Rock Zombie :

Diseño, Desarrollo y Comercialización
de un videojuego independiente

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA EN INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

Rock Zombie: Diseño, Desarrollo y Comercialización de un videojuego independiente

Miguel José García Corchero

Septiembre, 2014

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

Rock Zombie: Diseño, Desarrollo y Comercialización de un videojuego independiente

Autor: Miguel José García Corchero
Director: Carlos González Morcillo

Septiembre, 2014

Miguel José García Corchero
E-mail : miguelgarciacorchero@gmail.com
Teléfono : +34 635 123 380
Website : http:www.quaternionstudio.com

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

A mis padres Miguel y Encarnación,
mi hermana Celia
y mi pareja Tamara.

RESUMEN

La industria de desarrollo de videojuegos es a día de hoy más fructífera económicamente que la industria del cine y la música juntas. Según el *«Libro blanco del desarrollo Español de videojuegos»* realizado por el *«DEV»* (Asociación de Desarrolladores Españoles de Videojuegos) con el apoyo de *«ADESE»* (Asociación de Desarrolladores y Editores de Software de Entretenimiento), donde se analiza el estado actual del sector, se estima que la facturación del sector crecerá hasta el año 2017 a una tasa anual del 23,7 %. Esto supone que el sector alcanzará en dicho ejercicio un volumen de negocio superior a los 723 millones de euros sólo en nuestro país. Este hecho, junto con el auge de los mercados de venta digitales, ha traído modelos de negocio que permiten a desarrolladores de todo el mundo autoeditar su software. Este nuevo campo de juego es conocido como *«desarrollo de videojuegos independientes»*, y en los últimos años ha ganado mucha popularidad entre los usuarios gracias a fenómenos como los *Humble Bundle* o el auge de plataformas de venta como *Steam*.

El presente Trabajo Fin de Grado surge con el objetivo de crear un videojuego con las características necesarias para ser un producto atractivo orientado a los potenciales clientes objetivo de las diversas plataformas de venta de videojuegos. Para ello abordarán las fases de diseño, desarrollo y comercialización del mismo.

En la fase de diseño se cubrirá el "qué" se quiere realizar, orientando la construcción de un producto hacia un público objetivo concreto.

En la fase de desarrollo se abordará la resolución en sí de la problemática de "cómo" implementar un videojuego que sea funcional en múltiples plataformas con recursos *hardware* limitados.

Por último se planteará la fase de comercialización, donde se discutirán las diferentes plataformas de distribución digital y se presentará un plan de negocio que permita hacer de este desarrollo un producto comercial viable.

ABSTRACT

Nowadays the video game industry is bigger than film and music industry together. According to the "White Paper of the Spanish game development", made by "DEV" (Asociación de Desarrolladores Españoles de Videojuegos) and supported by "ADESE" (Asociación de Desarrolladores y Editores de Software de Entretenimiento), it is estimated that the sector's turnover will grow at a annual rate of 23.7 % until 2017. This means that the sector will reach 723 million of euros that year in Spain. This fact, and the fast rising of digital markets, has brought new business models that allow to worldwide developers the opportunity of publishing their own software and video games.

This new field is known as "independent video game development", and in recent years has gained a great amount of popularity among users due to successful initiatives such as the Humble Bundles or Steam.

This Project what created with the objective of making a concrete video game with the requirements of being an attractive product for the differents kinds of customers in each game platform. To achieve that it will focus on the phases of design, development and commercialization.

In the first phase, the design of the product to target a specific audience will be covered.

The second phase (development) is focused on the resolution of the problem, solving the problems related to the implementation of the game. The game must run in multiple hardware platforms with limited computing resources.

Finally, in the commercialization stage, we will pay attention to the specific platforms and markets where the game will be published. This step also offers a careful discussion and final remarks about the different business models and marketing plans to make the development a profitable investment.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

A mis padres les agradezco haberme permitido estudiar esta carrera con su apoyo económico y anímico, sin el cual no hubiera sido posible realizarla.

A mi hermana por estar siempre ahí, por haberme acompañado a charlas y a cursos muy lejos de casa.

A Tamara por su apoyo y conversaciones motivadoras en los duros momentos del desarrollo de este videojuego que me han ayudado a que consiguiera terminarlo. También por ser la primera persona que prueba mis juegos y la que me da opiniones sinceras sobre los mismos.

Por último le agradezco a Carlos haberme enseñado una valiosa lección que fui aprendiendo ya desde mis primeros años de universidad cuando empecé a trabajar con él. No importa lo bueno que sea el trabajo que le lleves, él siempre es bastante crítico. Al principio resulta molesto, después empiezas a ser más crítico con tu propio trabajo obteniendo mejores resultados en el proceso y por último aprendes la gran lección : “En cada cosa que hagas se crítico con tu trabajo y no te conformes con un buen resultado, esfuérzate para conseguir el resultado definitivo”.

CAPÍTULO 1. INTRODUCCIÓN

El auge de los dispositivos móviles, las nuevas formas de control, así como los nuevos mercados de venta de aplicaciones digitales han traído modelos de negocio que permiten que desarrolladores de todo el mundo puedan autoeditar su software pudiendo obtener rendimiento económico de ello. Estos recientes fenómenos dibujan el campo de juego del *«desarrollo de videojuegos independientes»*.

Si hablamos de cifras, según el informe anual de *«La sociedad de la Información en España»* realizado por *Telefónica* [49], el 63 % de los usuarios de móvil en España tienen *smarphones*. En 2012 el 43 % de los internautas utilizaron su dispositivo móvil para conectarse a *internet*, lo que supone un incremento de 210 % respecto a 2011.

Por otro lado la industria de los videojuegos se cristaliza como un sector que a día de hoy supera en ingresos a la del cine, teniendo recientemente casos como el lanzamiento de *«GTA V»*, un videojuego en el que se invirtieron 250 millones de dolares y 4 años de trabajo durante los cuales *RockStar*, la empresa desarrolladora dio trabajo a multitud de trabajadores del sector, y tras salir a la venta recaudó 800 millones de dolares en sus primeras 24 horas [89].

1.1. QUÉ ES UN VIDEOJUEGO

Un videojuego [15] es un software especial en el que el usuario interactúa, por medio de un controlador, sobre un dispositivo capaz de representar información de forma visual. Este dispositivo *hardware* sobre el que se ejecuta este software es conocido como “plataforma” y puede ser desde una computadora, una máquina recreativa, una videoconsola o un *smartphone*.

El objetivo principal de un videojuego puede ser múltiple. Inicialmente fueron concebidos como forma de diversión o entretenimiento aunque a día de hoy se utilizan en diversas áreas como la educación, como medio de expresión para contar historias, e incluso como si-

muladores para el entrenamiento de pilotos en vuelos espaciales. Últimamente también están ganando mucha importancia los "*Serious Games*", cuya aplicación en medicina permite ayudar en terapias de rehabilitación, o en el aprendizaje de personas con dificultades cognitivas.

Al dispositivo de entrada usado para manejar el videojuego se lo conoce como controlador, y varía dependiendo de la plataforma, pudiendo ser un teclado, un ratón, un *joystick*, una pantalla táctil, un dispositivo de captura de movimientos mediante sensores e incluso un casco de realidad virtual con capacidades de *tracking*.

Por lo general, los videojuegos emplean diversas formas de comunicación con el usuario. El principal método es mediante el uso de información visual en forma de imágenes que se dibujan a gran velocidad sobre el dispositivo de representación para conseguir la sensación de movimiento. También hacen uso de otros elementos para conseguir una inmersión adecuada del usuario empleado el uso de sonidos, vibración de los controladores de entrada, etc.

1.2. ELEMENTOS ESTRUCTURALES

Un videojuego puede contemplarse desde dos perspectivas principales [15]:

- **Perspectiva del cliente :** El cliente de este tipo de software no aprecia el producto por sus capacidades técnicas o el arquitectura tecnológica utilizada para desarrollarlo, sino que mide su calidad por lo que percibe como producto final, valorando aspectos como:

  Jugabilidad : Se espera que el juego tenga mecánicas jugables que nos plantean un reto, que nos hagan pensar o esforzarnos para conseguir una mejor marca o forma de superar los desafíos propuestos.

  Calidad gráfica / sonora : El aspecto visual o sonoro percibido es muy importante, pues son los estímulos principales que el jugador recibe. En general, cuanto mejor sean sus gráficos y más calidad artística tengan, mejor valorado será el videojuego.

  Contenido : Si el videojuego contiene muchos niveles, variedad de situaciones, elementos a coleccionar, modificar o con los que interactuar será mejor percibido que si su contenido es más limitado.

  Guión / Historia : No es imprescindible que un videojuego cuente una historia, pero si lo hace y la historia es buena será un elemento de enganche para que el jugador continúe queriendo saber más y por tanto quiera continuar avanzando en el videojuego.

Interactividad : Todo videojuego permite una interacción limitada con el entorno o elementos del mismo. El rango varía entre muy alta, haciendo que lo percibamos como una simulación de un entorno virtual en la que estamos inmersos, o limitada haciendo que lo percibamos como si viéramos una película.

- **Perspectiva tecnológica :** Si miramos el videojuego desde la perspectiva tecnológica nos encontramos con que está formado por diversos componentes.

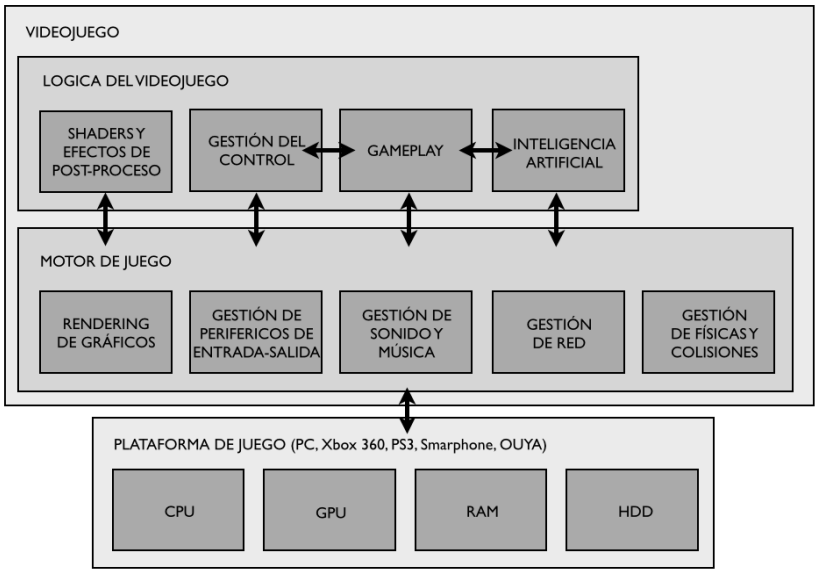


Figura 1.1: Componentes de un videojuego

Motor del videojuego : El motor de videojuego [12] es la parte del mismo que se comunica con el *hardware* de la plataforma sobre la que se está ejecutando. Es el encargado de proporcionar los mecanismos de comunicación con los controladores, de la lectura y escritura de información sobre el almacenamiento secundario, de la reproducción de sonidos o de la renderización de gráficos para su visualización sobre un dispositivo de representación. Si el motor de videojuego es bueno las capacidades disponibles para implementar una buena lógica de videojuego serán mejores.

Lógica del videojuego : La lógica del videojuego es el código del videojuego que modela el comportamiento. Mediante la lógica del videojuego creamos unas reglas, que haciendo uso del motor de videojuego, permiten procesar las entradas del jugador y devolver una respuesta a modo de información visual o sonora. Para ello se emplea el uso de recursos externos o *assets*. Si la lógica de videojuego es buena el jugador se

divertirá más o percibirá mayor interactividad con los elementos del mismo.

***Assets* :** Llamamos *asset* a todo el contenido generado para ser utilizado en el videojuego. Es decir: imágenes de texturas, modelos tridimensionales, animaciones de huesos, textos, sonidos, música o todo elemento pero que por sí sólo, sin la lógica del videojuego, son algo estático. Si las imágenes, sonidos, animaciones y los modelos 3D son buenos, entonces se percibirá como un contenido individual de más calidad sobre un conjunto mayor. Es importante guardar una coherencia de estilos y uniformidad entre los *assets* para que el resultado final pueda considerarse como una cohesión armónica de elementos y no como una mezcla de recursos sueltos de diferentes estilos.

1.3. IMPACTO SOCIO-ECONÓMICO

A lo largo del año se lanzan al mercado numerosos videojuegos. Algunos recuperan su inversión inicial y muchos otros tienen pérdidas para la compañía que los comercializa. No obstante el éxito de unos pocos tapa el fracaso de otros muchos.

Si miramos este hecho desde la perspectiva de un trabajador de una empresa podemos decir que el mero hecho de que se hagan videojuegos es bueno pues justifica ese puesto de trabajo durante el desarrollo del mismo, independientemente de si se recupera o no la inversión inicial.

Mirándolo desde la perspectiva de la empresa que comercializa el videojuego es una inversión no carente de riesgo. Muchos expertos [64] dicen que de cada 10 videojuegos que se hacen, 3 o 4 de ellos son rentables por sí mismos, y sólo 1 o 2 de ellos aporta capital suficiente como para poder financiar los siguientes videojuegos de esa empresa.

La realidad es que se trata de una industria dura y muy competitiva. Sin embargo, existen grandes casos de éxito como [30]:

- ***«Candy Crush Saga»* [29] (*King*) / Móviles :** En 2013 la compañía presento sus ingresos de 1800 millones, que se deben en un 78 % a lo que genera el juego con sus 93 millones de usuarios diarios.
- ***«Clash of Clans»* [70] (*SuperCell*) / Móviles :** Con sólo tres juegos en el mercado para móviles, estos finlandeses que fundaron la empresa en el 2010 generan 2,4 millones de dólares al día gracias a su juego.

- ***«Angry Birds»*** **[71] (*Rovio*) / Móviles :** Este caso es particularmente especial, pues tras 6 meses sin casi repercusión y sin generar ingresos este juego obtuvo un crecimiento increíble a raíz de una inversión en marketing. Consiguiendo en 2012, 80 millones de beneficio sobre los 210 millones de ingresos.

- ***«Minecraft»*** **[69] (*Mojang*) / *PC* :** Comenzó como un proyecto de una única persona que empezó a vender el juego cuando aún se trataba de una *beta*. En 2013 cerró el año fiscal con 128 millones de dolares sobre el beneficio de 322 millones de dólares.

- ***«GTA V»*** **[72] (*Rockstar*) / Consolas :** Este caso es de una empresa consolidada en el sector, que generó un videojuego durante 4 años de desarrollo a un coste de unos 250 millones de dólares y que sólo 24 horas después de ponerlo a la venta ya habían generado 800 millones de dólares.

- ***«Mario Kart Wii»*** **[73] (Nintendo) / Consolas :** Este caso también es de una empresa consolidada en el sector. A día de hoy se trata de uno de los videojuegos más vendidos de todos los tiempos con 35 millones de copias.

1.4. JUSTIFICACIÓN

El objetivo de este Trabajo de Fin de Grado (TFG), es la creación de un videojuego independiente completo, abordando las 3 macro-etapas en su creación: Diseño, Desarrollo y Comercialización. Como objetivo implícito del TFG se establece maximizar la rentabilidad económica del producto, tratando de amortizar el esfuerzo realizado.

El proceso de creación en 3 etapas se enfocará en dos áreas principales.

- 1. Diseño y desarrollo de un producto que sea percibido por los usuarios como un videojuego de calidad. Para ello habrá que estudiar los factores que influyen en las valoraciones de los usuarios.

- 2. Adopción de un modelo de negocio y de los canales de distribución adecuados a las características del producto. Se realizará un estudio y un plan de marketing que haga llegar el videojuego a la mayor cantidad de público potencial posible.

1.5. ESTRUCTURA DEL DOCUMENTO

Este documento se ha redactado respetando la normativa de trabajos de fin de grado de la Escuela Superior de Informática (UCLM) y se encuentra dividido en los siguientes capítulos:

Capítulo 2 : Objetivos

En este capítulo se recoge el objetivo general y los subobjetivos concretos de este trabajo de fin de grado.

Capítulo 3 : Antecedentes

En este capítulo se realizan los antecedentes sobre la evolución histórica y los tipos de videojuegos, herramientas y métodos de desarrollo y comercialización más utilizados en la actualidad.

Capítulo 4 : Método de trabajo

En este capítulo se describe la metodología elegida para llevar a cabo el desarrollo de este proyecto y se describe cuáles han sido las herramientas utilizadas para su realización.

Capítulo 5 : Resultado final

En este capítulo se describe el diseño e implementación del sistema, detallando los problemas encontrados y las soluciones aportadas. El capítulo se centra en los diferentes sistemas que componen el videojuego y las relaciones entre los mismos, además de detallar los patrones empleados y las soluciones tecnológicas adoptadas. También se resume la evolución del proyecto y se analizan los costes asociados al mismo.

Capítulo 6 : Conclusiones y propuestas

En este capítulo se analiza el trabajo realizado y se comentan algunas propuestas de posibles líneas de trabajo futuro.

Capítulo 7 : Bibliografía

En este capítulo se añaden las fuentes bibliográficas consultadas por el alumno para la realización del presente TFG.

Apéndices :

En los diferentes apéndices de este documento se adjunta el siguiente material:

- A. **Manual del videojuego** : El manual del videojuego contiene una breve descripción de cómo instalarlo en cada uno de los sistemas operativos compatibles. También se incluye un listado de los comandos empleados y las teclas asociadas a cada uno de ellos.
- B. **Imágenes finales del videojuego** : En este anexo se adjunta una colección de imágenes del videojuego donde puede apreciarse el resultado final.
- C. **Organización del código fuente** : En este anexo se describe como se ha estructurado el código fuente que puede encontrarse en el *DVD* adjunto.
- D. ***Assets* externos utilizados** : En este anexo se describe un listado de los *assets* de terceros que han sido adquiridos para incluirse en el videojuego.
- E. **Proyectos previos realizados por el alumno** : En este anexo se describen algunos de los proyecto realizados por el alumno en el ámbito de la programación gráfica, videojuegos o infografía. La realización de estos proyectos anteriores ha sido clave para disponer de la experiencia necesaria que ha permitido que el presente videojuego tenga la calidad que se ha logrado.

CAPÍTULO 2. OBJETIVOS

En este capítulo se enumeran y describen los objetivos y subobjetivos planteados para el presente Trabajo Fin de Grado.

2.1. OBJETIVO GENERAL

El objetivo principal de este Trabajo Fin de Grado puede definirse como la creación de un videojuego independiente[1] completo, abordando las 3 macro-etapas en su creación: Diseño, Desarrollo y Comercialización.

Cada una de estas etapas definen a su vez un objetivo principal que tiene influencia en las etapas posteriores. Además el objetivo principal de cada etapa se concreta en una serie de subobjetivos que se detallan en los siguientes apartados.

2.2. SUBOBJETIVOS ESPECÍFICOS

2.2.1. Etapa de diseño

El objetivo principal de esta etapa es la definición de un documento de diseño del videojuego que asegure la construcción de un producto de calidad y que minimice los posibles riesgos y problemas en las etapas posteriores.

El objetivo principal se alcanza en base a la consecución los siguientes subobjetivos:

- **Diseño de videojuego basado en niveles :** El videojuego desarrollará una historia conforme se avanza en los diferentes niveles, en los cuales el jugador enfrentará múltiples situaciones, como superar niveles normales, niveles con situaciones especiales y encuentros con jefes finales que harán uso de mecánicas específicas.

[1] Un videojuego independiente (o *«indie game»*) es aquel que es creado por un individuo o un grupo pequeño sin el apoyo financiero de productores externos o distribuidores

- ***Items* coleccionables y logros desbloqueables :** El videojuego contará con elementos secretos que se podrán coleccionar. También se incluirán desafíos que se le plantearán al jugador a modo de logros desbloqueables. Estos dos elementos contribuyen a aumentar la complejidad y el número de horas jugables del videojuego.

- **Estudio de publico objetivo:** Se realizará un estudio que permita definir el tipo de género y características deseables en un producto de este tipo para orientar el producto a un publico objetivo concreto, maximizando así que el producto encaje con el perfil de ese tipo de cliente.

- **Análisis de productos similares / competencia:** Se realizará un análisis de productos similares y de la competencia que permita determinar cuales son los puntos fuertes de esos productos. De este modo se determinará en que aspectos poner más atención de modo que se pueda garantizar que el producto a desarrollar sea viable comercialmente.

2.2.2. Etapa de desarrollo

El objetivo principal de esta etapa es la implementación de un producto software a partir del documento de diseño del videojuego. Este software debe funcionar cumpliendo con los estándares de calidad adecuados en las plataformas objetivo para las que ha sido planteado, llevando a cabo las optimizaciones o adaptaciones necesarias para garantizar el correcto funcionamiento.

El objetivo principal se alcanza en base a la consecución los siguientes subobjetivos:

- **Uso de gráficos tridimensionales :** Se emplearán modelados *3D* con animación mediante esqueletos y uso de texturas mediante la técnica de *UV-Mapping*.

- **Respuesta en tiempo real :** El videojuego debe reaccionar a la entrada del usuario con una frecuencia adecuada para garantizar la sensación de movimiento y fluided necesarias. Para ello debe estar optimizado para alcanzar la frecuencia de 30 o 60 *frames* de renderizado por segundo dependiendo de la plataforma seleccionada.

- **Multiplataforma :** El videojuego se diseñará pensando en que sea multiplataforma, del tal forma que en el futuro se comercialice en diversas plataformas diferentes tras pasar por una etapa de adaptación al *hardware* final de dicha plataforma.

- **Uso de *shaders* programables :** Esta funcionalidad se considera el estándar a día de hoy en lo que a renderizado de videojuegos se refiere. Para ello se creará una biblioteca

de *shaders* compatibles con los chips de tarjetas gráficas de ordenadores de escritorio pero también compatibles con chips gráficos de dispositivos móviles. El uso de este elemento contribuirá a obtener un acabado visual más elaborado.

- **Efectos *postproceso* avanzados :** El uso de efectos como *Bloom*, *Depth Of Field* o la corrección de color de la imagen resultante del renderizado se considera también como un estándar de los videojuegos. Para ello se emplearán técnicas específicas que permitan garantizar el uso fluido de estos efectos en ordenadores de escritorio y dispositivos móviles.

- **Sombras dinámicas :** También se emplearán los mecanismos necesarios para garantizar el uso de iluminación y sombras dinámicas en tiempo real.

2.2.3. Etapa de comercialización

El objetivo principal de esta etapa es la puesta a la venta de un producto software. Para ello es necesario desarrollar un plan de negocio que cubra aspectos como las campañas de marketing y distribución necesarias para permitir hacer llegar el producto a los potenciales clientes.

El objetivo principal se alcanza en base a la consecución los siguientes subobjetivos:

- **Estudio de plataformas de publicación :** Se llevará a cabo un estudio para determinar cuales son los canales de venta más adecuados, eligiendo también las plataformas *hardware* en la que se va a publicar.

- **Elección del modelo de negocio :** Para cada plataforma objetivo se determinará el modelo de negocio más adecuado en función de los hábitos de los usuarios potenciales.

- **Plan de marketing :** Se realizará un plan de acciones de marketing que permitan dar a conocer el videojuego entre los usuarios de distintas plataformas.

- **Distribución y presentación en medios :** Se realizará la puesta a la venta en cada uno de los canales de venta seleccionados y se pondrá en marcha el plan de marketing. Como resultado de estas acciones se tendrá presencia en prensa para apoyar la comercialización del videojuego.

CAPÍTULO 3. ANTECEDENTES

En este capítulo se introducirán los campos y las tecnologías relacionadas con el videojuego como industria y se sentarán bases que permitirán entender las referencias y la terminología de este documento.

Para continuar con el enfoque seguido en la elaboración del presente Trabajo fin de Grado, a continuación se realizará una división de el contenido en las secciones de Diseño, Desarrollo y Comercialización.

3.1. INTRODUCCIÓN HISTÓRICA

Con origen en la década de los 40, tras el fin de la Segunda Guerra Mundial, se comenzaron a desarrollar las primeras computadores programables como el *«ENIAC»*. Estas computadores permitían implementar programas y algunos de los que se desarrollaron en su momento recreaban juegos como el ajedrez.

Más adelante surgieron lo que conoceríamos como los primeros videojuegos modernos [28] un poco antes de la década de los 60, cuando *William Higinbotham*, un ingeniero norteamericano que había participado en el *Proyecto Manhattan*, presentó un juego llamado *«Tennis for Two»* en el cual recreaba una partida de tenis desde una visión lateral utilizando líneas y puntos. En la este programa, dos jugadores utilizaban la pantalla de un osciloscopio como elemento de representación, y unos controladores de entrada que el mismo *Higinbotham* había construido para interaccionar con el videojuego en tiempo real.

En los 70 surgieron las primeras máquinas recreativas, siendo *«Galaxy Game»* la primera de ellas. Estaban diseñadas para recibir monedas a cambio de jugar una partida. Desafortunadamente el precio que se requería para construir ese *hardware* era mucho mayor que lo que podía recaudarse en un salón recreativo habitual.

Los 80 es lo que conoceríamos como la edad de oro del videojuego. En esta época España estuvo a la cabeza de esta industria. Sobre estas circunstancias concretas y su importancia futura se profundizará un poco más en otras secciones de este mismo capítulo.

Los 90 trajeron nuevas máquinas más potentes, y con ello más posibilidades pero a su vez más complejidad a la hora de crear videojuegos. Esto hizo que los equipos necesarios para ello aumentaran en número hasta formar grandes equipos de profesionales muy preparados. El reinado de los pequeños equipos de desarrollo había terminado.

Sobre el 2000 la industria se consolidó definitivamente, los videojuegos empezaron a tomarse como un producto serio sobre el cual sustentar una industria millonaria y la forma de realizar videojuegos se enfocaría desde una perspectiva más ingenieril y menos artesanal.

La gran revolución llegó hace unos años cuando los *smartphones* y el auge de las tiendas de venta de software mediante descarga digital, cambiaron otra vez las reglas del juego. A partir de este momento un conjunto de factores como es la capacidad limitada de estos dispositivos, junto con la evolución de las herramientas de creación, permitían que nuevamente un par de jóvenes desde un garaje pudieran explotar al máximo las oportunidades comerciales de ese *hardware*.

3.2. DISEÑO

3.2.1. Introducción al diseño de un videojuego

En este apartado se habla sobre los aspectos fundamentales relativos al diseño de un videojuego [23]. La lectura introductoria de este apartado permitirá entender mejor lo expuesto en el capítulo 5 donde se describe el diseño adoptado para llevar a cabo el desarrollo de *«Rock Zombie»*.

En la creación de un videojuego, la primera etapa a realizar es la de diseño. Para ello hay aprender el lenguaje específico del videojuego. Lo que se busca no es obtener una pieza de tecnología punta, sino hacer uso de la tecnología existente para crear un producto que destaque por su historia, gráficos, interactividad y diversión que proporciona.

Una vez que se tiene un diseño, hay que dominar las herramientas de desarrollo, la tecnología software y emplear los conocimientos necesarios para transformar ese documento de

diseño en un producto ejecutable, que funcione sobre un determinado hardware.

Como tercer paso en este complejo proceso, una vez se domina el propio lenguaje del videojuego y se domina la tecnología, es necesario conocer cómo funciona el mercado, qué tipo de modelos de negocio y de comercialización existen, y aplicar un correcto plan de marketing y distribución para hacer que el videojuego llegue a el publico objetivo de forma rentable.

3.2.2. Documento de diseño del videojuego

La realización del Documento de diseño del videojuego (*GDD*) [90] es el primer paso en el proceso de creación de un videojuego. En él se recogen los detalles sobre el diseño de juego. Se detallan elementos como la jugabilidad, historia, personajes, progresión, entorno, ambientación, etc. Es parecido a un documento de especificación de requisitos, pero mucho más flexible y específicamente orientado al desarrollo de videojuegos.

En el documento se explican datos simples sobre el producto. En las diferentes secciones se va llegando a mayor grado de detalle hasta definir completamente cómo va a ser el videojuego. Es necesario cubrir aspectos como controles, pantallas de juego o mecánicas jugables. El *GDD* se utiliza para plasmar todas las ideas y conceptos que aparecerán en el videojuego.

Uso del documento

El *GDD* se utiliza para multitud de propósitos. A continuación se enumeran algunos de los más importantes.

- **Detallar ideas :** Podemos explicar las características con las que contará nuestro videojuego, pero si no hay un documento que lo respalde, mediante una especificación formal del mismo, se quedará en el aire. Para ello se utiliza a modo de organización de los pensamientos del creador y resto del equipo. También será útil para visualizar una estructura del proyecto y darse cuenta de posibles problemas en etapas tempranas.

- **Tener un contexto común :** En un equipo creativo no es raro que tras una sesión de *brainstorming* cada miembro del equipo acabe con una idea sobre lo que el juego final va a ser. Con la elaboración de este documento, se consigue plasmar un consenso común entre las diferentes decisiones a tomar. También es una forma de asegurarse de que todo el personal de el equipo trabaje con un objetivo común y concreto.

- **Organizar la fase de desarrollo :** En ingeniería del software, la fase de análisis prepara y organiza la forma en la que la implementación se lleva a cabo. Gracias al *GDD* tenemos una estructura clara para organizar las siguientes etapas de desarrollo. Y de este modo quedan acotados y detallados cada uno de los aspectos que hay que desarrollar.
- **Comunicación :** Para realizar una correcta comunicación con agentes externos es de vital la existencia de un buen *GDD*. En el caso de un juego comercial, el equipo podrá buscar financiación de un "publisher" o presentar la idea a un grupo de inversores. Para tener garantías de éxito en este tipo de tareas es necesario un *GDD* detallado y un prototipo funcional del producto.

Estructura típica del *GDD* [91]

- **Página de resumen** : Nombre, información de *copyright*, número de versión, autor, fecha de publicación.
- **Índice del documento**
- **Historia de versiones** : Descripción de los prototipos, fecha de creación y detalle de dificultades e información recopilada.
- **Sección 1** : Descripción del videojuego
  - **Concepto del juego** : Descripción de la idea básica del videojuego.
  - **Conjunto de características** : Descripción de las características que va a tener el videojuego.
  - **Género y público objetivo** : Descripción del público al que va dirigido y sobre qué género se van a desarrollar las mecánicas.
  - **Estilo y ambientación característicos** : Descripción del estilo característico del videojuego.
  - **Definición de la profundidad** : Detalle del número de localizaciones, número de niveles, número de personajes, número de armas, etc.
- **Sección 2** : Jugabilidad y mecánicas [7]
  - **Progresión del videojuego** : Descripción de los diferentes elementos que pueden evolucionar en el videojuego. Definir conforme a que valores o diferentes características pueden cambiar.

- **Estructura** : Misiones, puzzles, estructura del desafío.
- **Objetivos del videojuego** : Definición del objetivo general del videojuego.
- **Mecánicas jugables** : Físicas, movimiento, objetos, acciones, combate, economía, etc.
- **Flujo de pantallas** : Gráfico de cada pantalla, esquema de menús, opciones de los menús.
- **Rejugabildiad y partida** : Descripción del estado de la partida, qué cosas se salvan, qué *checkpoints* hay, etc.

- **Sección 3** : Historia y personajes (*Story bible*)

  - **Historia y narrativa** : Esquema de elementos, progresión del videojuego, consideración de licencias, *Cut Scenes*.
  - **Mundo del videojuego** : Descripción del mundo. Por cada área hay que detallar características físicas, niveles dentro del mundo, conexión entre áreas.
  - **Personajes** : Por cada personaje hay que detallar historia, personalidad, aspecto, animaciones, habilidades especiales, relevancia en la historia, relaciones con otros personajes.

- **Sección 4** : Niveles

  Para cada uno de los niveles se detallaría lo siguiente :

  - **Sinopsis** : Explicación de la relación con la trama de este nivel concreto, conexión con otras áreas e importancia dentro del conjunto general.
  - **Objetivos** : Objetivo del nivel.
  - **Descripción física** : Descripción de que estructuras se va a encontrar el jugador cuando llegue a ese nivel.
  - **Mapa** : Mapa del nivel con una descripción de los diferentes elementos.
  - **Camino crítico** : Camino que permite finalizar el nivel, aún sin recolectar todos los secretos o descubrir todas las zonas.
  - **Eventos** : Descripción de los eventos que pueden surgir en ese nivel concreto.

- **Sección 5** : Interface

  - **Sistema visual** : Que *HUD*, menús, cámaras y modelos de iluminación se van a emplear.
  - **Sistema de control** : Con qué configuración de botones se va a utilizar el videojuego.
  - **Audio** : Qué tipo de sonido ambiental se va a emplear en cada una de las partes del videojuego.
  - **Música** : Qué tipo de música se va a emplear en cada ambiente.
  - **Efectos de sonido** : Qué tipo de efectos de sonido se van a emplear para aportar *feedback* al jugador.
  - **Sistema de ayuda o tutorial** : Descripción de el sistema de apoyo al jugador. En qué momento va a mostrar información, bajo qué criterios y qué tipo de ayuda se va a proporcionar.

- **Sección 6** : Inteligencia artificial [22]
  - ***IA* de los oponentes** : Definición de comportamientos para enemigos.
  - ***IA* de *NPC's*** : Definición de comportamientos para otros personajes no jugables.
  - ***IA* de soporte** : Jugador, detección de colisiones, *pathfinding*.

- **Sección 7** : Apartado técnico (*Thecnical Bible*)
  - ***Hardware* objetivo** : Dispositivo sobre el que se va a ejecutar el videojuego.
  - ***Hardware* empleado** :*Hardware* y software con el que se va a desarrollar el videojuego.
  - **Desarrollo** : Procedimientos y estándares que se van a emplear en el desarrollo.
  - ***Game Engine*** : Descripción del motor de videojuegos a utilizar.
  - ***Network*** : Características de red del videojuego y cómo van a implementarse.
  - **Lenguaje de *scripting*** : Descripción del lenguaje con el que se va a definir el diseño de niveles.
  - Varios

- **Sección 8** : Apartado artístico (*Art Bible*)
  - ***Concept art*** : Lista de *concept art's* empleados para describir la parte artística del videojuego.

- **Guías de estilo** : Qué estilos se van a utilizar para cada uno de los elementos del videojuego.
- **Personajes** : Descripción de los personajes del videojuego desde el punto de vista artístico.
- **Entornos** : Descripción de los entornos del videojuego desde el punto de vista artístico.
- **Objetos** : Descripción de los objetos del videojuego, interactivos y decorativos.
- ***Cut scenes*** : Descripción de las *cut scenes*. Se explica en qué contexto se reproducen y qué tipo de actores están involucrados. También qué repercusiones tiene sobre la trama cada una de estas escenas.
- **Varios** : Resto de cosas que tienen que ver con el apartado artístico.

- **Sección 9** : Software auxiliar

  - **Editor** : Si el videojuego va a contar con un editor de niveles hay que describirlo aquí, sus características, cómo se usa, etc.
  - **Instalador** : Qué instalador o software auxiliar se va a utilizar para desplegar el videojuego en la plataforma objetivo.
  - **Actualización del software** : Hay que detallar qué mecanismos de software se van a emplear para extender el videojuego en el futuro.

- **Sección 10** : Gestión del proyecto [13]

  - **Planificación detallada** : Se debe realizar una planificación detallada del proceso de realización del videojuego, las fases, hitos y etapas de desarrollo.
  - **Presupuesto** : Descripción del presupuesto, a qué se va a destinar y qué duración de proyecto se puede permitir con esa cantidad.
  - **Análisis de riesgos** : Análisis de la inversión desde el punto de vista financiero.
  - **Plan de localización** : Explicación de cómo se va a localizar el producto a las diferentes culturas y países. Qué modelos de negocio se van a adoptar en cada país, qué idiomas se van a soportar, etc.
  - **Plan de *testing*** : Hay que definir qué dispositivos se van a utilizar, qué fases de pruebas se van a llevar a cabo. Si se va a externalizar o se va a tener a un equipo de *testers* en plantilla, etc.

- **Apéndices** :
  - **Arte** : Lista de modelos, texturas, animaciones, efectos, gráficos de interface y lista de *cut scenes*.
  - **Sonido** : Sonidos de ambiente, sonidos del interface, efectos de sonido del videojuego.
  - **Música** : Banda sonora, música de diferentes eventos, acción, victoria, derrota.
  - **Voces** : Lista de líneas de cada actor

3.2.3. Perfiles involucrados en el diseño y desarrollo de un videojuego

A continuación se detallan el conjunto de profesionales que forman un equipo típico de desarrollo de un videojuego [21]. A menudo una misma persona cubre varios perfiles y si el equipo de desarrollo es pequeño pueden agruparse gran cantidad de estos perfiles profesionales en una única persona.

- **Productor :** Es el encargado de gestionar el proceso de realización del videojuego. Trazando planes e hitos en el desarrollo, encargándose de gestionar la financiación disponible y reestructurar el proceso para alcanzar las fechas pactadas y así cumplir objetivos. Suele ser una persona con amplia experiencia en el desarrollo de videojuegos y conoce todas las partes aunque no necesariamente debe tener un conocimiento profundo de cada una de ellas.
- **Diseño / Diseño de juego :** Encargado de diseñar las mecánicas jugables del videojuego, de darle un contexto y un trasfondo al mundo que se va a reflejar. Debe conocer todo tipo de juegos, diversas formas de control y plantear un diseño de juego que sea atractivo para el jugador. Es el encargado de hacer que te diviertas. Suele ser una persona con bastante experiencia en el desarrollo de videojuegos y es capaz de tener en cuenta el trabajo de los otros perfiles para realizar diseños que se puedan resolver con una solución tecnológica.
- **Diseño / Diseño de niveles :** Encargado de coger las mecánicas propuestas por el diseñador de juego así como los elementos diseñados por los artistas y los elementos de programación de los programadores para construir los niveles del videojuego. Junta el trabajo de varios perfiles para generar la experiencia jugable.
- **Diseño / Guionistas y escritores :** Este tipo de perfiles escriben la historia, realizan el árbol de diálogos o le dan un trasfondo al mundo virtual que se va a recrear.

- **Arte / Artista de conceptos *2D* :** Son los encargados de traducir a imágenes de conceptos la visión del diseñador de juego. Suelen usar acuarelas, métodos tradicionales o ilustración digital para generar *artworks* que se usará de referencia para transmitir el concepto del videojuego al resto del equipo.

- **Arte / Modelador *3D* :** Son los encargados de generar la geometría tridimensional a partir de los conceptos *2D* de los que se parte. Este tipo de perfiles crean los modelos *3D* de los personajes y escenarios.

- **Arte / Animador :** A partir de los modelos *3D* creados por los modeladores *3D* el animador les da vida mediante animación basada en esqueletos. Para ello crea las animaciones que luego se visualizarán en el juego.

- **Arte / Artista técnico :** El artista técnico es el nexo de unión entre los artistas y los programadores, encargándose de que los modelos, texturas y resto de material de arte encaje en las necesidades técnicas que requiere el motor de videojuegos con el que lo están desarrollando.

- **Programación / Programador de mecánicas :** Es el encargado de programar los comportamientos de la aplicación bajo todo tipo de circunstancias. Debe traducir a código las mecánicas propuestas por el diseñador de videojuego.

- **Programación / Programador gráfico :** Es el encargado de programar el motor gráfico del videojuego o los componentes auxiliares al mismo como *Shaders* u otros efectos avanzados.

- **Música / Técnico de efectos de sonido :** Realiza la grabación y edición de los diferentes efectos de sonido que se reproducirán en el videojuego.

- **Música / Compositor de la banda sonora :** Músico o conjunto de músicos que componen la banda sonora del videojuego. En algunos casos se emplea una orquesta completa para esta tarea.

- **Testing / *Qualitty Assurance* :** Es el encargado de asegurar la calidad del producto, manteniendo la comunicación con los testers, los programadores y artistas para cumplir con los objetivos de calidad establecidos por el productor del videojuego.

- **Testing / *Tester* :** Es el encargado de probar el videojuego durante todas sus fases de desarrollo, informando de los *bugs* encontrados y haciendo un reporte técnico sobre

la situación en la que se han producido, posibles causas y qué posibles soluciones se podrían adoptar.

- **Localización y traducción :** Esta serie de perfiles adaptan el videojuego no sólo al idioma del país donde va a comercializarse, sino que adaptan las referencias a la cultura local.

- ***Community Managment* :** Encargado de representar a la empresa o el videojuego que se está desarrollando en *internet*. Responsable de comunicación y de tratar con la prensa para obtener la máxima repercusión posible.

Además de esta serie de perfiles es normal contar con diferentes grados de experiencia por cada uno de los perfiles (*Junior* y *Senior*) y además tener un director por cada departamento, como el director del equipo de desarrollo o el director de arte. Estos perfiles de dirección son los encargados de guiar al equipo con una visión común.

3.2.4. El videojuego como producto

En este apartado se tratan aspectos relativos al videojuego como producto en sí y se discuten los diferentes aspectos que lo conforman. En los siguientes apartados se hablará sobre la interactividad, dificultad, la progresión o la estructura típica de lo que es un videojuego.

Interactividad y diversión

El videojuego debe divertir al usuario [2], y para ello se hace uso de la interactividad mediante la cual el jugador puede cambiar cosas sobre el videojuego y obtener un resultado o *feedback* de esas acciones.

Se trata de dar la sensación de libertad suficiente. Hacer que el jugador se sumerja en el videojuego, para que tenga la ilusión de controlar las cosas que pasan en el mismo y conseguir que se divierta.

Contar una historia

Los videojuegos antiguos simplemente planteaban situaciones de acción donde el jugador interaccionaba. Con el avance en la complejidad de los videojuegos, hoy día se cuentan con guiones tan elaborados como los que cuentan las superproducciones de *hollywood*.

El uso de la historia contribuye a involucrar al jugador en los eventos que suceden dentro del videojuego y a llenar las partes que no se cuentan de manera interactiva. Para ello se hace uso de secuencias cinemáticas o diálogos.

Figura 3.1: Imagen del videojuego *The Walking Dead* [119]

En determinados videojuegos la historia constituye el eje central del mismo y el jugador querrá continuar jugando no porque las mecánicas del videojuego sean adictivas, sino porque quiere conocer más sobre esa historia a la que se encuentra enganchado. Un ejemplo de esta tendencia es el videojuego de *«The Walking Dead»* [92] que puede apreciarse en la figura 3.1.

Desafío para el jugador y gestión de la dificultad

Los videojuegos deben plantear retos [6], dando sensación de progreso y generar satisfacción cuando se superan. La naturaleza de estos retos pueden ser mentales o de habilidad y hacen que el jugador ponga su atención en la resolución de los mismos.

La gestión de la dificultad es un aspecto muy importante, pues si se cuenta con una dificultad muy baja el jugador perderá interés y abandonará el videojuego. Por otro lado si la dificultad es muy alta el jugador se sentirá frustrado y también abandonará el videojuego sin haberse divertido.

Para conseguir la sensación de reto asumible es necesario llevar una gestión de la dificultad que permita al jugador sentir que está superando situaciones que le ponen a prueba, pero

que a su vez no supongan un reto tan alto que le haga sentir frustración. Lograr el equilibrio adecuado se traducirá en que el jugador se divierta con el videojuego.

En los videojuegos se manejan curvas de dificultad crecientes a lo largo de los niveles, y se establece un riguroso proceso de *testing* probándolo sobre usuarios finales para obtener datos y balancear los diferentes aspectos del juego. De estas formas se consigue una curva de dificultad adecuada.

Progresión del jugador

El jugador debe de sentir que evoluciona y se involucra con la historia [3]. Para ello se incluyen factores de personalización del personaje que se van habilitando a medida que el jugador progresa en la trama. En la Figura 3.2 se muestra una captura de pantalla de las opciones de personalización del avatar en *Diablo 3*.

Figura 3.2: Imagen del videojuego *Diablo 3* [93]

Al dar a elegir estos factores diferenciadores, el jugador ajusta el personaje a su gusto y se une al mismo permitiendo involucrarse más en la trama y los sucesos del videojuego.

Factores de enganche

Además de la progresión del jugador y el uso de una correcta curva de dificultad se proponen otra serie de factores que hacen que el jugador quiera continuar avanzando. Algunos de estos elementos son los logros desbloqueables y los *items* coleccionables.

Un logro desbloqueable es un objetivo secundario que se plantea antes del inicio del juego y que su obtención no es necesaria para completar el juego pero que si se hace, se obtienen algunas ventajas como mayor puntuación o más monedas. El uso de estos logros contribuyen a que el jugador quiera continuar jugando para completar todos los objetivos, aunque ya hayan superado los objetivos principales del videojuego.

Un *ítem* coleccionable es algún elemento del propio videojuego que ha sido dispuesto de manera secreta a lo largo del mismo o que se permite comprarlo si se han coleccionado la cantidad suficiente de monedas. Este tipo de *items* permiten modificar la jugabilidad del videojuego en sucesivas partidas, aportando trajes alternativos, armás más potentes u otras ventajas para que el jugador quiera probarlas y continúe jugando una vez se hayan superado los objetivos principales del videojuego.

Como se ha discutido anteriormente, otro elemento fundamental es la historia del videojuego, que igualmente contribuye a ”enganchar“ al jugador al producto.

Interfaz de usuario

La interfaz de usuario debe proporcionar toda la información que el jugador necesita y aportar el *feedback* necesario para que el jugador pueda leer la situación de lo que está pasando en el entorno virtual.

El uso de textos, colores o gráficos contribuyen a que el jugador pueda acceder a la información y no se sienta perdido o con desconocimiento de lo que está pasando en el entorno virtual.

Si no se tiene una buena interfaz de usuario y no se le proporciona al jugador el *feedback* correcto este se desorientará y acabará por abandonar el videojuego.

Estructura típica de un videojuego

Un videojuego es un tipo de software de entretenimiento que está pensado para todo tipo de públicos y en consecuencia debe tener una estructura y forma de acceder a las opciones muy sencilla y asistida. A continuación se describen las partes típicas de las que consta un videojuego tradicional.

- **Menú :** El menú del juego permite acceder a jugar los diferentes niveles o modos de juego. Desde aquí iniciamos la primera partida o continuamos en las sucesivas.
- **Opciones típicas :** Es habitual contar con un menú de configuración donde ajustar los métodos de control y los ajustes gráficos.
- **División en niveles :** El videojuego suele estar dividido en niveles o escenas sucesivas. La dificultad suele ir creciendo conforme avanzamos en los diferentes niveles según una curva de dificultad establecida.
- **Elementos coleccionables / desbloqueables :** Suele haber otra serie de menús, como es el menú donde se nos informa de los logros que podemos desbloquear y otros menús donde podemos personalizar o comprar más *items* que modifican el comportamiento de la partida del propio videojuego.

3.2.5. Géneros de videojuegos actuales

En la actualizad hay multitud de géneros de videojuegos establecidos y es raro que se hagan videojuegos que no pertenezcan a un género existente.

- **Sociales :** Videojuegos que se caracterizan por la interacción con otros jugadores para avanzar. Muy populares en las redes sociales y con una buena base de usuarios actualmente.
- **Casual :** Videojuegos sencillos, divertidos y sin complicaciones. Accesibles para todo tipo de personas sin que haya jugado previamente a videojuegos. Muy comunes en dispositivos móviles.
- **Simuladores de conducción :** Videojuegos que simulan la experiencia de conducir un vehículo, ya sea un coche de carreras, un avión de pasajeros o un tren.
- **Simuladores deportivos :** Videojuegos que simulan diferentes deportes como Fútbol, Baloncesto, Tenis, etc...

- ***First / Third Person Shooters* :** Videojuegos cuyas mecánicas principales se basan en disparar a enemigos para avanzar por el escenario. Pueden ser en primera o tercera persona dependiendo de la posición de la cámara.

Figura 3.3: Algunos Géneros de Videojuegos : a.Social [96], b.Casual [98], c.Simulación de conducción [99], d.Simulador deportivo [100]

- **Juegos de estrategia :** Videojuegos cuyas mecánicas se basan en la gestión de recursos o ejércitos. Los hay bélicos, pero también otros que permiten gestionar el desarrollo de una ciudad o la construcción de tu propio vecindario.

- **Juegos de rol :** Videojuegos cuyas mecánicas se centran en la narración de una historia compleja en la que es posible decidir sobre la evolución del personaje. En este tipo de videojuegos se visitan multitud de entornos en busca de objetos y experiencia para el personaje.

- **Aventuras :** Videojuegos donde debemos encontrar objetos, saltar a plataformas, o

librarnos de enemigos para vivir una aventura en la que la trama y los diálogos cobran mucha importancia.

Figura 3.4: Algunos Géneros de Videojuegos actuales : a.*First Person Shooter* [112], b.Estrategia [111], c.Rol [95], d.Aventuras [97]

- ***Sandbox* :** Videojuegos que simulan un mundo virtual y sobre el se dispone una trama que por lo general requiere la interacción con otros personajes virtuales, vehículos, armas, etc.

- **Juego de lucha :** Videojuego de lucha entre personajes virtuales donde el objetivo es derrotar al adversario.

- ***Hack & Slash* :** Mezcla de los juegos de aventuras y rol donde la atención se enfoca más en el combate cuerpo a cuerpo del personaje con los enemigos que se encuentran en el escenario. Importa menos la trama y más la acción.

- ***Survival Horror* :** Variante de los juegos de aventuras donde la trama se centra en el terror y se busca que el jugador cuente con pocos recursos para sobrevivir en un entorno hostil.

Figura 3.5: Algunos Géneros de Videojuegos : a.*Sandbox* [116] , b.Lucha [114] , c.*Hack & Slash* [113], d.*Survival horror* [117]

3.2.6. Géneros de videojuegos clásicos

En la actualidad, existen algunos géneros de videojuegos clásicos que no están siendo explotados por grandes compañías. Este tipo de géneros están siendo abordados por desarrolladores independientes con muy buenos resultados, pues aunque para el público generalista no despiertan el mismo interés que otro tipo de géneros actuales, sí que hay un nicho de mercado muy importante que demanda este tipo de géneros clásicos y que encuentran muy poca representación en los videojuegos actuales.

Figura 3.6: Algunos Géneros de Videojuegos : a.*Metroidvania* [104] , b.Aventura gráfica [102]

- ***Shoot 'em up de scroll lateral* :** Videojuegos de acción arcade que basan sus mecánicas principales en el manejo de un personaje que avanza disparando a enemigos de forma lateral. Podríamos considerarlos los precursores de los *First / Third Person Shooters*.
- ***Beat 'em up de scroll lateral* :** Videojuegos de acción arcade que basan sus mecánicas principales en el manejo de un personaje que avanza luchando con enemigos de forma lateral y haciendo uso de combate cuerpo a cuerpo o armas cortas. Podríamos considerarlos los precursores de los *Hack & Slash*.

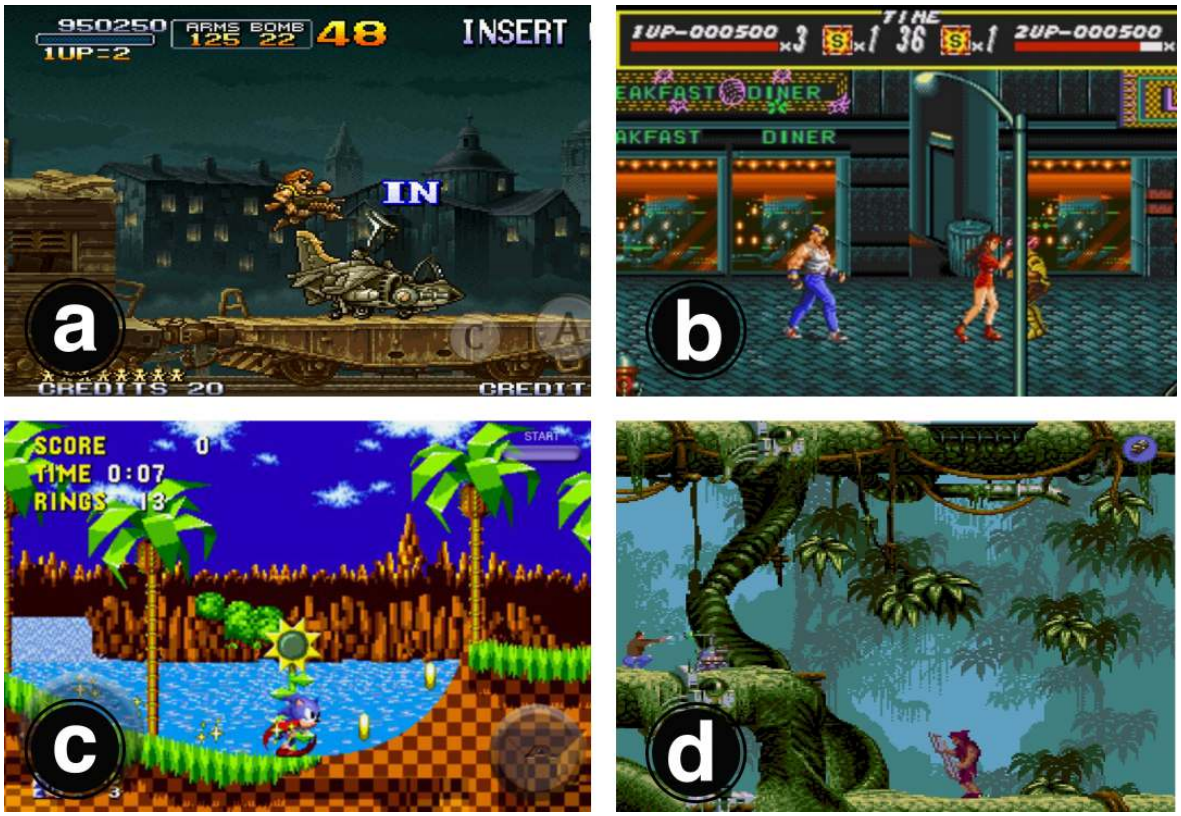


Figura 3.7: Algunos Géneros de Videojuegos : a.*Shoot 'em up* [101] , b.*Beat 'em up* [110] , c.Plataformas [115] , d.Plataformas cinemático [103]

- **Juegos de plataformas :** Videojuegos en los que la mecánica principal es avanzar por el escenario e ir saltando a diferentes puntos para evitar peligros o eliminar enemigos. Podríamos considerarlos los precursores de los videojuegos de Aventuras.
- **Plataformas cinemáticos :** Videojuegos de plataformas con tramas más complejas y estilo gráfico más maduro en el que el objetivo no es saltar por el escenario para evitar

peligros, sino formar parte de una trama más elaborada en la que resolver diferentes objetivos. Podríamos considerarlos los precursores de los videojuegos de Aventuras.

- ***Metroidvania* :** Videojuegos de plataformas en los que el avance no es lineal y prima mucho el desarrollo del personaje y la búsqueda de objetos especiales repartidos por diferentes zonas. Podríamos considerarlos como influyentes en la forma de hacer los videojuegos de rol y aventuras actuales. Caracterizados por el uso de *backtracking* en el recorrido de los niveles.

- **Aventuras gráficas :** Videojuegos de historias donde el desarrollo de diálogos entre el jugador y los demás personajes virtuales son decididos por el usuario. Estas decisiones determinan el avance la historia. También caracterizados por plantear enigmas o *puzzles* al jugador. Podríamos considerarlos los precursores de los videojuegos de Aventuras.

3.3. DESARROLLO

3.3.1. Desarrollo de videojuegos

El desarrollo de un videojuego se lleva a cabo por un equipo de profesionales multidisciplinares utilizando herramientas comerciales o de creación propia.

Una vez que el videojuego ha sido desarrollado como proyecto software se comercializa utilizando un modelo de negocio concreto y apoyándose en un plan de distribución que permita dar rentabilidad al producto. Estos aspectos se analizarán en mayor detalle en el capítulo 5.

3.3.2. Herramientas disponibles

En la industria del videojuego, las empresas buscan maximizar el beneficio del producto que se se pretende construir. Para ello suele ser más rentable no reinventar nuevamente la rueda si no es necesario. Para ello se parte de soluciones tecnológicas ya establecidas y que aporten garantías en el desarrollo del proyecto que se pretende desarrollar. Es por esto que la mayoría de las empresas de videojuegos desarrollan los videojuegos utilizando motores de juegos ya existentes y cuyas funcionalidades encajen con el tipo de videojuego que se pretende desarrollar.

En otros casos algunas empresas desarrollan un motor de juego propio. Por ejemplo cuando quieren hacer una inversión a largo plazo o quieren compartir su uso en diferentes juegos de la misma compañía. Este tipo de casos no es el más común y suele ser motivado por querer cubrir funcionalidades que los motores existentes no soportan.

Otro tipo de empresas como *Crytek* o *Epic* desarrollan motores de videojuego comerciales que licencian al resto de empresas de desarrollo de videojuegos para que adquieran esa tecnología en lugar de desarrollarla desde cero.

En este apartado se describirán las principales *APIs* gráficas, *Frameworks* y motores de videojuego utilizados en la actualidad.

APIs gráficas

Una *API* (*Application Programming Interface*) [53] es un conjunto de funciones que realizan tareas específicas. Cuando hablamos de *API* gráfica nos referimos a un conjunto de funciones para inicializar por ejemplo los modos gráficos, realizar el copiado de gráficos de la memoria del computador a la memoria gráfica o pintado de elementos en pantalla. Actualmente existen varias *APIs* que permiten realizar la misma tarea y que están disponibles para diferentes sistemas operativos.

- ***Direct X* :** Es una *API* multimedia creada por *Microsoft* en 1995 que consta de *Direct3D* para la parte gráfica, *DirectSound* junto con *DirectMusic* para la parte de audio y *DirectInput* para la parte de teclados y *joysticks*. También incluye *DirectPlay* para comunicación de datos en redes. Esta *API* puede ser utilizada para soportar la programación de videojuegos en entornos *Windows* y de *XBox*.

- ***OpenGL* :** Es una *API* multiplataforma creada por *Silicon Graphics* en 1992 que maneja sólo el aspecto gráfico del sistema dejando el control de los otros aspectos a otras *APIs* específicas de cada una de esas tareas. Se trata de una *API* multiplataforma y puede usarse para soportar la programación de videojuegos en entornos *Windows*, *GNU/Linux*, *Mac*, *Playstation* y multitud de otros dispositivos.

- ***OpenGL ES* :** Es un subconjunto de la *API* de *OpenGL* que está optimizado para funcionar en dispositivos móviles. Actualmente está soportado por multitud de dispositivos como *smartphones* y *tablets*.

No suele ser común utilizar la *API* gráfico directamente para realizar videojuegos. Lo más normal es construir un *framework* o un motor de videojuego que haga uso de varias *APIs* y sobre eso construir la base de desarrollo del videojuego.

Frameworks

Un *framework* define en términos generales un conjunto estandarizado de conceptos, practicas y criterios para enfocar un tipo de problemática particular que sirve para enfrentar y resolver problemas de una determinada clase. En concreto los *frameworks* que se presentan a continuación hacen uso de las *APIs* gráficas y presentan una serie de interfaces de programación que permiten facilitar la tarea de pintado de gráficos, gestión del sonido o gestión de la entrada de control del usuario.

- ***Cocos 2D* [55] :** Es un *framework* multiplataforma y *open source* que permite desarrollar videojuegos en *2D*. Tiene con características de soporte de *Sprites*, así como efectos o transiciones entre escenas. Cuenta con una versión que permite generar videojuegos para dispositivos móviles. Internamente utiliza *OpenGL* para renderizar los gráficos.

- ***Ogre 3D* [56] :** Es un *framework* de renderizado *3D* orientado a escenas y escrito en lenguaje de programación *C++*. Sus bibliotecas evitan tener que utilizar capas inferiores de las librerías gráficas como *OpenGL* o *Direct3D*.

- ***SDL* [57] :** Es una *framework* multiplataforma que está formado por varias bibliotecas que permiten gestionar la renderización de contenido *2D*, la gestión de sonido o gestión de el control. Permite la comunicación con *APIs* como *OpenGL* para acceder a renderización de gráficos en *3D*. Se suele usar en combinación con otros *frameworks*.

Algunos videojuegos se desarrollan utilizando *frameworks*, no obstante es común crear herramientas que hagan uso de estos *frameworks* y que permitan realizar por ejemplo el diseño de niveles u otros elementos del videojuego. En este caso podríamos considerar que se ha desarrollado un mini motor de videojuegos haciendo uso de un *framework* existente, y que sobre este se ha construido el videojuego.

Motores de juego

La línea entre *framework* y motor de videojuegos [54] no está bien definida, a menudo la única diferencia es que el motor de juego además de hacer uso de las *APIs* gráficas y

frameworks, también ofrece una serie de herramientas auxiliares como un editor de niveles, gestor de *assets*, gestor de importación de formatos o comunicación con el compilador. En la mayoría de los videojuegos comerciales se suele utilizar un motor de juego sobre el que desarrollar el propio videojuego. A continuación se describen algunos de los motores de videojuego más utilizados actualmente [58]. Algunos de ellos son comerciales, mientras que otros son privados de las empresas que los han desarrollado específicamente para sus videojuegos.

- ***«Unity3D»* [59] :** Motor de videojuegos multiplataforma creado por *Unity Technologies*. Esta disponible como plataforma de desarrollo para *Windows* y *Mac OS X* y permite crear videojuegos para *Windows*, *Mac OS X*, *GNU/Linux*, *Xbox 360*, *Playstation3*, *Playstation Vita*, *Wii*, *Wii U*, *iOS*, *Android* y *Windows Phone*. Actualmente es una de las elecciones preferente de los desarrolladores independientes pues su carácter multiplataforma permite adaptar el videojuego fácilmente a multitud de dispositivos de videojuegos.

- ***«Unreal Engine»* [62] :** Uno de los motores técnicamente más potentes de la actualidad. Utilizado ampliamente por multitud de estudios en la realización de videojuegos *triple A*. Usado en juegos como *«Gears of War»*, *«Mass Effect»*, *«Bioshock»*.

- ***«CryEngine»* :** *«CryEngine»* es una joya tecnológica, podríamos considerarlo como el motor más potente en características técnicas. No es tan licenciado por los estudios como *«Unreal Engine»* y es menos versátil pero sus características lo establecen como uno de los motores más consolidados de la actualidad. Usado en juegos como *«Far Cry»* o *«Crysis»*.

- ***«Frosbite 3»* :** Este motor ha sido desarrollado por la empresa *Electronics Art* y es utilizado en la realización de muchos de los juegos que realiza esta editora. Destacan sus punteras características técnicas y su uso en la industria, que lo ponen a la altura de *«Unreal Engine»* y *«CryEngine»*. Usado en juegos como *«Battlefield 4»* o *«Need For Speed : Rivals»*.

- ***«Source Engine»* :** Este es otro motor que es muy conocido en la industria. Usado en videojuegos como *«Half Life 2»*, *«Portal»* o *«Team Fortress 2»* y licenciado por algunas empresas. Tiene menos características técnica pero cuenta con un excepcional código de red que permite desarrollar videojuegos online con la máxima fiabilidad posible.

- ***«Anvil Engine»*** **:** Este motor no ha recibido mucha publicidad pero es el encargado de mover videojuegos como *«Assasin's Creed»* o *«Prince of Persia»* y tiene características de generación de animaciones en tiempo real con mundos abiertos y gráficos detallados. Su principal característica es que puede renderizar escenarios de una manera completamente escalable para entornos enormes.

- ***«Geo-Mod»*** **:** Una de las cosas más difíciles de equilibrar en las consolas es el poder de procesamiento. Este motor creado por *Vollition* y utilizado en *«Red Faction : Guerrilla»* permite equilibrar todos los elementos de un entorno *sandbox* abierto y dinámico generando un estable resultado de renderización de 30 *FPS* permitiendo destruir cualquier estructura del entorno de manera dinámica.

- ***«EGO Engine»*** **:** Este motor de videojuegos está centrado principalmente en proporcionar las herramientas necesarias para realizar videojuegos de carreras. Ha sido utilizado en juegos como *«Grid 2»* o *«Dirt 3»*. Permite tener un sistema de gestión de daños en vehículos y un sistema de partículas muy avanzado.

- ***«MT Framework»*** **:** Este motor fue creado para desarrollar los videojuegos de *Capcom* de la pasada generación de consolas. Ha sido utilizado en videojuegos como *«Resident Evil 5»*, *«Lost Planet»* 1 y 2 o *«Devil May Cry 4»*. Permite una gran flexibilidad a la hora de desarrollar tanto para *PC* como Consola, y se adapta fácilmente a la plataforma de videojuegos.

- ***«Rage»*** **:** Este es uno de los motores de videojuegos más importantes de toda la década. Es propietario de *Rockstar* y lo utiliza en sus videojuegos de mundo abierto como *«GTA V»* o *«Red Dead Redemption»*. Este motor permite gestionar mundos abiertos sin cargas y haciendo uso de su integración con el motor de animación física *Euphoria* que permite realizar animaciones dinámicas de cuerpos bípedos para dotar a los videojuegos de una interactividad total en cualquier tipo de situaciones.

- ***«id Tech»*** **:** El padre de *«Doom»* y de los motores gráficos modernos *John Carmack* ha sido el encargado de desarrollar este motor de videojuegos. En su quinta versión ha servido como base para multitud de videojuegos *first person shooters* actuales. Utilizado en juegos como *«Quake»*, *«Doom»*, *«Prey»*, *«Wolfenstein»* o *«Rage»*.

- ***«IW Engine»*** **:** Este motor de videojuegos se desarrolló por *Infinity Ward* y sirve como base para todos los videojuegos de la saga *«Call Of Dutty»* que han salido al mercado. Actualmente se ha quedado un poco desfasado en el apartado de renderización, no

obstante destaca por su rendimiento y código de red que permiten generar videojuegos con el mejor rendimiento de la parte *online* del producto.

3.3.3. Introducción al desarrollo de videojuegos

En este apartado se analizan brevemente los fundamentos matemáticos de la parte tecnológica de un videojuego. La lectura introductoria de este apartado permitirá entender mejor lo expuesto en el capítulo 5 donde se describe la arquitectura de la solución tecnológica para llevar a cabo el desarrollo de este videojuego.

Introducción

Desde el punto de vista del usuario, un videojuego puede definirse como una aplicación software que responde a una serie de eventos, redibujando la escena y generando una serie de respuestas adicionales (sonido en los altavoces, vibraciones en dispositivos de control, etc...).

Habitualmente el motor gráfico trabaja con geometría descrita mediante mallas triangulares. Las técnicas empleadas para optimizar el despliegue de esta geometría, junto con las propiedades de materiales, texturas e iluminación varían dependiendo del tipo de videojuego que se está desarrollando. A un alto nivel de abstracción podemos ver el proceso de *rendering*(también llamado renderizado o renderización) como el proceso encargado de convertir la descripción de una escena tridimensional en una imagen bidimensional. En esencia el proceso de renderizado de una escena *3D* requiere los siguientes elementos [11]:

- **Superficies :** La geometría de los objetos que forman la escena debe ser definida empleando alguna representación matemática para su posterior procesamiento por parte del computador.

- **Cámara :** La situación del visor debe ser definida mediante un par (posición, rotación) en el espacio *3D*. El plano de imagen de esta cámara virtual definirá el resultado del proceso de renderizado. Para imágenes generadas en perspectiva, el volumen de visualización define una pirámide truncada que selecciona los objetos que serán representados en la escena. Esta pirámide se denomina *Frustrum*.

- **Fuentes de luz :** Las fuentes de luz emiten rayos que interactúan con las superficies e impactarán en el plano de imagen. Dependiendo el modo de simulación de estos impactos de luz tendremos diferentes métodos de renderizado.

- **Propiedades de las superficies :** En este apartado se incluyen las propiedades de materiales y texturas que describen el modelo de rebote de los fotones sobre las superficies.

Pipeline gráfico

El proceso de visualizar una escena en *3D* [11] mediante gráficos por computador es similar al que se realiza cuando se toma una fotografía real. En la figura inferior se muestran los pasos generales del *Pipeline* asociado a la transformación de una escena *3D* hasta su representación final en el dispositivo de visualización.

El *Pipeline* está dividido en etapas funcionales. Algunas de estas etapas se realizan en paralelo y otras secuencialmente.

Figura 3.8: *Pipeline* gráfico

- **Etapa de aplicación :** La etapa de aplicación se ejecuta en la *CPU*. Actualmente la mayoría de las *CPU's* son multinúcleo, por lo que el diseño de esta aplicación se realiza

mediante diferentes hilos de ejecución. En esta etapa se ejecutan las tareas asociadas al cálculo de la posición de los modelos *3D* mediante simulaciones físicas, detección de colisiones, gestión de la entrada del usuario.

- **Etapa de geometría :** En la proyección hacia la pantalla de cada objeto *3D* se transforma en diferentes sistemas de coordenadas. A los vértices de cada modelo se le aplican la denominada Transformación de Modelado para posicionar y orientarlo con respecto del Sistema de Coordenadas Universal obteniendo así las Coordenadas del Mundo. Como este sistema de coordenadas es único, tras aplicar la transformación de modelado a cada objeto, ahora todas las coordenadas estarán expresadas en el mismo espacio. La posición y orientación de la cámara nos determinará que objetos aparecerán en la imagen final.

  Habitualmente el *pipeline* contiene una etapa adicional intermedia que se denomina *Vertex Shader* o Sombreado de Vértice que consiste en obtener la representación del material del objeto modelando las transformaciones en las fuentes de luz, utilizando los vectores normales a los puntos de la superficie, información de color, etc. La Transformación de Proyección convierte el volumen de visualización en un cubo unitario. Este volumen de visualización se define mediante planos de recorte *3D* y define todos los elementos que serán visualizados.

  Existen multitud de métodos de proyección, aunque como veremos más adelante, los más empleados son la ortográfica (o paralela) y la perspectiva. Tras la proyección, el volumen de visualización se transforma en Coordenadas Normalizadas (obteniendo el cubo unitario), donde los modelos son proyectados de *3D* a *2D*. La coordenada Z se guarda habitualmente en un *buffer* de profundidad llamado *Z-Buffer*. Únicamente los objetos que están dentro del volumen de visualización deben ser generados en la imagen final. Los objetos que están totalmente dentro del volumen de visualización serán copiados íntegramente a la siguiente etapa del *pipeline*. Sin embargo, aquellos que estén parcialmente incluidas necesitan ser recortadas, generando nuevos vértices en el límite del recorte. Esta operación de Transformación de Recorte se realiza automáticamente por el *hardware* de la tarjeta gráfica. Finalmente la Transformación de Pantalla toma como entrada las coordenadas de la etapa anterior y produce las denominadas Coordenadas de Pantalla, que ajustan las coordenadas x e y del cubo unitario a las dimensiones de ventana finales.

- **Etapa de rasterización :** A partir de los vértices proyectados y la información asociada a su sombreado obtenida de la etapa anterior, la etapa de rasterización se encarga de calcular los colores finales que se asignarán al os *pixeles* de los objetos. En la primera etapa del *pipeline* llamada Configuración de triángulos, se calculan las coordenadas *2D* que definen el contorno de cada triangulo. Esta información es utilizada en la siguiente etapa y normalmente se implementa directamente en *hardware* dedicado. A continuación en la etapa de Recorrido de Triangulo se generan fragmentos para la parte de cada *pixel* que pertenece al triángulo. El recorrido del triángulo se basa por tanto en encontrar los *pixeles* que forman parte del triángulo, y se denomina *Triange Transversal*.

  El fragmento se calcula interpolando la información de los tres vértices definidos en la etapa de Configuración de Triángulos y contiene información calculada sobre la profundidad desde la cámara y el sombreado. La información interpolada de la etapa anterior se utiliza en el *Pixel Shader* para aplicar sombreado a nivel de *pixel*. Finalmente en la etapa de Fusión se almacena la información del color de cada *pixel* en un *array* de colores denominado *Color buffer*. Para ello se combina el resultado de los fragmentos que son visibles de la etapa de Sombreado de *Pixel*. La visibilidad se suele resolver en la mayoría de los casos mediante un *buffer* de profundidad *Z-Buffer*, empleando la información que almacenan los fragmentos.

Proyección ortográfica

Un modo de visualización muy utilizado en aplicaciones de *CAD* es la proyección de objetos empleando líneas paralelas sobre el plano de proyección mediante la denominada proyección ortográfica. En este modo de proyección se conserva las proporciones relativas entre objetos, independientemente de su distancia.

Este modo de proyección es ideal para los videojuegos en *2D* pues los sprites que se pintan en pantalla no deben verse desformados por su posición en la misma, sino que siempre deben de pintarse con el mismo tamaño independientemente de la distancia a la que se encuentren de la cámara.

Sprite 2D

Un *sprite* es un conjunto de *píxeles* que se pintan sobre la pantalla en una posición determinada. Si variamos la imagen que se pinta sobre ese sprite concreto obtendremos la sensación de animación, utilizándose esta técnica para representar personajes de un videojuego u otros objetos interactivos.

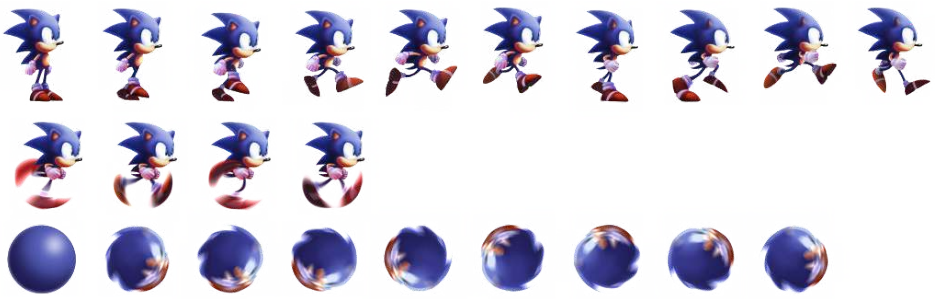


Figura 3.9: Imagen usada en un Sprite 2D [154]

Proyección en perspectiva

Mediante la proyección en perspectiva se proyectan los puntos hasta el plano de visualización empleando trayectorias convergentes en un punto. Esto hace que los objetos situados más distantes del plano de visualización, aparezcan más pequeños en la imagen. Las escenas generadas utilizando este modelo de proyección son más realistas, ya que ésta es la manera en que el ojo humano y las cámaras físicas forman imágenes.

En la proyección en perspectiva, las líneas paralelas convergen en un punto, de forma que los objetos más cercanos se muestran de un tamaño mayor que los lejanos.

Matemáticas para videojuegos

Para manipular los elementos de un entorno tridimensional se describe ese mundo mediante un modelo matemático sobre el que podemos realizar cálculos y transformaciones que dan como resultado la representación del videojuego y como fin último es la obtención de una imagen que se visualiza por pantalla. En este modelo matemático se emplean diferentes elementos que se definen a continuación y que nos permiten manipular el entorno virtual.

- **Puntos :** Un punto puede definirse como una localización en un espacio n-dimensional. En el caso de los videojuegos, este espacio suele ser bidimensional o tridimensional.

El tratamiento discreto de los valores asociados a la posición de los puntos en el espacio exige elegir el tamaño mínimo y máximo que tendrán los objetos en nuestro videojuego.

- **Coordenadas :** Es crítico definir el convenio empleado relativo al sistema de coordenadas y las unidades utilizadas entre programadores y artistas. Algunas de las alternativas posibles en este apartado son Coordenadas Cartesianas, Coordenadas Cilíndricas o Coordenadas Esféricas.

- **Vectores :** Un vector es una tupla n-dimensional que tiene una longitud(denominada módulo), una dirección y un sentido. Puede ser representado mediante una flecha. Dos vectores son iguales si tienen la misma longitud, dirección y sentido. Los vectores son entidades libres que no están ancladas a ninguna posición en el espacio. Algunas de las operaciones que se realizan con vectores son Suma, Resta, Módulo, Normalización, Producto Escalar, Producto Vectorial.

- **Quaternios :** Los quaternios (*quaternion*) fueron propuestos en 1843 por *William R. Hamilton* como extensión de los números complejos. Los *quaternion's* representan una 4-tupla. También pueden verse como un vector de tres dimensiones que representa una posición, dirección y sentido, más un cuarto factor que representa la rotación respecto al eje que define el vector. Se suelen utilizar en videojuegos para representar la rotación de determinados elementos como cámaras o huesos de las estructuras esqueletales de los modelos tridimensionales. Algunas de las operaciones que se realizan con los quaternions son Suma, Multiplicación, Inverso, o Interpolación linéales y esféricas.

- **Matrices y transformación en el espacio :** En la representación de gráficos *3D* es necesario contar con herramientas para la transformación de objetos básicos que compondrán la escena. En general podemos decir que una transformación toma como entrada elementos como vértices y vectores y los convierte de alguna manera para obtener un resultado diferente. Las transformaciones básicas utilizadas son la traslación, la rotación y el cambio de escala. A menudo cuando queremos cambiar la localización de un objeto habitualmente necesitamos especificar una combinación de translaciones y rotaciones en el mismo para obtener este resultado. En muchas aplicaciones gráficas los objetos deben transformarse geométricamente de forma constante (por ejemplo, en el caso de una animación, en la que en cada *frame* el objeto debe cambiar de posición). En el ámbito de los videojuegos, es habitual que aunque un objeto permanezca inmóvil, es necesario cambiar la posición de la cámara virtual para que se ajuste a la

interacción con el jugador. De este modo resulta crítica la eficiencia en la realización de estas transformaciones. Para conseguir una transformación en el espacio tridimensional que codifique la translación, rotación y escala se introducen las matrices, ya que mediante ecuaciones matemáticas podemos calcular una matriz, la cual si aplicamos mediante multiplicación a un conjunto de puntos del espacio, estos alterarán sus propiedades en base a las transformaciones codificadas en la matriz. Las operaciones típicas que realizamos con matrices son la composición y la transformación inversa.

- **Proyecciones :** Las proyecciones, cambios de plano o de sistemas de coordenadas o dimensiones suele ser otra de las operaciones típicas en los videojuegos y se utilizan ampliamente en el *pipeline* de renderizado, además de para obtener efectos como reflexiones o sombras en tiempo real.

Modelos *3D*

Utilizando las bases matemáticas introducidas anteriormente se describen y manipulan elementos tridimensionales para formar representaciones de objetos del mundo real, como objetos estáticos inanimados o como elementos dinámicos en el caso de personajes.

- **Vértices y caras :** Los puntos almacenados en un sistema de coordenadas locales definen los vértices del modelo y la relación entre ellos las caras o faces. Una cara es un polígono formado por tres o más vértices, aunque lo más común es utilizar como primitiva el triángulo pues el *hardware* de las tarjetas gráficas trabaja internamente con este tipo de primitivas. Un modelo *3D* no es más que una lista de vértices y una lista de triángulos. Cada triángulo esta formado por tres de esos vértices.

- **Texturas y mapeado *UV* :** Además de la definición de vértices y caras, los modelos suelen llevar asociadas texturas que se aplican directamente sobre estos vértices. Para ello se despliega la geometría tridimensional sobre un plano realizando una proyección y asociando cada punto con una coordenada de un espacio bidimensional. De esta forma cuando se renderiza en *3D* la geometría lleva pegada y ajustada esa imagen sobre el propio modelo.

Figura 3.10: Modelo tridimensional formado por triángulos [145]

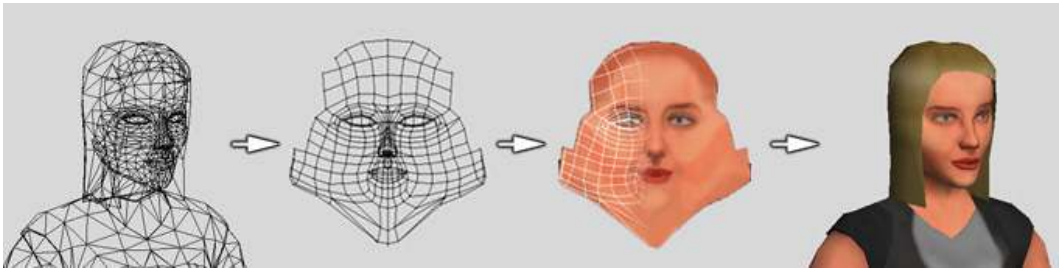


Figura 3.11: Mapeado de textura mediante *UV-Mapping* [155]

- **Materiales y *shaders* :** Las texturas aplicadas sobre el modelo están asignadas a materiales que definen a su vez el comportamiento de la superficie con respecto a las iluminación. Para ello los materiales van asociados con los *shaders* que son los que implementan estos algoritmos matemáticos. Estas dos cuestiones se tratarán mas adelante en este mismo apartado.

- **Animaciones :** En videojuegos es habitual contar con un sistema de animación basada en esqueletos. Para ello se asocian los vértices *3D* del modelo a determinados huesos virtuales y se define la deformación o peso de los mismos sobre la mall. A este proceso

de asociación se le llama *rigging* y permite animar el modelo posteriormente. Una vez realizada esta asociación se procede a animar el esqueleto mediante la aplicación de cinemática directa o cinemática inversa y empleando esquemas de interpolación basados en *frames* claves. La reproducción de esta animación en tiempo real da como resultado que los modelos tridimensionales se desformen conforme las restricciones y asociaciones de su esqueleto les permiten, dando como resultado la sensación de animación de esas estructuras tridimensionales.

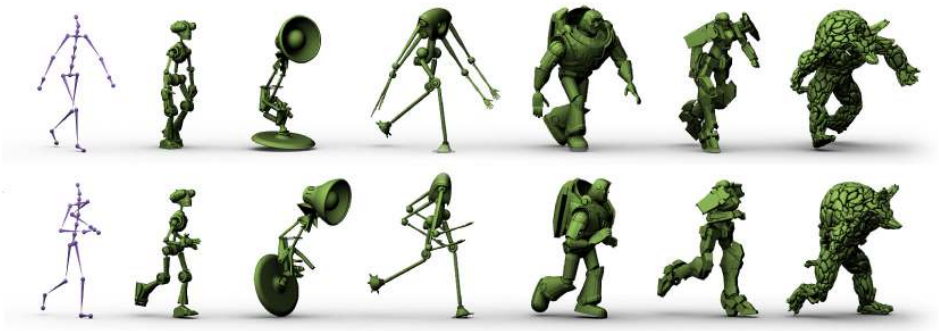


Figura 3.12: Rigging de modelos 3D [150]

- **Formatos de archivo para modelos *3D* :** En el desarrollo de videojuegos existen múltiples formatos *3D* que nos permiten almacenar información sobre los modelos, las coordenadas de texturas, el esqueleto que llevan asociado, o la animación del propio esqueleto. Algunos de los formatos más utilizados son *.OBJ*, *.FBX*, *.Blend*, *.X*, *.3DS*, etc

Materiales

Los materiales describen las propiedades físicas de los objetos relativas a como reflejan la luz incidente. Obviamente, el aspecto final obtenido será dependiente tanto de las propiedades del material, como de la propia definición de las fuentes de luz. De este modo materiales e iluminación están íntimamente relacionados.

Usando algoritmos y partiendo de las propiedades geométricas y de materiales especificadas numéricamente es posible simular la reflexión y propagación de la luz en una escena. A mayor precisión mayor nivel de realismo en la imagen resultado.

A un alto nivel de abstracción, podemos realizar una primera taxonomía de métodos de renderizado [26] entre aquellos que realizan una simulación de iluminación local, teniendo en cuenta únicamente una interacción de la luz con las superficies, o los métodos de iluminación global que tratan de calcular todas las interacciones de la luz con las superficies de la escena. La potencia de cálculo actual hace inviable el uso de métodos de iluminación global. Así, en los motores gráficos actuales los materiales definen cómo se refleja la luz en los objetos, empleando un esquema de iluminación local. En cierto modo, el sombrado es equivalente a pintar con luz. Los modelos de sombreado más ampliamente utilizados en gráficos interactivos, que fueron inicialmente desarrollados en los años 70 son los siguientes:

- **Sombreado difuso :** Muchos objetos del mundo real tienen una acabado eminentemente mate (sin brillo). Este tipo de materiales reflejan la luz en todas las direcciones, debido principalmente a las rugosidades microscópicas del material. Como efecto visual, el resultado de la iluminación es mayor cuando la luz incide perpendicularmente en la superficie. La intensidad final viene determinada por el ángulo que forma la luz y la superficie. En su expresión más simple, el modelo de reflexión difuso de *Lambert* dice que el color de una superficie es proporcional al coseno del ángulo formado entre la normal de la superficie y el vector de dirección de la fuente de luz.

- **Sombreado especular :** Es el empleado para simular los brillos de algunas superficies, como materiales pulidos, pintura plástica, etc. Una característica principal del sombreado especular es que el brillo se mueve con el observador. Esto implica que es necesario tener en cuenta el vector del observador.

- **Sombreado ambiental :** Esta componente básica permite añadir una aproximación a la iluminación global de la escena. Simplemente añade un color base independiente de la posición del observador y de la fuente de luz. De esta forma se evitan los tonos absolutamente negros debidos a la falta de iluminación global, añadiendo este término constante.

- **Sombreado de emisión :** Finalmente este término permite añadir una simulación de la iluminación propia del objeto. No obstante, debido a la falta de interacción con otras superficies (incluso con las caras poligonales del propio objeto), suele emplearse como una alternativa al sombreado ambiental a nivel local. El efecto es como tener un objeto que emite luz pero cuyos rayos no interactúan con ninguna superficie de la escena.

Figura 3.13: Modelos gráficos de sombreado [26]

El sombreado final de la superficie se obtiene como combinación de los cuatro modos de sombreado anteriores. Esta aproximación es una simplificación del modelo de reflexión físicamente correcto definido por la Función de Distribución de Reflactancia Bidireccional *BRDF*(*Bidirectional Reflactance Distribution Function*). Esta función define cómo se refleja la luz en cualquier superficie opaca, y es empleada en motores de renderizado fotorrealistas.

Fuentes de luz

Las fuentes de luz pueden representarse de diversas formas, dependiendo de las características que queramos simular en la etapa de renderizado. En videojuegos suelen permitirse tres tipos de fuentes de luz directamente soportadas por el *hardware* de aceleración gráfico:

- Las fuentes puntuales (*point lights*) irradian energía en todas las direcciones a partir de un punto que define su posición en el espacio. Este tipo de fuentes permite variar su posición pero no su dirección. En realidad, las fuentes de luz puntuales no existen como tal en el mundo físico.
- Uno de los tipos de fuentes más sencillo de simular son las denominadas fuentes direccionales (*directional lights*), que pueden considerarse fuentes situadas a una distancia muy grande, por lo que los rayos viajan en una única dirección en la escena (son paralelos entre sí). El sol podría ser modelado mediante una fuente de luz direccional. De este modo, la dirección viene determinada por un vector l (especificado en coordenadas universales). Este tipo de fuentes de luz no tienen por tanto una posición asociada (únicamente dirección).
- Finalmente os focos (*spot lights*) son en cierto modo similares a las fuentes de luz puntuales, pero añadiendo una dirección de emisión. Los focos arrojan luz en forma

cónica o piramidal en una dirección específica. De este modo, requieren un parámetro de dirección, además de dos ángulos para definir los conos de emisión interno y externo.

Las fuentes de luz permiten especificar multitud de parámetros y propiedades, como el color difuso y especular. Una fuente de luz puede definir un color de iluminación difuso (como si el cristal de la bombilla estuviera tintado), y un color diferente para el brillo especular. Ambas propiedades están directamente relacionadas con el modo en el que reflejarán la luz los materiales.

Shaders

Un shader [9] es un pequeño programa que se ejecuta en la *GPU* y que permite implementar un comportamiento personalizado en las diferentes etapas del *pipeline* de renderizado. Alterando el cálculo sobre los vértices, lineas, triángulos o *pixeles* que se calculan.

- ***Vertex Shader* :** Los *vertex shaders* permiten aplicar transformaciones y deformaciones a nivel de vértice. Este *shader* se aplica en la primera etapa del *pipeline* de la *GPU*. En esta etapa los flujos de datos que la *CPU* envía a la tarjeta son procesados y se aplican las matrices de transformación especificadas por el usuario. A este nivel, el *vertex shader* sólo trabaja con la información relativa a los vértices como la posición, vector normal, color y coordenadas de textura. Algunas operaciones clásicas que se implementan empleando este tipo de shaders son efectos de lente, ojo de pez, deformaciones de objetos o animaciones de las coordenadas de textura.

- ***Geometry Shader* :** Los *geometry shaders* facilitan la creación y destrucción de primitivas geométricas en la *GPU* en tiempo de ejecución (vértices, líneas y triángulos). Este tipo de *shaders* se emplean para la simulación de pelo, para encontrar los bordes de los objetos o para la implementación de algunas técnicas avanzadas como la simulación de telas o diversas técnicas de subdivisión como la *tesselación*.

- ***Pixel Shader* :** A nivel de *pixel shader* se pueden aplicar operaciones a los *pixeles* resultantes de la rasterización, permitiendo definir complejas ecuaciones de sombreado que serán evaluadas para cada *pixel* de la imagen. El *pixel shader* tiene influencia únicamente sobre el fragmento que está manejando. Esto implica que no puede aplicar ninguna transformación sobre fragmentos vecinos. El uso principal que se da a este tipo de shaders es establecimiento mediante código de color y la profundidad asociada

al fragmento. Actualmente se emplea para aplicar multitud de efectos como *normal mapping* [19] o reflexiones.

Los *shaders* son ampliamente utilizados en videojuegos para implementar diferentes efectos [25] como el comportamiento de las superficies con respecto a la luz incidente, la modificación de la altura de determinados vértices para simular desplazamientos de las mayas *3D*, la simulación de sombreado de geometrías virtualmente inexistentes como en el caso del *normal mapping* o incluso para suavizar la geometría existente añadiendo más polígonos como cuando se usa teselación en *DirectX11*.

En la Figura 3.14 puede apreciarse la misma geometría, a la izquierda renderizada con un pipeline estándar y a la derecha renderizada aplicando *pixel shaders* sobre el *pipeline* de renderizado para obtener una imagen resultante con un aspecto diferente. Entre estos dos resultados diferentes lo único que cambia es el uso de diferentes *shaders* programables pues la geometría original que se envía desde la *CPU* a la *GPU* es la misma.

Figura 3.14: Renderizado Sin *shaders* VS Renderizado Con *shaders* [151]

Modelos de iluminación

En función del tipo de shaders que implementemos para nuestro videojuego podremos obtener iluminación por caras, por vértices, por pixel o incluso emplear avanzados sistemas

de iluminación *HDRi* a partir de imágenes de alto rango dinámico [1] como origen de la luz.

El uso de una u otra técnica suele implementarse a nivel de *shader* y varía en rendimiento, desde una de las más básicas como la iluminación por vértice hasta una de las más complejas y habituales de hoy día como es la iluminación por *píxel*.

Billboards

Un *billboard* es un polígono con una textura [26] que cuenta con un vector de orientación sensible a la posición de la cámara. Dicho vector se toma como referencia para cambiar la orientación y para permanecer siempre perpendicular a la cámara. Esta técnica suele emplearse para colocar elementos en el escenario como hierba, arboles en la distancia u objetos secundarios que en lugar de renderizarse en 3D, hacen uso de una textura que se renderizará sin deformación sobre un plano que se estará siempre siempre orientado con su vector normal apuntando hacia la cámara.

Figura 3.15: Árboles renderizados utilizando *billboards* [140]

En determinadas ocasiones la textura del plano puede ser animada haciendo uso de texturas de *Sprites 2D*.

Efectos de partículas

Un sistema de partículas [20] es un conjunto de objetos separados en movimiento de acuerdo a algún algoritmo. Su objetivo principal es la simulación de fuego, humo, explosiones, flujos de líquido u otros efectos en los cuales interviene la interacción de muchos objetos.

Cada uno de esos objetos del sistema de partículas puede ser un objeto tridimensional, un punto con color en el espacio o un *billboard* individual, siendo esto último lo más común. En el caso de usar *billboards* es común que la textura individual que se pinta en el *billboard* sea animada, así como el movimiento y flujo de vida de esta partícula individual.

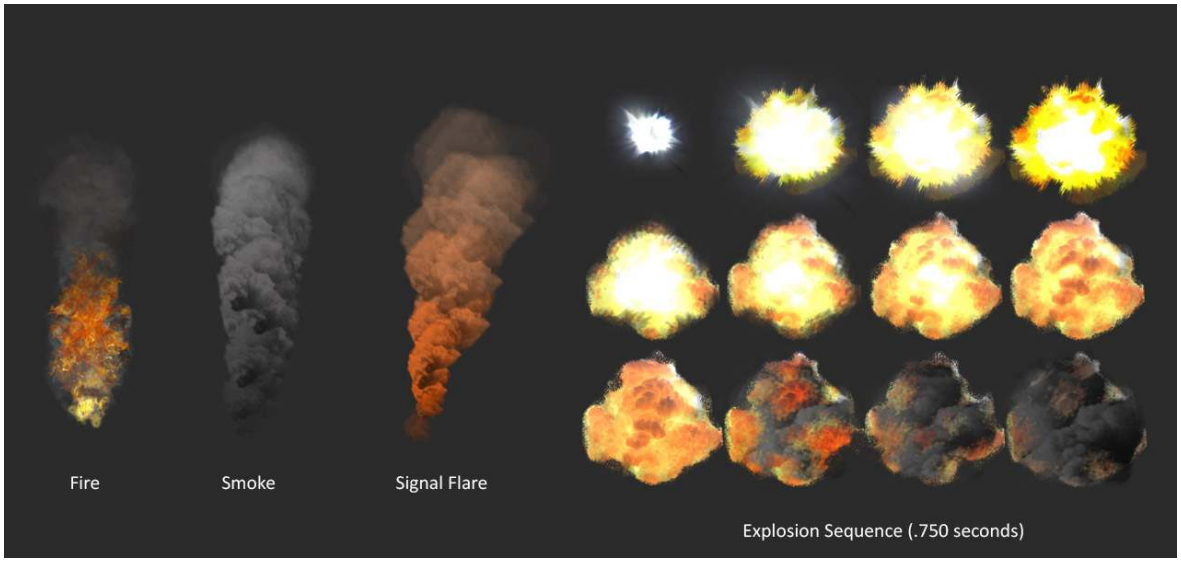


Figura 3.16: Textura en los *billboards* de efectos de partículas [146]

Este tipo de efectos de partículas son ampliamente utilizados en el desarrollo de videojuegos y le aportan dinamismo y espectacularidad visual.

En la Figura 3.17 puede apreciarse su uso para recrear un choque de vehículos donde se ha dotado a la escena de humo y efectos de chispas haciendo uso de un sistema de partículas de tipo *billboard*.

Figura 3.17: Ejemplo de uso de partículas [147]

Reflexiones precalculadas

Para representar reflexiones en videojuegos es común utilizar la técnica de *enviroment mapping*. En el caso de utilizar un cubo como mecanismo de precálculo de la reflexión (*cube mapping*) es necesario realizar un renderizado previo en las seis direcciones del espacio a partir de un punto de origen.

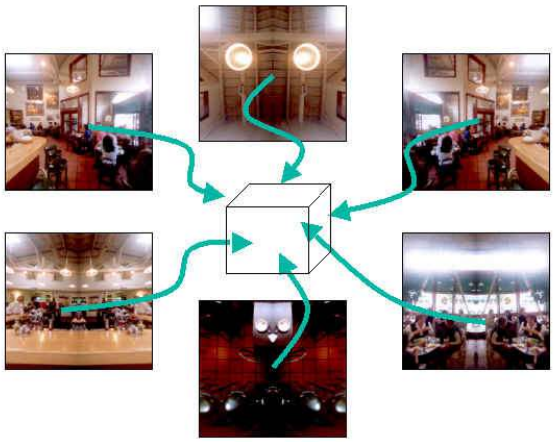


Figura 3.18: Renderización de *cubemaps* [142]

El resultado de renderizar estas imágenes formarán un cubo que será proyectado sobre los objetos que se encuentren en este punto del espacio. En la figura 3.19 podemos ver como esta técnica se ha empleado sobre el punto del espacio que ocupa la esfera central. Nótese

que la reflexión se visualizará de forma diferente si cambia el punto de vista de la cámara pero no se actualizará si cambia el entorno. Si por ejemplo la calavera se moviera, en la esfera central seguiría reflejándose sobre el punto original donde se generó cuando se hizo el renderizado estático del *cubemap*.

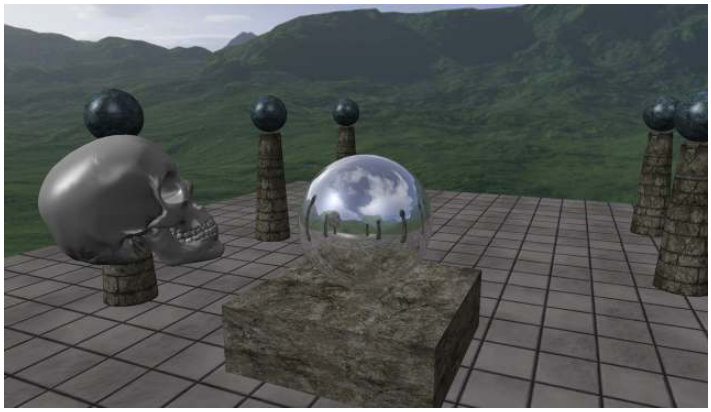


Figura 3.19: Uso de *cubemap* en renderización de escena en tiempo real [141]

Reflexiones dinámicas

Si se quiere disponer de reflexiones en tiempo real para que los objetos puedan reflejar en tiempo real los cambios del entorno dinámicamente es necesario emplear *cubemaps* que se actualicen cada *frame*.

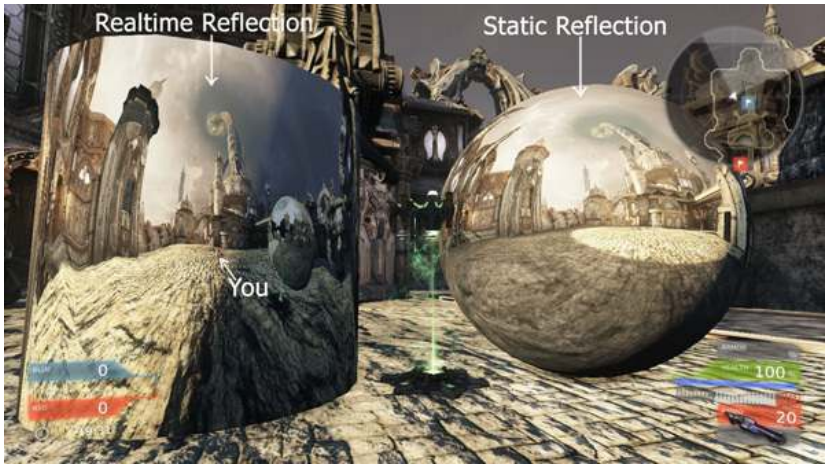


Figura 3.20: *Cubemap* dinámico VS *Cubemap* estático [148]

También se emplean otras técnicas según el tipo de geometría. Por ejemplo en planos

consiste en calcular la matriz de reflexión de la cámara con respecto a la superficie, realizar un renderizado a textura aplicando a la cámara esa matriz de deformación y ajustar la proyección de las coordenadas de textura del plano para proyectar el resultado del render a textura.

Figura 3.21: Reflexiones sobre plano en tiempo real [149]

En la practica se suelen utilizar reflexiones de *cubemap* estáticos para determinados objetos de menor importancia, reflexiones de *cubemap* dinámicos para objetos con geometría compleja que se desplazan a gran velocidad como vehículos, y reflexiones planas para elementos como el suelo o superficies de agua.

Sombras precalculadas

El renderizado de sombras es otro de los elementos que más realismo aporta en un videojuego. Lo habitual es precalcular las sombras de los elementos estáticos del escenario mediante la técnica de *lightmapping* empleando algoritmos de *Ray-tracing*. Esta técnica se discute en mayor detalle en el apartado “Mecanismos de optimización de rendimiento”. Como resumen diremos que esta técnica consiste en representar la sombra sobre el entorno tridimensional haciendo uso de la técnica de *UV-Mapping* sobre un segundo canal de información.

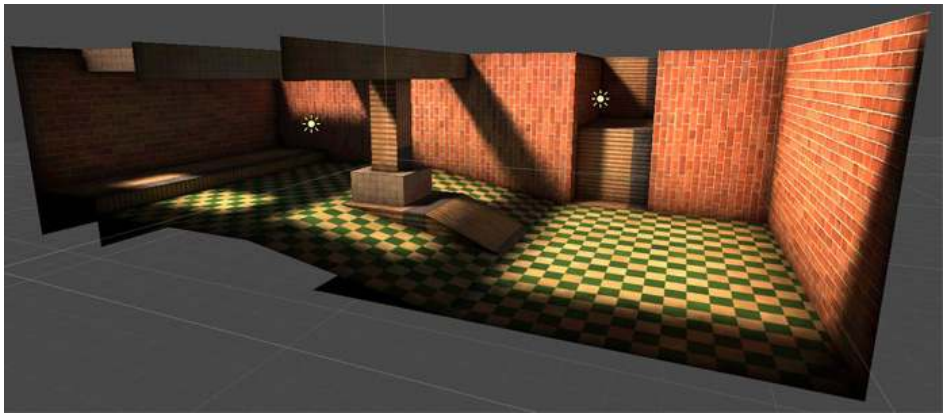


Figura 3.22: Uso de sombras precalculadas en escena tridimensional [152]

Sombras dinámicas

Adicionalmente a las sombras precalculadas sobre el escenario del videojuego se representarán otra serie de actores que posen movimiento o son interactivos y para los cuales es necesario renderizar la sombra en tiempo real.

En la imagen inferior podemos apreciar cómo para el entorno se han precalculado las sombras de forma estática utilizando sombras difusas, mientras que la silla que puede moverse y es interactiva proyecta una sombra dura. Esta sombra se renderiza en tiempo real y cambiará en función de la posición que ocupe la silla en el espacio tridimensional.

Figura 3.23: Integración de sombreado en tiempo real sobre escena con sombreado precalculado [153]

La imagen de la Figura 3.23 tiene bastante contraste de calidad entre las sombras precalculadas y las sombras dinámicas, no llegando a quedar bien integradas. No obstante los videojuegos modernos son capaces de renderizar las sombras de tiempo real mediante algoritmos que las hacen indistinguibles de las sombras precalculadas obteniendo una integración total de la imagen resultante.

Para la implementación de esta técnica se pueden seguir diferentes aproximaciones como las sombras basadas en *Stencil Buffer*, las sombras basadas en render a textura o las sombras volumétricas.

Efectos de postprocesado

Los efectos de postprocesado supone emplear *shaders* a pantalla completa sobre la imagen resultante del renderizado. Mediante este método se puede alterar la imagen en base a otras imágenes renderizadas de forma auxiliar o en base a los algoritmos que se implementen en estos *shaders*.

Se utilizan para realizar efectos [62] como la corrección del color de la imagen resultante o la recreación de efectos como la profundidad de campo donde se simula el enfoque de una lente. Este tipo de efectos mejoran mucho la calidad visual y son ampliamente utilizados en los videojuegos modernos.

- ***Antialiasing* :** Esta técnica se emplea para suavizar las líneas y eliminar los *jagged* que aparecen cuando se rasteriza la imagen resultante del renderizado. Hay algoritmos que permiten suavizar estos defectos mediante el uso de *shaders* de postproceso y hay otra serie de algoritmos que requieren un renderizado en *oversampling* tomando dos, cuatro u ocho muestras por cada *pixel* de los que aparecen en la imagen resultante.

Figura 3.24: Escena virtual renderizada (Izquierda) sin *Antialiasing* y (Derecha) con *Antialiasing* [139]

- ***Color Grading*** **:** El propósito del *color grading*, *tone mapping* o efectos como la corrección de color contribuyen a adaptar la imagen resultante a una paleta de color que encaje más artísticamente con las sensaciones que se quiere provocar en el jugador. De modo que la misma escena puede parecer otra diferente ya sea si se aplican colores cálidos o fríos y reflejando con ello diferentes sentimientos cuando se produce su visionado.

Figura 3.25: Imagen virtual renderizada utilizando el efecto de corrección de color [139]

- ***Bloom*** **:** Este efecto recrea un fenómeno de la luz que se percibe en el mundo real y que contribuye a aumentar el realismo en la imagen resultante. Puede apreciarse a simple vista cuando se miran directamente fuentes de luz muy brillantes que se encuentran sobre un fondo mucho más oscuro.

Figura 3.26: Imagen virtual renderizada utilizando el efecto *Bloom* [143] [139]

- ***Depth Of Field*** **:** Este efecto aplica desenfoque basado en la distancia al punto focal para simular el efecto que ocurre en las cámaras del mundo real. También se emplea

para centrar el foco de interés de la imagen sobre determinados elementos dando un aspecto más cinematográfico.

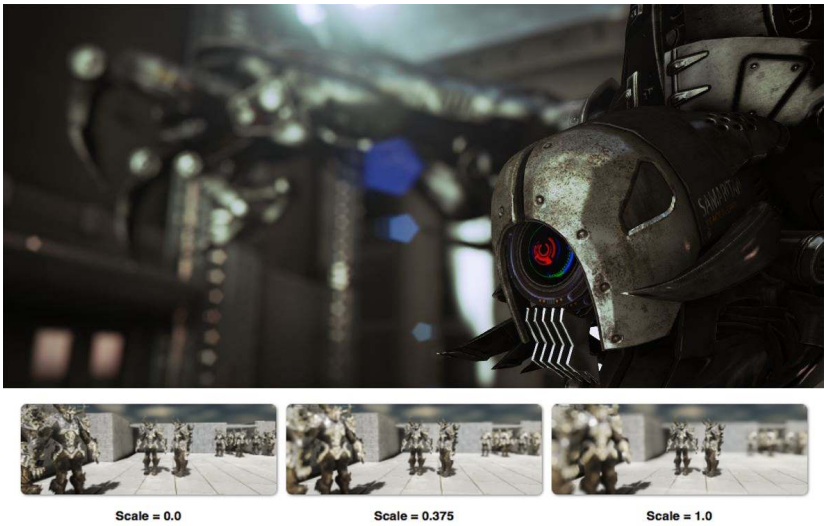


Figura 3.27: Imagen virtual renderizada utilizando el efecto de postprocesado *Depth Of Field* [139]

- ***Eye adaptation*** **:** La adaptación del ojo o exposición automática es un efecto que se aplica para recrear el fenómeno de ajuste del ojo humano cuando se pasa de unas condiciones de poca luz a mucha luz o viceversa.

Figura 3.28: Imagen virtual renderizada utilizando el efecto de adaptación al ojo [139]

- ***Lens flare*** **:** Este efecto recrea los efectos en las lentes de la cámaras que hacen que se muestren imperfecciones en el color cuando se enfoca directamente a una fuente de luz muy potente.

Figura 3.29: Imagen virtual renderizada con *Lens Flares* [139]

- ***Vignette* :** Este efecto simula el oscurecimiento cerca de los bordes de las lentes de las cámaras del mundo real.

Figura 3.30: Imagen virtual renderizada utilizando el efecto de postprocesado *Vignetting* [139]

- ***Color aberration* :** Este efecto simula el desplazamiento de color que se produce en las cámaras del mundo real cerca de los bordes de la imagen.

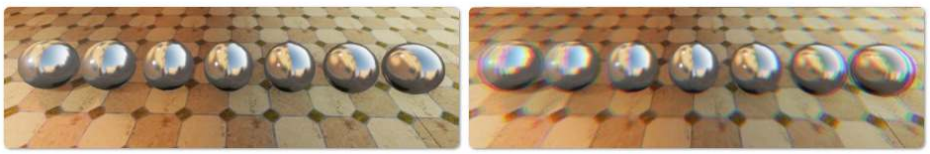


Figura 3.31: Imagen virtual renderizada utilizando el efecto de postprocesado *Color Aberration* [139]

- ***Motion blur* :** Este efecto de desenfoque de movimiento simula el rastro dejado en una fotografía o en la retina por objetos que se mueven a mucha velocidad, percibiendo como borroso los contornos de los elementos que poseen más velocidad.

Figura 3.32: Imagen virtual renderizada utilizando el efecto de postprocesado *Motion Blur* [144]

Mecanismos de optimización de rendimiento

Para garantizar el correcto rendimiento del videojuego se emplean técnicas de optimización [8] que permiten alcanzar las elevadas tasas de renderización de *frames* requeridas para percibir interactividad en los videojuegos.

- **Técnicas de partición del espacio :** Las técnicas de partición del espacio permiten precalcular qué zonas del mapa son visibles desde cada uno de los puntos del espacio y cuáles han sido ocluidas por otros elementos que se encuentran delante de la cámara para ese punto del espacio. Estas técnicas permiten reducir la cantidad de geometría a pintar. El precalcular este proceso agiliza bastante el pintado cuando tenemos mapas en los que la propia geometría del escenario oculta otras construcciones que se encuentran detrás. También descarga el intercambio de datos entre la memoria principal y la memoria de vídeo.

- ***LOD* :** La técnica de *Level of Detail* consiste en disponer de varias versiones de la geometría tridimensional que se está representando con diferentes grados de detalle de modo que en función de la distancia de la cámara al modelo *3D* que se va a pintar se utiliza una de más o menos resolución. De esta forma los elementos lejanos se pintarán utilizando la versión de baja calidad pero el resultado será muy parecido, pues a dicha distancia no se perciben los detalles.

- ***Lightmapping*** **:** La técnica de *ligthmapping* consiste en precalcular la iluminación de la escena y plasmar las sombras y colores que alcanza cada superficie sobre una textura. Para ello se despliega la textura sobre la superficie tridimensional utilizando la técnica de *UV-Mapping* sobre el segundo canal de *UV* de los modelos *3D*. Esto permite tener calculadas y ”pegadas“ las sombras de los elementos estáticos sobre la escena, evitando tener que realizar este tipo de cálculos en tiempo real.

- ***Lightprobes*** **:** La técnica de *lightprobes* consiste en precalcular la iluminación de la escena mediante *Ray-tracing*. Para ello se captura la iluminación indirecta en determinados sumideros de luz que se sitúan sobre la escena. Estos puntos guardarán información sobre la cantidad de luz y el color recibidos en ese punto del espacio. Después en tiempo real se realizará una media geométrica para la posición, donde se pretende conocer la iluminación indirecta en ese punto del espacio teniendo. En este cálculo se tiene en cuenta la cantidad de luz recibida en los sumideros precalculados que se encuentren cercanos en esa región del espacio. Este tipo de técnicas permite simular iluminación indirecta de forma dinámica sobre personajes y otros elementos de la escena. La forma en que se logra es alterando la componente ambiente del material sin tener que hacer el cálculo de la misma en tiempo real, algo que sería demasiado costoso para alcanzar una tasa de frames adecuados. Es habitual utilizar esta técnica en conjunto con la técnica de *ligthmapping*.

Simulación física

La simulación física [5] es una de las partes cruciales de muchos géneros de videojuegos y una de las que originan mayor sensación de interactividad. En la programación de simulación física es necesario definir cómo se comporta cada elemento de la escena con respecto a las fuerzas de la gravedad a las cuales están sometidos, así como a efectos tales como el viento, o las fuerzas individuales que ejerce cada uno de los elementos.

También se utiliza la simulación física para realizar el cálculo de colisiones que nos permite determinar cuándo el personaje ha chocado con una pared, ha entrado en una determinada zona en la que debe disparar un evento o ha sido alcanzado por un proyectil enemigo.

Lo más básico es que el sistema de gestión de física permita la interacción entre los llamados cuerpos rígidos, es decir elementos geométricos que no sufren deformación. En los videojuegos modernos también se hace uso de otros elementos como la simulación física de

la deformación de los músculos, pelo o uso de cuerpos blandos en los personajes.

Otra de las áreas en las que se emplea la simulación física es para el cálculo de los movimientos de vehículos y las fuerzas que generan cada uno de las ruedas o elementos de propulsión de los mismos.

No hay que olvidar que la simulación física también tiene estrecha relación con el sistema de animaciones ”basada en esqueletos“ de los personajes y en determinados casos como en el caso de los *ragdolls* se usa para calcular la interacción realista de un esqueleto bajo las fuerzas a las que se ve sometido, restringiendo el movimiento de cada uno de los huesos en determinados ejes y aplicando fuerzas individuales sobre otros en función de la posición del cuerpo en el espacio tridimensional.

Inteligencia artificial

Un videojuego que recrea comportamientos de actores virtuales controlados por el programa debe proporcionar la ilusión de que estos entes toman decisiones y acciones en base a unos criterios que parecen lógicos para la consecución de sus objetivos. Para ello se dota a cada uno de los elementos de una inteligencia artificial que es gestionada por el propio videojuego y que tiene como misión simular el comportamiento realista y adaptable al entorno en el que se encuentran [5].

A menudo estos actores virtuales deben tomar decisiones de forma individual o como grupo y hacerlo de una forma que le parezca lógica al jugador.

- ***NPC's* :** Un *NPC* (*non-player character*) es un personaje no jugable que forma parte del mundo virtual pero sobre el cual el jugador no puede tomar decisiones o controlarlo directamente. Es normal que este tipo de entidades tengan una inteligencia artificial [22] propia y sean reactivos a las acciones del propio jugador. Este tipo de actores pueden ser enemigos, personajes amigos o personajes meramente decorativos pero interactivos que forman parte del mundo virtual.

- **Autómatas y estados :** Para algunos comportamientos a veces es suficiente con un autómata definido por estados y transiciones que en función de los eventos que perciben a su alrededor, cambian su estado para manifestar un comportamiento u otro. Este tipo de modelos más simple supone una alternativa en contraposición a otro tipo de inteligencias artificiales que utilizan heurísticas más complejas para dar un comportamiento

más adaptable a todo tipo de situaciones.

- **Búsqueda de caminos :** La búsqueda de caminos es otra de las áreas de interés en lo que a inteligencia artificial en videojuegos se refiere. Esto es debido a que en determinados géneros, como los videojuegos de estrategia, las unidades individuales deben encontrar el camino hacia un punto determinado del mapa, para ello se utilizan algoritmos de inteligencia artificial como el *pathfinding* o $A*$.

Gestión de recursos *hardware*

Un videojuego es un software especializado de alto rendimiento en tiempo real con unas necesidades de respuesta a la entrada de usuario que requieren un renderizado de 30 o 60 imágenes por segundo mientras se gestiona la persistencia de los eventos que suceden, la reproducción de contenido multimedia, o el uso de la red para videojuegos online. Conseguir este tipo de resultados requiere una gestión optima de cada uno de los sistemas del *hardware* que se utilizan.

- **Gestión de procesador :** El procesador [4] se utiliza para calcular los sucesos y resultados a los eventos del mundo virtual, así como para gestionar la inteligencia artificial de los enemigos y hacer el cálculo de los elementos físicos del videojuego. Para que el procesador no suponga un cuello de botella se implementan métodos que permitan segmentar los cálculos de este tipo de tareas de manera secuencial y discreta en los diferentes *frames* de renderizado. A menudo se utiliza procesamiento multihilo para garantizar una concurrencia adecuada de cada uno de los procesos que funcionan en un videojuego.

- **Gestión de memoria :** La memoria principal debe guardar los gráficos, estructuras e información utilizada por cada elemento del juego. Todo lo que se ve en pantalla debe estar en memoria, por lo que es normal utilizar políticas de remplazo que favorezcan el tener los recursos necesarios cargados en memoria para el momento que van a ser utilizados. La memoria de vídeo guarda información como texturas, listas de pintado de vértices o información de *shaders*. A menudo la memoria principal suele ser más numerosa que la memoria de vídeo y en determinados momentos se produce un intercambio de información desde la memoria principal a la memoria de vídeo. En algunas arquitecturas como los dispositivos móviles, la memoria principal y la memoria de vídeo está unificada, facilitando así el uso versátil de la misma.

- **Gestión de gráficos :** La tarjeta gráfica (*GPU*), o el chip gráfico en el caso de los dispositivos móviles, es el encargado de renderizar la información como texturas, modelos *3D* y *shaders* para dar una salida a pantalla. Cuenta con una memoria de video de alta velocidad donde se almacenan los datos que van a ser pintados por pantalla. A menudo tienen una arquitectura paralela que le permite procesar diferentes fragmentos de forma simultanea para obtener un rendimiento que permita alcanzar altas tasas de *frames* por segundo. Se debe realizar una gestión de gráficos que permita balancear la carga gráfica para que el pintado de *frames* se realice de forma constante, sin tener altibajos que repercutan en cómo se percibe el resultado por parte del usuario.

- **Gestión de sonido y multimedia :** La tarjeta de sonido o chip de sonido (en el caso de los dispositivos móviles) es la parte encargada de reproducir sonidos. En casos como los dispositivos móviles es necesario hacer una gestión adecuada de sonidos para no reproducir más de un número específico de sonidos simultáneos, pues disponen de pocas unidades de descompresión de sonido por *hardware*. Este hecho requiere que el gestor de sonidos tenga que decidir qué sonidos reproducir y cuáles parar en momentos de máxima carga.

- **Gestión de entrada y salida :** El usuario se comunica con el videojuego a través de una pantalla táctil, un ratón, un *joystick*, u otro dispositivo de control. Es necesario recoger esta entrada con una frecuencia adecuada y dar una respuesta a tiempo para que el jugador no perciba *lag* entre los comandos que introduce y el resultado que obtiene por pantalla.

- **Gestión de almacenamiento y lectura de información :** El videojuego debe gestionar el progreso [10] en el mismo permitiendo volver más tarde y continuar por donde se quedó el jugador. Esto requiere la gestión de la persistencia de la información teniendo que controlar el almacenamiento y lectura de información sobre la partida e incluso el cálculo de estadísticas de juego para otorgar logros u objetivos secundarios en determinadas circunstancias.

- **Gestión de red :** En el caso de los videojuegos *online* se debe gestionar el *hardware* de red para implementar una arquitectura que permita integrar la entrada de usuarios que se encuentran en otra ubicación física. Este tipo de videojuegos añaden una complejidad extra, pues deben garantizar que el videojuego funciona en tiempo real para todos los jugadores y que los eventos del mundo virtual son consistentes y están sincronizados.

3.4. COMERCIALIZACIÓN

3.4.1. Estado de la industria

Entendemos por industria del videojuego al conglomerado de empresas y desarrolladores que producen videojuegos generando con ello puestos de trabajo y capital que se queda dentro del país donde se lleva a cabo la actividad.

Según el reciente estudio publicado en el libro blanco del desarrollo Español de los videojuegos [27] que cuenta con el apoyo del ICEX podemos extraer que el estado de la industria del videojuego en España pasa por uno de los mejores momentos de su historia, comparable sólo a la edad de oro del videojuego en España.

Algunas de los resultados y datos que arroja el estudio son por ejemplo el cambio en la cadena de valor en la creación de videojuegos. Anteriormente los agentes contaban con un papel muy definido, sin tener libertad ni posibilidad de desarrollar nuevos modelos de negocio.

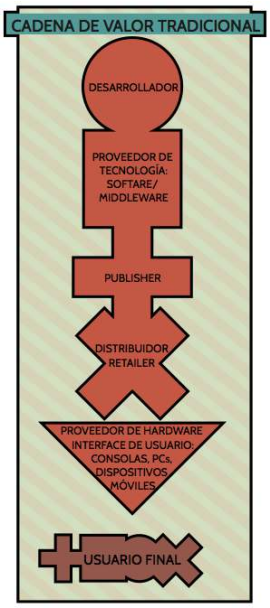


Figura 3.33: Cadena de valor tradicional del videojuego [27]

La cadena de valor tradicional estaba compuesta por desarrolladores, encargados de la creación del producto, editoras encargadas de adaptar el producto a las necesidades de cada mercado y distribuidores encargados de comercializar el producto en cada mercado nacional.

Puede apreciarse esta estructura tradicional de la cadena de valor del videojuego en la Figura 3.33.

Un aspecto esencial en la cadena de valor tradicional era la financiación de los videojuegos. La producción de videojuegos para consola y *PC* se caracteriza por unos altos costes de desarrollo que eran asumidos por el *publisher*. Siendo este el que ejercía el control sobre la empresa que desarrollaba el producto.

La irrupción de *internet* y los nuevos métodos de distribución han comenzado a reorganizar las funciones y la forma en la que interactúan los actores en cada uno de los niveles de la cadena de valor. Uno de los agentes más afectados por la irrupción de *internet* son los encargados de la distribución logística, habiendo dejado de ser relevante este proceso en el segmento de los bienes digitales donde el coste de distribución se considera marginal.

Actualmente el *publisher* opta por distribuir los videojuegos directamente a través de las propias plataformas de venta que proporcionan los fabricantes del *hardware*.

Por otro lado los desarrolladores independientes también son aceptados en estas plataformas de venta proporcionadas por los fabricantes, por lo que después de mucho tiempo se abre la puerta para que un producto de un desarrollador independiente se encuentre en el mismo canal de distribución que el producto de un *publisher* ya establecido.

En este punto lo único que declinará la venta a favor de un producto u otro será la calidad del producto en cuestión y la cantidad de recursos de marketing que se inviertan en él para darlo a conocer. No obstante este cambio de reglas del mercado abre la puerta a que nuevas empresas vetadas anteriormente por la maquinaria comercial establecida tengan ahora una oportunidad de vender sus productos. Este hecho hace que la cadena de valor actual se establezca como se puede apreciar en la Figura 3.34.

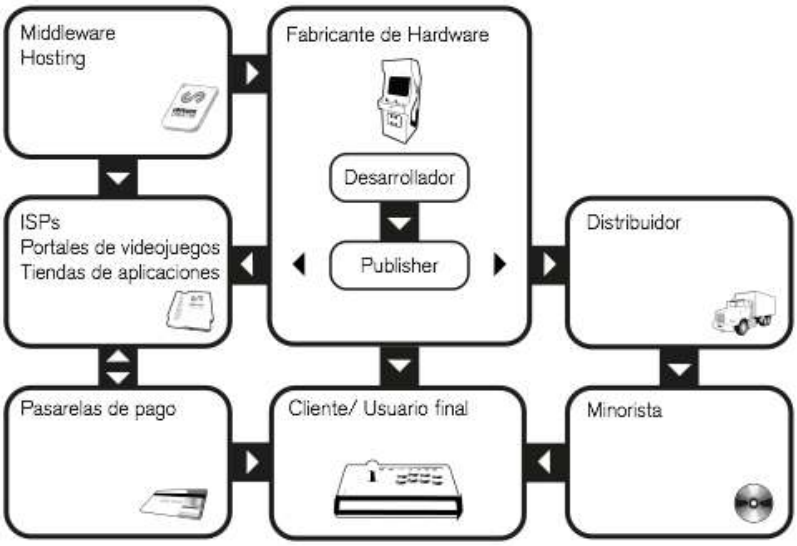


Figura 3.34: Cadena de valor actual del videojuego [27]

A su vez la entrada de nuevos competidores a un mercado donde los *publishers* ya no tienen el control total ha desencadenado el nacimiento de nuevos modelos de negocio que ser perfilan como muy rentables, quedando actualmente una segmentación clara en los siguientes:

- ***Pay to play* o *Premium*:** Es el modelo tradicional, el videojuego se compra en formato físico o pagado por su descarga.
- ***Freemium* :** Modelo de negocio que consiste en facilitar al usuario una versión gratuita del videojuego. Se trata de la versión completa pero sólo permite jugar hasta cierto punto. A partir del cual es necesario pagar el precio total del producto para poder continuar. A efectos prácticos se trata de añadir el modelo de negocio *Pay to Play* a la propia demostración gratuita del videojuego.
- ***Free to play* (*F2P*) :** Modelo de negocio que consiste en facilitar al usuario una versión gratuita del videojuego con la posibilidad de adquirir mejoras o más funcionalidades mediante micro pagos dentro de la aplicación.
- **Publicidad :** El videojuego incluye publicidad y obtiene ingresos por ello. De esto hay varias alternativas:
  - ***In Game Advertising* :** Los juegos contienen publicidad dentro de ellos.

- ***Around-Game Advertising*** **:** La publicidad rodea al juego, pudiendo aparecer antes o después de jugar. Esto es bastante común en juegos online.
- ***Advergaming*** **:** Una marca comercial promociona específicamente todo el juego. Por ejemplo cuando se hace una nueva película, se realiza un *advergaming* gratuito que promociona la película y atrae público a su visionado. También lo hacen marcas de refrescos o comida.

Otro de los efectos en el cambio de la cadena de valor tradicional del videojuego es el surgimiento de nuevas figuras que toman ahora gran valor como los localizadores y los dinamizadores. Los localizadores ya formaban parte de la cadena de valor tradicional, no obstante están adquiriendo un papel muy relevante reforzado por la internacionalización de los videojuegos debido a *internet*. Por otro lado el dinamizador adquiere un papel significativo administrado comunidades virtuales y aumentando la relevancia de los productos para los que trabaja.

En estos últimos años también ha habido un auge de los juegos serios (*serious games*), que son aquellos que se utilizan con fines diferentes al entretenimiento, ya sea educación, medicina, salud, cultura o entrenamiento para defensa. Cada vez son más utilizados y ya hay empresas que se dedican a realizar este tipo de videojuegos.

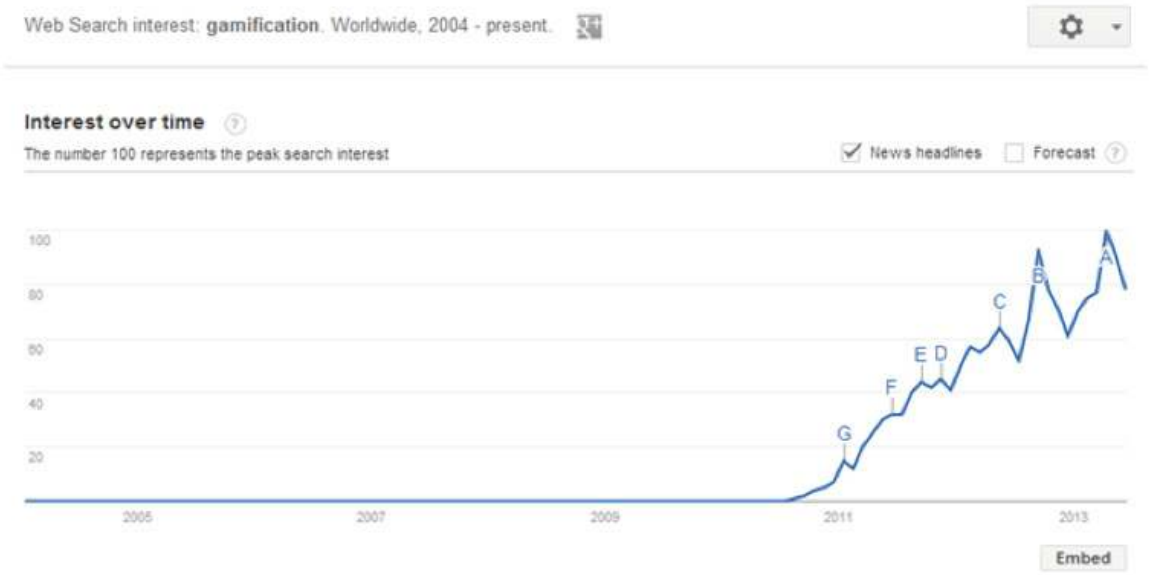


Figura 3.35: Búsqueda del término “gamificación“ en titulares de google desde 2004.

También la gamificación (*gamification*) se consolida como un movimiento de reciente creación pero que está teniendo un crecimiento exponencial en los últimos años. Se considera gamificación a la aplicación de mecánicas de videojuego, motivaciones, recompensas , gestión de la progresión y consecución de objetivos a organizaciones tradicionales como empresas u otro tipo de servicios y productos para el gran público. El uso de estas técnicas hace que el usuario se involucre de una forma más activa y le resulte más gratificante la tarea que realiza. Se usa ampliamente en los procesos de aprendizaje o en las organizaciones para premiar la consecución de objetivos en lugar de castigar la no consecución de los mismos.

El interés en la gamificación se puede medir de muchas formas, una de las más habituales es usar las tendencias de búsqueda de *google* o el número de resultados. En el gráfico de la figura 3.35 podemos ver un crecimiento realmente grande en los últimos años.

3.4.2. Perspectiva histórica del desarrollo de videojuegos en España

La denominada Edad de oro del videojuego Español [51] fue una época que duró aproximadamente entre los años 1983 y 1993, durante la cual España llegó a ser el segundo productor europeo de software de entretenimiento para máquinas de 8 bits, sobre todo para Spectrum, detrás sólo del Reino Unido.

Hablamos de un tiempo en el que marcas de origen patrio como *Dinamic*, *Opera Soft*, *Zigurat*, *Zafiro* o *Topo Soft* encabezaban las las ventas de un mercado que estaba naciendo.

En este mapa de productoras nacionales con una progresiva incorporación de profesionales extranjeros convivía un numeroso grupo de desarrolladores autónomos con el talento necesario para que sus creaciones despuntaran entre las listas de los juegos más vendidos.

En la época dorada del videojuego en España, los superheroes que inspiraban a los desarrolladores fueron tan pintorescos como *Don Quijote*, *Mortadelo*, *Curro Jiménez*, *Don Juan* o el *Capitán Sevilla*. En los juegos deportivos contaban con el apoyo de las figuras del momento como *Butragueño*, *Sanchez Vicario*, *Sito Pons*, *Perico Delgado* o *Fernando Martín*.

Algunos títulos quedarían marcados para siempre en la historia de los videojuegos como *«Olé toro»*, *«Desperado»*, *«Fernando Martín»* o *«Abu Simbel Profanation»*. Sin olvidar el videojuego desarrollado por *Paco Menéndez llamado «La abadía del crimen»* al que muchos catalogan como el mejor videojuego concebido en España.

Con la llegada de los años noventa [50] y la llegada de sistemas más potentes que permitían desarrollar videojuegos más complejos, la industria en España no supo adaptarse a las estructuras empresariales que se requerían para el desarrollo de este tipo de videojuegos más complejos y eso supuso el declive absoluto de la industria en España.

Tendrían que pasar casi 20 años para que se percibiera un cambio en esta tendencia, y ocurrieran hechos como que en 2009 el congreso de los diputados declarara a los videojuegos como Bien de Interés Cultural.

Por otro lado en los últimos años se han venido desarrollando una serie de juegos en España que de nuevo vuelven a encabezar las listas de los más vendidos a nivel mundial. Este movimiento nos sitúa de nuevo en el mapa mundial de la industria del desarrollo de videojuegos y está cimentando una industria más estable que antaño. Podríamos decir que aunque acaba de nacer recientemente ya se percibe como una de las industrias productivas más fructíferas de nuestro territorio.

Algunos de estos referentes actuales serían *Mercury Steam*, *Tequila Works* , *Digital Legends*, *Novarama*, *Digital Toys*, *Enjoy Up*, *Over the Top Games*, *Abylight*, o *U-Play* entre muchos de los que desarrollan productos de calidad.

3.4.3. Perspectiva actual del desarrollo de videojuegos en España

Enlazando con el punto anterior sobre el estado actual del desarrollo de videojuegos en España [52] y según los datos de libro blanco del desarrollo Español de videojuegos redactado por la asociación *DEV* y con el apoyo del *ICEX* podemos concluir que la salud de la industria del videojuego en España pasa por uno de los mejores momentos de su historia.

El sector de los videojuegos en España está compuesto por 330 empresas con un reparto territorial como se puede apreciar en la figura 3.4.

Los empleados que trabajan en estas empresas y su antigüedad como trabajadores, se refleja el gráfico inferior y nos permite apoyar la idea de que se trata de un sector muy nuevo, pero también con mucho crecimiento por delante y que ahora es un buen momento de apostar por él.

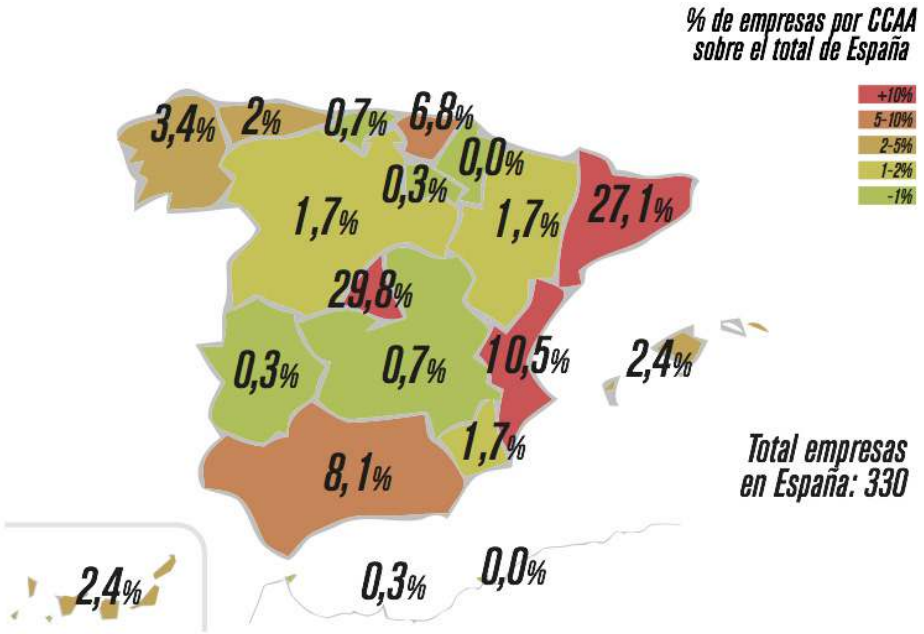


Figura 3.36: Distribución territorial de empresas en España [27]

Si hablamos del capital invertido, según el informe se refleja que el 97 % del capital que se invierte es nacional, frente a un 3 % extranjero. Por lo que los beneficios de ese capital también repercuten en la riqueza dentro de nuestro territorio.

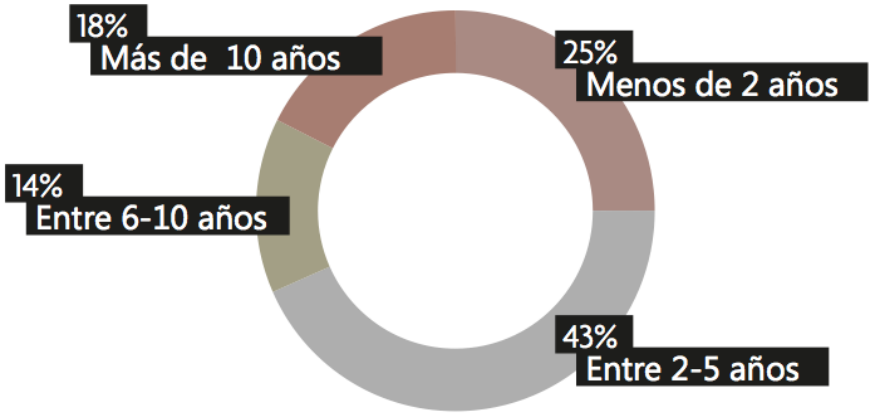


Figura 3.37: Distribución de antigüedad de los empleados [27]

En cuanto a la facturación se alcanza el 78 % con ventas en formato digital mientras que el 22 % proviene de ventas en soporte físico, por lo que queda claro que España ha encontrado en este formato una alternativa de futuro y la mayoría de sus empresas adoptan modelos

de negocio que encajan con este canal de distribución. Este hecho viene motivado porque la mayoría de los consumidores de los productos de videojuego Españoles lo hacen fuera de nuestras fronteras, quedando patente la importancia de la figura de los localizadores para adaptar los productos realizados a un mercado internacional donde el Inglés es el lenguaje dominante. A continuación se muestra la distribución de la facturación por modelos de negocio y la evolución prevista para la facturación en los siguientes años.

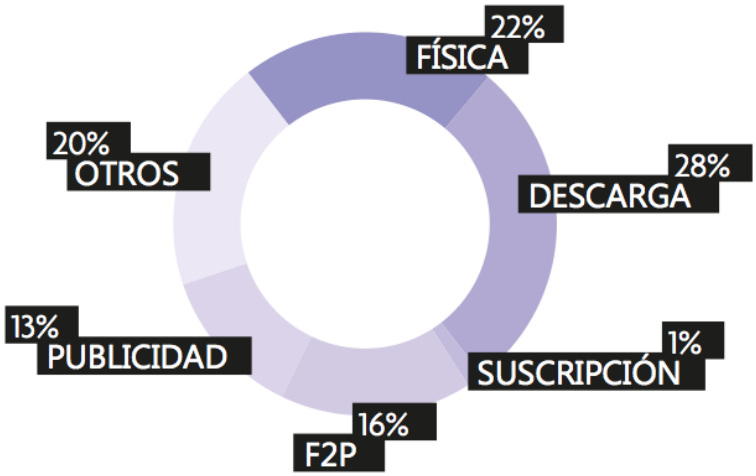


Figura 3.38: Facturación segmentada por modelo de negocio [27]

Figura 3.39: Crecimiento esperado de empleo en el sector [27]

Se estima que la facturación del sector crecerá hasta el año 2017 a una tasa anual compuesta(*CAGR 2013-2017*) del 23,7 %. Esto supone que el sector alcanzará en dicho ejercicio un volumen de negocio superior a los 723 millones de euros. Este importante incremento

se sitúa en un contexto de crecimiento sostenido del mercado internacional y en un entorno macroeconómico de recuperación tras la crisis padecida en los últimos años.

Si analizamos los tipos de perfiles profesionales y la retribución media de los mismos podemos segmentar los sueldos anuales de estos perfiles en los siguientes grupos:

- **Dirección de proyectos :** ~40.000 €
- **Desarrollo software y programación :** ~34.000 €
- **Diseño de videojuegos :** ~27.500 €
- **Marketing y distribución :** ~27.000 €
- **Creación de gráficos, animación y modelado :** ~25.500 €

En relación al empleo que el sector estima generar, la distribución por tipo de perfil se detalla en la siguiente gráfica, siendo el perfil de programador el que mayor demanda va a tener en los próximos años:

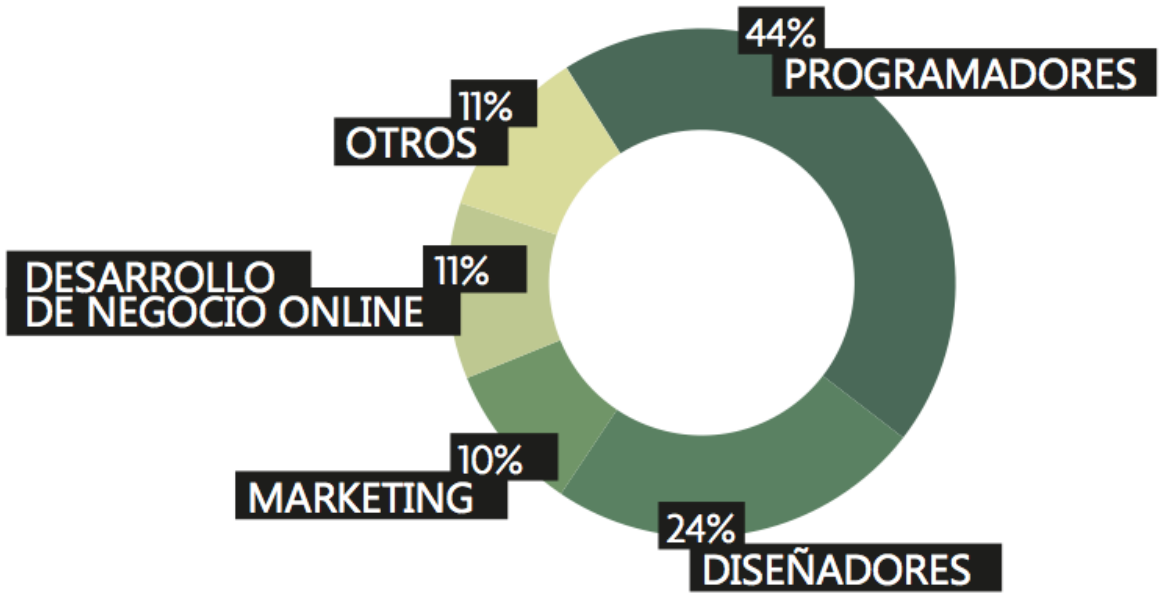


Figura 3.40: Estimación de contratación segmentado por perfiles profesionales [27]

Otro de los indicativos del estado de la industria del videojuego en España es la existencia de apoyo público contando con los siguientes tipos de ayudas a las que adherirse :

- Ayuda del ministerio de cultura de acción y promoción cultural
- Ayuda del ministerio de cultura a la inversión de capital
- Ayudas de internacionalización del *ICEX*
- Plan Avanza del ministerio de industria
- Plan de impulso de la economía digital y los contenidos digitales

También existen iniciativas privadas de apoyo a la industria del videojuego como:

- ***Gamelab* :** Organizado desde 2004, *Gamelab* busca favorecer la visibilidad de la industria Española ofreciendo una feria con doble vertiente. Por una parte es un punto de encuentro de profesionales del sector y por otra parte una feria de presentación de las novedades que están por llegar de mano de las empresas Españolas.
- ***3D Wire* :** Evento centrado en la dinamización del ocio de la mano de la industria digital especializada en la creación de videojuegos, animaciones y diseño por ordenador que se celebra en Segovia.

3.4.4. Videojuegos desarrollados en España

A continuación se nombran algunos de los videojuegos actuales que se han producido en España o que se están produciendo en estos momentos.

Videojuegos AAA: Consideramos *triple A (AAA)* a los videojuegos que cuentan con una inversión económica que les permite desarrollar un producto con unos valores de producción muy elevados. Se considera este tipo de producciones a juegos con presupuestos de entre 30 y 50 millones de euros. Aunque actualmente existen excepciones como *«GTA V»* (~250 millones) y *«Destiny»* (~400 millones). Y también hay excepciones de videojuegos con un presupuesto algo menor, pero que aún así consiguen obtener un producto de una calidad competitiva.

Como ejemplo de algunos de estos proyectos Españoles tenemos a los siguientes:

- ***«Castlevania: Lords of Shadows 2»* [84] :** Este juego es la secuela del videojuego *«Castlevania : Lords of Shadows»* de 2010. Basado en el clásico *«Castlevania»* de *Konami* donde vivimos las aventuras de la familia *Belmont* en su lucha contra *Dracula*. Realizado por *Mercury Steam* se perfila como uno de los puntales de la industria Española.

- ***«DeadLight»* [85] :** Videojuego de tipo plataformas cinemático con una ambientación única y un acabado realmente pulido hacen que a día de hoy este sea otro de los mejores representantes de videojuego en España. Realizado por *Tequila Works*.

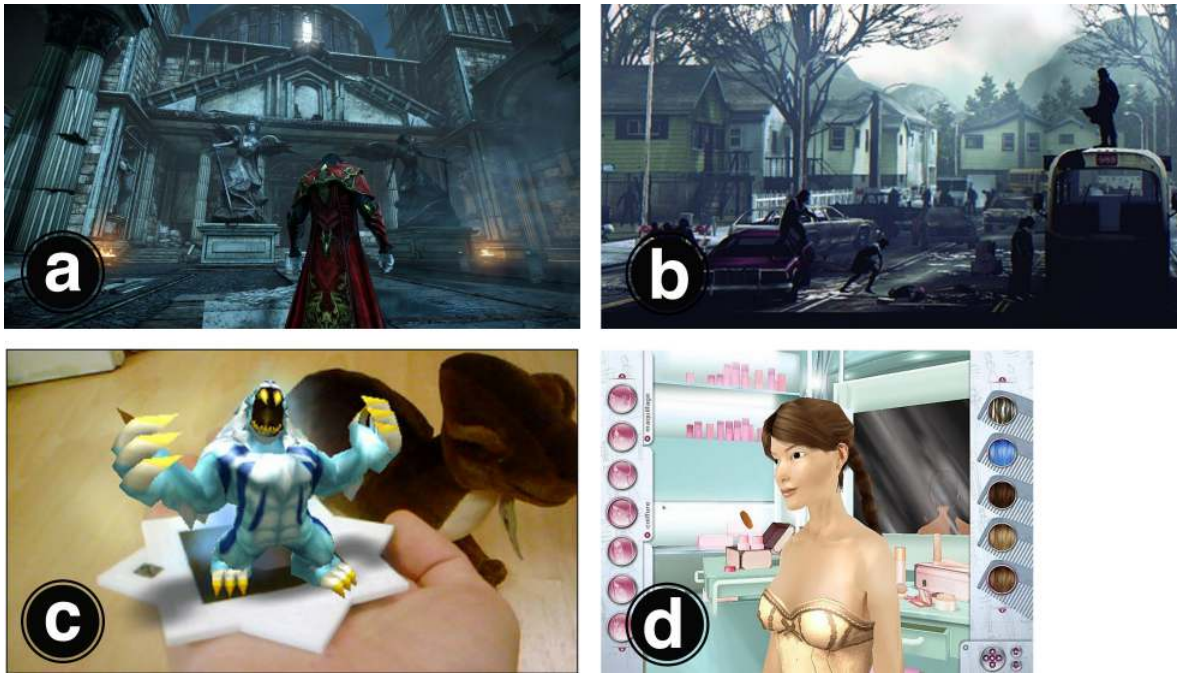


Figura 3.41: Algunos dejemplos de Videojuegos *AAA* desarrollados por empresas Españolas : a.*«Castlevania 2 : Lords Of Shadows»* [129] , b.*«Deadlight»* [130] , c.*«Invizimals»* [132] , d.*«Imagina a ser diseñadora de moda»* [131]

- ***«Invizimals»* [86] :** *«Invizimals»* es un videojuego para *PSP* creado por la compañía *Novarama* y comercializado por *Sony* que utiliza la innovación del campo de la realidad aumentada para ofrecer una propuesta nunca vista en el campo de los videojuegos.

- ***«Imagina a ser diseñadora de moda»* [87] :** Este juego es un ejemplo del buen acierto a la hora de hacer un estudio del *target* objetivo. El resultado es un videojuego para

un sector de los usuarios sin representantes de calidad y que le ha llevado a establecerse como uno de los videojuegos más vendidos de todos los tiempos en España. Fue realizado por *Virtual Toys*.

- ***«Zack Zero»* [88] :** Este videojuego es un juego de plataformas y aventuras basado en la exitosa serie de televisión del mismo nombre, creado en 2012 por *Crocodile Entertaiment* y destacando por la calidad final del producto, así como su acogida por público y crítica.

Videojuegos Independientes : Consideramos videojuegos independientes a videojuegos que se han realizado sin el apoyo de una editora que apoye la inversión. Este tipo de videojuegos se caracterizan por contar con menor presupuesto, oscilando el presupuesto típico de este tipo de videojuegos entre los 50.000€ y 200.000€. También hay casos de videojuegos más pequeños y más grandes pero queda patente la diferencia de presupuesto con respecto a los *triple A*. No obstante este tipo de videojuegos tienen algunas ventajas, como por ejemplo la libertad creativa al no depender de una editora con valores tradicionales. Otra de las ventajas es que no tienen que vender tantas unidades para ser rentables, considerándose inversiones menos arriesgadas y pudiendo crear productos con una rentabilidad adecuada a la inversión inicial.

Figura 3.42: Algunos ejemplos de Videojuegos independientes desarrollados en España : a.*«Zack Zero»* [133] , b.*«Nihilumbra»* [137] , c.*«Paradise Lost: First Contact»* [138] , d.*«Candle»* [134]

Como ejemplo de algunos de estos proyectos Españoles tenemos a los siguientes:

- ***«Nihilumbra»* [79] :** Videojuego de *puzzles* mezclados con plataformas realizado por *BeautiFun Games* y que se lanzó en *iOS* originalmente cosechando muy buenos resultados entre los usuarios.

- ***«Paradise Lost: First Contact»* [80] :** En este videojuego controlamos a una planta alienígena que ha caído a la tierra y que tiene que escapar de una instalación donde la tienen bajo estudio. Este proyecto está siendo desarrollado por la empresa *Rastree Works* y ha sido financiado a través de la plataforma de *crowfunding KickStarter*.

- ***«Candle»* [81] :** Videojuego de *puzzles* que destaca porque todos sus gráficos han sido pintados a mano utilizando técnicas tradicionales. Este proyecto está siendo desarrollado por la empresa *Teku Studio* y ha sido financiado a través de la plataforma de *crowfunding KickStarter*.

- ***«Gods Will be watching»* [82] :** Este proyecto destaca porque la temática central es la toma de decisiones morales sobre un grupo de personajes que debe sobrevivir en un ambiente hostil. Este proyecto está siendo desarrollado por la empresa *Deconstructeam* y ha sido financiado a través de la plataforma de *crowfunding KickStarter*.

Figura 3.43: Algunos ejemplos de Videojuegos independientes desarrollados en España : a.*«Gods Will be watching»* [136] , b.*«Dead Synchronicity»* [135]

- ***«Dead Synchronicity»* [83] :** Videojuego que combina temáticas *sci-fi* con ambientes distópicos, bajo un sistema de juego de aventuras *point and click*. Este proyecto está siendo desarrollado por la empresa *Fictiorama Studios* y ha sido financiado a través de la plataforma de *crowfunding KickStarter*.

3.4.5. Perspectiva internacional del desarrollo de videojuegos

Si analizamos la perspectiva global de la industria podemos decir que los ingresos estimados a nivel mundial del mercado del videojuego para el año 2013 son de 70.400 millones de dólares, un 6.2 % más que en 2012. En 2016 se estima que el mercado de videojuegos alcance los 86.100 millones de dólares, impulsado por el fuerte crecimiento en países emergentes gracias al aumento de la conectividad a *Internet* y al incremento de la penetración de los dispositivos móviles (*smartphones* y *tablets*).

Se estima que el número de jugadores activos a nivel mundial crezca hasta los 1.550 millones en 2016, con un *CAGR 49 (2012-2016)* del 5,9 %. Si analizamos la cuota de mercado por regiones y tipo de plataforma tenemos los gráficos de las Figuras 3.44 y 3.45.

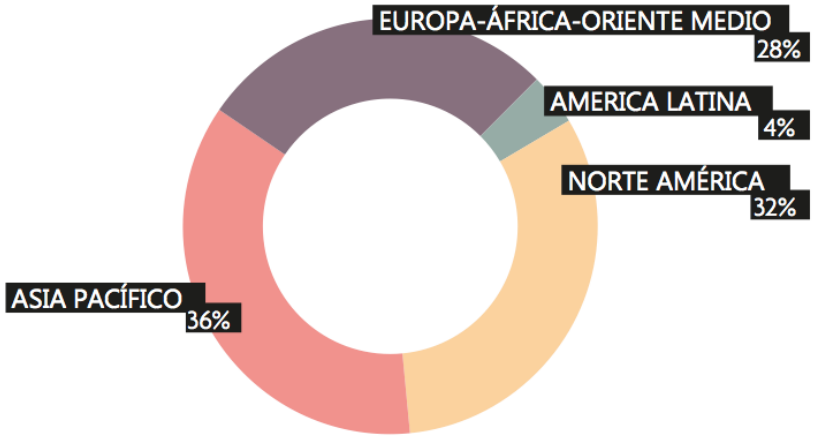


Figura 3.44: Cuota de mercado por regiones [27]

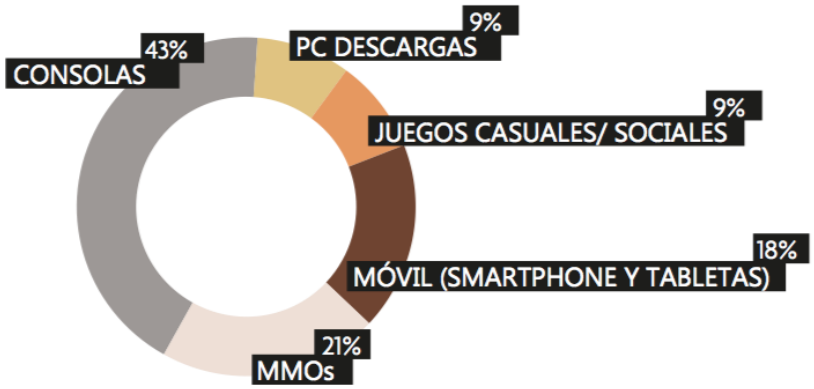


Figura 3.45: Cuota de mercado por tipo de plataforma [27]

3.4.6. Videojuegos desarrollados internacionalmente

A continuación se nombran algunos de los videojuegos actuales que se han producido en el mundo o que se están produciendo en estos momentos.

Videojuegos *AAA* : Como ejemplo de algunos de estos proyectos tenemos a los siguientes:

- ***«Uncharted 3»* [74] :** Aventura de acción en tercera persona desarrollada por *Naughty Dog*. Destaca el tratamiento cinemático de la acción y el desarrollo de los personajes.
- ***«Starcraft 2»* [75] :** Videojuego de estrategia en tiempo real que se ha consolidado como uno de los más jugados en países como Korea del Sur. En dichos países hay ligas nacionales de deportes electrónicos que superan en cuota de pantalla a deportes como el fútbol o baloncesto.

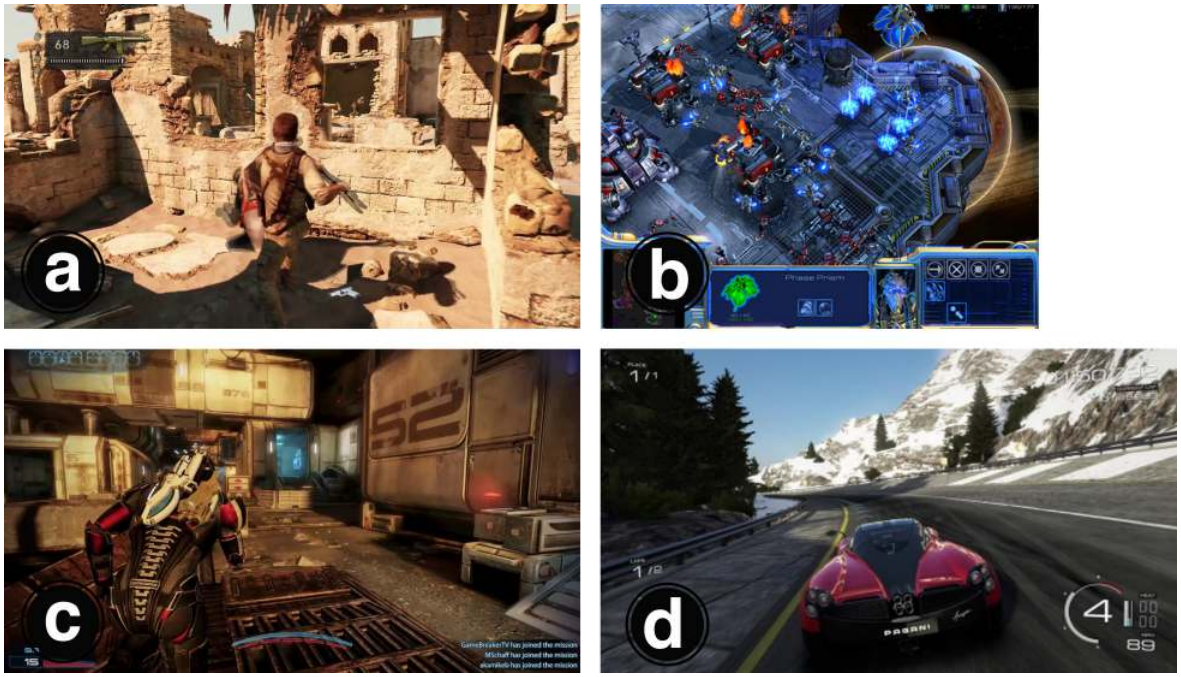


Figura 3.46: Algunos ejemplos de Videojuegos *AAA* desarrollados por empresas extrangeras : a.*«Uncharted 3»* [123] , b.*«Starcraft 2»* [122] , c.*«Mass Effect 3»* [121] , d.*«Forza Motorsport 5»* [120]

- ***«Mass Effect 3»* [76] :** Aventura de acción en tercera persona desarrollada por *Biowa-re*. Se caracteriza por desplegar un ambiente de ciencia ficción y permitir la toma de

decisiones sobre eventos que afectan al desarrollo de la trama.

- ***«Forza Motorsport 5»* [77] :** Simulador de conducción desarrollado por *Turn 10 Studios* y publicado por *Microsoft* para su nueva consola *Xbox One*. Constituye otra entrega más de una saga consagrada que vende millones de copias con cada entrega.
- ***«Fifa 14»* [78] :** Simulador deportivo desarrollado por *EA Canada* y publicado por *Electronics Arts*. Es la entrega anual de este simulador y está disponible en multitud de plataformas de juego diferentes.

Nótese que entre estos videojuegos muchos se tratan de secuelas, terceras partes o hasta quintas partes de una misma propiedad intelectual. Este conservadurismo de la industria de los *triple A* y el no contar con propuestas nuevas y creativas es uno de los motivos por el cual los videojuegos independientes están emergiendo con más fuerza entre los jugadores, pues se trata en muchos de los casos de propuestas frescas y originales que aunque con menos presupuesto consiguen ilusionar a los jugadores tanto o más que las inversiones millonarias detrás de los *triple A*.

Videojuegos Independientes : Como ejemplo de algunos de estos videojuegos independientes tenemos a los siguientes:

- ***«Limbo»* [65] :** Videojuego de tipo *plataformas-puzzle* desarrollado por la compañía independiente *PlayDead Studios* y que destaca por el uso monocromático del color con el que consigue generar un ambiente opresivo hacia el jugador.
- ***«Fez»* [66] :** Videojuego de *plataformas-puzzle* desarrollado por el desarrollador independiente *Phil Fish* y que hace uso de la perspectiva para el planteamiento de puzzles que plantean un desafío al jugador.

Figura 3.47: Algunos ejemplos de Videojuegos independientes desarrollados fuera de España : a.*«Braid»* [124] , b.*«Minecraft»* [127]

- ***«Super Meat Boy»* [67] :** Desarrollado por *Edmund MCMillen* y *Tommy Refenes*. Es el *remake* de un título realizado en *Flash* originalmente y que cosechó mucho éxito. Tanto es así que este *remake* ha llegado a multitud de plataformas considerándose como uno de los primeros videojuegos que inició la ola de atención mediática y de público sobre el sector de los videojuegos independientes.

- ***«Braid»* [68] :** Videojuego desarrollado por *Jonathan Blow* para el servicio *Xbox Live Arcade* que se caracteriza por el estilo de gráficos pintados a mano con el uso de mecánicas de rebobinado del tiempo para establecerse como otro de los fenómenos indies más conocidos de los últimos años.

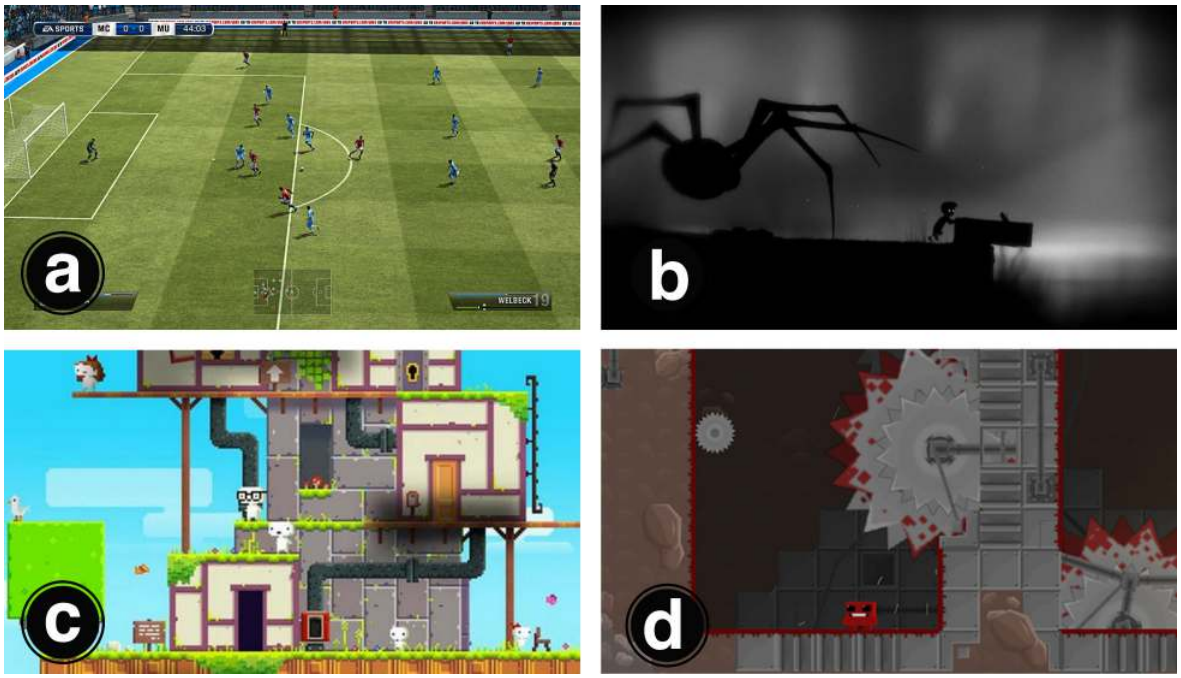


Figura 3.48: Algunos ejemplos de Videojuegos desarrollados fuera de España : a.*«Fifa 14»* [118] , b.*«Limbo»* [126] , c.*«Fez»* [125] , d.*«Super Meat Boy»* [128]

- ***«Minecraft»* [69] :** *«Minecraft»* es un videojuego creado por *Markus Notch Persson* que nos introduce dentro de un mundo persistente *online* donde podemos crear todo tipo de estructuras a partir de cubos y realizar tareas de forma colaborativa con otros jugadores.

3.4.7. Dispositivos comerciales

Actualmente tenemos disponibles una gran gama de dispositivos donde se comercializan videojuegos. Algunos de estos dispositivos son multifuncionales como un *PC* doméstico o los *smarphones*, mientras que otros son dispositivos concebidos únicamente para su uso con videojuegos como las videoconsolas:

- ***PC / Mac*** **:** El *PC* está viviendo uno de los mejores momentos en lo que a plataforma de venta de videojuegos se refiere. Este hecho ha sido motivado principalmente por plataformas de venta digital como *Steam*. El modelo de distribución dominante en este dispositivo es la descarga digital y hay cabida a juegos con modelos de negocio *Pay to Play* y también *Free To Play*.

- **Micro consolas como *OUYA*** **:** Estos dispositivos han emergido recientemente y aunque no cuentan con una base de publico mayoritario en estos momentos, están empezando a ganar presencia en la industria. En el caso de *OUYA* cuenta con sistema operativo *Android*. Modelo de distribución dominante en formato digital y *Freemium*.

- ***Smarphones / Tablets iOS*** **:** Diversos dispositivos con sistema operativo *iOS*. Modelo de distribución únicamente digital y modelo de negocio *Pay to Play*, *Free to Play* y *Freemium*.

- ***Smarphones / Tablets Android*** **:** Diversos dispositivos con sistema operativo *Android*. Modelo de distribución únicamente digital y modelo de negocio *Pay to Play*, *Free to Play*, y sobre todo Publicidad.

- **Videoconsola *Wii*** **:** Videoconsola de *Nintendo* de anterior generación. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.

- **Videoconsola *Wii U*** **:** Videoconsola de nueva generación de *Nintendo*. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.

- **Videoconsola *Nintendo 3DS*** **:** Videoconsola portátil de *Nintendo*. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.

- **VideoConsola *Xbox 360*** **:** Videoconsola de *Microsoft* de anterior generación. Todavía vigente a día de hoy y con gran base de usuarios potenciales. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.

- **VideoConsola *Xbox One* :** Videoconsola de nueva generación de *Microsoft*. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.
- **VideoConsola *PS3* :** Videoconsola de *Sony* de anterior generación. Todavía vigente a día de hoy y con gran base de usuarios potenciales. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.
- **VideoConsola *PS4* :** Videoconsola de nueva generación de *Sony*. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.
- **Videoconsola *Sony PSP Vita***: Videoconsola portátil de *Sony*. Modelo de distribución dominante en formato físico y modelo de negocio dominante *Pay to Play*.

3.4.8. Plataformas de distribución digital

El auge de los plataformas de distribución digital a día de hoy es un hecho. En algunos sistemas como las videoconsolas conviven con la distribución física, pero también hay dispositivos en los que el único modelo de distribución de contenido es el digital. A continuación se analizan algunas de las más importantes plataformas de distribución digital de la actualidad:

- ***Xbox Live* :** Tienda de distribución digital de *Microsoft* para videoconsolas *Xbox*.

  La entrada a esta plataforma es cerrada. Sólo se puede publicar si *Microsoft* te da autorización expresa o se realiza este proceso a través de un *publisher* autorizado.

- ***Playstation Network* :** Tienda de distribución digital de *Sony* para videoconsolas *Playstation*.

  La entrada a esta plataforma es cerrada. Sólo se puede publicar si *Sony* te da autorización expresa o se realiza este proceso a través de un *publisher* autorizado.

- ***eShop* :** Tienda de distribución digital de *Nintendo*.

  La entrada a esta plataforma es cerrada. Sólo se puede publicar si *Nintendo* te da autorización expresa o se realiza este proceso a través de un *publisher* autorizado.

- ***Steam* :** Tienda de distribución digital de *Valve*. Es la plataforma dominante en el mundo del *PC*.

  La entrada a la plataforma es cerrada, para poder publicar en *Steam* puede realizarse de dos formas :

- 1. A través de un *publisher* autorizado.
- 2. A través de *Steam Greenlight*. Este sistema requiere subir imágenes y vídeos del juego a un sistema de votación de la comunidad. Si el juego recibe suficientes votos y despierta interés entre los usuarios de la plataforma entonces te permiten publicarlo.

- ***Desura*** **:** Tienda de distribución digital. Es una plataforma muy notable en el mundo *indie* del *PC*. A menudo se utiliza para publicar videojuegos independientes como escalón previo a *Steam*. De esta forma se puede ganar el suficiente prestigio como para que estos juegos sean aceptados en la otra plataforma.

  La entrada a esta plataforma requiere que el software pase un proceso de certificación y cualquier desarrollador independiente puede enviar su software. No obstante es necesario que el producto cumpla unos determinados estándares de calidad para ser aprobado.

- ***App Store*** **:** Tienda de distribución digital de *Apple* para dispositivos con sistema operativo *iOS*.

  La entrada a esta plataforma es abierta. Cualquier desarrollador que pague la licencia puede publicar.

- ***Google Play*** **:** Tienda de distribución digital de *Google* para dispositivos con sistema operativo *Android*.

  La entrada a esta plataforma es abierta. Cualquier desarrollador que pague la licencia puede publicar.

Existen más plataformas de distribución digital con menor base de usuarios, no obstante lo más importante para la rentabilidad del proyecto es distribuir el videojuego en un canal que permita obtener buena visibilidad.

3.4.9. *Bundles*

Entre el mundo de los desarrolladores independientes surgió otro fenómeno denominado *«Humble Bundle»*. En origen se trataba de dar a conocer determinados videojuegos independientes y repartir parte del dinero recaudado en obras de caridad.

Para ello el usuario compra un paquete de videojuegos y puede pagar lo que el quiera de entre unos varemos de precio. También es posible decidir qué parte de ese dinero va a obras

de caridad y qué parte a los desarrolladores del videojuego.

Este fenómeno caló hondo entre los jugadores y surgieron multitud de alternativas similares. Algunas de las plataformas más conocidas son las siguientes:

- Humble Bundle
- Indie Royale
- Bundle Stars
- Indie Gala
- Groupees
- Little Big Bunch
- The Greenlight Bundle
- Indie Game Stand

Si bien este tipo de propuestas no sirven para obtener un gran volumen de negocio, si que proporcionan mucha visibilidad, y han ayudado a muchos desarrolladores a ser conocidos y poder recibir los votos suficientes en *Steam Greenlight*.

La entrada a este tipo de propuestas es cerrada. Para ello cada plataforma organizadora del *bundle* selecciona una serie de juegos con potencial y se pone en contacto con el desarrollador para llegar a un acuerdo.

3.4.10. Comportamiento de las ventas y demografía

Cuando se lanza un nuevo videojuego al mercado suele tener un periodo de mayor visibilidad que es cuando más se vende. El funcionamiento de las ventas se rige por el fenómeno *The long tail*. Puede apreciarse el comportamiento en ventas en la Figura 3.49.

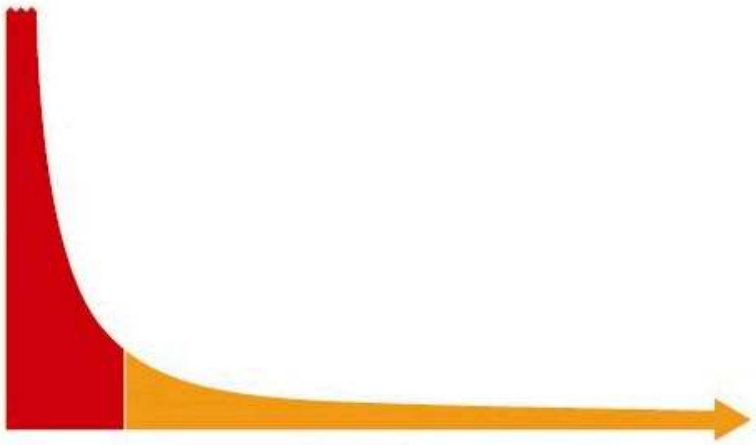


Figura 3.49: Distribución *Long Tail* [94]

Si se sigue trabajando la comunicación para que el videojuego sea popular seguirá recibiendo ventas y con ello el área de larga cola seguirá creciendo. Con el suficiente tiempo este área amarilla puede ser más fructífera que el área roja de ventas iniciales.

Otra posible estrategia es enfocarse en disponer un número alto de productos con bajo volumen de ventas, pues la suma de las colas de estas ventas puede ser mayor que el área señalada en rojo que obtendría un *hit* de ventas en su primer periodo.

En cuanto a la demografía, según los datos recolectados durante 4 años y más de 500.000 descargas por parte de la empresa *«Atomic Flavor SL»* sobre el *App Store* se pueden obtener lo siguientes gráficos:

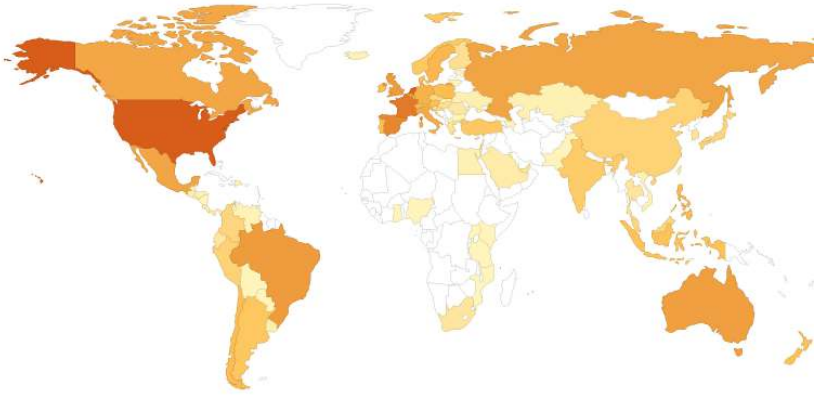


Figura 3.50: Gráfico de rentabilidad por territorios

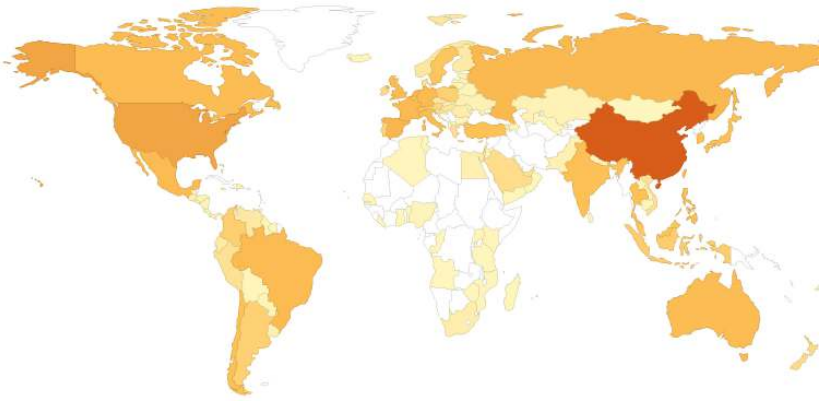


Figura 3.51: Gráfico de descargas por territorios

En la figura 3.11 se visualiza la rentabilidad por territorios, mientras que en la figura 3.12 se visualiza el número de descargas por territorio. Las áreas más oscuras significan un volumen mayor.

Este tipo de datos nos permite inferir los modelos de negocio que funcionan mejor en cada territorio y conocer cómo funciona el mercado en dicha plataforma.

La conclusión tras analizar los datos propios y compartir información con otros equipos de desarrollo es bastante clara :

- ***USA* :** El mercado de videojuegos por excelencia. De aquí se saca la mayor rentabilidad. Los usuarios están dispuestos a pagar por el software y el modelo *Pay to Play* funciona bien.
- **China :** El mercado con más volumen de descargas gratuitas. No pagan por el software, pero es un tipo de público con mucha base de usuarios. Si se genera una comunidad adecuada en torno a videojuegos *Free To Play* es posible obtener bastante beneficio de la base de usuarios que sí gasta dinero (2 % - 5 %).
- **Resto de países :** Podríamos contemplar otros territorios como Canadá, Reino Unido, Australia o Alemania. En estos países también es posible obtener cierta rentabilidad, no obstante suele ser menor comparado con *USA*.
- **España :** El caso concreto de España no es diferente a otros países como Francia, o Italia. El lanzar un producto en estos territorios genera menos beneficios. Es por eso que dichas versiones suelen ser traducciones o conversiones de las versiones previas.

CAPÍTULO 4. MÉTODO DE TRABAJO

En este capítulo se explica la metodología de trabajo elegida, así como que herramientas de software y *hardware* han sido empleadas para la elaboración del videojuego. También se detallan los *assets* externos que han sido empleados.

4.1. METODOLOGÍA DE TRABAJO

Cuando se enfrenta el desarrollo de un videojuego con equipos de desarrollo en los que trabajan personas con diferentes perfiles profesionales, la metodología de trabajo más adecuada e utilizada en la industria es *Scrum*. No obstante, para el presente TFG, se plantea el uso de la metodología de prototipado evolutivo por los siguientes motivos:

- Se trata de una única persona, que realizará el proceso de diseño del videojuego, así como su posterior diseño software, implementación y construcción del producto final. No sería adecuado aplicar una metodología de trabajo ágil pensada para la interacción en equipos más numerosos y dinámicos.

- En cuanto a la naturaleza del proyecto, una metodología de desarrollo con prototipado evolutivo permite ir modificando el videojuego de una forma dinámica y adaptándolo a las necesidades. También permite anticiparse a problemas al ir teniendo un producto que poder probar con cada uno de los prototipos.

Esta metodología de prototipado evolutivo es arriesgada, pero permite hacer una aproximación rápida al desarrollo de un prototipo funcional. De esta forma se puede ir recogiendo *feedback* que será utilizado para mejorar el videojuego en la siguiente interacción.

El primer prototipo está basado en las mecánicas jugables propuestas en el documento de diseño del videojuego. Posteriormente se han realizado modificaciones en cada uno de los aspectos para obtener la calidad final en cada uno de los apartados.

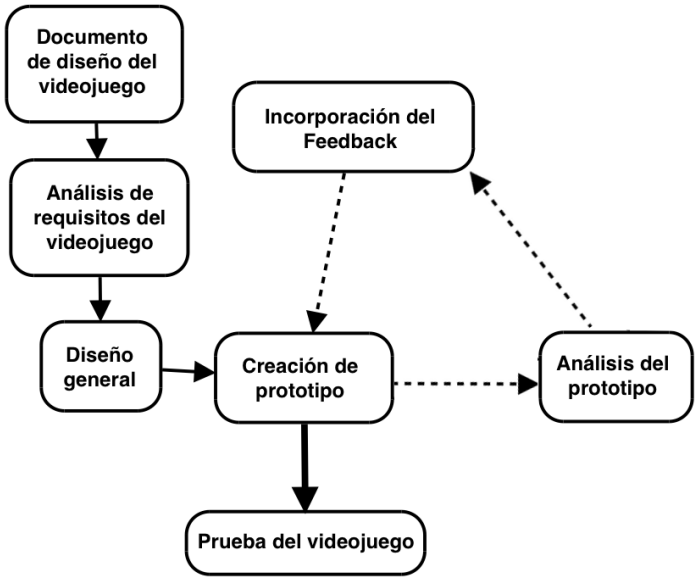


Figura 4.1: Flujo de trabajo con la metodología de prototipado evolutivo

Tal como puede apreciarse en la figura 4.1, este proceso evolutivo ha permitido construir el videojuego, mientras se tienen versiones jugables. Sobre estas versiones se han podido sacar conclusiones y anticipar decisiones para guiar el desarrollo y evitar problemas futuros.

En el capítulo 5 se detallan cada uno de los prototipos obtenidos.

4.2. HERRAMIENTAS

El desarrollo del videojuego se ha realizado utilizando el motor *«Unity3D»* [16] en sus versiones 3.5 y 4.3. Además se han empleado herramientas de síntesis de imagen digital y edición de audio para crear y editar los *assets* del videojuego.

4.2.1. Lenguajes

El motor *«Unity3D»* [60] permite implementar la lógica del videojuego mediante el paradigma de programación orientado a componentes [63]. Se considera que el nivel de abstracción de los componentes es más alto que el de los objetos. Dichos componentes no comparten estado, ya que realizan su comunicación mediante el intercambio de mensajes que contienen datos.

«Unity3D» organiza el contenido [14] mediante escenas. Una escena es una jerarquía en

árbol de objetos de juego (clase *GameObjects*). Una escena puede representar el nivel aislado de un videojuego o también el sistema de menús que permite iniciar el juego en otros niveles.

La interrelación entre los diferentes componentes y las asociaciones entre elementos que conforman la arquitectura del videojuego fueron realizadas a nivel de *prefabs*. *Un prefab* es la serialización de una parte del árbol de la escena a la que se le asocia un nombre y que permite su posterior instanciación en diferentes escenas en tiempo de ejecución.

Cada elemento individual dentro de la jerarquía de un *prefab* puede contener uno o varios componentes. Los componentes pueden ser modelos *3D*, objetos físicos, punteros a sonidos, texturas, materiales, o *scripts* de comportamiento.

En nuestro caso prestaremos especial interés a los *scripts* de comportamiento que pueden ser especificados utilizando los siguientes lenguajes:

- ***Javascript* :** *«Unity3D»* proporciona una *API* que permite acceder a sus funcionalidades empleando los lenguajes *Javascript*, *C#* o *Boo*. El lenguaje utilizado principalmente en la implementación de la lógica del videojuego ha sido *Javascript*.
- ***C#* :** Para extender y modificar el comportamiento de algunos de los componentes incluidos de serie en *«Unity3D»* se ha utilizado el lenguaje en el que están originalmente implementados. Estos componentes han tenido que ser modificados para adaptarlos a las necesidades concretas del desarrollo, como por ejemplo la realización de optimizaciones que garanticen el correcto funcionamiento en dispositivos móviles de funciones proporcionadas por *«Unity3D»* originalmente para ordenadores de escritorio.
- ***«LaTeX»* :** Lenguaje empleado para realizar la maquetación de la documentación en su distribución MacTeX-2014.

También hay que prestar especial atención a los materiales, pues todo material tiene asociado a un *shader* que define su las propiedades de este durante la renderización en el *pipeline* gráfico. Los *shaders* han sido especificados utilizando los siguientes lenguajes:

- ***Shaderlab* :** Es el lenguaje estándar de *«Unity3D»* para la descripción de *shaders*. Después este código es traducido automáticamente por un compilador del motor a *GLSL* o *HLSL* en el momento de compilación.

- ***GLSL* :** También es posible describir los *shaders* utilizando código *GLSL*, en este caso se empleó directamente GLSL en determinados *shaders* en los que se quería especificar con una granularidad más fina, consiguiendo mayor grado de optimización.

En el caso concreto de este proyecto se han empleado unas 37.000 lineas de código para definir los *scripts* de comportamiento y unas 3500 lineas de código para definir los *shaders*.

El proyecto se ha estructurado en 92 escenas, entre las que se definen los diferentes niveles, menús, escenas de carga, escenas auxiliares de vídeo, escenas de visualización de cómics, etc.

4.2.2. *Hardware*

Para la implementación y pruebas del videojuego se han empleado los siguientes dispositivos:

- Ordenador portátil MacBook Pro i7 2.2GHz, 8GB de *RAM* y tarjeta gráfica AMD Radeon HD 6750M 512 MB con sistema operativo Mac Os X 10.7.
- Monitor externo Apple Led Cinema Display 27”
- Ordenador sobremesa con procesador Quad Core 2.4GHz, 4GB de *RAM* y tarjeta gráfica Nvidia GeForce 250 Gts 512 MB con sistemas operativos *Ubuntu GNU/Linux* 14.04 y *Windows* 7
- *GameStop USB Joystick*
- Teléfono iPhone 5 con sistema operativo *iOS* 7.1.
- *Tablet* iPad 3 con sistema operativo *iOS* 7.1.
- Teléfono Nexus 4 con sistema operativo *Android* 4.2.
- Videoconsola *OUYA* [61] con sistema operativo *Android* 4.1.

4.2.3. Software

En este apartado se especifica el conjunto de software utilizado para la implementación y pruebas del videojuego.

Sistemas operativos

El desarrollo del videojuego ha sido realizado bajo el sistema operativo *Mac OS X* 10.7. y 10.9. Para ello se ha realizando una compilación cruzada a los diferentes sistemas operativos objetivo de las versiones finales del videojuego.

Para probar las diferentes versiones del juego se han utilizado también los sistemas operativos *Ubuntu GNU/Linux* 14.04., *Windows* 7, *Android* 4.1. y 4.2. e *iOS* 7.1.

Software empleado

Además del motor de videojuegos *«Unity3D»* se han empleado las siguientes herramientas :

- ***«MonoDevelop»* 4.0.1. :** Entorno *IDE* que permite la escritura de código en los lenguajes *C#*, *Javascript*, *Shaderlab* y *GLSL.*
- ***«Blender 3D»* 2.49. :** Software de modelado y síntesis de imagen digital empleado para realizar o editar el modelado de los *assets 3D* del videojuego.
- ***«Pixelmator»* 3.2. :** Software de retoque fotográfico empleado para realizar o editar los gráficos de los *assets 2D* del videojuego.
- ***«Comic life 2»* 2.2.6. :** Software empleado para añadir los bocadillos a los cómics que se muestran en el videojuego para contar la historia.
- ***«Crazy Bump»* Beta 2 :** Software empleado para la creación de mapas de normales a partir de texturas con información de color.
- ***«Audacity»* 2.0.3. :** Software empleado para realizar o editar los sonidos y música de los *assets* de audio del videojuego.
- ***«Libreoffice»* 4.1.1.2. :** Software empleado para realizar el primer borrador de el presente documento.

4.3. UTILIZACIÓN DE *ASSETS* EXTERNOS

Puesto que el presente videojuego ha sido desarrollado por una única persona sin un equipo de modeladores *3D*, grafistas *2D* o músicos se ha hecho uso de contenido externo con

licencia libre de derechos que permite utilizar estos *assets* como contenido incluido dentro de un videojuego comercial. No es posible en cualquier caso distribuir de forma libre este contenido sino como contenido incluido en la compilación final del videojuego.

Este contenido ha sido editado y modificado para utilizar junto con el contenido de *assets* desarrollados de forma original por el alumno. El objetivo de esta adaptación es que todo el contenido del videojuego guarde una coherencia en estilo y características del mismo.

Para un listado completo de los assets externos con los que se ha contado puede consultarse el anexo D de este documento.

4.3.1. Modelos *3D*

Se han empleado modelos *3D* adquiridos en los siguientes portales:

- *Turbosquid* [31] : Tienda digital de venta de modelos 3D utilizado por empresas como *«Industrial Light & Magic»*, *«Activision»*, *«Sony»* o *«Pixar»*.
- *3DModels & Textures* [32] : Tienda digital de venta de paquetes de modelos 3D y texturas organizados de forma temática.
- *3DRT* [33] : Tienda digital de venta de paquetes de modelos 3D y texturas organizados de forma temática.
- *The3dstudio* [34] : Tienda digital de venta de modelos 3D en diversos formatos.

Este contenido ha sido utilizado como apoyo para ambientar y conformar los diferentes niveles del videojuego.

4.3.2. Vectores y imágenes *2D*

Se han empleado vectores *2D* adquiridos en *Vector Stock* [35]. Este contenido ha sido utilizado como apoyo para ambientar y conformar los diferentes elementos de los menús.

4.3.3. Música / Sonidos

Se han empleado música y efectos de sonido adquiridos en los siguientes portales:

- *Opuzz* [36] : Tienda digital que vende *CD-ROM's* temáticos para usar como banda sonora en diferentes producciones multimedia.

- *Audiojungle* [37] : Tienda digital de venta de paquetes de canciones y efectos de sonido.

Este contenido ha sido utilizado para componer la banda sonora que se escucha en los diferentes niveles del videojuego.

4.3.4. Biblioteca de efectos de partículas

Se han empleado bibliotecas de efectos de partículas adquiridas del *Asset Store* [38] de *«Unity3D»*. Este contenido ha sido utilizado para representar efectos que se aprecian en el videojuego como la sangre, las texturas del fuego o el impacto de las balas sobre las diferentes superficies.

4.3.5. Traducción

También se han contado con los servicios de traducción de un profesional para garantizar que el videojuego haga uso del lenguaje Inglés de forma correcta.

CAPÍTULO 5. RESULTADO FINAL

En este capítulo se describe el producto final obtenido para el presente TFG. Para ello se mantendrá la división de etapas planteadas en la construcción del producto: Diseño, Desarrollo y Comercialización. En cada una de estas secciones se tratarán diferentes perspectivas del mismo proyecto y se abordarán los detalles relativos a cada apartado. Por último se incluye la sección de evolución y costes del proyecto donde se hace un resumen de las diferentes iteraciones del desarrollo.

5.1. DISEÑO

A continuación se describe el Documento de Diseño de Videojuego (*GDD*) de *«Rock Zombie»*. Aunque la elaboración de un *GDD* para un videojuego como el propuesto en el presente TFG podría requerir cientos de páginas de documentación, en la siguiente sección se ha sintetizado la información más importante para garantizar un correcto diseño de producto final.

5.1.1. Descripción del videojuego

Sinopsis

¿Qué pasaría si el concierto de rock de una banda femenina fuera invadido por una horda de *zombies*? La respuesta a esta pregunta es este videojuego que combina jugabilidad clásica de la era dorada de los arcades con gráficos tridimensionales de última generación.

Figura 5.1: Imagen final del videojuego

Características principales

Las características principales del videojuego son:

- Gráficos tridimensionales con uso de *shaders* y efectos postproceso para dotarlos de la calidad requerida por los jugadores actuales.
- Historia contada por medio de viñetas de cómics.
- 20 niveles y *tutorial* de iniciación.
- 4 jefes finales con diferentes mecánicas jugables.
- Dos niveles especiales de conducción (coche y moto).
- Tres personajes seleccionables con diferentes armas y trajes para cada uno de ellos.
- Ataques cuerpo a cuerpo y ataques mágicos.
- Banda sonora que combina diferentes géneros de *Rock & Roll* y *Heavy Metal*.
- Uso de logros.
- Uso de *items* coleccionables.

Género

El género del videojuego es *Beat 'em up* de scroll lateral. También conocidos como *Brawler*.

Es un género de videojuegos en el que se destaca el combate cuerpo a cuerpo entre el protagonista y un gran número de enemigos. El objetivo típico este tipo género de videojuegos es luchar contra oleadas de enemigos para ir avanzando por el nivel.

Los *beat 'em ups* son un género separados de los videojuegos de lucha tradicionales. Este género ocurre sobre un nivel largo, con el desplazamiento de la pantalla cuando el jugador se mueve por el escenario. Los juegos de lucha competitiva incluyen una mayor variedad de ataques que el jugador puede usar, mientras que los *beat 'em ups* ofrecen menos ataques con

un esquema de control más simple.

Figura 5.2: Imagen de *«Golden Axe»* [105], uno de los referentes clásicos del género *Beat 'em up*

Además los *beat 'em up* emplean escenarios de desplazamiento horizontal, algunas de ellos con un jefe final. La mayoría de los *beat 'em ups* se caracterizan por que, además de moverse de izquierda a derecha (y/o saltar/esquivar), los jugadores también pueden moverse en profundidad, dentro y fuera de la escena.

Productos similares

Actualmente se comercializan pocos videojuegos de este género, entre algunos de los referentes del momento podríamos destacar (ver Figura 5.3) :

- ***Castle Crashers*** : Estética *cartoon* y ambientación medieval. Es *gameplay* es de *Beat 'em up* clásico.
- ***Charlie Murder*** : Estética *punk* y uso de referencias musicales. Es *gameplay* es de *Beat 'em up* clásico.
- ***Sacred Citadel*** : Estética estilizada y ambientación medieval. Es *gameplay* es de *Beat 'em up* pero no es posible moverse en profundidad.
- ***Shank*** : Estética *cartoon* y ambientación moderna. Es *gameplay* es de *Shoot 'em up* mezclado con *Beat 'em up*. No es posible moverse en profundidad.

Figura 5.3: a.*Charlie Murder* [107], b.*Castle Crashers* [106], c.*Sacred Citadel* [108], d.*Shank* [109]

La mayoría de estos videojuegos tienen en común la utilización del *2D* como modo gráfico. En el presente videojuego se planea la utilización de *3D* para aportar efectos novedosos hasta el momento en el género como el uso de cámaras cinemáticas, el uso de repeticiones a cámara lenta o el cambio del punto de vista.

Estudio de público objetivo

Los resultados del estudio del público objetivo se pueden consultar en el apartado "Comercialización" de este mismo capítulo.

Estilo y ambientación

El videojuego cuenta con una ambientación de tipo película de *Zombies* donde todo es caótico, oscuro, sucio y lluvioso.

Figura 5.4: Imagen de uno de los escenarios del videojuego

Para recrear esta ambientación se ha empleado un estilo de colores y personajes estilizados (ver Figuras 5.4 y 5.5), optando por esta solución en contra del fotorealismo para aportarle un toque más distinguible a la estética del proyecto.

Figura 5.5: Imagen de uno de los personajes del videojuego

Los personajes principales son una banda de *rock* femenina con vestimenta de cuero y

estética *metal*.

Localizaciones

El videojuego transcurre en las siguientes localizaciones :

- ***Escenario de concierto*** : Nivel 0
- ***Calles de ciudad*** : Niveles 1, 2, 3, 15, 18
- ***Edificio interior, servicios*** : Nivel 4
- ***Edificio interior, cocinas*** : Niveles 5, 6
- ***Parque*** : Nivel 7
- ***Cementerio*** : Niveles 8 y 9
- ***Estación de tren*** : Nivel 10
- ***Polígono industrial exterior*** : Nivel 11
- ***Polígono industrial interior*** : Niveles 12 y 13
- ***Alcantarillas*** : Nivel 14
- ***Autopista*** : Niveles 16 y 17
- ***Edificio interior, hospital*** : Niveles 19, 20

5.1.2. Jugabilidad y mecánicas

Gameplay

En este apartado se detallan los 3 componentes que definen todo *gameplay*, el personaje, el tipo de cámaras y la forma de controlar el videojuego. En videojuegos esto es llamado "Las tres C's : *Character*, *Camera*, *Control*"

Si cualquiera de estos tres apartados fallara el juego estaría mal diseñado.

Personaje

El personaje que controlará el jugador es una chica que pertenece a un grupo de *rock* y a su vez es una bruja de un clan ancestral que usa su magia para hacer el bien y combatir las

amenazas malignas de este mundo.

La estética de los personajes es con vestimenta de cuero, colores de pelo chillones y complexión atlética (ver Figura 5.6).

Figura 5.6: Estética de los personajes

Cámara

El personaje se visualizará en tercera persona, desde una perspectiva lateral sobre la que podremos controlar al personaje (ver Figura 5.7). La cámara seguirá al jugador siempre que se encuentre dentro de los límites preestablecidos.

Figura 5.7: Tipo de cámara estándar con *scroll* lateral

También habrá zonas específicas de cámara donde se proporcionará una perspectiva más adecuada de determinados elementos de interés.

Figura 5.8: Camara oblicua para adaptarse al callejón

Por último habrá determinadas zonas donde la cámara realizará un recorrido libre cuando el jugador entre. Este tipo de rodajes de cámara no será interactivo y ayudarán a añadir el acabado cinematográfico al juego y proporcionar espectacularidad. Por otro lado este tipo de rodajes de cámara servirá de nexo de unión entre diferentes áreas de acción, dando al jugador un respiro en estos momentos no jugables.

Figura 5.9: Rodaje de cámara

Control

El control que ejerceremos sobre el juego será manejar el movimiento del personaje, pudiendo caminar, correr o realizar volteretas como movimiento evasivo para evitar ataques de enemigos.

El personaje podrá realizar ataques cuerpo a cuerpo para eliminar a los enemigos de forma cercana o ataques mágicos desde lejos si ha recargado la barra de magia previamente.

También habrá niveles en lo que el jugador conducirá un vehículo que acelera solo y lo único que deberá realizar sera desplazarlo de un lado a otro de la pantalla para eliminar enemigos y evitar obstáculos.

Estructura

Para asegurar la diversión del jugador y la correcta estructuración del contenido, para el diseño se ha seguido la regla de los tres 5's.

5 segundos Vista

A los 5 segundos de ver el videojuego el jugador debe saber el tipo de juego del que se trata, el tono del mismo y entender que tipo de objetivos o peligros puede tener. Tras ver una captura del videojuego la idea que se forme el jugador sobre como será el juego debe ser acertada para que no se produzca una diferencia entre lo esperado y recibido.

5 minutos Vista

A los 5 minutos de estar jugando, el jugador debe haber experimentado una pequeña misión del videojuego y tiene que haber encontrado interesante el universo del juego y el tipo de experiencias que se le plantean. Si esto no es así el jugador podría abandonar el juego en este punto.

5 horas Vista

A las 5 horas de juego, el jugador debe haber entendido el guión que le plantea el juego, los diferentes elementos del mismo, qué tipo de desafíos debe superar y qué conflictos debe

solucionar. Si estos elementos son atractivos se sentirá atraído por solucionarlos y concluir el videojuego, pero por contra si no paran de introducirse nuevos elementos o la estructura general es caótica el jugador puede sentirse tentando a abandonar el videojuego.

Progresión

La progresión del juego se lleva conforme a tres criterios:

- **Compleción de niveles :** El jugador irá completando niveles y la finalización de los mismos les permitirá volver a jugarlos en el futuro. También se irán desbloqueando los cómic conforme el jugador vaya alcanzando ese punto de la historia.
- **Consecución de *items* :** Conforme el jugador avance en los niveles se le irá entregando como recompensa diferentes cantidades de monedas en función de el grado de consecución.
- **Consecución de logros :** Al jugador se le plantearán diferentes retos secundarios cuya consecución le hará conseguir los diferentes logros del videojuego.

Reto y recompensas

El esquema de dificultad adoptado por el juego es una curva de dificultad creciente. Para aumentar la dificultad conforme aumentan los niveles se adoptan los siguientes métodos:

- Enemigos con variantes más complejas conforme aumentan los niveles. Uso de enemigos de tipo fuego, de tipo vómito y perros.
- Grupos de enemigos más numerosos conforme aumentan los niveles.
- Más cantidad de *items* de peligros sobre el escenario como minas, raíces, telarañas, veneno o fuego.
- Número de enemigos activos más numeroso por grupo. En los primeros niveles aunque un grupo tenga 8 enemigos sólo 2 persiguen y atacan simultáneamente al personaje. A medida que aumentan los niveles este número de enemigos activos de manera simultánea aumenta también.

Además de la dificultad creciente en cada nivel se establecerán 3 niveles de dificultad: *Casual*, *Normal* y *Hardcore* que tendrán el siguiente impacto:

- **Dificultad *casual* :** Los enemigos quitan la mitad y el personaje del jugador les quita el doble con los ataques. El modificador para calculo de puntuación se sitúa en 0.5, obteniendo menor cantidad de monedas aunque se completen los niveles en este nivel de dificultad. Se permite continuar la partida cuando el jugador pierde todas sus vidas.
- **Dificultad *normal* :** El modificador para calculo de puntuación se sitúa en 1.0, obteniendo la cantidad estándar de monedas. Se permite continuar la partida cuando el jugador pierde todas sus vidas.
- **Dificultad *hardcore* :** Igual que la dificultad normal pero cuando el jugador pierde todas las vidas la partida acaba sin posibilidad de continuar. El modificador para eñ cálculo de puntuación se sitúa en 1.5, obteniendo mayor cantidad de monedas si se juega en esta dificultad.

5.1.3. Historia

Desglose de la trama

La historia transcurre a lo largo de los siguientes tipos de elementos:

- **Vídeo :** Se muestran contenido multimedia en forma de vídeo. Es usado para la presentación del videojuego y para el vídeo de conclusión.
- **Cómic :** Se muestran los hechos, así como los diálogos entre los diferentes actores del videojuego.
- **Nivel jugable :** El jugador es el protagonista y con sus acciones avanza en la historia controlando al personaje.

A continuación se describe el transcurso de la historia usando cursiva para referirse a los sucesos que se desarrollarán de manera no interactiva en las viñetas de cómic y con estilo de letra normal para referiste a los sucesos que protagoniza el jugador. Para los vídeos multimedia se utilizará negrita.

- **Video inicial : Se presenta el videojuego mostrando de que se trata, los personajes jugables y el tipo de elementos que se encontrará el jugador.**
- ***Cómic 1:*** *Se muestra cómo llegan los personajes al concierto, al principio no se las ve directamente pues se reserva su presentación para cuando se abre el telón directamente en el concierto. En este punto se presentan los personajes que el jugador podrá*

controlar y se inicia el concierto. Llegada la mitad del concierto las protagonistas se van al camerino. Al volver de nuevo para la segunda parte del concierto se encuentran con que en la parte del público todo está lleno de zombies. Estos enemigos empiezan a perseguirlas y echan a correr...

- **Nivel 0 :** Te encuentras todavía en el escenario y tienes que eliminar a los primeros enemigos que te van saliendo. Para ello se te irán mostrando las diferentes acciones que puedes realizar a modo de tutorial y a medida que las vas completando se va avanzando en el nivel y en la realización del tutorial. Llegado a un punto el juego ya te ha mostrado todas las acciones que puede hacer tu personaje y al haberlas completado se da el *tutorial* por finalizado. A partir de aquí se entra en modo libre y debes acabar el nivel eliminando a los enemigos y completando las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- **Nivel 1 :** El personaje ya ha salido del edificio donde se realizaba el concierto y empiezas el nivel en la calle. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 2 :*** *Las protagonistas dialogan en la calle sobre qué ruta van a tomar a continuación.*

- **Nivel 2 :** El personaje continúa en la calle. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 3 :*** *Después de una lucha con algunos enemigos una de las protagonistas sugiere la posibilidad de coger una moto para despejar el camino mientras las otras protagonistas contienen la amenaza del punto actual.*

- **Nivel 3 :** El personaje continúa en la calle. Deberás avanzar y eliminar a los enemigos que te encuentres haciendo uso del vehículo, así como evitar los obstáculos y peligros que hay en la carretera. Para ello el control del personaje es a sobre un vehículo.

- ***Cómic 4 :*** *La protagonista choca contra un edificio e incrusta el vehículo dentro del mismo.*

- **Nivel 4 :** Después de chocar con la moto la protagonista se encuentra en el interior de un edificio y ahora continúas jugando a través de este escenario. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.
- **Nivel 5 :** Continúas en el interior de edificios pero la ambientación cambia pues te has desplazado hacia el edificio contiguo, que aunque sigue siendo un escenario interior, cuenta con una ambientación diferente. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.
- ***Cómic 5 :*** *Las protagonistas otra vez reunidas se encuentran con el primer enemigo final.*
- **Nivel 6 :** Después encontrarte con el primer jefe final que hace uso de mecánicas especiales debes superarlo. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.
- ***Cómic 6 :*** *El primer enemigo final es derrotado y su onda expansiva desplaza a las protagonistas hacia fuera del edificio. Después deciden continuar su camino a través del parque.*
- **Nivel 7 :** Después de eliminar al jefe final y salir del edificio continúas por el parque. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.
- **Nivel 8 :** Después del parque pasas a la siguiente zona contigua a este, el cementerio. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.
- ***Cómic 7:*** *Al final del cementerio se encuentran muchas telarañas y por último dan con el segundo jefe final, la araña gigante.*
- **Nivel 9 :** Tras encontrarte con el segundo jefe final que hace uso de mecánicas especiales debes superarlo. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 8 :*** *El segundo enemigo final es derrotado y las protagonistas bromean un poco al respecto.*

- **Nivel 10 :** Después de eliminar al segundo jefe final el jugador avanza hasta la nueva zona de la estación de tren. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 9 :*** *Al llegar al polígono industrial se encuentran con fuerte presencia militar que las identifica como enemigas. Los militares abren fuego y las protagonistas descubren que a partir de ahora se añadirá este tipo de enemigos a su lista de peligros.*

- **Nivel 11 :** Al final de la estación de tren se encuentra la entrada al polígono industrial, después de cruzarla te encuentras dentro del polígono industrial. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- **Nivel 12 :** Has avanzado por el polígono industrial y ahora estás dentro de uno de los edificios. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 10 :*** *Eliminados todos los militares aún queda uno, el tercer jefe final. En estas viñetas se presenta y se preparan para iniciar el tercer enfrentamiento con un jefe final.*

- **Nivel 13 :** Después de encontrarte con el tercer jefe final que hace uso de mecánicas especiales debes superarlo. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 11 :*** *El tercer jefe final ha sido eliminado y este le da las pistas de cómo combatir la amenaza zombie que los militares habían creado experimentando con armas químicas.*

- **Nivel 14 :** Después de eliminar al tercer jefe final las protagonistas toman un atajo por las alcantarillas para volver de nuevo a las calles. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para

ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- **Nivel 15 :** Ya estás en la calle otra vez. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 12 :*** *Las protagonistas han avanzado por la calle y deben llegar a la otra parte de la ciudad, en estas viñetas se muestra cómo encuentran un coche y deciden cogerlo para cruzar la autopista.*

- **Nivel 16 :** El personaje se encuentra en la autopista. Deberás avanzar y eliminar a los enemigos que te encuentres haciendo uso del vehículo, así como evitar los obstáculos y peligros que hay en la carretera. Para ello el control del personaje es a sobre un vehículo.

- ***Cómic 13 :*** *Dentro del coche las protagonistas hablan sobre como eliminar la amenaza final tras las indicaciones que les ha dado el tercer jefe final. Al final se encuentran con que la carretera por la que circulan está rota y deben bajarse del coche para continuar a pie.*

- **Nivel 17 :** Después de haber avanzado gran parte del camino hacia la otra parte de la ciudad por la autopista debes completar la última parte del camino a pie. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- **Nivel 18 :** Ya has cruzado la autopista y has llegado a la otra zona de la ciudad, que se encuentra derruida tras un ataque militar para erradicar la amenaza de los zombies. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 14 :*** *Ya han llegado a su destino, están tras las puertas del hospital. Ahora sólo queda entrar y encontrarse con el cuarto jefe final.*

- **Nivel 19 :** Después de avanzar los últimos tramos de la ciudad se ha llegado al edificio donde se encuentra el enemigo final, en el hospital. En este nivel deberás avanzar

por dentro del edificio hasta que des con el enemigo que supondrá el encuentro final. Deberás avanzar y eliminar a los enemigos que te encuentres, así como resolver las situaciones que se te plantean. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 15 :*** *Ya han encontrado al último jefe final, el enfrentamiento está dispuesto y se preparan para ello.*

- **Nivel 20 :** Después de encontrarte con el cuarto jefe final que hace uso de mecánicas especiales debes superarlo. Para ello el control del personaje es a pie, haciendo uso de tu guitarra como arma y de la magia que dispone el personaje.

- ***Cómic 16 :*** *Ya has eliminado la última amenaza. En estas viñetas se acaba de resolver la trama del videojuego y al final se deja la puerta abierta para que pueda haber capítulos futuros en el que las protagonistas combatan otra serie de amenazas.*

- **Video final : Tras cerrar la trama en el cómic 16 se introduce como video se ha introducido como recompensa final un video divertido en el que las protagonistas tocan música sobre un escenario mientras que los zombies bailan abajo una famosa coreografía.**

En este apartado se han omitido los diálogos de las diferentes viñetas de cómic pues eso debe consultarse en el documento de guión. No incluido en el presente proyecto por quedar fuera del alcance presente trabajo y no aportar contenido de valor técnico al mismo. El tipo de resultado en la realización de los cómics que se ha obtenido en la versión actual puede apreciarse en las Figuras 5.10, 5.11 y 5.12.

Figura 5.10: Imagen cómic 1

Figura 5.11: Imagen cómic 2

Figura 5.12: Imagen cómic 3

La portada de todos los cómics empleados en la trama del videojuego puede apreciarse a continuación en la Figura 5.13.

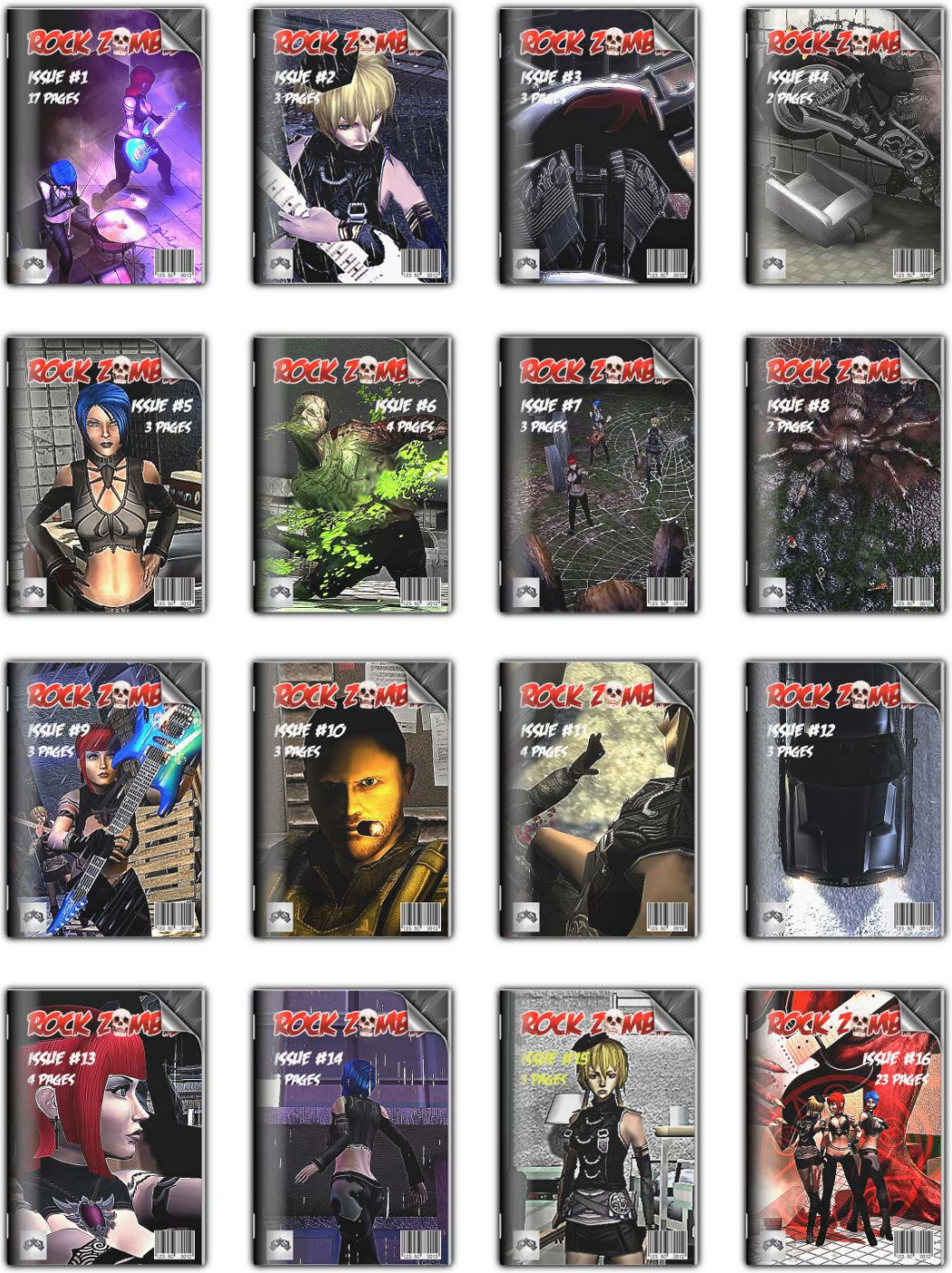


Figura 5.13: Portadas de los cómics

Power ups

En el juego hay determinados *items* que nos proporcionan ventajas temporales o permanentes y cuyo uso modifica el estado de la partida. Tenemos los siguientes:

- **Vida extra :** Disco de vinilo que proporciona una vida extra.

Figura 5.14: *Item* de vida extra

- ***Medikit* :** Cinta de *cassette* que rellena la barra de vida del jugador.

Figura 5.15: *Item medikit*

- **Bote de magia :** Bote de magia que rellena la cantidad de magia del jugador.

Figura 5.16: Bote de magia

- **Monedas :** Monedas que pasan a nuestra cuenta personal cuando las coge el jugador.

Figura 5.17: Monedas

Objetos rompibles

En el juego hay determinados objetos como cajas, barreras o barriles que tienen el logo de *Rock Zombie*. Este tipo de objetos se puede explotar soltando monedas u otros *power ups*.

Figura 5.18: Objeto rompible

Peligros

En el juego nos encontraremos con los siguientes peligros repartidos por el escenario:

- **Charco de veneno :** Zona con veneno que nos daña al pasar por encima.
- **Chorro de fuego :** Chorro de fuego que se activa de manera intermitente a modo de fuga de gas desde una tubería.
- **Fuego :** Zona con fuego que nos daña al acercarse.
- **Mina :** Mina sensible a la proximidad que explota al acercarse.
- **Raíces :** Charco de arenas movedizas que despliega unas raíces sensibles a la proximidad al acercarse.
- **Telaraña :** Telaraña colocada en el suelo a modo de trampa que se pliega cuando pasa el jugador por encima dañándolo.
- **Electricidad :** Zona con fuego que nos daña al acercarse.

Objetos de bloqueo

En el juego nos encontraremos con objetos que obstruyen nuestro paso y no nos permiten pasar hasta que los rompamos. Los objetos son los siguientes:

- **Barreras :** Una barrera que obstruye el paso. En niveles 1, 2 y 4.
- **Coche :** Coche que obstruye el paso. En niveles 2 y 17.
- **Puertas :** Puertas bloqueadas que obstruyen el paso. En niveles 4, 5 y 19.
- **Nevera :** Zona con agua y nevera con cables que producen chispas. Deberemos romper la nevera para que la electricidad no llegue al agua y poder pasar. En el nivel 5.
- **Estatua :** Estatua que obstruye el paso. En el nivel 7.
- **Cajetín eléctrico :** Zona con agua y una cajetín eléctrico con cables que producen chispas. Deberemos romper el cajetín eléctrico para que la electricidad no llegue al agua y poder pasar. En el nivel11
- **Fuego :** Zona de fuego que nos impide el paso. Deberemos de romper un depósito de agua cercano para liberarlo sobre la zona y apagar el fuego. En el nivel 12.
- **Fuga venenosa :** Fuga de líquido venenoso desde una tubería. Deberemos de romper la tubería por otro sitio para que la fuga salga en otra dirección y nos permita pasar. En el nivel 14.

Objetos decorativos

En el juego también hay elementos interactivos decorativos que no afectan a la jugabilidad pero contribuyen a mejorar la ambientación del juego. Nos encontraremos con los siguientes:

- **Rata :** Una rata que se mueve por el escenario.
- **Bandada Pájaros :** Bandada de pájaros que permanecen posados sobre el escenario hasta que pasas cerca y salen volando.
- **Pájaro individual :** Pájaro que permanece posado sobre le escenario.
- **Perro :** Perro que permanece quieto sobre el escenario hasta que pasas cerca y echa a correr.

- **Gato :** Gato que permanece quieto sobre el escenario hasta que pasas cerca y echa a correr.
- **Caída de escombros :** Escombros que caen cuando se pasa por determinada zona.
- **Rotura de cristales :** Cristales que se rompen cuando se pasa por determinada zona.
- **Gotas de agua techo :** Goteras que caen del techo en interiores.
- **Caída de polvo techo :** Caída de polvo y escombros en determinados puntos de los niveles interiores.

Items coleccionables

En este menú el jugador podrá comprar una caja sorpresa por un determinado precio en monedas pero no verá qué ítem le ha tocado hasta que se abra dicha caja. La lista de items que pueden desbloquearse desde el museo *zombie* por orden de aparición detrás de cada caja sorpresa es la siguiente:

- ***Artwork 1* :** 300 monedas
- ***Zoe figure 2* :** 100 monedas
- ***Artwork 2* :** 1050 monedas
- ***Zoe alt costume* :** 100 monedas
- ***Artwork 3* :** 750 monedas
- ***Artwork 4* :** 1000 monedas
- ***Weapon 2* :** 100 monedas
- ***Artwork 5* :** 300 monedas
- ***Spiders figure* :** 1500 monedas
- ***Artwork 6* :** 100 monedas
- ***Marines figure* :** 1000 monedas
- ***Artwork 7* :** 100 monedas
- ***Artwork 8* :** 100 monedas

- ***Artwork 9*** **:** 150 monedas
- ***Artwork 10*** **:** 1000 monedas
- ***Zoe figure 1*** **:** 300 monedas
- ***Artwork 11*** **:** 750 monedas
- ***Artwork 12*** **:** 250 monedas
- ***Sasha figure 2*** **:** 100 monedas
- ***Artwork 13*** **:** 1300 monedas
- ***Crystal figure 1*** **:** 300 monedas
- ***Artwork 14*** **:** 100 monedas
- ***Artwork 15*** **:** 1500 monedas
- ***Sasha alt costume*** **:** 300 monedas
- ***Artwork 16*** **:** 100 monedas
- ***Final boss figure*** **:** 500 monedas
- ***Artwork 17*** **:** 100 monedas
- ***Artwork 18*** **:** 500 monedas
- ***Spider boss figure*** **:** 300 monedas
- ***Artwork 19*** **:** 1000 monedas
- ***Artwork 20*** **:** 650 monedas
- ***Crystal figure 2*** **:** 900 monedas
- ***Artwork 21*** **:** 500 monedas
- ***Artwork 22*** **:** 1000 monedas
- ***Zombies figure 2*** **:** 300 monedas
- ***Artwork 23*** **:** 1100 monedas

- ***Artwork 24*** **:** 1800 monedas
- ***Marine boss figure*** **:** 300 monedas
- ***Artwork 25*** **:** 200 monedas
- ***Fat Zombie figure*** **:** 100 monedas
- ***Artwork 26*** **:** 100 monedas
- ***Zombies figure 1*** **:** 1050 monedas
- ***Artwork 27*** **:** 250 monedas
- ***Artwork 28*** **:** 50 monedas
- ***Weapon 3*** **:** 100 monedas
- ***Artwork 29*** **:** 700 monedas
- ***Crystal alt costume*** **:** 400 monedas
- ***Artwork 30*** **:** 250 monedas
- ***Sasha figure 1*** **:** 200 monedas
- ***Artwork 31*** **:** 50 monedas

Logros desbloqueables

La lista de logros que se plantean en el videojuego son los siguientes:

- ***Miss universe*** **:** Consigue todos los logros
- ***Red flavour*** **:** Completa el juego con el personaje *Zoe*
- ***Yellow flavour*** **:** Completa el juego con el personaje *Sasha*
- ***Blue flavour*** **:** Completa el juego con el personaje *Crystal*
- ***Purist*** **:** Completa el juego en máxima dificultad sin utilizar ningún *continue*
- ***Collector*** **:** Desbloquea todos *items* coleccionables del museo *zombie*.

- ***Master of metal*** **:** Consigue los logros *'Rock & Roll (gold)'*, *'Heavy Metal (gold)'*, *'Black Metal (gold)'* y *'Death Metal (gold)'*
- ***Rock and roll (silver)*** **:** Realiza 10 veces el combo *rock & roll*
- ***Rock and roll (gold)*** **:** Realiza 50 veces el combo *rock & roll*
- ***Heavy metal (silver)*** **:** Realiza 10 veces el combo *heavy metal*
- ***Heavy metal (gold)*** **:** Realiza 50 veces el combo *heavy metal*
- ***Black metal (silver)*** **:** Realiza 10 veces el combo *black metal*
- ***Black metal (gold)*** **:** Realiza 30 veces el combo *black metal*
- ***Death metal (silver)*** **:** Realiza 5 veces el combo *death metal*
- ***Death metal (gold)*** **:** Realiza 20 veces el combo *death metal*
- ***Love in trenches (Bronze)*** **:** Elimina 25 marines
- ***Love in trenches (Silver)*** **:** Elimina 50 marines
- ***Love in trenches (Gold)*** **:** Elimina 100 marines
- ***Zombie killer (Bronze)*** **:** Elimina 500 *zombies*
- ***Zombie killer (Silver)*** **:** Elimina 2500 *zombies*
- ***Zombie killer (Gold)*** **:** Elimina 5000 *zombies*
- ***Arachnophobia (Bronze)*** **:** Elimina 50 arañas
- ***Arachnophobia (Silver)*** **:** Elimina 100 arañas
- ***Arachnophobia (Gold)*** **:** Elimina 200 arañas
- ***Hot girl (Bronze)*** **:** Destruye 25 objetos explosivos
- ***Hot girl (Silver)*** **:** Destruye 50 objetos explosivos
- ***Hot girl (Gold)*** **:** Destruye 100 objetos explosivos

5.1.4. Niveles

Sistema de jugabilidad de cada nivel

El juego presenta diferentes tipos de mecánicas jugables en función del nivel. A continuación se describe el tipo de jugabilidad que el jugador encontrará.

- **Nivel *tutorial* :** La jugabilidad del nivel *tutorial* enseña la configuración de botones y comandos para que el jugador se familiarice con las acciones y elementos del videojuego. El tipo de jugabilidad de el nivel de tutorial es el mismo que el nivel estándar pero mostrando información contextual sobre diferentes elementos del videojuego.
- **Nivel estándar :** La jugabilidad estándar del videojuego consiste en avanzar por el escenario eliminando los diferentes enemigos que salen a nuestro paso. Esta jugabilidad se describe en detalle en el apartado "Esquema de jugabilidad de un nivel".
- **Nivel vehículo :** En el nivel de tipo vehículo el jugador controlará una moto o un coche que puede moverse de izquierda a derecha. El objetivo es eliminar la mayor cantidad de enemigos mientras se esquivan los obstáculos que el jugador encontrará por el camino (ver Figura 5.19).

Figura 5.19: Jugabilidad en nivel de vehículos

Como obstáculos en este tipo de nivel el jugador deberá esquivar los contenedores, árboles, barreras, resto de coches, zonas de lava, zonas de veneno, roturas en la carretera y salientes de edificios (ver Figura 5.20).

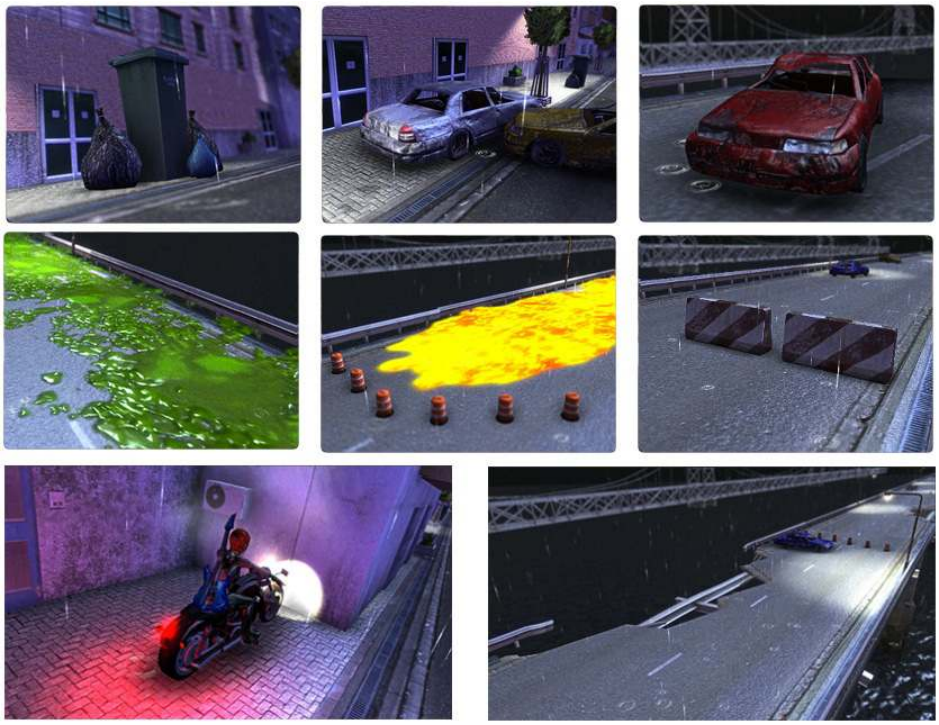


Figura 5.20: Tipos de peligros en nivel de vehículos

- **Jefes finales :** El videojuego presenta diferentes enfrentamientos con jefes finales. Cada uno de ellos tiene una serie de mecánicas y comportamientos diferentes que será necesario contrarrestar para superarlos.

Desglose de niveles

A continuación se muestra un desglose de los niveles señalando dónde se han colocado las diferentes zonas de cámaras, los bloqueos del nivel y los rodajes de cámara.

Los nombres de los niveles se han elegido como guiño a elementos culturales de la música, cine o series de televisión. Para ello se ha partido de nombres clásicos y se ha modificado determinadas palabras o juegos de palabras para evitar problemas de derechos y aportar un tono desenfadado a la obra.

Tutorial - Nivel 0 : *I Love Hack 'N Slash*

Figura 5.21: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 0

Figura 5.22: Detalle 1 de 3 del nivel 0

Figura 5.23: Detalle 2 de 3 del nivel 0

Figura 5.24: Detalle 3 de 3 del nivel 0

Estándar - Nivel 1 : *The girls are back in town*

Figura 5.25: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 1

Figura 5.26: Detalle 1 de 3 del nivel 1

Figura 5.27: Detalle 2 de 3 del nivel 1

Figura 5.28: Detalle 3 de 3 del nivel 1

Estándar - Nivel 2 : *Blood & the city*

Figura 5.29: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 2

Figura 5.30: Detalle 1 de 3 del nivel 2

Figura 5.31: Detalle 2 de 3 del nivel 2

Figura 5.32: Detalle 3 de 3 del nivel 2

Vehículo - Nivel 3 : *Back in Red*

Figura 5.33: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 3

Figura 5.34: Detalle 1 de 3 del nivel 3

Figura 5.35: Detalle 2 de 3 del nivel 3

Figura 5.36: Detalle 3 de 3 del nivel 3

Estándar - Nivel 4 : *Blood on the water*

Figura 5.37: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 4

Figura 5.38: Detalle 1 de 3 del nivel 4

Figura 5.39: Detalle 2 de 3 del nivel 4

Figura 5.40: Detalle 3 de 3 del nivel 4

Estándar - Nivel 5 : *Old jazz serenade*

Figura 5.41: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 5

Figura 5.42: Detalle 1 de 3 del nivel 5

Figura 5.43: Detalle 2 de 3 del nivel 5

Figura 5.44: Detalle 3 de 3 del nivel 5

Jefe final - Nivel 6 : *Enter fatman*

Figura 5.45: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 6

Figura 5.46: Detalle 1 de 3 del nivel 6

Figura 5.47: Detalle 2 de 3 del nivel 6

Figura 5.48: Detalle 3 de 3 del nivel 6

Estándar - Nivel 7 : *The ballad of headless statues*

Figura 5.49: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 7

Figura 5.50: Detalle 1 de 3 del nivel 7

Figura 5.51: Detalle 2 de 3 del nivel 7

Figura 5.52: Detalle 3 de 3 del nivel 7

Estándar - Nivel 8 : *Fallen angels*

Figura 5.53: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 8

Figura 5.54: Detalle 1 de 3 del nivel 8

Figura 5.55: Detalle 2 de 3 del nivel 8

Figura 5.56: Detalle 3 de 3 del nivel 8

Jefe final - Nivel 9 : *Rock and roll queen*

Figura 5.57: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 9

Figura 5.58: Detalle 1 de 3 del nivel 9

Figura 5.59: Detalle 2 de 3 del nivel 9

Figura 5.60: Detalle 3 de 3 del nivel 9

Estándar - Nivel 10 : *Symphony of delusion*

Figura 5.61: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 10

Figura 5.62: Detalle 1 de 3 del nivel 10

Figura 5.63: Detalle 2 de 3 del nivel 10

Figura 5.64: Detalle 3 de 3 del nivel 10

Estándar - Nivel 11 : *And Justice for us*

Figura 5.65: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 11

Figura 5.66: Detalle 1 de 3 del nivel 11

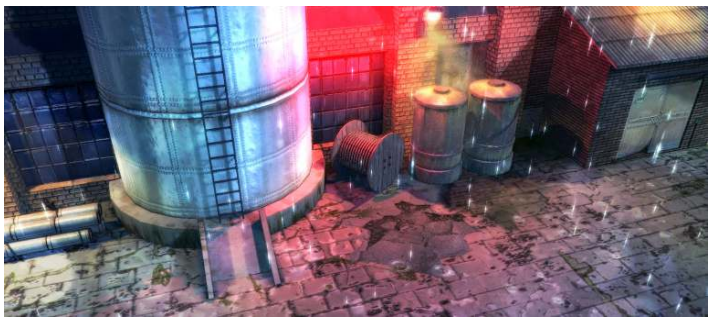


Figura 5.67: Detalle 2 de 3 del nivel 11

Figura 5.68: Detalle 3 de 3 del nivel 11

Estándar - Nivel 12 : *Sympathy for the army*

Figura 5.69: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 12

Figura 5.70: Detalle 1 de 3 del nivel 12

Figura 5.71: Detalle 2 de 3 del nivel 12

Figura 5.72: Detalle 3 de 3 del nivel 12

Jefe final - Nivel 13 : *Heavy metal jacket*

Figura 5.73: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 13

Figura 5.74: Detalle 1 de 3 del nivel 13

Figura 5.75: Detalle 2 de 3 del nivel 13

Figura 5.76: Detalle 3 de 3 del nivel 13

Estándar - Nivel 14 : *Smells like teen zombie*

Figura 5.77: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 14

Figura 5.78: Detalle 1 de 3 del nivel 14

Figura 5.79: Detalle 2 de 3 del nivel 14

Figura 5.80: Detalle 3 de 3 del nivel 14

Estándar - Nivel 15 : *Babe, where's my car?*

Figura 5.81: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 15

Figura 5.82: Detalle 1 de 3 del nivel 15

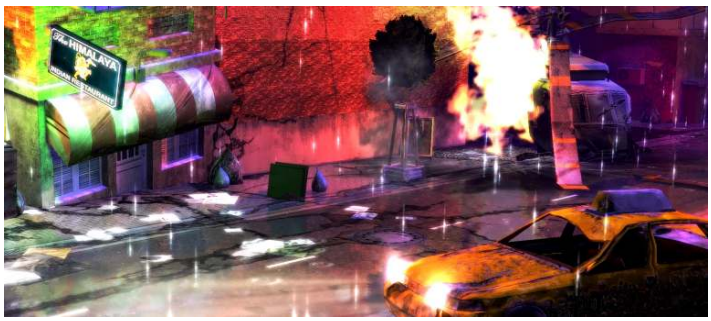


Figura 5.83: Detalle 2 de 3 del nivel 15

Figura 5.84: Detalle 3 de 3 del nivel 15

Vehículo - Nivel 16 : *Angel of asphalt*

Figura 5.85: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 16

Figura 5.86: Detalle 1 de 3 del nivel 16

Figura 5.87: Detalle 2 de 3 del nivel 16

Figura 5.88: Detalle 3 de 3 del nivel 16

Estándar - Nivel 17 : *Highway to paradise*

Figura 5.89: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 17

Figura 5.90: Detalle 1 de 3 del nivel 17

Figura 5.91: Detalle 2 de 3 del nivel 17

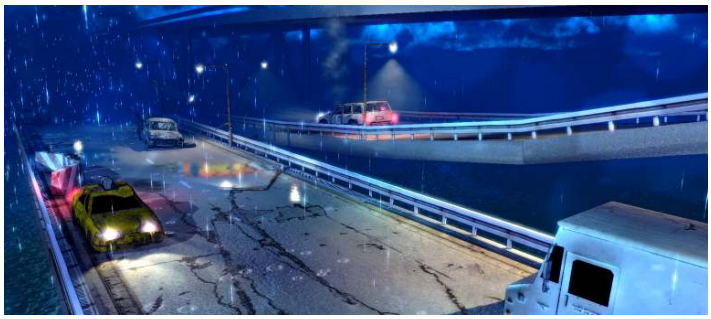


Figura 5.92: Detalle 3 de 3 del nivel 17

Estándar - Nivel 18 : *Bohemian night*

Figura 5.93: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 18

Figura 5.94: Detalle 1 de 3 del nivel 18

Figura 5.95: Detalle 2 de 3 del nivel 18

Figura 5.96: Detalle 3 de 3 del nivel 18

Estándar - Nivel 19 : *Last waltz*

Figura 5.97: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 19

Figura 5.98: Detalle 1 de 3 del nivel 19

Figura 5.99: Detalle 2 de 3 del nivel 19

Figura 5.100: Detalle 3 de 3 del nivel 19

Jefe final - Nivel 20 : *La petite mort*

Figura 5.101: Distribución de cámaras, bloqueos y rodajes de cámara del nivel 20

Figura 5.102: Detalle 1 de 3 del nivel 20

Figura 5.103: Detalle 2 de 3 del nivel 20

Figura 5.104: Detalle 3 de 3 del nivel 20

5.1.5. Interface

Como principal elemento de *feedback* se usará el *HUD* que se muestra por pantalla para mostrar información como la que puede apreciarse en la figura 5.105:

Figura 5.105: *HUD* del videojuego

- **Zona superior izquierda :** El personaje seleccionado, la vida disponible, la magia disponible y el número de vidas.
- **Zona superior derecha :** Número de golpes encadenados para combo, modificador *X2* o *X3* si se hubiera superado un número específico de golpes.
- **Zona inferior izquierda :** Información sobre qué botón pulsar para obtener información de contexto.
- **Zona inferior derecha :** Número de puntos y monedas que se llevan por ahora en el transcurso del nivel.
- **Zona central :** Se muestra la vida de los enemigos como una barra, también la puntuación obtenida al realizar determinados golpes y se modifica el color de los personajes (principales y enemigos) si alguna acción les resta nivel de vida.

Cuando alguno de los enemigos sufra daño se dará el siguiente *feedback* :

- La figura del enemigo se pondrá en rojo.
- Su barra de vida individual bajará.
- Se instanciarán efectos de partículas en el punto del impacto.
- Temblor leve de la cámara virtual.

Cuando el jugador sufra daño se dará el siguiente *feedback* :

- *Feedback* sonoro a modo de quejido del personaje.
- Su barra de vida individual bajará.
- Se instanciarán efectos de partículas en el punto del impacto.
- Uso del efecto *chromatic aberration* sobre la pantalla.
- Temblor intenso de la cámara virtual.

Cuando haya explosiones, temblores o impactos :

- *Feedback* sonoro del evento.
- Vibración del dispositivo en caso de *smartphones*.
- Temblor muy intenso de la cámara virtual.

También se información en pantalla en los siguientes eventos :

- Cuando el jugador haya terminado con todos los enemigos se mostrará el indicativo *GO* sobre la pantalla para que le jugador avance hasta la siguiente zona.
- Cuando haya enemigos activos pero no se estén visualizando por pantalla habrá una flecha y un indicativo que mostrará el texto *"Near Enemy"* apuntando hacia la zona donde encontraremos esos enemigos.
- Cuando haya que romper un determinado elemento para poder pasar por una zona habrá un indicativo que nos mostrará el texto *"Break IT"* apuntando sobre el objeto con el que tenemos que interactuar.

5.1.6. Inteligencia Artificial

La inteligencia artificial adoptada, así como un detalle de este subsistema concreto puede consultarse en el apartado "Desarrollo" de este mismo capítulo.

5.1.7. Apartado técnico

Hardware objetivo

El *hardware* en el que va a comercializarse el producto final puede consultarse en el apartado "Comercialización" de este mismo capítulo.

Desarrollo

La arquitectura tecnológicas adoptada, así como un detalle de cada subsistema pueden consultarse en el apartado "Desarrollo" de este mismo capítulo.

Game Engine

Las herramientas de desarrollo y metodología empleada pueden consultarse en el capítulo 4.

5.1.8. Gestión del proyecto

Evolución detallada de las iteraciones

La evolución de las iteraciones y la gestión del proyecto puede consultarse en el apartado "Evolución y Costes" de este mismo capítulo.

Presupuesto

El presupuesto del videojuego puede consultarse en el apartado "Evolución y Costes" de este mismo capítulo.

Análisis de riesgos

El análisis de riegos del proyecto puede consultarse en el apartado "Comercialización" de este mismo capítulo.

Plan de localización

El plan de localización futura del videojuego puede consultarse en el capítulo 6.

Plan de *testing*

La información relativa a las pruebas de los diferentes prototipos puede consultarse en el apartado “Evolución y Costes” de este mismo capítulo.

5.2. DESARROLLO

En este apartado se describe la arquitectura del videojuego a través de un enfoque *top-down*, comenzando con una descripción general y refinando la especificación progresivamente en cada apartado donde se cubrirán los aspectos específicos de cada subsistema, las problemáticas detectadas y la solución tecnológica que se plantea para su resolución.

5.2.1. Descripción general de la arquitectura del videojuego

La arquitectura del videojuego está compuesta a grandes rasgos por los siguientes subsistemas tal como puede apreciarse en la figura 5.106.

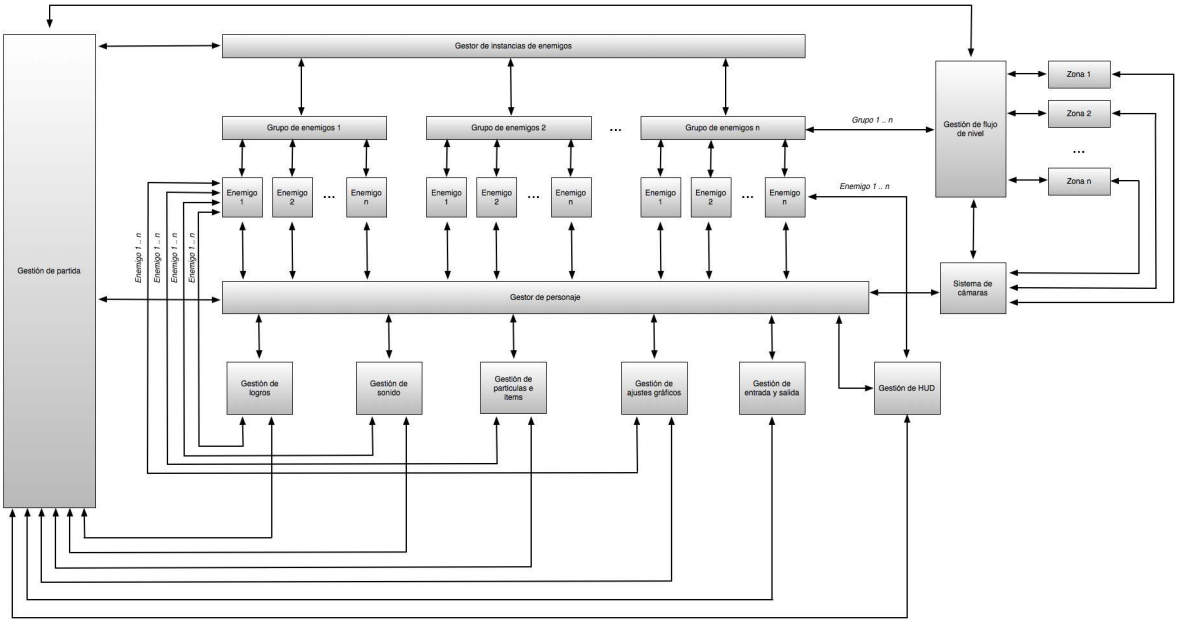


Figura 5.106: Esquema de la arquitectura software de un nivel de *«Rock Zombie»*

- **Gestión de partida** : Este subsistema se encarga de controlar el flujo de la partida, con acciones tales como el inicio del nivel, la gestión del menú o el estado de Pausa. También gestiona el número de vidas del personaje de modo que si llega a cero debe de cargar la escena *Game Over*. Por otro lado sirve de nexo de unión entre diversos componentes para gestionar el *gameplay*.

- **Gestión de flujo de nivel** : Controla en qué zona se encuentra el personaje, qué barreras tiene a cada lado que le impidan avanzar el *scroll* y qué zonas de cámaras hay

activas.

- **Gestión de cámaras** : El sistema de gestión de cámaras interpola la posición de la cámara conforme a la posición del jugador. También se encarga de realizar los rodajes de cámara y el mezclado suave entre zonas.

- **Gestión de personaje** : El sistema de gestión del personaje se encarga de controlar las acciones del personaje.

- **Gestión de enemigos** : Cada enemigo es individual pero va asociado con un grupo. El sistema de gestión de enemigos se encarga de dar un comportamiento adecuado a los enemigos.

- **Gestión de grupos** : Los grupos gestionan a los enemigos asociados.

- **Gestión de creación de enemigos** : El sistema de creación de enemigos se emplea para reciclar instancias y devolverlas a los grupos conforme las demandan. De esa forma no hace falta instanciar desde el principio del nivel a todos los enemigos, sino el subconjunto mínimo necesario para hacerlos funcionar de manera aislada en cada grupo.

- **Gestión de partículas e *items*** : El sistema de gestión de partículas controla los *items* y efectos de partículas para llevar una política de *pooling* y permitir reutilizar las instancias.

- **Gestión de *hud*** : El sistema de *hud* actualiza la información que se muestra por pantalla como la cantidad de vida, de magia o la vida de los enemigos.

- **Gestión de sonido** : El sistema de gestión de sonido reproduce la música y efectos de sonido y se encarga de gestionar el número de recursos que se reproducen simultáneamente para aplicar una política de uso.

- **Gestión del controlador de E/S** : El sistema de entrada y salida permite obtener el control de teclados y *joysticks* para aplicarlo sobre los elementos del videojuego como el personaje o los menús.

- **Gestión de logros** : El sistema de gestión de logros lleva una estadística de los sucesos que pasan en el juego y determina cuándo se ha cumplido con un objetivo secundario para proporcionar un logro al jugador.

- **Gestión de ajustes gráficos** : El sistema de gestión de ajustes gráficos permite configurar los valores gráficos de sombras, reflexiones, cantidad de luces, resolución, brillo, contraste, etc. Estos valores permiten adaptar la visualización al *hardware* en el que se está ejecutando el videojuego. El sistema soporta un cambio de valores *en caliente* propagando los cambios hacia todas las entidades dependientes de adoptar esos valores.

5.2.2. Descripción general del comportamiento del sistema

El comportamiento general tiene como meta la simulación de un entorno virtual con elementos que se comportan de manera individual y simultánea. Cada sistema tiene unos objetivos, y conforme a su comportamiento, generan eventos que influyen a otros sistemas. Estos eventos y la relación entre los diferentes sistemas se implementan mediante el paso de mensajes.

Para simular este entorno de elementos simultáneos sin tener concurrencia se hace uso de una ejecución en tiempo real que realiza 60 pasadas por cada segundo. En cada una de estas pasadas o *frames* de renderizado se actualizan los diferentes estados de cada sistema y se recogen o generan los diferentes eventos que emite cada sistema.

El sistema esta optimizado para garantizar que cada una de esas pasadas de renderizado pueda ejecutarse en un tiempo máximo de 1/60 segundos, teniendo en cuenta el *hardware* específico sobre el que está ejecutándose el videojuego. Si este requisito no se cumpliera el videojuego no proporcionaría la respuesta de interactividad necesaria para percibir que el movimiento de los diferentes elementos obedece a nuestros valores de entrada.

Política de carga de recursos

Para garantizar que el videojuego funciona correctamente en todas las plataformas se ha optado por adoptar una política de gestión de recursos que carga el contenido de un nivel en memoria desde el principio. Después conforme se van necesitando esos recursos se hace uso de ellos teniéndolos ya pre-cargados.

Para adaptar los sistemas a este modo de gestionar los recursos se ha tenido que desarrollar políticas específicas para sistemas como la gestión de partículas, gestión de sonidos, gestión de creación de enemigos o gestión de creación de niveles procedurales. Estos subsistemas se tratarán en profundidad en los siguientes apartados donde se detalla el funciona-

miento de cada uno de estos sistemas de forma individual.

Para garantizar que el videojuego funcione con un amplio rango de dispositivos se ha establecido el requisito de que cada nivel, tras realizar la carga de elementos en memoria, no ocupe más de 512 *Mb* de *RAM* con todos los elementos y sistemas cargados. Esta restricción permitirá ejecutar el juego sin tener que hacer carga y descarga de elementos bajo demanda. Esta restricción permitirá su despliegue en dispositivos móviles con 1 *GB* de memoria *RAM*, dejando los otros 512 *Mb* restantes para el correcto funcionamiento del sistema operativo *Android* o *iOS*.

Solución tecnológica empleada

Para llevar a cabo la implementación de la arquitectura se ha empleado el motor de videojuego *Unity3D* y se ha hecho uso del paradigma de programación orientada a componentes. En el capítulo 4 del presente documento se detallan los aspectos principales relativos a la gestión de este motor.

Con el fin de crear una arquitectura tecnológica viable que permita ejecutar el videojuego y gestionar los eventos y sucesos de un nivel se ha estructurado el sistema como muestra la figura 5.106.

Algunos de esos subsistemas son autónomos y no requieren de otros para su funcionamiento como es el caso de Gestión de la entrada y salida o Gestión de logros. Para gestionar el resto de componentes y las conexiones entre ellos se ha desarrollado un sistema específico, Gestor de componentes.

5.2.3. Gestión de componentes

Problemática

Los sistemas que gestionan el flujo de una partida están acoplados y requieren de una comunicación constante entre ellos. En *Unity3D* puede crearse una puntero de referencia a otro *script* en cualquier punto del código. Este tipo de tareas se crean habitualmente para gestionar el paso de mensajes entre componentes. Esta aproximación no es eficiente, ya que requiere crear un puntero cada vez que es necesario enviar un mensaje. De igual modo, tampoco es aconsejable que cada componente haga la gestión de los punteros que utiliza al principio, pues unos *scripts* tienen dependencias de otros y no es posible controlar el orden

de creación individual de dichos punteros. El problema demanda una solución que pasa por crear un sistema intermedio que gestione la creación de estos punteros de forma ordenada y proporcione su referencia al resto de *scripts* que requieran comunicación con otros.

Solución tecnológica empleada

Se ha optado por mantener cada *script* individual como componentes estándar y desarrollar otro subsistema que denominaremos Gestor de componentes que se encarga de gestionar los punteros a cada uno de los componentes del sistema.

El patrón que se ha usado para implementar este sistema es ***Singleton*** y su objetivo consiste en crear los punteros que enlazan con cada uno de los componentes del sistema para servírselo a su vez a cada uno de los otros componentes que los demandan y permitir así la agrupación centralizada de estos punteros actuando también como ***Facade*** con los diferentes componentes que requieren de una comunicación con otros sistemas.

Este componente también tiene en cuenta un requisito importante, y es el orden de creación de estos punteros, pues algunas de las tareas de inicialización requieren de que el puntero a otros subsistemas estén disponibles previamente.

Los punteros a componentes que crea y proporciona este Gestor de componentes son los siguientes:

- ***Sonidos*** : Puntero al *script* del gestor de sonidos.
- ***Game*** : Puntero al *script* del gestor de partida.
- ***Player*** : Puntero al *script* de comportamiento del jugador.
- ***AvanzarObjeto*** : Puntero al *script* de comportamiento de vehículo.
- ***FXManager*** : Puntero al *script* del gestor de efectos avanzados.
- ***CamaraShake*** : Puntero al *script* de temblor de la cámara.
- ***FinalBoss1*** : Puntero al *script* de comportamiento del jefe final 1.
- ***FinalBoss2*** : Puntero al *script* de comportamiento del jefe final 2.
- ***FinalBoss3*** : Puntero al *script* de comportamiento del jefe final 3.

- ***FinalBoss4*** : Puntero al *script* de comportamiento del jefe final 4.
- ***Shadows*** : Puntero al *script* de configuración global de las sombras.
- ***Camara*** : Puntero al *script* de comportamiento de la cámara principal.
- ***Moto*** : Puntero al *script* de comportamiento de la motocicleta.
- ***GestorDerrumbamiento*** : Puntero al *script* de comportamiento de columnas destructibles.
- ***InterpoladorDeCamara*** : Puntero al *script* de comportamiento de la interpolación de la cámara.
- ***Tutorial*** : Puntero al *script* de comportamiento del tutorial del primer nivel.
- ***TutorialVehiculos*** : Puntero al *script* de comportamiento del tutorial resumido de vehículos.
- ***InstanceManager*** : Puntero al *script* del sistema de creación de instancias de partículas y objetos de juego.
- ***MaterialPool*** : Puntero al *script* que proporciona los diferentes materiales preparados para usar en el sistema de composición de enemigos que reutiliza las instancias.
- ***HudSystem*** : Puntero al *script* que proporciona el comportamiento del *HUD*.
- ***InstanciaEnemigosDinamica*** : Puntero al *script* del gestor de creación de enemigos en los jefes finales.
- ***GestorDeBloqueos*** : Puntero al *script* del gestor de bloqueos y barreras invisibles del nivel.
- ***GestorAnimacionFinal*** : Puntero al *script* que gestiona la animación final de los personajes cuando se supera un nivel.
- ***GestorDeAnimacionesBoss*** : Puntero al *script* que gestiona la animación final de los jefes finales.
- ***ColaEnemigos*** : Puntero al *script* que gestiona la cola de creación de enemigos.

- ***GestorBarrasEnemigos*** : Puntero al *script* que gestiona las barras de vida de los enemigos y su posicionamiento en el *HUD*.

Según el nivel algunos de estos punteros serán creados y otros no. Esto dependerá de cuáles tienen asociado un *Transform* en el inspector de *Unity3D* para el nivel concreto.

Hay que tener en cuenta que este sistema no proporciona el puntero de todos los scripts del videojuego, sino sólo a aquellos que por su interconexión con otros requieren paso de mensajes.

5.2.4. Gestión de partida

Problemática

En este videojuego se pretende recrear un entorno virtual en el que intervienen muchos actores : El personaje, los enemigos, las cámaras, el entorno, etc. Para que estos actores se comporten conforme a un contexto común surge la necesidad de un sistema que actúe a modo de director de orquesta. Dicho sistema debe conocer a cada entidad del nivel para comunicarles cuándo el juego se encuentra en pausa, cuándo ha terminado la partida o cuándo ha muerto el jugador principal. Además cada entidad que requiera invocar el cambio de estado de estas reglas comunes también debe conocer a este sistema.

Solución tecnológica empleada

Se ha desarrollado un sistema denominado Gestión de partida. El objetivo general del sistema es la sincronización de la partida entre las diferentes entidades. Para ello se disponen unas reglas que permiten marcar el inicio, pausa, fin del nivel, muerte del jugador y otra serie de eventos comunes que afectan a los actores de la partida en curso. Estos eventos pueden ser disparados por otras entidades, y cuando se lanzan afectan al comportamiento de la partida.

El patrón que se ha usado para implementar este sistema es ***Facade***. Entre algunas de las tareas que permite realizar este sistema podemos destacar :

- Iniciar partida e invocar la animación inicial.
- Finalizar partida e invocar la animación final.

- Pausar y reanudar el juego.
- Gestionar el control del menú de pausa.
- Dar paso a la escena de *Game Over* si el jugador no dispone de más vidas.
- Activar y desactivar *HUD* para los rodajes de cámara.
- Iniciar secuencia de *slow motion* para una cámara.
- Invocando diferentes grados de vibración para la cámara activa de la partida.
- Hacer *respawn* del jugador tras su muerte.
- Invocar el inicio de tutoriales.
- Llevar un control de tiempo y estadísticas de la partida.

5.2.5. Gestión de flujo de nivel

Problemática

Cuando el jugador comienza un nivel no puede avanzar por toda la geometría del entorno hasta el final, porque entonces lo recorrería en poco tiempo. Además esto implicaría que se ha ido encontrando con todos los enemigos y los ha ido esquivando, generando tras de sí una fila de antagonistas que lo persiguen por el nivel y que se le acumularán al final del mismo.

Este problema no sólo va en contra de la jugabilidad, sino del propio rendimiento del sistema pues no es posible renderizar un número elevado de enemigos en pantalla al mismo tiempo.

Solución tecnológica empleada

Para evitar este problema el jugador sólo puede avanzar en una región acotada del nivel, y en dicha zona luchar con un número acotado de enemigos. Tras superar estas amenazas se le debe permitir avanzar hasta otra zona donde se encontrará con retos nuevos.

Para ello el nivel está subdividido conceptualmente en fases que se van sucediendo a medida que se superen los enemigos, amenazas o retos que se le plantean al jugador.

Para modelar el comportamiento de este sistema se ha construido un sistema que activa y desactiva las diferentes fases en la escena de manera secuencial conforme a los eventos que recibe de otras entidades. Una fase es un *GameObject* colocado sobre una escena y en cuya jerarquía se encuentran :

- **Una o varias zonas de cámara :** Estas zonas de cámara son controladas por el sistema de gestión de cámaras.
- **Una o varias barreras físicas :** Estas barreras son geometría que se utiliza en el cálculo de colisiones por parte del sistema de gestión de físicas. Impiden que el jugador avance a través de ellas y se colocan al inicio y final de la zona para delimitar el paso del jugador.

En este sistema pueden invocarse las siguientes tareas por parte de otros actores de la escena :

- **Poner fase :** Establece la fase directamente, activando o desactivando para ello las zonas que correspondan.
- **Aumentar fase :** Aumenta la fase a la siguiente, activando o desactivando para ello las zonas que correspondan. Puede realizarse de forma suave invocando al sistema de cámaras para que interpole las posiciones de la cámara actual con la de la nueva fase o de manera brusca.
- **Mostrar *GO* :** Cuando se produce un cambio de fase, se puede invocar al sistema de interface que muestra un mensaje en pantalla para que el jugador tenga conocimiento de que puede avanzar.

La interpolación suave de las cámaras en un cambio de fase dependen del propio sistema de gestión de cámaras.

Las fases dentro de los niveles se han dividido en dos tipos :

- **Fase Enlace :** En este tipo de fase sirve de enlace entre dos fases de acción. No hay enemigos en estas fases y sólo sirven para que el jugador se pueda desplazar desde una zona acotada a la siguiente zona acotada. Generalmente cuando se alcance un punto determinado del escenario se pasará a la siguiente fase que será una fase de acción. A continuación nos referiremos a este tipo de fases como "Fase".

- **Fase Acción :** En este tipo de fase el jugador debe eliminar una serie de enemigos o superar una serie de desafíos propuestos. La consecución de estos objetos hará que se aumente la fase hacia la siguiente, que se tratará de una fase de enlace. A continuación nos referiremos a este tipo de fases como "*Fase*".

Eventos de cambio de fase :

El cambio de una fase hacia la siguiente puede ser disparada por los siguientes eventos:

- Eliminación de todos los enemigos previstos para esa fase
- Detección de que el jugador ha alcanzado un determinado punto del escenario
- Lanzamiento de un rodaje de cámara

Además como resultado de finalización de la última fase del nivel se disparará el lanzamiento de la animación de fin de nivel.

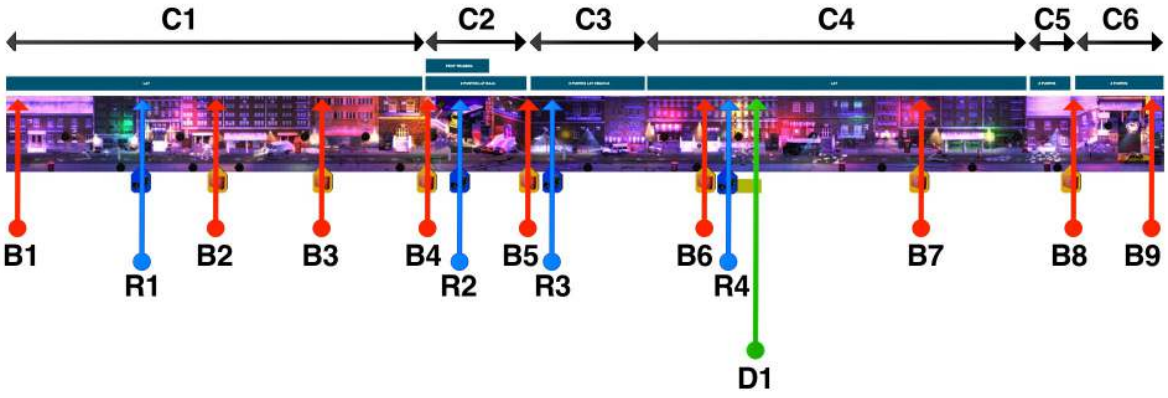


Figura 5.107: Desglose de *gameplay* del nivel 1

En la Figura 5.107 se muestran las zonas de cámara bloqueos, rodajes y desafíos del primer nivel de *«Rock Zombie»*. Los colores y leyendas de la imagen tienen el siguiente significado :

- **C1 - C6 :** Zonas de cámara diferentes por las que el jugador se puede desplazar.

- **B1 - B9 :** Bloqueos que no permiten avanzar al jugador a menos que haya superado la eliminación de todos los enemigos o desafíos de la fase que lleva asociadas esos bloqueos.
- **R1 - R4 :** Rodajes de cámara que se dispararán cuando el jugador llegue a ese punto y que resaltarán o centrarán la atención del jugador sobre algún elemento del escenario.
- **D1 :** Desafío planteado al jugador. No podrá avanzar hasta que lo supere.

A continuación se detallan las diferentes zonas de cámaras :

- **C1 :** Zona de cámara con *scroll* lateral
- **C2 :** Zona de cámara de interpolación entre 2 puntos en la zona frontal y en la zona trasera tenemos una zona de cámara en detalle
- **C3 :** Zona de cámara de interpolación entre 2 puntos
- **C4 :** Zona de cámara con *scroll* lateral
- **C5 :** Zona de cámara de interpolación entre 2 puntos
- **C6 :** Zona de cámara de interpolación entre 4 puntos

A continuación se detallan los diferentes rodajes de cámara :

- **R1 :** La cámara muestra que sobre la carretera hay restos de sangre de *zombies*
- **R2 :** La cámara muestra cómo en el callejón hay monedas tras una barrera
- **R3 :** La cámara muestra cómo hay un coche con un *zombie* dentro atrapado
- **R4 :** La cámara muestra cómo el personaje se ha topado con una barrera que debe superar para poder continuar

En cuanto al desafío **D1** el jugador ve impedido el paso por unas barreras y para poder continuar debe utilizar los diferentes ataques con los que cuenta el personaje para romperlas y pasar.

Los bloqueos desde **B1** hasta **B9** suponen paredes imaginarias que impiden que el jugador se vaya de la zona sobre la que se centra la acción. Este tipo de límites permiten acotar el número de sucesos que están pasando en un momento determinado y a controlar mejor dónde se encuentra el jugador para proporcionar una jugabilidad adaptada a cada zona, no teniéndose así bajones de intensidad de acción o acumulación de muchos enemigos en una misma zona.

Ejemplo del comportamiento del sistema

A continuación se describe como se comportaría el sistema en la ejecución del nivel 1. Dicho nivel dura aproximadamente 5 minutos para el jugador y está dividido en 15 zonas para gestionar el comportamiento del mismo.

- **Inicio :** Se muestran las imágenes de presentación del nivel junto con el título del nivel
- **Animación inicial :** Se muestra la animación inicial del nivel
- **Fase 0 :** El jugador comienza el nivel y tiene permitido moverse entre las barreras B1 y B2. Cuando avanza un poco se encuentra con el rodaje de cámara que dispara R1. Tras avanzar un poco más se dispara la *Fase* 1 por contacto con un *trigger*.
- ***Fase* 1 :** El jugador se encuentra con el grupo 1-1 de *zombies* que tiene que eliminar. Tras eliminarlos se pasa a la fase Fase 2
- **Fase 2 :** El jugador puede avanzar desde B1 hasta B3. Cuando avanza más allá de B2 se encuentra con un *trigger* que dispara la *Fase* 3
- ***Fase* 3 :** El jugador puede avanzar desde B2 hasta B3. Salen a su encuentro el grupo de *zombies* 2-1. Cuando elimina a este grupo sale el grupo de *zombies* 2-2, cuando elimina al grupo de *zombies* 2-3 se aumenta la fase hasta Fase 4
- **Fase 4 :** El jugador puede avanzar desde B2 hasta B4. Cuando avanza más allá de B3 se aumenta hasta la *Fase* 5.
- ***Fase* 5 :** El jugador puede avanzar desde B3 hasta B4. Salen a su encuentro el grupo de *zombies* 4-1. Cuando elimina a este grupo se avanza hasta la Fase 6.
- **Fase 6 :** El jugador puede avanzar desde B3 hasta B5. Después de avanzar un poco, el jugador entra en el *trigger* que dispara el rodaje de cámara R2. Después de terminar el rodaje de cámara R2 se aumenta a *Fase* 7.

- ***Fase* 7 :** El jugador puede avanzar desde B4 hasta B5. Salen a su encuentro el grupo de *zombies* 6-1 y después de eliminarlos el grupo de *zombies* 6-2. Después de eliminar este último grupo se aumenta a Fase 8.

- **Fase 8 :** El jugador puede avanzar desde B4 hasta B6. Cuando avanza un poco el jugador entra en el *trigger* que dispara el rodaje de cámara R3. Después de terminar el rodaje de cámara R3 se aumenta a *Fase* 9.

- ***Fase* 9 :** El jugador puede avanzar desde B5 hasta B6. Salen a su encuentro el grupo de *zombies* 9-1 y después de eliminarlos el grupo 9-2 y 9-3. Tras eliminar a este último grupo se aumenta a Fase 10.

- **Fase 10 :** El jugador puede avanzar desde B5 hasta B7. Después de avanzar un poco se encuentra con el *trigger* que dispara el rodaje de cámara R4. Después de terminar el rodaje de cámara se aumenta a *Fase* 11.

- ***Fase* 11 :** El jugador puede avanzar desde B6 hasta B7 pero en medio tiene una barrera, por lo que deberá eliminarla para poder continuar. Tras eliminarla y pasar salen a su encuentro el grupo de enemigos 11-1 disparados por un *trigger*. Tras eliminarlos salen a su encuentro el grupo de enemigos 11-2 y 11-3. Después de eliminarlos se aumenta a Fase 12.

- **Fase 12 :** El jugador puede avanzar desde B6 hasta B8. Después de pasar más allá de B7 se aumenta a *Fase* 13.

- ***Fase* 13 :** El jugador puede avanzar desde B7 hasta B8. Salen a su encuentro el grupo de enemigos 13-1 y tras eliminarlos el grupo 13-2. Después de eliminar este último grupo se aumenta a Fase 14.

- **Fase 14 :** El jugador puede avanzar desde B7 hasta B9. Después de avanzar más allá de B8 se pasa a *Fase 15*.

- ***Fase* 15 :** El jugador puede avanzar desde B8 hasta B9. Salen a su encuentro el grupo de *zombies* 15-1. Después de eliminarlos al ser el último grupo del nivel se dispara la animación final del nivel.

- **Animación final :** Se muestra la animación final del nivel

- **Resultados :** Se muestran las estadísticas de la partida y las recompensas obtenidas.

5.2.6. Gestión de cámaras

Problemática

La geometría del escenario no es uniforme, contiene pasillos en profundidad, zonas con obstáculos y diferentes alturas y esquemas de edificios u objetos. Este hecho presenta el problema de que no es posible disponer de un único algoritmo que modele el comportamiento de la cámara que sigue al jugador.

El problema a resolver es proporcionar al jugador un punto de vista del personaje adecuado para cada una de las zonas del escenario.

Por otro lado se pretende disponer de un sistema que permita realizar un recorrido de cámara para poder realizar animaciones dentro de la partida.

Solución tecnológica empleada

Para solucionar esta problemática se dispone de un sistema de interpolación de cámaras que permite proporcionar la posición y rotación adecuadas para la cámara que sigue al personaje personaje. Para ello cada *frame* de renderizado se detecta qué zona de cámara dinámica está activa en función de la posición del jugador y se realiza un cálculo de la posición de la cámara con respecto a su algoritmo de comportamiento concreto.

Además este sistema gestiona los rodajes de cámara. Se trata de cámaras no interactivas que son disparadas cuando el personaje alcanza una determinada zona del escenario y se usan para mostrar o centrar la atención sobre algún elemento concreto del nivel.

Por último, este sistema es el encargado de mezclar la animación de dos cámaras cuando se produce el cambio de una fase a la siguiente. Consiguiendo con este efecto una continuidad en el comportamiento de la cámara a lo largo del nivel y las diferentes zonas concretas.

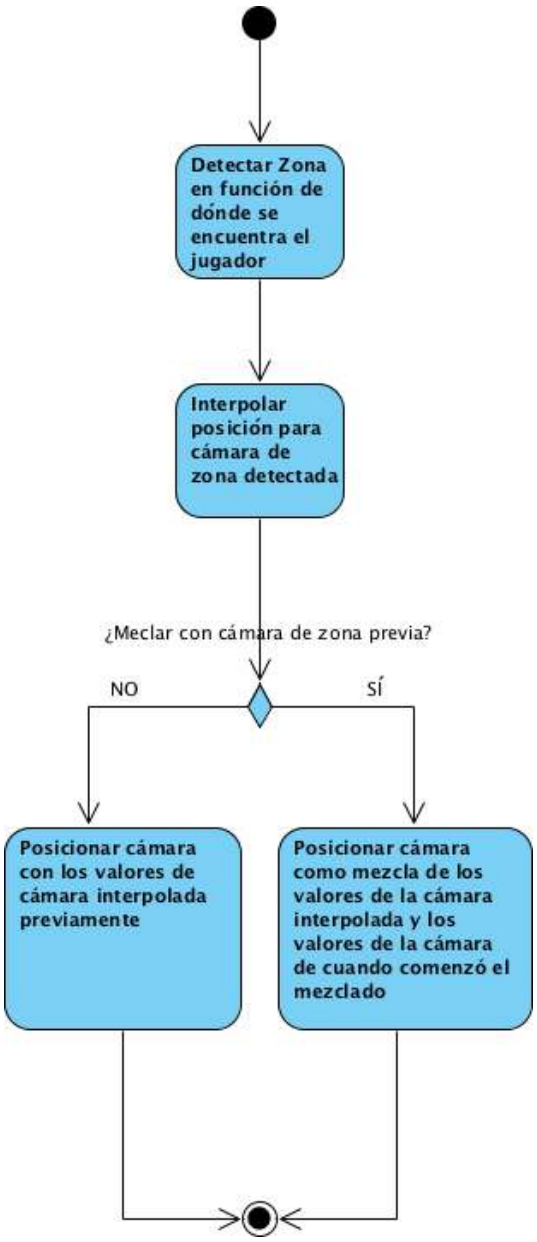


Figura 5.108: Mezclado de cámara entre zonas

Este modo de mezclado puede ser invocado por el sistema de gestión de flujo de nivel cuando se produce un cambio de fase.

Cámaras dinámicas

Las zonas de cámara dinámicas son regiones del espacio que llevan asociado un determinado algoritmo de comportamiento que permite decidir la posición de esta cámara. Cada

una de estas regiones tiene asociada una prioridad por lo que pueden solaparse en el nivel.

Para lograr este objetivo se ha hecho uso de los mecanismos de herencia, teniendo un puntero de clase *Camara* en cada una de las zonas. Este puntero se enlazará con una instancia de clase hija y mediante el polimorfismo se conseguirá que cada zona tenga un comportamiento específico.

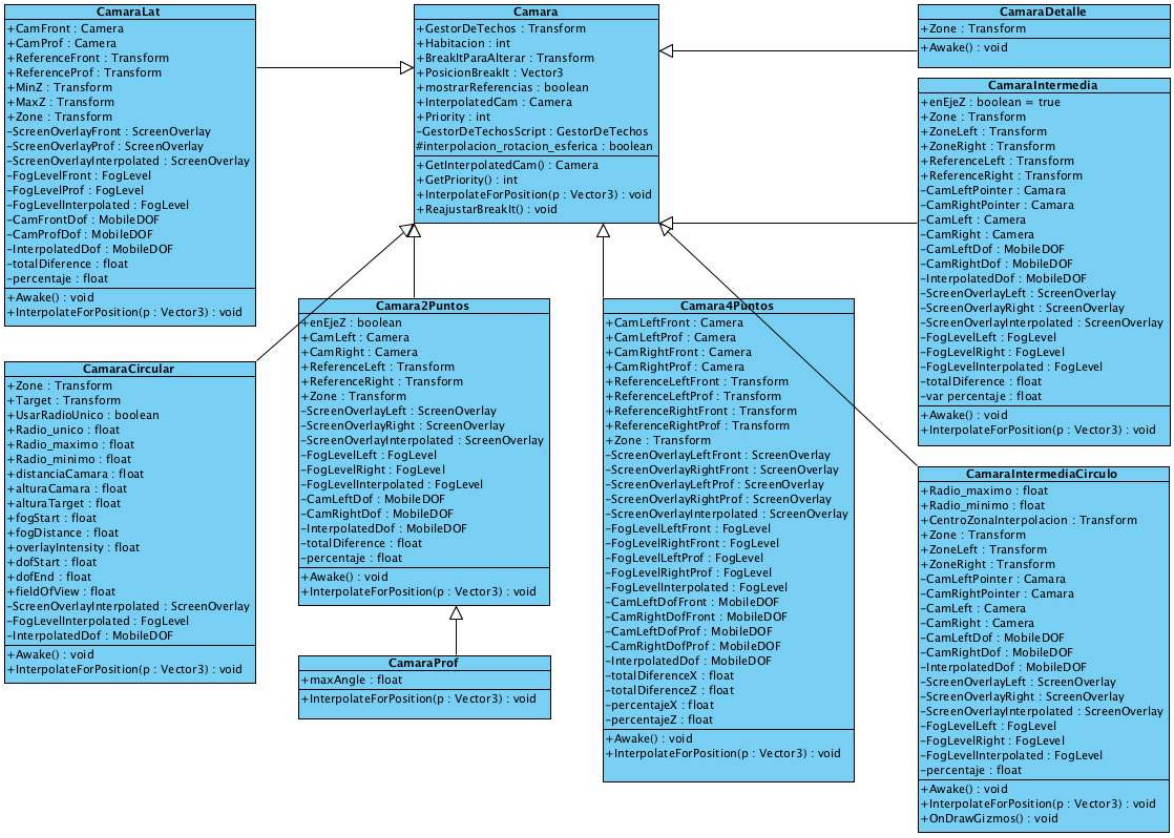


Figura 5.109: Diagrama de clases de los diferentes tipos de cámaras

A continuación se describe el comportamiento del algoritmo de cada uno de los tipos de clase hijo :

- **Cámara de 2 puntos :** Este tipo de cámara (ver Figura 5.110) viene definida por 2 cámaras y 2 posiciones de referencia. Cada cámara va asociada a uno de esos puntos de de referencia en el espacio. Se realizará una interpolación que mezclará la posición, rotación y *FOV* de cada una de estas cámaras en función de la distancia del personaje a cada una de estas posiciones de referencia.

Figura 5.110: Cámara de 2 puntos

- **Cámara de 4 puntos :** Este tipo de cámara (ver Figura 5.111) viene definida por 4 cámaras y 4 posiciones de referencia. Cada cámara va asociada a uno de esos puntos de de referencia en el espacio. Se realizará una interpolación que mezclará la posición, rotación y *FOV* de cada una de estas cámaras en función de la distancia del personaje a cada una de estas posiciones de referencia.

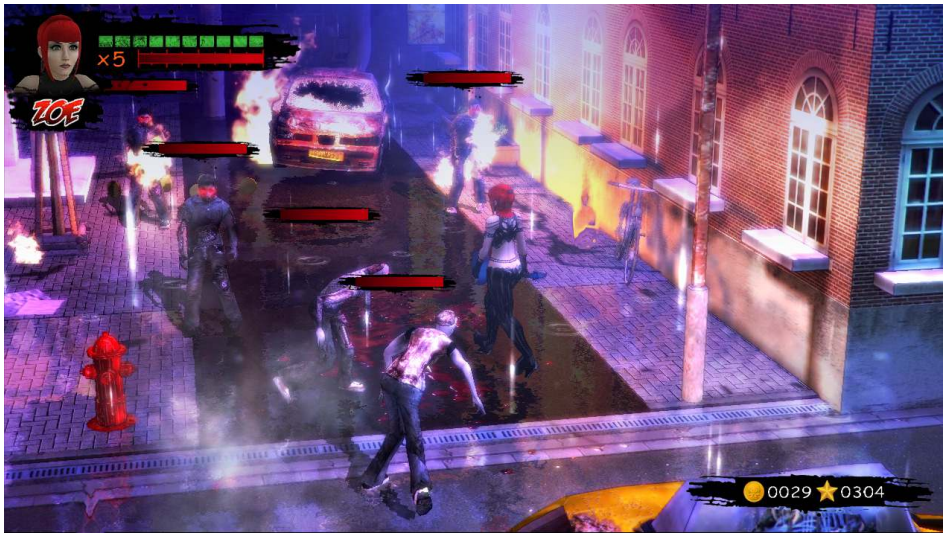


Figura 5.111: Cámara de 4 puntos

- **Cámara de detalle :** Es una cámara fija que muestra una posición, rotación y *FOV* determinados cuando el jugador entra en esta zona. Suele usarse solapando otras cámaras y usando una prioridad más alta para ver en detalle una región del nivel.

Figura 5.112: Cámara de detalle

- **Cámara de *scroll* lateral :** Es una cámara que viene definida por una coordenada Z mínima y una máxima. Sigue al jugador a no ser que se salga de esos rangos de Z. También varía su orientación si el jugador se mueve en profundidad.

Figura 5.113: Cámara de *scroll* lateral

- **Cámara profundidad :** Cámara que sigue al jugador a lo largo de un pasillo en pro-

fundidad. Permite especificar un rango de rotación máximo hacia los lados del pasillo. Viene definida por 2 cámaras, 2 posiciones de referencia y el ángulo de rotación permitido.

- **Cámara vehículo :** Cámara que sigue al vehículo en los niveles de conducción. Realiza una interpolación suave en los giros.

Figura 5.114: Cámara vehículo

- **Camara intermedia :** Se dispone de un tipo de zona especial que permite colocarse entre otras 2 zonas de las cámaras anteriormente nombradas. La misión de la zona de cámara intermedia es realizar una interpolación entre la zona 1 y la zona 2 de modo que el mezclado de una zona a otra sea suave.

El jugador no debe percibir los diferentes tipos de cámara, se tienen que colocar utilizando las zonas intermedias y los saltos entre zonas de forma que el jugador experimente que la acción siempre está pasando desde el mejor punto de vista posible. Para el jugador, la elección del tipo de cámara o zona de cada parte del nivel debe ser transparente.

Rodajes de cámara

Un rodaje es el recorrido de una cámara entre dos posiciones diferentes. Para ello se realiza una interpolación con respecto a una serie de valores a lo largo de un tiempo, dando como resultado el cambio de posición y rotación de la cámara de forma animada. Dicha interpolación se realiza con respecto a los siguientes elementos:

- ***Trigger* :** Región que hará saltar la animación de la cámara si el jugador entra en esta zona.

- **Cámara inicial :** Posición desde la que comienza la animación definida por una posición, rotación y *FOV* de la cámara.

- **Cámara final :** Posición hasta la que que tiene que llegar la animación definida por una posición, rotación y *FOV* de la cámara.

- **Duración inicial :** Tiempo a detener la cámara en posición inicial antes de comenzar la interpolación.

- **Duración interpolación :** Duración de la interpolación entre posición inicial y final.

- **Duración final :** Tiempo a detener la cámara en posición final después de acabar la interpolación.

- **Duración vuelta :** Se puede definir si realizar otra interpolación desde la posición final hasta la posición donde se encuentra la cámara dinámica asociada al jugador en el momento actual. Si se realiza la interpolación de vuelta se puede especificar el tiempo que durará.

5.2.7. Gestión de personaje

Problemática

En el videojuego hay disponibles 3 personajes, *Zoe*, *Sasha* y *Crystal*. Cada uno de ellos tiene diferente balanceo de velocidad, fuerza, resistencia y magia.

Figura 5.115: Habilidades de los personajes

Se debe identificar las acciones que va a poder realizar el personaje y desarrollar un sistema que permita al jugador llevarlas a cabo. Además el comportamiento de este sistema debe ser sensible a los diferentes puntos de habilidades que tenga el personaje seleccionado.

Análisis de las acciones del jugador

La Figura 5.116 muestra las acciones que puede realizar el jugador.

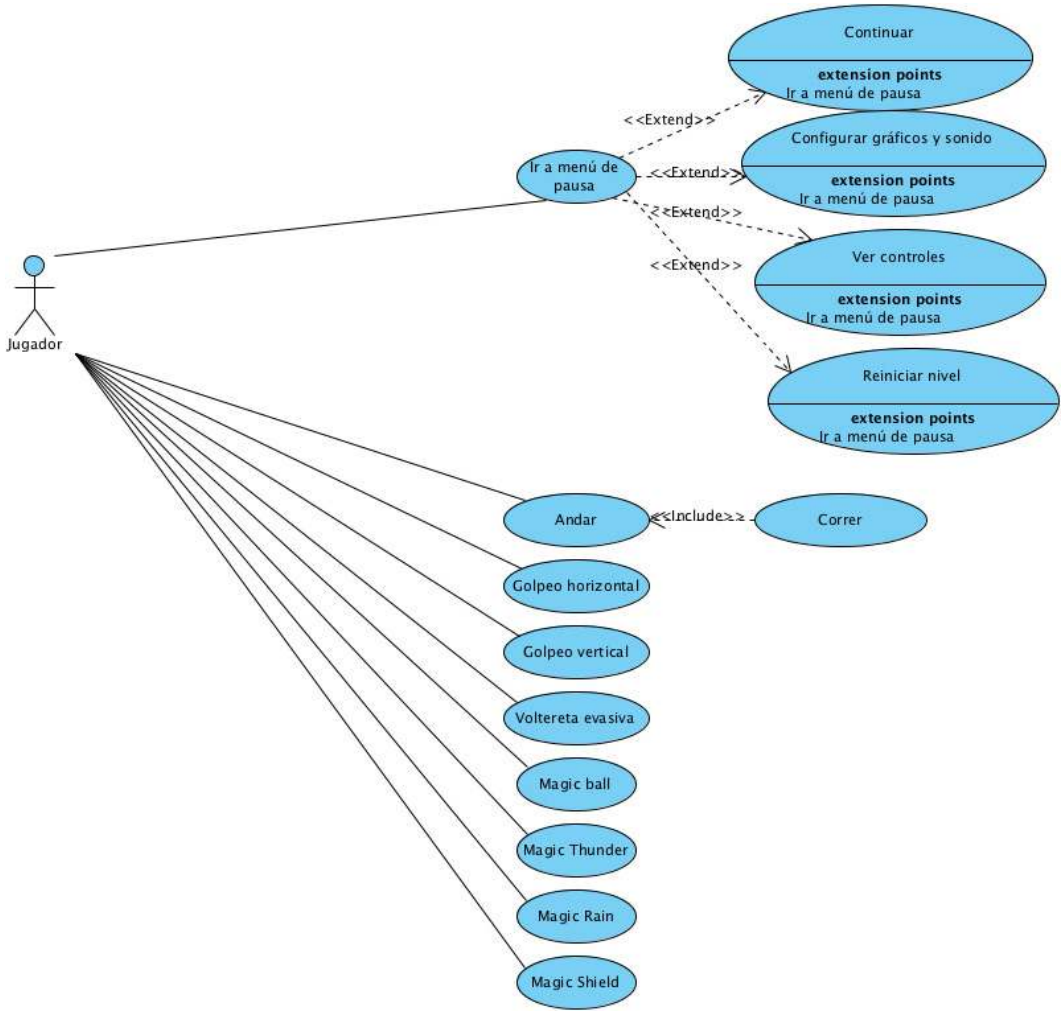


Figura 5.116: Acciones disponibles para el jugador

- **Andar :** El personaje caminará a una velocidad específica dependiendo de la cantidad de movimiento del *stick* analógica. Esto permite andar despacio si se quiere precisión movimiento el *stick* muy poco y andar más rápido si se mueve el *stick* en una dirección de forma completa.

- **Correr :** El personaje también podrá correr si además de mover el *stick* presiona el botón de correr.

Figura 5.117: Personaje corriendo

- **Voltereta evasiva :** El personaje podrá realizar un movimiento evasivo si presiona el botón de voltereta y mueve el *stick* hacia una dirección.

Figura 5.118: Personaje realizando movimiento evasivo

- **Golpeo horizontal :** El personaje realizará un ataque que barrerá su posición de forma horizontal para alcanzar el máximo de enemigos.

Figura 5.119: Personaje realizando ataque horizontal

- **Golpeo vertical :** El personaje realizará un ataque vertical que barrerá su posición de forma vertical para alcanzar enemigos que se encuentran en el suelo o a una altura inferior a su posición.

Figura 5.120: Personaje realizando ataque vertical

- ***Magic Shield*** **:** El personaje podrá cubrirse utilizando el escudo mágico si presiona el botón de magia y el botón de evasión. Este escudo va consumiendo magia mientras está activo.

Figura 5.121: Personaje cubriéndose con escudo mágico

- ***Magic Ball*** **:** El personaje podrá lanzar una bola mágica si presiona el botón de de magia y el botón de ataque horizontal de forma simultánea. Este ataque consume 1/8 de la barra de magia.

Figura 5.122: Personaje lanzando ataque *Magic Ball*

- ***Magic Thunder* :** El personaje podrá lanzar un rayo mágico que desintegrará a los enemigos que toque si presiona el botón de de magia y el botón de ataque horizontal de forma simultánea. Este ataque consume 1/4 de la barra de magia.

Figura 5.123: Personaje lanzando ataque *Magic Thunder*

- ***Magic Rain* :** El personaje podrá lanzar una lluvia de rayos mágicos que desintegrará a los enemigos que se encuentren en un radio determinado si presiona el botón de magia y el botón de evasión de forma simultánea. Este ataque consume 1/2 de la barra de magia.

Figura 5.124: Personaje lanzando ataque *Magic Rain*

También pueden realizar los siguientes *combos* concatenando diferentes movimientos. El uso de combos proporcionará mayor cantidad de puntuación.

Figura 5.125: Listado de *combos*

Solución tecnológica empleada

Para desarrollar un sistema que modele el comportamiento descrito anteriormente se ha empleado el patrón ***Composite***. Este patrón permite construir objetos complejos a partir de otros más simples gracias a la composición y una estructura en forma de árbol. A continuación se muestra un esquema donde se puede contemplar las relaciones entre cada uno de los componentes.

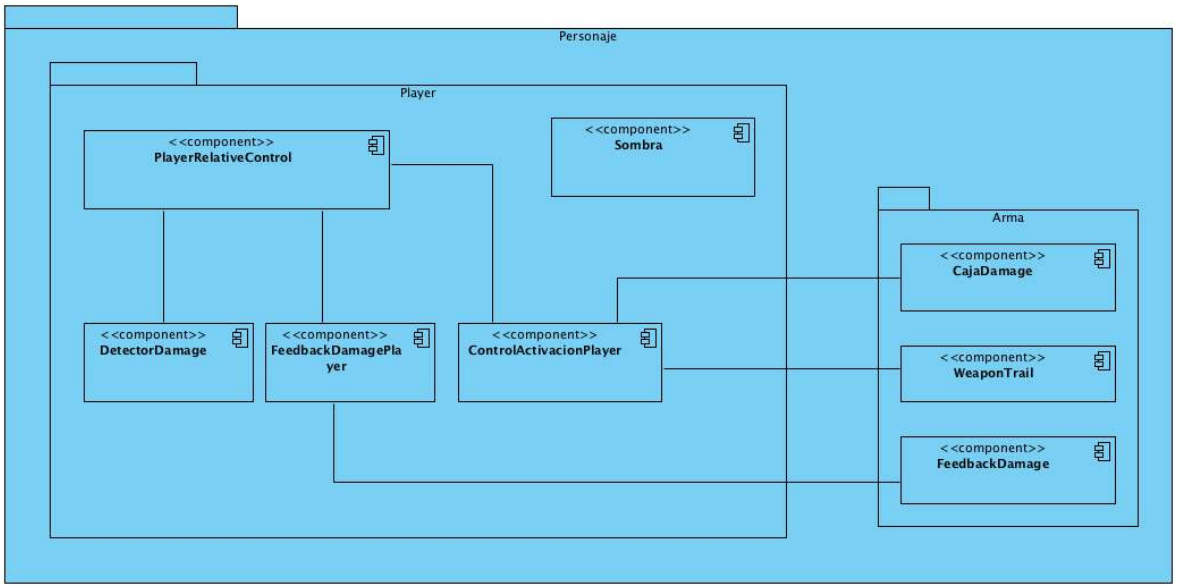


Figura 5.126: Diagrama de componentes del sistema del personaje

Comportamiento general del sistema

El comportamiento del sistema se actualiza en cada pasada del bucle del juego. El siguiente diagrama describe el flujo de acciones que se llevan a cabo para modelar el comportamiento del jugador.

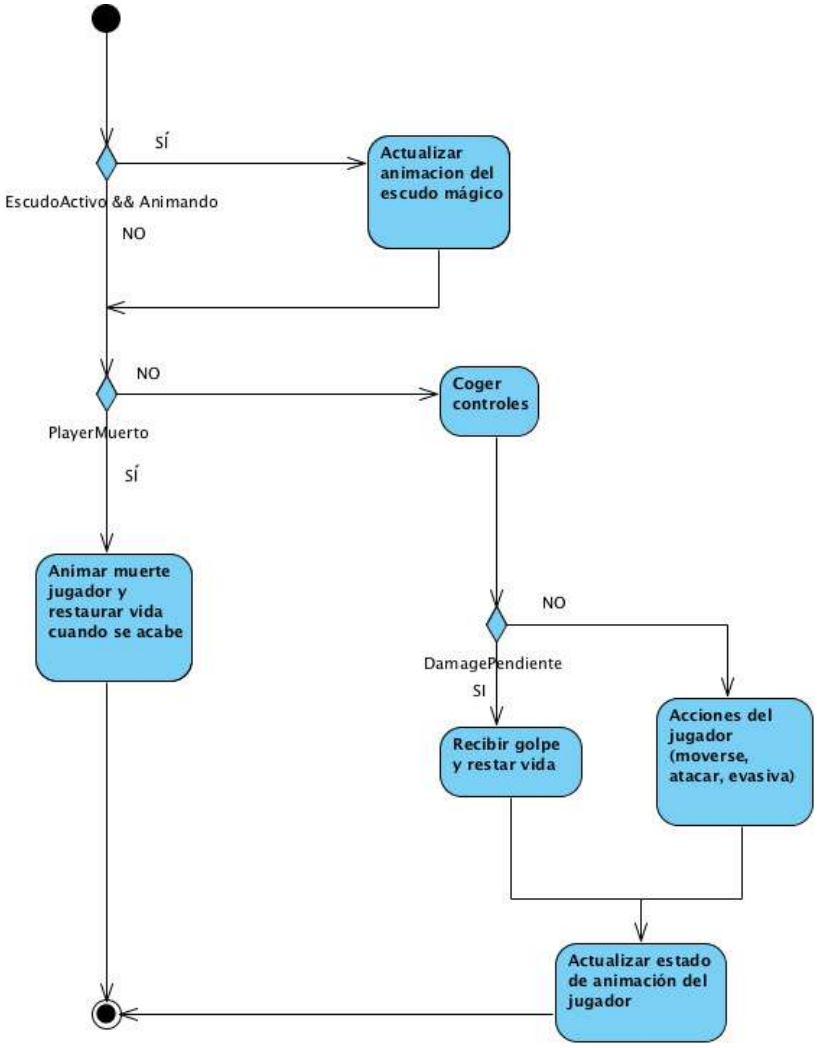


Figura 5.127: Diagrama de actividad del sistema del personaje

Paso de mensajes

Para realizar las diferentes acciones del personaje se realiza una comunicación entre los diferentes componentes, logrando en el proceso que el sistema funcione globalmente. A continuación se muestra la secuencia de mensajes entre los diferentes componentes para algunas de las acciones :

- **Golpear :** El jugador inicia la acción de realizar un movimiento de ataque.

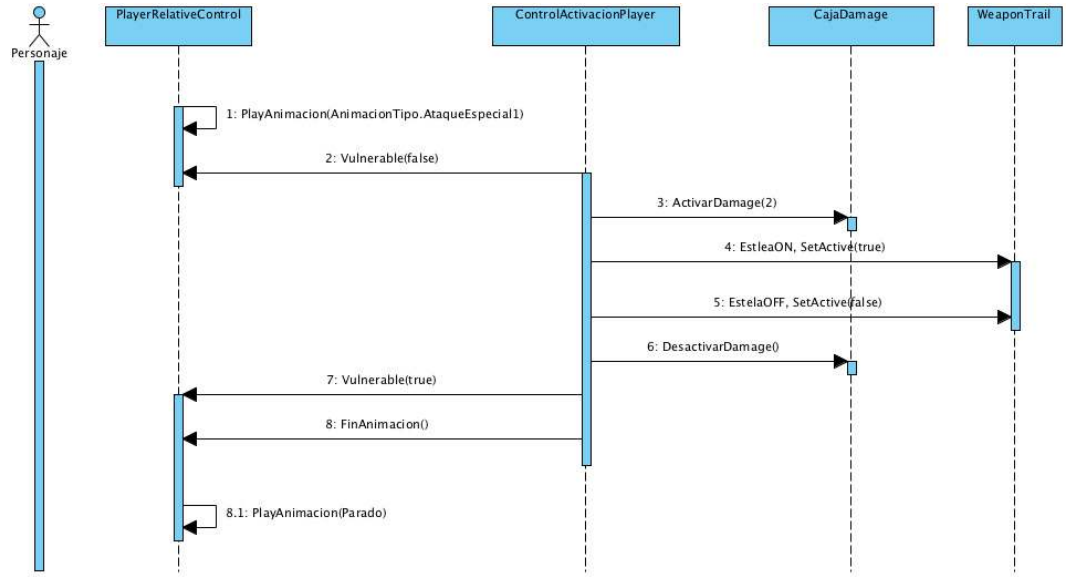


Figura 5.128: Diagrama de secuencia para la acción dar golpe

- **Recibir golpe :** El jugador recibe un golpe por parte de un enemigo.

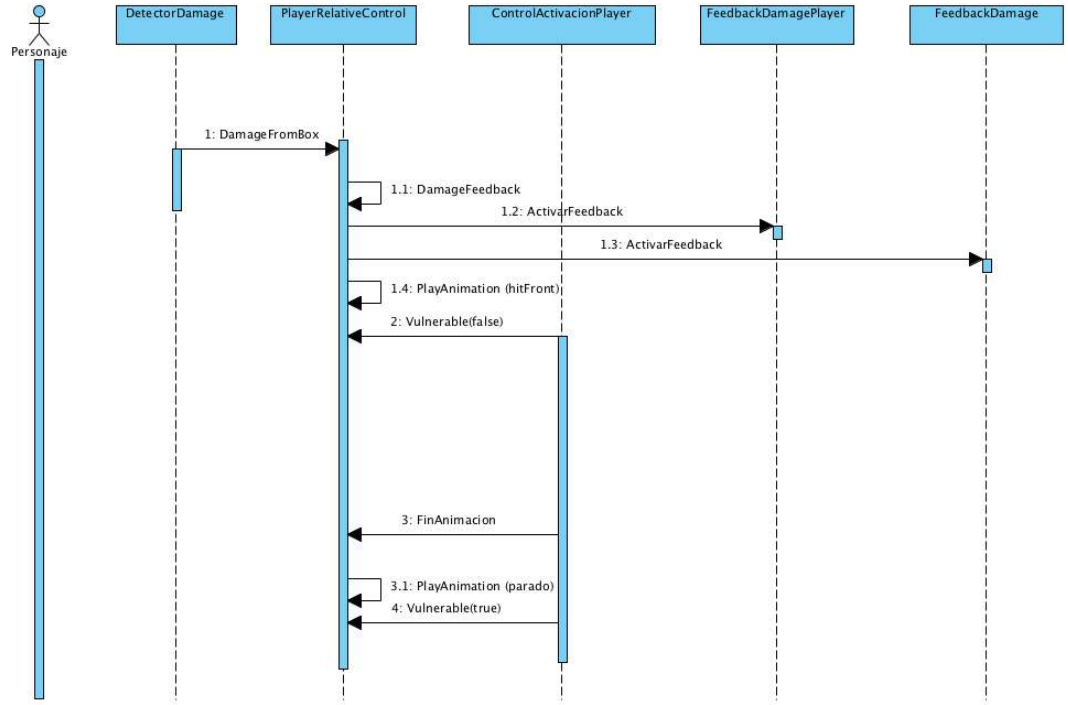


Figura 5.129: Diagrama de secuencia para el evento recibir golpe

Autómata de animaciones del personaje principal

Para modelar el sistema de animaciones del personaje se ha implementado un autómata finito determinista. En función del estado en el que se encuentre el sistema de animaciones se realizará la interpolación para el *frame* de animación concreto. Además este sistema permite disparar eventos en las transiciones, pudiendo notificar a otros componentes de que se ha iniciado o parado una animación para que actúen en consecuencia.

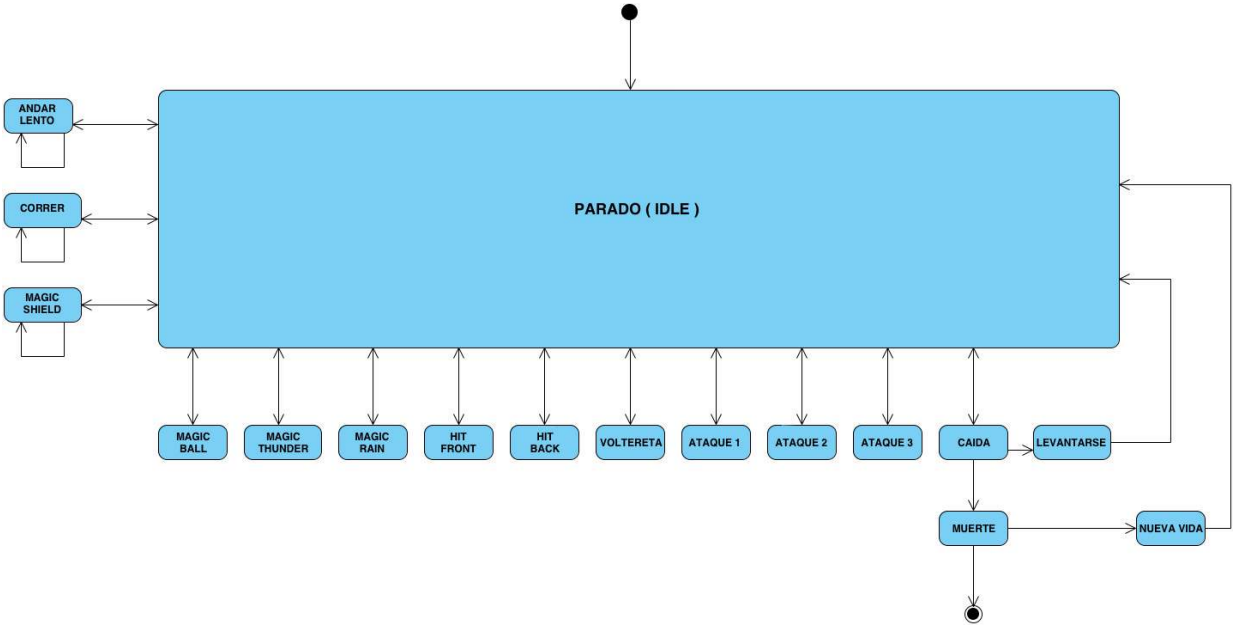


Figura 5.130: Autómata finito determinista que define el comportamiento del sistema de animaciones del personaje

5.2.8. Gestión de grupos y enemigos

Problemática

Tal como se ha comentado en los apartados anteriores el nivel se subdivide en fases. En estas fases se disponen una serie de grupos de enemigos que atacarán al jugador. El comportamiento de los enemigos debe ser coherente con el comportamiento del grupo para gestionar así la curva de dificultad de una manera aceptable para la jugabilidad.

De este modo en los primeros niveles, aunque salgan en pantalla un grupo de 8 enemigos no todos atacarán al mismo tiempo al personaje, sino que sólo un subconjunto de ellos

realizarán ataques.

Este tipo de comportamiento pone de manifiesto no solo la necesidad de un sistema de enemigos que los gestione de manera individual, sino también el disponer de un sistema que permita organizarlos en forma de grupo para que se comporten coherentemente.

Análisis del comportamiento del sistema

Los enemigos se organizan en grupos. Cada grupo de enemigos lleva asociados un número de enemigos a su cargo y un valor entero que determina la cantidad de enemigos en estado de ataque.

El grupo es el encargado de comprobar cuántos enemigos están en estado de ataque hacia el jugador en un momento dado y se encargará de parar o activar los que se encuentren más cercanos al jugador.

Los enemigos disponen de diferentes grados de velocidad y su comportamiento básico es quedarse esperando o entrar en estado de ataque persiguiendo al jugador hasta que entran en una distancia a la cual realizar un ataque. Entre ataque y ataque hay un tiempo que puede ser configurado. En el videojuego disponemos de los siguientes tipos de enemigos :

- ***Zombie*** **:** Te persigue y realiza ataques cuerpo a cuerpo. Pueden eliminarse con el ataque horizontal, el ataque vertical o los ataques mágicos. Su velocidad puede ser lenta, normal o rápida.
- ***ZombieCrawler*** **:** Te persigue y realiza ataques cuerpo a cuerpo desde el suelo. Para eliminarlos es necesario utilizar el ataque vertical o los ataques mágicos. Su velocidad puede ser lenta o normal
- ***ZombieFuego*** **:** Te persigue y realiza ataques cuerpo a cuerpo. Cuando muere explota quitando vida si el personaje se encuentra demasiado cerca. Pueden eliminarse con el ataque horizontal, el ataque vertical o los ataques mágicos. Su velocidad puede ser lenta, normal o rápida.
- ***ZombieEscupe*** **:** Te persigue y realiza ataques de vómito a media distancia. Si entras en un radio cercano también realiza ataques cuerpo a cuerpo. Pueden eliminarse con el ataque horizontal, el ataque vertical o los ataques mágicos. Su velocidad puede ser

lenta, normal o rápida.

Figura 5.131: Tipos de enemigos 1 de 2

- ***Spider*** **:** Te persigue y realiza ataques cuerpo a cuerpo desde el suelo. Para eliminarlos es necesario utilizar el ataque vertical o los ataques mágicos. Su velocidad puede ser lenta o normal.

- ***Perro*** **:** Te persigue y realiza ataques cuerpo a cuerpo desde el suelo. Para eliminarlos es necesario utilizar el ataque vertical o los ataques mágicos. Su velocidad es rápida.

- ***MarinePistola*** **:** Te persigue y realiza ataques de disparos a larga distancia. Si entras en un radio cercano también realiza ataques cuerpo a cuerpo. Pueden eliminarse con el ataque horizontal, el ataque vertical o los ataques mágicos. Su velocidad puede ser normal o rápida.

- ***MarineGranada*** **:** Te persigue y realiza ataques de disparo de granadas a larga distancia. Si entras en un radio cercano también realiza ataques cuerpo a cuerpo. Pueden eliminarse con el ataque horizontal, el ataque vertical o los ataques mágicos. Su velocidad puede ser normal o rápida.

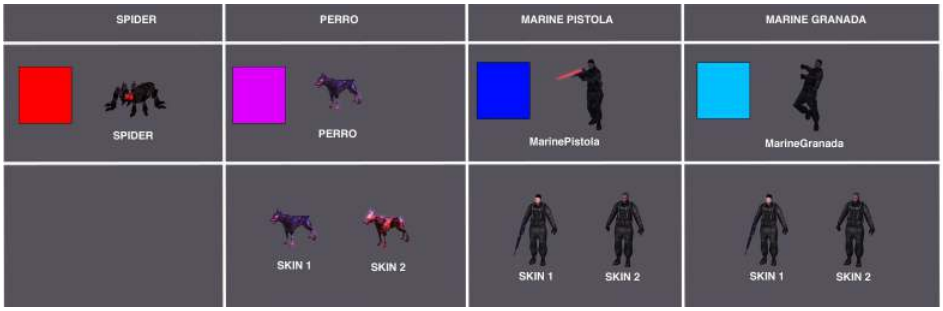


Figura 5.132: Tipos de enemigos 2 de 2

Cada tipo de enemigo tiene diferentes variantes y a su vez cada variante puede tener diferentes *skins*. Esto influye directamente en el gestor de instancias de enemigos que permite componer enemigos diferentes en apariencia con la misma instancia.

Solución tecnológica empleada para grupos

El comportamiento de una grupo se ha implementado utilizando una lista de enemigos y un conjunto de valores de entrada conforme a los que el gestor toma decisiones. Este componente toma las decisiones de cómo gestionar los enemigos y sobre qué eventos disparar. Estas decisiones se toman en base a los siguientes parámetros de entrada :

- Conjunto de *placeholders* de enemigos (Lista) : Se trata de el conjunto de enemigos pertenecientes a ese grupo. Para ver más información sobre cómo se instancian y gestionan hay que consultar la información sobre el sistema de Gestión de creación de enemigos.
- Número de enemigos simultáneos que atacan (Entero) : Este valor permite determinar cuántos enemigos estarán en estado de ataque de manera simultánea para ese grupo.
- Aumentar fase al derrotar a todos los enemigos del grupo (*Booleano*) : Si se activa, el grupo invocará el evento de aumentar fase cuando se derroten todos los enemigos de ese grupo.
- Invocar efecto de *slow motion* al derrotar a todos los enemigos del grupo (*Booleano*) : Si se activa, el grupo activará el efecto *slow motion* para visualizar una cámara cercana que cubra el plano de la acción.
- Acercar enemigos al jugador si no son visibles (*Booleano*) : Si se activa el grupo acercará a los límites del *scroll* percibidos por el jugador los enemigos que están muy lejos.

Esta acción solo se llevará a cabo si no hay enemigos visibles en pantalla. Contribuye a aumentar el ritmo de juego, de modo que el jugador no tenga que esperar hasta que los enemigos más lentos hagan su aparición por los márgenes de la pantalla.

- Rodaje de cámara de presentación (*Transform*) : Si se activa se invocará un rodaje de cámara justo antes de iniciar el grupo para realizar una presentación de los enemigos del grupo.

- Invocar grupo al finalizar (*Transform*) : Si se enlaza un grupo en este puntero, se invocará el lanzamiento de este otro grupo cuando se finalice con todos los enemigos del grupo actual. Esto sirve para disponer de varios grupos que se instancian de forma secuencial para una misma fase.

Solución tecnológica empleada para enemigos individuales

Para desarrollar un sistema que modele el comportamiento de un enemigo individual se ha empleado el patrón ***Composite***. Este patrón permite construir objetos complejos a partir de otros más simples gracias a la composición y una estructura en forma de árbol. A continuación se muestra un esquema donde se puede contemplar las relaciones entre cada uno de los componentes.

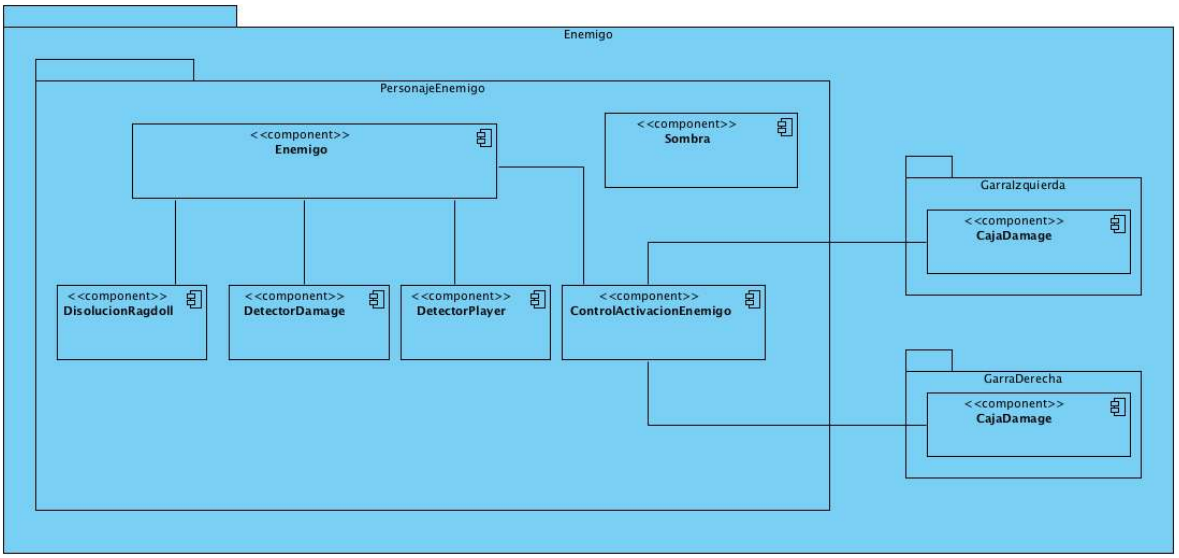


Figura 5.133: Diagrama de componentes del sistema del enemigo

Comportamiento general del sistema

El comportamiento del sistema se actualiza en cada pasada del bucle del juego. El siguiente esquema describe el flujo de acciones que se llevan a cabo para modelar el comportamiento del enemigo.

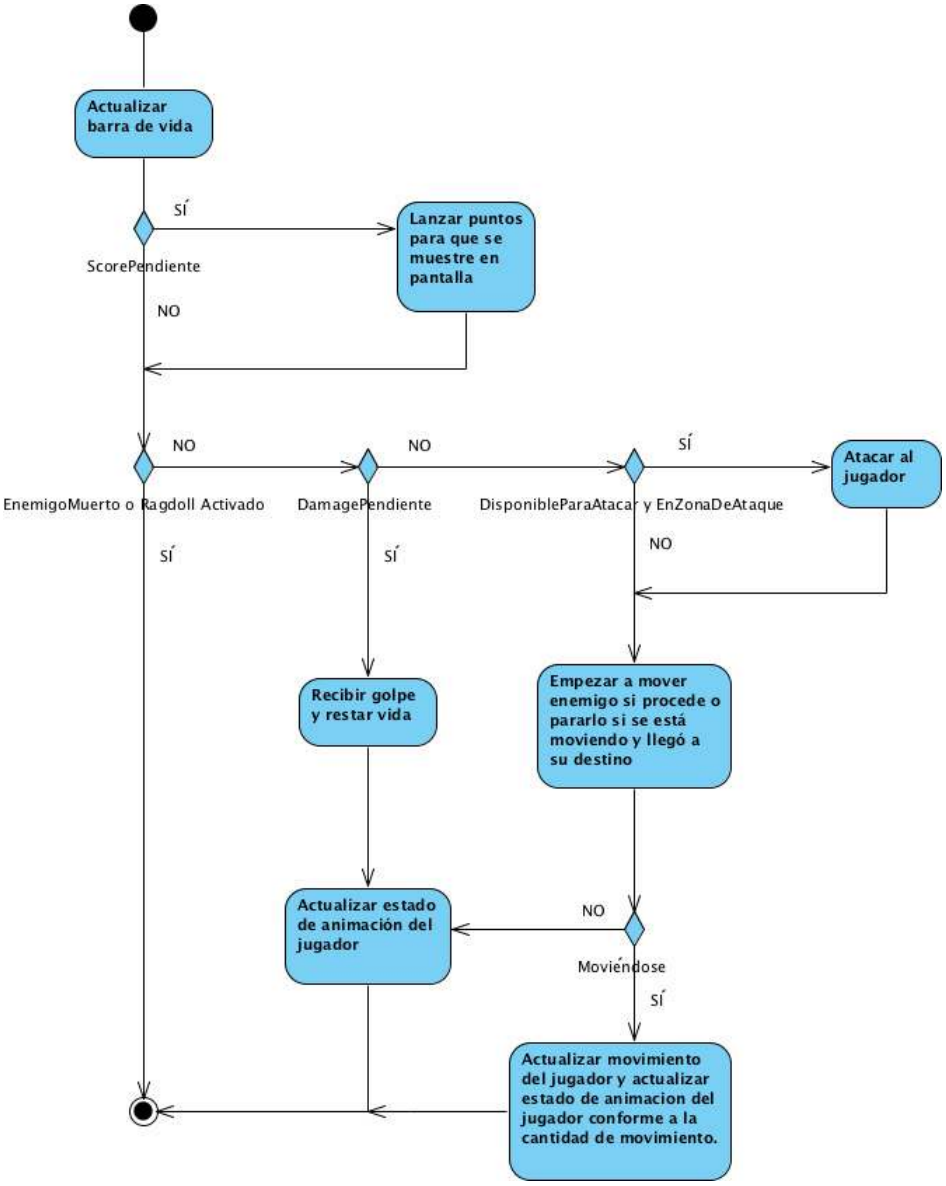


Figura 5.134: Diagrama de actividad del sistema del enemigo

Paso de mensajes

Para realizar las diferentes acciones del enemigo se realiza una comunicación entre los diferentes componentes, logrando en el proceso que el sistema funcione globalmente. A continuación se muestra un ejemplo de la secuencia de mensajes entre los diferentes componentes cuando el enemigo recibe un golpeo por parte del jugador.

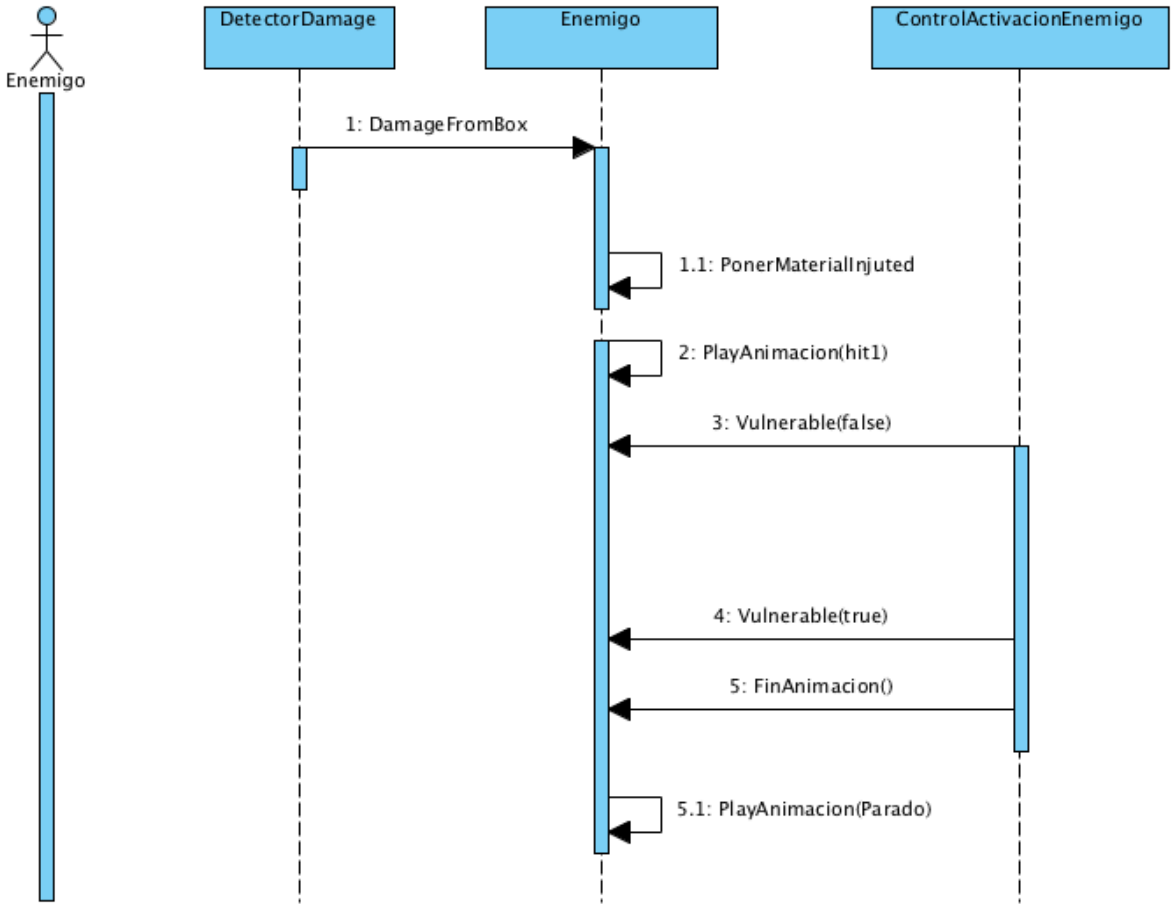


Figura 5.135: Diagrama de secuencia para el evento recibir golpe

Autómata de animaciones del personaje enemigo

Para modelar el sistema de animaciones del enemigo se ha implementado mediante un autómata finito determinista. En función del estado en el que se encuentre el sistema de animaciones se realizará la interpolación para el *frame* de animación concreto. Además este

sistema permite disparar eventos en las transiciones, pudiendo notificar a otros componentes de que se ha iniciado o parado una animación para que actúen en consecuencia.

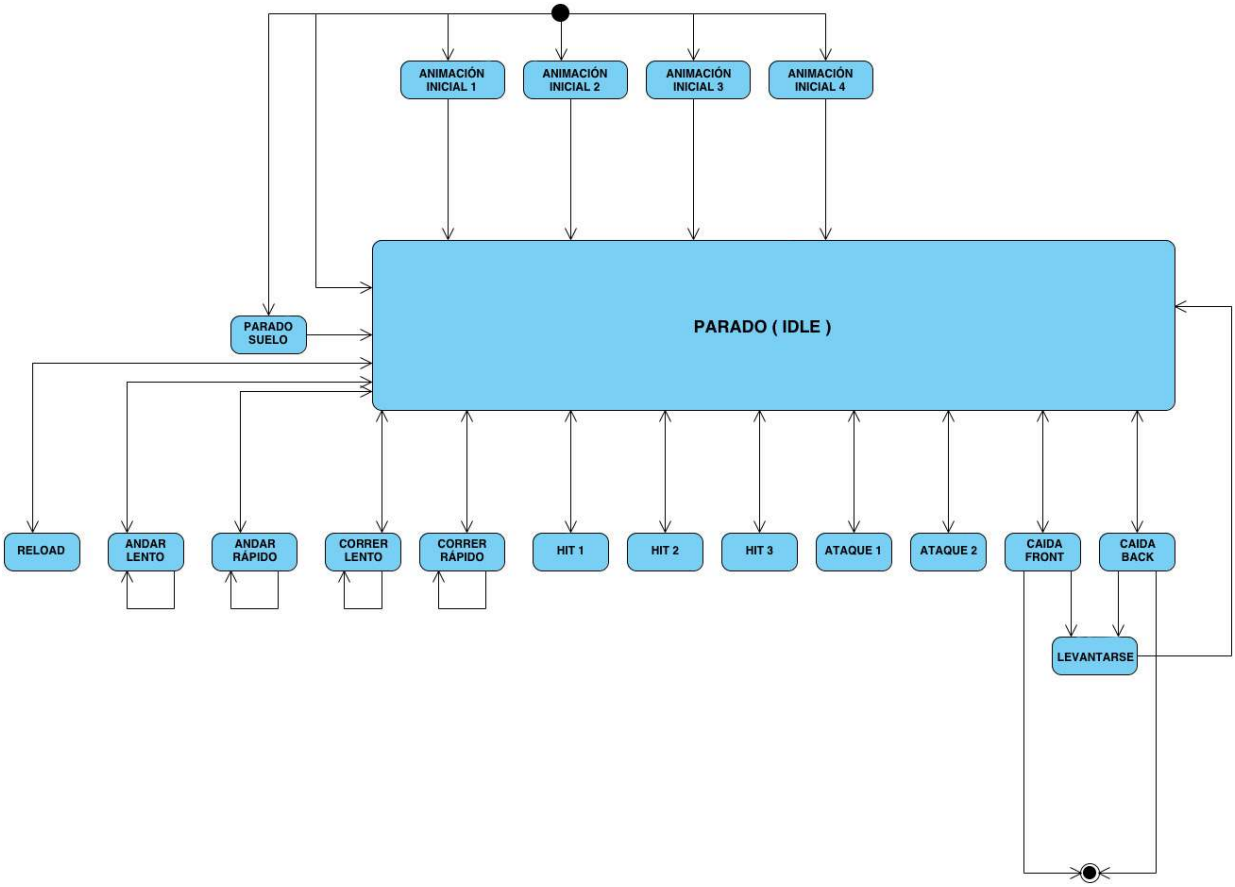


Figura 5.136: Autómata finito determinista que define el comportamiento del sistema de animaciones del enemigo

5.2.9. Gestión de creación de enemigos

Problemática

En un nivel de un videojuego puede haber más de 200 enemigos. El disponer de todas esas instancias en memoria de forma simultánea no es posible pues se sobrepasarían los límites de 512 *Mb* de *RAM* fijados para que este videojuego permita funcionar en sistemas con una cantidad de memoria acotada.

Una posible solución sería cargar los enemigos según van saliendo y descargarlos de memoria cuando no se utilicen. El problema que presenta esta solución es que esa acción

provoca un relentizamiento del sistema cuando se lleva a cabo.

Queda patente la necesidad de un sistema que permita distribuir sobre el nivel miles de enemigos sin exceder la cantidad de memoria fijada y sin realizar cargas intermedias.

Solución tecnológica empleada

La solución empleada consta de dos partes, los *placeholders* y el sistema de gestión de instancias.

Placeholders : Un *placeholders* es una entidad que desplegaremos sobre la escena del videojuego para determinar que en ese punto debe de posicionarse la instancia de un enemigo. Estas entidades van asociadas a un grupo concreto y permiten al gestor de instancias determinar que tipo de enemigos necesitará cada grupo para un momento específico del flujo de la partida.

Los *placeholders* pueden ser de tantos tipos como diferentes enemigos disponemos. Para modelar el sistema se ha empleado el siguiente diagrama de clases :

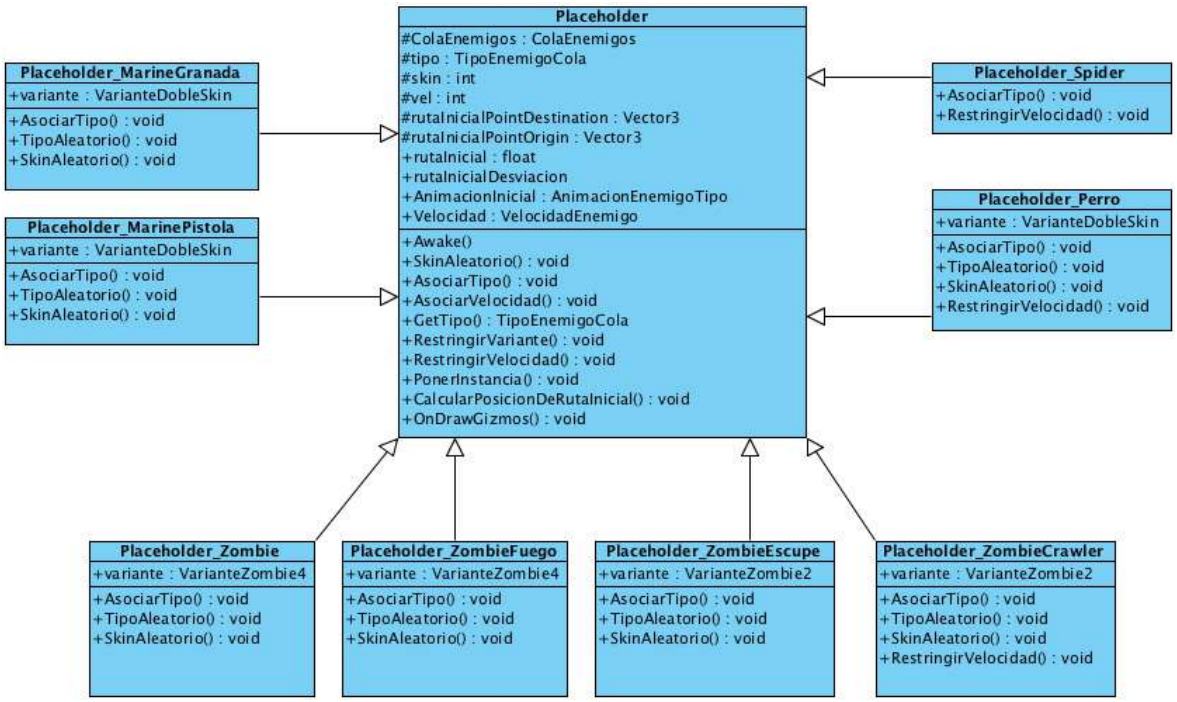


Figura 5.137: Diagrama de clases de los diferentes *placeholders* de enemigos

Los *placeholders* permiten indicarle al sistema de gestión de instancias en qué posición y rotación debe disponerse un tipo de enemigo concreto. Además permiten especificar una ruta de movimiento para el enemigo y una animación inicial de instanciación.

Sistema de gestión de instancias : Esta solución implementa el patrón ***Object Pool***. Es un patrón de diseño de software que usa un conjunto de objetos inicializados preparados para su uso.

El sistema de gestión de instancias realiza el trabajo de creación de enemigos cuando se carga la escena. Para ello recorre todos los grupos del nivel y determina para cada tipo diferente de enemigos cuántas instancias serán necesarias de forma simultánea para permitir jugar la partida.

Después crea una cola por cada tipo de enemigos y mete en ellas el número de instancias mínimo calculado anteriormente para cada tipo de enemigo en función de los *placeholders* encontrados.

Más adelante cuando los grupos demandan *n* cantidad de instancias de diferentes tipos para desplegar en las posiciones de sus *placeholders*, este sistema devuelve instancias mediante una política de gestión *FIFO* en sus colas. Estas instancias se reinician y se cargan con los valores almacenados en cada *placeholder*, dotando al enemigo de los comportamientos y valores necesarios para cada punto de instanciación.

En general, según los datos obtenidos en el videojuego esta técnica permite reducir el número de instancias a su décima parte. Para posicionar 200 enemigos en el nivel, en lugar de 200 instancias sólo es necesario disponer de 20. Esto aporta fluidez al videojuego y además permite su despliegue en plataformas más limitadas en cuanto a cantidad de memoria *RAM* como un *smpartphone*.

5.2.10. Gestión de jefes finales

Problemática

Los jefes finales son un tipo de enemigos a los que el jugador tendrá que enfrentarse en 4 momentos puntuales del videojuego. Cada uno de ellos debe tener un comportamiento muy definido y diferente al anterior. La tarea del jugador es encontrar el punto débil de este tipo de enemigos y explotar su vulnerabilidad hasta acabar con ellos.

Solución tecnológica empleada

El comportamiento individual de cada uno de estos enemigos finales sólo se reproducirá una vez en todo el videojuego para cada uno de ellos. Esto nos lleva a la decisión de optar por una implementación más estática y simple para cada una de estas entidades. Para ello se ha creado un autómata finito determinista que define el comportamiento de cada uno de los jefes finales y se ha implementado de forma más simple en comparación con los enemigos generales.

Comportamiento específico del jefe final 1

Este jefe final se trata de un *zombie* que ha comido demasiado y arrojará una lluvia de vómito cuando le realices un ataque cuerpo a cuerpo.

La forma de eliminarlo es atacando e inmediatamente después realizando una maniobra evasiva hacia otra dirección.

Figura 5.138: Jefe final 1

Este enemigo te perseguirá en todo momento a una velocidad normal y realizará un ataque cuerpo a cuerpo cuando el jugador se encuentre a cierta distancia cercana.

El autómata finito determinista implementado para modelar el comportamiento de esta entidad es el siguiente :

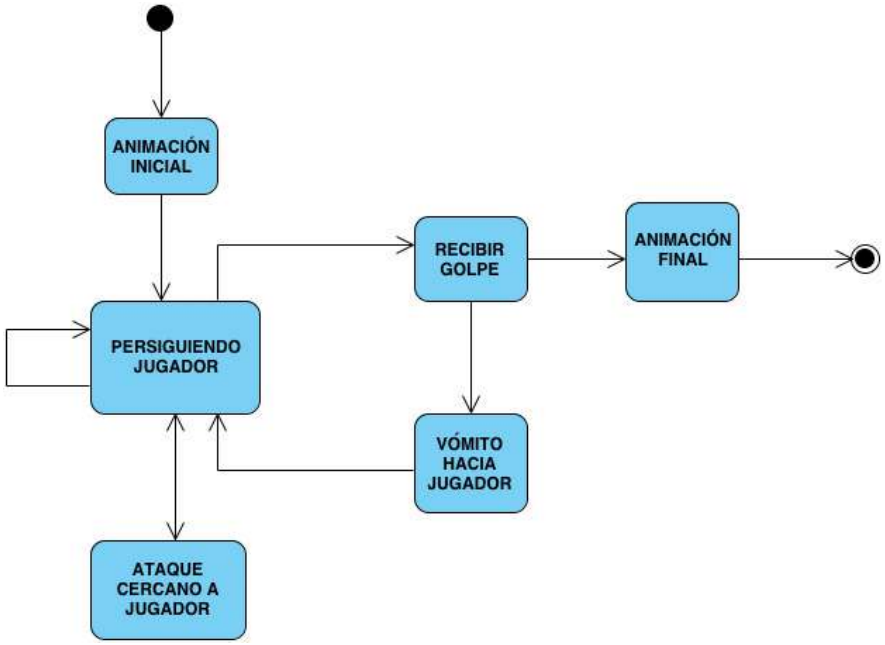


Figura 5.139: Autómata de comportamiento del jefe final 1

Comportamiento específico del jefe final 2

Este jefe final se trata de una araña gigante que pondrá huevos en el escenario. De los huevos saldrán arañas más pequeñas que te perseguirán y atacarán.

Figura 5.140: Jefe final 2

El enemigo sólo es vulnerable mientras pone huevos, y después de ponerlos se esconde por lo que para derrotarle habrá que eliminar todas las crías que salen de los huevos y atacarle cuando salga a poner más huevos.

El autómata finito determinista implementado para modelar el comportamiento de esta entidad es el siguiente :

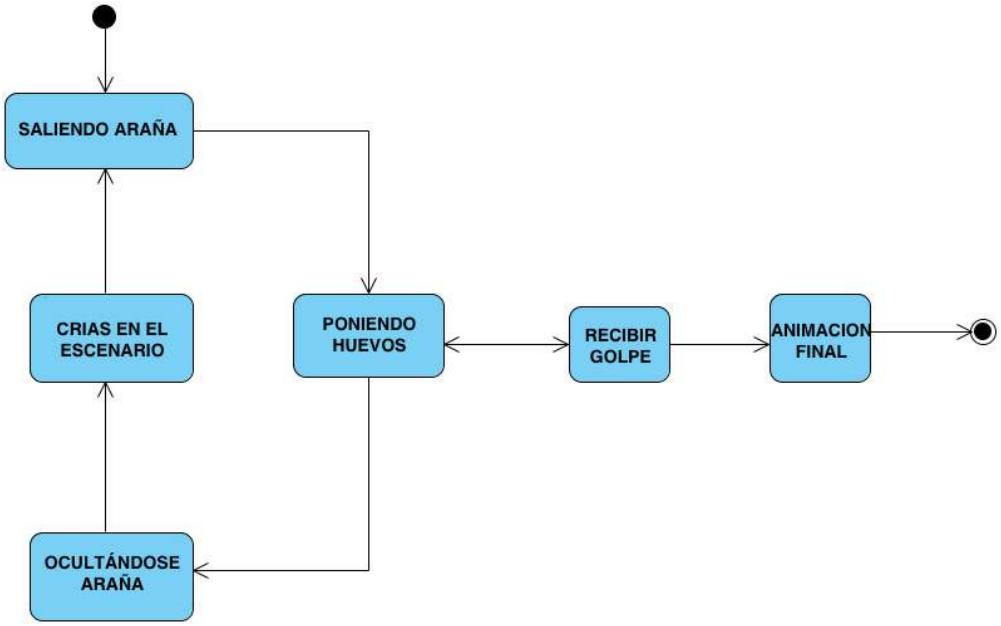


Figura 5.141: Autómata de comportamiento del jefe final 2

Comportamiento específico del jefe final 3

Este jefe final se trata de un marine con una ametralladora. Como única cobertura entre él y nosotros tendremos una serie de columnas detrás de las que podremos escondernos.

No podemos pasar demasiado tiempo detrás de una columna pues si esta recibe muchas balas seguidas se calentará y se derrumbará quedando al descubierto nuestra cobertura.

Figura 5.142: Jefe final 3

El modo de derrotar a este enemigo es alternar la cobertura entre las diferentes columnas hasta que se quede sin balas y deba recargar su arma. En este momento el jugador deberá controlar al personaje para que salga de detrás de las columnas y acercarse a realizar un ataque sobre el enemigo. Tras realizar este ataque el enemigo empezará a recargar más rápidamente por lo que habrá que volver a cubrirse y esperar que vuelva a quedarse sin balas.

El autómata finito determinista implementado para modelar el comportamiento de esta entidad es el siguiente :

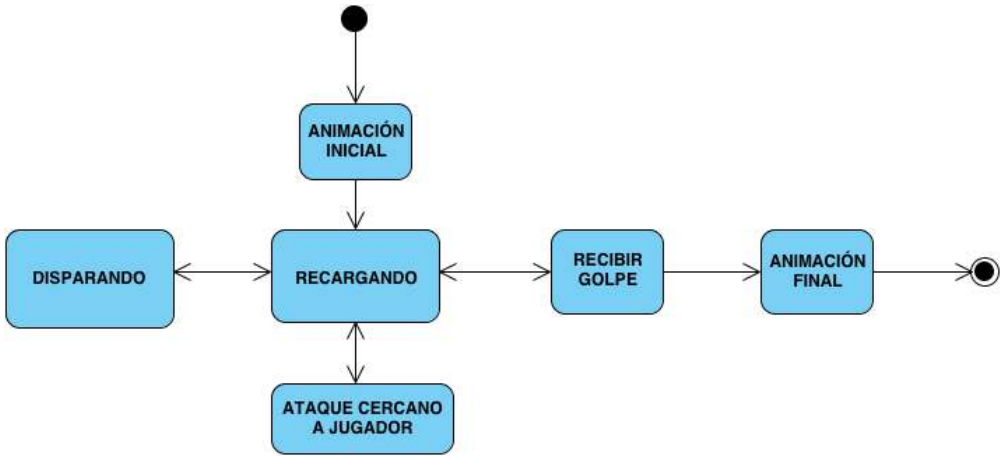


Figura 5.143: Autómata de comportamiento del jefe final 3

Comportamiento específico del jefe final 4

Este jefe final se trata de una mutación gigante de *zombie* que nos escupe fuego y nos golpea con uno de sus brazos.

Figura 5.144: Jefe final 4

Para derrotarle debemos esquivar el fuego que nos lanza en tres ocasiones, después realizará un ataque con su brazo que también debemos esquivar y eso nos dará una oportunidad de atacar el brazo mientras trata de sacarlo del suelo donde se le ha quedado encajado. Tras sacar el brazo vuelve a empezar el ciclo de fuego y debemos de proceder de la misma forma hasta que agotemos toda la vida del enemigo.

El autómata finito determinista implementado para modelar el comportamiento de esta entidad es el siguiente :

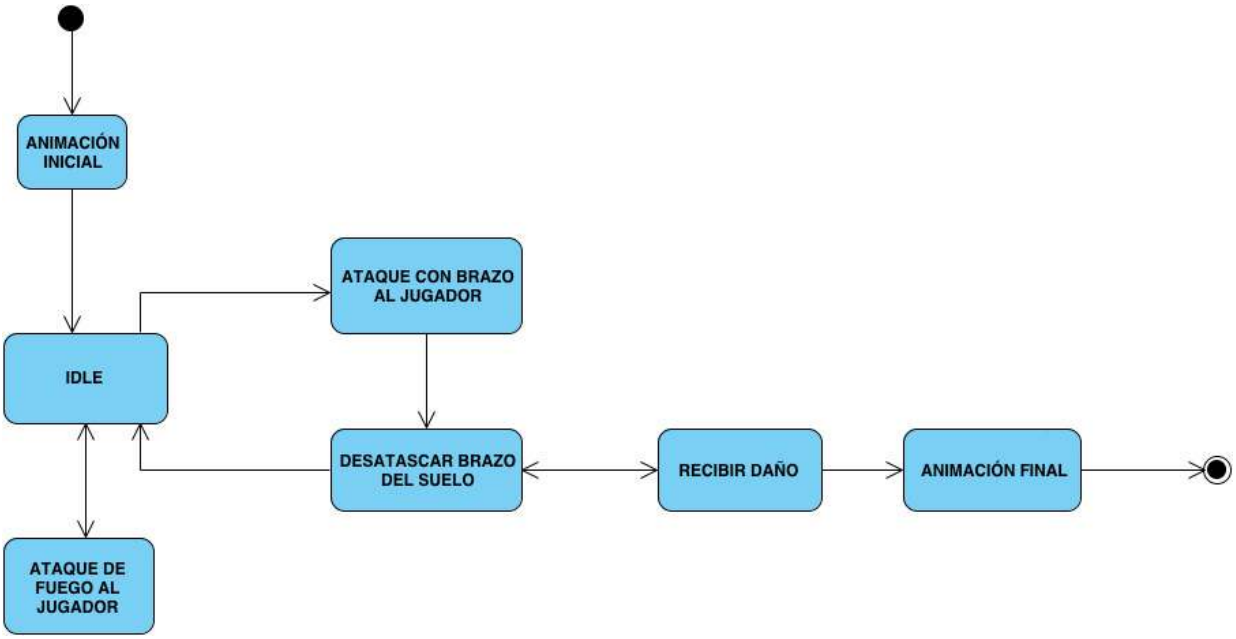


Figura 5.145: Autómata de comportamiento del jefe final 4

5.2.11. Gestión de partículas e ítems

Problemática

En un videojuego se reproducen una serie de efectos de partículas y además se dispone de una serie de objetos de juego como balas, botiquines, monedas o diferentes *power ups*. Uno de los problemas es que estos elementos deben disponerse en pantalla de una forma acotada, es decir si hay varios enemigos y todos están generando explosiones, no pueden pintarse en pantalla de forma simultánea 20 de estos efectos. El motivo es que para ese *frame* concreto el rendimiento se resentirá y el jugador experimentaría ralentizaciones.

Otro de los problemas es la carga de dichos efectos. Si se realiza bajo demanda, el rendimiento del juego dependerá de que estos efectos se invoquen o no. Sufriendo altibajos en la renderización simultánea de tantos efectos.

Debido a estos problemas surge la necesidad de un sistema que permita desplegar efectos sin realizar cargas en el momento del despliegue y que además lleve un control de cuántos de estos efectos se disponen de forma simultánea.

Solución tecnológica empleada

Esta solución implementa el patrón ***Object Pool***. Es un patrón de diseño de software que usa un conjunto de objetos inicializados preparados para su uso.

El sistema de gestión de partículas e *items* realiza el trabajo de creación de estas entidades cuando se carga la escena. Después crea una cola por cada tipo de enemigos y mete en ellas el número de instancias máximo de efectos simultáneos fijado para ese efecto concreto.

Más adelante cuando las entidades del videojuego demandan el despliegue de un efecto o *item*, el sistema devuelve una instancia para desplegar en la posición requerida. Este sistema emplea una política *FIFO* en sus colas.

Por otro lado se cuenta con un sistema auxiliar que emplea patrón ***Facade***. Este sistema sirve de nexo entre las entidades del videojuego y el sistema de gestión de partículas. De ese modo es posible centralizar todas las llamadas de creación de efectos para tener una interface más sencilla y transparente al uso de esta técnica.

Los tipos de colas que dispone el sistema son las siguientes :

- *ExplosionMoco*
- *ProyectilEnemigoMoco*
- *FlashBang*
- *FuegoMuerte*
- *VenenoMuerte*
- *CogerMoneda*

- *CogerMagic*
- *CogerVida*
- *CogerMedikit*
- *FireExplosion*
- *PolvoSueloGrande*
- *ExplosionAire*
- *PolvoSuelo*
- *PolvoTecho*
- *AcidBomb*
- *FirePrenderSmall*
- *FirePrenderMedium*
- *PisadaGordo*
- *ChorroSangreAmarillo*
- *ChorroSangreRojo*
- *ExplosionNido*
- *ExplosionNormal*
- *ExplosionSpider*
- *ExplosionVenom*
- *FuegoProyectil*
- *GolpeConcrete*
- *GolpeDirt*
- *GolpeMadera*
- *GolpeMetal*

- *GolpeSueloAmarillo*
- *GolpeSueloAzul*
- *GolpeSueloRojo*
- *GolpeWater*
- *HondaCaidaPrefab*
- *MuerteP1*
- *MuerteP2*
- *MuerteP3*
- *NuevaVidaAmarillo*
- *NuevaVidaAzul*
- *NuevaVidaRojo*
- *Punch*
- *PunchJugador*
- *PunchText*
- *Sangre1*
- *Sangre1Amarilla*
- *Sangre2*
- *SangreExplosion*
- *SangreExplosionAmarillo*
- *SangreExplosionAmarilloNOSUELO*
- *SangreExplosionFuego*
- *SangreExplosionFuegoNOSUELO*
- *SangreExplosionNOSUELO*

- *DestelloFuerteAzul*
- *DestelloFuerteRojo*
- *DestelloFuerteAmarillo*
- *DisparoMagia1Roja*
- *DisparoMagia2Amarillo*
- *DisparoMagia3Azul*
- *DisparoMagiaFuerte1Roja*
- *DisparoMagiaFuerte2Amarillo*
- *DisparoMagiaFuerte3Azul*
- *GolpeEscenarioAmarillo*
- *GolpeEscenarioAzul*
- *GolpeEscenarioRojo*
- *ImpactoMagia1Roja*
- *ImpactoMagia2Amarilla*
- *ImpactoMagia3Azul*
- *MagiaDeFuego1Rojo*
- *MagiaDeFuego2Amarillo*
- *MagiaDeFuego3Azul*
- *MagiaFallo1Rojo*
- *MagiaFallo2Amarillo*
- *MagiaFallo3Azul*
- *Combo1*
- *Combo2*

- *Combo3*
- *Combo4*
- *Score120*
- *Score160*
- *Score180*
- *Score20*
- *Score240*
- *Score40*
- *Score60*
- *Score80*
- *Granada*
- *Magic*
- *Medikit*
- *Moneda1*
- *Moneda2*
- *Moneda3*
- *ProyectilEnemigoSuper*
- *Vida*
- *RoturaSuelo*

5.2.12. Gestión de construcción procedural de niveles

Problemática

Solución tecnológica empleada

Esta solución implementa el patrón ***Object Pool***. Es un patrón de diseño de software que usa un conjunto de objetos inicializados preparados para su uso.

El sistema de gestión de construción de niveles procedurales realiza el trabajo de creación de bloques reutilizables cuando se carga la escena. Después crea una cola por cada tipo de bloque y mete en ellas el número de instancias mínimo para satisfacer el uso simultáneo de estos bloques pre construidos sin que el usuario perciba cuándo están siendo usados.

Por otro lado se tienen una serie de bloques simples sustitutivos que están dispuestos en el nivel a modo de representantes de estas entidades más complejas. A medida que el usuario avanza por el nivel, cuando el sistema detecta que necesita renderizar uno de estos bloques de sustitución, realiza una llamada al sistema de construcción procedural de niveles y le pide el bloque complejo asociado para ponerlo en ese punto del escenario.

Dicho bloque complejo permanecerá allí, hasta que sea demandado por el sistema para colocarlo en otro punto. Para ello se emplea una política *FIFO* en sus colas. De este modo el objeto colocado no se volverá a mover a otro punto hasta que todos los anteriores de la cola lo hayan hecho antes que él.

El uso de esta técnica permitió que los niveles de vehículos pudieran ajustarse a las limitaciones de memoria de 512 *Mb* de *RAM*. En la primera versión del prototipo sin el uso de esta técnica, el nivel de la motocicleta ocupaba en memoria 1.5*Gb* de *RAM*. Después de implementar este sistema y adaptar el nivel para su uso, pasó a ocupar en memoria 400 *Mb* de *RAM*. Esta optimización permite el despliegue de ese tipo de niveles en plataformas móviles.

Los tipos de colas que dispone el sistema son las siguientes :

- *Carteles luminosos Decorativos 1 - 25*
- *ArbolRompible*
- *AutopistaBasura1*

- *AutopistaBasura2*
- *AutopistaCoche1*
- *AutopistaCoche2*
- *AutopistaCoche3*
- *AutopistaCoche4*
- *AutopistaCuboBasura*
- *CasaHexagonal*
- *CasaSaliente1*
- *CasaSaliente3*
- *CasaEsquina*
- *ContenedorMetal*
- *Edificio1*
- *Edificio2*
- *Edificio3*
- *Edificio4*
- *EdificioSaliente1*
- *EdificioSaliente2*
- *Gasolinera1*
- *Gasolinera2*
- *Luz*
- *Macetero*
- *Magma*
- *Monedas*

- *Torre1*
- *Torre2*
- *ZonaVeneno*

5.2.13. Gestión de HUD

Problemática

El principal problema es que el sistema de *GUI* que proporciona *Unity3D* es muy lento. Surge la necesidad de crear un sistema que proporcione un rendimiento adecuado en el renderizado. Además es necesario estructurar dicho sistema para que proporcione una interface que permita alterar el *HUD* desde el resto de actores del videojuego de una manera centralizada y sencilla.

Solución tecnológica empleada

Para resolver el problema se plantea un sistema de *HUD* que emplea el patrón ***Facade***, de este modo el resto de entidades del videojuego pueden interactuar con el sistema de *HUD* de una manera simple.

Por otro lado se ha optado por no utilizar el sistema *GUI* nativo de *Unity3D*. En su lugar se ha desplegado un sistema que emplea el uso de polígonos con mapeado de texturas y renderización. Estos polígonos se pintan cada *frame* directamente sobre el *framebuffer* justo después de haber realizado el render de la cámara virtual del videojuego. Esta mejora a proporcionado un aumento del rendimiento de entorno al 10 % con respecto a los primeros prototipos que utilizaban el sistema nativo.

5.2.14. Gestión de sonido

Problemática

La gestión del sonido supone otra de las grandes problemáticas de un videojuego. Por un lado cada vez que se utiliza un sonido debe de ser cargado, además cada entidad puede duplicar dicha carga si no se dispone de un sistema gestor que centralice esta tarea.

Por otro lado en dispositivos móviles existe una fuerte limitación del número de sonidos que pueden reproducirse de manera simultánea por hardware. Por este motivo es necesario implementar unas políticas de gestión que permitan acotar el número de efectos simultáneos.

Solución tecnológica empleada

Para resolver el problema propuesto se plantea un sistema de *HUD* que emplea el patrón ***Facade***, de este modo el resto de entidades del videojuego pueden invocar la reproducción de sonidos de una manera simple.

Esta tarea permite centralizar la carga de estos *assets* para realizarla una única vez al principio del nivel. Este sistema también lleva un control de la reproducción de sonidos, de modo que para reproducir un nuevo sonido debe parar los que actualmente se estén reproduciendo si el número es muy elevado. Para ello se emplea una política de uso *FIFO*.

Además de esto, el sistema carga los valores de volumen de sonido y música para aplicarlos a todos los sonidos que gestiona. También debe notificar de un cambio en estos valores a las entidades y sistemas que hagan uso del sonido. Para esta tarea el sistema también emplea el patrón ***Observer***.

5.2.15. Gestión del controlador de E/S

Problemática

La gestión del control en un videojuego es una tarea que requiere un sistema que centralice toda esta comunicación con el *hardware* bajo un mismo sistema común. De esta forma puede ser accesible desde diferentes actores.

Además el sistema debe ser transparente al tipo de control empleado para controlar el videojuego, proporcionando para ello una interface única al resto de entidades. Dicha interface debe permanecer invariable, independientemente de que el control que use el personaje sea un *joystick*, un teclado o una pantalla táctil.

Solución tecnológica empleada

Para resolver esta problemática, en la creación del sistema de control se han empleado los patrones ***Singleton*** y ***Proxy***. De esta forma todas las entidades pueden acceder a la misma instancia del sistema, y además el sistema funcionará como un intermediario con cada uno de los diferentes tipos de control, estando enlazado con uno u otro en función de las preferencias del jugador.

5.2.16. Gestión logros

Problemática

La gestión de logros requiere llevar un control de la estadísticas del videojuego, así como su persistencia. Con el conocimiento de estos datos, cada vez que se produzca un cambio en ellos, el sistema debe comprobar si se ha cumplido alguno de los objetivos secundarios del videojuego.

En caso afirmativo el sistema debe proporcionar el logro adecuado al jugador, guardar la persistencia y notificar por pantalla su consecución.

Solución tecnológica empleada

Para resolver la implementación de un sistema de estas características se ha empleado el patrón ***Singleton***. La ventaja que proporciona es que todas las entidades que proporcionan estadísticas a este sistema se comunicarán con la misma instancia, evitando problemas de coherencia en la persistencia de información de juego, y teniendo una zona común donde se lleva el control de estadísticas.

Por otro lado, este sistema se apoya en el sistema de Gestión de *HUD* para notificar de manera gráfica mediante una animación en pantalla que se ha conseguido un logro.

5.2.17. Gestión de la física

Problemática

La gestión de las colisiones, físicas complejas y animación de esqueletos basados en física es un problema en sí mismo tan grande como la creación de un motor gráfico desde cero. Por suerte *Unity3D* proporciona una interface de programación para utilizar el motor de físicas *Physics* de *NVidia*.

La problemática concreta en el uso de la física en este videojuego es delegar correctamente cada uno de los cálculos a este motor.

Solución tecnológica empleada

Se ha empleado el uso integrado de esta solución software para las siguientes tareas:

- **Detección de colisiones :** Se ha empleado la detección de colisiones para delimitar las posiciones por las que puede o no pasar el jugador. De este modo acotamos por dónde se puede mover para que no atraviese objetos del entorno. La forma de utilizar este método consiste en añadir a los objetos tridimensionales unos objetos físicos de menor resolución y que son empleados en el cálculo de la simulación física.

- ***Triggers* :** Un *trigger* es una región del espacio que emite un evento cuando un determinado objeto entra o sale de ella. Se han empleado *triggers* por ejemplo para detectar cuando un enemigo recibe un golpe o cuando un personaje llega a una determinada zona.

- ***Ragdolls* :** Cuando un enemigo o personaje muere su cuerpo se anima conforme a la fuerza del impacto que ha recibido. Esto se realiza de forma dinámica mediante el sistema de *ragdolls* que proporciona el motor de física.

- ***Layers* :** Para gestionar correctamente todo el uso de colisiones, *ragdolls*, *triggers* y demás componentes se ha empleado el uso de diferentes capas de cálculo físico. De este modo unas colisiones no intervienen con otras, minimizando así el número de cálculos a realizar.

5.2.18. Renderización y efectos gráficos

Problemática

El videojuego debe de poder renderizarse en multitud de dispositivos de *hardware*, incluyendo plataformas móviles y debe de proporcionar unos efectos y técnicas de renderización compatibles con cada una de estas plataformas.

Solución tecnológica empleada

Para resolver esta tarea se han empleado diversas técnicas de optimización y adaptación del renderizado para hacerlo compatible con todas las plataformas soportadas. Esto se discute a continuación en las siguientes subsecciones.

Por otro lado se ha creado un sistema que permite gestionar las diferentes calidades gráficas de todos los efectos y entidades del videojuego. Para ello se emplea un sistema que hace uso del patrón ***Observer***. De esta forma puede realizarse un cambio de calidad de ajustes gráficos en tiempo de ejecución que permite propagar los cambios a cada entidad dependiente de este sistema. Después cada entidad es responsable de ajustarse a esos valores

de calidad gráfica para ofrecer una renderización adecuada a diverso tipo de plataformas y configuraciones de *hardware*.

Iluminación y sombras

En el videojuego se han empleado cálculos de sombras en tiempo real. *Unity3D* proporciona un sistema de sombras, pero las que se requerían para este videojuego necesitaban emplear el modo de renderizado *Deferred Rendering*.

Como esta técnica de renderizado no es posible aplicarla en algunos dispositivos móviles se optó por implementar un sistema de sombras basado en proyectores y *render* a textura que fuera compatible con el modo de renderizado *Forward Rendering*. Este modo permite su despliegue de forma optima en plataformas móviles, además de ofrecer un mejor rendimiento en la versión de escritorio.

Reflexiones y agua

En el videojuego se han empleado efectos de reflexiones en tiempo real. *Unity3D* ya proporciona estos efectos de serie para su uso. No obstante no son compatibles con dispositivos móviles debido a que están implementados utilizando técnicas y *shaders* pensados para equipos de escritorio.

Para el presente videojuego se realizo una adaptación de dichos efectos para que fuera posible su uso en plataformas móviles y funcionaran con un rendimiento adecuado. Para ello se modificó el código de estos sistemas y se cambiaron los *shaders* empleados en el procesamiento de cada uno de estos efectos de post procesado.

Optimización de sombras y reflexiones

Además de las citadas optimizaciones en cada uno de los sistemas se realizó un sistema de balanceo de carga para los sistemas de reflexión y sombras dinámicas, de modo que permita que estos dos sistemas funcionen de manera simultánea sin sobrecargar demasiado al sistema.

Para ello se emplea una política de gestión de recursos. En un *frame* concreto este componente da permiso de uso de la *cpu* a uno de los sistemas y se lo deniega al otro. Con ello se consigue que para un *frame*, o se actualizan los reflejos, o se actualizan las sombras, pero no

las dos cosas de manera simultánea. El resultado, es que si el videojuego funciona a 60 *FPS*, cada uno de estos sistemas lo hace a 30 *FPS*, realizando su trabajo de cómputo en *frames* alternos.

La adopción de esta política incrementó notablemente el rendimiento del videojuego sin que este efecto sea perceptible para el ojo humano, el cual no es capaz de asimilar más de 24 *FPS*.

Cull Layers

En el desarrollo del videojuego se han empleado diferentes capas de *culling*. El motivo es disponer de diferentes conos de *frustrum* en función del tipo de contenido. Con ello se consigue que en función de la capa empleada para renderizar un determinado objeto *3D* este será o no visible en función de la distancia a la cámara y su capa concreta.

Por ejemplo, los edificios, suelo y estructuras más grandes tienen una distancia de dibujado de 1.0, mientras que los *decals*, objetos de decoración y detalles tienen una distancia de dibujado de 0.5. Eso significa que al renderizar una escena se pintará hasta la máxima distancia de dibujado las estructuras más importantes, mientras que los detalles serán influidos por otro cono de descarte más restrictivo en el proceso de renderizado.

El uso de esta técnica proporciona mayor rendimiento al videojuego y el usuario no notará este efecto debido a la niebla que se pinta en el nivel. En cambio se realizan menos llamadas de dibujado en comparación con toda la geometría que se renderizaría si se compartiera una misma distancia de dibujado máxima.

Efectos de post procesado

En el videojuego se han empleado efectos de post procesado como el *Bloom*, *Depth of Field* o la corrección de color. *Unity3D* ya proporciona estos efectos de serie para su uso. No obstante no son compatibles con dispositivos móviles debido a que están implementados utilizando técnicas y *shaders* pensados para equipos de escritorio.

Para el presente videojuego se realizo una adaptación de dichos efectos para que fuera posible su uso en plataformas móviles y funcionaran con un rendimiento adecuado. Para ello se modificó el código de estos sistemas y se cambiaron los *shaders* empleados en el procesamiento de cada uno de estos efectos de post procesado.

Biblioteca de *shaders*

Gracias a la biblioteca de *shaders* desarrollada se ha alcanzado nivel de calidad gráfica con un rendimiento adecuado. Estos *shaders* permitan su renderización en *pipelines* de tarjetas gráficas punteras pero también en circuitería de móvil.

5.2.19. Resumen de patrones empleados

Los patrones de creación, estructura y comportamiento suponen un estándar en la creación de software. Esto se debe a la probada eficacia y ahorro de tiempo que suponen para el diseño de soluciones software, y además proporcionan un enfoque común y reconocible para que el resto de profesionales pueden entender y extender dicho software si es necesario.

La mayoría de los patrones están pensados para su implementación y despliegue bajo un paradigma de orientación a objetos. No obstante en el presente TFG se ha empleado el paradigma de programación orientado a componentes propuesto por *Unity3D*.

Este cambio de paradigma supone un *handicap*, no obstante se han construido las soluciones de los diferentes sistemas, adaptándolos en la medida de lo posible al uso de patrones.

A continuación se ofrece una lista resumen de los patrones que se han empleado :

- ***Facade*** **:** Proporciona una interface unificada para un conjunto de sistemas. Este patrón de proporciona una interface de alto nivel fácil de utilizar.

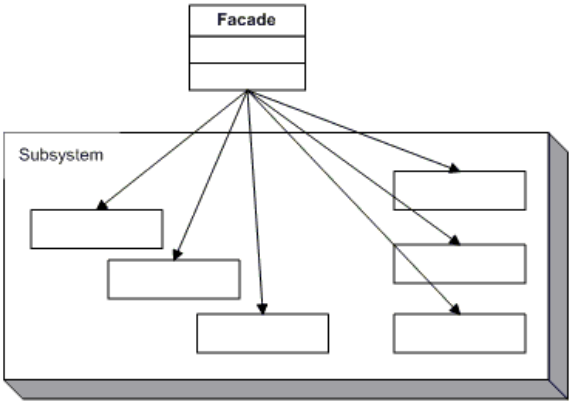


Figura 5.146: Esquema del patrón *Facade* [156]

Empleado en Gestión de componentes, Gestión de partida, Gestión de partículas y Gestión de sonido.

- ***Singleton*** **:** Se asegura que sólo existe una instancia de un determinado sistema y se proporciona un punto de acceso global a él para todas las demás entidades.

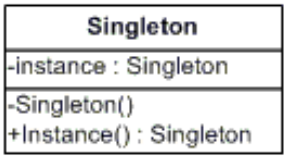


Figura 5.147: Esquema del patrón *Singleton* [156]

Empleado en Gestión de componentes, Gestión de entrada y salida de control y Gestión de logros.

- ***Object Pool*** **:** Es un patrón de diseño de software que usa un conjunto de objetos inicializados preparados para su uso.

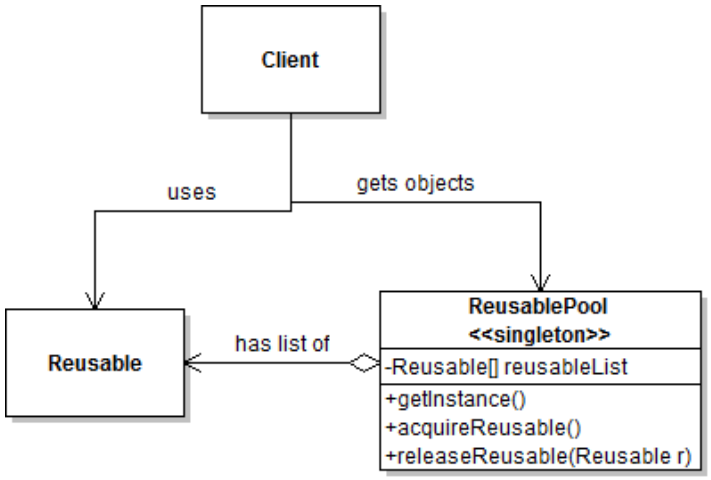


Figura 5.148: Esquema del patrón *Object Pool* [157]

Empleado en Gestión de partículas, Gestión de creación de enemigos y Gestión de construcción procedural de niveles.

- ***Proxy*** **:** Proporciona un representante o sustituto de otro objeto para que los sistemas pueden controlarlo a través de este.

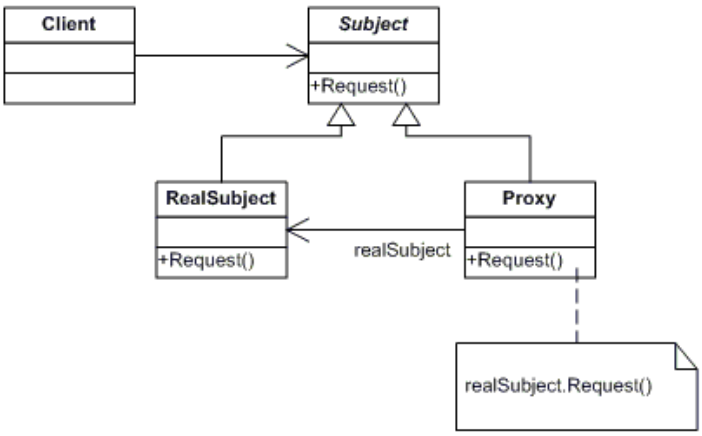


Figura 5.149: Esquema del patrón *Proxy* [156]

Empleado en Gestión de entrada y salida de control.

- ***Observer*** **:** Define una dependencia entre objetos que proporciona la posibilidad de propagar los cambios de estado de un sistema hacia los que son observadores del mismo.

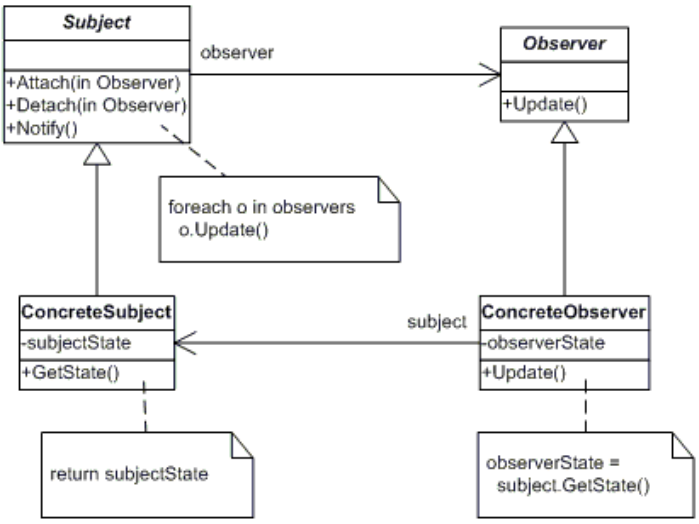


Figura 5.150: Esquema del patrón *Observer* [156]

Empleado en Gestión de sonido y Gestión de ajustes gráficos.

- ***Composite*** **:** Este patrón permite construir objetos complejos a partir de otros más simples gracias a la composición y una estructura en forma de árbol.

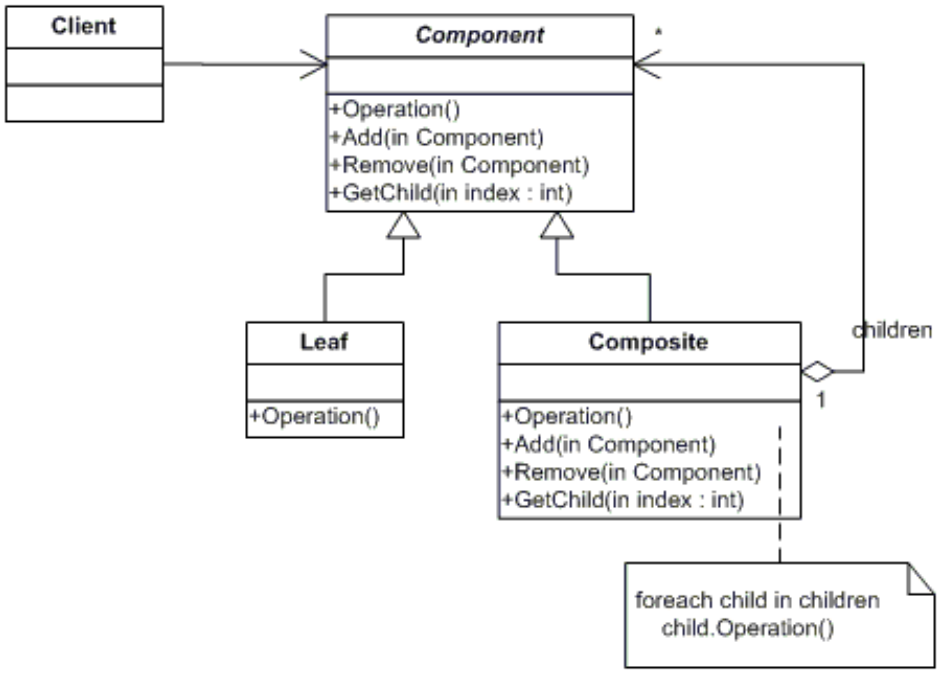


Figura 5.151: Esquema del patrón *Composite* [156]

Empleado en Personaje y Enemigos.

- ***State*** **:** Permite a un objeto alterar su comportamiento cuando su estado interno cambia.

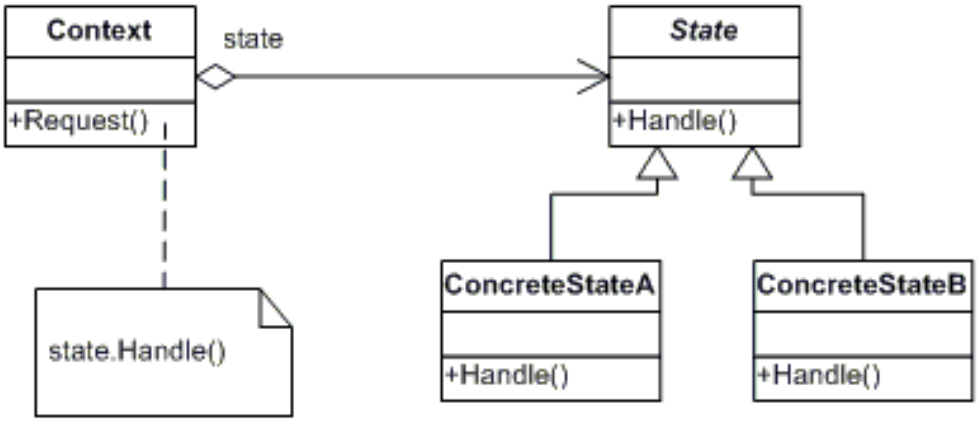


Figura 5.152: Esquema del patrón *State* [156]

Empleado en Jefes finales, Personaje y Enemigos.

5.3. COMERCIALIZACIÓN

A continuación se describe un plan de negocio [48] con un alcance acotado. El plan de negocio recogido en este TFG se ha limitado a los aspectos más esenciales. De otro modo, la extensión y profundidad del mismo requeriría incluso mayor número de páginas que el TFG completo.

5.3.1. Descripción del producto y valor distintivo

Este apartado contiene una explicación detallada del concepto básico y de las características del producto concreto que se presenta en este proyecto.

Las funcionalidades básicas del presente videojuego es la de crear un producto comercial multiplataforma que sea original, que tenga una calidad técnica y artística a la altura de los estándares actuales y que sea de un genero que no esté demasiado saturado a día de hoy. En nuestro caso se trata de un *Beat 'em up* de scroll lateral.

El soporte tecnológico inicial en el cual se pretende ofrecer será en multiples familias familias:

- Ordenadores de escritorio : *Windows* / *GNU/Linux* y *Mac OS X*
- Dispositivos móviles : *Smartphones*, *Tablet* y la consola *OUYA* con sistema operativo *Android*.
- Videoconsolas : Si se consigue llegar a un acuerdo con un *publisher* se podría dar soporte a *Wii U*, *Xbox One* y *PS4*.

El origen de la idea de negocios no es otro que atacar un sector que tiene un crecimiento muy bueno y de el cual se tienen los conocimientos necesarios para tener unas mínimas garantías de éxito en la puesta en marcha de este proyecto.

El público objetivo al que va dirigido sería :

- Usuarios de videojuegos independientes, que valoran ideas originales por encima de la calidad técnica.
- Usuarios de móviles que quieren videojuegos más profundos en comparación con los videojuegos casuales habituales en estas plataformas

- Usuarios que no encuentran videojuegos actuales pero de géneros que disfrutaban antiguamente.

En cuanto al valor único y carácter distintivo del mismo podemos considerar que el presente videojuego tiene un planteamiento original debido al carácter de los protagonistas del mismo, así como su estética original y el estilo de juego rescatado de los géneros clásicos que a día de hoy han quedado un poco abandonados.

5.3.2. Mercado potencial

Este apartado contiene una explicación del estado del mercado actual, así como del público potencial para este tipo de productos.

El tamaño del mercado es muy grande, como se ha justificado según los datos proporcionados en el capítulo 3 de este documento, y en cuanto al crecimiento esperado según determinados estudios [49] marcan una tendencia de crecimiento para 2015 de 158 % respecto a 2010, donde podemos considerarlo como uno de los sectores con mayor crecimiento en los últimos años.

El grado de consolidación es alto, siendo a día de hoy la industria del videojuego una de los sectores más fructíferos, superando en los últimos años a la industria de la música y el cine juntas.

Los factores clave de éxito en este mercado es tener una buena presencia en los medios y contar con el apoyo de los *influencers* más destacados. Esto se conseguirá con buen plan de marketing pero no es suficiente, ya que a su vez el contenido debe ser de calidad para que los usuarios lo valoren como algo original y no como otro producto más.

Las barreras de entrada a este mercado son la necesidad de unos conocimientos muy específicos, no sólo de la parte tecnológica de cómo se desarrolla un videojuego, sino de el funcionamiento del mercado y de cómo conseguir que un producto con una calidad adecuada tenga buena repercusión. Otra de las barreras de entrada es la necesidad de un presupuesto adecuado para el proyecto, pues durante el periodo de desarrollo no se obtendrá ningún ingreso. Sólo en la fase de comercialización puede recuperarse la inversión si el proyecto se ha finalizado con éxito y tiene buena aceptación comercial.

La evolución y crecimiento del mercado de videojuegos es exponencial a día de hoy. No obstante no deben de perderse de vista los antecedentes históricos ya tratados previamente, pues la época actual guarda un fuerte paralelismo con la edad de oro del videojuego Español y la previsión personal de la situación es que este crecimiento acabará de la misma forma. Se sufrirá un crecimiento en complejidad de los desarrollos, hasta un punto en que la complejidad a la hora de realizar un producto competitivo sea una barrera de entrada muy difícil de salvar, momento en el cual algunos estudios independientes actuales habrán subido de nivel y se habrán cristalizado como empresas más estables y de mayor envergadura o se disolverán irremediablemente como sucedió entonces. Lo que se tiene a nuestro favor es el uso de la historia como modelo de previsión para no cometer los mismos errores y para adelantarse a los cambios que están por venir, permitiéndonos adoptar una posición ventajosa para cuando llegue ese momento. Entonces si se consigue soportar ese punto crítico, el beneficio será mucho mayor a partir de entonces pues el número de competidores se reducirá drásticamente. Este tipo de suposiciones aún no se perciben en los datos del sector de los videojuegos independientes o para dispositivos móviles por lo que es de prever que de producirse este suceso futuro aún no puede ni vislumbrase en el horizonte.

La tendencia actual de los videojuegos es hacia el *free to play*. Esto es ofrecer los videojuegos de manera gratuita como servicio y cobrar por contenido de pago dentro del juego. Entonces se tiene una base de usuarios mayoritaria que juega gratis pero de los cuales un pequeño porcentaje en torno al 5 % realizan pagos para conseguir determinados elementos del videojuego. Este modelo de negocio se contemplará para la realización de los siguientes videojuegos, no obstante para el presente videojuego se ha optado por emplear otros modelos que encajan mejor con el tipo de producto.

La segmentación de usuarios está marcada fuertemente entre usuarios de tipo *hardcore* y casuales. Los jugadores *hardcore* juegan principalmente a videojuegos punteros en máquinas potentes como *PC's* o consolas de nueva generación. Este tipo de jugadores requiere productos más elaborados y de mayor presupuesto, y aunque este tipo de público gasta más dinero en videojuegos también es una base objetiva menor. Los jugadores casuales utilizan las redes sociales o sus dispositivos móviles para jugar a videojuegos, y aunque el número de este tipo de jugadores supera ampliamente al número de jugadores *hardcore*, también gastan menos dinero de manera individual. Las ventajas de este tipo de público es que no requiere productos con una base tecnológica tan fuerte como el otro tipo de jugadores.

A día de hoy el sector *casual* es más rentable, no obstante es menos fiel. El publico *casual* suele ser cliente de una venta, mientras que al jugador *hardcore* es más sencillo fidelizarlos si se le da el producto que cubre sus elevados niveles de necesidad, y este tipo de jugadores está dispuesto a pagar más dinero.

Como los dos segmentos de mercado son atractivos para la propuesta de este videojuego se ha optado por una estrategia que se encuentra entre los dos segmentos, ya que se va a realizar un videojuego que estará disponible en dispositivos móviles pero con unos niveles de producción y profundidad del contenido muy por encima de la media de los videojuegos casuales. De este modo se aprovecha la gran base de usuarios de estos dispositivos y se les dará un contenido de muy buena calidad. Para ello la propuesta debe apoyarse en una buena campaña de marketing para poder llegar a toda esa base de público.

El mercado en cuanto a jugadores casuales se mueven más por campañas virales, recomendaciones de sus amigos y el tipo de producto que está de moda en ese momento.

Puesto que la concepción del videojuego no es de tipo casual también permite ser publicada en plataformas de escritorio con garantías de ser comercializable. En este tipo de plataformas el videojuego queda lejos de poder competir con las producciones *triple A* que tanto demandan los jugadores *hardcore*, no obstante representa un buen candidato para competir con el resto de videojuegos independientes que también tienen una fuerte demanda entre los usuarios. Los jugadores *hardcore* comprarán el producto si realmente tiene calidad, son un público entendido que sabe lo que quiere y que está dispuesto a pagarlo cuando lo encuentra.

5.3.3. Competencia

Este apartado se analiza la competencia para identificar posibles estrategias que permitan conseguir mayor éxito en la comercialización del producto.

El entorno competitivo es elevado puesto que cada mes salen a la venta miles de videojuegos nuevos para móviles. No obstante, solo una fracción mínima de estos representa competidores reales pues muchos de estos videojuegos no tienen un acabado de calidad suficiente, y de los que cuentan con la calidad suficiente a su vez hay una gran mayoría que no tiene la repercusión adecuada pues no se realiza un plan de marketing y comercialización que permita llevar el producto a sus clientes potenciales.

En cuanto a los competidores de entornos de sobremesa con videojuegos sobre *PC* o las consolas domésticas, podríamos considerar que ese tipo de videojuegos juega en otra liga superior en comparación con el tipo de producto que se pretende realizar en este plan de negocio. Estos videojuegos considerados *triple* A recaudan cantidades monetarias en unos ordenes de magnitud muy superior, no obstante el costo necesario en tiempo y recursos económicos para realizar este tipo de inversiones también es de unos ordenes de magnitud muy superior. Esto entraña unos riesgos en la inversión que los deja fuera del alcance planteado para este proyecto.

Es muy difícil compararse con los competidores, pues algunos de los criterios como volumen de ventas, crecimiento, cuota de mercado o segmentación de clientes son datos desconocidos. En cuanto a otros factores conocidos tendríamos :

- **Precios :** El precio del presente producto se colocará sensiblemente por debajo del precio de los competidores. Para ello se dispondrá de un precio de 2.39 € en dispositivos móviles y 4.99 € en plataformas de escritorio. El estándar por el que se rigen los competidores para este tipo de juegos con mismo nivel de elaboración suele ser 4.99 € en dispositivos móviles y 9.99 € en plataformas de escritorio.

- **Posicionamiento :** Aún no se puede puede comparar el posicionamiento con respecto a los diferentes competidores, no obstante este dato estará disponible una vez se hayan publicado todas las versiones del presente videojuego, pudiendo obtener datos sobre en que posición se encuentra el producto en las diferentes tiendas de descarga digital en comparación con el resto de competidores.

- **Canales de distribución :** Los canales de distribución en los que podrá adquirirse el producto son los mismos en los que se encuentra la competencia.

En dispositivos móviles la estrategia adoptada por la competencia esta fuertemente dividida en varias áreas:

- **Enfoque *indie* :** Hay grandes joyas que destacan por su originalidad, gusto en los detalles y calidad global del producto en todos los sentidos. Hay pocas propuestas de este tipo, pero cuando surgen tienen muy buena acogida. Podríamos nombrar *«Monument Valley»*, *«Republique»*, *«Plants Vs Zombies»*, *«The Room»*, *«Device 6»*, *«Angry Bird»*, *«Cut The Rope»*, etc...

- **Enfoque *casual* :** Videojuegos virales, generalmente *free to play* donde para avanzar es necesario involucrar a otros amigos y en los que el beneficio se genera mediante micro pagos.

- **Enfoque copia de *hardcore* :** Videojuegos con unos valores de producción elevados, concebidos por grandes equipos multidisciplinarios trabajando para copiar su equivalente original de consola. Este tipo de productos se venden bien, pero tiene poca alma. Se perciben como cartón piedra, demasiado sintéticos donde la copia se percibe como evidente.

- **Enfoque *port* :** Algunos videojuegos son un *port* de videojuegos que tuvieron éxito en consolas anteriores. Este tipo de proyectos solo se vende bien si la conversión se ha realizado con cuidado, permitiendo un control adecuado y obteniendo un resultado que cumpla con los niveles de calidad de hoy día. Se podrían dar muchos ejemplos de malos *ports*, pero por nombrar algunos de los mejores *ports* tendríamos videojuegos como *«Injustice: Gods Among us»*, *«Castle of Illusion»* o *«Xcom: Enemy Unknown»*.

La competencia tiene la fortaleza de que son capaces de captar la atención del público mayoritario. También cuentan con más medios económicos que les permite tener equipos de profesionales con cientos de empleados. Esto les permite sacar mayor número de productos.

En cuanto a las debilidades de la competencia es que sus propuestas son trilladas, realizando copias de copias de mecánicas más que establecidas y sobre las cuales el grado de innovación representa un factor muy pequeño.

La ventana competitiva respecto a a los competidores es que el presente videojuego tiene muy buena base tecnológica, estando a la altura de sus competidores en acabado visual. También la propuesta original y bastante fresca en comparación con los videojuegos de hoy día que tienden a innovar muy poco.

En cuanto a la potencial reacción de los competidores ante el lanzamiento de este producto es difícil calibrar si la repercusión que se conseguirá es suficiente para lograr una reacción. De producirse, algunos tratarían de copiar el producto, lo cual requerirá al menos del mismo tiempo o mucho dinero si se quiere conseguir un resultado a la altura del producto.

5.3.4. Modelo de negocios y plan financiero

En esta sección se detallan todas las lineas de ingresos. El plan financiero detalla el gasto previsto de forma anual basado en hipótesis razonables.

La primera inversión de 55.000€ ya se ha realizado tal como se detalla en el apartado "Evolución y Costes" de este mismo capítulo.

Una vez esté disponibles en las plataformas de descarga digital las 6 versiones comerciales del videojuego (*Windows*, *GNU/Linux*, *Mac OS X*, *iOS*, *Android* y *OUYA*) se planea alcanzar el *breakeven*[1] en un plazo de 2 años.

Teniendo en cuenta que los precios de venta son:

- ***Windows*, *GNU/Linux*, *Mac OS X*, *OUYA* :** 4.99€
- ***iOS*, *Android* :** 2.39€

Consideramos también que el beneficio percibido por cada venta una vez se paga el porcentaje de cada plataforma, se declaran impuestos y se pagan los gastos generados es del 40 %.

Y consideramos que las 6 versiones venden por igual, en tal caso consideramos el precio medio del producto como (4.99+2.39)/2= 3.69€.

Es decir por cada copia vendida se percibe 3.69*0.4 = ~ 1,5€

Entonces para alcanzar el *breakeven* y recuperar la inversión en 2 años (730 * 1.5 * x = 55.000) habría que vender una media de 50 copias al día. Este número es elevado, no obstante puede alcanzarse teniendo en cuenta que debe repartirse entre las 6 versiones del juego (8 copias por versión) y que el publico potencial que puede comprar el juego es realmente grande.

Ver apartado "Estrategia de marketing y ventas" de este plan de negocios para conocer en detalle los modelos de negocio adoptados, los objetivos de número de descargas y los ratios de conversión esperados.

[1] Punto de equilibrio o umbral de rentabilidad es el número mínimo de unidades que una empresa necesita vender para que el beneficio en ese momento sea cero

Como mínimo se plantea el proyecto para un ciclo de vida de 3 años, por lo que el tercer año estaríamos recibiendo un beneficio de ~27.000 €, si el ritmo de ventas continúa estable, habiendo amortizado la inversión inicial.

Si este primer proyecto es rentable se podría plantear una inversión para escalar la propuesta de negocio contratando a profesionales para la creación de un equipo multidisciplinar. La inversión necesaria cubriría los gastos de contratación de personal para la realización de nuevos videojuegos durante los próximos 3 años. Se estima que con 500.000 € la empresa podría funcionar con una plantilla de 5 profesionales durante los próximos 3 años y en ese tiempo podrían realizarse en el orden de 10 proyectos de la calidad y profundidad del videojuego presentado en este proyecto.

5.3.5. Estado de desarrollo y plan de implantación

En esta sección se detalla el estado en el que se encuentra el producto y el plan de implantación para el lanzamiento del mismo.

El calendario de implantación de las diferentes versiones es el siguiente :

- Junio de 2014 : Versión *Windows*, *GNU/Linux* y *Mac OS X*
- Septiembre de 2014 : Versión *iOS*
- Octubre de 2014 : Versión *Android* y *OUYA*

Cada versión tendrá un modelo de negocio y precio diferente para adaptarse al tipo de mercado al que va dirigido. Este tipo de políticas se detallarán en el apartado ”Estrategia de marketing y ventas“ de este plan de negocios.

Como principales hitos tendríamos la publicación de cada una de las 6 versiones a corto plazo. A medio plazo habría como hito la consecución de el *breakeven* y a largo plazo la formación de un equipo multidisciplinario que permitiera realizar los siguientes videojuegos.

En cuanto a las interconexiones entre los distintos grupos de trabajo en este caso no hay, al ser el alumno el único integrante del proyecto. No obstante el tener la visión conjunta de todos los procesos aporta el conocimiento suficiente para optimizar otras tareas como el marketing o los modelos de negocio de las diferentes versiones.

5.3.6. Alianzas estratégicas

En el futuro se planea buscar un *publisher* para contar con mayor músculo de marketing en la publicación de los siguientes videojuegos. Este tipo de editoras permiten llegar más lejos pues están bien posicionadas en el mercado y a cambio se quedan con una parte de los beneficios de la venta.

Los *publishers* también ofrecen servicios de financiación para videojuegos futuros, por lo que tras presentar un prototipo pueden adelantarte el dinero para poder desarrollar dichos proyectos. Para este tipo de tareas es necesario firmar un acuerdo que sea fructífero para las dos partes una vez se realice la publicación del producto.

5.3.7. Estrategia de marketing y ventas

En esta sección se describe como se planea mantener en el tiempo el posicionamiento del producto y se describe que estrategia de marketing va a emplearse para el lanzamiento del producto.

Medios de marketing a emplear :

- Difusión en redes sociales *Twitter*, *Facebook* y *Youtube*.
- Concursos sobre estas redes sociales con sorteo de una copia digital del videojuego.
- Promoción cruzada en los otros productos realizados previamente.
- Descuentos eventuales en el precio del videojuego.
- Inclusión en *Bundles* (*Little Indie*, *Indie Royal* ...)
- Publicación en *Steam Greenlight*
- Inclusión en redes de publicidad en móviles como *AdMob*
- Envío de versión gratuita a los principales *influencers*. *Blogs*, medios especializados y *youtubers*.

Se planea emplear el 50 % de los beneficios de cada copia vendida (es decir el 20 % del precio total del producto) para la realización de acciones de marketing que repercutan en más ventas futuras. Este dinero puede invertirse en campañas de redes sociales desde la que atraer a nuevos usuarios o en publicidad integrada en otros juegos, que permitan alcanzar el

segmento del público adecuado para dirigir la publicidad.

Para el futuro se planea también una estrategia de viralización en las versiones móviles cuando se realiza la compra del videojuego. Para ello se ofrecerán dos modalidades de pago:

- 1. Pago 100 % de la cuantía del precio
- 2. Pago con descuento del 50 % a cambio de realizar la publicación en tu muro de *Facebook* o *timeline* de *Twitter* de que acabas de comprar ese videojuego, poniendo un *link* a la tienda donde puede adquirirse para que otros usuarios puedan verlo. Este método requiere la integración de la *api* de *Facebook* y *Twitter* en el núcleo del videojuego para realizar estas acciones, no obstante es algo que el alumno ya ha realizado anteriormente con éxito en otros videojuegos.

El presente videojuego tendrá un modelo de negocio diferente para cada una de las plataformas de lanzamiento. Estas decisiones se han tomado en base a los hábitos de compra de los usuarios de las diferentes plataformas, adoptando para cada plataforma el modelo más idóneo.

- ***Pay To Play* :** Este modelo de negocio se aplicará en las versiones de Windows, Linux y Mac OS X. El usuario deberá pagar por el videojuego a través de la tienda de venta digital para poder descargarlo.
- ***Freemium* :** Este modelo de negocio se aplicará en las versiones de iOS y OUYA. El usuario podrá descargar gratuitamente el videojuego para jugar el tutorial y el primer nivel. A partir de ese punto deberá pagar si desea continuar jugando. El pago se realizará mediante los mecanismos de *In-App Purchases* de cada plataforma.
- **Gratis con *In Game Advertising* :** Este modelo de negocio se aplicará a la versión de Android. El usuario podrá descargar gratuitamente el videojuego para jugarlo completo pero con publicidad integrada. Adicionalmente se da la posibilidad de realizar un pago mediante los mecanismos de *In-App Purchases* para eliminar la publicidad.

En función a los datos que se disponen de los videojuegos y *apps* realizadas anteriormente se sabe que el *ratio* de conversión entre usuarios que descargan la aplicación/demo y los que realizan el pago *in-app* varía de un 5 % a un 10 %. Estos datos nos llevan a inferir que para obtener 8 conversiones diarias en la versión de *iOS* por ejemplo necesitaríamos que el videojuego tuviera unas 100 descargas al día. Este número entra dentro de los margenes normales de este tipo de mercados y se considera un objetivo de fácil consecución.

5.3.8. Principales riesgos y estrategia de salida

En esta sección se describen los principales riesgos que podría sufrir el producto, y qué posibles estrategias de contingencia podrían ser adoptadas para sobreponerse a tales circunstancias.

Riesgos básicos :

- **Crecimiento menor del esperado :** Este problema podría darse, no consiguiendo generar el número de descargas y conversiones necesarias para garantizar la rentabilidad del producto. Para paliar este riesgo se podría aplicar alguna de las posibles estrategias de contingencia propuestas.
- **Costes mayores a los previstos :** Es difícil que este problema pudiera darse, pues el producto ya se encuentra finalizado y la inversión ya se ha realizado.
- **Entrada inesperada de un competidores :** Ya es un sector con bastante competencia por lo que la entrada de un nuevo competidor no cambia en nada las reglas del juego.
- **Falta de ajuste entre el producto y las necesidades del púbico objetivos :** El producto podría no gustarle al sector de usuarios para el que ha sido diseñado. Para paliar este riesgo se podría aplicar alguna de las posibles estrategias de contingencia propuestas.

Posibles estrategias de contingencia :

- **Modificación del producto :** Se podría modificar el producto para adaptarlo a las características necesarias para convertirlo en un producto más atractivo.
- **Modificación del segmento de mercado potencial :** Es complicado cambiar de segmento pues algunas de las decisiones tempranas que se tomaron afectan al tipo del segmento al que va dirigido el producto.
- **Alianza con alguno de los principales líderes :** Se podría negociar una alianza con alguno de los *publishers* del mercado móvil para que vendan el videojuego bajo su sello. Esto te exige darles un porcentaje de el beneficio, pero por contra ellos tienen mayor capacidad de llegar al público objetivo.

- **Venta total o parcial de la compañía a una empresa del sector más potente :** Se podría vender el videojuego a otra empresa que tenga más capacidad económica y con ello puedan obtener más rendimiento al tener mejores campañas de marketing.
- **Venta o explotación de la tecnología y su patente :** Algunas de las tecnologías diseñadas para el videojuego descritas en el apartado de Desarrollo de este capítulo se pueden vender de manera individual para que otros estudios partan de esas soluciones tecnológicas y no tengan que implementarlas desde cero.
- **Venta de la base de clientes :** Se puede distribuir el videojuego de manera gratuita e incrustar publicidad en todas las versiones, poniendo a disposición de las empresas que realizan campañas de promoción cruzada toda esa base de usuarios activos de la que se disponga, generando con ello ingresos por publicidad.

5.4. EVOLUCIÓN Y COSTES

A continuación se describen las fases llevadas a cabo para la obtención del producto final. En cada uno de los puntos siguientes se hará referencia a diferentes capítulos de este documento, donde se explica en mayor detalle los diferentes elementos, que se han ido generando en dichas etapas.

5.4.1. Análisis de requisitos

Antes de realizar el análisis de requisitos primero se realizó un estudio del tipo de producto a crear teniendo en cuenta el público objetivo, el tamaño del mercado, la inversión necesaria y los elementos fundamentales que debería tener. Esta información puede consultarse al principio de este mismo capítulo.

Una vez se tuvo claro qué tipo de videojuego se iba a realizar y qué tipo de funcionalidades y características básicas debía tener para ser atractivo comercialmente se pasó a la realización del documento de diseño del videojuego. Este documento describe cómo va a ser el videojuego de principio a fin, cubriendo aspectos como la historia, el tipo de enemigos, las acciones del personaje, el tipo de niveles, el tipo y funcionamiento de cada *ítem*, etc. El *GDD* no contiene información técnica sobre cómo van a ser implementas cada una de estas funcionalidades.

5.4.2. Diseño del Sistema

Para poder afrontar las funcionalidades descritas en el análisis de requisitos se hizo el diseño de una arquitectura, que mediante el uso de diferentes sistemas interrelacionados entre sí, permitiera desarrollar el videojuego de forma modular.

Puede consultarse la información completa de la arquitectura del sistema y el detalle de cada uno de los sistemas en el aparatado “Desarrollo” de este capítulo.

En esta fase se inició la realización de los *assets* más básicos con un escenario de prueba y un personaje con el que ir desarrollando y probando los diferentes sistemas.

5.4.3. Diseño del Programa

En esta fase se realizaron los algoritmos necesarios para que los sistemas fundamentales descritos en el diseño del sistema pudieran funcionar y empezar a desarrollar otras funciona-

lidades que hicieran uso de estos sistemas.

Puede consultarse la información completa de los algoritmos o métodos utilizados por los diferentes sistemas en el aparatado "Desarrollo" de este capítulo.

En esta fase se desarrolló una primera versión de los *assets* más críticos como los personajes, enemigos o *items*, pero el diseño de niveles y el grafismo permaneció en forma de *placeholders*.

5.4.4. Iteraciones

En este apartado se muestra un resumen de los sistemas implementados en cada una de las iteraciones. Para conocer el funcionamiento en detalle de cada una de las partes mencionadas se puede revisar el aparatado "Desarrollo" de este capítulo.

En cada prototipo se fueron realizando pruebas unitarias de cada sistema desarrollado. También pruebas de integración para garantizar que la comunicación entre los diferentes sistemas funcionara de manera correcta.

La frecuencia de creación de prototipos fue mayor al principio del proyecto, debido a que había bastantes cosas que probar, determinar y tener una idea clara de hacia qué resultado se encaminaba el proyecto. Mas adelante los ciclos se fueron haciendo más largos, pues el proyecto tenía una base más sólida y unos objetivos más claros a los que enforcar cada nuevo prototipo.

Iteración 0 :

- **Fecha :** 10 de Enero, 2013
- **Detalle :**
  - Implementación muy simple de personaje con cámara fija
  - Implementación muy simple de un autómata finito determinista que hace de enemigo
- **Resultados del prototipo :** Primer prototipo. Se probó el concepto y se determinó la viabilidad del proyecto. Se implementaron los sistemas de una forma muy rápida y sin

que las soluciones software fueran generales.

Iteración 1 :

- **Fecha :** 15 de Enero, 2013
- **Detalle :**
  - Se realizó una implementación de enemigos más funcional, sensible a diferentes eventos
  - Se implementó una cámara sensible al movimiento del personaje
  - Se implementó el sistema de gestión de partida
  - Primera versión del *HUD*
- **Resultados del prototipo :** Versión un poco más flexible del código y más pulida del concepto. Se determinó que había que añadir una animación de cámara al inicio y al final del nivel.

Iteración 2 :

- **Fecha :** 22 de Enero, 2013
- **Detalle :**
  - Se implementó el recorrido de cámara al principio y final del nivel
  - Conexión del nivel jugable con la escena de *Game Over*
  - Se implementó el recuento de puntos finales y la pausa
- **Resultados del prototipo :** Se validó el concepto de partida de principio a fin. Para el siguiente prototipo hay que pulir el acabado gráfico del nivel.

Iteración 3 :

- **Fecha :** 1 de Febrero, 2013
- **Detalle :**
  - Se finalizaron todas las mecánicas del nivel 1
  - Se añadieron algunos elementos de arte al nivel

  - Se implementó el sistema de partículas
  - Implementación del sistema de gestión de sonido

- **Resultados del prototipo :** Resultado del prototipo bueno, en el siguiente prototipo hay que crear un esqueleto del proyecto con conexión al resto de menús y escenas del videojuego.

Iteración 4 :

- **Fecha :** 8 de Febrero, 2013
- **Detalle :**
  - Implementación del sistema de menús
  - Conexión de las diferentes escenas de menús con el primer nivel y la escena de carga
- **Resultados del prototipo :** Primer esqueleto del proyecto creado, hay que añadir el resto de elementos.

Iteración 5 :

- **Fecha :** 22 de Marzo, 2013
- **Detalle :**
  - Se añadió la introducción
  - Se implementaron los menús
  - Se implementó el sistema de lectura de cómics
  - Se definieron todas las escenas del esqueleto del proyecto, aún sin desarrollar cada uno de los niveles
- **Resultados del prototipo :** Esqueleto validado y completo, hay que finalizar los menús y elementos gráficos entre niveles en los siguientes prototipos.

Iteración 6 :

- **Fecha :** 29 de Marzo, 2013
- **Detalle :**
  - Se rediseño el sistema de sonido para invocarlo mediante una fachada
  - Se añadió la música final de todas las escenas y niveles
- **Resultados del prototipo :** El juego ya se podía percibir con entidad. El primer era nivel jugable y el prototipo permitía navegación entre menús. Para los siguientes prototipos hay que añadir un tutorial y empezar a implementar el resto de mecánicas.

Iteración 7 :

- **Fecha :** 5 de abril, 2013
- **Detalle :**
  - Se implementó el tutorial en el nivel 0
  - Se implementó la primera versión de las mecánicas de jefes finales
- **Resultados del prototipo :** Se validaron las nuevas mecánicas implementadas. Hay que añadir los sistemas de logros e *items* de museo *zombie* para los siguientes prototipos.

Iteración 8 :

- **Fecha :** 10 Abril, 2013
- **Detalle :**
  - Se implementó el menú de logros
  - Se añadió el sistema de estadísticas y notificación de logros
- **Resultados del prototipo :** Se validaron el resto de elementos auxiliares al juego añadidos y validados. Para el siguiente prototipo hay que optimizar el nivel jugable porque el *gameplay* va bastante lento.

Iteración 9 :

- **Fecha :** 12 Abril, 2013
- **Detalle :**
  - Se realizaron optimizaciones en el nivel 1
  - Se determinó el *baseline* gráfico a partir de las modificaciones que se realizaron en el nivel.
- **Resultados del prototipo :** Se ha realizado la primera fase de optimización, el rendimiento es mejor. También se ha optado por ajustar todo el videojuego a la proporción 16:9, pues hasta este punto se soportaban varios *ratios* de resolución, pero es muy difícil ajustar todas las cámaras para los diferentes *ratios*. Por lo tanto el siguiente prototipo deberá centrarse en esta modificación.

Iteración 10 :

- **Fecha :** 19 de Abril, 2013
- **Detalle :**
  - Adaptación de las cámaras y sistema de *HUD* a aspecto 16:9
  - Se finalizó toda la parte de lectura de *cómics* con el material final
- **Resultados del prototipo :** El nivel 1 está optimizado y ejecutándose con *ratio* de aspecto fijo. Ahora se añaden bandas negras para adaptarse a otros *ratios* diferentes. Se determinó que hay que seguir optimizando el videojuego.

Iteración 11 :

- **Fecha :** 23 de Abril, 2013
- **Detalle :**
  - Refactorización de código para adaptarlo al sistema de gestión de punteros
  - Implementación de gestión de instancias
- **Resultados del prototipo :** Se ha añadido un sistema de gestión de punteros a los componentes *scripts* para disminuir la creación de dichos punteros en cada *frame*. Esta mejora aporta velocidad pero hay que seguir optimizando el prototipo.

Iteración 12 :

- **Fecha :** 3 de Mayo, 2013
- **Detalle :**
  - Re diseño y refactorización del sistema de gestión de partículas
  - Añadido el sistema de gestión de *items*
- **Resultados del prototipo :** Se modifica el sistema de gestión de partículas mediante uso de una fachada. A partir de ahora los prototipos se centrarán en seguir implementando mecánicas.

Iteración 13 :

- **Fecha :** 10 de Mayo, 2013
- **Detalle :**
  - Implementación del museo *zombie*
  - Se conectó la persistencia del sistema de gestión de *items* desbloqueables con el museo *zombie*
- **Resultados del prototipo :** Se implementaron más mecánicas auxiliares a la partida y se validaron. A continuación hay que finalizar los niveles de conducción pues los menús y el primer nivel ya se encuentran acabados.

Iteración 14 :

- **Fecha :** 22 de Mayo, 2013
- **Detalle :** Se finalizaron los dos niveles de vehículos.
- **Resultados del prototipo :** Implementación de los niveles de vehículos. Para los siguientes prototipos hay que seguir añadiendo mecánicas.

Iteración 15 :

- **Fecha :** 7 de Junio de 2013
- **Detalle :**

  - Se añadieron los tutoriales a los jefes finales
  - También se implementó la funcionaldiad de selección de nivel libre en el menú

- **Resultados del prototipo :** Se añadieron funcionalidades al menú y a los jefes finales. El siguiente prototipo debe depurar las mecánicas y dejar cerrada la implementación del esqueleto del videojuego, así como el nivel tutorial, el nivel normal, los niveles de conducción y los jefes finales.

Iteración 16 :

Prototipo A (Ver Figura 5.153)

- **Fecha :** 28 de Junio de 2013
- **Detalle :**
  - Menús depurados
  - Nivel tutorial depurado
  - Nivel 1 depurado
  - Niveles de conducción depurados
  - Menús finalizados
  - Implementación del sistema de gestión de entrada y salida personalizable
- **Resultados del prototipo :** Primer prototipo con las mecánicas implementadas. A simple vista se percibe como correcto, pero la soluciones tecnológicas no son genéricas. Hay sistemas como las cámaras o los enemigos muy poco flexibles y la optimización sigue sin ser buena. La conclusión es que hay que seguir mejorando esos aspectos para tener un producto de calidad.

Iteración 17 :

- **Fecha :** 22 de Agosto de 2013
- **Detalle :**
  - Optimización del nivel tutorial
  - Optimización del nivel 1

  - Implementación del efecto *Depth Of Field* de forma compatible con plataformas móviles
  - Implementación del efecto *Bloom* de forma compatible con plataformas móviles
  - Implementación de sombras dinámicas de forma compatible con plataformas móviles
  - Creación de biblioteca de *shaders* propios ajustados para ser multiplataforma
  - Optimización del sistema de reflexiones que viene con *Unity* para hacerlo funcionar en plataformas móviles
  - Optimización del sistema de reflexiones
  - Implementación del sistema de gestión de ajustes gráficos para gestionar los diferentes valores de calidad gráfica
- **Resultados del prototipo :** Se ha optimizado el prototipo y el resultado es bastante mejor. Se puede continuar detallando más la estructura del proyecto.

Iteración 18 :

- **Fecha :** 11 de Septiembre 2013
- **Detalle :**
  - Diseño de niveles completo
  - Esqueleto del videojuego que se puede recorrer pero sin enemigos en cada nivel
- **Resultados del prototipo :** El videojuego ya permite cargar todos los niveles, la mayoría de ellos están vacíos sin enemigos pero puedes llegar al final y acabar el nivel para pasar al siguiente. También se ha detectado que los niveles de conducción no acaban de encajar del todo, se debe cambiar el punto de vista para hacerlo más sencillo. También hay que pulir un poco más los jefes finales, se perciben demasiado bruscos en comparación con el resto de niveles. Añadir cámaras de inicio y fin de nivel.

Iteración 19 :

- **Fecha :** 12 de Octubre 2013
- **Detalle :**
  - Reestructuración de los niveles de vehículos

- Se añadieron nuevas animaciones iniciales y finales en el sistema de jefes finales

- **Resultados del prototipo :** El videojuego se percibe mejor en los jefes finales y niveles de conducción. Se dan por finalizadas las mecánicas de nivel normal, nivel tutorial, niveles de jefes finales y niveles de conducción. Hay que seguir optimizando para depurar más el producto.

Iteración 20 :

- **Fecha :** 1 de Noviembre 2013

- **Detalle :**

  - Se rediseño el sistema de *HUD* para que fuera más optimo
  - Se re diseñó el sistema de gestión de partículas para optimizarlo mediante la técnica de *pooling*
  - Se realizaron optimizaciones en los *shaders*
  - Se optimizaron los efectos de postproceso

- **Resultados del prototipo :** Prototipo con una mejora de optimización adecuada. Los siguientes prototipos irán enfocados a terminar el grafismo de niveles e ir desarrollando el resto de *Assets* que no son definitivos.

Iteración 21 :

- **Fecha :** 20 de Diciembre 2013

- **Detalle :**

  - Realización del *Lightmapping* completo de todos los niveles
  - *Asset* gráficos terminados en la mayoría de niveles

- **Resultados del prototipo :** Gráficos acabados, con esqueleto del videojuego completo. Los niveles se pueden recorrer pero no hay enemigos ni cámaras sensibles al contexto de cada situación. Hay que re diseñar el sistema de enemigos y cámaras para que permita un diseño de niveles más flexible.

Iteración 22 :

- **Fecha :** 10 de Enero 2014
- **Detalle :**
  - Se re diseñó el sistema de cámaras para hacerlo más flexible para el diseño de niveles
  - Se colocaron las zonas de cámara en cada uno de los niveles
  - Se realizó la implementación del sistema de flujo de nivel
- **Resultados del prototipo :** Con el sistema de cámaras rediseñado se han distribuido por todos los niveles y ahora el videojuego puede recorrerse de principio a fin. Los gráficos están terminados pero falta re diseñar el sistema de enemigos y colocarlos en los diferentes niveles.

Iteración 23 :

- **Fecha :** 7 de Febrero 2014
- **Detalle :**
  - Se rediseñó el sistema de enemigos de nuevo, para que fuera más flexible en el diseño de niveles
  - Se reestructuraron el nivel tutorial y nivel 1 conforme al nuevo sistema de enemigos
- **Resultados del prototipo :** Sistema de enemigos completo. Se han añadido los enemigos, peligros y bloqueos a los niveles 0 y 1. A continuación hay que extender esto a todos los niveles.

Iteración 24 :

- **Fecha :** 25 de Marzo 2014
- **Detalle :**
  - Se terminó el diseño de niveles con todos los enemigos y cámaras
  - Se realizaron algunas optimizaciones más
  - Se pulió el acabado gráfico en determinados apartados

- Modificación del sistema de gestión de entrada y salida para adaptarlo a los nuevos controles de *Joystick* y teclado

- **Resultados del prototipo :** Todo el videojuego es jugable. Hay partes con muchos enemigos, partes con pocas, zonas con baja calidad de textura, zonas con defectos o fallos en algunas zonas de cámara. Aunque el videojuego está completo se percibe como sin pulir. Hay que refinar un poco los detalles y conseguir una versión final con todo bien trabajado.

Iteración 25 :

Prototipo B (ver Figura 5.153)

- **Fecha :** 30 de Mayo 2014

- **Detalle :** Versión final con todo implementado, todas las optimizaciones y todo pulido hasta el final. Preparado para la realización de testing.

- **Resultados del prototipo :** Versión del videojuego con todo implementado, detalles pulidos y bien optimizado. Se han eliminado la mayoría de los *bugs* en el proceso de prototipado, pero aun así el videojuego no está 100 % probado. A partir de ahora se usará este prototipo para hacer el *testing* final.

5.4.5. Comparativa de prototipos

En este apartado se realiza una comparación entre de uno de los prototipos de mitad del desarrollo y el último de ellos. Los niveles comparados están implementados al 100 % en los dos, pero guardan las siguientes diferencias:

Prototipo A (Junio de 2013)

- Consumo de memoria *RAM* ~1.5*Gb*.

- Rendimiento medio ~30 *FPS* a 800x600 pixeles de resolución.

- Modelo de renderizado con *pipeline Deferred Rendering*. Sólo compatible con dispositivos de escritorio.

- Uso de los efectos de postproceso que vienen de serie con *Unity3D*. Sólo aptos para dispositivos de escritorio.

- Uso de los shaders vienen de serie con *Unity3D*. Sólo aptos para dispositivos de escritorio.

Prototipo B (Mayo de 2014)

- Consumo de memoria *RAM* ~512*Mb*.
- Rendimiento medio ~60 *FPS* a 1920x1080 pixeles de resolución.
- Uso de los efectos de postproceso propios. Compatible con dispositivos que soporten *OpenGL ES 2.0*.
- Biblioteca de *shaders* propia. Compatible con dispositivos que soporten *OpenGL ES 2.0*.
- Modelo de renderizado con *pipeline Forward Rendering*. Compatible con dispositivos que soporten *OpenGL ES 2.0.*
- Interface *HUD* mejorada.
- Nueva cámara en las escenas de vehículos.
- Sistema de cámaras nuevo.
- Sistema de partículas optimizado con *pooling*.
- Uso cola de gestión de instancias de enemigos inteligente.

En las Figuras 5.153, 5.154 y 5.155 puede apreciarse el cambio de aspecto entre los dos prototipos.

Figura 5.153: Comparación entre prototipos 1 de 3. Arriba el prototipo A, abajo el prototipo B

Figura 5.154: Comparación entre prototipos 2 de 3. Arriba el prototipo A, abajo el prototipo B

Figura 5.155: Comparación entre prototipos 3 de 3. Arriba el prototipo A, abajo el prototipo B

5.4.6. Pruebas sobre el prototipo final

Además de las pruebas unitarias y de integración realizadas a cada uno de los prototipos durante el desarrollo evolutivo, se pasó a realizar pruebas de todo el sistema sobre la versión final. Para la generación de la versión 1.0. se hicieron pruebas en los siguientes dispositivos:

- Portátil *Mac* con sistema operativo *Mac OS X 10.7.*
- *PC* con sistemas operativos *Windows 7* y *Ubuntu GNU/Linux 14.04*

En cada una de las versiones se realizaron las siguientes comprobaciones directamente sobre el producto final, utilizando los propios controles del videojuego:

- Instalación desde cero para que no tenga valores guardados, comprobar que se hace bien la persistencia de desbloqueo de *items*, la gestión del dinero, los logros o el progreso de la partida.
- Verificar que la persistencia de los datos se mantiene después de reiniciar el juego en múltiples ocasiones.

- Cambiar todos los valores gráficos y comprobar que el juego funciona correctamente en cada uno de ellos.
- Probar que funcionan todos los ajustes de control y sonido.
- Probar los dos modos de control: teclado y *Joystick*.
- Verificar que las leyendas de todas las pantallas sean consistentes con el modo de control seleccionado.
- Activar la pausa dentro del juego y comprobar que funciona correctamente y que te permite volver al juego.
- Probar los 3 niveles de dificultad y ver que los enemigos se comportan de forma acorde.
- Probar a morir en los 3 niveles de dificultad para ver en cuáles permite continuar y en cuáles no.
- Probar todos los movimientos en los 3 personajes.
- Probar todas las magias en los 3 personajes.
- Pasarse el juego con cada uno de los 3 personajes.
- Obtener todos los logros.
- Desbloquear todos los elementos del *zombie museum*.

Para probar toda esta batería, sin que nada falle el software y sin pararse se emplean unas 10 horas por versión. Como resultado de la detección y subsanación de los diferentes *bugs*, al final de esta etapa se obtuvo la versión 1.0. del videojuego.

5.4.7. Fase de beta privada

Después se obtener la versión 1.0. del videojuego se inició un periodo de *beta* privada. Para ello se distribuyó el videojuego entre diferentes compañeros y amigos que ayudaron reportando algunos *bugs* y aportando el siguiente *feedback*:

- Para invocar la tecla *F1* en los *Mac* hay que pulsar la tecla función. Al darle directamente en el videojuego a la tecla *F1* se cambian los valores de brillo de la pantalla.

- Problema en el *wizard* de configuración gráfica en algunos *Mac* cuando se pone la pantalla completa.
- Los tres puntos suspensivos de *Loading* parpadean demasiado rápido.
- Errores de traducción en los textos.
- Algunas onomatopeyas no existen en el idioma Inglés.
- Problema con el ejecutable de *GNU/Linux* 64 y dependencias de 32 *Bits*.
- Problema en el tutorial usando teclado, donde la tecla 'O' se usa simultáneamente para quitar el *tutorial* y para acceder a un comando.

Después de la corrección de dichos *bugs* detectados sobre la versión 1.0. y como resultado del proceso de *beta* privada, se obtuvo la versión 1.1., considerándose esta como la versión comercial del juego.

5.4.8. Implantación

En este caso, la fase de implantación se corresponde con la publicación del software en las diferentes plataformas de distribución digital. Actualmente algunas de las versiones del videojuego se encuentran ya publicadas y es posible comprar el videojuego. Otras de las versiones están pendientes de publicación, siendo esto un proceso largo de verificación y aceptación que involucra a entidades como *Google*, *Apple*, *Steam*, o *Desura*, y que comienza una vez ha sido completada la finalización del producto software. Para conocer más detalles sobre el estado de desarrollo y plan de implantación puede consultarse el "Plan de negocio" en el capítulo 5.

5.4.9. Mantenimiento

Se prevé un ciclo de vida de este software mínimo de unos 3 años, coincidiendo con el plan de negocio propuesto. En este periodo se correjirán los posibles errores que puedan surgir. Además hay previsto dedicar recursos a la mejora y adaptación del producto. Estas propuestas de mejora para el futuro se detallan en el capítulo 6 de este documento.

5.4.10. Coste de desarrollo

El periodo de implementación del presente Proyecto de Fin de Grado comprende desde Noviembre del 2012 hasta Mayo del 2014 (1 año y 6 meses) a razón de una media de 40

horas semanales. Esto suma un total de 2880 horas de trabajo. En este tiempo se ha desarrollado un producto con ~40.000 lineas de código fuente y ~20.000 *assets* gráficos, sonoros y de otros tipos multimedia. El videojuego resultante dura una media de 4 horas en su primera partida y unas 10 horas si se quiere desbloquear todo el contenido que ofrece.

A continuación se recogen los costes en recursos humanos, costes de *hardware*, y costes en licencias de software. Para el coste de recursos humanos se han considerado los 18 meses de trabajo. Y como referencia del valor de sueldo se ha usado el coste medio de un perfil de desarrollador de videojuegos en España, que asciende a una media de 30.000 € anuales según el Libro Blanco de desarrollo Español de videojuegos [27].

Recurso	**Cantidad**	**Coste (euros / unidad)**
Recursos humanos	1	~45.000 €
Ordenador portatil *Mac Book Pro 15”*	1	~1.600 €
Monitor Apple Led Cinema Display 27”	1	~1.000 €
Ordenador sobremesa *Quad Core 2.4GHz*	1	~700 €
Móvil *iPhone 5*	1	~600 €
Tableta *iPad 3*	1	~600 €
Móvil *Android Nexus 4*	1	~400 €
Ouya Kit Desarrollo	1	~600 €
Ouya versión comercial	1	~100 €
Assets externos utilizados[2]	*	~800 €
Servicios externos de traducción	1	~200 €
Licencia *«Unity3D»* escritorio	1	~1.200 €
Licencia *«Unity3D» iOS*	1	~1.200 €
Licencia *«Unity3D» Android*	1	~1.200 €
Licencia *Google Play*	1	~25 €
Licencia *iOS App Store*	1	~80 €
Licencia *Mac App Store*	1	~80 €
Total		**~55.000 €**

Tabla 5.1: Coste estimado del desarrollo del videojuego.

5.4.11. Estadísticas del código fuente

En la tabla 6.2. se muestra un resumen del número de ficheros y lineas de código de los diferentes sistemas y apartados del proyecto. Para ver un desglose más detallado puede consultarse el anexo C del presente documento.

Subsistema	**Número de archivos**	**Líneas de código**
Menús / gestión de logros	21	~7.400
Control del personaje, Enemigos y grupos	62	~11.200
Sistema de juego, sonido y cámaras	57	~11.000
Items	52	~2.700
Renderización	34	~4.500
Biblioteca de *Shaders*	67	~3.500
Total	293	**~40.000**

Tabla 5.2: Estadísticas del código fuente.

5.4.12. Distribución del trabajo

A continuación se muestra una distribución del tipo de trabajo en el total del videojuego. Esta estimación se ha realizado a partir de el número de tareas completadas de cada área y de el tiempo utilizado para ello. También puede apreciarse en forma de gráfico en la figura 6.1.

Trabajo de *game designer* / *level designer* / guionista (20 % del total) :

- Diseño de mecánicas jugables (7.5 % del total)
- Diseño niveles (10 % del total)
- Guión (2.5 % del total)

Trabajo de grafista 3D / 2D (40 % del total) :

- Diseño de menús (2 % del total)
- Diseño de interface (2 % del total)
- Creación de cómics (9 % del total)
- Modelado / Texturizado de escenarios y *props* (20 % del total)
- Adaptación de recursos externos (2 % del total)
- *Setup* de iluminación (5 % del total)

Trabajo de programación / *testeo* / *QA* (40 % del total) :

- Diseño del software (2 % del total)
- Implementación de prototipos (18 % del total)
- Implementación de optimizaciones (8 % del total)
- Adaptación los requisitos de cada plataforma (4 % del total)
- Pruebas (8 % del total)

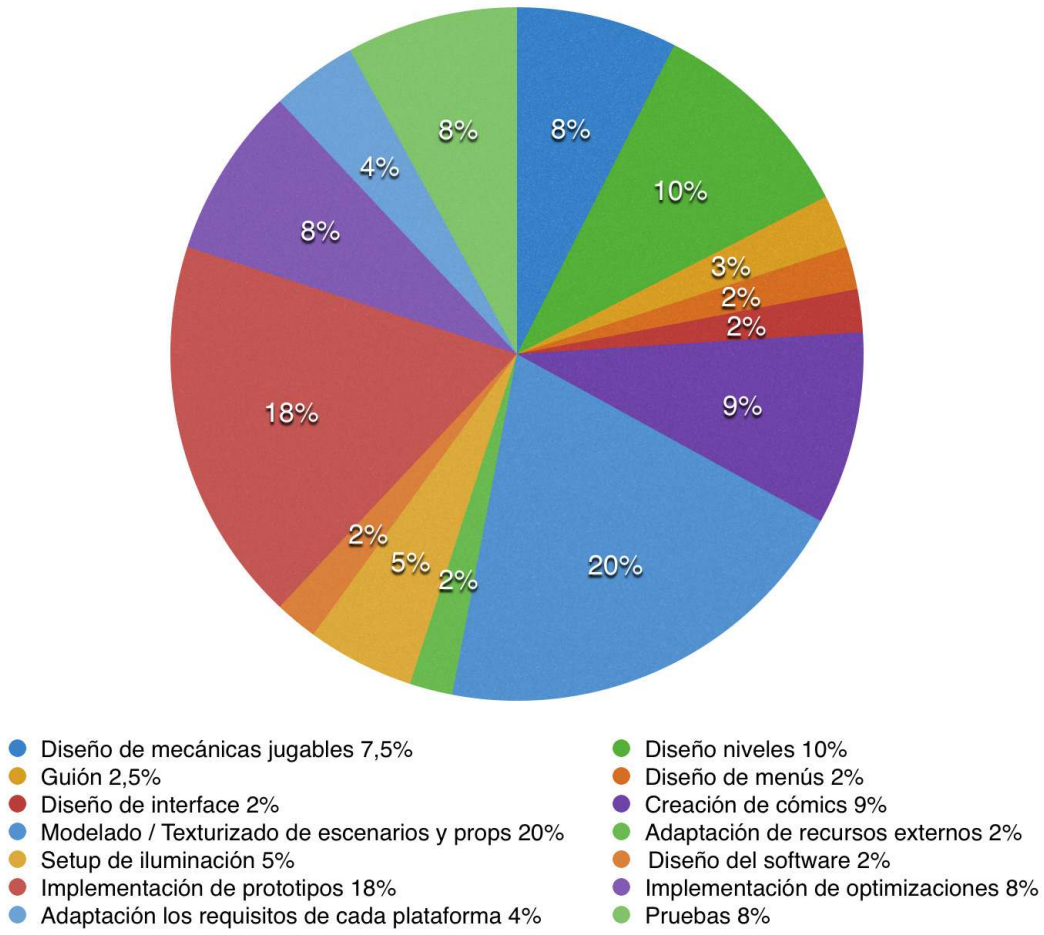


Figura 5.156: Distribución del tiempo en función del tipo de tarea

5.4.13. Otros datos del videojuego

A continuación se muestran algunos datos estadísticos del software de modo que se pueda apreciar el alcance de la producción :

- Número de horas trabajadas en el proyecto : ~2800
- Número de lineas del código fuente : ~40.000
- Número de assets del proyecto : ~20.000
- Duración en primera partida completa : ~4 horas
- Duración estimada de partida con 100 % desbloqueado y logros conseguidos : ~10 horas
- Número de niveles : 21
- Número de viñetas de cómic : ~300
- Número de items desbloqueables : 50
- Número de logros propuestos : 27
- Número de escenas en *Unity3D* : 95
- Duración estimada promedio por nivel : ~5 minutos
- Número de objetos *3D* promedio por nivel : ~200
- Número de enemigo promedio por nivel : ~100
- Número de materiales promedio por nivel : ~680
- Número de texturas promedio por nivel : ~1700
- Número de *clips* de audio promedio por nivel : ~100
- Número de *GameObjects* alojados en memoria promedio por nivel : ~6600
- Cantidad de memoria *RAM* consumida promedio por nivel : ~512Mb
- Número medio de *drawcalls* realizadas promedio por nivel : ~150
- Número medio de triángulos renderizados por frame promedio por nivel : ~100.000

CAPÍTULO 6. CONCLUSIONES Y PROPUESTAS

En este capítulo se describen los objetivos alcanzados y se proponen una serie de líneas de trabajo futuro para mejorar y extender la propuesta comercial. Se recomienda consultar la web oficial[1] del videojuego donde puede verse un *trailer* y consultar las últimas novedades sobre las plataformas en las que se ha publicado.

6.1. OBJETIVOS ALCANZADOS

Se han cumplido todos los objetivos descritos en el capítulo 2. El presente videojuego multiplataforma constituye una propuesta comercial con características de alta calidad y con una base tecnológica a la altura de los estándares actuales del videojuego. Este objetivo general se ha conseguido en base a los siguientes subobjetivos :

6.1.1. Etapa de diseño

- **Diseño de videojuego basado en niveles :** El videojuego se ha dividido en niveles. Para ello se ha vertebrado sobre una historia que evoluciona conforme el jugador avanza en el cumplimiento de los distintos niveles.

  Esta estructura es muy buena, pues permite que en el futuro se pueda extender la trama con historias alternativas y nuevos niveles. Esto se plantea en las propuesta de trabajo futuro de este mismo capítulo.

- ***Items* coleccionables y logros desbloqueables :** También se han implementado elementos que permiten crear mayor nivel de enganche como son el uso de *items* coleccionables que pueden comprarse con las monedas que se recojan en los diversos niveles. Por otro lado se proponen logros desbloqueables que tienen un efecto directo en la implicación del jugador para tratar de conseguirlos. Estos desafíos secundarios te los plantea el videojuego en forma de retos y acciones específicas. Si se consiguen aportan al jugador la recompensa de monedas extra.

[1] http://www.rockzombiegame.com

Estos elementos de diseño permiten alargar la duración del videojuego desde las 4 horas de la primera vuelta hasta 10 horas si se quiere desbloquear todo el contenido. Estas funcionalidades permiten que el producto tenga mayor grado de rejugabilidad.

- **Estudio de publico objetivo :** Según el estudio de publico objetivo se determinó realizar un videojuego que se adapte a los siguientes perfiles de usuarios :

  - Usuarios de videojuegos independientes, que valoran ideas originales por encima de la calidad técnica.
  - Usuarios de móviles que quieren videojuegos más profundos en comparación con los videojuegos casuales habituales en estas plataformas.
  - Usuarios que quieren disfrutar de géneros de videojuego clásicos con gráficos actuales.

  Estos perfiles de público objetivo representan un nicho de mercado poco saturado en plataformas móviles.

- **Análisis de productos similares / competencia :** Al estudiar los productos de la competencia se han detectado diversas áreas en las que poder destacar:

  - Por un lado se han empleado gráficos tridimensionales en oposición a los gráficos *2D* de la mayoría de los videojuegos existentes del mismo género. Esta decisión ha permitido añadir rodajes de cámara y movimientos más cinematográficos en el tratamiento de la acción, dotando así al videojuego de un acabado visualmente más impactante.
  - Por otro lado se ha dotado al videojuego de mayor duración, pues lo normal para un videojuego arcade de este tipo es una hora. En cambio el presente videojuego tiene una duración de cuatro horas en su primera vuelta, aportando más contenido para que el jugador pueda disfrutar del título de una forma más duradera.

6.1.2. Etapa de desarrollo

- **Uso de gráficos tridimensionales :** Se han utilizado gráficos *3D* con modelos de una cantidad de polígonos que los hace adecuados tanto para máquinas de escritorio como para dispositivos móviles. La ventaja de la cámara elegida por defecto en el videojuego

hace que al visualizarse los objetos desde esa distancia el resultado se perciba muy detallado.

Esta solución es mejor en comparación con gráficos *2D*, porque al disponer de gráficos vectoriales, el contenido se puede renderizar en diferentes resoluciones de pantalla sin que se pierda calidad de imagen. Con ello se logra cubrir el espectro de las resoluciones más exigentes de dispositivos *Full HD* y también las resoluciones más restrictivas de teléfonos móviles.

- **Respuesta en tiempo real :** La respuesta óptima a la entrada de usuario y la renderización fue otro de los puntos críticos del proyecto. Conseguir una tasa de *frames* adecuada para percibir el movimiento como fluido es complicado en dispositivos más limitados. No obstante consiguió salvarse satisfactoriamente mediante el uso de las técnicas de optimización empleadas.

  Esto permite que el videojuego se pueda disfrutar en altas resoluciones y tasas de refresco de 60 *frames* por segundo y que se garantice una experiencia de usuario fluida.

- **Multiplataforma :** Se ha conseguido portar el videojuego a múltiples plataformas gracias al enfoque que se le ha dado al contenido desde el principio. Para ello se ha desarrollado todo el contenido pensando en que sea compatible con chips de dispositivos móviles, pero sin perder calidad en plataformas de escritorio.

  La adopción de múltiples versiones es algo clave para rentabilizar correctamente el videojuego, pues permite que sea distribuido en diferentes canales de venta. Además para cada plataforma se ha implementado un modelo de negocio y forma de control diferentes en función de los hábitos de consumo de los usuarios.

- **Uso de *shaders* programables :** Gracias a la biblioteca de *shaders* desarrollada se ha alcanzado nivel de calidad gráfica con un rendimiento adecuado. Estos *shaders* permiten su renderización en *pipelines* de tarjetas gráficas punteras pero también en circuitería de móvil.

  El hecho de plantear los *shaders* desde un primer momento como universales para que funcionaran en estas dos familias de *hardware* supuso una dificultad considerable. Se ha utilizado para lograr efectos avanzados como la distorsión del fuego, la reflexión del agua o las reflexiones especulares con normales en los objetos usando una sola *drawcall*[2].

[2]Termino utilizado en *OpenGL ES* para referirse al pintado de geometría mediante una lista compilada

- **Efectos *postproceso* avanzados :** Se han implementado efectos que se consideran un estándar entre los clientes de videojuego actuales como *Bloom*, *DOF* o corrección de color. Para ello el sistema de gestión de calidad, puede ajustarse específicamente al tipo de dispositivo en el que se está ejecutando. Esto permite que el videojuego pueda tener un rendimiento adecuado en una gran gama de dispositivos.

  Para lograr este objetivo fue crucial la implementación de efectos avanzados por métodos compatibles con chips limitados como los que montan los dispositivos móviles.

- **Sombras dinámicas :** Se han empleado técnicas avanzadas de renderización de sombras en tiempo real e iluminación dinámica y se ha conseguido que estas técnicas funcionen correctamente en dispositivos móviles. Para ello se emplearon mecanismos de optimización que se detallan ampliamente en el capítulo 5 de este documento.

  El uso de iluminación dinámica y sombras en tiempo real no suelen tenerse en cuenta en versiones móviles. Por lo que su inclusión en el videojuego de forma compatible con todas las versiones ha supuesto un esfuerzo extra, pero aporta un valor añadido a la calidad de estas versiones.

6.1.3. Etapa de comercialización

La etapa de comercialización es un proceso que conlleva la publicación, promoción y mantenimiento del software en cada una de las plataformas objetivo. Actualmente el videojuego se encuentra en esta etapa para algunas de las versiones del mismo, y otras se encuentran pendiente de implantación en los diferentes mercados conforme al calendario previsto descrito en el capítulo 5.

- **Estudio de plataformas de publicación :** Según el estudio de las diferentes plataformas de publicación se determinó que el videojuego debía publicarse en Steam Greenlight, Desura, Mac App Store, App Store, Google Play y Ouya Store entre otras.

  En algunas de estas plataformas ya se han pasado las certificaciones necesarias, logrando la publicación y consiguiendo un alto grado de impacto:

  - En Desura el videojuego consiguió permanecer entre los cinco primeros puestos de popularidad en las tres primeras semanas siguientes a la publicación.
  - En Steam Greenlight [40] se ha conseguido a día 8/7/2014 estar al 82 % de progreso de llegar a la cola de publicación con 15.468 visitantes únicos, 4.392 votos

positivos y 277 comentarios de los usuarios. Estos datos se corresponden a tres semanas después de su publicación. Esta plataforma da acceso a Steam como se describe en el capítulo 3 y es el canal de distribución líder para los usuarios de PC. Para ello se requiere que la comunidad apoye el proyecto. Si comparamos el presente videojuego con otros proyectos, se contemplan buenas perspectivas de poder alcanzar la publicación en Steam. Otros proyectos exitosos han tardado meses en poder llegar a la cola de publicación, y la gran mayoría no lo logran porque no tienen suficiente apoyo de la comunidad.

En plataformas móviles aún no se ha lanzado. Según el calendario de implantación se ha propuesto la publicación y fase de promoción de estas versiones a partir de Septiembre de 2014. El motivo es repartir en el tiempo las fechas de lanzamiento, es poder dedicar el tiempo necesario a la promoción de cada versión.

- **Elección del modelo de negocio :**

  En función del estudio de los usuarios de cada plataforma se determinó que sus hábitos y forma de consumir videojuegos es diferente. Por este motivo se han planteado modelos de negocio diferentes en cada plataforma para maximizar el éxito comercial del proyecto. La distribución de modelos de negocio adoptada es la siguiente :

  - ***Pay To Play* :** Este modelo de negocio se aplicará en las versiones de Windows, Linux y Mac OS X. El usuario deberá pagar por el videojuego a través de la tienda de venta digital para poder descargarlo.
  - ***Freemium* :** Este modelo de negocio se aplicará en las versiones de iOS y OUYA. El usuario podrá descargar gratuitamente el videojuego para jugar el tutorial y el primer nivel. A partir de ese punto deberá pagar si desea continuar jugando. El pago se realizará mediante los mecanismos de *In-App Purchases* de cada plataforma.
  - **Gratis con *In Game Advertising* :** Este modelo de negocio se aplicará a la versión de Android. El usuario podrá descargar gratuitamente el videojuego para jugarlo completo pero con publicidad integrada. Adicionalmente se da la posibilidad de realizar un pago mediante los mecanismos de *In-App Purchases* para eliminar la publicidad.

- **Plan de marketing :** Se ha desplegado correctamente el plan de marketing en las versiones publicadas hasta la fecha. Para ello se creó un *kit* de prensa digital que se

envió a los diferentes medios online para que pudieran disponer del producto y así analizarlo en sus respectivos medios.

Se han llevado a cabo acciones como las siguientes :

- Envío del *kit* de prensa a medios específicos de videojuego, y también a multitud de *youtubers* y principales *influencers* del momento en el ámbito de los videojuegos.
- Publicación de contenido en redes sociales como *Twitter*, *Facebook* y *Linkedin* para dar a conocer el producto, distribuyendo información sobre el juego y los diferentes canales de venta en los idiomas Español e Inglés.
- Asistencia a la feria nacional de desarrollo de videojuegos *Gamelab* donde se entregaron panfletos y *DVD-ROM's* entre los asistentes y medios de prensa. Esta acción también permitió tomar contacto con *publishers* para negociar la publicación del videojuego en la plataforma *Wii U*. Por el momento estas negociaciones están abiertas, pudiendo ser una opción factible si el juego cosecha suficiente éxito en el resto de versiones.

La estrategia de viralización para la versión de plataformas con pago *In-App* se llevará a cabo cuando se publiquen estas versiones. Esta estrategia consiste en tener dos precios de compra, uno de ellos al 50 % de descuento y otro normal. Para realizar el pago *In-App* con descuento debes dar permiso para publicar un *Tweet* en el que se le enviará a tus seguidores un mensaje haciendo publicidad del videojuego.

- **Distribución y presentación en medios :**

  A día de hoy, ya se han conseguido grandes hitos en cuanto a distribución y presencia en medios :

  - **Indie Royale Bundle 15 :** El videojuego fue seleccionado para formar parte de el paquete de videojuegos independientes Indie Royale Bundle 15. En dicho paquete, los beneficios se reparten entre los desarrolladores, y una parte va destinada a caridad. En la semana que duró la promoción se vendieron más de 2500 unidades del bundle.
  - **Medios de prensa nacionales e internacionales :** El videojuego ha tenido buena presencia en prensa, destacando su mención en medios de reconocido prestigio

com *«Meristation»* [42], *«Mashthosebuttons»* [43], *«RetroInvaders»* [44], *«Gamingonlinux»* [45], *«Shdownloads»* [46] o *«RetroManiacMagazine»* [47] entre otros.

- **Presencia en *youtube* :** También se ha tenido buena presencia en youtube, con más de 20 canales realizando *gameplays* para sus seguidores. Solo hay que hacer una búsqueda del termino "rock zombie gameplay" en *youtube* para comprobar el impacto en la comunidad de jugones.
- **Otros canales :** Además de los canales oficiales, se ha detectado la presencia del videojuego en más de 100 portales de descarga pirata de software. Esto no aporta beneficios directamente, pero la piratería es parte de la industria de los contenidos digitales. La parte positiva de esto es que el videojuego se ha pirateado en tal grado es porque despierta interés en la comunidad. Por otro lado esos impactos en entre los usuarios, aunque ilegítimos contribuyen a dar a conocer el producto por medio del "boca a boca".

Es de esperar que cuando el resto de versiones se hayan implantado se consiga tener aún más repercusión y poder llegar a más clientes potenciales.

6.2. PROPUESTAS DE TRABAJO FUTURO

El presente videojuego pretende ser una activo comercial a largo plazo tal como se ha dejado patente en el plan de negocios descrito en el capítulo 5. Para garantizar el ciclo de vida adecuado de este software, a continuación se describen algunas propuestas de trabajo futuro :

- **Ampliación del contenido mediante paquetes descargables :** La lógica del videojuego ha sido diseñada de tal forma que sea posible añadir nuevos niveles y elementos de forma sencilla. Esto permite ampliar el videojuego mediante paquetes descargables que añadan historias paralelas al mismo, nuevos modos de juego, nuevos objetos coleccionables, nuevos logros desbloqueables o incluso nuevos capítulos sobre la trama principal de la historia. Este tipo de añadidos se podrán comprar dentro del propio software del videojuego y permitirán ampliar el ciclo de vida del proyecto software.

  Para realizar esta ampliación habría que implementar en el software el mecanismo de descarga de contenido adicional e integración de este contenido dentro del programa. En esta modificación se estima que supondría el trabajo de una semana del alumno.

Para realizar cada una de las ampliaciones descargables se estima que el alumno necesitaría un mínimo de 3 meses para crear paquetes de contenido de un 25 % de la longitud total del videojuego base. Este trabajo conllevaría labores de escritura de guión, diseño de niveles, diseño de mecánicas de nuevos enemigos, modelado y texturizado de escenarios y programación de nuevas funcionalidades.

- **Localización a diferentes idiomas :** Actualmente el videojuego sólo está disponible en el idioma Inglés. En futuras iteraciones de mejora se planea soportar nuevos idiomas. Para ello se contará con traductores de diversas lenguas y se ampliará la implementación del software para que soporte la modalidad de múltiples idiomas. Los idiomas contemplados por orden de implantación serán Chino, Japonés, Español, Alemán y Francés.

  Para realizar esta ampliación habría que implementar en el software el mecanismo de soporte para múltiples idiomas. Esto tendría impacto sobre los textos de los menús, los cómics y los textos de dentro del videojuego. De este modo cada vez que se cargase una nueva escena se consultaría el idioma configurado y se daría diferente valor a cada uno de estos elementos localizables que en este momento son estáticos.

  Se estima que el alumno podría realizar esta modificación en un plazo de dos semanas. En cuanto a cada nueva traducciones a idioma extranjero se estima que haría falta gastar 200 € por cada una contratando los servicios de un traductor externo. Esta cifra ha sido calculada en base al número de palabras de texto contenidas en el videojuego y el coste de traducción por palabra de la plataforma *ICanLocalize* [39].

- **Adaptación a nuevas plataformas y nuevos modelos de negocio :** También está previsto portar el videojuego a plataformas como *Windows 8*, en su modalidad de teléfonos y *tablets*. Por otro lado, el auge de las nuevas plataformas de videojuegos (*Wii U*, *XBox One*, *PS4*) y las nuevas políticas de apertura a desarrolladores independientes se considera como posible una publicación futura del presente software en estos sistemas. En cuanto a nuevos modelos de negocio, se tiene previsto un plan de comercialización para el mercado chino de *smartphones*. Este mercado tiene unas reglas diferente, pues la comercialización del software no depende de las plataformas de venta digital, sino de los operadores de telefonía. Para ello se presentará una propuesta que permita la venta del videojuego previa traducción al lenguaje Chino y la adaptación al modelo *free to play* para adecuarlo al tipo de software de entretenimiento que se consume en ese país.

  Se estima que el alumno podría realizar la portabilidad a plataformas como Windows

8 en su modalidad de teléfonos móviles en el plazo de dos semanas. Este tiempo se emplearía principalmente en optimizaciones específicas para la plataforma, de modo que garanticen el correcto funcionamiento.

En cuanto a versiones de consola se estima que el alumno podría realizar cada nueva versión en el plazo de un mes por plataforma. Sería necesario dedicar parte del tiempo a labores de optimización y adaptación a dicha plataforma, pero la mayor cantidad del tiempo se emplearía en cumplir con las guías de estilo y políticas de publicación de cada plataforma concreta.

- **Adaptación a nuevas características gráficas en los dispositivos móviles :** Puesto que el presente videojuego se ha diseñado teniendo como objetivo que sea multiplataforma y debido al sistema de gestión de calidad gráfica implementada que permite ajustarse a las características del *hardware*, se pueden obtener diferentes perfiles dependiendo del *hardware* donde se ejecuta. En los móviles actuales, debido a sus características técnicas limitadas, se está adoptado un perfil con gráficos más restrictivos. No obstante según aumenten las capacidades de los terminales y tablets en los próximos años se cuenta con un gran margen disponible para seguir ofreciendo una calidad puntera en estos dispositivos. Se estima que en 3 años el *hardware* gráfico de los móviles sea equivalente al de los equipos portátiles de hoy en día. En ese tiempo el videojuego podrá seguir siendo puntero técnicamente con muy pocos cambios en el software. Esta estimación coincide con el ciclo de vida comercial mínimo previsto para este producto.

  Por el momento sólo habría que tocar los valores de un fichero del código fuente para escalar la calidad gráfica. Esto es debido a que el propio videojuego incluye un configurador de ajustes gráficos y sólo habría que cambiar los valores máximos. Además en el futuro sería posible modificar la biblioteca de *shaders* para aportar más calidad a los materiales o efectos postproceso. Esta modificación no requeriría cambiar nada de el código base del videojuego.

6.3. CONCLUSIÓN PERSONAL

La elaboración del presente Trabajo Fin de Grado me ha permitido aplicar la mayor parte de los conocimientos aprendidos durante mis años de estudio en la Escuela Superior de Informática, junto con otros conocimientos adquiridos de manera autodidacta, para la obtención de un producto software con calidad profesional. Además me permite apostar por otra

rama de la informática con unas salidas profesionales muy altas.

La realización del presente videojuego es para mi un gran éxito. Por un lado, si la acogida comercial es positiva, este proyecto puede cristalizar en el desarrollo de una empresa con la cual realizar más videojuegos y dar trabajo a otros profesionales de mi misma rama laboral. En cambio, si el producto no es rentable, sigue constituyendo un éxito personal. La mera finalización de un proyecto completo de esta envergadura abre la puerta a la contratación por empresas del sector que demanden personas con formación académica en el desarrollo de software y con experiencia previa en el desarrollo de videojuegos.

La ingeniería del software, estructura de datos, la informática gráfica, los autómatas y lenguajes formales, los procesadores de lenguajes, el estudio de sistemas operativos, la inteligencia artificial, la algoritmia de ampliación de programación, la síntesis de imagen digital y junto con otras disciplinas me han demostrado su utilidad y su potencial, ayudándome a desarrollar mis capacidades como analista y desarrollador de software.

Como conclusión estoy muy satisfecho de haber elegido la titulación de Grado en Ingeniería Informática y más aún de haberme especializado en esta rama concreta del desarrollo de videojuegos.

CAPÍTULO 7. BIBLIOGRAFÍA

[1] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering (3rd Edition)*. AK Peters, 2008. ISBN-10: 1568814240, ISBN-13: 978-1568814247.

[2] E. Adams. *Fundamentals of Game Design (3rd Edition)*. New Riders, 2013. ISBN-10: 0321929675, ISBN-13: 978-0321929679.

[3] E. Adams , J. Dormans. *Game Mechanics: Advanced Game Design (Voices That Matter)*. New Riders, 2012. ISBN-10: 0321820274, ISBN-13: 978-0321820273.

[4] A. Alfred, A. V., Lam, M., Sethi, R., Ullman, J. D. *Compiladores: Principios, Técnicas y Herramientas*. Addison-Wesley, 2008. ISBN-10: 9702611334, ISBN-13: 978-9702611332.

[5] M. Buckland. *Programming Game AI by Example*. Wordware Game Developer's Library, 2004. ISBN-10: 1556220782, ISBN-13: 978-1556220784.

[6] C. Czerkawski, C. Czerkawska. *Breaking Into Video Game Design - A Beginner's Guide*. Wordarts, 2011. ASIN: B0061U4HZM.

[7] T. Dunniway, J. Novak. *Game Development Essentials: Gameplay Mechanics*. Cengage Learning, 2008. ISBN-10: 1418052698, ISBN-13: 978-1418052690.

[8] D.S.C. Dalmau. *Core techniques and algorithms in game programming*. New Riders Pub, 2004. ISBN-10: 0131020099, ISBN-13: 978-0131020092.

[9] W. Engel *ShaderX7: Advanced Rendering Techniques. Course Technology*. Cengage Learning, 2009. ISBN-10: 1584505982, ISBN-13: 978-1584505983.

[10] E. Gamma. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995. ISBN-10: 0201633612, ISBN-13: 978-0201633610.

[11] C. González Morcillo, J. A. Albusac Jiménez, C. Mora Castro, S. Fernández Durán. *Desarrollo de Videojuegos: Programación Gráfica (2ª Edición)*. Bubok, 2013. ISBN-10: 978-84-686-4026-6.

[12] J. Gregory. *Game Engine Architecture*. AK Peters, 2009. ISBN-10: 1568814135, ISBN-13: 978-1568814131.

[13] J. Hight, J. Novak. *Game Development Essentials: Game Project Management*. Cengage Learning, 2007. ISBN-10: 1418015415, ISBN-13: 978-1418015411.

[14] R. Henson Creighton. *Unity 4.x Game Development by Example Beginner's Guide*. Packt Publishing, 2013. ISBN-10: 1849695261, ISBN-13: 978-1849695268.

[15] M. Menard. *Game Development with Unity*. Course Technology PTR: Stacy L. Hiquet, 2011. ISBN-10: 1435456580, ISBN-13: 978-1435456587.

[16] J. W. Murray. *C# Game Programming Cookbook for Unity 3D*. A K Peters/CRC Press, 2014. ISBN-10: 1466581409, ISBN-13: 978-1466581401.

[17] I. Millington and J. Funge. *Artificial Intelligence for Games*. Morgan Kaufmann, 2009. ISBN-10: 0123747317, ISBN-13: 978-0123747310

[18] I. Millington, J. Funge. *Artificial Intelligence for Games (2nd edition)*. CRC Press, 2009. ISBN-10: 0123747317, ISBN-13: 978-0123747310.

[19] H. Nguyen. *GPU Gems 3*. Addison-Wesley Professional, 2007. ISBN-10: 0321515269, ISBN-13: 978-0321515261.

[20] J. Novak. *Game Development Essentials: An Introduction (3rd edition)*. Cengage Learning, 2011. ISBN-10: 1111307652, ISBN-13: 978-1111307653.

[21] J. P. Ordoñez. *Power-Ups: Conviertete en un profesional de los videojuegos*. Plan B Editorial, 2013. ISBN: 978-84-941128-3-6.

[22] S.J. Russell and Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 1998 ISBN-10: 0136042597, ISBN-13: 978-0136042594 .

[23] S. Rogers. *Level Up! The Guide to Great Video Game Design (2nd edition)*. Wiley, 2014. ISBN-10: 1118877160, ISBN-13: 978-1118877166.

[24] A. Silberschatz, P.B. Galvin, and G. Gagne. *Fundamentos de Sistemas Operativos*. McGraw Hill, 2006. ISBN-10: 8448146417, ISBN-13: 978-8448146412.

[25] S. St-Laurent. *Shaders for Game Programmers and Artists (Premier Press Game Development)*. Course Technology PTR, 2004. ISBN-10: 1592000924, ISBN-13: 978-1592000920.

[26] D. Villa, F. Moya, M. A. Redondo, J. López, F. J. Villanueva, M. García, J. L. González. *Desarrollo de Videojuegos: Técnicas Avanzadas (2ª Edición)*. Bubok, 2013. ISBN-10: 978-84-686-4026-9.

[27] Informe "libro blanco del desarrollo Español de videojuegos"
http://www.dev.org.es/images/stories/docs/LibroBlancoDEV%20alta_compr.pdf

[28] Información sobre la historia de los videojuegos
http://es.wikipedia.org/wiki/Historia_de_los_videojuegos

[29] Información de ventas sobre el videojuego *Candy Crush*
http://www.teinteresa.es/tecno/internet/Candy-Crush-corona-rey-apps_0_1087692413.html

[30] Información sobre los videojuegos más rentables
http://www.vidaextra.com/industria/los-juegos-mas-rentables/o-cuando-el-dinero-te-sale-por-las-orejas

[31] Página web de *TurboSquid*
http://www.turbosquid.com

[32] Página web de *3DModels & Textures*
http://www.3dmodels-textures.com/store/

[33] Página web de *3DRT*
http://3drt.com/store/

[34] Página web de *The 3D Studio*
http://www.the3dstudio.com

[35] Página web de *Vector Stock*
http://www.vectorstock.com

[36] Página web de *Opuzz*
http://www.opuzz.com

[37] Página web de *Audio Jungle*
http://audiojungle.net

[38] *Asset Store* de *Unity3D*
https://www.assetstore.unity3d.com

[39] Página web de *ICanLocalize*
http://www.icanlocalize.com/site/

[40] Página web de *Steam Greenlight* de *«Rock Zombie»*
http://steamcommunity.com/sharedfiles/filedetails/?id=268218453

[41] Página web del *Indie Royale Bundle*
http://www.indieroyale.com/blog

[42] Noticia en *Meristation* sobre *«Rock Zombie»*
http://www.meristation.com/pc/noticias/rock-zombie-el-heavy-metal-espanol-a-golpe-de-guitarra/58/1990289

[43] Noticia en *Mash Those Buttons* sobre *«Rock Zombie»*
http://mashthosebuttons.com/2014/07/smash-zombie-faces-as-a-member-of-an-all-ladies-rock\-band-in-rock-zombie/

[44] Noticia en *Retro Invaders* sobre *«Rock Zombie»*
http://retroinvaders.com/es/50934/rock-zombie-te-lleva-de\-vuelta-a-los-origenes-del-beat-em-up-mas-clasico

[45] Noticia en *Gaming On Linux* sobre *«Rock Zombie»*
http://www.gamingonlinux.com/articles/rock-zombie-action-game-released-for-linux.3931

[46] Noticia en *Shdownloads* sobre *«Rock Zombie»*
http://www.shdownloads.com.ar/2014/07/analisis-rock-zombie-pc-2014.html

[47] Noticia en *Retro Maniac Magazine* sobre *«Rock Zombie»*
http://retromaniacmagazine.blogspot.com.es/2014/06/rock-zombie-te-lleva-de-vuelta-los.html

[48] Información sobre como redactar un plan de negocio
http://es.wikipedia.org/wiki/Plan_de_negocio

[49] Información sobre el crecimiento del mercado móvil
http://www.poderpda.com/noticias/el-mercado-de-\
desarrollo/de-apps-moviles-alcanzara-100-mil-millones-de-\
dolares-en-2015/

[50] Perspectiva histórica de la edad de oro del videojuego en España
http://www.elmundo.es/elmundo/2011/09/25/valencia/
1316934951.html

[51] Información sobre la edad de oro del software Español
http://es.wikipedia.org/wiki/Edad_de_oro_del_software_
espa~nol

[52] Información sobre los mejores y peores videojuegos Españoles
http://blog.kelkoo.es/tecnologia/videojuegos/
espanoles-los-5-mejores-y-peores-2012-07

[53] Información sobre *Apis* Gráficas
http://arraysgame.blogspot.com.es/2011/09/apis-graficas.
html

[54] Información sobre motores de videojuego
http://es.wikipedia.org/wiki/Motor_de_videojuego

[55] Información sobre *Cocos 2D*
http://es.wikipedia.org/wiki/Cocos2d

[56] Información sobre *Ogre 3D*
http://es.wikipedia.org/wiki/OGRE_3D

[57] Información sobre *SDL*
http://es.wikipedia.org/wiki/SDL

[58] Información sobre los 10 motores gráficos más importantes de la industria
http://www.neoteo.com/top-10-los-motores-graficos-mas-importantes

[59] Información sobre *Unity3D*
http://es.wikipedia.org/wiki/Unity_(software)

[60] Página oficial de*Unity3D*
http://www.unity3d.com

[61] Página oficial de *Ouya*
http://http://www.ouya.tv

[62] Uso de efectos de post proceso *unreal engine*
https://docs.unrealengine.com/latest/FRA/Engine/Rendering/PostProcessEffects/index.html

[63] Información sobre Programación orientada a componentes
http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software_basada_en_componentes

[64] Información sobre rentabilidad de el desarrollo de videojuegos
http://www.meristation.com/playstation-3/noticias/solo-el-40-de-los-juegos-de-sony-ganan-dinero/61/1991184

[65] Información sobre el videojuego *Limbo*
http://es.wikipedia.org/wiki/Limbo_(videojuego)

[66] Información sobre el videojuego *Fez*
http://en.wikipedia.org/wiki/Fez_(video_game)

[67] Información sobre el videojuego *Super Meat Boy*
http://es.wikipedia.org/wiki/Super_Meat_Boy

[68] Información sobre el videojuego *Braid*
http://es.wikipedia.org/wiki/Braid_(videojuego)

[69] Información sobre el videojuego *Minecraft*
https://minecraft.net

[70] Información sobre el videojuego *Clash Of Clans*
http://www.supercell.net/games/view/clash-of-clans

[71] Información sobre el videojuego *Angry Birds*
https://www.angrybirds.com

[72] Información sobre el videojuego *GTA V*
http://www.rockstargames.com/V/es/

[73] Información sobre el videojuego *Mario Kart Wii*
http://www.mariokart.com/wii/launch/

[74] Información sobre el videojuego *Uncharted 3*
http://en.wikipedia.org/wiki/Uncharted_3:_Drake's_Deception

[75] Información sobre el videojuego *Starcraft 2*
http://en.wikipedia.org/wiki/StarCraft_II:_Wings_of_Liberty

[76] Información sobre el videojuego *Mass Effect 3*
http://en.wikipedia.org/wiki/Mass_Effect

[77] Información sobre el videojuego *Forza Motorsport 5*
http://en.wikipedia.org/wiki/Forza_Motorsport_5

[78] Información sobre el videojuego *Fifa 14*
http://en.wikipedia.org/wiki/FIFA_14

[79] Información sobre el videojuego *Nihilumbra*
http://en.wikipedia.org/wiki/Nihilumbra

[80] Información sobre el videojuego *Paradise Lost First Contact*
http://www.asthreeworks.com/games/

[81] Información sobre el videojuego *Candle*
https://www.kickstarter.com/projects/spaniardmike/candle-a-dynamic-graphic-adventure

[82] Información sobre el videojuego *Gods Will Be Watching*
http://www.deconstructeam.com/games/gods-will-be-watching/

[83] Información sobre el videojuego *Dead Synchronicity*
http://www.deadsynchronicity.com

[84] Información sobre el videojuego *Castlevania Lords Of Shadows 2*
http://en.wikipedia.org/wiki/Castlevania:_Lords_of_Shadow_2

[85] Información sobre el videojuego *DeadLight*
http://en.wikipedia.org/wiki/Deadlight

[86] Información sobre el videojuego *Invizimals*
http://es.wikipedia.org/wiki/Invizimals

[87] Información sobre el videojuego *Imagina a ser diseñadora de moda*
http://www.ubi.com/ES/Games/Info.aspx?pId=6030

[88] Información sobre el videojuego *ZackZero*
http://en.wikipedia.org/wiki/Zack_Zero

[89] Noticia sobre la recaudación económica en el videojuego *GTA V*
http://www.europapress.es/portaltic/videojuegos/noticia-gta-recauda-mas-800-millones/dolares-primer-dia-20130919102424.html

[90] Información sobre el *GDD*
http://www.gamasutra.com/view/feature/3384/the_anatomy_of_a_design_document_.php

[91] *Template* de un *GDD*
http://www.google.es/url?url=http://www-personal.engin.umd.umich.edu/~bmaxim/cis488/BaldwinGameDesignDocumentTemplate.doc&rct=j&sa=U&ei=OY3NTOOFK8-dOpHnnJEB&sqi=2&ved=0CBwQFjAA&sig2=neDZkwkd6bcKQ-CMr9VRXA&q=game+design+document+template&usg=AFQjCNHGRaFPUKkLZPbB9xcpTi0ozNrtSw&cad=rja

[92] Imagen del videojuego *The Walking Dead*
http://www.vrutal.com/noticias/329-los-10-juegos\-con-mejor-trama-de-la-pasada-generacion

[93] Imagen del videojuego *Diablo 3*
http://guides.gamepressure.com/diabloiii/guide.asp?ID=15225

[94] Imagen ilustrativa del efecto conocido como *Long Tail*
http://lab.vente-privee.com/les-strategies-de-\

referencement-naturel-a-linternational-12-la-gestion\
-du-contenu/

[95] Imagen del videojuego *Diablo 3*
http://www.gry-online.pl/S024.asp?ID=1239&PART=5020

[96] Imagen del videojuego *Farmville*
http://blog.youngworks.nl/signalen/sociaal-gamen

[97] Imagen del videojuego *Tomb Raider*
http://vgtoday.net/2013/02/tomb-raider-monastery\
-escape-gameplay-video/tomb-raider-2/

[98] Imagen del videojuego *Angry Bird*
http://ultradownloads.com.br/baixar-jogo/tiro-ao-alvo/
Angry-Birds/

[99] Imagen del videojuego *Gran Turismo 6*
http://www.gamerzona.com/2013/11/11/
impresiones-de-gran-turismo-6-para-ps3/

[100] Imagen del videojuego *NBA 2k14*
http://www.gamerzona.com/analisis/nba-2k14/

[101] Imagen del videojuego *Metal Slug*
http://www.ferra.ru/ru/apps/games/2013/12/15/
metal-slug-ii-metal-slug-x-prodolzhenie-retro-banketa.
html

[102] Imagen del videojuego *Monkey Island 2*
http://www.juegoviejo.com/juego/monkey-island-2\
-lechucks-revenge-3-5/

[103] Imagen del videojuego *Flashback*
http://www.dotnes.com

[104] Imagen del videojuego *Castlevania clásico*
http://www.playstationgeneration.it/2010/04/
i-giochi-che-hanno-definito-playstation.html

[105] Imagen del videojuego *Golden Axe*
http://www.the-arcade.ie/2014/02/interview-super-retro-world/

[106] Imagen del videojuego *Castle Crashers*
http://www.castlecrashers.com

[107] Imagen del videojuego *Charlie Murder*
http://imcharliemurder.com

[108] Imagen del videojuego *Sacred Citadel*
http://www.deepsilver.com/game-view/view/game/access16/sacred-citadel-13/

[109] Imagen del videojuego *Shank*
http://www.kleientertainment.com/games/shank/

[110] Imagen del videojuego *Street of Rage*
http://www.zonared.com/pc/analisis/analisis-retro-streets\-of-rage-md-ms-gg/

[111] Imagen del videojuego *Rome Total War*
http://www.supergameplay.com.br/sony-revela-novo-video-com\-gameplay-do-exclusivo-ps4-the-order-1886/

[112] Imagen del videojuego *Battlefield 4*
http://www.battlefield.com/es/battlefield-4/images

[113] Imagen del videojuego *Devil May Cry*
http://regionps.com/dmc-devil-may-cry/

[114] Imagen del videojuego *Mortal Kombat 9*
http://www.gameview.com.br/Video/Mortal-Kombat-Komplete\-Edition--Pc-Scorpion-vs-Sub-Zero.aspx?v=1l5r3

[115] Imagen del videojuego *Sonic*
http://www.3djuegos.com/comunidad-foros/tema/16099100/0/analisis-sonic-the-hedgehog-megadrive/

[116] Imagen del videojuego *GTA IV*
http://livinggames.ru/xbox360-pc/item/603-gta-5.html

[117] Imagen del videojuego *Silent Hill Downpour*
http://www.meristation.com/xbox-360/silent-hill-downpour/juego/1533703

[118] Imagen del videojuego *Fifa 14*
http://www.ea.com/es/fifa14

[119] Imagen del videojuego *The Walking Dead*
http://www.niubie.com/2012/03/el-juego-the-walking\-dead-llegara-por-capitulos-empezando-en-abril/

[120] Imagen del videojuego *Forza Motorsport 5*
http://tierragamer.com/review-forza-motorsport-5/

[121] Imagen del videojuego *Mass Effect 3*
http://oxcgn.com/2012/04/26/oxcgns-mass-effect-3\-review-a-triumphant-endgame/

[122] Imagen del videojuego *Starcraft 2*
http://eu.battle.net/sc2/es/

[123] Imagen del videojuego *Uncharted 3*
http://es.mashpedia.com/Uncharted_3:_La_traición_de_Drake

[124] Imagen del videojuego *Braid*
http://store.steampowered.com/app/26800/?l=spanish

[125] Imagen del videojuego *Fez*
http://www.eurogamer.es/articles/fez-analisis

[126] Imagen del videojuego *Limbo*
http://www.mediavida.com/foro/juegos/resultados-top-mediavida-best-indie-games-ever-473816

[127] Imagen del videojuego *Minecraft*
http://www.retrorequiem.com/minecraft-review-pc\-version-simply-perfect/

[128] Imagen del videojuego *Super Meat Boy*
http://www.poligonu.com/2012_10_01_archive.html

[129] Imagen del videojuego *Castlevania2*
http://www.trueachievements.com/n15262/castlevania-lords-of-shadow-2-special-editions.htm

[130] Imagen del videojuego *Deadlight*
http://www.meristation.com/xbox-360/deadlight/analisis-juego/1766167

[131] Imagen del videojuego *Imagina ser diseñadora de moda*
http://www.3djuegos.com/juegos/pc/2630/fashion-designer/

[132] Imagen del videojuego *Invizimals*
http://www.3djuegos.com/15532/invizimals-la-alianza/

[133] Imagen del videojuego *Zack Zero*
http://zackzero.com/?lang=es

[134] Imagen del videojuego *Candle*
http://www.tekustudios.com

[135] Imagen del videojuego *Dead Syncrhonicity*
http://es.ign.com/pc/70413/news/dead-synchronicity\-una-aventura-a-la-espanola

[136] Imagen del videojuego *Gods Will BeWatching*
http://www.deconstructeam.com/games/gods-will-be-watching-es/

[137] Imagen del videojuego *Nihilumbra*
http://store.steampowered.com/app/252670/?l=spanish

[138] Imagen del videojuego *Paradise Lost*
http://www.vidaextra.com/aventura-plataformas/paradise-lost-first-contact-celebra-su-ultimo-dia-en-kickstarter

[139] Efectos gráficos de *Unreal Engine 4*
https://docs.unrealengine.com/latest/INT/Engine/Rendering/PostProcessEffects/index.html

[140] Imagen utilizada efecto de uso de *cubemaps* 2
http://www.bundysoft.com/docs/doku.php?id=l3dt:gallery

[141] Imagen utilizada para ilustrar el uso de *cubemaps* 1
http://blog.csdn.net/bonchoix/article/details/8279813

[142] Imagen utilizada para ilustrar el uso de *cubemaps* 2
http://aass.oru.se/~mbl/avdg/lectures-2008/

[143] Imagen utilizada para ilustrar el efecto *bloom*
http://en.wikipedia.org/wiki/Bloom_(shader_effect)

[144] Imagen utilizada para ilustrar el efecto *motion blur*
http://www.fxguide.com/featured/unreal-engine-4-next\
-gen-gaming-effects/

[145] Imagen utilizada de modelo *3D* poligonal
http://www.residentevilsh.com/foro/viewtopic.php?f=1&t=
27630

[146] Imagen utilizada con efecto de partículas 1
http://www.moddb.com/games/red-alert-a-path-beyond/
images/particle-effects

[147] Imagen utilizada con efecto de partículas 2
http://store.steampowered.com/app/228380/?l=spanish

[148] Imagen utilizada con reflexiones *cubemap* en dinámicos
http://cgi.tutsplus.com/articles/working-with\
-advanced-reflections-in-udk--cg-18783

[149] Imagen utilizada con reflexiones planas
http://www.indiedb.com/features/cryengine-going\
-subscription-based

[150] Imagen utilizada *Rigging*
http://candle3d.com/3ds-max/3d-animation.html

[151] Imagen utilizada comparativa *Vertex Shader vs Pixel Shader*
http://www.ict456.com/wp-content/uploads/2012/10/
pixel-shader1.jpg

[152] Imagen utilizada de sombras precalculadas 1
http://docs.cryengine.com/display/SDKDOC4/Static+vs.+Dynamic+Lighting

[153] Imagen utilizada de sombras precalculadas 2
http://forum.unity3d.com/threads/bsp-style-geometry\-creation-tool-with-lightmapping.33446/

[154] Imagen utilizada de *Sprite*
http://www.deviantart.com/morelikethis/309594606

[155] Imagen utilizada de *UV-Mapping*
http://support.quest3d.com/index.php?title=Chapter_2.4:_Surface_properties

[156] Imágenes de patrones 1 de 2
http://www.dofactory.com/Patterns/Patterns.aspx

[157] Imágenes de patrones 2 de 2
http://codebuild.blogspot.com.es/2013/04/a-theorical-look-into-objectresource.html

APÉNDICE A. MANUAL DE USUARIO

En este anexo se detalla cómo realizar la instalación del videojuego para cada versión (En el DVD adjunto pueden encontrarse los instaladores para Mac OS X, GNU/Linux y Microsoft Windows). A continuación también se explica de una manera resumida cómo controlar el videojuego en cada una de las versiones, puesto que este *software* ya incluye un *tutorial* interactivo en el que el jugador puede familiarizarse con los controles de la plataforma en la que lo está ejecutando.

A.1. INSTALACIÓN

- **Mac OS X:**
  1. Mover el fichero RockZombie.app a la carpeta Aplicaciones
  2. Ejecutar el fichero RockZombie.app

- **GNU/Linux:**
  1. Mover el contenido de la carpeta Rock Zombie al disco duro
  2. Añadir permisos de escritura al fichero RockZombie/RockZombie.x86
  3. Ejecutar el fichero RockZombie.x86 desde un terminal o escritorio gráfico

- **Windows:**
  1. Mover el contenido de la carpeta Rock Zombie al disco duro
  2. Ejecutar el fichero RockZombie.exe desde el disco duro

- **iOS:**
  1. Descargar e instalar el videojuego desde App Store
  2. Ejecutar una vez instalado desde el dispositivo

- **Android:**
  1. Descargar e instalar el videojuego desde Google Play
  2. Ejecutar una vez instalado desde el dispositivo

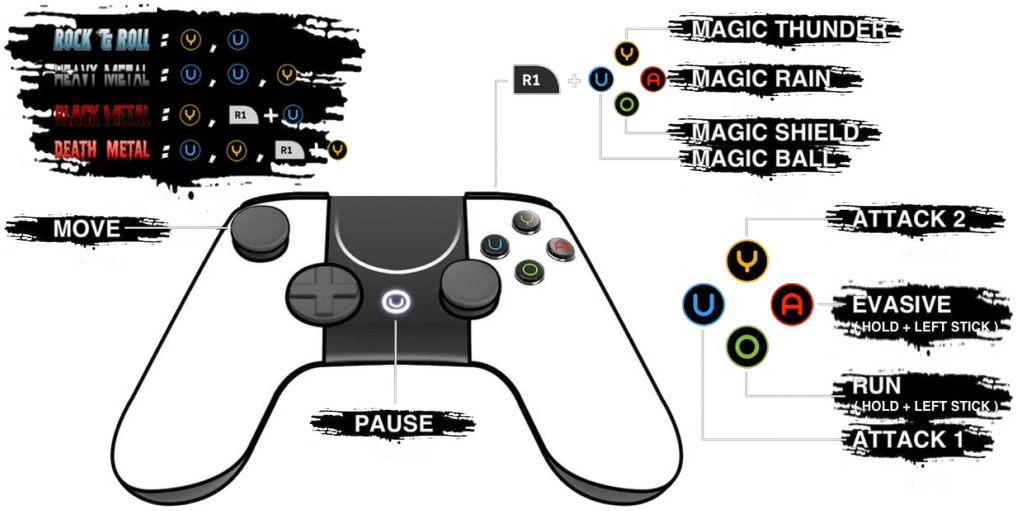
- **OUYA:**
  1. Descargar e instalar el videojuego desde el store oficial de OUYA
  2. Ejecutar una vez instalado desde el menú Play de la consola

A.2. CONTROL

El videojuego podrá controlarse con diversos métodos de control en función de la plataforma en la que se está ejecutando.

- **Mac OS X/ GNU/Linux / Windows:**

- **OUYA:**



- **iOS / Android:**

  Se emplearán los botones virtuales que se pintan sobre la pantalla y los acelerómetros de estos dispositivos en el caso de los niveles de conducción de vehículos. Para ver más detalles se debe completar el *tutorial* específico de estas versiones.

APÉNDICE B. IMÁGENES FINALES DEL VIDEOJUEGO

A continuación se muestran algunas capturas realizadas *in-game*. También puede verse un *trailer* del videojuego en la web oficial de Rock Zombie[1]

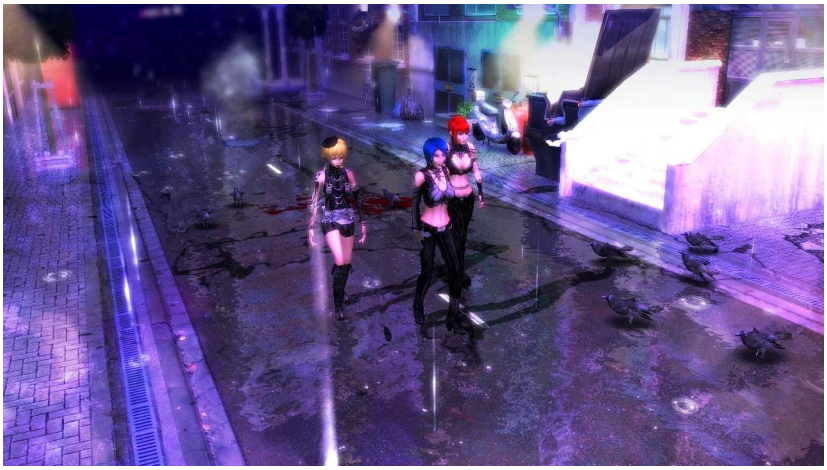


Figura B.1: Rock Zombie captura *in-game* 1

Figura B.2: Rock Zombie captura *in-game* 2

[1] http://www.rockzombiegame.com

Figura B.3: Rock Zombie captura *in-game* 3

Figura B.4: Rock Zombie captura *in-game* 4

Figura B.5: Rock Zombie captura *in-game* 5

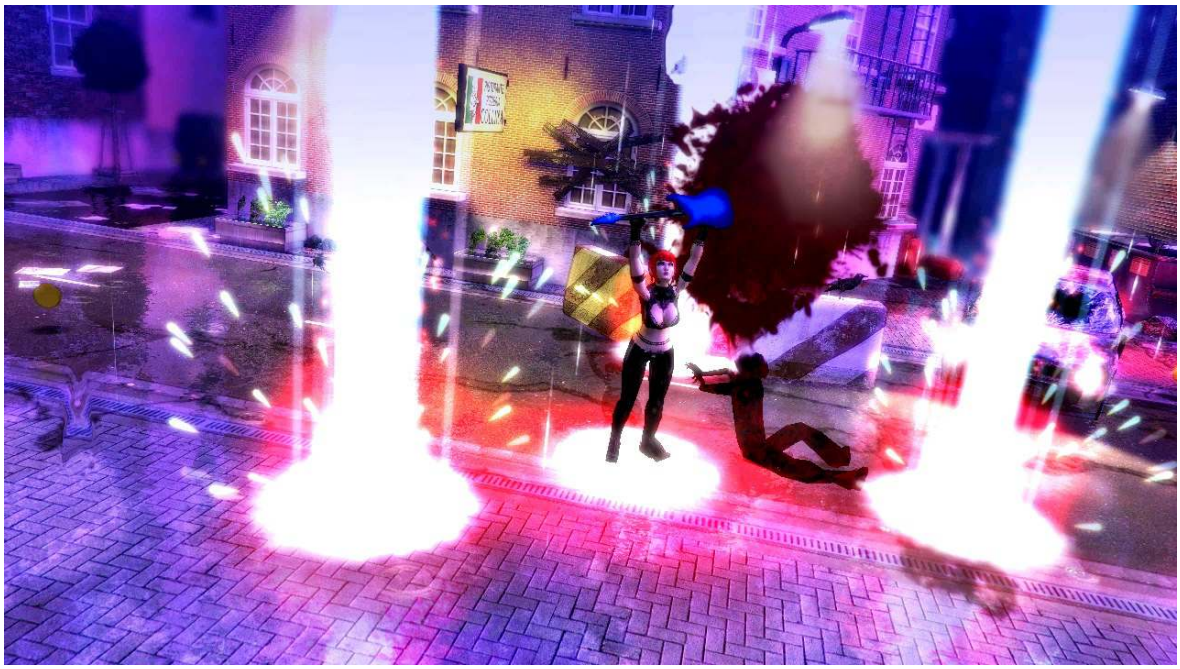


Figura B.6: Rock Zombie captura *in-game* 6

Figura B.7: Rock Zombie captura *in-game* 7

Figura B.8: Rock Zombie captura *in-game* 8

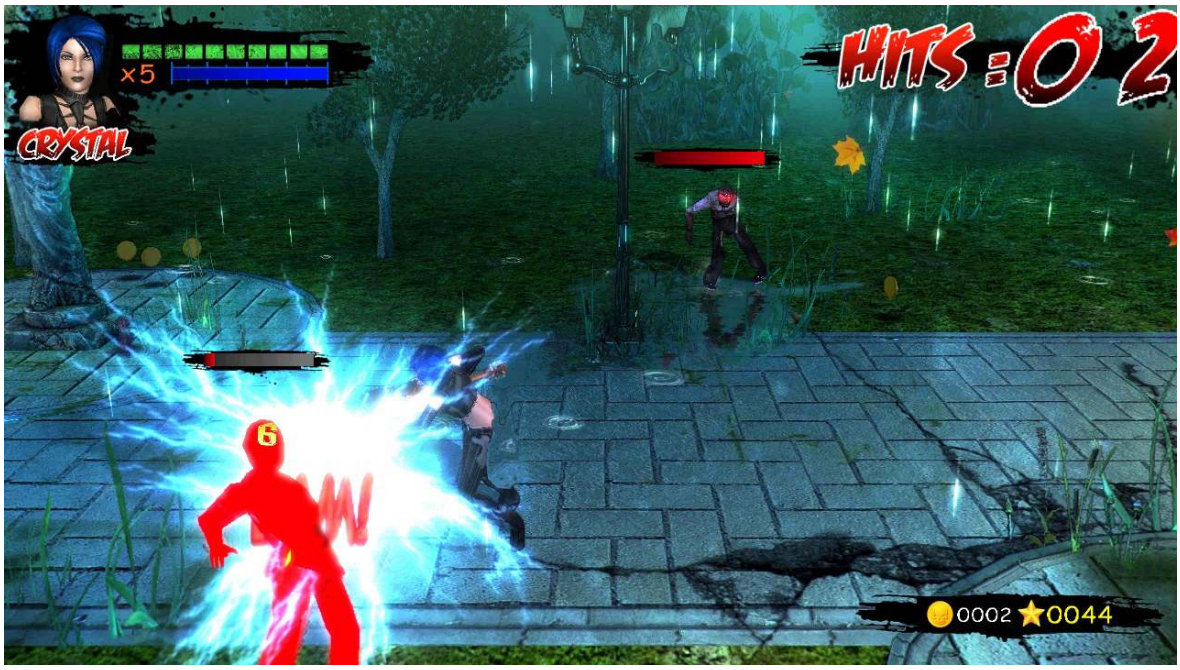


Figura B.9: Rock Zombie captura *in-game* 9

Figura B.10: Rock Zombie captura *in-game* 10

APÉNDICE C. ORGANIZACIÓN DEL CÓDIGO FUENTE

C.1. ORGANIZACIÓN DEL CÓDIGO FUENTE

Los *assets* empleados para la realización del videojuego como modelos *3D*, música o gráficos no han sido incluidos para no vulnerar las licencias individuales de este contenido que sólo permite su distribución como elemento incluido en el binario resultante de la compilación final del videojuego y no su distribución a terceros de manera libre.

En cuanto al código fuente, dada su extensión, (unas 40.000 lineas), únicamente se incluye en su versión electrónica en el *CD* adjunto a este documento. El código fuente se estructura en los siguientes directorios:

C.2. MENÚS / GESTIÓN DE LOGROS (~7400 LINEAS)

Este directorio contiene los *scripts* encargados de dar comportamiento al sistema de menús y otros sistemas secundarios como la gestión de logros.

C.3. CONTROL DEL PERSONAJE, ENEMIGOS Y GRUPOS (~11200 LINEAS)

Este directorio contiene los *scripts* encargados de dar comportamiento al control del personaje, así como la gestión del sistema de enemigos y grupos.

C.4. SISTEMA DE JUEGO, SONIDO Y CÁMARAS (~11000 LINEAS)

Este directorio contiene los *scripts* encargados de definir la arquitectura general del *gameplay*, así como los subsistemas de sonido y cámaras.

C.5. ITEMS (~2700 LINEAS)

Este directorio contiene los *scripts* encargados de definir el comportamiento de cada *ítem* individual como pueden ser los *power ups* o *npc's* que dan mayor vida y dinamismo a los niveles del videojuego.

C.6. RENDERIZACIÓN (~8000 LINEAS)

Este directorio contiene los scripts encargados de codificar los efectos de *postprocesado* o renderización de agua y sombras en tiempo real. También se incluye la biblioteca de *shaders* generada para el videojuego.

APÉNDICE D. *ASSETS* EXTERNOS UTILIZADOS

Turbosquid [31]

- **Modelos *3D* base de los personajes principales (modificados para cambiar trajes)**
  3 modelos *3D* con 11 texturas.
- **Modelos *3D* y texturas de las guitarras**
  3 Modelos *3D* con 6 texturas.
- **Modelo *3D* de coche empleado en el nivel de conducción**
  1 Modelo *3D* con 8 texturas.
- **Modelos *3D* y texturas de vehículos militares destruidos**
  15 modelos *3D* con 45 texturas.
- **Modelos *3D* y texturas de vehículos destruidos**
  10 modelos *3D* con 30 texturas.
- **Modelos *3D* y texturas de araña (modificada para adaptar a necesidades específicas)**
  1 Modelo *3D* y 3 texturas.
- **Modelos *3D* y texturas de elementos de decoración de cocina (frigorífico, microondas, mesitas, etc)**
  20 modelos *3D* y 40 texturas.
- **Modelos *3D* y texturas de elementos de decoración de hospital (camas, sillas, carritos, etc)**
  30 modelos *3D* y 30 texturas.

3DModels & Textures [32]

- ***Pack* modelos *3D* y texturas de ambiente industriales**
  69 modelos *3D*, ~200 texturas.

- ***Pack* modelos *3D* y texturas de ciudad**
  34 modelos *3D*, ~100 texturas.
- ***Pack* modelos *3D* y texturas de carreteras y puentes**
  50 modelos *3D*, ~150 texturas.
- ***Pack* modelos *3D* y texturas de estatuas**
  18 modelos *3D*, 54 texturas.
- ***Pack* modelos *3D* y texturas de objetos de almacén como cajas, estanterías, mesas y carritos**
  20 modelos *3D* y ~50 texturas.

***3DRT* [33]**

- ***Pack* modelos *3D* y texturas de *zombies***
  6 Modelos *3D*, 20 texturas, 31 animaciones.
- ***Pack* modelos *3D* y texturas de *zombies* perros**
  2 Modelos *3D*, 10 texturas, 31 animaciones.
- ***Pack* modelos *3D* y texturas de militares**
  2 modelos *3D*, 6 texturas de *skin*, 86 animaciones.
- ***Pack* modelos *3D* motoristas y motocicletas**
  4 Modelos *3D*, 30 texturas, 28 animaciones.

***The3dstudio* [34]**

- **Modelo *3D* de un Taxi**
  1 *Modelo 3D* con 1 textura
- **Modelo *3D* de un Coche de policía**
  1 Modelo *3D* con 1 textura
- **Modelo *3D* de una Furgoneta**
  1 Modelo *3D* con 1 textura
- **Modelo *3D* de un vehículo todoterreno**
  1 Modelo *3D* con 1 textura

Vector Stock **[35]**

- **Siluetas utilizadas en las pantallas de carga y menús**
  11 vectores
- **Elementos utilizados para crear el interface de usuario de los diferentes menús**
  15 vectores
- **Imágenes utilizadas para representar cada logro**
  17 vectores

Opuzz **[36]**

- ***Pack*** **música *Hard Edge Action***
  10 Canciones
- ***Pack*** **música *Rock and Nu Metal***
  10 Canciones

Audiojungle **[37]**

- ***Pack*** **música *Energy Metal***
  6 canciones
- ***Pack*** **música *Hard Rock***
  6 canciones
- ***Pack*** **música *Nu Metal***
  6 canciones

Asset Store **de *«Unity3D»* [38]**

- **Efectos de partículas de magia 1**
  65 efectos
- **Efectos de partículas de magia 2**
  40 efectos
- **Efectos de partículas de líquido**
  40 efectos

- **Efectos de partículas de sangre**
  18 efectos
- **Efectos de partículas de fuego**
  3 efectos

APÉNDICE E. PROYECTOS PREVIOS REALIZADOS

En este anexo se describen los trabajos y experiencia previa del alumno. El objeto de añadir dicha información a este documento es dejar patente la necesidad de un *background* previo en realización de proyectos que ha permitido conseguir realizar un videojuego con los valores de producción como los que se han obtenido en este Trabajo de Fin de Grado.

A continuación se han seleccionado sólo los trabajos que están relacionados con videojuegos, programación gráfica, desarrollo en plataformas móviles o infografía. Quedando fuera otros proyectos que no tienen conexión directamente con las disciplinas necesarias para el desarrollo de un videojuego.

E.1. LEGEND LANGUAGE

Tareas : Programador / Diseñador de interface / Grafista.

Descripción : Videojuego que se realizó como practica para la asignatura "Procesadores de Lenguajes" en la *UCLM*. Se trata de un lenguaje para descripción de niveles en juego con mecánicas tipo *Diablo*.

Diseñe el lenguaje, el analizador léxico / sintáctico / semántico y la lógica del juego que permitía cargar niveles jugables especificados en dicho lenguaje.

Tecnología : *JFlex* / *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

```
LEVEL          ::== [TAGS] level ID "{" [SETTINGS] {NODE} "}"
ID             ::== LETTER { LETTER | INT }
LETTER         ::== a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y |
z | A | B | C | D | E | F | G | H | I | J | K | I | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
INT            ::== 0 | 1 {INT} | 2 {INT} | 3 {INT} | 4 {INT} | 5 {INT} | 6 {INT} | 7 {INT} | 8 {INT} | 9 {INT}
FLOAT          ::== INT "." INT
TAGS           ::== tags "{" ASSIGN { ASSIGN } "}"
ASSIGN         ::== CATEGORY ID "=" INT
CATEGORY       ::== climate | theme | type | rotation | time
SETTINGS       ::== settings "{" [VALUES] "}"
NODE           ::== BLOCK | ENEMY | OBJECT
BLOCK          ::== block ID "{" [VALUES] "}"
ENEMY          ::== enemy ID "{" [VALUES] "}"
OBJECT         ::== object ID "{" [VALUES] "}"
VALUES         ::== VALUES , VALUE | VALUE
VALUE          ::== CATEGORY_ASSIGN | LIFE | POSITION
CATEGORY_ASSIGN  ::== CATEGORY "=" (INT | ID)
POSITION         ::== position "=" "(" INT "," INT ")"
LIFE             ::== life "=" FLOAT
COMMENT        ::== /* Cualquier carácter ASCII */
```

Figura E.1: Gramática en notación EBNF

Figura E.2: Capturas *in-game* de niveles generados a partir de fichero fuente en lenguaje *Legend Language*

E.2. THE ESCAPE

Tareas : Programador / *Game designer* / *Level designer* / Gráficos.

Descripción : Juego de infiltración táctica desde la perspectiva de un preso que quiere escapar de la cárcel. El proyecto se canceló por falta de presupuesto para finalizarlo, aunque llegó al estado de prototipo funcional (Demo jugable).

Tecnología : *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

Figura E.3: Capturas *in-game* de *The Escape*

E.3. MAZE CITY

Tareas : Programador / *Game designer* / *Level designer* / Gráficos.

Descripción : Juego casual para móviles. Consiste en encontrar la salida a laberintos. Salió a la venta para *iOS*, *Android* y *OUYA*.

Tecnología : *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

Figura E.4: Capturas *in-game* de *Maze City*

E.4. CHRISTMAS SHOOTER

Tareas : Programador / *Game designer* / *Level designer* / Gráficos.

Descripción : Juego casual para móviles. Consiste en ir tirando regalos desde el trineo de papa noel. Dispone de integración con *GameCenter*, *Facebook* y *Twitter*. Salió a la venta para *iOS*.

Tecnología : *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

Figura E.5: Capturas *in-game* de *Christmas Shooter*

E.5. SPACE SOKOBAN

Tareas : Programador / *Game designer* / *Level designer* / Gráficos.

Descripción : Juego casual para móviles. Consiste en resolver puzzles de cajas. Salió a la venta para *iOS*, *Android* y *OUYA*.

Tecnología : *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

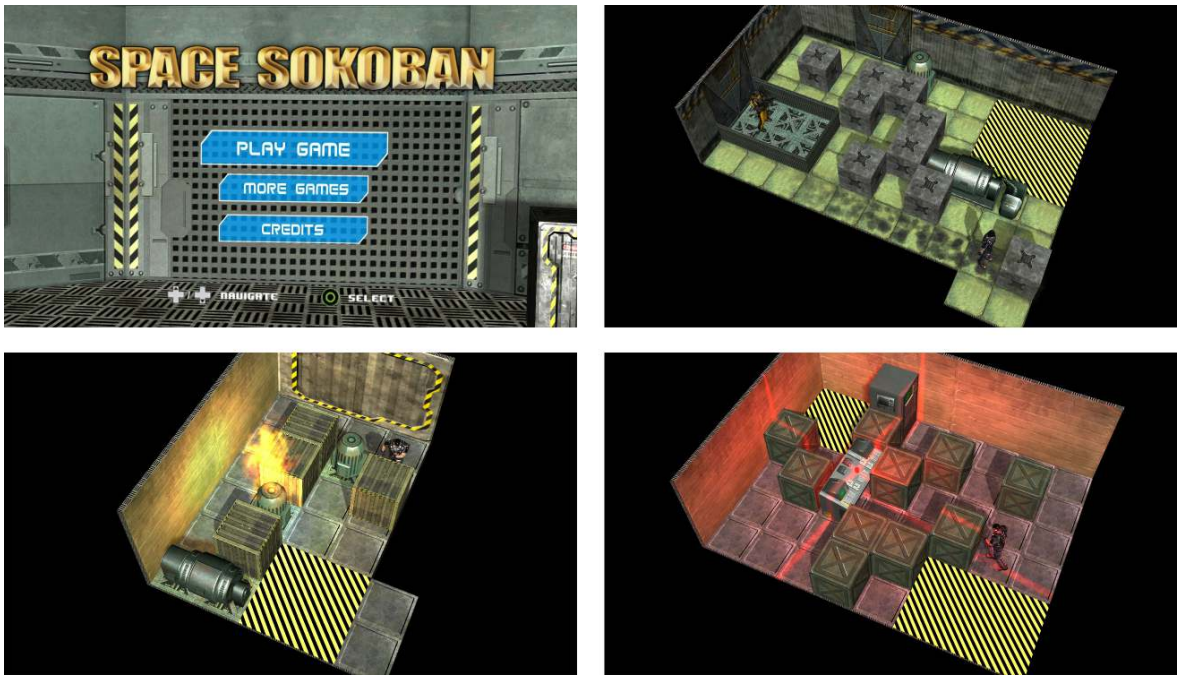


Figura E.6: Capturas *in-game* de *Space Sokoban*

E.6. SPITING BOY

Tareas : Programador / Game designer / Level designer / Gráficos. (En colaboración con socio fundador en la empresa propia *«Atomic Flavor SL»*).

Descripción : Juego inspirado en el clásico de *Beavis and Butthead*. El proyecto se canceló por falta de presupuesto para finalizarlo, aun así existen videos de *gameplay* y *demo* jugable.

Tecnología : *C#* / *Javascript* / *Shaderlab* / Motor gráfico *Unity3D*.

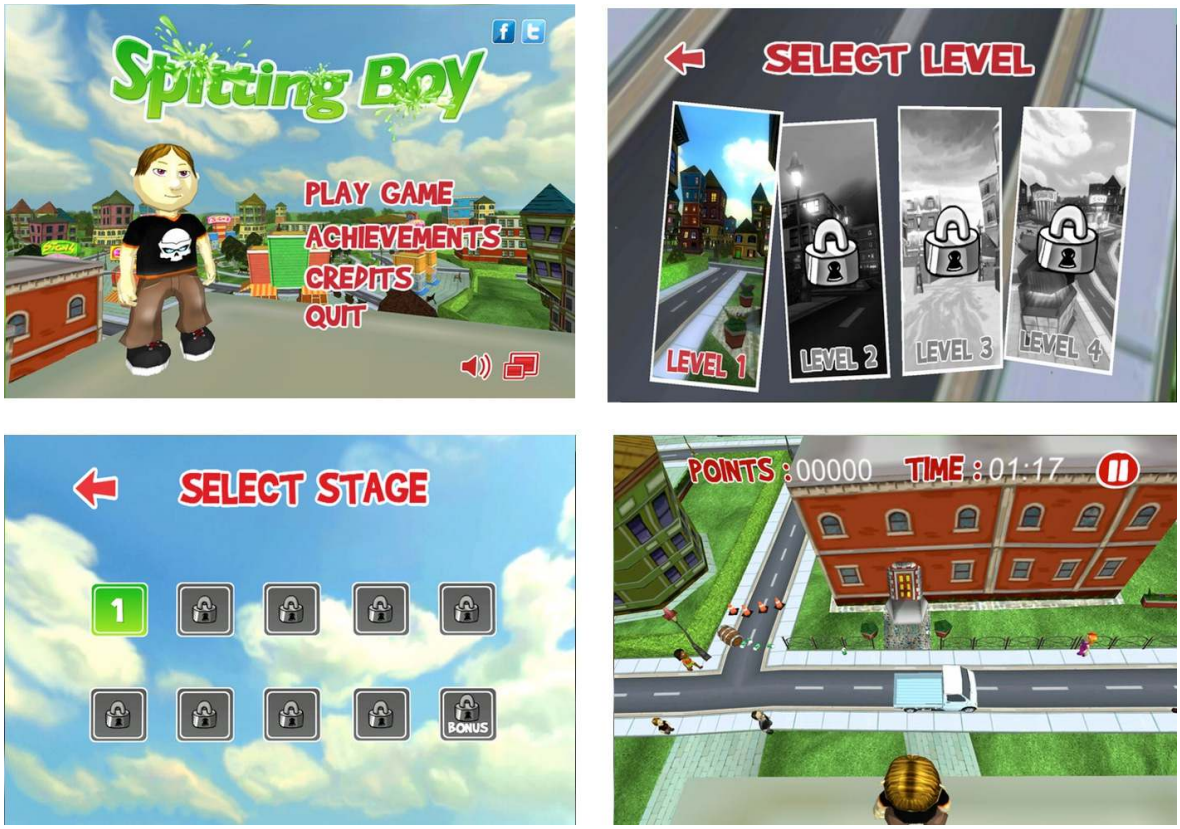


Figura E.7: Capturas *in-game* de *Spiting Boy*

E.7. OPERATION

Tareas : Programador. (En colaboración con socio fundador en la empresa propia *«Atomic Flavor SL»*).

Descripción : Juego casual para *tablets*. Te plantea el reto de realizar una operación a muñecos de tipo *toon*.

Tecnología : *Objective C* / *Cocoa* / *Quartz2D* / Motor gráfico propio.

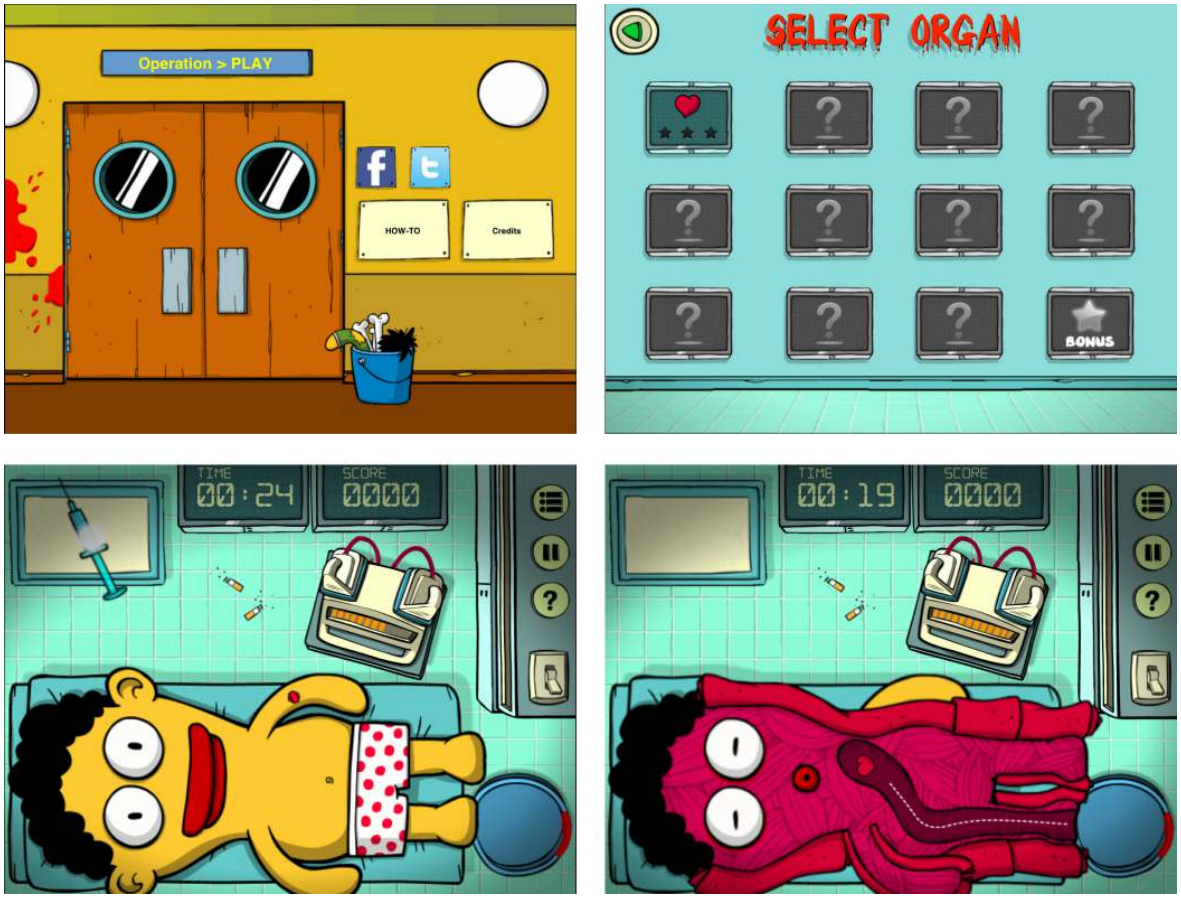


Figura E.8: Capturas *in-game* de *Operation*

E.8. THE BOX

Tareas : Programador / Diseñador de interface / Grafista. (En colaboración con socio fundador en la empresa propia *«Atomic Flavor SL»*).

Descripción : Juego casual para móviles. Un juego que combina reflejos y memoria. Salió a la venta para *iOS* (*iPhone e iPad*).

Tecnología : *Objective C* / *Cocoa* / *Quartz2D* / Motor gráfico propio.

Figura E.9: Capturas *in-game* de *The Box*

E.9. PHOTOPUZZLE

Tareas : Programador / Diseñador de interface / Grafista. (En colaboración con otros dos compañeros de la *«UCLM»*).

Descripción : Juego casual para móviles que consiste en resolver un puzzle de bloques deslizantes. Salió a la venta para *iOS* (*iPhone*).

Tecnología : *C++* / *OpenGL ES* / Motor gráfico propio.

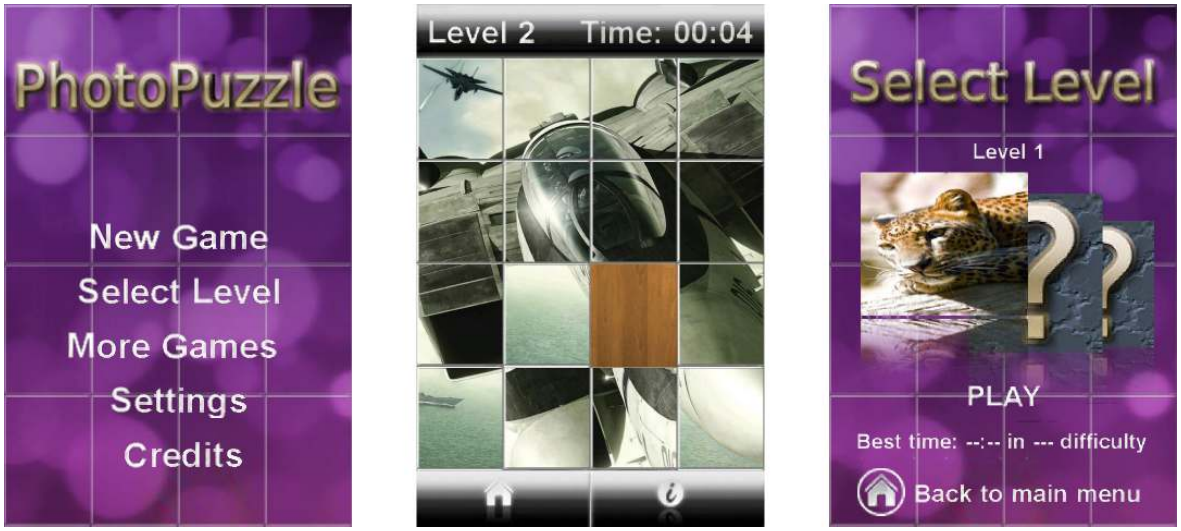


Figura E.10: Capturas *in-game* de *Photopuzzle*

E.10. APLICACIONES MÓVILES AUTO PUBLICADAS

Tareas : Programador / Diseño de interface / Grafista.

Descripción : Realización de más de 20 aplicaciones auto publicadas en la tiendas de venta de aplicaciones *App Store*.

Tecnología : *Objective C* / *Cocoa*.

Figura E.11: Capturas de diversas *Apps*

E.11. APLICACIÓN DE VISUALIZACIÓN MÉDICA

Tareas : Programador / Diseño de interface. (En colaboración con socio fundador en la empresa propia *«Atomic Flavor SL»*).

Descripción : Esta aplicación permite visualizar órganos recreados en *3D* y poder pintar anotaciones sobre ellos. Aplicación didáctica y de uso interno para médicos. Salió a la venta en la plataforma *iOS*(*iPad*).

Tecnología : *Objective C* / *Cocoa* / *C++* / *OpenGL ES* / Motor gráfico propio.

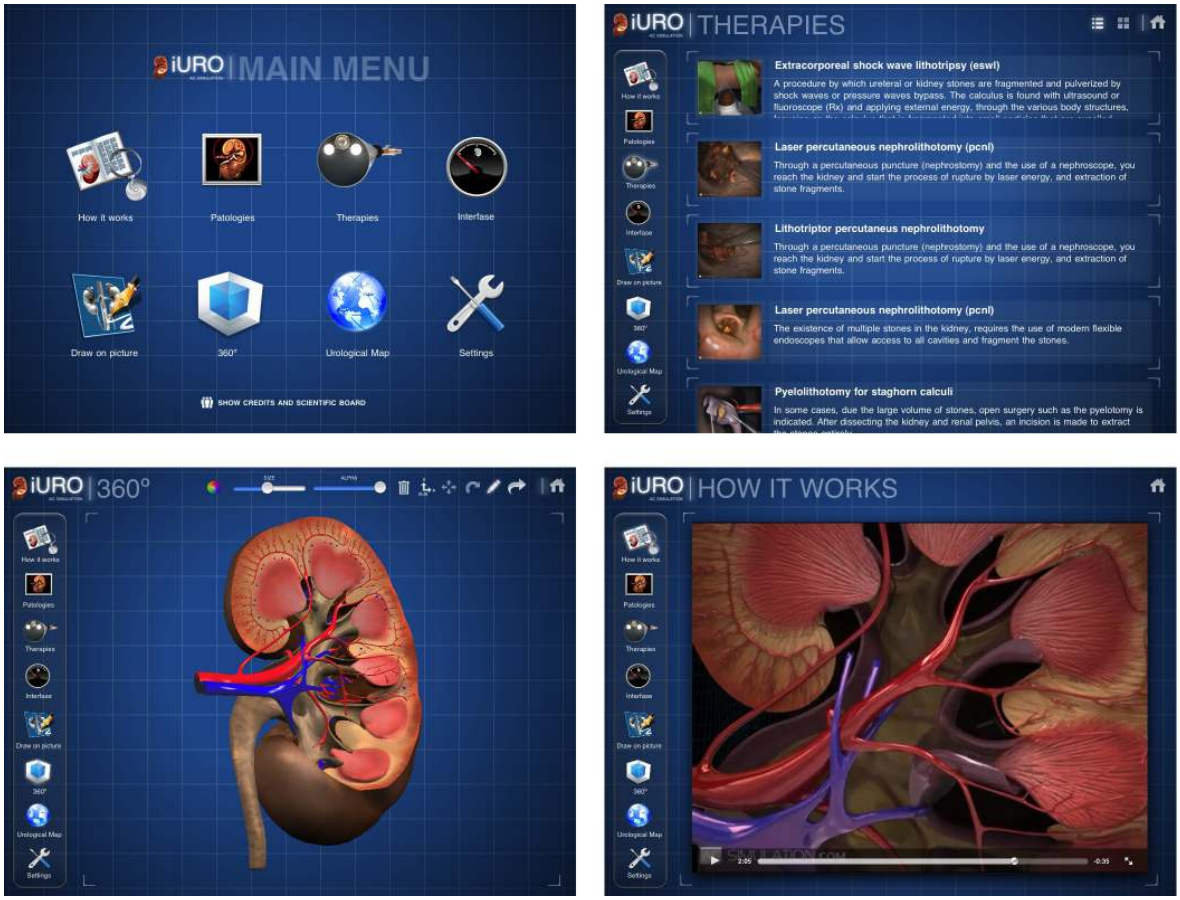


Figura E.12: Aplicación de visualización médica para *iPad*

E.12. APLICACIONES MÓVILES PARA EMPRESAS

Tareas : Programador / Diseño de interface / Grafista. (En colaboración con socio fundador en la empresa propia *«Atomic Flavor SL»*).

Descripción : Realización de más de 50 aplicaciones para *iOS*. Entre los clientes más destacados se hicieron aplicaciones para *«Santillana»*, *«Federopticos»*, *«Groupalia»*, *«Presidencia de Aragón»*, *«Adif»*, *«Banco Santander»* o *«Games Tribune Magazine»*.

Tecnología : *Objective C* / *Cocoa*.

Figura E.13: Capturas de diversas *Apps*

E.13. MUSEO VIRTUAL DE LA INFORMÁTICA

Tareas : Programador / Diseñador de interface. (En colaboración con un director de proyecto y dos compañeros de la *«UCLM»*).

Descripción : Desarrollo de la aplicación del punto de información del museo de la informática de la Escuela Superior de Informática en Ciudad Real. Para este proyecto desarrollé el motor gráfico en *C++* y *OpenGL*, así como los formatos de fichero y *plugins* para programas externos que permiten extender el contenido del museo sin recompilar el código. Se presentó al concurso europeo de proyectos de software libre *«Europrix»* y quedamos entre los finalistas, obteniendo el galardón *quality seal*.

Tecnología : *C++* / *OpenGL ES* / Motor gráfico propio.

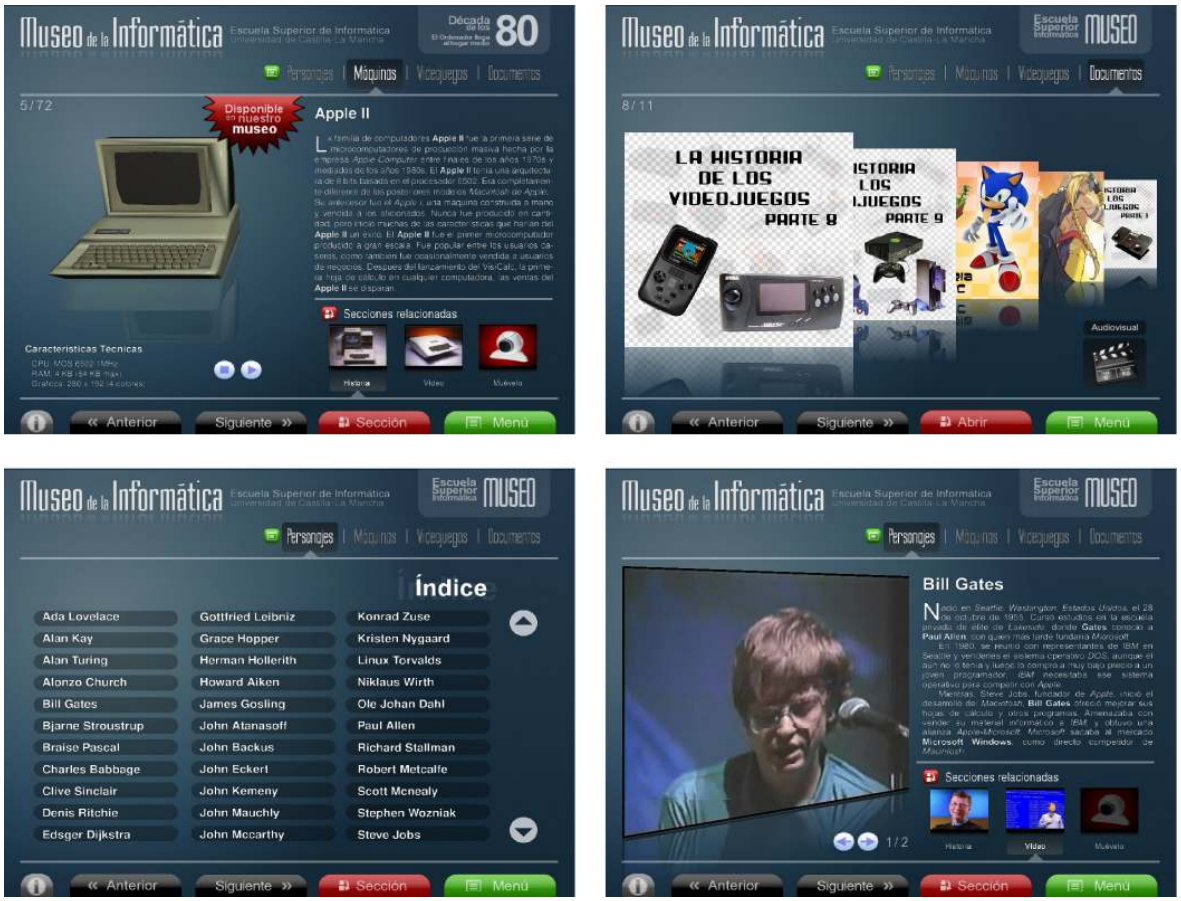


Figura E.14: Capturas de la aplicación del Museo Virtual de la informática

E.14. VISITAS VIRTUALES

Tareas : Modelado / Texturizado / *Rendering*. (En colaboración con otros dos compañeros de la *«UCLM»* y el director del proyecto).

Descripción : Trabajé como artista *3D* para los videos de "Visita Virtual al Hospital de Ciudad Real", "Visita Virtual al servicio de Radioterapia" y "Visita Virtual al servicio de Nefrología".

Tecnología : *Blender*, *Gimp*, *Yafray*, y la granja de *render* del Servicio de Supercomputación de la Universidad de Castilla-La Mancha.

Figura E.15: Capturas de "Visita Virtual al servicio de Nefrología"

E.15. INFOARQUITECTURA

Tareas : Modelado / Texturizado / *Rendering*. (En colaboración con otros dos compañeros de la *«UCLM»*).

Descripción : Realización de trabajos de infoarquitectura para inmobiliarias y realización de sintesis digital de edificios y pisos pilotos.

Tecnología : *Blender* y *Yafray*.

Figura E.16: *Renders* e integración con imágen real de varios proyectos realizados para imobiliarias