TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

"Dragon´s Curse: Un videojuego RPG con MonoGame"

Curso 2018/2019

Alumno/a: Alejandro Urdiales Manzanaro

Director/es:
Antonio Leopoldo Corral Liria

Agradecimientos

Me gustaría agradecer en primer lugar a mi tutor durante este Trabajo Fin de Grado, Antonio Corral, el cual siempre ha estado disponible para resolver las posibles dudas que me iban surgiendo y ha tenido una paciencia infinita con mis malas planificaciones de tiempo.

En segundo lugar, me gustaría agradecer de manera especial a mi hermano, Eduardo, que ha suplido con su increíble capacidad para dibujar (aunque sea con un ratón en el ordenador) mi mínimo nivel artístico en varias ocasiones. Sin su ayuda, seguramente el apartado gráfico del videojuego se habría resentido.

También me gustaría dar las gracias a mi familia por la fe ilimitada que tienen en mis capacidades y todo el apoyo que me brindan día a día con este Trabajo Fin de Grado y la vida en general. Mereciendo un agradecimiento mayor mis padres, Isa y Paco, los cuales llevan esto al límite y son gran al apoyo que he tenido durante estos dos últimos e intensos meses.

Me gustaría dedicar también un agradecimiento especial a todos mis amigos y a mis compañeros del grado, los cuales son al fin y al cabo amigos que se han cruzado en mi vida por escoger esta carrera. Les quiero dar las gracias por estar siempre pendientes de cómo estaba avanzando el videojuego, por hacerme el favor de dedicar parte de su tiempo en probarlo y aguantar mis charlas y quejas interminables sobre este.

Finalmente, me gustaría dar las gracias a la Universidad de Almería y sus profesores del Grado en Ingeniería Informática, los cuales han conseguido que pasará de tener conocimientos nulos sobre el desarrollo de software a que durante este Trabajo Fin de Grado haya sido capaz de desarrollar mi propio videojuego.

Contenido

Tabla de figuras

1. Introducción

El desarrollo de software, de la misma manera que el desarrollo de cualquier producto, es una tarea compleja y que está compuesta a su vez de distintas subtareas de distinta índole complejas por separado. Además, el desarrollo de software cuenta a su vez con unos plazos impuestos para ser completado, lo cual complica la posibilidad de que los desarrolladores realicen todas las subtareas necesarias de un desarrollo software. La solución para este problema es la utilización de librerías, las cuales se tratan de un conjunto de implementaciones de comportamiento ya desarrolladas para facilitar el trabajo del desarrollador. Sin embargo, los desarrolladores también pueden crear sus propias librerías para facilitar los trabajos futuros, aunque esto no es siempre posible por los plazos mencionados anteriormente.

Un claro ejemplo de uso de librerías es el desarrollo de videojuegos, ámbito en el que se conoce a estas librerías como motor del videojuego[1]. A la hora de crear videojuegos se utilizan grandes librerías para manejar ciertos comportamientos tales como el sonido, el renderizado o las físicas. Como hemos mencionado antes, pueden estar ya creadas y ser utilizadas directamente o ser creadas por los desarrolladores para permitir comportamientos específicos del desarrollo actual, lo que suelen hacer únicamente los equipos de desarrollos grandes y con respaldo debido al costo de tiempo y recursos que requiere. Mencionar que, en ciertos casos, conlleva incluso más tiempo, presupuesto y esfuerzo desarrollar el motor a utilizar que el propio desarrollo del videojuego.

[2]Los motores de videojuegos se pueden clasificar de diversas formas que nos permitirán elegir uno u otro en función de las necesidades de nuestro proyecto y del presupuesto con el que se cuenta. Algunas formas de clasificarlos serían las siguientes: tipo de licencia, donde nos encontramos motores comerciales (Unity o CryEngine) y motores OpenSource (MonoGame o Ogre3D); dimensiones que permite, donde tenemos motores que permiten 2D (MonoGame) y motores que permiten también el desarrollo 3D (Unreal Engine); lenguaje en el que está desarrollado, donde encontramos una amplia gama que abarca desde Java hasta C#; plataformas donde se puede ejecutar, donde tenemos tanto motores monoplataforma como multiplataforma.

[2]Además de las mencionadas en el párrafo anterior, existe una división muy significativa para los motores de videojuegos, la forma en la que se utilizan. Por un lado, tenemos lo motores que permiten desarrollar usando simplemente un método de “arrastrar y soltar” (drag and drop), lo cual facilita la inclusión en el mundo del desarrollo de videojuegos al no ser necesario conocer un lenguaje de programación. Aun así, estos motores permiten que el desarrollador programe scripts sencillos para implementar comportamientos requeridos en el videojuego. Algunos motores de videojuegos de este tipo serían Unity y Game Maker. Por el otro lado, contamos con los motores de videojuegos que se podrían considerar como frameworks, los cuales ofrecen facilidades en aspectos básicos como el renderizado o el sonido pero que obligan al desarrollador a implementar otros como las físicas o el sistema de colisiones. Dentro de estos se encuentran motores como MonoGame y Libgdx.

Teniendo en cuenta las limitaciones de tiempo que se tiene en un trabajo de fin de grado y las posibilidades económicas propias, se han buscado motores ya implementados y gratuitos. La cantidad de tiempo y los recursos necesarios para desarrollar un motor propio y un videojuego serían excesivos y por lo tanto se descarta la idea de crear un motor de videojuegos propio durante este trabajo de fin de grado. Por lo tanto, se ha enfocado la búsqueda en motores gratuitos, que permitan programar en C#, desarrollar juegos en 2D y que estén ampliamente documentados. Aplicando estos filtros, el motor de videojuegos más convincente para el desarrollo de este trabajo de fin de grado ha sido MonoGame.

[3]MonoGame se trata de un motor de videojuegos del tipo framework que, como se ha comentado anteriormente, ofrece ciertos comportamientos necesarios para el desarrollo de videojuegos, pero que a su vez permite implementar de manera más específica otros comportamientos como el sistema de colisiones o las físicas. Este framework es gratuito, OpenSource y está enfocado en la realización de videojuegos 2D para casi todas la mayoría de plataformas mediante el uso de las tecnologías Microsoft. Esto último se debe a que MonoGame es una implementación de código abierto del motor de videojuegos de Microsoft, XNA Framework[4][5], el cual actualmente se encuentra sin soporte alguno y desactualizado. Sin embargo, MonoGame es actualizado periódicamente y se encuentra en continuo crecimiento, permitiendo incluso el desarrollo en plataformas actuales como PlayStation 4 o Xbox One. Además, mencionar que MonoGame tiene una extensa y útil documentación, complementada con toda la perteneciente a XNA Framework, y una comunidad de desarrolladores muy activa que participan en los foros de la propia página de MonoGame.

Los videojuegos implican a diversas disciplinas: gráficos, sonido, inteligencia artificial, físicas, etc. Gracias a MonoGame, en este trabajo de fin de grado las partes de gráficos y sonido las gestiona el propio framework y solo se necesitará hacer uso de las implementaciones de este. Sin embargo, será necesario llevar a cabo diversos aspectos como los algoritmos del sistema de colisiones, la inteligencia artificial, el uso de diversas estructuras de datos, la aplicación de patrones de diseño[6] o el tratamiento de archivos, por lo que se pondrán en práctica bastantes conocimientos de los adquiridos en el grado.

La elección entre los distintos géneros dentro de los videojuegos, los cuales se tratarán a posteriori, sería otro aspecto de suma importancia. Esta importancia viene de que no es lo mismo realizar el desarrollo de un videojuego de plataformas, en el cual el jugador tiene muy poca o nula interacción con otros personajes dentro del juego y se busca más la dificultad a la hora de superar los puzles o retos que nos ofrecen las plataformas del mapa; de un videojuego del tipo shooter, donde lo más importante suele ser el multijugador y los sistemas de físicas y colisiones; o un videojuego RPG, en el que el jugador interacciona en gran medida con todo el entorno del mapa y los personaje y se permite una gran personalización del aspecto físico del personaje y sus habilidades. Tras investigar más a fondo sobre todos estos géneros[7], y como se puede apreciar en el título del proyecto, el género elegido ha sido el RPG, ya que los videojuegos RPG siempre han llamado mucho la atención del autor y probablemente sea el que mejor se adapte a las ideas ya concebidas para el videojuego a realizar.

1.1. Motivación profesional y personal

Este proyecto de Trabajo de Fin de Grado supone un extra de motivación tanto personal como profesional para el autor, que se complementa con la posibilidad que ofrece de aplicar los conocimientos adquiridos durante el grado.

El gusto por los videojuegos es algo que siempre ha estado muy presente para el autor, desde que por primera vez disfrutara de ellos con el clásico Pokemon Amarillo de Game Boy hasta la actualidad en la que disfruta con mayor madurez de juegos como The Last of Us o Shadow of the Colossus en consolas de última generación. Durante este periodo, dicho gusto por los videojuegos fue evolucionando en una pasión por estos, que a su vez conllevó un interés más concreto en el desarrollo de estos y la posibilidad de “crear” sus propios juegos. Para poder llegar a cumplir este “deseo”, el autor se decantó fácilmente por escoger el Grado en Ingeniería Informática, el cual le permitiría además adquirir conocimientos sobre informática en general que también es una de sus pasiones.

Por otro lado, está el aspecto de motivación profesional para el autor, el cual tiene la creación de su propio estudio de desarrollo de videojuegos como mayor objetivo en su futura vida laboral. Además, según los estudios mostrados en el Libro Blanco del Desarrollo Español de Videojuegos 2017[8], el futuro de la industria del videojuego es muy prometedor y cuenta con un crecimiento anual estimado del 8.2% como se aprecia en la *Figura 1*. Este dato junto a los datos actuales sobre el dinero que generan (116.000 millones de dólares en 2017) invita a pensar que los videojuegos, en un futuro no muy lejano acabarán desbancando en el mundo audiovisual al resto de competidores (cine, televisión, música, etc), las cuales dominaron y dominan el mercado de lo audiovisual. También es importante destacar, que esta mayor importancia conllevará el aumento de trabajos relacionados con la industria y sus sueldos correspondientes, siendo una apuesta de futuro y la cual se está llenando de jóvenes profesionales con gran talento.

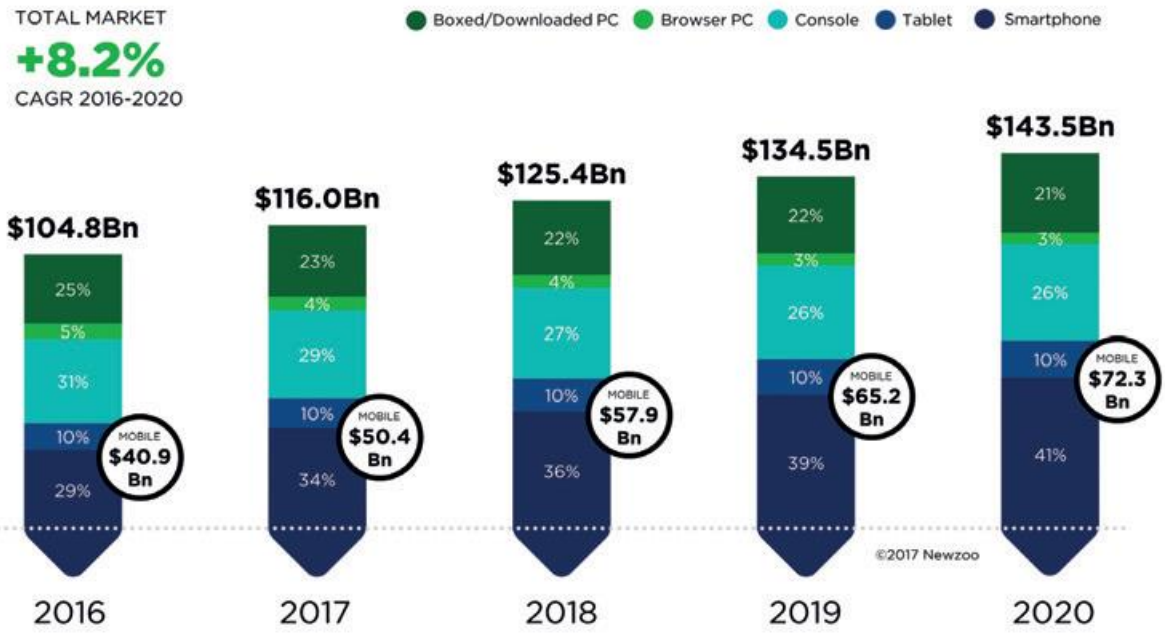


Figura 1. Evolución estimada del mercado mundial del videojuego[8]

Por lo tanto, este Trabajo Fin de Grado supondrá el primer reto del autor en su carrera por cumplir sus aspiraciones. Además, al no contar con experiencia alguna en el desarrollo de videojuegos, se tratará de un excelente ejercicio autodidacta en el que se pondrán a prueba las capacidades del mismo.

1.2. Objetivos

El objetivo principal de este Trabajo Fin de Grado sería desarrollar un videojuego en 2D con MonoGame, para lo que se precisan los siguientes objetivos:

- Crear un sistema de gestión de estados para evitar dependencias entre ellos
- Crear y diseñar menús para el juego
- Crear clases abstractas como plantillas para crear entidades del juego
- Crear y diseñar los personajes del jugador como entidades animadas
- Crear y diseñar los NPCs (Non-Player Character) como entidades animadas
- Crear y diseñar elementos del mapa como entidades estáticas o animadas

- Crear el estado de nivel en el que el jugador pueda tomar el control de su personaje e interactuar con el resto de elementos del nivel
- Crear una cámara que siga al jugador por la pantalla
- Implementar un sistema de físicas y colisiones para moverse por el escenario
- Permitir pausar el juego
- Añadir el aspecto sonoro al videojuego
- Permitir guardar y cargar el progreso realizado por el jugador
- Crear una cámara que siga al jugador por la pantalla

1.3. Planificación temporal

Como en todo proyecto software, es importante contar con una planificación del trabajo a realizar teniendo en cuenta las horas de trabajo dedicadas por día y los recursos. Es por esto, que como parte del trabajo inicial de este Trabajo Fin de Grado se realizó una planificación temporal estimada, la cual se muestra en la *Figura 2*.

Figura 2. Planificación temporal estimada para el proyecto Dragon´s Curse

En esta planificación se han tenido en cuenta los distintos procesos y etapas a llevar a cabo durante un proyecto software, específicamente en el desarrollo de un videojuego, teniendo en mente una jornada de cuatro horas por día, la cual se puede ver aumentada o reducida en función de las necesidades de dicho proyecto.

En el apartado "3.6. Planificación temporal del proyecto" se realizará una nueva planificación temporal, pero esta vez se tratará de la planificación que ha sido llevada a cabo. Una vez que se exponga esta nueva planificación sería interesante llevar a cabo una comparación con la planificación estimada, determinando porque han podido surgir cambios en las estimaciones para ciertos procesos del proyecto.

1.4. Estructura del documento

A lo largo del capítulo actual de esta memoria se ha llevado a cabo una introducción al proyecto que rodea al Trabajo Fin de Grado del autor, explicando además las motivaciones de este respecto a este proyecto y los objetivos a cumplir para su correcta finalización

A continuación, en el capítulo "2. Videojuegos" se expondrán diferentes conceptos necesarios y generales sobre el mundo de los videojuegos y su desarrollo. A todo esto, se le añadirá un breve repaso por la historia de los videojuegos y un estudio del mercado y la industria tanto a nivel internacional como nacional.

En el capítulo "3. Dragon´s Curse" se explicará en mayor profundidad el motor de videojuegos que se va a utilizar para la realización de este Trabajo Fin de Grado, MonoGame. Tras esto, se llevará a cabo un repaso por los distintos procesos del desarrollo del videojuego Dragon´s Curse y se expondrán las herramientas utilizadas así como la planificación temporal llevada a cabo finalmente.

En el capítulo "4. Resultados" se mostrará el resultado final del proyecto, con capturas de las pantallas del videojuego y explicaciones sobre las distintas funcionalidades implementadas.

En el capítulo "5. Conclusiones y trabajo futuro" se llevará a cabo un trabajo introspectivo en el cual se expondrán las conclusiones derivadas del desarrollo de este Trabajo Fin de Grado y se comentarán posibles trabajos futuros y ampliaciones sobre el resultado final.

Por último, en el capítulo "6. Bibliografía" se encontrarán todas las referencias bibliográficas de las que se ha sacado información para la realización de este Trabajo Fin de Grado.

2. Videojuegos

En este apartado se hablará en mayor profundidad de qué es un videojuego, comenzando por una definición básica de los videojuegos y comentando brevemente su historia. Tras esto, se introducirán unos conceptos básicos tales como los controladores, las plataformas, los géneros y los motores de videojuegos. Otro aspecto a tratar será el desarrollo de videojuegos, en el cual se comentará el proceso que conlleva y los roles que participan en este. Finalmente, se estudiará la actualidad del mercado de esta industria tanto a nivel global como a nivel español, añadiendo finalmente el impacto que los videojuegos suponen en la sociedad.

2.1. ¿Qué es un videojuego?

Un videojuego se trata de un juego electrónico el cual se visualiza mediante el uso de una pantalla o dispositivo de visualización[9]. En otras palabras, un videojuego se trata de un software orientado al entretenimiento que, haciendo uso de unos controladores y una plataforma, permite al usuario o usuarios interactuar con este y que le responda mediante dispositivos de salida tales como una pantalla, dispositivos de sonido, etc.[10]. Por lo tanto, se puede apreciar que la mayor característica de los videojuegos dentro del mundo del entretenimiento, y donde se diferencia de otros medios como el cine o la música, es la interacción con el usuario, la cual permitirá que la experiencia sea prácticamente única para cada individuo.

2.2. Conceptos básicos en el mundo de los videojuegos

2.2.1. Controlador

Un controlador de videojuegos se trata de un dispositivo de entrada que permite al jugador interactuar con el videojuego y controlar el funcionamiento de este[11].

Si se habla de controladores dentro del mundo de los videojuegos, se tiene que existen multitud de tipos de dispositivos muy diferentes entre sí que tienen en común una funcionalidad común, la interacción del usuario o jugador con el videojuego. Algunos de estos tipos de controladores serían los siguientes:

2.2.1.1. Mando o gamepad

Un mando en el mundo de los videojuegos se trata de un dispositivo de control que utiliza el jugador para interactuar con el juego. Este cuenta básicamente con controladores del movimiento y botones para enviar distintas señales al videojuego[12].

Figura 3. Gamepad de la videoconsola SNES/Super Famicom[13]

Los controladores de movimiento de un mando o gamepad consistían inicialmente en crucetas D-Pad, las cuales permiten transmitir al videojuego la dirección indicada por el jugador mediante un valor de 0 o 1 como es en el caso del mando de la SNES de la *Figura 3*. Con el paso del tiempo aparecieron los joysticks o palancas de movimiento, los cuales envían el ángulo de dirección y la inclinación deseados por el jugador al videojuego para permitir un movimiento más específico que el permitido por las crucetas. En la actualidad, el estándar en cuanto a controladores de movimiento es la inclusión de una cruceta D-Pad y uno o dos joysticks en el gamepad, permitiendo el uso de ambos a elección del usuario. Esto es visible en mandos de última generación como el de PlayStation 4 representado en la *Figura 4*.

Figura 4. Gamepad de la videoconsola PS4 (DualShock 4)[14]

Los botones del mando o gamepad tienen la función principal de enviar una señal concreta (específica para cada botón) al videojuego para que se ejecute una acción asociada a dicho botón. En un principio, estos botones se limitaban únicamente a situarse en el frontal del mando, pero en la actualidad casi como norma general todos los mandos suelen contar además con unos botones traseros denominados triggers o gatillos (los cuales se pueden apreciar en el gamepad de la Figura 4). Estos triggers, además de contar con la funcionalidad básica de los botones antes explicada, permiten al usuario y al gamepad añadir al envío de la señal la intensidad de pulsación que se le aplica a dicho trigger, añadiendo un grado más de detalle a dicha señal.

Destacar por último, que las consolas portátiles, las cuales se tratarán en mayor profundidad en el apartado 2.3.2., no necesitan en su mayoría de un gamepad o mando externo, ya que dentro del hardware de estas se incluyen los controladores de movimiento y los botones directamente sin necesidad de un dispositivo externo (véase en la Figura 5).

Figura 5. Consola New Nintendo 3DS XL con controladores de movimiento y botones incluidos[15]

2.2.1.2. Teclado y ratón

El teclado y el ratón se tratan de dos controladores que se complementan el uno al otro y constituyen el controlador de videojuegos de ordenador principal del mercado.

El teclado se trata de un dispositivo que cuenta con un conjunto de letras, las cuales al ser pulsadas envían señales cifradas al ordenador (o al dispositivo al que esté conectado) y este las interpreta para producir una respuesta[16]. En los videojuegos, el teclado suele ser utilizado para trasladar al juego las interacciones del usuario mediante funcionalidades internas de este, siendo la más conocida la de pulsar la "cruceta" WASD[17] o las flechas de dirección para mover al personaje del jugador.

El ratón es un periférico o controlador utilizado principalmente en ordenadores cuya funcionalidad es la de mover el cursor que se muestra por pantalla y permitir acciones como la selección[18]. Su uso en los videojuegos suele estar asociado al movimiento de la cámara interna del videojuego y a la realización de acciones tan diversas como la selección de un elemento del juego o la de disparar en los juegos del genero de disparos o shooter.

2.2.1.3. Pantalla táctil

La pantalla táctil se trata de un controlador de videojuegos relativamente reciente, el cual se trata de una pantalla que permite la interacción del usuario o jugador mediante toques[19]. Por lo tanto, esta permite eliminar la necesidad de botones o controladores de movimiento físicos para interactuar con el videojuego.

Este controlador suele presentarse mayoritariamente en dispositivos móviles, en los cuales se ha convertido en un estándar. Sin embargo, es también interesante mencionar que ciertas plataformas (en su mayoría pertenecientes a Nintendo[20]) también incluyen pantallas táctiles en los mandos o incluso en la propia plataforma, como el caso de la New Nintendo 3DS XL que se puede apreciar en la Figura 5.

2.2.1.4. Sensores de movimiento

En este tipo de controladores nos encontramos con dispositivos que, de distintas maneras y mediante distintos mecanismos, buscan capturar el movimiento del jugador para transmitirlo hacia el videojuego y que este responda ante esto.

Figura 6.Wiimote de la videoconsola Wii[21]

Dentro del sector de los sensores de movimiento nos podemos encontrar principalmente con los mandos de movimiento, lo cuales tratan de capturar el movimiento mediante acelerómetros, sensores ópticos, magnetómetros, etc. (como puedan ser Wiimote[21] representado en la *Figura* 6 o Playstation Move[22]); y las cámaras de reconocimiento corporal, las cuales hacen uso de cámaras RGB, sensores de profundidad, etc. para reconocer el cuerpo del usuario y la posición de este para que pueda interactuar como si se encontrara dentro del videojuego (algunos ejemplos serían Kinect[23] o Playstation Eye[24]). Mencionar además, que ambos tipos de dispositivos se complementan en algunos casos como pueda ser el del uso de Playstation Eye para utilizar Playstation Move. Para concluir con este tipo de controladores, mencionar que en esta categoría también se podrían introducir los cascos de realidad virtual, ya que aunque su función principal sea el aspecto visual también cuentan con capacidades para el reconocimiento del movimiento de la cabeza[25].

2.2.1.5. Otros

Finalmente, en torno a los controladores de videojuegos, sería importante mencionar algunos que, aunque sigan en el mercado, no cuentan con la importancia de los explicados durante este apartado. Estos controladores podrían ser las arcades, los joystick y en general todos los controladores que permiten la simulación de un elemento real para crear una mayor inmersión en el juego (como pueda ser un volante, una pistola de luz o una guitarra).

2.2.2. Plataforma

Una plataforma de videojuegos se trata de un conjunto de elementos hardware y software que permiten al usuario la posibilidad de ejecutar el software conocido como videojuego. Dentro de las plataformas de videojuegos nos encontramos con una gran variedad tanto de marcas como de tipos de plataformas, pero la clasificación más general que se podría hacer es la que las divide en PCs, consolas, arcades y dispositivos móviles[26]. Cabe mencionar, antes de explicar en mayor profundidad estos tipos de plataformas, que únicamente las consolas y las arcades fueron diseñadas con el propósito explícito de ser usadas para jugar a videojuegos.

2.2.2.1. PC (Personal Computer)

Los PCs u ordenadores (ambos términos se usan indistintamente) se tratan de plataformas informáticas cuya funcionalidad nos permite desde realizar tareas de ofimática hasta jugar a videojuegos. La

composición que conocemos como PC suele ser la formada por una ordenador de sobremesa, una pantalla, un teclado y un ratón o por un único dispositivo que integra todos los anteriormente mencionados conocido como ordenador portátil.

Estos dispositivos permiten por tanto ser denominados como plataformas de videojuegos aunque no fueran concebidos con la idea de ser definidos como tal. Sin embargo, en la actualidad muchos PCs son fabricados con la idea de que principalmente cumplan la función de ejecutar un videojuego con un alto rendimiento, para lo cual se eligen componentes hardware orientados a esto. A estos PCs se les suele conocer como PCs gaming.

2.2.2.2. Consolas

Las consolas se tratan de la plataforma de videojuegos por excelencia, las cuales se tratan de dispositivos capaces de ejecutar videojuegos desde un cartucho, un disco compacto, una tarjeta de memoria o desde el propio almacenamiento de la consolas (para lo cual se necesita que el juego haya sido descargado en este)[27]. En la actualidad, las consolas han ampliado sus posibilidades no solamente ofreciendo la posibilidad de jugar a videojuegos, sino también permitiendo disfrutar de otros contenidos multimedia, interaccionar con otras personas de forma online o navegar por Internet[27]. Estas plataformas de videojuegos podrían clasificarse tanto por generaciones[28], de las cuales existen ocho distintas, como por su movilidad, donde nos encontramos con consolas de sobremesa, portátiles o híbridas. A continuación se va a proceder a realizar esta última clasificación mencionando además las consolas de última generación en cada tipo.

Las consolas de sobremesa están concebidas para el uso doméstico y ofrecer una mayor potencia que el resto de tipos de consolas. Dentro de esta categoría nos categoría nos encontraríamos con consolas de la octava generación como puedan ser PlayStation 4, Xbox One o Wii U, mostradas en la *Figura 7*.

Figura 7. Consolas de sobremesa de la octava generación[59]

Las consolas portátiles tienen como principal ventaja la posibilidad de permitir jugar a videojuegos desde cualquier lugar sin necesidad de disponer de una toma de corriente donde tener enchufada constantemente la consolas. Sin embargo, esta ventaja que ofrecen las consolas portátiles se ve contrapuesta a la menor potencia que ofrecen en comparación con las consolas de sobremesa. Dentro de la octava generación nos encontraríamos con consolas como Nintendo 3DS o Playstation Vita (mostradas en la *Figura 8*), por lo que se puede apreciar que prácticamente el único fabricante que sigue apostando por este tipo de consolas es Nintendo.

Figura 8. Consolas portátiles PlayStation Vita y Nintendo 3DS[60]

Las consolas híbridas son la mayor novedad del mercado gracias a Nintendo y su Nintendo Switch, anteriormente conocida como NX Project. Este tipo de consolas cogen las ventajas de los dos tipos anteriores, permitiendo al usuario poder cambiar entre los modos sobremesa (obteniendo mayor potencia pero perdiendo la movilidad) y portátil (obteniendo la posibilidad de jugar donde sea pero perdiendo potencia). Cabe destacar que hasta la fecha la única consola híbrida oficial (ya que existen modificaciones de usuarios sobre consolas sobremesa para hacerlas híbridas) es Nintendo Switch (*Figura 9*), pero actualmente se especula con que PlayStation 5, la consola de Sony para la novena generación, también se tratará de una consola híbrida[29].

Figura 9. Consola híbrida Nintendo Switch[61]

2.2.2.3. Arcades

Las máquinas arcades[30] son probablemente de los primeros dispositivos para jugar a videojuegos que estaban orientados para todos tipos de públicos. Se tratan de máquinas recreativas al igual que las tragaperras o los pinballs, pero destinadas a disfrutar de videojuegos para lo que hacen uso de una pantalla y diversos controladores (botones, joysticks, pistolas, etc.) como se aprecia en el ejemplo de la *Figura 10*. En la actualidad, las máquinas arcades cuentan con un público bastante escaso a nivel internacional debido a la expansión de las consolas, mientras que si es destacable su popularidad en países asiáticos como Japón en los que han conseguido subsistir hasta hoy en día.

Figura 10. Máquina arcade[62]

2.2.2.4. Dispositivos móviles

Los dispositivos móviles se tratan de otros dispositivos que, al igual que el PC, no estaban orientados a su uso como plataforma de videojuegos, ya que su función principal es la de la comunicación entre dos personas. Sin embargo, con el paso del tiempo y el aumento de la potencia en estos dispositivos se ha conseguido adaptarlos como plataformas para el uso de videojuegos, llegando al punto de que en la actualidad cualquier Smartphone es más potente como plataforma de videojuegos que consolas de sobremesa de quinta o sexta generación.

2.2.3. Géneros de videojuegos

Los géneros dentro del mundo de los videojuegos hacen referencia a una cosa distinta que en el mundo del cine o la literatura, donde normalmente se distinguen por la temática ambiental de la película o libro. En los videojuegos, el aspecto de la temática ambiental también podría ser considera como una clasificación, pero no tan importante como para ser considerada como una clasificación por género. Por tanto, al referirnos a géneros de videojuegos se habla del tipo de gameplay que ofrece el videojuego[31]. El gameplay de un videojuego o jugabilidad hace referencia a la forma en la que se juega el videojuego, es decir, se trataría de un conjunto de aspectos como puedan ser las reglas de juego, los objetivos, la trama, etc.[32]

Un videojuego por tanto, se podría clasificar dentro de uno de los distintos géneros existentes, pero además hay que tener en cuenta que esta clasificación no es de carácter único por lo que dicho videojuego puede hallarse al mismo tiempo englobado en distintos géneros cogiendo cosas de ambos. Es importante mencionar que durante el la historia de los videojuegos han ido surgiendo diferentes subgéneros dentro de los propios géneros de videojuegos, siendo algunos de ellos considerados incluso como géneros independientes. Sin embargo, en este caso se va a proponer una clasificación correspondiente a los géneros originales dentro del mundo de los videojuegos[33].

2.2.3.1. Videojuegos de acción

El género de acción[33] dentro del mundo de los videojuegos incluye en sí a juegos que proponen al usuario la superación de niveles mediante su habilidad o destreza. Para esto se fuerza al usuario a decidir la estrategia a seguir de una forma rápida y desarrollar la velocidad de reacción y la coordinación ojos-manos (en el caso de que el controlador sea manipulado con las manos). Durante los niveles a superar el jugador se encontrará con una serie de peligros en forma de enemigos u obstáculos, teniendo que derrotar a un enemigo final más fuerte al final de cada nivel. Se podría decir que en este género es de

gran importancia la capacidad del jugador para reconocer patrones de acción de los enemigos, lo cual facilitará la elaboración de su estrategia.

Dentro de los videojuegos de acción se podrían distinguir cuatro subgéneros principales[34], ampliables a cinco en el caso de incluir los shooters o videojuegos de disparos:

- Videojuegos de lucha: Se tratan de juegos en los que el jugador con su personaje se enfrenta en combates de uno contra uno contra otro personaje controlado por la máquina o por otro jugador. Uno de los aspectos más interesantes de este subgénero es la variedad de movimientos con la que cuenta cada personaje, ofreciendo al jugador la posibilidad de decidir qué estrategia usar con cada personajes o viceversa[35]. Algunos ejemplos de juegos de lucha serían *Street Fighter* o *Mortal Kombat*.
- Videojuegos de Beat´em Up: Se tratan de juegos en los cuales el jugador va avanzando con un personaje por un nivel en una única dirección mientras lucha con grandes cantidades de enemigos que van apareciendo durante este recorrido, teniendo que enfrentarse al final de este a un enemigo mucho más poderoso. Normalmente suelen ser juegos muy abiertos al concepto multijugador, lo que ayuda a la consecución del objetivo haciendo uso de estrategias de equipo[36]. Algunos ejemplos del subgénero serían *Double Dragon* o *Streets of Rage*.
- Videojuegos de Hack ´n´ Slash: Estos juegos derivan de los anteriormente mencionados Beat´em Up, con la peculiaridad de que el personaje suele portar un arma blanca como por ejemplo una espada, un hacha o una lanza. Además, suelen incluir un aspecto propio de los juegos de rol o RPG como puede ser la mejora de las habilidades del personaje[37]. Algunos ejemplos de este subgénero serían *Diablo* o *God of War*.
- Videojuegos de plataformas: En este subgénero se encuentran videojuegos en los cuales el jugador debe avanzar por un nivel haciendo uso de plataformas a las que debe llegar haciendo uso de saltos y otras habilidades acrobáticas. Para hacer más dinámico este gameplay, se introducen normalmente potenciadores que cambian las habilidades del jugador por una cantidad determinada de tiempo[38]. Ejemplos de juegos de plataformas serían *Super Mario Bros* o *Sonic the Hedgehog*.

2.2.3.2. Videojuegos de disparos

Los videojuegos de disparos o shooters son un subgénero de los videojuegos de acción, pero que por la magnitud que han obtenido a lo largo del tiempo se pueden considerar como un género propio. La característica principal de este tipo de videojuegos es que el personaje manejado por el jugador siempre cuenta con un arma de fuego sin la cual no puede interaccionar con el entorno, lo que la hace indispensable para el gameplay[33].

Dentro del género shooter se pueden apreciar cuatro subgéneros principales[39]:

- Videojuegos de First Person Shooter (FPS): La principal característica de este tipos de juegos es que la acción siempre se desarrolla utilizando una cámara en primera persona, la cual se sitúa en el personaje controlado por el jugador y nos permite ver únicamente el arma y/o el brazo de este[40]. Algunos ejemplos de videojuegos pertenecientes a este subgénero serían *Call of Duty* o *Doom*.
- Videojuegos de Third Person Shooter (TPS): Al igual que los First Person Shooter, los videojuegos de este subgénero cuentan con que su característica principal es la posición de la cámara respecto del personaje del jugador. En este caso, la cámara se ubica detrás del personaje,

mostrándolo al jugador, a la altura del hombro para facilitar el apuntado[41]. Ejemplos de videojuegos TFS serían *Gears of War* o *Max Payne*.

- Videojuegos de Shoot´em Up: El gameplay de los videojuegos de este subgénero se basa en avanzar por un mapa o nivel controlando a un personaje, el cual en muchas ocasiones suele ser un vehículo, con el cual eliminar a multitud de enemigos al mismo tiempo mientras se esquivan los disparos de estos[42]. En este subgénero se encuentran videojuegos como *Space Invaders* o *Contra*.
- Videojuegos de disparos sobre raíles: Este subgénero se caracteriza por limitar las tareas del jugador al apuntado y disparo del arma del personaje, estando la movilidad por el nivel ya fijada sobre unos raíles virtuales[43]. Dentro de este subgénero nos encontramos con videojuegos como *The House of the Dead* o *Crisis Zone*.

2.2.3.3. Videojuegos de estrategia

Los videojuegos de estrategia se basan principalmente en el hecho de que el jugador debe controlar un grupo de personajes amplio planificando la estrategia a llevar por estos para cumplir el objetivo principal, teniendo en cuenta las consecuencias de las decisiones que tome[33].

Este género se podría dividir principalmente en dos subgéneros principales, los de estrategia en tiempo real y los de estrategia por turnos[44]:

- Videojuegos de estrategia en tiempo real: En este subgénero de los videojuegos de estrategia la característica principal del gameplay es la continuidad en los sucesos y acciones que ocurren en el juego, omitiendo cualquier pausa entre las acciones determinadas por el jugador y lo que sucede en consecuencia de estas. De esta manera, se recompensa la agilidad mental y la capacidad de reacción del jugador además de su capacidad estratégica[45]. Algunos juegos de este subgénero podrían ser *Age of Empires* o *Imperivm*.
- Videojuegos de estrategia por turnos: En contraposición a los videojuegos de estrategia en tiempo real se encuentran los videojuegos de estrategia por turnos, en los cuales el tiempo dentro del juego se pausa mientras el jugador decide la estrategia a llevar a cabo. Durante su turno, el jugador cuenta con un número de movimientos máximos a realizar, tras los cuales el turno pasa al enemigo y así consecutivamente. En estos videojuegos se prima mucho más la capacidad estratégica y analítica del jugador, antes que la agilidad mental y la capacidad de reacción[46]. En este subgénero se encuentran videojuegos tales como *XCOM: Enemy Unknown* o *Advance Wars*.

2.2.3.4. Videojuegos de aventura

Los videojuegos de aventuras basan su gameplay en la narración de una historia principal para la cual el jugador necesita realizar tareas investigación, exploración y resolución de enigmas. Durante estas fases, el gameplay puede incluir la posibilidad de combates entre el jugador y enemigos controlados por la máquina para hacer más variado el transcurso del videojuego[33].

El género de aventuras cuenta con multitud de subgéneros dentro de él, aunque la mayoría se tratan de combinaciones de este con otro género como puedan ser el género de aventuras o el RPG. Sin embargo, existen dos subgéneros de los juegos de aventuras que se podrían definir dentro de este género sin combinarlos con otro más[47]:

- Videojuegos de aventura conversacional: En este subgénero la trama se desarrolla de manera textual, ocasionalmente con alguna imagen, y haciendo que la interacción del jugador con esta trama sea mediante comandos de texto usando lenguaje natural[48]. Algunos ejemplos de subgénero serían *El Hobbit* o *Mistery House*.
- Videojuegos de aventura gráfica: Los videojuegos de este subgénero son la evolución de las aventuras conversacionales, donde se introduce un componente gráfico mucho mayor. En este tipo de videojuegos, el jugador debe interactuar con otros personajes no jugadores y con objetos del nivel para resolver los puzles que constituyen la trama del juego[49]. Dentro de este subgénero se encuentra videojuegos como *Life Is Strange* o *Ace Attorney*.

2.2.3.5. Videojuegos de rol

En los juegos de rol o RPG (Role Playing Game) el jugador tiene como principal tarea que escoger un rol y desarrollarlo durante la trama de dicho videojuego, la cual es un aspecto muy importante en el gameplay de este género de videojuegos. Normalmente estos juegos suelen basar su gameplay en el desarrollo de una historia lineal o con alternativas bastante profunda, en la cual se insertan además multitud de subhistorias más pequeñas que complementan a la trama principal. Sin embargo, cabe destacar que muchos de estos juegos también suelen incluir un apartado multijugador en el cual los jugadores pueden "diseñar" sus propias historias y objetivos mientras juegan[33].

Dentro de este subgénero se incluyen multitud de subgéneros, que al igual que en el género de aventuras, suelen ser combinaciones con otros tipos de videojuegos[50]. Las combinaciones más recurrentes suelen ser rol-acción (con ejemplos como *Kingdom Hearts* o *The Elder Scrolls*) y rol-aventura (con ejemplos como *The Legend of Zelda* o *Assassins Creed: Odyssey*).

Sin embargo, cabe destacar un subgénero bastante importante dentro del género de rol, los MMORPG. Los MMORPG o Massive Multiplayer Online Role Playing Game se tratan de videojuegos de rol cuya principal característica es el carácter exclusivo online de su gameplay, el cual sitúa al jugador en un mundo virtual con cientos o miles de jugadores al mismo tiempo e incita a la interacción, colaboración y competición entre ellos para cumplir sus objetivos[51]. Algunos ejemplos de este subgénero tan importante de los RPG serían *World of Warcraft* o *Star Wars: The Old Republic*.

2.2.3.6. Videojuegos simulación

El género de simulación dentro de los videojuegos hace referencia a todos aquellos juegos, que como su nombre bien indica, tratan de simular situaciones reales de la manera más fiel posible para mejorar las habilidades y destrezas de un profesional o para probar las posibilidades técnicas, sociales o físicas del jugador[33]. Todo esto hace que este tipo de videojuegos no cuente normalmente con un objetivo final, sino que más bien tratan simplemente de recrear una experiencia lo más real posible[52].

Dentro de este género, podemos encontrar que sus videojuegos se suelen clasificar por su ámbito temático: simuladores de vida (*Los Sims*), simuladores de gestión deportiva (*Football Managerk*), simuladores de vehículos (*Euro Truck Simulator*), etc.

2.2.3.7. Videojuegos educativos, curativos, preventivos y de agilidad mental

Los videojuegos incluidos en este género tienen como finalidad principal la mejora del jugador en aspectos como la agilidad cerebral y aspectos educativos, ya sean a nivel de pasatiempo o como terapia

cognitiva/social[33]. Algunos de estos videojuegos podrían ser *Brain Training Infernal del Dr. Kawashima* (el cual pretende mejorar los conocimientos matemáticos y la agilidad mental además de ser una gran ayuda para los niños con TDAH), *Proyecto Kokori* (el cual propone el aprendizaje de la biología mediante misiones cortas con distintos niveles de dificultad) o *Reigns: Her Majesty* (el cual pone al jugador en el papel de una reina que debe hacer frente a los desafíos que suponen para ella la sociedad patriarcal y machista en la que vive).

2.2.3.8. Videojuegos musicales

Los videojuegos musicales son aquellos que ponen al jugador a interactuar con los procesos musicales, ya sean de creación, combinación y armonía, afinación o destreza musical[33]. Por lo tanto, se podría decir que estos videojuegos tratan de recrear de manera más o menos realista el ámbito que rodea a la música, por lo que en algunas clasificaciones podrían ser considerados como un subgénero de los videojuegos de simulación. Dentro de este género nos encontramos con juegos como *Guitar Hero*, *Just Dance* o *Singstar*.

2.2.3.9. Videojuegos deportivos

Los videojuegos deportivos son un tipo de videojuegos que tratan de recrear un deporte concreto, para cuyo gameplay el jugador debe conocer las reglas de dicho deporte concreto y demostrar capacidades como la agilidad ojos-controlador y la capacidad de reacción ante los problemas que puedan suceder durante la recreación[33]. Al igual que con los videojuegos musicales, los videojuegos deportivos pueden ser considerados en algunas clasificaciones como videojuegos de simulación, pero sin embargo también puede ser considerados como un género propio debido a la variedad que existe dentro del género. Videojuegos deportivos podrían ser la saga *FIFA*, la saga *NBA2K* o la saga *Formula 1*.

2.2.4. Motor de videojuegos

En el apartado dedicado a la introducción de este Trabajo Fin de Grado se ha mencionado en varias ocasiones el término motor de videojuegos, el cual es una parte muy importante dentro del desarrollo de videojuegos. Sin embargo, en este apartado se pretende explicar un poco más el concepto y su origen, comentando posteriormente algunos ejemplos dentro de la variedad de motores de videojuegos que existen en el mercado, omitiendo Monogame ya que será explicado a posteriori.

2.2.4.1. ¿Qué es un motor de videojuegos?

[53]Al igual que ocurre en otras disciplinas de la informática, en el desarrollo de videojuegos existen herramientas que facilitan dicho desarrollo automatizando determinadas tareas y ocultando a vista del desarrollador la complejidad de ciertos procesos de bajo nivel.

El término motor de videojuegos surgió a mediados de los años 90 con la aparición del juego *Doom* de la compañía *id Software* bajo la dirección de John Carmack. Esta afirmación sobre el surgimiento del término se corrobora en el hecho de que *Doom* fue el primer videojuego diseñado con una arquitectura orientada a la reutilización mediante la separación de esta en módulos principales (renderizado gráfico, sistema de colisiones, sistema de audio, etc.).

Este planteamiento llevado a cabo por *id Software* facilitaba en gran medida la reutilización del software de un videojuego, siendo cada vez más popular gracias al uso por parte de los desarrolladores de módulos o juegos previamente licenciados para crear los suyos propios. Dicho de otra manera, gracias a este concepto se podía desarrollar un videojuego del mismo tipo sin modificar apenas el motor o núcleo de este, centrándose mayormente en el diseño artístico y las reglas del juego.

El concepto traído por los motores de videojuegos ha ido evolucionando con el tiempo expandiéndose y dando lugar a mods creados por desarrolladores independientes o la creación de grandes librerías, bibliotecas e incluso lenguajes de programación cuya finalidad es facilitar el desarrollo de videojuegos. A día de hoy, la gran mayoría de las empresas relacionadas con el desarrollo de videojuegos acaban o utilizando motores de terceras, lo cual puede suponer un ahorro económico y de tiempo, o desarrollando motores de videojuegos propios para luego licenciarlos y lanzarlos al mercado, obteniendo así sus beneficios.

Sin embargo, los motores de videojuegos no ofrecen una arquitectura preparada para el desarrollo de cualquier videojuego, ya que existe una cierta dependencia entre el motor y el género de videojuegos al que está destinado. Por esta razón, no podemos afirmar que sean reusables al cien por ciento, ya que las dependencias entre el motor y el género no nos dejarían, por ejemplo, desarrollar fácilmente un videojuego de simulación con un motor orientado a juegos de aventura.

Para concluir con este apartado, es importante comentar la evolución que están sufriendo actualmente los motores de videojuegos respecto a la generalidad mencionada en el párrafo anterior. En la actualidad, se está consiguiendo poco a poco desarrollar motores de uso general para distintos géneros de videojuegos, aunque la dicho avances aún no son suficientes. Esto conlleva a que la mayoría de veces cuando un estudio o desarrolladora quiere utilizar un motor concreto para su videojuego, necesite realizar ciertas personalizaciones sobre este para cumplir las necesidades de su juego.

2.2.4.2. Unity

[54]Unity es un motor de videojuegos desarrollado por *Unity Technologies* el cual funciona como herramienta para la creación de contenido interactivo y su distribución. Las posibilidades que ofrece son desde la creación de videojuegos hasta la realización de arte interactivo, pasando por recorridos arquitectónicos o simulaciones de entrenamiento online.

El editor de Unity, conocido como Unity Editor, es una interfaz totalmente integrada y extensible que permite al desarrollador crear sus propios videojuegos tanto en 2D como en 3D. Además, ofrece la posibilidad de crear escenas con terrenos, luces, sonidos, interfaces de usuario, personajes, físicas, etc. Otros dos aspectos a comentar son la posibilidad de programar las interacciones del jugador mediante scripts, en lenguajes tan comunes como Javascript o C# para luego ejecutarlos en Mono, y la posibilidad de probar el juego en cualquier punto de su desarrollo mediante el modo Play.

A nivel de plataformas para las que se puede desarrollar, Unity ofrece una amplia gama en la que se ofrecen la distribución para más de 25 plataformas distintas entre móvil, escritorio, consola, TV, VR, AR y la web. Además, solo es necesario una compilación única para la distribución en todas estas plataformas.

Algunos juegos desarrollados con Unity serían *Hollow Knight* de *Team Cherry*, *Cuphead* de *StudioMDHR* u *Ori and the Blind Forest* de *Moon Studios*.

2.2.4.3. CyEngine

[55]CryEngine es un motor de videojuegos desarrollado por Crytek el cual se encuentra actualmente en su versión 5. Este motor de videojuegos fue lanzado originalmente para la realización de una demo técnica de las tarjetas gráficas NVIDIA, pero gracias a los buenos resultados que consiguió el estudio acabo desarrollando *FarCry*, una saga de videojuegos que en la actualidad es de las más conocidas a nivel mundial.

Se trata de un motor desarrollado en C#, C++ y Lua el cual cuenta con infinidad de características tales como un sistema de físicas propio, altas capacidades gráficas o un sistema de inteligencia artificial para las entidades del videojuego. Además, cabe destacar que es un motor utilizado por una de las grandes compañías del desarrollo de videojuegos como es Ubisoft, y que otorga un gran apoyo a las desarrolladoras indies, ofreciendo el motor de manera totalmente libre y organizando competiciones entre desarrolladoras independientes con premios para estas.

Algunos videojuegos desarrollados con CryEngine serían *Everybody´s Gone to the Rapture* de *The Chinese Room*, *Prey* de *Arkane Studios* o *Kingdom Come: Deliverance* de *Warhore Studios*.

2.2.4.4. Unreal Engine

[56]Unreal Engine es un motor de videojuegos desarrollado por Epic Games que actualmente se encuentra en su versión 4. Al igual que CryEngine, Unreal Engine no fue concebido con la idea de ser distribuido al mercado, ya que en principio fue creado únicamente para el videojuego *Unreal*, pero al final acabo siendo comercializado debido a la opción de mercado que se le abrió a la empresa gracias a las necesidades de los desarrolladores.

Utilizando este motor es posible desarrollar utilizando C# o Blueprints, las cuales se definen como "cajas" que contienen scripts internos y que pueden ser conectadas unas con otras para definir la interacción del jugador con el videojuego. Su característica principal es su capacidad gráfica, la cual es considerada como la mejor probablemente dentro del mercado, especialmente en cuanto a los efectos de partículas, los VFX o el hiperrealismo de los materiales dentro del videojuego.

Algunos videojuegos que hacen uso de Unreal Engine son *Dragon Ball FigtherZ* de *Arc System Works* o los futuros *Kingdom Hearts III* y *Final Fantasy VII* de *Square Enix*.

2.3. Desarrollo de videojuegos

El desarrollo de videojuegos, al igual que cualquier desarrollo software, cuenta con un proceso de desarrollo establecido en el que deben intervenir todos los miembros del equipo de desarrollo durante distintas fases de este. La mayor diferencia con otros tipos de software es la que genera la composición del equipo, ya que en el desarrollo de videojuegos los equipos suelen tener cabida para artistas gráficos, artistas de sonido o diseñadores del gameplay (también conocidos como diseñadores de videojuegos) para los cuales hay que reservar ciertas fases del desarrollo que normalmente serían de desarrollo software como tal.

A continuación se va explicar brevemente la composición típica de un equipo de desarrollo de videojuegos y se hablará a posteriori de las fases dentro del proceso de desarrollo.

2.3.1. Roles en un equipo de desarrollo de videojuegos

2.3.1.1. Ingenieros

[57]Los ingenieros se tratan de los responsables de diseñar e implementar el software necesario para el videojuego y las herramientas que dan soporte a dicho videojuego. Normalmente, suelen dividirse en dos grandes grupos:

- **Programadores del núcleo:** Se tratan de los programadores cuyo trabajo está enfocado al desarrollo del código principal del videojuego y del motor de este, en el caso de que se parta de la idea de crear un nuevo motor.
- **Programadores de herramientas:** Se tratan de los programadores encargados del desarrollo de las herramientas a utilizar por el resto del equipo de desarrollo para conseguir mayor eficiencia a la hora de trabajar.

Además de esta gran clasificación, independientemente cada programador suele orientarse hacia ciertas disciplinas concretas, como puedan ser la programación gráfica o la inteligencia artificial. Sin embargo, cada día es más común encontrarse con los conocidos como ingenieros transversales dentro de los equipos pequeños, ya que estos no cuentan con suficiente capital como para tener en nómina a especialistas de las distintas disciplinas.

Finalmente, mencionar que dentro de estos equipos de desarrollos se suele otorgar al desarrollador con más experiencia, conocido como senior, ciertas dotes de liderazgo y supervisión. Esto implica que, además de dedicarse al desarrollo como tal, deberá llevar a cabo cierta toma de decisiones desde el aspecto técnico y deberá cumplir ciertas tareas de gestión del personal desde una perspectiva propia de los recursos humanos

2.3.1.2. Artistas

[57]Los artistas son los encargados de la creación de todo el contenido audio-visual del videojuego, como puedan ser los escenarios, los personajes, las animaciones, etc. Al igual que en el caso de los ingenieros, suelen dividirse en distintos grupos:

- **Artistas de concepto:** Se tratan de los responsables de crear los bocetos iniciales del videojuegos, los cuales permiten al resto del equipo de desarrollo hacerse una idea principal de cuál debe ser el aspecto principal de videojuego. Su función suele ser de las más importantes en las primeras fases del desarrollo.
- **Modeladores:** Son los responsables de la generación de cualquier contenido 3D del videojuego, desde los escenarios de este hasta cualquier entidad interna del mundo virtual.
- **Artistas de texturizado:** Realizan un trabajo bastante similar al de los modeladores pero en el ámbito del contenido gráfica 2D, como puedan ser las texturas o imágenes planas dentro del videojuego. Su trabajo en muchas ocasiones es utilizado por los modeladores para aplicar texturas a los modelos 3D.
- **Artistas de iluminación:** Se tratan de los responsables de gestionar las distintas fuentes de luz dentro del videojuego y sus propiedades estáticas y dinámicas.
- **Animadores:** Su tarea principal es la de implementar el movimiento de las entidades dinámicas del videojuego para que estas reflejen realismo, como por ejemplo a la hora de reflejar la manera de caminar de un personaje concreto.
- **Diseñadores de sonido:** Son los encargados de crear e integrar todo el aspecto sonoro que rodea al videojuego, desde las canciones de la banda sonora hasta los efectos más simples de sonido.
- **Actores:** Este suele ser un rol que no está presente en todos los videojuegos, debido a que es un extra para aportar realismo para la integración del jugador al videojuego. Normalmente suelen ser o actores de doblaje, los cuales se dedican exclusivamente a aportar una voz a los distintos personajes, o actores de captura de movimientos, los cuales facilitan el trabajo de los animadores para que las animaciones acaben siendo más realistas.

Mencionar como conclusión, que dentro del sector de los artistas también se suele presentar una figura de artista senior, el cual se encarga de la supervisión de los diferentes aspectos vinculados al componente artístico.

2.3.1.3. Diseñadores de juego

[57]Los diseñadores de juego son los responsables de diseñar todo el contenido relacionado con el gameplay, es decir, la parte interactiva del videojuego que genera la experiencia del jugador. Dentro de los diseñadores de juego se podrían apreciar tres niveles. Por un lado se tienen a los diseñadores que trabajan a un nivel de detalle menor y se centran en aspectos como la historia, la secuencia de niveles o los objetivos principales del videojuego. Por otro lado se encontrarían los diseñadores que se centran en un nivel concreto del mundo virtual del videojuego, con lo cual toman decisiones del tipo de qué tipos de enemigos tendrá ese nivel y con qué frecuencia aparecerán o los subobjetivos a completar en dicho nivel. Por último, ciertos diseñadores enfocan su tarea a un nivel más técnico, trabajando a menudo mano a mano con los ingenieros y/o realizando tareas de programación mediante scripts en leguajes de alto nivel como puedan ser Lua o Python.

2.3.1.4. Productores

[57]Los productores son una figura “opcional” dentro del equipo de desarrollos de videojuegos, ya que es posible desarrollar un videojuego sin contar con la figura de este como suele ser el caso en las empresas más pequeñas. Sus funciones suele variar entre la gestoría de agenda y recursos humanos, las de un diseñador senior o las de enlace entre el equipo de desarrollo y la unidad de negocios de la empresa.

2.3.1.5. Editores

[57]Aunque no suelen pertenecer como tal al equipo de desarrollo, ya que suele ser una empresa ajena con la que el equipo de desarrollo firma un contrato que le ofrezca las mejores condiciones, los editores o publishers son los encargados de llevar a cabo todas las tareas tanto de marketing del videojuego como de su distribución. En función del tipo de contrato que se realice con estas, se suele hablar de equipos de desarrollo first-party, los cuales trabajan directamente para una empresa de fabricación de consolas, o los equipos de desarrollo third-party, los cuales trabajan con editoras ajenas a las distintas plataformas y que se encargan de distribuir el videojuego en las plataformas que puedan suponer mayores beneficios.

2.3.2. Proceso del desarrollo de videojuegos

2.3.2.1. Pre-producción

[58]La etapa de pre-producción o fase de diseño es la fase en la que se planifica el proyecto centrándose en la idea sobre el videojuego y la producción documentación clara y fácil de entender sobre el diseño inicial. Para esto, el primer proceso a establecer es el de la realización de brainstorming o lluvia de ideas, mediante el cual se establecerán las ideas principales sobre el videojuego sobre las que se trabajará durante el desarrollo. Tras esto se comenzará a documentar el proyecto, al mismo tiempo que se van realizando prototipos del arte final de este y buscando o generando herramientas que faciliten el trabajo a posteriori. De forma gráfica, el proceso de pre-producción a llevar a cabo se refleja en el siguiente esquema de la *Figura 11*:

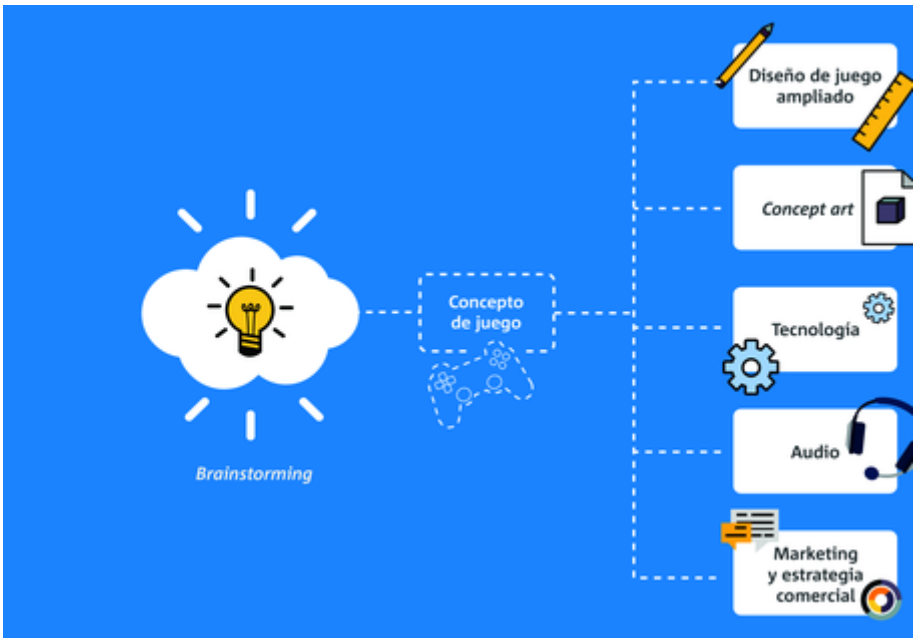


Figura 11. Esquema gráfico del proceso de pre-producción[58]

Finalmente, algunas recomendaciones para realizar esta primera fase de pre-producción serían las siguientes:

- Limitar el tiempo de brainstorming y realizar pausas en este, ya que muchas veces es necesario refrescar las ideas.
- En función del tamaño del equipo de desarrollo, es conveniente involucrar a todos los miembros o únicamente a los líderes de cada sección de este a la hora de desarrollar el primer documento oficial del proyecto, el concepto de juego.
- El concepto de juego debería incluir los siguientes aspectos: ficha del producto, descripción y objetivo del juego, mecánicas del juego, referencias y análisis de los posibles riesgos y sus soluciones.
- Una vez terminado dicho documento, este debe ser distribuido entre todos los miembros del equipo para que lo revisen y se consiga establecer una línea de trabajo única
- En ciertas ocasiones es posible sustituir el concepto de juego por un documento mayor conocido como el ten-page-document, en el cual se amplía la información del anterior con descripciones sobre los personajes, los enemigos, los escenarios, etc. Este documento suele estar más orientado al sector artístico del equipo de desarrollo y al apartado encargado del marketing y la relación con los editores.

2.3.2.2. Producción

[58]La producción es la etapa más larga del proceso de desarrollo de videojuegos y, por lo tanto, la más costosa a nivel económico y de recursos. Durante esta fase, todas las secciones dentro del equipo de desarrollo trabajan codo con codo para ir generando los ítems que componen el videojuego, mientras se van mejorando las herramientas y sistemas.

Algunos de los aspectos más importantes que ocurren durante este periodo son la programación del código del videojuego, el diseño de niveles, la producción artística y musical o la realización de pruebas

por parte de los testers (personal encargado principalmente de poner a prueba el videojuego y encontrar los fallos o bugs de est).

Mencionar además, que durante este proceso de producción se van generando distintas versiones del videojuego conocidas como alfa, beta y release:

- **Alfa:** se trata de la primera versión oficial del videojuego en la cual se permite al usuario conocer y probar un gameplay muy limitado pero definitorio sobre el que acabará siendo. Esta versión alfa solía estar orientada para realizar las primeras demostraciones a los editores o publishers, pero en la actualidad los estudios de desarrollo suelen ofrecer un acceso abierto o por invitación a los usuarios para que mientras jueguen generen complejos informes para mejorar la experiencia de usuario, la corrección de bugs, etc.
- **Beta:** la versión beta es una versión mucho más cercana a lo que será la versión de lanzamiento o release, aunque suele tener ciertas limitaciones en cuanto a contenido. En esta versión, la aparición de bugs es mucho menor que en la versión alfa, siendo su objetivo ser testeada por el mayor número de usuarios posibles para que el equipo pueda recibir un primer feedback de la comunidad con respecto a su videojuego.
- **Release:** esta versión se trata de la versión de lanzamiento, es decir, la que se distribuye en las plataformas de venta de videojuegos y la que podrán disfrutar al cien por cien los jugadores. Esta versión seguirá mejorándose durante el periodo de post-producción e incluso ampliándose con contenido adicional, pero podría considerarse como una versión final y completa del videojuego. Mencionar además que su fecha de entrega es la más importante a nivel de editores y publishers, ya que las campañas de marketing suelen ir ligadas a esta fecha y además puede suponer una importante diferencia en cuanto al éxito y los beneficios generados por el videojuego.

2.3.2.3. Post-producción

[58]Tras lanzar al mercado la versión release, el proyecto pasaría a la fase de post-producción, en la cual el equipo de desarrollo se centra en mejorar el juego a nivel técnico resolviendo los bugs que hayan podido aparecer tras el lanzamiento, mejorando las funcionalidades y algoritmos que definen los comportamientos y mecánicas del videojuego y añadiendo contenido adicional para extender la vida útil del videojuego. Este concepto de vida útil es bastante importante en los videojuegos, ya que dependiendo de la duración de esta el estudio de desarrollo obtendrá más o menos beneficios para poder continuar y mejorar su trabajo. Otra forma de mantener dicha vida útil sería mediante la introducción, ya sea durante la producción o la post-producción, sistemas de rankings y puntuaciones online que motiven al jugador a marcarse objetivos y superarse.

2.4. Breve historia de los videojuegos

La historia de los videojuegos es como tal una historia relativamente breve, ya que, como veremos durante esta sección, los orígenes de esta datan de 1958. Aun llevando únicamente 60 años entre nosotros, los videojuegos han sufrida una evolución estratosférica durante este periodo, pasando de Tennis for Two (considerado como el primer videojuego de la historia) a juegos de realidad virtual o con gráficos ultrarrealistas que acabarán desembocando en mundos virtuales donde los jugadores puedan realizar todo aquello que no pueden en la vida real.

En el transcurso de este apartado, se irán definiendo distintas etapas dentro del mundo de los videojuegos que el autor considera como suficientemente importantes como para dedicarles cierto espacio en este Trabajo Fin de Grado.

2.4.1. De Higginbotham al *Pac-Man*

[33]En el año 1958, el físico estadounidense William Higginbotham utilizó un programa militar, cuya finalidad era la de calcular las trayectorias de los misiles nucleares, para recrear junto a dos mandos y un osciloscopio a modo de monitor el movimiento de una pelota de tenis. De esta manera, William acabó creando así, sin ser consciente, el considerado como primer videojuego (ya que contaba con un monitor que lo convertía en "video") de la historia, *Tennis for Two*, el cual se trataba de un pequeño juego de tenis de visión lateral que calculaba el movimiento de la bola y permitía elegir la altura de la red. Aun así, Higginbotham no lo vio como algo más que un simple pasatiempo para los físicos que pasaban por el laboratorio donde él trabajaba y no lo llegó a patentar, debido a que estaba más centrados en sus investigaciones y su activismo antinuclear.

Años después de este primer videojuego, el estudiante Steve Russell del Instituto de Tecnología de Massachusetts evolucionó un proyecto propio llamado *Spacewar* para poder llevarlo a una pantalla de rayos catódicos utilizando un ordenador DEC PDP-1. De esta manera, *Spacewar* se convirtió en el segundo videojuego de la historia, pero al igual que con *Tennis for Two*, Steve Russell estaba más centrado en sus investigaciones que en patentar dicha idea y verla como algo más que un pasatiempos. Es en este momento fue cuando apareció Nolan Bushnell, un empresario que si vio más posibilidades para *Spacewars* y, al no contar este con una patente, lo plagió creando *Computer Space*, el cual trataría de distribuir junto a la empresa de recreativas *Nutting*, aunque no tuvo el éxito esperado.

Sin embargo, la empresa *Nutting* se quedó con los derechos de *Computer Space* y consiguió finalmente hacer rentable la máquina recreativa que permitía jugar a dicho videojuego, volviéndose este muy popular entre los adolescentes de la época. Por otro lado, Bushnell acabó creando la empresa archiconocida *Atari* en 1972 y desarrollando bajo esta un juego conocido como *Pong* (el cual era prácticamente una copia del *Tennis for Two* de Higginbotham), el cual consiguió un éxito inmediato. Gracias al éxito de estas dos empresas con sus respectivos juegos, fueron surgiendo pequeñas empresas que querían agregarse a la carrera en torno al diseño de videojuegos, de la cual era *Atari* el mayor exponente llegando a desarrollar un juego al mes durante 1974.

Una vez que la industria del videojuego se asentó, la empresa *Atari* de Nolan Bushnell aprovechó todos los beneficios obtenidos a través de sus arcades de *Pong* y otros juegos para adentrarse en el mundo de las consolas domésticas, siendo la primera de estas la Magnavox Odyssey en 1972. Con el paso del tiempo y la mejora de microprocesadores, la revolución de las consolas domésticas llegó a la industria a través de consolas como la Atari 2600 o la Intellivision de *Mattel*.

Entrados en los años ochenta y una vez superados los estragos de la Segunda Guerra Mundial, el mundo audiovisual evolucionó hacia un estilo más pop o intrascendente que permitiera al usuario disfrutar sin complicaciones, lo cual se reflejó por supuesto en el mundo de los videojuegos. Las pioneras en este aspecto fueron las empresas japonesas *Namco* y *Nintendo*, las cuales trajeron a la industria dos de los grandes clásicos de la industria como son *Pac-Man* (*Figura* 12) de Tōru Iwatani y *Super Mario Bros* de Shigeru Miyamoto respectivamente, los cuales trajeron una mayor estética pop y alegre convirtiéndose ambos en los juegos más comercializados de los años 80.

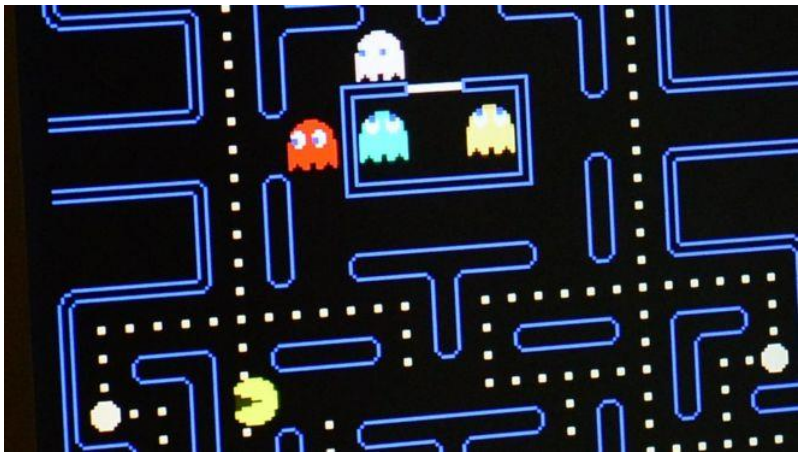


Figura 12. Captura del videojuego Pac-Man[63]

2.4.2. De los Atari a la época dorada del Spectrum

[33]Con la llegada de las consolas domésticas, llegó a su vez el formato de los cartuchos, los cuales permitían disfrutar en una misma plataforma de una amplia variedad de videojuegos con distinto gameplay y no de una cantidad preestablecida para cada consola. Viendo el potencial del mercado de los videojuegos, la empresa audiovisual *Warner* decidió llevar a cabo la compra de *Atari*. Sin embargo, esto coincidió con un momento en el que la proliferación de consolas y empresas incompatibles entre sí, el plagio constante en las mecánicas del gameplay y la falta de títulos atractivos para el jugador se llegó a la considerada primera crisis de la industria del videojuego a finales de los setenta. Esta crisis se prolongó durante varios años perjudicando a la industria durante este tiempo, pero aun así gracias a esta el mercado se saneó y marcó un nuevo camino en el mundo de los videojuegos. Las consecuencias de esta crisis fueron muy distintas a lo largo del globo: los estadounidenses trataron de reinventarse, mientras que el mercado asiático aposto fuertemente por la consola *NES* (*Nintendo Entertainment System*) de la empresa japonesa *Nintendo* y, por otro lado, en Europa triunfaron consolas como *Spectrum*, *Commodore* o *Amstrad*, las consolas de la generación de los 8 bits (la cual además fue una muy buena época para los desarrolladores españoles).

Figura 13. Consola ZX Spectrum de la empresa Sinclair Research[64]

Merece mención especial la consola lanzada en 1982 por la empresa británica *Sinclair Research*, la *ZX Spectrum*, mostrada en la *Figura 13*. Se trataba de una consola doméstica barata, asequible y de fácil manejo que estaba preparada para la nueva generación de los 8 bits, lo cual la popularizó automática entre los adolescentes de la época. Las características principales a nivel técnico fueron el aumento a quince colores gráficos y su procesador de 3.5Mhz con el famoso bus de datos de 8 bits, el cual se

convirtió en un estándar de la generación y de la cual coge su nombre esta. Pero quizás una de los aspectos más importantes que consiguió la *Spectrum* fue el desarrollo de la imaginación y los subgéneros que surgieron alrededor de esta. En pocas palabras, se podría hablar de que esta consola, junto a la *NES* de *Nintendo*, marcaron un antes y un después en la industria.

2.4.3. De los 8 bits al *Castillo de Wolfenstein* 3D

[33]Llegaron los años noventa y llegó a la industria de los videojuegos el formato del CD-ROM, las nuevas capacidades digitales, más memoria, más limpieza, más velocidad en los procesadores, etc. En definitiva, se podría decir que en los años noventa llegó una revolución tecnológica sin igual al mundo de los videojuegos, la cual vino acompañada de los novedosos gráficos 3D. El primer videojuego que implantó esta tendencia gráfica fue *Virtual Racing* de la consola *MegaDrive*, pero fue con la llegada de los 32 y 64 bits junto a dos de las consolas más importantes de la historia, *Nintendo 64* de *Nintendo* y *PlayStation* de *Sony Computer Entertainment*, cuando los gráficos 3D se instauraron como estándar en los videojuegos (a excepción de que en los últimos años se han vuelto a poner de moda los juegos tanto en 2D o 2.5D como *Dragon Ball FighterZ* o *Megaman 11* en gran medida por la nostalgia hacia lo retro).

En paralelo a toda esta evolución de las consolas domésticas, las máquinas recreativas o arcades iban perdiendo fuelle de manera exponencial, por lo cual decidieron realizar una gran apuesta por los grandes soportes físicos (simuladores de aviones, coches, pistolas, etc.). Esta apuesta fue suficiente para mantener en cierta manera el mercado de las arcades, sin embargo ya no eran capaces de competir de ninguna manera con las consolas de sobremesa y los ordenadores personales.

Figura 14. Captura del videojuego Wolfenstein 3D[65]

Mencionar de manera destacada, que durante esta época de la industria de los videojuegos se desarrolló para PC el videojuego *Wolfenstein 3D* de *id Software*, un videojuego de disparos en primera persona que cambió nuevamente la forma de jugar. Este juego combinaba gráficos rasterizados en paredes, cuadros y puertas con personajes 2D adaptados a la visión del jugador, como se aprecia en la *Figura 14*.

2.4.4. La gran industria del videojuego

[33]Con el paso del tiempo, fueron pocas las empresas desarrolladoras de consolas que consiguieron mantener una cuota de mercado lo suficiente como para competir con las siempre fieles ventas de los ordenadores personales o PC, ya que estos no solo cumplían la función de ejecutar videojuegos sino que

también cumplían tareas que poco a poco iban siendo diarias y "obligatorias" como la ofimática o la navegación por Internet. Estas empresas que se consiguieron mantener con el paso de las generaciones fueron *Sony Computer Entertainment*, cuya consola principal en la actualidad es *PlayStation 4*, *Microsoft*, con *Xbox One*, y *Nintendo*, representada por su famosa consola híbrida *Nintendo Switch*. Es importante mencionar en este aspecto, la comparación eterna entre dichas consolas y el PC, la cual lleva a los jugadores a discutir si es preferible la potencia y menor coste de los PC o la cantidad de exclusivos y la inmersión que generan las consolas.

Otro aspecto a destacar en la evolución de los videojuegos, sería la cantidad de formas de jugar que existen en la actualidad, donde probablemente la empresa japonesa *Nintendo* tenga gran parte de culpa. En el año 2006, *Nintendo* lanzó al mercado una consola de menor potencia que sus competidoras, *PlayStation 3* y *Xbox 360*, pero con una característica revolucionadora en cuanto gameplay, ya que su controlador principal se trataba del *Wiimote*, un mando que reconoce el movimiento del jugador mediante el uso de la detección espacial y los giroscopios. Además, este no fue el único controlador revolucionario en cuanto al gameplay del que disfrutarían los jugadores de *Wii*, ya que a posteriori se añadió el soporte *Wii Fit*, una tabla que utiliza el control gravitatorio del peso del cuerpo para su aplicación a diferentes aspectos como la realización de ejercicio físico al mismo tiempo que se juega o de yoga. Resaltar además, que estas tecnologías introducidas por *Nintendo* junto a las mejoras gráficas (llegando al realismo otorgado por el Full HD o el 4K), probablemente puedan ser consideradas como precursoras de lo que hoy en día se denomina como la inmersión total de los videojuegos, la realidad virtual mediante gafas y controladores similares al *Wiimote* (como se puede apreciar en el conjunto de controladores de *Oculus Rift* de la *Figura 15*).

Figura 15. Casco de realidad virtual Oculus Rift y sus controladores Touch System[66]

Paralelamente a todo esto, llegan al mercado los primeros móviles 3G y *smartphones*, con sus procesadores más potentes. Es esta característica la cual hace que estos nuevos dispositivos sean considerados como pequeñas consolas portátiles, además de todas sus funciones principales relacionadas con la comunicación. El mercado del videojuego orientado a este tipo de dispositivos es mucho más casual que en el caso de las consolas y el PC, ofreciendo así títulos como *Candy Crush* o *Clash Royale* que permiten entretener al jugador en sus ratos libres sin necesitar de una gran cantidad de tiempo. Además, es con la llegada de los videojuegos a los móviles cuando los videojuegos estrella de otras generaciones vuelven a resurgir, llegando multitud de *remakes* o remasterizaciones de juegos como *Super Mario Bros*, *Pokemon* o *Final Fantasy*.

Un aspecto fundamental para entender el estado actual de los videojuegos, sería la capacidad *online* de estos. En la actualidad, prácticamente cualquier videojuego cuenta con cierto matiz online ya sea mediante rankings, competiciones o interacciones, lo cual, como se ha mencionado anteriormente en este Trabajo Fin de Grado, es una característica muy importante a la hora de alargar la vida de un videojuego. Mención aparte requerirían los videojuegos conocidos como MMO (Massive Multiplayer Online), los cuales han creado una nueva generación de jugadores y un nuevo modo de relación social entre estos. Videojuegos como *League of Legends* o *World of Warcraft* han abierto cabida a la reunión de millones de jugadores en un mundo virtual y la interacción entre estos, generando así comunidades internacionales alrededor de estos videojuegos.

Hablando desde un punto de vista más profesional, el mundo del desarrollo de videojuegos ha evolucionado a la par que estos, fomentando desde el ámbito educativo el desarrollo de profesionales de la industria. En las universidades se empiezan a encontrar grados universitarios en Diseño y Desarrollo de Videojuegos, los cuales permiten a los jóvenes involucrarse y formarse en el ámbito al que muchos consideran como una pasión. Cada vez empiezan a aparecer más estudios sobre los videojuegos con temáticas desde el desarrollo de estos hasta el estudio de los jugadores de un juego concreto. Es también importante la introducción de los videojuegos por parte de profesionales de la educación y la psicología, permitiendo el desarrollo personal o la detección y ayuda respecto a los problemas psicológicos y sociales haciendo uso de los videojuegos.

2.4.5. Los eSports

[33]Finalmente, para finalizar con este breve recorrido por la historia de los videojuegos, se va a proceder a comentar uno de los aspectos más candentes de estos, los eSports o deportes electrónicos.

Los eSports surgen en gran medida a través de la plataforma gratuita de contenido audiovisual *Youtube*. La dificultad de los videojuegos o su precio hacen que muchos usuarios recurran a partidas guiadas por *youtubers* que enseñan a su público el gameplay de un juego o simplemente se dedican a jugar y comentar al mismo tiempo, haciendo uso de sus propios sellos personales para enganchar al espectador. Gracias estos creadores de contenido y sus gameplays, se empiezan a popularizar juegos donde prima la competición como puedan ser *Call of Duty*, *FIFA* o *League of Legends*, los cuales allanan el camino hacia el futuro de la competición virtual: los eSports y sus competiciones masivas (*Figura 16*).

Figura 16. Final de la League of Legends World Championship 2018[67]

En la actualidad, el fenómeno emergente de los deportes electrónicos es capaz de llenar recintos con sus competiciones oficiales y atraer a marcas e inversores gracias a la cantidad de beneficios que generan. Estas competiciones o torneos suelen ser retransmitidas mediante *streamings* online los cuales cuentan con millones y millones de visitas, hasta el punto de rivalizar con medios de comunicación tan famosos como la televisión o la radio.

Por otro lado está por discutir si es posible considerarlos como deportes al uso, en torno a lo cual existe un debate muy amplio y sobre el que se ha especulado incluso la posible inclusión de los eSports como deportes olímpicos. Pero lo cierto es que los jugadores profesionales dentro de este ámbito están sometidos a retos físicos de concentración, habilidad y rapidez mental similares a los de los profesionales de otros deportes.

2.5. Actualidad de la industria de los videojuegos

A la hora de desarrollar este apartado, el autor ha hecho uso de los conocimientos y estadísticas ofrecidas por el Libro Blanco del Desarrollo Español de Videojuegos 2017[8], que ofrece datos basados en un análisis exhaustivo de la industria del videojuego en España. Estos datos se recogieron durante el mes de mayo de 2017.

2.5.1. La industria del videojuego a nivel mundial

Se estima que la industria del videojuego es la industria tecnológica con mayor crecimiento a nivel mundial. Este dato se debe a la venta online, a que los videojuegos alcanzan más ámbitos además del ocio, al impulso de la banda ancha en todo el mundo y a los nuevos modelos de negocio como el *free to play* o la introducción de publicidad en los videojuegos. Además, se trata de una industria global y que permite la localización de sus productos en cualquier parte del mundo.

Según estimaciones realizadas en el Libro Blanco del Desarrollo Español de Videojuegos 2017, se especula con que el mercado de los videojuegos crecerá un 8.2% anualmente entre los años 2018 y 2020, estimando unos ingresos aproximados de 143.5 billones de dólares (un 23.7% más que en 2017).

Además, se especula que a nivel mundial existen 2200 millones de jugadores y jugadoras en el mundo, representando las mujeres un 47% de esta cifra (lo cual es un dato a favor de la igualdad en torno al mundo de los jugadores de videojuegos). Es destacable también que un 47% de los jugadores, es decir, mil millones de jugadores gastan dinero mientras juegan en las tiendas virtuales de los videojuegos.

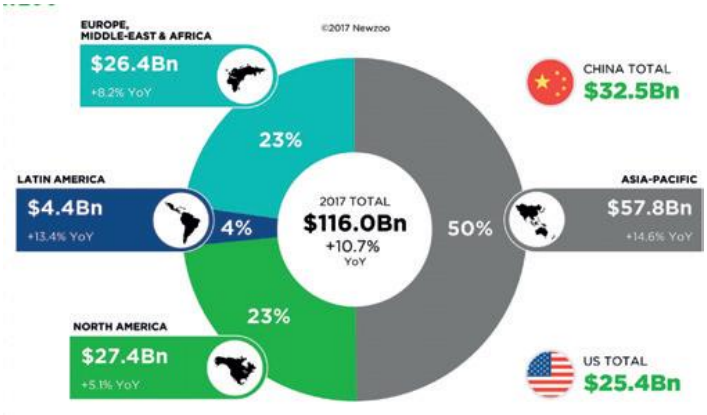


Figura 17. Mercado mundial de videojuegos en 2017 por regiones[8]

Como se aprecia en la *Figura 17*, se puede apreciar que la región más potente dentro del mercado de los videojuegos sería la región Asia-Pacífico, que cuenta con un sorprendente 50% de la cuota de mercado global. Sin embargo, es destacable mencionar que la segunda potencia a nivel de país, tras China, no se trata de un país asiático sino que se trata de Estados Unidos con 25.4 billones de dólares generados.

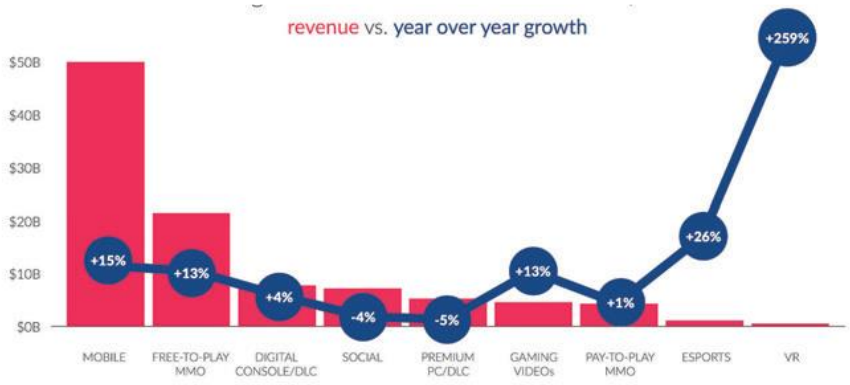


Figura 18. Mercado mundial del videojuego en 2017 por modelo de negocio[8]

Para finalizar con el mercado global, es importante apreciar en la *Figura 18* como los tres mercados con mayor crecimiento anual en la actualidad son la realidad virtual, los eSports y los videojuegos para móviles. Este dato ratifica el hecho de que la realidad virtual es vista como el futuro inmediato de los videojuegos, como se ha mencionado anteriormente en este Trabajo Fin de Grado.

2.5.2. La industria del videojuego a nivel español

En el ámbito del mercado español, los ingresos generados durante el 2017 ascienden a 1.9 billones de dólares, situándose como el cuarto país que más videojuegos consume, siendo Francia, Reino Unido y Alemania las siguientes respectivamente. Además su tasa de crecimiento especulada para los años 2018, 2019 y 2020 se encuentra en la media global con 8.2%.

El aspecto a destacar principalmente en España es el del crecimiento de la industria y las empresas que lo forman, habiendo sido creadas en los últimos diez años el 80% de las empresas que están activas hoy en día (*Figura 19*). Además, es apreciable como dichas empresas en su mayoría se tratan de estudios indies o independientes y por lo tanto de pequeñas empresas con menos de 10 empleados (*Figura 20*).

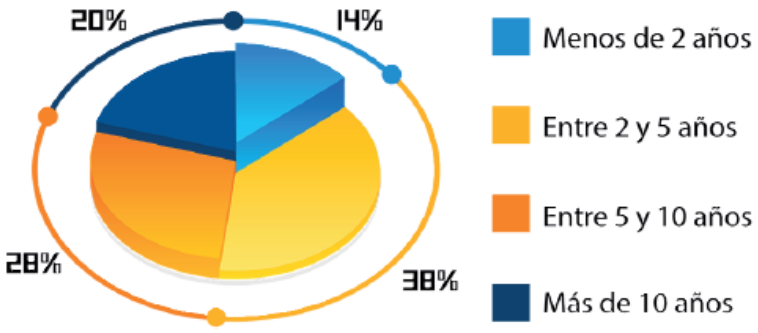


Figura 19. Distribución de las empresas de videojuegos españolas por antigüedad[8]

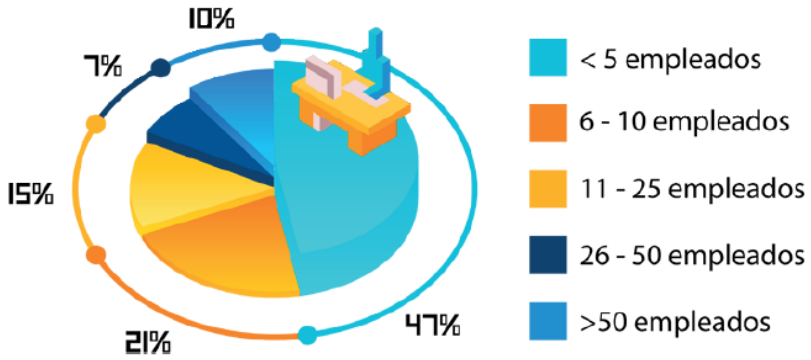


Figura 20. Distribución de las empresas de videojuegos españolas por empleados[8]

Otro dato a comentar es la creación de empleo que generará la industria del videojuego entre los años 2018 y 2020, cuyo crecimiento se estima sobre el 20% anual llegando a 11420 trabajadores del sector aproximadamente para 2020 (*Figura 21*).

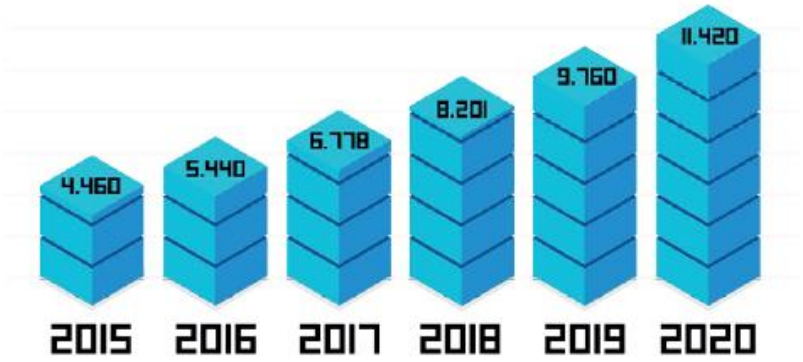


Figura 21. Evolución prevista del empleo en el sector (número de empleados/as) [8]

Finalmente, como se aprecia en la *Figura 22*, la distribución de las empresas de desarrollo de videojuegos o relacionadas con el sector se encuentra principalmente en las comunidades más "capitales" dentro del territorio español, como son Madrid, Cataluña y la Comunidad Valenciana entre las que acumulan un 68.1% de las empresas del sector en España. Destacar además, que Cataluña, con Barcelona a la cabeza, es la región con mayor crecimiento a costa de la disminución del peso de Madrid en este aspecto. Tras estas tres potencias, se encontrarían comunidades en auge dentro del sector como son Andalucía y el País Vasco, las cuales acaparan un importante 14.6%.

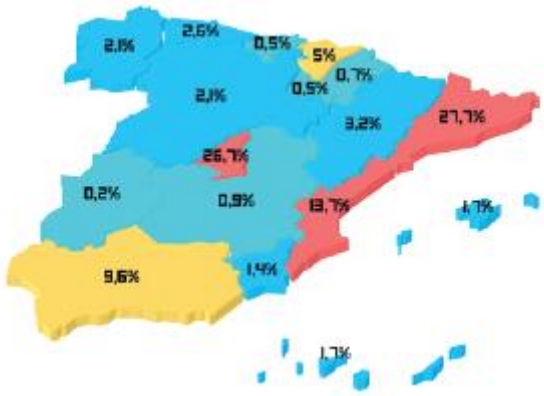


Figura 22. Distribución territorial de las empresas y estudios de videojuegos[8]

3. Dragon´s Curse

3.1. Introducción

En este apartado se llevará a cabo una explicación inicial sobre el motor de videojuegos usado en este Trabajo Fin de Grado, MonoGame. Durante esta explicación se profundizará en este motor de videojuegos concreto de una manera mayor a la realizada en el apartado "2.2.4. Motor de videojuegos" con otros motores (Unity, CryEngine y Unreal Engine) que fueron usados de ejemplos. Tras esta introducción y explicación de MonoGame, se procederá a analizar el proyecto sobre el que trata esta memoria, Dragon´s Curse. En este análisis se estudiarán aspectos como el análisis inicial del proyecto, el diseño de este, la implementación llevada a cabo, el periodo de pruebas y el feedback recibido, la planificación temporal llevada a cabo y las herramientas utilizadas.

3.2. MonoGame

[3]MonoGame (*Figura 23*) es un framework, más concretamente un motor de videojuegos, el cual surge como una versión Open Source del framework de desarrollo de videojuegos de Microsoft, Microsoft XNA 4. El motivo principal de crear una versión Open Source de XNA era la posibilidad de lanzar los juegos desarrollados con este framework en otras plataformas, ya que solo permitía llevar a cabo proyectos para las plataformas de la propia Microsoft como Xbox o Windows. Además, el hecho de que en abril de 2014 Microsoft decidiera dejar de soporte a su framework de desarrollo de videojuegos[68] hizo más necesario aun MonoGame para que los desarrolladores de XNA pudieran continuar y no tirar por la borda sus desarrollos, además de abrirles el camino hacia la portabilidad de los juegos multiplataforma.

Figura 23. Logo del motor de videojuegos MonoGame[3]

Podría decirse entonces, que la característica diferenciadora de MonoGame respecto a XNA 4 sería la posibilidad de crear videojuegos portables a las distintas plataformas del mercado, ampliando así considerablemente las posibilidades ofrecidas por XNA en este aspecto. En la actualidad, MonoGame cuenta con una parte abierta para todo el mundo, la cual permite desarrollar para dispositivos *smarts* (smartphones de los tres principales sistemas operativos y televisiones inteligentes) y para los principales sistemas operativos de PC, y una parte adicional para desarrolladores registrados, la cual permite ampliar estos dispositivos anteriores al mundo de las consolas actuales. [3]Por lo tanto, el abanico final del que dispone un desarrollador MonoGame sería el siguiente:

- **Ordenadores o PCs:**
  - Windows 10 (Univeral Windows Platform[69])
  - Windows Win32 (OpenGL[70] y DirectX[71])
  - Linux (OpenGL)
  - Mac OS X (OpenGL)

- **Dispositivos móviles/tablets:**
  - Android (OpenGL)
  - iPhone/iPad (OpenGL)
  - Windows Phone 10 (Universal Windows Platform)
  - Televisión
  - tvOS
- **Consolas:**
  - PlayStation 4 (Sony)
  - PlayStation Vita (Sony)
  - Xbox One (Microsoft)
  - Nintendo Switch (Nintendo)

Para terminar con esta breve introducción a MonoGame, es apreciable destacar algunos de los videojuegos portados de XNA o desarrollados directamente sobre este motor, entre los cuales se encuentran títulos como *Fez*, *Stardew Valley*, *Apotheon*, *Celeste* o *Skulls of the Shogun*.

3.2.1. Game1.cs y el bucle de juego

Al generar un proyecto de MonoGame, la base que proporciona este motor de videojuegos son dos clases denominadas como Program.cs y Game1.cs (cuyo nombre suele ser sustituido por el nombre del proyecto) y una carpeta llamada Content, la cual permitirá al desarrollador almacenar todo el aspecto artístico necesario para el videojuego. La clase Program.cs se trata de una clase bastante sencilla, la cual se encarga de generar y ejecutar un objeto del tipo Game1 haciendo uso del atributo STAThread que se encarga de que el videojuego use un único hilo destinado al objeto Game1.

Mención aparte merece la clase Game1.cs, ya que esta es realmente la que implementa toda la funcionalidad del videojuego y la que se encarga de gestionar todos sus aspectos. Por un lado, esta clase contiene dos variables importantísimas en MonoGame: un objeto del tipo GraphicsDeviceManager, el cual cuenta con un atributo GraphicsDevice que permite que el videojuego acceda a la GPU y se hace necesario para cualquier acción gráfica, y un objeto del tipo SpriteBatch, el cual se trata del encargado de mostrar los sprites y los textos por pantalla haciendo uso de la propiedad GraphicsDevice del objeto GraphicsDeviceManager mencionado anteriormente. Por otro lado, esta clase hereda de la clase Game de Monogame, implementando 5 métodos a sobrescribir que marcan el flujo interno dentro del juego (marcado en la *Figura 24*):

- **Initialize:** este es método es llamado una única vez por ejecución y en él se inicializan todos los objetos y variables necesarios para la ejecución de la clase Game1.
- **LoadContent:** es el método que se ejecuta tras Initialize y realiza la carga de todos los aspectos artísticos del videojuego. Este método se realiza una única vez, al igual que Initialize, por ejecución y es el que abre paso al bucle que recorrerá el videojuego durante todo su funcionamiento.
- **UnloadContent:** se trata del método opuesto a LoadContent y, al igual que este, está relacionado con la parte artística del videojuego. Sin embargo, su función es la de ser ejecutado cuando se sale del bucle mencionado en LoadContent y descargar el contenido artístico antes de la finalización de la ejecución global del videojuego.
- **Update:** este es el método que implementa toda la parte lógica del videojuego, comprobando los cambios ocurridos desde su última ejecución y actuando en consecuencia a estos.

- **Draw:** es el método encargado de mostrar en pantalla todos los elementos necesarios para el videojuego, para lo cual debe hacer uso de las variables GraphicsDevice y SpriteBatch.

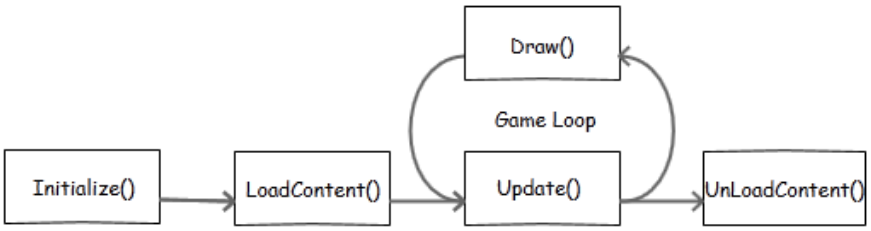


Figura 24. Diagrama del bucle de juego en MonoGame[72]

[4]En referencia a todo este aspecto del bucle de juego, es importante apreciar la diferencia a la hora de ejecutarse entre el software de un videojuego y el del resto de aplicaciones convencionales. Las aplicaciones convencionales suelen funcionar y ejecutarse mediante el registro de eventos, quedándose la aplicación en una especie de estado de *standby* entre los distintos eventos que van ocurriendo en su ejecución e informan a la aplicación de cambios. Sin embargo, los videojuegos funcionan mediante *polling*, lo cual hace que este se actualice constantemente y sea él mismo quien se encargue de informase del sistema sobre los cambios ejecutados por el usuario o por otros agentes. Esta diferencia se debe a que al contar con animaciones y aspectos ajenos a los posibles eventos que se generen, si los videojuegos estuvieran planteados para ejecutarse en función de un registro de estos eventos, el videojuego daría una impresión de estar "congelado" mientras que el usuario no realice acción alguno sobre el controlador.

3.2.2. Características de MonoGame

Este subapartado servirá como introducción a las características principales que necesita conocer cualquier desarrollador que use MonoGame, además del bucle de juego explicado en el subapartado anterior.

3.2.2.1. Sistema de entrada de datos del usuario

El sistema de entrada ofrecido por MonoGame permite detección tanto de cambio en un teclado, en un ratón o en un mando o gamepad. Estos cambios en los distintos controladores son gestionados mediante unas clases específicas que acceden a estructuras concretas para cada uno de ellos, el teclado hace uso de la clase Keyboard con la estructura KeyboardState, el ratón hace uso de la clase Mouse con la estructura MouseState y el mando hace uso de la clase GamePad con la estructura GamePadState. Al usar estas clases, MonoGame accede al estado actual de cada uno, guardado en las estructuras mencionadas, mediante el método GetState y obtiene distintas propiedades que interpretar.

La estructura KeyboardState cuenta con una serie de métodos que comprueban el estado de todas las teclas a la vez o de una tecla concreta. Para realizar la comprobación de todas estas teclas, representadas mediante la enumeración Keys, se hace uso del método GetPressedKeys, el cual devuelve un array con todas las teclas que se encuentren pulsadas en el momento de utilizar dicho método. Si lo que se necesita es comprobar el estado de una tecla en particular, se cuenta con dos métodos: el método IsKeyDown, al cual se le pasa como parámetro una tecla de la enumeración Keys y devuelve un bool en función de si la tecla está pulsada, y el método IsKeyUp, el cual realiza la misma función que IsKeyDown pero devolviendo el bool en función de si la tecla no está pulsada.

La estructura MouseState permite comprobar dos aspectos fundamentales del estado de un ratón: la posición en la pantalla del cursor y el estado de los botones de este. Para comprobar la posición del cursor, existen dos posibilidades: la propiedad Position, que devuelve un objeto de la clase Point de MonoGame (la cual hace referencia a una coordenada con su valor X e Y), o las propiedades X e Y, con las cuales se obtienen los valores concretos de posición del cursor respecto a los ejes de coordenadas X e Y. Por otro lado, para comprobar el estado de los botones del ratón se cuentan con propiedades concretas para cada botón disponible que devuelven un estado ButtonState de Monogame (presionado o no presionado), añadiendo además el control del scroll de la rueda o botón central del ratón.

La estructura GamePadState se podría dividir en tres comprobaciones principales: el estado de los elementos analógicos (triggers o gatillos y joysticks), el estado de los botones (de dirección y de acción) y el estado del mando en sí (conectado o no conectado). El estado de los elementos analógicos se comprueba mediante dos propiedades como son Triggers, que permite distinguir entre los distintos gatillos del mando y obtener sus valores analógicos, y ThumbSticks, que permite distinguir entre los dos joysticks con los que suele contar un mando y obtener sus valores analógicos. Por otro lado, el estado de los botones se comprueba mediante las propiedades DPad, la cual permite distinguir entre los botones direccionales del mando, y Buttons, la cual incluye a los distintos botones de acción del mando, con las cuales se obtienen valores de estado ButtonState. Por último, GamePadState permite comprobar si el mando esta encendido o no mediante la propiedad IsConnected, lo cual es bastante útil a la hora de incluir la modalidad multijugador en un videojuego.

3.2.2.2. Mostrar elementos gráficos por pantalla

El motor de videojuegos MonoGame cuenta con dos tipos de elementos gráficos básicos para mostrar por pantalla: sprites y textos. Ambos hacen uso para ser mostrados por pantalla del objeto SpriteBatch de la clase Game1.cs, mencionado anteriormente en el apartado “3.2.1. Game1.cs y el bucle de juego”.

Una cosa que tienen ambos elementos gráficos en común es la forma de comenzar y finalizar el dibujado, ya que, se vaya a mostrar uno, otro o ambos, se debe hacer uso de los métodos de SpriteBatch Begin (con alguna de sus sobrecargas) antes de empezar y End tras terminar. Otro aspecto en común sería la necesidad de pasar al método Draw o DrawString (métodos con distintas sobrecargas utilizados para mostrar los sprites y los textos respectivamente) como parámetro tanto una coordenada o área de destino dentro del sistema de coordenadas usado por MonoGame, el cual se muestra en la *Figura 25*, y un color representado por la clase Color de MonoGame, que servirá para aplicar un tinte al sprite o un color a la fuente del texto.

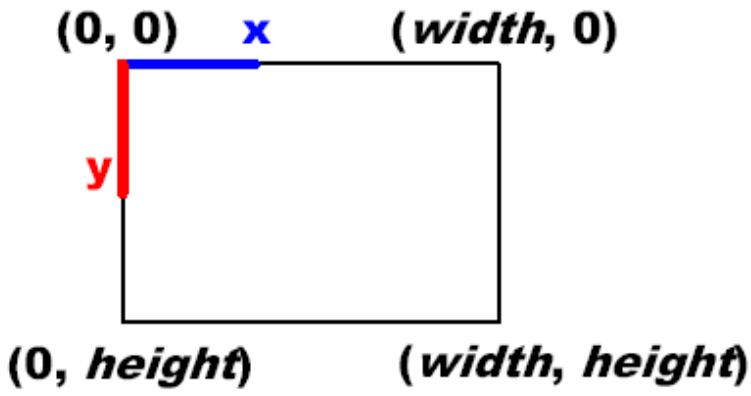


Figura 25. Sistema de coordenadas usado por MonoGame[73]

Una vez mencionados los aspectos en común entre el dibujado de sprites y textos, es hora de nombrar sus aspectos propios, empezando por los del método Draw y los sprites. El método Draw que permite el mostrado de sprites necesita mínimo, además de los dos parámetros mencionados en el párrafo anterior, que se le pase una imagen usando la clase Texture2D de MonoGame. Si hablamos de otras sobrecargas del método, se pueden utilizar más parámetros como un área de origen dentro de la imagen para que solo se muestre parte de esta, una cierta rotación de la imagen, un origen para esta rotación, una cambio en la escala de la imagen respecto a la original, un efecto de volteo o una profundidad de plano para mostrar la imagen.

Por otro lado, el método DrawString que permite el mostrado por pantalla de cadenas de texto necesita de dos parámetros más además de los comunes con Draw. Estos se tratan de una cadena de texto y un objeto de la clase SpriteFont de MonoGame, el cual describe las características de la fuente a utilizar para la cadena de texto. Además, mencionar que este método cuenta con unas sobrecargas similares a las de Draw, añadiendo parámetros como rotación, escala, volteo o profundidad.

Finalmente, es importante introducir en cierta manera una característica del mostrado por pantalla muy relacionada con la profundidad de plano mencionada anteriormente. El objetivo principal de esta profundidad es decidir en capa o plano serán mostrados las imágenes o cadenas de texto, permitiendo que unas se coloquen encima de otras. Para poder hacer uso de esta profundidad es necesario utilizar sobrecargas del método Begin de SpriteBatch que permiten el uso de un parámetro de la clase SpriteSortMode. Esta clase se trata de una enumeración que cuenta con distintas posibilidades en cuanto al tema de la profundidad, siendo estas las siguientes:

- **Deferred:** esta opción hace que las llamadas al método Draw o al método DrawString no se realicen hasta que no se llame al método End de SpriteBatch, tras lo cual se van ejecutando en orden de llamada sin tener en cuenta la profundidad que se le pueda haber indicado.
- **Immediate:** esta opción permite que el SpriteBatch vaya dibujando las distintos sprites o textos en el momento en el que se realizan las llamadas a Draw o DrawString, todo ello sin tener en cuenta la profundidad que se le pueda haber indicado.
- **BackToFront:** al hacer uso de esta opción se obliga al SpriteBatch a realizar un proceso similar al de la opción Deferred, pero ordenando las llamadas en orden ascendente en función de la profundidad que se haya podido indicar.
- **FrontToBack:** con esta opción se consigue el mismo efecto que con BackToFront, pero cambiando el orden ascendente por uno descendente.
- **Texture:** esta última opción realiza un proceso similar a Deferred ordenando las llamadas en función de la imagen a la que pertenezcan, ya que una imagen puede contar con distintas llamadas para el mostrado de sprites.

3.2.2.3. Reproducción de sonidos background y efectos de sonido

En cuanto al aspecto sonoro del videojuego, MonoGame utiliza dos opciones a la hora de renderizar las diferentes pistas de audio o sonidos. La primera de ella es mediante objetos de la clase SoundEffect, los cuales se obtienen a partir de un buffer de bytes que contiene la información del sonido y a los cuales se aplican tanto una frecuencia en hercios como un tipo de canal (mono o estéreo). Estos objetos a su vez cuentan con una limitación en cuanto a funciones que se pueden aplicar al sonido, por lo cual se generan instancias de la clase SoundEffectInstance que amplían más las opciones de reproducción. La segunda opción se trata del uso objetos de la clase Cue, siendo estos obtenidos a partir de unos archivos generados

por el programa Microsoft Cross-Platform Audio Creation Tool. En general, ambas opciones ofrecen a la larga las mismas opciones de reproducción como puedan ser el activar, pausar, utilizarla en bucle o la comprobación del estado de un sonido concreto, por lo que son dos opciones validas a la hora de gestionar el sonido de un videojuego.

Mencionar que en este proyecto, el autor se ha decantado por la utilización de los objetos Cue debido a su conocimiento previo sobre el programa Microsoft Cross-Platform Audio Creation Tool, por lo que además se considera interesante realizar una explicación breve de su funcionamiento. [74]Este programa, conocido también por su abreviatura XACT, se trata de una herramienta de audio-autoría gráfica distribuido por Microsoft como parte del SDK de DirectX, para posteriormente formar parte del entorno del motor de videojuegos XNA. Su función principal es la de permitir al usuario gestionar grandes cantidades de archivos sonoros, editar estos sonidos haciendo uso de su sencilla interfaz gráfica y utilizarlos mediante la API del framework utilizado, ya sea MonoGame o XNA. Todo esto lo realiza mediante tres archivos principales que genera al compilar el trabajo: un banco o almacén de pistas de audio en formato WAV, un banco o almacén de objetos Cue, las cuales se generan a partir de las pistas WAV mencionadas anteriormente y controlan todas las características del audio final, y un motor de audio que permite interpretar los otros dos archivos anteriores. El mayor problema quizás de esta herramienta es que Microsoft le dejó de dar soporte al mismo tiempo que lo hacía con el framework XNA, lo cual la dejó estancada en la versión 3 de la misma y sin vista de recibir actualizaciones en un futuro. Sin embargo, al permitir MonoGame su uso en las versiones más actualizadas, se ha convertido en una alternativa bastante potente a la hora de gestionar el aspecto sonoro al usar MonoGame y su framework de desarrollo.

3.2.2.4. MonoGame Pipeline

Finalmente, es importante mencionar una de las facilidades que proporciona MonoGame a la hora de compilar el contenido artístico del videojuego, MonoGame Pipeline. Esta herramienta se trata de una actualización del Content Pipeline de XNA, el cual permite la compilación de manera única, efectiva y veloz de todo el contenido multimedia del videojuego necesario. Esta compilación se basa en dos pasos principales: la importación del archivo concreto y su procesamiento al formato .xnb (XNA binary). Haciendo uso de este procedimiento, se consigue que la carga del contenido multimedia se realice en una única lectura y, por lo tanto, se evita tener que cargar cada elemento por separado, lo cual podría suponer que el jugador tenga que esperar una cantidad de tiempo suficientemente grande como para empeorar el gameplay del juego.

Sin embargo, en este proyecto se ha optado por no hacer uso de esta herramienta, a excepción de para gestionar los archivos .spritefont que definen las fuentes usadas a la hora de mostrar texto, y cargar los elementos en sus formatos originales. Esta decisión se debe a que, al ser un proyecto pequeño y con una carga de contenido multimedia entre comillas mínima, se ha optado por una solución alternativa como es la de tener una clase gestora de los Recursos, la cual se explicara posteriormente durante el apartado “3.4. Diseño e implementación”. En un futuro, como se destacará en el apartado “5. Conclusiones y trabajo futuro”, se podría contemplar el migrar todo el contenido multimedia a esta herramienta conforme aumente el volumen de este contenido.

3.3. Análisis

El concepto inicial que el autor planteó para su videojuego era el de plasmar un gameplay del estilo de los juegos de rol o RPG que siempre le han fascinado, como puedan ser *The Legend of Zelda*, *Pokemon* o *Kingdom Hearts*, pero añadiendo ciertos matices que lo hicieran distinto, como puede ser el hecho de realizar también todo el apartado artístico personalmente. Es por esta razón, que este concepto inicial fue demasiado ambicioso y más si se tenía en cuenta el poder completar el videojuego a nivel de trama narrativa y escenarios. En la *Figura 26*, se puede apreciar esta idea o concepto inicial, al cual habría que añadir todo el proceso artístico requerido para poder apreciar la ambición del proyecto.

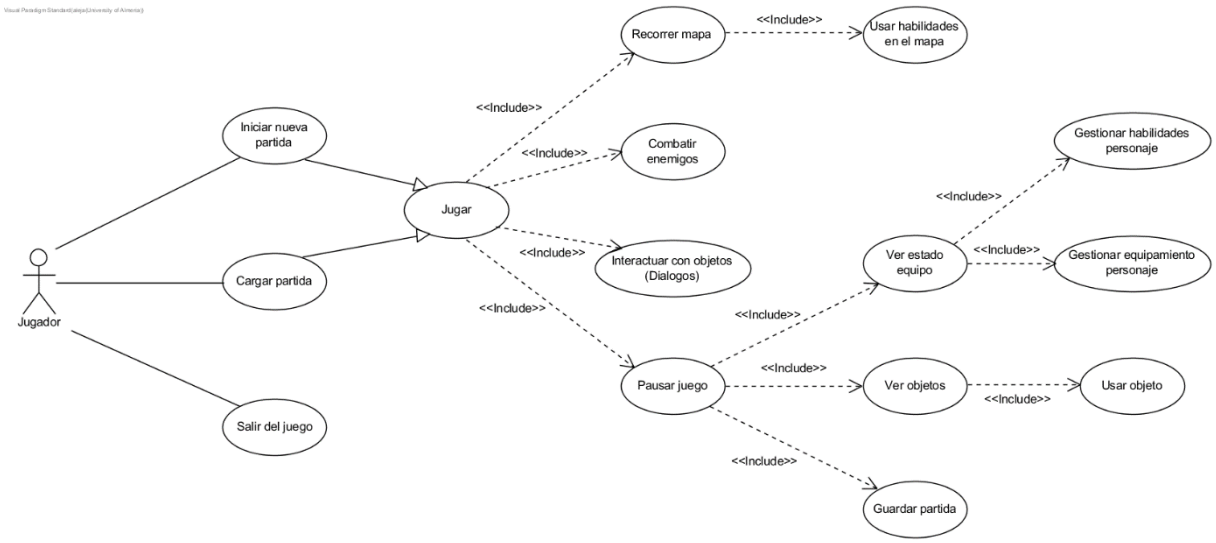


Figura 26. Diagrama de casos de uso inicial del proyecto

Una vez iba avanzando el proyecto y se acercó la fecha de la convocatoria de septiembre, el autor llegó a la conclusión de que el proyecto quizás se quedaba un poco grande para el tiempo que podía dedicar al Trabajo Fin de Grado. Es por esto por lo que se decidió a acotar el proyecto y centrarse solamente en la parte inicial de la trama narrativa y su correspondiente apartado artístico, con vistas de conseguir un videojuego más pequeño pero de mayor calidad para la convocatoria de diciembre. Por lo tanto, el concepto inicial del proyecto sufrió una serie de recortes, los cuales serían implementados a posteriori como trabajo futuro, y derivó en el representado por la *Figura 27*.

Figura 27. Diagrama de casos de uso final del proyecto

A continuación, se va a proceder a explicar más detalladamente en qué consiste el gameplay del videojuego, ya que el diagrama de la *Figura 27* solo se trata de una vista muy general de este. Para esta explicación se van a exponer cinco partes diferenciadas dentro del videojuego: el mapa, los combates, las interacciones/diálogos, los menús y los estados de transición.

3.3.1. Mapa

Recorrer los mapas es en gran medida la parte del gameplay que consumirá más tiempo para el jugador. Estos mapas estarían formados por distintas texturas y objetos, contando algunos de estos últimos la posibilidad de generar interacción con el jugador. En líneas generales, el mapa sirve como recorrido para que el jugador avance en la trama narrativa y, por lo tanto, en el videojuego, contando con las siguientes mecánicas:

- El jugador podrá desplazarse por el mapa desde un punto de vista cenital.
- El mapa contará con ciertas texturas y objetos atravesables que cumplirán la función de delimitar el espacio por el que se puede mover el jugador.
- El jugador podrá interaccionar con distintos objetos del mapa, generando diálogos o combates en función del tipo de estos objetos.
- Si el jugador colisiona con objetos como el fuego y similares recibirá una cantidad X de daño en su vida, en función del tipo del objeto.
- El jugador tendrá que resolver ciertos “puzles” para poder avanzar a zonas más avanzadas del mapa.

3.3.2. Combates

Los combates son una de las partes más interesantes del gameplay, ya que en ellos se pone a prueba las habilidades del jugador a la hora de realizar una estrategia y reaccionar ante problemas. En estos combates, el jugador deberá hacer uso de su equipo de personajes para hacer frente a un equipo de enemigos. En función del desarrollo del combate, el jugador podrá seguir avanzando por el mapa o perderá la partida. Las mecánicas de estos combates se podrían resumir en los siguientes puntos:

- El jugador podrá elegir entre lanzar un ataque normal, lanzar una habilidad con su coste de puntos de habilidad determinado, usar un objeto o huir en el turno de cada personaje del equipo.
- Cada personaje del equipo que este vivo deberá ser dirigido para realizar una acción antes de comenzar el lanzamiento de ataques entre personajes y enemigos.
- Una vez que el jugador decida qué hacer con cada personaje, se generarán mediante probabilidades los ataques de los distintos enemigos y se decidirá el orden de todos los ataques (tanto de personajes como de enemigos) en función de un atributo de velocidad de cada entidad de combate.
- Existirán varios tipos de ataques o movimientos: ataques normales, habilidades de daño, habilidades de daño múltiple, habilidades de daño con efecto, habilidades estadísticas y habilidades especiales.
- Para cada ataque o habilidad de daño se decidirá si es crítico o no, es decir, si interviene la defensa del receptor o no, en función de una probabilidad dada por el atributo de habilidad de la entidad de combate lanzadora.
- Si todos los enemigos son derrotados, los personajes vivos del equipo del jugador recibirán una cantidad de experiencia acorde a los enemigos derrotados, permitiendo así que los personajes puedan subir de nivel y mejorar sus estadísticas.

- Si todos los personajes del jugador son derrotados, el jugador habrá perdido la partida y se le llevará a una pantalla de game over.
- El jugador podrá huir de los combates, a excepción de cuando se enfrente a un jefe o subjefe enemigo en cuyo caso se le informará de que no puede huir.

3.3.3. Interacción/Diálogos

La interacción del jugador con los elementos del mapa en el género de rol o RPG es muy importante debido a la carga narrativa de estos, por lo que se ha tratado como una de las partes importantes dentro del gameplay de Dragon´s Curse. Esta interacción genera diálogos, además de otras posibilidades adicionales, que permiten guiar al jugador por la trama narrativa y cuenta con la siguiente mecánica:

- El jugador podrá avanzar por las distintas líneas de diálogo concretas de cada entidad interaccionable del mapa.
- Ciertas entidades otorgan al jugador de uno o varios objetos concretos que se añaden a su inventario.
- Ciertas entidades podrán generar combates a los que el jugador deberá enfrentarse.

3.3.4. Menús

El videojuego cuenta con una cantidad considerable de menús que permiten al jugador realizar ciertas acciones propias del gameplay, las cuales pueden ser desde salir del juego hasta equipar una pieza de armadura a un personaje concreto. Los distintos menús incorporados finalmente en el juego han sido los siguientes:

- Un menú principal que permite al jugador empezar una nueva partida o salir del juego, a lo que se podría añadir en una ampliación futura la opción de cargar una partida como se contempló en el concepto inicial del proyecto.
- Un menú de pausa desde el cual se puede acceder a los menús de equipo y objetos, además de poder continuar la partida o regresar al menú principal mencionado anteriormente. En este menú se contemplaría también la opción de guardar partida, la cual se trata de un trabajo futuro de este proyecto.
- Un menú de equipo que permite realizar la gestión de los personajes del jugador, cuyas posibilidades serían comprobar el estado de la vida y los puntos de habilidad del personaje, visualizar el equipamiento portado por el personaje y cambiar dicho equipamiento.
- Un menú de objetos que ofrece la posibilidad de revisar el inventario de objetos del que dispone el jugador y hacer uso de estos si es posible.

3.3.5. Estados de transición

Las pantallas de transición son un aspecto importante en un videojuego, ya que aportan al jugador una sensación de continuidad al ser un paso intermedio entre los distintos estados principales del videojuego. En esta versión final del proyecto se ha optado por contar con dos estados de transición principales: la pantalla “Cargando”, la cual intenta transmitir al jugador la sensación de que se están cargando los distintos elementos de la partida, y la pantalla “Game Over”, la cual indica al jugador que ha perdido la partida y lo traslada al menú principal. Además de estos estados de transición, sería interesante añadir como trabajo futuro algunos más que otorguen al juego de mayor continuidad y eviten saltos bruscos entre estados.

3.4. Diseño e implementación

En cuanto al diseño e implementación del videojuego Dragon´s Curse, se ha seguido una metodología orientada a objetos y se han utilizado distintos patrones de diseño como State o Factory Method. A continuación, se va a llevar a cabo una explicación detallada de las distintas partes que componen el proyecto, para lo cual se hará uso de distintos diagramas UML que facilitarán la compresión de dichas partes.

3.4.1. Almacén de recursos

La primera parte a resaltar del proyecto sería la creación de un almacén de recursos, con el cual se accederá a los distintos contenidos artísticos del videojuego. Este elemento ha sido concebido con la idea de disminuir el número de veces que se cargan los elementos multimedia, que podría ascender a un número bastante elevado, a una única vez cuando se inicializa el videojuego. Por lo tanto, cada vez que se necesite cargar un elemento artístico se recurrirá a una instancia de este cargada en alguna de las estructuras que componen la clase Recursos, las cuales se pueden apreciar en la *Figura 28*.

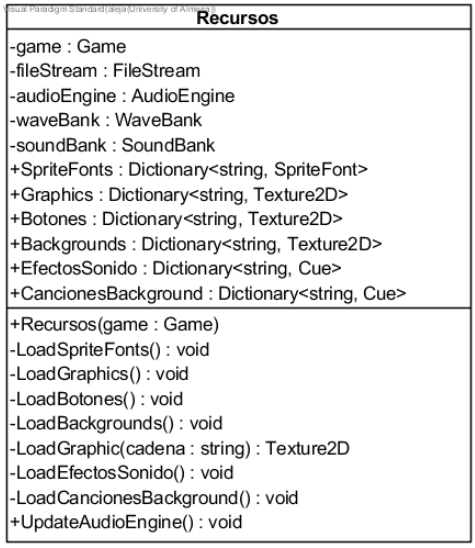


Figura 28. Diagrama de clases de la clase Recurso

Como se puede observar en la *Figura 28*, además de las estructuras para los distintos elementos artísticos, se cuenta con métodos que implementan la carga de estos elementos ya sea a través de los archivos generados por la herramienta XACT o a través de una variable del tipo FileStream. Estos métodos mencionados serían llamados durante el constructor de la clase Recursos, la cual se inicializa en el método LoadContent de DragonsCurse (originalmente nombrada como Game1).

Por último, es destacable mencionar la presencia del método UpdateAudioEngine, el cual permite el refresco interno de las pistas y sonidos almacenados en el motor de sonido generado por XACT.

3.4.2. Gestor de estados del videojuego

Uno de los aspectos más importantes en el desarrollo de un videojuego es la concepción de este como una sucesión de estados, los cuales implementan las distintas partes del gameplay. Estos estados cuentan cada uno con su implementación y lógica propias, las cuales deben ser ejecutadas desde los métodos de DragonsCurse heredados de la clase Game de MonoGame. Sin embargo, si todos estos estados fueran implementados en la clase DragonsCurse, esta clase presentaría síntomas del code smell “Clase

extensa"[75] y, por lo tanto, sería necesaria una solución a este problema. Dicha solución se trata de aislar el código de cada estado en clases específicas para cada uno de ellos y contar con un gestor de estados del videojuego, el cual sea llamado desde la clase DragonsCurse y se encargue de decidir qué estado es el que debe de ejecutarse. La implementación de dicha solución se ve reflejada, a continuación, en la *Figura 29*.

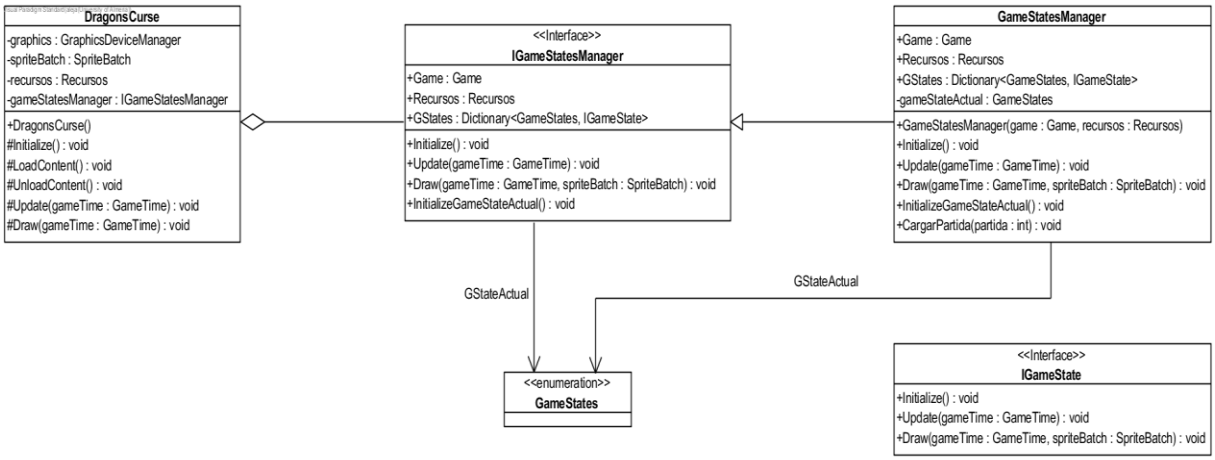


Figura 29. Diagrama de clases del gestor de estados del videojuego

Como se puede apreciar en el diagrama de clases de la *Figura 29*, la clase DragonsCurse contendría un objeto que implemente la interfaz IGameStatesManager, el cual se encargará de gestionar los distintos estados del videojuego como se ha mencionado anteriormente.

La interfaz IGameStatesManager obliga a las clases que la implementan a que cuenten con una serie de variables necesarias para poder gestionar el código de los distintos estados del videojuego: una referencia a un objeto Game para acceder a distintas propiedades del videojuego, como el tamaño de la pantalla; una referencia a un objeto de la clase Recursos, en el cual se almacena todo el apartado multimedia del videojuego; un diccionario que permita almacenar los distintos estados del videojuego asociando cada elemento de la enumeración GameStates a un objeto que implemente la interfaz IGameState, la cual implementan las clases correspondientes a los estados del videojuego; y una propiedad que permita conocer cuál es el estado actual de entre todos los de la enumeración GameStates. Además de estas variables, esta interfaz fuerza a sus clases hijas a implementar cuatro métodos que gestionan los distintos estados: el método Initialize, el cual permite reiniciar el diccionario de estados sin necesidad de eliminar el objeto en la clase DragonsCurse; el método Update, con el cual se ejecuta a su vez el método Update del objeto IGameState actual; el método Draw, que realiza el mismo procedimiento que Update pero con el mostrado por pantalla; y el método InitializeGameStateActual, con el cual se lleva a cabo la inicialización del pertinente estado actual del gestor.

Por otro lado, se tendría la clase GameStatesManager, la cual realiza la implementación de la interfaz IGameStatesManager y es considerada como el gestor de estados del videojuego principal, es decir, el que contiene la clase DragonsCurse. En este gestor de estados, el cual no es el único dentro del videojuego, se cargan en el método Initialize los principales estados del videojuego (InicioJuego, MenuPrincipal, Cargando y Juego) y se asigna a la variable GStateActual el estado inicial del juego (InicioJuego). Comentar, que en GameStatesManager cada vez que se le asigna un estado a la variable GStateActual, este estado se guarda en otra variable privada y se llama al método InitializeGameStateActual en el cual se inicializa dicho estado. Es importante también mencionar, que los métodos Update y Draw siguen una secuencia, la cual es mostrada en las figuras *Figura 30* y *Figura*

31, en la cual se realizan llamadas a los métodos del mismo nombre desde la clase DragonsCurse hasta el objeto IGameState actual.

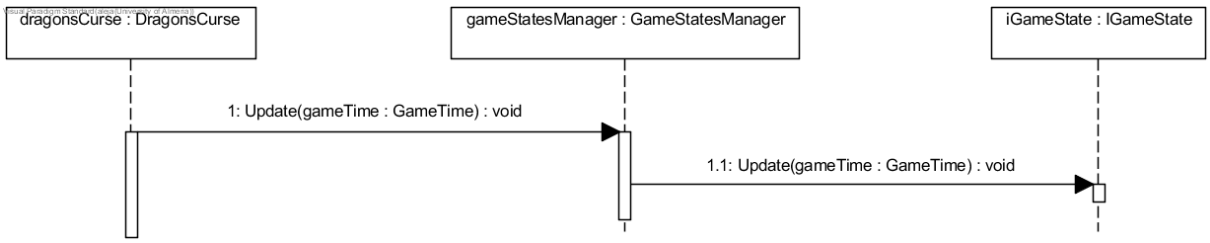


Figura 30. Diagrama de secuencias del proceso llevado a cabo para la actualización del videojuego

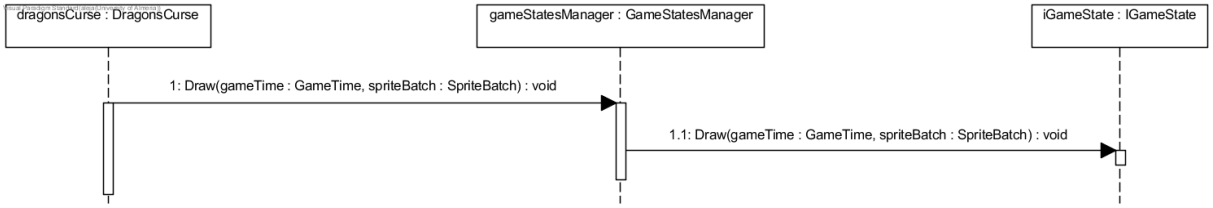


Figura 31. Diagrama de secuencias del proceso llevado a cabo para el mostrado por pantalla del videojuego

Para finalizar con el gestor de estados de videojuegos, es importante mencionar que para el diseño e implementación de este se ha utilizado el patrón de diseño State. Este patrón permite a un objeto alterar su comportamiento en función de su estado interno[6][77]. Si se observa la *Figura 32*, se pueden apreciar correspondencias con el diagrama de clases de la *Figura 29*: la clase Context se vería reflejada en la clase GameStatesManager, la clase o interfaz State se vería reflejada en IGameState y la relación de agregación entre Context y State se traduciría al diccionario de GameStates asociados a IGameStates que se encuentra en GameStatesManager.

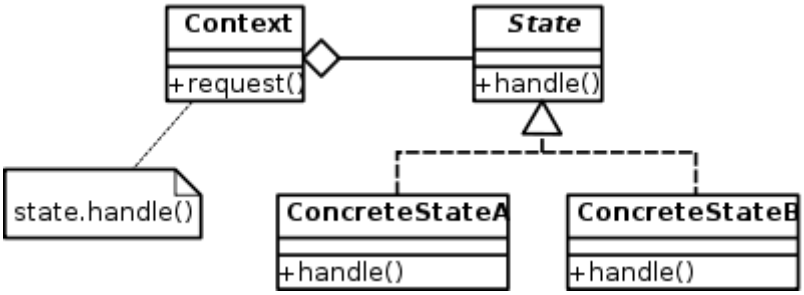


Figura 32. Diagrama de clases del patrón de diseño State[76]

3.4.3. Menús

Como se ha mencionado en el apartado "3.3. Análisis", los menús dentro de un videojuego son una parte muy diferenciada dentro de lo que es el conjunto del gameplay. Estos menús permiten al jugador realizar operaciones que versan desde salir del juego o comenzar una nueva partida hasta equipar un arma o usar un objeto sobre un personaje. Dentro de los menús de Dragon´s Curse, nos encontramos con dos tipos principales de menús: los menús estáticos, los cuales cuentan con una serie de botones entre los que el jugador puede navegar y seleccionar, y los menús de lista, en los que se muestra al jugador una serie de ítems en una lista por la que puede navegar y seleccionar uno de estos ítems. En cierta manera, estos dos tipos de menús parecen muy similares, pero el uso de la palabra "lista" en el segundo de los tipos quiere reflejar la posibilidad de que en estos menús pueden existir ciertos ítems que no se puedan mostrar en pantalla (por razones de espacio normalmente), lo cual se consigue desplazándose el jugador por la lista.

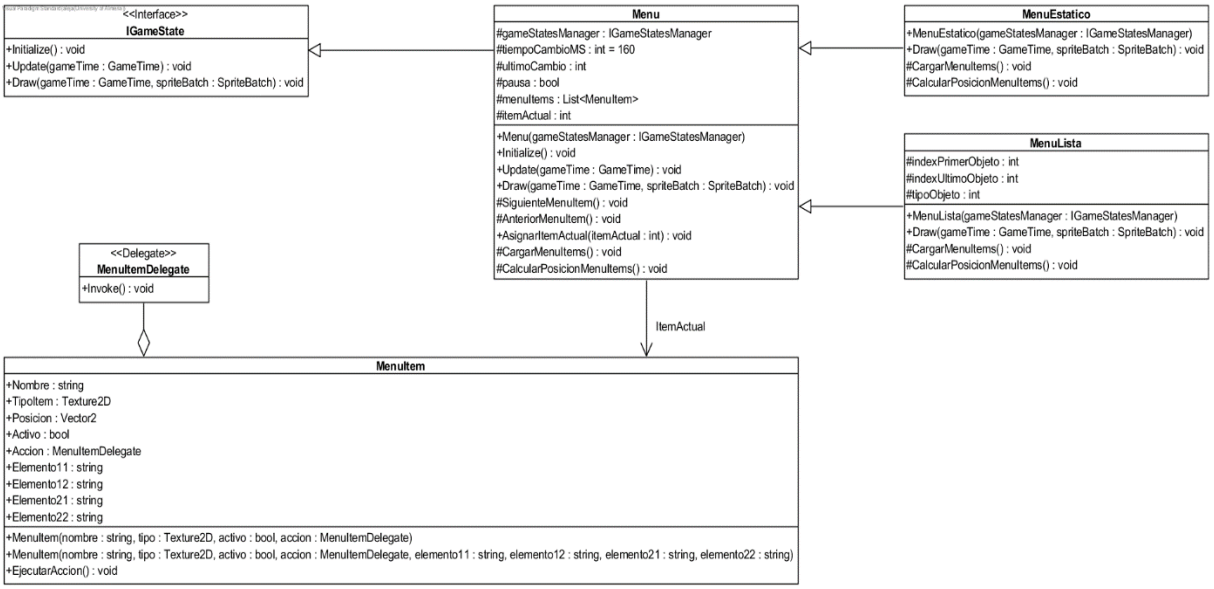


Figura 33. Diagrama de clases de la implementación de los menús

Observando la *Figura 33*, la cual representa el diagrama de clases de la implementación de estos menús, se puede apreciar como los menús serían considerados como un estado del videojuego, heredando por lo tanto de la interfaz IGameState y siendo gestionados por un IGameStatesManager. Además, los menús cuentan con una clase abstracta Menu que contiene las características generales de todos los menús del juego, incluyendo algunos métodos cuya implementación delega en los tipos concretos de menú representados por sus clases hijas. Las variables con las cuenta la clase Menu serían: una referencia al gestor de estados correspondiente, mediante un objeto IGameStatesManager; ciertas variables que permiten tener un contador con el que regular el tiempo entre cada interacción del jugador; una lista de objetos del tipo MenuItem, la cual representa el conjunto de botones con los que cuenta dicho menú; y dos variables, un objeto del tipo MenuItem y un entero que indica su posición en la lista, para poder acceder a las propiedades y a la función asignadas al botón actual. Por otro lado, los métodos que presenta la clase Menu son los heredados de la interfaz IGameState y otros que permiten tanto el proceso de inicialización de los distintos ítems del menú como el desplazamiento a través de ellos. Destacar que los métodos cuya implementación delega la clase Menu a sus clases hijas serían los siguientes: Draw, que implementa el mostrado por pantalla del menú; CargarMenuItems, que crea los distintos MenuItem en función del menú; y CalcularPosicionMenuItems, con el cual se le asigna una posición en pantalla a los distintos MenuItem.

Antes de exponer el proceso llevado a cabo cuando se inicializa un objeto del tipo Menu, es interesante describir brevemente la clase MenuItem, la cual se ha mencionado en varias ocasiones durante el párrafo anterior pero no se ha explicado. Esta clase permite representar como objetos independientes los distintos ítems del menú, permitiendo acceder de esta manera a sus propiedades internas. La propiedad principal de estos objetos sería la que representa a la acción que ejecutan dichos ítems dentro del menú, para lo cual se ha hecho uso de un delegado MenuItemDelegate que llame al método que se le asigna en el constructor. Otras propiedades incluidas en MenuItem serían las siguientes: un nombre, el cual representa la acción que ejecuta el ítem; una serie de valores representativos del ítem, los cuales se pasan como cadenas de caracteres para poder ser reflejados en el ítem o no; una textura para indicar el tipo del ítem y si esta seleccionado o no; una posición definida por una variable del tipo Vector2 de MonoGame, la cual se asigna durante el método CalcularPosicionMenuItems de la clase Menu; y un elemento booleano que permite identificar si el ítem puede ser seleccionado o no.

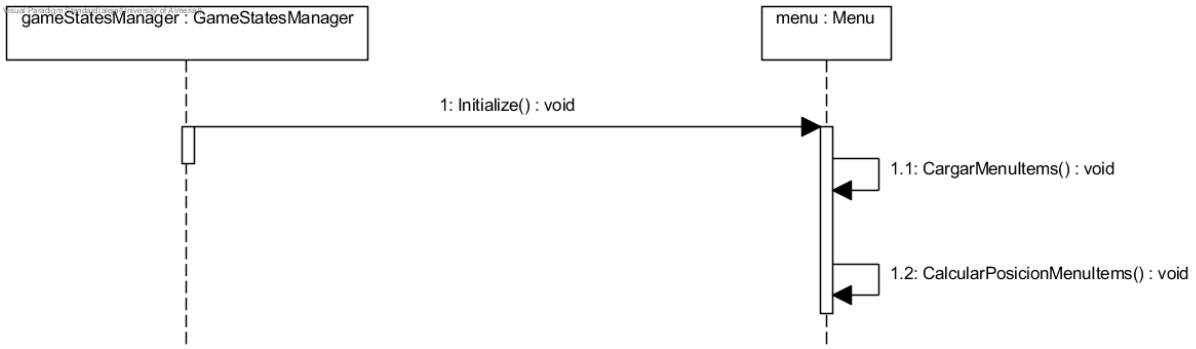


Figura 34. Diagrama de secuencias del proceso llevado a cabo al inicializar un menú

En la *Figura 34*, se puede apreciar cual es el proceso de inicialización de un objeto de la clase Menu, en el cual se lleva a cabo la carga de los distintos MenuItems y el cálculo de su posición en la pantalla. Este proceso cuenta con una implementación distinta para cada menú, ya que, como se ha comentado anteriormente, los métodos CargarMenuItems y CalcularPosicionMenuItems son abstractos en la clase Menu y se delega su implementación a las distintas clases hijas.

Para finalizar con esta sección dedicada a los menús, si se realiza cierto estudio sobre los diagramas presentados en la *Figura 33* y la *Figura 34* se podrá apreciar que, a la hora de diseñar e implementar los menús, se ha hecho uso del patrón de diseño Template Method (representado en la *Figura 35*). Este patrón de diseño permite a una clase abstracta delegar la implementación de ciertas etapas de un método a sus subclases[6][79]. En este caso, y como se ha mencionado varias veces anteriormente, la clase abstracta Menu delega partes de su inicialización (CargarMenuItems y CalcularPosicionMenuItems) a sus clases hijas, las cuales están divididas en menús que heredan de MenuEstatico (represemtados en la *Figura 36*) y menús que heredan de MenuLista (representados en la *Figura 46*).

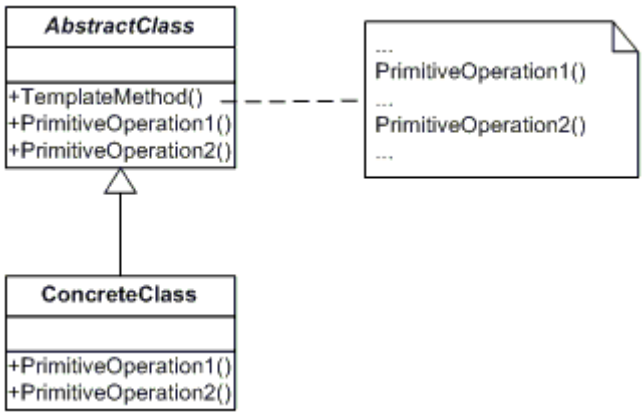


Figura 35. Diagrama de clases del patrón de diseño Template Method[78]

3.4.3.1. Menús estáticos

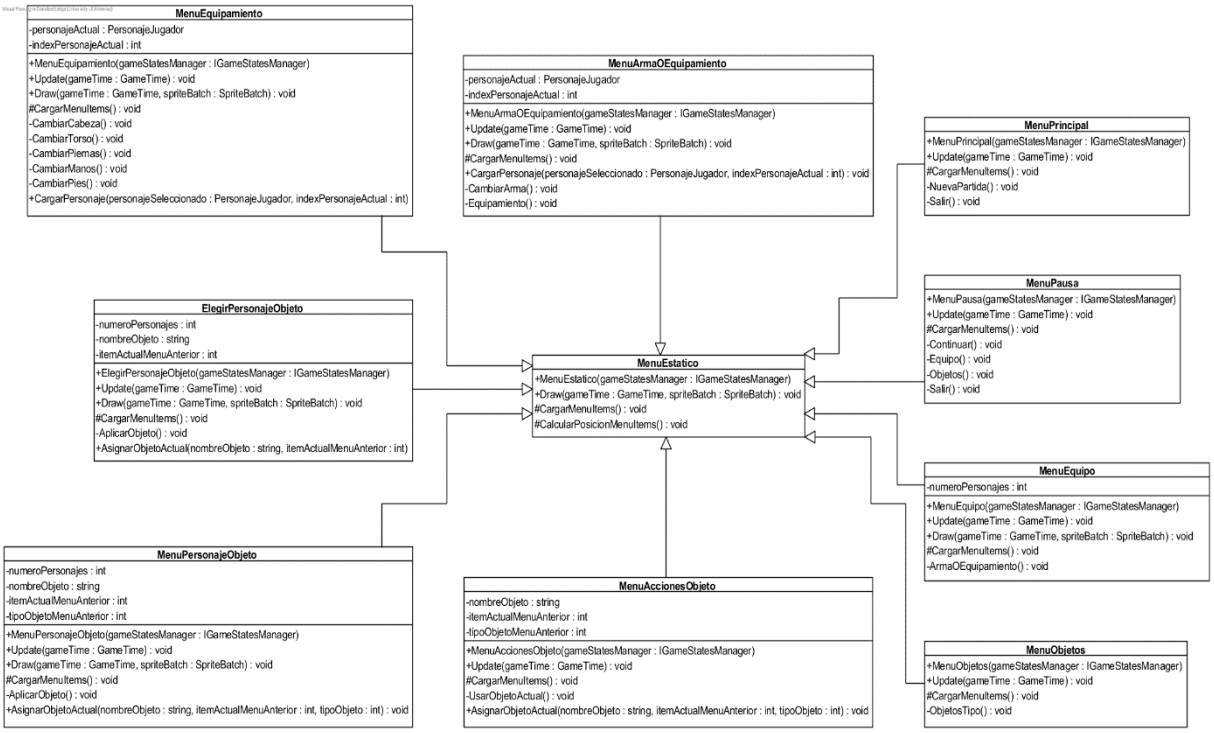


Figura 36. Diagrama de clases de los menús estáticos

- **MenuPrincipal:** Se trata del menú inicial del videojuego, al cual se accede desde el estado de transición InicioJuego, desde el menú MenuPausa perteneciente al estado Juego y desde la pantalla de transición GameOver. Además, este menú permite al jugador seleccionar una opción entre "Nueva Partida", que le llevará a la pantalla de transición Cargando para posteriormente llegar al estado Juego, y "Salir", opción que finalizará la ejecución del videojuego. Todo esto representado en el diagrama de estados de la *Figura 37*.

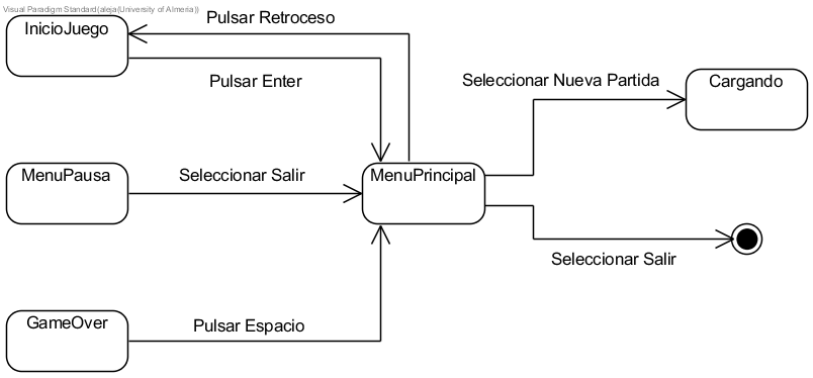


Figura 37. Diagrama de estados de MenuPrincipal

- **MenuPausa:** Este menú permite al jugador poner en pausa el juego, una vez se encuentra en el estado Mapa, para poder acceder mediante los distintos botones a MenuEquipo, MenuObjetos, MenuPrincipal o al estado Mapa de nuevo. Cada una de estas posibilidades cuenta con su camino inverso hacia MenuPausa a excepción de la opción que lleva a MenuPrincipal, como se puede apreciar en la *Figura 38*.

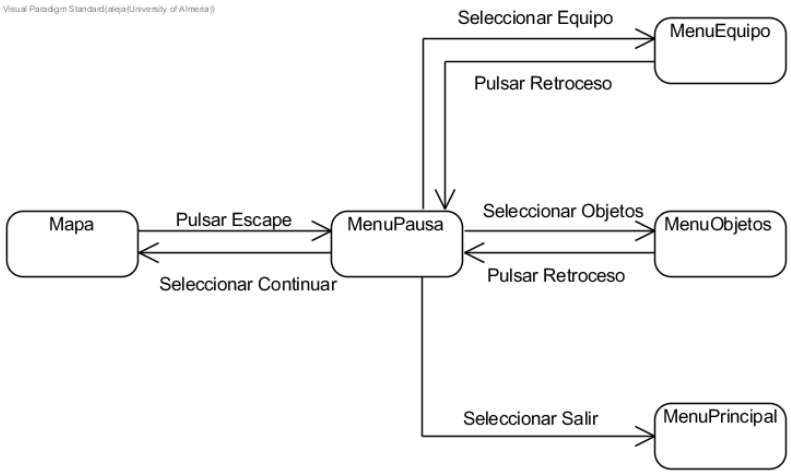


Figura 38. Diagrama de estados de MenuPausa

- **MenuEquipo:** En este menú, el jugador podrá consultar el estado de los personajes de su equipo y acceder a la gestión del equipamiento de estos en MenuArmaOEquipamiento. Además, desde este menú se puede volver hacia MenuPausa, para lo cual bastará con pulsar la tecla Retroceso. Todo esto representado en el diagrama de estados de la *Figura 39*.

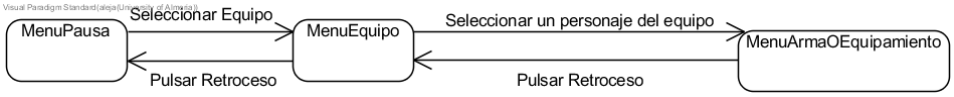


Figura 39. Diagrama de estados de MenuEquipo

- **MenuArmaOEquipamiento:** Como se ha mencionado en el punto anterior, este menú es accesible seleccionando un personaje en MenuEquipo. A su vez, permite la posibilidad al jugador de acceder a MenuEquipamientoConcreto, donde se podrá cambiar el arma del personaje, seleccionando la opción Arma o acceder a MenuEquipamiento, donde se podrán gestionar las distintas piezas de armadura del personaje, seleccionando la opción Equipamiento. Todo esto representado en el diagrama de estados de la *Figura 40*.

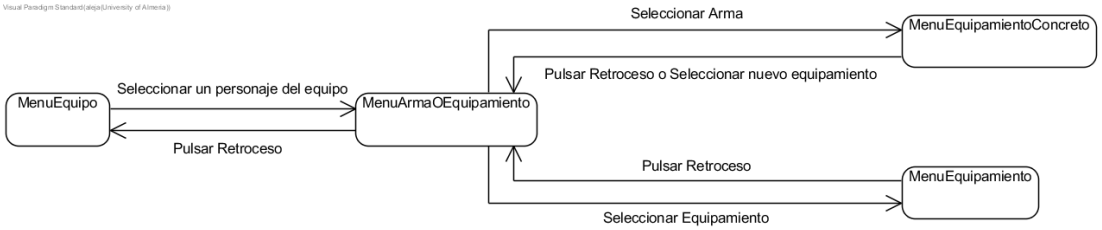


Figura 40. Diagrama de estados de MenuArmaOEquipamiento

- **MenuEquipamiento:** A través de este menú el jugador podrá visualizar las distintas partes de armadura que lleva el personaje, el cual fue seleccionado previamente en MenuEquipo. Si se decide cambiar alguna de estas partes, este menú permite también acceder a MenuEquipamientoConcreto seleccionando la opción de dicha parte de armadura. Todo esto representado en el diagrama de estados de la *Figura 41*.

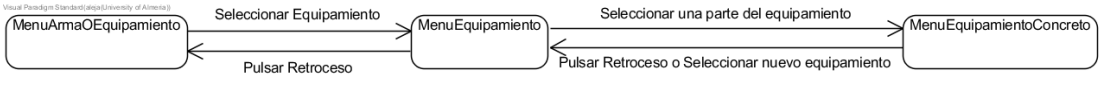


Figura 41. Diagrama de estados de MenuEquipamiento

- **MenuObjetos:** La funcionalidad que ofrece este menú, accesible desde MenuPausa, es bastante sencilla, ya que solo cuenta con la posibilidad de elegir qué tipo de objetos del inventario (Variedad, Combate o Especial) se quiere consultar. Una vez tomada esta decisión, el jugador podrá acceder a MenuObjetosTipo seleccionando el ítem del menú correspondiente al tipo decidido. Todo esto representado en el diagrama de estados de la *Figura 42*.

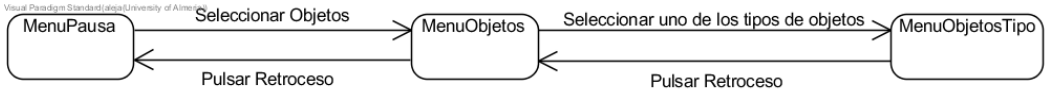


Figura 42. Diagrama de estados de MenuObjetos

- **MenuAccionesObjeto:** Este menú cuenta con una única función por el momento, ya que está planeado que como trabajo futuro se permita que el jugador realice más acciones sobre los distintos objetos. Esta funcionalidad consiste en acceder al MenuPersonajeObjeto para poder usar el objeto seleccionado en MenuObjetosTipo, al cual también se puede volver para seleccionar otro objeto distinto del actual. Todo esto representado en el diagrama de estados de la *Figura 43*.

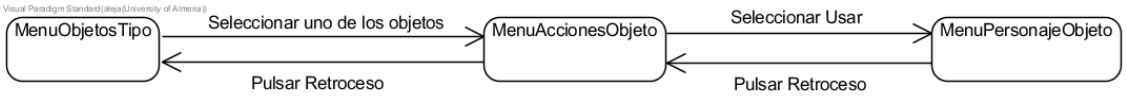


Figura 43. Diagrama de estados de MenuAccionesObjeto

- **MenuPersonajeObjeto:** Se trata de un menú que, siendo idéntico a MenuEquipo en cuanto a mostrado por pantalla, permite al jugador seleccionar sobre qué personaje decide usar el objeto seleccionado anteriormente en MenuObjetosTipo. Una vez seleccionado dicho personaje, el jugador volverá a MenuObjetosTipo, donde se habrá restado la cantidad del objeto usado. Todo esto representado en el diagrama de estados de la *Figura 44*.

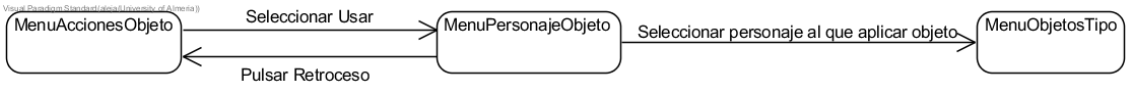


Figura 44. Diagrama de estados de MenuPersonajeObjeto

- **ElegirPersonajeObjeto:** Este menú perteneciente al estado Combate es idéntico en todos los sentidos a MenuPersonajeObjeto, siendo las únicas diferencias sus puntos de acceso (ElegirObjeto) y destino (InicioTurno o LanzamientoAtaques) como se puede apreciar al comparar la *Figura 45* con la *Figura 44*.

Figura 45. Diagrama de estados de ElegirPersonajeObjeto

3.4.3.2. Menús de lista

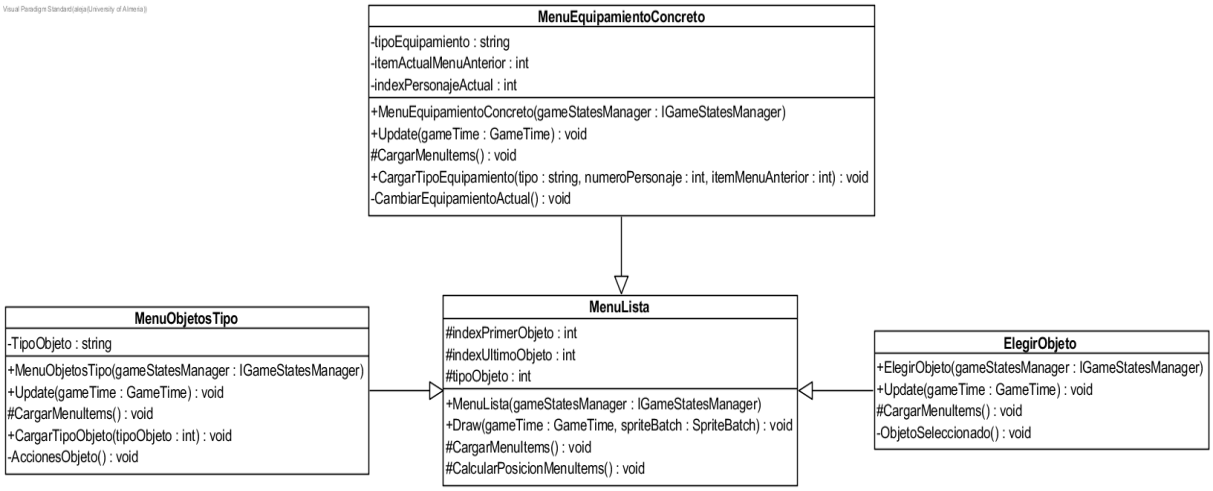


Figura 46. Diagrama de clases de los menús de lista

- **MenuEquipamientoConcreto:** Este menú permite al jugador seleccionar una pieza de equipamiento o arma de las disponibles en el inventario, la cual sustituirá a la que tenga equipada actualmente el personaje seleccionado en MenuEquipo. En función de si el punto de acceso a este menú ha sido MenuEquipamiento o MenuArmaEquipamiento, una vez que el jugador seleccione la nueva pieza de equipamiento o arma, o en su defecto haya pulsado la tecla Retroceder, se le devolverá al menú desde el que haya accedido a MenuEquipamientoConcreto. Todo esto representado en el diagrama de estados de la *Figura 47*.

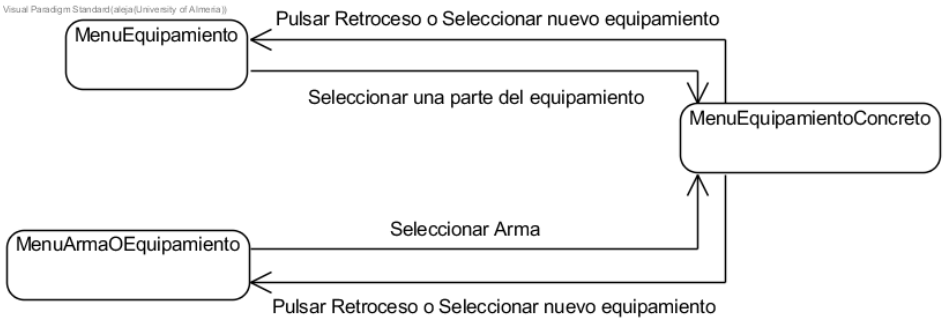


Figura 47. Diagrama de estados de MenuEquipamientoConcreto

- **MenuObjetosTipo:** Este menú varía su contenido en función del tipo de objeto seleccionado en MenuObjetos, permitiendo al jugador seleccionar uno de los objetos disponibles en el inventario de ese tipo. Una vez se ha seleccionado el objeto deseado, el jugador será trasladado a MenuAccionesObjeto, donde decidirá qué acción desea ejercer respecto al objeto. Todo esto representado en el diagrama de estados de la *Figura 48*.

Figura 48. Diagrama de estados de MenuObjetosTipo

- **ElegirObjeto:** Si se comparan la *Figura 48* y la *Figura 49*, se puede apreciar como ElegirObjeto cuenta con la misma funcionalidad que MenuObjetosTipo, con la diferencia de que se accede a este desde el estado de combate InicioTuno (seleccionando la opción OBJETOS) y que, debido a que en combate la única acción a realizar con un objeto es la de ser aplicado a un personaje, este menú accede directamente a otro menú que permite seleccionar el personaje objetivo, ElegirPersonajeObjeto. Todo esto representado en el diagrama de estados de la *Figura 49*.

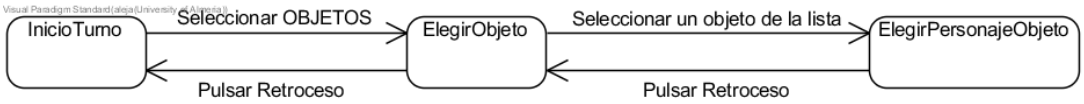


Figura 49. Diagrama de estados de ElegirObjeto

3.4.4. Transiciones

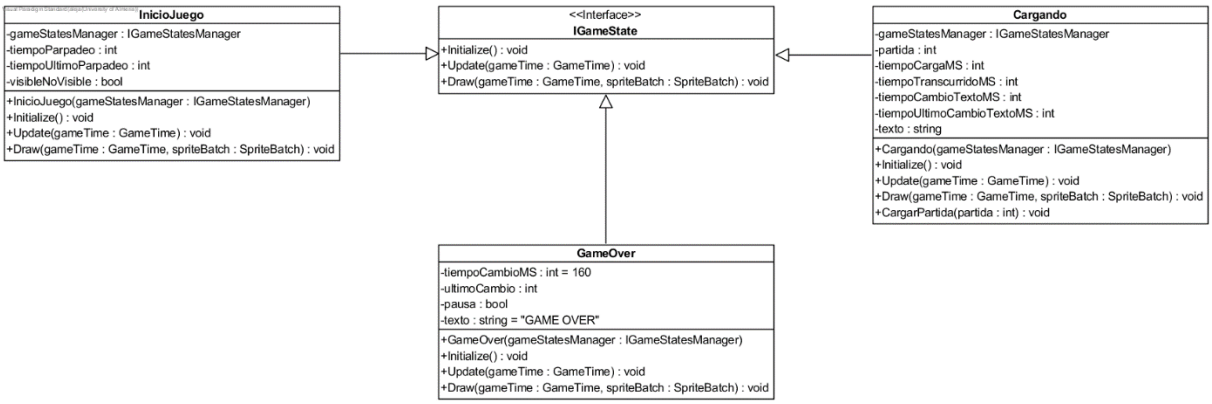


Figura 50. Diagrama de clases de los estados de transición

Las transiciones son, sin duda, la parte más simple dentro del videojuego. Su única función como tal es la de hacer que el jugador sienta una sensación de continuidad cuando va avanzando por los distintos estados del videojuego, lo cual consigue gracias al feedback que produce mostrar una pantalla intermedia entre el estado del que se procede y al que se va a pasar. Además, dentro de su implementación, la cual se trata de una herencia directa de la interfaz IGameState, se suele recurrir a distintas formas de pasar al estado siguiente, como esperar una determinada cantidad de tiempo o realizar alguna acción sobre el controlador usado.

Como se puede apreciar en el diagrama de la *Figura 50*, se puede observar que se han llegado a implementar tres estados de transición distintos para esta versión final:

- **InicioJuego:** es el primer estado que inicia el videojuego y en él se muestra al jugador una pantalla con el título de juego de fondo y el texto “Pulsa ENTER para comenzar” parpadeando. Como bien indica el texto que se muestra, si el jugador pulsa la tecla Enter podrá acceder al estado MenuPrincipal, desde el cual también se podrá volver a InicioJuego pulsando la tecla Retroceso. Todo esto representado en el diagrama de estados de la *Figura 51*.

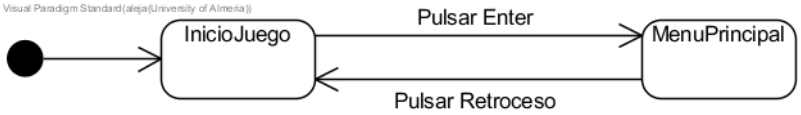


Figura 51. Diagrama de estados de InicioJuego

- **Cargando:** este es seguramente el estado de transición usado más frecuentemente en el videojuego, ya que sirve como paso desde el MenuPrincipal a lo que es el juego en sí, representado por el estado Juego. El contenido de Cargando consta de una pantalla en la que se muestra el texto "CARGANDO", al cual se le van añadiendo puntos suspensivos con el paso del tiempo. La salida de este estado hacia el estado Juego viene dada por la espera del jugador por un tiempo determinado, lo cual hace ver a este que se están cargando los datos necesarios para inicializar el siguiente estado. Todo esto representado en el diagrama de estados de la *Figura 52*.

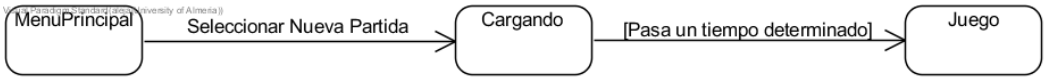


Figura 52. Diagrama de estados de Cargando

- **GameOver:** se trata del estado que indica al jugador que todos los personajes de su equipo han muerto, ya sea durante el estado Mapa o el estado Combate, y por lo tanto ha perdido la partida, lo cual se representa mostrando por pantalla el texto "GAME OVER". Una vez se le ha mandado al jugador esta información, se le permite acceder al estado MenuPrincipal para empezar una nueva partida o quitar el juego usando la tecla Espacio. Todo esto representado en el diagrama de estados de la *Figura 53*.

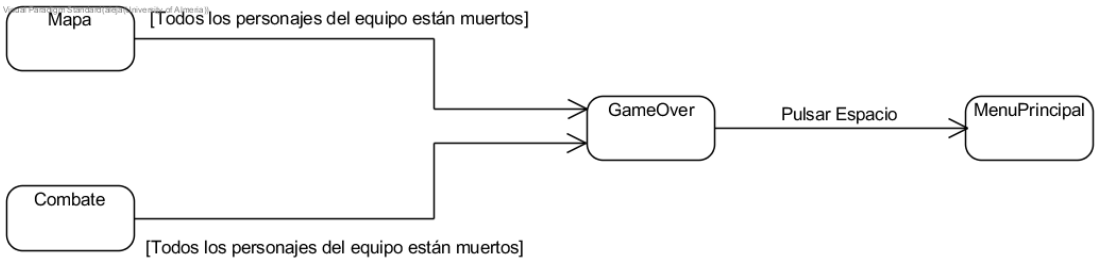


Figura 53. Diagrama de estados de GameOver

3.4.5. Juego

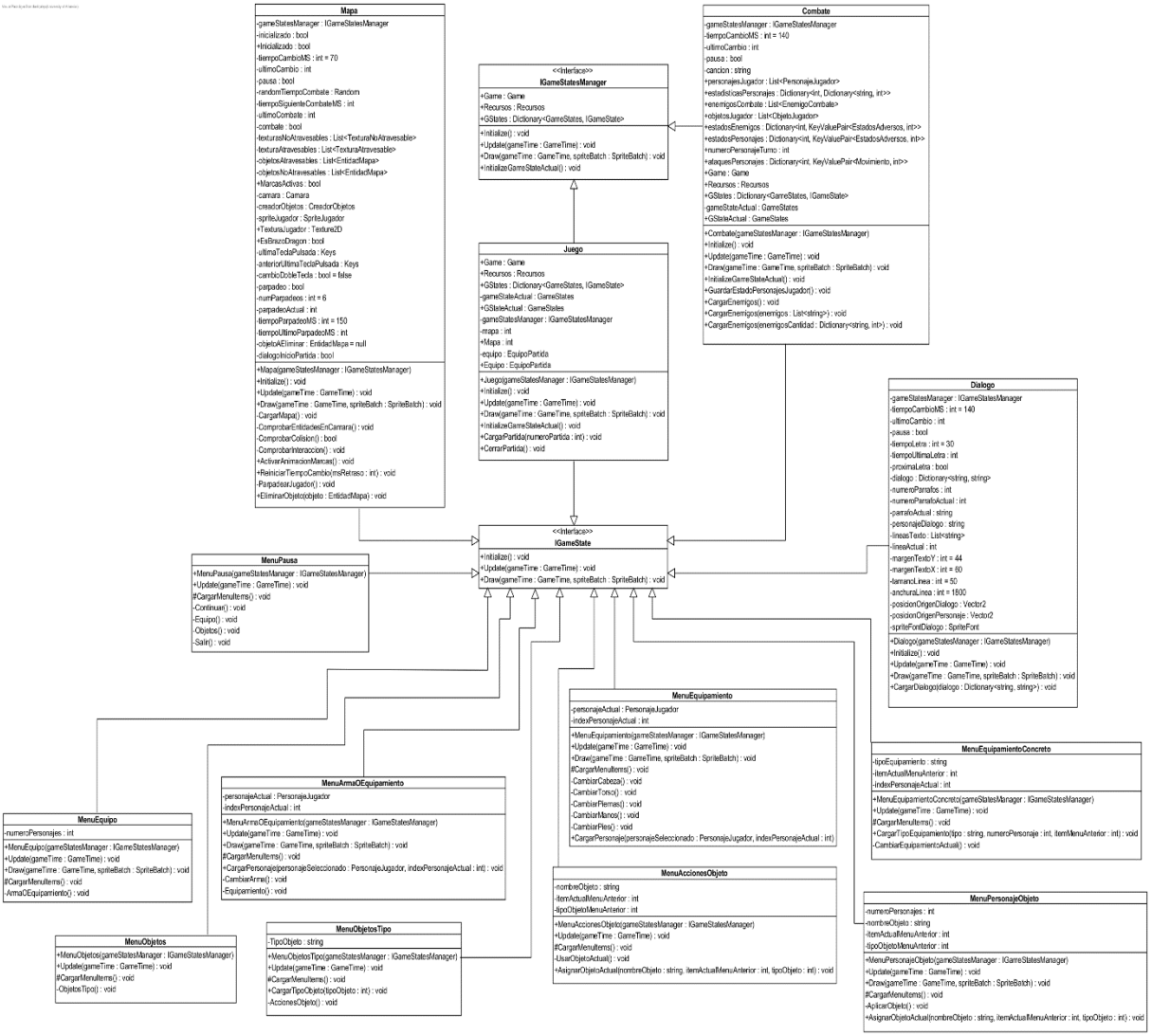


Figura 54. Diagrama de clases del estado Juego junto a sus estados

En contraposición a la simpleza de los estados de transición, el estado Juego es seguramente el más amplio de todos los estados del videojuego. Este estado contempla toda la implementación que es necesaria para, una vez iniciada una partida, poder disfrutar de todo el gameplay del videojuego. Debido a que esta implementación es demasiado grande como para llevarla a cabo en una única clase, se ha considerado al estado Juego como un gestor de estados propio dentro del gestor principal del videojuego, por lo que se tendría una nueva herencia de la interfaz IGameStatesManager como se aprecia en la *Figura 54*.

Sin embargo, este nuevo gestor de estados del videojuego cuenta con una diferencia interesante respecto al anterior. En el gestor principal, cada vez que se asignaba a la variable GStateActual un nuevo estado este se inicializaba como parte del proceso, pero en el gestor Juego esto no es siempre así. Juego hace una excepción para el estado Mapa, para el cual se comprueba si está inicializado y, en caso de que lo esté, no se lleva a cabo una nueva inicialización. Este aspecto se debe a que cuando se pasa del estado Mapa a cualquier otro estado no se debe “interrumpir” la ejecución sino que se debe pausar, ya que si cuando se vuelve al estado Mapa desde uno de los estados de Juego se realizara un proceso de inicialización se perdería todo el recorrido del jugador por el mapa y se le haría comenzar de nuevo.

En la *Figura 54* se puede observar como existen dos grupos diferenciados dentro de los estados de Juego. Por un lado, estarían los estados relacionados con menús, los cuales han sido explicados previamente en el apartado "3.4.3 Menús" y no se llevará a cabo una nueva explicación. Pero por otro lado, es posible agrupar tres estados de Juego que no se tratan de menús, siendo estos Mapa, Dialogo y Combate, los cuales están asociados entre sí como se aprecia en el diagrama de estados de la *Figura 55*. Estos tres estados serían a su vez, como sus propios nombres indican, lo estados asociados a tres de los aspectos del videojuego destacados durante el apartado "3.3. Análisis": el mapa, la interacción o diálogo y el combate. Es por esto, que a continuación, en los siguientes apartados, se va a desarrollar la explicación de la implementación de cada uno de estos estados y las distintas partes que lo componen, ya que estas merecen apartados individuales para ser desarrolladas.

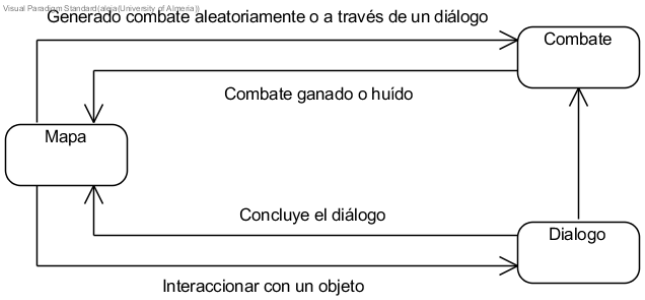


Figura 55. Diagrama de estados de Mapa, Combate y Dialogo

3.4.6. Mapa

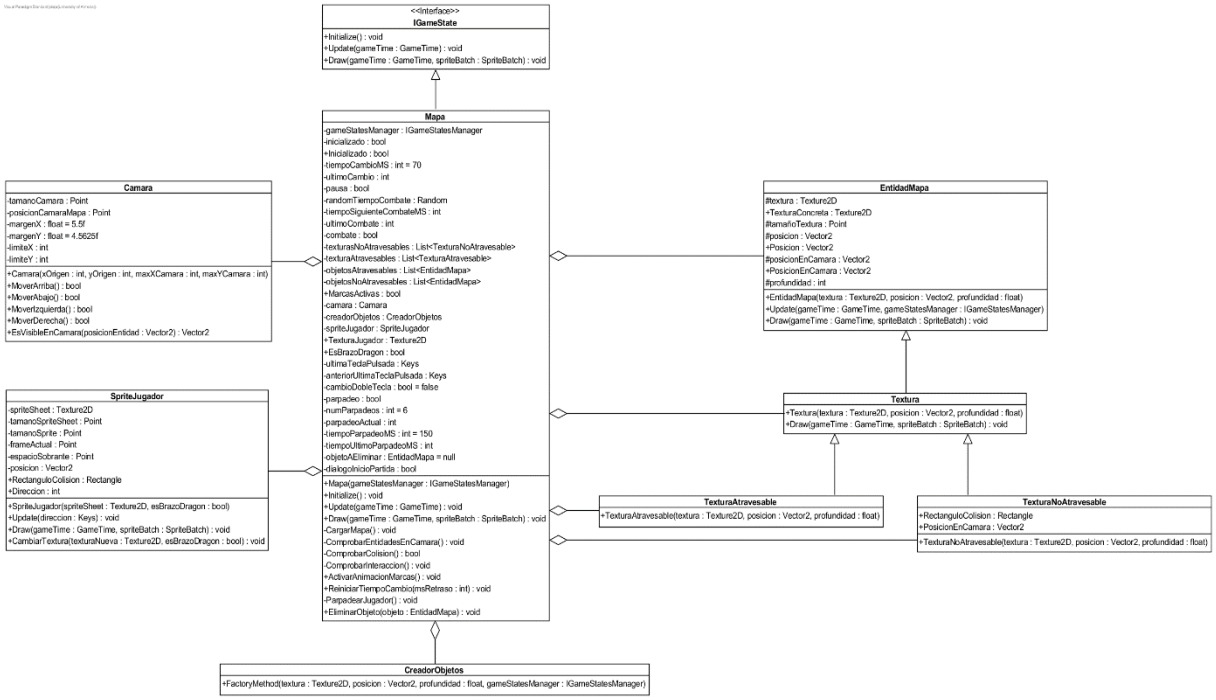


Figura 56. Diagrama de clases del estado Mapa

Durante el videojuego Dragon´s Curse, el jugador deberá recorrer unas serie de mapas, aunque en la versión actual solo existe uno, para poder avanzar por la trama narrativa, ir mejorando las estadísticas de su equipo subiendo niveles, obtener nuevas piezas de equipamiento o disfrutar del reto que puedan suponer los combates entre otros aspectos. Es por esto que el recorrido del mapa es una de los elementos principales del videojuego, contando con una implementación amplia que ofrezca variedad durante el transcurso de este proceso.

La clase Mapa, representada en la *Figura 56*, está compuesta por una serie de variables necesarias para implementar dicha variedad en su gameplay: una referencia al gestor de estados al que pertenece, mediante la cual se obtienen el contenido artístico del videojuego almacenado en Recursos y otros datos propios del videojuego; una serie de variables que permiten llevar contadores de tiempo para controlar la interacción del jugador, la generación de combates aleatorios o el parpadeo sobre el sprite del jugador cuando este sufre daño al colisionar con un objeto que provoque este daño; una lista de entidades del mapa por cada capa de este; un objeto del tipo SpriteJugador, el cual representa el sprite usado para representar al jugador; una cámara desplazable, la cual permite controlar que entidades del mapa se van a mostrar por pantalla; y un objeto CreadorObjetos sobre el que delegar la creación de los objetos del mapa. Además de estas variables o propiedades, este estado de Juego cuenta con una serie de métodos que permiten cargar las distintas capas del mapa desde archivos CSV en las listas mencionadas anteriormente (CargarMapa), comprobar que entidades se deben mostrar por pantalla y sus correspondientes posibles colisiones e interacciones (ComprobarEntidadesEnCamara, ComprobarColision y ComprobarInteracción), activar distintas funcionalidades del mapa concreto (ActivarAnimacionMarcas, ParpadearJugador y EliminarObjeto) o pausar durante un tiempo determinado el movimiento del jugador (ReiniciarTiempoCambio).

Para finalizar con el estado Mapa, se ha considerado interesante exponer, mediante la *Figura 57*, la *Figura 58* y la *Figura 59*, los procesos de inicialización, actualizado y mostrado por pantalla de este estado, ya que a través de esto se puede apreciar el uso de los distintos métodos mencionados anteriormente.

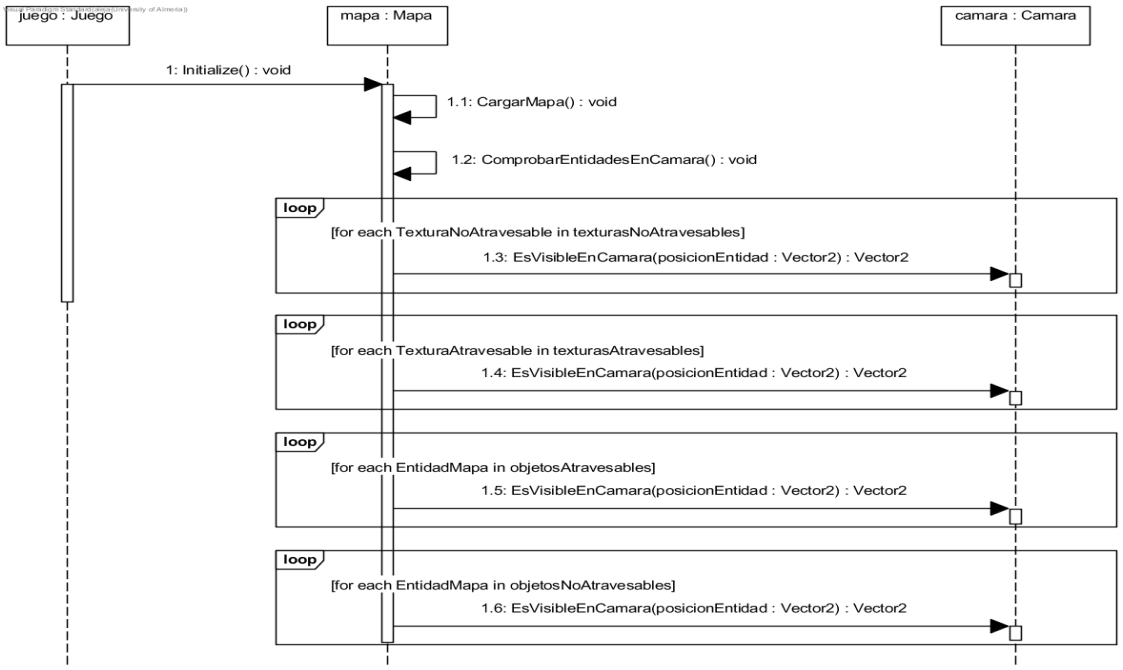


Figura 57. Diagrama de secuencias del proceso llevado a cabo al inicializar el estado Mapa

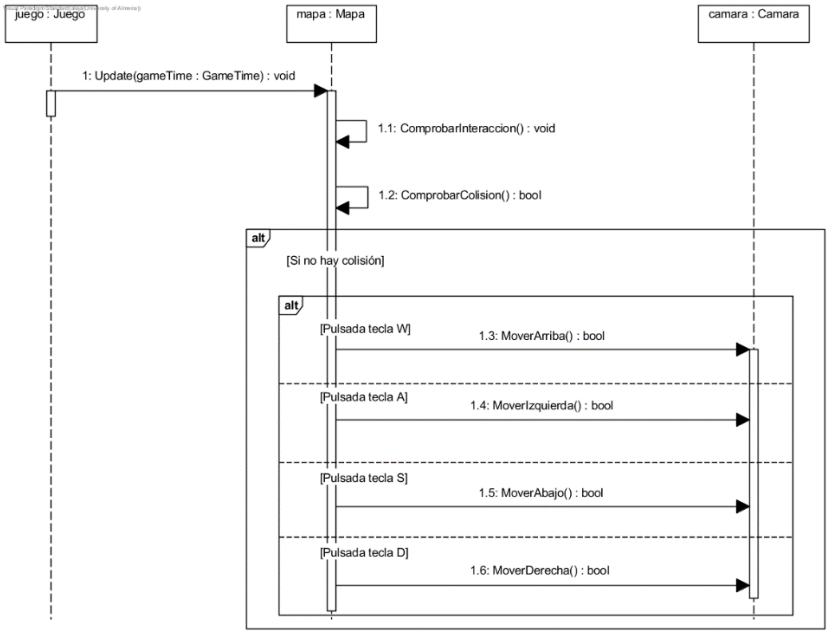


Figura 58. Diagrama de secuencias del proceso llevado a cabo al actualizar el estado Mapa

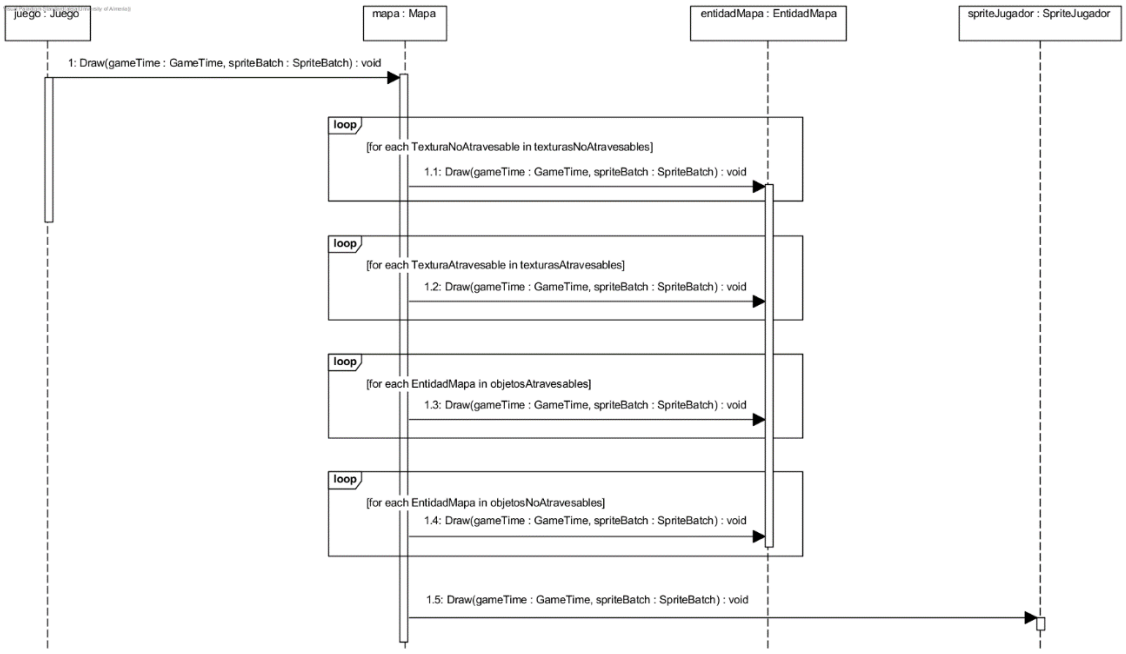


Figura 59. Diagrama de secuencias del proceso llevado a cabo al mostrar por pantalla el estado Mapa

3.4.7. Cámara

Como se ha mencionado en el aparta anterior, el estado Mapa necesita de un objeto del tipo Camara, el cual permita controlar qué entidades del mapa son visibles en pantalla, por lo que se deben mostrar, y cuáles no. Este objeto Camara (representado por la clase de la *Figura 60*) cuenta con una serie de atributos que hacen referencia al tamaño y la posición de la cámara, la cual es un poco más grande que el tamaño de pantalla dado por Game para cargar los objetos adyacentes. A su vez, esta clase cuenta con una serie de métodos que permiten el desplazamiento de la cámara por el mapa (MoverArriba, MoverAbajo, MoverIzquierda y MoverDerecha) y un método que realiza la comprobación de si una entidad del mapa se encuentra en el área de la cámara.

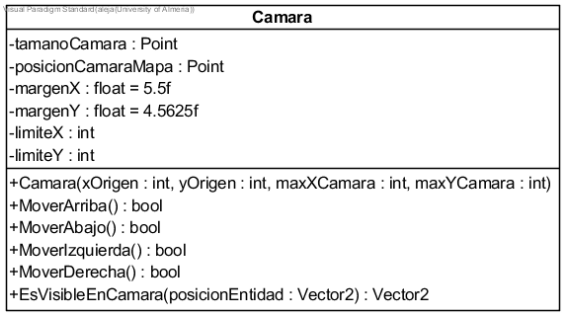


Figura 60. Clase Camara

3.4.8. Entidades del mapa

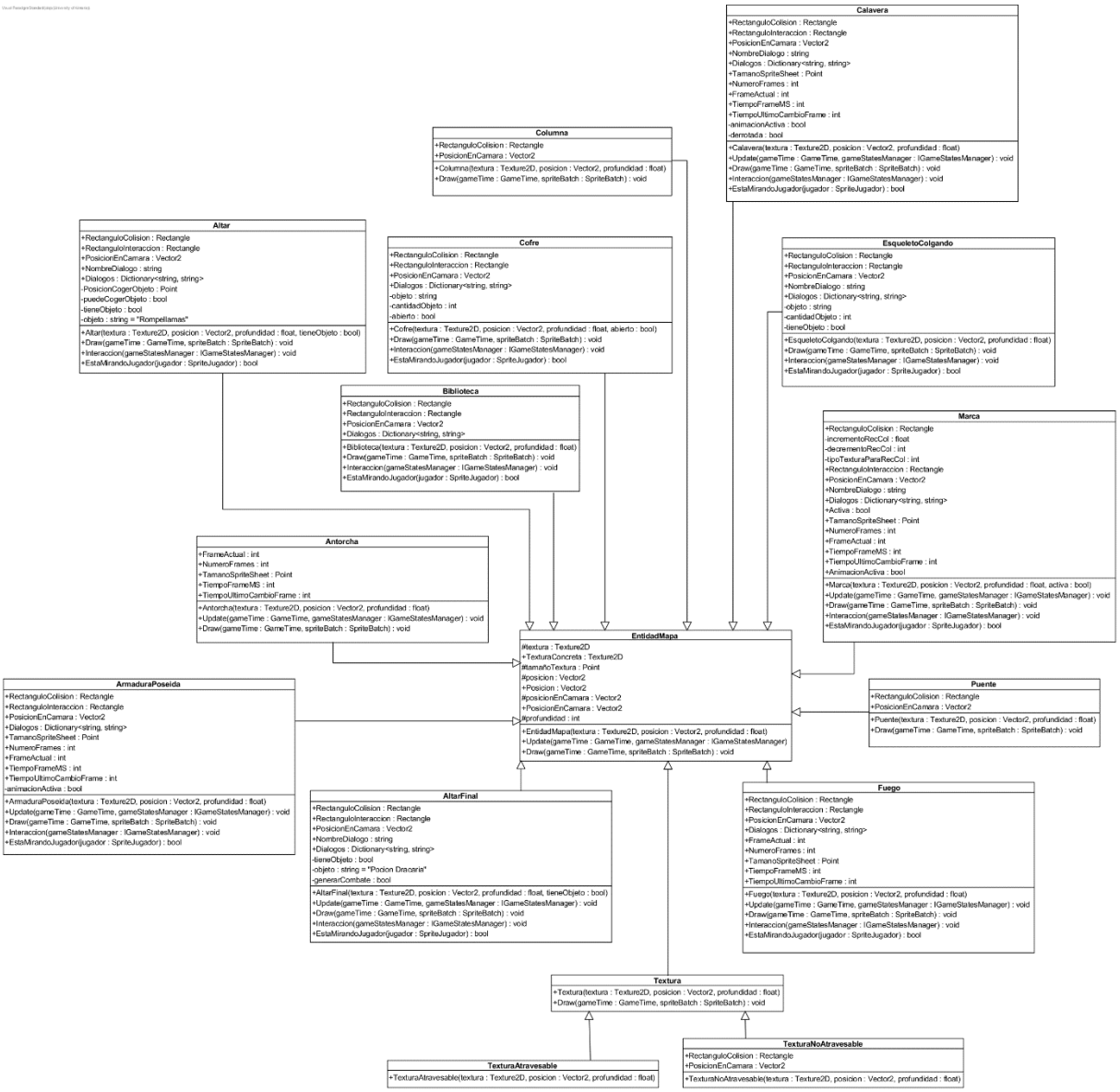


Figura 61. Diagrama de clases de las entidades del mapa

Cuando se habla del estado del videojuego Mapa, es imprescindible tener en mente que este está formado por distintas entidades que han de ser mostradas por pantalla y con las cual el jugador puede colisionar o interaccionar. Estas entidades han sido reflejadas como clases que heredan de la clase abstracta EntidadMapa, como se puede ver en la *Figura 61*, y que pueden implementar o no distintas interfaces

que se presentarán en subapartados posteriores. Además, es posible hacer una división general de estas entidades: las texturas, las cuales son clases que heredan de la clase abstracta Textura (TexturaAtravesable y TexturaNoAtravesable); y los objetos, los cuales son representados por clases individuales para cada uno.

3.4.8.1. IEntidadAnimada

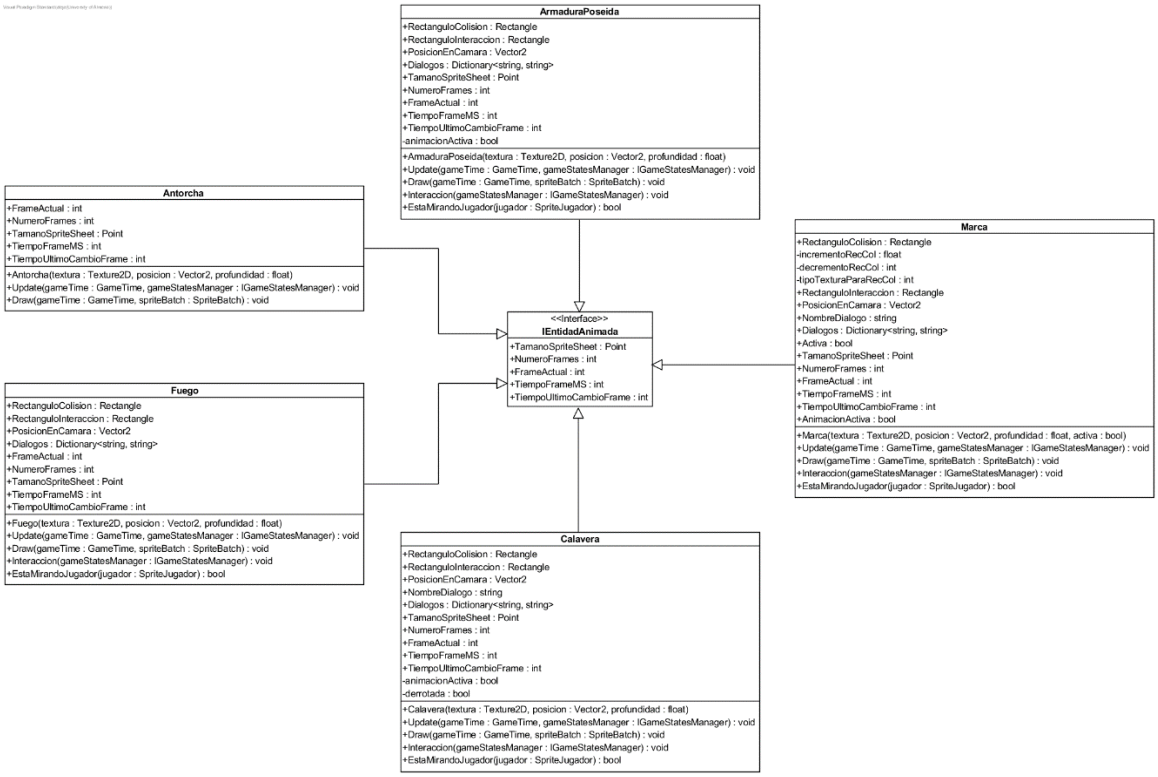


Figura 62. Diagrama de clases de la interfaz IEntidadAnimada y las clases que la implementan

La interfaz IEntidadAnimada (representada en la *Figura 62*) permite a las clases que la implementa contar con una animación que se refleje durante su mostrado por pantalla, otorgando de esta manera una sensación de movimiento o viveza al jugador. Para realizar esta función, las clases hijas de esta interfaz deben implementar una serie de variables o atributos: el tamaño de la hoja de animación (TamanoSpriteSheet), el número de frames de dicha hoja de animación (NumeroFrames), un contador que indique el frame actual (FrameActual), la duración que debe tener cada frame de la animación (TiempoFrameMS) y una variable donde almacenar el tiempo transcurrido desde el último cambio de frame (TiempoUltimoCambioFrame). Además, es fundamental y necesario que dichas clases sobrescriban el método de actualización Update, en el cual se comprobará el avance por la secuencia de frames.

3.4.8.2. IEntidadColisionable

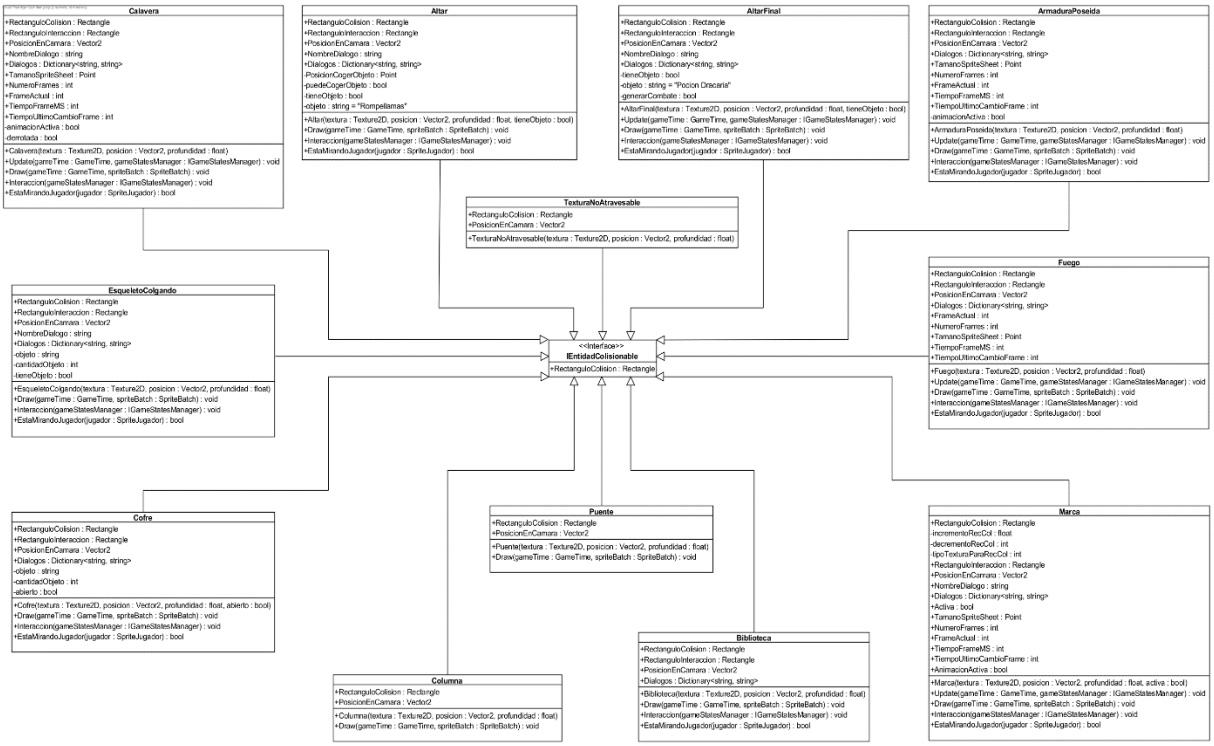


Figura 63. Diagrama de clases de la interfaz IEntidadColisionable y las clases que la implementan

La interfaz IEntidadColisionable (representada por la *Figura 63*) permite a las clases que la implementan ser objeto de colisión por parte del sprite del jugador, ejerciendo muchas veces la función de delimitar el espacio por donde puede moverse el jugador en el mapa. Para realizar esta función, la interfaz obliga a sus clases hijas a implementar una variable que represente el área de colisión del objeto concreto (RectanguloColision), la cual será utilizada por la clase del estado Mapa a la hora de comprobar si el jugador ha colisionado con alguna entidad.

3.4.8.3. IEntidadInteraccionable

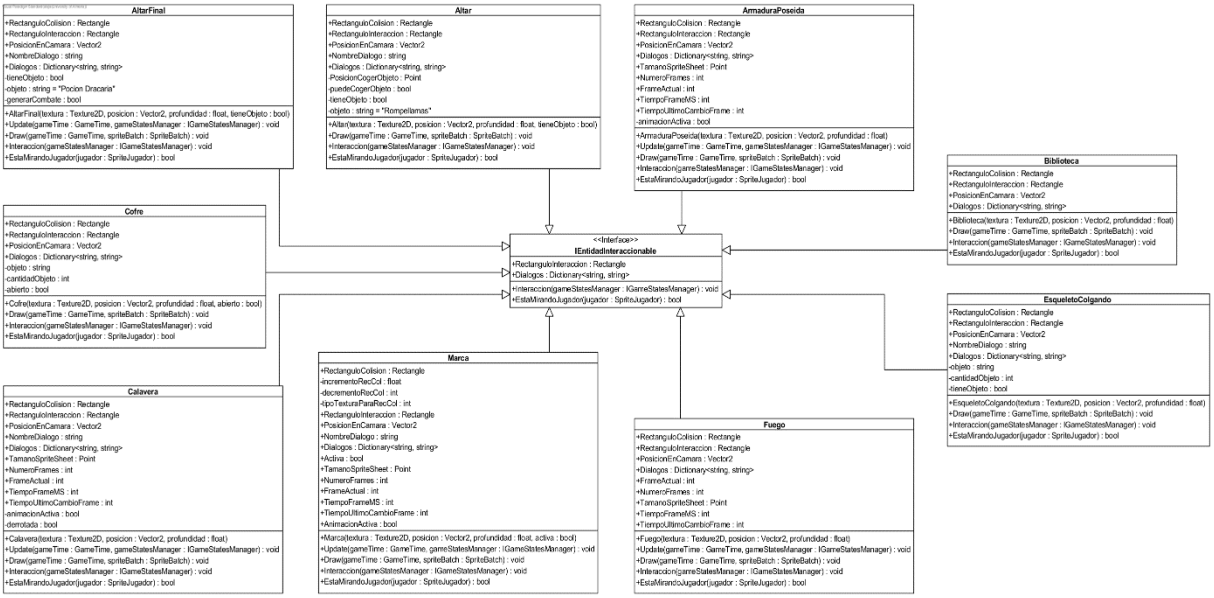


Figura 64. Diagrama de clases de la interfaz IEntidadInteraccionable y las clases que la implementan

La interfaz IEntidadInteraccionable (representada por la *Figura 64*) permite a las clases que la implementa la funcionalidad para ser interaccionadas por el jugador durante su recorrido por el mapa y generar distintos resultados en función del objeto. Para realizar esta funcionalidad, la interfaz obliga a sus clases hijas la incorporación de una variable que representa el área de interacción de la entidad (RectanguloInteraccion) y otra variable que almacena en una estructura Dictionary los diálogos correspondientes a la interacción de cada objeto concreto (Dialogos). Además de estasa variables, la interfaz también obliga a implementar dos métodos necesarios para la interaccion: Interaccion, el cual implementa toda la funcionalidad que pueda requerir la interacción; y EstaMirandoJugador, cuya función es comprobar si el jugador se encuentra en el área de interacción y si, además, está mirando en dirección a la entidad interaccionable.

3.4.9. Creación de objetos del mapa

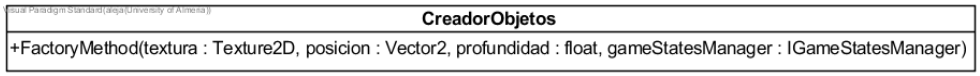


Figura 65. Clase CreadorObjetos

Una vez presentados e implementados los distintos tipos de entidades del mapa surge una situación que puede suponer un problema, la creación desde la clase Mapa de estas clases hijas de EntidadMapa. Las distintas variedades de la clase Textura no suponen ningún problema al contar con clases generales que engloban la construcción de estas (TexturaAtravesable y TexturaNoAtravesable), pero el problema radica más en cómo implementar de manera más general la construcción de los objetos que son implementados en clases individuales y con constructores que necesitan de distintos parámetros.

La solución escogida para plantar cara al problema que se ha mencionado es el uso del patrón de diseño Factory Method. Este patrón de diseño permite la creación de objetos de un subtipo determinado de una clase abstracta o interfaz a través de una clase factoría a la que se delega dicha construcción[6][80][81]. La estructura de este patrón se puede apreciar en la *Figura 66*, en la cual la clase Client se correspondería con el estado Mapa, la clase Creator con la clase CreadorObjetos (representada por la *Figura 65*) y la interfaz IProduct con la clase abstracta EntidadMapa.

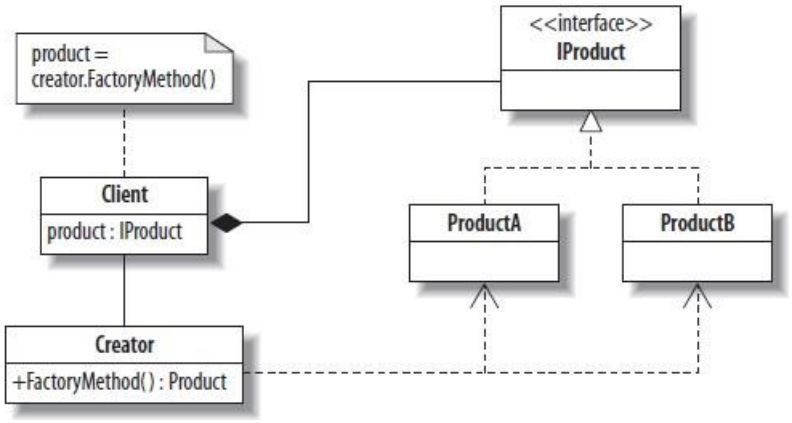


Figura 66. Diagrama de clases del patrón de diseño Factory Method[81]

3.4.10. Interacción/Diálogos

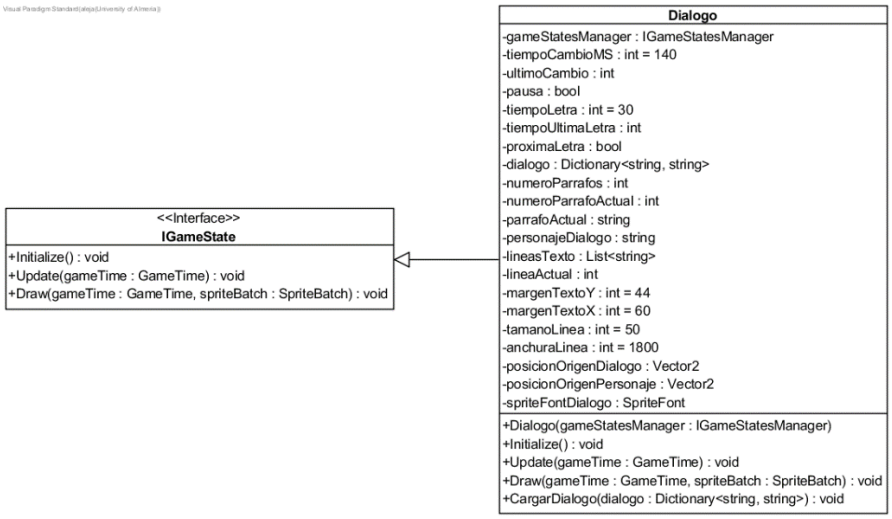


Figura 67. Diagrama de clases del estado Dialogo

En un videojuego del género de Dragon´s Curse, es decir, del género RPG, la trama narrativa supone uno de los grandes valores a realzar. Es por esto, que la implementación de las interacciones y diálogos en la que esta se presenta es una de las que el juego hará mayo uso, siendo una de las partes importantes como se indica anteriormente en el apartado “3.3. Análisis”.

En este estado del videojuego (representado en la *Figura 67*), se le mostrará al jugador una caja de diálogo en la que aparecerán progresivamente una serie de líneas de texto, y en ciertas ocasiones sprites de entidades representativas, asociadas a la interacción con entidades concretas del mapa, por las cuales este podrá ir avanzando hasta llegar a un final en el que será transportado al estado Mapa o al estado Combate en función del diálogo (apreciándose esto en el diagrama de estados de la *Figura 55*). Para llevar a cabo este proceso, la clase Dialogo cuenta con una serie de atributos o variables necesarias: un contador para controlar el tiempo entre las interacciones del jugador, el cual tiene como objetivo evitar que el jugador se salte inmediatamente todas las líneas de diálogo de una sola vez; un contador para permitir que gestionar el hecho de que el texto se muestre letra a letra automáticamente, lo cual aporta al jugador una sensación de continuidad al no mostrar todo el texto de golpe; una serie de variables que permiten almacenar los diálogos, más sus posibles entidades, y gestionar el transcurso a través de estos; y ciertas variables que representan características necesarias para el mostrado por pantalla de la caja de diálogo y las líneas de texto que contiene.

El resto de la implementación de la clase Dialogo es la que hereda de la interfaz IGameState, la cual no necesita de una nueva explicación. Sin embargo, si puede resultar interesante comentar el método CargarDialogo, el cual, como su nombre indica, lleva a cabo la carga de los distintos diálogos que serán necesarios durante el transcurso del estado. Estos diálogos son obtenidos por las distintas entidades interaccionables del mapa a través de archivos XML que representan las propiedades de estos, siendo la estructura de estos archivos la mostrada en la *Figura 68*. Esta estructura permite obtener los distintos párrafos del diálogo y sus posibles entidades asociadas, añadiendo en algunos casos atributos relacionados con la obtención de objetos que pueda generar la interacción.

```
<Dialogo id="319-62">
  <Parrafos>
    <Parrafo>
      <Texto>Parece que este esqueleto tiene algo dentro...</Texto>
      <Personaje>null</Personaje>
    </Parrafo>
    <Parrafo>
      <Texto>Has obtenido POCIÓN.</Texto>
      <Personaje>null</Personaje>
    </Parrafo>
  </Parrafos>
  <Objeto>Poción</Objeto>
  <Cantidad>1</Cantidad>
</Dialogo>
```

Figura 68. Ejemplo de estructura de los nodos del archivo XML contenedor de los diálogos

3.4.11. Gestión del equipo

Otros dos aspectos fundamentales que diferencia a los videojuegos del género de rol del resto, serían la capacidad de personalización de los personajes que controla el jugador, mediante niveles, habilidades o equipamientos, y el hecho de contar con un inventario de equipo, el cual permita almacenar los distintos objetos recogidos por el mapa para su posterior uso. Estas dos mecánicas del gameplay permiten que cada jugador que participe en el videojuego pueda decidir y tener una experiencia de juego personalizada, ya que no todo el mundo personalizará a los personajes de la misma manera o recogerá los mismos objetos durante su partida.

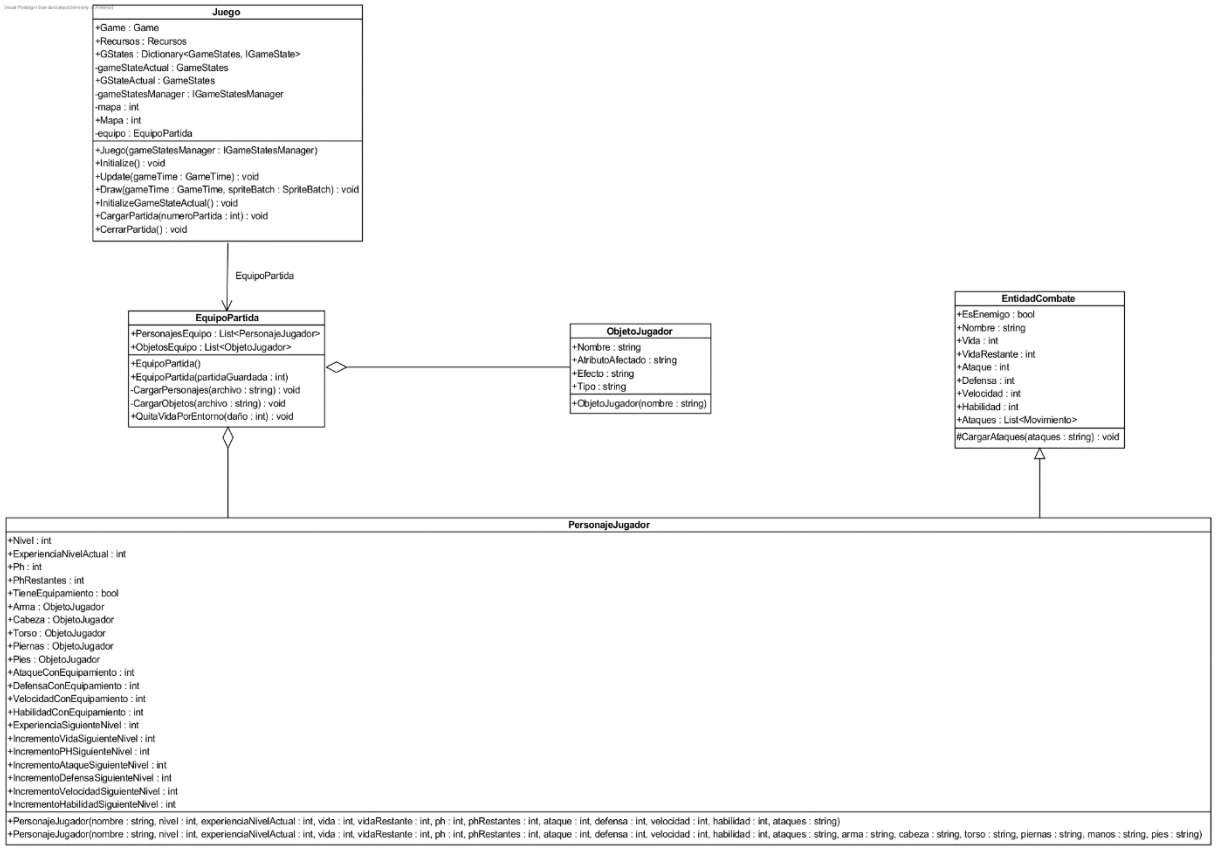


Figura 69. Diagrama de clases de la gestión del equipo

Este aspecto del gameplay ha sido implementado a través de la clase EquipoPartida (representada en el diagrama de clases de la *Figura 69*), la cual almacena en ella tanto al equipo de personajes del jugador como los objetos disponibles en el inventario a través de listas. A través de esta clase, de la cual se tiene una instancia en la clase del gestor de estados Juego, se realiza la carga de dichas listas mediante la lectura de dos archivos XML relacionados con la partida que se está jugando, uno que define el equipo de personajes y sus distintos atributos (*Figura 70*) y otro que indica los objetos del jugador junto a las respectivas cantidades de estos (*Figura 71*).

```
<Personaje>
  <Nombre>Athros</Nombre>
  <Nivel>10</Nivel>
  <ExperienciaNivelActual>0</ExperienciaNivelActual>
  <Vida>300</Vida>
  <VidaRestante>300</VidaRestante>
  <PH>15</PH>
  <PHRestantes>15</PHRestantes>
  <Ataque>40</Ataque>
  <Defensa>40</Defensa>
  <Velocidad>40</Velocidad>
  <Habilidad>40</Habilidad>
  <Ataques>Estoque kethardiano,Hoja magna,Sangre real,Grito de guerra</Ataques>
  <TieneEquipamiento>true</TieneEquipamiento>
  <Arma>Espada de acero kethardiano</Arma>
  <Cabeza>Diadema real kethardiana</Cabeza>
  <Torso>Coraza de acero kethardiano</Torso>
  <Piernas>Mallas de acero kethardiano</Piernas>
  <Manos>Guantes reales kethardianos</Manos>
  <Pies>Botas kethardianas</Pies>
</Personaje>
```

Figura 70. Ejemplo de estructura de los nodos del archivo XML del equipo de personajes

```
<Objeto>
  <Nombre>Poción</Nombre>
  <Cantidad>5</Cantidad>
</Objeto>
```

Figura 71. Ejemplo de estructura de los nodos del archivo XML del inventario

A continuación, en los siguiente subapartados, se va a proceder a explicar las clases usadas para representar a los personajes del jugador, PersonajeJugador, y los objetos del inventario, ObjetoJugador, con el objetivo de que se entienda mejora la representación que de cada uno.

3.4.11.1. Equipo de personajes

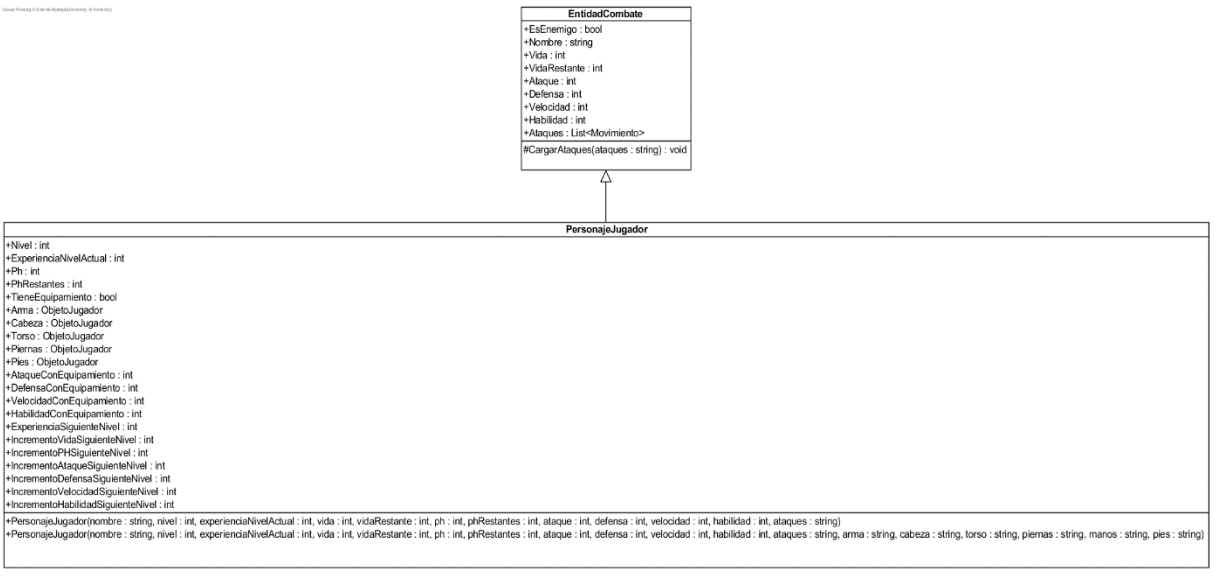


Figura 72. Diagrama de clases del equipo de personajes

La clase PersonajeJugador (representada en la *Figura 72*) se trata de una clase hija de EntidadCombate, la cual será posteriormente explicada durante el apartado "3.4.13. Entidades de combate", y que implementa todas las características de un personaje relacionadas con el sistema de niveles de experiencia y el equipamiento.

Por un lado, para implementar el sistema de niveles de experiencia esta clase cuenta con los atributos enteros Nivel, ExperienciaNivelActual y ExperienciaSiguienteNivel, mediante los cuales se gestiona el nivel actual del personaje y la experiencia que le hace falta obtener para subir al siguiente nivel. Además, cada nivel supone un incremento de las estadísticas del personaje, por lo que se cuenta con atributos numéricos enteros que informan sobre las estadísticas a incrementar cuando se sube al siguiente nivel y la cantidad de dicho incremento.

Por otro lado, la capacidad de un personaje para portar equipamiento genera una división en estos, ya que se ha decidido que en el videojuego Dragon´s Curse únicamente llevarán equipamiento los personajes principales a nivel de narrativa. En el caso de que el personaje tenga la capacidad de portar equipamiento, este deberá contar con una serie de atributos o propiedades que representan las distintas partes posibles de equipamiento (Arma, Cabeza, Torso, Piernas, Pies y Manos).

3.4.11.2. Objetos del equipo

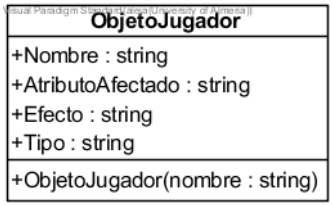


Figura 73. Clase ObjetoJugador

La clase ObjetoJugador (*Figura 73*) permite implementar y gestionar las distintas características que puede tener un objeto en el videojuego Dragon´s Curse. Estas características serían las siguientes:

- **Nombre:** hace referencia al nombre o id del objeto concreto y es el atributo que se pasa como parámetro al constructor de la clase.
- **AtributoAfectado:** permite decidir el atributo al que afecta del personaje sobre el que se usa el objeto. Este atributo afectado puede hacer referencia a una estadística del personaje o al estado adverso que sufra, los cuales están representados por la enumeración EstadosAdversos.
- **Efecto:** indica el incremento o decremento sufrido en las estadísticas del personaje o el estado adverso concreto que debe eliminar.
- **Tipo:** mediante esta propiedad del objeto se puede diferenciar entre los distintos tipos de objetos existentes en el videojuego Dragon´s Curse:
    - **Variedad:** objetos que se pueden utilizar durante el estado Mapa o el estado Combate, los cuales suelen estar relacionados con estadísticas como la vida del personaje, sus puntos de habilidad o el nivel.
    - **Combate:** objetos que solo permiten su uso durante el transcurso del estado Combate, pudiendo eliminar estados adversos o incrementando ciertas estadísticas del personaje para el combate en cuestión únicamente.
    - **Equipamiento (donde se incluyen los distintos tipos de equipamiento):** objetos que tienen como finalidad ser equipados por un personaje para incrementar sus estadísticas.

- **Especial:** objetos relacionados con el transcurso de la historia y de carácter único normalmente.

3.4.12. Combate

El estado Combate sería el último a explicar de los tres estados principales del gestor de estados Juego, además de que cuenta con una implementación especial debido a su mecánica o gameplay. Durante un combate, el jugador irá avanzando por una serie de fases y turnos en las cuales irá tomando decisiones que influyan en el devenir de dicho combate. Llevando a cabo un análisis más profundo sobre estas fases, el autor se percató de que estas cuentan con una definición y una estructura bastante similares a las de un estado de la interfaz IGameState, por lo que se tomó la decisión de implementar clases hijas de IGameState específicas para cada una de estas fases. Esta decisión llevó, a su vez, a contemplar el estado Combate como un nuevo gestor de estados de videojuego que gestionara los estados de las distintas fases, lo cual hace que la clase Combate implementa la interfaz IGameStatesManager además de IGameState. Por lo tanto, la estructura final del estado Combate quedaría tal como se puede aprecia en la *Figura 74*, contando a su vez con un proceso de ejecución como el representado por la *Figura 75*.

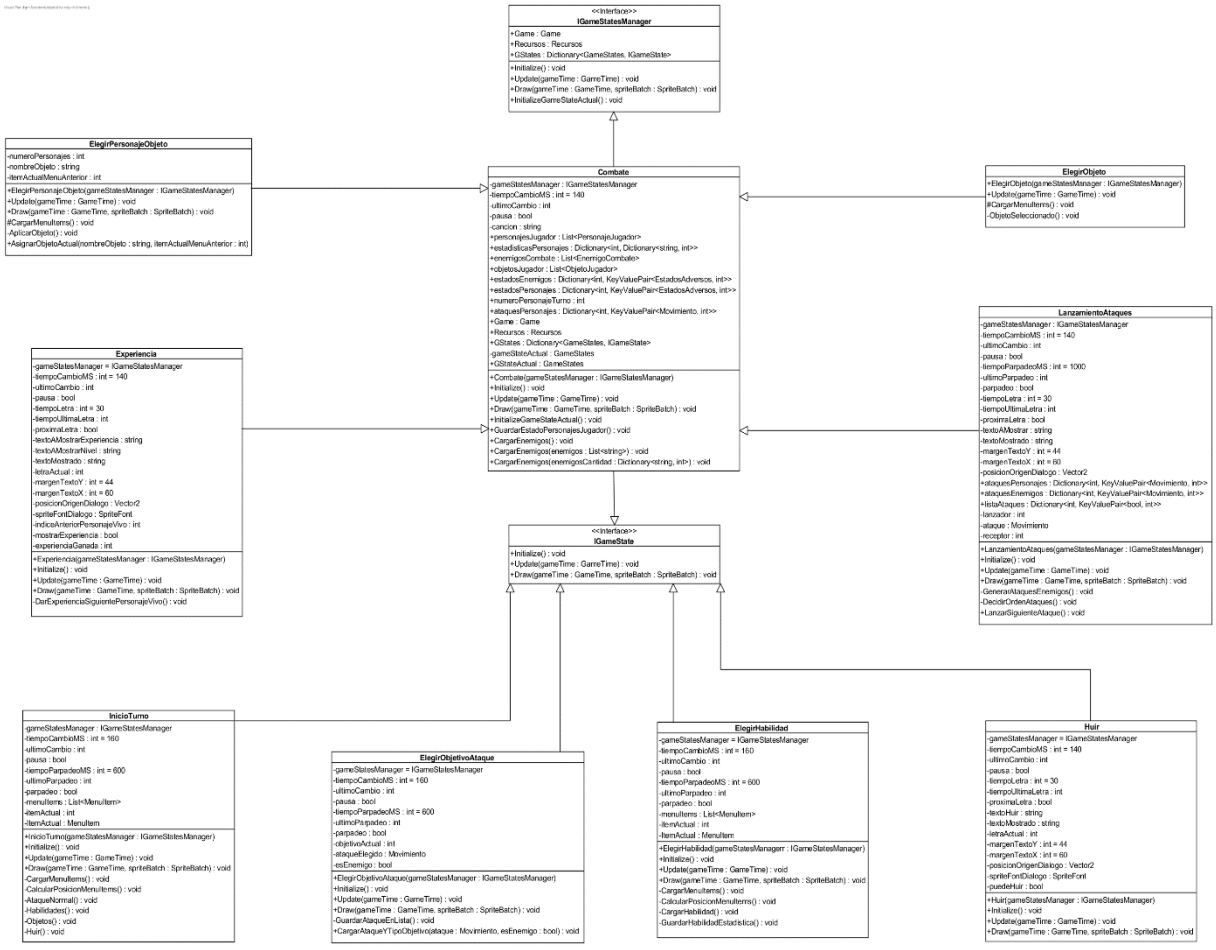


Figura 74. Diagrama de clases del estado Combate junto a sus estados

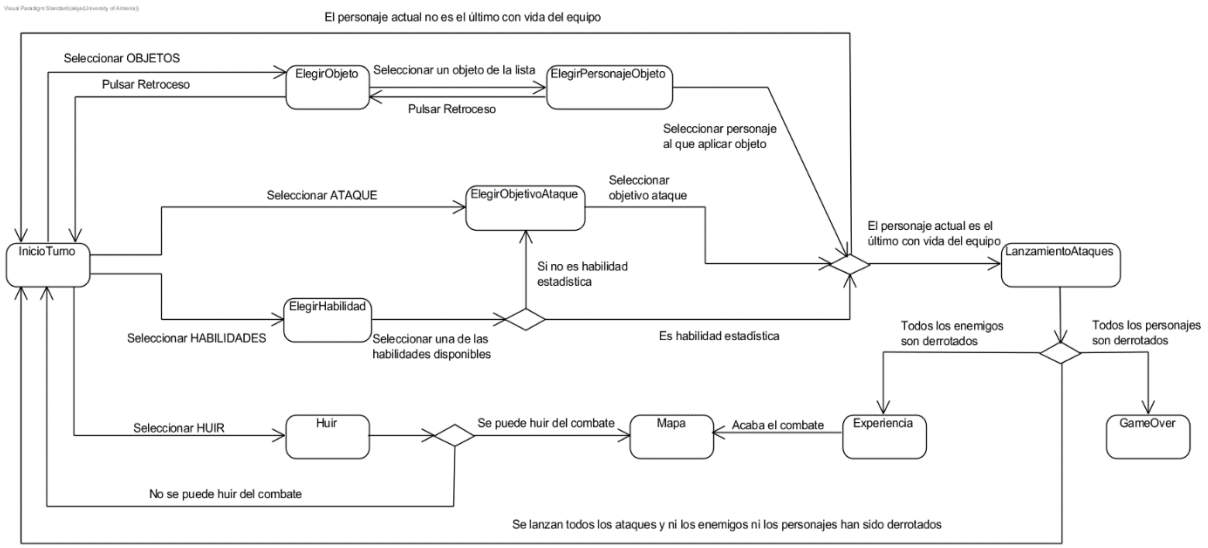


Figura 75. Diagrama de estados del proceso llevado a cabo en el estado Combate

Es importante destacar, que el estado Combate almacena en su clase referencias al inventario y al estado actual del equipo de personajes del jugador, separando a su vez en una estructura Dictionary las estadísticas Ataque, Defensa, Habilidad y Velocidad del personaje. Esto último se debe a que durante el combate dichas estadísticas puede verse modificadas por habilidades u objetos, manteniéndose dichas modificaciones solamente en el transcurso del combate y siendo restablecidas una vez que este termina. Otra característica a destacar de la clase Combate serían las variables que gestionan los estados adversos sufridos por los personajes y los enemigos durante este, siendo almacenados en una lista para cada tipo de entidad de combate. Estos estados adversos, como Quemado o Aturdido, son comprobados en el estado o fase InicioTurno, en la cual se realiza la acción asociada a dicho estado y se disminuye su duración si es necesario.

Antes de explicar la clase Movimiento, la cual es fundamental en el transcurso de Combate, se va a proceder a explicar brevemente los métodos que permiten generar los equipos de enemigos en Combate, los cuales son una pieza fundamental también en este estado. En el videojuego Dragon´s Curse se ha planteado que dicha generación sea posible de tres maneras distintas: sin pasar ningún parámetro al método CargarEnemigos, con lo cual se generarán entre uno y cinco enemigos y decidiendo el tipo de cada uno mediante probabilidad (siendo una excepción los jefes y subjefes); pasando una lista de tipos de enemigos como parámetro al método CargarEnemigos, generando entre cinco y el número de estos tipos de enemigos y decidiendo la cantidad de cada tipo mediante aleatoriedad; o pasando como parámetro al método CargarEnemigos una estructura Dictionary que contenga los tipos de enemigos necesarios y sus cantidades, las cuales deben sumar un máximo de cinco enemigos.

3.4.12.1. Movimientos o habilidades

Los movimientos o habilidades son un aspecto fundamental en el estado Combate como se ha mencionado anteriormente, ya que estos aportan variedad a la mecánica del combate y permiten al jugador desarrollar una estrategia para cada combate que no se base únicamente en realizar ataques básicos y usar objetos. Es por esto que han sido reflejados en una clase individual, Movimiento (*Figura 76*), la cual permite obtener las distintas características propias de un movimiento a través de un archivo XML (*Figura 77*). Estas características o atributos serían las siguientes:

- **Nombre:** hace referencia al nombre elegido para describir al movimiento o habilidad y ejerce además las funciones de identificador.
- **Tipo:** permite diferenciar las habilidades en una serie de tipos con efectos distintos:
    - **Ataque:** habilidad que incrementa el daño realizado por el personaje, calculando este como una suma de la estadística de ataque del personaje y el daño de la habilidad.
    - **Ataque multiple:** habilidad que realiza el mismo efecto que las del tipo Ataque, con la diferencia de que el objetivo de esta es todo el equipo contrario.
    - **Ataque con efecto:** habilidad que realiza el mismo efecto que las del tipo Ataque, con la diferencia de que cuenta con una probabilidad de producir un estado adverso en el objetivo.
    - **Estadistico:** habilidad que lleva a cabo modificaciones sobre los valores de las estadísticas del personaje que las lanza, pudiendo estas ser incrementos o decrementos.
    - **Especial:** habilidades que ejercen una efecto muy específico sobre el objetivo de estas, como puede ser la habilidad “Escudo humano” que redirige todos los ataques lanzados a un compañero de equipo hacia la entidad que lanza esta.
- **CostePH:** se trata de un atributo exclusivo de los movimientos no pertenecientes a enemigos, con el cual se comprueba si es posible usar dicho movimiento en función de los PH o puntos de habilidad que le queden al personaje que lo usa.
- **Probabilidad:** se trata de un atributo exclusivo de los movimientos pertenecientes a enemigos, con el cual se indica la probabilidad de que el enemigo utilice este movimiento y no los otros de los que dispone.
- **Daño:** se trata de un atributo exclusivo de los movimientos que generan daños, con el cual se indica el incremento que se realiza sobre el ataque del personaje que usa el movimiento.
- **Efecto:** se trata de un atributo exclusivo de los movimientos del tipo Ataque con efecto, con el cual se indica el estado adverso que puede generar en el oponente.
- **Ataque:** se trata de un atributo exclusivo de los movimientos del tipo Estadistico, con el cual se refleja la modificación en la estadística Ataque de la entidad de combate que lo utiliza.
- **Defensa:** se trata de un atributo exclusivo de los movimientos del tipo Estadistico, con el cual se refleja la modificación en la estadística Defensa de la entidad de combate que lo utiliza.
- **Velocidad:** se trata de un atributo exclusivo de los movimientos del tipo Estadistico, con el cual se refleja la modificación en la estadística Velocidad de la entidad de combate que lo utiliza.
- **Habilidad:** se trata de un atributo exclusivo de los movimientos del tipo Estadistico, con el cual se refleja la modificación en la estadística Habilidad de la entidad de combate que lo utiliza.
- **EsEnemigo:** hace referencia al hecho de si el movimiento se trata de un ataque exclusivo de los enemigos.

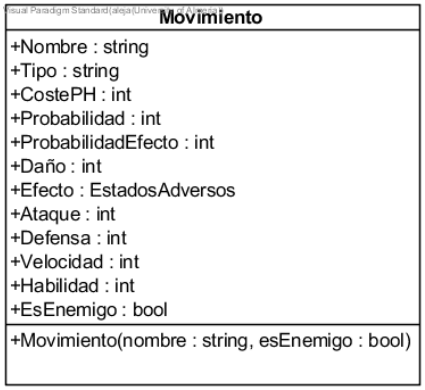


Figura 76. Clase Movimiento

```
<Movimiento id="Llama aúrea">
  <Tipo>Ataque con efecto</Tipo>
  <Probabilidad>45</Probabilidad>
  <Daño>60</Daño>
  <Efecto>Quemado</Efecto>
</Movimiento>
```

Figura 77. Ejemplo de estructura de los nodos del archivo XML de movimientos

3.4.13. Entidades de combate

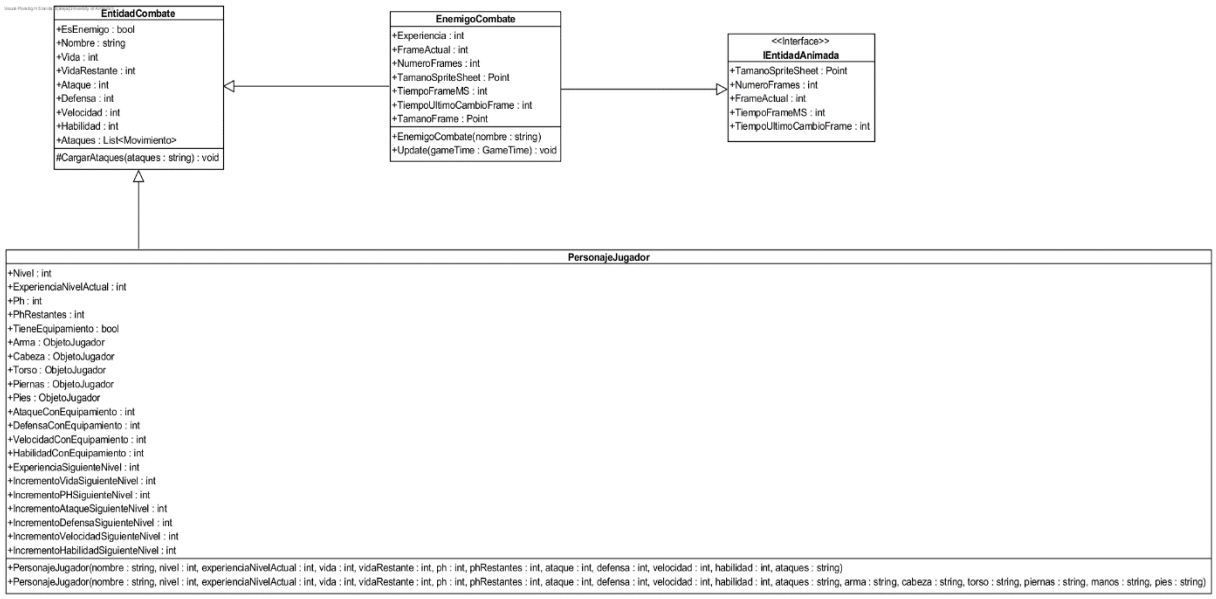


Figura 78. Diagrama de clases de las entidades de combate

Al igual que en el estado Mapa, el estado Combate cuenta con sus propias entidades (EntidadCombate) que son representadas en el diagrama de clases de la *Figura* 78 y permiten que el jugador puede hacer uso de ellas para enfrentarse a otro grupo de estas, estando por tanto diferenciadas en dos tipos: los personajes del jugador y los enemigos a los que se enfrenta. Sin embargo, aunque estas dos subclases de EntidadCombate cuenten con distintos atributos y e implementación, existen una serie de atributos comunes que implementa la propia EntidadCombate (si es enemigo, el nombre, las estadísticas de vida, ataque, defensa, velocidad y habilidad y la ristra de ataques de los que dispone) y un método común que permite cargar los ataques de la entidad desde el archivo XML de estos (CargarAtaques). Como dato interesante, que además no se ha mencionado anteriormente, el atributo Habilidad hace referencia a la probabilidad de que, cuando la entidad de combate realice un ataque, este sea crítico y no tenga en cuenta la defensa del objetivo.

Por un lado, los personajes del jugador son representados mediante la clase la PersonajeJugador, la cual ha sido explicada anteriormente en el subapartado “3.4.11.1 Equipo de personajes” y no necesita de una nueva explicación. Sin embargo, la representación de los enemigos en el combate es mediante la clase EnemigoCombate, la cual no ha sido explicada aún. Esta clase implementa la interfaz IEntidadAnimada, con lo cual se consigue que estas entidades ofrezcan cierta sensación de movimiento durante el combate. Además, cuenta con un atributo que hace referencia a la cantidad de experiencia, la cual es otorgada a los personajes vivos al final del combate por derrotar a dicho enemigo.

3.5. Pruebas

Durante el diseño e implementación del videojuego Dragon´s Curse y una vez obtenida una primera versión de este, se llevaron a cabo una serie de sesiones de pruebas de distinta índole y con distinto público objetivo.

En el proceso de desarrollo del videojuego, el autor iba a llevando a cabo pruebas muy primitivas continuamente sobre los aspectos individuales que se iban implementando, gracias a las cuales se encontraban bugs o errores graves /principales a corregir. A través de esta serie de pruebas, el videojuego pudo llegar a su versión alfa y estaba lista para ser probada por compañeros, amigos y familiares del autor, los cuales servirían como agente externo para la búsqueda de bugs y la producción de críticas constructivas de feedback.

Una vez que esta versión alfa fue distribuida entre los distintos "testers", se envió personalmente a cada uno de estos una encuesta[82], en la cual valorarían los distintos aspectos principales del videojuego haciendo uso de una puntuación numérica entre cero y diez (o una opción representativa en algunos casos) junto a un comentario con posibles mejoras del aspecto concreto por el que se pregunta. Además, se requerían dos respuestas más generales como son la aparición de bugs junto al contexto de estos y una valoración general del videojuego Dragon´s Curse. El objetivo por parte del autor para esta encuesta era contar con una serie de opinión de estos testers, gracias a la cual, se podrían distinguir trabajos futuros adicionales de ampliación o mejora que añadir a los obtenidos durante el recorte del diagrama de casos de uso de la *Figura 26*.

3.5.1. Resultados de la encuesta de feedback

En primera instancia, es interesante mostrar el objetivo más básico de esta encuesta y el cual influye en la experiencia de usuario, la aparición de bugs o errores durante la ejecución del videojuego. Como se representa en la *Figura 79*, más de la mitad de los jugadores del periodo de pruebas no han encontrado ninguno de estos bugs o errores, fruto seguramente de la primera fase de prueba llevada a cabo por el autor antes de distribuir la versión pública. Sin embargo, los usuarios que los han encontrado (habiendo encontrado como mucho uno por jugador) han sido en las fases de combate del gameplay, lo cual refleja que este es el aspecto con mayor margen de mejoras y correcciones.

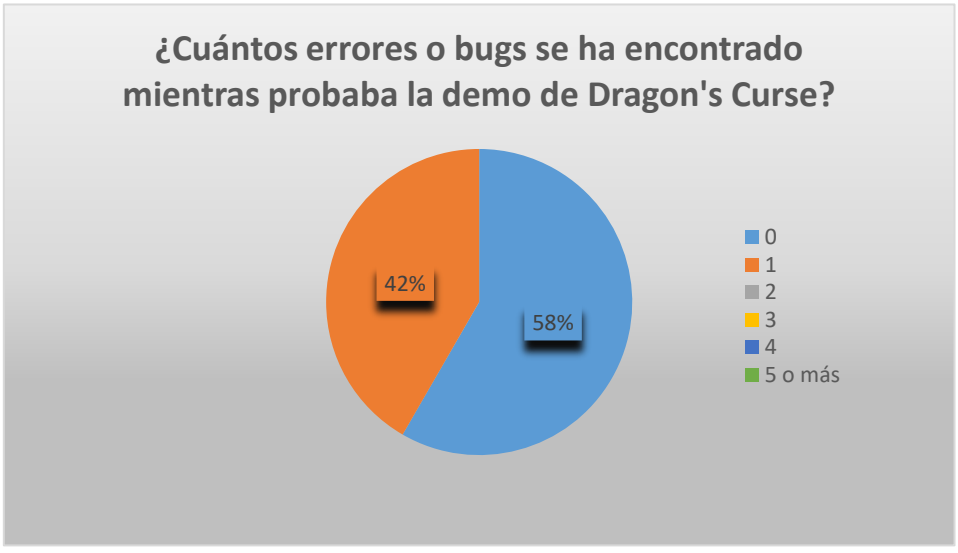


Figura 79. Respuestas respecto al número de bugs encontrados

Por los resultados de esta primera cuestión, en cuanto a los aspectos del combate se pregunta por distintas características como la tasa de aparición de combates (*Figura 80*), la dificultad de estos (*Figura 81*), la interfaz (*Figura 82*) y las mecánicas del gameplay de este aspecto (*Figura 83*). En cuanto a comentarios y posibles mejoras sugeridas, los jugadores parecen haberse puesto de acuerdo en el hecho de que la dificultad y la tasa de aparición de estos combates son correctas, sin embargo opinan que un cambio en la forma de indicar el objetivo y el lanzador de los ataques o habilidades mejoraría bastante la interfaz y, por tanto, en el gameplay de los combates.

Figura 80. Respuestas respecto a la tasa de aparición de combates

Figura 81. Respuestas respecto a la dificultad de los combates

Figura 82. Respuestas respecto a la valoración de la interfaz del combate

Figura 83. Respuestas respecto a la valoración del gameplay del combate

Por otro lado, los jugadores de las pruebas han valorado de una manera bastante equilibrada los la fase de recorrido del mapa, con algunas excepciones puntuales en notas negativas. Los aspectos encuestados alrededor de la fase del mapa son el gameplay del recorrido por este (*Figura 84*) y el diseño del mapa en cuestión (*Figura 85*), siendo el primero de estos aspectos mejor valorado y teniendo la monotonía del diseño como punto flaco. Además, uno de los comentarios más repetidos respecto al mapa es la posible mejora que supondría la inclusión de un minimapa, el cual ejerza las funciones de guía.

Figura 84. Respuestas respecto a la valoración del gameplay del recorrido por el mapa

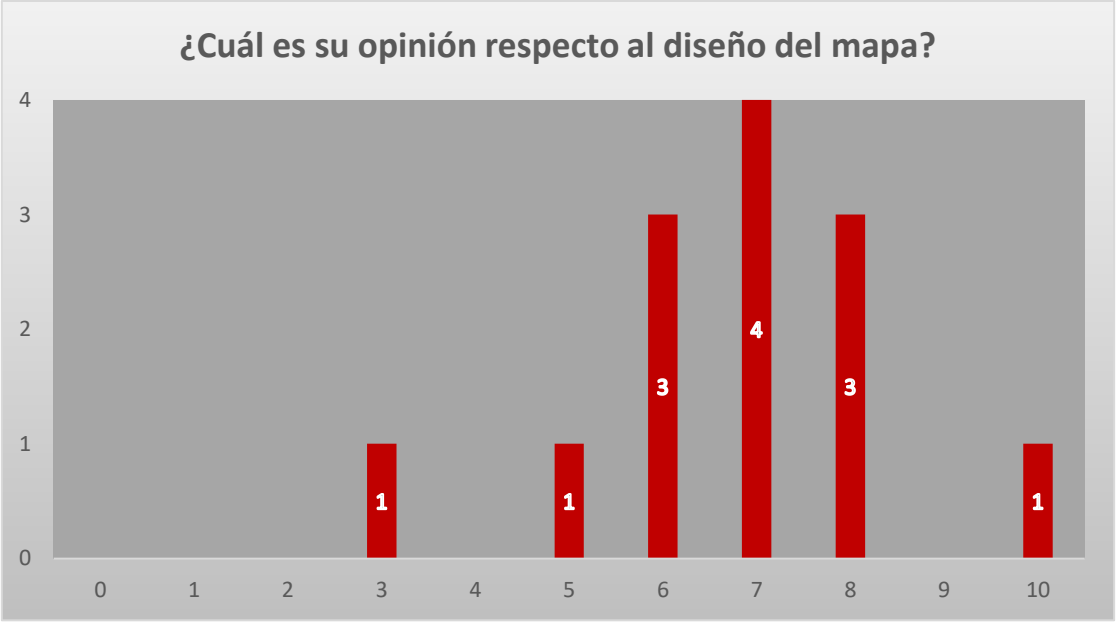


Figura 85. Respuestas respecto a la valoración del diseño del mapa

El aspecto de los diálogos y la narrativa era algo que preocupaba bastante al autor en cuanto a la opinión de esta encuesta, ya que la forma de presentarlos no era la que él había pensado durante las fases tempranas del proyecto y, además, consideraba que uno de los puntos flacos podía esta narrativa por parecer ridícula o aburrida. Sin embargo, a excepción de algún comentario que indicaba su simplicidad narrativa, este aspecto de Dragon´s Curse ha sido valorado positivamente como se aprecia en la *Figura 86* y la *Figura 87*, destacando el cuidado de los detalles en referente a que cada objeto tengo su propia interacción y esta sea única.

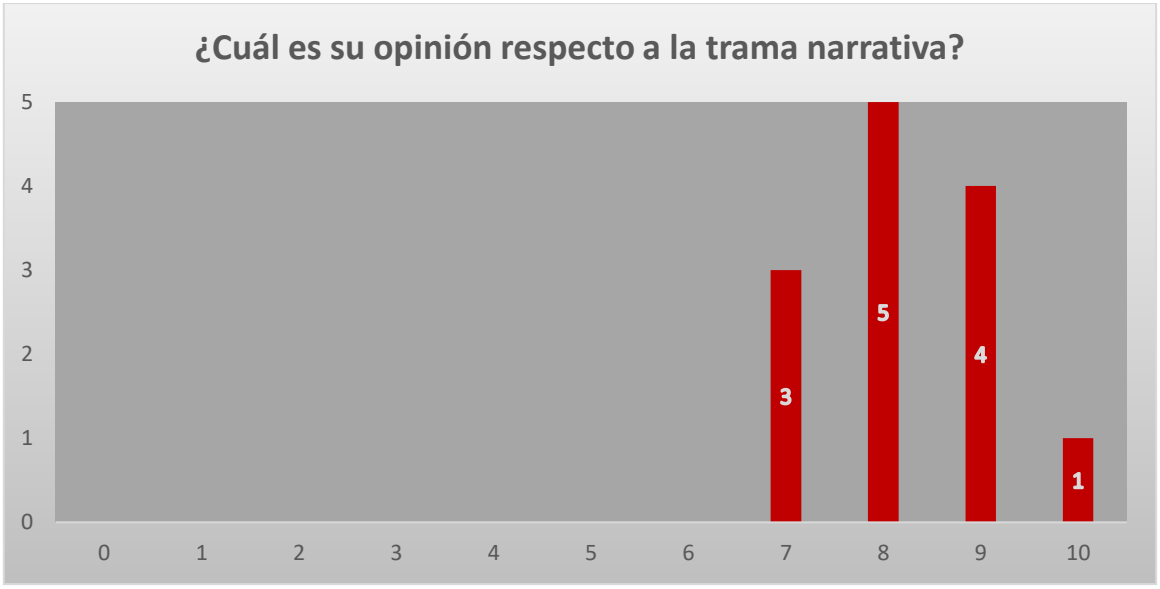


Figura 86. Respuestas respecto a la valoración de la trama narrativa

Figura 87. Respuestas respecto a la valoración de los diálogos

En cuanto a los menús, se ha obtenido un feedback general bastante positivo de los jugadores (*Figura 88*), los cuales han destacado que, si se añadiera cierta información sobre los objetos del inventario, estos menús estarían perfectos.

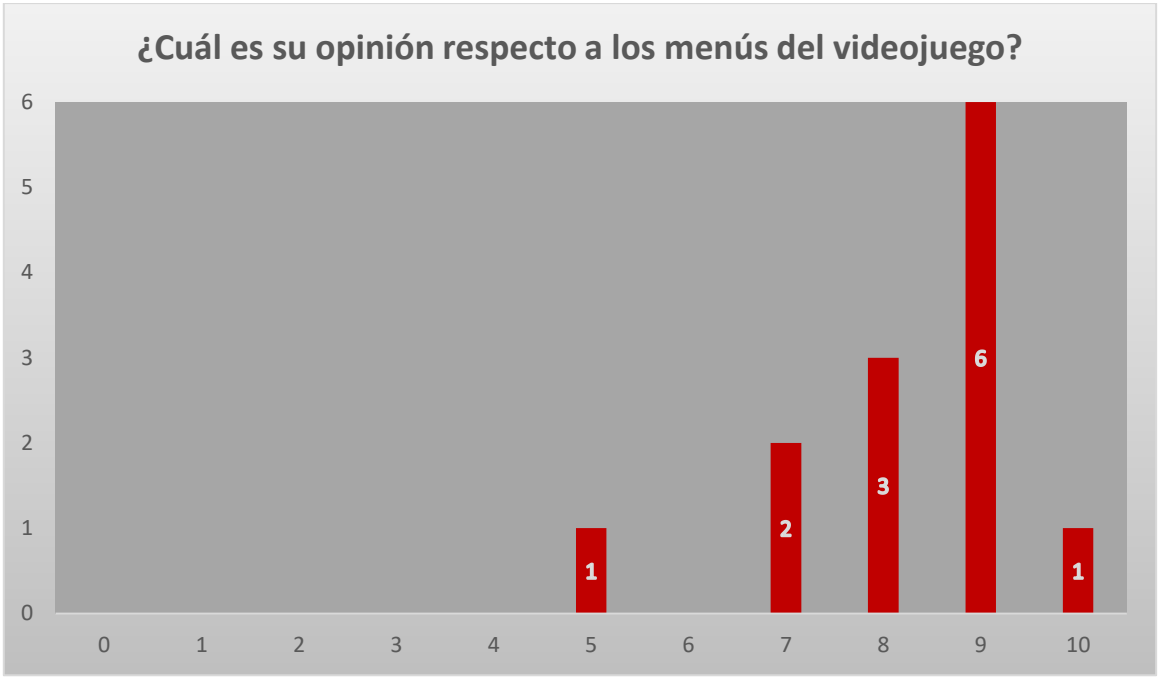


Figura 88. Respuestas respecto a la valoración de los menús

Como recompensa a la cantidad de trabajo que ha supuesto para el autor toda la generación del contenido artístico, el feedback de estos elementos, tanto gráficos como sonoros, ha sido el que mayor valoración ha obtenido por las respuestas de los jugadores reflejadas en la *Figura 89* y la *Figura 90*. Es importante destacar, que se ha valorado especialmente la variedad de este contenido artístico y su adecuación al contexto del videojuego.

Figura 89. Respuestas respecto a la valoración del contenido gráfico

Figura 90. Respuestas respecto a la valoración del apartado sonoro

Finalmente, como conclusión de esta encuesta se le preguntaba a los jugadores respecto a una valoración general del videojuego Dragon´s Curse junto a posibles comentarios de carácter general. Es en esta valoración cuando el autor se ha sentido más orgulloso del trabajo realizado, ya que esta ha obtenido un valor de 8,5 (*Figura 91*) y se ha destacado de manera general que el videojuego es interesante e invita a volver a ser jugado una vez haya sido completado a nivel de contenido.

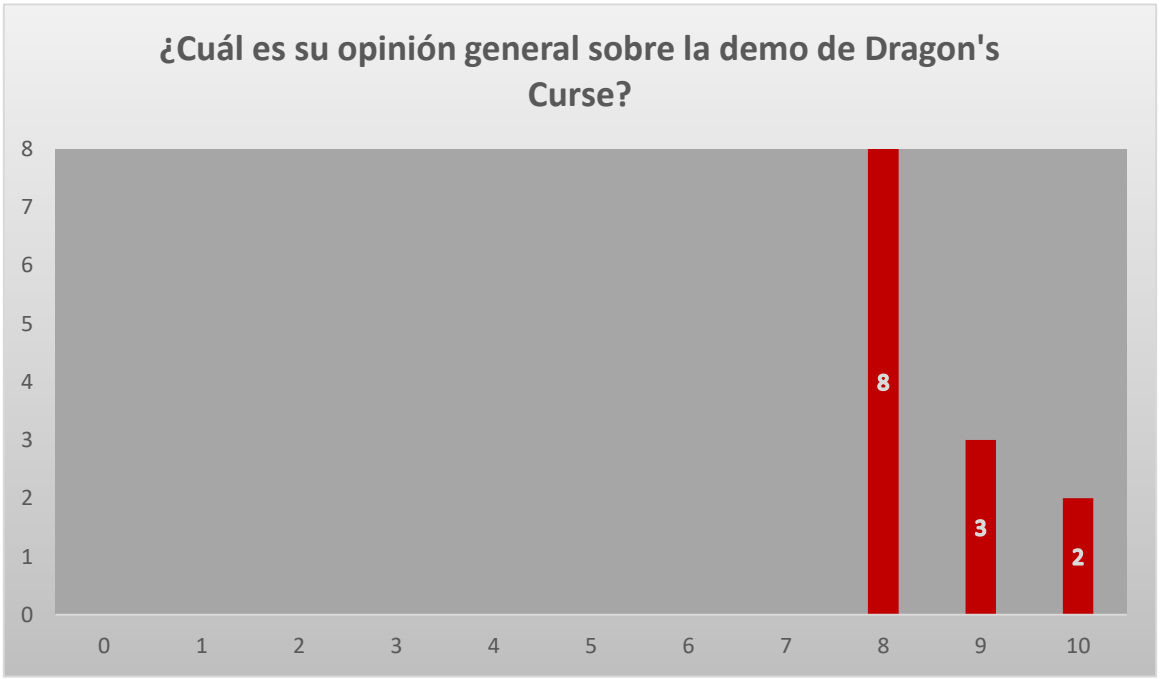


Figura 91. Respuestas respecto a la valoración general de la versión de prueba de Dragon´s Curse

3.6. Planificación temporal del proyecto

En este apartado referente a la planificación temporal del proyecto, se va a resumir en cierta medida el periodo que ha transcurrido desde que se decidió comenzar con este Trabajo Fin de Grado hasta el día de entrega de este, incluyendo los distintos problemas y épocas por la que ha pasado el proyecto.

Este Trabajo Fin de Grado fue comenzado alrededor de marzo de 2018 (teniendo en cuenta que la revisión bibliográfica fue llevada a cabo anteriormente para el anteproyecto), debido a la realización de las prácticas en empresa, incluidas en el programa del grado, y el hecho de que el autor se encontraba trabajando como profesor particular en una academia, lo cual se ocupaba los pocos huecos libres que quedaban en su horario. A principios de marzo, y una vez finalizado el periodo de prácticas en empresa, el autor decidió dejar aparcado el trabajo de profesor particular para poder dedicarse plenamente al segundo cuatrimestre de universidad y este Trabajo Fin de Grado, ya que la estimación inicial de tiempo realizada sobre el proyecto daba como resultado un periodo de seis meses aproximadamente teniendo en cuenta una jornada de cuatro horas al día dedicadas exclusivamente a este proyecto. Por lo tanto, las expectativas del autor respecto a la planificación le permitían poder presentar este Trabajo Fin de Grado en la convocatoria de septiembre, más teniendo en cuenta que en verano dispondría de más tiempo.

Sin embargo, la idea original del videojuego Dragon´s Curse era demasiado ambiciosa y excedería con creces la planificación temporal inicial, fruto quizás de que el autor, que no se había adentrado nunca en el mundo del desarrollo de videojuegos, suele disfrutar de juegos AAA[83] y estos crean grandes expectativas en desarrolladores amateurs de videojuegos. El problema fue que esta percepción sobre la idea inicial no surgió hasta que, durante el mes de junio, el autor se vio abrumado a la par que agobiado por la cantidad ingente de contenido artístico que era necesario para desarrollar un videojuego de tales magnitudes. Es importante mencionar que, aun no siendo un especialista o ni siquiera un gran dibujante, el autor tenía como una de las ideas inamovibles que todo el contenido gráfico del videojuego fuera generado por él, ya que quería que Dragon´s Curse contará con un apartado visual propio que sirviera como uno de los puntos fuertes de este. Por lo tanto, como consecuencia de este problema, se llevaron a cabo una serie de recortes a nivel de gameplay y narrativa que redujeran el contenido necesario, llegando hasta el concepto llevado a cabo en el cual el videojuego se podría considerar en una fase intermedia entre una versión alfa y una versión beta.

Una vez que la planificación temporal volvió a ser estimada, la idea pasó a ser la de realizar toda la parte de diseño que faltaba, tanto a nivel artístico como de diagramas, durante el mes de junio y la primera quincena de agosto para poder dedicar el tiempo entre esta primer quincena y la convocatoria de septiembre a intentar realizar todo el aspecto de implementación, pruebas y redacción de esta memoria. Pero como suele ocurrir en los primeros proyectos de cualquier desarrollador de software, la planificación temporal no se suele estimarse correctamente y este fue el caso, ya que no fue hasta una vez entrados en septiembre cuando el autor concluyó la fase de diseño. Por esta razón, la entrega del proyecto fue aplazada aún más hasta la convocatoria de diciembre de 2018, con su correspondiente nueva estimación de tiempo de las fases que quedaban por llevar a cabo.

Esta planificación se consiguió que fuera definitiva, obteniendo como resultado final la cronología representada en la *Figura 92* y la *Figura 93*.

Iteración	Duración	Tarea	Observaciones
1	14/02/18 – 28/02/18	Revisión bibliográfica	
2	01/03/18 – 09/03/18	Análisis sobre el proyecto	
3	09/03/18 – 10/09/18	Diseño	Durante este periodo se generaron los distintos diagramas del proyecto y el contenido artístico
4	18/10/18 – 19/10/18	Implementación del gestor de estados	Esta implementación incluye tanto las interfaces IGameState y IGameStatesManager como el gestor principal del videojuego usando el patrón de diseño State
5	20/10/18 – 20/10/18	Implementación del almacén de recursos	Se trata de la implementación de la clase Recursos y su instanciación en la clase principal DragonsCurse.cs
6	21/10/18 – 22/10/18	Implementación del menú principal	Esta implementación incluye las clases abstractas Menu y MenuEstatico, las clases estado MenuPrincipal e InicioJuego y la clase MenuItem
7	23/10/18 – 23/10/18	Implementación de la pantalla de carga	Se trata de la implementación del estado de transición Cargando
8	23/10/18 – 24/10/18	Implementación del gestor de estados Juego	Además de definir Juego como un nuevo gestor de estados, se definen los distintos estados de este gestor
9	24/10/18 – 31/10/18	Implementación del mapa	Esta implementación se trata de la de la clase estado Mapa y las clases SpriteJugador y Camara
10	01/11/18 – 01/11/18	Implementación de las texturas del mapa	Esta implementación incluye la de la clase abstracta Textura como la de sus subclases TexturaAtravesable y TexturaNoAtravesable y la interfaz IEntidadColisionable
11	02/11/18 – 06/11/18	Implementación de los distintos objetos del mapa	Esta implementación incluye la abstracción de Textura hacia una nueva clase EntidadMapa, la creación de las interfaces IEntidadAnimada e IEntidadInteraccionable y la creación de todas las clases que faltaban de entidades del mapa
12	08/11/18 – 09/11/18	Implementación del creador de objetos	Implementación de CreadorObjetos aplicando el patrón de diseño Factroy Method

13	11/11/18 – 12/11/18	Implementación del menú de pausa	Implementación de MenuPausa básica para permitir pausar, continuar y salir al menú principal
14	12/11/18 – 13/11/18	Implementación de los diálogos	Implementación del estado Dialogo y creación del archivo XML para los diálogos
15	13/11/18 – 13/11/18	Implementación del equipo de personajes	Esta implementación incluye los distintos menús internos del menú de pausa, la estructura dada por la clase EquipoJugador y la clase PersonajeJugador junto al XML de personajes del jugador
16	14/11/18 – 14/11/18	Implementación del inventario de objetos	Esta implementación incluye los distintos menús internos del menú de pausa, la ampliación de EquipoJugador y la clase ObjetoJugador junto al XML de objetos existentes
17	15/11/18 – 18/11/18	Implementación del sistema de combate	Esta implementación incluye la creación o ampliación de todas las clases relacionadas con la clase estado Combate, la cual implementa también la interfaz IGameStatesManager
18	19/11/18 – 20/11/18	Periodo de pruebas por parte del autor	Durante este periodo se realizaron pruebas exhaustivas para encontrar el mayor número de bugs antes de distribuir el juego para su testeo público
19	20/11/18 – 28/11/18	Realización de la memoria y las pruebas públicas de Dragon´s Curse	Antes de este periodo ya se había comenzado a redactar esta memoria, pero una vez en este periodo se focalizaron los esfuerzos en la redacción de esta y el envío de versiones al tutor

Figura 92. Cronología final del proyecto Dragon´s Curse

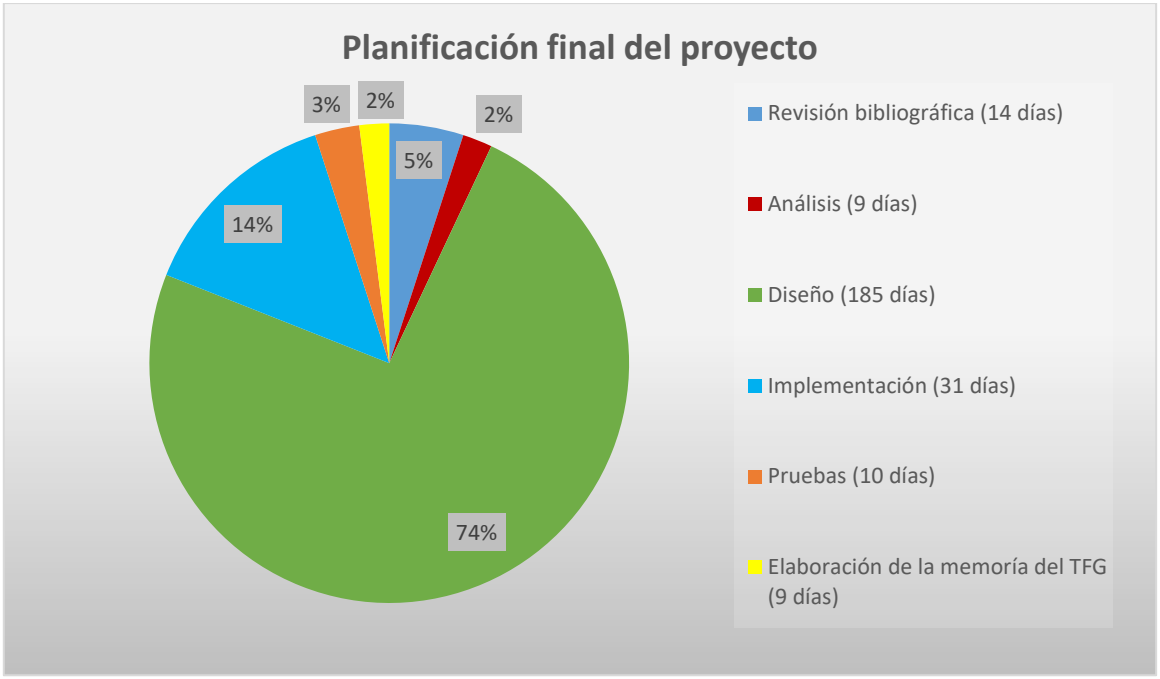


Figura 93. Planificación final del proyecto

3.7. Herramientas usadas

En todo el proceso de desarrollo de este Trabajo Fin de Grado se han utilizado diversas herramientas, sin las cuales habría sido imposible o enormemente más complicado la realización del proyecto. Estas herramientas son las siguientes:

- El lenguaje de programación utilizado ha sido C#, un leguaje orientado a objetos desarrollado por Microsoft.
- El entorno de desarrollo o IDE utilizado para el desarrollo de la implementación ha sido Visual Studio 2015.
- El motor de videojuegos o framework utilizado en este proyecto ha sido MonoGame.
- El sistema de control de versiones utilizado para gestionar el código de este proyecto ha sido Git, alojando el repositorio remoto en GitHub.
- Los diagramas UML utilizados durante esta memoria como documentación han sido generados usando la herramienta Visual Paradigm 14.2 distribuida por el Object Management Group.
- Para la creación del mapa como capas en archivos CSV se ha utilizado el editor de mapas gratuito Tiled.
- Para la creación de los distintos componentes gráficos del videojuego se han utilizado las herramientas Aseprite y Photoshop CS6.
- Como compilador del contenido de audio se ha utilizado la herramienta Microsoft Cross-Platform Audio Creation Tool 3 (XACT).
- Como compilador para ciertos aspectos multimedia del videojuego, como las fuentes del texto, se ha utilizado la herramienta complementaria de MonoGame, MonoGame Pipeline.
- Para la realización de la parte ofimática se han utilizado las herramientas Word, Excel y PowerPoint del paquete de ofimática Microsoft Office.

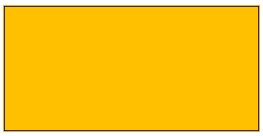
4. Resultados

Durante el transcurso de este apartado, se mostrarán capturas de pantallas del resultado final de este Trabajo Fin de Grado, el videojuego Dragon´s Curse. Teniendo en cuenta su magnitud, este proceso se dividirá en distintos subapartados en los cuales se mostrarán distintos aspectos individuales del producto obtenido.

4.1. Elementos gráficos diseñados

Como se ha mencionado en el apartado "3.6. Planificación temporal del proyecto", los elementos gráficos del videojuego han sido desarrollados íntegramente por el autor, contando este con la ayuda para algunos de estos elementos por parte de su hermano menor, el cual está estudiando el Grado en Bellas Artes de la Universidad de Granada. A continuación, se van a mostrar en varias tablas los distintos elementos gráficos diseñados agrupados en subgrupos, algunos usados finalmente y otros no:

- **Elementos de los menús:** estos elementos, representados en la tabla de la *Figura 94*, son principalmente los distintos botones usados en los menos, añadiendo la caja usada para los menú de lista, la info que se coloca en la esquina inferior derecha para facilitar el manejo por los menús y el fondo utilizado por estos.

Nombre	Elemento	Nombre	Elemento
BotonSeleccionado		BotonSeleccionado Equipo	
BotonSeleccionado Lista		BotonSin Seleccionar	
BotonSin SeleccionarEquipo		BotonSin SeleccionarLista	
BoxLista		InfoMenu	

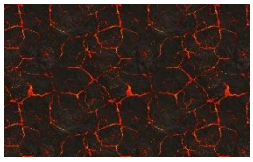
FondoMenu			

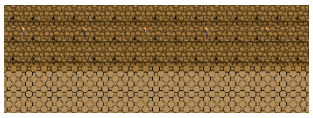
Figura 94. Elementos gráficos para los menús

- **Elementos de los diálogos o interacciones:** se tratan de los elementos necesitados para mostrar los distintos diálogos por pantalla y se tratan de la caja de diálogo y las entidades que pueden aparecer sobre esta, viéndose todos representados en la tabla de la *Figura 95*.

Nombre	Elemento	Nombre	Elemento
BoxDialogo		ArmaduraPoseida (Diálogo)	
Dragon (Diálogo)		PociónDracaria (Diálogo)	
ProtagonistaBrazo Dragon (Diálogo)		Protagonista (Diálogo)	

Figura 95. Elementos gráficos para los diálogos

- **Elementos del combate:** para los combates fueron necesario tres tipos de elementos gráficos diferenciados, representados todos en la tabla de la *Figura 96*: la interfaz del combate, el fondo utilizado para la zona central y los sprites y hojas de animaciones para las distintas entidades que participan en un combate.

Nombre	Elemento	Nombre	Elemento
BotonCombateNo Seleccionado		BotonCombate Seleccionado	
BoxCombate		BoxCombate Enemigos	
FondoCombate Mapa1		ArmaduraPoseida Animacion	
DragonAnimacion (Sin fuego)		DragonAnimacion	
ElementalFuego Animacion		Esqueleto Animacion	
Fantasma Animacion		PerroLobo Animacion	

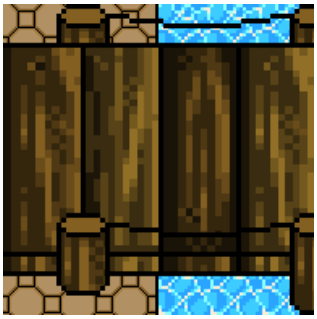
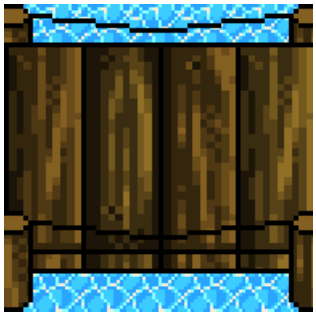
ProtagonistaBrazo DragonCombate		ProtagonistaBrazo NormalCombate	
SoldadoCombate			

Figura 96. Elementos gráficos para los combates

- **Objetos del mapa y entidades del mapa:** estos elementos, representados en la tabla de la *Figura 97*, identifican los distintos objetos con los que el jugador puede interaccionar o colisionar durante su recorrido del mapa. Son además, los que otorgan de mayor variedad al mapa y permiten, algunos, la obtención de objetos. Se incluye en este subgrupo a la hoja de animaciones del personaje protagonista, ya que al fin y al cabo este es una entidad del mapa.

Nombre	**Elemento**	**Nombre**	**Elemento**
Altar		AltarConObjeto	
AltarFinal		AltarFinalCon Objeto	

AntorchaAnimada		AntorchaAnimada Derecha	
AntorchaAnimada Izquierda		ArmaduraPoseida (Mapa)	
CalaveraCara Levantamiento		CalaveraDerecha Levantamiento	
CalaveraEspalda Levantamiento		CalaveraIzquierda Levantamiento	
EsqueletoColgando		CofreAbierto	
CofreAbierto Oxidado		CofreCerrado	
CofreCerrado Oxidado		Columna	

EstanteriaLibros		EstanteriaLibros Derecha	
EstanteriaLibros Izquierda		FuegoAnimacion	
Marca1Animacion		Marca2Animacion	
ParedCuevaMarca1		ParedCuevaMarca2	
PuenteIzquierda		PuenteCentro	

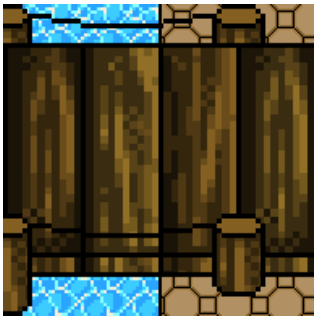
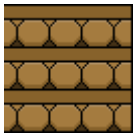
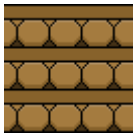
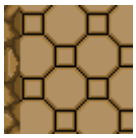
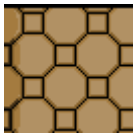
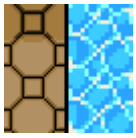
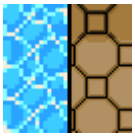
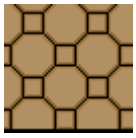
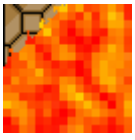
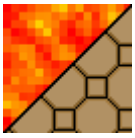
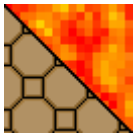
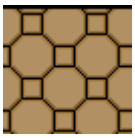
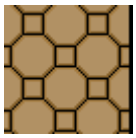
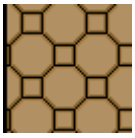
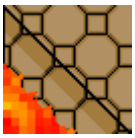
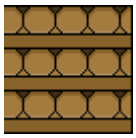
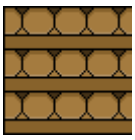
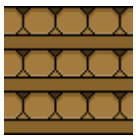
PuenteDerecha		ProtagonistaBrazo DragonSpriteSheet	
ProtagonistaBrazo NormalSpriteSheet			

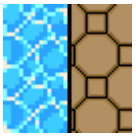
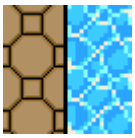
Figura 97. Elementos gráficos para los objetos y entidades del mapa

- **Texturas del mapa:** este último subgrupo de elementos gráficos, representados en la tabla de la *Figura 98*, incluye a todas las texturas utilizadas para diseñar las capas bases del mapa. Gracias al uso de estas, es posible delimitar por qué zonas puede pasar el jugador y cuáles no pertenecen al camino del mapa.

Nombre	Elemento	Nombre	Elemento
LosaRota		EscalerasMarrones IzquierdaDescender	
EscalerasMarrones DerechaDescender		EscalerasMarrones Descender	

BordeLosaArriba		BordeLosaIzquierda	
BordeLosaAbajo		BordeLosaDerecha	
BordeParedAbajo		LosaArribaSombra Izquierda	
LosaArribaSombra Drecha		LosaIzquierdaAgua SombraArriba	
LosaDerechaAgua SombraArriba		LosaAbajo	
LosaLavaAbajo Derecha2		LosaLavaAbajo Derecha1	

LosaLavaArriba Izquierda		LosaLavaArriba Derecha	
LosaArriba		LosaDerecha	
LosaIzquierda		LosaDerechaLava Abajo	
LosaIzquierdaLava Abajo		LosaLavaAbajo Izquierda2	
LosaLavaAbajo Izquierda1		EscalerasMarrones Derecha	
EscalerasMarrones Izquierda		EscalerasMarrones	

LosaLavaAbajo		LosaDerechaAgua	
LosaIzquierdaAgua		Techo	
BordePared Izquierda		BordeParedDerecha	
LosaSombraArriba Derecha		LosaSombra Derecha	
LosaRotaSombra ArribaIzquierda		LosaSombra Izquierda	
LosaSombraArriba Izquierda		BordeParedArriba	

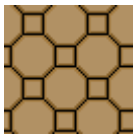
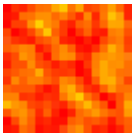
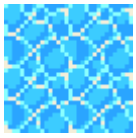
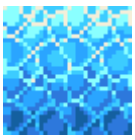
LosaRotaSombra Arriba		LosaSombraArriba	
ParedCueva		Losa	
Lava		Agua1	
Agua2			

Figura 98. Elementos gráficos para las texturas del mapa

4.2. Estados de transición

Figura 99. Resultado final de los estados de transición

Los estados de transición son el resultado más simple del proyecto, como se mencionó en el apartado "3.4.4. Transiciones", ya que únicamente cuentan con un texto y una imagen de fondo, utilizada también por los menús del siguiente apartado. La primera de las capturas de la *Figura 99* representa a la pantalla inicial del videojuego Dragon´s Curse (estado InicioJuego), la segunda a la pantalla de carga de la partida que se va a jugar (estado Cargando) y la última de estas capturas representa la pantalla de finalizacion de la partida (estado GameOver).

4.3. Menús

Los menús del videojuego son agrupados en dos grupos distintos, como se mencionó durante el apartado "3.4.3 Menús": los menús estáticos y los menú de lista. Los primeros se tratan de la mayoría de estos menús, contando únicamente con dos menús de lista que representan las situaciones en las que se muestra el inventario del jugador (el cual se filtra por tipos de objetos). Además, se puede apreciar como para estos menús se han utilizado dos colores similares para representar la selección entre los distintos ítems del menú, siendo el que representa la selección uno más claro y el que representa la no selección uno más oscuro. Siguiendo un orden de izquierda a derecha y de arriba hacia abajo, las capturas de la *Figura 100* representan los siguientes menús: el menú principal (estado MenuPrincipal), el menú de pausa (estado MenuPausa), el menú de equipo (estado MenuEquipo), el menú que permite seleccionar entre cambiar de arma o cambiar alguna pieza de equipamiento (estado MenuArmaOEquipamiento), el menú que permite seleccionar la pieza del equipamiento a cambiar (estado MenuEquipamiento), el menú que permite seleccionar una nueva arma o pieza de equipamiento (estado MenuEquipamientoConcreto), el menú que permite seleccionar el tipo de objetos del inventario a consultar (estado MenuObjetos), el menú de lista para objetos de un tipo concreto (estado MenuObjetosTipo), el menú con las diferentes acciones que realizar con un objeto(estado MenuAccionesObjeto) y el menú que permite aplicar un objeto a un personaje (estado MenuPersonajeObjeto).

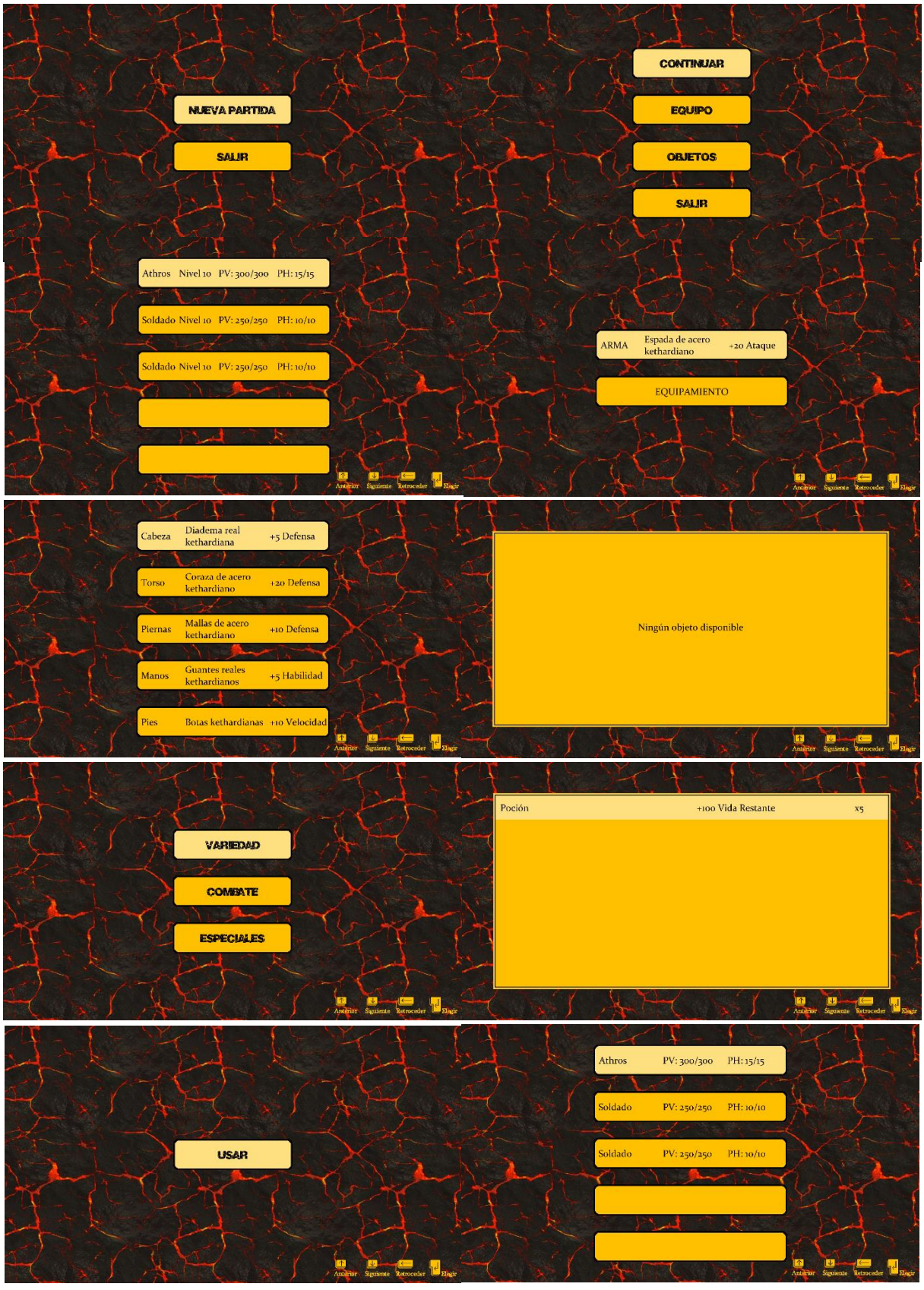


Figura 100. Resultado final de los menús

4.4. Mapa

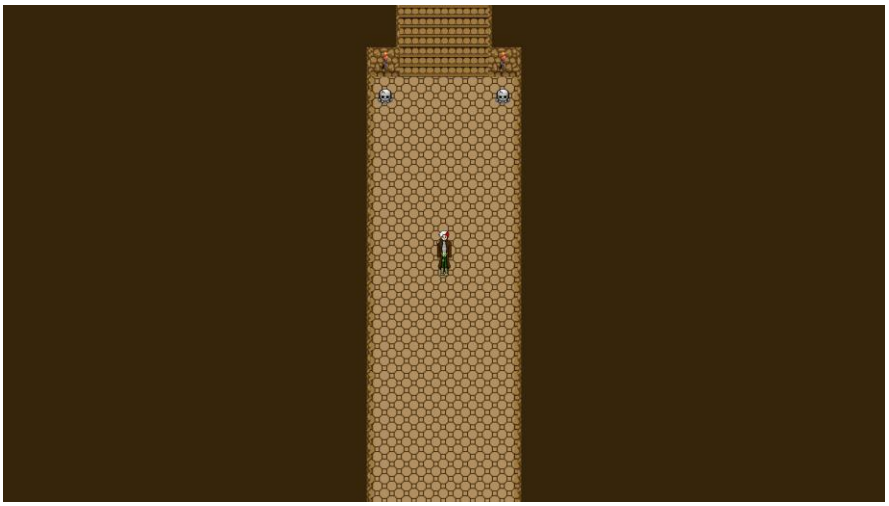


Figura 101. Resultado final del mapa

El recorrido por el mapa es la fase más habitual en el videojuego Dragon´s Curse, representado en la *Figura 101*, y es por tanto el aspecto con un mayor resultado final en cuanto contenido. En este podemos observar tanto texturas y objetos como entidades del mapa ya sean enemigos inactivos o el sprite del personaje principal, siendo este último el único que se mueve a través del mapa. Este movimiento por el mapa es gestionado a través de una cámara que se dedica a comprobar, en función del movimiento del jugador, que elementos pertenecientes al mapa debe mostrar.

4.5. Diálogos/Interacción

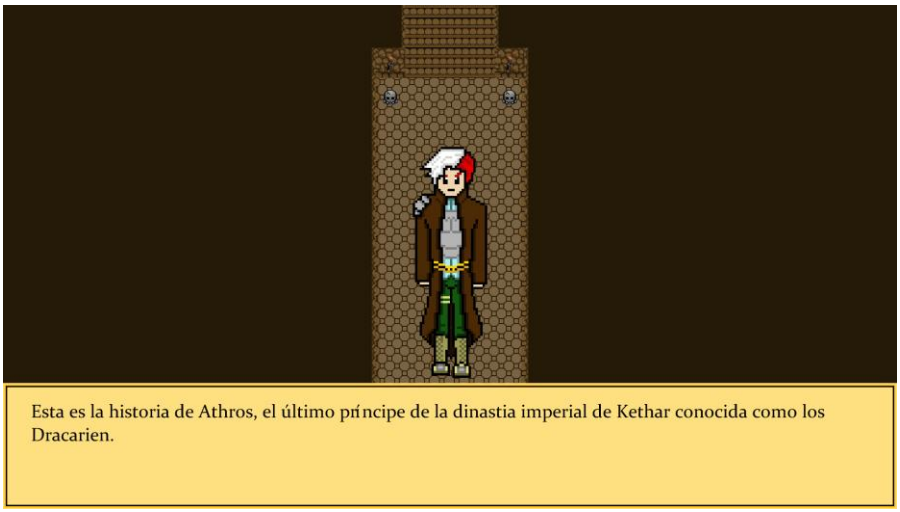


Figura 102. Resultado final de los diálogos

Los diálogos del videojuego Dragon´s Curse, aun siendo una de las fases más sencillas en cuanto a implementación, son una de las fases más importantes debido a que permiten al jugador disfrutar de uno de los puntos fuertes de los videojuegos RPG, la narrativa. Estos diálogos van mostrando letra a letra, aunque el jugador puede autocompletarlos pulsando la barra espaciadora, las distintas conversaciones de los personajes o el narrador junto a un personaje u objeto representativo en la conversación, lo cual se aprecia en la captura de la *Figura 102*.

4.6. Combate

En cuanto a los combates de Dragon´s Curse, se ha obtenido un resultado similar en apariencia a videojuegos como *Pokemon* o *Final Fantasy* en los cuales se presenta un fondo, genérico para cada mapa o localización, junto a una interfaz superior e inferior que permite conocer el estado de las entidades parte del combate y las posibles acciones que puede realizar el jugador. Además, se presentan unos enemigos y personajes enfrentados entre sí y que se tratan de los contendientes del combate. Es importante indicar que en los combates, además de la representación de este en las distintas fases, existen ciertos menús internos que permiten el uso de objetos y se representan como los resultados finales mostrados en el apartado "4.3. Menús". Siguiendo un orden de izquierda a derecha y de arriba hacia abajo, las capturas de la *Figura 103* representan las siguientes fases del combate: el comienzo del turno de un personaje (estado InicioTurno), la elección de una habilidad por parte de un personaje (estado ElegirHabilidad), la elección de un objetivo para un ataque normal o una habilidad (estado ElegirObjetivoAtaque), el menú para elegir un objeto que usar durante el turno de un personaje (estado ElegirObjeto), el menú que permite elegir sobre qué personaje lanzar el objeto seleccionado (estado ElegirPersonajeObjeto), la huida de un combate por parte del jugador (estado Huir), el lanzamiento de las distintas habilidades o ataques de las entidades del combate que estén vivas (estado LanzamientoAtaques), el diálogo de obtención de experiencia al derrotar a los enemigos de un combate (estado Experiencia) y el diálogo que indica la subida de nivel de uno de los personajes del jugador y muestra los incrementos que esto conlleva en sus estadísticas (estado Experiencia).

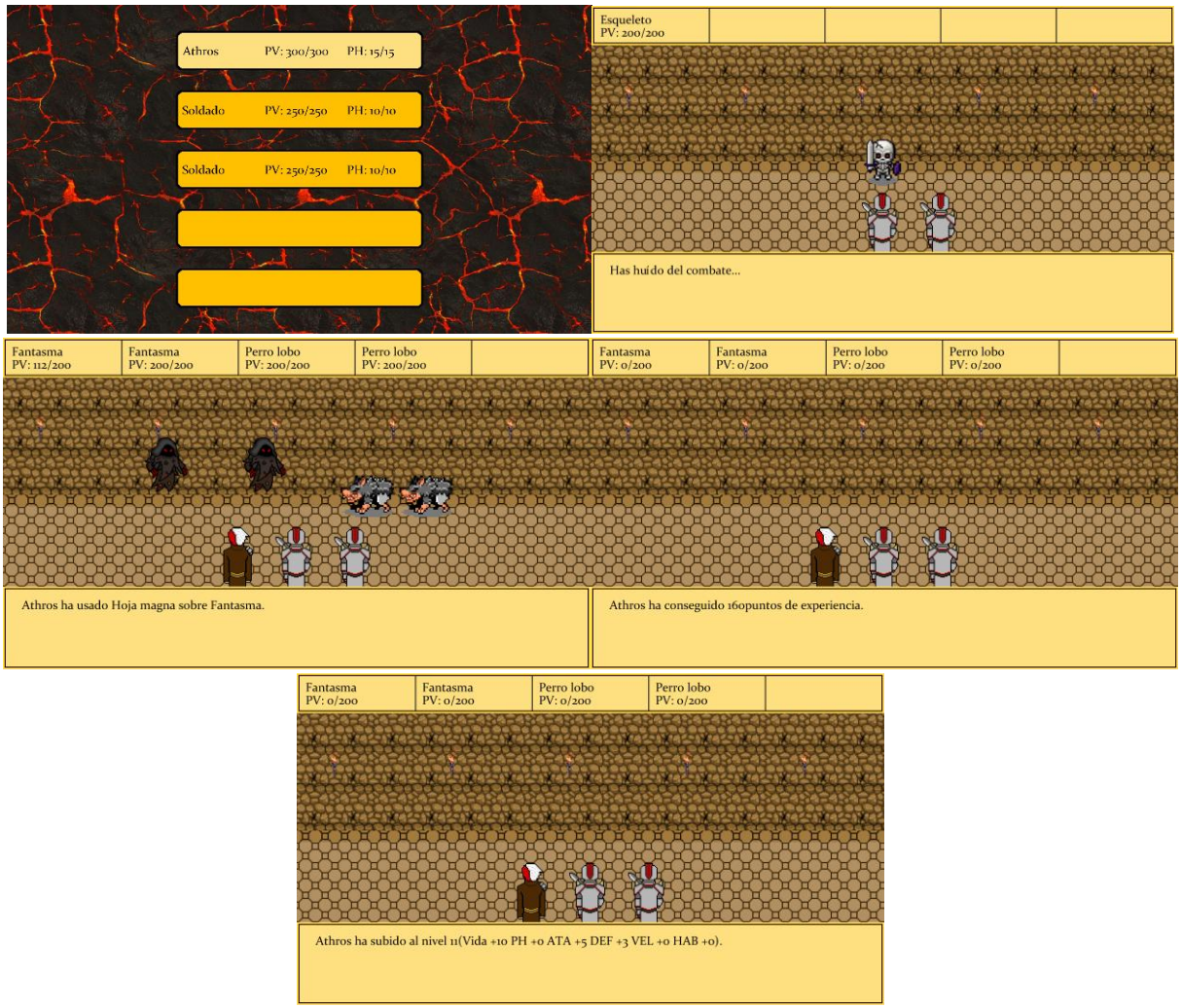


Figura 103. Resultado final de los combates

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Este Trabajo Fin de Grado ha conseguido que el autor haya cumplido un sueño que llevaba persiguiendo desde hace muchos años, llevar a cabo el desarrollo de un videojuego propio con el que intentar aportar su pequeño granito de arena a una industria que le ha dado muchos de los mejores momentos de su vida. Gracias a este proyecto llevado a cabo durante el Trabajo Fin de Grado, ha podido poner a prueba de manera real y práctica los conocimientos adquiridos durante los cuatro años del grado. Además, podría considerarse como un gran paso hacia su meta de conseguir trabajar como desarrollador de videojuegos profesional, ya que en el proceso de realización de este Trabajo Fin de Grado ha podido conocer de primera mano cuales son los pasos a llevar a cabo para desarrollar un videojuego.

Dragon´s Curse, el cual se muestra por última vez en esta memoria en la *Figura 104*, es por tanto una mezcla entre el reflejo de las aspiraciones y metas del autor y el reflejo de los conocimientos adquiridos durante sus estudios universitarios de grado, como puedan ser el uso de patrones de diseño o de estimar una planificación temporal para un proyecto.

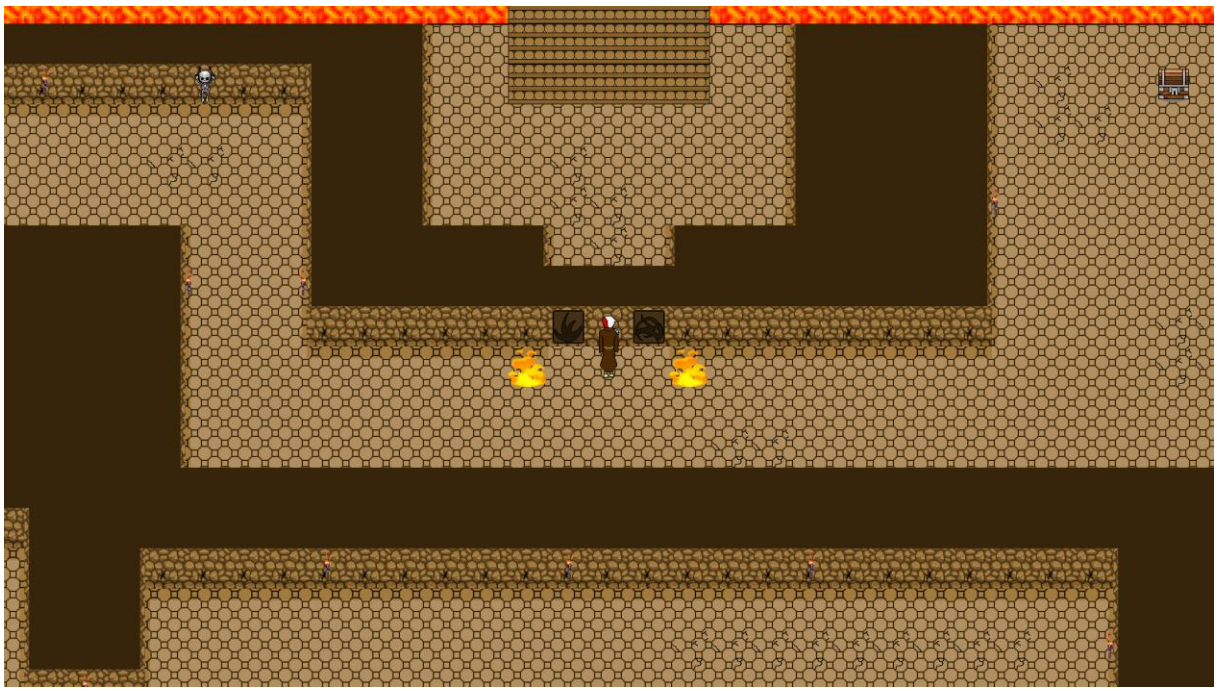


Figura 104. Dragon´s Curse: un videojuego RPG realizado con MonoGame

A continuación, se va realizar una lista de las aportaciones que el autor cree que ha obtenido de la ejecución de este proyecto:

- Actualizar sus conocimientos sobre los videojuegos y su industria.
- Entender que es lo que significa un motor de videojuegos en el desarrollo de estos y para qué es útil.
- Conocimientos sobre el uso de un motor de videojuegos concreto, en este caso MonoGame, con el que volver a desarrollar más proyectos e ideas diferentes.

- Realizar una aplicación práctica real de los conocimientos adquiridos en el Grado en Ingeniería Informática, como puedan ser el uso adecuado de patrones de diseño o la estimación de una planificación temporal.
- Aprender a subdividir los problemas grandes y complicados en problemas abordables y más pequeños, como en el caso del uso de un gestor de estados.
- Mejorar la forma de documentar un proyecto correctamente y redactar de manera formal.

5.2. Trabajo futuro

Dragon´s Curse es realmente un videojuego incompleto que ni de lejos se encuentra en una fase de release, por lo cual cuenta con una cantidad enorme de trabajo futuro para poder abarcar toda la idea inicial del autor y mejorar ciertos errores que han aparecido durante el periodo de pruebas públicas. Este trabajo futuro se podría dividir a su vez en dos grupos diferenciados: las mejoras o correcciones y las ampliaciones de contenido que aporten mayor variedad al gameplay.

Por un lado, las posibles mejoras o correcciones a llevar a cabo serían las siguientes:

- Implementar un sistema de partidas guardadas que permita al jugador guardar la partida desde el menú de pausa para que pueda ser cargada en otro momento desde el menú principal. Esta mejora se podría llevar a cabo mediante el almacenamiento de los distintos elementos de una partida en diferentes archivos XML, a los cuales el estado Juego recurra cuando se inicialice.
- Permitir cambiar el idioma del juego, lo cual sería un punto a favor de la internacionalización y posterior posible distribución del videojuego en el mercado global.
- Permitir cambiar entre distintas resoluciones de pantalla para no limitar en exceso el rango de plataformas que pueden ejecutar Dragon´s Curse, lo cual es un problema respecto a una posible distribución en el mercado.
- Introducir un sistema de gestión de entrada que incluya más controladores permitidos, como puedan ser el ratón o un gamepad, y permita personalizar los controles del videojuego, permitiendo al jugador decidir la manera más cómoda de disfrutar del videojuego.
- Introducir un sistema de preferencias generales del videojuego con el que el jugador pueda regular aspectos relacionados con la imagen o el sonido.
- Mejorar los aspectos relacionados con los combates para que estos se vean más continuos y fluidos. Algunos ejemplos de mejoras en este sentido serían añadir animaciones a los personajes del jugador, identificar los distintos ataques con sonidos individuales o mejorar el sistema de selección para que sea menos brusco.
- Buscar mayor equilibrio en la dificultad del juego, ya que ciertos jugadores de las pruebas han indicado que la dificultad es demasiada en según qué situaciones.

Por otro lado, las ampliaciones que se podrían llevar a cabo para hacer más variado y completo el videojuego serían las siguientes:

- Completar el videojuego a nivel de trama narrativa y mapas, ya que debido al tiempo con el que se contaba se redujo este aspecto al primer mapa únicamente.
- Introducir habilidades que el personaje del jugador puede ejecutar en su recorrido por el mapa, desencadenando así nuevos caminos alternativos en el mapa o desbloqueando el camino principal.

- Añadir al sistema de experiencia y niveles la capacidad de que el personaje pueda aprender nuevos movimientos al subir de nivel, decidiendo si sustituir este por alguno de los que ya dispone.
- Ampliar el catálogo de texturas y objetos del mapa para dotar una mayor variedad en el diseño de este, ya que algunos jugadores de las pruebas han comentado que se les hacía monótono el recorrido del mapa por el diseño de este.
- Añadir un mayor número de sprites para el diálogo, consiguiendo de esta manera que los ya existentes cuenten con mayor variedad de expresiones o tipos.
- Añadir más posibles personajes que se puedan unir a tu equipo y con los cuales conformar un equipo acorde a la manera de jugar de cada jugador.
- Incluir un minimapa que aparezca en el estado Mapa para permitir al jugador orientarse por las zonas de este que ya haya visitado, lo cual ha sido una de las posibles mejoras comentadas por los jugadores de la prueba encuestados.
- Introducirse, por parte del autor, en cuanto a la generación de sonido y música para poder contar con banda sonora y efectos de sonido originales, ya que el apartado sonoro actual ha sido obtenido a través de las distintas páginas de distribución con licencia Creative Commons que se referencia en la bibliografía.

6. Bibliografía

[1] Ruelas, U. (2017). ¿Qué es un motor de videojuegos (game engine)? Recuperado de Coding or Not: https://codingornot.com/que-es-un-motor-de-videojuegos-game-engine. Última consulta: noviembre de 2018.

[2] Depto. Ciencia de la Computación e IA de la Universidad de Alicante. (2015). VIDEOJUEGOS 2. Motores para videojuegos. Recuperado de https://moodle2015-16.ua.es/moodle/pluginfile.php/12347/mod_resource/content/5/vii-02-motores.pdf. Última consulta: noviembre de 2018.

[3] Documentación de Monogame. (s.f.). Recuperado de Monogame: http://www.monogame.net/documentation/. Última consulta: noviembre de 2018.

[4] Reed, A. (2010). Learning XNA 4.0. O´Reilly. Última consulta: noviembre de 2018.

[5] Walker, M. (2011). El framework de XNA. Recuperado de Aprendiendo XNA. Experiencias de un recién llegado: https://aprendiendoxna.wordpress.com/articulos/xna/el-framework-de-xna/. Última consulta: noviembre de 2018.

[6] Debrauwer, L. (2012). Patrones de diseño para C#. Ediciones ENI. Última consulta: noviembre de 2018.

[7] Scyable. (2017). Aprende sobre los distintos géneros de videojuegos actuales. Recuperado de TierraGamer: http://www.tierragamer.com/aprende-los-generos-de-videojuegos/. Última consulta: noviembre de 2018.

[8] Desarrollo Español de Videojuegos (DEV). (2017). Libro Blanco del Desarrollo Español de Videojuegos 2017. Recuperado de http://www.europacreativamedia.cat/rcs_auth/convocatories/libro_blanco_dev_2017.pdf. Última consulta: noviembre de 2018.

[9] Real Academia Española (RAE). (2017). Definición de videojuego. Recuperado de http://dle.rae.es/?id=bmnbNU7. Última consulta: noviembre de 2018.

[10] Pérez Porto, J., & Gardey, A. (2013). Definicion.de: Definición de videojuego. Recuperado de https://definicion.de/videojuego/. Última consulta: noviembre de 2018.

[11] Pérez Porto, J., & Merino, M. (2014). Definicion.de: Definición de controlador. Recuperado de https://definicion.de/controlador/. Última consulta: noviembre de 2018.

[12] Definición de Gamepad. (2013). Recuperado de http://www.gamerdic.es/termino/gamepad. Última consulta: noviembre de 2018.

[13] Gamepad - Wikipedia. (s.f.). Recuperado de https://en.wikipedia.org/wiki/Gamepad. Última consulta: noviembre de 2018.

[14] Ps4 Controller For Pc | gitfsforsubs. (s.f.-b). Recuperado de https://www.hpssociety.info/news/ps4-controller-for-pc.html. Última consulta: noviembre de 2018.

[15] Consola New Nintendo 3DS XL. (s.f.-b). Recuperado de https://www.nintendo.com/es_LA/3ds/new-nintendo-3ds. Última consulta: noviembre de 2018.

[16] Pérez Porto, J., & Merino, M. (2009). Definicion.de: Definición de teclado. Recuperado de https://definicion.de/teclado/. Última consulta: noviembre de 2018.

[17] Urban Dictionary: wasd. (s.f.-b). Recuperado de https://www.urbandictionary.com/define.php?term=wasd. Última consulta: noviembre de 2018.

[18] Pérez Porto, J., & Merino, M. (2013). Definicion.de: Definición de ratón. Recuperado de https://definicion.de/raton/. Última consulta: noviembre de 2018.

[19] Pérez Porto, J., & Merino, M. (2013). Definicion.de: Definición de pantalla táctil. Recuperado de https://definicion.de/pantalla-tactil/. Última consulta: noviembre de 2018.

[20] Wikipedia. (s.f.). Nintendo - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Nintendo. Última consulta: noviembre de 2018.

[21] Wikipedia. (s.f.). Wiimote - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Wiimote. Última consulta: noviembre de 2018.

[22] Playstation. (s.f.). Mando de movimiento Playstation Move. Recuperado de https://www.playstation.com/es-es/explore/accessories/playstation-move-motion-controller/. Última consulta: noviembre de 2018.

[23] Wikipedia. (s.f.). Kinect - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Kinect. Última consulta: noviembre de 2018.

[24] Wikipedia. (s.f.). Playstation Eye - Wikipedia. Recuperado de https://en.wikipedia.org/wiki/PlayStation_Eye. Última consulta: noviembre de 2018.

[25] Teseo, T. (s.f.). Cascos de Realidad Virtual | Teseo Noticias. Recuperado de https://teseo.es/noticias/cascos-de-realidad-virtual/. Última consulta: noviembre de 2018.

[26] Wikipedia. (s.f.). Plataforma de videojuegos - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Plataforma_de_videojuegos. Última consulta: noviembre de 2018.

[27] Pérez Porto, J., & Gardey, A. (2013). Definicion.de: Definición de consola. Recuperado de https://definicion.de/consola/. Última consulta: noviembre de 2018.

[28] MBertrand (2016). La evolución de las videoconsolas en 8 generaciones y más de 40 dispositivos (infografía). Recuperado de https://www.nobbot.com/redes/evolucion-de-las-videoconsolas-infografia/. Última consulta: noviembre de 2018.

[29] Casillas, D. (2018). 5 increíbles consolas híbridas. Recuperado de https://www.metrolatam.com/hub/noticias/2018/04/16/5-increibles-consolas-hibridas.html. Última consulta: noviembre de 2018.

[30] Wikipedia. (s.f.). Arcade - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Arcade. Última consulta: noviembre de 2018.

[31] Wikipedia. (s.f.). Género de videojuegos - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/G%C3%A9nero_de_videojuegos. Última consulta: noviembre de 2018.

[32] What is Gameplay? - Definition from Techopedia. (s.f.-b). Recuperado de https://www.techopedia.com/definition/1911/gameplay. Última consulta: noviembre de 2018.

[33] Deusto Formación. (s.f.). Curso de Desarrollo de Videojuegos en Unity - 1. Orígenes y evolución de los videojuegos. Recuperado de http://campus.deustoformacion.com/Portals/DEUSTO/Minisites/Videojuegos/#module/&item=EPI-0-1. Última consulta: noviembre de 2018.

[34] Wikipedia. (s.f.). Videojuegos de acción - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Videojuego_de_acci%C3%B3n. Última consulta: noviembre de 2018.

[35] Definición de Lucha. (2013). Recuperado de http://www.gamerdic.es/termino/lucha. Última consulta: noviembre de 2018.

[36] Definición de Beat'em Up. (2013). Recuperado de http://www.gamerdic.es/termino/beat-em-up. Última consulta: noviembre de 2018.

[37] Definición de Hack ´n´ Slash. (2013). Recuperado de http://www.gamerdic.es/termino/hack-n-slash. Última consulta: noviembre de 2018.

[38] Definición de Plataformas. (2013). Recuperado de http://www.gamerdic.es/termino/plataformas. Última consulta: noviembre de 2018.
[39] Wikipedia. (s.f.). Videojuegos de disparos - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Videojuego_de_disparos. Última consulta: noviembre de 2018.
[40] Definición de First Person Shooter. (2013). Recuperado de http://www.gamerdic.es/termino/first-person-shooter. Última consulta: noviembre de 2018.
[41] Definición de Third Person Shooter. (2013). Recuperado de http://www.gamerdic.es/termino/third-person-shooter. Última consulta: noviembre de 2018.
[42] Definición de Shoot´em Up. (2013). Recuperado de http://www.gamerdic.es/termino/shoot-em-up. Última consulta: noviembre de 2018.
[43] Definición de Shooter On Rails. (2013). Recuperado de http://www.gamerdic.es/termino/shooter-on-rails. Última consulta: noviembre de 2018.
[44] Wikipedia. (s.f.). Videojuegos de estrategia - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Videojuego_de_estrategia. Última consulta: noviembre de 2018.
[45] Definición de Estrategia en tiempo real. (2013). Recuperado de http://www.gamerdic.es/termino/estrategia-en-tiempo-real. Última consulta: noviembre de 2018.
[46] Definición de Estrategia por turnos. (2013). Recuperado de http://www.gamerdic.es/termino/estrategia-por-turnos. Última consulta: noviembre de 2018.
[47] Wikipedia. (s.f.). Videojuegos de aventura - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Videojuego_de_aventura. Última consulta: noviembre de 2018.
[48] Definición de Aventura conversacional. (2013). Recuperado de http://www.gamerdic.es/termino/aventura-conversacional. Última consulta: noviembre de 2018.
[49] Definición de Aventura gráfica. (2013). Recuperado de http://www.gamerdic.es/termino/aventura-grafica. Última consulta: noviembre de 2018.
[50] Wikipedia. (s.f.). Videojuegos de rol - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Videojuego_de_rol. Última consulta: noviembre de 2018.
[51] Definición de MMORPG. (2013). Recuperado de http://www.gamerdic.es/termino/mmorpg. Última consulta: noviembre de 2018.
[52] Definición de Simulación. (2013). Recuperado de http://www.gamerdic.es/termino/simulacion. Última consulta: noviembre de 2018.
[53] Vallejo, D., & Martín, C. (2015). Volumen 1. Arquitectura del Motor. In D. Vallejo, C. González, & D. Villa (Eds.), Desarrollo de Videojuegos. Un Enfoque Práctico (4ª ed., pp. 7–8). Createspace Independent. Última consulta: noviembre de 2018.
[54] Web de Unity. Recuperado de https://unity3d.com/. Última consulta: noviembre de 2018.
[55] Web de CryEngine. Recuperado de https://www.cryengine.com/. Última consulta: noviembre de 2018.
[56] Web de Unreal Engine. Recuperado de https://www.unrealengine.com. Última consulta: noviembre de 2018.

[57] Vallejo, D., & Martín, C. (2015). Volumen 1. Arquitectura del Motor. In D. Vallejo, C. González, & D. Villa (Eds.), Desarrollo de Videojuegos. Un Enfoque Práctico (4ª ed., pp. 3–5). Createspace Independent. Última consulta: noviembre de 2018.

[58] Deusto Formación. (s.f.). Curso de Desarrollo de Videojuegos en Unity - 3. Diagrama de procesos de un videojuego. Recuperado de http://campus.deustoformacion.com/Portals/DEUSTO/Minisites/Videojuegos/#module/&item=EPI-0-3. Última consulta: noviembre de 2018.

[59] Sánchez, C. (2016). ¿Qué plataforma de octava generación manda en Alemania? Recuperado de https://juegosadn.eleconomista.es/que-plataforma-de-octava-generacion-manda-en-alemania-no-93333/. Última consulta: noviembre de 2018.

[60] Neurogadget. (2016). Nintendo 3DS vs Sony PS Vita: Which One Should You Choose? Recuperado de https://neurogadget.net/2016/11/13/nintendo-3ds-vs-sony-ps-vita-one-choose/46284. Última consulta: noviembre de 2018.

[61] Amazon. (s.f.). Amazon.com: Nintendo Switch with Joy-Con Pick You Own System Bundle Options: Edition, Color, Games, Extra Controller, Case: Nintendo: Computers & Accessories. Recuperado de https://www.amazon.com/Nintendo-Switch-System-Bundle-Options/dp/B077CVQVN1. Última consulta: noviembre de 2018.

[62] MercadoLibre. (2017). Maquina Multujuegos Arcade(1720). Recuperado de https://articulo.mercadolibre.com.ar/MLA-620444565-maquina-multujuegos-arcade1720-_JM?quantity=1. Última consulta: noviembre de 2018.

[63] BBC Mundo. (2017). 3 claves que hicieron de Pac-Man uno de los videojuegos más exitosos de la historia. Recuperado de https://www.laprensa.com.ni/2017/01/31/tecnologia/2175047-3-claves-que-hicieron-de-pac-man-uno-de-los-videojuegos-mas-exitosos-de-la-historia. Última consulta: noviembre de 2018.

[64] Wikipedia. (s.f.). Sinclair ZX Spectrum. Recuperado de https://es.wikipedia.org/wiki/Sinclair_ZX_Spectrum. Última consulta: noviembre de 2018.

[65] GamePressure. (2017). Wolfenstein 3D GAME MOD Gerolf's SS v.270216 - download. Recuperado de https://www.gamepressure.com/download.asp?ID=60625. Última consulta: noviembre de 2018.

[66] Amazon. (s.f.-b). Windows 8 - Oculus Rift and Touch Controllers [Bundle]: Amazon.es: Videojuegos. Recuperado de https://www.amazon.es/Windows-Oculus-Touch-Controllers-Bundle/dp/B073X8N1YW/ref=sr_1_1?ie=UTF8. Última consulta: noviembre de 2018.

[67] Goslin, A. (2018). League of Legends World Championship 2018: Qualified teams, seeding and more. Recuperado de https://www.riftherald.com/lol-worlds/2018/8/23/17718838/worlds-2018-league-qualified-teams-seeds. Última consulta: noviembre de 2018.

[68] Rose, M. (2013). It's official: XNA is dead. Recuperado de http://www.gamasutra.com/view/news/185894/Its_official_XNA_is_dead.php. Última consulta: noviembre de 2018.

[69] Microsoft. (2018). ¿Qué es una aplicación para Plataforma universal de Windows (UWP)? - UWP app developer. Recuperado de https://docs.microsoft.com/es-es/windows/uwp/get-started/universal-application-platform-guide. Última consulta: noviembre de 2018.

[70] Khronos Group. (s.f.). OpenGL News Archives. Recuperado de https://www.opengl.org/documentation/. Última consulta: noviembre de 2018.

[71] Microsoft. (s.f.). DirectX 11 | Descargar gratis. Recuperado de https://www.directx.com.es/. Última consulta: noviembre de 2018.

[72] Bottiau, D. (2018). Create a game with MonoGame using C#/XAML (Windows Store app). Recuperado de https://medium.com/@dbottiau/create-a-game-with-monogame-using-c-xaml-windows-store-app-603396408800. Última consulta: noviembre de 2018.

[73] Whitaker, R. B. (s.f.). Monogame - Introduction To 2D Graphics - RB Whitaker's Wiki. Recuperado de http://rbwhitaker.wikidot.com/monogame-introduction-to-2d-graphics. Última consulta: noviembre de 2018.

[74] Wikipedia. (s.f.). Cross-platform Audio Creation Tool - Wikipedia. Recuperado de https://en.wikipedia.org/wiki/Cross-platform_Audio_Creation_Tool. Última consulta: noviembre de 2018.

[75] Refactoring Guru. (s.f.). Large Class. Recuperado de https://refactoring.guru/smells/large-class. Última consulta: noviembre de 2018.

[76] Wikipedia. (s.f.-f). State (patrón de diseño) - Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/State_(patr%C3%B3n_de_dise%C3%B1o) . Última consulta: noviembre de 2018.

[77] Refactoring Guru. (s.f.). State. Recuperado de https://refactoring.guru/design-patterns/state. Última consulta: noviembre de 2018.

[78] Template Method .NET Design Pattern in C# and VB - dofactory.com. (s.f.). Recuperado de https://www.dofactory.com/net/template-method-design-pattern. Última consulta: noviembre de 2018.

[79] Refactoring Guru. (s.f.). Template Method. Recuperado de https://refactoring.guru/design-patterns/template-method. Última consulta: noviembre de 2018.

[80] Refactoring Guru. (s.f.). Factory Method. Recuperado de https://refactoring.guru/design-patterns/factory-method. Última consulta: noviembre de 2018.

[81] Corral, A.L. (s.f.). Patrones de Diseño I. In UAL (Ed.), Temario de Desarrollo Rápido de Aplicaciones (pp. 78–88). Última consulta: noviembre de 2018.

[82] Urdiales, A.(2018,27 noviembre).Cuestionario de feedback Dragon's Curse.Recuperado de https://docs.google.com/forms/d/e/1FAIpQLScPPKcQK2ifVo6ED47QbOwe0UzUwFOYOgog8t0k-1VLvB-3Dw/viewform?usp=send_form. Última consulta: noviembre de 2018.

[83] Definición de Triple A. (2013). Recuperado de http://www.gamerdic.es/termino/triple-a. Última consulta: noviembre de 2018.

[84] Fuente de las fuentes de letra. Recuperado de https://www.dafont.com/es/. Última consulta: noviembre de 2018.

[85] Fuente del apartado sonoro. Recuperado de http://dig.ccmixter.org. Última consulta: noviembre de 2018.

[86] Fuente del apartado sonoro. Recuperado de http://soundimage.org. Última consulta: noviembre de 2018.

[87] Fuente del apartado sonoro. Recuperado de https://hartwigmedia.com. Última consulta: noviembre de 2018.

[88] Fuente del apartado sonoro. Recuperado de http://www.flashkit.com. Última consulta: noviembre de 2018.

[89] Fuente de ciertos recursos gráficos que sirvieron como inspiración. Recuperado de https://opengameart.org. Última consulta: noviembre de 2018.