego tradicional.

El coste de desarrollar un videojuego serio, que además utilice la última tecnología, como en nuestro caso la realidad virtual, es muy alto. Hay que tener muy presente que desarrollar un videojuego es un proceso muy laborioso, que requiere de participación de muchos profesionales, tanto técnicos como creativos, para que el producto sea viable en tiempo y forma, y se obtenga el máximo rendimiento promocional del producto objeto de la promoción. Para futuros desarrollos sería conveniente que se trabajara en equipo para mejorar, no solo la rapidez en el desarrollo, si no para aportar poder compartimentar mejor el trabajo

En este proyecto se ha aprendido a desarrollar una versión funcional de un videojuego, incluida la documentación técnica relacionada con el mismo. Se han adquirido conocimientos sobre animación, lógica de mecánicas, comunicación entre objetos, diseño de escenarios, interfaz de usuario, entre otras.

Durante el desarrollo del proyecto se han aprendido procedimientos de: Metodologías agiles, diseño de videojuegos y documentación técnica.

Capitulo IX

Líneas de trabajo futuro

La principal línea de trabajo futuro debería ser la de aumentar la funcionalidad del videojuego en todos los apartados para que se adapte al diseño original descrito en el GDD.

Al mismo tiempo, hay que intentar mantener una fase beta de pruebas finales con usuarios reales, en cada iteración y desarrollo de nueva versión del producto.

Siguiendo estas líneas de actuación, se ha desarrollado un resumen de las siguientes dos iteraciones, con los elementos más importantes que se deberían añadir en el videojuego si se continuara su trabajo.

Los principales objetivos a largo plazo se clasifican en las siguientes fases:

Fase II:

- Implementación de un sistema de malla entorno al modelo del castillo para permitir ubicar defensas fijas en emplazamientos concretos a elección del jugador.
- Crear un sistema de puntuaciones online.
- Crear los modelos de los diferentes tipos de defensas y la interacción con los zombis.
- Incorporar más tipos de flecha para aumentar la funcionalidad de la mecánica FPS del videojuego.
- Añadir y mejorar los efectos de sonido, incluir textos narrados por la protagonista para introducir la historia del videojuego.
- Pruebas Beta

Fase III:

- Crear un sistema de transiciones día/noche que ayude al jugador a identificar las fases del juego (descanso/oleadas)
- Aumentar la ambientación natural, para mejorar la sensación inmersiva del juego
- Mejorar la interfaz de usuario para que cuadre con el hilo narrativo del videojuego
- Aumentar el tipo de zombis, con diferentes características para aprovechar todas las mecánicas del videojuego.
- Añadir objetos con los que el jugador pueda interactuar para narrar la historia.
- Pruebas Beta

Apéndices

Apéndice A

Acrónimos

FPS: First Person Shooter, en español, tirador de primera persona.

GDD: Game Design Document, en español, documento de diseño de videojuego.

3D: 3 Dimensiones.

RTS: Real Time Strategy, en español, estrategia a tiempo real.

VR: Virtual reality, en español, realidad virtual.

MDA: Mecanicas-Dinamicas-Aestéticas.

API: Application Programming Interfaces, en español, interfaz de programación de aplicaciones.

SDK: Software Development Kit, en español, kit de desarrollo de software.

DLC: Downloadable Content, en español, contenido descargable.

PC: Personal Computer, en español, ordenador personal.

MOBA: Multiplayer Online Battle Arena, en español, arenas de batalla multijugador

MMORPG: Massively Multiplayer Online Role-Playing Game, en español, videojuegos de rol multijugador masivos en línea.

PS: Playstation.

GML: Game Marker Languaje.

IDE: Integrated Development Environment, en español, entorno de desarrollo integrado.

2D: Dos dimensiones.

Fps: Fotogramas por segundo.

TFG: Trabajo de fin de grado.

US: User Story, en español, Historia de usuario.

LOPD: Ley Oficial de Protección de Datos.

Spawn: Zona designada para la reaparición de jugadores o personajes no jugables

Collider: Los componentes Collider definen la forma de un objeto para los propósitos de colisiones físicas.

Mesh: Un Mesh o Malla, es un componente que define la forma que tiene un objeto y que permite a este objeto ser representado en un entorno en 3D

Navigation Agent: Los componentes Navigation Agent o NavMeshAgent le ayudarán a usted crear personajes que se eviten a ellos mismos mientras se mueven hacia su objetivo. Los Agentes razonan

acerca del mundo del juego utilizando el NavMesh y saben cómo evitarse entre ellos al igual que otros obstáculos en movimiento. Pathfinding (Encuentra caminos) y el razonamiento espacial son manejados utilizando la API de scripting del NavMesh Agent.

NavMesh: Representa la forma de un objeto para interactuar con los NavMesh Agent y que estos no colisionen.

RigidBody: Los Rigidbodies le permiten a sus GameObjects actuar bajo el control de la física. El Rigidbody puede recibir fuerza y aceleración para hacer que sus objetos se muevan en una manera realista.

Navigator: El sistema de Navegación le permite crear personajes que pueden moverse de forma inteligente en el mundo del juego, utilizando meshes de navegación que son creadas automáticamente de la geometría de su escena.

SkyBox: Los Skyboxes son una envoltura alrededor de su escena entera que muestra cómo el mundo se vería más allá de su geometría.

Localposition: En Unity hay diferentes sistemas de representación para ubicar objetos dentro de las escenas, global position y local position. Todas las posiciones que hacen referencia a global position utilizan un sistema de referencia basado en la escena completa, de manera que el punto (0,0,0) sería el centro de toda la escena. Sin embargo cuando hacemos referencia a localPosition, usamos un sistema de referencia local y por lo tanto solo afecta al objeto en cuestión, el punto (0,0,0) definiría el centro del objeto.

Apéndice B

Manual de uso

Para inicial el juego haga doble clic en el archivo MotaTD.exe. Asegúrese de tener iniciado SteamVR y las gafas de RV conectadas y configuradas.

Se iniciará el juego con una pantalla con un formulario, en el que podrá introducir el nombre con el que quiere jugar, mediante un teclado virtual, que tendrá que utilizar golpeando las teclas con los mandos de realidad virtual.

Posteriormente se cargará la escena principal del juego, puede utilizar la pulsación del pad central de cualquiera de los controladores para desplazarse apuntando a la dirección seleccionada y soltando la pulsación del pad. El desplazamiento del jugador está restringido al interior del castillo.

Para interactuar con los objetos del juego, como el arco, deberá acercarse a ellos y cogerlos con ayuda del gatillo de uno de los controladores. Una vez tenga sujeto el arco podrá disparar, emulando el apoyo de la flecha en el reposaflechas.

El tensado de la cuerda para el disparo de la flecha, se realiza mientras tenga la flecha anclada al reposaflechas, manteniendo pulsado el gatillo del controlador que sujeta la flecha. Soltará la flecha si esta la cuerda tensada y suelta el gatillo del controlador. Apunte con ayuda del controlador que sujeta el arco, alineándolo con el agarre de la flecha.

Si quiere desplegar el minimapa, deberá pulsar el botón de opciones, ubicado encima del pad táctil de cada controlador. Mientras tenga el minimapa desplegado, podrá desplazarse rápidamente entre los marcadores del minimapa. Cambie de marcador usando los gatillos del controlador y vuelva a pulsar el botón de opciones cuando esté seleccionado el marcador con la ubicación deseada de teletransporte.

Puede interactuar con los botones de los paneles acercando el controlador al botón y apretando el gatillo del controlador.

Para cambiar el tipo de flecha, de normal a helada, puede apoyar el dedo en los laterales izquierdo y derecho del pad, no hace falta presionar, solo apoyar. El lateral derecho carga una flecha de hielo, si tuviera. El izquierdo carga una flecha normal y son las flechas por defecto.

Puede salir en cualquier momento del juego con ayuda de la interfaz de steamVR.

Apéndice C

Contenido de la memoria USB

La memoria USB contendrá:

- El ejecutable del videojuego
- La memoria del trabajo de fin de grado en pdf
- El repositorio de código fuente del proyecto.

Apéndice D

Encuestas pruebas Fase BETA

Cuestionario Beta Mota TD	
En una escala del 1 al 10, ¿Le ha resultado el juego entretenido? Siendo 1 la calificación más baja	[1-10]
En caso de que la respuesta a la anterior pregunta sea menor que 5, indicar los aspectos que no le han gustado.	[TEXT]
¿Considera que el juego es intuitivo? ¿Es fácil de aprender a manejar al jugador?	[SI | NO]
En caso de que la respuesta anterior haya sido “NO”, indicar el porqué.	[TEXT]
¿Consideraría a este videojuego sencillo?	[SI | NO]
En caso de que la respuesta anterior haya sido “NO”, indicar el porqué.	[TEXT]
¿Ha encontrado algún punto en el que el juego le haya parecido frustrante?	[SI | NO]
En caso de que la respuesta anterior haya sido “SI”, indicar en que oleada.	[NUMBER]
En una escala del 1 al 10, ¿Cómo de atractivo le ha resultado el entorno del videojuego? Siendo 1 la calificación más baja	[1-10]
En una escala del 1 al 10, ¿Cómo de atractivos le han resultado los modelos de los Zombies? Siendo 1 la calificación más baja	[1-10]
En una escala del 1 al 10, puntúe el sistema de movimiento del videojuego. Siendo 1 la calificación más baja	[1-10]
En caso de que la respuesta a la anterior pregunta sea menor que 5, indicar los aspectos que no le han gustado.	[TEXT]
¿Conoce el escenario en el que está ambientado el videojuego? (en referencia al castillo)	[SI | NO]

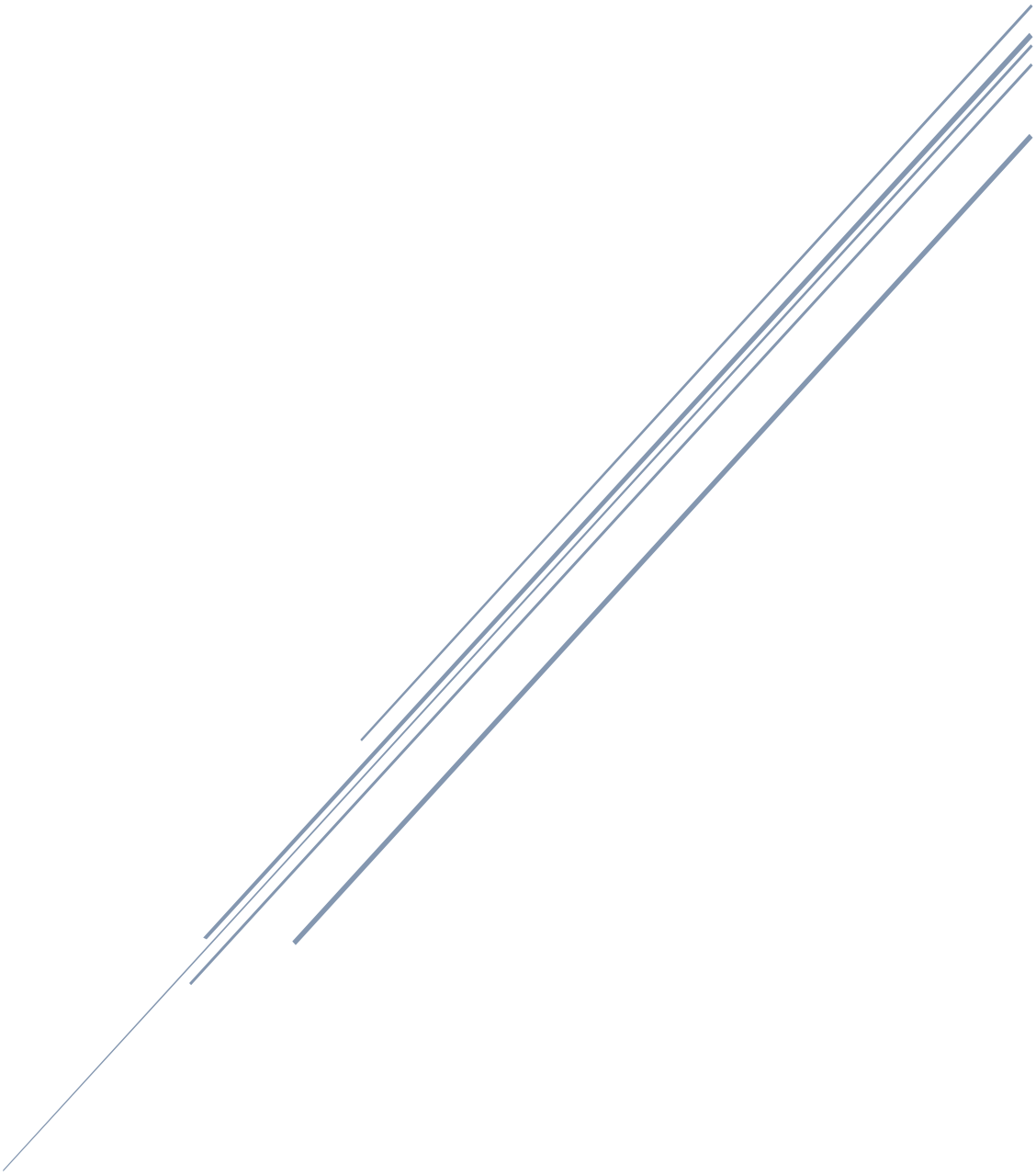
En caso de que la respuesta anterior haya sido “NO”, indicar si le gustaría visitarlo.	[SI | NO]
¿Ha encontrado algún error en el juego que le haya permitido tener determinada ventaja para superar alguna oleada o todas?	[SI | NO]
En caso de que la respuesta anterior haya sido “SI”, indicar cuales.	[TEXT]

Apéndice E

Documento de Diseño de Juego. GDD

Documento de Diseño del Juego

Mota Tower Defense VR



Alfredo Fernández ramos

Este documento de diseño es para un videojuego de acción/estrategia en primera persona, que use la realidad virtual llamado Mota Tower Defense VR (Mota TDVR).

Lo innovador del juego es enlazar dos estilos de juego, FPS[1] y RTS[2], en uno solo utilizando la tecnología de realidad Virtual para ello. El videojuego partirá de un proyecto real, la virtualización y modelado en 3D del castillo de la Mota (Medina del Campo, Valladolid), realizado por la empresa Irzón y se construirá el juego en base a ese modelo.

El documento servirá de guía de consulta al personal involucrado en el desarrollo del juego, estableciendo las principales líneas de diseño para poder desarrollar el mismo con las ideas principales predefinidas.

Se podrá ampliar la información de este documento, a fin de esclarecer cualquier duda que pudiera surgir en el proceso de desarrollo.

[1] First Person Shooter (Juego de disparos en primera persona)
[2] Real Time Strategy (Juego de estrategia en tiempo real)

1. Visión General

Mota tower defense, es un juego de acción que mezcla los componentes específicos de los clásicos tower defense, basados en la categoría RTS de videojuegos. El componente principal en el que se va a basar el juego es en la jugabilidad siendo este un factor crítico, acompañado de una historia corta para aumentar la inmersión en el videojuego. En éste el jugador controla a Cristina una Ingeniera de diseño industrial que trabajo en la creación de un modelo en 3D del castillo, la historia trascurre en un ambiente apocalíptico zombi, con la única idea de sobrevivir y dado que conoce perfectamente el potencial defensivo del castillo, decide ir a este para protegerse de los zombis.

El objetivo principal del jugador será manejar a Cristina y decidir cómo evitar que las hordas de zombis entren al castillo y defenderse de estas. Para ello dispondrá de un arco y un número ilimitado de flechas y un panel de mando en el que podrá gestionar las defensas del castillo. En cada ronda llegarán nuevos supervivientes, que podrán utilizar estas defensas.

El juego recompensará la habilidad del jugador con el manejo del arco, así como la táctica empleada a la hora de distribuir sus defensas, aportando dos maneras de defenderse, una táctica y otra directa, pudiendo eliminar a los zombis con sus propias manos demostrando la destreza del jugador.

Estará disponible para pc, siendo necesario usar gafas de realidad virtual y 2 controladores para las manos. El desarrollo se realizará bajo HTC VIVE marca líder del mercado en tecnología de realidad virtual, y gafas especialmente creadas para videojuegos en RV.

Con un diseño realista de ciencia ficción, el agente motivador principal a explotar será la competitividad multijugador basándose en el número de oleadas que pueden soportar. Aunque el juego sea exclusivo para un jugador, el número máximo de oleadas será registrado y asociado a un identificador de usuario, estas puntuaciones estarán publicadas en un servidor web para consulta de cualquier persona que se registre en la plataforma. El juego también explota otras facetas psicológicas en el jugador, como poder evadirse de la realidad, usando para esto un escenario ficticio apocalíptico. La sensación de inmersión inducida por el componente tecnológico utilizado (realidad virtual), aumentará este efecto psicológico. Estas facetas se entrelazan para generar en el jugador un entretenimiento constante, pero que no busca mantener al jugador en constante tensión, lo que permitirá al juego formar parte de una actividad des estresante para el jugador.

A pesar de que el objetivo principal sea fomentar la competitividad entre los jugadores, el jugador podrá recorrer la mayor parte del castillo, explotando, también, el componente psicológico de exploración en el jugador.

Para mejorar la sensación de inmersión se intentará utilizar actores de doblaje para la narración de diálogos que expliquen la historia o que añadan factores de tensión al juego en momentos críticos, a modo de aviso y de motivación en determinados momentos del juego.

Con estos componentes buscaremos que los jugadores sientan la necesidad de jugar para demostrar que son los mejores, componente competitivo, y a su vez que disfruten de la recreación de un escenario de ciencia ficción, con unos modelos hiperrealistas, que les motive a visitar el castillo de la Mota y los alrededores, componente publicitario, y tengan una sensación de inmersión total en el escenario.

2. Mecánicas de juego

Visión general

El juego utilizará la RV como visión principal y los controladores de este equipo para desplazarse y controlar los elementos accesibles del entorno.

El jugador deberá evitar que los zombis entren al castillo donde permanecerá durante todo el tiempo de juego. Para ello dispondrá de dos herramientas, un arco y flechas que podrá disparar usando los controladores del equipo de realidad virtual y un panel para controlar la asignación de supervivientes[3] a las defensas del castillo[i], que atacaran automáticamente ayudando en el objetivo de que los zombis no conquistes en castillo. El usuario podrá explotar su naturaleza aventurera explorando el castillo para conocer más detalles de la historia.

Cámara

La visión del juego se centra en la realidad virtual, desde la perspectiva del jugador. Podrá desplazarse y realizar giros de 360º para observar el entorno construido, siempre que cuente con un equipo de RV que pueda detectar tanto el movimiento relativo del jugador, como el movimiento de la cabeza, lo que le permitirá ver el mundo virtual como si se tratase de la realidad. La cámara contará de mecanismos para evitar la colisión con objetos y que el jugador no pueda ver atravesando paredes, aumentando la sensación de realismo.

Interfaz gráfica de usuario

La pantalla del jugador contará con una capa de información mire a donde mire, en esta se representarán los elementos más importantes del juego:

- **Contador de vidas:** En la parte superior derecha se mostrará un indicador con el número de vidas restantes, un número representativo del número de zombis que pueden entrar al castillo antes de que termine el juego, hay que tener en cuenta que no todos los zombis contabilizan igual[4].

- **Contador de flechas:** En la esquina inferior derecha se mostrará el número de proyectiles restantes que puedes usar en las oleadas con el arco, junto con un icono representativo que indique el tipo de flecha[5]. La flecha seleccionada tendrá un reborde coloreado, cambiará si el jugador interactúa con el sistema para cambiar de tipo de munición.

- **Contador de oleadas:** En la parte superior centrado se encontrará un contador que represente la siguiente/actual oleada, este número cambiará cuando entremos en los turnos de asignación de defensas justo al terminar cada oleada de zombis (Se explica en el apartado Mecánicas de combate).

- **Contador de supervivientes:** En la esquina inferior izquierda podremos ver el número de supervivientes asignados a defensas del castillo / el número de supervivientes totales que tenemos en el castillo.

- **Cuadros de diálogo:** Si interactuamos con un objeto que tenga un texto explicativo del mismo o de parte de la historia, nos aparecerá en el centro de la pantalla un cuadro con este texto,

[3] Ver apartado Mecánicas de Combate, subapartado Defensas del castillo.
[4] Ver apartado Mecánicas de Combate, subapartado Enemigos.
[5] Ver apartado Mecánicas de Combate, subapartado Combate con arco

claro y legible, aunque también se pueda leer el texto en el propio objeto antes de interactuar con él.

- **Contador de zombis restantes:** En cada oleada se podrá consultar en la parte superior derecha, justo debajo del número de vidas, el número de zombis “vivos” que quedan para terminar la oleada.

- **Minimapa en 3D:** Cuando el jugador lo desee podrá desplegar una reproducción en tres dimensiones del castillo y sus alrededores, con la que podrá ver la situación actual del castillo bajo asedio, el estado de las defensas, las oleadas de zombis en tiempo real, así como poder transportarse automáticamente a las zonas principales del mismo

Estos elementos estarán siempre disponibles en cualquier momento incluidos en la visión del jugador, existen varios menús de interacción que podrán ser desplegados si se cumplen las condiciones:

- **Menú de operaciones:** Éste solo estará disponible desde la sala de mando, en lo más alto de la torre del homenaje, y se activará al interactuar con la mesa de operaciones. Se desplegará un mapa en 3D como el minimapa, pero en éste se podrá interactuar con los espacios de la maqueta designados a defensas. Al seleccionar cada parte nos mostrará un cuadro de dialogo con el estado de la defensa:
    a) puntos de vida restantes / totales si se trata de una defensa pasiva.
    b) número de supervivientes mínimos necesarios / máximos si se trata de una defensa activa. También indicará el número de supervivientes asignados actualmente.
    c) Si se trata de una defensa consumible, se mostrará un indicador para que el jugador sepa si está activa.
    d) En cualquiera de estos cuadros de diálogos se incluirá una opción para asignar supervivientes en caso de que se desee mejorar la defensa, repararla o rellenarla si se trata de una defensa consumible.

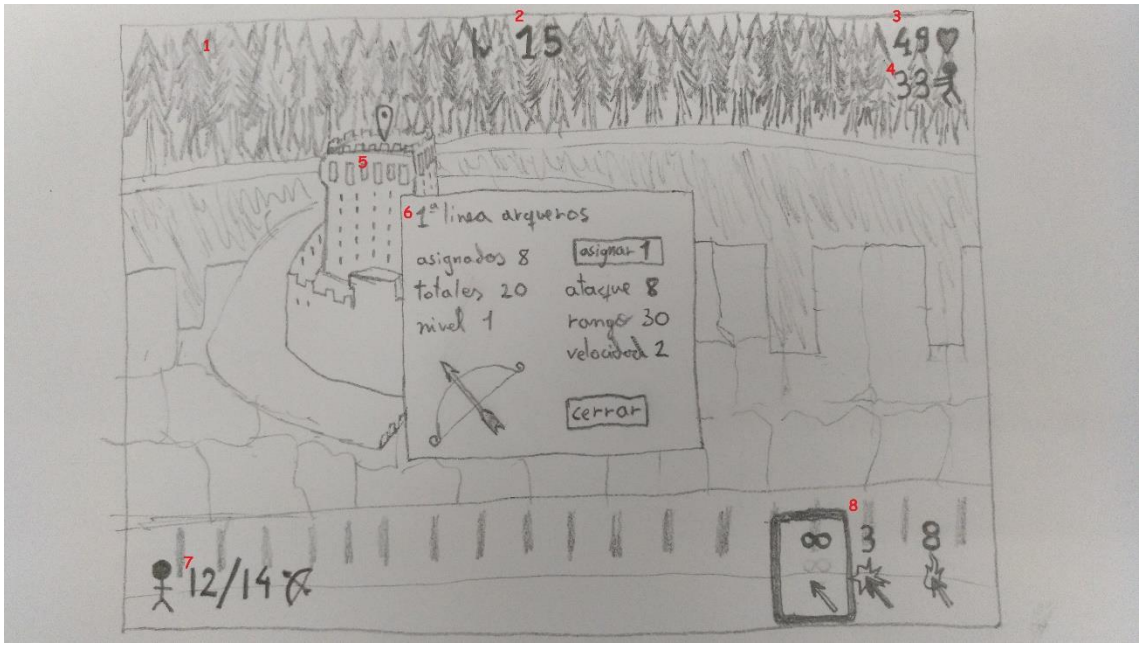


Figura 1: Boceto de la visión del jugador, incluida la interfaz de Usuario

1: Entorno de juego

2: Oleada actual

3: Puntuación en zombis permitidos dentro del castillo

4: Zombis restantes en la oleada actual

5: Minimapa

6: Menú de gestión de defensa

7: Supervivientes asignados / supervivientes totales

8: Selector de munición, cantidad de munición por tipo

Lista de controles

La mayoría de las acciones del jugador serán de interacción con los elementos del castillo, así como diálogos y menús seleccionables en estos. Cuando disponga de arco y flechas equipadas podrá disparar a los enemigos haciendo uso de los mandos y del movimiento propio del jugador en la vida real, que será interpretado por el juego como un lanzamiento de flecha. Esta última funcionalidad viene heredada del api de Steam VR.

- **Desplazarse**: El jugador podrá desplazarse por el mapa haciendo uso del sistema de movimiento integrado en los sistemas de realidad virtual. Seleccionará el botón del mando que permite el desplazamiento y lo mantendrá pulsado mientras selecciona el área al que se quiere mover, indicado mediante un cuadro en el suelo que representa el área por la que se puede

desplazar en “estático” (sin usar esta funcionalidad, con la detección de movimiento fuera del juego). Al soltar el botón aparecerá en el área seleccionada.

- **Mostrar minimapa**: Pulsando un botón del controlador de VR aparecerá justo delante de la posición relativa del jugador un mapa en tres dimensiones del castillo completo. Podrá interactuar con el minimapa para desplazarse rápidamente a otras zonas del castillo.

- **Mirar alrededor**: Al girar la cabeza, y con esta las gafas de VR podrá ver todo lo que le rodea, no necesita funcionalidad extra para la visión.

- **Acción**: Podrá interactuar con los objetos, diálogos, menús del juego con el gatillo del controlador. Para ello se acercará con uno de los mandos, con el que quiera interactuar, a un objeto y pulsará el gatillo, en función de si se trata de un agarre o de una interacción simple, tendrá que mantener el gatillo presionado o solo pulsarlo.

Movimiento general

En el juego existen tres tipos de movimiento del jugador, movimiento normal, tele transporte y movimiento en estático:

- **Movimiento Estático**: El jugador y la cámara están asociados de manera que los movimientos que haga el usuario con la cabeza o las manos que sujetan los controladores, se registrarán en el juego. Esto incluye los pequeños desplazamientos que el jugador haga: dependiendo de la configuración y del espacio para jugar del que disponga, el usuario podrá desplazarse más o menos usando este tipo de movimiento, siendo la única limitación el espacio que tenga fuera del juego y la capacidad de detección de los sistemas de VR. Dentro del juego dispondrá de un área marcada en el suelo con la que se indicará el espacio que puede recorrer, idealmente, sin poner en peligro su integridad física.

- **Movimiento Normal**: Debido a las limitaciones físicas del movimiento en estático, el jugador dispondrá de una vía de desplazamiento más cómoda, sin tener que moverse en el mundo real. Ésta consiste en un desplazamiento utilizando el mando del sistema de realidad virtual. Mantendrá un botón presionado mientras apunta con el mando al lugar donde se quiere transportar. Estará limitado a las zonas libres de obstáculos por las que se pueda pasar, y aparecerá un rectángulo indicador del área por la que podrá desplazarse utilizando el movimiento estático a modo de pre visualización.

- **Tele transporte**: Para facilitar el movimiento por un entorno virtual tan grande, se dispondrá de un sistema de viaje rápido a zonas clave del castillo que será accesible desde el mini mapa en todo momento. Al sacar el mini mapa podrá interactuar con estas zonas directamente, estarán marcadas en el mismo y el jugador deberá utilizar el controlador para marcarlas y con un botón podrá activarlas para moverse a estas zonas.

Superficies

El movimiento del jugador será libre, pudiendo desplazarse a cualquier punto dentro del castillo- Las zonas no accesibles estarán limitadas basándose en la técnica de las colisiones. Si no puede pasar por una zona es porque no hay espacio para pasar: el jugador no podrá atravesar paredes, ni objetos que tengan un cuerpo con las colisiones activas. De esta manera, la superficie por la que pueda desplazarse será intuitiva y el jugador podrá darse cuenta rápidamente cuando no puede pasar por

una zona. En el caso del movimiento Normal, el usuario solo podrá seleccionar zonas habilitadas evitando que se desplace entre paredes o que atraviese objetos.

Interactuando con objetos
El jugador podrá interactuar con determinados objetos dentro del castillo, destacarán con el resto del entorno a fin de facilitar la identificación de estos por parte del usuario. Usando los controladores del sistema, podrá acercar estos al objeto y pulsando un botón interactuar con estos, los podrá coger. A partir de ese momento formarán parte del controlador, por lo que cualquier movimiento que el jugador realice con el brazo se verá reflejado en este objeto. Así generamos un sistema de movimiento del objeto basado en los movimientos fuera del juego, con este sistema podrá rotar, mover, e incluso lanzar el objeto.

3. Mecánicas de combate

En la siguiente sección se detallarán las mecánicas de combate con las oleadas de zombis, separando en dos secciones las que serán las mecánicas más importantes del juego.

Combate con arco
En el momento en el que se acceda al arco se podrá utilizar en las oleadas. Éste usará flechas de diferentes tipos. Las flechas normales serán infinitas. Las flechas especiales deberán ser generadas mediante el uso de supervinientes, cuantos más se designe a la función de elaboración de flechas más se generarán al comienzo de cada ronda. El tiempo exacto en relación con el número de supervivientes asignados se concretará en fase de desarrollo, para facilitar el equilibrado de dificultad y mejorar las mecánicas, Se podrá seleccionar el tipo de munición desde el controlador con un botón que permutará entre los diferentes tipos de flechas disponibles.

Los diferentes tipos de flechas que estarán disponibles en el juego para ser usadas son las siguientes:

- **Flecha normal**: Con una cantidad infinita de estas. Al disparar generaran daños en los zombis pudiendo llegar a matarlos si estas generan los suficientes daños como para reducir los puntos de vida del zombi a 0, siempre que impacten directamente con éstos. El daño que estas generarán, será constante de manera que serán útiles contra las hordas más básicas de zombis o para asistir al resto de defensas.

- **Flecha explosiva**: Estarán disponibles si se dedican supervivientes a la elaboración de flechas. Su daño no será individual sino de área. Desde el centro del impacto se generara una esfera explosiva que dañara a todos los zombis a los que toque. Serán muy útiles para momentos puntuales en los que haya muchos enemigos agrupados.

- **Flecha incendiaria**: Estarán disponibles si se dedican supervivientes a la elaboración de flechas. A diferencia de las flechas explosivas, estas no dañan en el momento del impacto, sino que mantienen una superficie de fuego en el punto de impacto que dañará a todos los enemigos que pasen por esta. Su daño inicial será menor que el de las flechas explosivas, pero al durar una cantidad de tiempo, el daño total en el tiempo que se pueda causar puede llegar a ser mayor. Será útil para ir debilitando a las oleadas de zombis que pasen por un determinado punto.

Será decisión del jugador la estrategia que prefiera a la hora de asignar supervivientes. De esta manera se le da libertad para que juegue de la forma que desee: si prefiere una defensa más directa basada en su habilidad, utilizando la mecánica del arco y las diferentes flechas o, si por el contrario, prefiere delegar esa función en sus defensas y no disparar ni una flecha. En cualquier caso, será muy recomendable que utilice ambos sistemas para facilitar la defensa del castillo.

Las físicas del disparo vendrán heredadas del sistema de disparo con arco de VR, y solo se modificará las interacciones que tengan las flechas al impactar. Así el jugador tendrá que imitar fuera del juego el tensado del arco y soltar el gatillo para soltar la flecha.

A las flechas normales se les podrá aplicar fuego si acercan está a una fuente de fuego, estas, a parte del daño inicial de la flecha al impactar, tendrán un pequeño daño en el tiempo causado por el fuego, se le irán reduciendo los puntos de vida segundo a segundo hasta que estos lleguen a 0. Esta mecánica será útil contra enemigos con muchos puntos de vida[6] ya que podremos generar más daño usando solo una flecha.

Nota: Cualquier objeto que podamos tirar hará daño a los zombis al impactar, pueden surgir nuevas ideas de defensa usando estos objetos.

Defensas del castillo

Desde el menú de gestión de defensas el jugador podrá asignar supervivientes a las defensas, esta será la manera de "pagar" nuevas defensas, reparaciones, mejoras y generación de consumibles. Se habilitará únicamente desde la torre del homenaje una zona de control de defensas para este fin, en el que mediante una visión en miniatura del castillo se podrá seleccionar las defensas a modificar. Éstas se gestionan mediante menús de dialogo que aparecen al seleccionar una de las defensas, y solo podrán ser gestionadas al finalizar cada una de las rondas.

En el castillo estarán distribuidas zonas habilitadas para diferentes tipos de unidades de defensa, en función con el tipo de defensa que sea aumentar el número de supervivientes asignados a cada una generará un efecto u otro:

- **Defensas Pasivas:** Vendrán ya por defecto instaladas en el castillo, su característica principal será los puntos de vida que les queden. Los zombis atacaran estas defensas justo antes de entrar en el castillo, y serán el último punto de defensa de éste. Su vida estará distribuida en cuartos, cada superviviente que asignemos permitirá reparar un cuarto de los puntos de vida. No se permitirá asignar supervivientes si no le falta al menos un cuarto de vida. El superviviente que asignemos no estará disponible en ninguna otra defensa del castillo hasta la siguiente ronda. Siendo ésta una forma de penalizar al jugador si desea volver a reparar sus puertas:

[6] Ver subapartado Enemigos.

que no pueda asignar esos supervivientes a más defensas, añadiéndole complejidad a las decisiones tácticas que el jugador decida.

- **Defensas Activas:** Para la utilización de estas, se tendrá que asignar al menos a un superviviente a cada una para su uso. Defensas de este tipo son arqueros y cañones. Los arqueros se desplegarán en conjunto, es decir, en líneas de arqueros. Las diferentes líneas de arqueros tendrán un número máximo de supervivientes asignados a las mismas que se concretará en fase de desarrollo a fin de mejorar el equilibrio de la dificultad. Al asignar este máximo de supervivientes a una línea, ésta no podrá disparar más flechas que el número de supervivientes asignados, sin embargo, se podrá aumentar el nivel de la defensa, lo que permitirá seguir asignando supervivientes hasta volver a llegar al máximo, Esto mejorará la velocidad de disparo de la línea de defensa. Asignar más supervivientes a las líneas de arqueros implicará que disparen más flechas aumentando el daño infligido a zombis y el área de actuación. Por otra parte los cañones son unidades de defensa unitarias, y asignar más de un superviviente a cada cañón permitirá que este dispare más rápido. El cañón tiene un daño muy elevado, pero una velocidad de ataque muy lenta. Será recomendado para enemigos con muchos puntos de vida. Los arqueros, en cambio, tienen mucho menos daño, pero atacan rápidamente y son perfectos para limpiar rápidamente oleadas con muchos zombis de poca vida.

- **Defensas consumibles:** Estas son defensas especiales que se activan como trampas automáticamente, justo antes de que lleguen a las zonas con defensas pasivas del castillo. Son calderos con brea que solo tienen un uso, para recargarlas hay que asignar un superviviente a cada una. Tienen ataque en área y un daño muy elevado. Se suelen usar como comodines en caso de que no se pueda contener una oleada muy grande de zombis con poca vida para evitar que las defensas pasivas sufran daño en lo que terminan de ser eliminados.

Enemigos

El castillo se verá asediado por oleadas de zombis. En cada oleada incrementará el número de zombis y se irán añadiendo nuevos tipos de enemigos, aumentando así la dificultad de cada ronda y motivando al jugador a realizar cambios constantes en la asignación de supervivientes. Los enemigos atacarán el castillo empezando siempre desde el mismo sector del castillo, el lado más alejado de la puerta de acceso, y recorrerán el camino exterior que bordea el castillo hasta llegar a la puerta principal. En futuras versiones se irá aumentando el área de "spawn"[7], de manera que en últimas rondas puedan aparecer desde cualquier parte exterior del castillo para atacar la puerta. Siempre seguirán los mismos caminos independientemente del tipo de zombi que sea. Los diferentes tipos de zombis son los siguientes:

- **Zángano:** Unidad de ataque básica, las primeras oleadas únicamente incluirán este tipo de enemigos. Se trata de unidades muy lentas (V), con daño de ataque básico (DA), velocidad de ataque básica también (VA), y pocos puntos de vida (PV). Para el cálculo de la puntuación esta unidad otorgará 1 punto al ser eliminada.

[7] Zona de reaparición de los enemigos.

- Parámetros[8]:
  - V = 2.0
  - DA = 1.0
  - VA = 1.0
  - PV = 1.0

- **Runner:** Evolución de los zánganos, atacan y se desplazan más rápido que estos, pero tienen puntos de vida parecidos. Para el cálculo de la puntuación esta unidad otorgará 2 puntos al ser eliminada.
  - Estadísticas:
    - V = 5.0
    - AD = 1.0
    - VA = 2.0
    - PV = 1.0

- **Juggernaut:** Zombi pesado, se mueven lentos pero tienen muchos puntos de vida, su daño a las defensas pasivas es muy elevado, deben ser eliminados rápidamente para evitar que las defensas pasivas pierdan muchos puntos de vida. Para el cálculo de la puntuación esta unidad otorgará 5 puntos al ser eliminada.
  - Estadísticas:
    - V = 1.0
    - AD = 5.0
    - VA = 1.0
    - PV = 5.0

Zombi Boss: Estos son la unidad más difícil de combatir del juego, estos enemigos tendrán estadísticas potenciadas dependiendo del tipo de boss. Estos tipos serán mezclas de los tipos anteriores. Por ejemplo, un zombi boss puede ser Runner Zángano, que será muy rápido, tendrá un ataque rápido pero débil y una cantidad de puntos de vida moderada. Otro ejemplo podría ser un Runner Juggernaut que será mucho más rápido que los Juggernauts y tendrá mucha más vida que los Runners. La primera aparición de los zombi bosses será tardía pero su frecuencia de aparición irá aumentado a medida que avancemos con las rondas. Para el cálculo de la puntuación esta unidad otorgará 10 puntos al ser eliminada.

- Estadísticas Runner Zángano:
  - V = 12.0
  - AD = 2.0
  - VA = 3.0
  - PV = 20.0

- Estadísticas Runner Juggernaut:
  - V = 8.0
  - AD = 3.0
  - VA = 3.0
  - PV = 20.0

- Estadísticas Zángano Juggernaut:
  - V = 1.0
  - AD = 10.0
  - VA = 2.0

[8] Los parámetros de los enemigos son valores numéricos representativos de las propiedades que estos tienen, por ejemplo: los puntos de vida de un zombi, su velocidad, el daño que hacen a las defensas pasivas, etc...

- PV = 50.0

4. El Juego

Niveles

El juego utilizará un sistema de niveles basado en oleadas “infinito”.

Los niveles o rondas constan de 2 fases la primero es de preparación de la defensa, serán estas fases las idóneas para explorar el castillo, la segunda será la propia de la defensa del castillo y cuándo empezarán a aparecer enemigos. En la segunda fase de cada ronda irán apareciendo progresivamente nuevos y mejores enemigos al mismo tiempo que aumenta su número. El jugador debe ser capaz de repeler a todos. El castillo no puede soportar que entren más de 50 puntos en zombis por la puerta principal. Una vez se alcance este número el juego habrá acabado y la máxima puntuación obtenida se calcula sumando el número de puntos de cada zombi eliminado.

Para comenzar cada ronda se dispondrá de un botón en el menú de gestión de defensas

Oleadas
Las 5 oleadas iniciales servirán para que el jugador aprenda a identificar los diferentes tipos de zombis.

- La oleada inicial tendrá 10 zombis de tipo Zángano.
- La segunda oleada tendrá 10 zombis tipo Runner.
- La tercera oleada tendrá 5 zombis tipo Juggernaut.
- La cuarta oleada tendrá 10 Zánganos, 5 Runner y 3 Juggernaut.
- La quinta oleada incluirá únicamente los 3 tipos de zombi boss

A partir de la 5 oleada, el cálculo de enemigos por oleada será incremental basado en la oleada base.

- Sexta oleada, oleada Base: Puntos totales 60 Distribución:
    - 30 Zánganos
    - 10 Runners
    - 2 Juggernaut

En las rondas sucesivas se incrementarán los zombis en estas proporciones:

- +10 Zánganos/ronda
- +5 Runners/ronda
- +1 Juggernaut/ronda

A partir de la ronda número 10 se introducirá un nuevo Zombi Boss cada 10 rondas.

Nivel de práctica
El juego como tal no tendrá un nivel de práctica, a modo tutorial, se le informará al jugador de los controles al principio del juego, mediante diálogos y monólogos del personaje. Las primeras rondas bastarán como introducción por su facilidad, pero debido a que las mecánicas del juego no son demasiado complejas no se precisa de un nivel 0 de iniciación, es preferible que el jugador domine las mecánicas del juego a medida que va jugando partidas. En este estilo de juego, el jugador debe explorar todas las posibilidades tácticas mediante la experimentación y la práctica. Utilizaremos el mismo principio para nuestro videojuego, lo que hará que las primeras veces que se juegue tenga una curva de aprendizaje muy elevada, pero que sea también muy fácil mejorar con las partidas sucesivas,

debido a la sencillez de las mecánicas, para motivar al jugador a que siga jugando, se fomentará la competitividad mediante un “leaderboard”.

5. Inteligencia Artificial

Se aplicará una IA muy básica tanto a los enemigos, como a las defensas, tal como se muestra a continuación.

IA Enemiga
Los zombis se irán generando desde una o más posiciones de varias posibles localizaciones del mapa y avanzaran por las rutas predefinidas, que llevan a las defensas pasivas, cuando detecten que una defensa pasiva está dentro de su radio de acción, la empezarán a atacar, hasta que esta pierda los puntos de vida y quede destruida, en ese momento podrán entrar al castillo y se irá reduciendo el contador de vidas.

IA Aliada (defensas)
En el caso de las defensas, sus instrucciones son sencillas atacarán a todo enemigo/zombi que entre dentro de su radio de acción, la velocidad, el tipo de ataque dependerá de la defensa, y su radio de acción también.

6. Historia

Contexto
Año 2020, Se declara un estado de emergencia a nivel mundial, la gente es atacada por criaturas que una vez fueron humanas. Nadie sabe de dónde han salido, se rumorea que varias empresas farmacéuticas experimentaban con agentes infecciosos microscópicos acelulares para combatir un determinado cáncer cerebral cuya tasa de incidencia en la sociedad estaba aumentando sin explicación alguna.

Nunca aceptaron esta vacuna oficialmente, pero sí que se aplicó en pacientes menores. No se sabe más acerca del patógeno. La gente especula sobre posibles mutaciones del virus generadas por el propio sistema autoinmune del ser humano, claro que son solo hipótesis.

La población ve reducido su número a pasos agigantados a medida que el patógeno se extiende en la sociedad. Nadie puede contener lo que denominan las autoridades competentes como “la Plaga”.

La única esperanza de la población es esconderse o huir... la pregunta es: ¿Dónde esconderse, donde huir?

Narrativa In-Game

La historia será expuesta a modo introductorio al jugador antes de empezar a jugar. Sin embargo, la trama solo informará al jugador sobre la situación actual de Cristina, de que huye y a donde, el resto de la trama la tendrá que ir averiguando a medida que transcurran las rondas, donde al llegar nuevos supervivientes le contarán partes de la historia. Al explorar el castillo se encontrara con objetos que forzarán a la protagonista a contar historias en forma de monólogos, que complementaran a la historia principal que formaran parte de la "meta-narrativa" con la que se desea que el jugador experimente y disfrute de unos créditos "in-game", en estos se contarán historias sobre los autores y colaboradores del videojuego.

Aunque la narrativa no sea el fuerte del videojuego, se buscará generar en el jugador una sensación de tensión constante, fomentada por el conocimiento de parte de esta historia y los diálogos del personaje. Los únicos puntos de baja tensión que va a encontrar son entre oleadas, donde planificará la defensa, y si lo desea puede explorar el castillo para conocer más detalles sobre el juego, el castillo, o la relación que tiene la protagonista con este.

Recuerdo 1: Al encontrar una sala con viales rotos:

Cuando Cristina encuentre unos viales en el suelo, confirmará los rumores que todo el mundo conocía sobre las investigaciones que realizan en el castillo. Al mismo tiempo que descubrirá porque se inició la Plaga.

Recuerdo 2: Al encontrar una foto del equipo de desarrollo:

Hablará sobre un equipo de desarrollo de videojuegos, que estaba trabajando en un juego basado en este castillo cuando estalló el incidente.

Recuerdo 3: Al encontrar el arco:

Cuando Cristina encuentre el arco, explicará que el arco puede que lo usara algún miembro del equipo de farmacéuticos para defenderse de los zombis, aunque expone que también pudo usarlo el equipo de desarrollo.

Recuerdo 4: Al encontrar la carta de un desarrollador:

Confirmará que el arco pertenecía a Anderson, el jefe del equipo de desarrollo del videojuego "Mota Tower Defense". Se explicará que el equipo de desarrollo se encontraba dentro del castillo cuando ocurrió el incidente. Tenían permiso para recorrer algunas zonas del castillo por parte de la empresa "Ibercare" la precursora de la Plaga.

A los recuerdos les complementarán las historias de los supervivientes que lleguen. El número del superviviente describe la ronda en la que esta historia se introduce.

Las historias siempre se introducen al terminar una ronda en el periodo de pausa.

Historia superviviente #1:

Daniel Bustillo es el primer superviviente que llega al castillo. Trabajaba de químico para Ibercare, tenía que incorporarse a la plantilla que trabajaba en el castillo, pero la plaga le pilló en medio del viaje,

contará a Cristina algún detalle de la información que le ha proporcionado la empresa en relación a su trabajo.

Historia superviviente #5:
Mike Lambert, un virólogo especializado en pandemias, será el segundo superviviente con narrativa en llegar, llegará con la 5ª oleada. Explicará porque el virus se propago tan rápidamente, al mismo tiempo que explicará que existe la posibilidad de que fuera algo premeditado, introduciendo la conspiración gubernamental.

Historia superviviente #10:
El último superviviente con historia, Abel Garrido, es político y el que más información proporcionará sobre lo que está sucediendo. A parte de explicar cómo el mundo se encuentra sumido en el caos, nos confesará que el proyecto no consistía en encontrar una cura contra el cáncer, querían reducir grupos concretos de población.

Personaje
Cristina, una ingeniera en diseño industrial, sabe dónde puede refugiarse de la plaga. Se acuerda de su trabajo de fin de grado, se basó en el castillo de la Mota en medina del Campo, Valladolid, y sabe perfectamente el potencial defensivo de esa fortaleza medieval, y sabe que sus defensas se pueden rehabilitar fácilmente, en el peor de los casos es un sitio discreto y grande donde guarecerse de los zombis. Lo que Cris no sabe es que dentro del castillo encontrará pruebas de la utilización de la vacuna AN_HR3 en niños, porqué se utilizó el castillo como base de aplicación de la vacuna, y lo que ocurrió con el equipo lo irá descubriendo a medida que recorra el castillo.

7. Multimedia

Música Principal
El juego tendrá dos temas principales el primero, de temática clásica tranquila y relajante, para los periodos de descanso en cada ronda. El segundo se intentará que sea algo estridente y de ritmo muy animado, para aumentar la tensión del jugador y buscar, mediante el ritmo acelerado, mejorar su respuesta ante las oleadas de zombis.

Ambos temas, al igual que el resto de ficheros multimedia, se elegirán en fase de desarrollo para que encajen lo mejor posible con las escenas del juego.

Sonidos Ambiente
Se utilizarán sonidos ambientales para aumentar la sensación de inmersión en el videojuego.

Se clasifican en tres tipos de sonidos ambientales:

Sonido Entorno:
Se utilizará un sonido de entorno complementario al tema principal en las fases de descanso, ejemplos de este tipo de sonidos son: cantos de pájaro, viento, etcétera.

Voces humanas:
Gritos, y sonidos que parezcan humanos durante todo el juego, simulando las voces de los supervivientes. Buscarán aumentar la sensación de inmersión. También servirán de aviso cuando el enemigo esté en la puerta principal o cuando esta esté a punto de ser destruida.

Ruidos Zombis:
Se añadirán ruidos de zombis durante toda la fase de defensa de oleadas, complementarios al tema principal correspondiente. Los sonidos generados por las hordas de zombis deberán ser altos e impactantes para el jugador a fin de aumentar la tensión y mejorar la sensación de terror.

Doblaje
Parte de la historia será narrada en forma de monólogos por parte de la protagonista. Se realizará un trabajo de doblaje para esta parte.

Bibliografía

[1] «https://dle.rae.es/juego,» [En línea].

[2] C. R. d. Álamo, *Uso de Realidad Virtual para recorridos sobre edificios historicos,* Valladolid: UVA, 2018.

[3] «https://www.educativa.com/blog-articulos/gamificacion-el-aprendizaje-divertido/,» [En línea].

[4] C. C. Abt, «Serious Games.,» *New York: Viking Press,* 1970.

[5] J. V. P. Alfonso, «https://web.archive.org/web/20070227153854/http://www.exelweiss.com/blog/37/advergaming-cuestiones-basicas/,» [En línea].

[6] L. y. Z. Hunicke, «“Challenges in Games AI",» de *National Conference of Artificial Intelligence*, USA, 2004.

[7] J. Schell, The Art of Game Design.

[8] «https://latam.historyplay.tv/hoy-en-la-historia/se-lanza-tennis-two-considerado-el-primer-videojuego-de-la-historia,» [En línea].

[9] L. Booker, «https://www.kotaku.com.au/2008/07/pandemic_working_on_new_open_world_sandbox_ip/,» [En línea].

[10] R. Bartle, «HEARTS, CLUBS, DIAMONDS, SPADES: PLAYERS WHO SUIT MUDS,» 1996.

[11] «https://www.thefreedictionary.com/glitch,» [En línea].

[12] «https://www.hobbyconsolas.com/noticias/10-juegos-mas-populares-twitch-2018-353207,» [En línea].

[13] «https://dle.rae.es/srv/fetch?id=VH7cofQ,» [En línea].

[14] D. Pogue, «https://www.scientificamerican.com/article/pogue-6-electronic-devices-you-can-control-with-your-thoughts/,» 2012. [En línea].

[15] C. R. d. Álamo, Uso de Realidad Virtual para recorridos sobre edificios históricos, Valladolid: Uva, 2018.

[16] A. Fernández, GDD Mota Tower Defense.

[17] «https://support.steampowered.com/kb_article.php?ref=1131-WSFG-3320&l=spanish,» [En línea].

[18] D. Pogue, «The Secret History of ‘Easter Eggs’,» *The new York Times,* 8 Agosto 2019.

[19] «https://web.archive.org/web/20050305102027/http://www.irosf.com/q/zine/article/10013,» [En línea].

[20] «https://tldp.org/HOWTO/Software-Proj-Mgmt-HOWTO/users.html#ALPHABETA,» [En línea].

[21] «http://www.gamerdic.es/termino/aventura-grafica/,» [En línea].

[22] «https://ecetia.com/2013/05/existen-1200-millones-de-videojugadores-en-el-mundo,» [En línea].

[23] «https://www.europapress.es/portaltic/videojuegos/noticia-43-jugadores-videojuegos-son-mujeres-2019-son-19-mas-hace-dos-anos-20190531144706.html,» [En línea].

[24] C. Livingston, «The battle royale games of 2021,» *https://www.pcgamer.com/battle-royale-games/,* 2021.

[25] «https://www.redbull.com/cl-es/los-t%C3%ADtulos-m%C3%A1s-jugados-actualmente-en-pc,» [En línea].

[26] «https://www.mcvuk.com/business-news/moba-the-story-so-far/,» [En línea].

[27] «https://hipertextual.com/2019/03/fortnite-250-millones-jugadores,» [En línea].

[28] «https://es.digitaltrends.com/realidad-virtual/mejores-juegos-htc-vive/,» [En línea].

[29] «https://web.archive.org/web/20071111013522/http://www.punch.co.uk/cartoonhistory02.html, » [En línea].

[30] «https://peyda.itch.io/towers-guard,» [En línea].

[31] «https://yrgo-game-creator.itch.io/medieval-archer,» [En línea].

[32] «https://www.oculus.com/experiences/rift/1184265584927845/?locale=es_ES,» [En línea].

[33] «https://www.oculus.com/experiences/rift/2773876492638340/,» [En línea].

[34] «https://devpost.com/software/castle-defense-ghcjx1,» [En línea].

[35] «https://survivr-game.com/game/tower-defense-vr,» [En línea].

[36] «http://www.wrackgame.com/?ref=steemhunt,» [En línea].

[37] A. C. Carrasco, «https://blogs.upm.es/observatoriogate/2018/07/04/que-es-un-motor-de-videojuegos/,» [En línea].

[38] A. Appel, «Some techniques for shading machine renderings of solids,» *IBM Research Center.*

[39] «https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html,» [En línea].

[40] «https://unity.com/es,» [En línea].

[41] «https://developer.valvesoftware.com/wiki/Source_Engine_Features,» [En línea].

[42] «http://osvr.github.io/whitepapers/introduction_to_osvr/,» [En línea].

[43] «https://www.yoyogames.com/,» [En línea].

[44] I. Ramos Salavert y M. D. Lozano Pérez, Ingeniería del software y bases de datos: tendencias actuales, Castilla la Mancha: Universidad de Castilla La Mancha, 2000.

[45] «https://code.visualstudio.com/,» [En línea].

[46] «http://www.sublimelinter.com/en/v3.10.10/about.html,» [En línea].

[47] «https://visualstudio.microsoft.com/es/,» [En línea].

[48] «https://azure.microsoft.com/es-es/,» [En línea].

[49] «https://docs.unity3d.com/Manual/UsingTheEditor.html,» [En línea].

[50] «https://docs.unity3d.com/Manual/ScriptingConcepts.html,» [En línea].

[51] «https://learn.unity.com/tutorial/assets-resources-and-assetbundles#5c7f8528edbc2a002053b5a6,» [En línea].

[52] P. Zackariasson y T. L. Wilson, The Video Game Industry: Formation, Present State, and Future, 2012.

[53] «https://www.boe.es/biblioteca_juridica/codigos/codigo.php?modo=1&id=037_Prevencion_de_riesgos_laborales,» [En línea].

[54] «https://www.gamerdic.es/termino/bug/#:~:text=Un%20bug%20es%20un%20error,o%20directamente%20no%20permitir%20ejecutarlo.,» [En línea].

[55] «https://www.gamerdic.es/termino/glitch,» [En línea].

[56] «https://es.ign.com/reportaje/92847/feature/la-curva-de-dificultad-en-videojuegos,» [En línea].

[57] «https://protecciondatos-lopd.com/empresas/datos-especialmente-protegidos-sensibles/#LOPD_2019,» [En línea].

[58] «https://es.wikipedia.org,» [En línea].

[59] «https://es.wikipedia.org/wiki/Ludificaci%C3%B3n,» [En línea].

[60] «http://ougam.ucr.ac.cr/index.php/comunidad/guia/que-es-un-area-de-influencia,» [En línea].

[61] C. R. d. Álamo, Uso de Realidad Virtual para recorridos sobre edificios históricos, Valladolid: Universidad de Valladolid, 2018.

[62] «http://www.desconsolados.com/analisis/beat-saber/,» [En línea].

[63] «http://www.uva.es/export/sites/uva/2.docencia/2.01.grados/2.01.05.areaestudiantes/_documentos/Normativa-trabajo-fin-de-grado.pdf,» [En línea].