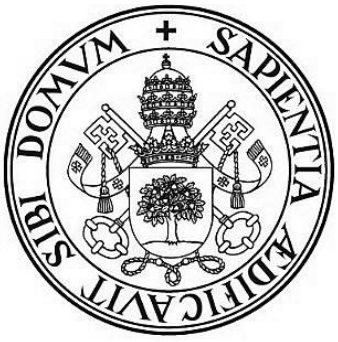


Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Tecnologías de la Información

Aplicación del juego La Colmena mediante Unreal Engine 4

Autor:

D. Rubén Moya Vázquez

Tutor:

Dña. Margarita Gonzalo Tasis

Agradecimientos

Quisiera agradecer el cariño, apoyo y comprensión a todos los que han tenido que lidiar conmigo desde el comienzo del proceso que culmina con este trabajo.

A mis abuelos, Carmen y Telmo, las raíces que me han hecho crecer y me han mantenido firme. A Sandra, mi pareja, el sol del que dependo y marca mis días. A mi madre y su marido, Almudena y Rubén, el tronco y el injerto sin los cuales este árbol nunca podría dar frutos. A mi hermano Mario, el retoño de lo que llegará a ser un árbol aún más grande.

Por último, agradezco a mi tutora, Margarita, la comprensión, apoyo, interés y entusiasmo que ha aportado al proyecto desde su germen, en una charla tras la corrección de un examen de IPC.

Resumen

El mercado del ocio digital es actualmente uno de los sectores laborales relacionados con la ingeniería informática que más volumen de beneficios tiene y más rápido crece. Según el Anuario de 2016 de la *AEVI* (*Agencia Española de Videojuegos*) [54] [85] se estima que este crecimiento sigue un ritmo del 7.5% anual y solo en España factura más de 1.000 millones de euros al año, el doble que la industria cinematográfica.

Teniendo en cuenta esta coyuntura, se presenta este TFG como porfolio de habilidades y conocimientos adquiridos, además de como presentación del aprendizaje realizado en el uso del motor de videojuegos más empleado del sector, Unreal Engine.

Abstract

Digital leisure market is, nowadays, one of the top growing and largest in terms of profit, labour sectors related to the Computer Sciences. According to the AEVI's 2016 yearbook (Videogames Spanish Agency) [54] [85] it is estimated that this growth continued at a rapid pace of 7.5% per year and just in Spain, the invoicing volume generated surpassed 1000 million euros, twice than the film industry.

Considering that situation, we submit this final thesis as a portfolio of evidence showing the competence of skills gained through training and the learning process done in the videogame industry's top game engine, Unreal Engine 4.

Índice

Índice de Ilustraciones

Índice de Figuras

Índice de Tablas

1 Introducción

En este primer capítulo realizaremos un breve preámbulo al contenido del resto de la memoria y el trabajo que ésta representa.

Describiremos las motivaciones que nos han llevado a realizar dicho proyecto, los conocimientos previos con los que partimos y situaremos el alcance del proyecto a partir de una serie de objetivos a perseguir.

Posteriormente realizaremos una descripción, a modo de marco conceptual, del producto final, el estado en el que se encuentra la industria del entretenimiento digital y las herramientas utilizadas más frecuentemente.

1.1 Motivaciones

El ocio y la expresión artística son dos aspectos fundamentales que describen la psicología humana. Desde sus orígenes, el hombre ha desarrollado multitud de formas de expresarse y divertirse conforme la tecnología le iba otorgando nuevas herramientas para ello. Al igual que las cámaras fotográficas y el cinematógrafo nos trajeron el cine y la fotografía, la informática dio lugar al entretenimiento digital.

Este avance brindó la posibilidad de aunar características propias de diversas actividades en un único producto; siendo así el videojuego un componente software con finalidad lúdica que a la vez puede servir como medio de expresión artística, para contar historias o como plataforma educativa o terapéutica.

Teniendo en cuenta el potencial que alberga el videojuego como producto [10] [11] y como medio de expresión y persiguiendo el objetivo de aprender el proceso de ingeniería que subyace en las técnicas aplicadas por la industria del ocio en su desarrollo; hemos decidido adaptar un juego de mesa existente a videojuego usando las mismas herramientas que se usan actualmente en dicha industria.

1.2 Objetivos

Con la finalidad de enfocar de forma más efectiva tanto el proceso de aprendizaje de las nuevas herramientas como la implementación del producto final, nos hemos planteado una serie de objetivos a perseguir. Estos hitos nos servirán tanto para planificar y subdividir en tareas todo el proceso como para medir el grado de conformidad final con el software desarrollado.

Mientras que algunos de los siguientes puntos son fundamentales para la correcta consecución del proyecto y pueden considerarse metas, otros son aspectos deseables o funcionalidades a desarrollar como mejora en posteriores iteraciones.

- Comprender el proceso de producción de un videojuego usando un motor de desarrollo.
- Adaptar las técnicas de ingeniería de software aprendidas a lo largo de la carrera al proceso de desarrollo de un videojuego [20].
- Conocer, comprender y aprender a utilizar las herramientas que nos brinda un motor de desarrollo de videojuegos.
- Analizar las mecánicas, piezas y situaciones del juego a desarrollar.
- Gestionar el tablero, sistema de turnos y piezas en juego de cada jugador.
- Analizar productos similares en busca de factores comunes, puntos que potenciar y otros que se deberían evitar.
- Conocer y comprender el uso y peculiaridades del framework de *C++* usada para programar en *Unreal Engine*.
- Conocer y comprender el sistema *Blueprint* de visual scripting incluido en *Unreal Engine*.
- Comprender el sistema de gestión de modos de juego, niveles y menús de *Unreal Engine*.
- Familiarizarse con el sistema de físicas, materiales y partículas que conforman el soporte de modelado gráfico de *Unreal Engine*.
- Familiarizarse con el soporte para el desarrollo de videojuegos multiplataforma y multiregión.
- Aprender y aplicar conceptos de gestión de proyectos a la hora de tomar decisiones y fijar plazos.

1.3 Conocimientos Previos

1.3.1 Videojuego

Un videojuego [14] [18] es una pieza de software mediante la que uno o varios usuarios interactúan, por medio de un controlador, obteniendo una respuesta visual, generalmente de video. La finalidad principal de un videojuego es el ocio, aunque cada vez con más frecuencia, puede usarse con finalidades pedagógicas o terapéuticas.

Un videojuego puede ejecutarse en una gran variedad de soportes, que van desde computadoras, videoconsolas o sistemas móviles hasta sistemas de realidad virtual o realidad aumentada como las gafas *Oculus Rift* [52] de *Facebook* o el set *PlayStation VR* [51] de *Sony*.

1.3.2 Juego de Mesa

Un juego de mesa es una actividad en el que intervienen dos o más jugadores y que generalmente se practica en torno a una mesa o tablero. El juego se conforma de un conjunto de reglas, una situación inicial, una serie de piezas, cartas o fichas y una situación final en la que uno, varios o ninguno de los jugadores queda victorioso.

Un juego de mesa suele enfrentar las habilidades de los jugadores, ya sea su capacidad deductiva, su razonamiento estratégico o destreza manual, pero también existen aquellos que se basan puramente en el azar.

1.3.3 Motor de videojuegos

Un motor de videojuegos es una colección de rutinas de programación y modelado que permiten el diseño implementación y representación de un videojuego. La función de un motor de videojuego es dar soporte al desarrollo del mismo proveyendo al desarrollador de diversos componentes como:

- Motor de renderizado para gráficos 2D y 3D.
- Motor físico.
- Gestor de medios audiovisuales
- Soporte para animación
- Soporte para inteligencia artificial
- Gestor de redes
- Administrador de memoria

El motor de videojuegos brinda la posibilidad de gestionar, reutilizar y adaptar componentes ya creados para poder incluirlos en otros videojuegos o desarrollar para distintas plataformas de forma eficiente.

1.4 Organización

Con el objetivo de situar al lector, realizaremos una breve descripción de los capítulos de los que se compone este trabajo.

- Capítulo 1: Introducción. Se presentan los objetivos y motivaciones que han incitado a la realización de este proyecto.
- Capítulo 2: Estado del Arte. Se sitúa el proyecto realizado en su contexto histórico, teórico y técnico.
- Capítulo 3: Documento de Diseño del Videojuego. Se describen las principales características del juego a desarrollar, incluyendo bocetos del mismo.
- Capítulo 4: Desarrollo del Proyecto. Se plantea el proceso de desarrollo que seguiremos y se plantean los documentos de gestión de riesgos y planificación.
- Capítulo 5: Análisis. Se detallan los requisitos funcionales y casos de uso, incluyendo la versión de análisis del diagrama de clases y los principales diagramas de secuencia.
- Capítulo 6: Diseño. Se describe la arquitectura lógica seguida para el desarrollo del proyecto, así como los patrones utilizados; todo ello apoyado los diagramas pertinentes.
- Capítulo 7: Implementación. Se describen los aspectos más destacables del proceso de implementación.
- Capítulo 8: Pruebas. Se presentan las pruebas realizadas al software, así como sus resultados.
- Capítulo 9: Manuales. Se detallan los manuales de instalación, administración y uso del proyecto desarrollado.
- Capítulo 10: Conclusiones. Valoración personal del proyecto y la consecución de los objetivos por parte del equipo de desarrollo, además de describirse posibles líneas de desarrollo a futuro con el objetivo mejorar el producto.

2 Estado del Arte

El objetivo de este capítulo es aportar un marco teórico y conceptual al resto de la memoria. Para ello, describiremos la evolución de la industria del entretenimiento digital, su origen, principales hitos, estado actual y previsiones de futuro. Además, detallaremos el papel que juega el motor de videojuegos en la creación del producto final, los distintos motores que hay y las ventajas e inconvenientes que representa el motor seleccionado frente al resto de los competidores.

Finalmente, analizaremos los principales conceptos arquitectónicos, herramientas y *plugins* de Unreal Engine que usaremos durante realización de nuestro proyecto.

2.1 La Industria del Videojuego. Orígenes y evolución

2.1.1 Orígenes

El primer juego electrónico del que se tiene constancia, *Cathode-ray tube amusement device* [25], data de 1947. Dicho juego, creado por Thomas T. Goldsmith Jr. y Estle Ray Man, era un simulador de lanzamiento de misiles basado en el sistema de radar utilizado en la Segunda Guerra Mundial. Para su funcionamiento se utilizaba un tubo de rayos catódicos y una serie de circuitos analógicos que controlaban tanto el haz de rayos como el punto que hacía las veces de objetivo.

Debido a su carácter analógico, diversas opiniones no consideran que *Cathode-ray tube amusement device* fuese el primer videojuego. En su lugar, otorgan dicho título a *OXO* (Alexander S. Douglas, 1952), juego basado

en el popular “tres en raya”. *OXO* [79] se caracteriza por ser el primer juego en utilizar una pantalla grafica digital y por estar programado para la computadora *EDSAC*.

Un año antes, en 1951, un ingeniero alemán llamado *Ralph Baer* que trabajaba como técnico de televisores indagó en la posibilidad de conseguir que los aparatos de televisión pudiesen ser más interactivos. No sería hasta 1968 cuando crease el primer prototipo de videoconsola de la historia. La “*Brown box*” [24], como llaman los coleccionistas al prototipo original, acabó dando lugar en enero de 1972 a la que sería la primera videoconsola en salir al mercado, la *Magnabox Odyssey*. En un año, pese a que solo se vendiese en los almacenes de la marca *Magnabox* y que los vendedores asegurasen que solo funcionaría con televisores de su marca, consiguió vender 100.000 unidades a un precio de 100 dólares. La *Magnabox Odyssey* era una consola sin cpu, ram, sistema de almacenamiento y cuyo catálogo final llegó únicamente a los 28 videojuegos; pero pese a todas sus pegas consiguió generar la llamada “*Locura del Pong*” que obligo a que otras marcas iniciasen la carrera por tener su propia videoconsola. Debido a esto, se considera a *Ralph Baer* el padre de los videojuegos, ya que originó el mercado del ocio digital doméstico.

Figura 1. Magnabox Odyssey y controlador.

El mismo año que dio a luz a la primera videoconsola también trajo, aunque con menos repercusión, la primera máquina recreativa. Bill Pits, un estudiante de la Universidad de Stanford, que había descubierto el juego *Spacewar!* recientemente, se alió con Hugh Tuck, ingeniero mecánico, para crear la primera máquina operada con monedas que reprodujese el juego anteriormente mencionado. La PDP-11, maquina resultante de dicho proceso de investigación, salió al mercado con el irrisorio precio de 20.000 dólares, lo que unido a sus escasos beneficios (cada partida costaba 10 centavos) provocaron que el proyecto fracasase. Aun así, este fracaso abrió a su vez el mercado para las conocidas maquinas *arcade* que acabarían teniendo su momento de gloria en la década de 1980.

Además de por iniciar el mercado del ocio digital doméstico y anticipar los posteriores salones de máquinas recreativas, 1972 está marcado en el calendario de la industria de los videojuegos por haber sido el año en el que apareció el primer juego multiplataforma en comercializarse de manera masiva. Dicho juego es el archiconocido *Pong*, creado por Nolan Bushnell y publicado por su compañía, *Atari* [27] en noviembre de 1972. El juego, basado en el ping pong, consistía en dos líneas verticales, a modo de raquetas, que eran controladas por los jugadores y una línea punteada, a modo de red, que dividía la pantalla en dos. Permitía jugar tanto contra un oponente controlado por la maquina como contra otro jugador y el objetivo final era ser el jugador con más puntos acumulados al acabar el tiempo de partida. Si bien habíamos dicho anteriormente que fue la consola de Magnabox la responsable del inicio de la conocida como “*Locura del Pong*” con juegos de hockey y tenis, fue el juego de Atari el que consiguió tener una mayor relevancia final gracias a no depender de una única plataforma. Este hecho desagradó profundamente a la marca de televisores, que acabaría ganado una demanda por infracción de patente

frente a la empresa de Bushnell, por la que Atari acabaría pagando 700.000 dólares a Magnavox. Cabe destacar que, pese a que la multa pueda parecer alta para una industria tan reciente, se estima que a finales de 1974 había, solo en Estados Unidos, en torno a 100.000 máquinas recreativas que generaban un beneficio a Atari de unos 250 millones de dólares anuales.

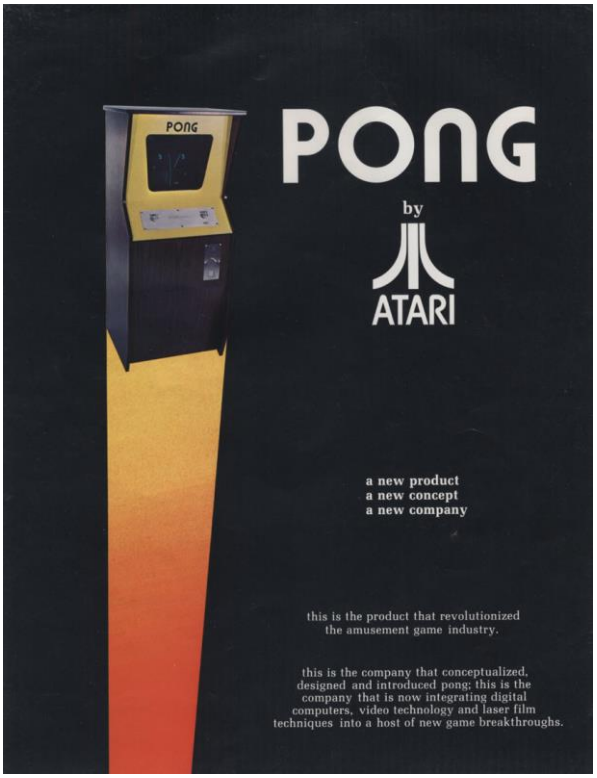


Figura 2. Anuncio de la máquina recreativa Pong de Atari.

2.1.2 La Edad de Oro de los videojuegos

La industria de los videojuegos [19] había empezado a funcionar con un claro líder, Atari. Gracias a juegos como *Rebound*, *Night Driver* o *Space Race* junto con su nueva consola, la Atari 2600, la compañía se había colocado claramente como líder del incipiente mercado del ocio digital. Este hecho, unido a la aparición de *Space Invaders* en verano de 1978 de la mano de *Taito Corporation*, fue el desencadenante de lo que comúnmente se conoce como "La Edad de Oro de los Videojuegos" **[23]** (1978-1983). En lo que duró este periodo, los beneficios se multiplicaron mensualmente una media de un 5%, pasando de los 450 millones de dólares de facturación con los que se cerró 1978 a los más de 5.300 millones de dólares que se llegarían a generar en 1982.

El creciente interés del público por los videojuegos unido a unas cifras que no dejaban de crecer provocó que muchas compañías relacionadas con el sector del entretenimiento creasen sus propias divisiones de videojuegos. Fue en este momento cuando *LucasArts* [29] (*LucasFilms*), *Fox Interactive* (*20th Century Fox*) y filiales de *Parker Brothers* o *Walt Disney Pictures* empezaron a despuntar como estudios líderes en la creación de videojuegos.

La explosión de interés que estaban generando los videojuegos se podía explicar en gran medida por dos enormes mejoras en el apartado estético: los colores y los gráficos vectoriales. Estos avances permitieron crear videojuegos más complejos y estéticamente más llamativos que los jugadores acogían con mucho entusiasmo. El

primer juego en introducir colores en sus gráficos fue *Galaxian* de *Namco*, que pese a seguir en consonancia con su predecesor, *Space Invaders*, sembró el germen de una nueva evolución en el sector.

Fue en esta época cuando Töru Iwatani creó el juego conocido por ser el más popular de la historia, Pac-man. La idea inicial de Iwatani cuando desarrolló Pac-man fue la de crear un juego de laberintos no basado en la violencia y con un protagonista que no fuera necesariamente masculino. En aquella época, la cultura de los videojuegos era mayoritariamente masculina y centrada en juegos de acción, por eso Iwatani quiso crear un juego para todos los públicos basado puramente en comer. Para los villanos, Iwatani recicló unos personajes de uno de sus anteriores juegos, Cutie Q. Pac-man fue portado y clonado repetidas veces y en todas las plataformas disponibles en la época; además, fue el juego responsable de iniciar la industria del merchandising de videojuegos.

Figura 3. Merchandising original de Pac-man.

Con el fin de la época dorada de los videojuegos también terminó la hegemonía de Atari. Mientras que la compañía de Bushnell seguía siendo líder en cifras, se había quedado atrás en cuanto a avances se refería. Esa fisura la aprovechó Hiroshi Yamauchi, dueño de la juguetera *Nintendo* [26], con su primera versión de la popular *Famicom*.

La *Famicom* [37] fue una consola revolucionaria en muchos aspectos y supuso un éxito a tal nivel que Nintendo se vio obligada en generar un sistema de licencias a terceros para que estudios de videojuegos externos desarrollasen para su consola, siguiendo unos estándares de calidad, debido a una demanda de juegos que la compañía no podía cubrir por sí misma. *Nintendo* se reservaba el derecho a decidir que juegos se publicaban y cuáles no, según el nivel de adecuación a sus estándares. Esta idea revolucionaria supuso un éxito rotundo y gracias a ella juegos como *Super Mario Bros., The Legend of Zelda, Dragon Quest* o *Final Fantasy* se unieron al catálogo de éxitos que encabezaba *Donkey Kong*.

Nintendo [13] había vuelto a abrir la puerta del mercado de los videojuegos a las empresas japonesas y compañías como *Capcom* (*Megaman, Ghosts'n Goblins y Cadillacs & Dinosaurs*), *Konami* (*Frogger y Contra*) o *Sega* (*Altered Beast y Golden Axe*) se situaron entre las más importantes del sector.

Figura 4. Famicom con juego de Super Mario Bros.

Por otro lado, Europa, que hasta la fecha se encontraba en el mercado del ocio digital únicamente como consumidor, vio como empezaban a surgir desarrolladores autóctonos centrados en el sector de las videoconsolas y los ordenadores personales. En concreto uno de los más relevantes fue el británico *Clive Sinclair*, quien en 1980 sacó al mercado el *Sinclair ZX80*. Un año después, con la llegada de su versión mejorada, el *ZX81*, que contaba con una memoria ampliable hasta 64Kb y un precio muy reducido, el mercado del ocio domestico europeo dio finalmente la bienvenida a los ordenadores personales. La siguiente versión del pc de *Sinclair*, el *ZX Spectrum*, llego al mercado en 1982 arrasando con sus competidores.

Las posibilidades que ofrecía este hardware motivaron la aparición de juegos que supusieron la creación de nuevos géneros de videojuegos como es el caso de *Football Manager* (1982, *Kevin Toms*) que consistiendo en un gestor de equipos de futbol en modo texto dio origen al género de los simuladores de gestión deportiva o el *Manic Miner* (1983, *Matthew Smith*), que definió los juegos de plataformas. En 1983 llegaría al *ZX Spectrum* otro juego histórico por su carácter innovador, el *Knight Lore* de *Ultimate (*la actual *Rare Ltd.)*. Este juego pasaría a la historia por ser el primer videojuego para el *ZX Spectrum* en perspectiva isométrica, lo cual encantó a la crítica y redundó en un tremendo éxito de ventas. Posteriormente la técnica *Filmation* (o *Filmation engine*) usada en la creación de la mencionada perspectiva 3D sería replicada en otros videojuegos como *Alien 8* (1985, *Ultimate*), *Batman* (1986, *Ocean Software*) o *Head Over Heels* (1987, *Ocean Software*).

En 1984, *Alan Michael Sugar*, saco al mercado el *Amstrad CPC 464*, que no solo contaba con un hardware con mejores características que sus competidores si no que, debido a que se había estado desarrollando juegos para prototipos anteriores de la máquina, salió a la venta con un catálogo inicial de 50 juegos. Esto, unido a que el pack de lanzamiento incluía todos los periféricos necesarios para su funcionamiento, supuso que fuese un éxito en ventas tan grande que la compañía *Amstrad* acabaría comprando *Sinclair Research* dos años más tarde.

En términos de desarrollo de software, los 80s fueron una época de expansión y creatividad para la industria del entretenimiento europea. El mercado estaba liderado indiscutiblemente por estudios y distribuidores británicos como *Ocean* (*Batman*, *Head Over Heels* y *Operation Wolf*), *Imagine* (*Yie Ar Kung-Fu, Hyper Sports* y *M.O.V.I.E.*), Hewson Consultants (*Ranarama, Uridium* y *Nebulus*), *Elite* (*Commando, Ghost'n Goblins* y *1942*) o *U.S. Gold (World Cup USA '94, Winter Games* y *Street Fighter*) pero en otros países del sur de Europa comenzaron a germinar estudios con gran potencial.

2.1.2.1 Edad de Oro del Software Español

En España todo empezó con el juego *"Bugaboo (The Flea)"* de *Indescomp*, publicado en 1983 en Gran Bretaña, que llegaría a nuestro país posteriormente con el nombre *La Pulga*. El juego en cuestión comenzaba con una animación en la que se veía caer a una pulga al fondo de una grieta y el jugador debía guiar a dicha pulga de vuelta a la superficie sorteando los peligros que se encontrase a su paso. Tuvo una tremenda acogida en el mercado europeo y sirvió como precedente de lo que acabaría siendo *"La Edad de Oro del Sofware Español"*.

Siguiendo el ejemplo de *Indescomp* (*La Pulga* y *Fred*), otros estudios como *Dinamic* (*Abu Simbel*, *Profanation* y *Camelot Warriors*), *Opera Soft* (*Livingstone supongo*, *Goody* y *La abadía del crimen*), *Topo Soft* (*Survivor*, *Mad Mix Game* y *Emilio Butragueño Fútbol*), *Erbe Software* (*Las tres luces de Glaurung* y *Whopper Chase*) o *Zigurat* (*Carlos Sainz. Campeonato del Mundo de Rallies*, *Curro Jimenez* y *El misterio del Nilo*) comenzaron a hacer videojuegos de gran calidad que recibían una gran acogida por parte del público internacional.

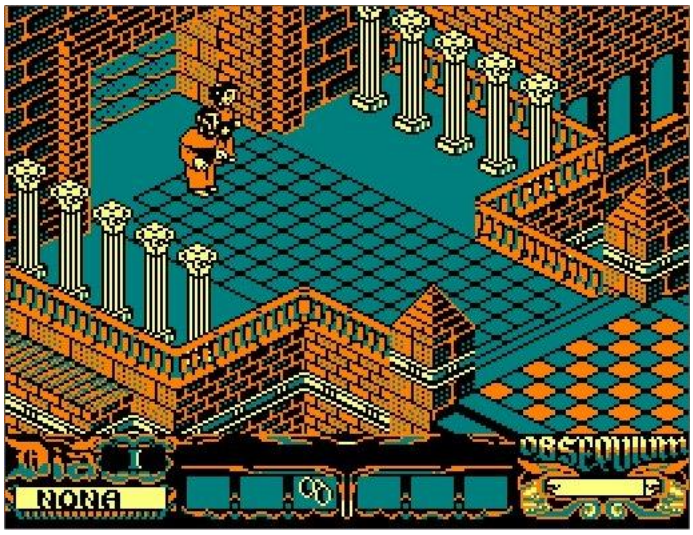


Figura 5. La abadía del crimen para Amstrad CPC.

Cabe destacar en particular La abadía del crimen por ser una obra de culto de la época dorada del software español. Basado en el libro de *Umberto Eco El nombre de la rosa*, el juego nunca llegó a tener éxito económico, pero aun así es considerado uno de los mejores juegos de la época de los 8bits. El juego, que cuenta con una perspectiva isométrica, consiste en averiguar quién es el responsable de una serie de crímenes cometidos en la abadía poniéndonos en la piel del monje Guillermo de Occam. Para ello se debía realizar las tareas que el abad nos encomendaba a la vez que se iba resolviendo una serie de puzles que nos iban acercando a nuestro objetivo. Finalmente, el juego contaba con un límite de 7 días, en tiempo dentro del juego, conforme al cual se otorgaba una puntuación final una vez superado.

El juego se versionó para *MSX, PC, Amstrad CPC* y *Spectrum 128*, siendo considerado uno de los 10 mejores juegos realizados para esta última plataforma.

La década de los 80s terminaría con el duelo entre la *NES* (*Nintendo Entertainment System*, nombre con el que se rebautizó a la *Famicom* en EEUU) y la *Master System* de *Sega* por hacerse con el control de los mercados occidentales [12] tras la debacle de *Atari*. Para la industria española, el fin de década supuso a su vez el fin de la edad de oro, dado que la llegada de las consolas de 16 bits supuso la debacle de las compañías españolas, que no supieron adaptarse al cambio.

2.1.3 Las máquinas de 16 bits y la guerra de las consolas

Para la industria de los videojuegos, la década de los 90s supuso la consolidación definitiva del sector. Las máquinas de 8 bits dieron paso por fin a computadoras más avanzadas que ofrecían mucha más potencia a los desarrolladores. Contrariamente a lo que cabe esperar, los 90 también trajeron, al menos en sus primeros años, un estancamiento en la creatividad de los estudios de desarrollo.

Por otra parte, dicho estancamiento supuso a su vez un renacimiento del genero de las aventuras gráficas [28], también conocido como *point-and-click* por su mecánica de interacción, liderado por *LucasArts*, filial interactiva de *Lucasfilm*. En juegos como *Secret of Monkey Island* (1990), Indiana *Jones and the Fate of the Atlantis* (1992) y *Maniac Mansion: Day of Tentacle*, LucasArts aplicaba técnicas cinematográficas con la intención de crear películas interactivas. El estilo narrativo unido a la simplicidad del menú de acciones *SCUMM* [30] (*Script Creation Utility for Maniac Mansion*) creado unos años antes para el juego *Maniac Mansion* provocó una gran acogida por parte del público.

Viendo los resultados tan positivos que LucasArts estaba obteniendo, otros estudios se lanzaron a crear sus propias aventuras gráficas. Entre las más destacadas podemos encontrar *Another World* (1991), del estudio francés *Delphine Software* o *Simon the Sorcerer* (1993) distribuido por la española *Erbe Software*.

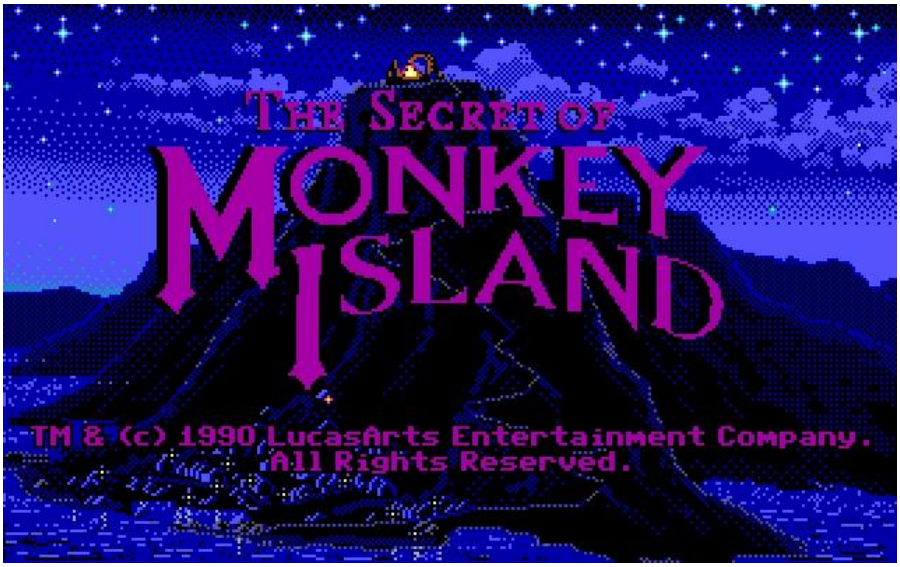


Figura 6. The Secret of Monkey Island de LucasArts para PC.

Este generó alcanzaría su máximo exponente a finales de los 90s gracias a la aplicación de las últimas técnicas de animación, el doblaje de los juegos y las secuencias cinemáticas. De esta generación son obras como *Broken Sword: La leyenda de los templarios* (1996, *Revolution Software*), *Hollywood Monsters* (1997, *Péndulo Studios* [32]) o *Grim Fandango* (1998, *LucasArts*). Posteriormente, la burbuja de las aventuras graficas acabaría pinchándose por culpa de una falta de innovación que acabo aburriendo a los jugadores.

Junto con el género de las aventuras gráficas, los 90s fueron la década en la que se definieron otros dos géneros, la simulación y la estrategia. Todo empezó en 1989 con dos desarrolladores, *Will Wright* y *Peter Molyneux*, que por separado sentaron los precedentes de ambos géneros al publicar ese año *SimCity* y *Populous*, respectivamente. Mientras que ambos juegos cuentan con una perspectiva cenital que le da al jugador la sensación de omnipresencia divina; el objetivo final y la lógica interna de ambos juegos los separa en géneros distintos.

Por un lado, *SimCity* cimentó los pilares de los simuladores de gestión urbanística aplicando funcionalidades de los editores de imágenes y principios del interfaz de *Apple* a la idea de crear ciudades. El objetivo del juego era desarrollar una ciudad perfecta en términos utópicos superando por el camino las diferentes complicaciones que se planteasen a través de la planificación y la toma de decisiones.

Populous, por el contrario, es considerado el primer juego del subgénero estratégico de la simulación divina. La finalidad del juego era ponerse en la piel de una deidad que guía a su pueblo con el objetivo de proliferar y eliminar a los seguidores de otras divinidades. Para ello, el jugador contaba con una serie de acciones clasificables en, acciones de construcción, intervenciones divinas y reclutamiento de aldeanos. Fue un juego aclamado tanto por la crítica como por el público y destaca además por ser uno de los primeros con una historia no lineal que tiene en cuenta la experiencia del usuario.

Figura 7. Captura de pantalla del juego Populous.

Dos años después, en 1991, aparecería de la mano del desarrollador *Sid Meier*, el juego *Civilization*, que siguiendo la estela de los anteriormente citados, terminaría de afianzar el subgénero de la estrategia por turnos (*Turn-Based Strategy*, *TBS*).

Posteriormente aparecerían otros juegos que ayudarían a seguir definiendo los géneros de simulación y estrategia, entre ellos se encuentran obras aclamadas por la crítica como *Warcraft: Orcs &* Humans (1994, *Blizzard Enterteinment*), *Age of Empires* (1997, *Ensemble Studios*), *Theme Hospital* (1997, *Bullfrog Productions*), *RailRoad Tycoon II* (1998, *PopTop Software*) y el español *Commandos: Behind Enemy Lines* (1998, *Pyro Studios* [31]).

En cuanto al hardware que daba soporte a estos juegos, Los 80s acabaron con la guerra entre la consola de *Nintendo* y la de *Sega*, que acabaría ganando esta última. Respecto a los equipos sobremesa, la llegada masiva de *PC* a los hogares de todo el mundo supuso la práctica extinción de otras máquinas como la *Atari ST* o la *Comodore Amiga*. Es decir, los 90s comienzan con el *PC* y la *Sega Mega Drive* [44] reinando con casi absoluta hegemonía. Este clima de monopolio fue roto únicamente por la incursión en el mundo de las consolas portátiles de *Nintendo* con su *Game Boy*.

La *Game Boy* [42], ideada por *Gunpei Yokoi*, que a su vez era el creador de las anteriores *Game&Watch* de *Nintendo,* no tardó en erigirse como reina absoluta de las consolas portátiles. Este éxito se debió en gran medida a su gran catálogo, que contaba, entre otros juegos, con el *Tetris*, que llegó a vender más de 30 millones de copias. Pese a que compañías como *Atari*, con su *Lynx*, o *Sega* con su *Game Gear*, intentaron hacerle competencia al fenómeno *Game Boy*, la llegada de la *Game Boy Color*, actualización con gráficos a color y mejores prestaciones de la consola de Nintendo, hizo que toda competencia fuese meramente simbólica.

Figura 8. Anuncio de la Game Boy original.

Las ventas globales de la portátil de Nintendo llegaron a ser astronómicas, lo que unido a la posterior salida de la *Super Nintendo* [38] (*SNES*) provocó un cambio en el mercado del ocio doméstico. Actualmente, la Game Boy sigue siendo la tercera consola más vendida de la historia con casi 119 millones de ejemplares vendidos entre su versión clásica y la versión a color.

La *Super Nintendo*, también conocida como *Super NES*, era la sucesora directa de la *Famicom* y comenzó a comercializarse en 1991. Gracias al chip *SuperFX* con que contaban algunos de los cartuchos de la consola y a su procesador de 16 bits, fue la primera máquina capaz de reproducir juegos completamente en *3D*. En concreto, el juego *Star Fox* (1993, *Argonaut Software*) fue el primer videojuego publicado completamente en tres dimensiones.

Viendo en peligro su liderazgo en el mercado de las consolas por culpa de la *SNES*, *Sega* se apresuró a sacar su *Sega Mega-CD* [45] para intentar paliar el varapalo. El *Sega Mega-CD* era un periférico hardware para la *Sega Mega Drive* que ampliaba sus características hasta igualar a la *SNES* en muchos aspectos. Incluía un lector de *Mega-CD*, que ampliaba hasta 150 veces la memoria disponible para los juegos respecto a los cartuchos y permitía a la *Sega Mega Drive* funcionar como centro multimedia, siendo capaz de reproducir música y funcionar como karaoke.

Figura 9. Sega Mega Drive con periférico Sega CD Modelo 1.

Por otro lado, el *Sega Mega-CD* tenía preinstalados codecs de *FMV* (*Full Motion Video*), lo que le permitía reproducir pequeños fragmentos de video a baja resolución dentro de los juegos. Este avance, que podía haber supuesto una clara ventaja de Sega sobre su rival, fue en realidad contraproducente debido a la excesiva afición de los desarrolladores de la época por abusar del renderizado y la digitalización en detrimento de la jugabilidad. Esto provocó que los juegos para el Sega Mega-CD fuesen básicamente, videos con bifurcaciones en las que se le presentaba una elección al usuario, lo cual no acabo de convencer al público.

En la década de los 90s tuvo lugar otro avance fundamental para el desarrollo de videojuegos que redefinió en gran medida el proceso de desarrollo. Dicho avance fue la aparición de los primeros motores de videojuegos en 3D.

Todo empezó cuando *John Romero*, desarrollador experimentado de videojuegos, se hizo cargo de la publicación especializada *Gamer's Edge* perteneciente a la revista *Softdisk*. En dicha publicación trabajaban a su vez *Adrian Carmack*, como ilustrador y *John Carmack* y *Tom Hall* como programadores. Trabajando conjuntamente, *John Romero* descubrió que años antes, como pasatiempo, *John Carmack* y *Tom Hall* habían realizado una versión completamente en 3D del *Super Mario Bros. 3*. Viendo potencial que tenía dicha versión, *John Romero* decidió ponerse en contacto con *Nintendo* quien en última instancia se negó a licenciar dicho juego. Pese a que la citada versión nunca llegase a publicarse, los cuatro responsables de *Gamer's Edge* vieron el potencial latente en los videojuegos 3D y se mantuvieron a la espera de vientos más favorables.

Por aquella entonces, *Paul Neurath*, antiguo compañero de *John Romero*, que había montado un pequeño equipo de investigación en el ámbito de los videojuegos con exalumnos del *MIT*, estaba trabajando con *Chris Green* en el que sería el primer **motor de videojuegos 3D**. Dicho motor fue especialmente relevante por que permitía añadir imágenes reales a las superficies poligonales de los objetos 3D (las llamadas **texturas**). Poco después, al enterarse *John Romero* de los avances de su amigo y *Chris Green*, informó a *John Carmack*, que se lanzó a desarrollar su propio motor con características similares. Esta motivación desembocaría en la fundación del estudio de videojuegos *id Software*, por parte de *John Romero*, *John Carmack*, *Tom Hall* y *Adrian Carmack* y en la publicación de *Catacombs 3-D*, publicado en 1991 en la revista *Softdisk*, considerado el primer juego del genero de acción en primera persona (*First Person Shooter*, *FPS*).

Figura 10. Captura del juego Catacombs 3-D.

Un año más tarde, Wolfenstein 3D (id Software), que continuaba en la línea de Catacombs, mejoraría el motor de videojuegos y la jugabilidad y llegaría a vender en solo un año más de 100.000 copias. Hasta ese momento cada estudio realizaba internamente todas las tareas de programación relacionadas con el producto final, pero fue con la tremenda acogida de su último juego cuando id Software se lanzó en 1993 a vender licencias de uso de su motor. El primer juego externo a id Software en usar su motor sería Shadowcasters, de Raven Software, pero no tardarían en unírsele otras producciones de distintos estudios.

Ese mismo año, *id Software* sacaría al mercado *DOOM*, un juego que destacaba por su contenido extremadamente violento y temática adulta. El juego causó tal controversia que convirtió *John Romero* en un personaje mediático y provocó que el mundo de los videojuegos diese el salto definitivo de las dos dimensiones al 3D. Además, *DOOM* fue el primer juego que oficialmente permitía el ***modding***, que consistía en la modificación de diversos aspectos del juego a través del código del mismo por parte de la comunidad de jugadores. Esto, unido a que fue uno de los primeros juegos en permitir el juego en línea ya fuese a través de red local o internet y que fomentase el uso de shareware, provocó que a día de hoy se considere a *DOOM* como uno de los juegos más relevantes de la historia de los videojuegos.

El auge de juegos extremadamente violentos como *DOOM*, *Mortal Kombat* o *Street Fighters* provocó que la norteamericana *ESRB* (*Entertainment Software Rating Board*) endureciese su sistema de clasificación. Esto provocó que el mercado de las videoconsolas y maquinas arcade cayese notablemente. En 1994, el volumen de ventas de videoconsolas solo alcanzaba el 17% del total del mercado del ocio doméstico. En un clima tan adverso como éste, *Sega* sacó su nueva máquina de 32 bits. La *Sega Saturn* [46] salió al mercado en noviembre de 1994 con unas características muy potentes. Contaba con un procesador de dos núcleos a 28.6 MHz, una RAM de 2 Mb, profundidad de color de 24 bits y una unidad de CD de doble velocidad. A pesar de ello, el hecho de que su catálogo contase con una mayoría de títulos exclusivos para Japón y el lanzamiento inmediatamente posterior de la *PlayStation* [33] de *Sony*, provocaron que las ventas de la videoconsola de *Sega* fuesen prácticamente anecdóticas.

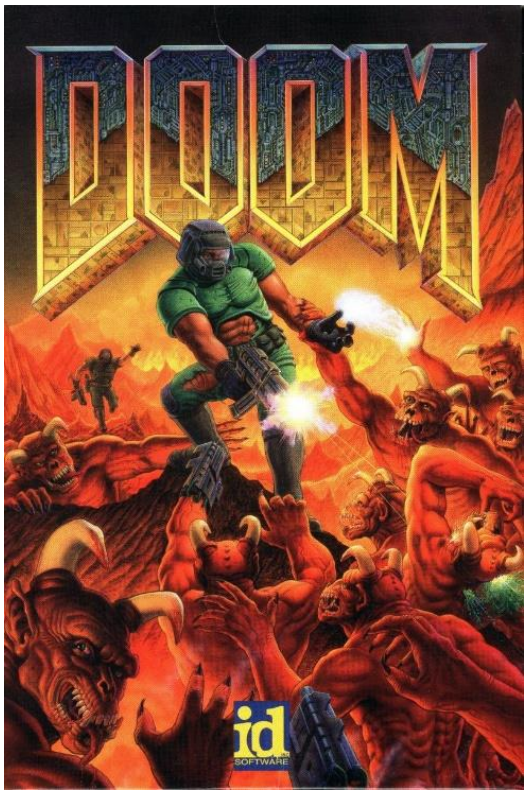


Figura 11. Caratula del juego DOOM de id Software.

La incursión de *Sony* en el mercado de las videoconsolas es la conclusión de una aventura que empezó años antes con la creación de un prototipo de videoconsola basada en la tecnología *CD-ROM* de la mano de *Nintendo*. Durante el proceso de desarrollo de dicho prototipo, *Nintendo* renunció a la colaboración para marcharse a trabajar en su lugar con la empresa de productos electrónicos *Philips*. Es en ese momento cuando *Norio Ohga*, presidente de *Sony*, ordena la creación de *Sony Computer Entertainment*, su división de videojuegos y pone al mando de ella a *Ken Kutaragi* con el objetivo de aplicar lo aprendido durante la colaboración con *Nintendo* a la creación de una videoconsola capaz de liderar el mercado.

La *PlayStation*, abreviada como *PS1*, tomó su nombre del prototipo creado junto a *Nintendo*, la *PlayStation X* y se presentó al público en Japón en diciembre de 1994, tan solo un mes después de la salida de la *Sega Saturn*. Sobre el papel, el sistema de Sony era ligeramente menos potente que el de Sega; pero la facilidad de desarrollo para la *PlayStation* comparado con las dificultades que presentaba la *Sega Saturn* y diversas innovaciones que presentaba la consola de *Sony* como un mando más ergonómico y las tarjetas de memoria extraíbles, decantaron la balanza, tanto para consumidores como para desarrolladores, hacia la *PS1*, que acabaría vendiendo un millón de unidades en menos de 6 meses.

En mayo de 1995, unos meses más tarde de su salida en los mercados orientales, se presentó la *PlayStation* a los consumidores occidentales durante la primera gran feria de los videojuegos, el *E3*. En dicha feria, además de presentarse la nueva consola de Sony, se presentó la que sería su competencia por parte de *Nintendo*, la *Ultra 64*, que finalmente se llamaría *Nintendo 64* [43]. Dicha confrontación tuvo tal relevancia que el *Electronic Entertainment Expo* o *E3* se celebra desde entonces de forma anual en la ciudad de Los Ángeles promovido por la organización *Entertainment Software Association*.

Figura 12. Consola PlayStation de Sony Computer Entertainment.

La *Nintendo 64* salió al mercado con unas especificaciones muy superiores a sus rivales y con el apoyo de grandes estudios del sector como *Rare*. Sin embargo, el hecho de que sus juegos se comercializasen en cartuchos, que eran mucho más caros que los *CD-ROM* de la competencia, supuso la desventaja que acabaría dejando a la *PlayStation* vía libre para coronarse como la reina de la quinta generación de videoconsolas de forma indiscutible.

El fin de la década de los 90 posicionaría de forma irrefutable a *Sony Computer Entertainment* como referente en el sector del ocio doméstico. Su liderazgo vendría de la mano del fin de las restricciones de edad a la hora de desarrollar videojuegos. Gracias a ello, se empezó a desarrollar historias más densas y juegos de géneros dirigidos exclusivamente a los adultos como los *shooters* o los *survival horror*. Fue así como juegos como *Resident Evil* (1996, *Capcom*), *Final Fantasy VII* (1997, *Squaresoft*), *Grand Theft Auto* (1997, *Rockstar Games*), *Metal Gear Solid* (1998, *Konami*) o *Silent Hill* (1999, *Konami*) vieron la luz iniciando una segunda revolución creativa.

2.1.4 La Sexta Generación y los nuevos géneros

La industria de los videojuegos empezó el siglo XXI con las mejores expectativas posibles. La sexta generación de consolas y la apertura a públicos no convencionales dio alas a la creatividad de los desarrolladores. Dichos desarrolladores además ya no tenían nada que ver con los programadores de videojuegos artesanos del siglo XX, si no que eran profesionales altamente cualificados y formados específicamente para la labor que llevaban a cabo. El programador amateur que desarrollaba prácticamente en solitario los juegos dejó paso a equipos de profesionales multidisciplinares organizados con el objetivo de ofrecer un producto de calidad a través de un proceso de ingeniería.

En cuanto al hardware, con el cambio de siglo llegó la que sería la sexta generación de videoconsolas y con ella la muerte definitiva de las máquinas recreativas. La primera en llegar sería la *Dreamcast* de *Sega*, en 1999. Fue la primera videoconsola en incluir un modem integrado para jugar online. Otras de sus características principales eran su procesador *Hitachi*, su peculiar tarjeta de memoria (*Visual Memory System, VMS*) que incluía botones, pantalla y se integraba en el mando y su desacertada apuesta por el *GD-ROM* como soporte para los videojuegos. A día de hoy esta consola sigue siendo la última que ha sacado al mercado *Sega*, que a partir de ese momento se centraría en el desarrollo de videojuegos para otras compañías.

Un año después, en marzo del año 2000, *Sony* sacaría al mercado la *PlayStation 2* [34] que supondría una mejora en tecnología respecto a la *PlayStation* original, manteniendo la misma filosofía. Fue la primera consola en dar el salto a los procesadores de 128 bits y en apostar por el *DVD* como soporte para sus juegos. La consola de *Sony* también apostó por el juego online con periféricos destinados a ello como el headset, pero no llegó a integrar el adaptador *Ethernet* en la consola hasta la segunda versión de la *PS2*. Otro de sus aciertos fue el de usar una adaptación de su mando con joysticks para la *PlayStation*, el *DualShock*, como controlador predeterminado.

Figura 13. Mando DualShock 2 para la PlayStation 2.

El *DualShock 2* mejoraba ligeramente las formas y materiales respecto a su versión anterior para hacerlo más cómodo y ergonómico manteniendo la vibración y el doble joystick que le caracterizaba. Además, disminuyeron su peso y modificaron los botones para que fuesen sensibles a un rango de 8 bits de presión, lo cual abría nuevas posibilidades a los desarrolladores. Las innovaciones de la familia de controladores *DualShock* y en concreto, los relativos a su segunda versión, fueron tan relevantes y tan bien acogidas por público y desarrolladores que actualmente conceptos como los rangos de presión, los dos joysticks y el sistema de vibración son estándares en la práctica totalidad de los controladores de videojuegos.

Todas estas mejoras, unidas a uno de los catálogos de juegos más extensos de la historia (más de 10.000 títulos) causaron que a día de hoy la *PlayStation 2* siga siendo la consola más vendida de la historia con casi 160 millones de unidades vendidas entre sus dos modelos.

En 2001 entraría en el mercado de las videoconsolas, de forma bastante inesperada, un nuevo competidor. *Microsoft* presento su *Xbox* [47] que, si bien no tuvo tanto éxito como la consola de *Sony*, sentó el precedente de lo que a día de hoy en día es una de las empresas más relevantes en la producción de videojuegos y sistemas de ocio doméstico. Ese mismo año, *Nintendo* sacó su *GameCube*, consola que debido a decisiones tan malas como no abrir su catálogo a públicos más adultos o usar un soporte exclusivo (*GameCube Optical Disc*, *GOD*) resultó en graves pérdidas para la empresa nipona, que desde ese momento decidió centrarse en el mercado de las consolas portátiles.

Esta situación de relativa estabilidad en el mercado se mantuvo durante la primera mitad de la década. En *PC* triunfaban los juegos de estrategia y *FPS* centrados en el multijugador online; *PlayStation 2* dominaba el mercado de las videoconsolas con una gran diferencia respecto a sus rivales y *Nintendo* tenía el practico monopolio de las consolas portátiles pese a los intentos de *Nokia* con su *N-Gage* (2003) por quitarle parte del mercado. Fue en ese momento cuando *Nintendo* decidió dar un paso al frente con su nueva estrategia presentando la *Nintendo DS* [39] en noviembre de 2004. Dicha consola contaba con elementos innovadores como el uso de dos pantallas, siendo una de ellas táctil. *Sony*, al igual que *Nokia* anteriormente, intentó robar mercado a la nueva portátil de *Nintendo* sacando la *PSP* (*PlayStation Portable*) en marzo de 2005, pero sus cifras de ventas no fueron las esperadas.

Figura 14. Nintendo DS

Una vez *Nintendo* hubo movido ficha con su nueva portátil, tanto *Sony* como *Microsoft* se pusieron en marcha proveyendo que *Nintendo* no tardaría en sacar una nueva consola de sobremesa. Así fue como *Microsoft* forzó la aparición de la séptima generación de consolas al sacar en noviembre de 2005 su *Xbox 360* [48]. Esta salida provocó una reacción en cadena que hizo que pocos meses después *Sony* sacase su *PlayStation 3* [35] y posteriormente *Nintendo*, en noviembre de 2006, sorprendió al mundo con la consola más innovadora en funcionalidad hasta la fecha. La *Wii* [40], como se llamó dicha consola, no gustó entre los jugadores habituales, pero encantó a los jugadores casuales por su control basado en el movimiento y sus juegos sencillos pensados, en la mayoría de los casos, para jugar en grupo.

En cuanto al software, una parte del público considera que con la industrialización del sector vino la perdida en la creatividad por parte de los desarrolladores. Esta opinión se debe mayoritariamente a que las grandes empresas preferían apostar por sagas con personajes conocidos o géneros exitosos en lugar de propuestas nuevas. Pese a que esta tendencia estaba generalizada y muchas compañías se enfocaban en desarrollar juegos con gran potencial grafico en lugar de producciones innovadoras, seguían apareciendo de forma constante nuevos géneros, franquicias y mejoras en la jugabilidad a un ritmo incluso mayor que en épocas anteriores. Ejemplos de esta constante evolución son el *Guitar Hero* de *Activision*, el simulador social *The Sims* de *Electronic Arts* o los *MMORPG* (*Masive Multiplayer Online Role-Play Game*) como el *World of Warcraft* de *Blizzard Entertainment*.

Dos de los juegos más aclamados de la generación por crítica y público serían *Grand Theft Auto III* y *Halo: Combat Evolved*. Dichos juegos, pese a pertenecer uno de ellos a una franquicia de éxito y el otro a un género de moda en aquella época, supusieron un derroche de calidad y originalidad tal que les situó en el panteón de los juegos de forma irrefutable.

Figura 15. Grand Theft Auto III para PlayStation 2.

El *GTA III*, juego publicado por *Rockstar Games* en 2001, se caracterizaba por un argumento complejo y denso más similar a los que se podían ver en el cine que a los de los videojuegos de la época. Esto, unido a ser uno de los precursores del subgénero *sandbox*, caracterizado por juegos no lineales en los que el mundo es abierto y hay cierta libertad de acción, lo situó como uno de los juegos más relevantes de la generación del cual han bebido muchos otros títulos importantes.

En cuanto a *Halo: Combat Evolved* (2001, *Bungie Studios*), su relevancia viene por ser el primer *FPS* centrado en el multijugador exclusivo de consola. Esto, pese a poder parecer una cuestión trivial, supuso un avance enorme en distintos aspectos. En el aspecto de la jugabilidad, el hecho de conseguir adaptar un género, pensado para ser jugado con teclado y ratón, a un mando de videoconsola a través de los joysticks era ya un logro importante. Pero los desarrolladores no se quedaron ahí, desarrollaron un sistema de controles que a día de hoy es un estándar de consola similar al estándar *WASD* de *PC*. La otra cuestión que hizo de *Halo* un juego destacable es que fue el verdadero responsable de popularizar el juego online en consolas. Antes de *Halo*, hubo diversos juegos que intentaron potenciar el juego a través de conexión, pero ninguno de ellos tuvo el éxito suficiente. *Halo*, consiguió que el multijugador online fuese tan divertido y relevante para el público que *Microsoft* hizo de ello su principal estandarte, centrando su catálogo en aquellos juegos que contasen con esta funcionalidad.

Figura 16. Halo: Combat Evolved para Xbox.

La primera década del siglo XXI acabaría con un nuevo clima de estabilidad en el que *Microsoft* y *Sony* se encontraban en su propia guerra fría por el liderazgo del mercado de las videoconsolas. *Nintendo* seguía siendo la compañía líder en ventas de portátiles, pero empezaba a ver su hegemonía amenazada debido a la aparición en 2007 de los primeros *smartphones*. En *PC*, la aparición de plataformas suministradoras de juegos en formato digital como *Steam* (2003, *Valve Corporation*) democratizó los precios de los juegos, lo que empezó a atraer a nuevos jugadores. En cuanto al desarrollo de videojuegos, el mercado seguía copado por las grandes empresas multinacionales pero la reducción en los costes de producción y la aparición de motores de videojuegos gratuitos o cuya licencia dependía del margen de beneficios, propició la reaparición de los estudios independientes, centrados sobre todo en el mercado de los *PC* y los *smartphones*.

2.1.5 Situación actual: La Gran Revolución Creativa

La séptima generación de consolas tardó en tener su relevo más de lo que habían tardado las anteriores. Esto se debía, entre otros motivos, a que las máquinas de *Microsoft* y *Sony* seguían teniendo potencia suficiente para los juegos que se estaban desarrollando. Además, el volumen de ventas de las tres grandes consolas de sobremesa seguía siendo el esperado, entre otros motivos debido a la salida al mercado de periféricos o mejoras en el firmware y software que daba soporte a nuevas funcionalidades y dinámicas de juego, sin necesidad de cambiar el hardware principal. Dos de los periféricos mencionados fueron la cámara *Kinect*, de *Microsoft* y el *PlayStation Move* de *Sony*. Dichos dispositivos retomaban el intento, que había fracasado en generaciones anteriores, de dar el salto definitivo a la ***realidad aumentada*** y la ***realidad virtual*** [50].

> *"La **realidad aumentada** (Augmented Reality, RA) es el término que se usa para definir la visión de un entorno físico del mundo real, a través de un dispositivo tecnológico, es decir, los elementos físicos tangibles se combinan con elementos virtuales, logrando de esta manera crear una realidad mixta en tiempo real" –* ***Wikipedia (2017).***
>
> *"La **realidad virtual** (Virtual Reality, VR) es el término que se usa para describir un entorno tridimensional generado por ordenador que se puede explorar y con el cual el usuario puede interactuar. Dicho usuario se convierte en parte del entorno o se sumerge en el mismo de forma que puede manipular objetos o realizar una serie de acciones." –* ***Virtual Reality Society*** [80] **(2017).**

Kinect salió al mercado en noviembre de 2010 como periférico basado en la realidad aumentada para *Xbox 360*. Un año después saldría un kit para *PC* que supondría todo un logro en defensa de los procesos de ***gamificación*** [15]; dado que acabaría usándose en programas de rehabilitación de pacientes, gestión de historiales clínicos en entornos esterilizados y otros proyectos no relacionados con la industria del ocio digital debido a que su coste era muy inferior a otros sistemas similares de la misma calidad. Este dispositivo cuenta con una cámara dual *RGB*, un sensor de profundidad, un micrófono de múltiples matrices y un procesador dedicado que le permite reconocer comandos de voz, gestos, objetos en tres dimensiones y caras.

Pese a que su catálogo no llegó a superar la veintena de títulos y que las ventas no fueron las esperadas, *Kinect* es a día de hoy uno de los periféricos más relevantes que ha salido de la industria de los videojuegos dada su potencial aplicación a diferentes tareas y las posibilidades de mejora y desarrollo que tiene.

> *"La **gamificación** es el uso de las mecánicas del juego, su estética y el pensamiento de juego para involucrar a la gente, motivar la acción, promover el aprendizaje y resolver problemas" – Karl Kapp, "The Gamification of Learning and Instruction"*

PlayStation Move por otro lado, representa un avance con menos repercusiones fuera del sector de los videojuegos, pero igualmente relevante. Su tecnología une conceptos similares al *Kinect* de *Microsoft* y el *WiiMote*, controlador de la consola *Wii*, de *Nintendo*. En concreto, el periférico constaba de un mando con sensor de movimiento y una esfera luminosa en uno de sus extremos y una cámara, *PlayStation Eye*, que detectaba la posición y el movimiento del mando. Aunque, al igual que *Kinect*, su éxito no fuese el esperado, se le considera uno de los precursores de los mandos utilizados como periféricos de las máquinas de realidad virtual actuales.

Una vez la tecnología de la séptima generación se había exprimido hasta el límite, empezó a ser evidente la llegada de una octava generación de consolas, que se preveía mucho más potente, pero sin innovaciones relevantes. Los primeros en dar el salto a la nueva generación fueron *Nintendo*, con su *Wii U* en noviembre de 2012. Pese a que se esperaba que *Nintendo* sacase una consola con una potencia superior, dado que la *Wii* no estuvo a la altura del rendimiento de sus competidoras, la *Wii U* resulto ser una actualización de su precursora, con un controlador nuevo que integraba una pantalla y una potencia equivalente a la de *Xbox 360*. El público no acogió muy bien las características de la consola que, pese a que en el momento de redactar este trabajo todavía no ha salido a la venta su sucesora, se tradujo en un fracaso por parte de *Nintendo* en cuanto a ventas.

Las siguientes en salir al mercado, con una semana de diferencia, fueron la *PlayStation 4* [36] de *Sony* y la *Xbox One* [49] de *Microsoft* en noviembre de 2013. Ambas son consolas con un hardware muy similar ya que cuentan con un procesador de 8 núcleos de 1.6-1.75 GHz, discos duros integrados de 500 Gb y 1 Tb y formatos de imagen *FullHD*. Las diferencias más relevantes entre ambas consolas son la *GPU* ligeramente más potente de la *PlayStation 4* y la utilización de *Windows 10* como sistema operativo de *Xbox One*.

Respecto al mercado de las consolas portátiles, *Nintendo* sigue siendo la empresa líder en ventas gracias a la salida en 2011 de su *Nintendo 3DS*; actualización de su portátil anterior con mejoras en potencia, cámara dual para sacar fotos en tres dimensiones y una pantalla con tecnología estereoscópica que permite la visualización de juego en *3D* sin necesidad de gafas. Aunque *Nintendo* sigue teniendo el practico monopolio del mercado de las portátiles, la cifra de ventas se está viendo seriamente afectada por los nuevos dispositivos móviles y empieza a ser cada vez más evidente que el mercado de las consolas portátiles acabará siendo absorbido por el de los *smartphones* y *tablets*.

En *PC*, el abaratamiento de hardware con especificaciones iguales o superiores al de las consolas unido al surgimiento de los juegos *free-to-play* [16] y webs como *Humble Bundle* que ofrecían paquetes de juegos en formato digital a precios muy asequibles donando parte de la recaudación a causas benéficas, supuso un relanzamiento de la plataforma cuando estaba siendo desplazada del mercado del ocio digital. Los juegos de simulación y el *modding* no solo seguían siendo un pilar fundamental de *PC,* sino que eran prácticamente exclusivos de esta plataforma.

El *modding* era, y es, una práctica tan relevante en *PC* que es la responsable de la aparición no solo de juegos nuevos a partir de la modificación de otros, si no de géneros enteros y de propulsar el mercado de los juegos *indie*

[21] [22]. En concreto, el género *MOBA* (*Multiplayer Online Battle Arena*) considerado actualmente el más popular de *PC* y uno de los mejores en cifras en todo el mercado de videojuegos; debe su aparición a un *mod* del *Warcraft III: The Frozen Throne* (2003, *Blizzard Entertainment*) llamado *Defense of the Ancient*. *DotA: AllStars*, como se conoció a dicho *mod* creado por *Steve Feak* basándose en un mapa de un *modder* conocido como *Eul*; se hizo tan popular y proponía una jugabilidad y un catálogo de personajes, acciones y objetos tan distintos del juego base, que era considerado un juego a parte por la comunidad. Varios años más tarde, *Steve Feak* unió fuerzas con *Steve Mescon*, *Brandon Beck* y *Marc Merrill* y fundaron *Riot Games* con el objetivo de crear un juego que recogiera el testigo de *DotA*.

Figura 17. Captura de pantalla del aspecto original del juego League of Legends.

League of Legends (*LoL, Riot Games*) saldría al mercado en octubre de 2009 bajo el modelo de marketing *free-to-play*. Su lanzamiento fue un éxito tal que catapultó a los juegos *free-to-play* y al género *MOBA* a los primeros puestos de beneficios en *PC*, llegando a ser en 2015 líder de mercado con un volumen de beneficios superior a los 1.600 millones de dólares, según un estudio realizado por el portal *SuperData*, y relanzando de manera efectiva el sector de los *eSports* (deportes electrónicos). A raíz de su éxito saldrían al mercado otros juegos, con la misma tónica, financiados por las grandes empresas del sector y con el objetivo de aprovechar la popularidad del modelo. Algunos ejemplos son *DotA 2* (2013, *Valve Corporation*), *Heroes of the Storm* (2015, *Blizzard Entertainment*), *Paragon* (2016, *Epic Games*), Battleborn (2016, *Gearbox Software*) y *Overwatch* (2016, *Blizzard Entertainment*).

Respecto a los desarrolladores, la década actual es una época propicia para la creatividad y la asunción de riesgos. Factores como el fomento del *modding* por parte de las empresas desarrolladoras de videojuegos con herramientas oficiales; la facilidad de distribución y marketing que suponen plataformas como *Steam Greenlight* (2012, Valve Corporation), *Google Play* (2008, Google) o *App Store* (2008, Apple); los nuevos modelos de financiación como el *crowdfunding* o el *free-to-play* con microtransacciones y la aparición de herramientas de desarrollo gratuitas o con financiación subrogada a los beneficios obtenidos como *Unreal Engine 4* (2015, *Epic Games*), *Unity 5* (2015, *Unity Technologies*)o *Game Maker Studio* (2009, *YoYo Games*), han abaratado notablemente los costes de desarrollo y facilitado la difusión del producto final a escala global. Esto ha provocado la aparición de grandes títulos de estudios independientes como el anteriormente mencionado *League of Legends* (2009, Riot Games) o la saga *The Witcher* (2008, *CD Projekt RED STUDIO*), aclamados por crítica y público, que ha obligado a las grandes empresas del sector a colaborar con los estudios *indie*.

En la actualidad, la creación de estudios universitarios enfocados en el mercado de los videojuegos, la diversidad de herramientas abiertas para el aprendizaje y la actividad y colaboración de la comunidad de

desarrolladores ha fomentado la aparición de profesionales altamente cualificados capaces de asumir grandes proyectos con poca financiación y equipos pequeños. Todo esto, unido a la diversidad de plataformas, géneros, modelos de marketing y la apuesta firme de instituciones y empresas por el mercado del ocio digital, ha desencadenado una nueva época dorada para el sector de los videojuegos.

Figura 18. Nintendo Switch, primera consola de la novena generación.

El futuro [53] de la industria de los videojuegos estará marcado, según el análisis de expertos como *Devinder Kumar*, *CFO* de *AMD*, o *Damian Thong*, jefe de investigación tecnológica de *Macquaire Group*, por una novena generación de consolas, ya en desarrollo, centradas en modos de juego innovadores y métodos de interacción más naturales para el público general como la realidad aumentada o la realidad virtual. La primera consola en salir de dicha novena generación es la *Nintendo Switch* [41], en marzo de 2017, pero se pronostica que las herederas de *PS4* y *Xbox One* no tardarán más de dos años en salir a la venta. A nivel laboral, se prevé un aumento de la aparición de estudios y empresas en el sector que responda a la creciente demanda de productos de ocio digital. Otro de los aspectos fundamentales que marcará el futuro de sector es la *gamificación*, ya que la infinidad de técnicas, tecnologías y productos que está generando el mercado de los videojuegos tienen un demostrado éxito en otros sectores como el médico o el inmobiliario.

2.2 Unreal Engine 4. Características y comparativa con otros motores

Para el desarrollo de nuestro videojuego hemos decidido utilizar el motor *Unreal Engine* en su versión 4.15. El objetivo es poder asumir el proyecto de creación de un videojuego con un nivel de calidad alto aprovechando contenidos y funcionales administradas por el núcleo del motor; evitando así tener que desarrollar todos los aspectos del juego desde cero.

Figura 19. Logo de Unreal Engine 4.

2.2.1 Historia del motor

El motor *Unreal Engine* [56] vio la luz por primera vez en 1998 de la mano del estudio *Epic Games* y su primera versión ya integraba funcionalidades como renderizado, detección de colisiones, inteligencia artificial, visibilidad, opciones de red y manipulación de archivos del sistema.

En 2002 apareció su segunda versión, que contaba con un nuevo motor de físicas llamado *Karma physics* e integraba *Unreal Editor 3* como motor de creación de mapas. *UE2* pasó por una reescritura completa de núcleo y motor de renderizado que le aportó, junto con los motores de físicas y el nuevo editor, mejoras en el rendimiento, un editor del sistema de partículas, físicas de vehículos y soporte para *PlayStation 2*, *GameCube*, *Xbox* y máquinas de 64 bits.

La tercera generación del motor aparecería en 2006 y estaría diseñado para *Xbox 360*, *PS3* y *PC* con soporte para *DirectX9* y *DirectX10*. Gracias a la reescritura, por segunda vez, de su núcleo, el motor permitía la aplicación de técnicas graficas avanzadas como el *HDDR* (*High-Dynamic-Range Rendering*), las sombras dinámicas y el uso de *normal mapping*. Otro aspecto importante era que en esta versión se eliminó el motor *Karma physics* para usar en su lugar el *PhysX* de *NVidia*. Además, se incluyó el motor *FaceFX* que facilitaba la creación de animaciones faciales en los personajes. Dos años más tarde, *Epic Games* lanzó una versión mejorada del motor con características como una mejora en la potencia de renderizado, físicas de líquidos y texturas corporales más realistas, rutinas avanzadas de luces y sombras y una inteligencia artificial mejorada.

Dos de los hitos más importantes de *Unreal Engine 3* fue la asociación en 2007 de *Epic Games* y *Sony* para optimizar el motor para el desarrollo de juegos para *PlayStation 3* y el lanzamiento del *UDK* (*Unreal Development Kit*) en 2009, que era una versión gratuita del motor con licencia condicionada a los beneficios obtenidos por los juegos creados con el motor. Esto provoco que casi la práctica totalidad de estudios de videojuegos y grandes distribuidoras que desarrollaban para *PlayStation 3* se asociase a su vez con *Epic Games*.

En 2015 *Epic Games* sorprendería a la industria de los videojuegos liberando de forma parcialmente gratuita, bajo el slogan "*if you love something, set it free*", el motor *Unreal Engine 4* con todas las actualizaciones que reciba con la única condición de que, si un proyecto realizado en UE4 obtenía más de 3000 dólares de beneficios, *Epic Games* recibiría el 5% de los beneficios trimestrales de la obra. Cualquier proyecto educativo o sin ánimo no se verá gravado con ningún royalty.

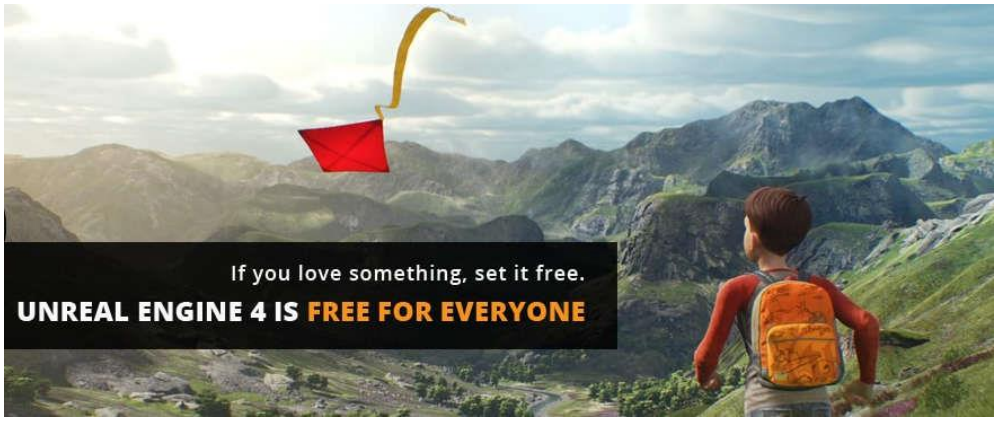


Figura 20. Anuncio de Unreal Engine 4 por parte de Epic Games.

A día de hoy el motor se encuentra en su versión 4.15 y cuenta con una asociación académica con más de 100 universidades de prestigio y con más de 60 grandes clientes a la altura de *Disney*, *Microsoft*, la *NASA*, *NVidia*, *Intel*, *Apple*, *Activision* o *Google*.

2.2.2 Características

Las principales características de este motor de videojuegos [55] son las siguientes:

- ✓ Soporte multiplataforma. (*Windows, Mac, Linux, PS4, Xbox One, Nintendo Switch, SteamOS, iOS, Android* y *HTML5*)
- ✓ Soporte para realidad virtual. (*PlayStation VR, Oculus Rift, Samsung Gear VR, HTC Vive* y *Google Daydream*)
- ✓ Compatibilidad total con funcionalidades de renderizado de *DirectX11* y *DirectX12*.
- ✓ *Cascade VFX*. Editor de partículas completo y detallado optimizado para un bajo consumo de *GPU*.
- ✓ Acceso completo al código fuente.
- ✓ Soporte para programación en lenguaje *C++*.
- ✓ *Blueprint*. Lenguaje de scripting visual.
- ✓ *Debugger* de *Blueprint* "en caliente".
- ✓ *Persona*. Editor de secuencias de animación.
- ✓ *Matinee*. Editor de secuencias cinemáticas.
- ✓ Editores de pos-procesado de audio y video.
- ✓ Cuenta con un gran soporte al aprendizaje en varios idiomas. Documentación, una extensa bibliografía, videotutoriales, clases en directo vía *Twitch* y wiki.
- ✓ Actualmente es el motor más utilizado por las empresas del sector de los videojuegos.
- ✓ Cuenta con una comunidad de desarrolladores activa, colaborativa y extensa.
- ✓ Gratuito hasta 3000 dólares de beneficio. 5% de royalties a partir de dicha cifra.
- ✓ Bazar con contenidos de la comunidad.

2.2.3 Comparativa

A continuación, presentamos otros tres grandes motores de videojuegos de uso generalizado con una breve descripción y sus características generales. Tras esto, expondremos las conclusiones y los motivos que nos han llevado a tomar la decisión de usar *Unreal Engine 4* para este proyecto.

Figura 21. Logo de CryEngine.

2.2.3.1 CryEngine V

CryEngine [61] es un motor de videojuegos creado por la desarrolladora *Crytek* con el propósito original de ser el motor de demostración de *NVidia*. El primer juego realizado usando este motor fue *Far Cry* (2004, *Crytek*) y tuvo tal éxito que en 2006 *Ubisoft* compró todos los derechos sobre el motor. *CryEngine* ha pasado por 4 versiones y la actual, *CryEngine V*, ha supuesto un cambio de modelo de negocio; siendo 100% gratuito independientemente de los beneficios del proyecto.

Sus principales características son:

- ✓ Soporte multiplataforma. (*Windows, Linux, PS4, Xbox One, Nintendo Switch, iOS* y *Android*)
- ✓ Soporte para realidad virtual. (*PlayStation VR, Oculus Rift, HTC Vive* y *OSVR*)
- ✓ Compatibilidad total con *DirectX12*.
- ✓ Soporte para programación en lenguaje *C++*.
- ✓ *PBR* (*Physically Based Rendering*). Materiales y luz fotorrealistas gracias al renderizado basado en físicas realistas.
- ✓ Soporte para teselado acelerado por hardware en todos los objetos modelados.
- ✓ *SVOGI* (*Voxel-Based Global Illumination*). Iluminación indirecta y oclusión ambienta fotorrealista.
- ✓ Sistema de Inteligencia Artificial Avanzada integrado.
- ✓ Soporte de archivos *FBX*.
- ✓ *Trackview*. Editor de secuencias cinemáticas.
- ✓ *MNM* (*Multi-Layer Navigation Mesh*). Navegación dinámica de las estructuras de datos para la resolución de colisiones.
- ✓ Cuenta con un gran soporte al aprendizaje en varios idiomas. Documentación, bibliografía, videotutoriales y *webinars*.
- ✓ Cuenta con un gran soporte para la comunidad de desarrolladores.
- ✓ Totalmente gratuito.
- ✓ Bazar con contenidos de la comunidad.

Figura 22. Logo de Unity.

2.2.3.2 Unity 5

Unity [57] es un motor de videojuegos desarrollado por *Unity Tecnologies* en 2005 para su juego *GooBall*. El juego no tuvo ningún éxito, pero sus desarrolladores vieron en *Unity* la posibilidad de ofrecer a las compañías desarrolladoras un motor competente a un precio asequible. *Unity* fue lanzado en 2006 y su éxito actual se debe a que ofreció a los estudios de desarrollo *indie* un motor competente con un precio por licencia mucho más bajo que el que tenían el resto de motores usados por los grandes estudios.

La versión actual, *Unity 5* [58], cuenta con las siguientes características:

- ✓ Soporte multiplataforma. (*Windows, Mac, Linux, SteamOS, WebGL, PS4, Xbox One, Nintendo Switch, Nintendo Wii, iOS, Android, Windows Phone, Tizen, Fire OS, Android TV, Samsung SmartTV, tvOS* y *Facebook Gameroom)*
- ✓ Soporte para realidad virtual. (*PlayStation VR, Oculus Rift, HTC Vive, Samsung Gear VR, Microsoft HoloLens, Steam VR, Google Cardboard* y *Google Daydream*)
- ✓ NVidia PhysX 3.3 como motor de físicas integrado.
- ✓ Soporte para programación en lenguaje *C#* y *Javascript*.
- ✓ *Pipeline* de importación extenso con múltiples formatos de video, audio, texto e imagen.
- ✓ Enlighten [73] de ARM para la gestión de la iluminación global.
- ✓ *Umbra3D* [77]. Sistema integrado de oclusión ambiental y teselado.
- ✓ Cuenta con un gran soporte al aprendizaje en varios idiomas. Documentación, bibliografía y videotutoriales.
- ✓ Cuenta con una certificación propia.
- ✓ Es el motor más usado por los estudios *indie*.

- ✓ Cuenta con un gran soporte para la comunidad de desarrolladores.
- ✓ Diferentes licencias (personal, plus, profesional y empresa).
- ✓ Bazar con contenidos de la comunidad.

Figura 23. Logo de GameMaker: Studio.

2.2.3.3 GameMaker: Studio

GameMaker [59] es un motor de desarrollo de videojuegos basado en *Delphi* a partir de un *SDK*. Fue creado en la década de los 90s por el profesor universitario *Mark Overmars* con el objetivo de crear una herramienta de animación que pudiese ser utilizada por estudiantes con pocos conocimientos de programación. Su primera versión fue liberada en 1999 y actualmente, su última versión, *Game Maker: Studio 2*, se encuentra en fase de beta pública.

Las características [60] más reseñables que presenta son:

- ✓ Soporte multiplataforma. (*Windows, Mac, Ubuntu, HTML5, PS3, PS4, Xbox One, PSVita, iOS, Android, Windows Phone, Tizen, Amazon Fire*)
- ✓ Utiliza *GML* (*GameMaker Language*) basado en *C*.
- ✓ *Box2D* [72]. Motor de físicas en 2D.
- ✓ *Spine* [75]. Motor de animación 2D.
- ✓ Compatible con el uso de *shaders*.
- ✓ Cuenta con un buen soporte al aprendizaje en inglés. Documentación y videotutoriales.
- ✓ Cuenta con un buen soporte para la comunidad de desarrolladores.
- ✓ Diferentes licencias (free, profesional y master collection).
- ✓ Bazar con contenidos de la comunidad.

2.2.3.4 Conclusiones y justificación de la elección

Los cuatro motores de videojuegos presentados en esta comparativa son buenas opciones a la hora de realizar un videojuego. Todos ellos tienen un conjunto muy bueno de herramientas abalado por grandes títulos creados a partir de cada uno de ellos.

Si lo que se pretende es crear un juego de gran fotorealismo en el que prime el aspecto gráfico y la percepción de un entorno realista e inmersivo por parte del usuario, la mejor opción es el motor *CryEngine V*.

Si el objetivo es crear un juego en dos dimensiones no especialmente ambicioso en cuanto a mecánicas o gráficos y con una inversión pequeña, la mejor opción sería *GameMaker: Studio*.

En cuanto a *Unity* y *Unreal Engine*, en muchos aspectos son muy parecidos y en algunos casos usan hasta los mismos componentes. Hay aspectos diferenciadores relevantes como el sistema *Blueprint* de visual scripting, el uso de *C++* o *C#* o la diferencia entre licencias de uso que se deben valorar a la hora de tomar la decisión de que motor usar.

En nuestro caso, nos hemos decantado por *Unreal Engine* por varios motivos. El primero de ellos es aprovechar el desarrollo de este proyecto para formarme en el uso de las herramientas más utilizadas en las

grandes empresas del sector de los videojuegos. Como hemos citado anteriormente, los cuatro motores presentados tienen productos de éxito en su haber y aunque *Unity* es el motor más utilizado por los pequeños estudios independientes, el motor y el lenguaje de programación más ligados a las grandes productoras de videojuegos y al grueso del mercado del ocio digital son *Unreal Engine* y *C++*.

Otro motivo importante es el coste en tiempo y esfuerzo a asumir para aprender a usar uno de los motores y desarrollar un proyecto en solitario con él. Pese a que a priori podría parecer que este motivo decanta la balanza en contra de *Unreal Engine*, este motor cuenta con la mayor colección de material formativo y la comunidad más activa y colaborativa de los cuatro mencionados. Esto hace que, aunque *Unreal Engine* sea un motor más complejo que *GameMaker* y de una complejidad similar a *Unity*, las posibilidades de aprender los conceptos necesarios para acometer el proyecto y cualquier complicación que pudiera aparecer en el proceso sean mayores al usar *Unreal Engine* que usando cualquiera de las alternativas.

Además, *Unreal Engine* cuenta con *Blueprint*, un sistema de visual scripting que simplifica la programación de eventos, las funciones de la *IA* y otros factores que podrían ser excesivamente complejos a la hora de acometerlos usando únicamente un lenguaje de programación convencional.

Por último, la diversidad y potencia de las herramientas que componen *Unreal Engine* solo es comparable al catálogo de características de *CryEngine*. Teniendo en cuenta que el motor de *Epic Games* tiene una curva de aprendizaje más pronunciada que el de *Crytek*, lo que nos permitirá controlar sus funcionalidades mucho antes, obtener un mayor grado de comprensión del proceso de desarrollo y obtener mejores resultados.

Todas estas razones no hacen que *Unreal Engine* sea un mejor motor de videojuegos que el resto de los comparados, sino que es el que mejor cubre las necesidades de nuestro proyecto. Es importante dejar claro que no hay una herramienta mejor o una peor, cada una de ellas se adapta mejor a un propósito definido.

2.3 Tecnologías complementarias utilizadas junto con Unreal Engine

A continuación, vamos a realizar una descripción de las principales tecnologías utilizadas tanto sobre el motor *Unreal Engine* como de forma adyacente al mismo. De esta forma no solo aportaremos mayor comprensión al funcionamiento de *UE4,* sino que podremos aportar una visión de conjunto de todas las técnicas y tecnologías que intervienen en el proceso de desarrollo de un videojuego.

2.3.1 *Gameplay Framework*

El *Gameplay Framework* [64] es el conjunto estandarizado de clases y métodos de *Unreal Engine 4* que definen la lógica y comportamiento de los componentes de un juego creado con este motor. Dichas clases y métodos están codificados en *C++*, pero se pueden utilizar tanto en *C++* como en *Blueprint*.

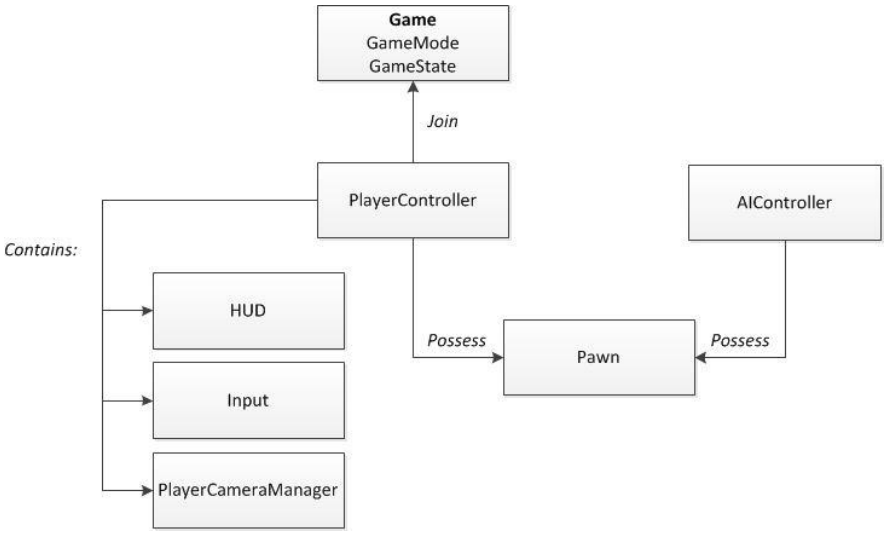


Figura 24. Diagrama de flujo de la comunicación entre clases del Gameplay Framework.

Entre las funcionalidades incluidas se encuentran aquellas necesarias para representar jugadores, aliados, enemigos, objetos interactivos, así como aquellas encargadas de permitir el control de avatares por parte del jugador o la *IA*. Por último, incluye clases que representan los diferentes estados y reglas que rigen tanto el juego como a cada jugador.

2.3.2 Microsoft Visual Studio 2015

Visual Studio [62] es el *IDE* con el que se integra el *Gameplay Framework* de *Unreal Engine 4* sobre *Windows* y la versión de 2015 es un prerrequisito para la instalación de *UE4* en este sistema operativo.

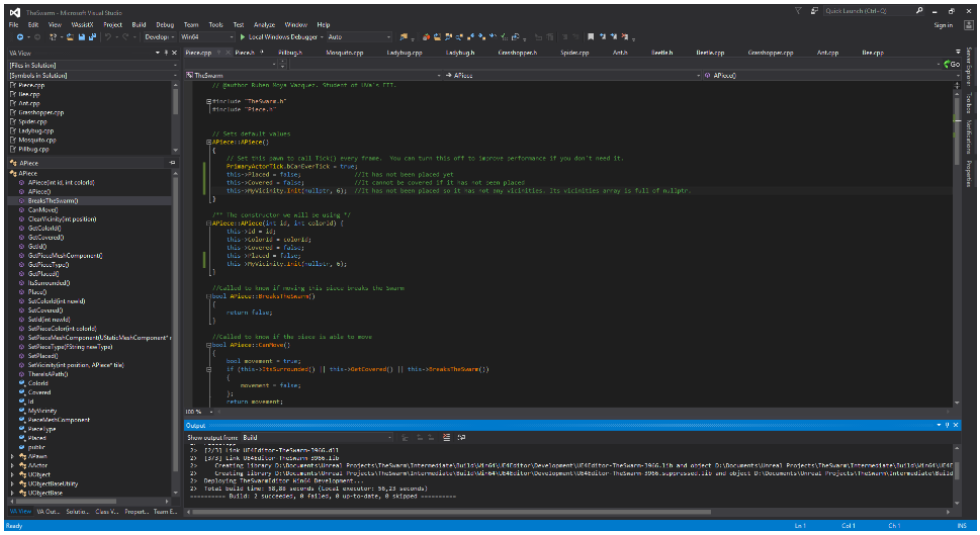


Figura 25. Captura de pantalla del Visual Studio 2015 con Gameplay Framework.

Es un entorno de desarrollo integrado disponible para una variada cantidad de lenguajes de programación, pero en nuestro caso lo utilizaremos para implementar y compilar el código *C++* con el que daremos forma a la lógica del juego.

Además, utilizaremos el plugin *Visual Assist* [63] de *Whole Tomato Software*, muy extendido entre la comunidad de desarrolladores de *UE4* ya que agiliza y facilita muchas tareas rutinarias relacionadas con *C++*.

2.3.3 Blueprint

Blueprint [65] es el sistema de *visual scripting* utilizado por *UE4*. Es una de las funcionalidades más potentes del motor ya que permite a gente con pocos conocimientos de programación crear un juego prácticamente al completo sin escribir una sola línea de código.

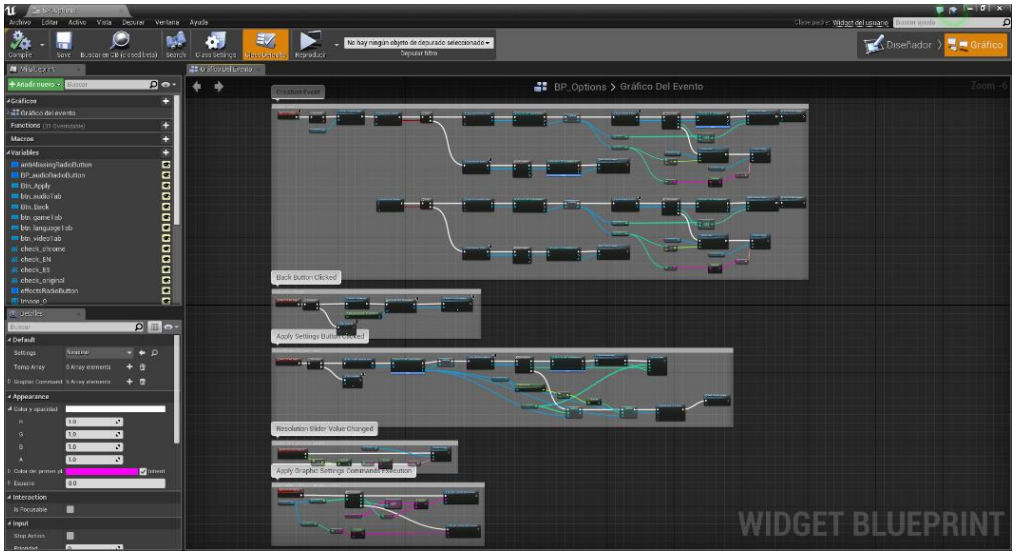


Figura 26. Captura de pantalla de parte del archivo Blueprint del widget de Opciones.

Sirve a su vez como nexo de unión entre los diseñadores de niveles e interfaz y los programadores, ya que muchas funciones y objetos creados a partir de código son accesibles vía *Blueprint* y viceversa. Es una herramienta muy potente que simplifica la creación de widgets, navegación entre interfaces de usuario, conservación de configuraciones y la captura de eventos.

2.3.4 Autodesk Maya 2017

Autodesk Maya [67] es una herramienta de desarrollo de gráficos 3D que nos permite crear y exportar con facilidad todos los modelos y mallas estáticas generadas en este programa a nuestro proyecto. Permite a su vez *skeletal meshes* con los que realizar animaciones de personajes que luego exportar a nuestro juego.

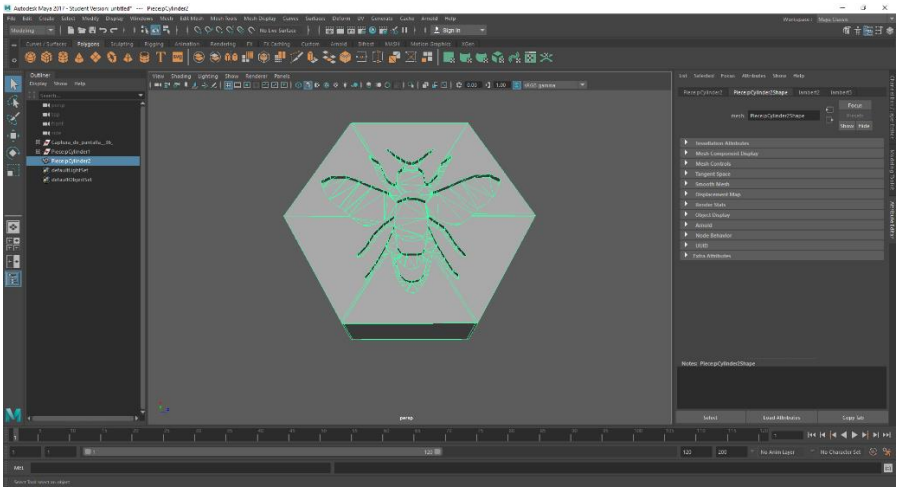


Figura 27. Captura de pantalla de la creación de una de las mallas estáticas del juego.

En nuestro caso, como se puede ver en la figura anterior, lo utilizamos para la creación de las piezas y las casillas.

2.3.5 Editor de Materiales

El editor de materiales [66] es una de las diversas herramientas integradas en Unreal Engine 4. Está en concreto consiste en una interfaz gráfica basada en nodos que nos permite la creación de materiales a los que aplicarles cualidades físicas y posteriormente asignárselos a mallas, tanto estáticas como *skeletal meshes*.

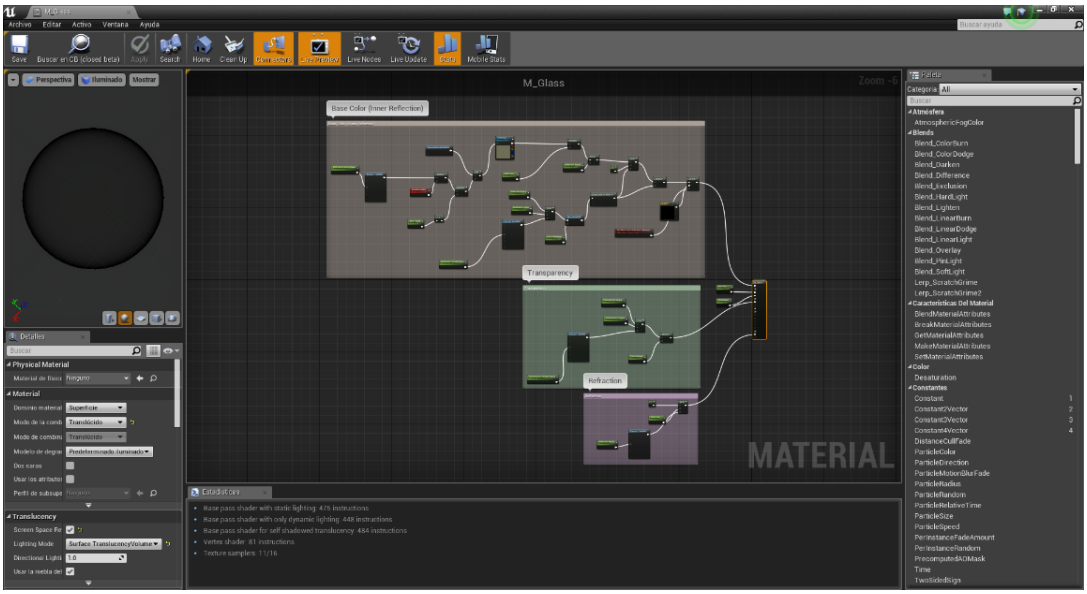


Figura 28. Captura de pantalla de Editor de Materiales.

De esta forma podemos, a partir de texturas simples, crear materiales muy complejos como el agua, la madera, el cristal o metales con cualidades físicas como la refracción de la luz, la opacidad o la emisión de luz.

2.3.6 Paint.net

Paint.net [68] es un editor fotográfico gratuito que compagina la potencia de edición de otros editores como Adobe Photoshop con la simplicidad del tradicional Paint. Aunque en un primer momento pudiese parecer que trivial, todo el proceso de creación de fondos, iconos, botones, logos y otro tipo de imágenes ha sido realizado usando este programa.

Figura 29. Captura del proceso de creación del logo.

3 Documento de Diseño del Videojuego (GDD)

A continuación, desarrollaremos el documento de diseño del videojuego o *GDD* (*Game Design Document*) [1] [82] [83] [84]. En él se especifican diversas características del juego que lo definen y sientan las pautas del desarrollo. Dado que gran parte del *GDD* son los documentos de análisis y diseño e implementación habituales de cualquier producto software y que dichos aspectos los trataremos en capítulos posteriores de este documento, reduciremos el *GDD* a los puntos característicos del mismo. Así mismo, como otros puntos ya los hemos tratado con anterioridad, también nos los ahorraremos.

Teniendo en cuenta que el juego de mesa del que partimos para crear nuestro videojuego ya existe, otro punto fundamental será el de describir las reglas de juego, las piezas, sus movimientos característicos y aspectos puntualmente destacables que lo diferencian de otros juegos similares.

Por último, realizaremos un pequeño análisis de videojuegos similares para buscar aspectos deseables a replicar en nuestro juego y fallos comunes a evitar.

3.1 Motivación del proyecto

El objetivo del videojuego es implementar una versión 3D, entretenida y dinámica del juego *Hive: La Colmena* diseñado por *John Yianni*, publicado por *Gen42 Games* [81] y ganador de la edición de 2006 de *Mensa Select Mind Games*. El juego será desarrollado por *Rubén Moya Vázquez* usando *Unreal Engine 4.15* con el objetivo final de superar el *Trabajo de Fin de Grado* de la titulación de *Grado en Ingeniería Informática* con mención en *Tecnologías de la Información*.

3.2 Detalles básicos del videojuego.

Figura 30. Logo de The Swarm

3.2.1 Nombre

Nuestro juego se llamará *The Swarm* haciendo alusión a los conceptos que caracterizan al juego de mesa del que proviene la idea original, *Hive: La Colmena*.

3.2.2 Genero

Estrategia/Casual/Indie.

3.2.3 Plataforma y requisitos mínimos

Nuestro juego estará disponible para PC y sus requisitos mínimos son los siguientes:

- Procesador de 1 GHz de 32 bits
- 1 GB de RAM
- Tarjeta gráfica DirectX 10

3.2.4 Objetivo Demográfico

Debido a que el videojuego será una adaptación del juego de mesa original *Hive: La Colmena* y no pretendemos añadir ningún aspecto que pueda modificar el rango de edades a las que está dirigido; el público objetivo será el mismo que el del juego del que partimos únicamente acotado por cuestiones de idioma. Es decir, jugadores mayores de 9 años de habla española o inglesa.

3.2.5 Perspectiva

Por defecto el juego se muestra en perspectiva isométrica con capacidad para rotar 360º sobre el eje z. Admite la opción de perspectiva cenital.

3.3 Objetivo del videojuego

El objetivo del juego es rodear por completo a la Abeja Reina del oponente mientras se evita que el rival haga lo mismo. A la hora de rodear a la reina rival cuentan todas las piezas en juego, tanto las propias como las del oponente. El primer jugador en conseguir dicha meta gana la partida.

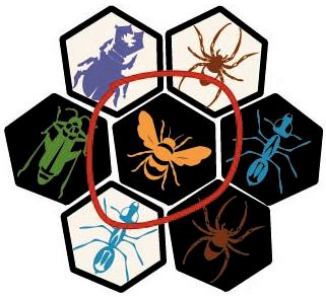


Ilustración 1. Representación del objetivo del juego.

3.4 Asunciones previas

- Todos los usuarios del juego disponen de los dispositivos necesarios para ejecutar el juego.
- La versión inicial del juego a desarrollar es la versión básica de *Hive: La Colmena* y sus expansiones.
- Los usuarios del juego entienden los conceptos básicos habituales de navegación por menús.
- El modo de juego se desarrolla, por turnos, en la misma instancia del juego.
- No se desarrollarán modos de juego en red.

3.5 Dinámica de juego y reglas

3.5.1 Preparación

En caso de estar jugando a la versión básica, cada jugador tiene a su disposición 11 piezas (blancas o negras). Estas 11 piezas constaran de: 1xAbeja Reina, 2xEscarabajo, 2xAraña, 3xSaltamontes y 3xHormiga Soldado.

En caso de jugar con las expansiones, al set básico de piezas se añadirán 1xMosquito, 1xMariquita y 1xCochinilla por jugador.

3.5.2 Situación inicial

El juego comienza cuando uno de los jugadores sitúa una de sus piezas en el tablero de juego. A continuación, el otro jugador deberá hacer lo mismo colocando la pieza que seleccione en contacto con cualquiera de los bordes de la pieza colocada anteriormente por el rival. A partir de este momento, los participantes juegan por turnos colocando piezas o moviendo las que ya se encuentren en juego bajo su control.

3.5.3 Reglas de colocación y movimiento

Las reglas están extraídas directamente del manual de instrucciones incluido en la edición original del juego.

3.5.3.1 Colocación

Se pueden introducir nuevas piezas en cualquier momento. Al colocar una pieza nueva, con excepción de la primera, esta no puede quedar en contacto con ninguna pieza rival.

No es necesario colocar todas las piezas para ganar, pero una vez una pieza entre en juego, no podrá ser retirada.

3.5.3.2 Colocación de la Abeja Reina

Cada jugador debe colocar su *Abeja Reina* dentro de sus cuatro primeros turnos. Si al llegar el cuarto turno no ha colocado aun su *Abeja Reina*, este deberá ser su movimiento.

3.5.3.3 Regla de la colmena

Las piezas en juego forman el tablero de juego, al que llamamos "*la colmena*" y pueden extenderse en cualquier dirección.

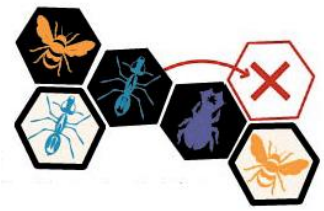


Ilustración 2. Representación de la Regla de la Colmena.

No se podrá realizar ningún movimiento que suponga la división de la colmena incluso cuando este movimiento reconecta la colmena ya que, en el proceso, la colmena se encontraría dividida. Por tanto, ninguna pieza en juego puede estar desconectada del resto de la colmena.

Ilustración 3. Representación complementaria de la Regla de la Colmena.

3.5.3.4 Regla de movimiento

Una vez un jugador haya colocado su abeja reina en juego y no antes, podrá elegir entre seguir colocando nuevas piezas en la colmena o mover una de las que ya se encuentren en juego.

Cada insecto tiene una forma particular de moverse en la colmena. Una vez una pieza está en juego ésta ya puede moverse con libertad sin restricciones de contacto con las piezas del contrario.

Ilustración 4. Representación de la regla de movimiento.

Tal y como dicta la *Regla de la colmena*, si una pieza es el único nexo de unión entre dos secciones de la colmena, esta no podrá moverse.

Ilustración 5. Representación de situación contraria a la regla de movimiento.

Los insectos solo podrán moverse deslizándose; si una pieza está rodeada hasta tal punto que no puede moverse físicamente con un deslizamiento, entonces no podrá hacerlo. Las únicas excepciones a esta regla son el *Saltamontes*, que puede saltar hacia o desde un espacio libre; el *Escarabajo* y la *Mariquita*, que pueden encaramarse a otras piezas; una pieza movida por efecto de la *Cochinilla* y el *Mosquito* siempre y cuando éste se encuentre inicialmente en contacto con un *Saltamontes*, un *Escarabajo* o una *Mariquita*.

Ilustración 6. Representación complementaria contraria a la regla de movimiento.

Si un jugador ya ha situado todas sus piezas en juego y/o es incapaz de mover o colocar ninguna de ellas su turno pasará al oponente. Se continuará saltando el turno de este jugador hasta que le sea posible colocar o mover una pieza o acabe el juego.

3.5.4 Piezas

Para describir las piezas distinguiremos entre el set básico de piezas y las expansiones con las que se puede contar, o no, a la hora de jugar.

3.5.4.1 Set Básico

Estas piezas forman el conjunto mínimo de piezas indispensables a la hora de jugar a *Hive: La Colmena*.

3.5.4.1.1 Abeja Reina

La *Abeja Reina* puede mover únicamente un espacio por turno, respetando las reglas anteriormente citadas.

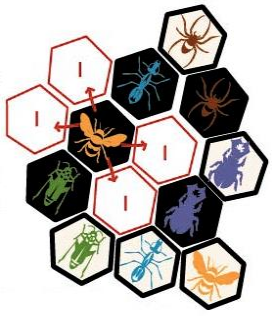


Ilustración 7. Movimiento de la Abeja Reina.

3.5.4.1.2 Escarabajo

El *Escarabajo*, al igual que la *Abeja Reina*, puede mover solo un espacio por turno, pero a diferencia del resto de insectos del set básico puede subirse encima de otro insecto indistintamente del color de la pieza que quede debajo. La pieza situada bajo el escarabajo quedará bloqueada y a efectos de colocación de nuevas piezas, esa posición será del color del escarabajo que se encuentre en la parte superior.

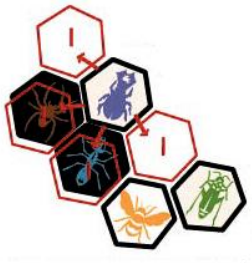


Ilustración 8. Movimiento del Escarabajo.

Un *Escarabajo* situado sobre otra pieza puede ser bloqueado por otro escarabajo que le sea colocado encima. No hay restricciones a la hora de apilar los escarabajos, pudiendo llegar a estar los cuatro apilados sobre una quinta ficha.

A la hora de colocar un escarabajo en juego, debe hacerse sin tener en cuenta su movimiento especial. Es decir, un *Escarabajo* no puede entrar en juego directamente apilado sobre otra ficha.

3.5.4.1.3 Araña

La *Araña* puede moverse exactamente tres espacios por turno. Este movimiento deberá realizarse manteniendo siempre el contacto con la colmena y sin poder retroceder en el proceso.

Ilustración 9. Movimiento de la Araña.

3.5.4.1.4 Saltamontes

El *Saltamontes*, al contrario que el resto de piezas, no puede moverse alrededor de la colmena. En su lugar, salta desde su posición al siguiente espacio libre siguiendo una línea recta de piezas conectadas. El *Saltamontes* no puede saltar por encima de espacios vacíos.

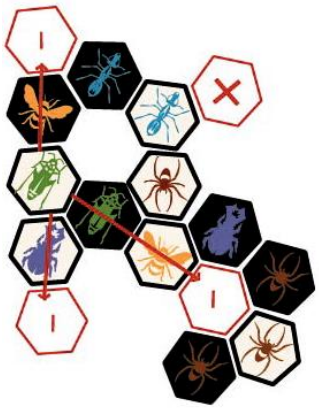


Ilustración 10. Movimiento del Saltamontes.

3.5.4.1.5 Hormiga Soldado

La *Hormiga Soldado* puede moverse a cualquier alrededor de la colmena. Puede moverse el número de espacios que nos convenga siempre que respete las reglas comunes de movimiento.

Ilustración 11. Movimiento de la Hormiga Soldado.

3.5.4.2 Expansiones

Estas piezas pertenecen a expansiones posteriores a la salida de la edición original del juego. A la hora de jugar se puede añadir cualquier subconjunto de ellas siempre que se añada por igual para ambos jugadores.

3.5.4.2.1 Mosquito

El *Mosquito* entra en juego como el resto de piezas, pero a la hora de moverse adquiere las características de movimiento de una de las piezas con las que esté en contacto, a elección del jugador.

Si se sube a la colmena actuando como un Escarabajo, continuará moviéndose como un Escarabajo hasta que baje de ésta.

Si un mosquito solo está en contacto con otro mosquito, éste no podrá moverse hasta que entre en contacto con una pieza distinta.

3.5.4.2.2 Mariquita

La *Mariquita* tiene un movimiento similar al resultado de mezclar a la *Araña* y el *Escarabajo*. Debe moverse exactamente tres posiciones siendo las dos primeras por encima de la colmena y la última para bajar de esta. No puede moverse alrededor de la colmena ni puede acabar su movimiento sobre ésta, por lo que no puede bloquear piezas como el *Escarabajo*, pero al igual que éste, puede moverse hacia y desde espacios inaccesibles para otras piezas.

3.5.4.2.3 Cochinilla

La Cochinilla mueve como la abeja reina; un espacio cada vez, pero también tiene una capacidad especial y es que puede mover una pieza adyacente no apilada, amiga o enemiga, dos espacios, el primero encima de la Cochinilla y el segundo a un espacio vacío adyacente a la misma.

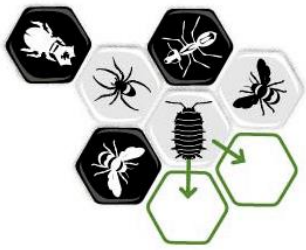


Ilustración 12. Movimiento de la Cochinilla.

Excepciones

- La Cochinilla no puede mover la última pieza que haya movido el contrario.
- La Cochinilla no puede mover una pieza apilada.
- La Cochinilla no puede mover una pieza si con ello rompe la colmena (no puede contravenir la regla de la colmena).
- La Cochinilla no puede mover una pieza a una posición con piezas apiladas (no puede contravenir las reglas de movimiento).

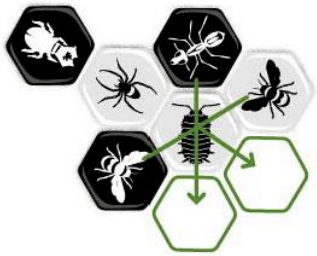


Ilustración 13. Habilidad especial de la Cochinilla.

Cualquier pieza que haya sido movida por la Cochinilla no puede moverse en el siguiente turno ni usar sus capacidades especiales. El Mosquito puede imitar el movimiento o la capacidad especial de la Cochinilla incluso cuando la Cochinilla ha sido movida por otro jugador y debe permanecer inmóvil ese turno.

3.5.5 Final del juego

El juego termina en cuanto una de las dos Abejas Reinas es rodeada por completo, independientemente del color de las piezas que la rodeen. El jugador al que pertenezca la Abeja Reina rodeada es el que pierde la partida.

Se puede producir una situación de empate si ocurre una de las siguientes dos situaciones:

- El movimiento de la última pieza en rodear una Abeja Reina provoca que la otra quede también rodeada.
- Los dos jugadores han llegado a una situación en la que se ven obligados a mover dos piezas constantemente para pasar turno sin que esto varíe el desarrollo de la partida.
-

3.5.6 Jugabilidad

A continuación, se describen determinados aspectos relacionados con el videojuego tal y como lo percibe el usuario. Estos aspectos no dependen del juego de mesa en el que nuestro videojuego está inspirado, sino que son puramente relativos a nuestra adaptación.

3.5.6.1 Modos de Juego

El videojuego cuenta con un único modo de juego. Dicho modo de juego es “Jugador VS Jugador” en el que dos usuarios, por turnos, juegan una partida usando la misma máquina.

3.5.6.2 Elementos del juego

3.5.6.2.1 Tablero

En la versión original del juego, éste no tiene tablero, sino que son las propias piezas las que lo forman. Para trasladar *Hive: La Colmena* a un videojuego, por el contrario, necesitaremos que sí que exista dicho tablero, que tendrá ciertas particularidades.

En primer lugar, el tablero, que tendrá forma de hexágono, estará formado por casillas hexagonales y la distancia entre dos lados opuestos será de 32 casillas (las 14 fichas que puede tener en juego cada jugador, más dos casillas extra por cada extremo que aporten sensación de infinitud). Con el objetivo de perseguir dicha sensación de tablero infinito, la fila de casillas perimetral que forma la silueta del hexágono estará difuminada.

En caso de que el juego derive en la situación de que un movimiento provoca que una pieza ocupe una de las casillas colindantes con las casillas perimetrales, en el siguiente turno el tablero aparecerá modificado para restaurar la situación inicial, manteniendo estática la colocación de las piezas.

3.5.6.2.2 Piezas

Las piezas son prismas hexagonales de color blanco o negro según el jugador al que representen. Dichas piezas tienen, en su cara superior, el icono que indica el tipo de pieza que simbolizan en distintos colores de forma que sean fácilmente reconocibles. Con el objetivo de aumentar la accesibilidad a jugadores con problemas de visualización cromática, se podrá activar una opción para jugar con las fichas en blanco y negro de la edición *Hive Carbon*.

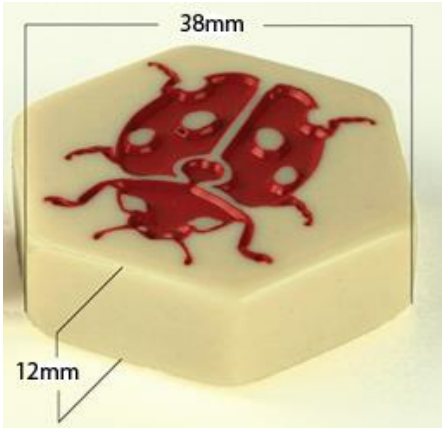


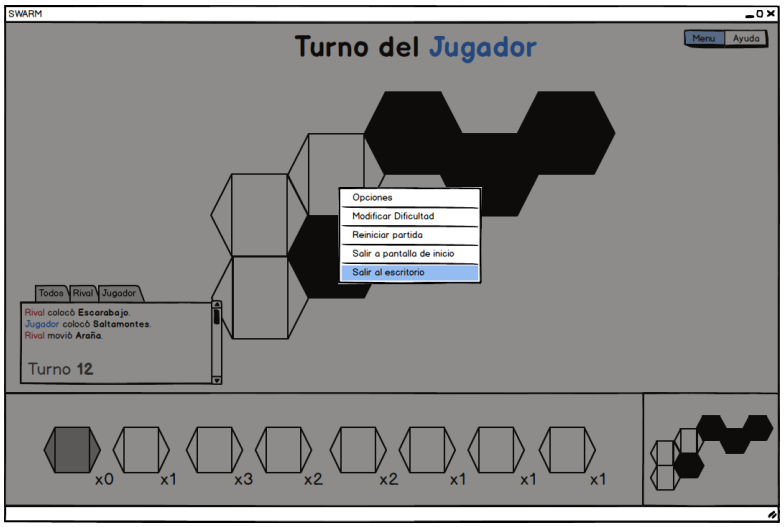
Figura 31. Dimensiones de las piezas del juego original.

Las piezas tienen unas medidas de 12mm de alto, 22mm de ancho cada lado del hexágono y una distancia entre caras opuestas de 38mm. Con el objetivo de mantener el "*look and feel*" que percibirá el jugador aumentando la visibilidad de las piezas, mantendremos esa relación de aspecto, pero en unas medidas mayores.

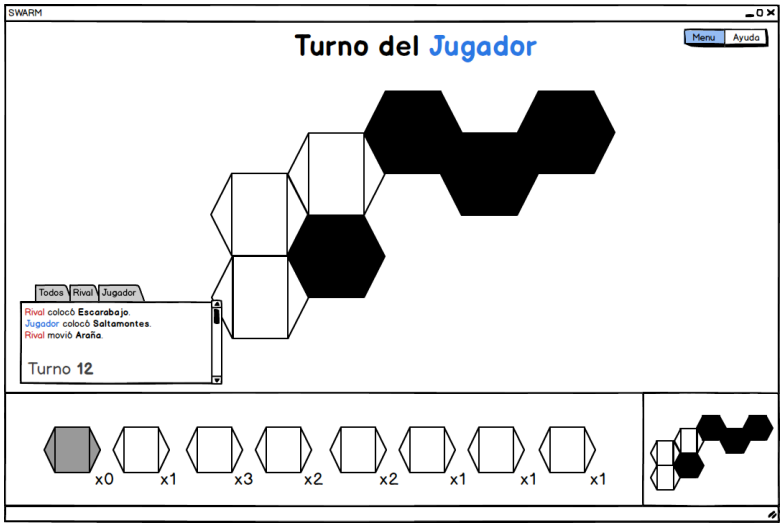
3.5.7 HUD

El *HUD* (*Heads-Up Display*) es el conjunto de toda la información que se muestra constantemente al usuario. Dicho conjunto de información está estructurado conforme a las convenciones sociales y estándares habituales en este tipo de juegos.

En primer lugar, en la parte superior derecha de la pantalla se mostrará una pequeña barra con acceso al menú de opciones y el menú de ayuda. Al hacer click sobre el botón del menú, se pausará la partida y se desplegará un menú con opciones para salir a la pantalla de inicio, salir al escritorio, reiniciar la partida y modificar la dificultad. Por otra parte, al hacer click en el menú de ayuda se mostrarán dos opciones, una para activar o desactivar la ayuda al movimiento de piezas dentro del juego y la otra para ver el manual de juego.



En segundo lugar, el 80% superior de la pantalla estará ocupado por el espacio de juego, que está compuesto por el tablero y sus alrededores vistos inicialmente en una perspectiva isométrica sobre la que podremos rotar, hacer zoom y cambiar a perspectiva cenital. Dentro de esta sección, en la parte superior central se indica a quien pertenece el turno actual, en texto e icono coloreado. En la parte inferior izquierda de esta sección se muestra el historial de movimientos en modo *scroll*, en transparencia mientras no se sitúe el cursor sobre él, con pestañas para filtrar los movimientos del jugador, del rival y todos los realizados. Bajo dicho historial se sitúa el contador de turnos que nos indica que turno es el actual.



Por último, el 20% inferior de la pantalla está ocupado por nuestro banquillo de piezas, en el que se muestran las piezas que no hemos puesto en juego aun y en el extremo derecho un “mini-mapa” con una visión cenital simplificada de la partida.

3.6 Análisis de videojuegos similares existentes

Con el objetivo de crear un juego que proporcione una experiencia lo más divertida y gratificante posible para los jugadores, siguiendo un procedimiento profesional de desarrollo y manteniendo unos estándares de calidad y usabilidad adecuados; vamos a realizar un análisis de videojuegos similares al que pretendemos realizar. De este análisis obtendremos una serie de aspectos positivos a replicar en nuestro proyecto y un conjunto de errores o características poco adecuadas que deseamos evitar.

Teniendo en cuenta que *Hive: La Colmena* es un juego de mesa relativamente popular, para el análisis de videojuegos existentes contamos con las 3 versiones distintas del juego que cuentan con licencia oficial por parte de *Gen42 Games*. Además, para realizar un análisis de mayor calidad, analizaremos tres de los juegos de mesa mejor valorados por la crítica, para *PC*. De esta forma no solo tendremos un análisis de aspectos propios del juego a desarrollar sino también de estándares y normas de uso de juegos del mismo género.

De todos los aspectos positivos y negativos observados en el análisis de estos juegos, obtendremos una serie de requisitos funcionales y no funcionales que se verán reflejados en el posterior documento de análisis.

3.6.1 Análisis de versiones existentes de *Hive: La Colmena*

Las tres versiones de *Hive: La Colmena* con licencia oficial por parte de *Gen42 Games* son *Hive*, de *BlueLine Games*, lanzado en 2013 vía *Steam* para *PC, Mac y Linux*, disponible también para *Xbox 360*; *Hive™ - board game for two*, de *Leviteo Ltd.*, lanzado para *iOS* y *Android* en 2014 y la versión web disponible en *BoardSpace*, de la que no hemos podido obtener información respecto a la fecha de publicación y autor. Como podemos ver, las tres versiones tienes soportes, fechas y desarrolladores distintos, pero para sacar patrones replicables nos abstraeremos de dichos factores y nos fijaremos únicamente en funcionalidad y comportamiento.

3.6.1.1 Hive de BlueLine Games

La versión de *Hive: La Colmena* para *PC* distribuida vía *Steam* [87] es la versión más completa y actualizada de todas las que cuentan con licencia por parte de *Gen42 Games*. Por defecto se ejecuta en modo ventana, lo que podría llegar a ser desagradable para algunos jugadores, pero esto se puede corregir a través del menú de opciones.

Figura 32. Captura del menú principal de Hive.

Su menú principal, creado a base de botones hexagonales unidos a modo de fichas del juego, cuenta con cuatro modos de juego como submenú de la opción “Jugar”, un enlace a la tienda de contenidos, créditos, estadísticas y las reglas del juego.

El primer defecto que encontramos con solo navegar es su interfaz poco cuidada, con botones, ventanas y menús de estética bastante rudimentaria. Además, desde las opciones no podemos seleccionar el idioma del juego, que estará ligado al idioma que tenemos vinculado a nuestro perfil de *Steam*. Por otro lado, a la hora de seleccionar el modo de juego, no se explica en que consiste cada uno y si bien “VS Ordenador”, “2 Jugador” y “En Línea” son bastante evidentes para los jugadores habituales, “Pasar Turno y Jugar” no se entiende bien en que consiste. Como añadido, una vez averiguado en que consiste cada modo, vemos que “Pasar Turno y Jugar” y “2 Jugador” solo se distinguen en caso de que los dos jugadores, en el segundo modo, cuenten con un controlador distinto, de otra forma son exactamente el mismo modo de juego. Finalmente, durante la partida, las piezas de cada jugador se colocan en los laterales de la pantalla, en lugar de la parte inferior y superior como es costumbre y si bien en cada lateral aparecen imágenes representativas de cada jugador, puede llegar a ser confuso hasta acostumbrarse.

Figura 33. Captura del estado final de la partida.

Una vez elegido el modo de juego, los modos locales nos redirigen a la pantalla de configuración de partida en la que podremos elegir quien comienza, que color usa cada jugador, que expansiones se usan, si se usan reglas de torneo y en caso de jugar contra el ordenador, la dificultad de la IA.

3.6.1.2 Hive™ - board game for two de Leviteo Ltd.

La versión para dispositivos móviles de *Hive: La Colmena* [88] es probablemente la peor de las versiones en cuanto a calidad del software si tenemos en cuenta que es la más reciente. Cuenta con acabados y una navegación poco profesional, está exclusivamente en inglés, el multijugador no funciona debido a constantes errores en el servidor y el menú de opciones, así como el configurador de partidas, que solo cuenta con las opciones de cambiar los colores de las fichas y añadir dos de las tres expansiones, fallan constantemente a excepción del submenú de sonido que es el único que funciona de manera esperada.

Figura 34. Menú principal de Hive™ - board game for two.

Durante la partida, no avisa de que jugador es el turno actual, ni nos informa de que debemos poner la abeja en el cuarto turno si no la habíamos colocado anteriormente, lo cual provoca que el usuario perciba que el juego se ha bloqueado o ha fallado en caso de seleccionar la abeja para ser colocada. Además, una vez la partida acaba, no se informa de que jugador ha ganado y directamente se salta a un letrero con el texto "GAME OVER".

Figura 35. Captura durante una partida mediante el método hotseat.

Por otro lado, los puzles son un modo de juego en solitario interesante que va aumentado en complejidad según se van superando los niveles. La forma de presentar cada puzle deja que desear por su falta de dinamismo y sus tipografías y botones carentes de estilos.

3.6.1.3 Hive web, versión de BoardSpace

BoardSpace es un portal web que cuenta con una serie de *applets java* que nos permiten jugar a diversos juegos, entre ellos, *Hive: La Colmena* [89]. Pese a contar con una estética y una navegación bastante anticuadas, a nivel de jugabilidad, funciones y configuración es muy completo, superando de largo a la app de *Leviteo Ltd.*

Obviando la estética y navegación arcaicas y la ausencia total de animaciones, el primer problema que encontramos es que, al requerir el uso de java para poder jugar, algunos navegadores como Google Chrome no son compatibles. Esto se soluciona a través de la descarga e instalación de un acceso directo que utiliza la *máquina virtual java* que tengamos instalada en nuestro equipo.

Los menús de la aplicación son muy completos, pero a la vez complejos y enrevesados. La interfaz de usuario está mal distribuida y cuenta con mucho espacio desaprovechado. El único componente que cuenta con animaciones es el historial de movimientos. Al igual que la aplicación de *Leviteo Ltd.* la ayuda es un enlace al archivo pdf oficial de las reglas de juego.

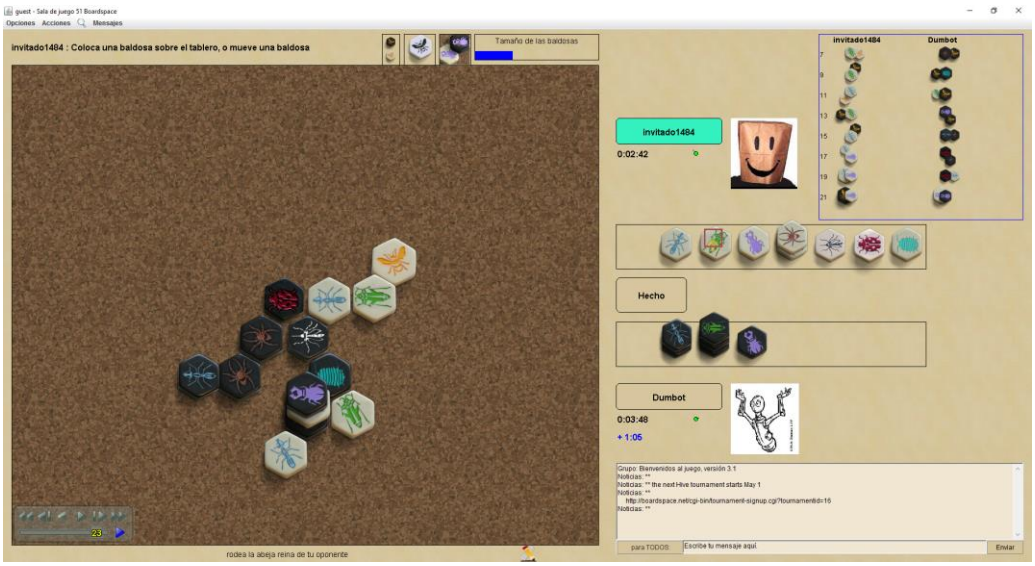


Figura 36. Captura de pantalla de una partida contra la IA en BoardSpace.

Como aspecto positivo, cabe destacar que cuenta con un historial de movimientos visual que nos permite reproducir la partida hasta el último movimiento realizado y un historial en modo iconográfico de los movimientos realizados por cada jugador. Además, cuenta con un chat que a la hora de jugar online puede ser muy útil.

3.6.2 Análisis de juegos de PC del genero estrategia/tabletop

A la hora de elegir los otros tres juegos a evaluar para nuestro análisis hemos querido tener en cuenta el número de jugadores activos con los que cuentan, la similitud con el juego a desarrollar, las valoraciones de los medios y las opiniones de la comunidad. Por ello, los elegidos han sido *Hearthstone* (2014, *Blizzard Entertainment*), juego de cartas aclamado por la crítica con más de 50 millones de jugadores repartidos entre dispositivos móviles, PC y Mac; *Civilization VI* (2016, *Firaxis*), sexta edición del juego de estrategia por turnos más popular, valorado con un 9.4 sobre 10 por la publicación estadounidense IGN y por último, *Tabletop Simulator* (2015, *Berserk Games*), videojuego indie disponible para PC y sistemas de realidad virtual, que engloba la mayor cantidad de juegos de mesa disponibles bajo el mismo título.

3.6.2.1 Hearthstone

Hearthstone [90] es posiblemente el juego que, de todos los analizados, mejor cuida los aspectos referentes a la usabilidad. Su proceso de carga, inicio y acceso al menú principal está integrado dentro de una única animación que integra las noticias y novedades en caso de que las haya. Esto hace que el proceso de carga inicial a penas se perciba por el usuario.

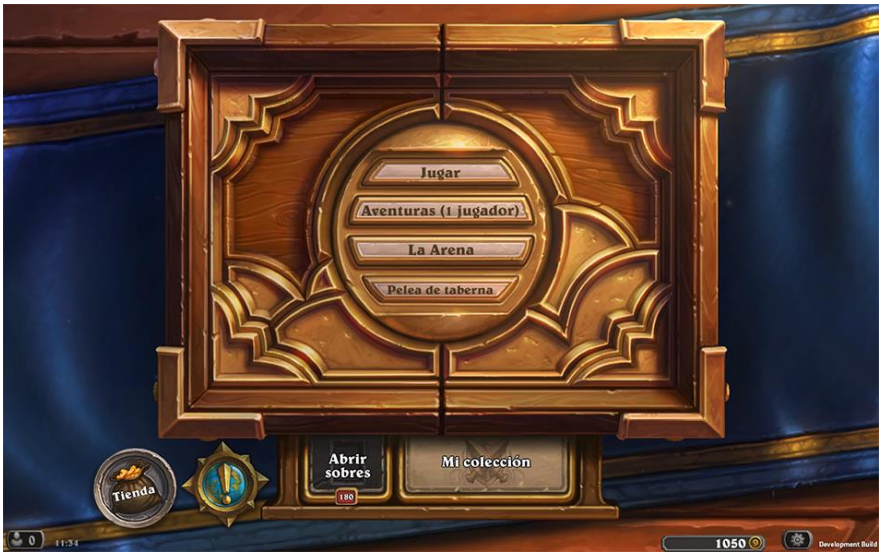


Figura 37. Captura del menú principal de Hearthstone.

El menú principal cuenta con cuatro opciones principales ligadas a modos de juego, que pueden estar presentes pero deshabilitadas en algunos casos; cuatro opciones secundarias relacionadas con aspectos de personalización y adquisición de nuevas cartas vía microtransacciones; animaciones y un aspecto visual integrado con el fondo y el resto de la navegación lo que lo hace sencillo, rápido y fácil de usar.

Durante la partida, *Hearthstone* cuenta con una serie de funcionalidades que facilitan el dinamismo en la partida y el seguimiento de esta por parte de los jugadores. Algunas de estas funcionalidades son un historial de acciones de los jugadores visual, avisos sonoros y visuales de inactividad, principio y fin de turno y un contador visual de tiempo cuando se nos va a acabar el turno.

El principal aspecto negativo de *Hearthstone* es la ausencia total de acceso a información de ayuda o tutoriales desde la aplicación. Si bien es cierto que la primera vez que se juega sí que se realiza un breve ejercicio de demostración, una vez realizado no hay más información accesible desde la propia aplicación.

Como apunte final, cabe destacar una curiosidad que el resto de los juegos analizados no presentan. Las partidas y diversos modos de juego cuentan con una serie de tableros diferentes con animaciones interactivas que, aunque no modifican la dinámica de juego, aportan un aspecto único a cada partida haciendo que resulten menos monótonas al usuario habitual y alargando de esta forma la vida útil del juego.

3.6.2.2 Civilization VI

La sexta entrega numerada de la saga *Civilization* [92] es uno de los mayores ejemplos de calidad y versatilidad del genero de estrategia por turnos. Pese a ser un juego con un nivel de complejidad relativamente elevado, el acceso constante a ayuda, el *HUD* optimizado y correctamente distribuido y el sistema de alertas, mantienen al jugador bien informado de todo lo que ocurre durante la partida y de las posibles acciones que tiene a su disposición.

Figura 38. Captura del menú principal de Civilization VI.

Civilization VI sigue la estela de sus predecesores y eso se nota en la calidad de todos los aspectos del videojuego. Cuenta con un menú principal elegante, con las opciones habituales de configuración, modos de juego, opciones y ayuda, además de un tutorial extenso pero fácil de seguir.

El nivel de personalización de cada partida gracias al configurador de partidas es, sin lugar a dudas, el más alto de todos los juegos analizados. Esto hace que el juego mantenga mucho más tiempo el interés del jugador y, por tanto, prolonga su vida útil.

Encontrarle defectos no relativos a la propia lógica interna del juego a *Civilization VI* es muy difícil dado que nos encontramos ante el juego abanderado de su género en el que se aprecia como sus desarrolladores han querido cuidar todos los aspectos apreciables por el jugador. Aun así, debido a su nivel de complejidad, el exceso de información y posibilidades, además de la larga duración de sus partidas son aspectos que pueden saturar a jugadores que no sean habituales del género.

3.6.2.3 Tabletop Simulator

De todos los juegos analizados *Tabletop Simulator* [91] es, sin lugar a dudas, el más versátil en cuanto a opciones de juego. Al ser un simulador de juegos de mesa que nos permite crear nuevos juegos dentro del mismo, sus opciones y modos de juego son prácticamente ilimitados. Por tanto, no analizaremos ningún minijuego o modo de juego concreto sino los aspectos generales que definen el juego, así como su interfaz, navegación y principales características.

Tabletop Simulator es un juego, publicado en 2015 por el estudio *indie Berserk Games* gracias a un *crowdfunding* en *Kickstarter*, que se encuentra enmarcado dentro de los géneros de *juegos de mesa*, *estrategia* y *sandbox*. Su versión básica, sin contar con *mods* o contenido descargable, cuenta con más de veinte juegos distintos entre juegos de mesa, puzles y juegos de azar. Cuenta con diversos modos de juego entre los que se encuentran multijugador cooperativo y competitivo, juego en solitario y juego en red local. De todos los juegos analizados es el único que cuenta con soporte para realidad virtual, en concreto para *HTC Vive* y junto con *Hearthstone*, es el otro único juego cuyo multijugador es global e independiente de la plataforma.

Figura 39. Captura del menú principal de Tabletop Simulator.

El menú principal es simple y cuenta con las opciones comunes al resto de juegos de su género. Este videojuego cuenta con distintas dinámicas de juego según el juego al que estemos jugando y no todas las acciones y formas de llevarlas a cabo son triviales, por tanto, en esta ocasión son, aun si cabe, mucho más importantes el tutorial y las ayudas para el jugador. Teniendo en cuenta la complejidad de sus distintas dinámicas de juego, un aspecto muy positivo es que permite al jugador usar el controlador que desee (mando de consola, *pads* de realidad virtual o teclado y ratón) y configurar el esquema de controles a su gusto.

El interfaz de usuario es bastante minimalista y solo cuenta con una barra de menú en la zona superior de la pantalla, con opciones relativas a la configuración de la partida, la mesa con el juego al que estemos jugando, un fondo de pantalla estático y una ventana de chat con transparencia en la esquina inferior izquierda que nos permite comunicarnos con el resto de jugadores. Un aspecto positivo es que cuenta con un chat de voz integrado, lo que nos permite omitir el chat escrito y comunicarnos con el resto de jugadores de forma hablada.

El principal aspecto negativo del juego teniendo en cuenta los premios que ha recibido y el número de jugadores es que tanto la interfaz como los juegos se encuentran exclusivamente en inglés. Otro aspecto que podría ser considerado negativo, aunque a priori puede parecer que no lo es, es el hecho de que sea un juego *sandbox* que nos otorga en algunos casos un exceso de libertad. Pese a que los juegos tienen unas reglas establecidas, nada obliga al jugador a respetarlas, pudiendo llegar a hacer trampas, equivocarse o incluso tirar la mesa con todo lo que hay sobre ella en un arrebato de ira.

Figura 40. Captura de parte del catálogo inicial de juegos de Tabletop Simulator.

Por último, cabe destacar su integración total con ***Steam Workshop***, plataforma de *Steam* que ofrece soporte a la modificación de videojuegos por parte de los usuarios. Si bien *Civilization VI* también cuenta con dicha integración, es en *Tabletop Simulator* donde se ve la potencia de otorgarle dichas herramientas a los usuarios ya que el juego actualmente cuenta con casi 14.000 *mods* diferentes subidos por la comunidad que demuestran que con este videojuego se puede replicar prácticamente cualquier juego de mesa existente.

3.7 Bocetos del juego

A continuación, adjuntamos una serie de bocetos [2] que ilustran el diseño inicial del videojuego y su comportamiento y navegación. Dichos bocetos son de la fase conceptual del desarrollo y podrían estar sujetos a cambios respecto a la versión final del juego.

Para la realización de los siguientes bosquejos hemos utilizado la herramienta *Balsamiq Mockups* [69] en su versión 3.5.8.

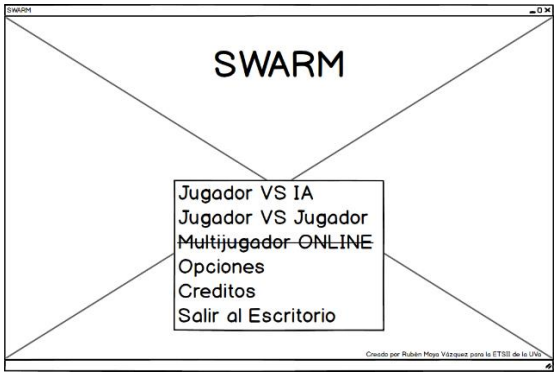


Figura 41. Boceto del Menú Principal.

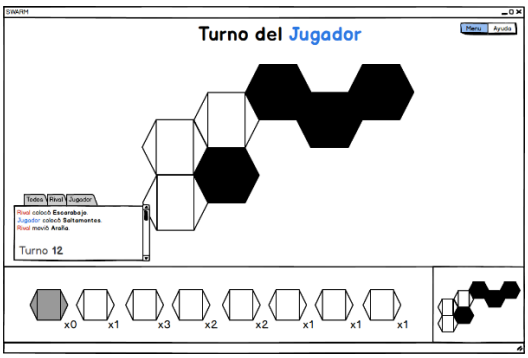


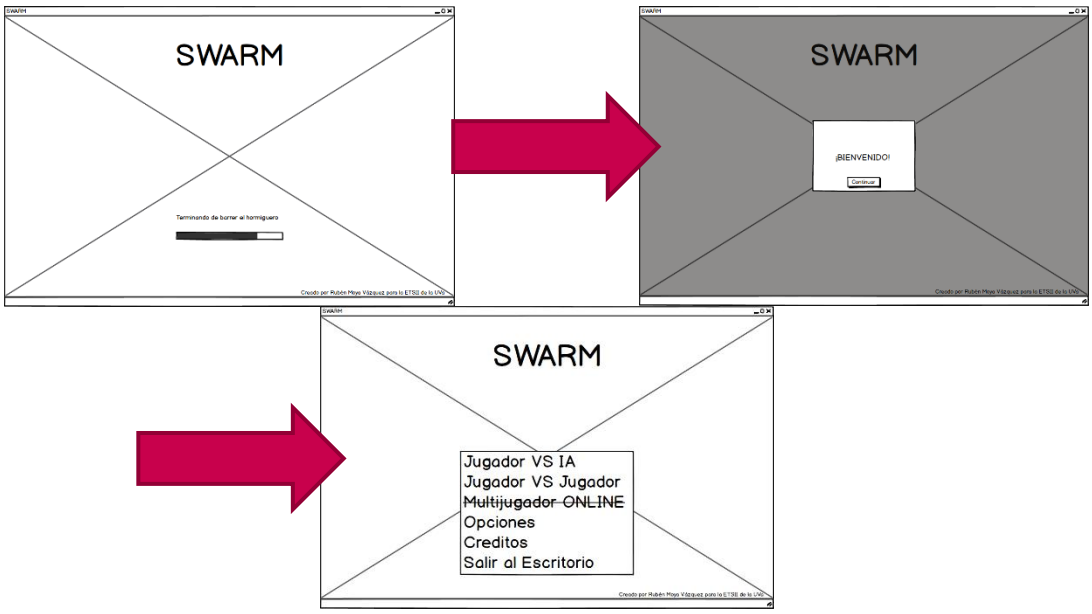
Figura 42. Boceto de Partida.

3.8 Prototipado

El prototipado del juego lo realizamos imprimiendo, a partir de los bocetos del apartado anterior, una serie escenas de situaciones supuestas relacionadas con la versión a desarrollar del juego, las piezas del juego original y dos voluntarios conocedores de la versión original del juego. Las situaciones planteadas se describen a continuación con apoyo visual:

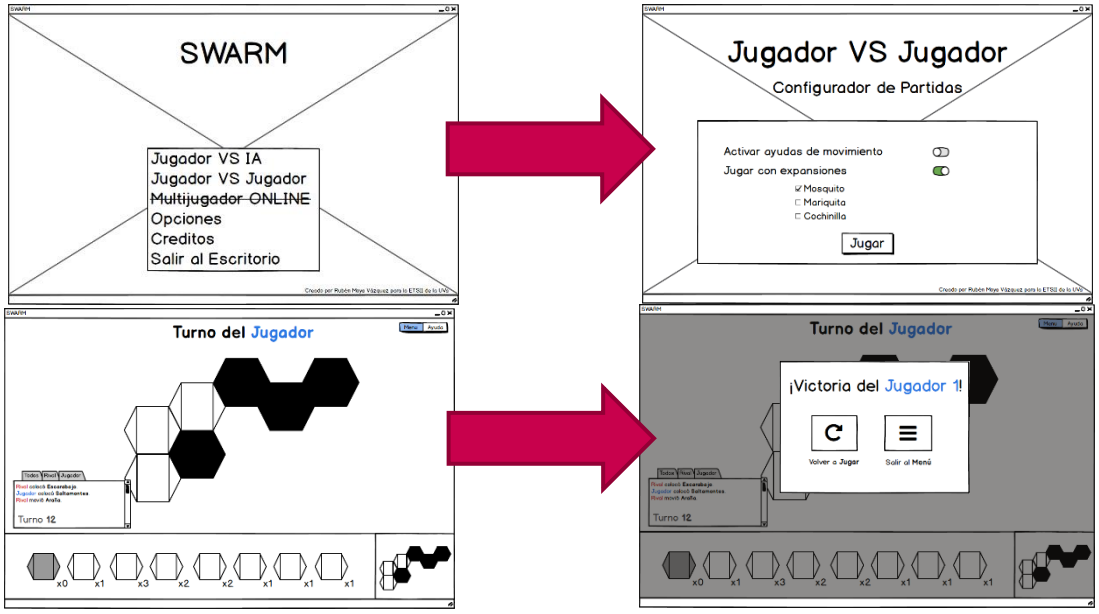
3.8.1 Inicio del videojuego

1. Pantalla inicial para indicar que se está realizando el proceso preliminar de carga de información. Para dinamizar la espera, se mostrará una breve animación con el logo del juego y en caso de que se prolongue más allá, mensajes en clave de humor.
2. En caso de existir avisos, mensajes o eventos, se mostrarán en este punto.
3. Despliegue del menú principal.



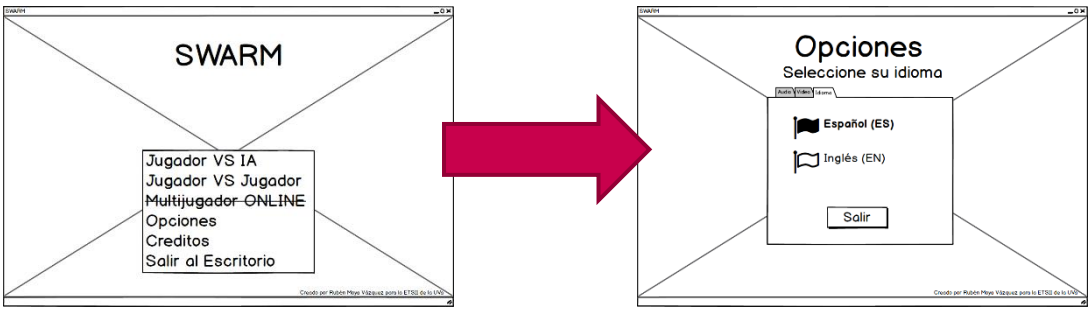
3.8.2 Jugar partida

1. En el menú principal, se seleccionará el modo de juego al que se desea jugar.
2. Se abrirá el menú de configuración de partida, donde dependiendo del modo de juego, podremos configurar una serie de opciones relativas a la partida, como las expansiones disponibles o la dificultad de la IA en caso de jugar contra la máquina.
3. Se desarrollará la partida por turnos hasta llegar a la situación final.
4. Se mostrará un mensaje informando del contendiente que ha ganado la partida y dando la opción de jugar otra partida o volver al menú inicial.



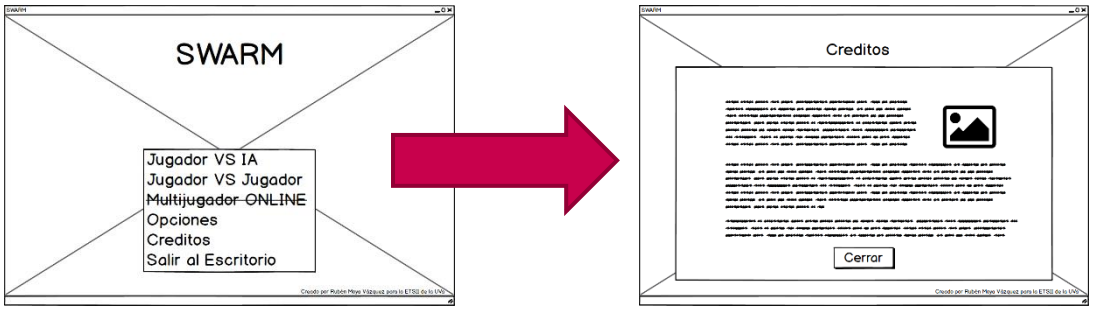
3.8.3 Configurar opciones generales

1. En el menú inicial, se pulsará en el botón “Opciones”.
2. Se abrirá el menú de opciones en el que se podrán configurar una serie de cuestiones relativas a los aspectos generales del juego, como el idioma o la configuración de sonido.
3. Si se realiza algún cambio, a la hora de salir de dicho menú, se nos pedirá confirmarlo de forma que sea permanente. En caso contrario se revertirán los cambios.



3.8.4 Ver Créditos

1. En el menú inicial, se pulsará en el botón “Créditos”.
2. Se abrirá un documento estático en el que se muestran los datos relativos a todos los componentes y personas involucrados en el desarrollo del proyecto, así como los agradecimientos.
3. Al pulsar “Cerrar” se devolverá al usuario al menú principal.



4 Desarrollo del Proyecto

En este capítulo detallaremos el proceso de desarrollo de software que seguiremos a la hora de llevar a cabo nuestro proyecto. Una vez descrito el proceso seleccionado, procederemos a desarrollar los documentos iniciales relativos al arranque del proyecto.

El primero de estos documentos es el documento de Gestión de Riesgos, en el cual detallamos los riesgos encontrados que podrían aparecer durante el proceso de desarrollo del proyecto, así como contramedidas para evitar su aparición e ideas para mitigar sus efectos.

El segundo y más importante es la Planificación del Proyecto, que consiste en la subdivisión de las fases del proyecto en tareas unitarias y la distribución de las mismas entre los recursos disponibles. De esta forma, el equipo de desarrollo sabrá que debe hacer en cada momento y cuánto tiempo tiene para hacerlo. Además, dicho documento incluye los costes del proyecto, con el objetivo de estimar el valor que debería tener el producto definitivo.

Finalmente se encuentra el documento de Seguimiento del Proyecto. El cual sirve para describir la planificación que se ha seguido finalmente y se registran los motivos para cualquier incidencia que se haya producido con el objetivo de evitar retrasos en futuros proyectos y planificar de manera más realista.

4.1 Proceso de Desarrollo

Para la realización de este proyecto hemos decidido seguir el *Proceso Racional Unificado*, *RUP* [71]. Este proceso de desarrollo de software tiene las siguientes características:

- **Desarrollo iterativo.** El proceso de desarrollo del software sigue un ciclo de vida que añade funcionalidad y estructura al proyecto con cada iteración del mismo.

- **Administración de requisitos.** Este proceso de desarrollo está enfocado en el análisis, realización y cumplimentación de los requisitos del sistema a través de la realización de los *Casos de Uso*.
- **Uso de arquitectura basada en componentes.** RUP enfoca el diseño de la arquitectura de sistema en el uso de módulos reutilizables frente al desarrollo de software monolítico.
- **Control de cambios.** Los cambios realizados en cada iteración del ciclo de vida quedan registrados en diferentes versiones y documentación de forma que cualquier diferencia entre el software diseñado y la versión final está justificada.
- **Modelado visual del software.** A través de los diferentes modelos de las fases de análisis y diseño se puede realizar un boceto de gran fidelidad respecto a la implementación final del software.
- **Verificación de la calidad del software.** RUP incluye una serie de pruebas de calidad que deben realizarse para asegurar una cantidad mínima asumible de posibles errores.

Todo proceso basado en RUP consta de cuatro fases caracterizadas por sus respectivos hitos. Dichas fases son:

1. **Inicio.** Esta fase tiene por hitos definir el proyecto y establecer el alcance y los objetivos del mismo. Los principales documentos entregables de esta fase con el de gestión de riesgos y la planificación del proyecto.
2. **Elaboración.** Esta fase se caracteriza por la identificación de los requisitos y casos de uso del sistema y la realización del proceso de diseño de la arquitectura del sistema. A su vez, esta fase se puede dividir en dos, separando la parte más abstracta (Fase de Análisis) de la parte más concreta (Fase de diseño), aportando en cada una de ellas su correspondiente documento.
3. **Construcción.** Esta fase consiste en la implementación del software partiendo de los documentos recabados en las anteriores fases. Al final de esta fase, la totalidad de los requisitos y casos de uso deben estar cubiertos por la funcionalidad del sistema.
4. **Transición.** Esta fase está enfocada en el proceso de validación a través de la realización de pruebas y la corrección de errores. Los entregables característicos de esta fase son el documento de pruebas y los manuales de instalación, administración y uso.

Estas fases se dividen a su vez en iteraciones de forma que la carga de trabajo pueda distribuirse de forma más homogénea entre los recursos y el tiempo del que disponemos. De la siguiente figura podemos extraer que mientras las fases de inicio y elaboración suelen tener una y dos iteraciones respectivamente, las fases de construcción y transición tienen un número indeterminado de iteraciones a definir por el líder del proyecto y el alcance del mismo.

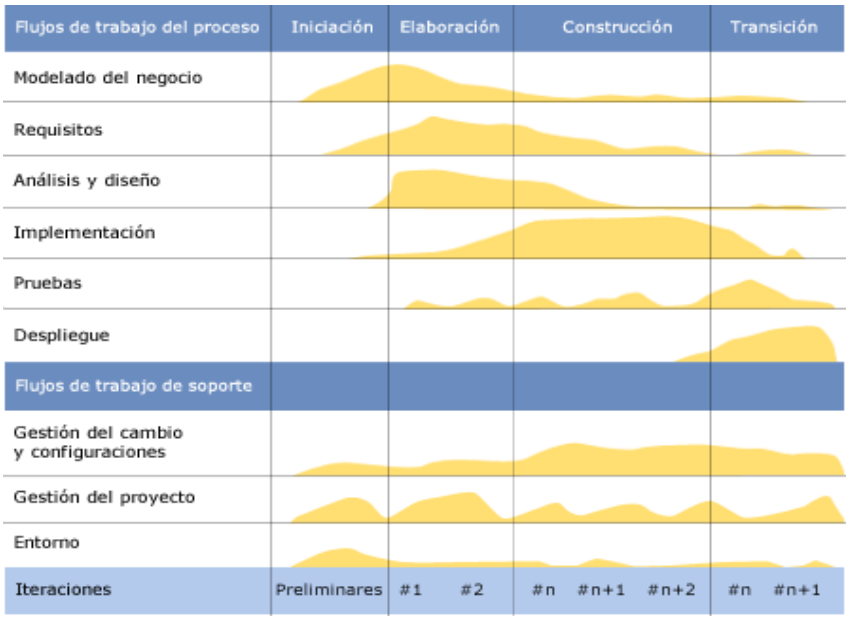


Figura 43. Relación de la distribución de carga de trabajo por fases e iteraciones.

4.2 Gestión de Riesgos

En este apartado se especifican los riesgos encontrados, así como la posibilidad de que desencadenen un fallo, las consecuencias del mismo y las contramedidas a aplicar.

Con el objetivo de medir la relevancia a través del impacto y la posibilidad de ocurrencia, clasificaremos el producto de estas dimensiones de la siguiente forma:

		PROBABILIDAD DE OCURRENCIA			
		Baja	Media	Alta	Muy Alta
Impacto	Crítico				
	Importante				
	Asumible				
	Irrelevante				

Tabla 1. Pantone de la severidad de un riesgo conforme al impacto y la probabilidad de ocurrencia.

Tal y como vemos en la tabla, un riesgo puede clasificarse en función de su probabilidad de ocurrencia y de su impacto.

Dependiendo del impacto económico y en horas de trabajo dedicadas a mitigar las consecuencias, un riesgo puede clasificarse en las siguientes cuatro categorías de mayor a menor severidad:

1. **Critico.** Riesgo critico es aquel que supone incremento de al menos un tercio del total del tiempo y/o presupuesto asignados a la tarea a la que afecte.
2. **Importante.** Riesgo critico es aquel que supone incremento de entre un 20% y un tercio del total del tiempo y/o presupuesto asignados a la tarea a la que afecte.
3. **Asumible.** Riesgo critico es aquel que supone incremento de entre un 10% y un 20% del total del tiempo y/o presupuesto asignados a la tarea a la que afecte.
4. **Irrelevante.** Riesgo critico es aquel que supone incremento de menos de un 10% del total del tiempo y/o presupuesto asignados a la tarea a la que afecte.

Respecto a la probabilidad de ocurrencia de un riesgo concreto a lo largo del proyecto, la clasificación de los riesgos, de mayor a menor, es la siguiente:

1. **Muy Alta.** Más del 75% de probabilidades de que ocurra.
2. **Alta.** Entre un 50% y un 75% de probabilidades de ocurrencia.
3. **Media.** Entre un 20% y un 50% de probabilidades de ocurrencia.
4. **Baja.** Menos del 20% de probabilidades de que ocurra.

Por último, todos los riesgos, indistintamente de su severidad, son clasificables en una de las siguientes categorías:

- **Riesgos de Proyecto.** Todos aquellos riesgos relativos a los recursos disponibles, la distribución de carga y las restricciones relativas a los mismos.

- **Riesgos de Proceso.** Aquellos relativos a la mala praxis en lo referente a documentación, proceso de desarrollo de software y planificación.
- **Riesgos de Producto.** Relacionados con la falta de experiencia en el dominio del problema del proyecto.

4.2.1 Listado de riesgos encontrados

A continuación, detallamos los riesgos que hemos encontrado relevantes tras realizar el proceso de análisis. Se incluyen planes de contingencia y mitigación además de recomendaciones estratégicas para evitar su aparición.

RISK-001	Problemas en la ejecución de Unreal Engine 4
Categoría	Producto
Probabilidad	Media
Impacto	Crítico
Descripción	Debido a la nula experiencia inicial con el motor de videojuegos Unreal Engine 4 podría no lograrse el objetivo establecido de replicar una experiencia realista de juego relativa al juego de la colmena.
Fases afectadas	➢ Construcción
Gestión del riesgo	
Estrategia	Investigar el riesgo
Plan de mitigación	Realización de pruebas de concepto previas que nos ayuden a calcular la viabilidad del proyecto
Plan de contingencia	Replantear el alcance del proyecto de forma que nos planteemos un objetivo más asumible de acuerdo a la información recabada.

Tabla 2. RISK-001. Problemas en la ejecución de Unreal Engine 4.

RISK-002	Problemas con el PC de Desarrollo
Categoría	Proyecto
Probabilidad	Baja
Impacto	Asumible
Descripción	Un fallo en el equipo con el que se está desarrollando el proyecto puede redundar en pérdidas de información, errores y retrasos.
Fases afectadas	➢ Inicio. ➢ Elaboración. ➢ Construcción. ➢ Transición.
Gestión del riesgo	
Estrategia	Evitar el riesgo
Plan de mitigación	Evitar realizar actualizaciones del sistema, instalar software nuevo y tratar adecuadamente el hardware en lo que dure el proceso de desarrollo. Realizar copias versionadas, tanto de la documentación como del proyecto, en distintos soportes
Plan de contingencia	Recuperar información desde la última versión estable y trabajar en otro equipo disponible.

Tabla 3. RISK-002. Problemas con el PC de Desarrollo.

RISK-003	Cambios en los requisitos del sistema
Categoría	Proceso
Probabilidad	Media
Impacto	Importante
Descripción	Cuanto más tarde se desee modificar uno de los requisitos más coste tendrá acomodar el desarrollo a la nueva versión. Un cambio en una de las fases finales puede tener consecuencias graves.
Fases afectadas	➢ Elaboración. ➢ Construcción. ➢ Transición.
Gestión del riesgo	
Estrategia	Protegerse del riesgo
Plan de mitigación	Fijar especial atención en las fases inicial y de elaboración de forma que la mayoría de requisitos importantes queden registrados durante dichas fases.
Plan de contingencia	Realizar las modificaciones pertinentes para que el nuevo planteamiento tenga sentido y reorganizar la planificación teniendo en cuenta el retraso generado.

Tabla 4. RISK-003. Cambios en los requisitos del sistema.

RISK-004	Aparición de errores de diseño
Categoría	Proceso
Probabilidad	Baja
Impacto	Importante
Descripción	Un diseño incompleto, erróneo o demasiado abstracto puede redundar en problemas a la hora de implementar la arquitectura final.
Fases afectadas	➢ Elaboración. ➢ Construcción.
Gestión del riesgo	
Estrategia	Protegerse del riesgo
Plan de mitigación	Realizar especial hincapié en la fase de elaboración realizando una segunda iteración en el caso de considerarlo necesario para finalizar con un diseño pulcro y lógico.
Plan de contingencia	En caso de que sea necesario, volver a la fase de elaboración para realizar una nueva iteración que corrija los errores encontrados.

Tabla 5. RISK-004. Aparición de errores de diseño.

RISK-005	Inconveniencias en el proceso de desarrollo del software
Categoría	Proceso
Probabilidad	Media
Impacto	Importante
Descripción	La aplicación ineficiente o errónea de algoritmos, la falta de conocimientos para la correcta aplicación de la lógica del juego o el fallo al comprender el paradigma subyacente al motor de desarrollo son problemas que pueden retrasar la finalización de la fase de construcción.

Fases afectadas	➢ Construcción.
Gestión del riesgo	
Estrategia	Investigar el riesgo
Plan de mitigación	Intentar aplicar algoritmos y soluciones simples, generalizadas y fáciles de extender y comprender. Evitar las filigranas y detalles que redunden en un exceso de tiempo de construcción.
Plan de contingencia	Consultar la documentación disponible y a la comunidad de desarrolladores en caso de cualquier conflicto.

Tabla 6. RISK-005. Inconveniencias en el proceso de desarrollo del software.

RISK-006	**Retraso respecto a la planificación inicial**
Categoría	Proceso
Probabilidad	Alta
Impacto	Importante
Descripción	Debido a una mala planificación, situaciones fortuitas y responsabilidades personales de los miembros del equipo se pueden producir retrasos y no finalizar el proyecto en las fechas previstas.
Fases afectadas	➢ Inicio. ➢ Elaboración. ➢ Construcción. ➢ Transición.
Gestión del riesgo	
Estrategia	Retener el riesgo
Plan de mitigación	Planificar tareas ajustando los objetivos y dejando holgura a las metas de forma que haya espacio para adaptarse a imprevistos.
Plan de contingencia	Incremento de velocidad en el desarrollo y realización de horas extra en el caso de que se crea necesario.

Tabla 7. RISK-006. Retraso respecto a la planificación inicial.

4.3 Roles y funciones de los recursos disponibles

A continuación, detallaremos los roles que desempeña cada uno de los participantes y las responsabilidades ligadas a cada uno de esos roles.

Participante: Margarita Gonzalo Tasis	
Rol	**Responsabilidades**
Cliente	El cliente es el responsable de fijar los objetivos y requisitos del sistema, además de orientar en la toma de decisiones al equipo de desarrollo de forma que la solución realizada satisfaga los objetivos de la mejor manera posible.
Informador	Se encarga de informar, apoyar, orientar y dirigir el proceso de desarrollo de forma que oriente al equipo en el mejor camino para la consecución de los objetivos, basándose en su experiencia y conocimientos.

Tabla 8. Roles y responsabilidades de Margarita Gonzalo Tasis.

Participante: Rubén Moya Vázquez	
Rol	**Responsabilidades**
Jefe de proyecto	La función del jefe de proyecto es planificar, coordinar y gestionar los recursos de forma que el proyecto se realice dentro de plazo y con un coste adecuado.
Analista	Su función consiste en recabar información relativa al proyecto a desarrollar. De esta forma, será el responsable de realizar la recolección de requisitos y casos de uso y un modelo de alto nivel de la solución que servirá como apoyo al trabajo del diseñador.
Diseñador Software	Realiza el diseño de la arquitectura y los modelos de comunicación entre los distintos actores del sistema de forma que consiga una solución fiable, eficiente y adecuada a los objetivos.
Programador	Programa la lógica interna del juego y la interfaz de usuario.
Diseñador Grafico	Diseña e implementa todos los aspectos gráficos relacionados con el HUD, la navegación y el apartado visual del juego.
Verificador	Es el responsable de realizar el proceso de pruebas, validar el software y detectar e informar de los errores que encuentre de forma que puedan ser solventados rápidamente.
Traductor	Responsable de hacer la localización del videojuego a los idiomas seleccionados.

Tabla 9. Roles y responsabilidades de Rubén Moya Vázquez.

4.4 Planificación del Proyecto

A continuación, describiremos mediante diagramas de Gantt creados utilizando *GanttProject* [78], las distintas fases planificadas inicialmente para el desarrollo del proyecto y adjuntaremos tablas de costes de los recursos utilizados durante el desarrollo del mismo, con el objetivo de obtener el coste final estimado del proyecto.

Teniendo en cuenta la escasa disponibilidad de los recursos humanos involucrados en el proyecto, el plan de fases desarrollado es el siguiente:

- Fase 1. **Fase de Inicio.** En esta fase desarrollaremos el GDD y los capítulos iniciales de la documentación. Duración estimada: 2 semanas.
- Fase 2. **Primera Fase de Elaboración.** Realización de la fase de análisis y la documentación pertinente. Duración estimada: 2 semanas.
- Fase 3. **Segunda Fase de Elaboración.** Realización de la fase de diseño y la documentación pertinente. Duración estimada: 2 semanas.
- Fase 4. **Primera Fase de Construcción.** En esta primera fase desarrollaremos los componentes gráficos necesarios para la construcción del juego (botones, piezas, materiales, etc.). Duración estimada: 2 semanas.
- Fase 5. **Segunda Fase de Construcción.** Construcción del nivel de juego, el HUD y el menú de navegación. Duración estimada: 3 semanas.
- Fase 6. **Tercera Fase de Construcción.** Programación de la lógica interna del juego. Duración estimada: 3 semanas.
- Fase 7. **Fase de Transición.** Realizaremos los manuales, las pruebas y las tareas relativas al cierre de proyecto y finalización de la documentación. Duración estimada: 1 semana y media (10 días).

Como resultado, el proyecto tendrá una duración total estimada ligeramente superior a 15 semanas, partiendo de la fecha de inicio 10/03/2016 y contando con la fecha 26/06/2016 como fecha final considerada.

4.4.1 Diagrama de Gantt

El siguiente diagrama de Gantt representa la planificación de las fases acordadas con anterioridad. Dicha planificación inicial no contempla las situaciones descritas en la sección de Seguimiento del Proyecto, ya que fue realizada antes de que aconteciesen. Así mismo, la asignación temporal está distribuida en semanas en lugar de en jornadas, tal y como se ha explicado en el apartado anterior, debido a la disponibilidad variable de los recursos.

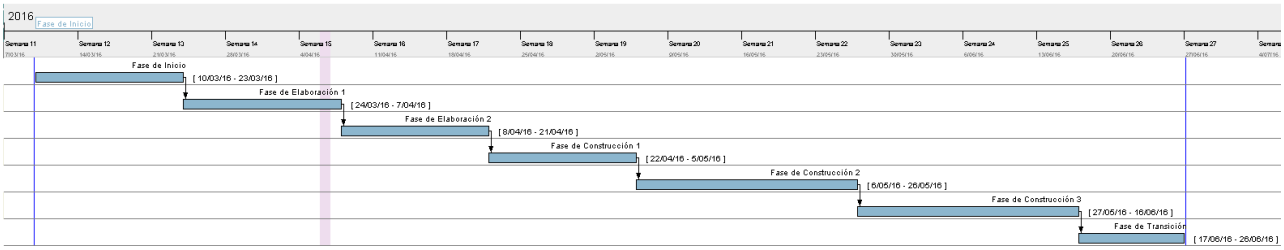


Figura 44. Diagrama de Gantt de la Planificación Inicial.

4.4.2 Costes de Producción

Para finalizar con la planificación del proyecto, desglosaremos brevemente los costes de producción estimados conforme a los recursos necesarios, tanto humanos como materiales, aportando una cifra final aproximada de lo que podría costar el total del proyecto.

El baremo de precios relativos a los roles de los diversos recursos humanos se ha estimado a partir del precio medio obtenido de distintas fuentes y ofertas, por tanto, podrían no ser exactos.

Rol	Precio/hora	Horas estimadas	Coste total
Jefe de Proyecto	60€	30	**1800€**

Diseñador Software	50€	50	**2500€**
Analista	40€	50	**2000€**
Programador	40€	85	**3400€**
Diseñador Grafico	20€	55	**1100€**
Verificador	25€	22	**550€**
Traductor	39€	8	**312€**

Tabla 10. Costes de los Recursos Humanos.

Concepto	Precio mensual	Coste total
Amortización del Hardware	44€	176€
Conexión ADSL	50€	200€
Suministro Eléctrico	70€	280€
Alquiler de Local	*Gratis*	*Gratis*
Licencia de Unreal Engine 4	*Gratis*	*Gratis*
Licencia de Autodesk Maya	*Gratis*	*Gratis*
Licencia de Word 2016	*Gratis*	*Gratis*
Licencia de Astah Professional	*Gratis*	*Gratis*
Banda Sonora del Juego	*Gratis*	*Gratis*
Componentes digitales de terceros	*Gratis*	*Gratis*
Material de Oficina	5€	20€

Tabla 11. Costes de los Recursos Materiales y de carácter tecnológico.

Una vez desglosados todos los costes relativos al desarrollo del proyecto, el montante total asciende a **12.338€** de los cuales **11.662€** son de costes humanos y **676€** de costes materiales.

4.5 Seguimiento del Proyecto

A continuación, desglosaremos los costes finales reales en tiempo de las distintas fases. Además, aportaremos justificación a las variaciones respecto a la planificación inicial, así como a los retrasos y pausas han ocurrido durante el desarrollo del proyecto. De esta forma, confrontaremos la planificación inicial con la planificación que realmente se ha seguido, con el objetivo de aprender de los errores cometidos a la hora de estimar el consumo de recursos y de esta forma evitar cometer nuevos errores en futuros proyectos.

En primer lugar, mostraremos el diagrama de Gantt en el que se contempla el tiempo real que ha llevado cada tarea, así como los periodos de inactividad entre tareas. Una vez visto, pasaremos a detallar las situaciones encontradas en cada fase.

Figura 45. Diagrama de Gantt de la Planificación Real.

Como se puede apreciar a simple vista en el diagrama anterior, ha habido grandes variaciones respecto a la planificación inicial. En concreto, dos de impacto relativamente leve en las fases de Inicio y Elaboración y dos de una severidad critica, antes de la Fase de Inicio y entre la Fase de Elaboración 1 y la Fase de Elaboración 2.

El retraso relativo a la fecha de comienzo, prevista inicialmente para el 10 de marzo y que, en realidad, tuvo lugar un mes después; se debió a la falta de disponibilidad de recursos de tipo hardware. Un problema en el equipo de desarrollo ocasionó que fuese irrecuperable para el proyecto y por consiguiente, fue necesario adquirir uno nuevo, manteniéndose el equipo de trabajo inactivo en cuanto al desarrollo del proyecto.

Una vez empezado el desarrollo, la Fase de Inicio se prolongó una semana más de la cuenta debido a la actividad laboral paralela de los recursos implicados.

El retraso en la planificación de la fase anterior, redundo no solo en que la primera fase de elaboración comenzase más tarde, sino que se prolongó durante la misma, lo que supuso que ésta, a su vez, se prolongase acumulando otra semana de retraso.

El gran periodo de inactividad enmarcado entre el final de la primera fase de elaboración y la segunda, se debió a la aparición de inconveniencias de carácter personal, que unidas a la actividad laboral paralela de los miembros del equipo de trabajo redundaron en un conflicto de intereses que obligó al jefe de proyecto a paralizar el mismo mientras durasen dichas inconveniencias.

Con la intención de no desaprovechar por completo el periodo de inactividad entre fases. Se tomó la decisión de, en la medida de lo posible, utilizar el tiempo para el proceso de formación. Es en ese momento cuando se leyeron las referencias del 1 al 16 del índice bibliográfico.

Finalmente, se logró retomar el proyecto con efectividad la tercera semana de abril. A partir de ese momento, habiendo aprovechado en cierto grado el periodo de inactividad para la formación del equipo, pudo seguirse la planificación con la duración de las fases estimada inicialmente.

Por tanto, si se omiten los periodos de inactividad por causa mayor, el proyecto siguió con la planificación establecida, asumiendo un sobrecoste en tiempo de aproximadamente 2 semanas.

5 Análisis

El objetivo que perseguimos en este capítulo es definir las principales características y especificaciones de nuestro sistema de forma que luego sean de utilidad en las posteriores fases de diseño e implementación. Para organizar correctamente este capítulo, lo hemos subdividido en cinco secciones que recogen los principales documentos entregables en la fase de análisis de un proyecto de software.

La primera sección recoge el propósito general de nuestro proyecto, definiendo los objetivos del sistema y el correcto funcionamiento del mimo.

La segunda sección define los límites de nuestro sistema, concretando las situaciones y comportamientos que pertenecen al mismo y aquellos que escapan a su alcance.

La tercera sección sirve para enmarcar el documento de análisis de requisitos, tanto funcionales como no funcionales, de nuestro sistema. Para el desarrollo de dicho documento tendremos en cuenta toda la información aportada anteriormente, tanto en este capítulo como en el GDD.

La cuarta sección engloba la especificación y realización de los casos de uso de análisis [17], donde se plantearán los principales escenarios y situaciones alternativas de la funcionalidad de nuestro sistema.

Por último, la quinta sección enmarca el modelo de dominio del sistema, en el que se definirá el organigrama de clases de análisis de nuestro sistema y se aporta una justificación al mismo.

5.1 Propósito General del Sistema

El sistema debe permitir al usuario disfrutar de una partida del juego *Hive: La Colmena* contra otro jugador en el mismo dispositivo. Para ello, utilizaremos el método de juego multijugador conocido como *hotseat*, que permite a los jugadores jugar por turnos en el mismo dispositivo.

El sistema deberá permitir al usuario configurar la partida a su gusto con diferentes opciones. Todo aspecto personalizable de la partida deberá estar a disposición del usuario para ser configurado a su gusto.

El sistema deberá permitir al usuario configurar las opciones de personalización relativas al propio sistema. Cuestiones tales como el idioma, la configuración de efectos de sonido y música y aspectos gráficos, deberán estar a disposición del usuario.

5.2 Alcance del Sistema

El sistema se apoyará en el motor gráfico y contenidos básicos de *Unreal Engine 4* para la realización de entornos y componentes del juego.

El sistema se apoyará en el motor de videojuegos de *Unreal Engine 4* para la realización de la navegación y lógica interna del videojuego.

El sistema no asumirá, en su primer lanzamiento, la realización de un modo multijugador online ni el desarrollo de una Inteligencia Artificial para jugar en solitario.

5.3 Análisis de Requisitos del Sistema

Para poder considerar que el sistema cumple con los mínimos de calidad planteados como objetivos, se deberán satisfacer los requisitos planteados a continuación. Dichos requisitos se han obtenido y catalogado siguiendo el modelo *FURPS+*.

5.3.1 Modelo FURPS+

El modelo *FURPS+* [70] es un sistema de análisis y catalogación de requisitos creado por *Robert Grady* para la empresa *Hewlett-Packard*. Las siglas *FURPS* provienen de las siguientes características observables a la hora de catalogar un requisito:

- Funcionalidad (***F**unctionality*).
- Usabilidad (***U**sability*).
- Fiabilidad (***R**eliability*).
- Rendimiento (***P**erformance*).
- Sostenibilidad (***S**upportability*).

Además, el '+' que acompaña a dicho acrónimo sirve como recordatorio para otras cuestiones que debemos tener en cuenta. Dichos requisitos adyacentes son:

- Requisitos de Diseño.

- Requisitos de Implementación.
- Requisitos de Interfaz.
- Requisitos Físicos.

5.3.2 Requisitos Funcionales

Los requisitos aquí presentados muestran el comportamiento deseado para nuestro sistema, su navegación y la respuesta esperada a las acciones a realizar por parte del usuario final. Deben cubrir todos los aspectos de la lógica interna de nuestro sistema y serán tenidos en cuenta a la hora de implementar todos los aspectos relativos a la funcionalidad del mismo.

RF-001	**Jugar Partida**
Descripción	El sistema debe permitir al usuario jugar una partida de *Hive: La Colmena* contra otro usuario usando el mismo dispositivo.

Tabla 12. RF-001. Jugar Partida.

RF-002	**Seleccionar Piezas**
Descripción	El sistema debe permitir al usuario elegir si desea jugar con las expansiones y con cuales de ellas desea jugar.

Tabla 13. RF-002. Seleccionar Piezas.

RF-003	**Seleccionar Jugador Inicial**
Descripción	El sistema debe permitir al usuario elegir que jugador comenzará la partida.

Tabla 14. RF-003. Seleccionar Jugador Inicial.

RF-004	**Seleccionar Color**
Descripción	El sistema debe permitir al usuario elegir el color de las fichas con las que va a jugar.

Tabla 15. RF-004. Seleccionar Color.

RF-005	**Aplicar Reglas de Torneo**
Descripción	El sistema deberá permitir al usuario decidir si quiere jugar con reglas de torneo.

Tabla 16. RF-005. Aplicar Reglas de Torneo.

RF-006	**Acceder al Manual**
Descripción	El sistema deberá tener información de las reglas de juego accesible al usuario desde cualquier punto de la aplicación.

Tabla 17. RF-006. Acceder al Manual.

RF-007	**Pausar Partida**
Descripción	El sistema deberá permitir al usuario pausar la partida.

Tabla 18. RF-007. Pausar Partida.

RF-008	**Abandonar Partida**
Descripción	El sistema deberá permitir al usuario salir de la partida en cualquier momento.

Tabla 19. RF-008. Abandonar Partida.

RF-009	**Configurar Audio**
Descripción	El sistema deberá permitir al usuario configurar el volumen de la música y los efectos del juego.

Tabla 20. RF-009. Configurar Audio.

RF-010	**Configurar Video**
Descripción	El sistema deberá permitir al usuario modificar las opciones graficas a su gusto.

Tabla 21. RF-010. Configurar Video.

RF-011	**Cambiar Idioma**
Descripción	El sistema deberá permitir al usuario cambiar el idioma del mismo.

Tabla 22. RF-011. Cambiar Idioma.

RF-012	Cambiar Estilo de las Piezas
Descripción	El sistema deberá permitir al usuario cambiar el estilo visual de las piezas entre los dos con los que cuenta el juego de mesa original.

Tabla 23. RF-012. Cambiar Estilo de las Piezas.

RF-013	Ver Créditos
Descripción	El sistema deberá permitir al usuario ver los créditos y agradecimientos a las personas y medios usados en el desarrollo del proyecto.

Tabla 24. RF-013. Ver Créditos.

RF-014	Colocar Pieza
Descripción	El sistema deberá asistir al jugador a la hora de colocar una pieza, resaltando aquellas casillas en las que pueda ser colocada.

Tabla 25. RF-014. Colocar Pieza

.

RF-015	Mover Pieza
Descripción	El sistema deberá asistir al jugador a la hora de mover una pieza, resaltando aquellas casillas a las que se pueda mover la pieza de acuerdo a las reglas de movimiento.

Tabla 26. RF-015. Mover Pieza.

RF-016	Usar Habilidad de Pieza
Descripción	El sistema deberá asistir al jugador a la hora de usar la habilidad de la pieza “Cochinilla”. Para ello, resaltará aquellas piezas sobre las que se pueda utilizar su habilidad según las reglas.

Tabla 27. RF-016. Usar Habilidad de Pieza

RF-017	Ver Configuración Actual
Descripción	El sistema deberá permitir al usuario consultar la configuración actual del sistema.

Tabla 28. RF-017. Ver Configuración Actual.

5.3.3 Requisitos No Funcionales

Los requisitos aquí presentados muestran el grado de cumplimiento de los estándares de calidad por parte de la operación del sistema a realizar. Sirven para medir atributos de idoneidad, no relativos a la funcionalidad, dentro del modelo FURPS+.

RNF-001	**Motor de Desarrollo**
Descripción	El sistema deberá implementarse usando *Unreal Engine 4* como motor base del juego.

Tabla 29. RNF-001. Motor de Desarrollo.

RNF-002	**Lenguaje de Programación**
Descripción	El trabajo de programación del sistema deberá realizarse en *C++*.

Tabla 30. RNF-002. Lenguaje de Programación.

RNF-003	**Aplicación de Diseño 3D**
Descripción	Los modelos específicos del juego se realizarán utilizando *Autodesk Maya*.

Tabla 31. RNF-003. Aplicación de Diseño 3D.

RNF-004	**Plataforma Objetivo**
Descripción	El sistema se desarrollará para funcionar en PC.

Tabla 32. RNF-004. Plataforma Objetivo.

RNF-005	**Apoyo Audiovisual**
Descripción	El sistema contará con apoyo audiovisual que permitirá al usuario darse cuenta de los cambios de estado de la partida.

Tabla 33. RNF-005. Apoyo Audiovisual.

RNF-006	**Localización**
Descripción	El sistema estará completamente traducido al inglés y al castellano.

Tabla 34. RNF-006. Localización.

RNF-007	**Contenidos Gratuitos**
Descripción	El sistema estará formado por contenidos propios y contenidos de carácter gratuito del bazar de *Unreal Engine 4*.

Tabla 35. RNF-007. Contenidos Gratuitos.

RNF-008	**Transiciones y Animaciones**
Descripción	El sistema deberá transmitir fluidez y dinamismo. Para ello las transiciones y animaciones no duraran en ningún caso más de 5 segundos.

Tabla 36. RNF-008. Transiciones y Animaciones.

RNF-009	**Información Accesible**
Descripción	El sistema deberá garantizar al usuario poder consultar las reglas de juego en cualquier momento. Para ello el manual no se encontrará nunca a más de dos click de distancia respecto a la vista actual.

Tabla 37. Información Accesible.

RNF-010	**Usabilidad**
Descripción	El sistema deberá cumplir los requisitos mínimos de usabilidad relativos al tipo de aplicación al que pertenece.

Tabla 38. RNF-010. Usabilidad.

RNF-011	**Escalabilidad**
Descripción	El sistema será escalable, se deberá poder agregar funcionalidad al mismo sin afectar al funcionamiento del resto de características.

Tabla 39. RNF-011. Escalabilidad.

RNF-012	**Accesibilidad Auditiva**
Descripción	El sistema será accesible para personas con diversidad auditiva. Para ello permitirá configurar el audio y toda señal auditiva tendrá apoyo visual.

Tabla 40. RNF-012. Accesibilidad Auditiva.

RNF-013	**Accesibilidad Visual**

Descripción	El sistema será accesible para persona con cierto grado de daltonismo. Para ello, las piezas dispondrán del estilo *Carbon* en blanco y negro.

Tabla 41. RNF-013. Accesibilidad Visual.

5.3.4 Modelo de Casos de Uso

Con el modelo de casos de uso [76] adjunto a continuación pretendemos dar una visión global de las funcionalidades concretas que se podrán realizar mediante nuestro sistema y como el usuario y ellas mismas se relacionan. Cada funcionalidad concreta y completa está representada por un caso de uso, el usuario por un actor y las relaciones entre ellos por líneas de uso o dependencia.

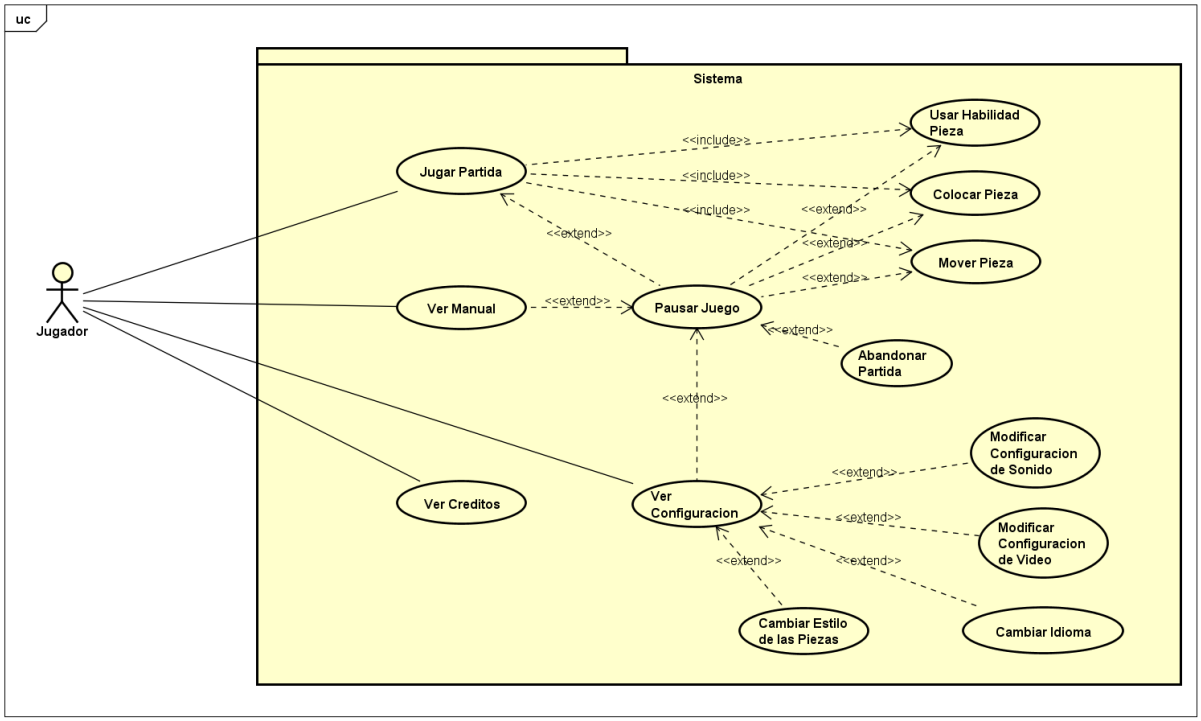


Figura 46. Diagrama de Casos de Uso del Sistema.

5.3.4.1 Actores de nuestro sistema

Un actor es cualquier entidad externa al sistema que se relaciona con el mismo a través de los medios de interacción disponibles. En el caso de nuestro videojuego, el único papel que puede desempeñar el usuario final es el de Jugador, tal y como se representa en el Modelo de Casos de Uso.

ACT-001	Jugador
Descripción	Este actor representa a cada usuario que interactúa con el sistema con el objetivo de jugar una partida a nuestro juego.

Tabla 42. ACT-001. Jugador.

5.3.5 Especificación de Casos de Uso y Diagramas de Secuencia

En este caso pretendemos analizar las acciones, situaciones, clases y el orden en el que se involucran estas en el desarrollo de cada funcionalidad de nuestro sistema. De esta forma podremos observar el desarrollo esperado de los acontecimientos que forman parte de una funcionalidad concreta, así como los flujos alternativos a seguir en caso de producirse una situación de excepción. Para ello cada caso de uso de los representados en el Modelo de Casos de Uso.

5.3.5.1 UC-001. Jugar Partida

Caso de Uso	UC-001. Jugar Partida.	
Fuentes	Rubén Moya Vázquez	
Actor	ACT-001. Jugador	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario seleccione la opción de juego "Jugador VS Jugador".	
Flujo principal	Paso	Acción
	1	El **actor** pulsa la opción del menú principal "Jugador VS Jugador".
	2	El **sistema** cambia a la vista de configuración de partida y muestra los pre-ajustes.
	3	El **actor** configura la partida a su gusto y pulsa el botón "Jugar".
	4	El **sistema** cambia a la vista de juego, inicia el proceso principal de juego con los parámetros configurados y cede el turno al jugador que debe comenzar. Comunica el inicio del turno del usuario y se mantiene a la espera.
	5	El **actor** realiza el caso de uso **UC-002. Colocar Pieza.**
	6	El **sistema** comprueba que no se ha llegado a ningún estado final, informa del cambio de turno. Comprueba que el usuario al que pertenece el nuevo turno ha colocado su Abeja Reina y habilita la opción de mover piezas.
	7	El **actor** realiza el caso de uso **UC-003. Mover Pieza.**
	8	El **sistema** comprueba que no se ha llegado a ningún estado final, informa del cambio de turno. Comprueba que el usuario al que pertenece el nuevo turno ha colocado su Cochinilla y habilita la acción de usar su habilidad.
	9	El **actor** realiza el caso de uso **UC-004. Usar Habilidad de Pieza.**
	10	El **sistema** comprueba que el juego se encuentra en una situación final, informa al usuario del resultado de la partida, finaliza el proceso de juego y plantea al usuario la opción de salir o jugar otra partida.
	11	El **actor** selecciona la opción "Salir".
	12	El **sistema** comprueba la opción seleccionada, cierra la vista de juego y vuelve a la vista del menú principal finalizando el caso de uso.
Flujos alternativos	Paso	Acción
	3	El **actor** pulsa en el botón "Volver" para salir de la configuración de partida.
	a	El **sistema** cierra la vista de configuración de partida y vuelve a la vista del menú principal finalizando el caso de uso.

	4	El **actor** ha dejado la opción del jugador que iniciará la partida en la opción "Aleatorio".	
		a	El **sistema** calcula de manera aleatoria el jugador que deberá empezar la partida, desembocando en el **paso 5** para el jugador seleccionado.
	5	El **actor** pulsa el icono de menú o la tecla "Esc".	
		a	El **sistema** realiza el caso de uso **UC-005. Pausar Juego.**
	6	El **sistema** comprueba que el usuario no ha colocado aun su Abeja Reina.	
		a	El **sistema** informa al usuario del cambio de turno y desemboca al **paso 5**.
		El **actor** pulsa el icono de menú o la tecla "Esc".	
		b	El **sistema** realiza el caso de uso **UC-005. Pausar Juego.**
	8	El **sistema** comprueba que el usuario no ha colocado aun su Cochinilla.	
		a	El **sistema** informa al usuario del cambio de turno y desemboca al **paso 7**.
		El **actor** pulsa el icono de menú o la tecla "Esc".	
		b	El **sistema** realiza el caso de uso **UC-005. Pausar Juego.**
	10	El **sistema** comprueba que no se ha llegado a una situación final.	
		a	El **sistema** informa al usuario del cambio de turno y desemboca al **paso 9**.
		El **actor** pulsa el icono de menú o la tecla "Esc".	
		b	El **sistema** realiza el caso de uso **UC-005. Pausar Juego.**
	11	El **actor** selecciona "Volver a Jugar" en lugar de "Salir".	
		a	El **sistema** comprueba la opción seleccionada, cierra la vista de juego y abre la vista de configuración de partida volviendo al **paso 2**.
Precondiciones	El **sistema** se encuentra en ejecución y el usuario está situado en el menú principal.		
Postcondiciones	Los **actores** han disfrutado de una partida completa mediante el método *hotseat* llegando a una de las situaciones finales posibles.		
Requisitos relacionados	**RF-001, RF-002, RF-003, RF-004, RF-005, RF-007, RF-014, RF-015** y **RF-016.**		
Puntos de extensión	**1**	Entre el paso 5 y el paso 8, todos son candidatos validos como puntos de extensión para el caso de uso **UC-005. Pausar Juego.**	
Notas	Este caso de uso representa la funcionalidad principal del juego, así como el bucle de acciones hasta llegar al estado final de la partida. Hemos simplificado los flujos alternativos con el objetivo de facilitar la lectura y comprensión.		

Tabla 43. UC-001. Jugar Partida.

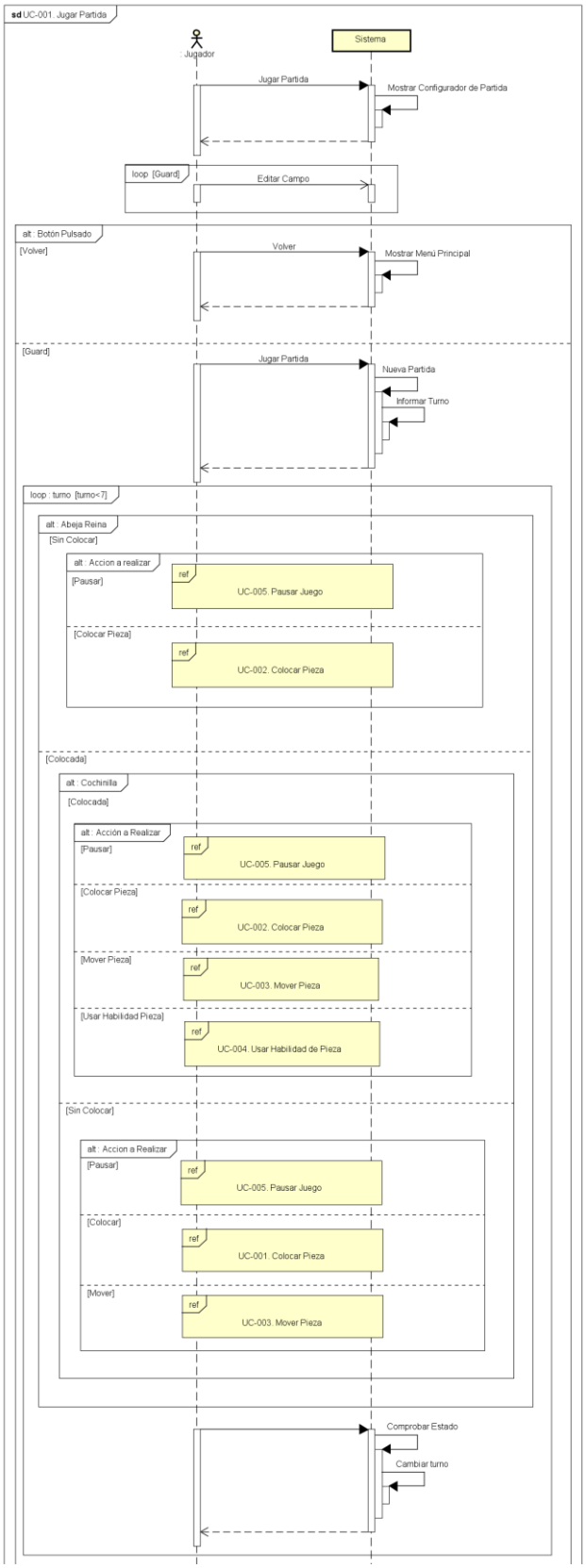


Figura 47. Diagrama de Secuencia de UC-001. Jugar Partida. Parte 1.

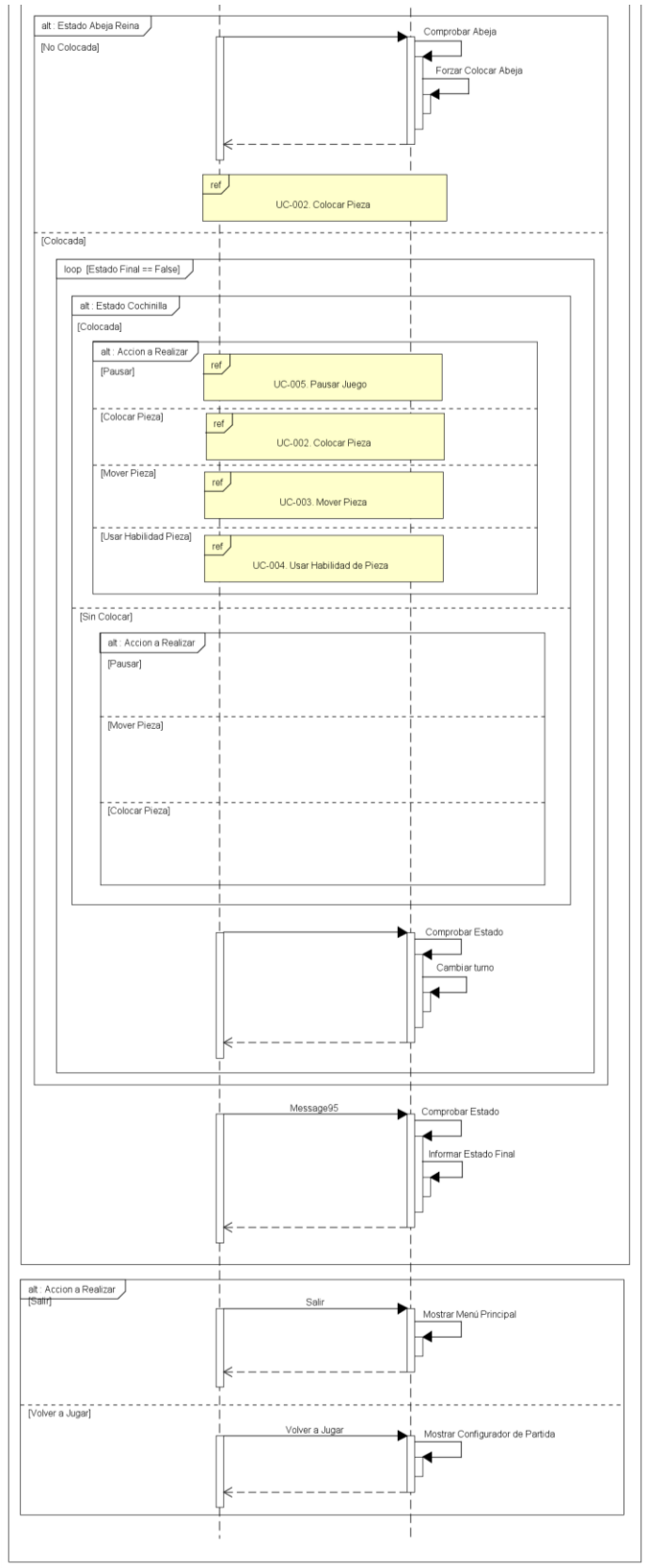


Figura 48. Diagrama de Secuencia de UC-001. Jugar Partida. Parte 2.

5.3.5.2 UC-002. Colocar Pieza

Caso de Uso	**UC-002. Colocar Pieza.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario pulsa el botón del HUD de la partida para colocar una pieza en juego durante su turno.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa el botón que representa a una pieza disponible para colocar.
	2	El **sistema** comprueba que puede colocar la pieza y resalta las posiciones disponibles para colocar la pieza.
	3	El **actor** pulsa sobre la posición en la que desea colocar la pieza.
	4	El **sistema** coloca la pieza en la posición seleccionada, disminuyendo el contador de piezas disponibles del tipo colocado, actualizando el estado de la casilla, pieza y piezas adyacentes y finaliza el turno del jugador.
Flujos alternativos	**Paso**	**Acción**
	1	El **actor** pulsa el icono de menú o la tecla “Esc”.
		a El **sistema** realiza el caso de uso **UC-005. Pausar Juego.**
	2	El **sistema** comprueba que la pieza seleccionada no puede ser colocada.
		a El **sistema** informa de que se está jugando con la opción de “Reglas de torneo” y la Abeja Reina no puede colocarse en el primer turno retornando al **paso 1**.
		b El **sistema** informa de que el jugador se encuentra en su cuarto turno y no ha colocado aun su Abeja Reina retornando al **paso 1**.
	3	El **actor** pulsa en cualquier sitio excepto una de las casillas válidas para la colocación de la ficha.
		a El **sistema** cancela la acción y el caso de uso queda sin efecto.
Precondiciones	La partida se encuentra en ejecución y el usuario se encuentra en su turno.	
Postcondiciones	El usuario ha colocado correctamente la ficha del tipo deseado en la casilla deseada.	
Requisitos relacionados	**RF-014.**	
Puntos de extensión	**1**	Cualquier paso entre los **pasos 1** y **3** es candidato valido como punto de extensión para el caso de uso **UC-004. Pausar Juego** tal y como se muestra en el flujo alternativo del **paso 1**.
Notas	Se han simplificado los flujos alternativos relativo a la extensión del caso de uso **UC-004. Pausar Juego** con el objetivo de facilitar la comprensión del flujo de acontecimientos.	

Tabla 44. UC-002. Colocar Pieza.

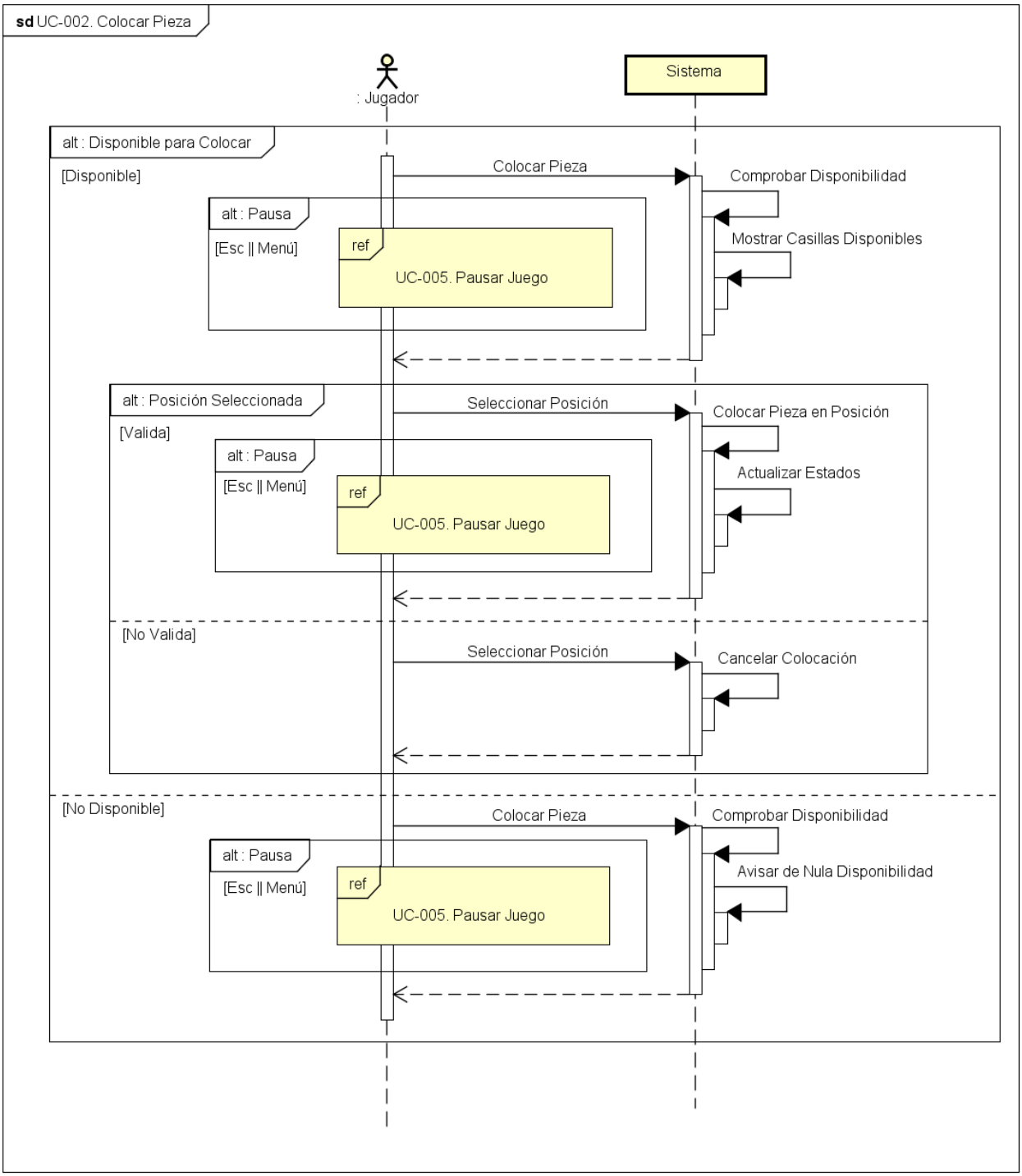


Figura 49. Diagrama de Secuencia UC-002. Colocar Pieza.

5.3.5.3 UC-003. Mover Pieza

Caso de Uso	**UC-003. Mover Pieza.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario haga click sobre una de sus piezas durante su turno.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa sobre la pieza que desea mover.
	2	El **sistema** comprueba que la pieza seleccionada puede moverse, busca las casillas disponibles para su movimiento y las resalta.
	3	El **actor** pulsa sobre una de las casillas resaltadas.
	4	El **sistema** realiza la animación de movimiento, actualiza las casillas de origen y destino, actualiza el estado de la pieza, de las antiguas piezas adyacentes y las nuevas y finaliza el turno del jugador.
Flujos alternativos	**Paso**	**Acción**
	2	El **sistema** comprueba que dicha pieza no puede moverse.
		a El sistema resalta la pieza en rojo durante dos segundos, emite una señal sonora y vuelve al **paso 1**.
	3	El **actor** pulsa cualquier sitio menos una casilla resaltada.
		a El **sistema** cancela la acción y el caso de uso queda sin efecto.
Precondiciones	La partida se encuentra en curso en el turno del actor y éste ha colocado su Abeja Reina en juego.	
Postcondiciones	El turno del actor ha finalizado con la realización exitosa del movimiento de pieza entre dos casillas y la actualización de todos los estados relacionados con dicho movimiento.	
Requisitos relacionados	**RF-015.**	
Puntos de extensión	**1**	Cualquier paso entre los **pasos 1** y **3** es candidato valido como punto de extensión para el caso de uso **UC-004. Pausar Juego** tal y como se muestra en el flujo alternativo del **paso 1**.
Notas	Durante la ejecución de este caso de uso no se contempla el estado en el que el usuario no ha colocado aun su Abeja Reina debido a que esta situación es una precondición para poder realizar este caso de uso.	

Tabla 45. UC-003. Mover Pieza.

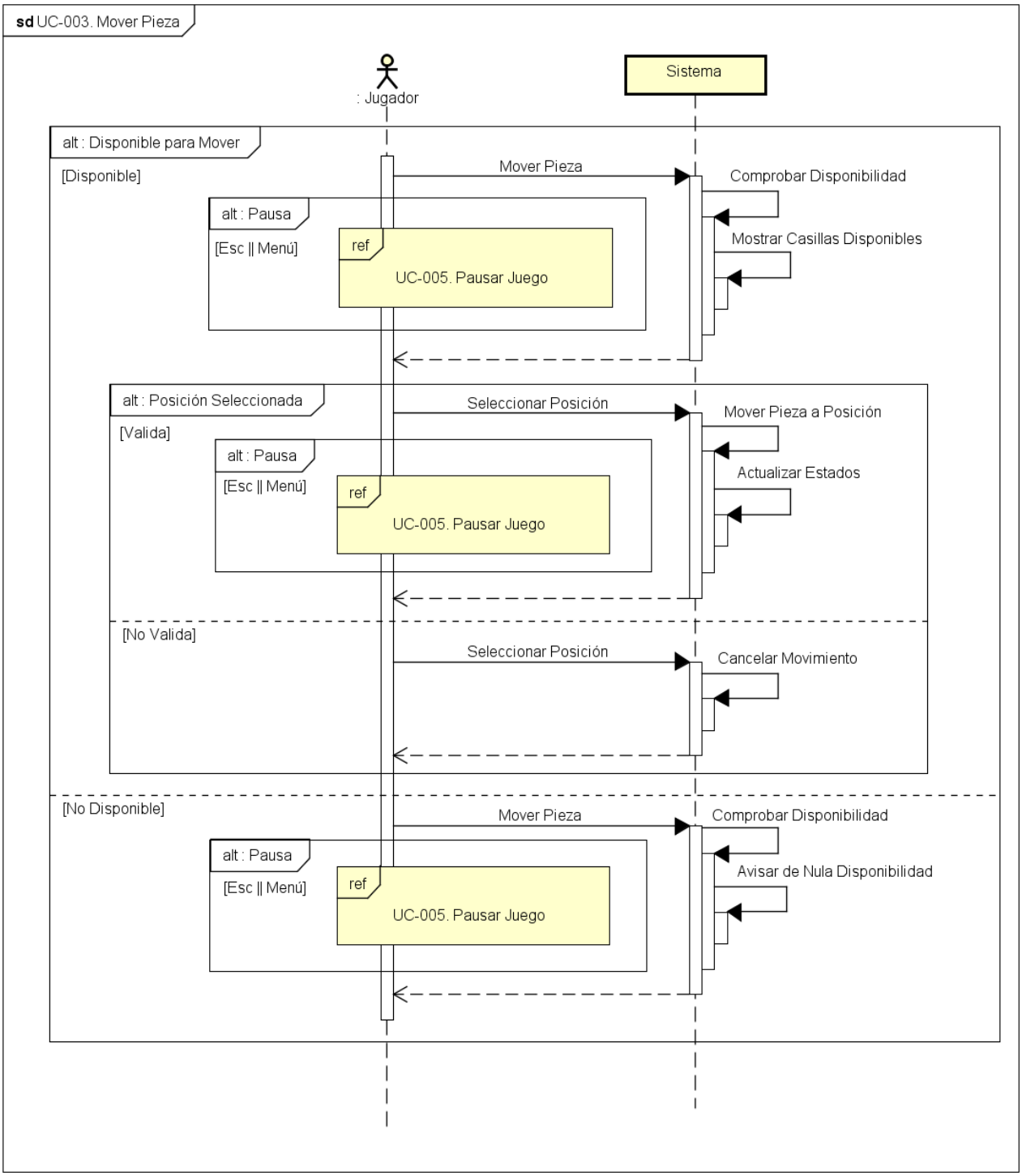


Figura 50. Diagrama de Secuencia de UC-003. Mover Pieza.

5.3.5.4 UC-004. Usar Habilidad de Pieza

Caso de Uso	**UC-004. Usar Habilidad de Pieza.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario haga click sobre una de las piezas adyacentes a su Cochinilla.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa sobre la pieza que desea mover.
	2	El **sistema** comprueba que la pieza seleccionada es vecina de la Cochinilla del jugador y no está bloqueada. Posteriormente muestra al jugador las casillas disponibles para el movimiento.
	3	El **actor** pulsa sobre una de las casillas resaltadas.
	4	El **sistema** realiza la animación de movimiento, actualiza las casillas de origen y destino, actualiza el estado de la pieza, de las antiguas piezas adyacentes y las nuevas y finaliza el turno del jugador.
Flujos alternativos	**Paso**	**Acción**
	2	El **sistema** comprueba que dicha pieza no puede moverse.
		a El sistema resalta la pieza en rojo durante dos segundos, emite una señal sonora y vuelve al **paso 1**.
	3	El **actor** pulsa cualquier sitio menos una casilla resaltada.
		a El **sistema** cancela la acción y el caso de uso queda sin efecto.
Precondiciones	La partida se encuentra en curso en el turno del actor, éste ha colocado su Cochinilla y su Abeja Reina en juego anteriormente y la pieza seleccionada es vecina de su Cochinilla.	
Postcondiciones	El turno del actor ha finalizado con la realización exitosa del movimiento de pieza entre dos casillas y la actualización de todos los estados relacionados con dicho movimiento.	
Requisitos relacionados	**RF-016.**	
Puntos de extensión	**1**	Cualquier paso entre los **pasos 1** y **3** es candidato valido como punto de extensión para el caso de uso **UC-004. Pausar Juego** tal y como se muestra en el flujo alternativo del **paso 1**.
Notas	Este caso de uso es bastante especial dentro de la lógica del juego ya que solo se aplica a una pieza en concreto de cada jugador.	

Tabla 46. UC-004. Usar Habilidad de Pieza.

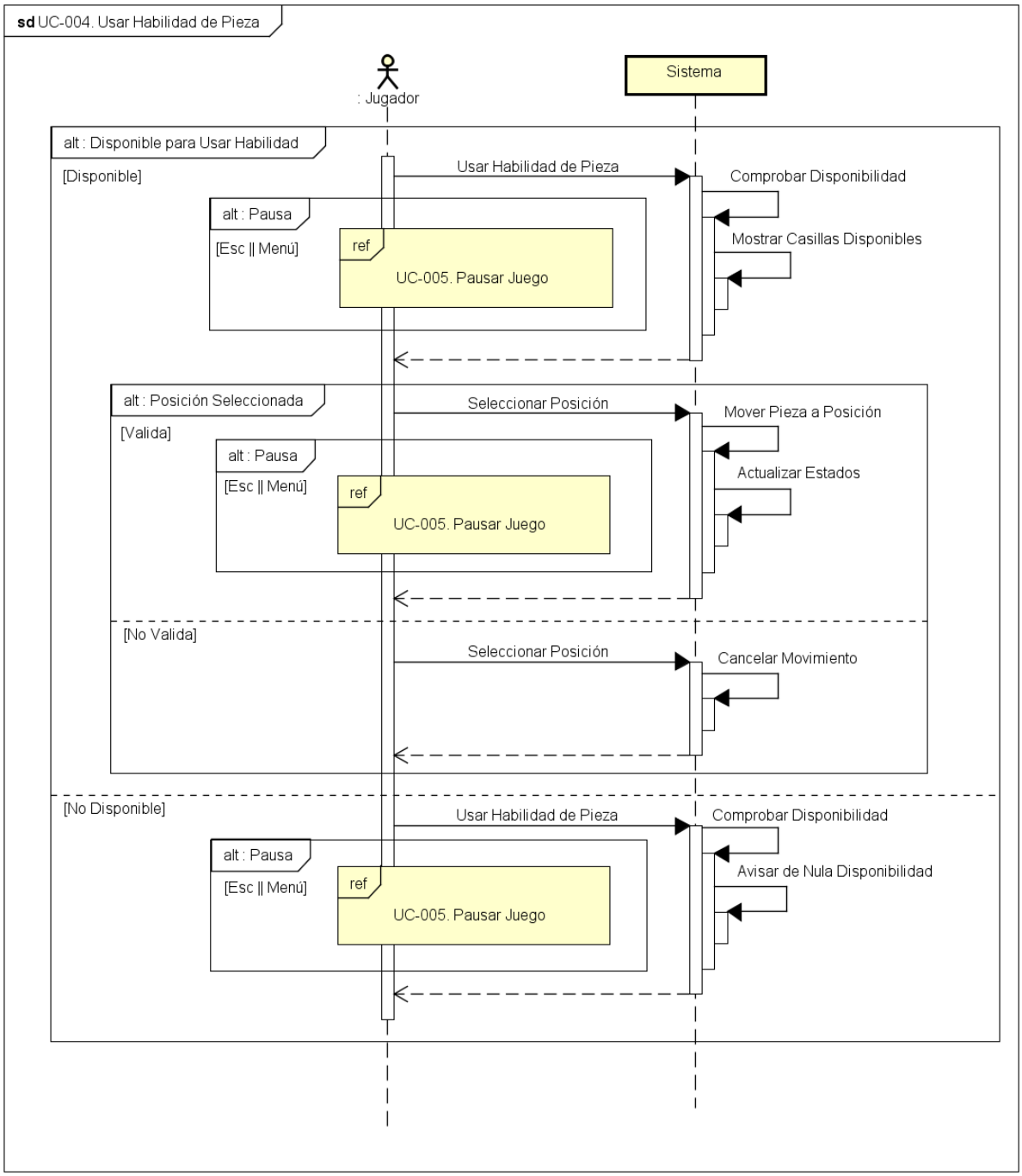


Figura 51. Diagrama de Secuencia de UC-004. Usar Habilidad de Pieza.

5.3.5.5 UC-005. Pausar Juego

Caso de Uso	**UC-005. Pausar Juego.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario pulse el botón de "Menú" del HUD o presione la tecla Esc.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa la opción del menú principal "Menú".
	2	El **sistema** cambia a la vista del menú de pausa y pausa el juego.
	3	El **actor** pulsa sobre el botón "Continuar Partida" para volver a la partida.
	4	El **sistema** cierra el menú de pausa y finaliza la pausa recuperando el juego en su estado previo y finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	1	El **actor** pulsa la tecla "Esc".
	a	El **sistema** actúa como si se hubiese pulsado el botón "Menú", desembocando en el **paso 2**.
	3	El **actor** pulsa la tecla "Esc".
	a	El **sistema** actúa como si se hubiese pulsado el botón "Volver al Juego", desembocando en el **paso 4**.
Precondiciones	El sistema se encuentra en ejecución y la partida se encuentra en curso.	
Postcondiciones	El usuario se encuentra de nuevo en la partida, en el mismo estado que antes de que la pausara.	
Requisitos relacionados	**RF-007.**	
Puntos de extensión	**1**	El paso 3 es un punto de extensión valido para el caso de uso **UC-006. Abandonar Partida.**
	2	El paso 3 es un punto de extensión valido para el caso de uso **UC-007. Ver Manual.**
	3	El paso 3 es un punto de extensión valido para el caso de uso **UC-009. Ver Configuración.**
Notas	Este caso de uso cumple dos funciones. En primer lugar, permite a los jugadores pausar la partida hasta que decidan retomarla. Por otro lado, permite al usuario consultar las reglas de juego, la configuración o salir de la partida si no desea terminarla.	

Tabla 47. UC-005. Pausar Juego.

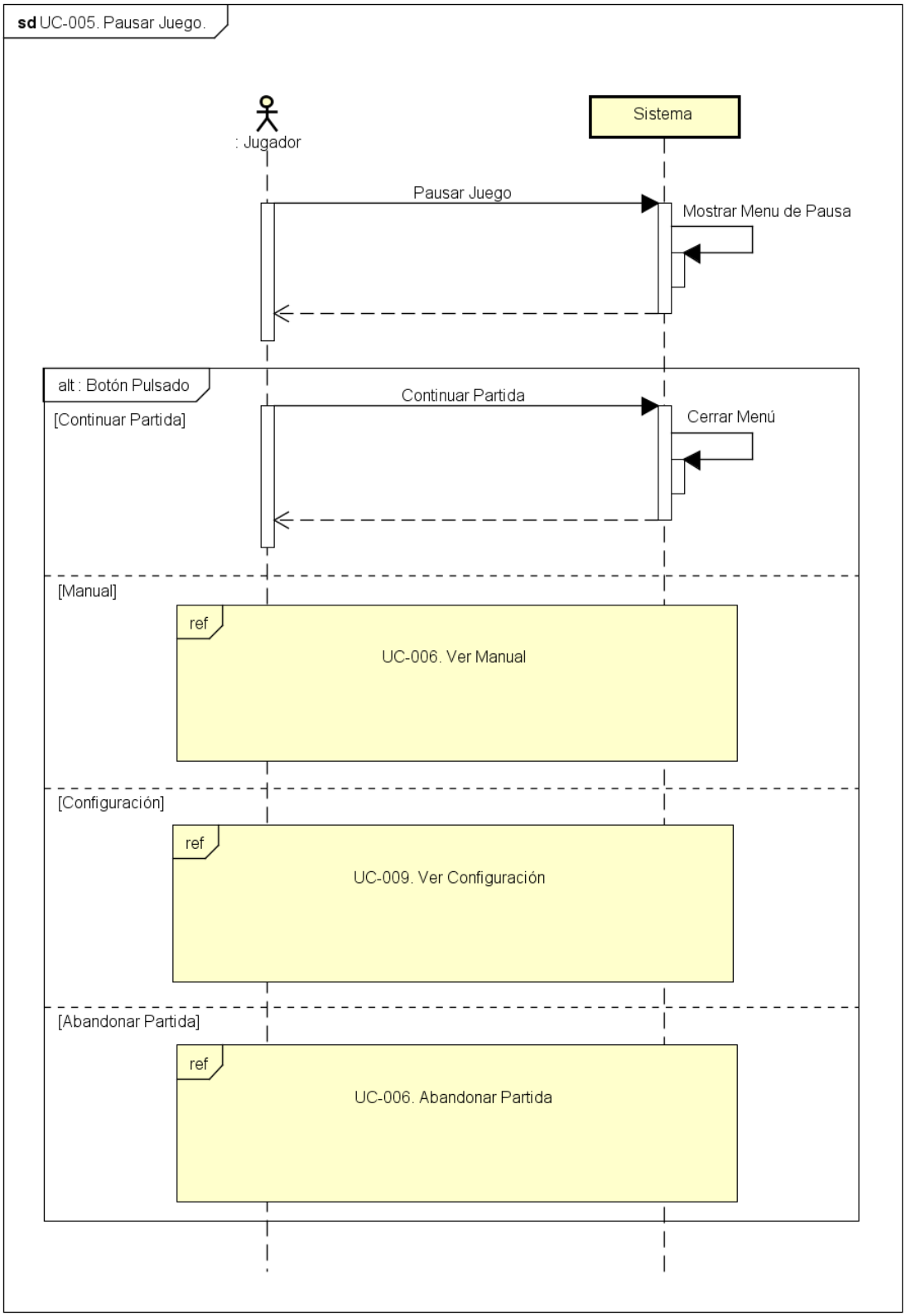


Figura 52. Diagrama de Secuencia de UC-005. Pausar Juego.

5.3.5.6 UC-006. Abandonar Partida

Caso de Uso	**UC-006. Abandonar Partida.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario pulse el botón "Abandonar Partida" del menú de pausa.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa la opción del menú pausa "Abandonar Partida".
	2	El **sistema** solicita la confirmación de acción por parte del usuario.
	3	El **actor** pulsa sobre el botón "Salir" para salir.
	4	El **sistema** cierra la vista de partida y vuelve a la vista del menú principal, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	3	El **actor** pulsa la tecla "Seguir Jugando".
	a	El **sistema** devuelve al usuario al menú de pausa quedando el caso de uso sin efecto.
Precondiciones	El sistema se encuentra en ejecución y el usuario está situado el menú de pausa.	
Postcondiciones	El usuario se encuentra de nuevo en el menú principal habiendo finalizado la partida.	
Requisitos relacionados	**RF-008.**	
Notas	Nos cercioramos de que el usuario está seguro de que desea salir antes de hacerlo siguiendo con las buenas prácticas en interacción persona-computadora.	

Tabla 48. UC-006. Abandonar Partida.

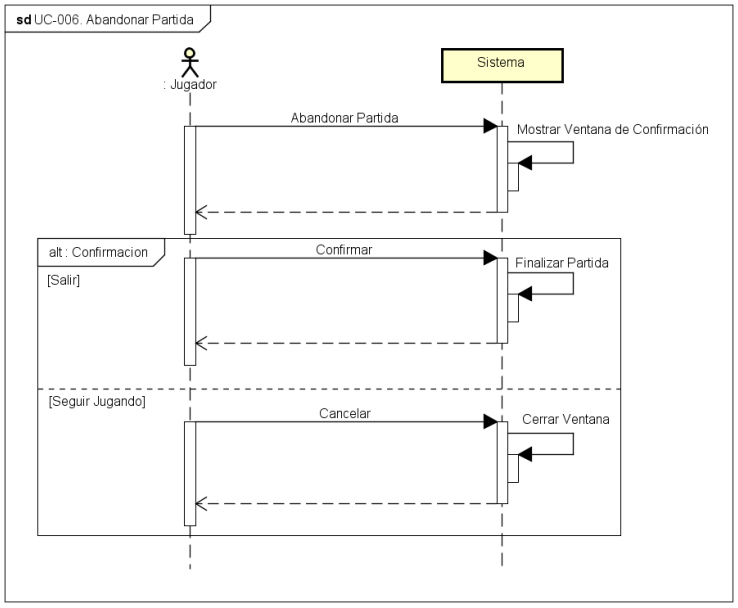


Figura 53. Diagrama de Secuencia de UC-006. Abandonar Partida.

5.3.5.7 UC-007. Ver Manual

Caso de Uso	**UC-007. Ver Manual.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario seleccione la opción del menú “Manual de Juego”.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa la opción del menú “Manual de Juego”.
	2	El **sistema** cambia a la vista del manual y muestra las reglas.
	3	El **actor** pulsa sobre el botón “Volver” para salir.
	4	El **sistema** cierra la vista del manual y vuelve a la vista del menú en el que se encontrase previamente el actor, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	3	El **actor** pulsa la tecla “Esc”.
	a	El **sistema** actúa como si se hubiese pulsado el botón “Volver”, desembocando en el **paso 4**.
Precondiciones	El sistema se encuentra en ejecución y el usuario está situado en el menú principal o en el menú de pausa.	
Postcondiciones	El usuario se encuentra de nuevo en el menú del que proviniese tras haber visto correctamente el manual de usuario.	
Requisitos relacionados	**RF-006.**	
Notas	El manual se muestra siguiendo la tendencia actual de mostrar la información en modo *scroll* en lugar de por páginas, esto se aprecia en el hecho de que no hay pasos en el flujo principal ni en flujos alternativos referidos a avanzar o retroceder página.	

Tabla 49. UC-007. Ver Manual.

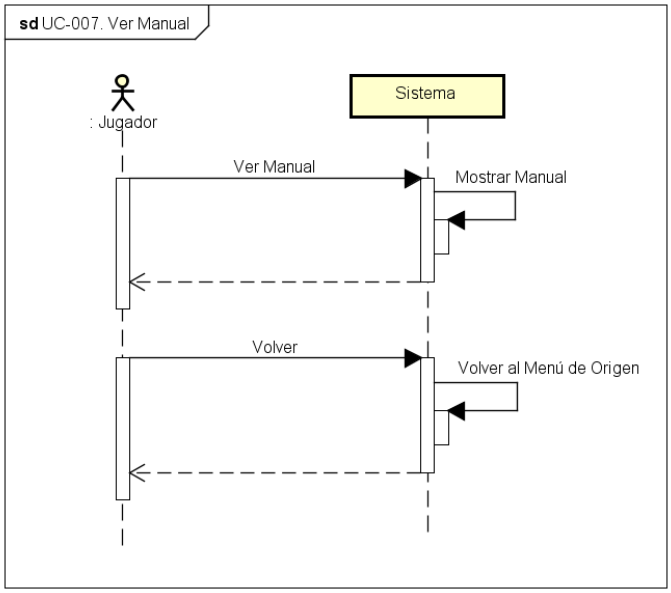


Figura 54. Diagrama de Secuencia de UC-007. Ver Manual.

5.3.5.8 UC-008. Ver Créditos

Caso de Uso	**UC-008. Ver Créditos.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario seleccione la opción del menú “Créditos”.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa la opción del menú principal “Créditos”.
	2	El **sistema** cambia a la vista de los créditos y los muestra.
	3	El **actor** pulsa sobre el botón “Volver” para salir.
	4	El **sistema** cierra la vista de créditos y vuelve a la vista del menú principal, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	3	El **actor** pulsa la tecla “Esc”.
	a	El **sistema** actúa como si se hubiese pulsado el botón “Volver”, desembocando en el **paso 4**.
Precondiciones	El sistema se encuentra en ejecución y el usuario está situado en el menú principal.	
Postcondiciones	El usuario se encuentra en el menú principal tras haber visto correctamente los créditos.	
Requisitos relacionados	**RF-013.**	
Notas	Al igual que el manual, los créditos también se muestran en modo *scroll*, con la misma consecuencia expuesta en las notas del caso de uso **UC-007. Ver Manual**.	

Tabla 50. UC-008. Ver Créditos.

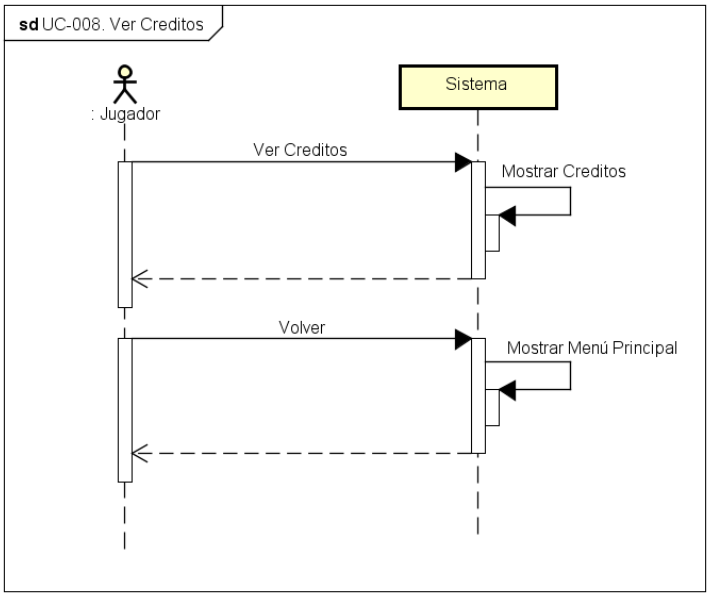


Figura 55. Diagrama de Secuencia de UC-008. Ver Créditos.

5.3.5.9 UC-009. Ver Configuración

Caso de Uso	**UC-009. Ver Configuración.**	
Fuentes	Rubén Moya Vázquez	
Actor	**Act-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario seleccione la opción del menú "Opciones".	
Flujo principal	**Paso**	**Acción**
	1	El **actor** pulsa la opción del menú "Opciones".
	2	El **sistema** cambia a la vista de configuración y muestra el menú de configuración.
	3	El **actor** hace click sobre la pestaña que desea visualizar.
	4	El **sistema** muestra la pestaña seleccionada.
	5	El **actor** pulsa en el botón "Volver" para salir.
	6	El **sistema** cierra la vista de configuración y vuelve a la vista del menú de partida, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	3	El **actor** pulsa en el botón "Volver" o la tecla "Esc".
		El **sistema** responde realizando el paso 6.
	5	El **actor** pulsa la tecla "Esc".
		El **sistema** responde realizando el paso 6.
Precondiciones	El sistema se encuentra en ejecución y el usuario está situado en el menú principal o en el menú de pausa del juego.	
Postcondiciones	El usuario se encuentra de vuelta en el menú del que proviniese tras haber visto correctamente la configuración del sistema.	
Requisitos relacionados	**RF-017.**	
Puntos de extensión	**1**	Una vez realizado el paso 4 del flujo principal, el caso de uso actual puede extenderse al caso de uso **UC-010. Modificar Configuración de Sonido.**
	2	Una vez realizado el paso 2 del flujo principal, el caso de uso actual puede extenderse al caso de uso **UC-011. Modificar Configuración de Video.**
	3	Una vez realizado el paso 4 del flujo principal, el caso de uso actual puede extenderse al caso de uso **UC-012. Cambiar Idioma.**
	4	Una vez realizado el paso 4 del flujo principal, el caso de uso actual puede extenderse al caso de uso **UC-013. Cambiar Estilo de las Piezas.**
Notas	El punto de extensión no es el mismo para **UC-011. Modificar Configuración de Video** que para el resto de casos de extensión porque la primera pestaña en cargar es la de la configuración de video.	

Tabla 51. UC-009. Ver Configuración.

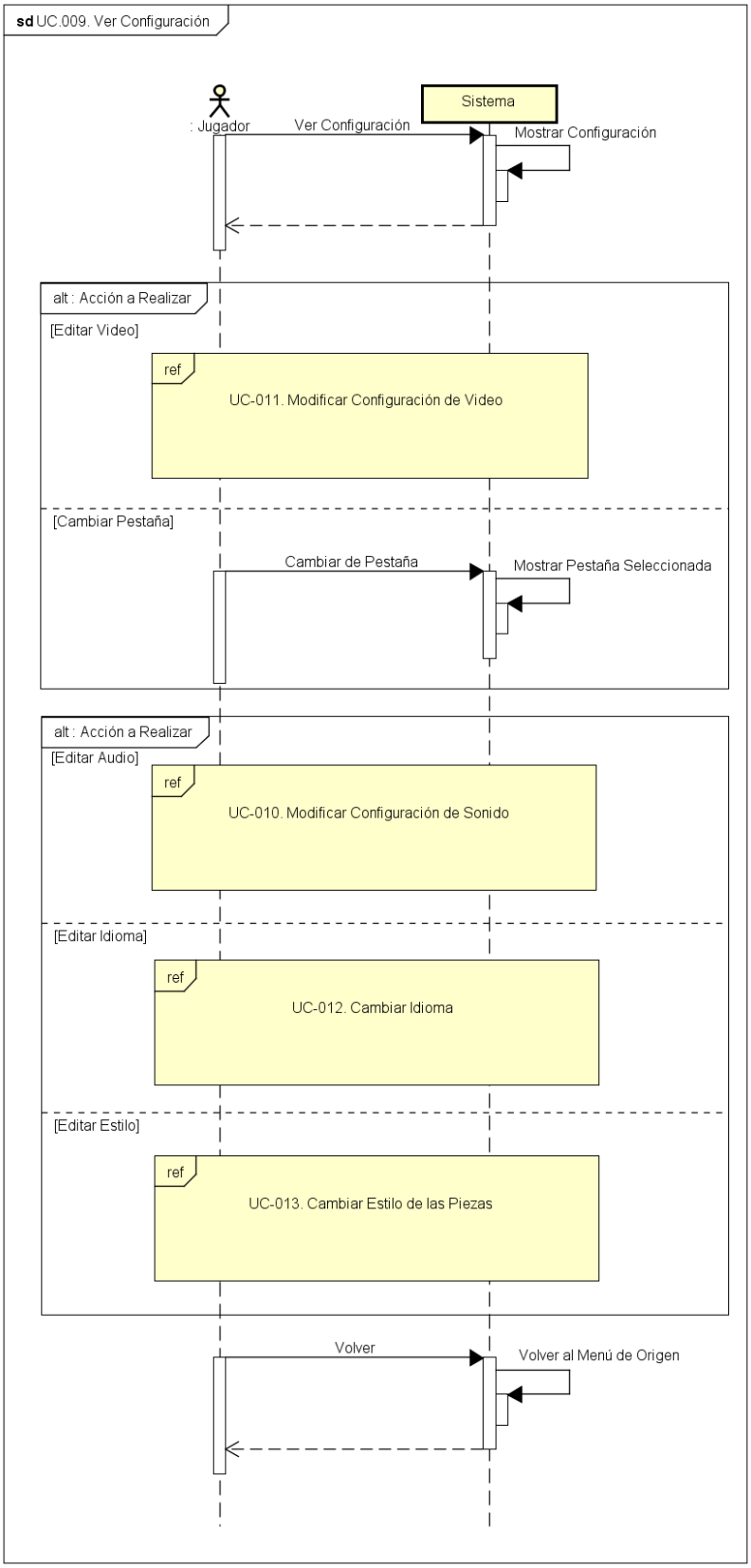


Figura 56. Diagrama de Secuencia de UC-009. Ver Configuración.

5.3.5.10 UC-010. Modificar Configuración de Sonido

Caso de Uso	**UC-010. Modificar Configuración de Sonido.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario desee realizar un cambio en la configuración de sonido.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** cambia una o más de las configuraciones de la pestaña "Audio".
	2	El **actor** pulsa el botón "Aplicar Configuración" para guardar los cambios.
	3	El **sistema** sobrescribe la antigua configuración con los cambios aplicados, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	2	El **actor** no pulsa en ningún momento "Aplicar Configuración".
	a	El **sistema** no cambiará la configuración y al salir de la vista actual se perderán los cambios, quedando el caso de uso sin efecto.
Precondiciones	El usuario se encuentra realizando el caso de uso **UC-009. Ver Configuración** y aplica un cambio en alguna de las opciones de la pestaña "Audio".	
Postcondiciones	El cambio aplicado por el usuario se guarda correctamente en la configuración del sistema.	
Requisitos relacionados	**RF-009.**	
Notas	El flujo de este caso de uso puede combinarse con los relativos a los casos de uso **UC-011. Modificar Configuración de Video**, **UC-012. Cambiar Idioma** y **UC-013. Cambiar Estilo de las Piezas**.	

Tabla 52. UC-010. Modificar Configuración de Sonido.

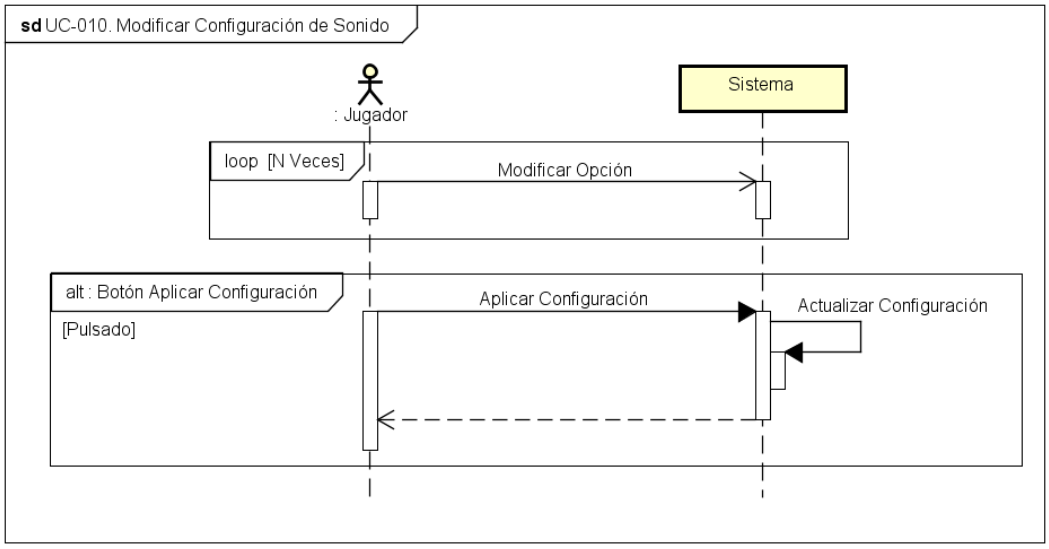


Figura 57. Diagrama de Secuencia de UC-010. Modificar Configuración de Sonido.

5.3.5.11 UC-011. Modificar Configuración de Video

Caso de Uso	**UC-011. Modificar Configuración de Video.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario desee realizar un cambio en la configuración de video.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** cambia una o más de las configuraciones de la pestaña "Video".
	2	El **actor** pulsa el botón "Aplicar Configuración" para guardar los cambios.
	3	El **sistema** sobrescribe la antigua configuración con los cambios aplicados, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	2	El **actor** no pulsa en ningún momento "Aplicar Configuración".
	a	El **sistema** no cambiará la configuración y al salir de la vista actual se perderán los cambios, quedando el caso de uso sin efecto.
Precondiciones	El usuario se encuentra realizando el caso de uso **UC-009. Ver Configuración** y aplica un cambio en alguna de las opciones de la pestaña "Video".	
Postcondiciones	El cambio aplicado por el usuario se guarda correctamente en la configuración del sistema.	
Requisitos relacionados	**RF-010.**	
Notas	El flujo de este caso de uso puede combinarse con los relativos a los casos de uso **UC-010. Modificar Configuración de Sonido**, **UC-012. Cambiar Idioma** y **UC-013. Cambiar Estilo de las Piezas**.	

Tabla 53. UC-011. Modificar Configuración de Video.

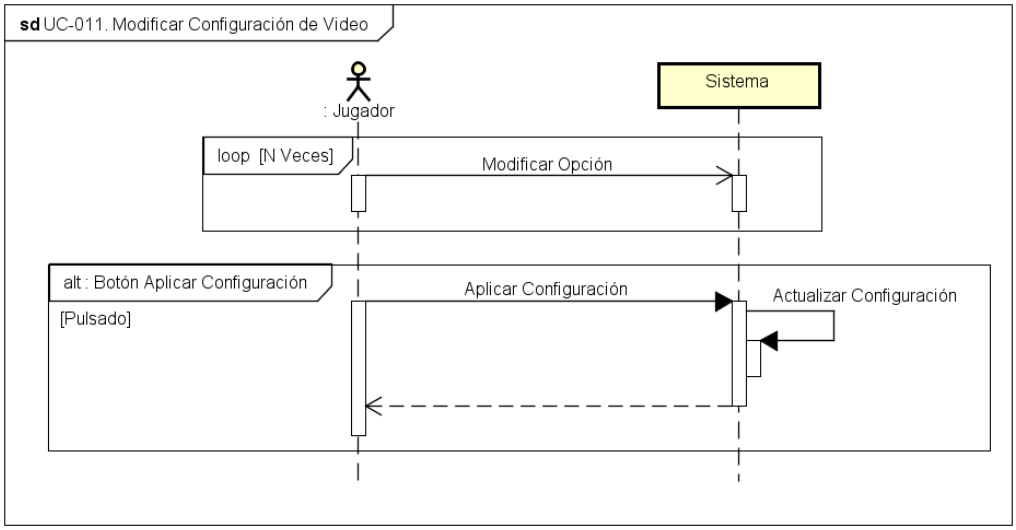


Figura 58. Diagrama de Secuencia de UC-011. Modificar Configuración de Video.

5.3.5.12 UC-012. Cambiar Idioma

Caso de Uso	**UC-012. Cambiar Idioma.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario desee cambiar el idioma en el que se muestra el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** cambia el idioma en la pestaña "Idioma".
	2	El **actor** pulsa el botón "Aplicar Configuración" para guardar los cambios.
	3	El **sistema** sobrescribe la antigua configuración con los cambios aplicados, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	2	El **actor** no pulsa en ningún momento "Aplicar Configuración".
	a	El **sistema** no cambiará la configuración y al salir de la vista actual se perderán los cambios, quedando el caso de uso sin efecto.
Precondiciones	El usuario se encuentra realizando el caso de uso **UC-009. Ver Configuración** y aplica un cambio en la pestaña "Idioma".	
Postcondiciones	El cambio aplicado por el usuario se guarda correctamente en la configuración del sistema.	
Requisitos relacionados	**RF-011.**	
Notas	El flujo de este caso de uso puede combinarse con los relativos a los casos de uso **UC-010. Modificar Configuración de Sonido**, **UC-011. Modificar Configuración de Video** y **UC-013. Cambiar Estilo de las Piezas**.	

Tabla 54. UC-012. Cambiar Idioma.

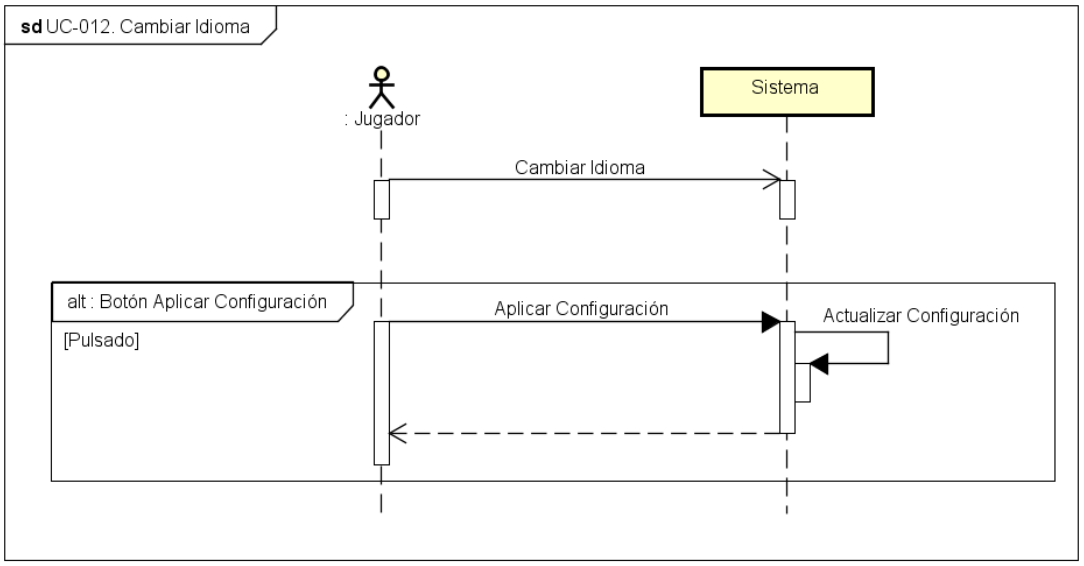


Figura 59. Diagrama de Secuencia de UC-012. Cambiar Idioma.

5.3.5.13 UC-013. Cambiar Estilo de las Piezas

Caso de Uso	**UC-013. Cambiar Estilo de las Piezas.**	
Fuentes	Rubén Moya Vázquez	
Actor	**ACT-001. Jugador**	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario desee cambiar el estilo de las piezas.	
Flujo principal	**Paso**	**Acción**
	1	El **actor** cambia el estilo de las piezas en la pestaña "Configuración del Juego".
	2	El **actor** pulsa el botón "Aplicar Configuración" para guardar los cambios.
	3	El **sistema** sobrescribe la antigua configuración con los cambios aplicados, finalizando el caso de uso.
Flujos alternativos	**Paso**	**Acción**
	2	El **actor** no pulsa en ningún momento "Aplicar Configuración".
	a	El **sistema** no cambiará la configuración y al salir de la vista actual se perderán los cambios, quedando el caso de uso sin efecto.
Precondiciones	El usuario se encuentra realizando el caso de uso **UC-009. Ver Configuración** y aplica un cambio en alguna de las opciones de la pestaña "Configuración del Juego".	
Postcondiciones	El cambio aplicado por el usuario se guarda correctamente en la configuración del sistema.	
Requisitos relacionados	**RF-012.**	
Notas	El flujo de este caso de uso puede combinarse con los relativos a los casos de uso **UC-010. Modificar Configuración de Sonido**, **UC-011. Modificar Configuración de Video** y **UC-012. Cambiar Idioma**.	

Tabla 55. UC-013. Cambiar Estilo de las Piezas.

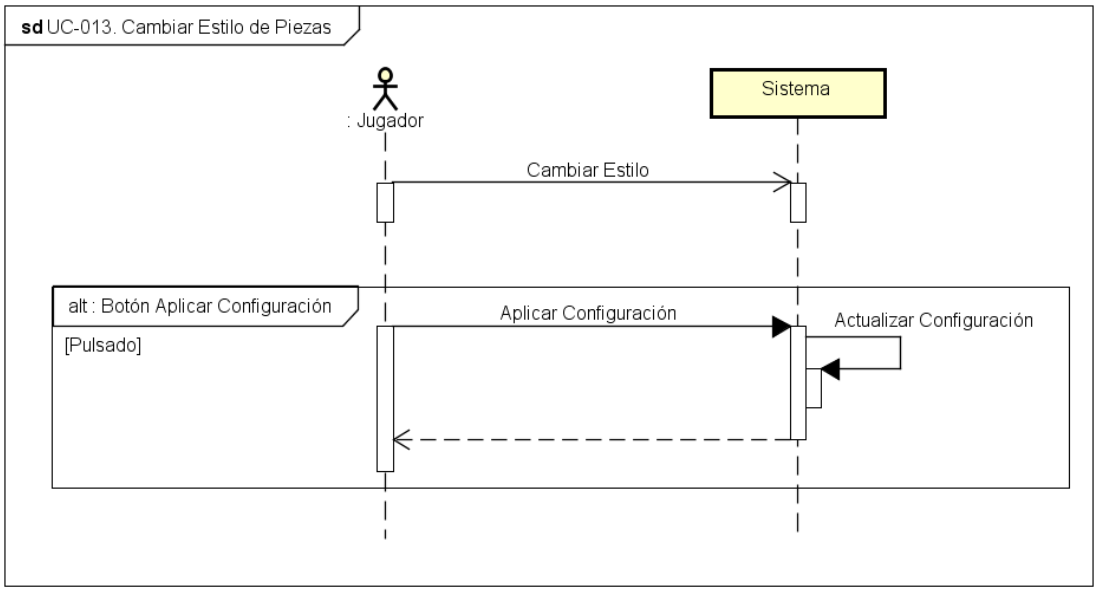


Figura 60. Diagrama de Secuencia de UC-013. Cambiar Estilo de las Piezas.

5.3.6 Modelo de Dominio del Sistema

Con el objetivo de proporcionar una visión de conjunto del funcionamiento del sistema y mostrar las principales clases de análisis que actúan en el desarrollo de la funcionalidad del mismo, adjuntamos el diagrama de clases y una breve descripción de cada clase representada.

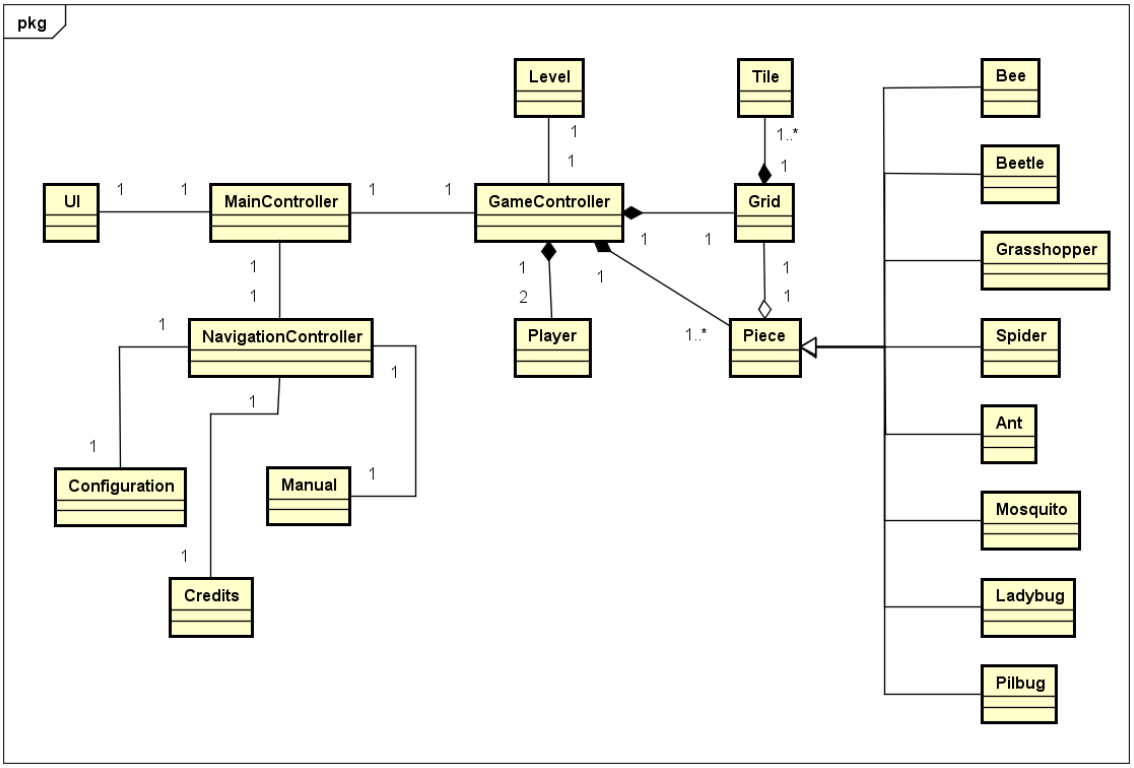


Figura 61. Modelo de Dominio del sistema.

5.3.6.1 Clases de Análisis

A continuación, describiremos brevemente cada una de las clases involucradas en el Modelo de Dominio aclarando su relevancia y el papel que desempeñan en el sistema. Estas clases no representan plenamente a las clases finales de las que se compondrá el sistema, sino que son versiones de un nivel de abstracción superior, pertenecientes al dominio de la solución.

5.3.6.1.1 UI

Es la interfaz de usuario y como tal se encarga de representar y transmitir la información y acciones del usuario y del sistema de forma que sean comprensibles para el primero. Siguiendo el patrón de diseño MVC, esta clase actuará bajo el rol de "vista".

5.3.6.1.2 MainController

Es la clase de arranque o clase principal de nuestro sistema, de la que parte todo el funcionamiento del mismo y gestiona de manera central el flujo de acción e información. Siguiendo el patrón de diseño MVC, actúa bajo el rol de "controlador", delegando el control de la lógica del juego a la clase ***GameController*** y la lógica de la navegación y los elementos de configuración del sistema a ***NavigationController***.

5.3.6.1.3 NavigationController

Es la clase encargada de controlar la lógica relativa a la navegación por los menús de juego y la configuración del mismo. Efectúa el rol de "controlador" del patrón MVC, gestionando toda la lógica no relacionada con el propio juego. Se comunica únicamente con la interfaz de usuario, vía ***MainController***, con la clase ***Configurativo*** del modelo y las vistas estáticas **Credits** y **Manual**.

5.3.6.1.4 GameController

Es la clase encargada de gestionar la lógica relativa al propio juego, por tanto, se ocupa de cuestiones relativas a la gestión de turnos, situación inicial y final, estado del tablero y gestión de los jugadores y sus acciones.

5.3.6.1.5 Configuration

Clase del modelo que contiene la configuración actual de audio, idioma y video del sistema y los métodos necesarios para mostrar y modificar la información relativa a dichas configuraciones.

5.3.6.1.6 Credits

Vista estática con el contenido de los créditos.

5.3.6.1.7 Manual

Vista estática con el contenido del manual.

5.3.6.1.8 Level

Nivel de juego que representa el entorno en el que se realizará la partida. Contiene referencias a todo el contenido gráfico, cámaras, iluminación, HUD, etc.…

5.3.6.1.9 Player

Clase que representa a cada uno de los contrincantes de la partida. Presenta métodos para informar y realizar acciones de movimiento y colocación de piezas, así como el número de piezas de cada tipo disponibles para colocar. Como atributo principal, cuenta con el identificador del color de sus piezas, utilizado para marcar el turno y las piezas con las que puede interactuar.

5.3.6.1.10 Grid

Representación relativa al modelo del sistema del tablero de juego. Contiene información relativa a la colocación de todas las piezas en juego y métodos para obtener información relativa a movimientos realizables, piezas en juego y estado actual de la partida.

5.3.6.1.11 Tile

Representa a cada una de las casillas del tablero. Contiene información relativa a su posición, su estado y si está ocupada o no.

5.3.6.1.12 Piece

Clase abstracta relativa al modelo del sistema que representa todas y cada una de las piezas disponibles para jugar en una partida. Su principal método, implementado en sus clases herederas, es el que define el modo en que una ficha se mueve según el tipo del que sea.

Sirve de adaptador entre el método para mover piezas de la clase ***Player*** y el de cada pieza.

6 Diseño

En este capítulo nos centraremos en describir la arquitectura lógica de nuestro sistema, así como el patrón arquitectónico usado en su diseño y la versión de este.

Una vez seleccionado el patrón arquitectónico que utilizaremos, pasaremos a describir el proceso de adaptarlo a nuestras necesidades, utilizando como guía visual el diagrama de paquetes y aportando una explicación de las clases contenidas en cada paquete y su funcionalidad.

Finalmente, expondremos brevemente los patrones de diseño aplicados a la creación de nuestro proyecto, con el objetivo de dar una visión más en profundidad al trabajo realizado.

Cabe destacar que omitiremos intencionalmente la explicación de la arquitectura física ya que nuestro proyecto se ejecuta únicamente en un equipo, sin comunicación de ningún tipo con el exterior.

6.1 Arquitectura Lógica

Un videojuego es un producto software muy extenso y complejo. Si no se observa adecuadamente en su conjunto, la arquitectura seleccionada para el producto puede no ser la adecuada. En este caso, si atendiésemos solo al sistema de ventanas de menú, podríamos deducir erróneamente que el patrón de diseño arquitectónico ante el que nos encontramos es el *Modelo-Vista-Controlador*. Pero estaríamos equivocados, ya que nos encontramos ante un caso que debe afrontarse aplicando la *Arquitectura Dirigida por Eventos* (*EDA*). Dicha arquitectura promueve la creación, detección, consumo y reacción a eventos; entendiendo por evento todo aquello que provoque un cambio significativo en un estado.

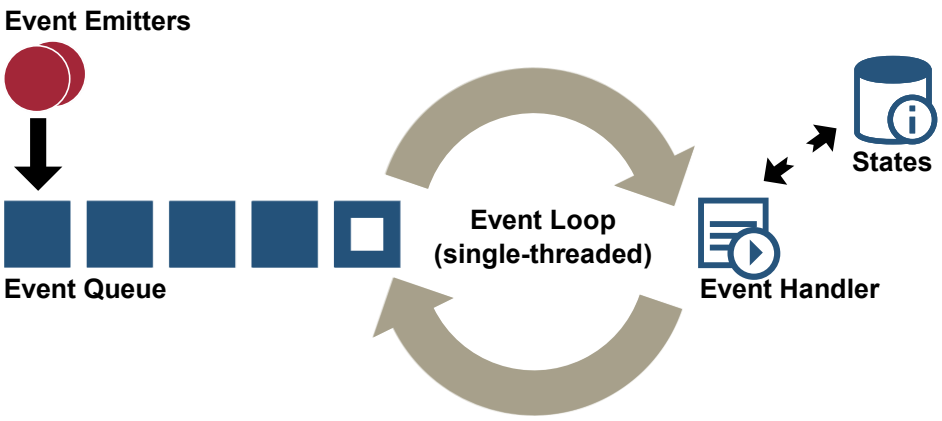


Figura 62. Arquitectura EDA.

La arquitectura EDA se basa en cuatro capas lógicas que se encargan de detectar el evento, capturar la información relativa al mismo, transmitirla al receptor encargado de gestionarla y generar un conjunto no vacío de respuestas a dicho evento. Las cuatro capas mencionadas y sus funciones asignadas son:

1. **Generador de Eventos.** Esta capa es la encargada de detectar la aparición de un hecho y generar un evento en consecuencia. Los generadores de eventos responden a la aparición de hechos concretos y transmiten los eventos generados al Canal de Eventos.

2. **Canal de Eventos.** Esta capa es el medio mediante el cual los eventos generados por cada uno de los generadores de eventos son transmitidos al Motor de Procesamiento. Puede haber más de un canal funcionando a la vez y una vez entregados los eventos a la cola de procesamiento del Motor, dichos eventos pueden ser procesados de manera síncrona o asíncrona.

3. **Motor de Procesamiento de Eventos.** El Motor de Procesamiento de Eventos es el encargado de identificar un evento y asignarle la respuesta adecuada. La reacción ejecutada a un evento por el Motor de Procesamiento puede desencadenar a su vez una serie de reacciones en consecuencia.

4. **Respuesta al Evento.** Es el resultado final de la captura del evento. En esta capa se muestran las consecuencias definitivas de la aparición del evento que disparó la ejecución.

Hay varios estilos a la hora de aplicar la arquitectura *EDA* a un desarrollo software. En nuestro caso, por el tipo de eventos que se pueden generar durante la ejecución de nuestro sistema, el estilo que debemos aplicar es el *Procesamiento por Flujo de Eventos* (*Event Stream Processing, ESP*). En este estilo de EDA, los eventos son sometidos a procesamiento secuencial de los mismos, lo que permite conducir el flujo de información.

6.2 Diagrama de Diseño

A continuación, mostraremos el diagrama de paquetes que representa la adaptación del patrón EDA a nuestro sistema. Posteriormente detallaremos el contenido de dichos paquetes para mayor comprensión del modelo y el sistema en si.

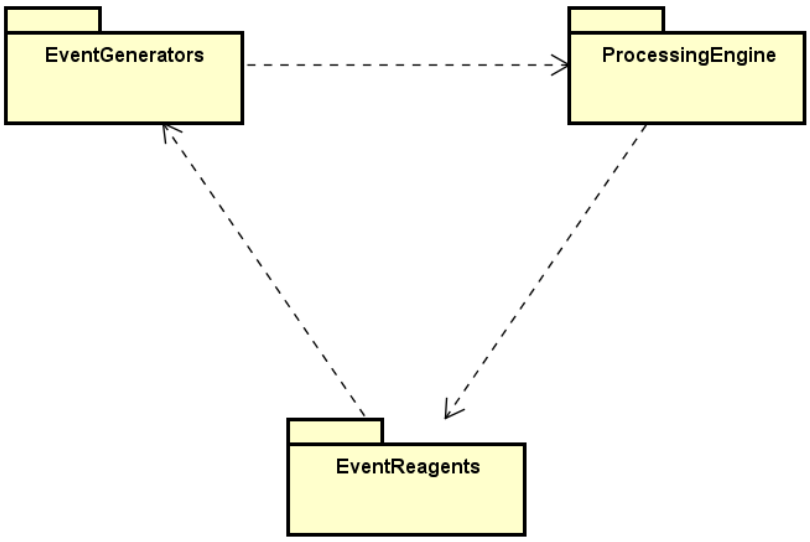


Figura 63. Diagrama de Paquetes del sistema.

Tal y como se puede ver en la imagen, hemos omitido el paquete relativo al *Canal de Eventos*. Esto se debe principalmente al hecho de que el único elemento que actúa bajo ese rol es el nivel de juego, cuya lógica está codificada en la clase *Blueprint* de nivel, ligada al mismo.

Por otra parte, las clases más representativas contenidas en cada paquete son las siguientes.

- **EventGenerators.** Las clases bajo este paquete son generadores de eventos que capturan la aparición de un hecho relevante y crean un evento en consecuencia.
  - ***HUD.*** El HUD es junto con los elementos físicos que representan a las piezas y casillas, el mayor representante del paquete. Su función es simplificar y acotar las acciones realizadas por el usuario y emitir eventos para cada una de las mismas.
  - ***MainMenu.*** El menú principal y el resto de ventanas de menú son generadores de eventos que permiten al usuario navegar y aplicar cambios en la configuración del sistema o en la partida.

- **ProcesingEngine.** Bajo este paquete se encuentra la clase encargada de organizar y gestionar los eventos generados por el paquete anterior y asignárselos a una de las clases del siguiente paquete, para su respuesta.
  - ***GameMode.*** El modo de juego es el procesador de eventos principal. No solo gestiona los eventos relativos al flujo de la partida, sino todos los relativos al juego, ya que también actúa, en mayor o menor grado, como motor de procesamiento para los eventos de las clases del menú.

- **EventReagents.** Este paquete contiene todas las clases encargadas de reaccionar frente a la aparición de un evento concreto. Dichas clases no solo actúan en consecuencia, sino que pueden desencadenar la aparición de nuevos hechos que generen nuevos eventos.
  - ***Player.*** Responde a los eventos relativos al estado de los jugadores.
  - ***Piece.*** Clase abstracta que actúa como interfaz adaptadora entre el motor de procesamiento y cada pieza concreta a la hora de responder a los eventos relativos a los estados de las mismas.
  - ***Grid.*** Clase que responde a los eventos que provocan cambios en los estados del tablero.

6.3 Patrones Utilizados

Durante el proceso del desarrollo de nuestro proyecto, aplicaremos muchos patrones de diseño software [7] tanto directa como indirectamente. Dicha situación es deseable ya que concuerda con los principios de reutilización y modularidad del diseño de software orientado a objetos.

La gran mayoría de los patrones que incluirá nuestro proyecto serán transparentes al diseñador y al programador, ya que están integrados en el *framework* que sustenta a nuestro juego. Como, por ejemplo, el patrón *Comando*, aplicado a la asignación de acciones a las teclas pulsadas en el teclado o el patrón *Flyweight*, aplicado en el follaje con el que se puebla el nivel de juego. Por ese motivo, teniendo en cuenta que solo podemos describir aquellos patrones que utilizamos activamente y con total seguridad, solo describiremos aquellos que hemos aplicado intencionalmente en nuestro diseño del software a desarrollar.

El único patrón utilizado activamente en nuestro sistema es el patrón *Adaptador*. Como puede verse en la siguiente imagen, usamos el patrón adaptador con el objetivo de abstraer el tipo de pieza con el que se interactúa en cada momento a la hora de solicitar la ejecución de los métodos utilizados para saber si la pieza puede ser movida y las casillas alcanzables por la misma.

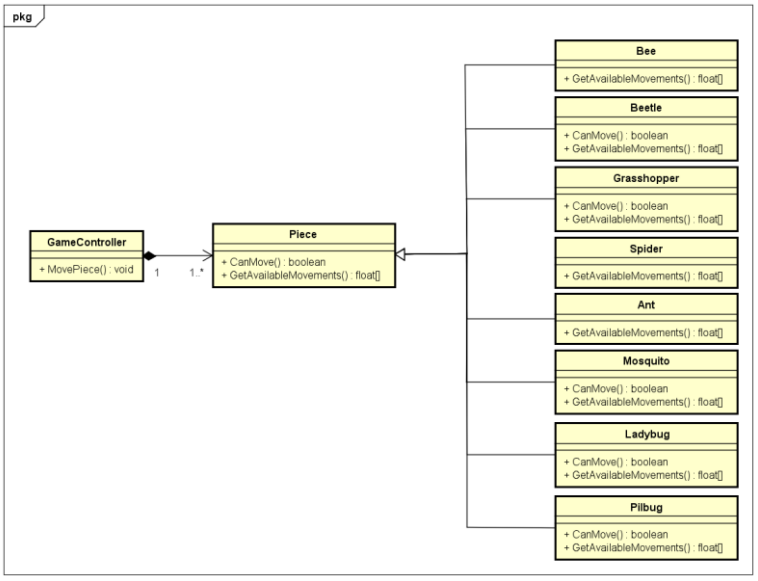


Figura 64. Aplicación del Patrón Adaptador.

En caso de haber tenido un modelo de clases distinto, en el que las piezas se relacionaban directamente con el tablero que las contenía, podíamos haber utilizado en su lugar el patrón *Flyweight*, pero las peculiaridades de nuestro tablero "*Grid*" impedían dicha situación.

7 Implementación

A continuación, describiremos los aspectos más destacables de la fase de implementación y sus tareas relacionadas.

La implementación de un videojuego tiene muchas actividades distintas, de las cuales la mayoría no tienen que ver con la programación, sino con el diseño gráfico. Aun así, destacaremos puntos relevantes de dichas áreas con el objetivo de dar una imagen del total de las tareas realizadas por el equipo de desarrollo.

7.1 Desarrollo de la Interfaz de Usuario

La interfaz de usuario está formada por dos componentes fundamentales, la navegación y el *HUD* [5] [6]. Ambos componentes han sido desarrollados usando el sistema de visual scripting *Blueprint*, pero los componentes usados como base para su creación y su funcionamiento les hacen claramente diferentes.

7.1.1 Navegación

La navegación es el conjunto de ventanas de menú por el que el usuario se desplaza mientras la partida no está en ejecución. Dichas ventanas han sido generadas tomando como base clases *Blueprint* del tipo widget, lo cual nos permite aunar la vista y su controlador en un mismo componente lógico, simplificando su edición y comunicación con otros componentes.

Figura 65. Menú Principal del Juego.

Tal y como se ve a continuación el editor de widgets *Blueprint* cuenta con dos vistas principales. Por un lado, está la pestaña "Diseño" que nos permite editar el aspecto y componentes gráficos de la vista de nuestro widget y por el otro, la pestaña de "Gráfico", que nos permite editar la funcionalidad del controlador relativo a dicha vista.

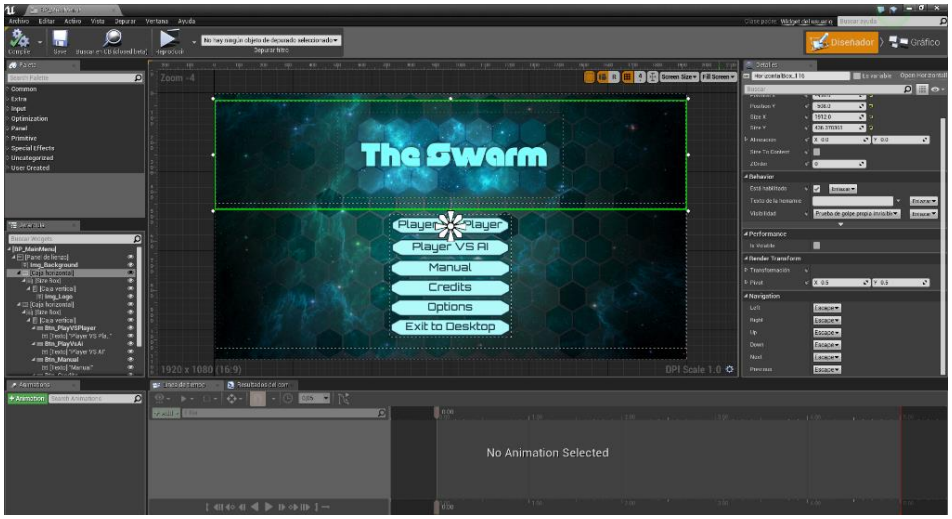


Figura 66. Vista de Diseño.

Crear la navegación a partir de dichos componentes no solo facilitó la edición de estilos y personalización de componentes, sino que automatizó el proceso de captura de eventos y el flujo de comunicación entre los distintos widgets que componen la navegación.

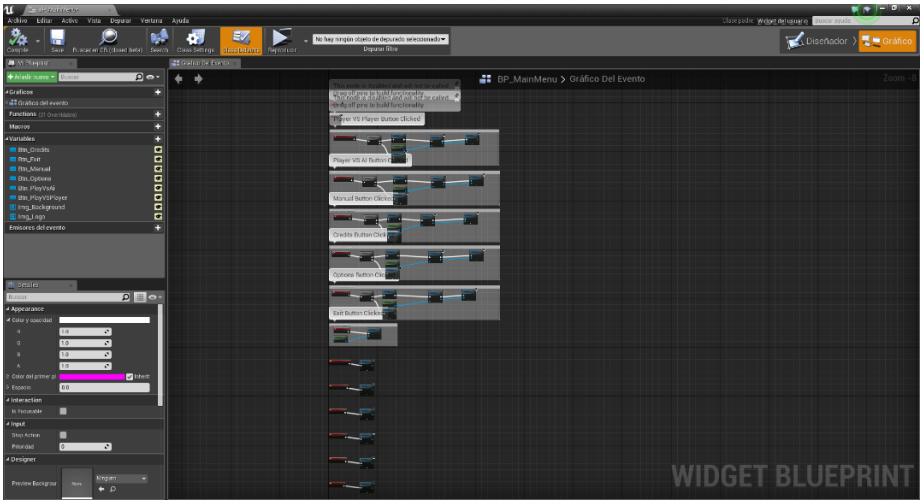


Figura 67. Vista de Grafico.

Por último, cabe destacar el hecho de que para guardar la configuración del sistema no utilizamos una base de datos, sino que utilizamos una clase *Blueprint* del tipo *SaveGame* que de cara al programador funciona como una clase del modelo, pero internamente gestiona la persistencia de los datos de manera automática.

7.1.2 *HUD*

El *HUD* o *Head-Up Display* es el sistema que nos permite mantener un flujo de comunicación constante con el usuario durante la partida, manteniéndole informado del estado de la partida y permitiéndole acciones como seleccionar la ficha que desea colocar o pausar el juego.

Para la creación del *HUD* hemos utilizado una clase *Blueprint* del mismo nombre. El editor de esta clase es prácticamente idéntico al de los widgets *Blueprint* y su principal diferencia con estos es que esta clase esta optimizada para ser asignada como HUD de un nivel de forma automática y facilita la gestión de variables de nuestras clases C++ a través de código Blueprint. De esta forma, podremos mantener actualizado en pantalla el número de piezas disponibles para colocar o el color del jugador al que corresponde el turno de una manera mucho más eficiente.

7.2 Diseño del Nivel

El diseño de nivel [8] [9] engloba todas las tareas necesarias para crear el entorno en el que se desarrollará el juego. Desde la creación y edición del terreno y sus texturas, iluminación y sonidos ambientales, hasta la posición de la cámara y sus movimientos.

En nuestro caso era el aspecto más desdeñable del desarrollo del juego ya que fundamentalmente, lo único que necesitamos para poder jugar es una superficie lisa relativamente extensa. Aun así, quisimos darle un aspecto visual cuidado, natural y campestre, ya que el juego trata de insectos.

7.2.1 Terreno y Follaje

Para la creación de nuestro mapa quisimos representar un pequeño valle por el que pasa un riachuelo. Para ello, en primer lugar. utilizamos una serie de recursos disponibles en el paquete de contenido básico y creamos nuestro material "*M_Landscape*" el cual utilizamos para aplicar texturas y granularidad al terreno modelado a partir del editor de niveles.

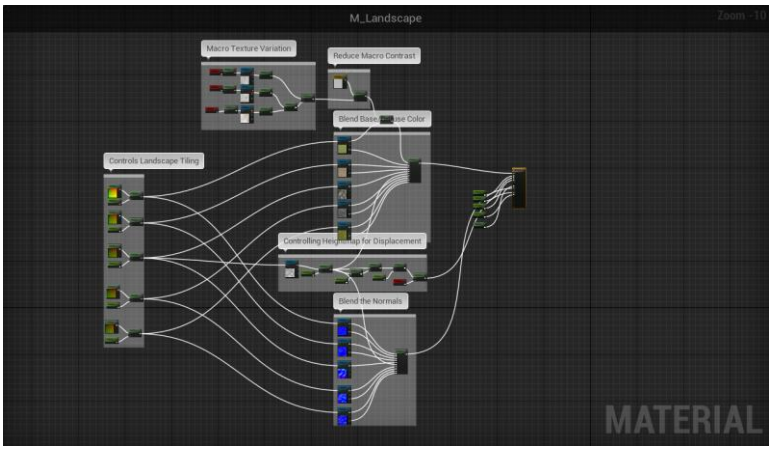


Figura 68. M_Landscape visto desde el editor de materiales.

Una vez creado el terreno, habiendo aplicado las texturas y creado la iluminación ambiental, poblamos dicho terreno de contenido extraído de la biblioteca de contenido gratuito de *Unreal Engine* como rocas, hierba, flores, árboles o arbustos.

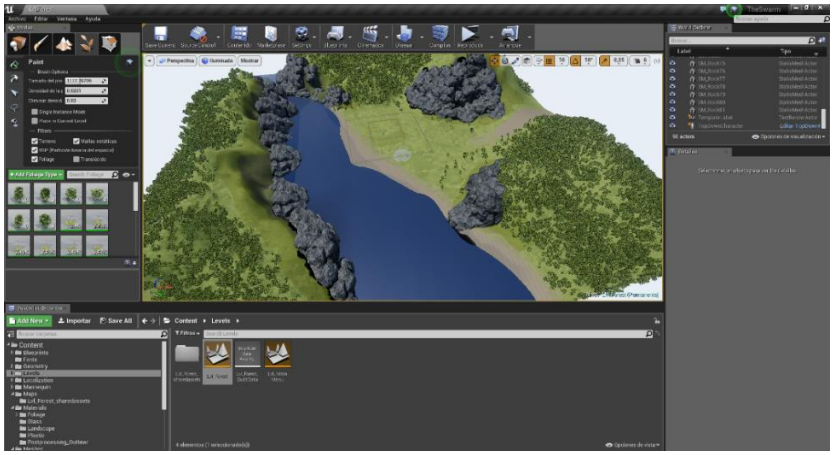


Figura 69. Mapa de Juego visto desde el editor de niveles.

Podríamos haber profundizado más en esta área y haber creado un mapa más fotorrealista, pero dada la planificación y el uso de recursos respecto a la mejora en el resultado final, consideramos adecuado el nivel de calidad conseguido con las acciones realizadas.

7.2.2 *Blueprint* de Nivel

Todo nivel de juego en *Unreal Engine* tiene ligado, a modo de controlador empotrado, una clase *Blueprint* de Nivel. Dicha clase sirve para inicializar funcionalidades en la carga del mapa y dependiendo del juego y sus mecánicas puede llegar a ser muy complejo.

En nuestro caso utilizamos dicha clase únicamente como nexo de conexión con el *HUD* y para iniciar ciertos aspectos del juego con el nivel, como los sonidos ambientales o la música de fondo.

7.3 Diseño de Materiales y Estructuras

El diseño de materiales y estructuras es uno de los pilares fundamentales del desarrollo de un videojuego. En nuestro caso, por la simplicidad estética del juego, no ha tenido tanta carga de trabajo, ya que los modelos a realizar han sido únicamente los de las piezas y el usado para las casillas vacías y estos eran, en todos los casos, prismas de base hexagonal con un dibujo grabado, en el caso de las piezas, o plano, en el caso de la casilla.

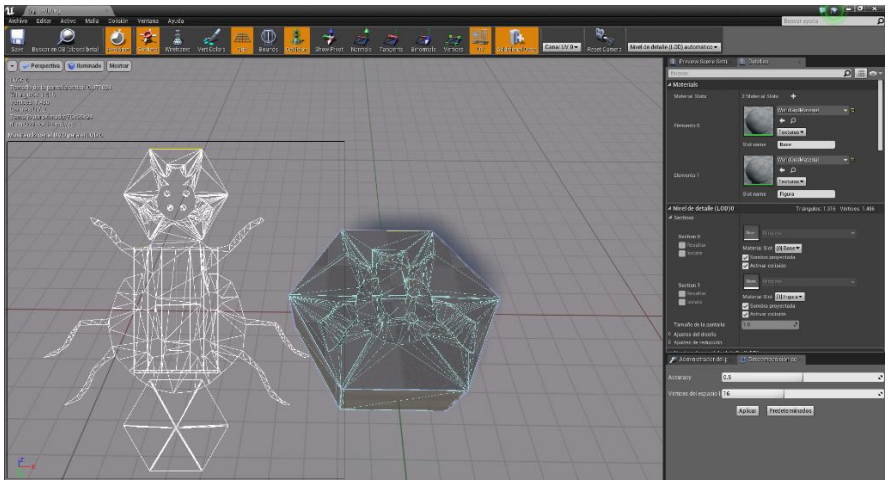


Figura 70. Pieza Mariquita en el editor de mallas estáticas de UE4.

Los materiales sin embargo han requerido más trabajo, ya que no solo requeríamos uno para cada color de las piezas y otro, cristalino, para las casillas, sino además, uno para cada tipo de iluminación que quisiésemos usar, a través de volúmenes de post-procesado, para resaltar las piezas y casillas según estuviesen disponibles para mover, colocar o usar habilidades, o no.

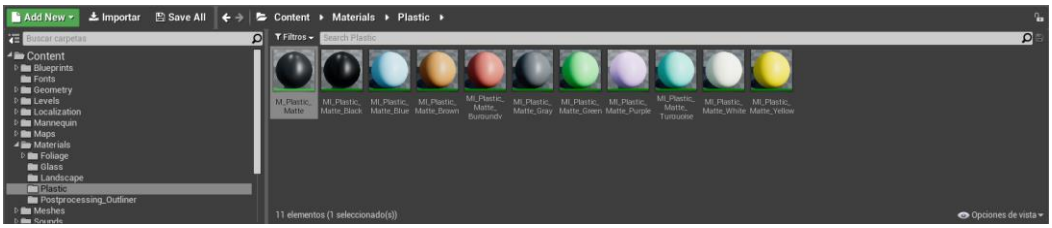


Figura 71. Muestra de parte de los materiales utilizados en el juego.

El resto de materiales y estructuras utilizados, como árboles, hojas, hierba o piedras, han sido extraídos de la biblioteca de contenidos gratuitos de *Unreal Engine* y se usan de acuerdo a su licencia en el caso de tenerla.

7.4 Programación del Videojuego

EL juego de la colmena es un juego con una complejidad relativamente particular a la hora de transcribir sus reglas y funcionamiento a un lenguaje de programación [3] [4]. Esto se debe fundamentalmente al hecho de que el juego original no tiene tablero, ya que las propias piezas lo forman y dichas piezas son hexagonales, complicando la posibilidad de transcribir las posiciones a coordenadas cartesianas o índices de una matriz bidimensional. Dichas cuestiones las hemos resuelto con la creación de nuestra clase *Grid*.

7.4.1 Grid Hexagonal

La clase *Grid*, representa el tablero de casillas hexagonales sobre el que jugamos. Dicho tablero es una malla con forma hexagonal compuesta a su vez de hexágonos y posee un sistema de coordenadas cubicas [74] que nos permite establecer unos ejes y vectores direccionales con los que operar para obtener todas las funcionalidades que necesitamos.

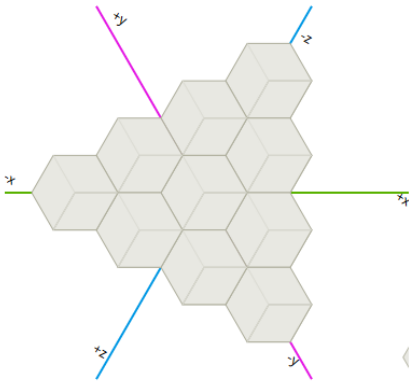


Ilustración 14. Sistema de coordenadas cubicas.

Esta clase funciona como biblioteca de funciones auxiliares sobre las que construimos toda la lógica de las reglas y movimientos de piezas de nuestro juego. Cuenta con funciones que nos permiten buscar caminos entre posiciones, teniendo en cuenta o no casillas ocupadas; detectar áreas cerradas o inaccesibles, calcular distancias, obtener las celdas vecinas a una, dadas sus coordenadas y más.

Teniendo en cuenta todo lo mencionado anteriormente y que la funcionalidad de dicho tablero únicamente depende de 2 clases que no necesitan comunicación con otras para desarrollar su funcionalidad interna (Grid y Tile), podemos afirmar que hemos creado una biblioteca de funciones plenamente reutilizable y que nos permitiría en el futuro la aplicación de la misma a una gran variedad de juegos de distintos géneros.

7.5 Localización

La localización es el proceso mediante el cual adaptamos la internacionalización existente en un producto software para una región concreta. En nuestro caso, hemos aprovechado las herramientas de internacionalización disponibles en *Unreal Engine* para localizar el juego a los países hispanohablantes y los países anglosajones. Para ello contamos con una subdivisión cultural siguiendo la norma *ISO-639* para cada una de las dos culturas y, ya que nuestro juego solo requiere traducción en los textos, una serie de traducciones para cadenas de texto identificadas por claves en el espacio de nombres.

El idioma por defecto será el inglés, pero desde la ventana de configuración, en la pestaña de idioma, se puede cambiar al español.

8 Pruebas

En este capítulo desarrollaremos las pruebas realizadas como parte de la fase final del desarrollo del proyecto, así como los resultados obtenidos y las correcciones aplicadas en caso de detectar un fallo. Las pruebas que registramos a continuación son únicamente las de Caja Negra, debido a que, al constar el equipo de desarrollo de un solo miembro y ser este el encargado de implementar todo el código del sistema, las pruebas de Caja Blanca no extraerían ninguna conclusión efectiva; ya que el responsable de la realización de las mismas se encuentra viciado respecto al código que debe analizar.

8.1 Pruebas de Caja Negra

Las pruebas de Caja Negra se construyen en base a los requisitos del sistema y el objetivo de las mismas es acotar los valores obtenidos respecto a los esperados según el rango de entradas posibles. De esta manera, se podrá medir el grado de cumplimiento de dichos requisitos.

El objetivo de las pruebas no es verificar la corrección del software, sino detectar errores en el mismo. Por tanto, una prueba tiene éxito cuando el resultado de la ejecución de la misma no es el esperado.

Para documentar el proceso de aplicación de las pruebas de Caja Negra utilizaremos las siguientes tablas como registro. En los casos en los que correspondiese, se realizaría a su vez un Análisis de Valores Límite para cubrir todas las posibles entradas.

Navegación	
PCN-001	Ver Manual en español
Versión	1.0
Descripción	Un usuario solicita ver el manual
Resultado esperado	El manual se muestra completamente y correctamente en español
Aplicación	Versión final
Valoración	OK

Tabla 56. PCN-001.

Navegación	
PCN-002	Ver Créditos en español
Versión	1.0
Descripción	Un usuario solicita ver los créditos
Resultado esperado	Los créditos se muestran completamente y correctamente en español
Aplicación	Versión final
Valoración	OK

Tabla 57. PCN-002.

Navegación	
PCN-003	Ver Manual en inglés
Versión	1.0
Descripción	Un usuario solicita ver el manual
Resultado esperado	El manual se muestra completamente y correctamente en inglés
Aplicación	Versión final
Valoración	OK

Tabla 58. PCN-003.

Navegación	
PCN-004	Ver Créditos en inglés
Versión	1.0
Descripción	Un usuario solicita ver los créditos
Resultado esperado	Los créditos se muestran completamente y correctamente en inglés
Aplicación	Versión final
Valoración	OK

Tabla 59. PCN-004.

Configuración	
PCN-005	Ver configuración en español
Versión	0.5
Descripción	Un usuario solicita ver la configuración del juego
Resultado esperado	Las opciones de configuración se muestran completamente y correctamente en español
Aplicación	Versión en desarrollo
Valoración	Incorrecto, los valores de gráficos no pueden traducirse

Tabla 60. PCN-005.

Configuración	
PCN-006	Ver configuración en español
Versión	1.0
Descripción	Un usuario solicita ver la configuración del juego
Resultado esperado	Las opciones de configuración se muestran completamente y correctamente en español
Aplicación	Versión final
Valoración	OK

Tabla 61. PCN-006.

Configuración	
PCN-007	Ver configuración en ingles
Versión	1.0
Descripción	Un usuario solicita ver la configuración de juego
Resultado esperado	Las opciones de configuración se muestran completamente y correctamente en inglés
Aplicación	Versión final
Valoración	OK

Tabla 62. PCN-007.

Configuración	
PCN-008	Modificar configuración de Gráficos
Versión	1.0
Descripción	Un usuario solicita modificar la configuración gráfica
Resultado esperado	La configuración se aplica y se guarda
Aplicación	Versión final
Valoración	OK

Tabla 63. PCN-008.

Configuración	
PCN-009	Modificar configuración de Sonido
Versión	1.0
Descripción	Un usuario solicita modificar la configuración de sonido
Resultado esperado	Una vez aplicada, la configuración de sonido se mantiene
Aplicación	Versión final
Valoración	OK

Tabla 64. PCN-009.

Configuración	
PCN-010	Cambiar el idioma
Versión	1.0
Descripción	Un usuario solicita cambiar el idioma
Resultado esperado	Una vez aplicados los cambios, la interfaz se muestra en el idioma seleccionado
Aplicación	Versión final
Valoración	OK

Tabla 65. PCN-010.

Configuración	
PCN-011	Cambiar el estilo de las piezas
Versión	1.0
Descripción	Un usuario solicita cambiar el estilo de las piezas
Resultado esperado	El estilo se cambia correctamente
Aplicación	Versión final
Valoración	OK

Tabla 66. PCN-011.

Jugar	
PCN-012	Configurar Partida
Versión	1.0
Descripción	Un usuario solicita modificar la configuración por defecto de la partida
Resultado esperado	La partida se genera de acuerdo con la nueva configuración
Aplicación	Versión final
Valoración	Incorrecto, la configuración se modifica y guarda, pero no se aplica totalmente.

Tabla 67. PCN-012.

Jugar	
PCN-013	Colocar Pieza
Versión	1.0
Descripción	Un usuario solicita colocar una pieza concreta
Resultado esperado	Una vez ha seleccionado una pieza, el sistema permite al jugador colocarla en una de las casillas disponibles
Aplicación	Versión final
Valoración	Incorrecto, llegado el momento, se selecciona la pieza, se muestra el peón que actúa como representante de la posición, pero no se fija a la malla.

Tabla 68. PCN-013.

Jugar	
PCN-014	Moverse por el Mapa
Versión	1.0
Descripción	Un usuario se mueve alrededor del mapa con la cámara
Resultado esperado	El mapa se muestra correctamente en todo momento
Aplicación	Versión final
Valoración	OK

Tabla 69. PCN-014.

Jugar	
PCN-015	Cambiar turno
Versión	1.0
Descripción	Un usuario mueve o coloca una pieza.
Resultado esperado	El sistema reacciona a la acción cambiando el turno del jugador.
Aplicación	Versión final
Valoración	Incorrecto, dado que el ciclo de juego está paralizado en la funcionalidad de la prueba **PCN-0013**

Tabla 70. PCN-015.

Jugar	
PCN-016	Mover Pieza
Versión	1.0
Descripción	Un usuario solicita mover una pieza colocada según sus reglas de movimiento
Resultado esperado	Se muestran las casillas accesibles y tras seleccionar una, se cambia su ubicación
Aplicación	Versión final
Valoración	Incorrecto, dado que el ciclo de juego está paralizado en la funcionalidad de la prueba **PCN-0013**

Tabla 71. PCN-016.

Jugar	
PCN-017	Ver Manual desde la partida
Versión	1.0
Descripción	Un usuario solicita ver el manual desde el menú de pausa
Resultado esperado	El menú se muestra sin cerrar la partida
Aplicación	Versión final
Valoración	OK

Tabla 72. PCN-017.

Jugar	
PCN-018	Ver Configuración desde la partida
Versión	1.0
Descripción	Un usuario solicita ver la configuración del juego desde el menú de pausa
Resultado esperado	Se muestra al usuario la configuración del juego sin cerrar la partida
Aplicación	Versión final
Valoración	OK

Tabla 73. PCN-018.

Jugar	
PCN-019	Cambiar configuración activa durante la partida
Versión	1.0
Descripción	Un usuario solicita modificar y aplicar una opción de configuración
Resultado esperado	La nueva configuración se aplica en el momento
Aplicación	Versión final
Valoración	OK

Tabla 74. PCN-019.

Jugar	
PCN-020	Volver al juego
Versión	1.0
Descripción	Un usuario solicita reanudar la partida
Resultado esperado	La partida continua en el punto en el que se había quedado al pausar.
Aplicación	Versión final
Valoración	OK

Tabla 75. PCN-020.

Jugar	
PCN-021	Abandonar Partida
Versión	1.0
Descripción	Un usuario solicita salir de la partida
Resultado esperado	La partida finaliza tras pedir confirmación
Aplicación	Versión final
Valoración	OK

Tabla 76. PCN-021.

Jugar	
PCN-022	Salir al Escritorio
Versión	1.0
Descripción	Un usuario solicita salir del juego desde el menú de pausa
Resultado esperado	El juego se cierra tras pedir confirmación
Aplicación	Versión final
Valoración	OK

Tabla 77. PCN-022.

Jugar	
PCN-023	Finalizar Partida
Versión	1.0
Descripción	Un usuario lleva la partida a un estado final
Resultado esperado	Una vez una abeja reina sea rodeada, la partida acaba y se muestra la ventana de fin de partida con la información pertinente.
Aplicación	Versión final
Valoración	Incorrecto, dado que el ciclo de juego está paralizado en la funcionalidad de la prueba **PCN-0013**

Tabla 78. PCN-023.

9 Manuales

En este capítulo expondremos dos documentos de gran relevancia para el usuario final. El primero de ellos es el "Manual de Instalación", que explica paso a paso el proceso a seguir para poder instalar en nuestro sistema el videojuego. El segundo de ellos, el "Manual de Usuario", muestra todas las posibles acciones a realizar en nuestro videojuego y los pasos a seguir para realizarlas.

Habitualmente suele adjuntarse a su vez el "Manual de Administración" que describe tareas específicas del usuario administrador del sistema relativas a actualizaciones, permisos, gestión de bases de datos, conexiones y demás. En nuestro caso, por la funcionalidad y los componentes de nuestro sistema, en los que no interviene ningún usuario a parte del jugador final, consideramos innecesario dicho manual.

9.1 Manual de Instalación

El proceso de instalación es realmente trivial, ya que lo que tenemos inicialmente es el paquete del juego en el que se incluye el archivo ejecutable del mismo. Por tanto, el proceso de instalación es tan simple como:

1. Copiar la carpeta contenedora en el punto de instalación seleccionado.
2. Ejecutar el juego haciendo doble click sobre el archivo con el nombre "*TheSwarm.exe*" (En Windows).

En caso de querer crear un acceso directo al juego para evitar tener que navegar hasta el archivo de nuevo, simplemente debemos hacer click derecho sobre el archivo mencionado anteriormente, pulsar en la opción "Crear Acceso Directo" y arrastrar dicho archivo a nuestro escritorio o el punto en el que deseemos colocarlo.

9.2 Manual de Usuario

A continuación, describiremos paso a paso las distintas acciones que el jugador puede realizar en este videojuego. Todas las explicaciones están apoyadas con una serie de imágenes explicativas que facilitarán la comprensión de las funcionalidades.

9.2.1 Inicio

Tras la introducción, el juego mostrará el menú principal. Dicho menú, como puede verse a continuación, cuenta con 5 opciones; todas ellas acompañadas de mensajes aclaratorios que aportan información relativa al contenido de cada una al pasar el ratón por encima.

Ilustración 15. Menú de Inicio explicado.

9.2.2 Jugar Partida

Al acceder a la opción del menú "Jugador VS Jugador" se nos mostrará la ventana de configuración de partida tal y como se muestra en la siguiente ilustración. Dicha ventana tiene las siguientes opciones:

- **Primer Turno.** Permite elegir quien empezará la partida.
- **Color del Jugador 1.** Permite elegir el color de cada jugador.
- **Reglas de Torneo.** Permite elegir si se jugará con reglas de torneo o no.
- **Jugar con Expansiones.** Permite seleccionar si se jugará con las expansiones o solo con el set básico. Habilitar dicha opción permite acceder a las siguientes:
    - **Mariquita.** Activar en caso de que se quiera jugar con la Mariquita.
    - **Mosquito.** Activar en caso de que se quiera jugar con el Mosquito.
    - **Cochinilla.** Activar en caso de que se quiera jugar con la Cochinilla.

Ilustración 16. Configurador de Partidas.

Una ver configurada la partida, se debe pulsa el botón "Jugar" que iniciará la partida tal y como la haya configurado el usuario. En su turno, un jugador puede optar entre mover una pieza en juego bajo su control, colocar una pieza nueva o utilizar, en el caso de que ésta esté en juego, la habilidad de su Cochinilla.

Para colocar una pieza en juego, se debe pulsar sobre la casilla del HUD que representa a la ficha que se desea colocar y posteriormente pulsar en la casilla en la que se desea colocar dicha pieza de entre todas las iluminadas.

Ilustración 17. Colocar pieza.

Para mover una pieza en juego, se debe pulsar sobre la pieza deseada y posteriormente pulsar sobre la casilla a la que se desea mover la pieza de entre todas las iluminadas.

Usar la habilidad de la Cochinilla es igual que mover una pieza en juego con la salvedad de que no importa que no sea del color del usuario al que pertenece la Cochinilla.

Una vez se ha llegado a una situación final, el juego informará a los participantes de quien ha resultado vencedor y permitirá jugar una nueva partida, volviendo al configurador de partida con "Volver a Jugar" o salir al menú principal con "Salir al Menú".

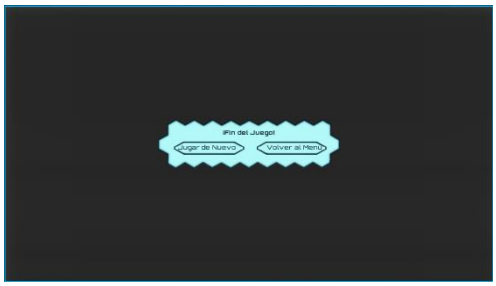


Ilustración 18. Menú de fin de partida.

9.2.2.1 Descripción del HUD.

El HUD es el canal de comunicación entre el jugador y el estado del juego. Es decir, mantiene al jugador informado del estado del juego, las acciones que puede realizar y los cambios que se producen, a la vez aportan los medios para que éste interactúe con el juego.

En este juego, el HUD está compuesto por las siguientes secciones, tal y como puede verse en la figura de apoyo:

1. **Área de Juego.** Esta sección de la pantalla es donde se mostrará la partida según se vaya desarrollando.
2. **Letrero de Turno.** Muestra a que jugador pertenece el turno actual.
3. **Botón de Menú.** Este botón permite al usuario pausar el juego y acceder al menú de pausa.
4. **Mis Piezas.** Esta zona contiene las piezas disponibles para ser colocadas e indicaciones para que el usuario sepa cuantas le quedan de cada tipo.

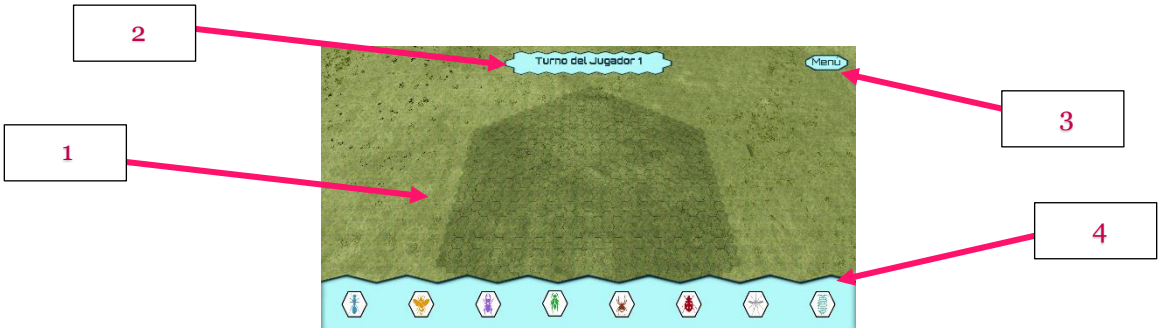


Ilustración 19. HUD.

9.2.3 Manual de Juego.

Al manual de juego se puede acceder, bien vía Menú Principal, o en partida a través del Menú de Pausa. Una vez pulsada la opción "Manual" en cualquiera de los dos menús mencionados, se mostrará el manual con las reglas de juego. Dicho manual se muestra en modo scroll, por tanto, para verlo completo se debe desplazar la vista con la ruleta central del ratón o arrastrando la barra lateral como se indica en la imagen.

Ilustración 20. Manual de Juego.

Por último, para salir del manual se debe pulsar el botón "Volver al Menú" que devolverá al usuario al menú en que se encontrase antes de ver el manual.

9.2.4 Opciones de Configuración.

Al igual que pasa con el manual de juego, a la configuración del sistema se puede acceder desde el Menú Principal o desde el Menú de Pausa, pulsando la opción correspondiente. Una vez se accede al Menú de Opciones de Configuración se puede ver que está subdividido en varias pestañas.

Para aplicar y guardar los cambios que hayamos realizado en la configuración de una o varias pestañas será necesario pulsar el botón "Aplicar Configuración" antes de salir. En caso contrario no se guardarán los cambios.

Finalmente, al igual que con el manual, para salir se debe pulsar el botón "Volver" que devolverá al usuario al menú en el que se encontrase antes de acceder a la configuración.

A continuación, se describen en detalle las opciones relativas a cada pestaña de la configuración.

9.2.4.1 Opciones de Video

En esta pestaña se pueden configurar las opciones relativas al aspecto visual del juego. Las opciones disponibles para graduar a gusto del usuario son las siguientes:

- **Escala de Resolución.** Densidad de pixeles por pulgada respecto a la imagen original.
- **Campo de Visión.** Distancia de visión.
- **Anti-Aliasing.** Calidad del suavizado de bordes.
- **Post-Procesado.**
- **Sombras.** Calidad del sombreado.
- **Texturas.** Calidad de las texturas de las superficies.
- **Efectos.** Calidad de los efectos y el sistema de partículas.

Ilustración 21. Pestaña de configuración gráfica.

Una vez realizado algún cambio, en caso de que se desee mantener dicho cambio, se deberá pulsar el botón "Aplicar Configuración" antes de salir del Menú de Configuración. En caso contrario, no se guardarán los cambios.

9.2.4.2 Opciones de Audio

En esta pestaña se pueden configurar opciones relativas al sonido del juego. Las opciones disponibles para modificar a gusto del usuario son las siguientes:

- **Volumen de la Música.** Volumen al que se reproduce la banda sonora del juego.
- **Volumen de Efectos.** Volumen al que se reproducen los efectos y sonidos ambientales.

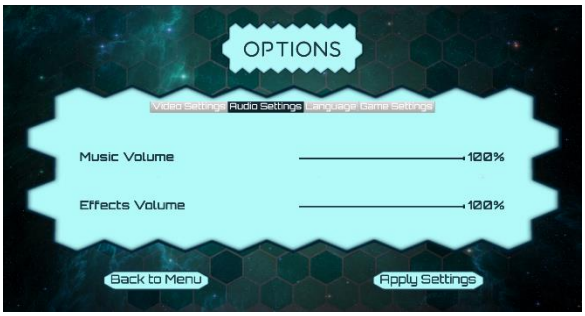


Ilustración 22. Pestaña de configuración de sonido.

Al realizar algún cambio se deberá pulsar el botón "Aplicar Configuración" antes de salir del Menú de Configuración en caso de que deseemos mantenerlo.

9.2.4.3 Idioma

Esta es la pestaña de selección de idioma del juego.

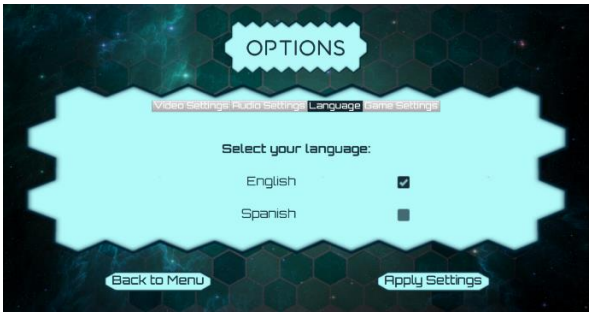


Ilustración 23. Pestaña de selección de idioma.

Para seleccionar el idioma en el que se desee que se muestre el juego se debe activar dicha opción y guardar los cambios pulsando "Aplicar Configuración".

9.2.4.4 Estilo de las Piezas

Esta es la pestaña de selección de estilo de las piezas del Juego. El juego de mesa original del que se ha partido para realizar este juego, Hive: La Colmena, cuenta con dos versiones. La versión "Original" a color y la versión "Carbón" en blanco y negro.

Ilustración 24. Pestaña de selección de estilo de las piezas.

Para seleccionar la opción con la que se desee jugar se debe activar dicha opción y guardar los cambios pulsando "Aplicar Configuración".

9.2.5 Créditos

Para acceder a los créditos y agradecimientos del juego se debe pulsar la opción “Créditos” del Menú Principal. Esto hará que se muestre la página de créditos, que al igual que el manual, se muestra en modo scroll.

Ilustración 25. Ventana de Créditos.

Para desplazarse por los créditos se debe utilizar la ruleta central del ratón o la barra lateral de desplazamiento. Una vez vistos los créditos, salir pulsando “Volver”.

9.3 Atajos de teclado

Para la gestión de la cámara se han desarrollado una serie de atajos de teclado que facilitan su funcionamiento.

9.3.1 Desplazamiento

El desplazamiento puede realizarse bien mediante el patrón de teclas convencionales WASD o con el ratón gracias a la detección de la proximidad de los bordes. Si pulsamos la tecla Mayus mientras nos desplazamos, aumentaremos la velocidad de desplazamiento.

9.3.2 Rotación e Inclinación

La rotación se gestiona pulsando el botón central del ratón a la vez que lo movemos. En caso de querer recuperar la posición inicial, debemos pulsar Alt + Botón central del ratón.

9.3.3 Zoom

El zoom se gestiona a partir de la ruleta central del ratón. Desplazar la ruleta hacia arriba aplica zoom a la visualización y desplazarla de nuevo hacia abajo lo reduce. Para devolver la cámara al zoom por defecto pulsar Ctrl + Botón central del ratón.

10 Conclusiones

Para finalizar este Trabajo de Fin de Grado quisiéramos exponer las conclusiones extraídas de la realización del mismo. Esta valoración se hace de acuerdo a los objetivos conseguidos y el proceso realizado.

Por último, describiremos una serie de futuras líneas de desarrollo que podrían seguirse a la hora de ampliar el proyecto realizado.

10.1 Conclusiones generales.

Al comenzar este proyecto nos planteamos como objetivos, no solo aprender las metodologías utilizadas por los profesionales de la industria de los videojuegos a la hora de crear un nuevo producto, sino aplicar los conocimientos adquiridos durante nuestros estudios universitarios de manera que dicho proceso de aprendizaje y el desarrollo del proyecto objetivo se hiciesen siguiendo una metodología adecuada a los principios de desarrollo de la ingeniería de software.

Teniendo esto en cuenta, hemos desarrollado y seguido una planificación para el proyecto subdividida en fases y tareas con un propósito, orden y entregables bien definidos. Hemos analizado, diseñado y replicado las características de un juego de mesa premiado por su complejidad, usando en el proceso las herramientas estándares del sector. Hemos creado nuestra propia biblioteca de funcionalidades relacionadas con un tablero de piezas hexagonales, de forma que podamos integrar su funcionalidad fácilmente en futuros proyectos. Cada pieza y material del juego han sido modelados a imagen y semejanza de los componentes del juego original de forma que el aspecto y la sensación del usuario final sean óptimas. Todo ello se ha realizado persiguiendo el objetivo de que nuestro juego fuese capaz de captar el mayor publico posible y por ello mismo se efectuó el proceso de localización al español y el inglés y la producción multiplataforma del juego, de tal forma que sea ejecutable en cualquier ordenador indistintamente del sistema operativo del mismo.

Aun así, lo realmente relevante es que la realización de este proyecto ha brindado a los miembros del equipo de desarrollo la oportunidad de aplicar conocimientos y técnicas adquiridos durante toda la titulación en un único objetivo. De esta forma, el equipo se ha visto forzado a utilizar el conjunto de conocimientos de distintas áreas acumulado a lo largo del tiempo, lo que ha redundado en un proceso de afianzamiento e integración de dichas competencias de orígenes heterogéneos.

El hecho de que este proyecto sea, por su procedimiento, individual y por su objetivo, multidisciplinar, ha obligado a que los miembros del equipo ejerzan el papel de hombre renacentista y no solo apliquen y desarrollen sus habilidades relacionadas con la profesión informática, sino que también han tenido que adquirir y aplicar habilidades relacionadas con áreas de conocimiento como el arte digital, la edición fotográfica o el modelado 3D. Además, el hecho de que toda la responsabilidad recaiga sobre una única persona ha redundado en el aprendizaje de habilidades profesionales de carácter no técnico, como son la toma de decisiones, el control del estrés, la asunción de responsabilidades, la gestión del tiempo y otros recursos o el desarrollo del autoaprendizaje.

Por otra parte, realizar este proyecto nos ha permitido comprender, valorar y saber medir el trabajo que conlleva un producto software de una envergadura tan grande como es un videojuego.

Por último, quisiéramos destacar que el haber tenido la oportunidad de desarrollar este proyecto no solo nos ha permitido realizar una pequeña incursión al área profesional en el que nos gustaría desarrollarnos, sino contactar, aprender, compartir y conocer a otros profesionales y futuros profesionales del sector de tal forma que hemos podido iniciar nuestra red de contactos laborales.

10.2 Futuras líneas de desarrollo.

Debido al potencial de la idea, desde un primer momento tuvimos en cuenta las posibles líneas de desarrollo a seguir una vez acabado el proyecto. Pese a ello, el proceso de creación de un videojuego es muy denso por la cantidad de factores distintos que intervienen y teniendo en cuenta los recursos limitados a nuestra disposición decidimos centrarnos en el objetivo principal de replicar una experiencia de juego fiel al juego de mesa original. Aun así, a continuación, exponemos una serie de líneas de desarrollo contempladas por el valor añadido que aportarían al producto final:

- La primera y principal línea de desarrollo a futuro es la realización de nuevos modos de juego, tanto en solitario como en compañía, que permitan al jugador expandir su experiencia de juego. Para ello las opciones van desde la aplicación de algoritmos de teoría de juegos a la creación de modos en solitario; hasta la modificación del modo competitivo para su funcionamiento online vía servidor.

- Otra línea de trabajo interesante sería la creación de un registro de usuarios y un ranking de partidas global que fomentase la competencia entre jugadores de distintos niveles de experiencia. Esto no solo abriría las puertas a la creación de competiciones basadas en el juego, sino que facilitaría la creación de una comunidad de jugadores habituales en torno a nuestro juego.

Estas son futuras líneas de desarrollo que aportarían un mayor nivel de calidad al producto final y extenderían su vida útil. Algunas de ellas se contemplaron cuando el proyecto solo era un germen de lo que es, pero finalmente fueron relegadas a futuras fases de desarrollo en pos de realizar un trabajo que cumpliese con los plazos y estándares de calidad asumidos.

Bibliografía

A continuación, pasamos a enumerar todas las fuentes de información utilizadas en el desarrollo de este proyecto. Dichas fuentes se listan siguiendo los estándares normativos de la sexta edición del formato *APA* [86].

Índice Bibliográfico

[1] González, D. (2011). Diseño de videojuegos da forma a tus sueños. Paracuellos de Jarama, Madrid: Ra-Ma.

[2] González, D. (2014). Arte de videojuegos: da forma a tus sueños. Paracuellos de Jarama, Madrid: Ra-Ma.

[3] Sherif, W. (2015). Learning C++ by creating games with UE4: learn C++ programming with a fun, real-world application that allows you to create your own games. Birmingham, UK: Packt Publishing.

[4] Sherif, W. & Whittle, S. (2016). Unreal Engine 4 scripting with C++ cookbook: get the best out of your games by scripting them using UE4. Birmingham: Packt.

[5] Misra, N. (2015). Learning Unreal Engine Android game development: tap into the power of Unreal Engine 4 and create exciting games for the Android platform. Birmingham, UK: Packt Publishing.

[6] Carnall, B. (2016). Unreal Engine 4.X by example: an example-based practical guide to get you up and running with Unreal Engine 4.X. Birmingham, UK: Packt Publishing.

[7] Nystrom, R. (2014). Game programming patterns. United States: Genever Benning.

[8] Satheesh, P. (2016). Unreal Engine 4 game development essentials: master the basics of Unreal Engine 4 to build stunning video games. Birmingham: Packt Publishing.

[9] Moniem, M. (2016). Mastering Unreal Engine 4.X: take your game development skills to the next level with one of the best engines on the market. Birmingham, UK: Packt Publishing.

[10] Rodríguez, R. (2016). Videojuegos: la explosión digital que esta´ cambiando el mundo. Sevilla: Héroes de Papel.

[11] Ordoñez, J. (2013). Power-Ups: ¡conviértete en un profesional de los videojuegos! Palma de Mallorca: Plan B.

[12] Harris, B. (2017). Console Wars: Sega, Nintendo y la batalla que definió una generación. Sevilla: Héroes de Papel.

[13] Rodríguez, U. (2016). La Historia de Nintendo más de 125 años de entretenimiento. Palma de Mallorca: Dolmen.

[14] Redondo, I. (2014). Que´ es un videojuego?: claves para entender el mayor fenómeno cultural del siglo XXI. Sevilla: Arcade.

[15] Teixes, F. (2015). Gamificación: motivar jugando. Barcelona: UOC.

[16] Parren□o, J. (2010). Marketing y videojuegos: product placement, in-game advertisigin [sic] y advergaming. Madrid: ESIC.

[17] Stevens, P. & Pooley, R. (2007). Utilización de UML en ingeniería del software con objetos y componentes. Madrid: Pearson Addison Wesley.

[18] Videojuego. (2017). Es.wikipedia.org. Ultimo acceso 10 de abril de 2017, desde https://es.wikipedia.org/wiki/Videojuego

[19] Industria de los videojuegos. (2017). Es.wikipedia.org. Ultimo acceso 4 de abril de 2017, desde https://es.wikipedia.org/wiki/Industria_de_los_videojuegos

[20] Desarrollo de videojuegos. (2017). Es.wikipedia.org. Ultimo acceso 4 de abril de 2017, desde https://es.wikipedia.org/wiki/Desarrollo_de_videojuegos

[21] Desarrollo de videojuego independiente. (2017). Es.wikipedia.org. Ultimo acceso 17 de abril de 2017, desde https://es.wikipedia.org/wiki/Desarrollo_de_videojuegos_independiente

[22] Videojuego independiente. (2017). Es.wikipedia.org. Ultimo acceso 17 de abril de 2017, desde https://es.wikipedia.org/wiki/Videojuego_independiente

[23] Historia de los videojuegos. (2017). Es.wikipedia.org. Ultimo acceso 16 de abril de 2017, desde https://es.wikipedia.org/wiki/Historia_de_los_videojuegos

[24] Magnabox Odyssey. (2017). Es.wikipedia.org. Ultimo acceso 11 de abril de 2017, desde https://es.wikipedia.org/wiki/Magnavox_Odyssey

[25] Primer videojuego. (2017). Es.wikipedia.org. Ultimo acceso 10 de abril de 2017, desde https://es.wikipedia.org/wiki/Primer_videojuego

[26] Nintendo. (2017). Es.wikipedia.org. Ultimo acceso 16 de abril de 2017, desde https://es.wikipedia.org/wiki/Nintendo

[27] Atari. (2017). Es.wikipedia.org. Ultimo acceso 16 de abril de 2017, desde https://es.wikipedia.org/wiki/Atari

[28] Aventura gráfica. (2017). Es.wikipedia.org. Ultimo acceso 15 de abril de 2017, desde https://es.wikipedia.org/wiki/Aventura_gr%C3%A1fica

[29] LucasArts. (2017). Es.wikipedia.org. Ultimo acceso 15 de abril de 2017, desde https://es.wikipedia.org/wiki/LucasArts

[30] SCUMM. (2017). Es.wikipedia.org. Ultimo acceso 15 de abril de 2017, desde https://es.wikipedia.org/wiki/SCUMM

[31] Pyro Studios. (2017). Es.wikipedia.org. Ultimo acceso 15 de abril de 2017, desde https://es.wikipedia.org/wiki/Pyro_Studios

[32] Pendulo Studios, (2017). Es.wikipedia.org. Ultimo acceso 15 de abril de 2017, desde https://es.wikipedia.org/wiki/Pendulo_Studios,_S.L.

[33] PlayStation. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/PlayStation

[34] PlayStation 2. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/PlayStation_2

[35] PlayStation 3. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/PlayStation_3

[36] PlayStation 4. (2017). Es.wikipedia.org. Ultimo acceso 17 de abril de 2017, desde https://es.wikipedia.org/wiki/PlayStation_4

[37] Nintendo Entertainment System. (2017). Es.wikipedia.org. Ultimo acceso 10 de abril de 2017, desde https://es.wikipedia.org/wiki/Nintendo_Entertainment_System

[38] Super Nintendo. (2017). Es.wikipedia.org. Ultimo acceso 10 de abril de 2017, desde https://es.wikipedia.org/wiki/Super_Nintendo

[39] Nintendo DS. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/Nintendo_DS

[40] Wii. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/Wii

[41] Nintendo Switch. (2017). Es.wikipedia.org. Ultimo acceso 7 de junio de 2017, desde https://es.wikipedia.org/wiki/Nintendo_Switch

[42] Game Boy. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/Game_Boy

[43] Nintendo 64. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/Nintendo_64

[44] Mega Drive. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/Mega_Drive

[45] Sega Mega-CD. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/Sega_Mega-CD

[46] Sega Saturn. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/Mega_Drive

[47] Xbox. (2017). Es.wikipedia.org. Ultimo acceso 12 de abril de 2017, desde https://es.wikipedia.org/wiki/Xbox

[48] Xbox 360. (2017). Es.wikipedia.org. Ultimo acceso 14 de abril de 2017, desde https://es.wikipedia.org/wiki/Xbox_360

[49] Xbox One. (2017). Es.wikipedia.org. Ultimo acceso 17 de abril de 2017, desde https://es.wikipedia.org/wiki/Xbox_One

[50] Realidad Virtual. (2017). Es.wikipedia.org. Ultimo acceso 17 de abril de 2017, desde https://es.wikipedia.org/wiki/Realidad_virtual

[51] PlayStation VR. (2017). Es.wikipedia.org. Ultimo acceso 20 de abril de 2017, desde https://es.wikipedia.org/wiki/PlayStation_VR

[52] Oculus Rift. (2017). Es.wikipedia.org. Ultimo acceso 20 de abril de 2017, desde https://es.wikipedia.org/wiki/Oculus_Rift

[53] El Futuro del Videojuego. Informe de resultados. (2010). Ultimo acceso 20 de abril de 2017, desde http://www.aevi.org.es/web/wp-content/uploads/2015/12/fcuantitativa.pdf

[54] El videojuego en España. (2017). Asociación Española de Videojuegos. Ultimo acceso 2 de mayo de 2017, desde http://www.aevi.org.es/la-industria-del-videojuego/en-espana/

[55] Game Engine Technology by Unreal. (2017). Unrealengine.com. Ultimo acceso 2 de mayo de 2017, desde https://www.unrealengine.com/en-US/blog

[56] Unreal Engine. (2017). Es.wikipedia.org. Ultimo acceso 2 de mayo de 2017, desde https://es.wikipedia.org/wiki/Unreal_Engine

[57] Unity (motor de juego). (2017). Es.wikipedia.org. Ultimo acceso 2 de mayo de 2017, desde https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[58] Unity - Game Engine. (2017). Unity. Ultimo acceso 2 de mayo de 2017, desde https://unity3d.com/es

[59] GameMaker: Studio. (2017). Es.wikipedia.org. Ultimo acceso 2 de mayo de 2017, desde https://es.wikipedia.org/wiki/GameMaker:_Studio

[60] GameMaker | YoYo Games. (2017). Yoyo Games. Ultimo acceso 2 de mayo de 2017, desde https://www.yoyogames.com/gamemaker

[61] CRYENGINE | The complete solution for next generation game development by Crytek. (2017). Cryengine.com. Ultimo acceso 2 de mayo de 2017, desde https://www.cryengine.com/

[62] Bienvenido a Visual Studio 2015. (2017). Msdn.microsoft.com. Ultimo acceso 15 de mayo de 2017, desde https://msdn.microsoft.com/es-es/library/dd831853.aspx?f=255&MSPPError=-2147217396

[63] Visual Assist - a Visual Studio extension by Whole Tomato Software. (2017). Wholetomato.com. Ultimo acceso 15 de mayo de 2017, desde https://www.wholetomato.com/

[64] Gameplay Framework. (2017). Docs.unrealengine.com. Ultimo acceso 15 de mayo de 2017, desde https://docs.unrealengine.com/latest/INT/Gameplay/Framework/

[65] Blueprints Visual Scripting. (2017). Docs.unrealengine.com. Ultimo acceso 15 de mayo de 2017, desde https://docs.unrealengine.com/latest/INT/Engine/Blueprints/

[66] Material Editor Reference. (2017). Docs.unrealengine.com. Ultimo acceso 15 de mayo de 2017, desde https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/Editor/

[67] Comprar Maya | Software de animación por ordenador | Autodesk Tienda Online. (2017). Autodesk.es. Ultimo acceso 15 de mayo de 2017, desde https://www.autodesk.es/store/products/maya?term=1year&support=advanced

[68] Brewster, R. (2017). Paint.NET - Free Software for Digital Photo Editing. Getpaint.net. Ultimo acceso 15 de mayo de 2017, desde https://www.getpaint.net/

[69] Balsamiq Mockups | Balsamiq. (2017). Balsamiq.com. Ultimo acceso 20 de mayo de 2017, desde https://balsamiq.com/products/mockups/

[70] Capturing Architectural Requirements. (2017). Ibm.com. Ultimo acceso 27 de mayo de 2017, desde https://www.ibm.com/developerworks/rational/library/4706.html#N100A7

[71] Proceso Unificado Racional. (2017). Es.wikipedia.org. Ultimo acceso 25 de junio de 2017, desde https://es.wikipedia.org/wiki/Proceso_Unificado_Racional

[72] Catto, E. (2017). Box2D | A 2D Physics Engine for Games. Box2d.org. Ultimo acceso 17 de mayo de 2017, desde http://box2d.org/

[73] Enlighten | Enlighten | Real Time Global Illumination Solution. (2017). Enlighten | Real Time Global Illumination Solution. Ultimo acceso 15 de mayo de 2017, desde http://www.geomerics.com/enlighten/

[74] Hexagonal Grids. (2017). Redblobgames.com. Ultimo acceso 27 de junio de 2017, desde http://www.redblobgames.com/grids/hexagons/

[75] Spine: Software de animación en 2D para juegos de video. (2017). Es.esotericsoftware.com. Ultimo acceso 21 de mayo de 2017, desde http://es.esotericsoftware.com/

[76] Welcome To UML Web Site! (2017). Uml.org. Ultimo acceso 21 de junio de 2017, desde http://www.uml.org/

[77] Umbra – Any 3D content, any device. (2017). Umbra. Ultimo acceso 19 de mayo de 2017, desde http://umbra3d.com/

[78] GanttProject: free desktop project management app. (2017). Ganttproject.biz. Recuperado 2 July 2017, a partir de http://www.ganttproject.biz/

[79] Historia de los videojuegos. (2017). Fib.upc.edu. Recuperado 2 July 2017, a partir de http://www.fib.upc.edu/retro-informatica/historia/videojocs.html

[80] Virtual Reality - Virtual Reality. (2017). Virtual Reality. Recuperado 2 July 2017, a partir de https://www.vrs.org.uk/

[81] Antoniou, T. (2017). Hive. Gen42.com. Recuperado 2 July 2017, a partir de http://www.gen42.com/hive

[82] Kelly, J. & Kirkwood, R. & Fry, K. (2007). C++ Design Document. Team Kilo Kilo Foxtrot, desde https://es.scribd.com/doc/2056160/KiloKiloFoxtrot-FINAL#

[83] Viral Light Interactive. (2008). Game Design Document for the Action Adventure RPG Game – DeathWish 1931. Viral Light Interactive, desde https://es.scribd.com/doc/34994424/Viral-Light-Interactive-GDD

[84] González Sánchez, J. L. & Padilla Zea, N. & Gutiérrez, F. L. & Cabrera, M. J. (2009). De la Usabilidad a la Jugabilidad: Diseño de Videojuegos Centrado en el Jugador. Granada: Universidad de Granada.

[85] AEVI, Asociación Española de Videojuegos. (2017). 2016 anuario. Anuario de la industria del videojuego. España: Asociación Española de Videojuegos, desde http://www.aevi.org.es/web/wp-content/uploads/2017/06/ANUARIO_AEVI_2016.pdf

[86] Save Time and Improve your Marks with CiteThisForMe, The No. 1 Citation Tool. (2017). Cite This For Me. Ultimo acceso 2 de julio de 2017, desde http://www.citethisforme.com/

[87] Hive on Steam. (2017). Store.steampowered.com. Ultimo acceso 2 de julio de 2017, desde http://store.steampowered.com/app/251210/

[88] Hive for Android. (2017). Play.google.com. Ultimo acceso 2 de julio de 2017, desde https://play.google.com/store/apps/details?id=com.hive

[89] Hive at Boardspace.net. (2017). Boardspace.net. Ultimo acceso 2 de julio de 2017, desde http://www.boardspace.net/english/about_hive.html

[90] Hearthstone - Página oficial. (2017). Hearthstone. Ultimo acceso 2 de julio de 2017, desde https://eu.battle.net/hearthstone/es/

[91] Tabletop Simulator en Steam. (2017). Store.steampowered.com. Ultimo acceso 2 de julio de 2017, desde http://store.steampowered.com/app/286160/Tabletop_Simulator/

[92] Sid Meier's Civilization® VI en Steam. Store.steampowered.com. Ultimo acceso 2 de julio de 2017, desde http://store.steampowered.com/app/289070/Sid_Meiers_Civilization_VI/?l=spanish

Anexo

A continuación, detallaremos brevemente los contenidos adjuntados en el DVD que acompaña a esta memoria. Dichos contenidos complementan a la memoria aportando así una visión total del trabajo realizado y el esfuerzo desempeñado en la consecución del proyecto.

Contenidos

El listado de los contenidos que se pueden encontrar en el DVD es el siguiente:

- "***memoria.pdf***". Contiene el documento de la memoria del TFG en formato pdf.
- "***ManualDeInstalacion.pdf***". El manual con las directrices para la instalación del ejecutable del programa.
- "***ManualDeUsuario.pdf***". El manual con las acciones realizables por parte del usuario.
- "**TheSwarmCode.rar**". Carpeta comprimida con el código fuente del programa.
- "***TheSwarmEj.rar***". Carpeta comprimida con el ejecutable del programa.
- "***README.txt***". Archivo de texto plano con este mismo índice de contenidos.