Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

MÁSTER UNIVERSITARIO EN DESARROLLO DE VIDEOJUEGOS

Trabajo Fin de Máster

Conceptualización y Desarrollo de un Action RPG TPV en UNREAL para PC

Concept and Development of a PC aRPG video game with Unreal Engine 4

Autor Pablo Suárez López

D./Dña. **Vanesa Muñoz Cruz**, con N.I.F. 78698687-R profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A

Que la presente memoria titulada:

“Conceptualización y Desarrollo de un videojuego aRPG con Unreal Engine 4 para PC”

ha sido realizada bajo su dirección por D. Pablo Suárez López, con N.I.F. 54.063.860-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en San Cristóbal de la Laguna, a 7 de Septiembre de 2021.

Agradecimientos

A los componentes de Tirando Rol por crear esta historia y permitir que la convirtiera en un videojuego con este proyecto.

A la tutora del TFM por su tiempo y dedicación en este proyecto.

Resumen

El objetivo de este proyecto ha sido la elaboración de un prototipo de videojuego action Rol Play Game (aRPG) pasando por todas las fases del desarrollo, desde la conceptualización hasta el prototipado del mismo.

En este ensayo hemos cubierto la planificación y elaboración de un Documento de Diseño de Videojuego (GDD) en el que hemos incluido todos los aspectos técnicos, el guión, las mecánicas, personajes, niveles del juego así como los concept art y referencias para el desarrollo del prototipo.

En una segunda parte hemos explicado las clases principales del juego y los elementos que las componen, asimismo se han añadido imágenes del código con una breve explicación del mismo.

Palabras clave: videojuego, aRPG, Unreal Engine 4, UE4.

Abstract

The main aim of this project has been the development of an action Role Play Game (aRPG) videogame prototype passing through every development stage, since the concept through the prototype itself.

This essay has covered the planification and production of a Game Design Document (GDD) in which we have included every single technical aspect, the script, mechanics, characters, game levels as well as the concept art and references for the prototype development.

In the second part we have explained the game's main classes and its compounding elements, likewise we have added images of the code with a brief explanation

Keywords: videogame, aRPG, Unreal Engine 4, UE4

Índice

Capítulo 1
Introducción

Este proyecto pretende llevar a cabo el desarrollo de un videojuego desde su conceptualización hasta el prototipado.

En una primera fase desarrollaremos la idea y definiremos un público objetivo para marcar el camino a seguir de manera que el videojuego sea lo más rentable posible. Se decidirá el género del juego, las mecánicas, los flujos de partida, la historia y personajes, así como los diferentes niveles del mismo.

En una segunda parte expondremos las tecnologías que hemos utilizado para desarrollar el prototipo, así como una explicación de los diferentes códigos utilizados para llevarlo a cabo.

Los reglas de juego que vamos a utilizar estarán basados en el sistema de reglas D20 [1], que posee una Licencia de Juego Abierto (OGL [2]). Esto servirá de referencia a la hora de desarrollar las mecánicas del juego, ya que se tendrán que adaptar dichas reglas, pensadas para jugar por turnos y en un tablero, a un videojuego en tiempo real. Utilizaremos estas reglas que ya están definidas y validadas, ya que crear un nuevo sistema de reglas sería una tarea inviable para un proyecto individual.

Para este proyecto hemos realizado una colaboración con los compañeros de Tirando Rol [3], un podcast de juego actual de Dragones y Mazmorras (Dungeons and Dragons) en un mundo creado exclusivamente por ellos. La historia y el mundo que desarrollaremos será una adaptación de la obra original desarrollada por Ulises Martínez, el narrador de Tirando Rol; de manera que nuestro juego se entremezclará con las aventuras desarrolladas en el podcast, tal y como se verá en el capítulo 2.

1.1. Antecedentes

En la actualidad los videojuegos aRPG se encuentran entre los géneros más jugados, como se puede observar en la Figura 1. Su popularidad aumentó con la llegada de Dark Souls en 2011, que sentó las bases de los aRPG actuales.

Gaming genres with the biggest increase in popularity in 2021

Rank	Gaming genre	2020 peak April Twitch viewers	2021 peak April Twitch viewers	Difference in viewership	Percentage increase (%)
1	Action RPG	36,896	336,983	+300,087	+813%
2	Real-Time Strategy	13,136	76,351	+63,215	+481%
3	Fighting & Martial Arts	53,006	255,760	+202,754	+383%
4	Puzzle	88,104	386,571	+298,467	+339%

Figura 1. Popularidad de los aRPG. [4]

Más tarde, The Witcher 3 revolucionaría el apartado gráfico con un mundo abierto con una calidad, tanto técnica como gráfica sin precedentes, siendo uno de los títulos más vendidos en la historia de los videojuegos. A partir de entonces la cantidad de títulos bajo este género se disparó, dando lugar a grandes IPs como Bloodborne, Diablo III, Skyrim, Sekiro: Shadows Die Twice y muchos otros. En la Figura 2 se muestra el top de ventas global en Steam, donde vemos que los aRPG están entre los más vendidos en el mundo [5]

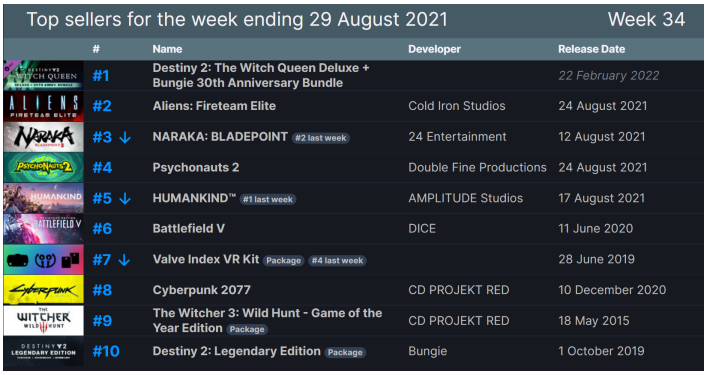


Figura 2. Top ventas global en Steam

Sus orígenes se encuentran en los juegos de rol convencionales, tales como Dragones y Mazmorras, en los cuales se actuaba por turnos y se lanzaban dados para realizar acciones dentro de un mundo de fantasía. Los primeros videojuegos RPG que vieron la luz se basaban en "*Dungeon Crawlers donde avanzábamos dando órdenes de texto* [6]"; para luego evolucionar a pasos agigantados hasta lo que conocemos hoy en día como videojuego de rol, o RPG.

A partir de este momento los títulos tenían tantas diferencias con los RPG originales que tuvieron que crear diferentes subgéneros para abarcar el amplio catálogo que se había desarrollado. Entre estos subgéneros encontramos los

Roguelike, Eastern RPG (entre los que encontramos los más conocidos creados en japón, jRPG); los Tactical RPG, Shooter RPG y los Action RPG, entre otros. [7]

Realmente, la gran mayoría de juegos suelen combinar diferentes subgéneros de RPG, e incluso algunos lo hacen con otros géneros completamente diferentes. También podemos encontrar juegos englobados en un sólo género, además de sagas con diferentes subgéneros. Un ejemplo claro de esto se trata de Pokemon; como podemos ver en la Figura 3, IP principal es claramente un JRPG, y complementa su catálogo con distintas sagas (o juegos) de subgéneros diferentes como Pokémon Mundo Misterioso, un Roguelike sin muerte permanente; o Pokémon Conquest que es un Tactical RPG. [8]

Figura 3. Diferentes sagas y géneros de Pokémon.

Es en este punto donde encontramos los action RPG, que difieren de sus padres en que la acción *"sucede en tiempo real y no por turnos"* [9]. Las características de este género consisten en que se juega con un único personaje que va mejorando sus características y se combate con él en primera y/o tercera persona; y que el videojuego transcurre en tiempo real, sin posibilidad de pausar el tiempo de juego para pensar en una estrategia.

1.2. Objetivo General

El objetivo principal de este proyecto es el desarrollo de una demo jugable de un videojuego RPG de acción en tercera persona con el motor Unreal Engine 4, en adelante UE4.

Con este prototipo podré adquirir los conocimientos básicos necesarios para desarrollar videojuegos con esta plataforma, así como crear un porfolio con el que mostrar mis capacidades.

1.3. Objetivos Específicos

Como objetivos específicos podemos encontrar:

- Conocer en profundidad el motor UE4 para el desarrollo de videojuegos en tercera persona.
- Descubrir y comprender el flujo de trabajo óptimo entre los programas de modelado y UE4.
- Desarrollar el Documento de Diseño de un Videojuego (GDD).
- Planificar y desarrollar un sistema de inventario, estadísticas y combate.
- Conceptualizar y crear un nivel de juego.
- Desarrollar una demo jugable.

1.4. Estado Actual de la Industria

La industria de los videojuegos es un mercado en alza, actualmente se trata de una de las industrias de ocio más rentables a nivel mundial, *"llegando a facturar 152.100 millones de dólares (133.670 millones de euros) en 2019"* [10]. Tras la pandemia sufrida en 2020 *"nos encontramos con que la industria del videojuego ha crecido un 20% respecto al año 2019 en todo el mundo hasta llegar a un montante de 174,9 mil millones de dólares en ventas, tanto de hardware como de software." "De hecho, tal ha sido el crecimiento, que la industria del ocio electrónico ha sido capaz de recaudar más ingresos por ventas que otras grandes formas de ocio como el deporte o el cine en Estados Unidos, y lo que es más, de manera conjunta."* [11].

Por otro lado, la evolución del género Acción ha sido al alza, como podemos ver en la Figura 4, donde se muestra la distribución de los géneros de videojuegos españoles, obteniendo un 10,8% de la cuota de mercado, siendo el género más popular según las métricas elaboradas por la Asociación de Videojuegos de España (AEVI) con más de 2,5 millones de unidades vendidas. [12]

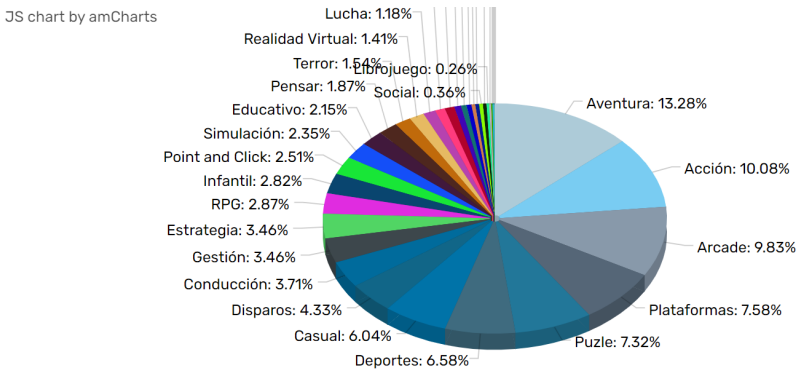


Figura 4. Distribución porcentual de videojuegos españoles por géneros [13]

En cuanto a los usuarios, tal como muestra la Figura 5, las métricas de videojugadores según AEVI en 2018 informan que hay más de 15 millones de

españoles usuarios de videojuegos, lo que supone prácticamente la mitad de la población en la franja de edad de los 6 a 64 años. Además, los españoles dedican una media de 6,7 horas a la semana a esta actividad, por debajo aún de Reino Unido (11,6 h/s), Alemania (8,3 h/s) y Francia (8,6 h/s). El grueso de los usuarios se encuentra entre los 15 y 44 años, siendo las consolas, smartphones y PC los dispositivos más utilizados para jugar.

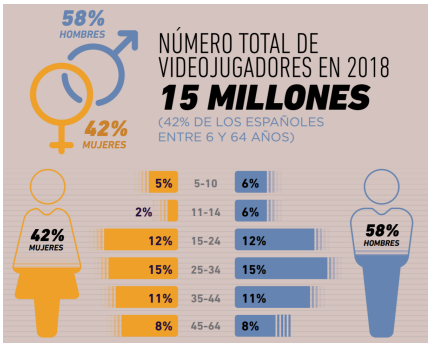


Figura 5. Métricas de videojugadores según AEVI en 2018

Capítulo 2
Documento de Diseño de Videojuegos

2.1. Información Técnica

Este es el documento de diseño del Proyecto aRPG desarrollado en este TFM, en el cual vamos a plasmar todos los elementos que debe incluir el juego. Tal y como hemos comentado en la introducción, la historia sobre la que desarrollaremos nuestro prototipo será una colaboración con el podcast Tirando Rol. Además, las reglas que vamos a utilizar serán, como también se ha indicado anteriormente, una adaptación del sistema de reglas D20, de licencia de juego abierto.

2.1.1. Concepto del juego

Hace 1.300 años el nigromante Salo Myoel estuvo a punto de conseguir un poder tal que ni los mismísimos dioses le podrían haber hecho frente, pero gracias a cinco valerosos héroes que dieron la vida para pararle consiguieron sellar al hechicero, sin embargo, al hacerlo, la magia quedó sellada con él.

En Proyecto aRPG encarnaremos a uno de los cinco aventureros que buscarán rescatar al continente de Cyrin de un mal que se acerca. La Magia ha desaparecido por siglos, pero ahora todo el poder está resurgiendo, incluyendo lo que hace más de mil años intentaron desterrar.

Junto con tus compañeros deberás investigar traiciones, detener rituales místicos, y luchar contra criaturas de las que sólo habías oído en cuentos de viejas, con el objetivo de salvar a Cyrin y a todos sus habitantes.

2.1.2. Características principales

Como nuestro héroe dispondrás de una serie de características con las que avanzar en la historia, entre las que se encuentran:

Juega como uno de los personajes del juego original: Podrás elegir entre los cinco personajes de la historia original de Tirando rol, para ello deberás encarnar la fuerza bruta de un bárbaro, el ingenio de un artificiero, la destreza de una exploradora, el coraje de un monje o la sabiduría de un clérigo.

Mejora a tu héroe: Un amplio sistema de características y talentos donde podrás personalizar cada aspecto de tu personaje antes de lanzarte a la acción. Cuenta

con unos atributos principales que potenciarán las habilidades de tu avatar, además de habilidades de combate y magia para personalizar al máximo a tu héroe.

Desarrolla tus habilidades: Como todo gran guerrero, podrás optar por combatir con armas, con hechizos, o una mezcla de ambos. Para ello deberás aprender de los mentores, practicar a diario y desarrollar tus talentos mientras descubres el maravilloso mundo de Cyrin.

Construye tu propia historia: Todas las misiones del juego tendrán repercusión en la historia principal, ya sea de manera directa o indirecta. Esto hace que el hilo narrativo varíe en función de si las llevas a cabo y cómo las completes.

Explora el mundo de Cyrin: Investiga el origen de la magia mientras descubres un mundo lleno de criaturas fantásticas y mazmorras llenas de tesoros con las que mejorar a tu personaje para convertirte en un gran héroe.

2.1.3. Género

Proyecto aRPG recoge elementos de diferentes géneros.

Juego de rol: Caracterizados por la interacción con el personaje, una historia profunda y una evolución del personaje a medida que la historia avanza. Proyecto aRPG se basa en este estilo al meterse en el papel del héroe de la historia, además de añadir un apartado de personalización y mejoras del personaje.

Acción en tercera persona: Los videojuegos de acción requieren que el jugador haga uso de sus reflejos, puntería y habilidad, a menudo en un contexto de combate o de superación de obstáculos y peligros. En Proyecto aRPG nos encontramos con un sistema de combate en tiempo real en el que el jugador tendrá que exprimir sus habilidades y tiempo de reacción a la vez que machaca botones para crear combinaciones poderosas.

Aventura: Son videojuegos en los que el protagonista debe avanzar en la trama interactuando con diversos personajes y objetos. En Proyecto RPG tendrás que dar vida a un personaje que trata de lograr sus sueños, a la vez que interactúa con el mundo para descubrir los misterios que oculta.

Sandbox: Se trata de videojuegos en los que se incorporan mecánicas y elementos de diseño que propician la libertad de acciones del jugador. Por otro lado, se incorporan las mecánicas lentamente de manera que el jugador experimenta con ellas poco a poco.

2.1.4. Plataformas

El juego estará dirigido a PC.

2.1.5. Público objetivo

Hemos desarrollado este juego teniendo en cuenta un público objetivo, principalmente, varón joven entre 15 y 25 años al que le gustan los videojuegos y dedica varias horas a la semana a jugar. Deben gustarle la aventura y exploración, así como la personalización de un personaje tanto en el apartado estético como mecánico.

2.2. Guión

2.2.1. Historia principal

2.2.1.1. Cinemática inicial

Han pasado 13 siglos desde que la magia se podía sentir en cualquier parte del mundo, 13 siglos desde que los Orcos y Goblins rondaban los caminos de la tierra y los gigantes eran un peligro para los viajeros en las montañas y los valles; poco más de 1.300 años desde la Gran Guerra Mágica; y aunque las generaciones actuales lo vean todo como historias exageradas de un tiempo en el que los hombres eran más inocentes, las cicatrices que nos dejaron aún son evidentes y están más vivas que nunca... Solo basta ver cómo vivimos y prestar un poco de atención.

O por qué crees, mi querido joven, que el entrenamiento marcial y estar unos años en la milicia es obligatorio en casi todas las ciudades sin importar la raza ¿Crees que jóvenes como tú deberían estar entrenados y preparados para la batalla si el miedo no estuviera presente en nuestra sociedad como una astilla de vidrio en una herida? Claro que no.

Las cicatrices siguen en el inconsciente del continente de Cyrin, no importa si eres un orgulloso elfo, un enano, un gnomo o perteneces a cualquier otra sociedad, es parte de lo que te ha formado, aunque no te des cuenta. Es por eso que, aunque llevemos más de mil años en paz, las naciones más grandes continúan gastando recursos en ejércitos y los puestos militares más altos son tan

bien vistos en la sociedad.Los regentes no son tontos, tenemos que estar listos, para que no pase lo de la última vez.

Los ancianos de las razas más longevas te lo pueden contar, aunque algunos tienen la mente un poco perdida por los años, ¿cómo era la vida antes del día del silencio, antes de que cinco héroes realizaran el ritual que salvó al mundo?

Eidnar, el clérigo que con su conocimiento sobre la palabra de los dioses y el orden del mundo opacaba cualquier biblioteca; Agreen, la hechicera cuyo poder era tan inmenso que solo era opacado por su belleza; Agro, medio hermano de Agreen, el Paladín más justo de todos y cuya espada podía partir montañas; Torag el Bárbaro, fuerte como la furia de un volcán; y Ludya, una druida que tenía una conexión con la naturaleza como nadie antes que ella. Ellos utilizaron todo su poder y conocimiento para apagar toda la magia en algún lugar de la isla de Dragharim, al sur del continente. Un movimiento muy drástico para el mundo, dirían algunos, pero la solución es fácil si la pones en una balanza... Un mundo sin magia, o un mundo dominado por el terror del Salo Myoel y sus ejércitos de temibles criaturas, en el que todos seríamos esclavos ¿Qué hubieras elegido?

Dicen que cuando la magia se apagó, hasta el aire se sentía distinto, y por uno o dos segundos, todo el mundo quedó callado; no se podía escuchar ni el aleteo de un insecto, y después de ese segundo tuvimos que acostumbrarnos a una nueva vida... Ciudades flotantes cayeron, las hadas desaparecieron, todo aquel que podía tocar la fibra del velo mágico y utilizarlo a su favor... simplemente perdió su poder, incluyendo Salo, el Cruel; esa era la idea. Y sin el poder de levantar a los muertos e invocar demonios, solo fue cuestión de tiempo, semanas incluso, para destruir su reino de terror y eliminar su desorganizado ejército de orcos y gigantes.

De los dioses mejor no hablamos, pues sin la conexión mágica en el mundo, sus milagros desaparecieron. Aunque eso sí, la fe de las personas sigue viva, seguimos adorándolos. Algunos piensan que es por costumbre, pero prefiero creer que es por esperanza.

Aunque permíteme contarte un secreto... Si eres curioso seguro lo habrás escuchado, se dice que la magia ha vuelto, y en los últimos años, los avistamientos de las maravillas se han ido multiplicando... Quien sabe, tal vez solo es cuestión de tiempo para que regrese, aunque esperemos que no traiga consigo la maldad que la obligó a apagarse en primer lugar.

2.2.1.2. Guión completo de la historia

Llegada al Campamento Erdia

Los personajes se conocen en el Campamento, cada uno de ellos llega por distintas razones. Este año parece ser uno como cualquier otro, pero algo extraño se está tramando y serán ellos quienes deberán descubrir de qué se trata.

Encuentro con el grupo (Presentación de los miembros)

Encuentro con la directora (Presentación del torneo)

Pruebas con los profesores

Una vez se hayan hecho las presentaciones, los profesores comenzarán a realizar una serie de pruebas a todos los recién llegados para averiguar sus capacidades. A efectos de juego se tratará de un tutorial de los comandos básicos para explicar orgánicamente al jugador los controles principales del juego.

En el caso de Fili se tratará de un puzzle que consistirá en equipar y usar diferentes objetos para poder avanzar por la prueba. Phos Grung te animará a entablar un combate con él eligiendo cualquier arma a tu elección. Finalmente, el profesor Caprice te hará internarte en el bosque para encontrar una serie de ingredientes, para ello deberás desplazarte por el mapa, saltar y escalar por diferentes sitios.

Torneo contra el Roble

Llegada del colegio del roble (Presentación profesores y alumnos)

Pruebas del torneo

Tiro con arco

Combate

Carrera de obstáculos

Exploración

La cabaña de Martell

Ganen o pierdan, los alumnos del roble se internan en el bosque. Cuando el grupo se da cuenta los siguen hasta la cabaña de Martell, pero al llegar, los alumnos del roble han desaparecido. Pensando que pueden estar dentro de la cabaña entran y descubren que está todo destrozado por marcas de garras. En la cabaña

encuentran el diario de Martell, donde hay dibujos extraños de un monstruo y entradas sobre transformaciones y locura; y además de esto, descubren que en las cavernas hay escondido algo importante, una niña llamada Nila.

Entre otros objetos que se encuentran en la cabaña, descubren un mapa de las cuevas cercanas al campamento.

El misterioso atacante

Al salir de la casa se encuentran a Balthazar siguiéndoles para robarles la información que acababan de obtener, pero una bestia enorme con forma de lobo los ataca. Tanto si logran vencerla como si no, la bestia huye en dirección a las montañas.

2.3. Mecánicas de Juego

2.3.1. Tipo de cámara

El tipo de cámara que se utilizará será 3D en tercera persona. Se controlará con el ratón y utilizará un componente que en UE4 se denomina Spring Arm que permite a la cámara acercarse al personaje cuando colisiona con algún objeto, en lugar de posicionarse tras él y perder la visualización del jugador. Adicionalmente se podrá controlar con un joystick.

2.3.2. Controles

El desplazamiento del personaje se realizará mediante el input del jugador pudiendo desplazarse en el espacio en 3D. También dispondrá de diferentes acciones de movimiento además del desplazamiento, como saltar y esquivar.

Todos los elementos interactivos se establecerán mediante el sistema de físicas, utilizando para ello las *hitbox*. Estas *hitbox* nos permitirán indicar al sistema cuándo se deben activar los diferentes eventos necesarios para interactuar entre los personajes y el entorno.

En el combate se utilizarán las *hitbox* para comprobar si se ha golpeado a otro actor. Por otro lado, el personaje podrá interactuar con el entorno, como abriendo y cerrando puertas, o recogiendo objetos dispersos por el mapa, mediante estas cajas de colisión; en el momento en que el personaje colisiona con una de ellas, activa el evento que avisa al jugador de que puede interactuar con dicho elemento.

Los controles de combate incluirán el ataque básico, el bloqueo, dos habilidades especiales, una habilidad definitiva y cuatro slots para objetos.

Las colisiones producidas serán:

Personaje - Personaje	Personaje - Arma	Personaje - Objeto
Arma - Arma	Arma - Hechizo	Arma - Objeto
Hechizo - Hechizo	Hechizo - Personaje	Hechizo - Objeto
Objeto - Objeto		

Para ver un listado detallado de las acciones y los diferentes inputs ver Apéndice A. Lista de controles

2.3.3. Guardado y carga

El sistema de guardado y carga se encargará de almacenar los datos necesarios para continuar el juego en cualquier momento. Para ello se creará una estructura de datos y un componente que guardará y cargará todas las variables necesarias para el correcto funcionamiento. Este sistema realizará un guardado automático cada vez que se alcance un punto de guardado, así como cada vez que se acceda a una de las pantallas de menú.

La estructura de datos tendrá dos niveles. En el primero guardaremos toda la información referente al personaje, como su nivel, las características, los objetos y habilidades que posee; y en el segundo nivel almacenaremos el estado del juego, la posición de los actores en el nivel y el estado de los mismos.

2.3.4. Sistema de reglas

2.3.4.1. Características

Las características determinan las capacidades principales de un personaje.

Atributos principales

En Proyecto aRPG existen seis atributos distintos: Fuerza, Destreza, Constitución, Inteligencia, Sabiduría y Carisma. Los tres primeros miden las capacidades físicas del personaje, mientras que los tres siguientes representan sus capacidades mentales y perceptivas.

- ***Fuerza (FUE)*:** Refleja la capacidad de esfuerzo físico del personaje, su musculatura y vigor. Mide la potencia física.
- ***Destreza (DES)*:** Determina los reflejos, el equilibrio y la agilidad. Mide la precisión de movimientos y la capacidad de uso de armas y objetos.
- ***Constitución (CON)*:** Indica el aguante y la vitalidad del personaje, así como su capacidad de soportar daños. Mide la resistencia.
- ***Inteligencia (INT)*:** Determina la capacidad de un personaje para aprender o razonar. Mide el razonamiento y la memoria.
- ***Sabiduría (SAB)*:** Recoge la agudeza sensorial del personaje y su capacidad de advertir su entorno. Mide la percepción y visión
- ***Carisma (CAR):*** Indica su capacidad para interactuar con otros. Mide la fuerza de personalidad.

Cada característica tendrá una puntuación que medirá no sólo las capacidades innatas, sino el entrenamiento y sus aptitudes en actividades relacionadas con dicha característica.

La media humana se encuentra entre el 10 y el 11, reflejando a un adulto promedio. Sin embargo, tras un entrenamiento muy duro y años de experiencia pueden llegar hasta el 18, y solo las grandes leyendas consiguen una puntuación de 20, la puntuación máxima para una característica de un personaje.

Atributos derivados

Los atributos derivados se obtienen realizando diferentes cálculos basados en las características principales. Son los que ofrecen al usuario una mayor comprensión del juego al comprender aquellas características directamente relacionadas con el gameplay.

- **Puntos de golpe (PG):** Representan la resistencia a heridas y la cantidad de daño que un personaje puede sufrir antes de caer inconsciente. Cuando recibe un ataque, los PG del personaje son reducidos en la cantidad reflejada por el daño del mismo.
- **Competencia, bonificador por competencia:** Representa la experiencia de un personaje para llevar a cabo tareas de manera más certera, desde las tareas más comunes como manejar una espada hasta las más complicadas como tallar una gema o realizar complejos hechizos.
- **Habilidad de ataque (Atk):** Mide la capacidad del personaje para golpear a su oponente, conseguir eludir su guardia y causarle daño. Se emplea con todo tipo de armas, ya sean cuerpo a cuerpo o a distancia. El cálculo de la habilidad de ataque se obtiene sumando la capacidad de ataque del arma; el modificador por Fuerza en ataques cuerpo a cuerpo, o Destreza en ataques a distancia; y el bonificador por competencia. Algunas armas

permiten utilizar tanto fuerza como destreza en ataques cuerpo a cuerpo cambiando el tipo de daño.

- **Clase de armadura (CA):** Indica la capacidad del personaje de esquivar y/o bloquear ataques. La CA incluye la armadura y el escudo que lleves, así como la dureza de la piel y otros modificadores. Cualquier personaje puede llevar cualquier tipo de armadura, pero solo podrás utilizarla correctamente si eres competente en dicho tipo, en caso contrario sufrirá penalizadores que pueden ir desde reducir el movimiento hasta impedir el lanzamiento de hechizos.
- **Maná:** Indican la cantidad de conjuros que un personaje lanzador de conjuros puede utilizar, así como el nivel de los conjuros que puede lanzar con ellos.

Características

Las características son las facultades que tiene un personaje para efectuar las acciones más diversas o poner en práctica conocimientos más variados, como trepar o conocer el uso de las plantas. Cada característica está ligada a uno de los atributos principales, de manera que cuanto mayor sea la característica base, mayor será el bonificador que aplique a dicha característica. Se puede encontrar una descripción de las características en las reglas básicas Dungeons & Dragons 5E [14].

Las características que se van a usar en el videojuegos son: Atletismo, Acrobacias, Juego de manos, Sigilo, Conocimiento Arcano, Historia, Investigación, Naturaleza, Religión, Medicina, Percepción, Perspicacia, Supervivencia, Trato con animales, Engaño, Interpretación, Intimidación y Persuasión.

2.3.5. Habilidades

El sistema de habilidades permitirá al jugador realizar acciones extraordinarias, tanto físicas como mágicas.

Las habilidades se desbloquean mediante un árbol de habilidad como el que vemos en la Figura 6, en el que necesitaremos puntos para poder avanzar por el mismo y obtener acceso a mayores poderes.

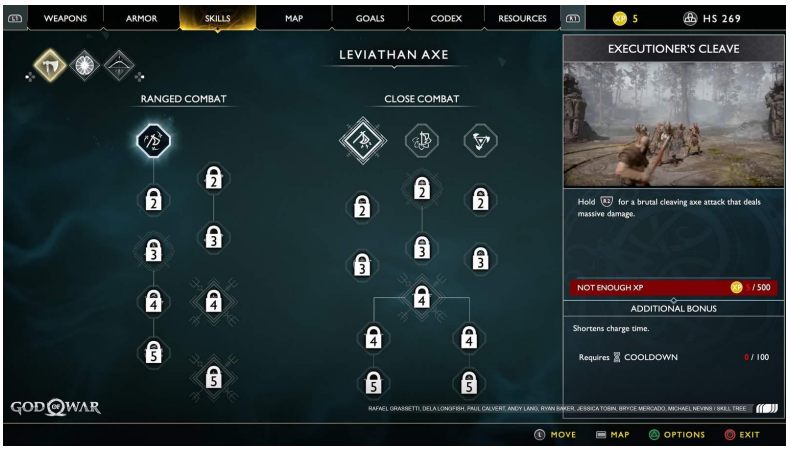


Figura 6. Árbol de habilidades.

El árbol de habilidad será distinto para cada uno de los personajes principales, según la clase a la que pertenezca. Los puntos de habilidad se conseguirán subiendo de nivel.

Se dividen en dos categorías, habilidades físicas y habilidades mágicas.

Las primeras consisten en ataques especiales que pueden realizar los personajes, tales como la furia del bárbaro, que le permite aumentar sus estadísticas de combate por un tiempo determinado; o el Castigo Divino del paladín, que inflige daño radiante adicional al objetivo.

Las habilidades mágicas se denominan Conjuros, y permiten al lanzador invocar bolas de fuego, escupir ácido, lanzar esquirlas de hielo o curar heridas.

2.3.6. Planificación del combate

El combate se desarrollará tanto cuerpo a cuerpo como a distancia.

Para el combate cuerpo a cuerpo, los personajes deben de estar próximos para poder golpear y/o ser golpeados. En el caso de combate a distancia, el ataque debe acertar en el objetivo para poder dañarlo.

En ambos casos se utilizará una caja de colisión o *hitbox*, como podemos ver en la Figura 7, que servirá como indicador de si el ataque ha golpeado al enemigo o no, además de una interfaz que llamaremos *damageable* que utilizaremos para comunicar los actores que intervengan en la colisión detectada.

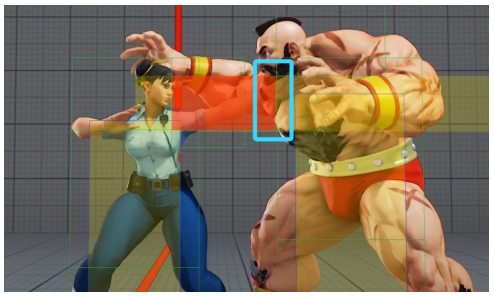


Figura 7. Caja de colisión o hitbox.

El diagrama de la Figura 8 muestra el flujo del combate.

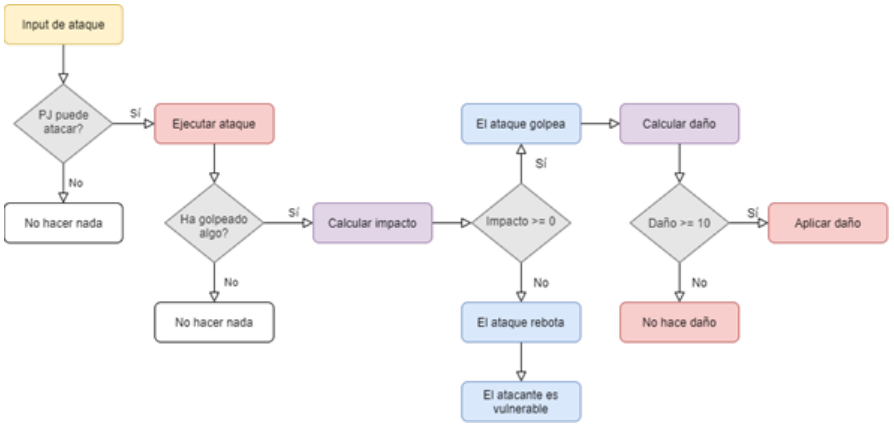


Figura 8. Diagrama de flujo de combate.

Las estadísticas asociadas al combate serán las siguientes:

Daño: Todas las armas producen un daño base determinado, al que sumaremos el bono de fuerza del personaje que la blande para obtener el daño final. Las armas están preparadas para ser blandidas a una o a dos manos, y esto indicará si sumaremos el modificador de fuerza una vez (para armas a una mano) o vez y media (para armas a dos manos). Algunas armas permiten utilizar el modificador de destreza en lugar del de fuerza modificando el tipo de daño que aplican.

Tipo de daño: Indica el origen del daño que va a causar el ataque según el arma que esté blandiendo o el tipo de conjuro a invocar. Se puede encontrar una descripción de los tipos de daño en las reglas básicas Dungeons & Dragons 5E [14]. Son los siguientes: Ácido, Contundente, Cortante, Frío, Fuego, Fuerza, Necrótico, Perforante, Psíquico, Radiante, Relámpago, Trueno y Veneno.

Resistencia al daño: Indica la capacidad de un personaje para disminuir las heridas recibidas de un tipo de daño. Si un personaje tiene resistencia a un tipo de daño, éste sólo sufrirá la mitad del mismo.

Vulnerabilidad al daño: Al contrario que la resistencia, si un personaje es vulnerable a un tipo de daño, recibirá el doble de daño de los ataques basados en dicho tipo.

Curación: Cuando un personaje sufre daño, sus puntos de golpe son reducidos, para recuperarse de estas heridas un personaje puede descansar y/o utilizar conjuros u objetos con la capacidad de curarse. Los puntos de golpe máximos no pueden ser aumentados mediante curación, salvo que la descripción diga lo contrario.

2.4. Estados del juego

El juego tendrá 3 estados: menú principal, juego y pausa. El flujo entre los estados se puede observar en la Figura 9.

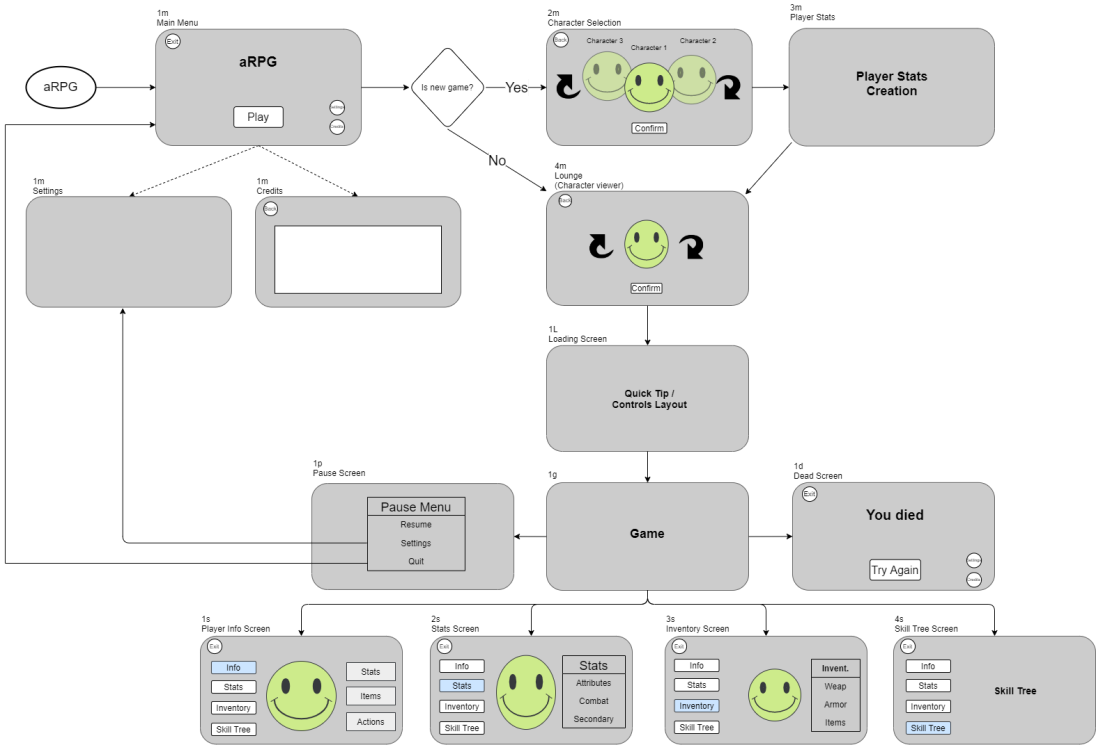


Figura 9. Flujo de los estados de juego.

2.5. Interfaz

Para la interfaz del juego haremos uso de un Manager HUD, este se encargará de comunicar el controlador del personaje con los diferentes estados y pantallas del juego.

A continuación se muestra un concept art de todas las pantallas de interfaz del juego, en las figuras 10, 11, 12, 13, 14, 15, 16 y 17.

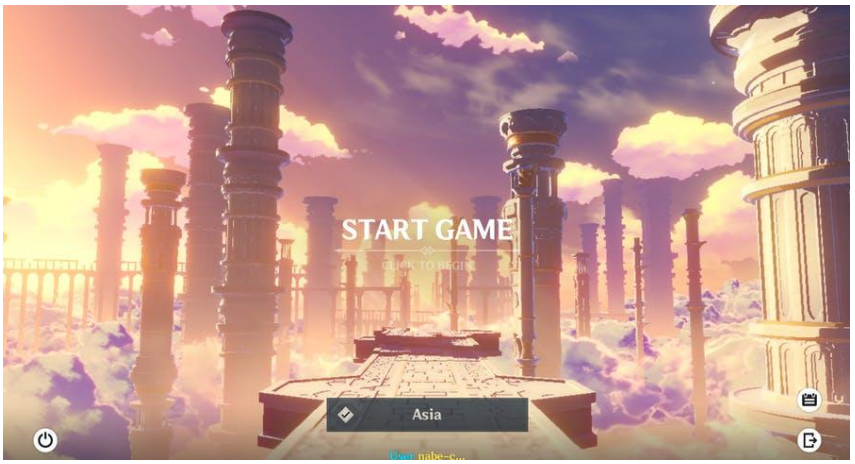


Figura 10. Concept art menú principal

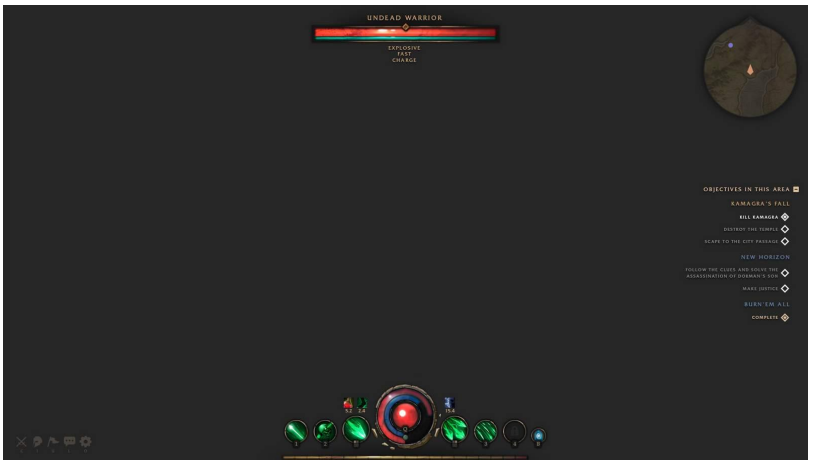


Figura 11. Concept art UI

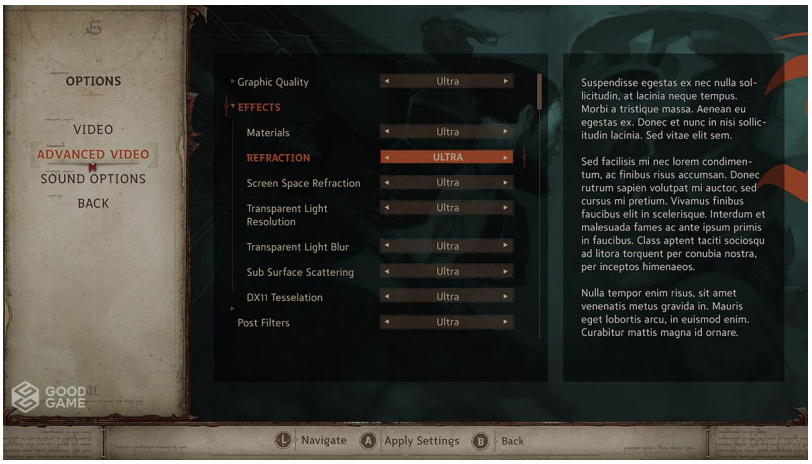


Figura 12. Concept art menú ajustes

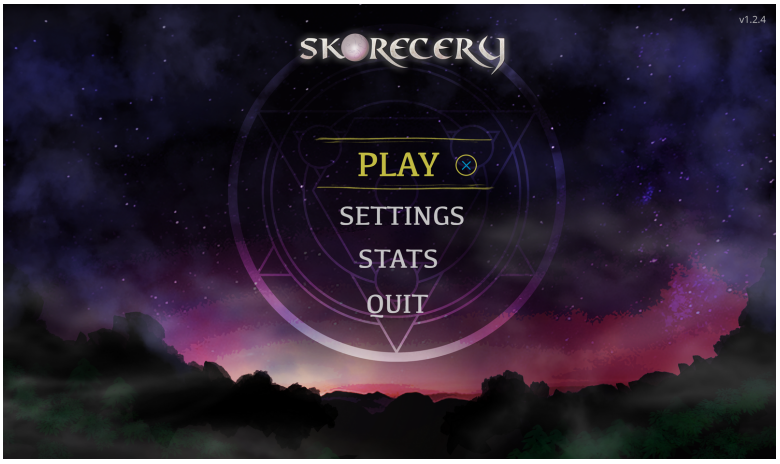


Figura 13. Concept art menú pausa

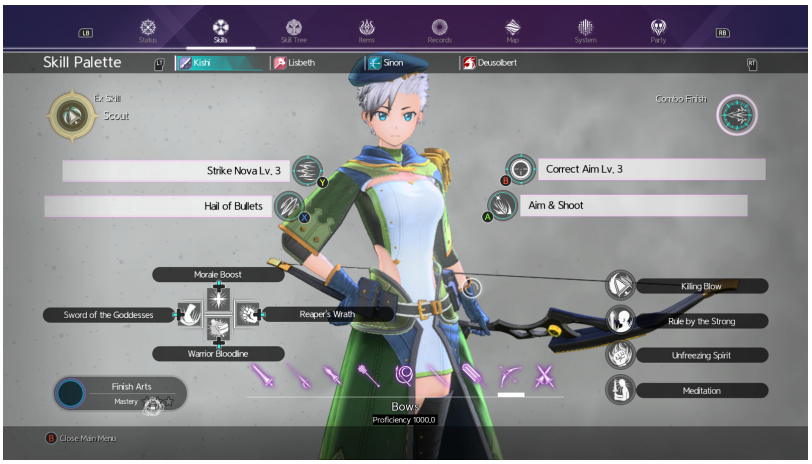


Figura 14. Concept art menú resumen personaje

Figura 15. Concept art menú estadísticas del personaje

Figura 16. Concept art menú inventario

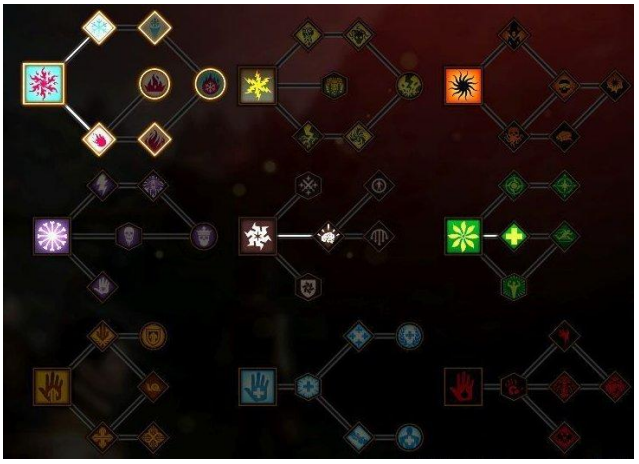


Figura 17. Concept art menú árbol de habilidades

2.6. Progresión del jugador

El jugador tendrá que ir avanzando por la historia a la vez que mejora a su personaje, para ello deberá realizar misiones y luchar contra enemigos que le darán experiencia, lo que le permitirá subir de nivel y desbloquear nuevas habilidades para volverse más poderoso.

Al comienzo del juego el personaje comenzará a nivel 1 con 0 puntos de experiencia (PE). Cada misión que complete, cada arco argumental que termine y cada enemigo al que derrote le dará PE que le permitirá avanzar al siguiente

nivel. Una vez desbloquee un nuevo nivel, tendrá acceso a nuevas habilidades, puntos de mejora de características y objetos de más poder.

2.7. Niveles

El mapa será un mundo abierto dividido en varias zonas, cada zona tendrá una dificultad y estética diferente, de manera que el jugador deberá ir avanzando según su nivel de dificultad y avance en la historia.

Al tratarse de un mundo abierto la cantidad de datos a cargar es enorme, por lo que muchos ordenadores tendrán dificultades para correrlo, por eso utilizaremos el sistema de carga de niveles de UE4 World Composition [14], el cual nos permitirá crear diferentes zonas que se irán cargando a medida que nos acerquemos a ellas. Esto unido a los diferentes LODs (niveles de detalle) nos ayudará a que la carga poligonal en pantalla sea la menor posible, pudiendo jugar al título en ordenadores menos potentes.

2.8. Personajes

El jugador podrá elegir entre cinco personajes predefinidos. Una vez seleccionados podrá modificar sus estadísticas y habilidades, de forma que será un personaje definido por el usuario.

Durante el juego nos encontraremos con diferentes NPCs repartidos por el mapa, son aquellos que no pueden ser controlados por el jugador y encontramos varias clases:

- Comerciantes: El jugador puede interactuar con ellos para comprar y/o vender objetos.
- Secundarios: Son importantes para el transcurso de la historia.
- Peticionarios: El jugador recibirá de ellos encargos a cambio de recompensas.
- Comunes: Personajes “de relleno” que hablan sobre la situación de la zona. En ocasiones pueden ofrecer pistas al jugador de posibles secretos.
- Enemigos: Son personajes que tratarán de luchar contra el jugador. Se dividen en dos tipos según su dificultad: Mob y Boss. Los primeros son enemigos base, con una IA (Inteligencia Artificial) básica que consistirá en atacar al ver al personaje y volver a su localización inicial si el jugador se aleja de su zona de influencia. En cuanto al segundo grupo, se trata de jefes finales con una IA más avanzada, que les permite realizar diferentes acciones según su salud y su localización en el mapa.

Para ver una descripción de los personajes en detalle ver Apéndice B. Personajes.

2.9. Items

Los objetos son elementos con los que los personajes pueden interactuar y utilizar. Se dividen en:

Interactivos: El personaje puede interactuar con ellos para activarlos y desactivarlos. Se puede desde abrir y cerrar puertas y cofres hasta encender lámparas o empujar objetos.

Utilizables: Son objetos que pueden ser usados desde el inventario para activar sus capacidades, por ejemplo una poción o una bomba de humo.

Equipables: Todos aquellos objetos que el personaje puede ponerse. Estos objetos modifican las características del personaje. Se clasifican en armaduras y armas.

2.10. Música y sonido

Se utilizarán assets de sonido de la biblioteca de UE4. Existirán diferentes canciones: una para la ciudad, otra para el bosque, otra para la montaña y otra que se activará cuando entremos en combate. Además de esto utilizaremos diferentes efectos de audio repartidos por el mapa que se podrán escuchar al acercarse a ellos, tales como sonidos de puertas, madera crujiendo, pájaros, etc.

Capítulo 3
Desarrollo del videojuego

En esta sección vamos a explicar cómo hemos desarrollado el juego, explicando algunos de los códigos más importantes. Hemos desarrollado el prototipo en el motor gráfico UE4 utilizando Blueprints, ya que permiten un prototipado rápido, pero como proyecto de futuro se planea pasar el código a C++ de manera que sea reutilizable para futuros proyectos.

Todo el código se encuentra en el prototipo final para su consulta.

3.1. Game Managers

Como estamos trabajando en UE4 hemos decidido implementar el código basándonos en las clases predefinidas del mismo. Por ello hemos utilizado los Game Managers existentes y modificado su comportamiento cuando ha sido necesario.

El objetivo de utilizar diferentes GameManagers consiste en dos partes. La primera es la persistencia de los datos entre niveles, por ello utilizamos el GameInstance, que es una clase que se crea al iniciar la aplicación y no se destruye hasta que se cierra, por lo que nos permite mantener los datos entre los distintos niveles tanto de juego como de menú. La segunda parte consiste en la posibilidad de ampliar la funcionalidad del prototipo añadiendo un modo multijugador, por lo que habrá clases que solo serán accesibles desde el cliente y otras desde el servidor.

A continuación explicaremos cada uno de los Game Managers y para qué los utilizamos.

3.1.1. Game Instance

El Game Instance es el único elemento del juego que persiste entre niveles, es por ello que hemos implementado un GameInstanceManager que se encargará de transmitir todos los datos que necesitamos sean persistentes, tales como el personaje elegido; las estadísticas, habilidades y objetos seleccionados; y el estado de los diferentes actores del juego, como si están muertos o no, si una puerta está abierta o no, o el estado de las misiones del juego.

Cuando el juego se inicia el GameInstanceManager se encarga de comprobar si existen datos de carga previa, y si es así comenzar el juego directamente con esos datos. En caso que no hayan datos previos, se encargará de ir almacenando las

variables necesarias mientras el jugador crea su personaje para la transferencia de datos al cargar el nivel, y una vez se comience a jugar, éste no volverá a ser llamado hasta que el jugador muera o salga de la partida. Es en ese momento es cuando se volverá a activar para recoger los datos necesarios y almacenarlos, ya que es el único actor que persiste entre cargas de niveles.

Si el usuario vuelve a jugar, el GameInstanceManager volverá a repetir el proceso anterior, pero en caso de que quiera salir de la aplicación, se encargará de llamar a las funciones de guardado de datos, para almacenar las variables necesarias para continuar la partida en otro momento.

3.2.1. Game Mode

En UE4 existe un Game Manager llamado GameModeBase, en el que se recogen todos los demás Game Managers que se van a utilizar en el juego. Este primer GameModeBase se puede sobreescribir en cada nivel para utilizar uno diferente. Hemos aprovechado esta funcionalidad para establecer un primer GameModeMenu que se encargará de gestionar el menú principal y el menú de muerte, y otro GameModeGameplay que se encarga de manejar el modo de juego y el menú de pausa.

El GameModeManager se utilizará para indicarle a la aplicación qué controladores tendrá que utilizar en cada momento, de manera que podamos utilizar un controlador distinto según si estamos jugando o seleccionando las habilidades de nuestro personaje en el menú principal, de manera que cada clase se encargará de realizar el trabajo que le corresponde, y no el de otros.

Finalmente, utilizaremos esta clase para inicializar todas las referencias de los demás GameManagers, como podemos ver en la Figura 18, s, ya que es el único elemento al que cualquier clase del juego puede acceder.

Figura 18. Función Initialize Manager Components.

3.2.2. Game State

En este Game Manager almacenaremos el estado de las variables del juego mientras se esté en el nivel de juego, es decir, una vez seleccionemos al personaje con el que jugaremos, cargaremos el mapa, y en este momento el GameInstanceManager se encargará de hacer llegar los valores necesarios al GameStateManager para que este se encargue a partir de entonces.

Una vez el juego haya terminado, ya sea porque el jugador ha muerto o porque ha salido de la partida, el GameStateManager se encargará de enviar los datos al GameInstanceManger para que este los maneje como necesite.

En definitiva, el GameInstanceManager se encarga de gestionar la carga y guardado de datos durante el menú principal y el de muerte; mientras que el GameStateManager se encarga de controlar los datos mientras se esté jugando.

3.2.3. Player Controller

En el Player Controller situaremos todos los eventos de input del jugador, de manera que se encargará de gestionar las llamadas a las funciones necesarias. Hemos utilizado un PlayerController en lugar de implementar los eventos directamente en el PlayerCharacter ya que nos abrirá un abanico de posibilidades a la hora de cambiar de personaje si se implementa la función en un futuro, así como el multijugador.

Al utilizar un PlayerController para los eventos, estamos separando los inputs del jugador de la acción que se realiza, de manera que si queremos realizar cualquier modificación o añadir funcionalidades tan solo tendríamos que referenciar las funciones necesarias en lugar de reescribir o modificar el código existente. Además, con el PlayerController podemos cambiar de personaje simplemente con cambiar el actor en el GameModeManager, de manera que podremos, con los mismos inputs, controlar un humano, un barco o un dragón incluso, ya que la implementación de las acciones se encuentra dentro del PlayerCharacter.

A continuación se muestran los eventos de input de entrada que controlan el movimiento y las acciones del personaje. Todos los eventos de entrada se gestionarán a través del InputManager propio de UE4, que nos permite modificar los botones que el jugador pulsa sin necesidad de cambiar el código.

Como podemos ver en la Figura 19., se recoge el input de entrada del jugador y se obtiene la referencia del HudManager a través del GameModeManager, tras esto se llama al método Toggle Player Inventory HUD para añadir o eliminar de la pantalla el inventario.

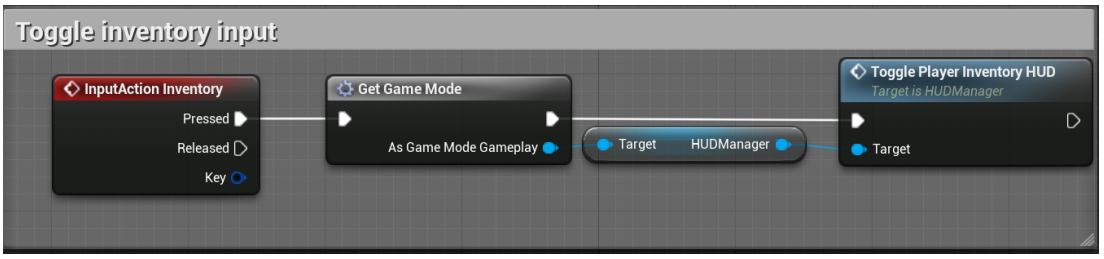


Figura 19. Input Event Inventory.

En la Figura 20, podemos ver como, tras recoger el input del jugador, se realiza un bucle que recorre todos los actores con los que el personaje está colisionando y se comprueba si implementa la interfaz BPI_Interactable. En caso afirmativo envía un mensaje a través de la interfaz al PlayerCharacter para que pueda ejecutar la funcionalidad requerida.

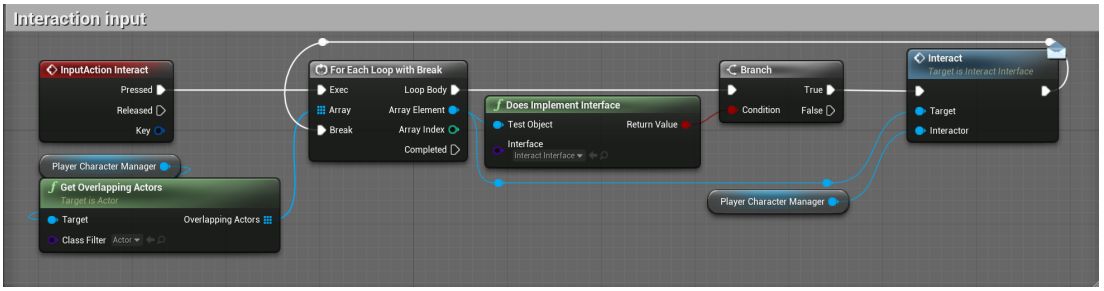


Figura 20. Input Event Interaction.

El script para el movimiento del personaje consta de dos partes, como podemos ver en la Figura 21.

En la primera se añade un movimiento hacia delante o hacia detrás según el valor devuelto por el Input del personaje, esto es, si pulsa el botón adelante, el valor será de 1, y si pulsa el de atrás el valor será -1, en caso contrario será 0.

En la segunda parte se realiza la misma acción anterior, solo que esta vez el valor viene del movimiento hacia la derecha, que será 1, y hacia la izquierda será -1, en caso contrario será 0.

Con estos dos inputs el personaje podrá moverse en el plano hacia cualquier dirección.

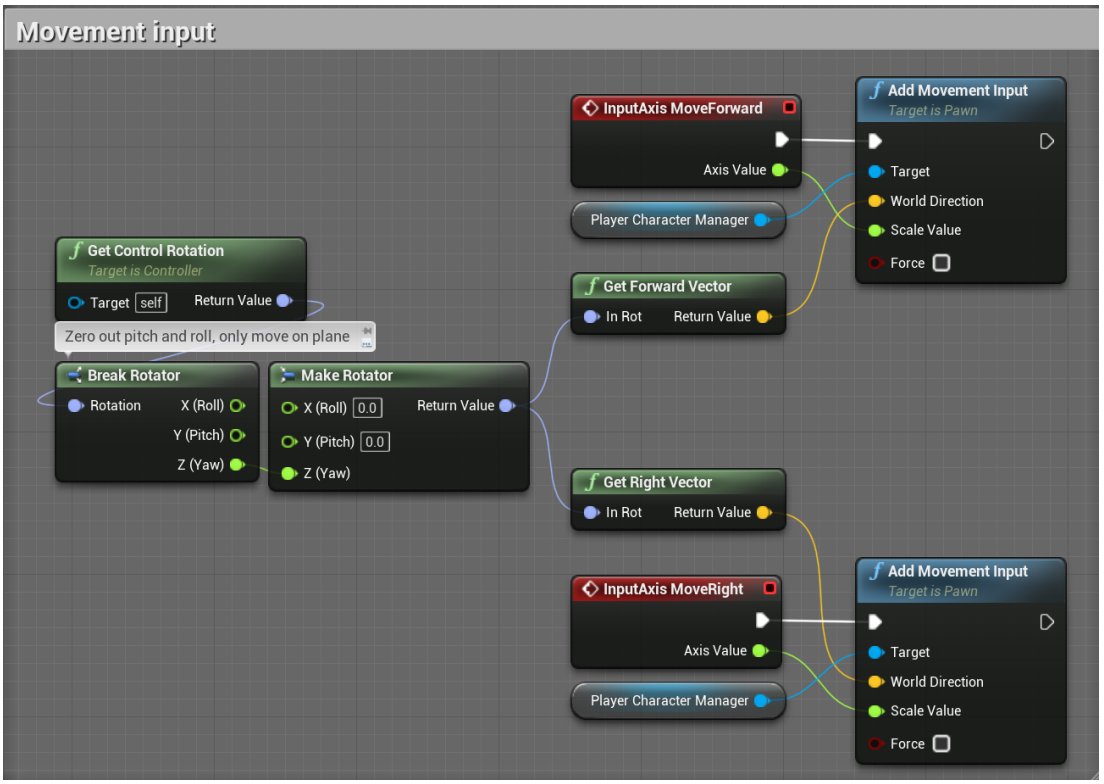


Figura 21. Input Event Movement Direction.

3.2.4. Player Character

Se trata de una clase donde implementaremos la funcionalidad de todas las acciones de input recibidas por el PlayerController, así como los diferentes componentes del personaje, como son las estadísticas, el inventario y las habilidades.

Será, además, el encargado de avisar al GameModeManager para que inicie todos los demás GameManagers. Esto es así, ya que se trata del último elemento en el bucle de juego en inicializarse, por lo que, una vez realice la llamada a la función *InitializeGame* todos los demás componentes estarán cargados en memoria, evitando así cualquier fallo de referencias.

El PlayerCharacter estará compuesto por los siguientes componentes, tal como podemos ver en la Figura 22. Player Character Viewport:

- CapsuleComponent (Root): Una cápsula de colisión que delimita el espacio del personaje en la escena. La utilizaremos para detectar las colisiones del jugador con el terreno y las interacciones con otros actores.
- SkeletalMeshActor: El “cuerpo” del personaje, que cambiará según la selección del mismo.
- SpringArm: Se utiliza para añadir la funcionalidad de contraer y expandir la distancia de la cámara al personaje ante colisiones con otros elementos.
- CámeraComponent: Lo que vemos en la pantalla, la cámara nos devuelve una representación gráfica de nuestra posición en el mapa.
- CharacterMovement: Una clase base de UE4 que permite al PlayerCharacter realizar acciones de movimiento. Además, posee algunas clases de movimiento predefinidas, como el salto.
- InventorySystem: En él recogeremos las variables referentes al sistema de inventario propias del personaje, tales como los objetos iniciales y el tamaño del mismo; así como una referencia para implementar las funciones necesarias.
- StatsSystem: El lugar donde almacenaremos todas las variables y las funcionalidades referentes a las estadísticas del personaje.
- SkillSystem: En este componente almacenaremos las habilidades que el personaje posee, y realizaremos llamadas a sus funciones cuando el jugador decida usar una de ellas.
- CombatSystem: Este componente nos servirá para llamar a las funciones de combate, reproducir animaciones y efectos de impacto, calcular daños y modificar la salud del personaje.

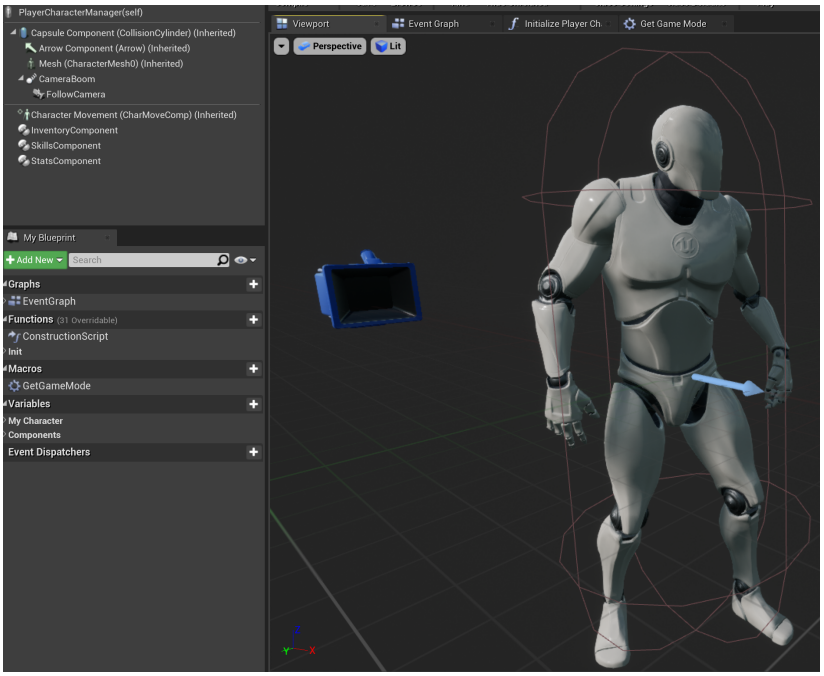


Figura 22. Player Character Viewport.

3.2.5. Player State

En este GameManager almacenaremos todas las variables referentes al personaje para realizar los cálculos necesarios y para guardar la partida. Será el encargado de comunicar a las demás clases el estado actual del jugador, tanto sus estadísticas como su inventario y posición en el mapa.

Utilizaremos esta clase para almacenar los datos que deban persistir más allá de la muerte del jugador, ya que, cuando el personaje muere, el PlayerCharacter es destruido y, al volver a jugar se crea uno nuevo, por lo que los datos se perderían si no se almacenan en una clase más persistente.

3.2.6. HUD Controller

Utilizaremos esta clase para gestionar los diferentes widgets y sus referencias, de esta manera cuando queramos acceder a la información de cualquier interfaz podremos hacerlo mediante las referencias almacenadas en este controlador.

En la Figura 23. Inicialización de la interfaz de usuario (UI) podemos ver cómo se obtiene el controlador del jugador y se crea un nuevo Player HUD. Después de esto, se almacena la información del widget y se añade a la pantalla.

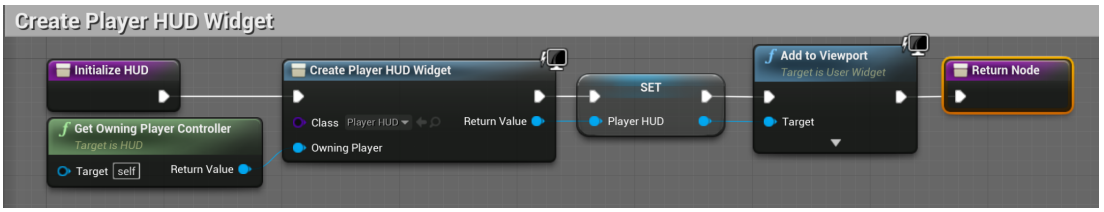


Figura 23. Inicialización de la interfaz de usuario (UI)

3.2. Inventory System

El sistema de inventario permitirá al jugador interactuar con los diferentes objetos posicionados en el mapa. Hemos diseñado el sistema para que sea completamente reutilizable, tal como se muestra en este prototipo, utilizando una clase padre que implementa la funcionalidad básica de interacción, y dejando cualquier posibilidad de modificación en las clases hijas. Un ejemplo de esto lo podemos ver en la Figura 24.

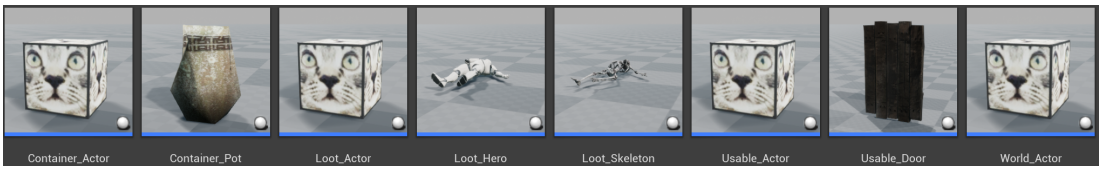


Figura 24. Ejemplos de herencia del sistema de inventario.

3.2.1. Inventory Component

Se trata del componente que gestiona todo el sistema de inventario.

La función principal es *AddToInventory*, que se encarga de comprobar si el objeto con el que se ha interactuado se puede añadir al inventario. Primero se comprueba si la cantidad que se quiere añadir es menor o igual al tamaño máximo de almacenamiento para poder calcular la cantidad de espacios que necesitaremos. Tras esto se crean las nuevas estructuras de datos que vamos a almacenar, con la cantidad máxima que podamos añadir. A continuación se comprueba si se puede guardar el objeto varias veces en el mismo espacio para, en caso positivo comprobar si existen espacios que no estén llenos y, en tal caso añadirlos hasta llenarlo; o en caso contrario comprobar si queda algún slot de inventario libre, para si no crearlo y añadir el objeto.

Por otra parte, encontramos la función *PrepareInventory* que se llamará cuando se inicialice el sistema de inventario. Como se puede ver en la Figura 25, este método redimensiona el inventario según el número de slots que se hayan indicado en la creación del componente. Tras esto realiza un bucle por cada uno de los slots, comprueba si no están vacíos y almacena los valores predeterminados del objeto y la cantidad correcta del mismo.

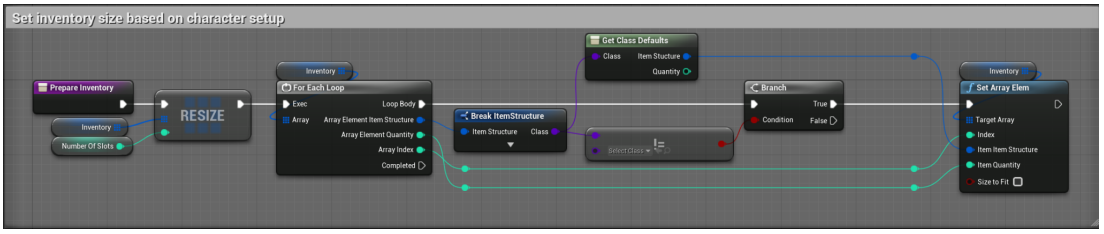


Figura 25. Función Prepare Inventory.

Con la función *HasSpaceInInventory,* podemos ver en la Figura 26, cómo buscamos dentro del array *Inventory* un índice que esté vacío. Esto nos devuelve un entero con el índice del espacio encontrado, o -1 si no encontramos ninguno. La función devuelve un booleano según si encontró un espacio o no.

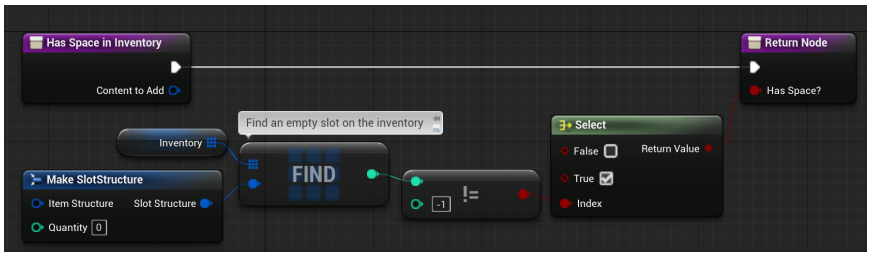


Figura 26. Función Has Space in Inventory.

3.2.1. BaseItem

Este actor es la clase padre de todos los elementos con los que el personaje va a poder interactuar. Contiene una cápsula de colisión y un Static Mesh Component que nos permitirá visualizar el objeto.

La principal funcionalidad de esta clase consiste en la implementación del evento *Interact* que pertenece a la interfaz *bpi_Interactable*. Podemos ver en la Figura 24, como en él, se nos devuelve un actor que es el que realiza la acción, en este caso el PlayerCharacter, y obtenemos el *InventoryComponent* de éste. Tras esto intentamos añadir al inventario el objeto con el que ha interactuado enviándole el

struct con la información del objeto. Si el objeto es recogido, entonces se destruye el actor, en caso contrario no se realiza ninguna acción.

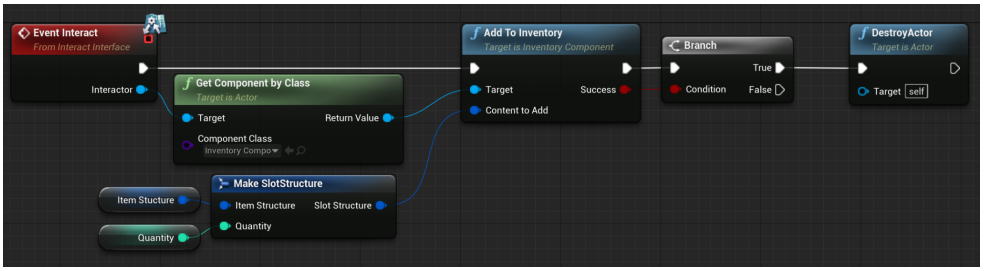


Figura 27. Evento de interacción en la clase padre BaseItem.

Con este actor podremos crear todos los objetos necesarios para el juego e implementar cualquier modificación a la hora de interactuar tan solo sobreescribiendo este evento. Por ejemplo, para crear una fuente de sanación tan solo tendríamos que crear un actor que sea hijo del *BaseItem* y sobreescribir el *EventInteract* de manera que, en lugar de añadir el objeto al inventario, obtuviera el componente *StatComponent* y modificase el valor de puntos de golpe.

3.2.1. BaseContainer

Esta clase nos servirá como padre de todos los contenedores del juego, desde cofres hasta vendedores, cualquier actor que almacene una serie de objetos será hija de *BaseContainer*.

La función principal volverá a ser el *EventInteract* de la interfaz *BPI_Interactable*. En este caso, el actor que ha activado esta función deberá ser el *PlayerCharacter*, para que ningún otro actor pueda interactuar con él; para ello casteamos al *PlayerCharacterManager,* como podemos ver en la Figura 28. y tan solo continuaremos si es él. A continuación reproduciremos una animación (que se implementará en los hijos) y se desactivará la colisión para evitar que se pueda volver a interactuar con él.

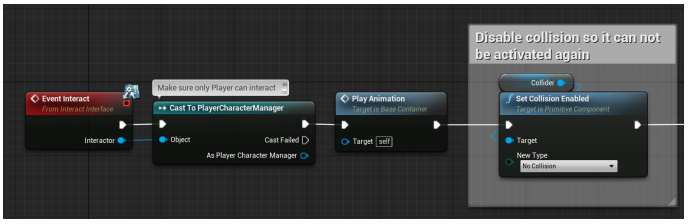


Figura 28. Inicio del evento de interacción en BaseContainer.

El resto de la funcionalidad como vemos en la Figura 29, consistirá en realizar un bucle por cada uno de los objetos e invocar al método *SpawnActor* para crear copias del mismo en el nivel.

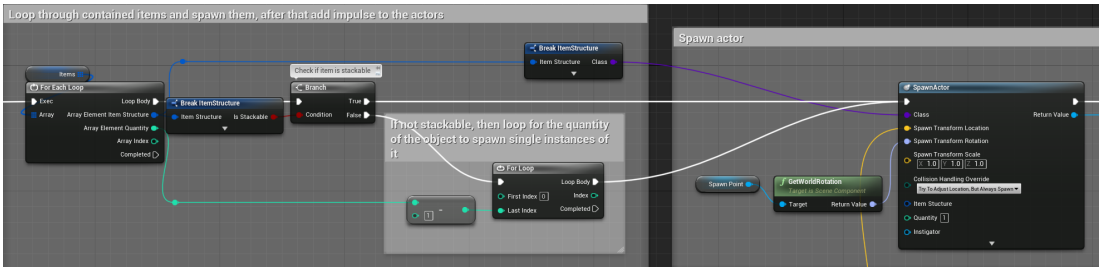


Figura 29. Continuación del evento de interacción en BaseContainer.

3.3. Stats System

Este sistema se encargará de almacenar y gestionar todas las estadísticas de los personajes.

3.3.1. StatsComponent

Este componente albergará la información correspondiente a las estadísticas del personaje y la funcionalidad que permitirá crear y modificar las mismas.

La inicialización se realizará obteniendo la tabla de datos *BaseCharacterStatData* y almacenando dicha información en el componente, tal como se ve en la Figura 30 de inicialización del componente StatsComponent.

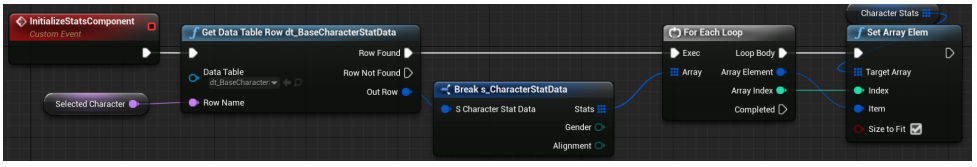


Figura 30. Inicialización del componente StatsComponent.

En la Figura 31, vemos cómo buscamos en el array *CharacterStats* un elemento con el mismo nombre que queremos ajustar. Una vez lo encontramos sobreescribimos el elemento con la nueva información y salimos del bucle.

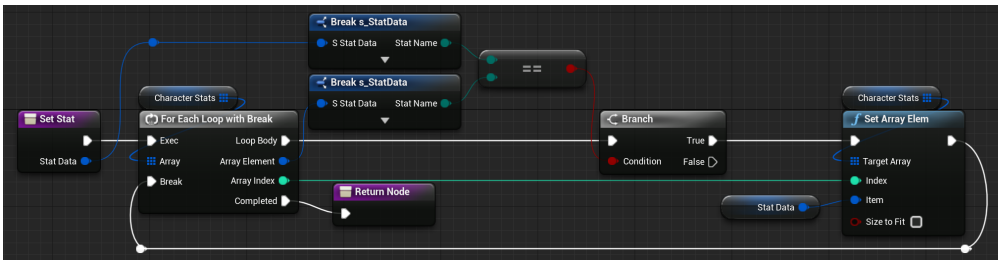


Figura 31. Función SetStat

Finalmente, encontramos el método *UpdateStatModifier*, que añade la funcionalidad de añadir modificadores a las características desde cualquier actor del juego, ya sea una pieza de equipo, una habilidad o un consumible.

En la Figura 32, podemos observar la función anteriormente mencionada. En ella se comienza buscando dentro de las características del personaje la que tenga el mismo nombre que la requerida, y una vez encontrada se almacenan localmente los datos y el índice donde se ha encontrado.

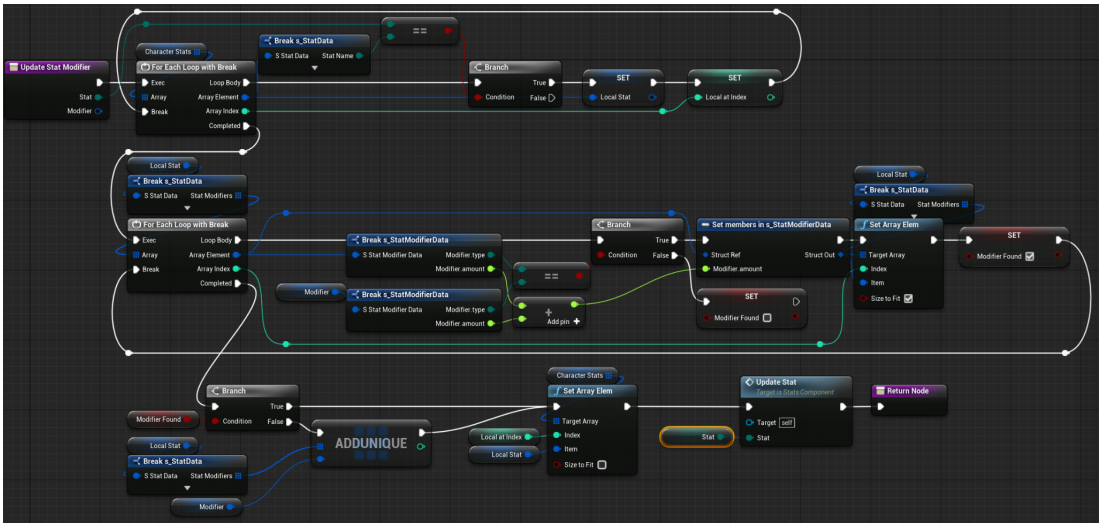


Figura 32. Función AddStatModifier.

A partir de ahí, se comprueba si ya posee un modificador con el mismo nombre, y en caso que así sea, suma el valor del modificador actual y el nuevo para luego almacenarlo dentro de un array local, ajustar un booleano de control a verdadero y terminar el bucle. En caso que no exista el modificador tan solo ajusta el booleano a falso y continúa con el bucle. Finalmente, si no se ha encontrado el modificador, se añade un elemento único al array local, para, tanto si se ha encontrado como si no, actualizar las estadísticas del personaje con el valor del

array local en el índice encontrado al principio, y terminar invocando al método *UpdateStat*.

Esta última función *UpdateStat* actualiza el valor actual de la estadística sumando al valor base los modificadores existentes, tal como podemos ver en la Figura 33.

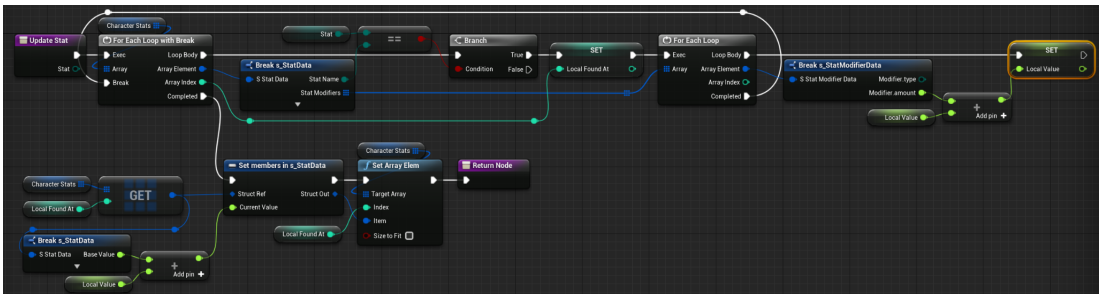


Figura 33. Función UpdateStat

En la Figura 34, podemos ver un ejemplo de implementación del sistema de estadísticas, en el que contamos con un atributo principal (fuerza) con una base de 6 puntos, al añadir varios modificadores quedan los siguientes valores: Equipamiento (2), Habilidad (9) y Objetos (-4). Al terminar la función *UpdateStat* el valor actual de fuerza es de 13 puntos [6+2+9+(-4)].

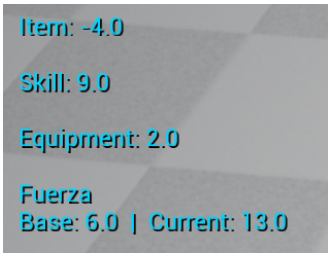


Figura 34. Ejemplo de implementación de StatSystem

3.3.3. Stats DataTable

En esta tabla de datos se almacena toda la información referente a las estadísticas iniciales de los personajes. Toma la estructura de datos del *struct s_CharacterStatData*.

3.4. Niveles

El nivel principal consiste en una isla desarrollada con el programa World Creator [16], desde el que se ha creado un mapa en escala de grises que luego se ha importado en UE4, ver Figura 35. A partir de aquí se han creado los materiales de terreno, se ha pintado el *Landscape,* y se han importado una serie de recursos de diferentes librerías como árboles, rocas y edificios que se han ido colocando en el mapa para crear el nivel.

Figura 35. Heightmap en escala de grises

En la Figura 36, podemos ver el nivel en modo depuración, con las cajas de colisión que impiden al jugador salir del mapa.

Figura 36. Vista del nivel en modo depuración.

Finalmente en las Figuras 37 y 38 se añaden algunas imágenes del nivel en modo de juego, desde donde se pueden observar los recursos incorporados, las texturas del terreno y los elementos de post-procesado utilizados.

Figura 37. Detalle del nivel en modo de juego.

Figura 38. Vista de pájaro del nivel de juego.

Conclusiones y Líneas futuras

El objetivo de este Trabajo de Fin de Máster consistía en investigar, planificar y desarrollar un prototipo de videojuego Action RPG en tercera persona utilizando el motor Unreal Engine 4.

Para ello hemos analizado las fases de desarrollo de un videojuego y creado un Documento de Diseño en el que se ha limitado el alcance del mismo, que era uno de los mayores problemas con los que los estudios se encuentran al crear un videojuego.

Hay que destacar que, aunque el objetivo de este proyecto no se ha conseguido en su totalidad, debido a la complejidad del sistema, así como el tiempo necesario para implementarlo en un prototipo jugable, se han desarrollado los sistemas de inventario y estadísticas, así como un nivel de juego completamente desde cero, creando el terreno en un software externo e importándolo en el motor, para luego crear todos los materiales necesarios para darle vida. Además, se han añadido diferentes volúmenes de control de iluminación, simulación de cielo y post procesado.

. Otra dificultad encontrada en la realización del trabajo ha sido la escasez de *assets* equipables para los personajes, ya que son recursos que varían demasiado de un proyecto a otro, por lo que no son rentables para los desarrolladores.

En lo referido a las líneas futuras, se propone continuar con el desarrollo del prototipo añadiendo el resto de funcionalidades que no se han podido implementar ya sea por su complejidad o por el tiempo que requieren. Así mismo, se plantea la posibilidad de crear tanto *assets* de personaje, como de objetos equipables, que son elementos que no se encuentran dentro del *marketplace* de UE4, por lo que abren un segmento de mercado dentro de este.

Summary and Conclusions

The purpose of this Master's Dissertation was about researching, planning and developing an Action RPG prototype in third person view using the Unreal Engine 4 engine.

To do so we have analyzed the development stages of a videogame and we have created a Game Design Document in which we have constrained its scope, this was one of the biggest issues with those who game development studios find when creating a video game.

We would like to highlight that, even though the aim of this project was not totally achieved, due to the complexity of the system as well as the time needed to implement it in a playable prototype, we have implemented the inventory and stats system, as well as a game level completely from zero, creating the terrain in an external software, and importing it to the engine, in order to create the materials needed to bring the game to life. We also have added different illumination control volumes, sky simulation and post processing. Another difficulty that we have encountered has been the limitation of player's equippable assets, as they are resources that change too much from one project to another, so they are not profitable to developers.

As further research is concerned, we would like to propose continuing with the prototype's development adding the functionalities that we haven't been able to implement for their complexity or the time they request. Likewise, we pose the possibility of creating the player's assets, as the equippable ones, because they are elements that we cannot find in the UE4 marketplace, so they create a market segment inside of it.

Bibliografía

[1] https://es.wikipedia.org/wiki/Sistema_d20
[2] https://es.wikipedia.org/wiki/Open_Game_License
[3] https://audioboom.com/channels/5017064
[4] https://invisioncommunity.co.uk/this-gaming-genre-is-gaining-the-most-popularity-in-2021/
[5] https://steamdb.info/topsellers/
[6] http://www.destinorpg.es/2015/08/historia-de-los-videojuegos-los.html
[7] https://imgvideogame.wordpress.com/2016/04/04/los-rpg-y-sus-subgeneros-mas-conocidos/
[8] https://imgvideogame.wordpress.com/2016/04/04/los-rpg-y-sus-subgeneros-mas-conocidos/
[9] https://www.geekno.com/glosario/arpg
[10] http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/
[11] https://as.com/meristation/2020/12/26/noticias/1608992024_963325.html
[12] http://www.aevi.org.es/web/wp-content/uploads/2020/04/AEVI-ANUARIO-2019.pdf
[13] https://www.devuego.es/bd/estadisticas/evolucion-generos-ano/
[14] https://codexarcana.org/wp-content/uploads/2020/02/reglas-b%C3%A1sicas-DD-5%C2%AA-0.7.pdf
[15] https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/LevelStreaming/WorldBrowser/
[16] https://www.world-creator.com/

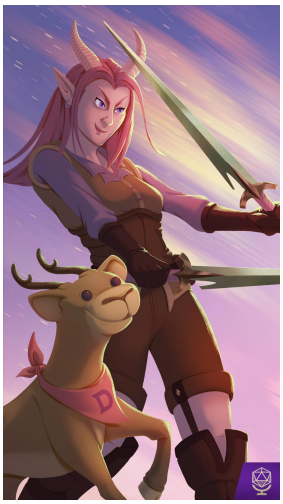
Apéndice A
Lista de controles

Input teclado	Input joystick	Acción
Ratón	Joystick derecho	Movimiento de la cámara
WASD	Joystick izquierdo	Movimiento del jugador
Barra espaciadora		Saltar
Shift izquierdo		Esquivar
Tab		Fijar objetivo
Botón izquierdo del ratón		Ataque básico
Botón derecho del ratón		Bloqueo
Q		Habilidad 1
E		Habilidad 2
R		Habilidad definitiva
1		Usar objeto 1
2		Usar objeto 2
3		Usar objeto 3
4		Usar objeto 4
C		Abrir menú de personaje
Escape		Abrir menú de pausa

Apéndice B
Personajes

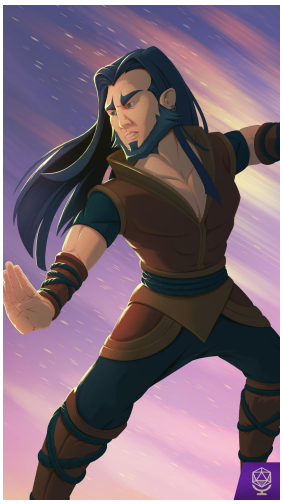
Thomas Von Falken

Gunther



Damaia

Skold



Mirok