



TRABAJO FIN DE GRADO

Manual técnico

Grado en Ingeniería Informática

Stealthy Spy, un juego de plataformas para dispositivos Android

Autor: Alberto Adamuz Priego

Directores: Juan Carlos Fernández Caballero

David Guijo Rubio

6 de junio de 2023



UNIVERSIDAD DE CÓRDOBA

Firma de conformidad del autor y del director del trabajo de fin de grado:

- **Autor:**

Fdo: Alberto Adamuz Priego

- **Directores:**

Fdo: Juan Carlos Fernández Caballero

Fdo: David Guijo Rubio

Índice de contenidos

I	Introducción	1
1	Introducción	3
2	Definición del problema	1
2.1	Definición del problema real	1
2.2	Definición del problema técnico	1
2.2.1	Funcionamiento	2
2.2.2	Entorno	2
2.2.3	Vida esperada	2
2.2.4	Ciclo de mantenimiento	2
2.2.5	Competencia	3
2.2.6	Aspecto externo	3
2.2.7	Estandarización	3
2.2.8	Calidad y fiabilidad	4
2.2.9	Fases de desarrollo	4
2.2.9.1	Análisis del problema	5
2.2.9.2	Diseño	5
2.2.9.3	Codificación	5
2.2.9.4	Pruebas	5
2.2.9.5	Distribución	5
2.2.9.6	Documentación	6
2.2.9.7	Planificación temporal	6
3	Objetivos	7
3.1	Objetivos formales	7

Índice de contenidos

3.2	Objetivos operacionales	8
4	Antecedentes	11
4.1	Origen de los videojuegos	11
4.2	Videojuegos de plataformas	13
5	Restricciones	17
5.1	Factores dato	17
5.2	Factores estratégicos	18
6	Recursos	21
6.1	Recursos humanos	21
6.2	Recursos software	22
6.3	Recursos hardware	22
II	Análisis del sistema y especificación de requisitos	23
7	Especificación de Requisitos	25
7.1	Identificación de los bloques funcionales	25
7.1.1	Bloque de construcción del escenario	26
7.1.2	Bloque de manejo del jugador	26
7.1.3	Bloque de gestión del sistema	27
7.1.4	Bloque de gestión de la interfaz	27
7.2	Requisitos funcionales	28
7.3	Requisitos de información	29
7.4	Requisitos no funcionales	29
7.5	Requisitos de la interfaz	30
7.6	Matriz de rastreabilidad	31
8	Análisis funcional	33
8.1	Usuarios del sistema	33

8.2 Análisis de casos de uso	34
8.2.1 Diagrama de contexto	34
8.2.2 Caso de uso 1. Comenzar juego	35
8.2.3 Caso de uso 1.1. Seleccionar dificultad	36
8.2.4 Caso de uso 1.2. Seleccionar mapa	37
8.2.5 Caso de uso 1.3. Pausar	38
8.2.6 Caso de uso 2. Configurar juego	39
8.2.7 Caso de uso 2.1. Ajustar volumen música	40
8.2.8 Caso de uso 2.2. Ajustar volumen efectos de sonido	41
8.2.9 Caso de uso 3. Ver puntuaciones	41
8.2.10 Caso de uso 4. Ver créditos	42
III Diseño del sistema	43
9 Arquitectura del sistema	45
10 Diseño de la interfaz	51
10.1 Menú principal	52
10.2 Menú de puntuaciones	53
10.3 Menú de configuración	54
10.4 Menú de selección de mapas	55
10.5 Interfaz del juego	56
10.6 Menú de pausa	57
10.7 Menú de fin de partida	58
10.8 Menú de créditos	60
IV Pruebas	61
11 Pruebas	63
11.1 Pruebas de aceptación	63

Índice de contenidos

11.1.1 Caso de prueba sobre CU-1, “Comenzar juego”	64
11.1.2 Caso de prueba sobre CU-1.1, “Seleccionar dificultad”	64
11.1.3 Caso de prueba sobre CU-1.3, “Pausar”	65
11.1.4 Caso de prueba sobre CU-2.1, “Ajustar volumen de la música”	65
11.1.5 Caso de prueba sobre CU-2.2, “Ajustar volumen de los efectos de sonido”	66
11.1.6 Caso de prueba sobre CU-3, “Ver puntuaciones”	66
11.1.7 Caso de prueba sobre la interfaz de la aplicación	67
11.1.7.1 Caso de prueba sobre los menús de la aplicación	67
11.2 Pruebas de integración	68
11.2.1 Caso de prueba 1	68
11.2.2 Caso de prueba 2	68
11.2.3 Caso de prueba 3	69
11.2.4 Caso de prueba 4	69
11.3 Pruebas de usuario	69
V Conclusiones y futuras mejoras	71
12 Conclusiones y futuras mejoras	73
12.1 Conclusiones sobre los objetivos formales	73
12.2 Conclusiones sobre los objetivos operacionales	74
12.3 Futuras mejoras	75
Bibliografía	75
VI Apéndice	81
A Manual de usuario	83
A.1 Estructura del manual	83
A.2 Requisitos del sistema	83

A.3 Instalación de la aplicación	84
A.4 Desinstalación de la aplicación	86
A.4.1 Guía de uso	89
A.4.1.1 Menú principal	89
A.4.1.2 Menú de puntuaciones	90
A.4.1.3 Menú de configuración	91
A.4.1.4 Menú de selección de mapas	92
A.5 Interfaz del juego	93
A.6 Menú de pausa	94
A.7 Menú de fin de partida	95
A.8 Menú de créditos	97

Índice de figuras

1.1	Super Mario Bros.	5
4.1	OXO. Primer videojuego creado	12
4.2	Tenis for Two	12
4.3	Spacewar!	13
4.4	Donkey Kong	14
4.5	Super Mario 64	15
4.6	Plataformas actuales	16
8.1	CU-0. Diagrama de contexto	34
8.2	CU-1. Comenzar juego	35
8.3	CU-2. Configurar juego	39
9.1	<i>Animator</i>	47
9.2	<i>Tile Palette</i>	48
9.3	Rejilla	48
9.4	<i>Camera prefab</i>	49
10.1	Menú principal	52
10.2	Menú de puntuaciones	53
10.3	Menú de configuración	54
10.4	Menú de selección de mapas	55
10.5	Interfaz del juego	56
10.6	Menú de pausa	57
10.7	Menú fin de partida 1	58
10.8	Menú fin de partida 2	59
10.9	Menú de créditos	60

Índice de figuras

A.1	Ventana de aviso por archivo dañino	84
A.2	Ventana de aviso por origen desconocido	85
A.3	Ventana de autorización de <i>Android</i>	85
A.4	Ventana de instalación	86
A.5	Stealthy Spy instalado	86
A.6	Desinstalar aplicación 1	87
A.7	Desinstalar aplicación 2	88
A.8	Menú principal	89
A.9	Menú de puntuaciones	90
A.10	Menú de configuración	91
A.11	Menú de selección de mapas	92
A.12	Interfaz del juego	93
A.13	Menú de pausa	94
A.14	Menú fin de partida 1	95
A.15	Menú fin de partida 2	96
A.16	Menú de créditos	97

Índice de tablas

2.1	Planificación temporal	6
7.1	Matriz de rastreabilidad	31

Parte I

Introducción

CAPÍTULO 1

INTRODUCCIÓN

La industria de los videojuegos se ha convertido en una de las industrias más grandes e importantes de la actualidad [1]. Este mercado ha ido creciendo exponencialmente y evolucionando hasta el punto de formar parte de la cultura popular, superando a otros grandes mercados de entretenimiento como la televisión o el cine.

Gran parte de este crecimiento es gracias a los llamados videojuegos *indie* o independientes [2]. Este tipo de videojuegos son creados por estudios pequeños o incluso por una sola persona. Durante los últimos años una gran cantidad de desarrolladores independientes se están dedicando a lanzar gran variedad de videojuegos, que pese a no tener un gran presupuesto o recursos como los videojuegos desarrollados por empresas, están consiguiendo hacerse hueco en el mercado [3] llegando a competir con grandes competidores como Nintendo o Sony.

Las causas de este crecimiento se deben principalmente a los nuevos métodos de distribución en línea y herramientas de desarrollo que permiten a los desarrolladores

1. Introducción

trabajar en videojuegos sin disponer de un gran equipo ni de un gran presupuesto. Además, la gran mayoría de este tipo de videojuegos se encuentran disponibles para móviles, dispositivos que prácticamente cualquier persona posee, lo que facilita su distribución al público.

La herramienta más potente a la hora de desarrollar un videojuego son los motores de videojuegos [4]. Un motor de videojuego es un término que hace referencia a una serie de librerías de programación que permiten el diseño, la creación y la representación de un videojuego. Muchos de ellos son gratuitos y ofrecen todo lo necesario para poder crear nuestro propio videojuego. Son tan populares y útiles que la mayoría de las grandes empresas deciden usarlos en vez de crear los suyos propios. Los mejores ejemplos son *Unity* [5] y *Unreal Engine* [6], dos motores con los que se han desarrollado varios de los juegos más populares y vendidos de la actualidad.

Cuando hablamos del desarrollo de videojuegos, además de la tecnología a utilizar y de la plataforma, es necesario hablar del género del videojuego [7]. Los géneros son categorías que se utilizan para clasificar y organizar a los videojuegos en función de las características claves de su jugabilidad. Este sistema se basa en las coincidencias que existen entre distintos videojuegos por su jugabilidad básica y se ha ido formando a medida que fueron surgiendo numerosos juegos con características en común que permitían agruparlos. Existen una enorme cantidad de géneros diferentes y muchos videojuegos pueden contener varios de estos géneros.

Uno de los géneros de videojuegos más populares son los de plataformas [8]. Este género se caracteriza por tener que caminar y saltar por plataformas, superando toda clase de obstáculos para llegar a una meta. También se caracteriza por una vista de desplazamiento horizontal hacia la izquierda o a la derecha. Es uno de los primeros géneros que se popularizó a comienzos de la década de 1980 y sigue estando muy presente en la

actualidad. En la figura 1.1 se muestra el videojuego que causó la gran explosión de este género, el famoso *Super Mario Bros* [9], lanzado en 1985 por Nintendo para la consola NES.



Figura 1.1: Super Mario Bros.

Con la continuidad y popularidad del género de plataformas, para este Trabajo Fin de Grado (TFG) se ha desarrollado un videojuego de dicho género para sistemas Android, ya que es una plataforma muy popular y de fácil acceso.

CAPÍTULO 2

DEFINICIÓN DEL PROBLEMA

En este capítulo se detallan las dos visiones del problema a resolver, el problema real y el problema técnico.

2.1. Definición del problema real

Este Trabajo de Fin de Grado (TFG) consiste en el diseño e implementación de un videojuego que funcione en dispositivos con sistema operativo Android. El videojuego consistirá en un género de plataformas 2D. El género de plataformas se caracteriza por la presencia de un personaje en 2D que puede desplazarse por diversos obstáculos para conseguir un determinado objetivo.

2.2. Definición del problema técnico

A continuación, se definen los condicionantes del problema técnico.

2. Definición del problema

2.2.1. Funcionamiento

El software a desarrollar basará su funcionamiento en las siguientes funcionalidades:

- Permitirá el control en 2D de un personaje dentro de varios mapas de plataformas.
- Permitirá un modo de selección de dificultad para los distintos mapas.
- Proporcionará un sistema para evitar o confrontar a los enemigos.
- Se mantendrá un registro de los mejores tiempos obtenidos en cada mapa según la dificultad.
- El juego se adaptará a la resolución de cualquier dispositivo Android.

2.2.2. Entorno

El videojuego estará disponible para cualquier dispositivo *Android* con una versión igual o superior a la 5.1. Esto permitirá que casi cualquier dispositivo pueda soportar el juego.

2.2.3. Vida esperada

La vida esperada depende del ciclo de mantenimiento y/o de posibles restricciones de software que puedan surgir en el futuro en la plataforma *Android* donde se instale.

2.2.4. Ciclo de mantenimiento

El software final será completamente funcional, constituyendo una versión definitiva del proyecto que se pretende desarrollar.

El desarrollo del mismo se realizará de tal forma que sean posibles futuras mejoras y ampliaciones sin que ello signifique un trabajo costoso para su realización.

2.2.5. Competencia

Existen en el mercado una enorme multitud de videojuegos de plataformas para dispositivos móviles, tanto de empresas grandes como de estudios pequeños. Sin embargo, el juego desarrollado en este TFG no pretende competir en el mercado, sino más bien servir de formación para el proyectista en la creación de un videojuego desde el inicio, a la vez que demostrar las competencias que se han adquirido durante sus estudios de grado.

2.2.6. Aspecto externo

El videojuego será presentado en un archivo .apk listo para su ejecución en *Android*. También se incluirá un manual técnico y de usuario, además del manual de código fuente del mismo.

Otra posibilidad es la distribución del juego de manera gratuita en *Google Play*, bajo sus políticas de privacidad, protección de datos y condiciones de uso.

2.2.7. Estandarización

La aplicación procurará seguir de la forma más fiel posible los estándares relacionados con los siguientes aspectos:

- **Diseño.** Se hará uso del paradigma de programación orientada a eventos.

También se seguirán estilos de identificación, de espaciado y convenciones de nombramiento para mantener un código legible y de fácil mantenimiento.

- **Implementación.** Se usará el lenguaje de programación estándar para el motor de videojuegos *Unity*, *C#*.

- **Documentación.** Uso de estándares en el desarrollo de la documentación, como en las referencias bibliográficas.

2.2.8. Calidad y fiabilidad

La calidad y la fiabilidad son dos factores fundamentales en el desarrollo, ya que de ellos depende el buen desempeño de la aplicación. Aquí se definen los aspectos a tener en cuenta a la hora del diseño y desarrollo de la misma:

- **Jugabilidad.** La jugabilidad es un factor determinante en un videojuego. El principal objetivo debe ser divertir y entretener de forma satisfactoria y creíble al usuario y una mala jugabilidad puede ser motivo suficiente para desechar el juego. Deben cuidarse detalles como la facilidad de uso, el diseño de una dificultad equilibrada, que no llegue a aburrir pero tampoco a desesperar al jugador, y el grado de satisfacción al cumplir los objetivos planteados.
- **Herramientas.** Se tendrán en cuenta las herramientas cuyas características y funcionalidades se adapten para el correcto desarrollo de la aplicación. En el capítulo 6 se ampliará la información aquí mencionada.
- **Errores y *bugs*.** La aplicación deberá controlar los distintos fallos que puedan ocurrir durante la ejecución de la misma. Por otro lado, los *bugs* se detectarán y solucionarán mediante un conjunto extensivo de pruebas durante el desarrollo y a la finalización del mismo para asegurar la corrección y consistencia de la aplicación. En el capítulo 11 se ampliará la información aquí mencionada.

2.2.9. Fases de desarrollo

A continuación se exponen las fases de desarrollo para este TFG, dividido en "Análisis del problema", "Diseño", "Codificación", "Pruebas", "Distribución" y "Documentación", para terminar con la planificación temporal prevista para llevarlo a cabo.

2.2.9.1. Análisis del problema

Se realizará un estudio previo sobre los videojuegos de plataformas más importantes del género para entender el funcionamiento básico del género.

Junto a este estudio previo, se realizará otro a modo de formación sobre las fases o requerimientos necesarios para realizar un videojuego en Android.

2.2.9.2. Diseño

Se realizarán unos diseños preliminares que permitan especificar el estilo artístico, el ciclo de juego, las mecánicas principales y secundarias, etc.

También se diseñará el conjunto de clases y rutinas necesarias para el videojuego.

2.2.9.3. Codificación

Se implementará usando las herramientas más apropiadas junto con el lenguaje de programación apropiado. El lector puede consultar más sobre ello en el capítulo 6.

2.2.9.4. Pruebas

Las pruebas se realizarán con distintos usuarios. Se irán reportando los fallos y mejoras que se detecten durante la fase de desarrollo y prototipado, para así garantizar la robustez, corrección y reusabilidad perseguidas por el proyecto. Una vez finalizada la fase de desarrollo se seguirán realizando distintas pruebas para asegurar estos objetivos. En el capítulo 11 se describen de manera más concreta algunos ejemplos sobre las pruebas realizadas.

2.2.9.5. Distribución

Se planteará la posibilidad de distribuirlo a través de *Google Play* bajo sus políticas de privacidad, protección de datos y condiciones de uso. Alternativamente, se puede

2. Definición del problema

distribuir bajo un fichero .apk.

2.2.9.6. Documentación

Esta fase comprende la redacción de toda la documentación relativa al proyecto (memoria):

- **Manual técnico.** Recoge la descripción y justificación de la solución desarrollada. Será ampliado conforme se progrese por las distintas fases del desarrollo.
- **Manual de usuario.** En él se indicará a los usuarios cómo manejar la aplicación y los distintos módulos que la componen.
- **Manual de código.** El código fuente desarrollado estará documentado para una mejor comprensión.

2.2.9.7. Planificación temporal

Aquí se detalla la planificación del proyecto. La duración total del proyecto sería aproximadamente de unas 300 horas distribuidas en 20 semanas, correspondiendo a cada una unas 15 horas de trabajo.

En la figura 2.1 se muestra la estimación de horas para cada fase de desarrollo.

Fase de Desarrollo	Horas	Suma
Análisis del problema	30	30
Diseño	60	90
Codificación	90	180
Pruebas	60	240
Distribución	15	255
Documentación	45	300

Tabla 2.1: Planificación temporal

CAPÍTULO 3

OBJETIVOS

En este capítulo se exponen los objetivos propuestos a alcanzar en este TFG, los cuales se han dividido en formales y operacionales.

3.1. Objetivos formales

A continuación se definen los objetivos formales que pretende cubrir este TFG:

- Comprender el proceso del desarrollo de videojuegos y cada una de las distintas fases que lo componen.
- Aprender a utilizar el motor gráfico *Unity*.
- Aprender a crear una interfaz usable y cómoda para el usuario.
- Desarrollar una documentación técnica asociada al proyecto.

3. Objetivos

3.2. Objetivos operacionales

En esta sección se exponen los objetivos operacionales que se pretenden cumplir:

- **OBJ-1. Módulo con el videojuego Stealthy Spy.** Se tratará de un videojuego de plataformas 2D que contará con las siguientes características:
 - El juego dispondrá de una lista de varios mapas que el usuario podrá rejugar infinitamente.
 - Existirán tres dificultades (fácil, normal, difícil). La dificultad influirá en el tiempo límite para completar el mapa, así como del número de obstáculos a superar.
 - Se registrarán las mejores puntuaciones obtenidas (relacionadas con el tiempo empleado) por el jugador para cada mapa y dificultad, que pondrán ser consultadas.
 - Dentro de la partida, el jugador tomará el rol de espía que deberá moverse por el escenario tratando de no ser advertido por los guardias y cámaras de seguridad para llegar a su objetivo en el tiempo establecido.
 - El mapa se dará por fallido si el temporizador llega a cero o si el jugador es detectado.
 - El jugador contará con la habilidad de incapacitar a enemigos por la espalda, dejándolos aturdidos e inmóviles durante un periodo de tiempo.
- **OBJ-2. Módulo de configuración.** Se dispondrá de un módulo de configuración para ajustar ciertos parámetros del juego como el volumen. Este menú será accesible tanto desde el menú principal como desde el menú de pausa durante la partida.
- **OBJ-3. Módulo de créditos.** La pantalla de créditos mostrará información sobre el proyecto.

- **OBJ-4. Interfaz.** La aplicación constará de una interfaz usable, fácil de manejar y comprender y llamativa para crear una buena experiencia al jugador. Deberá ser capaz de adaptarse al tamaño de pantalla de cualquier dispositivo *Android*.

CAPÍTULO 4

ANTECEDENTES

En este capítulo se hace un breve recorrido por los inicios de los videojuegos y sobre el género de plataformas.

4.1. Origen de los videojuegos

Los videojuegos llevan bastante tiempo con nosotros [10]. Ya desde los años 50 empezaron a surgir los primeros videojuegos del mercado. Aunque no se conoce exactamente cuál fue el primero, muchos le atribuyen este título al videojuego *Nought and crosses*, también llamado OXO, desarrollado por Alexander S. Douglas en 1952, véase figura 4.1. Se trataba de una versión del famoso juego de mesa tres en raya que permitía enfrentar a un jugador humano contra una máquina.

4. Antecedentes

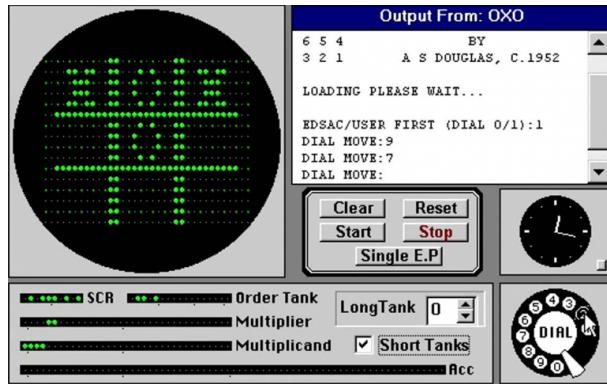


Figura 4.1: OXO. Primer videojuego creado

Unos años más tarde, William Higginbotham creó *Tennis for Two* [11], un simulador de tenis de mesa para entretenimiento. En la figura 4.2 se puede observar que no hacía uso de una perspectiva aérea para mostrar el desarrollo del juego, sino que utilizaba una vista lateral en la que se podía observar una pista que incluía como único detalle la red que separaba ambos campos. Los dos jugadores que podían participar en cada partida disponían de un controlador compuesto por un pulsador que servía para golpear la pelota virtual y un mando analógico con el que se controlaba la dirección de la bola.

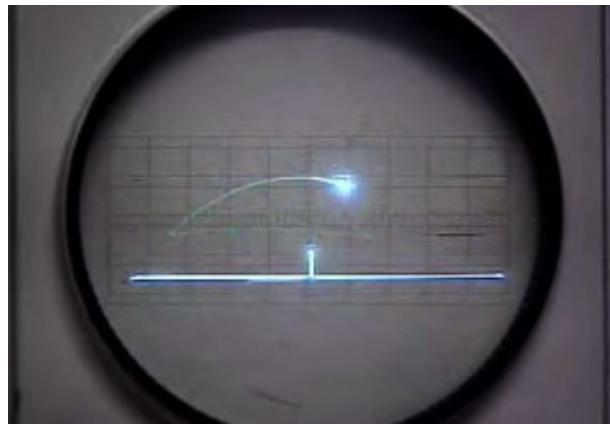


Figura 4.2: Tenis for Two

Cuatro años más tarde, Steve Russell, diseñó el videojuego *Spacewar!* [12] que consistía en que dos jugadores controlaban la dirección y la velocidad de dos naves espaciales que luchaban entre ellas. El videojuego funcionaba sobre un PDP-1, como se muestra en

la figura 4.3, y fue el primero en tener un cierto éxito, aunque apenas fue conocido fuera del ámbito universitario.



Figura 4.3: Spacewar!

La ascensión de los videojuegos llegó con la máquina recreativa *Pong* [13], muy similar al *Tennis for Two* pero utilizada en lugares públicos: bares, salones recreativos, aeropuertos, etc. El sistema fue diseñado por Al Alcorn para Nolan Bushnell en la recién fundada Atari. A partir de aquí, empezó a desarrollarse la tecnología, destacando las videoconsolas que poco a poco fueron dejando obsoletas a las máquinas recreativas. A su vez, la complejidad y tamaño de los videojuegos fueron aumentando y con ello llegaron ideas y temáticas revolucionarias que fueron expandiendo la industria hasta tener la gran variedad que tenemos hoy en día.

4.2. Videojuegos de plataformas

Centrándonos en el género que nos interesa para este proyecto, el género de plataformas es uno de los géneros fundamentales en la historia de los videojuegos. Los videojuegos de este género se caracterizan por tener que controlar un personaje que camina, corre, salta y escala entre plataformas mientras evita obstáculos [14].

Los videojuegos de plataformas se originaron a finales de los setenta y principios de los ochenta. Los primeros ejemplos de juegos de plataformas se limitaron a un campo de juego estático, generalmente visto de perfil. *Donkey Kong* [15], creado por Nintendo

4. Antecedentes

y lanzado en julio de 1981, fue el primer juego que permitió a los jugadores saltar por encima de los obstáculos y los agujeros, lo que lo convirtió en el primer verdadero juego de plataformas, véase figura 4.4. El protagonista era Jumpman que más tarde se convertiría en Mario. El juego ayudó a consolidar la posición de Nintendo como un nombre importante en la industria de los videojuegos a nivel internacional.

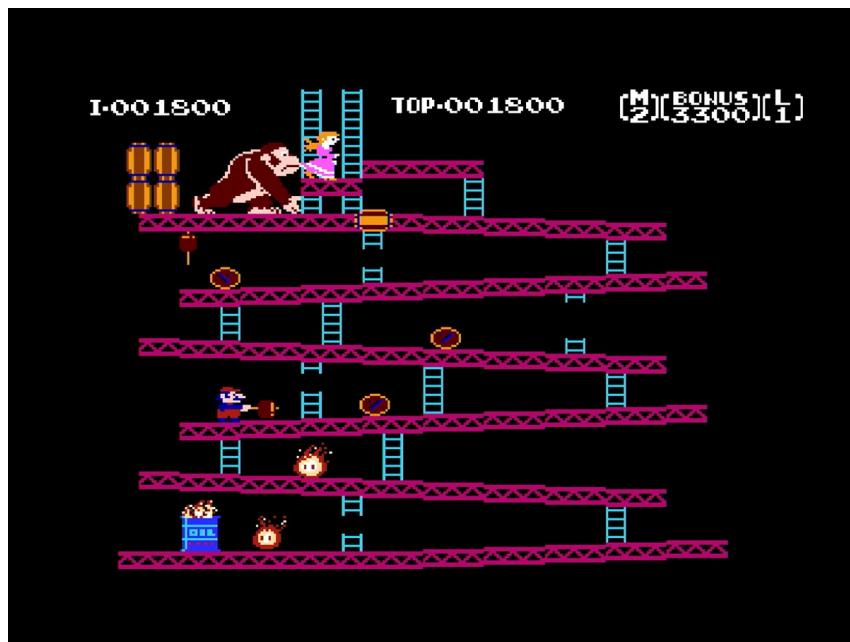


Figura 4.4: Donkey Kong

A partir de 1982, surgieron juegos de transición que no contaban con desplazamiento automático, pero tenían niveles que abarcaban varias pantallas conectadas. *Pitfall!* [16], lanzado para Atari 2600, presentaba niveles amplios y extendidos horizontalmente. Se convirtió en el juego más vendido del sistema y fue un gran avance para el género. *Manic Miner* [17] (1983) y su secuela *Jet Set Willy* (1984) [18] continuaron con este estilo de niveles multipantalla en ordenadores. En 1984, Epyx lanzó *Impossible Mission* [19], que amplió aún más el aspecto de exploración y sentó las bases para juegos como *Prince of Persia* [20].

Pero en 1985, el juego de plataformas de Nintendo, *Super Mario Bros.*, lanzado para NES se convirtió en el arquetipo para el resto de plataformas. Su éxito como juego llevó a

muchas compañías a ver los juegos de plataformas como vitales para su éxito, y contribuyó enormemente a popularizar el género durante la generación de consolas de 8 bits. Se vendieron 40 millones de unidades. El término "juegos de plataformas" se estandariza a través del éxito de Super Mario Bros. A raíz de eso empezaron a surgir una cantidad masiva de videojuegos de plataformas, manteniéndose como el género que más videojuegos tenía y contando con grandes títulos como *Metroid* [21], *Castlevania* [22], *Mega Man* [23] y *Sonic* [24].

Poco a poco, el lanzamiento de las nuevas consolas hizo que la atención de los jugadores se alejara de los géneros 2D, aunque seguían presentando algunos éxitos como *Rayman* [25]. En esta época fue cuando apareció el llamado 2.5D, que se refiere a videojuegos con estética 3D de desplazamiento lateral como el que aparece en la figura 4.5. La llegada de *Mario 64* [26] y *Crash Bandicoot* [27] puso de moda los plataformas en tres dimensiones y los bidimensionales quedaron en un segundo plano y casi desapareciendo con las siguientes generaciones de consolas.



Figura 4.5: Super Mario 64

Sin embargo, en estos últimos años los plataformas 2D se popularizaron de nuevo gracias a los desarrolladores independientes y a las plataformas de venta digital. A su vez, esto provocó que grandes empresas como Nintendo volvieran al género clásico. Podemos ver grandes representantes de juegos *indies* en la figura 4.6 como *Hollow Knight* [28], lanzado en 2017; *Ori and the Blind Forest* [29], lanzado en 2015, y su secuela *Ori and*

4. Antecedentes

the Will of the Wisps [30], lanzado en 2020. Gracias a este nuevo *boom*, los plataformas 2D siguen estando muy vivos en la actualidad y siguen manteniendo su importancia en el mercado.



(a) Hollow Knight



(b) Ori and The Will Of The Wisps

Figura 4.6: Plataformas actuales

CAPÍTULO 5

RESTRICCIONES

A continuación, se identifican los distintos factores dato y estratégicos que restringen la naturaleza del problema.

5.1. Factores dato

Como factores dato en este TFG se presentan los siguientes:

- **Limitación de recursos hardware.** Se cuentan con los recursos hardware del propio alumno que se detallarán en el capítulo 6.
- **Limitación de recursos software.** Los recursos software con los que se contará será el uso de software con licencias gratuitas o de software libre, siempre que sea posible.
- **Limitación de recursos humanos.** En la realización del proyecto se contará con los directores del proyecto: Juan Carlos Fernández Caballero y David Guijo Rubio.

- **Limitación de tiempo.** El proyecto está emplazado en un marco temporal reducido, ya que el reglamento de los trabajos fin de grado de la EPSC fija una carga de 300 horas para la realización de un TFG, por lo que esto ha de ser tenido en cuenta a la hora de realizar la aplicación.
- **Limitación de rendimiento.** La aplicación ha de estar optimizada para poder ser ejecutada en cualquier dispositivo móvil con sistema operativo *Android* y la versión mínima requerida.

5.2. Factores estratégicos

Los factores estratégicos para este TFG se indican a continuación:

- **Sistema Operativo.** La elección será *Android* debido a su gran cuota de mercado y a la facilidad y gratuitad que ofrece a la hora de publicar contenido en *Play Store*, su plataforma de descarga de aplicaciones.
- **Motor gráfico.** Se ha optado por la opción de *Unity* como motor gráfico debido a que ofrece una curva de aprendizaje baja, usa como lenguaje de programación *C#* y permite exportar a *Android*, además cuenta con una gran comunidad y documentación.
- **Lenguaje de programación.** El lenguaje de programación será *C#* porque es el lenguaje oficial para el desarrollo con *Unity* [5].
- **Entorno de desarrollo.** Se optó por *Visual Studio Community* [31], ya que es un entorno potente, con soporte oficial desde *Unity*, de fácil uso y con una amplia comunidad de foros y soporte por parte de otros usuarios.
- **Assets.** Se utilizarán *assets* gratuitos de distintas páginas oficiales y además la aplicación de *pixelart Piskel* [32] para crear algunos *assets* originales.
- **Control de versiones.** Existen numerosas herramientas especializadas en el control de versiones. Se ha elegido *Git* por ser la más usada y la que más soporte tiene en

la comunidad.

- **Repositorio de código.** Algunos de los repositorios basados en *Git* más extendidos y con más soporte son *GitHub* y *Bitbucket*. Se ha seleccionado *GitHub* [33] por estar más extendido entre los desarrolladores y tener más soporte.
- **Versión mínima de *Android* requerida.** La versión mínima para usar el videojuego será la versión 5.1 porque es la versión más antigua que soporta *Unity*.

CAPÍTULO 6

RECURSOS

En este capítulo se detallan los diferentes recursos, tanto humanos como materiales, que están involucrados en el desarrollo del proyecto.

6.1. Recursos humanos

Autor: Alberto Adamuz Priego.

Alumno de 4º curso del Grado de Ingeniería Informática.

Director: Juan Carlos Fernández Caballero.

Profesor Titular de la Universidad de Córdoba del Dpto. de Informática
y Análisis Numérico y miembro investigador del grupo AYRNA.

6. Recursos

Director: David Guijo Rubio

Investigador postdoctoral en la Universidad East Anglia (Norwich, Reino Unido) y miembro investigador del grupo AYRNA.

6.2. Recursos software

- **Sistema Operativo:** Windows 11
- **Entorno de desarrollo:** *Unity 2021.3.12f1*.
- **Entorno de programación:** *Microsoft Visual Studio Community 2022*.
- **Control de versiones:** *GitHub*.
- **Diseño de assets:** *Piskel*
- **Entorno para la elaboración de la documentación:** *LaTeX*

6.3. Recursos hardware

Equipo personal en el que se llevará a cabo el desarrollo:

- **Procesador:** Procesador Intel Core i7-9750H 2.60GHz
- **Memoria RAM:** 16GB
- **Tarjeta gráfica dedicada:** NVIDIA GeForce GTX 1650

Dispositivo móvil para realizar las pruebas en Android:

- **Memoria RAM:** 8GB
- **Memoria interna:** 256GB
- **Pantalla:** Super AMOLED 2400x1080
- **Procesador:** Snapdragon 778G 5G

Parte II

Análisis del sistema y especificación de requisitos

CAPÍTULO 7

ESPECIFICACIÓN DE REQUISITOS

En este capítulo se identifican los distintos bloques funcionales de la aplicación a desarrollar y se definen de forma técnica los requisitos de información, funcionales y no funcionales que definen el problema.

7.1. Identificación de los bloques funcionales

La composición de los bloques funcionales necesarios para desarrollar el videojuego son los siguientes:

- Bloque de construcción del escenario.
- Bloque de control del jugador.
- Bloque de gestión del sistema.
- Bloque de gestión de la interfaz.

7. Especificación de Requisitos

A continuación se realizará una descripción breve de cada bloque identificando la funcionalidad incluida en cada uno de ellos.

7.1.1. Bloque de construcción del escenario

Se identifican las funcionalidades relacionadas con la construcción del escenario del videojuego. Dispondremos de varios objetos con características y funciones variadas que se gestionarán en este bloque:

- **Plataformas.** Las plataformas componen el terreno donde se sitúa el jugador y los diferentes objetos con los que se podrá interactuar.
- **Jugador.** El jugador estará representado por un personaje que podrá moverse e interactuar con el escenario.
- **Enemigos.** Los enemigos serán entidades que el jugador deberá sortear o incapacitar para evitar ser detectado.
- **Camuflaje.** Por el mapa habrá ciertos objetos en los que si el jugador se coloca detrás de ellos, permanecerá invisible para los enemigos.
- **Temporizador.** En pantalla se mostrará el tiempo restante para completar el mapa.

7.1.2. Bloque de manejo del jugador

En este bloque se identifican las funcionalidades correspondientes al jugador para poder realizar las acciones permitidas durante la partida:

- **Movimiento del personaje.** El movimiento del jugador se realizará mediante gestos con la mano sobre un joystick visible en la pantalla sobre del dispositivo Android que se esté usando y servirá para mover al jugador de izquierda a derecha.
- **Salto.** El jugador podrá realizar un salto pulsando sobre un botón en pantalla.
- **Incapacitar.** El jugador podrá incapacitar a un enemigo pulsando un botón que aparecerá en la pantalla cuando este se encuentre situado detrás del objetivo.

7.1.3. Bloque de gestión del sistema

En este bloque se identifican una importante parte de las funcionalidades que son necesarias para el correcto funcionamiento y desempeño en el desarrollo del videojuego:

- **Gestionar el tiempo.** El sistema controlará el tiempo transcurrido en el videojuego durante una partida mediante un temporizador. Si el temporizador llega a cero, se dará por fracasada la partida.
- **Guardar mejores tiempos.** El sistema guardará en un fichero los mejores tiempos conseguidos por el jugador en cada mapa y para cada dificultad.
- **Cargar mejores tiempos** El sistema cargará un fichero de mejores tiempos al iniciar el videojuego para que el jugador pueda consultarlos.

7.1.4. Bloque de gestión de la interfaz

En este bloque se incluyen las distintas funcionalidades relacionadas con la interfaz de usuario y los menús que la componen:

- **Menú principal.** Es la pantalla que verá el usuario al iniciar la aplicación, desde la que podrá acceder a la configuración y a la selección de mapas.
- **Selección de mapas.** Se mostrará una lista de los mapas disponibles para iniciar la partida. También permitirá elegir la dificultad del mapa seleccionado y se mostrará la mejor puntuación obtenida para cada mapa en cada dificultad.
- **Modo pausa.** Interfaz a la que se podrá acceder durante la partida para volver al menú principal, cambiar algún aspecto de la configuración o reiniciar la partida.
- **Configuración.** Permitirá cambiar algunos ajustes del videojuego.
- **Pantalla de créditos.** Mostrará al usuario una pantalla con los créditos del juego.

7.2. Requisitos funcionales

A continuación se detallan los requisitos funcionales que debe cumplir el sistema:

- **RF-1.** El sistema permitirá elegir varios mapas que se irán desbloqueando a medida que se vayan completando.
- **RF-2.** El sistema permitirá elegir entre distintas dificultades: fácil, intermedio y difícil; que se diferenciarán por la cantidad máxima de tiempo y cantidad de obstáculos.
- **RF-3.** El sistema permitirá que el jugador se mueva y salte.
- **RF-4.** El sistema mantendrá un temporizador activo durante la partida para contabilizar el tiempo empleado en resolver el mapa.
- **RF-5.** El sistema finalizará la partida si el jugador llega al final del mapa o el temporizador llega a cero.
- **RF-6.** El sistema permitirá la pausa o finalización de la partida durante esta.
- **RF-7.** El sistema permitirá controlar el volumen de los sonidos desde el menú de configuración y el de pausa.
- **RF-8.** El sistema permitirá volver al menú principal desde la partida comenzada.
- **RF-9.** El sistema permitirá ver los créditos del videojuego accediendo desde el menú principal.
- **RF-10.** El sistema mantendrá un histórico de la mejor puntuación obtenida en cada mapa, basada en el tiempo empleado para resolverlo y dividido para cada dificultad.
- **RF-11.** El jugador podrá utilizar distintos objetos para ocultarse de los enemigos.
- **RF-12.** Los enemigos podrán detectar al jugador. Si el jugador es detectado durante demasiado tiempo, se considerará que ha sido descubierto y perderá la partida.
- **RF-13.** El jugador podrá incapacitar a un enemigo por la espalda, dejándolo inactivo durante un periodo de tiempo.

- **RF-14.** El jugador podrá intentar superar su mejor tiempo en partidas anteriores.

7.3. Requisitos de información

A continuación se describen los requisitos de información que deberá gestionar y almacenar el sistema:

- **RI-1.** El sistema deberá almacenar la información referente al estado actual (posición del jugador, posición de enemigos, etc.) en el que se encuentra la partida para poder reanudarse en el menú de pausa.
- **RI-2.** El sistema deberá almacenar la información necesaria para comprobar si el jugador ha reducido el tiempo empleado en finalizar el mapa durante anteriores partidas.
- **RI-3.** El sistema deberá almacenar la información referente a la configuración establecida por el usuario, como volumen de la música y de los efectos de sonido, etc.

7.4. Requisitos no funcionales

A continuación se detallan los requisitos no funcionales específicos de este sistema:

- **RNF-1.** La instalación y desinstalación de la aplicación deberá ser lo más sencilla posible.
- **RNF-2.** El tiempo de respuesta del sistema deberá ser el mínimo posible.
- **RNF-3.** En la medida de lo posible, el videojuego se debe poder ejecutar en la mayor cantidad de dispositivos Android.

7.5. Requisitos de la interfaz

La interfaz se desarrollará siguiendo varios puntos a tener en cuenta para el diseño del videojuego:

- **RINT-1.** La interfaz se adaptará para su ejecución óptima en dispositivos móviles con sistema operativo Android.
- **RINT-2.** Deberá ser sencilla y de fácil interpretación, para que cualquier tipo de usuario pueda navegar y jugar sin ningún problema. Para ello dispondrá de un menú principal y de varios submenús claros y concisos, con la mínima información necesaria en pantalla en cada momento.
- **RINT-3.** Durante la partida, los controles serán visibles en pantalla, permitiendo al usuario controlar con cada mano un aspecto del juego, como mover el personaje, saltar o realizar alguna acción especial.

7.6. Matriz de rastreabilidad

A continuación, en la tabla 7.1, se muestra la matriz de rastreabilidad que relaciona cada uno de los requisitos enumerados en las secciones anteriores con los objetivos del proyecto.

	OBJ-1	OBJ-2	OBJ-3	OBJ-4
RI-1	X			
RI-2	X			X
RI-3		X		
RF-1	X			
RF-2	X			
RF-3	X			
RF-4	X			
RF-5	X			
RF-6	X			
RF-7		X		
RF-8	X			
RF-9			X	
RF-10	X			
RF-11	X			
RF-12	X			
RF-13	X			
RF-14	X			
RINT-1				X
RINT-2				X
RINT-3	X			X

Tabla 7.1: Matriz de rastreabilidad

CAPÍTULO 8

ANÁLISIS FUNCIONAL

En este capítulo se realiza un análisis funcional del sistema, para lo cual se usará *UML*. Con objeto de obtener una mayor comprensión del problema se van a utilizar los diagramas de casos de uso mediante los cuales se puede exponer el comportamiento del sistema y su interacción con los usuarios.

En primer lugar, se identifican los usuarios que participan en el sistema, denominados “actores”. A continuación, mediante los casos de uso se establecen los escenarios que interactúan con los usuarios.

8.1. Usuarios del sistema

Los usuarios del sistema pueden ser tanto niños, jóvenes como adultos que quieran usar el videojuego, pero principalmente se dirige a los dos primeros. Cualquier persona que utilice el juego puede acceder a la misma funcionalidad, por lo que en los casos de uso se muestra solamente un actor.

8.2. Análisis de casos de uso

A continuación se describen y explican los distintos casos de uso que describen cómo el usuario puede interactuar con el videojuego.

8.2.1. Diagrama de contexto

En el diagrama de contexto representado en la figura 8.1 se muestra el flujo principal de la aplicación desde su inicio, dividiéndola en las funcionalidades más genéricas.

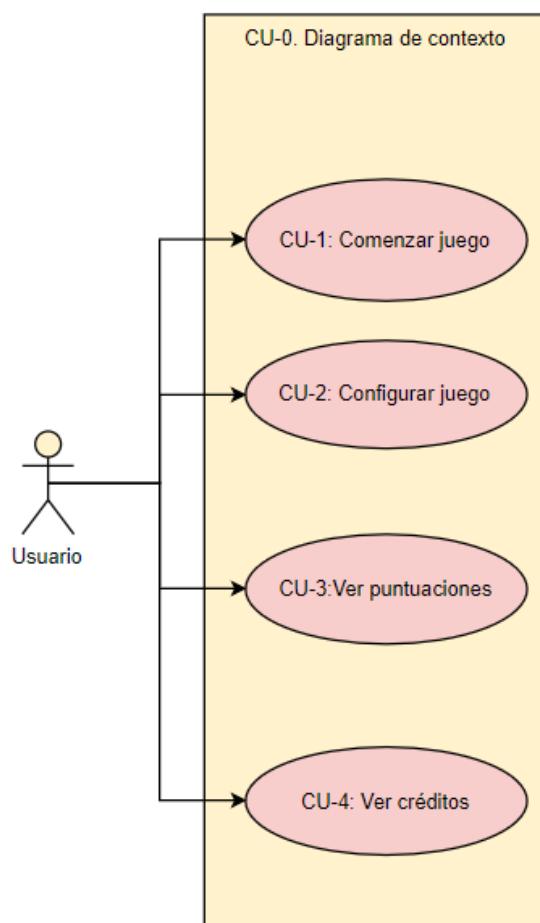


Figura 8.1: CU-0. Diagrama de contexto

La descripción del caso de uso CU-0 es la siguiente:

- **Nombre.** CU-0. Diagrama de contexto
- **Descripción.** Permite al usuario iniciar uno de los cuatro módulos de la aplicación.
- **Actores.** Usuario.
- **Casos de uso:**
 - CU-1. Comenzar juego: Permite iniciar una partida.
 - CU-2. Configurar juego: Permite modificar algunos parámetros del juego.
 - CU-3. Ver puntuaciones: Permite ver un registro de las mejores puntuaciones obtenidas.
 - CU-4. Ver créditos: Permite visualizar los créditos de la aplicación.
- **Precondiciones.** Haber iniciado la aplicación por parte del usuario.
- **Flujo principal.** El usuario elige la funcionalidad deseada para ser cargada.
- **Postcondiciones.** El módulo seleccionado ha sido cargado.

8.2.2. Caso de uso 1. Comenzar juego

El usuario puede jugar una partida como se especifica en la figura 8.2.

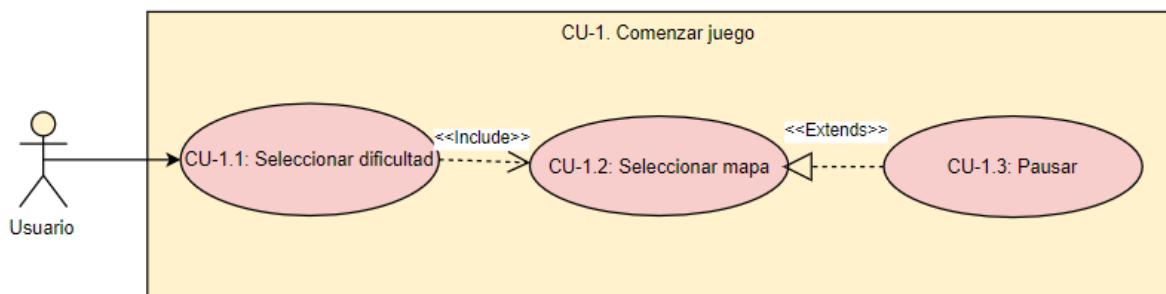


Figura 8.2: CU-1. Comenzar juego

8. Análisis funcional

La descripción del caso de uso CU-1 es la siguiente:

- **Nombre.** CU-1. Comenzar juego
- **Descripción.** Permite al usuario jugar una partida.
- **Actores.** Usuario.
- **Casos de uso:**
 - **CU-1.1. Seleccionar dificultad:** Permite al usuario seleccionar la dificultad del juego.
 - **CU-1.2. Seleccionar mapa:** Permite al usuario seleccionar el mapa que desea jugar.
 - **CU-1.3. Pausar:** Permite al usuario pausar la partida.
- **Precondiciones.** El usuario debe haber escogido la opción *Comenzar Juego* en el menú principal.
- **Flujo principal.**
 - 1. Se muestra una lista de mapas disponibles.
 - 2. El usuario elige la dificultad que desea.
 - 3. El usuario selecciona el mapa al que quiere jugar.
 - 4. Se carga el mapa seleccionado y comienza la partida.
- **Postcondiciones.** El mapa seleccionado ha sido cargado en la dificultad escogida y la partida ha iniciado.

8.2.3. Caso de uso 1.1. Seleccionar dificultad

El usuario debe seleccionar la dificultad a la que desea jugar. La descripción del caso de uso CU-1.1 es la siguiente:

- **Nombre.** CU-1.1. Seleccionar dificultad

- **Descripción.** Permite al usuario seleccionar la dificultad del juego.
- **Actores.** Usuario.
- **Precondiciones.** El usuario debe haber seleccionado la opción de *Comenzar juego* en el menú principal y la lista de mapas debe haber sido cargada.
- **Flujo principal.**
 - 1. El usuario selecciona una de las tres dificultades disponibles (fácil, normal y difícil) en la parte superior del menú.
- **Postcondiciones.** La dificultad es seleccionada y en cada mapa se muestra la mejor puntuación obtenida en esa dificultad

8.2.4. Caso de uso 1.2. Seleccionar mapa

El usuario debe seleccionar el mapa que desea jugar. La descripción del caso de uso CU-1.2 es la siguiente:

- **Nombre.** CU-1.2. Seleccionar mapa
- **Descripción.** Permite al usuario seleccionar un mapa para jugar.
- **Actores.** Usuario.
- **Precondiciones.** El usuario debe haber escogido la opción *Comenzar juego* en el menú principal y la lista de mapas debe haber sido cargada.
- **Flujo principal.**
 - 1. El usuario selecciona uno de los mapas disponibles en la lista.
- **Postcondiciones.** El mapa seleccionado es cargado en la dificultad escogida y comienza la partida.

8.2.5. Caso de uso 1.3. Pausar

El usuario puede pausar la partida actual en cualquier momento. La descripción del caso de uso CU-1.3 es la siguiente:

- **Nombre.** CU-1.3. Pausar
- **Descripción.** Permite al usuario pausar la partida
- **Actores.** Usuario.
- **Precondiciones.** Debe existir una partida cargada.
- **Flujo principal.** El usuario selecciona la opción de pausa y después puede elegir las siguientes funcionalidades:
 - **Reanudar juego:** Permite al usuario reanudar la partida actual en el momento y estado previo a la pausa.
 - **Reiniciar partida:** Permite al usuario empezar de nuevo la partida desde el comienzo en el mismo mapa y con la misma dificultad que la partida actual.
 - **Volver al menú principal:** Permite al usuario finalizar la partida y regresar al menú principal.
 - **Configuración:** Permite al usuario abrir el menú de configuración.
- **Postcondiciones.** Según la funcionalidad elegida por el usuario ocurrirá una de las siguientes situaciones:
 - **Reanuda la partida:** El usuario continúa la partida actual.
 - **Reiniciar la partida:** El usuario comienza una nueva partida en el mismo mapa y con la misma dificultad que la partida anterior.
 - **Finalizar la partida:** El usuario finaliza la partida actual considerándose como mapa no superado.
 - **Configuración:** Se abrirá el menú de configuración.

8.2.6. Caso de uso 2. Configurar juego

El usuario podrá acceder al menú de configuración para modificar algunos parámetros de la aplicación, como se representa en la figura 8.3.

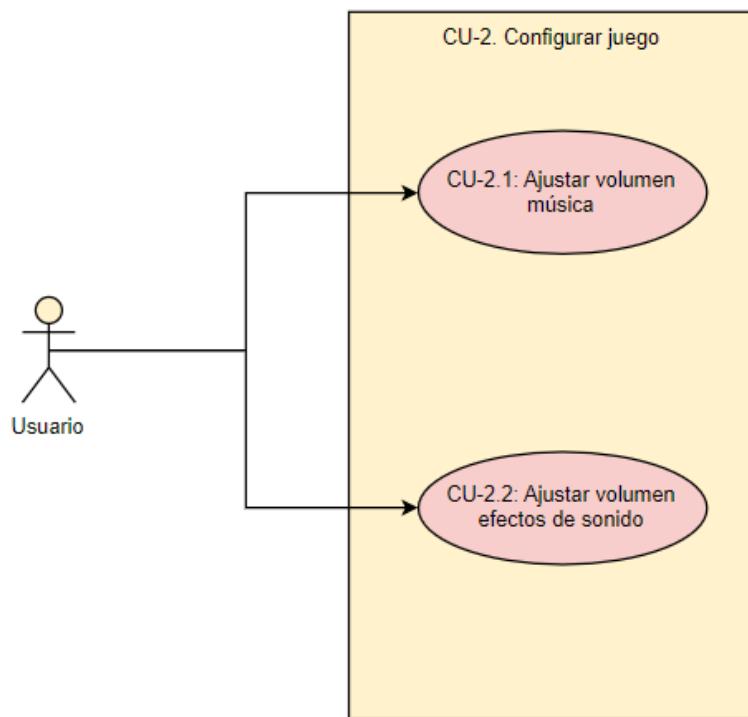


Figura 8.3: CU-2. Configurar juego

La descripción del caso de uso CU-2 es la siguiente:

- **Nombre.** CU-2. Configuración
- **Descripción.** Permite al usuario configurar algunos parámetros de la aplicación.
- **Actores.** Usuario.
- **Precondiciones.** El usuario debe haber seleccionado la opción *Configuración* del menú principal.

- **Casos de uso:**
 - **CU-2.1. Ajustar volumen de música:** Permite ajustar el volumen de la música del juego.
 - **CU- 2.2. Ajustar volumen de efectos de sonido:** Permite ajustar el volumen de los efectos de sonido del juego.
- **Flujo principal.** El usuario pulsa la opción *Configuración* en el menú principal o en el menú de pausa. Después puede seleccionar entre las siguientes funcionalidades.
 - **Ajustar música.** Permite al usuario modificar el volumen de la música del juego.
 - **Ajustar efectos de sonido.** Permite al usuario modificar el volumen de los efectos de sonido.
- **Postcondiciones.** El volumen de la música y de los efectos de sonidos se modificará a lo establecido por el usuario.

8.2.7. Caso de uso 2.1. Ajustar volumen música

El usuario podrá ajustar el volumen de la música. La descripción del caso de uso CU-2.1 es la siguiente:

- **Nombre.** CU-2.1. Ajustar volumen música
- **Descripción.** Permite al usuario modificar el volumen de la música del juego.
- **Actores.** Usuario.
- **Precondiciones.** El menú de configuración debe estar cargado.
- **Flujo principal.** El usuario ajusta el volumen de la música.
- **Postcondiciones.** El volumen de la música se ajusta a la configuración escogida.

8.2.8. Caso de uso 2.2. Ajustar volumen efectos de sonido

El usuario podrá ajustar el volumen de los efectos de sonido. La descripción del caso de uso CU-2.2 es la siguiente:

- **Nombre.** CU-2.2. Ajustar volumen efectos de sonido
- **Descripción.** Permite al usuario modificar el volumen de los efectos de sonido.
- **Actores.** Usuario.
- **Precondiciones.** El menú de configuración debe estar cargado.
- **Flujo principal.** El usuario ajusta el volumen de los efectos de sonido.
- **Postcondiciones.** El volumen de los efectos de sonido se ajusta a la configuración escogida.

8.2.9. Caso de uso 3. Ver puntuaciones

El usuario puede ver un registro de sus mejores puntuaciones en los diferentes mapas y dificultades. La descripción del caso de uso CU-3 es la siguiente:

- **Nombre.** CU-3. Ver puntuaciones.
- **Descripción.** Se muestra un registro de las mejores puntuaciones obtenidas.
- **Actores.** Usuario.
- **Precondiciones.** El usuario selecciona la opción *Puntuaciones* en el menú principal.
- **Flujo principal.**
 - 1. El usuario pulsa la opción *Puntuaciones*.
- **Postcondiciones.** Se carga un fichero con las mejores puntuaciones del jugador en todos los mapas y dificultades y se muestra por pantalla.

8.2.10. Caso de uso 4. Ver créditos

El usuario puede ver la pantalla de créditos. La descripción del caso de uso CU-4 es la siguiente:

- **Nombre.** CU-4. Ver créditos.
- **Descripción.** Se muestran los créditos del juego.
- **Actores.** Usuario.
- **Precondiciones.** El usuario selecciona la opción *Créditos* en el menú principal.
- **Flujo principal.**
 - 1. El usuario pulsa la opción *Créditos*.
- **Postcondiciones.** Se muestra la pantalla de créditos.

Parte III

Diseño del sistema

CAPÍTULO 9

ARQUITECTURA DEL SISTEMA

En cuanto al diseño y estructuración del código, este está dividido en una serie de *scripts* que se utilizan como componentes de los objetos y marcan el funcionamiento que deben tener los objetos que los contienen.

Por ejemplo, podemos tener el objeto «JUGADOR» y este posee los componentes «COLLIDER» (detecta las colisiones) y «PLAYER CONTROLLER» (recibe los *inputs*) y a su vez un objeto «ENEMIGO» que solo posea el componente «COLLIDER». El hecho de poseer el componente «COLLIDER» permitirá tanto a «JUGADOR» como a «ENEMIGO» detectar colisiones, pero solo «JUGADOR» recibirá los *inputs* del controlador porque solo él tiene el componente «PLAYER CONTROLLER». Esto permite reutilizar y ajustar ciertas funcionalidades en diferentes objetos de manera cómoda, haciendo que el sistema sea mucho más modular.

También para manejar el flujo de ejecución de la aplicación, se ha utilizado una arquitectura dirigida por eventos [34]. Esto consiste en una clase que detecta cuándo se produce un cambio significativo en la aplicación y se encarga de notificar a las clases suscritas a esos eventos para que lo procesen. Esto facilita la integración de nuevos *scripts*

9. Arquitectura del sistema

a la aplicación, aunque se debe tener especial cuidado porque no controla el orden en el que se ejecutan los *scripts*.

Para el diseño de la interfaz de usuario se usarán las herramientas internas del propio *Unity*.

Para comprobar el resultado obtenido por estas interfaces y para simular el sistema operativo *Android* en el mismo entorno se ha optado por usar “*Device Simulator*” [35]. Se trata de una herramienta de *Unity* que simula cómo se comportará la aplicación en varios dispositivos móviles.

La estructura del proyecto ha sido dividida en distintos módulos teniendo en cuenta el tipo de archivos y su funcionalidad.

- **Módulo Animation.** Contiene ficheros con las animaciones de personajes y objetos del escenario. Existen dos tipos de archivos:
 - **Animator Controller**, que es el encargado de controlar la animación de un objeto en la escena. Existe uno por cada objeto animado.
 - **Animator Clip**, son las distintas secuencias de sprites, representadas en orden temporal, que en conjunto forman una animación. Es decir, estos archivos no solo poseen los *sprites* que forman la animación sino que ya se establece el orden y duración de cada *sprite*. Pueden existir varios para cada objeto.

Cada archivo *Animator Clip* está asociado a un *Animator Controller* que a su vez se enlaza al componente *Animator* del objeto que se desea animar. Este componente, véase la figura 9.1, representa la secuencia de animaciones como una máquina de estados donde cada estado representa una animación o *Animator Clip* y cada transición indica cuando se cambia de animación.

Además en la barra vertical izquierda se muestran las variables que controlan esas transiciones cuyo valor se modifica desde código.

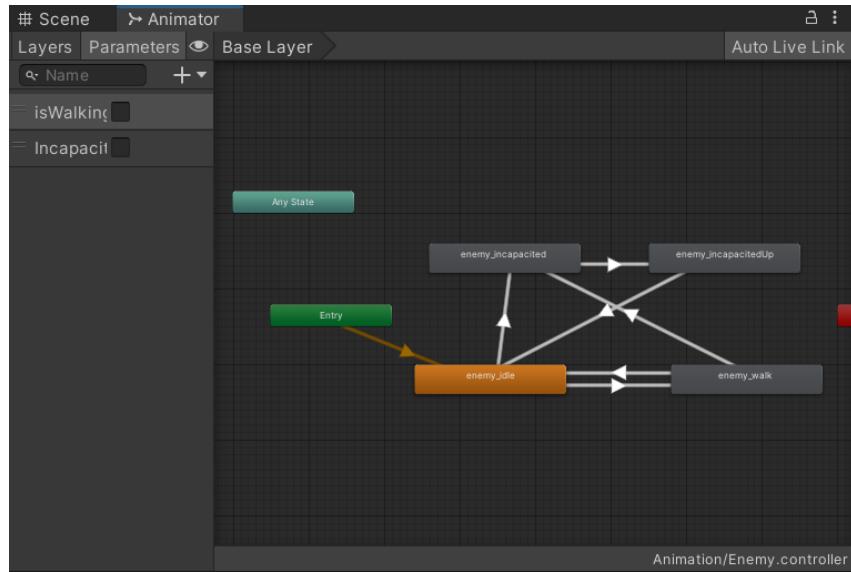


Figura 9.1: *Animator*

- **Módulo *Font*.** Contiene ficheros con las fuentes de texto utilizadas.
- **Módulo *Graphics*.** Contiene ficheros con todos los *sprites*, *tilesets* y texturas del juego. También contiene lo que se denomina *TilePalette*, esto es un archivo que guarda un conjunto de texturas que permite, desde la ventana *TilePalette* (veáse figura 9.2) del editor, dibujar en el escenario en una rejilla, como se muestra en la figura 9.3, facilitando la creación de mapas.

9. Arquitectura del sistema

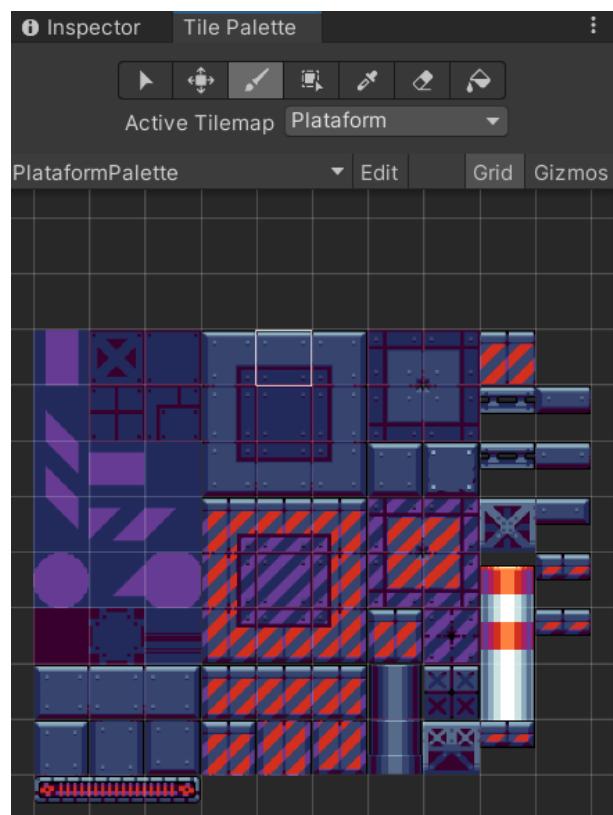


Figura 9.2: *Tile Palette*

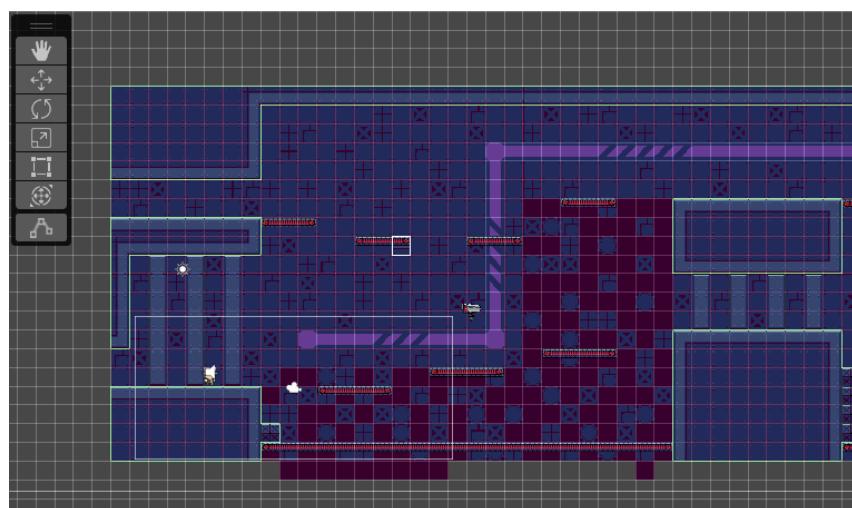


Figura 9.3: Rejilla

-
- **Módulo *Materials*.** Contiene ficheros de materiales, estos controlan la iluminación de los objetos que posean estos materiales en la escena.
 - **Módulo *Plugins*.** Contiene ficheros de los *plugins* utilizados. En primer lugar, se ha utilizado el plugin *Joystick Pack* que ofrece un *joystick* táctil ya configurado. También el *plugin* *DoTween* que permite hacer pequeñas animaciones (sobre todo se ha utilizado en la interfaz). Y por último *TextMeshPro*, que lo trae el motor por defecto, que proporciona una mejor calidad visual y brinda una flexibilidad increíble en lo que respecta al estilo y la textura del texto.
 - **Módulo *Prefabs*.** Contiene plantillas de objetos compuestos y configurados para instanciarlos más fácilmente. Por ejemplo, el fichero *Camera.prefab* es la plantilla que define al completo una cámara de vigilancia. Como se puede observar en la figura 9.4, este objeto está a su vez definido por varios objetos como la cabeza o el cono de visión y cada objeto tiene sus componentes ya configurados con unos valores preestablecidos.

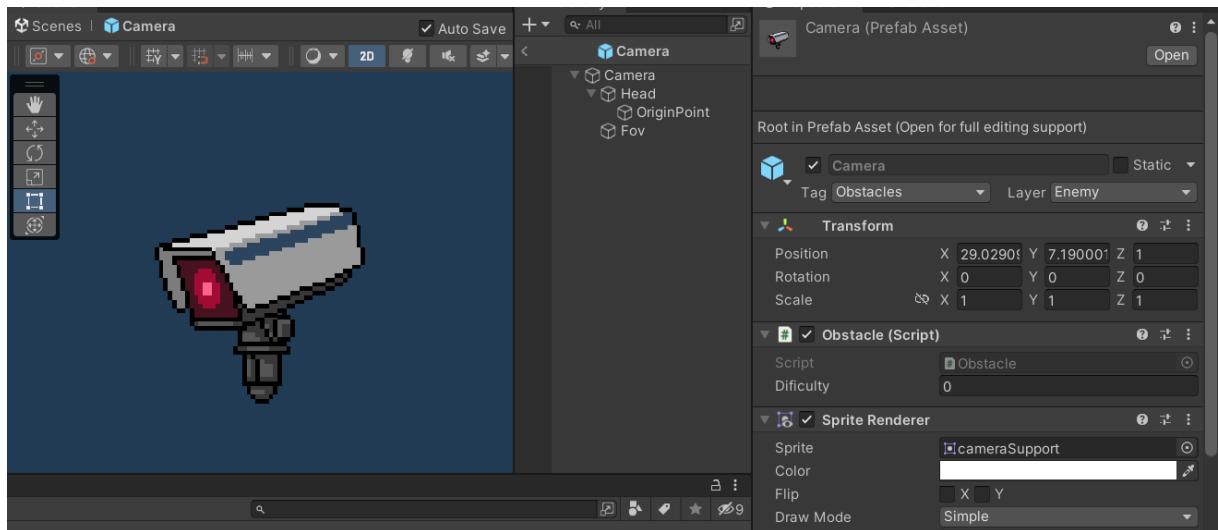


Figura 9.4: *Camera prefab*

-
- **Módulo *Scenes*.** Contiene las escenas en las que el jugador interacciona con los demás objetos. Existe la escena *MainScene* que corresponde al menú principal y

9. Arquitectura del sistema

después se encuentran los mapas que a cada uno corresponde una escena y también un *scriptableObject*. Este es una clase que permite almacenar grandes cantidades de datos compartidos independientes de instancias de *script*. En este caso se utilizan para almacenar la información de los mapas (nombre, escena, tiempo, etc.) y de esa forma se puede configurar fácilmente desde el editor.

- **Módulo *Scripts*.** Contiene los *scripts*, que son las clases que controlan el comportamiento de los objetos en la escena. Están divididos entre aquellos que controlan todo sobre la aplicación y los menús, y aquellos que controlan el *gameplay* y los mapas. El lector puede encontrar más información sobre cada uno de estos *scripts* en el manual de código de este TFG.
- **Módulo *Sounds*.** Contiene los ficheros de música y efectos de sonido utilizados en la aplicación.

CAPÍTULO 10

DISEÑO DE LA INTERFAZ

En este capítulo se establece la interfaz de la aplicación y se detallan las elecciones realizadas respecto a la disposición y ubicación de los diversos elementos en la pantalla, así como los menús que la conforman.

10.1. Menú principal

Es la ventana principal que se muestra al abrir la aplicación y permite acceder a los diferentes módulos de esta. En la figura A.8 se muestra su diseño.



Figura 10.1: Menú principal

- **1. Botón Puntuaciones.** Muestra un registro con las mejores puntuaciones obtenidas en los diferentes mapas.
- **2. Botón Iniciar Partida.** Abre el menú de selección de mapas para empezar una partida.
- **3. Botón Configuración.** Permite ajustar elementos de la configuración de la aplicación.
- **4. Botón Créditos.** Permite ver los créditos de la aplicación que contienen información de interés sobre la misma.
- **5. Botón Salir.** Permite cerrar la aplicación.

Para los botones se han utilizado iconos sin ningún tipo de texto debido a su simpleza y a que resulta bastante más intuitivo y agradable para el usuario.

10.2. Menú de puntuaciones

En este menú se muestran las 10 mejores puntuaciones, y una media de las mismas, obtenidas en cada dificultad para cada mapa. En la figura A.9 se muestra su diseño.



Figura 10.2: Menú de puntuaciones

- **1. Botones de selección.** Permiten desplazarse entre mapas.
- **2. Botón del mapa.** Al pulsar sobre el nombre del mapa se actualizará el panel de información con los resultados.
- **3. Panel de información.** Muestra en orden las 10 mejores puntuaciones para cada dificultad y, al final, la media de estas.
- **4. Botón de retorno.** Permite volver al menú principal.

10.3. Menú de configuración

El menú de configuración permite ajustar el volumen tanto de la música como de los efectos de sonido. En la figura A.10 se muestra su diseño.

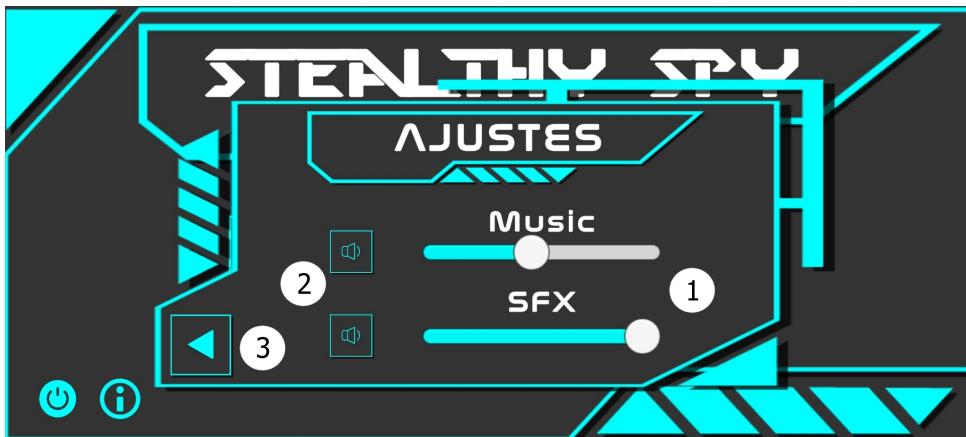


Figura 10.3: Menú de configuración

- **1. Barras de volumen.** El usuario puede desplazar estas barras para ajustar el volumen de la música o de los efectos.
- **2. Botón Silenciar.** Al pulsar sobre estos botones se establece el volumen a cero.
- **3. Botón de retorno.** Permite volver al menú principal.

10.4. Menú de selección de mapas

En este menú, el usuario puede seleccionar la dificultad y el mapa para comenzar una nueva partida. En la figura A.11 se muestra su diseño.



Figura 10.4: Menú de selección de mapas

- **1. Selector de dificultad.** Permite seleccionar la dificultad de la partida.
- **2. Selector de mapa.** Permiten desplazarse entre los diferentes mapas.
- **3. Botón Comenzar.** Para empezar la partida, el usuario solo debe pulsar el botón del mapa correspondiente. Además, sobre el botón se muestra información sobre la mejor puntuación obtenida en ese mapa para la dificultad seleccionada. También se muestra la puntuación media.
- **4. Botón de retorno.** Permite volver al menú principal.

10.5. Interfaz del juego

Esta es la interfaz con la que el jugador interactúa durante una partida. En la figura A.12 se muestra su diseño.



Figura 10.5: Interfaz del juego

- **1. Botón de pausa.** Permite abrir el menú de pausa.
- **2. Temporizador.** Muestra el tiempo restante para superar el mapa.
- **3. Joystick.** Permite al usuario desplazar al personaje.
- **4. Botón Saltar.** Permite al usuario hacer saltar al personaje si este está sobre el suelo.
- **5. Botón Incapacitar.** Permite al usuario incapacitar a los enemigos si se dan las condiciones.

10.6. Menú de pausa

Este menú es accesible durante el desarrollo de la partida y permite al usuario volver al menú principal o ajustar el volumen de la música y efectos de sonido. En la figura A.13 se muestra su diseño.

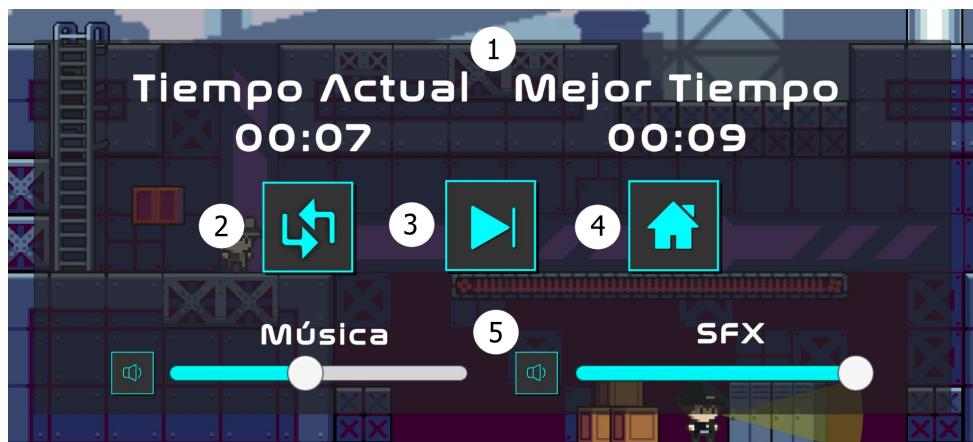


Figura 10.6: Menú de pausa

- **1. Panel de información.** A la izquierda se muestra el tiempo empleado en la partida actual y a la derecha el mejor tiempo registrado para ese mapa en la dificultad actual.
- **2. Botón Reiniciar.** Permite empezar la misma partida desde cero.
- **3. Botón Reanudar.** Permite reanudar la partida actual.
- **4. Botón *Home*.** Permite finalizar la partida y volver al menú principal.
- **5. Ajustar volumen.** Permite ajustar el volumen de la música y efectos de sonido como en el menú de configuración.

10.7. Menú de fin de partida

Existen dos versiones de este menú:

Cuando el jugador consigue superar el mapa se muestra la siguiente ventana (véase la figura A.14):



Figura 10.7: Menú fin de partida 1

- **1. Panel de información.** A la izquierda se muestra el tiempo empleado en la partida actual y a la derecha el mejor tiempo registrado para ese mapa en la dificultad actual. Si el nuevo tiempo es superior al anterior se mostrará en color amarillo.
- **2. Botón *Home*.** Permite finalizar la partida y volver al menú principal.

10.7. Menú de fin de partida

Cuando el jugador fracasa se muestra la siguiente ventana (véase la figura A.15):



Figura 10.8: Menú fin de partida 2

- **1. Mensaje de fracaso.** Se muestra la razón por la que el jugador fracasó (ser detectado o fin de tiempo).
- **2. Botón Reiniciar.** Permite empezar la misma partida desde cero.
- **3. Botón *Home*.** Permite finalizar la partida y volver al menú principal.

10.8. Menú de créditos

En este menú se puede ver información sobre este TFG. En la figura A.16 se muestra su diseño.



Figura 10.9: Menú de créditos

- **1. Panel de información.** Muestra información sobre la aplicación.
- **2. Botón Retorno.** Permite volver al menú principal.

Parte IV

Pruebas

CAPÍTULO 11

PRUEBAS

Durante las diversas etapas de desarrollo de la aplicación, se han llevado a cabo diversos procesos de pruebas y verificación con el fin de identificar y solucionar los posibles errores de codificación y diseño que pudieran surgir en cualquier momento.

A continuación se van a describir, a modo de ejemplo y para no extender en demasía el documento, algunos de las pruebas realizadas sobre la aplicación desarrollada.

11.1. Pruebas de aceptación

Se centran en determinar si un sistema o una aplicación cumple con los criterios de aceptación y los requisitos establecidos. Se llevan a cabo para verificar si el sistema se comporta de acuerdo con los casos de uso.

11.1.1. Caso de prueba sobre CU-1, “Comenzar juego”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-1, “Comenzar juego”:

- **Caso de prueba 1.** El componente *MapManager* crea y coloca en el *Scrollbar* los correspondientes *MapButton* al entrar al menú de selector de mapas.

En este caso de prueba se producía un error al entrar repetidamente al menú de selección de mapas, siempre y cuando no se realizara ninguna partida, que provocaba que se duplicaran los mapas. Esto se debía a que la función *CreateButton* estaba suscrita al evento *OnMapMenu*, lo que provocaba que cada vez que se activara el menú de selección de mapas se volvieran a crear nuevos botones. Para solucionarlo solo era necesario desuscribir la función una vez que finalizaba.

- **Caso de prueba 2.** Los botones para desplazar el *Scrollbar* funcionan de forma correcta. Este caso de prueba no detectó ningún error.

11.1.2. Caso de prueba sobre CU-1.1, “Seleccionar dificultad”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-1.1, “Seleccionar dificultad”:

- **Caso de prueba 1.** La información de cada *MapButton* se actualiza cuando se modifica la dificultad actual.

Este caso de prueba presentaba un error donde al cambiar la dificultad a Normal o Difícil, no se mostraba correctamente la puntuación media. Este error deriva de un error detectado en el **Caso de prueba 1 del CU-3, “Ver puntuaciones”**.

- **Caso de prueba 2.** La dificultad seleccionada se mantiene al salir del menú o al volver a entrar después de una partida. Este caso de prueba no detectó ningún error.

11.1.3. Caso de prueba sobre CU-1.3, “Pausar”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-1.3, “Pausar”:

- **Caso de prueba 1.** El botón de pausa activa el menú de pausa deteniendo la partida e impidiendo al usuario controlar al personaje. Este caso de prueba no detectó ningún error.
- **Caso de prueba 2.** El panel de información muestra el tiempo empleado en la partida actual y la mejor puntuación registrada en ese mapa para esa dificultad.

Este caso de prueba producía un *bug* donde siempre mostraba la misma puntuación media independientemente de la dificultad actual. Se trataba a un errata en el código donde se leía siempre la media de la dificultad Fácil.

11.1.4. Caso de prueba sobre CU-2.1, “Ajustar volumen de la música”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-2.1, “Ajustar volumen de la música”:

- **Caso de prueba 1.** El volumen de la música se configura adecuadamente. Este caso de prueba no detectó ningún error.
- **Caso de prueba 2.** El volumen de la música se almacena correctamente en la memoria manteniéndose entre mapas y entre sesiones del usuario. Este caso de prueba no detectó ningún error.
- **Caso de prueba 3.** El botón para silenciar música funciona correctamente.

Este caso de prueba generaba un *bug* visual que consistía en que el ícono del botón no se modificaba correctamente después de reestablecer el sonido. Se debía a que

11. Pruebas

no se habían asignado correctamente los iconos cuando se bajaba el volumen a cero manualmente.

11.1.5. Caso de prueba sobre CU-2.2, “Ajustar volumen de los efectos de sonido”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-2.2, “Ajustar volumen de los efectos de sonido”:

- **Caso de prueba 1.** El volumen de los efectos de sonido se configura adecuadamente.
Este caso de prueba no detectó ningún error.
- **Caso de prueba 2.** El volumen de los efectos de sonido se almacena correctamente en la memoria manteniéndose entre mapas y entre sesiones del usuario. Este caso de prueba no detectó ningún error.
- **Caso de prueba 3.** El botón para silenciar efectos de sonido funciona correctamente.

Este caso de prueba generaba un *bug* visual como el explicado en el **Caso de prueba 3 del CU-2.1 “Ajustar volumen de música”**.

11.1.6. Caso de prueba sobre CU-3, “Ver puntuaciones”

A continuación se exponen algunos de los casos de prueba realizados sobre el caso de uso CU-3, “Ver puntuaciones”:

- **Caso de prueba 1.** Las puntuaciones guardadas se cargan a la aplicación correctamente.

Este caso de prueba produjo un error en el que no se leía correctamente la puntuación media de cada dificultad al iniciar la aplicación. Como no se descubrió la causa del error se optó por calcular manualmente la media al arrancar la aplicación.

- **Caso de prueba 2.** Al entrar al menú de puntuaciones se crea correctamente un *ScoreButton* para cada mapa. Este caso de prueba no detectó ningún error.

11.1.7. Caso de prueba sobre la interfaz de la aplicación

A continuación se exponen algunos de los casos de prueba realizados sobre la interfaz de la aplicación:

- **Caso de prueba 1.** Al cargar la interfaz de juego se deben mostrar los controles de movimiento e incapacitar, el temporizador, el botón de pausa, y se han cargado los enemigos y el *tilemap*. Este caso de prueba no detectó ningún error.
- **Caso de prueba 2.** Los controles de movimiento y salto e incapacitar deben reaccionar a los *inputs* del jugador. Este caso de prueba no detectó ningún error.
- **Caso de prueba 3.** El temporizador debe actualizarse correctamente. Este caso de prueba no detectó ningún error.
- **Caso de prueba 4.** El botón de incapacitar solo puede usarse en las condiciones adecuadas.

Este caso de prueba generaba un *bug* que permitía utilizar el botón de incapacitar aunque no se encontrara un enemigo cerca. El problema era que no se desactivaba el botón después de haber detectado a un enemigo la primera vez.

11.1.7.1. Caso de prueba sobre los menús de la aplicación

A continuación se expone como ejemplo el resultado de las pruebas realizadas sobre los menús de la aplicación:

- **Caso de prueba 1.** La transición entre los distintos menus se produce de forma correcta.

Este caso de prueba presenta un pequeño *bug* con los *scrollbars* del menú de mapas y el menú de puntuaciones. En ambos, a partir de la segunda vez que se accedía a

ellos, los *scrollbars* aparecen situados en el último botón en lugar del primero como debería ser. Se intentó forzar la variable *value* a cero, que representa la posición inicial del *scrollbar*, y también se intentó mover la posición de este al cargar la interfaz. Sin embargo, ninguno de los métodos fue efectivo y no se encontró forma de solucionarlo.

- **Caso de prueba 2.** Las operaciones en los diversos menús se llevan a cabo sin problemas. Este caso de uso no detectó ningún error.

11.2. Pruebas de integración

Se centran en comprobar cómo interaccionan los diferentes componentes del software cuando se combinan y se comunican entre sí. El objetivo es detectar problemas de interoperabilidad y asegurar que los componentes integrados funcionen correctamente.

11.2.1. Caso de prueba 1

En esta prueba, se verificó que al iniciar una partida sin realizar cambios en la dificultad, se inicia automáticamente una partida en el mapa elegido con la configuración de dificultad establecida en fácil. Este caso de prueba no detectó ningún error.

11.2.2. Caso de prueba 2

En esta prueba, se verificó que al modificar la dificultad esta se refleja al iniciar una nueva partida. Para ello, se probaron en los diferentes mapas a iniciar una partida en cada dificultad y se comprobó que el tiempo disponible y los enemigos que aparecían eran los correspondientes a cada dificultad. Este caso de prueba no detectó ningún error.

11.2.3. Caso de prueba 3

En esta prueba, se verificó que al finalizar exitosamente una partida, la nueva puntuación se actualizara de forma correcta. Para ello, al finalizar una partida se accedía al menú de puntuaciones y se comprobaba que se hubiera almacenado la nueva puntuación correctamente.

Este caso de prueba producía un error en la función *RecalculateScores* de la clase *MapManager* debido a que por un fallo en la codificación de la función la nueva puntuación no se registraba adecuadamente. Fue necesario rehacer parte de la función.

11.2.4. Caso de prueba 4

En esta prueba, se comprobó que el personaje recibía correctamente los *inputs* del *joystick* y los botones. Para ello, se jugaron algunas partidas intentando realizar actos imposibles, como doble salto, usar el botón de incapacitar cuando no está permitido, usar el *joystick* para saltar, etc.

En este caso de prueba producía un *bug* que consistía en que el personaje realizaba la acción de incapacitar aunque no hubiera ningún enemigo cerca. Esto se debía a que no se desactivaba correctamente el botón cuando no se detectaban enemigos cercanos.

11.3. Pruebas de usuario

Son pruebas realizadas por personas externas al desarrollo, que representan a los usuarios finales del sistema. Estas pruebas se centran en evaluar la usabilidad, la facilidad de uso y la experiencia general del usuario al interactuar con la aplicación.

Para realizar estas pruebas se distribuyó una copia de la aplicación a varias personas y posteriormente se realizó una serie de preguntas sobre su experiencia en la aplicación. A continuación se muestran diferentes cambios o errores producidos a raíz de estas pruebas:

11. Pruebas

- Se detectó un *bug* que permitía al jugador escalar por las paredes saltando. El error era producido porque las paredes, junto con todo el escenario que posee colisiones, pertenece a la capa *Ground*, capa que el personaje detecta como suelo. Por tanto, si saltabas pegándote a las paredes, el personaje detectaba que estabas en el suelo y permitía volver a saltar. Para solucionarlo, se movieron los *tiles* que era paredes a otro *Tilemap* con una capa diferente.
- Algunos usuarios tenían dificultades al realizar saltos muy ajustados. Para intentar mejorar esto, se desplazó un poco el componente *GroundCheckPoint*, que se encarga de detectar el suelo, un poco hacia la parte posterior del personaje, en lugar de en el centro, para que los jugadores tengan más tiempo de reacción. De esta forma, el personaje seguirá detectando suelo aunque parte del *sprite* ya este fuera de la plataforma.
- Algunos usuarios consideraban que la música de fondo a máximo volumen solapaba a los demás sonidos. Se ha optado por capar el volumen máximo de la música para evitar esto.
- Algunos usuarios consideraban que el botón de retorno de los menús era demasiado pequeño. Se aumentó un poco su tamaño para que sea más cómodo.
- Se realizaron ajustes al salto del personaje según las críticas recibidas. En primer lugar, se modificó el código para que el personaje saltara en cuanto el usuario pulsara el botón y no al soltarlo, como funciona en *Unity* por defecto. También se permitió que al mantener el botón de salto el personaje realizara el siguiente salto al tocar el suelo tras unos segundos.

Parte V

Conclusiones y futuras mejoras

CAPÍTULO 12

CONCLUSIONES Y FUTURAS MEJORAS

En este capítulo se llevará a cabo un análisis de las conclusiones obtenidas. Se realizará una retrospectiva para verificar si se han cumplido los objetivos propuestos en el capítulo 3 de este manual y se discutirán todos los conocimientos adquiridos durante la ejecución del proyecto. Finalmente se comentarán un serie de futuras mejoras de la aplicación.

12.1. Conclusiones sobre los objetivos formales

Estos objetivos se refieren a los conocimientos adquiridos durante el desarrollo de este TFG. Tras el desarrollo del mismo, se ha cumplido lo siguiente:

- Se ha experimentado de primera mano cada uno de los procesos que componen el desarrollo de un videojuego, desde la lluvia de ideas y el diseño hasta la codificación

12. Conclusiones y futuras mejoras

y las pruebas.

- Se ha aprendido a utilizar el motor gráfico de *Unity* desde cero y utilizarlo para desarrollar un videojuego.
- Se ha diseñado una interfaz simple y fácil de entender por el usuario.
- Se ha escrito y desarrollado toda la documentación del proyecto, compuesta por el manual técnico, el manual de usuario y el manual de código.

12.2. Conclusiones sobre los objetivos operacionales

Estos objetivos son las metas específicas que la aplicación debía cumplir una vez finalizado el proyecto. Tras la finalización del mismo, se ha cumplido lo siguiente:

- Se ha desarrollado un videojuego de plataformas 2D para dispositivos *Android*.
- Se han diseñado varios mapas y tres dificultades para el desarrollo de las partidas.
- Se ha implementado un espia como personaje principal, sus movimientos a lo largo de los mapas y su capacidad de incapacitación sobre los guardias.
- Se ha creado un registro de puntuaciones para que el jugador pueda consultar sus mejores tiempos en cada mapa y en cada dificultad.
- Se han implementando obstáculos que añaden dificultad y vida al juego.
- Se ha desarrollado un módulo de configuración para que el jugador pueda ajustar el volumen de la música y de los efectos de sonido de la aplicación.
- Se ha desarrollado un entorno amigable y fácil de usar para que el usuario no tenga dificultad a la hora de entender el funcionamiento de la aplicación.
- Se han implementado enemigos incapacitantes del jugador espia a lo largo de los mapas.
- Se ha implementado un sistema de temporización relacionado con la finalización de una partida.

12.3. Futuras mejoras

El videojuego ha cumplido los requisitos que se plantearon al inicio del proyecto, sin embargo, durante el desarrollo de la aplicación y durante las pruebas de usuario han surgido ideas o aspectos a mejorar que podrían considerarse para futuras mejoras en la aplicación o en proyectos similares:

- **Mejorar el apartado gráfico.** Al haberse utilizado *assets* gratuitos, las limitaciones eran muy grandes a la hora de crear un buen apartado gráfico. Estas limitaciones se deben a la escasez de recursos disponibles, la dificultad para encontrar *assets* que se complementen entre sí y la falta de experiencia del desarrollador en diseño gráfico. Para futuras versiones podría considerarse utilizar algunos *assets* de pago o el aprendizaje de herramientas de diseño gráfico y animación.
- **Variedad de mapas y desafíos.** La idea del juego es contar con una gran variedad de mapas en los que el jugador pueda probar su habilidad. Esto incluye tanto nuevos mapas como nuevos tipos de obstáculos y mecánicas, como podría ser, por ejemplo, interruptores para accionar puertas.
- **Subir la aplicación a la *Google Play Store*.** Subir la aplicación a esta plataforma permitiría que muchos usuarios pudieran conocer el juego.
- **Puntuaciones y *rankings* globales.** Sería la idea más ambiciosa, pues consistiría en crear una clasificación global entre todos los jugadores donde cada jugador podría comparar sus puntuaciones con la de los demás. Esto implicaría un sistema de registro en la red, además de una base de datos que albergue esa información.

BIBLIOGRAFÍA

- [1] *En 2023 los videojuegos generarán 200 billones de dólares, más del 50 por ciento será por el sector móvil.* <https://mundoejecutivo.com.mx/tecnologia/en-2023-los-videojuegos-generaran-200-billones-mas-del-50-sera-por-el-sector-movil/>.
- [2] Alex Pareja: *La Evolución del Fenómeno Indie*, 2015. <https://www.startvideojuegos.com/la-evolucion-del-fenomeno-indie/>.
- [3] Reseña: *Hollow Knight – Calidad y cantidad en la cima del desarrollo indie*. <https://comiqueros.cl/resena-hollow-knight-calidad-y-cantidad-en-la-cima-del-desarrollo-indie/>.
- [4] Alberto Carrasco Carrasco: *¿Qué es un motor de videojuegos?*, 2018. <https://blogs.upm.es/observatoriogate/2018/07/04/que-es-un-motor-de-videojuegos/>.
- [5] *Plataforma de desarrollo en tiempo real de Unity / Motor de videojuegos.* <https://unity.com/es>.
- [6] *Unreal Engine: The most powerful real-time 3D creation tool.* <https://www.unrealengine.com/es-ES>.

BIBLIOGRAFÍA

- [7] Wikijuegos: *Géneros de videojuegos*. https://videojuegos.fandom.com/es/wiki/G%C3%A1neros_de_videojuegos.
 - [8] Wikijuegos: *Videojuego de plataformas*. https://videojuegos.fandom.com/es/wiki/Videojuego_de_plataformas.
 - [9] NintendoWiki: *Historia de los juegos de plataformas: Mundo 1-1*. https://nintendo.fandom.com/es/wiki/Super_Mario_Bros.
 - [10] Simone Belli y Cristian López Raventós: *Breve historia de los videojuegos*. Athenea Digital. Revista de Pensamiento e Investigación Social, (14), 2008, ISSN 1578-8946. <https://www.redalyc.org/pdf/537/53701409.pdf>.
 - [11] JJVelasco: *Tennis for Two, uno de los primeros videojuegos de la historia*. <https://hipertextual.com/2011/07/tennis-for-two-uno-de-los-primeros-videojuegos-de-la-historia>.
 - [12] Wikipedia: *Spacewar!* <https://es.wikipedia.org/wiki/Spacewar!>
 - [13] Angelus LLC: *Pong / Análisis de videojuegos*. <https://tus-videojuegos.com/pong/>.
 - [14] Pedro Díaz San Miguel: *LOS PLATAFORMAS 2D*, 2019. <https://portal.33bits.net/los-plataformas-bidimensionales/>.
 - [15] *Donkey Kong, el primer exitazo mundial de Nintendo*. <https://www.vidaextra.com/analisis/donkey-kong-primer-exitazo-mundial-nintendo>.
 - [16] Wikipedia: *Pitfall!* <https://en.wikipedia.org/wiki/Pitfall!>
 - [17] *Por qué el ‘Manic Miner’ de Matthew Smith fue un éxito absoluto*. <https://www.teknoplof.com/2020/10/20/por-que-el-manic-miner-de-matthew-smith-fue-un-exito-absoluto/>.
 - [18] Wikipedia: *Jet Set Willy*. https://es.wikipedia.org/wiki/Jet_Set_Willy.
-

BIBLIOGRAFÍA

- [19] Wikipedia: *Impossible Mission*. https://en.wikipedia.org/wiki/Impossible_Mission.
- [20] *Prince of Persia (1989)*. [https://princeofpersia.fandom.com/es/wiki/Prince_of_Persia_\(1989\)](https://princeofpersia.fandom.com/es/wiki/Prince_of_Persia_(1989)).
- [21] Wikipedia: *Metroid*. <https://es.wikipedia.org/wiki/Metroid>.
- [22] Wikipedia: *Castlevania*. <https://es.wikipedia.org/wiki/Castlevania>.
- [23] *Mega Man (Videojuego)*. [https://megaman.fandom.com/es/wiki/Mega_Man_\(Videojuego\)](https://megaman.fandom.com/es/wiki/Mega_Man_(Videojuego)).
- [24] Wikipedia: *Sonic the Hedgehog (serie)*. [https://es.wikipedia.org/wiki/Sonic_the_Hedgehog_\(serie\)](https://es.wikipedia.org/wiki/Sonic_the_Hedgehog_(serie)).
- [25] *Rayman*. <https://es.wikipedia.org/wiki/Rayman>.
- [26] Super Mario 64, un título que marcó un antes y un después, cumple 25 años. <https://vandal.elespanol.com/noticia/1350745584/super-mario-64-un-titulo-que-marco-un-antes-y-un-despues-cumple-25-anos/>.
- [27] *Crash Bandicoot*. [https://crash.fandom.com/es/wiki/Crash_Bandicoot_\(videojuego\)](https://crash.fandom.com/es/wiki/Crash_Bandicoot_(videojuego)).
- [28] *Hollow Knight*. https://es.wikipedia.org/wiki/Hollow_Knight.
- [29] David Plaza: *Ori and the Blind Forest, análisis de un juego con luz propia*. <https://xombitgames.com/2015/03/analisis-ori-and-the-blind-forest>.
- [30] Análisis *Ori and the Will of the Wisps* para Nintendo Switch. https://es.wikipedia.org/wiki/Ori_and_the_Will_of_the_Wisps.
- [31] *Visual Studio Community*. <https://visualstudio.microsoft.com/es/vs/community/>.

BIBLIOGRAFÍA

- [32] *Piskel*. <https://www.piskelapp.com/>.
- [33] *GitHub*. <https://github.com>.
- [34] *¿Qué es la arquitectura basada en eventos?* <https://www.redhat.com/es/topics/integration/what-is-event-driven-architecture>.
- [35] *Device Simulator 3.0*. <https://docs.unity3d.com/Packages/com.unity.device-simulator@3.0/manual/index.html>.

Parte VI

Apéndice

APÉNDICE A

MANUAL DE USUARIO

A continuación, se especifican los requisitos mínimos para ejecutar la aplicación, los pasos a seguir para instalar y desinstalarla y una guía sobre su uso.

A.1. Estructura del manual

Este manual se divide en los siguientes apartados:

- Requisitos del sistema.
- Instalación.
- Desinstalación.
- Guía de uso.

A.2. Requisitos del sistema

El requisito mínimo para poder ejecutar la aplicación es:

- Dispositivo con versión del sistema operativo *Android* igual o superior a 5.1

Si se cumple este requisito se puede asegurar que el dispositivo podrá ejecutar la aplicación correctamente, ya que no requiere muchos recursos.

A.3. Instalación de la aplicación

Como ejemplo de instalación se va a utilizar un dispositivo móvil con versión 13 de *Android*. Para llevar a cabo la instalación es necesario realizar los siguientes pasos:

- **Paso 1.** Abra el siguiente enlace en su navegador y pulse en “Download” para comenzar la descarga.
<https://www.mediafire.com/file/gpo3zr4d1ecia75/StealthySpy.apk/file>
- **Paso 2.** Por seguridad, el navegador preguntará si se desea descargar el archivo como se muestra en la figura A.1. Pulse “Descargar de todos modos” para descargar el archivo.

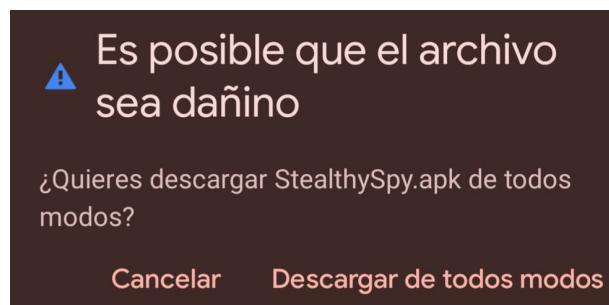


Figura A.1: Ventana de aviso por archivo dañino

- **Paso 3.** Al terminar la descarga, el sistema detectará que no se trata de una aplicación de *Google Play Store*. El sistema mostrará una alerta, como en la figura A.2 y pedirá que autorice al navegador para descargar archivos de origen desconocido. Pulse “Ajustes” y marque la casilla de su navegador como se muestra en la figura A.3

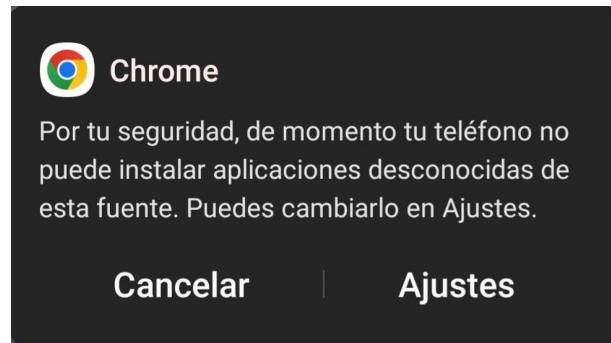


Figura A.2: Ventana de aviso por origen desconocido

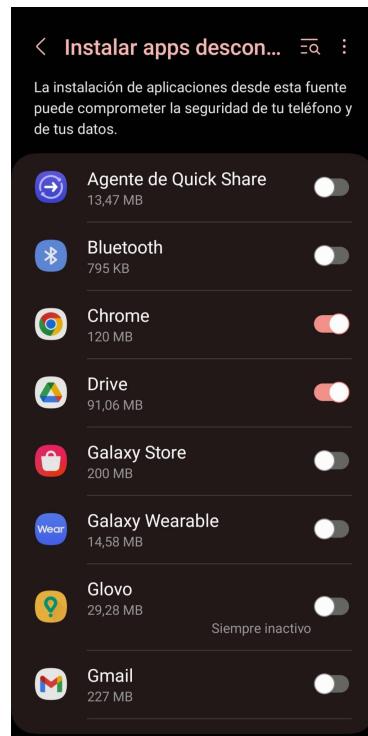


Figura A.3: Ventana de autorización de *Android*

- **Paso 4.** En la ventana que aparece, pulse la opción “Instalar”, tal y como muestra la figura A.4. Si no apareciera automáticamente la ventana, debe ir a las descargas de su navegador y pulsar sobre el archivo.

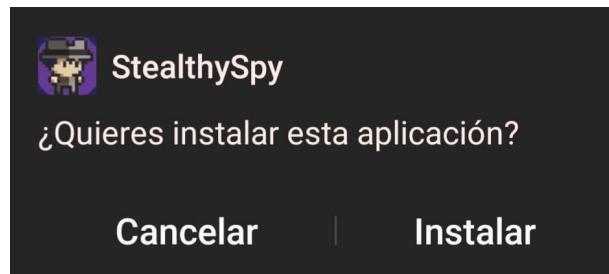


Figura A.4: Ventana de instalación

- **Paso 5.** Por último aparecerá una última ventana, como en la figura A.5. Esto significa que la aplicación ha sido instalada con éxito. Pulse “Abrir” para abrir la aplicación, o “Hecho” para continuar donde estaba.

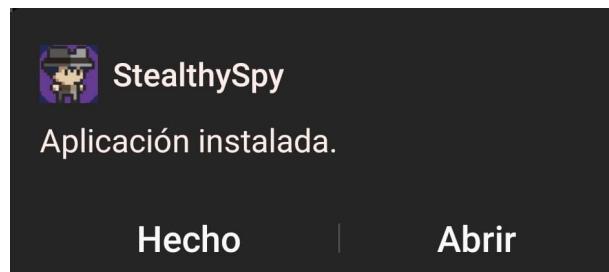


Figura A.5: Stealthy Spy instalado

A.4. Desinstalación de la aplicación

Como ejemplo de desinstalación se va a utilizar un dispositivo móvil con versión 13 de *Android*. Para llevar a cabo la desinstalación es necesario realizar los siguientes pasos:

- **Paso 1.** Manten pulsado el ícono de la aplicación y aparecerá un pequeño menú similar al de la figura A.6. Pulse la opción “Desinstalar”.

A.4. Desinstalación de la aplicación



Figura A.6: Desinstalar aplicación 1

- **Paso 2.** El sistema preguntará si se desea desinstalar la aplicación como muestra la figura A.7. Pulse “Aceptar”. Al finalizar, la aplicación habrá sido eliminada de su dispositivo.

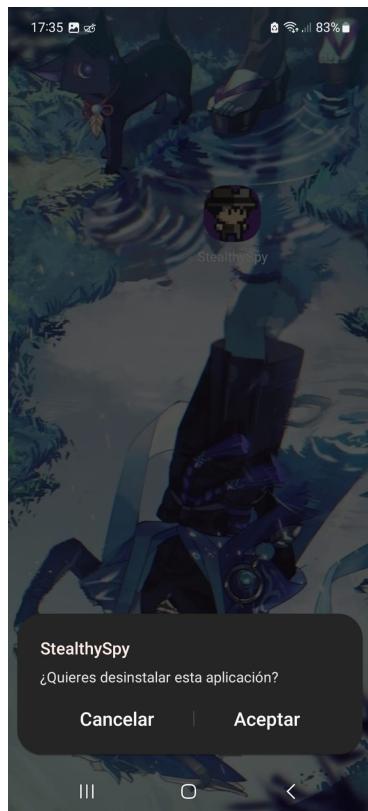


Figura A.7: Desinstalar aplicación 2

A.4.1. Guía de uso

En el siguiente apartado se detalla la interfaz de la aplicación, explicando cómo navegar en ella y brindando una descripción de la estructura y el contenido de las pantallas que conforman la aplicación.

A.4.1.1. Menú principal

Es la ventana principal que se muestra al abrir la aplicación y permite acceder a los diferentes módulos de esta. En la figura A.8 se muestra su diseño.



Figura A.8: Menú principal

- **1. Botón Puntuaciones.** Muestra un registro con las mejores puntuaciones obtenidas en los diferentes mapas.
- **2. Botón Iniciar Partida.** Abre el menú de selección de mapas para empezar una partida.
- **3. Botón Configuración.** Permite ajustar elementos de la configuración de la aplicación.
- **4. Botón Créditos.** Permite ver los créditos de la aplicación que contienen información de interés sobre la misma.
- **5. Botón Salir.** Permite cerrar la aplicación.

A.4.1.2. Menú de puntuaciones

En este menú se muestran las 10 mejores puntuaciones, y una media de las mismas, obtenidas en cada dificultad para cada mapa. En la figura A.9 se muestra su estructura.



Figura A.9: Menú de puntuaciones

- **1. Botones de selección.** Permiten desplazarse entre mapas.
- **2. Botón del mapa.** Al pulsar sobre el nombre del mapa se actualizará el panel de información con los resultados.
- **3. Panel de información.** Muestra en orden las 10 mejores puntuaciones para cada dificultad y, al final, la media de estas.
- **4. Botón de retorno.** Permite volver al menú principal.

A.4.1.3. Menú de configuración

El menú de configuración permite ajustar el volumen tanto de la música como de los efectos de sonido. En la figura A.10 se muestra su estructura.

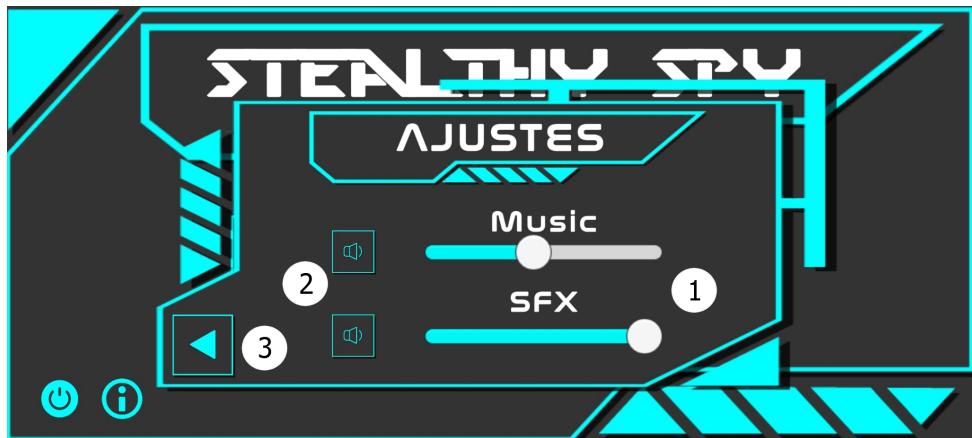


Figura A.10: Menú de configuración

- **1. Barras de volumen.** Pueden desplazarse estas barras para ajustar el volumen de la música o de los efectos.
- **2. Botón Silenciar.** Al pulsar sobre estos botones se establece el volumen a cero.
- **3. Botón de retorno.** Permite volver al menú principal.

A.4.1.4. Menú de selección de mapas

En este menú, se puede seleccionar la dificultad y el mapa para comenzar una nueva partida. En la figura A.11 se muestra su estructura.



Figura A.11: Menú de selección de mapas

- **1. Selector de dificultad.** Permite seleccionar la dificultad de la partida.
- **2. Selector de mapa.** Permiten desplazarse entre los diferentes mapas.
- **3. Botón Comenzar.** Para empezar la partida, solo debe pulsar el botón del mapa correspondiente. Además, sobre el botón se muestra información sobre la mejor puntuación obtenida en ese mapa para la dificultad seleccionada. También se muestra la puntuación media.
- **4. Botón de retorno.** Permite volver al menú principal.

A.5. Interfaz del juego

Esta es la interfaz que se muestra durante el transcurso de una partida. En la figura A.12 se muestra su diseño.

Durante la partida, el jugador puede desplazarse por el mapa, el cual está lleno de guardias y cámaras de vigilancia que podrán detectar al jugador. El objetivo es llegar al final del mapa antes de que el temporizador, situado en la parte superior de la pantalla, llegue a cero, en cuyo caso perderá la partida. También contará como fracaso el ser detectado por algún enemigo.

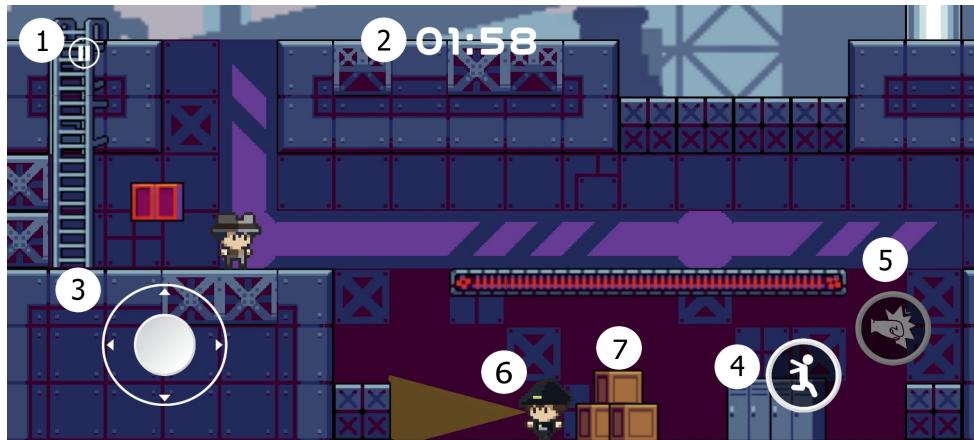


Figura A.12: Interfaz del juego

- **1. Botón de pausa.** Permite abrir el menú de pausa.
- **2. Temporizador.** Muestra el tiempo restante para superar el mapa.
- **3. Joystick.** Permite desplazar al personaje.
- **4. Botón Saltar.** Permite hacer saltar al personaje si este está sobre el suelo.
- **5. Botón Incapacitar.** Permite al usuario incapacitar a los enemigos. Esto solo es posible si el jugador se encuentra detrás del enemigo. Al pulsar el botón el enemigo quedará en el suelo durante unos segundos.
- **6. Enemigo.** El jugador debe evitar su área de visión para evitar ser detectado. Existen dos tipos: las cámaras, que están situadas en una posición fija y rotan su

campo de visión, y los guardias que se desplazan por el mapa. Estos últimos pueden ser incapacitados como se mencionó antes.

- **7. Cajas** En algunas zonas hay situadas unas cajas donde el jugador puede esconderse y evitará ser detectado mientras permanezca ahí.

A.6. Menú de pausa

Este menú es accesible durante el desarrollo de la partida y permite volver al menú principal o ajustar el volumen de la música y efectos de sonido. En la figura A.13 se muestra su diseño.

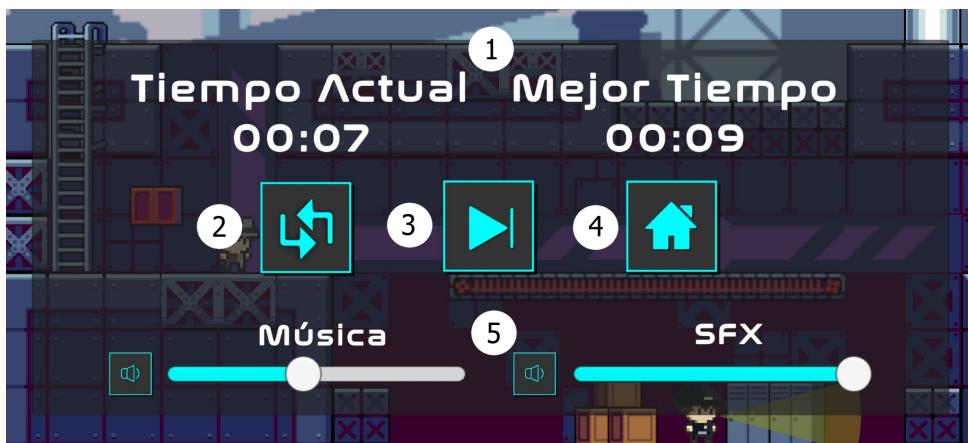


Figura A.13: Menú de pausa

- **1. Panel de información.** A la izquierda se muestra el tiempo empleado en la partida actual y a la derecha el mejor tiempo registrado para ese mapa en la dificultad actual.
- **2. Botón Reiniciar.** Permite empezar la misma partida desde cero.
- **3. Botón Reanudar.** Permite reanudar la partida actual.
- **4. Botón *Home*.** Permite finalizar la partida y volver al menú principal.
- **5. Ajustar volumen.** Permite ajustar el volumen de la música y efectos de sonido como en el menú de configuración.

A.7. Menú de fin de partida

Existen dos versiones de este menú:

Cuando el jugador consigue superar el mapa se muestra la siguiente ventana (véase la figura A.14):



Figura A.14: Menú fin de partida 1

- **1. Panel de información.** A la izquierda se muestra el tiempo empleado en la partida actual y a la derecha el mejor tiempo registrado para ese mapa en la dificultad actual. Si el nuevo tiempo es superior al anterior se mostrará en color amarillo.
- **2. Botón *Home*.** Permite finalizar la partida y volver al menú principal.

Cuando el jugador fracasa se muestra la siguiente ventana (véase la figura A.15):



Figura A.15: Menú fin de partida 2

- **1. Mensaje de fracaso.** Se muestra la razón por la que el jugador fracasó (ser detectado o fin de tiempo).
- **2. Botón Reiniciar.** Permite empezar la misma partida desde cero.
- **3. Botón *Home*.** Permite finalizar la partida y volver al menú principal.

A.8. Menú de créditos

En este menú se puede ver información sobre el desarrollo de la aplicación. En la figura A.16 se muestra su diseño.



Figura A.16: Menú de créditos

- **1. Panel de información.** Muestra información sobre la aplicación.
- **2. Botón Retorno.** Permite volver al menú principal.