Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Tecnologías de la Información

Desarrollo de un juego 2D para Android usando Unity

Autor:

D. Daniel González Pérez

Tutores:

Dña. Margarita Gonzalo Tasis
D. Joaquín Adiego Rodríguez

Agradecimientos

Quisiera agradecer a mis padres, por aguantarme a lo largo de todos estos años de tristezas y alegrías.

También agradezco a los compañeros que me han acompañado a lo largo de la carrera, haciendo de ésta una experiencia muy valiosa, especialmente a mis buenos compañeros Miguel, Álvaro y Borja.

Resumen

El mercado de los videojuegos es uno de los sectores laborales relacionados con la ingeniería informática que más crece actualmente. Según el Anuario de 2018 de la AEVI (Agencia Española de Videojuegos) [63] en 2018 el consumo de videojuegos facturó 1.530 millones de euros, un dato récord en nuestro país con un 12,6% más que el año anterior.

En base a esta premisa, se presenta este TFG como un primer paso para un posible futuro laboral al sector, haciendo uso del motor de videojuegos de moda para pequeñas empresas, Unity3D.

Abstract

The videogame market is one of the labour sectors related to computer engineering that is growing the most at present. According to AEVI's 2018 Yearbook (Agencia Española de Videojuegos) [63] in 2018 the consumption of video games billed 1,530 million euros, a record in our country with 12.6% more than the previous year.

Based on this, this TFG is presented as a first step for a possible future job in the sector, making use of the fashion video game engine for small businesses, Unity3D.

Índice

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Introducción

En este capítulo expondremos las motivaciones que nos han llevado a la realización de este proyecto, plantearemos los objetivos que trataremos de alcanzar con nuestro trabajo y listaremos los capítulos contenidos en esta memoria, junto con una breve descripción, para ayudar a situarse en la misma.

1.1 Motivaciones

La necesidad por entretenernos siempre ha sido parte de la psicología humana. Desde nuestros orígenes, siempre hemos buscado nuevas maneras de divertirnos, y, conforme avanzaba la tecnología, conseguíamos nuevas maneras de conseguirlo. Una vez llegada la era digital, se crearon los primeros videojuegos. Desde entonces, la industria de los videojuegos ha crecido con el paso del tiempo, siendo hoy en día un sector que mueve grandes cantidades de dinero.

Dentro de este sector, es de vital importancia el rol de los desarrolladores de software. Por ello nos hemos planteado el cómo aplicaríamos los conocimientos adquiridos a lo largo de la carrera para la creación de un videojuego.

Debido a nuestros conocimientos en el desarrollo de software, y al potencial que poseen los videojuegos como producto, hemos decidido desarrollar un videojuego propio, como un posible primer paso en la industria.

1.2 Objetivos

Con la finalidad de definir las ideas que queremos tratar en nuestro proyecto, nos hemos planteado una serie de objetivos a perseguir. Estos objetivos nos ayudarán a planificar y subdividir en tareas todo el proceso y a decidir si hemos alcanzado a realizar con éxito las ideas que nos hemos propuesto tratar.

• Conocer, comprender y aprender a utilizar las herramientas que nos brinda el motor de desarrollo de videojuegos Unity3D.

• Adaptar los conocimientos de ingeniería de software aprendidas a lo largo de la carrera al proceso de desarrollo de un videojuego.

• Analizar productos similares en busca de factores comunes, puntos que potenciar y otros que se deberían evitar.

• Desarrollar un videojuego de plataformas 2D roguelike que evite la sensación de ser repetitivo.

1.3 Organización

En este apartado realizaremos un breve resumen de los capítulos presentados en la memoria.

1. **Introducción**. se presentan las motivaciones que han incitado a la realización de este proyecto y los objetivos a cumplir con nuestro proyecto.
2. **Contexto**. Se intenta familiarizarse con los conceptos básicos pertinentes a nuestro videojuego. Para ello se presenta la historia de los videojuegos de plataformas, género al que pertenece nuestro juego.
3. **Planificación**. Se plantea el proceso de desarrollo, así como una planificación inicial a seguir, incluyendo roles de los participantes y costes de producción, acabando con el seguimiento del proyecto.
4. **Análisis**. Se detallan los requisitos funcionales y no funcionales de nuestro proyecto, los casos de uso que se puedan dar y el modelo de dominio pertinente.
5. **Diseño**. Se detalla la arquitectura lógica y los patrones que se han aplicado a nuestro proyecto.
6. **Implementación**. Se expone el proceso de implementación seguido para conseguir que nuestros niveles se generen de manera aleatoria en cada partida.
7. **Conclusiones**. Valoración personal del proyecto y posibles futuras aplicaciones.

Capítulo 2. Contexto

En este capítulo, expondremos algo de contexto sobre videojuegos, con énfasis en el género de las plataformas, con el fin de ayudar al lector a familiarizarse con el campo con el que vamos a trabajar.

2.1 Definición de videojuego

Los juegos han sido parte de nuestra sociedad desde hace miles de años. Dichos juegos han ido evolucionando a lo largo de los años, y, como era de esperar, a medidados del siglo 20 se comenzaron a utilizar las nuevas tecnologías para la producción de un nuevo tipo de juego, el videojuego.

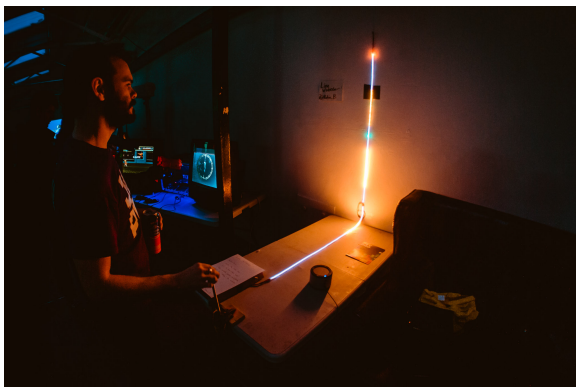

Ilustración 1: Juego Line Wobbler [55]

La definición más simple de videojuego es un “Juego electrónico que consta de un display e interacción con el usuario.” Aunque esta descripción parezca sencilla, esto abarca un amplio abanico de posibilidades, desde algo tan simple como *“Line Wobbler”* [55], un juego que consta de una tira de LEDs y un Joystick, hasta las más modernas consolas de videojuegos.

Hoy en día, la creación de un videojuego conlleva un gran trabajo conjunto de múltiples disciplinas, de entre las cuáles tiene un rol esencial la ingeniería informática, tanto para resolver simples problemas como puede ser tocar un botón, como para problemas de gran complejidad, como pueden ser la interacción entre múltiples elementos del sistema.

2.2 El género de las plataformas

El género de las plataformas nace en 1980 en las máquinas arcade, siendo los primeros juegos en explorar el género *“Space Panic”* [64] y *“Donkey Kong”* [58]. Estos juegon fueron los primeros en implementar gravedad y saltos para moverse a través de plataformas que flotan a través de la pantalla, lo cuál les diferenciaba de los otros juegos de su tiempo.

Debido a la poca capacidad de las consolas de la época, estos juegos no eran capaces de explotar el potencial contenido en este género, por lo que los juegos de plataformas no destacaron más allá de su boom inicial.

Ilustración 2: Donkey Kong [58]

Sin embargo, en 1985 el juego *“Super Mario Bros”* [65], acompañado de la consola NES [66], ocasionó la grán explosión del género de las plataformas. Super Mario Bros introdujo una serie de

elementos que se asentarían en el género: la recolección de ítems para conseguir vidas extra (en este caso, monedas), y la división en subniveles de los mundos.

En la primera mitad de los 90, el género se estandarizó por completo, entrando en el mundo de las consolas portátiles, donde se popularizó aún más, con la creación de juegos como *"Kirby's Dream Land"* [67] y *"Kid Dracula"* [68].

Ilustración 3: Kirby's Dream Land [79]

Con el avance de la tecnología, aparecieron nuevas consolas, capaces de generar gráficos 3D, y los juegos de plataforma pasarían a adaptarse a la tercera dimensión. Los primeros juegos mantenían la esencia lineal de los juegos de plataformas de toda la vida pero presentando gráficos tridimensionales. Este tipo de juegos se consideran "2.5D".

Ilustración 4: Super Mario 64 [80]

El primer videojuego de plataformas en conseguir escenarios completamente tridimensionales apareció en 1996, *"Super Mario 64"* [69], en el cual la libertad y la cantidad de movimientos en el escenario por fin consiguieron trasladar la jugabilidad, característica esencial de los juegos basados en plataformas, a la tercera dimensión.

Durante la primera década de los años 2000, el género reiteraría sobre la formula del Super Mario 64, sin recibir cambios notorios más allá de temática y visuales.

Sin embargo, a comienzos de la década de 2010, se produjo un auge en los juegos de plataformas de avance 2D, llegando a convertir a los plataformas 3D en un subgénero minoritario. Ésto es debido a que los juegos de plataformas 2D contaban con costes de desarrollo mucho menores, y permitían ofrecer un apartado artístico más elaborado. Ejemplos de juegos que volvieron a las 2 dimensiones son *"Rayman Origins"* [70] y *"Sonic Generations"* [71].

También fue notoria la popularidad de los juegos desarrollados por empresas independientes, que crearon videojuegos de plataformas 2D completamente nuevos como Super Meat Boy y Fez.

2.3 Los niveles preestablecidos

Los videojuegos suelen hacer uso de una serie de niveles. En cada uno de estos niveles, el desarrollador encapsula una serie de obstáculos a completar.

Ilustración 5: Niveles en el mundo 1 de New Super Mario Bros [81]

Un famoso ejemplo de ésto es la serie de juegos basados en plataformas "Super Mario". Este conjunto de juegos están divididos en mundos, que agrupan una serie de niveles. Cada mundo utiliza una temática diferente, ayudando así a darle variedad al juego, de manera que parezca que nuestro personaje, Mario, esté realizando una aventura por diferentes lugares exóticos.

Los niveles son preestablecidos, se diseñan una vez, y ese diseño permanece predeterminado para siempre. Ésto conlleva un problema, una vez el jugador ha completado dicho nivel, no necesita volverlo a completar. Así, cuando el usuario ha completado todos los niveles de un juego, dicho juego pierde su valor.

Podríamos argumentar que el jugador podría volver a completar un nivel previamente completado, pero ésto no es más que repetir algo ya hecho previamente, no tiene ningún valor añadido. Es sólo cuestión de tiempo que el valor del juego desaparezca por completo para el usuario.

Debido a la fama del Super Mario, muchos otros desarrolladores empezaron a crear juegos de este tipo. Ésto junto con el hecho de que al completar los niveles de dicho juego no le era de utilidad al usuario, creó un declive en el género, debido a que ya no le resultaba de interés al consumidor.

Sin embargo, a lo largo de los años, los desarrolladores de videojuegos han ideado sistemas que evitan dichos problemas, haciendo que el género de las plataformas siga manteniéndose como uno de los más populares actualmente.

"Super Meat Boy" [59] cuenta con una serie de niveles en los cuáles se le pide realizar al jugador movimientos muy precisos para superar los obstáculos. Éste juego sigue utilizando la estructura por niveles, pero cada nivel cuenta con un tiempo a superar y objetos que coleccionar. Si bien el jugador puede acabar el juego sin realizar dichas tareas, si el jugador busca una experiencia más intensa o extensa, siempre puede intentar conseguir tiempos menores en sus niveles.

Ilustración 6: Videojuego Super Meat Boy [82]

En *"Super Mario Maker"* [60], los usuarios pueden crear sus propios niveles y compartirlos con gente mediante Internet. Esto hace que la cantidad de contenido que el jugador puede experimentar es virtualmente infinito.

Ilustración 7: Videojuego Super Mario Maker [83]

Estos y otros elementos han conseguido que los videojuegos de plataformas siga siendo popular, pero el elemento que ha causado un mayor impacto en el mercado ha sido el uso de la aleatoriedad.

2.4 La generación aleatoria y los roguelike

Debido a la naturaleza de los ordenadores, no podemos generar números aleatorios mágicamente. Para ello se usan generadores de números pseudoaleatorios. Dichos generadores utilizan un algoritmo para generar números aparentemente aleatorios en base a un valor de entrada (Llamado semilla). Dado un mismo valor de entrada, el generador siempre generará la misma serie de resultados. Para evitar repetir los resultados, se utiliza un número de semilla diferente cada vez que se ejecuta el programa. Una forma de obtener ese número automáticamente siendo distinto cada vez, es utilizar el reloj del sistema.

La aleatoriedad ha tenido un papel muy importante en los juegos a lo largo de los años. De una manera u otra, en los juegos siempre podemos encontrar cosas que se generan aleatoriamente. Así, cada vez que el usuario juegue al juego, experimentará una experiencia distinta, en mayor o menor medida.

Dicha aleatoriedad se puede aplicar de multitud de maneras. El videojuego *"No Man's Sky"* [61], presenta al jugador con un universo repleto de planetas. Cada planeta genera su terreno y fauna de forma aleatoria (Llamado generación procedimental), de manera que, por muchos planetas que el jugador explore, cada uno será una nueva experiencia.

Ilustración 8: Diferentes planetas generados mediante aleatoriedad en No Man's Sky[84]

Pero también se podría aplicar aleatoriedad sobre otros elementos, que no sean parte del escenario. Un ejemplo muy usado de esta mecánica, son los objetos obtenidos al completar un objetivo, que pueden variar de cantidad y calidad.

En el videojuego *"Diablo III"* [62], el jugador recibe una cantidad de objetos aleatoria al completar un nivel, y dichos objetos a su vez son de una calidad aleatoria. A mayores de esto, los niveles son generados aleatoriamente al entrar a éstos, dentro de unas restricciones, de manera que por mucho que el usuario juegue un mismo nivel, la experiencia siempre será diferente.

Ilustración 9: Recompensas obtenidas aleatoriamente en Diablo III [85]

Ésta manera de construir niveles se ha hecho popular conforme avanza el tiempo, creando un nuevo género, el “Roguelike”, especialmente usado con plataformas 2D.

En los videojuegos Roguelike, el jugador se adentra en un mundo (Habitualmente una mazmorra) que contiene niveles que se generan de manera procedimental, de manera que cada partida contará con niveles completamente diferentes.

Habitualmente los juegos del género presentan las 2 siguientes características:

- Los niveles son generados aleatoriamente, así como, posiblemente, cualquier otro elemento del juego.
- Al morir, el jugador empezará la siguiente partida de cero, perdiendo todo lo que haya obtenido anteriormente.

Éste género ha sido muy popular desde que los juegos 2D han vuelto a estar en áuge, dado que los menores costes que implica que es fácil implementar un nuevo elemento, que a su vez llevará a una cantidad elevada de contenido diferente, manteniendo tanto los costes de dinero como los costes de rendimiento bajos.

Si bien las dos características expuestas son lo que más define el género, se pueden modificar ligeramente, manteniendo la esencia Roguelike. En “Rogue Legacy” [72], el jugador obtiene dinero al derrotar enemigos, el cuál puede gastar cuando es derrotado, haciendo que su personaje sea más fuerte, haciendo posible que el jugador pueda adentrarse a más habitaciones, las cuales aumentan de dificultad según se avanza.

Ilustración 10: Nivel de Rogue Legacy generado aleatorieamente [86]

Un ejemplo que mantiene la esencia del Roguelike es “The Binding of Isaac” [73], un juego que genera sus niveles aleatoriamente cada partida, y en cada nivel hay objetos esparcidos generados de manera aleatoria.

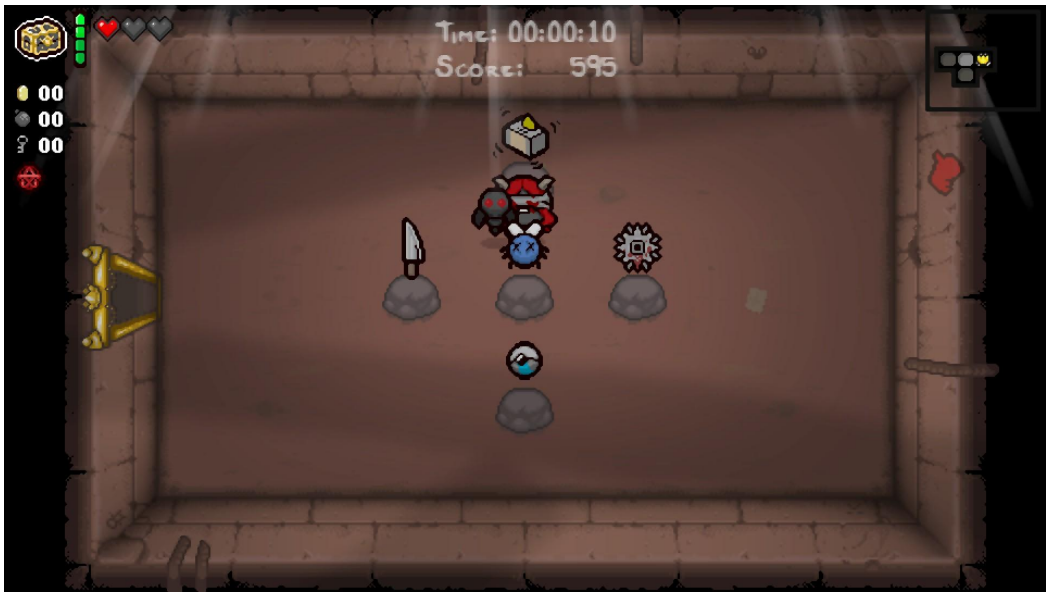

Ilustración 11: Objetos aleatorios en The Binding of Isaac [87]

Debido a la popularidad del género, y a los bajos costes que genera, parte muy influyente en juegos para dispositivos móviles, se ha optado por realizar un videojuego Roguelike, aplicando aleatoriedad para generar niveles diferentes en cada partida.

Capítulo 3. Planificación

En este capítulo realizaremos la gestión de riesgos, una primera planificación del proyecto, una previsión de costes de Recursos y el seguimiento del trabajo.

3.1 Gestión de riesgos

En este apartado realizaremos una compilación de posibles riesgos que puedan generar problemas a la hora de desarrollar nuestro proyecto con el objetivo de estar preparados frente a la posible ocurrencia de dichos riesgos. Identificaremos los riesgos del proyecto, los clasificaremos y propondremos soluciones para ellos.

Matriz de Impacto/Probabilidad

Impacto/ Probabilidad	Muy Alta	Alta	Media	Baja	Muy Baja
Catastrófico	Alto	Alto	Moderado	Moderado	Bajo
Crítico	Alto	Alto	Moderado	Bajo	Ninguno
Marginal	Moderado	Moderado	Bajo	Ninguno	Ninguno
Despreciable	Moderado	Bajo	Bajo	Ninguno	Ninguno

Tabla 1: Matriz de Riesgos

Según el **impacto** en tiempo/recursos, clasificamos el impacto de un riesgo de la siguiente manera:

- **Catastrófico**: Coste mayor del 30% del tiempo/recursos.
- **Crítico**: Coste de entre el 20 y el 30% del tiempo/recursos.
- **Marginal**: Coste de entre el 10 y el 20% del tiempo/recursos.
- **Despreciable**: Coste menor del 10% del tiempo/recursos.

Según la **probabilidad** de ocurrencia, clasificamos los riesgos de la siguiente manera:

- **Muy Alta**: Probabilidad mayor del 75%.
- **Alta**: Probabilidad de entre el 50% y el 75%.
- **Media**: Probabilidad de entre el 25% y el 50%.
- **Baja**: Probabilidad de entre el 10% y el 25%.
- **Muy Baja**: Probabilidad menor del 10%.

Listado de riesgos

Riesgo-01	**Retrasos respecto a la planificación inicial**		
Impacto	Crítico	Probabilidad	Alta
Riesgo	Alto		
Descripción	Se produce un retraso en la planificación inicialmente planteada.		
Estrategia	Evitar el riesgo		
Plan de contingencia	Adaptar la planificación aumentado los recursos para reducir el tiempo.		

Tabla 2: Riesgo-01: Retrasos respecto a la planificación inicial

Riesgo-02	**Falta de tiempo empleado**		
Valoración	Crítico	Probabilidad	Alta
Riesgo	Alto		
Descripción	Se emplea menos tiempo del esperado para una tarea.		
Estrategia	Evitar el riesgo		
Plan de contingencia	Incrementar el tiempo a emplear en otras tareas.		

Tabla 3: Riesgo-02: Falta de tiempo empleado

Riesgo-03	**Retrasos debido a complejidad del proyecto**		
Valoración	Marginal	Probabilidad	Alta
Riesgo	Moderado		
Descripción	El proyecto resulta ser más complejo de lo que se esperaba en un principio, de manera que requiere más trabajo.		
Estrategia	Investigar el riesgo		
Plan de contingencia	Incrementar la cantidad de tiempo a emplear de otras tareas.		

Tabla 4: Riesgo-03: Retrasos debido a complejidad del proyecto

Riesgo-04	**Cambio de requisitos**		
Valoración	Crítico	Probabilidad	Media
Riesgo	Moderado		
Descripción	Ocurre un cambio en los requisitos del proyecto, lo cuál requiere más tiempo y/o recursos.		
Estrategia	Evitar el riesgo		
Plan de contingencia	Realizar las modificaciones necesarias y adaptar la planificación teniendo en cuenta el coste de tiempo y/o recursos.		

Tabla 5: Riesgo-04: Cambio de requisitos

Riesgo-05	**Pérdida o destrucción de material**		
Valoración	Crítico	Probabilidad	Media
Riesgo	Moderado		
Descripción	Se pierden o destruyen materiales del proyecto.		
Estrategia	Protegerse del riesgo		
Plan de contingencia	Realizar las actividades necesarias para recuperar el material y adaptar la planificación teniendo en cuenta el coste de tiempo y/o recursos requeridos para ello.		

Tabla 6: Riesgo-05: Pérdida o destrucción de material

Riesgo-06	**Bajo rendimiento del ordenador**		
Valoración	Crítico	Probabilidad	Baja
Riesgo	Bajo		
Descripción	El ordenador presenta un rendimiento bajo, de manera que ralentiza o imposibilita el desarrollo del proyecto.		
Estrategia	Investigar el riesgo		
Plan de contingencia	Realizar el proyecto en otro ordenador		

Tabla 7: Riesgo-06: Bajo rendimiento del ordenador

Riesgo-07	**Fallo de la red**		
Valoración	Marginal	Probabilidad	Baja
Riesgo	Bajo		
Descripción	La red da problemas de conexión, de manera que impide acceder a recursos online.		
Estrategia	Investigar el riesgo		
Plan de contingencia	Trabajar sin conexión hasta que se resuelva y, en caso de ser necesario, conseguir acceso mediante otra red.		

Tabla 8: Riesgo-07: Fallo de la red

Riesgo-08	**Daños al material**		
Valoración	Catastrófico	Probabilidad	Muy Baja
Riesgo	Bajo		
Descripción	Daños a materiales físicos, como pueden ser el ordenador en el que se desarrolla el proyecto, produciendo grandes costes en tiempo y recursos.		
Estrategia	Protegerse del riesgo		
Plan de contingencia	Conseguir nuevos materiales y adaptar la planificación como sea necesario.		

Tabla 9: Riesgo-08: Daños al material

3.2 Roles y funciones

A continuación detallamos los roles que desempeña cada participante.

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 10: Roles y Responsabilidades de Margarita Gonzalo Tasis

Participante: Joaquín Adiego Rodriguez	
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 11: Roles y Responsabilidades de Joaquín Adiego Rodríguez

Participante: Daniel González Pérez	
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.
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.

Tabla 12: Roles y Responsabilidades de Daniel González Pérez

3.3 Planificación

Para la planificación de nuestro proyecto, utilizaremos una versión simplificada del Proceso Unificado.

En el proceso unificado, el proyecto se divide en tareas, de manera que por cada tarea, la cantidad de trabajo completado se va incrementando, de modo que según se avance en el proyecto, se puede observar como el proyecto va tomando forma.

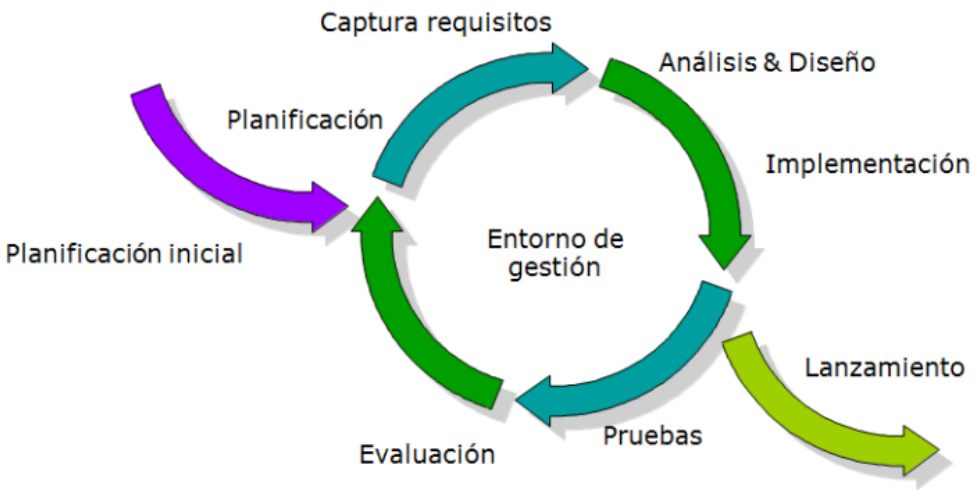

Ilustración 12: Proceso iterativo e incremental

En un proceso de desarrollo normal, por cada tarea se realiza un análisis, seguido del diseño y la implementación, y por último las pruebas y la evaluación de dicha tarea para dejarla completada.

Para agilizar el desarrollo, y debido a que no esperamos un cambio drástico de los requisitos durante éste, procederemos a relizar el análisis y el diseño de la aplicación al inicio del desarrollo, y, de ser necesario, se modificarán dichas partes según avance el proyecto.

De esta manera, vamos a dividir el proyecto en 5 fases:

- **Fase de aprendizaje**: En esta fase aprenderemos a usar las herramientas del software Unity3D para realizar las tareas del proyecto. (Tiempo estimado: 4 semanas)
- **Fase de análisis**: En esta fase realizaremos el análisis y la documentación pertinente. (Tiempo estimado: 2 semanas)
- **Fase de diseño**: En esta fase realizaremos el diseño y la documentación pertinente. (Tiempo estimado: 2 semanas)
- **Fase de implementación**: En esta fase desarrollaremos el software mediante Unity3D. (Tiempo estimado: 4 semanas)
- **Fase de documentación**: Finalización del documento. (Tiempo estimado: 3 semanas)

Y la fase de implementación estará subdividida en las siguientes 4 tareas, de 1 semana de duración cada una.

- **Menú Principal:** En esta tarea desarrollaremos el menú principal y las transiciones al modo de juego y al modo debug.
- **Juego parte 1 - Generación de nivel:** En esta tarea desarrollaremos la generación de niveles.
- **Juego parte 2 – Sistemas:** En esta tarea desarrollaremos los sistemas para el modo juego.
- **Modo Debug:** En esta tarea desarrollaremos las acciones del modo debug, en el cuál podremos observar cómo se generan los niveles aleatoriamente.

Terminando así el proyecto en la segunda mitad de Mayo según nuestra planificación inicial.

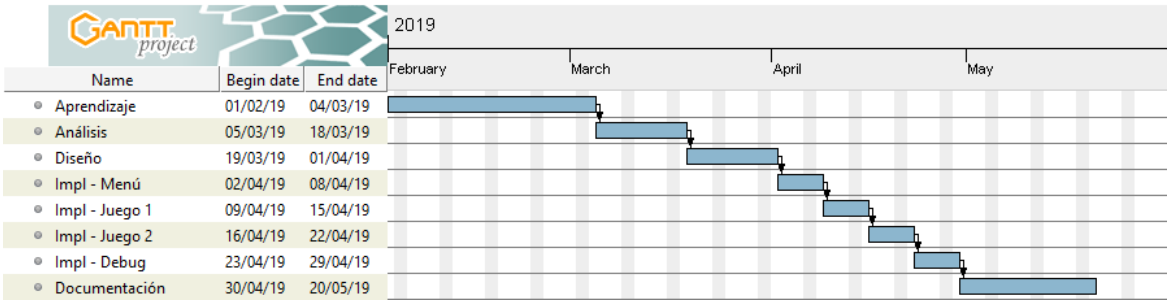

Ilustración 13: Diagrama de Gantt de la planificación inicial

3.4 Costes de Producción

A continuación desglosaremos los costes estimados asociados a nuestra planificación.

Rol	Precio/Hora	Horas estimadas	Coste total
Jefe de proyecto	60 €	20	1200 €
Diseñador Software	50 €	50	2500 €
Programador	40 €	50	2000 €
Diseñador Grafico	20 €	20	400 €

Tabla 13: Costes de Recursos Humanos

Concepto	Precio
Conexión a Internet	50 €
Electricidad	60 €
Licencia Unity3D	Gratis
Licencia GIMP	Gratis
Licencia LibreOffice	Gratis
Licencia Gantt Project	Gratis
Licencia Visual Paradigm	Gratis
Material de oficina (Papel, bolígrafos...)	3 €

Tabla 14: Costes de Recursos Materiales

Una vez desglosado, podemos ver que el precio asciende a un total de 6100 € en Recursos Humanos y 113 € en Recursos Materiales, sumando un total de 6213 €.

3.5 Seguimiento

Dutrante el desarrollo del proyecto se ha ido tomando nota de lo que se ha tardado en completar las tareas, de manera que obtenemos el siguiente diagrama de Gantt.

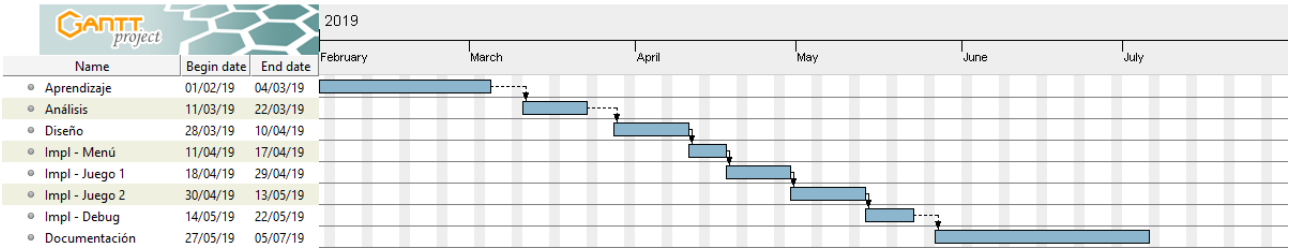

Ilustración 14: Diagrama de Gantt del seguimiento del proyecto

Como podemos observar, se ha tenido que retrasar la finalización hasta la primera mitad de Julio. La materialización de riesgos que habíamos previsto con anterioridad han ocasionado dicho retraso.

El primero de ellos es la falta de tiempo empleado antes de la realización de las fases de análisis y diseño, retrasando la fecha de finalización (Que es diferente a incrementar el tiempo que hemos usado para el proyecto)

El segundo es el tiempo empleado en las fases de implementación. Debido a la complejidad del proyecto, que no fue no prevista, la implementación ha supuesto un incremento de tiempo inesperado.

También hubo unos días en los que no se ha trabajado entre la finalización de la implementación y el inicio de la fase de documentación, debido a que resultaba de interés descansar unos días, puesto que el desarrollo del proyecto se ha llevado a cabo mientras el desarrollador trabajaba por las mañanas.

Por último, la duración del desarrollo de la documentación fue drásticamente incrementado, debido a las peticiones del cliente de desarrollar una documentación con un mayor nivel de detalle.

En total, hemos visto una desviación de 64 horas de desviación frente a la estimación original. Siendo la planificación original un total de 216 horas, y el resultado un total de 280 horas.

Capítulo 4. Análisis

En este capítulo definiremos las principales características y especificaciones de nuestra aplicación. Definiremos los requisitos funcionales, que definen la funcionalidad de nuestro juego, y los no funcionales, que definen cómo debe hacerlo. Expondremos el diagrama de casos de uso y definiremos dichos casos de uso, y, finalmente, compondremos el modelo de dominio de nuestro juego.

4.1 Requisitos funcionales

Los siguientes requisitos definen el comportamiento deseado de nuestra aplicación.

RF-01	Jugar partida
Descripción	Al usuario se le permitirá empezar una partida desde el menú principal.

Tabla 15: RF-01: Jugar partida

RF-02	Mover personaje
Descripción	Al usuario se le permitirá mover a su personaje, vertical y horizontalmente, durante el juego.

Tabla 16: RF-02: Mover personaje

RF-03	Recoger objetos
Descripción	Al usuario se le permitirá recoger objetos durante el juego.

Tabla 17: RF-03: Recoger objetos

RF-04	Final de partida
Descripción	El sistema finalizará la partida si el personaje toca un bloque malo o se acaba el tiempo.

Tabla 18: RF-04: Final de partida

RF-05	Reiniciar partida
Descripción	Al usuario se le permitirá reiniciar una partida cuando llegué a la pantalla de final de partida.

Tabla 19: RF-05: Reiniciar partida

RF-06	Volver al menú principal
Descripción	Al usuario se le permitirá volver al menú principal en cualquier momento.

Tabla 20: RF-06: Volver al menú principal

RF-07	Modo debug
Descripción	Al usuario se le permitirá entrar al modo debug desde el menú principal.

Tabla 21: RF-07: Modo debug

RF-08	Modo debug: Mover cámara
Descripción	Al usuario se le permitirá mover la cámara en el modo debug.

Tabla 22: RF-08: Modo debug: Mover cámara

RF-09	Modo debug: Zoom cámara
Descripción	Al usuario se le permitirá hacer zoom con la cámara en el modo debug.

Tabla 23: RF-09: Modo debug: Zoom cámara

RF-10	Modo debug: Rehacer nivel
Descripción	Al usuario se le permitirá rehacer el nivel del modo debug.

Tabla 24: RF-10: Modo debug: Rehacer nivel

RF-11	Modo debug: Rehacer habitaciones
Descripción	Al usuario se le permitirá rehacer las habitaciones de un nivel, manteniendo la estructura de éste, en el modo debug.

Tabla 25: RF-11: Modo debug: Rehacer habitaciones

RF-12	Modo debug: Cambiar tamaño nivel
Descripción	Al usuario se le permitirá cambiar el tamaño del nivel a generar en el modo debug.

Tabla 26: RF-12: Modo debug: Cambiar tamaño nivel

4.2 Requisitos no funcionales

Los siguientes requisitos presentan los cumplimientos de calidad de nuestra aplicación.

RNF-01	Restricción: Motor de desarrollo
Descripción	El sistema deberá ser desarrollado usando el motor Unity3D.

Tabla 27: RNF-01: Restricción: Motor de desarrollo

RNF-02	Restricción: Lenguaje de programación
Descripción	Los scripts deberán ser desarrollados usando C#.

Tabla 28: RNF-02: Restricción: Lenguaje de programación

RNF-03	Plataforma objetivo
Descripción	El sistema deberá estar hecho para funcionar en dispositivos Android.

Tabla 29: RNF-03: Plataforma objetivo

RNF-04	Fluidez en transiciones
Descripción	Las transiciones entre escenas no durarán más de 5 segundos bajo circunstancias normales.

Tabla 30: RNF-04: Fluidez en transiciones

RNF-05	Fluidez en controles
Descripción	El sistema deberá responder inmediatamente a las acciones del usuario.

Tabla 31: RNF-05: Fluidez en controles

RNF-06	Escalabilidad
Descripción	El sistema deberá ser escalable, de manera que se pueda añadir funcionalidad sin que afecte al resto de las funciones.

Tabla 32: RNF-06: Escalabilidad

4.3 Modelo de casos de uso

El siguiente diagrama presenta los diferentes casos de uso que pueda realizar un usuario al utilizar nuestra aplicación.

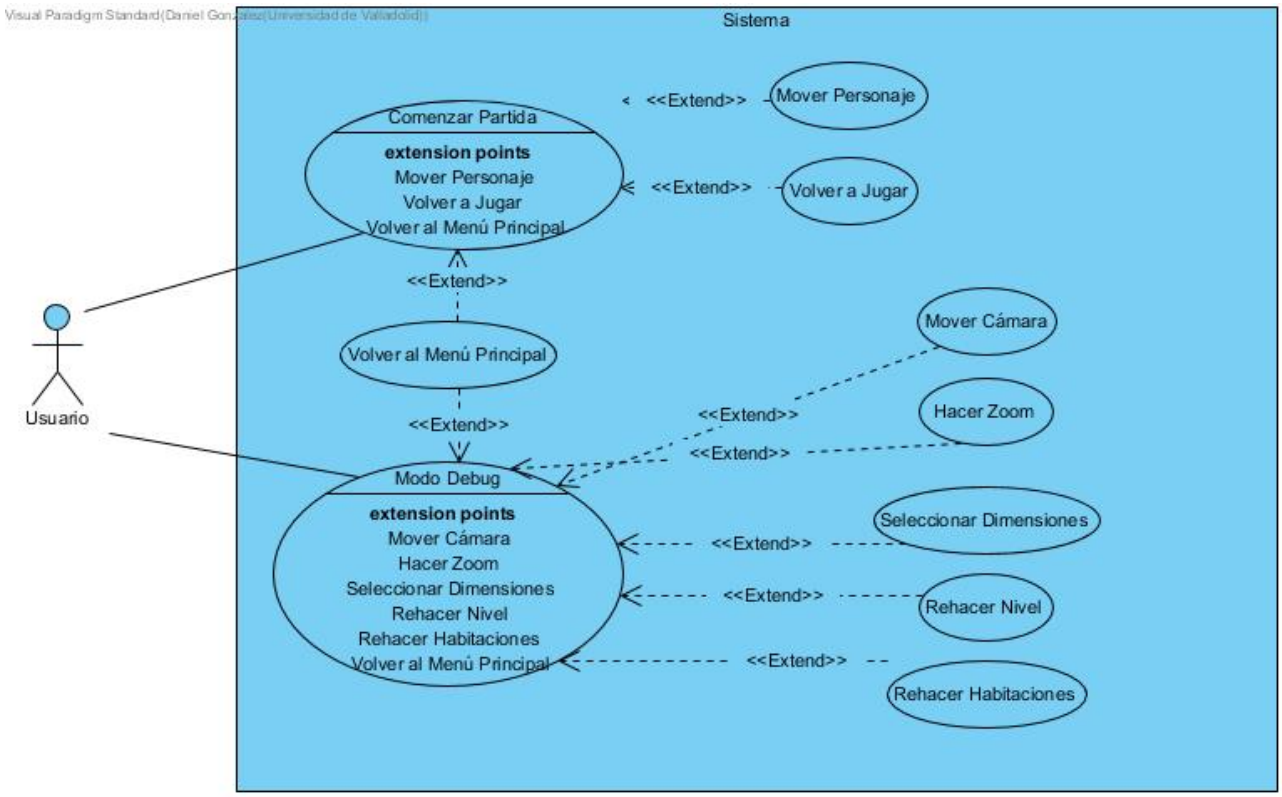

Ilustración 15: Modelo de Casos de Uso

4.4 Especificación de casos de uso

En este apartado especificamos más en detalle cada caso de uso presentado anteriormente.

CU-01	Comenzar Partida	
Descripción	El usuario comienza una partida nueva.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de jugar.
	2	El sistema cambia a la escena de Partida y crea una partida nueva.

Tabla 33: CU-01: Comenzar Partida

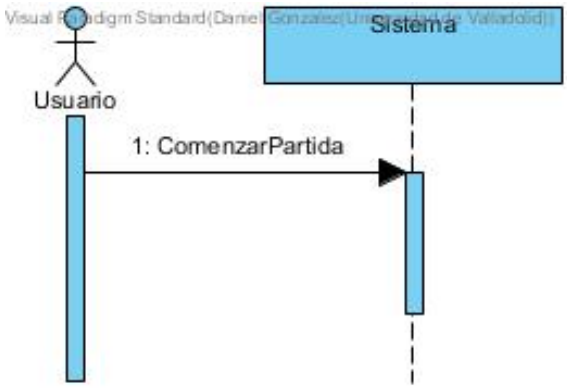

Ilustración 16: Diagrama de secuencia de CU-01: Comenzar Partida

CU-02	Mover Personaje	
Descripción	El usuario mueve su personaje hacia la izquierda, derecha o arriba.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de desplazarse.
	2	El sistema desplaza al personaje en la dirección elegida.

Tabla 34: CU-02: Mover Personaje

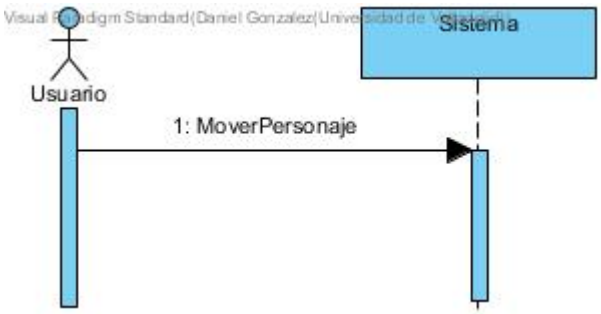

Ilustración 17: Diagrama de secuencia de CU-02: Mover Personaje

CU-03	Volver a Jugar	
Descripción	En la pantalla de game over, el jugador selecciona volver a jugar una partida.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de volver a jugar.
	2	El sistema crea una partida nueva.

Tabla 35: CU-03: Volver a Jugar

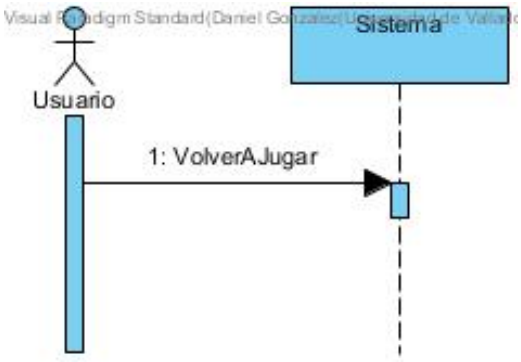

Ilustración 18: Diagrama de secuencia de CU-03: Volver a Jugar

CU-04	Volver al Menú Principal	
Descripción	El usuario vuelve al menú principal.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de volver al menú principal.
	2	El sistema cambia a la escena de Menú Principal.

Tabla 36: CU-04: Volver al Menú Principal

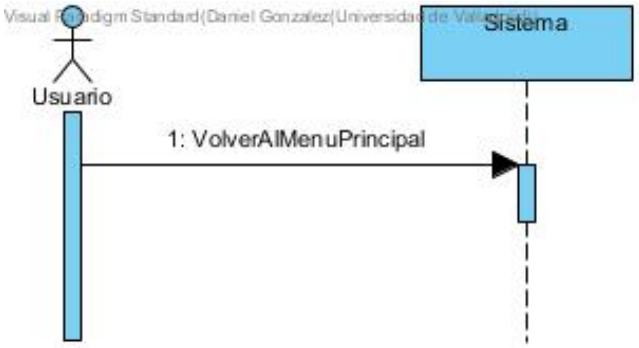

Ilustración 19: Diagrama de secuencia de CU-04: Volver al Menú Principal

CU-05	Modo Debug	
Descripción	El usuario entra al modo debug.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de modo debug.
	2	El sistema cambia a la escena de modo debug y crea un nivel.

Tabla 37: CU-05: Modo Debug

CU-06	Mover Cámara	
Descripción	El usuario mueve la cámara del modo debug en una de las 4 direcciones.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de mover la cámara.
	2	El sistema mueve la cámara en la dirección elegida.

Tabla 38: CU-06: Mover Cámara

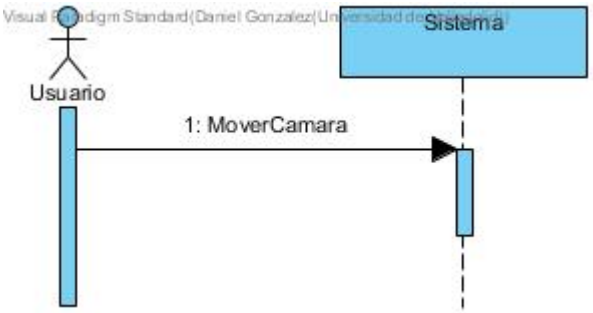

Ilustración 20: Diagrama de secuencia de CU-06: Mover Cámara

CU-07	Hacer Zoom	
Descripción	El usuario hace zoom con la cámara del modo debug.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de hacer zoom.
	2	El sistema hace zoom a la cámara en la dirección elegida.

Tabla 39: CU-07: Hacer Zoom

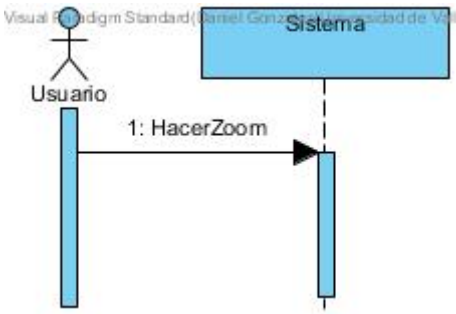

Ilustración 21: Diagrama de secuencia de CU-07: Hacer Zoom

CU-08	Seleccionar Dimensiones	
Descripción	El usuario cambia las dimensiones del nivel a generar.	
Flujo	Paso	Descripción
	1	El usuario toca el tamaño de la dimensión a cambiat.
	2	El sistema presenta el teclado al usuario.
	3	El usuario introduce la cantidad deseada.
	4	El sistema cambia la dimensión del nivel a la introducida.

Tabla 40: CU-08: Seleccionar Dimensiones

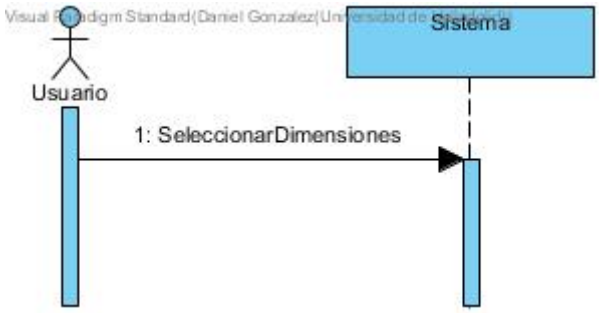

Ilustración 22: Diagrama de secuencia de CU-08: Seleccionar Dimensiones

CU-09	Rehacer Nivel	
Descripción	El usuario rehace el nivel.	
Flujo	Paso	Descripción
	1	El usuario toca el botón de rehacer el nivel.
	2	El sistema borra el nivel actual y crea uno nuevo.

Tabla 41: CU-09: Rehacer Nivel

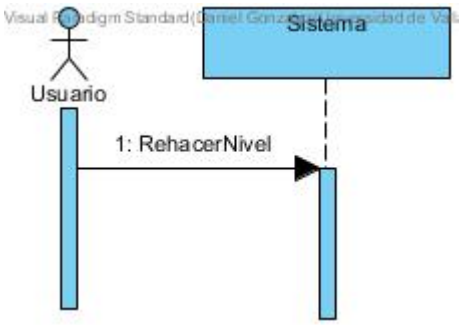

Ilustración 23: Diagrama de secuencia de CU-09: Rehacer Nivel

CU-10	Rehacer Habitaciones	
Descripción	El usuario rehace las habitaciones	
Flujo	Paso	Descripción
	1	El usuario toca el botón de rehacer las habitaciones.
	2	El sistema borra las habitaciones del nivel actual y crea nuevas.

Tabla 42: CU-10: Rehacer Habitaciones

Ilustración 24: Diagrama de secuencia de CU-10: Rehacer Habitaciones

4.5 Modelo de dominio

A continuación mostramos el modelo de dominio pertinente a nuestro análisis.

Debido a la complejidad del proyecto, las siguiente clases del análisis se presentan de manera genérica, sin ser éstas las clases implementadas finalmente.

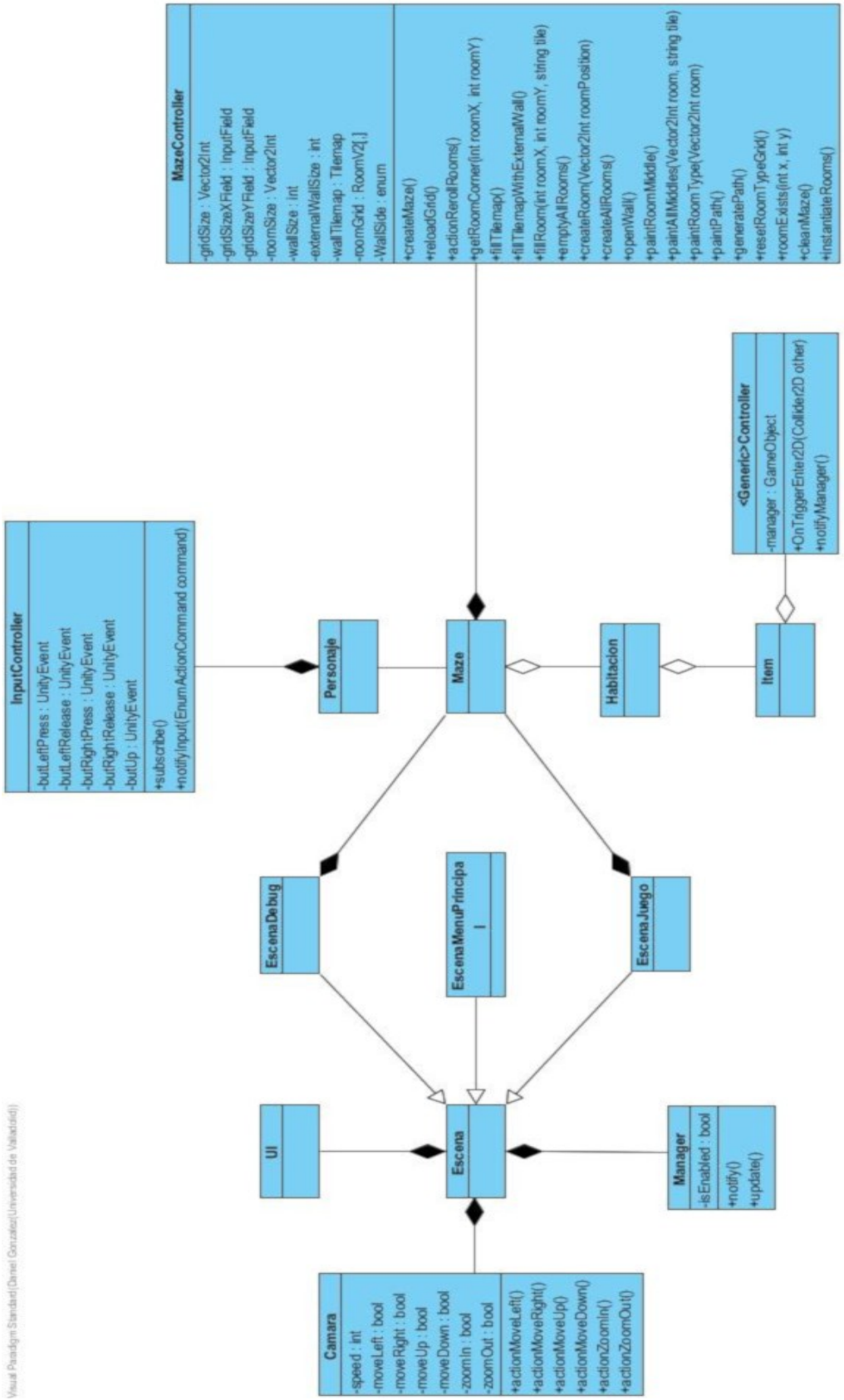

Ilustración 25: Diagrama de Dominio

Nuestro juego está compuesto por una variedad de clases. Dichas clases, aunque puedan parecer diferentes, interiormente siguen una lógica casi idéntica. Por ello, con el fin de no repetir todo para cada uno de los varios objetos, definiremos los diferentes tipos de clases que forman parte del proyecto.

- **Escenas**

Lo que se presenta en pantalla. Hay 3 escenas, la escena del menú principal, que lleva a las otras dos, la escena de juego, en la cual el usuario podrá jugar, y la escena de debug, en la que se puede visualizar la generación de niveles aleatorios. Las escenas cuentan con una cámara, para visualización, UI, para interfaz de usuario, y un conjunto de managers.

- **Managers**

Los managers son los objetos encargados de manejar los varios sistemas (Input, vida, puntuación…). Su tarea principal es la de comunicar cualquier evento que ocurra en el juego a los controllers para mantenerlos actualizados y, si se requiere, que reaccionen a dicho evento. Los eventos pueden provenir por parte del jugador o por parte de elementos del juego, que notifican a los managers mediante sus controllers. Por ejemplo, el manager de Input recibe un evento por parte del usuario, encapsula este evento para que los controllers puedan leerlo y se lo envía a todos los controllers relacionados con el manager.

- **Controllers**

Definen que acciones realiza el objeto frente a los eventos recibidos por parte de los managers. Una vez recibido un evento, los controllers son los encargados de realizar una de entre las acciones disponibles de su objeto para reaccionar a dicho evento. Por ejemplo, frente a un evento de “Botón saltar pulsado”, el input controller del personaje indicará al personaje que debe realizar la acción “Saltar”.

- **Maze**

El nivel generado aleatoriamente. Cada vez que el juego entra a la escena de juego, se genera un Maze (O nivel). Dicho Maze genera una serie de habitaciones de manera aleatoria, siguiendo una serie de reglas básicas y parámetros ajustables a elección del desarrollador. (Ver capítulo 6)

- **Habitacion**

Cada una de las habitaciones que contiene un Maze. El camino que genera un Maze está formado por habitaciones. Cada habitación presenta al jugador con un desafío a superar para continuar su camino, con la opción de recoger Items mientras lo hace. Éstas habitaciones, al igual que el camino del Maze, se generan de manera aleatoria, de manera que aunque el camino fuera el mismo las habitaciones a recorrer serían diferentes.

- **Item**

Objeto del juego que interaccionan con el personaje del jugador. Para dicho objetivo, los Items cuentan con un controller capaz de enviar una notificación a su respectivo manager. Por ejemplo, al ponerse en contacto con una pompa, ésta es destruida y notifica dicha interacción al manager de vida, para que la vida del jugador sea aumentada. Los objetos se generan dentro de las habitaciones, generalmente de manera aleatoria, aunque hay ciertas habitaciones con Items que siempre se generan de la misma manera, con el fin de que la generación aleatoria no se vuelva en contra del jugador.

- **Personaje**

Objeto que controla el jugador, capaz de moverse a través del Maze mediante el input del jugador. El objetivo del juego es usar al personaje para recolectar la mayor cantidad de puntos posibles.

Capítulo 5. Diseño

En este capítulo describiremos la arquitectura y los patrones adecuados para nuestro proyecto y los adaptaremos para un óptimo funcionamiento de nuestro proyecto.

5.1 Arquitectura lógica

En un videojuego realizado en Unity podemos diferenciar dos partes principales: Los objetos del juego y los sistemas que interactúan con dichos objetos.

Debido a esto, nos comviene usar una arquitectura que nos permita abstraer dichas partes, de manera que podamos trabajar sobre ellas individualmente sin afectar la una a la funcionalidad de la otra. Para esto hemos optado por escoger una arquitectura por capas, siendo las capas las siguientes:

- **Managers (Manejadores)**

En esta capa implementaremos los sistemas de nuestro videojuego, como vienen siendo los de escenas, input del usuario, puntuación, etc…

- **Controllers (Controladores)**

En esta capa implementaremos los controladores de los objetos. Estos controladores definirán como actuarán los objetos frente a diferentes eventos.

- **Objects (Objetos)**

En esta capa implementaremos la lógica de los objetos.

5.2 Patrones de diseño

Para un mejor funcionamiento de nuestra aplicación, hemos decidido aplicar los siguientes patrones de diseño.

5.2.1 Patrón observador

Debido a que en un videojuego el personaje que maneja el jugador están cambiando constantemente, necesitaremos una manera de actualizarlo. Unity cuenta con un método Update, que se ejecuta 60 veces por segundo. Usando este método se podrían realizar comprobaciones de input, sin embargo, realizar dicha tarea tantas veces por segundo es muy costoso.

Debido a esto hemos decidido realizar dichas comprobaciones mediante eventos, de manera que sólo se realice dicha tarea cuando sea necesaria. Para ello utilizaremos una adaptación del patrón observador.

En el patrón observador, tenemos 2 tipos de elementos, el sujeto y los observadores. El sujeto mantiene una lista de observadores, y los notifica de cualquier cambio. Es un patrón muy popular para aplicaciones basadas en eventos, como la nuestra.

Para ello crearemos una interfaz IsubscriberInput, que implementaremos a los objetos que queramos que sean observadores. En nuestra aplicación, el sujeto será un InputManager y nuestro observador será nuestro personaje. Debido al patrón, podemos tener un grupo de observadores, pero en nuestro caso solo queremos que sea el personaje que maneja el jugador el que reaccione ante el Input del usuario.

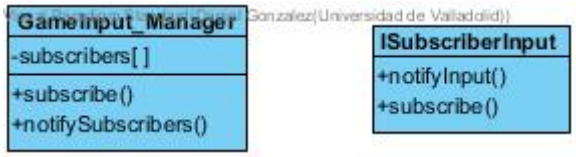

Ilustración 26: Aplicación del Patrón observador

5.2.2 Patrón comando

Si quisieramos hacer una aplicación muy sencilla, podríamos hacer que cada interacción del usuario le diga directamente a un objeto lo que tiene que hacer. Esto está bien para realizar pruebas sobre la funcionalidad de un objeto, sin embargo nos cierra las puertas a que la aplicación sea escalable y flexible ante cambios.

Para que nuestra aplicación pueda cambiar con relativa sencillez, implementaremos el patrón comando.

En el patrón comando, tenemos 3 elementos, el emisor, el receptor y el comando. Para que el receptor pueda realizar una tarea, el emisor encapsula los datos necesarios en un objeto comando y se lo envía al receptor.

Nosotros vamos a simplificar esto y en vez de enviar objetos, vamos a pasar un tipo ENUM, para que el receptor sepa lo que tiene que hacer.

En nuestra aplicación, un InputManager será el que envíe dicho comando a un InputController.

De esta manera conseguimos poder configurar cómo queremos que el usuario active una clase de acción y en el controller configuraremos cómo queremos que el objeto reaccione ante dicha acción.

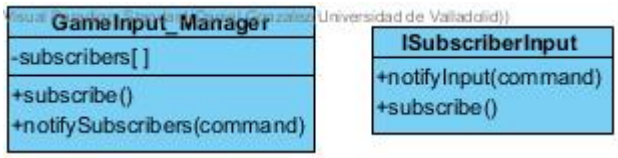

Ilustración 27: Aplicación del patrón comando

5.3 Diagramas de secuencia

A continuación presentamos los diagramas de secuencia para el diseño de los casos de uso previamente expuestos.

DS-01 Comenzar Partida

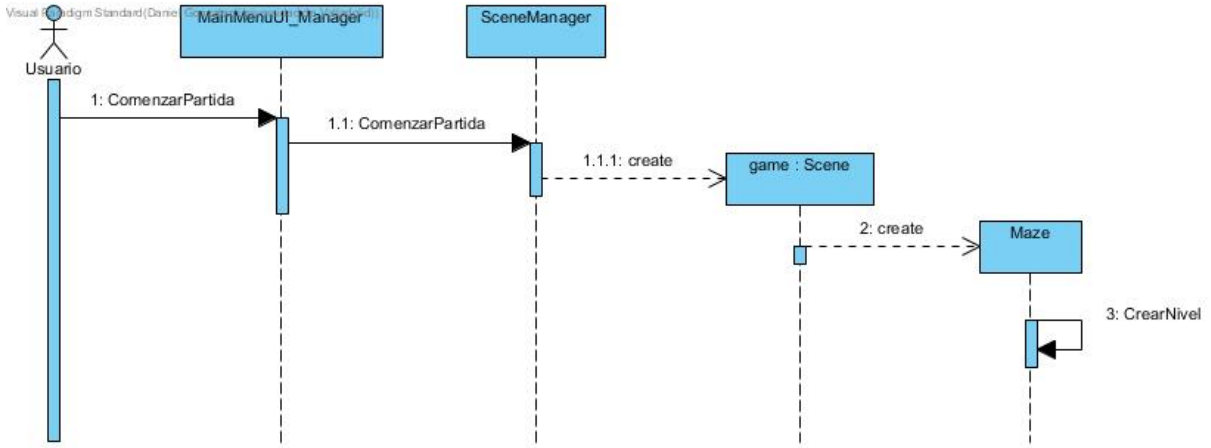

Ilustración 28: DS-01 Comenzar Partida

DS-02 Mover Personaje

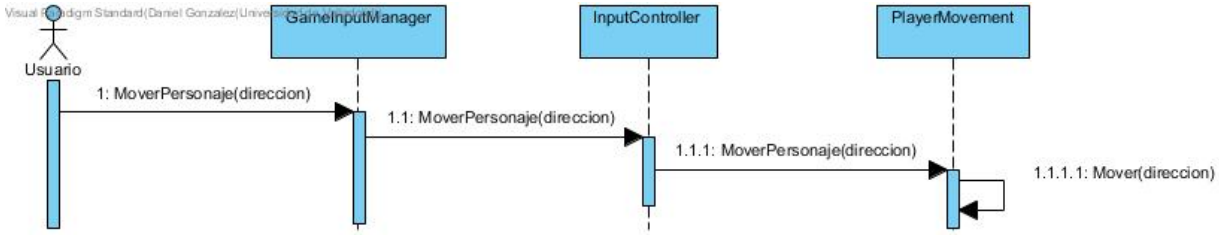

Ilustración 29: DS-02 Mover Personaje

DS-03 Volver a Jugar

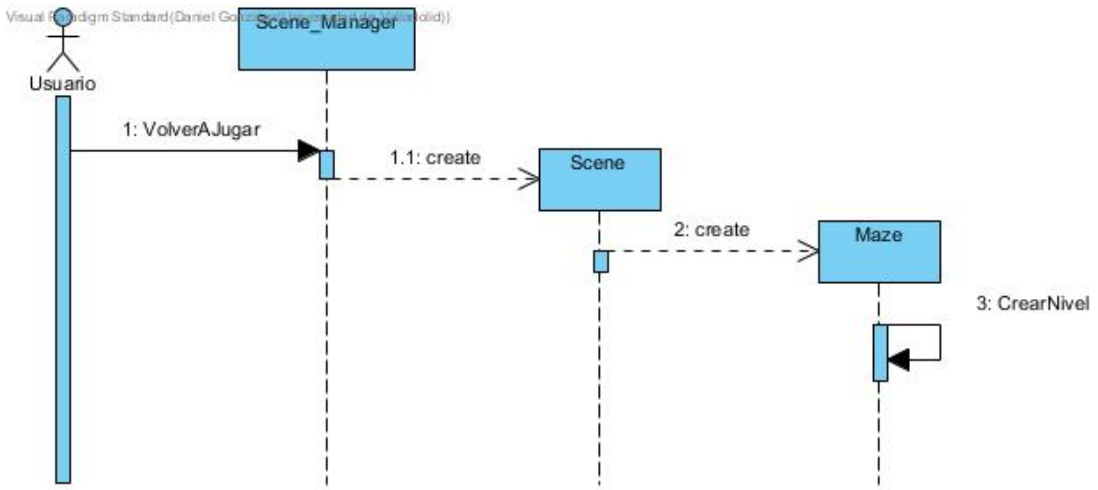

Ilustración 30: DS-03 Volver a Jugar

DS-04 Volver al Menú Principal

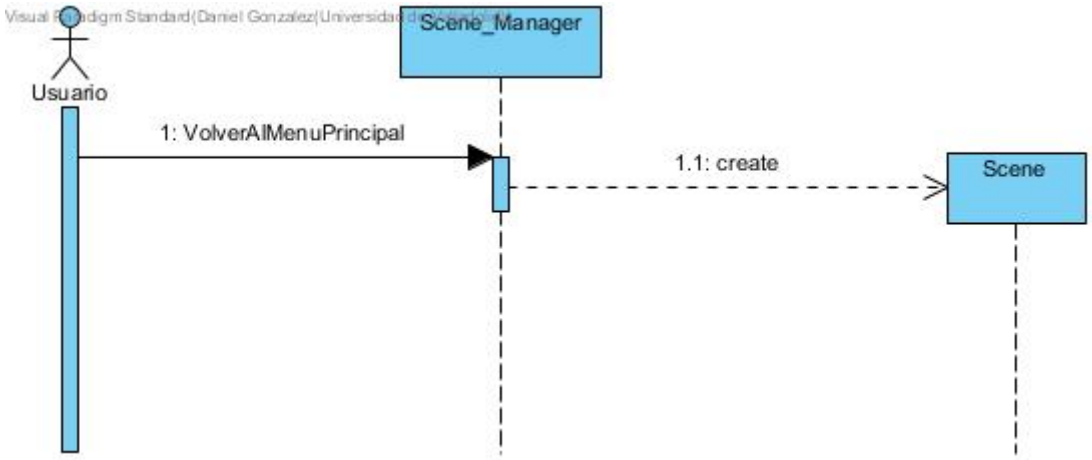

Ilustración 31: DS-04 Volver al Menú Principal

DS-05 Modo Debug

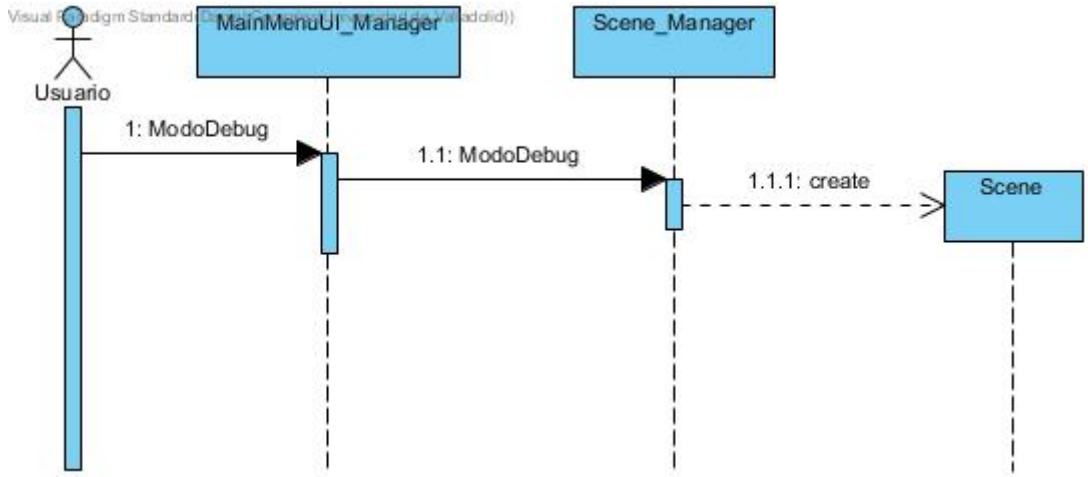

Ilustración 32: DS-05 Modo Debug

DS-06 Mover Cámara

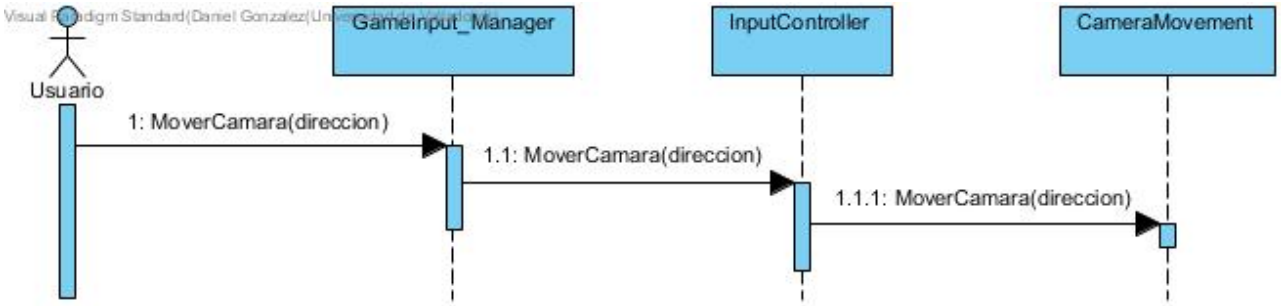

Ilustración 33: DS-06 Mover Cámara

DS-07 Hacer Zoom

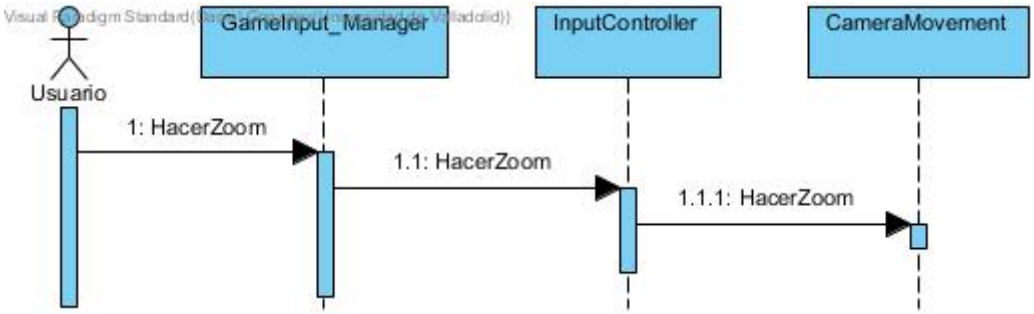

Ilustración 34: DS-07 Hacer Zoom

DS-08 Seleccionar dimensiones

Ilustración 35: DS-08 Seleccionar dimensiones

DS-09 Rehacer Nivel

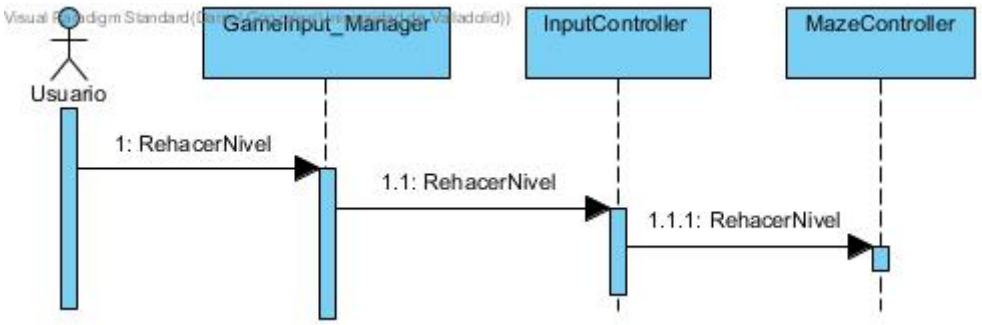

Ilustración 36: DS-09 Rehacer Nivel

DS-10 Rehacer Habitaciones

Ilustración 37: DS-10 Rehacer Habitaciones

5.4 Diagrama de paquetes

A continuación se puede ver el diagrama de paquetes de nuestro sistema. Nótese que debido a la modularidad interna de nuestro proyecto, no tenemos dependencias entre paquetes.

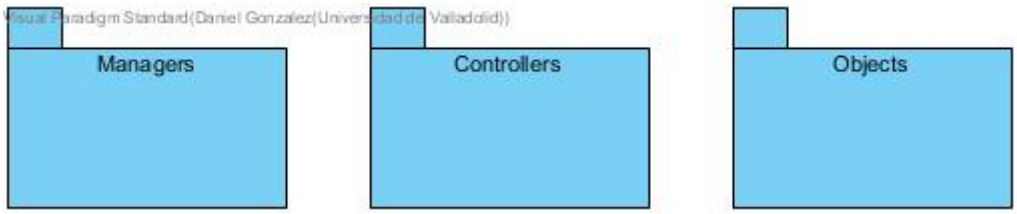

Ilustración 38: Diagrama de paquetes

Capítulo 6. Implementación

En este capítulo describiremos cómo hemos implementado la generación aleatoria de un nivel de nuestro juego, tanto para el camino que genera un Maze como para la generación de las habitaciones que conforman el camino del Maze

6.1 Maze

Para generar nuestros niveles, tenemos un objeto Maze, que genera dicho nivel en una matriz tablero de 2 dimensiones, donde se guardan las habitaciones. A este objeto se le pueden dar parametros de configuración varios, como son la cantidad de habitaciones, tamaño de dichas habitaciones y tamaño del muro que rodea el nivel. Esto hace que nuestro Maze pueda ser usado en variedad de situaciones.

Dicho ésto, para nuestro juego tenemos configurado el Maze para que genere niveles de 5 de ancho por 10 de alto, dado que tras pasar un tiempo probando dimensiones, son las que nos han parecido más adecuadas. Es de alta importancia configurar unas dimensiones adecuadas, dado que, aunque podemos configurar las dimensiones al desarrollar el juego y en el modo debug, no hemos implementado la función para cambiar dimensiones en el modo juego.

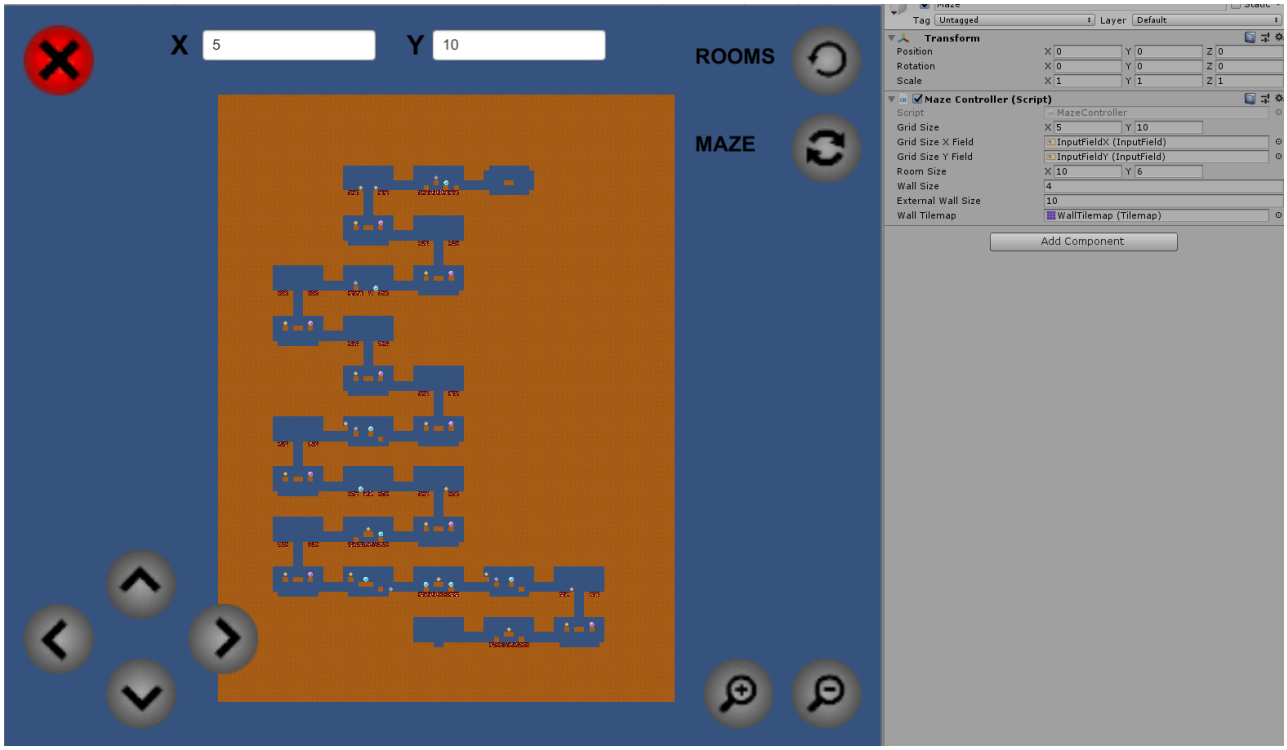

Ilustración 39: Parámetros para dar a un Maze

6.2 Generación de nivel

Para generar nuestro nivel, se siguen los siguientes pasos:

1. **Recargar tablero:** Limpia el nivel anterior, de haber alguno.
2. **Crear muro exterior:** Genera el muro exterior, para que el usuario no vea “vacío” más allá de los límites del nivel.
3. **Vaciar habitaciones:** Se asegura de que no hay nada ocupando el espacio de las habitaciones.
4. **Generar camino:** El algoritmo que genera el camino en nuestro tablero.
5. **Instanciar habitaciones:** Se generan las habitaciones según nuestro tablero.

En este apartado, definiremos cómo se genera nuestro camino de manera aleatoria.

Primero decidimos cuál será nuestra habitación de inicio. Para ello seleccionaremos al azar una de las habitaciones de la primera fila (Distribución uniforme). Una vez seleccionada, guardamos la posición de dicha habitación (Que actualizaremos según avancemos por el tablero). Una vez hecho esto, se entra en un bucle.

Cada vez que entramos a una fila nueva se selecciona una dirección a la que ir, dicha dirección será el lado opuesto al lado en el que se encuentre la primera habitación de la fila. Con esto evitamos que el algoritmo pueda crear un camino que se dirija siempre en la misma dirección.

Tras esta selección, en cada habitación en la que nos situemos, se escoge si el camino avanza en esa dirección o avanza a la siguiente fila. Seleccionamos al azar, de manera que halla un 5/7 de probabilidad de que el camino avance horizontalmente en esa fila, y un 2/7 de que pase a la siguiente fila. Ésta ha sido la probabilidad escogida, dado que una menor probabilidad resultaba en caminos horizontales muy cortos y una mayor probabilidad resultaba en caminos horizontales que habitualmente abarcaban todo el ancho de una fila, y en nuestro juego queríamos que el jugador pudiera pasar de fila más a menudo para no sentirse atascado. En caso de ser la primera habitación de una fila, el algortimo siempre avanzará horizontalmente, de manera que en cada fila siempre habrá al menos una habitación de bajada y otra de subida.

Este proceso se repetirá fila tras fila hasta la última fila del tablero. En dicha fila, cuando el algoritmo seleccione bajar de fila, se generará una habitación de final de nivel y el algoritmo se detendrá.

A continuación, mostramos una imagen de un camino generado por nuestro algoritmo.

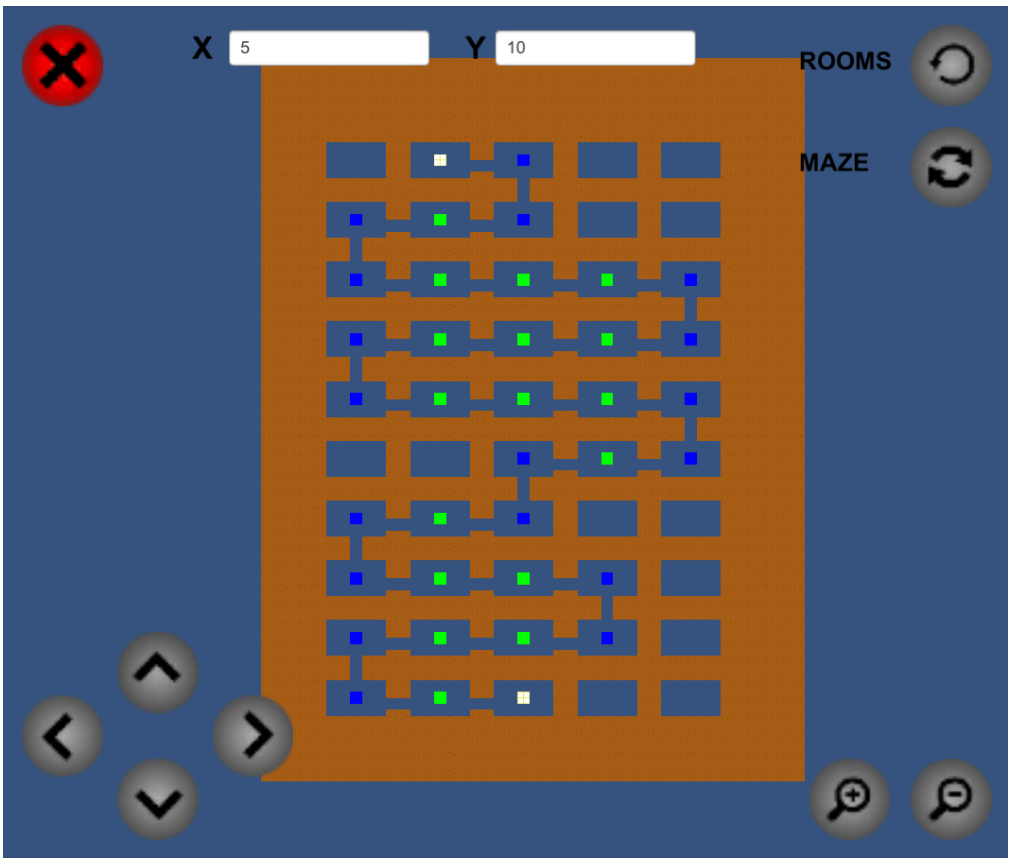

Ilustración 40: Demonstración algoritmo generador de caminos

- **Blanco:** Habitación de inicio o final de nivel.
- **Azul:** Habitación de avance vertical.
- **Verde:** Habitación de avance horizontal.

6.3 Generación de habitaciones

Una vez hemos generado el tablero de habitaciones, el objeto Maze creará las habitaciones en base a dicho tablero. En nuestro caso tenemos los siguientes tipos de habitaciones:

- **Vacía:** Una habitación que no forma parte del camino.
- **Inicial:** La primera habitación del nivel.
- **Camino:** Una habitación de desplazamiento horizontal.
- **Bajada:** La última habitación de una fila. Baja a la siguiente fila.
- **Subida:** La primera habitación de una fila. Siempre sigue después de una habitación de bajada.
- **Final:** La última habitación de un nivel.

En caso de ser una habitación vacía, se rellenará dicha habitación como si fuera una pared. Para las demás habitaciones, tenemos un grupo de habitaciones construidas previamente para cada tipo de habitación. A la hora de generar la habitación, se escoge aleatoriamente una habitación de ese tipo (Distribución uniforme)

A mayores, las habitaciones cuentan con **generadores** (de puntos y vida) Estos generadores están construidos de manera que al añadirlos a nuestras habitaciones, podamos seleccionar el tipo de objeto a generar y la probabilidad de que lo generen. Para nuestro juego, se ha seleccionado un 80% de probabilidad de que se generen, de manera que cree la apariencia de diferentes salas, pero en la mayoría de los casos el jugador siempre tenga algo con lo que interactuar. Además de estos generadores aleatorios, también hemos colocado generadores que siempre generen objetos, de manera que el jugador tenga la seguridad de recoger un mínimo de objetos en una partida.

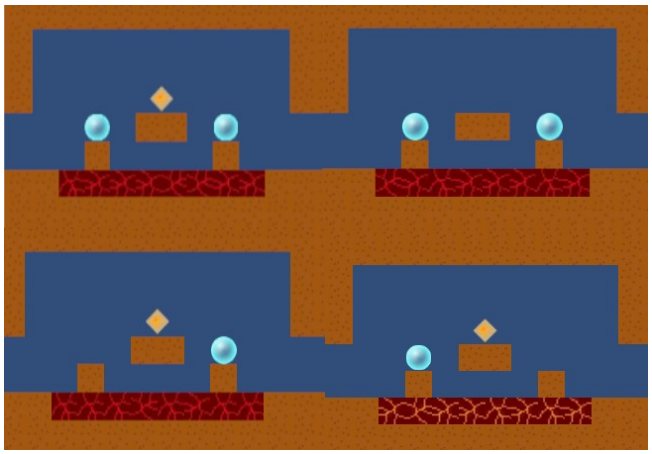

Ilustración 41: Demostración aleatoriedad en un tipo de habitación

Capítulo 7. Pruebas

En este capítulo expondremos las pruebas realizadas en las diferentes tareas de implementación de nuestro proyecto. Ya que el equipo de desarrollo cuenta con una sola persona, las pruebas de caja blanca no concluirían en ningún resultado que sea de utilidad, dado a que dicha persona es la encargada de desarrollar el código en un principio. Debido a esto, solo se realizarán pruebas de caja negra.

7.1 Pruebas de caja negra

El objetivo de las pruebas de caja negra es tratar de dar con posibles errores en nuestra aplicación realizando dichas pruebas directamente sobre nuestra aplicación. Esto nos servirá para demostrar la correcta funcionalidad de nuestra aplicación en caso de que no haya errores.

Para comprobar que no haya habido conflictos entre tareas, también realizaremos dichas pruebas en la versión final de nuestra aplicación.

PCN-01	Jugar partida
Tarea	Menú Principal
Resultado esperado	Al tocar el botón play del menú principal, el juego cambia a la escena de juego.
Valoración	OK
Valoración final	OK

Tabla 43: PCN-01: Jugar partida

PCN-02	Modo debug
Tarea	Menú Principal
Resultado esperado	Al tocar el botón de debug del menú principal, el juego cambia a la escena de modo debug.
Valoración	OK
Valoración final	OK

Tabla 44: PCN-02: Modo debug

PCN-03	Volver al menú principal
Tarea	Menú Principal
Resultado esperado	Al tocar el botón de volver al menú principal, se cambia de escena al menú principal
Valoración	OK
Valoración final	OK

Tabla 45: PCN-03: Volver al menú principal

PCN-04	Transición
Tarea	Menú Principal
Resultado esperado	Al cambiar de escena, se activa la transición.
Valoración	OK
Valoración final	OK

Tabla 46: PCN-04: Transición

PCN-05	Generación de nivel
Tarea	Juego parte 1 – Generación de nivel
Resultado esperado	Al comenzar una partida se genera un nivel.
Valoración	OK
Valoración final	OK

Tabla 47: PCN-05: Generación de nivel

PCN-06	Generación de camino
Tarea	Juego parte 1 – Generación de nivel
Resultado esperado	El camino generado por el nivel se genera aleatoriamente y de manera correcta.
Valoración	OK
Valoración final	OK

Tabla 48: PCN-06: Generación de nivel

PCN-07	Generación de habitaciones
Tarea	Juego parte 1 – Generación de nivel
Resultado esperado	Al generar un nivel, las habitaciones se generan en el camino y la habitación que se genera es aleatoria.
Valoración	OK
Valoración final	OK

Tabla 49: PCN-07: Generación de habitaciones

PCN-08-1	Mover personaje - Izquierda
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar el botón para moverse a la izquierda, el personaje se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 50: PCN-08-1: Mover personaje - Izquierda

PCN-08-2	Mover personaje - Derecha
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar el botón para moverse a la derecha, el personaje se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 51: PCN-08-2: Mover personaje - Derecha

PCN-08-3	Mover personaje - Saltar
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar el botón para saltar, el personaje se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 52: PCN-08-3: Mover personaje - Saltar

PCN-09-1	Recoger objetos – Pompa azul
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar una pompa azul con el personaje, ésta añade 1 segundo al tiempo y se elimina de la escena.
Valoración	OK
Valoración final	OK

Tabla 53: PCN-09-1: Recoger objetos - Pompa azul

PCN-09-2	Recoger objetos – Pompa rosa
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar una pompa rosa con el personaje, ésta añade 5 segundos al tiempo y se elimina de la escena.
Valoración	OK
Valoración final	OK

Tabla 54: PCN-09-2: Recoger objetos - Pompa rosa

PCN-09-3	Recoger objetos – Punto plateado
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar un punto plateado con el personaje, éste añade 1 punto a la puntuación y se elimina de la escena.
Valoración	OK
Valoración final	OK

Tabla 55: PCN-09-3: Recoger objetos - Punto plateado

PCN-09-4	Recoger objetos – Punto dorado
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar un punto dorado con el personaje, éste añade 3 puntos a la puntuación y se elimina de la escena.
Valoración	OK
Valoración final	OK

Tabla 56: PCN-09-4: Recoger objetos - Punto dorado

PCN-10-1	Final de partida – Por tiempo
Tarea	Juego parte 2 - Sistemas
Descripción	Al acabarse el tiempo, se finaliza la partida, apareciendo el menú de final de partida.
Valoración	OK
Valoración final	OK

Tabla 57: PCN-10-1: Final de partida - Por tiempo

PCN-10-2	Final de partida – Por lava
Tarea	Juego parte 2 - Sistemas
Descripción	Al tocar un bloque de lava con el personaje, se finaliza la partida, apareciendo el menú de final de partida.
Valoración	OK
Valoración final	OK

Tabla 58: PCN-10-2: Final de partida - Por lava

PCN-11	Reiniciar partida
Tarea	Juego parte 2 - Sistemas
Resultado esperado	Al tocar el botón play again de la pantalla de final de partida, se inicia una partida nueva.
Valoración	OK
Valoración final	OK

Tabla 59: PCN-11: Reiniciar partida

PCN-12-1	Modo debug: Mover cámara - Arriba
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para mover la cámara hacia arriba, la cámara se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 60: PCN-12-1: Modo debug: Mover cámara - Arriba

PCN-12-2	Modo debug: Mover cámara - Abajo
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para mover la cámara hacia abajo, la cámara se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 61: PCN-12-2: Modo debug: Mover cámara - Abajo

PCN-12-3	Modo debug: Mover cámara - Izquierda
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para mover la cámara hacia la izquierda, la cámara se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 62: PCN-12-3: Modo debug: Mover cámara - Izquierda

PCN-12-4	Modo debug: Mover cámara - Derecha
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para mover la cámara hacia la derecha, la cámara se desplaza en la dirección deseada.
Valoración	OK
Valoración final	OK

Tabla 63: PCN-12-4: Modo debug: Mover cámara - Derecha

PCN-12-1	Modo debug: Zoom cámara - In
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para hacer zoom in, la cámara hace zoom in.
Valoración	OK
Valoración final	OK

Tabla 64: PCN-12-1: Modo debug: Zoom cámara - In

PCN-12-2	Modo debug: Zoom cámara - Out
Tarea	Modo Debug
Resultado esperado	Al tocar el botón para hacer zoom out, la cámara hace zoom out.
Valoración	OK
Valoración final	OK

Tabla 65: PCN-12-2: Modo debug: Zoom cámara - Out

PCN-13	Modo debug: Rehacer nivel
Tarea	Modo Debug
Resultado esperado	Al tocar el botón de rehacer nivel, se genera un nuevo nivel.
Valoración	OK
Valoración final	OK

Tabla 66: PCN-13: Modo debug: Rehacer nivel

PCN-14	Modo debug: Rehacer habitaciones
Tarea	Modo Debug
Resultado esperado	Al tocar el botón de rehacer habitaciones, se generan nuevas habitaciones, manteniendo el mismo camino.
Valoración	OK
Valoración final	OK

Tabla 67: PCN-14: Modo debug: Rehacer habitaciones

PCN-15-1	Modo debug: Cambiar tamaño nivel - X
Tarea	Modo Debug
Resultado esperado	Al tocar el campo de texto de la dimensión X, aparece un teclado numérico para insertar la dimensión deseada, y al finalizar se cambia dicha dimensión. (Probado generando un nuevo nivel)
Valoración	OK
Valoración final	OK

Tabla 68: PCN-15-1: Modo debug: Cambiar tamaño nivel - X

PCN-15-2	Modo debug: Cambiar tamaño nivel - Y
Tarea	Modo Debug
Resultado esperado	Al tocar el campo de texto de la dimensión Y, aparece un teclado numérico para insertar la dimensión deseada, y al finalizar se cambia dicha dimensión. (Probado generando un nuevo nivel)
Valoración	OK
Valoración final	OK

Tabla 69: PCN-15-2: Modo debug: Cambiar tamaño nivel - Y

Capítulo 8. Conclusiones

Para finalizar nos gustaría exponer las conclusiones extraídas a lo largo de la realización de este Trabajo de Fin de Grado de acuerdo a los objetivos conseguidos y el trabajo realizado.

8.1 Conclusiones generales

Para el desarrollo de este trabajo nos planteamos aplicar los conocimientos adquiridos en la carrera de Ingeniería Informática para desarrollar un videojuego.

Gracias a estos conocimientos, hemos podido estructurar nuestro proyecto y planificar de antemano el trabajo a realizar, facilitando así en grán medida el desarrollo de éste.

Primero hemos realizado un análisis con el que hemos definido las caractéristicas que nuestra aplicación requiría.

Si bien es cierto que el desarrollo de software para un videojuego es una tarea de dimensiones inmensurables a primera vista, de manera que al empezar a indagar en el tema se podría estar perdido completamente, hemos descubierto que, efectivamente, podemos aplicar nuestros conocimientos para dicha tarea.

Tal ha sido su utilidad, que sin los modelos y patrones aprendidos a lo largo de estos años, nos habría resultado imposible desarrollar nuestra aplicación de una manera sostenible.

Por último, nos gustaría insistir, especialmente para otros posibles compañeros de nuestra titulación que puedan leer esta memoria en un futuro, que los conocimientos adquiridos a lo largo

de la titulación no son exclusivos para dicha titulación. Como bien hemos podido observar, éstos no son sólo aplicables a trabajos en empresas del sector de las tecnologías, si no que también son útiles, si no esenciales, en proyectos de otro carácter, como puede ser el desarrollo de videojuegos.

8.2 Futuras líneas de desarrollo

Pese a que esta aplicación ha sido desarrollada y presentada como Trabajo de Fin de Grado, todavía se podría seguir trabajando sobre ella.

A continuación presentamos una serie de líneas de desarrollo que aportarían un mayor valor a nuestra aplicación.

- La primera línea de desarrollo consistiría en mejorar el sistema de puntuación. Si bien el usuario puede ver la puntuación conseguida al final de cada partida, sería conveniente que nuestra aplicación fuera capaz de almacenar dicha puntuación. De esa manera el jugador podría saber cuál ha sido su mejor puntuación e intentar conseguir una mayor. Esto también abriría la posibilidad de comparar dicha puntuación con la conseguida con nuestros amigos, creando un ambiente de deportividad que incite a seguir jugando con nuestros amigos.
- Otra línea de trabajo sería incrementar la cantidad de contenido del juego. Esto puede constituir muchas cosas, como pueden ser nuevos objetos, diferentes personajes o nuevas habitaciones.

Dado que lo más duro de desarrollar son las bases de la aplicación, una vez hemos conseguido desarrollar éstas a lo largo de nuestro trabajo, sería conveniente hacer uso de dichas bases para expandir nuestra aplicación de manera rápida y sencilla.

Otra opción a tener muy en cuenta es la posibilidad de desarrollar aplicaciones completamente diferentes haciendo uso del trabajo realizado. Gracias a la modularidad aplicada podemos traspasar nuestro trabajo fácilmente a otros proyectos de manera muy sencilla, de manera que el desarrollo de nuevas aplicaciones sea más ágil.

Anexos

A continuación describiremos cómo instalar el juego en nuestro dispositivo Android y describiremos cómo utilizarlo, apoyándonos en imágenes para una más fácil comprensión.

A. Manual de Instalación

A.1 Requisitos

- Dispositivo Android
- Versión Android 4.1 (Jelly Bean, API level 16) o superior

A.2 Instalación

Para instalar el juego en nuestro dispositivo Android, tendremos que seguir los siguientes pasos:

- Mover el archivo Maze.apk a nuestro dispositivo. Una forma de hacer esto es conectando nuestro dispositivo a un ordenador mediante USB y permitiendo al dispositivo transferir archivos.
- Ejecutar el archivo. Esto se puede hacer con exploradores de archivos.
- Nuestro dispositivo podría pedirnos permiso para instalar la aplicación debido a que no viene de la tienda oficial de Android. En dicho caso, le daremos permiso.
- La aplicación quedará instalada en nuestro dispositivo y podremos ejecutarla tocando su icono.

B. Manual de Usuario

B.1. Menú principal

Desde el menú principal podemos acceder a los 2 modos de la aplicación.

- **Modo Juego:** En este modo se puede jugar a nuestro juego.
- **Modo Debug:** En este modo se pueden generar niveles aleatorios y observar el resultado.

Ilustración 42: Menú principal

Para acceder al modo juego pulsaremos en el cartel de “play”.
Para acceder al modo debug pulsaremos el icono del bicho (Comúnmente conocido como el icono de debug)

B.2. Juego

Una vez entramos en una partida, el jugador podrá mover a su personaje para conseguir puntos.

Para volver al menú principal, se debe tocar el botón en la esquina superior izquierda.

El jugador se podrá mover horizontalmente con los botones situados en la esquina inferior izquierda y saltar verticalmente con el botón situado en la esquina inferior derecha.

El juego se acabará si se agota el tiempo o si el personaje toca un bloque de lava.

El tiempo está representado por una barra en la esquina superior derecha. Debajo de esta se encuentra el contador de puntos.

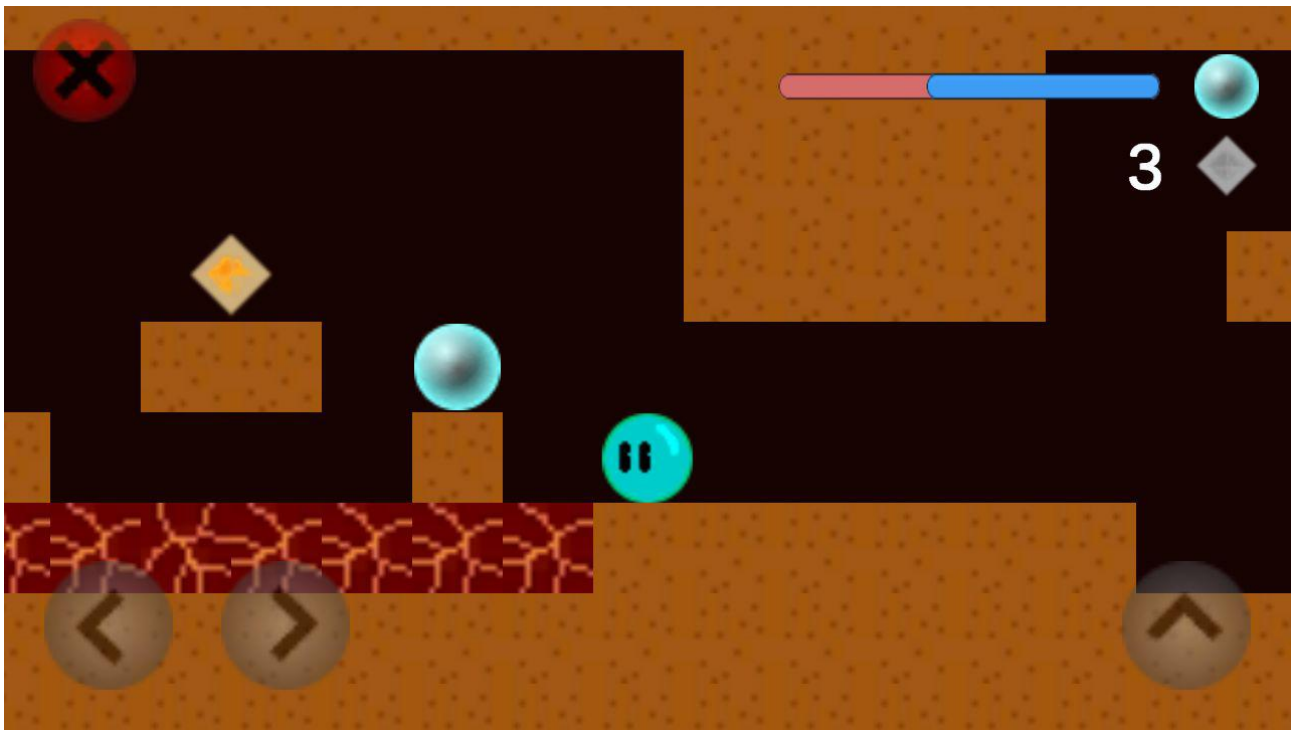

Ilustración 43: Partida del modo juego

El jugador puede recoger los siguientes objetos:

- **Pompa azul:** Incrementa el tiempo en 1 segundo.
- **Pompa Rosa:** Incrementa el tiempo en 5 segundo.
- **Punto plateado:** Incrementa en 1 los puntos.
- **Punto dorado:** Incrementa en 3 los puntos.

B.2.1 Pantalla de fin de juego

Una vez se acaba una partida, se mostrará la pantalla de fin de juego. En esta pantalla se puede ver la puntuación conseguida en dicha partida y se dan dos opciones al jugador. Volver al menú principal, de la misma manera que en la pantalla de juego, o volver a jugar una partida, mediante el botón de “play again”.

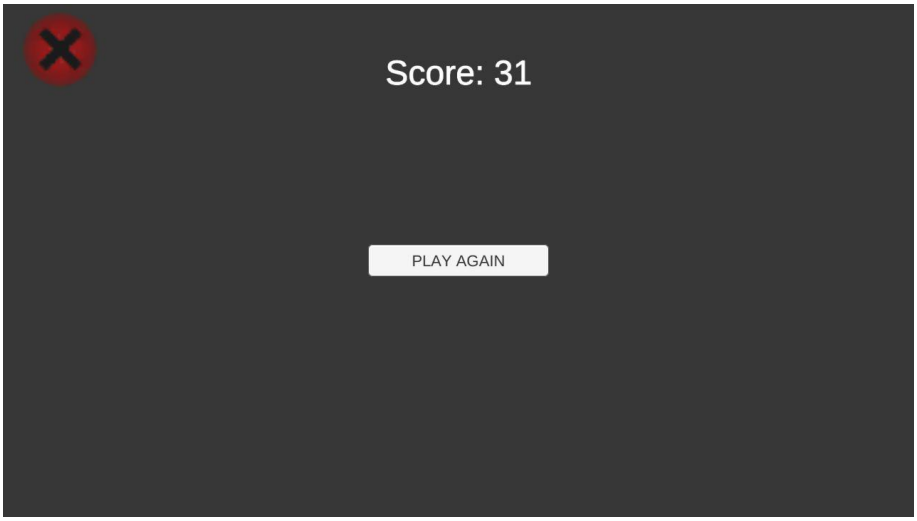

Ilustración 44: Pantalla de fin de juego

B.3. Modo debug

En este modo se pueden generar niveles y observar lo que generan.

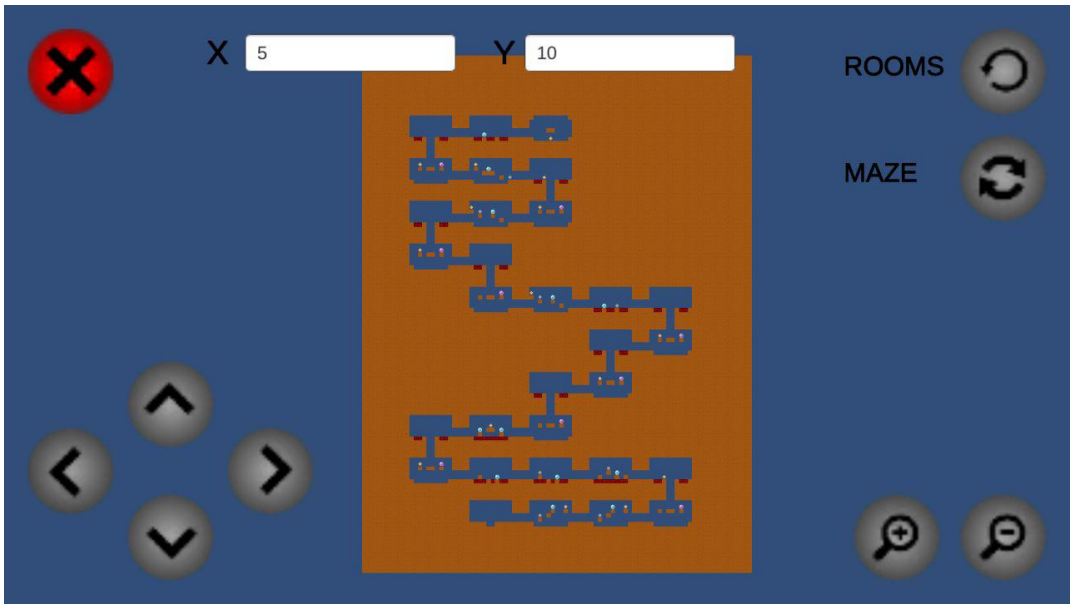

Ilustración 45: Modo debug

De nuevo, podemos volver al menú principal con el botón de la esquina superior izquierda.

Para poder observar el nivel, podemos mover la cámara con los botones en la esquina inferior izquierda y hacer zoom con los botones de la esquina inferior derecha.

Podemos seleccionar el tamaño del nivel a generar con los campos de texto en la parte superior, siendo X la cantidad de salas horizontalmente e Y la cantidad de salas verticalmente.

Para generar un nivel entero presionaremos el botón de “maze” (Para tamaños grandes esto puede tardar un tiempo)

Para cambiar las salas manteniendo el camino que estas forman, presionaremos el botón “rooms”.

B.4. Atajos de teclado

Para facilitar el uso de la aplicación en ordenadores, también tenemos a nuestra disposición los siguientes atajos:

Modo Juego:

- **Movimiento horizontal:** Flechas izquierda/derecha
- **Saltar:** Flecha arriba / Barra espaciadora

Modo Debug:

- **Movimiento de la cámara:** Flechas
- **Zoom In:** Z
- **Zoom Out:** X

Bibliografía

A continuación enumeramos las fuentes de información de las que hemos hecho uso a lo largo del transcurso de nuestro trabajo. Indicamos, donde aplique, autor, nombre de la publicación, año de publicacióm, título, fuente, y fecha y enlace del último acceso.

Índice bibliográfico

[1] Gibson, J. (2014). Introduction to Game Design, Prototyping and Developing.

[2] Nystrom, R. (2014). Game Programming Patterns

[3] Yu, D. (2016). Spelunky (Boss Fight Books Book 11)

[4] Collider2D.OnTriggerEnter2D(Collider2D) (2019). docs.unity3d.com. Último acceso 3 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/Collider2D.OnTriggerEnter2D.html

[5] Tile (2019). docs.unity3d.com. Último acceso 3 de Febrero de 2019, desde https://docs.unity3d.com/Manual/Tilemap-ScriptableTiles-Tile.html

[6] Introduction to the New Unity 2D Tilemap System (2019). www.raywenderlich.com. Último acceso 3 de Febrero de 2019, desde https://www.raywenderlich.com/23-introduction-to-the-new-unity-2d-tilemap-system

[7] How to make cool scene transitions in Unity (2018) Blackthornprod. www.youtube.com. Último acceso 3 de Febrero de 2019, desde https://www.youtube.com/watch?v=Qd2em_ts5vs

[8] Tilemap (2019). docs.unity3d.com. Último acceso 3 de Febrero de 2019, desde https://docs.unity3d.com/Manual/class-Tilemap.html

[9] Tilemap (2019). docs.unity3d.com. Último acceso 3 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/Tilemaps.Tilemap.html

[10] Custom Tilemap Brush To Spawn Prefabs (2019). unity3d.com. Último acceso 4 de Febrero de 2019, desde https://unity3d.com/learn/tutorials/topics/2d-game-creation/custom-tilemap-brush-spawn-prefabs

[11] Data Classes (2019). docs.unity3d.com. Último acceso 4 de Febrero de 2019, desde https://unity3d.com/learn/tutorials/topics/scripting/data-classes

[12] Rigidbody 2D - Official Unity Tutorial (2014) Unity. www.youtube.com. Último acceso 5 de Febrero de 2019, desde https://www.youtube.com/watch?v=rq6c2B_socs

[13] Better Jumping in Unity With Four Lines of Code (2017) Board to Bits Games. docs.unity3d.com. Último acceso 5 de Febrero de 2019, desde https://www.youtube.com/watch?v=7KiK0Aqtmzc

[14] How to make cool scene transitions in Unity (2018) Blackthornprod. docs.unity3d.com. Último acceso 5 de Febrero de 2019, desde https://www.youtube.com/watch?v=Qd2em_ts5vs

[15] Several signals of wood texture Vector (2019). www.freepik.com. Último acceso 6 de Febrero de 2019, desde https://www.freepik.com/free-vector/several-signals-wood-texture_933701.htm#page=1&index=30&query=wood%20sign

[16] How do you make an object respond to a click in C# (2019). answers.unity.com. Último acceso 6 de Febrero de 2019, desde https://answers.unity.com/questions/332085/how-do-you-make-an-object-respond-to-a-click-in-c.html

[17] Inspector Field for Scene Asset (2019). answers.unity.com. Último acceso 7 de Febrero de 2019, desde https://answers.unity.com/questions/242794/inspector-field-for-scene-asset.html

[18] Canvas (2019). docs.unity3d.com. Último acceso 7 de Febrero de 2019, desde https://docs.unity3d.com/Manual/UICanvas.html

[19] Adding The Game Controller (2019). unity3d.com. Último acceso 8 de Febrero de 2019, desde https://unity3d.com/learn/tutorials/topics/2d-game-creation/adding-game-controller

[20] Unity with MVC: How to Level Up Your Game Development (2019). www.toptal.com. Último acceso 18 de Febrero de 2019, desde https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development

[21] Calling function from other scripts c# (2019). forum.unity.com. Último acceso 21 de Febrero de 2019, desde https://forum.unity.com/threads/calling-function-from-other-scripts-c.57072/

[22] MonoBehaviour (2019). docs.unity3d.com. Último acceso 21 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[23] GameObject.FindGameObjectsWithTag (2019). docs.unity3d.com. Último acceso 22 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/GameObject.FindGameObjectsWithTag.html

[24] Debug.Log (2019). docs.unity3d.com. Último acceso 22 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/Debug.Log.html

[25] Input.GetAxis (2019). docs.unity3d.com. Último acceso 22 de Febrero de 2019, desde https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

[26] Holding down UI mobile button (2019). docs.unity3d.com. Último acceso 22 de Febrero de 2019, desde https://answers.unity.com/questions/1262535/holding-down-ui-mobile-button.html

[27] Landscape mode only (2019). answers.unity.com Último acceso 24 de Febrero de 2019, desde https://answers.unity.com/questions/774186/landscape-mode-only.html

[28] Designing UI for Multiple Resolutions (2019). docs.unity3d.com. Último acceso 24 de Febrero de 2019, desde https://docs.unity3d.com/Manual/HOWTO-UIMultiResolution.html

[29] Building your Unity game to an Android device for testing (2019). docs.unity3d.com. Último acceso 24 de Febrero de 2019, desde https://unity3d.com/learn/tutorials/topics/mobile-touch/building-your-unity-game-android-device-testing

[30] Game Programming Patterns (2019). www.gameprogrammingpatterns.com. Último acceso 30 de Marzo de 2019, desde http://www.gameprogrammingpatterns.com/contents.html

[31] Game programming patterns in Unity with C# - Command Pattern (2019). www.habrador.com. Último acceso 1 de Abril de 2019, desde https://www.habrador.com/tutorials/programming-patterns/1-command-pattern/

[32] Input (2019). docs.unity3d.com. Último acceso 18 de Abril de 2019, desde https://docs.unity3d.com/ScriptReference/Input.html

[33] Lists and Dictionaries (2019). docs.unity3d.com. Último acceso 18 de Abril de 2019, desde https://unity3d.com/learn/tutorials/modules/intermediate/scripting/lists-and-dictionaries

[34] Object.Destroy (2019). docs.unity3d.com. Último acceso 18 de Abril de 2019, desde https://docs.unity3d.com/ScriptReference/Object.Destroy.html

[35] Random.Range (2019). docs.unity3d.com. Último acceso 18 de Abril de 2019, desde https://docs.unity3d.com/ScriptReference/Random.Range.html

[36] Deleting a game object from within itself (2019). docs.unity3d.com. Último acceso 22 de Abril de 2019, desde https://forum.unity.com/threads/deleting-a-game-object-from-within-its-self.92840/

[37] Roll a Ball game: Displaying Score and Text (2015) Unity. www.youtube.com. Último acceso 2 de Mayo de 2019, desde https://www.youtube.com/watch?v=bFSLI2cmYYo

[38] Zoom out in orthographic view? (2019). answers.unity.com. Último acceso 2 de Mayo de 2019, desde https://answers.unity.com/questions/15252/zoom-out-in-orthographic-view.html

[39] Game Over (2019). unity3d.com. Último acceso 5 de Mayo de 2019, desde https://unity3d.com/learn/tutorials/projects/survival-shooter/game-over

[40] Interfaces (2019). unity3d.com. Último acceso 5 de Mayo de 2019, desde https://unity3d.com/learn/tutorials/topics/scripting/interfaces

[41] Enumerations (2019). learn.unity.com Último acceso 5 de Mayo de 2019, desde https://learn.unity.com/tutorial/enumerations?projectId=5c8920b4edbc2a113b6bc26a#5c8a6ee6edbc2a067d47537b

[42] switch (C# reference) (2019). docs.microsoft.com. Último acceso 5 de Mayo de 2019, desde https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch

[43] Camera.orthographicSize (2019). docs.unity3d.com. Último acceso 7 de Mayo de 2019, desde https://docs.unity3d.com/ScriptReference/Camera-orthographicSize.html

[44] Player Health (2019). docs.unity3d.com. Último acceso 7 de Mayo de 2019, desde https://unity3d.com/learn/tutorials/projects/survival-shooter/player-health

[45] RigidbodyConstraints2D (2019). docs.unity3d.com. Último acceso 7 de Mayo de 2019, desde https://docs.unity3d.com/ScriptReference/RigidbodyConstraints2D.html

[46] Object.Instantiate (2019). docs.unity3d.com. Último acceso 7 de Mayo de 2019, desde https://docs.unity3d.com/ScriptReference/Object.Instantiate.html

[47] How can I add script to a single tile? (2019). answers.unity.com. Último acceso 11 de Mayo de 2019, desde https://answers.unity.com/questions/1510435/how-can-i-add-script-to-a-single-tile.html

[48] Tilemap: Prefab Brush (2018) Unity. www.youtube.com. Último acceso 11 de Mayo de 2019, desde https://www.youtube.com/watch?v=UqhK6GpCgrM

[49] Tilemap Collider 2D and Composite Collider 2D (2019). unity3d.com. Último acceso 11 de Mayo de 2019, desde https://unity3d.com/learn/tutorials/topics/2d-game-creation/tilemap-collider-2d-and-composite-collider-2d?playlist=17093

[50] Fa2png.io (2019). fa2png.io. Último acceso 15 de Mayo de 2019, desde http://fa2png.io/

[51] 2D Animation in Unity (2018) Brackeys. www.youtube.com. Último acceso 15 de Mayo de 2019, desde https://www.youtube.com/watch?v=hkaysu1Z-N8

[52] Rotating exactly 90 degrees - specific direction (2019). forum.unity.com. Último acceso 18 de Mayo de 2019, desde https://forum.unity.com/threads/rotating-exactly-90-degrees-specific-direction-answered.44056/

[53] Allow only numbers to a textfield, and make it an int (2019). answers.unity.com. Último acceso 21 de Mayo de 2019, desde https://answers.unity.com/questions/867528/allow-only-numbers-to-a-textfield-and-make-it-an-i.html

[54] How would I get the text input from a text field into a variable? (2019). forum.unity.com. Último acceso 21 de Mayo de 2019, desde https://forum.unity.com/threads/how-would-i-get-the-text-input-from-a-text-field-into-a-variable.329821/

[55] Line Wobbler. wobblylabs.com. Último acceso 20 de Mayo de 2019, desde https://wobblylabs.com/projects/wobbler

[56] Pong. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Pong

[57] Space Invaders. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Space_Invaders

[58] Donkey Kong (video game). en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Donkey_Kong_(video_game)

[59] Super Meat Boy. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Super_Meat_Boy

[60] Super Mario Maker. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Super_Mario_Maker

[61] No Man's Sky. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/No_Man%27s_Sky

[62] Diablo III. en.wikipedia.org. Último acceso 20 de Mayo de 2019, desde https://en.wikipedia.org/wiki/Diablo_III

[63] El mercado del videojuego bate récords en España con un crecimiento del 12,6% y una facturación de 1.530 millones en 2018 (2019). http://www.aevi.org.es. Último acceso 28 de Junio de 2019, desde http://www.aevi.org.es/mercado-del-videojuego-bate-records-espana-crecimiento-del-126-una-facturacion-1-530-millones-2018/

[64] Space Panic. en.wikipedia.org. Último acceso 1 de Julio de 2019, desde https://en.wikipedia.org/wiki/Space_Panic

[65] Super Mario Bros. en.wikipedia.org. Último acceso 1 de Julio de 2019, desde https://en.wikipedia.org/wiki/Super_Mario_Bros.

[66] Nintendo Entertainment System. en.wikipedia.org. Último acceso 1 de Julio de 2019, desde https://en.wikipedia.org/wiki/Nintendo_Entertainment_System

[67] Kirby's Dream Land. en.wikipedia.org. Último acceso 1 de Julio de 2019, desde https://en.wikipedia.org/wiki/Kirby%27s_Dream_Land

[68] Kid Dracula. en.wikipedia.org. Último acceso 1 de Julio de 2019, desde https://en.wikipedia.org/wiki/Kid_Dracula_(1993_video_game)

[69] Super Mario 64. en.wikipedia.org. Último acceso 2 de Julio de 2019, desde https://en.wikipedia.org/wiki/Super_Mario_64

[70] Rayman Origins. en.wikipedia.org. Último acceso 2 de Julio de 2019, desde https://en.wikipedia.org/wiki/Rayman_Origins

[71] Sonic Generations. en.wikipedia.org. Último acceso 2 de Julio de 2019, desde https://en.wikipedia.org/wiki/Sonic_Generations

[72] Rogue Legacy. en.wikipedia.org. Último acceso 2 de Julio de 2019, desde https://en.wikipedia.org/wiki/Rogue_Legacy

[73] The Binding of Isaac. en.wikipedia.org. Último acceso 2 de Julio de 2019, desde https://en.wikipedia.org/wiki/The_Binding_of_Isaac_(video_game)

[74] Rubén Moya Vázquez. (2011). TFG: Aplicación del juego La Colmena mediante Unreal Engine 4

[75] Pablo de la Fuente Redondo. (2011). Transparencias asignatura Planificación y Gestión de Plataformas Informáticas

[76] Videojuego de Plataformas. es.wikipedia.org. Último acceso 4 de Julio de 2019, desde https://es.wikipedia.org/wiki/Videojuego_de_plataformas

[77] Roguelike. es.wikipedia.org. Último acceso 4 de Julio de 2019, desde https://es.wikipedia.org/wiki/Roguelike

[78] Generador de números pseudoaleatorios. es.wikipedia.org. Último acceso 4 de Julio de 2019, desde https://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios

[79] Kirby's Dream Land Part 1: Green Greens (2008). www.youtube.com. Último acceso 8 de Julio de 2019, desde https://www.youtube.com/watch?v=-EAw3CGscwg

[80] Super Mario 64 turns 20 and is still one of the greatest 3D games ever (2016) Unity. venturebeat.com. Último acceso 8 de Julio de 2019, desde https://venturebeat.com/2016/06/23/super-mario-64-turns-20-and-is-still-one-of-the-greatest-3d-games-ever/

[81] World 1 (New Super Mario Bros.). mario.fandom.com. Último acceso 8 de Julio de 2019, desde https://mario.fandom.com/wiki/World_1_(New_Super_Mario_Bros.)

[82] Review: Super Meat Boy (2016) kotaku.com. Último acceso 8 de Julio de 2019, desde https://kotaku.com/review-super-meat-boy-5674516

[83] Super Mario Maker. www.dlcompare.com. Último acceso 8 de Julio de 2019, desde https://www.dlcompare.com/games/100000045/digital-mario-maker-key-retail

[84] World Without End (2015) www.newyorker.com. Último acceso 8 de Julio de 2019, desde https://www.newyorker.com/magazine/2015/05/18/world-without-end-raffi-khatchadourian

[85] How To Get Tons of Loot in Diablo III's New Dungeons (2014) kotaku.com. Último acceso 8 de Julio de 2019, desde https://kotaku.com/how-to-get-tons-of-loot-in-diablo-iiis-new-dungeons-1629991701

[86] Rogue Legacy. www.playstation.com. Último acceso 8 de Julio de 2019, desde https://www.playstation.com/en-us/games/rogue-legacy-ps4/

[87] Insane Item Rooms. moddingofisaac.com. Último acceso 8 de Julio de 2019, desde https://moddingofisaac.com/mod/886/insane-item-rooms-afterbirth