



Escuela
Politécnica
Superior

Videojuego Multiplataforma



Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autora:

Sandra María Garzón Hernández

Tutor/es:

Miguel Ángel Lozano Ortega

Septiembre 2015



Universitat d'Alacant
Universidad de Alicante

Agradecimientos

El proyecto final de carrera representa la culminación de cualquier estudio universitario y el haberlo alcanzado finalmente, después de un gran esfuerzo y dedicación significa mucho y es todo un orgullo para mí. Agradecer a todas las personas que han hecho posible este proyecto:

Al profesorado de Ingeniería Multimedia por formarme como ingeniera, y en especial a mi tutor Miguel Ángel Lozano, por tener tanta paciencia conmigo y ayudarme en todo lo que necesitaba.

A mi familia, a mis padres y a mi hermana, por todo su apoyo incondicional y comprensión que me han brindado en todo momento a lo largo de mi vida y que sin ellos esto no habría sido posible.

A mi pareja David, amigos y compañeros de universidad, por estos cuatro años de carrera que siempre han estado ahí para solucionar dudas, apoyarme y animarme cuando lo necesitaba, además de haber convertido la etapa universitaria en la mejor etapa de mi vida.

Gracias.

Índice de Contenidos

1. INTRODUCCIÓN.....	15
1.1 RESUMEN	15
1.2 MOTIVACIÓN	15
1.3 OBJETIVOS	15
1.4 ESTRUCTURA DEL DOCUMENTO	16
2. LOS VIDEOJUEGOS.....	19
2.1 HISTORIA DE LOS VIDEOJUEGOS.....	19
2.2 ACTUALIDAD DE LOS VIDEOJUEGOS	21
2.3 LOS VIDEOJUEGOS EN DISPOSITIVOS MÓVILES.....	21
3. METODOLOGÍAS Y HERRAMIENTAS	24
3.1 METODOLOGÍA EMPLEADA.....	24
3.2 HERRAMIENTAS EMPLEADAS	24
3.3 ETAPAS EN EL DESARROLLO DE UN VIDEOJUEGO	27
4. MOTORES DE VIDEOJUEGOS.....	30
4.1 EJEMPLOS DE MOTORES.....	30
4.2 HERRAMIENTA USADA PARA EL PROYECTO	33
5. DISEÑO Y ESPECIFICACIÓN	34
5.1 ARGUMENTO.....	34
5.2 CONJUNTO DE CARACTERÍSTICAS	34
5.3 GÉNERO	34

5.4 AUDIENCIA.....	36
5.5 ASPECTO GRÁFICO.....	36
5.6 PERSONAJE.....	36
5.7 ZONAS.....	37
5.8 NIVELES.....	38
5.9 JUGABILIDAD.....	38
5.10 OBJETIVOS DEL JUEGO.....	39
5.11 FLUJO DEL JUEGO.....	39
5.12 MECÁNICAS.....	40
5.13 MOVIMIENTO.....	41
5.14 OBJETOS.....	41
5.15 ACCIONES Y PODERES.....	43
5.16 OBSTÁCULOS.....	44
5.17 ENEMIGOS.....	45
5.18 RETOS Y LOGROS.....	45
5.19 PANTALLAS.....	46
5.19.1 Flujo general de las pantallas.....	46
5.19.2 Pantallas del Juego.....	47
6. DESARROLLO E IMPLEMENTACIÓN.....	51
6.1 ESTRUCTURA GENERAL.....	51
6.2 PATRONES DE DISEÑO.....	52
6.2.1 Patrón Singleton.....	52

6.3 IMPLEMENTACIÓN DE EFECTOS	52
6.3.1 Fade In y Fade Out	53
6.3.2 Vibración de pantalla	55
6.3.3 Otros efectos	55
6.4 MOTOR DE FÍSICAS	55
6.4.1 Colisiones	56
6.4.1.1 Colisiones con Objetos y Obstáculos	57
6.4.1.2 Colisiones con límites de pantalla	59
6.5 GENERADORES DE OBJETOS Y OBSTÁCULOS	59
6.6 INTELIGENCIA ARTIFICIAL	59
6.8 GESTIÓN DE INPUTS	61
6.9 GRÁFICOS	63
6.9.1 Assets	63
6.9.2 Sprites	64
6.9.3 Animaciones	68
6.9.4 Resoluciones	69
6.9.4.1 Viewports en LibGDX	69
6.10. AUDIO	73
6.10.1 Efectos de Sonido	73
6.10.2 Música Ambiente	76
6.11 REQUISITOS FUNCIONALES	77
6.11.1 HUD	81

6.11.2 Controles	84
6.12 REQUISITOS NO FUNCIONALES	85
7. PUBLICACIÓN.....	87
7.1 GOOGLE PLAY	87
7.1.1 Publicación en Google Play	88
7.2 APP STORE.....	90
7.2.1 Publicación en App Store.....	91
8. PUBLICIDAD.....	93
8.1 ESTRATEGIAS DE PUBLICIDAD Y NEGOCIO EXISTENTES	93
8.2 ESTRATEGIA DE PUBLICIDAD Y NEGOCIO ELEGIDA.....	95
9. ESTUDIO DE VIABILIDAD.....	97
9.1 VIABILIDAD ECONÓMICA	97
9.2 VIABILIDAD TÉCNICA.....	97
9.3 VIABILIDAD LEGAL	98
9.4 ESTIMACIÓN DE COSTES	98
9.5 ANÁLISIS DAFO.....	101
9.6 ANÁLISIS DE RIESGOS	102
9.7 CONCLUSIÓN	105
10. HARDWARE Y SOFTWARE.....	106
10.1 HERRAMIENTAS HARDWARE EMPLEADAS.....	106
10.2 HERRAMIENTAS SOFTWARE EMPLEADAS	106
10.3 REQUISITOS HARDWARE Y SOFTWARE.....	106

11. CONCLUSIONES.....	107
11.1 OBJETIVOS ALCANZADOS	107
11.2 LÍNEAS FUTURAS DE TRABAJO	107
12. ANEXO: MANUAL DE INSTRUCCIONES	109
13. BIBLIOGRAFÍA Y REFERENCIAS	119

Índice de Figuras

Figura 1: Ejemplo actual de juego para móvil - Angry Birds	22
Figura 2: Ejemplo de juego actual para móvil - Cut the Rope.....	22
Figura 3: Compras incluidas en el juego Jetpack Joyride	23
Figura 4: Listas y Tarjetas en la herramienta Trello	25
Figura 5: Tarjetas de la herramienta Trello.....	26
Figura 6: Diagrama de las etapas de desarrollo de un videojuego.....	27
Figura 7: Bocetos del diseño previo del videojuego	28
Figura 8: ejemplo de runner, Wind Runner.....	35
Figura 9: ejemplo de runner por niveles, Bit Trip Runner.....	35
Figura 10: Concept del personaje principal, LHED.....	37
Figura 11: Sprite del personaje principal, LHED ampliada (izquierda) original (derecha). 37	
Figura 12: Movimiento del personaje principal	41
Figura 13: Objeto engranaje.....	42
Figura 14: Objeto batería	42
Figura 15: Objeto estrella de energía	42
Figura 16: Objeto esfera de salud.....	42
Figura 17: Obstáculo bomba	44
Figura 18: Explosión del obstáculo bomba	44
Figura 19: Explosión del obstáculo bomba	44
Figura 20: Enemigo del nivel de Jefe.....	45
Figura 21: Diagrama de flujo general de pantallas	46
Figura 22: Concept pantalla de comenzar	47
Figura 23: Concept menú principal	48
Figura 24: Concept menú de opciones	49
Figura 25: Bloques de un videojuego	51

Figura 26: Fade In	53
Figura 27: Fade Out	54
Figura 28: Fade de ambientación	54
Figura 29: Bounds	57
Figura 30: Diagrama funcional del método ColisionarConItems	58
Figura 31: Diagrama funcional del método ColisionarConObstaculos.....	58
Figura 32: IA de los cañones - movimiento	60
Figura 33: Comportamiento cañones del enemigo.....	61
Figura 34: Área de inputs	62
Figura 35: Inputs durante la partida	63
Figura 36: Assets del proyecto.....	64
Figura 37: Spritesheet del Personaje Principal.....	65
Figura 38: Spritesheet de los obstáculos	66
Figura 39: Spritesheet de los objetos	66
Figura 40: Spritesheet del HUD	66
Figura 41: Spritesheet de los escenarios.....	67
Figura 42: Frames de la animación del personaje corriendo	68
Figura 43: Visualización del StretchViewport	70
Figura 44: Visualización del FitViewport	71
Figura 45: Visualización del FillViewport	71
Figura 46: Visualización del ScreenViewport	72
Figura 47: Panel de control del programa gratuito para efectos de sonido SFXR	74
Figura 48: HUD del jugador - Nivel de recogida	81
Figura 49: Retratos del jugador - Nivel de recogida	82
Figura 50: Corazones de vida del jugador	82
Figura 51: HUD del jugador - Nivel jefe	83
Figura 52: Modo ataque	83

Figura 53: Modo defensa	83
Figura 54: Retratos del jugador para diálogos.....	84
Figura 55: HUD del jefe.....	84
Figura 56: Interfaz de Google Play	87
Figura 57: Google Play - Developer Console	88
Figura 58: Google Play - Añadir nueva aplicación.....	89
Figura 59: Google Play - Ficha para rellenar de la aplicación.....	89
Figura 60: Interfaz de App Store	90
Figura 61: Developer Apple - Member Center	91
Figura 62: iTunes Connect - Gestionar Aplicaciones	91
Figura 63: Ejemplo de posición y tipo de banner publicitario	96
Figura 64: Manual de Instrucciones - LHED.....	109
Figura 65: Manual de Instrucciones - Shiro	110
Figura 66: Manual de Instrucciones - Menú Selección de nivel.....	111
Figura 67: Manual de Instrucciones - Nivel de recogida	112
Figura 68: Manual de Instrucciones - Nivel de Jefe	113
Figura 69: Manual de Instrucciones - Combinaciones de botones	115
Figura 70: Manual de Instrucciones - Menú de pausa	116
Figura 71: Manual de Instrucciones - Game Over.....	117

Índice de Tablas

Tabla 1: Efectos de sonido empleados	76
Tabla 2: Música ambiente empleadas	76
Tabla 3: Requisitos Funcionales	80
Tabla 4: Requisitos no Funcionales	86
Tabla 5: Tareas y estimación de tiempos	100
Tabla 6: Estimación de costes	101
Tabla 7: Análisis DAFO	102
Tabla 8: Riesgo - Avería en la estación de trabajo.....	103
Tabla 9: Riesgo - Error de planificación	103
Tabla 10: Riesgo - Problemas de disponibilidad	103
Tabla 11: Riesgo - Desmotivación	104
Tabla 12: Riesgo - Errores de compatibilidad de herramientas	104
Tabla 13: Riesgo - Sobreestimación o infravaloración de las tareas	105

Terminología

- **Aspect Ratio:** relación de aspecto de una imagen, es la proporción entre su ancho y su alto.
- **Bounds:** referencia a la caja de colisión de un objeto.
- **Bug:** error en el software que puede hacer que el programa falle.
- **Concept:** concepto
- **Cooldown:** tiempo de espera para volver a realizar una acción.
- **Feedback:** reacción, respuesta u opinión que nos proporciona un interlocutor.
- **Frame:** Fotograma, imagen concreta dentro de una sucesión de imágenes en movimiento.
- **FPS:** Frames Per Seconds o Frames or Segundo.
- **Frame Rate:** también conocido como Frame Frequency, es la frecuencia con la que un dispositivo muestra imágenes consecutivas por pantalla.
- **Game Over:** Fin de juego.
- **HUD:** Heads-Up Display o visualización frontal, información que en todo momento se muestra en pantalla durante el desarrollo de una partida.
- **IA:** Inteligencia Artificial
- **Line Runner:** género de videojuego que se caracteriza por avanzar de forma constante en una misma dirección y esquivar obstáculos.
- **Loop:** ejecución en bucle
- **Píxel:** la unidad más pequeña de una imagen digital.
- **Sprite:** mapa de bits 2D que se dibuja por pantalla.
- **Sprite Sheet:** hoja o archivo donde se agrupan varios sprites
- **State Time:** tiempo que se mostrará un *frame* antes de sustituirlo por el siguiente
- **Viewport:** región de visión de los gráficos de una pantalla.

1. Introducción

1.1 Resumen

Hoy en día los dispositivos móviles se han transformado en una gran plataforma para el mundo de los videojuegos. Gracias a la evolución del hardware y de los recursos disponibles, actualmente los móviles son capaces de ofrecernos una enorme variedad de juegos. Sin embargo, esto también supone un problema para los desarrolladores, debido a la gran cantidad de móviles con características diferentes (resoluciones de pantalla, arquitecturas, sistemas operativos...). El desarrollar un videojuego para móviles que funcione en la gran mayoría de dispositivos es todo un reto para el desarrollador, por ello existen diversas librerías y herramientas que van dirigidas a resolver este problema, permitiendo desarrollar videojuegos y portarlos a las plataformas principales de móviles existentes.

1.2 Motivación

La motivación principal que ha llevado a realizar este proyecto es el desarrollo de un videojuego lo más completo posible acorde con la demanda actual del mercado, incluyendo la realización de todos los gráficos y parte visual del mismo y la publicación en las principales plataformas de distribución (Google Play y App Store). Por otro lado, que sirva también de experiencia y resulte un reto personal.

Aunque a lo largo de la carrera he tenido oportunidad de desarrollar otros proyectos similares de realización de un videojuego, siendo el último el desarrollado en cuarto de carrera en el ABP (Aprendizaje Basado en Proyectos), este es el primero que abordo en solitario que incluye todas las fases de un proyecto real, lo que me permite afianzar con más fuerza los conocimientos adquiridos en el grado.

1.3 Objetivos

Los objetivos generales de este proyecto son realizar un videojuego multiplataforma para dispositivos móviles empleando los conocimientos adquiridos a lo largo del grado de ingeniería multimedia cursado, diseñándolo de forma adecuada para su funcionamiento correcto en diferentes plataformas. Además, estudiar el mercado actual de videojuegos

para móviles y las tendencias de hoy en día, investigar y aprender el funcionamiento del motor que se usará para desarrollar el juego y cómo exporta a diferentes plataformas.

Con este proyecto se pretende alcanzar la meta de obtener el máximo de conocimiento sobre el desarrollo de un videojuego y al final tener un producto preparado para mostrar al público y al mercado.

1.4 Estructura del Documento

La estructura de este documento se ha enfocado de forma que no solo incluye la información y el contenido necesario para una memoria de un Proyecto de Fin Grado, sino que al ser el desarrollo de un videojuego, se ha enfocado una parte del mismo como un *Game Design Document* (GDD), que consiste en un documento de diseño de videojuegos. Este tipo de documento es muy útil a la hora de explicar todo el contenido de un juego y todos los requisitos que contendrá.

A continuación se enumerarán y se explicarán brevemente los apartados de este documento:

1. Introducción

La primera parte del documento donde nos encontramos en este momento. En este apartado se pretende hacer una introducción del proyecto en general, explicar los motivos y los objetivos del mismo. De esta forma, al acabar este apartado, se tendrá una visión de en qué consiste el proyecto y por qué se ha realizado.

2. Videojuegos

A continuación se hará una pequeña introducción al mundo de los videojuegos, su historia, actualidad y un estudio de mercado. Se estudiará además la demanda de los videojuegos en los dispositivos móviles hoy en día.

3. Metodologías

El siguiente apartado explicará las metodologías de trabajo y por qué son tan importantes a la hora de realizar un proyecto o trabajo. Se describirán las herramientas utilizadas para llevar organizadas las tareas y las etapas del desarrollo que se han llevado a cabo para hacer este proyecto.

4. Motores de Videojuegos

La cuarta parte del documento consiste en la introducción a los motores de videojuegos y la investigación de algunos más utilizados hoy en día para el desarrollo de videojuegos. También se explicará el motor utilizado en este proyecto, el porqué y las ventajas y desventajas que tiene frente a otros motores.

5. Diseño y Especificación

El apartado de diseño y especificación se centrará en explicar los aspectos más generales del videojuego a modo de GDD, como se mencionó anteriormente. En este apartado se incluirá el argumento, el género, la audiencia a la que va dirigida, qué aspecto gráfico tendrá, etc, además de explicar la jugabilidad, la mecánica y todos los elementos del mismo.

6. Desarrollo e Implementación

El apartado de desarrollo e implementación explicará cómo se ha desarrollado el videojuego. En este apartado se mostrará la estructura general, los patrones de diseño empleados, los efectos que se han desarrollado para el apartado visual del juego, el motor de físicas y la inteligencia artificial entre otros. También se incluye el apartado de Gráficos, que explicará cómo funcionan los *sprites* y las animaciones en el motor del juego y se mostrarán algunos de los gráficos usados para el proyecto, y en Audio se describirá cómo se han desarrollado los efectos de sonido y la música que incluirá el juego.

Por último en este punto, se describirán los Requisitos Funcionales, los Requisitos no Funcionales y el HUD del videojuego.

7. Publicación

En el siguiente apartado de Publicación se describirán las principales plataformas de publicación en dispositivos móviles y cómo publicar un juego en ellas.

8. Publicidad

En el apartado de Publicidad se explicará la API utilizada y se describirán los diversos métodos y estrategias para llegar al máximo números de usuarios posibles.

9. Estudio de Viabilidad

En el apartado de Estudio de viabilidad se indicará una estimación de los costes del proyecto y un análisis detallado de los riesgos junto a un análisis DAFO.

10. Hardware y Software

El apartado Hardware y Software contendrá un resumen de los programas y herramientas usadas en el proyecto para cada apartado del videojuego y su documento.

11. Conclusiones

En este apartado se mostrarán las conclusiones que se han obtenido realizando este proyecto y una vez finalizado, también los objetivos alcanzados y una valoración sobre el trabajo desarrollado. Además, las líneas futuras con los aspectos ampliables, como pueden ser nuevas funcionalidades y puntos en lo que se puede mejorar.

12. Anexo: Manual de Instrucciones

El penúltimo apartado, el Anexo, consiste en un manual de instrucciones donde se explicará el juego y servirá como tutorial para los usuarios.

13: Bibliografía y Referencias

Por último, la memoria concluye con la Bibliografía y Referencias, donde se expondrán enlaces de interés que se han utilizado para la investigación y el estudio de este proyecto y temas de interés relacionados con el tema del trabajo.

2. Los Videojuegos

Como es bien conocido, los videojuegos son una tecnología o aplicación interactiva orientada sobre todo al entretenimiento. Funciona a través de mandos o controles y permite simular experiencias mostradas en la pantalla de un televisor, un monitor de ordenador u otro dispositivo electrónico. Un videojuego recrea entornos virtuales en los que un jugador o varios jugadores pueden controlar un personaje o elemento del juego para conseguir unos determinados objetivos mediante unas reglas determinadas que se definen en el propio juego.

La diferencia que existe entre los videojuegos y otras formas de entretenimiento, como por ejemplo las películas, es que los videojuegos deben ser interactivos con el usuario, es decir, que el usuario, o en este caso el jugador, debe involucrarse activamente con el contenido del juego. A continuación se explicará cómo los videojuegos han llegado a ser lo que son hoy en día.

2.1 Historia de los videojuegos

El mundo de los videojuegos ha estado en constante desarrollo y ha experimentado una progresión imparable en todos los aspectos, pero para entender esto mejor es necesario echar un vistazo a sus orígenes y también a las compañías que han conseguido que este mercado llegase a ser como es ahora.

Durante bastante tiempo ha estado presente el debate de cuál es el primer videojuego de la historia, debido a todas las definiciones que se han ido estableciendo, pero se puede considerar el que ya usaba los gráficos de la computadora ESDAC, el Nought and Crosses, también conocido como OXO, desarrollado por Alexander S. Douglas en el año 1952.

Este juego pasó desapercibido porque solo fue mostrado de forma personal al profesorado y a los estudiantes de la Universidad de Cambridge. Permitía enfrentar a un jugador humano contra la máquina y consistía en una versión computarizada del tres en raya, el objetivo era conseguir completar una línea con los tres círculos o las tres cruces, siendo la línea horizontal, vertical o diagonal.

El siguiente juego que se desarrolló fue en 1958 por William Higginbotham, el Tennis for Two. Se trataba de un simulador de tenis de mesa para el entretenimiento de los visitantes del *Brookhaven National Laboratory*. Muchos consideran este el primer videojuego de la historia en vez del Nought and Crosses.

Tras el éxito del Tennis for Two, cuatro años más tarde un estudiante del Instituto de Tecnología de Massachussets dedicó seis meses de desarrollo del Space War, un videojuego que usaba gráficos vectoriales donde dos jugadores controlaban la dirección y la velocidad de dos naves que luchaban entre ellas. Más adelante se comenzaron a comercializar versiones de este en Estados Unidos a principios de los años 70.

El Pong fue el siguiente en alcanzar el éxito, pero en vez de comercializarse era utilizado en lugares públicos como bares o salones recreativos. Fue diseñado por Al Alcorn para Nolan Bushnell en la recién fundada Atari.

En 1972 se presentó el videojuego que fue la clave en los videojuegos como industria, el Space Invaders. En los siguientes años los videojuegos comenzaron a experimentar numerosos avances técnicos en cuanto a microprocesadores y chips de memoria, y fue cuando comenzaron a salir los sistemas domésticos como la Intellivision, la Colecovision y la Atari.

Pero aparte de los juegos para estas consolas, también triunfaron las máquinas recreativas con juegos como Pacman, Tron o Zaxxon.

En la década de los 80, Japón dio un fuerte giro a la industria con el éxito de la Famicon, conocida en occidente como la NES, consola desarrollada por Nintendo, mientras que en Europa se decantaba más por los microordenadores como el Spectrum o el Commodore 64.

La NES fue adoptada por los norteamericanos como principal sistema de videojuegos. A continuación fueron apareciendo nuevas consolas por parte de otras compañías como SEGA, con la Master System.

Las consolas evolucionaron considerablemente a partir de la década de los 90 gracias a la competición de la llamada "generación de 16 bits" compuesta por la Mega Drive, la SNES, o también Super Famicon, el PC Engine de NEC, conocido en occidente como Turbografx y la CPS Charger de Capcom. A su vez aparecieron más consolas y se introdujo la tecnología del CD-ROM. Los juegos siguieron evolucionando y se comenzó a trabajar con entornos tridimensionales (Doom 3D), principalmente en el campo del

ordenador, que rápidamente fueron ocupando un lugar importante en el mercado. Fue cuando surgió la llamada "generación de 32 bits" y salieron al mercado consolas como la PlayStation de Sony, la Sega Saturn y la Nintendo 64.

Las consolas portátiles también constituyeron un gran auge al mundo de los videojuegos, apareciendo la Game Boy, Game Gear o la Neo Geo Pocket. Durante muchos años fueron las dominadoras del mercado, siendo Nintendo el que más éxito tenía con sus consolas, debido a que siempre apostaba por la jugabilidad y mecánica de sus juegos.

2.2 Actualidad de los videojuegos

La industria de los videojuegos ha experimentado en los últimos años un increíble crecimiento, debido al desarrollo de la computación, la mejora en la capacidad de procesamiento, el alcance de imágenes más reales y la estrecha relación entre el cine y los videojuegos, con lo cual los consumidores reconocen los títulos antes.

Hoy en día nos encontramos en la séptima generación de consolas, estando la octava en activo desarrollo. Han aparecido una enorme variedad de géneros de juegos para todos los públicos y cada día más desarrolladores se implican en este mercado.

En la actualidad, el sector de los videojuegos ha pasado a generar más dinero que el sector del cine y la música juntos. En el caso de España, la industria del videojuego llegó a triplicar los beneficios del cine en el año 2014.

2.3 Los videojuegos en dispositivos móviles

Los dispositivos móviles se han convertido en una de las plataformas más grandes para los desarrolladores de videojuegos, pero, ¿cuál es el secreto de su éxito?

Los videojuegos móviles han sabido abrirse paso en el mercado gracias a que son sencillos de jugar y muy intuitivos. Si echamos un vistazo a los videojuegos para consolas de sobremesa o portátiles, nos ofrecen un amplio catálogo de juegos y una gran variedad, sin embargo, nos es algo tan común llevarlas siempre encima, al contrario que los móviles, que se han convertido hoy en día en una herramienta muy necesaria y que siempre tenemos a mano, por lo que en cualquier momento y en cualquier lugar podemos echar unas partidas.

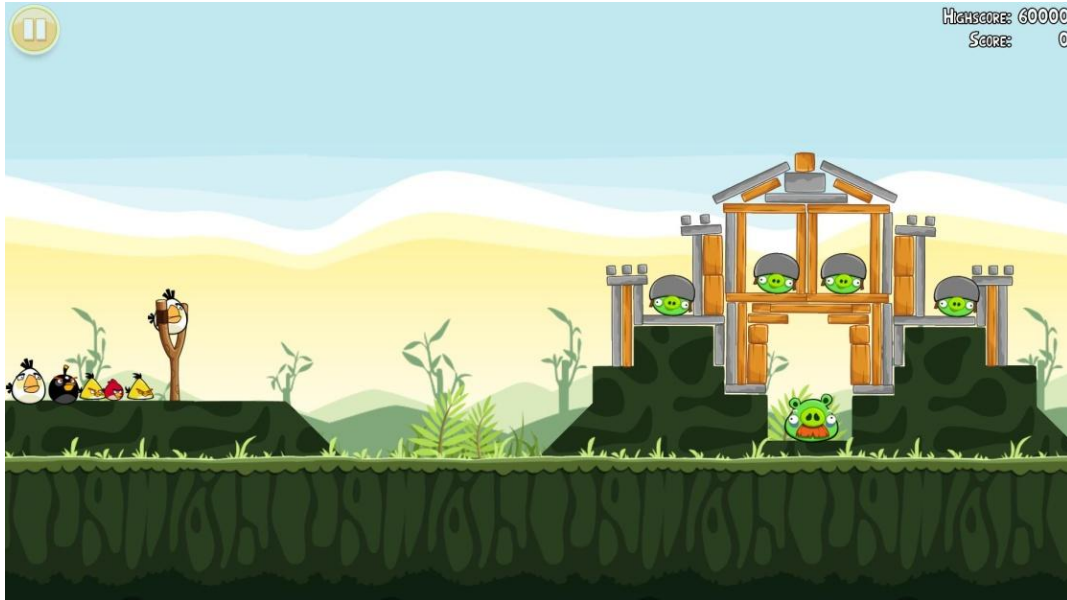


Figura 1: Ejemplo actual de juego para móvil - Angry Birds

A los juegos de móvil se les suelen llamar juegos casual, un nombre muy acertado ya que cualquiera que tenga un smartphone puede jugar. Con estos juegos se ha conseguido eliminar la barrera del público habitual, siendo cada vez más gente de todas las edades las que prueba estos juegos.



Figura 2: Ejemplo de juego actual para móvil - Cut the Rope

Otra razón del éxito de estos juegos es su bajo coste, llegando muchos de ellos a ser totalmente gratuitos o con modelos de negocio que permiten descargar el juego de forma

gratuita y pagando solo algunas mejoras o nuevas características del mismo. Este modelo de negocio es de los más utilizados hoy en día en la industria.



Figura 3: Compras incluidas en el juego Jetpack Joyride

3. Metodologías y Herramientas

Una de las fases más importantes y crítica a la hora de desarrollar un proyecto de grandes dimensiones como puede ser la elaboración de un videojuego es la organización y la planificación de este mismo. Tenemos que tener un orden y establecer distintas prioridades para que el proyecto que nos propongamos tenga su respectivo éxito, una mala organización y planificación puede desembocar en que el proyecto no llegue a finalizarse nunca, o que surjan más complicaciones y dificultades de las normales, que las tareas no se asignen de forma correcta y que aparezcan errores que retrasen el resultado del proyecto.

Para evitar eso, existen distintos tipos de metodologías de trabajo que se ajustan a cada proyecto y al grupo de trabajadores.

3.1 Metodología empleada

Para este proyecto se ha utilizado una metodología ágil, debido a que se consideraba que era la que mejor encajaba con el tipo de proyecto y por eso es la más utilizada en el desarrollo de videojuegos para dispositivos móviles.

La metodología ágil se caracteriza por ser iterativa e incremental. Son aquellas que permiten adaptar la forma de trabajo a las condiciones que requiere el proyecto, consiguiendo así flexibilidad e inmediatez, mayor velocidad y eficiencia a la hora de trabajar.

La metodología ágil en concreto que se ha usado en este proyecto es la metodología SUM, ya que es de las más usadas en el desarrollo de videojuegos. Esta metodología surge como la necesidad de cumplir con el objetivo de un proyecto con escaso personal y recursos, al ser una metodología ágil, tiende a un desarrollo rápido, preciso, adaptable y optimizado, lo cual encaja a la perfección con este proyecto.

3.2 Herramientas empleadas

Para este proyecto se ha optado por la herramienta de organización Trello. Se trata de una herramienta de gestión de proyectos muy sencilla y realmente fácil de usar. Cuenta, además del servicio online en su página web oficial, de una aplicación para dispositivos

móviles Android e iOS, permitiendo así poder gestionar tu proyecto desde tu móvil en cualquier lugar.

¿Cómo funciona esta herramienta? Después de crearnos nuestra cuenta en la página web, nos aparecerá nuestro escritorio. En este escritorio es donde podremos organizar todas nuestras tareas mediante listas, que se muestran de manera vertical, como podemos ver en la imagen a continuación. Dentro de esas listas tenemos unos objetos llamados tarjetas, que se pueden arrastrar y soltar en otras listas o reordenarse de la manera que cada uno desee. Las listas también las podemos desplazar por todo nuestro escritorio y ordenarlas como queramos.

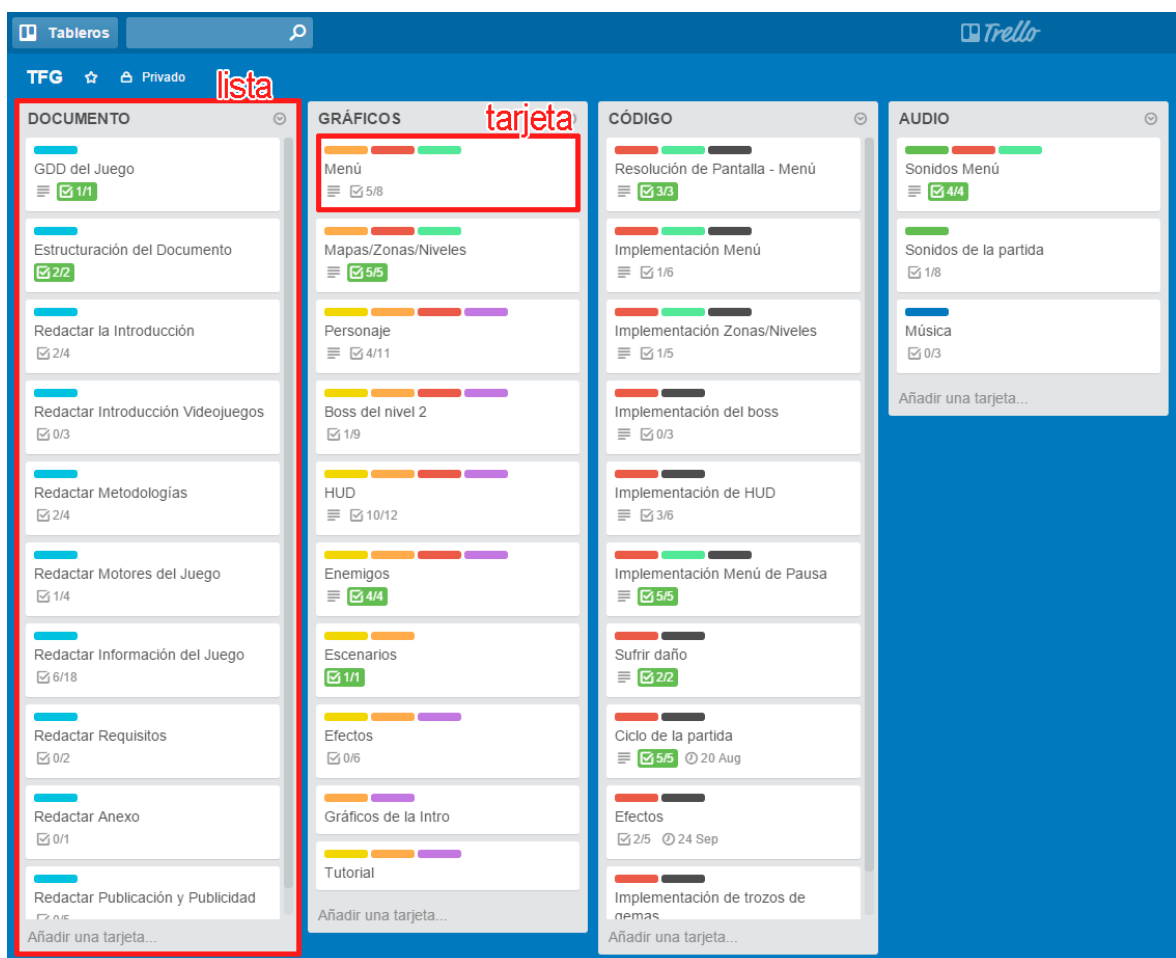


Figura 4: Listas y Tarjetas en la herramienta Trello

En cada tarjeta podemos tener listas de tareas, imágenes, archivos, asignarle etiquetas, añadir descripción y comentarios e incluso ponerle una fecha de vencimiento a esa tarea.

En el caso de este proyecto, se ha optado por dividir las tareas en cuatro grupos principales o listas, como se puede ver en la imagen anterior: Documento, gráficos,

código y audio. Dentro de cada lista se ha introducido todas las tareas relacionadas, por ejemplo, observamos en la siguiente imagen la tarea de los gráficos del menú. Se pueden observar los elementos que se han mencionado anteriormente.

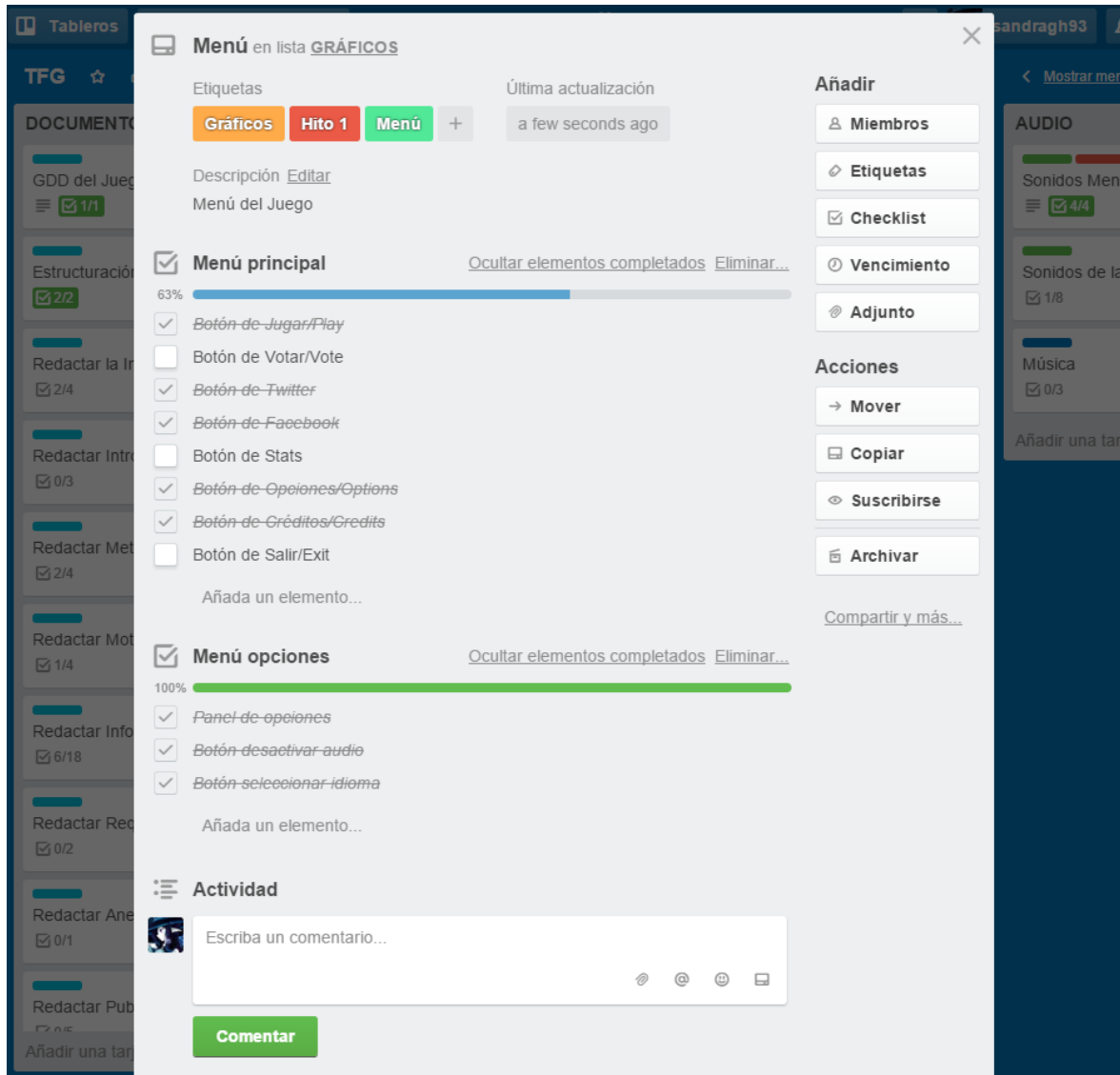


Figura 5: Tarjetas de la herramienta Trello

Las tarjetas de Trello son como pequeños post-it que se colocan en un tablón pero en formato digital, se puede buscar en ellas y compartirlas con otros usuarios. Además, si una tarea tiene una fecha de vencimiento, la herramienta te avisará, ya sea por correo electrónico o por la aplicación del móvil, si se tiene instalada de que esa tarea está a punto de finalizar.

Con esta herramienta se ha podido organizar de forma completa las tareas de este proyecto.

3.3 Etapas en el desarrollo de un videojuego

El desarrollo de un videojuego es un proceso largo que está formado por diferentes etapas. A pesar de que a la hora de desarrollar un proyecto, cada grupo de trabajadores o empresas establezcan sus propias etapas para desarrollar su producto, hay seis de ellas que son generales para todos, y son las seis siguientes:

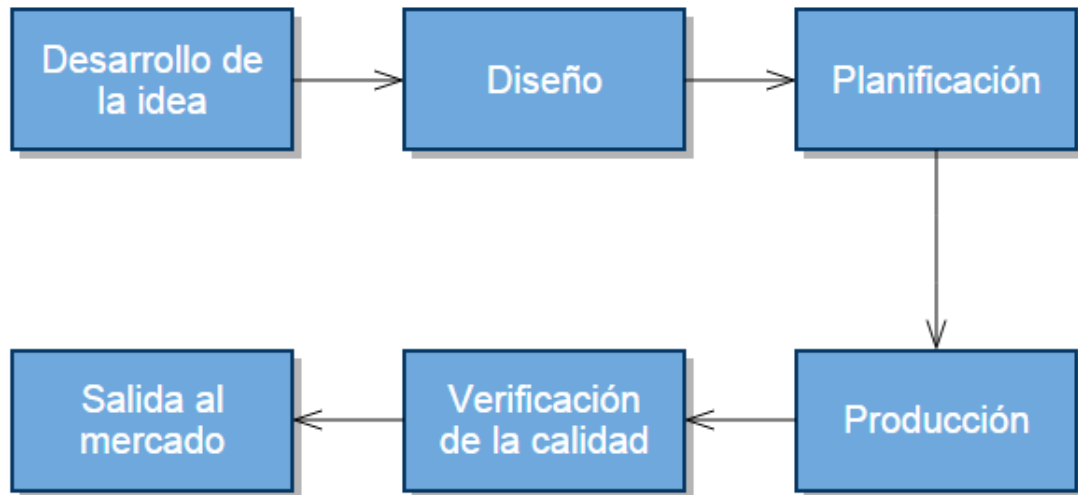


Figura 6: Diagrama de las etapas de desarrollo de un videojuego

Para este proyecto se han llevado a cabo estas mismas etapas, añadiendo algunos pasos que se han considerado oportunos.

Primera Etapa: Desarrollo de la idea

Antes de pensar en la idea del juego en sí, se realizó un pequeño estudio sobre los diferentes juegos de móvil que hoy en día resultan más atractivos para el público. Uno de los géneros que más me ha gustado desde siempre y que se trata de los más abundantes en los juegos para móviles es el género *runner*. Este género se caracteriza por el desplazamiento constante del personaje hacia la derecha o izquierda (aunque ésta última es menos común), y el objetivo es hacer que el personaje salte para esquivar los obstáculos pulsando la pantalla. Es una mecánica ideal para un juego de móvil, ya que se reducen los controles a principalmente uno, por lo que hace que sea una idea sencilla de jugar y bastante atractiva.

Pero no se quiere hacer para el proyecto simplemente un *runner*, se quiere que tenga algo distinto y que se diferencie de lo demás, así que manteniendo la mecánica de

manejar al personaje con un solo botón, se ha pensado en añadirle otros elementos que le dan más variedad, que sea un *runner* pero que a la vez tenga algo de acción, por lo que se ha pensado en implementar enemigos que consistan un reto de verdad y añadieran al juego un toque frenético y activo.

También se ha decidido la ambientación, me gusta mucho el tema futurista y se ha pensado que sería perfecto para los elementos de acción que se han querido añadir.

Segunda Etapa: Diseño

En esta etapa se empieza a desarrollar más detenidamente la idea, qué objetivos tendrá que conseguir el jugador, los niveles, cómo serán los enemigos, qué obstáculos habrá, los elementos que se irán recogiendo, cómo se progresa en el juego, etc. También se piensa en la historia de fondo y se realizan bocetos previos de cómo serán los niveles del juego y el personaje, sirviendo de guía para el diseño definitivo.



Figura 7: Bocetos del diseño previo del videojuego

Lo más importante de esta etapa es dejar el juego completamente pensado y definido, aunque haya detalles que podrán ser alterados, la jugabilidad principal tiene que estar clara desde el principio, esto sirve para poder hacerse una idea de si el juego puede ser entretenido y enganchar al jugador.

También se define la resolución del juego, el aspecto gráfico que tendría, la orientación de la pantalla y los controles.

Tercera Etapa: Planificación

Después de tener todo cuidadosamente detallado y definido, se pasa a establecer y dividir las tareas del desarrollo. Para esto se ha usado la herramienta Trello, explicada

anteriormente en el punto 3.2 Herramientas empleadas. Después de tener todas las tareas establecidas se indican fechas de vencimiento y límites para tener algún prototipo jugable con el que hacer pruebas, además de un estudio de viabilidad, como se puede ver en el apartado 9. Estudio de viabilidad.

En esta etapa también se investiga sobre los diversos motores de videojuegos y se elige el que se considera que se ajusta mejor a nuestras necesidades.

Cuarta Etapa: Producción

En esta etapa se comienza con el desarrollo en sí del juego. Lo primero que se realiza es un pequeño prototipo para ver si lo que se ha planificado se puede implementar sin problemas y la idea resulta atractiva. En el caso de que se vea algún fallo se procede a cambiar lo necesario. Para este prototipo no es necesario tener los gráficos definitivos del juego, es aconsejable usar figuras geométricas provisionales o usar incluso gráficos de otros juegos.

Después de comprobar que el prototipo funciona, se abarca el proyecto real, la programación y creación de los gráficos, animaciones, música, efectos de sonido, etc.

Quinta Etapa: Verificación de la calidad

Se trata de la última etapa del desarrollo del juego, una vez terminado una versión alpha, se realizan pruebas del mismo para comprobar que no hay errores ni ningún tipo de *bug*, que sea estable y que funcione en las plataformas que se tenían de objetivo.

Sexta Etapa: Salida al mercado y mantenimiento

Por último, la etapa final del desarrollo de un videojuego, la comercialización del mismo. En esta etapa se decide cómo se monetizará el juego, dónde se publicará y cómo se promocionará.

En este caso, se ha decidido sacar tras su finalización en Google Play de forma gratuita, obteniendo beneficio económico mediante la publicidad proporcionada por la plataforma AdMob. También se ha decidido tener en constante actualización para mantener enganchando a los jugadores y proporcionarles de forma constante nuevos elementos del juego.

4. Motores de Videojuegos

A la hora de empezar a desarrollar un videojuego es importante tener presente las herramientas de las que dispones y de las que necesitas, ya que por ejemplo, cuando hacer algún juego muy sencillo, como por ejemplo un juego de adivinar un número o el de las parejas, no se necesita tener demasiado código, pero cuando se quiere desarrollar un juego más complejo o de calidad comercial, es muy difícil hacer todo de forma manual. Muchos grupos de trabajo o empresas del sector se crean su propio motor de videojuegos o usan uno ya existente. Realizar tu propio motor no es para nada una tarea sencilla, ni mucho menos, por eso las empresas cada vez se inclinan más por usar un motor ya creado para centrarse más en sus productos.

Pero, ¿qué es exactamente un motor de videojuegos? Es un conjunto de herramientas de programación que permiten el diseño, la creación y la representación de un videojuego. Existen muchos tipos de motores, unos permiten crear videojuegos ofreciendo los recursos básicos necesarios mientras que con otros te permiten hacerlo sin escribir una línea de código, mediante interfaces de usuario gráficas o editores visuales. Hay motores que ofrecen herramientas para juegos en 2D y otros para juegos en 3D, o incluso para ambos, al igual que existen motores que operan tanto en consolas de videojuegos como en sistemas operativos.

Un motor de videojuegos se divide a su vez en varios motores. La funcionalidad básica de un motor es proveer al videojuego de un motor de renderizado para gráficos en 2D y 3D, es decir, un motor gráfico, también un motor físico o detector de colisiones, un motor de audio, de animaciones, inteligencia artificial, etc. Cada motor de videojuegos contiene más o menos motores, haciendo que el proceso de desarrollo de un videojuego varíe de forma notable por reutilizar o adaptar un mismo motor de videojuego para crear diferentes juegos.

4.1 Ejemplos de Motores

Existen una gran cantidad de motores de videojuegos hoy en día, cada uno ofreciendo sus propios recursos y sus propias características. Es difícil cuando se empieza a desarrollar un videojuego elegir el tipo de motor perfecto para usar, ya que obviamente lo normal que se suele buscar es que cubra todas nuestras necesidades y nos haga cuánto más fácil la tarea mejor.

Al igual que tenemos videojuegos en 2D y 3D, existen motores que se especializan en cada tipo de juego, o permiten desarrollar para ambos. Como hemos mencionado en las etapas de un desarrollo, el informarse de los diferentes motores actuales que existen es importante para decidir cuál se adapta mejor al proyecto que deseas realizar.

A continuación se mostrarán algunos ejemplos de algunos de los motores de videojuegos más utilizados hoy en día:

Source

Source es un motor de videojuegos desarrollado por Valve Corporations. Se trata de un motor multiplataforma orientado a ordenadores y consolas, permite portar los juegos desarrollados a Windows, Mac OS, Linux, XBOX 360 y PlayStation3.

Se puede descargar desde la plataforma de Steam y su uso es no comercial, sin embargo, se está desarrollando la nueva versión, Source 2, que será gratuita.

Página oficial: <http://source.valvesoftware.com>

Unity

Unity es un motor de videojuegos multiplataforma flexible y de gran potencia creado para diseñar juegos 3D y 2D para PC, consolas, dispositivos móviles y web. Fue desarrollado por Unity Technologies, siendo su objetivo el crear un motor que estuviera disponible y lo pudiera usar cualquier persona.

Una de las mayores ventajas de Unity es la gran cantidad de plataformas a las que se puede portar de forma muy sencilla nuestro juego, además nos ofrece una interfaz de usuario intuitiva y fácil de usar, pudiendo llegar a desarrollar un proyecto decente sin necesidad de programar.

Página Oficial: <http://unity3d.com/es/>

Unreal Engine 4

Unreal Engine 4 es un motor multiplataforma gratuito de gran potencia creado por la compañía Epic Games, que se ha convertido en uno de los principales motores gráficos usados en algunos juegos de la nueva generación de consolas.

Este motor permite crear juegos tanto en 2D como en 3D y portarlos a numerosas plataformas, como Android, iOS, PS4, XBOX ONE, Windows, MAC y Linux. Es muy versátil y con una gran cantidad de mejoras que no paran de añadir.

Las desventajas de Unreal Engine 4 es su complejidad a la hora de utilizarlo, que requiere muchas horas de trasteo para conseguir algo decente, y la necesidad de equipos de gran potencia para poder desarrollar en él.

Página oficial: <http://unrealengine.com>

LibGDX

Se trata de una *framework* para el desarrollo de videojuegos multiplataforma. Está enfocado a juegos de 2D, aunque con cada actualización se acerca más al 3D. Permite portar los juegos a los principales sistemas operativos de móviles (Android, iOS y BlackBerry), Windows, Mac, Linux y web (HTML5), de forma sencilla, escribiendo en un único proyecto el código del juego y exportarlo a las plataformas mencionadas sin modificar nada.

Página Oficial: <https://libgdx.badlogicgames.com/>

GameMaker

Game Maker no es exactamente un motor de videojuego, pero se le acerca bastante, se trata de una herramienta o software de desarrollo de videojuegos utilizada también por muchos desarrolladores. La ventaja es que se pueden desarrollar juegos sin la necesidad de conocer un lenguaje de programación, pues el Game Maker utiliza su propio lenguaje basado en *scripts*. Ofrece una interfaz intuitiva y sencilla y se pueden portar los juegos a PSVita, PS3, PS4 y Xbox One.

La desventaja es que la versión gratuita nos proporciona muy pocos recursos y si queremos hacer un videojuego más complejo será necesario actualizar a la de pago.

Página oficial: <http://www.yoyogames.com/studio>

Sprite Kit

Apple introdujo su propio motor de videojuegos 2D con iOS7. Este motor utiliza el lenguaje de programación *Swift* y *Objective C*. La ventaja de este motor es que es muy

sencillo de usar y crear rápidamente juegos 2D, sin embargo, solo permite juegos para iOS.

Guía de programación:

https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html

4.2 Herramienta usada para el proyecto

Para este proyecto se ha optado por usar el *framework* LibGDX y las razones son las siguientes:

- Es una herramienta que ofrece un buen y reconocido tanto rendimiento como estabilidad.
- Es una herramienta consolidada que permite realizar aplicación ligeras, ya que en los dispositivos móviles el tamaño de las aplicaciones es importante porque la memoria de estos es más limitada que un ordenador o consola.
- Es multiplataforma, permitiéndonos portar nuestra aplicación tanto a dispositivos móviles con Android e iOS y BlackBerry, como ordenador (Windows, Linux y MAC) y web (HTML5).
- Existe una gran cantidad de documentación, recursos y ayuda en la red actualizada continuamente.
- Posee un gran soporte de la librería por la comunidad, lo que permite una continua evolución de esta.
- Es una herramienta Open Source, licenciada por Apache 2.0.
- Por último, la experiencia previa y satisfactoria en trabajos anteriores.

5. Diseño y Especificación

En este apartado del documento se describirá de forma detallada todos los aspectos del videojuego que se va a realizar. Está orientado a un estilo *Game Design Document*, o GDD, ya que proporciona un buen análisis y explica de forma detallada todos los aspectos y elementos del videojuego.

5.1 Argumento

Para el argumento del juego, se ha pensado en una pequeña historia de trasfondo para enganchar más al jugador. La historia gira en torno a LHED, un misterioso androide que despierta dentro de una máquina de pruebas en un extraño laboratorio. Incapaz de recordar qué le ha pasado y cómo ha llegado allí, encuentra un fragmento de una gema con vida propia que le ayuda a salir de allí dándole el poder suficiente. Su objetivo ahora es conseguir escapar del lugar, recuperar sus recuerdos y conocer el origen de esa extraña gema, pero no le será tan sencillo, ya que se encontrará con otros androides con fragmentos de gemas similares a la suya que tratarán de detenerle.

5.2 Conjunto de características

- Un solo jugador
- Aspecto gráfico: Pixel Art (juego)
- Diversidad de zonas con varios niveles cada una
- Personaje con varios modos (ataque y defensa)
- Mini-jefes en determinados niveles
- Jefe final en cada zona
- Juego multiplataforma, Android e iOS
- Disponible en los idiomas Inglés y Español

5.3 Género

El juego se trata de un *runner* que mezcla elementos de un juego de acción. El género *runner* es un género donde el jugador debe avanzar constantemente en una misma

dirección, siendo la derecha la más común, y su objetivo es evitar obstáculos y avanzar lo máximo posible antes de morir.

Este tipo de juegos son normalmente de dos tipos, infinitos y finitos. Los *runners* infinitos consisten en un solo nivel que va cambiando de forma aleatoria en la mayoría de los casos y que cada partida es distinta a la anterior. Aquí el jugador tiene que aguantar el mayor tiempo posible y recoger el máximo número de objetos posibles antes de perder. Un ejemplo de este *runner* es el Wind Runner, que se muestra la imagen a continuación:



Figura 8: ejemplo de runner, Wind Runner

El *runner* finito suele ir por niveles los cuales tiene un final. Consiste en ir completando los niveles hasta finalizar el juego. Un ejemplo es el Bit Trip Runner.



Figura 9: ejemplo de runner por niveles, Bit Trip Runner

Lo que tienen en común ambos tipos es que el personaje suele morir en la mayoría de un solo golpe, es decir, solo posee un toque de vida. Para este juego se ha querido cambiar

eso y se le ha añadido una vida al personaje, por lo que lo que aguantará más de un golpe antes de morir. Por lo demás, se trataría del tipo *runner* finito.

Generalmente las acciones principales son saltar pulsando en la pantalla y esquivar los enemigos u obstáculos, sin embargo hay muchos que incluyen algún tipo de ataque, tanto cuerpo a cuerpo como con armas a distancia. Para el caso de este juego se explicará la mecánica que incluirá de ataque en las secciones [5.12 Mecánicas](#) y la sección [5.15 Acciones y poderes](#).

5.4 Audiencia

La audiencia que se pretende alcanzar con este juego es la mayor posible, por lo que aunque contenga elementos de acción y ataques se considera que es apto para todos los públicos.

5.5 Aspecto Gráfico

Para este proyecto se ha optado por diseñar y realizar personalmente todo el apartado visual, realizando desde cero tanto *concepts* del juego como todos los gráficos, incluyendo *sprites*, dibujos y animaciones.

Lo primero es decidir el tipo de gráficos que se usarían, si modelos 3D o gráficos 2D. Se ha elegido gráficos 2D porque se ha considerado que resultaría más atractivo y se conseguirá un resultado más pulido. Después de barajar la opción de hacer los gráficos con dibujos o no, finalmente se ha optado por un estilo pixel art, ya que se ha querido dar un toque más retro al juego.

Por lo tanto, el juego será con gráficos 2D en pixel art, salvo algunas imágenes que se realizarán mediante ilustraciones, además de los *concepts*.

5.6 Personaje

El personaje principal fue de las primeras cosas que se diseñó al empezar el proyecto, realizando un primer concept del estilo que se deseaba y que encajase bien con la ambientación del juego. Al ser el estilo pixel art, el personaje no debía tener demasiados detalles, ya que al pasarlo a *sprite* iba a ser complicado de animar y de realizar las diferentes acciones y posiciones.



Figura 10: Concept del personaje principal, LHED

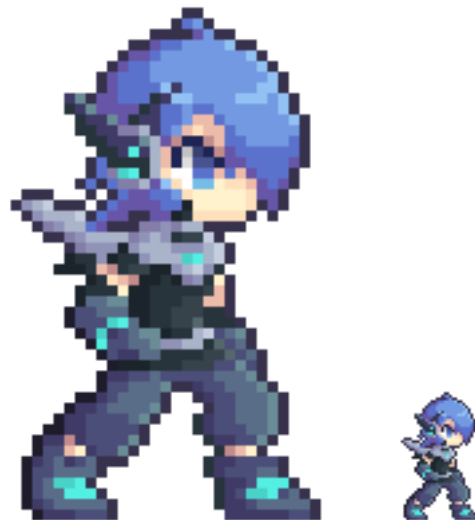


Figura 11: Sprite del personaje principal, LHED ampliada (izquierda) original (derecha)

5.7 Zonas

El juego se desarrollará en un mundo futurista. Habrá distintas zonas con varios niveles en cada una. Para pasar a la siguiente zona, el jugador deberá haber completado todos los niveles de la zona anterior. La primera zona estará situada y ambientada en un laboratorio.

5.8 Niveles

Habrán dos tipos de niveles diferentes, los niveles de recogida y los niveles de jefe. En los niveles de recogida el jugador tendrá que ir recolectando unos determinados objetos, que serán engranajes y baterías, además de ir esquivando los obstáculos que saldrán a su paso. El jugador completará el nivel si ha conseguido recoger al menos la mitad de los engranajes que hay en ese nivel. Si consigue menos de la mitad no podrá pasar al siguiente y tendrá que volver a intentarlo. Si además consigue todos los engranajes del nivel y las baterías que haya (siempre 3) completará el nivel al 100% y desbloqueará un logro.

En los niveles de jefe no habrá engranajes ni baterías que coger. En estos niveles aparecerá un enemigo que se situará en la parte izquierda de la pantalla de forma fija e irá persiguiendo y atacando al personaje. Durante la partida irán apareciendo unas estrellas de energía que el jugador deberá coger para realizar sus ataques y defenderse de los del enemigo. El jugador completará el nivel cuando ataque lo suficiente al jefe para dejar su vida a cero.

Cada zona contendrá un número total de 5 niveles, sin incluir el tutorial en la primera zona del laboratorio. Los niveles que tendrá la primera zona serán los siguientes:

Primera zona: Laboratorio

- Tutorial
- Primer nivel: nivel de recogida
- Segundo nivel: nivel de recogida
- Tercer nivel: nivel de jefe
- Cuarto nivel: nivel de recogida
- Quinto nivel: nivel de jefe final

5.9 Jugabilidad

El usuario progresará en el juego superando los distintos niveles de cada zona y completándolas para poder acceder a nuevas y así avanzar en la historia. En cada nivel, el usuario tendrá que: o bien recoger el mayor número de objetos que aparezcan por pantalla, o enfrentarse y vencer a los distintos enemigos que le vayan asaltando, recibiendo el menor daño posible.

5.10 Objetivos del Juego

Se pretende que los objetivos del juego sean sencillos con el propósito de que el usuario coja la dinámica del juego lo antes posible. Al tener la mira en usuarios de dispositivos móviles, donde estos pueden utilizar en cualquier lugar y durante periodos de tiempo no demasiado largos, se desea que la complejidad del juego no sea un impedimento para su utilización.

El usuario tendrá que completar todas las zonas y niveles del juego para poder acceder al final del juego y de la historia. También, el juego contiene una serie de retos o logros que incitarán al usuario a completarlos.

Pero el objetivo principal de este proyecto es que el usuario que juegue disfrute de esa experiencia de juego y que se mantenga enganchado hasta que lo complete.

5.11 Flujo del Juego

A continuación se describirá de forma detallada el flujo que sigue el juego desde que se ejecuta en el dispositivo móvil hasta el final de una partida del mismo:

Al iniciar la aplicación aparecerá la pantalla de carga, pasados unos segundos, el jugador verá una pantalla con el texto de 'pulsar en la pantalla para empezar'. Cuando pulse pasará al menú principal.

El menú principal mostrará una serie de opciones distintas:

- Botón de jugar: aparecerá el menú de selección de nivel.
- Botón de opciones: mostrará las opciones del juego.
- Botón de créditos: mostrará los créditos del juego.
- Botón de Twitter y Facebook: abrirá la correspondiente red social.
- Botón de votar: abrirá la plataforma de distribución para votar al juego.
- Botón de retos: mostrará los retos del juego.

Al pulsar en la opción de jugar, aparecerá un panel con la primera zona, el laboratorio, que contendrá los cinco niveles más el tutorial. Si es la primera vez que el usuario juega, solo estará disponible el tutorial, donde se le explicarán los controles y los objetivos del juego. Una vez completado, se desbloqueará el siguiente nivel, y así sucesivamente hasta completar los niveles de la zona.

Para poder pasarse un nivel de recogida, el jugador deberá controlar al personaje y recoger un mínimo de engranajes que irán apareciendo en pantalla, además de esquivar los obstáculos que vayan apareciendo. Cada nivel tendrá una cantidad mínima determinada de engranajes para poder pasarse dicho nivel. Si el jugador no consigue recolectar el mínimo de engranajes, no podrá acceder al siguiente nivel y tendrá que reintentarlo. Si consigue todos los engranajes de ese nivel, y además las 3 baterías, conseguirá el 100% de dicho nivel.

En los niveles con jefe o mini jefes, el jugador deberá esquivar los ataques que realicen estos enemigos y recoger las estrellas de energía para poder derrotarlos. Si dañan al personaje un número determinado de veces perderá la partida y tendrá que reintentarlo. Si consigue derrotar a los enemigos podrá acceder al siguiente nivel.

Una vez el jugador acabe un nivel, aparecerá una pantalla de resultados que indicará si se ha superado el nivel, si no, o si se ha superado al 100%. El jugador podrá elegir en esta pantalla entre cuatro opciones:

- Volver a jugar el mismo nivel
- Volver a la pantalla de zonas y niveles
- Salir al menú principal

5.12 Mecánicas

Personaje:

El personaje podrá realizar más de un salto, pudiendo así impulsarse en el aire. La máxima altura que podrá alcanzar será el techo. El jugador podrá controlar al personaje con toques en la pantalla, o bien manteniendo el dedo presionado en la pantalla para elevarse en el aire de forma constante.

El personaje siempre correrá a la misma velocidad. No acelerará ni se ralentizará, salvo para empezar a correr y al morir.

Colisiones:

El personaje irá corriendo sobre una plataforma y podrá saltar e impulsarse hasta llegar al techo. Las colisiones se realizarán entre el personaje y:

- Los engranajes

- Las baterías
- Las estrellas de energía
- Obstáculos
- Enemigos
- Los ataques de los enemigos
- Suelo y techo

5.13 Movimiento

Durante la partida, el personaje estará siempre corriendo hacia la derecha y podrá saltar varias veces en el aire. Podrá saltar todas las veces que el jugador quiera, quedándose en el aire si el jugador mantiene el toque en la pantalla.

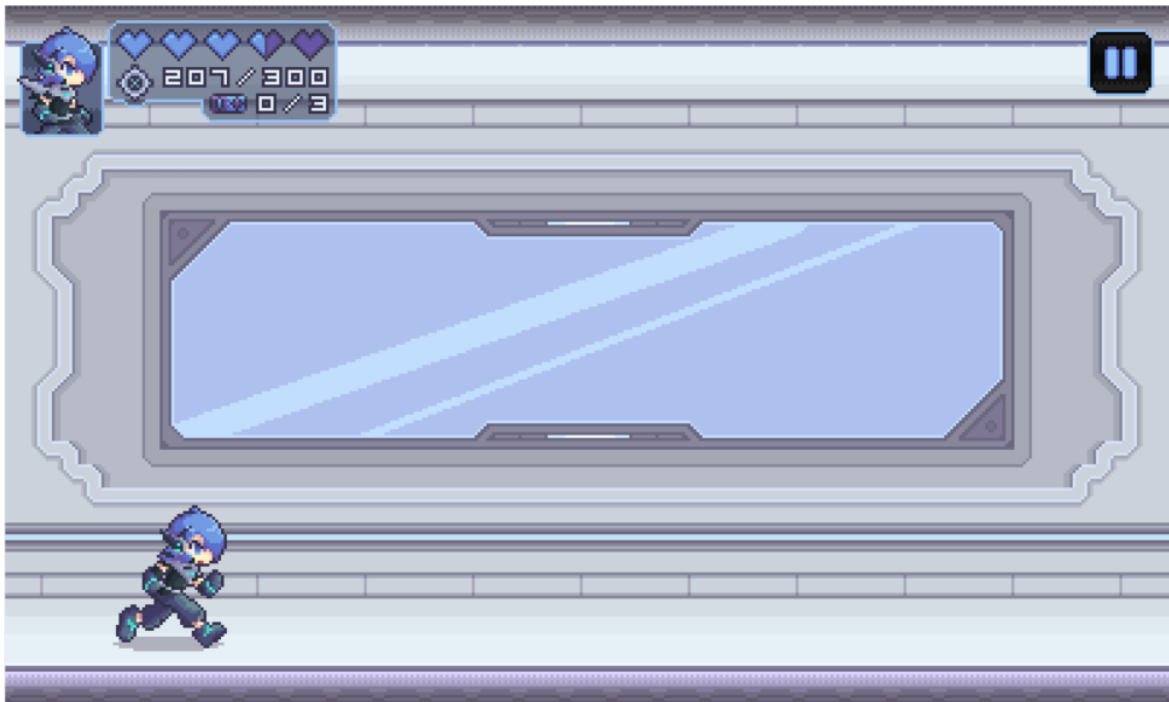


Figura 12: Movimiento del personaje principal

5.14 Objetos

En el juego, el jugador se irá encontrando a lo largo de la partida con una serie de objetos que irán apareciendo por la pantalla. Estos objetos serán necesarios para completar el nivel, como por ejemplos los engranajes, o serán objetos que ayudarán al jugador de forma pasiva, como las esferas de salud. Los objetos son los siguientes:

- Engranajes: en todos los niveles de recogida aparecerá una gran cantidad de engranajes repartidos por la pantalla que el jugador deberá ir recogiendo para completar el nivel.



Figura 13: Objeto engranaje

- Batería: una batería de energía. En cada nivel de recogida el jugador se encontrará con un máximo de 3 baterías. Estas baterías sirven para completar el 100% del nivel.



Figura 14: Objeto batería

- Estrella de energía: durante un nivel de jefe, las estrellas de energía serán la clave para que el jugador realice sus ataques y acabe así con los enemigos.



Figura 15: Objeto estrella de energía

- Esfera de salud: este objeto es muy poco común y podrá aparecer tanto en los niveles de recogida como en los de jefe. Se trata de un pequeño corazón que si el jugador lo coge recuperará un punto completo de salud.



Figura 16: Objeto esfera de salud

5.15 Acciones y poderes

En los niveles de jefe, el personaje podrá adoptar dos modos de lucha: Ataque y Defensa. Utilizará estos modos para derrotar a los enemigos que se vaya encontrando.

Cuando el jugador consiga 3 estrellas de energía, el personaje pasará a modo ataque. En ese momento, el juego se detendrá y aparecerán una serie de combinaciones de botones por la pantalla que el jugador deberá de acertar en un tiempo límite para poder realizar el ataque.

Para el modo ataque:

- Si acierta más de un 80% de las combinaciones, el personaje atacará al enemigo y le quitará un toque.
- Si acierta más de un 50% de las combinaciones, el personaje atacará al enemigo pero solo le quitará medio toque.
- Si acierta menos del 50%, el personaje no atacará, perdiendo así la oportunidad de dañar al enemigo.

Visualmente, el personaje atacará lanzando un potente láser hacia el enemigo.

Si el jugador no consigue coger una estrella de energía, el enemigo la obtendrá en su lugar. Si el enemigo consigue tener 3 estrellas de energía, atacará al jugador con un potente disparo. En este caso, el personaje pasará a un modo defensivo y al igual que con el modo de ataque, el juego se detendrá por unos segundos para que aparezca la serie de combinaciones.

Para el modo defensa:

- Si acierta el 100% de las combinaciones, el personaje se defenderá y no recibirá daño.
- Si acierta más de un 50% de las combinaciones, el personaje se defenderá pero recibirá parte del golpe, dañándolo medio toque.
- Si se acierta menos del 50% de las combinaciones, el personaje no se defenderá y el ataque le quitaría un toque de vida.

Visualmente, el personaje se defenderá formando un escudo de energía a su alrededor que absorberá el ataque del enemigo.

En ambos modos, el personaje se situará centrado verticalmente en la pantalla y el jugador no podrá manejarlo hasta que termine el ataque o la defensa.

5.16 Obstáculos

En los niveles de recogida aparecerán obstáculos por la pantalla que el jugador deberá de esquivar para no colisionar con ellos y recibir daño. En el primer nivel aparecerá solo un tipo de obstáculo, y se irán añadiendo más variedades en los siguientes niveles. Los obstáculos serán los siguientes:

- Bomba: este obstáculo aparecerá flotando en el aire, al colisionar el jugador con él producirá una fuerte explosión que quitará un corazón de vida al personaje.



Figura 17: Obstáculo bomba

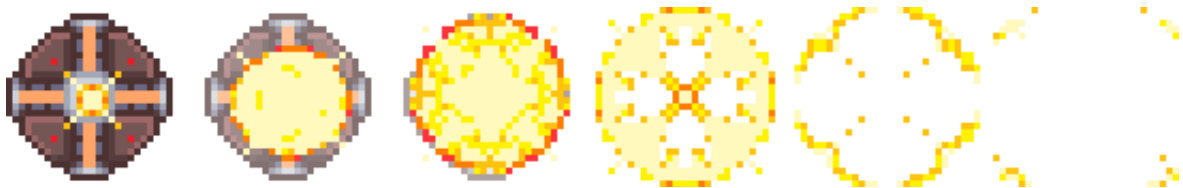


Figura 18: Explosión del obstáculo bomba

- Pinchos: estarán situados en el suelo y el techo, son pinchos afilados que quitarán al jugador medio corazón si los roza.

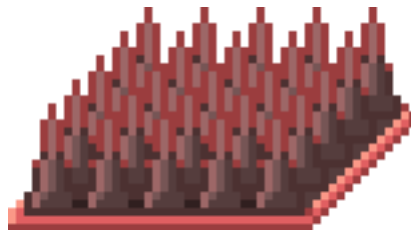


Figura 19: Explosión del obstáculo bomba

En los niveles de jefe, los obstáculos serán los ataques del mismo enemigo.

Cuando el jugador colisiona con estos obstáculos se activará un *cooldown*, es decir, un tiempo de respuesta de duración muy escasa que hará que el jugador se vuelva invulnerable durante ese tiempo. El *sprite* del personaje parpadeará levemente para indicar al jugador que ha sufrido daño y pueda reaccionar con tiempo.

5.17 Enemigos

En los niveles de jefe, el jugador se enfrentará a poderosos enemigos que intentarán frenarle el paso. El enemigo que el jugador se encontrará en el primer nivel de jefe será una enorme nave con dos cañones que irán disparando activamente. El comportamiento de estos cañones se explicará en el apartado 6.6 Inteligencia Artificial.

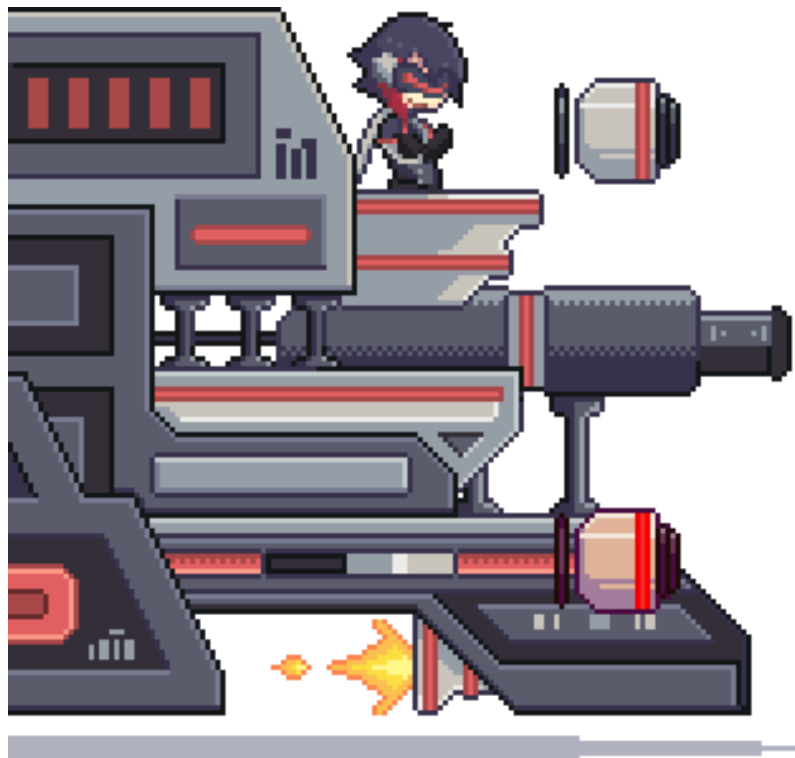


Figura 20: Enemigo del nivel de Jefe

Los enemigos de estos niveles se situarán en la parte izquierda de la pantalla.

5.18 Retos y logros

Se introducirá en el juego un sistema de logros proporcionado por la API de Google ([Google Play Game Services](https://developers.google.com/games/services/)) para motivar al jugador a seguir jugando y a completar desafíos.

Los logros y desafíos que incluirá el juego son los siguientes:

- Jugar por primera vez
- Completar el tutorial
- Completar el primer nivel
- Completar el segundo nivel sin recibir daño
- Completar un nivel al 100%
- Completar la primera zona
- Completar la primera zona al 100%
- Derrotar al jefe sin recibir daño

5.19 Pantallas

5.19.1 Flujo general de las pantallas

A continuación en este apartado se mostrará un diagrama con el flujo de las pantallas que contiene el juego:

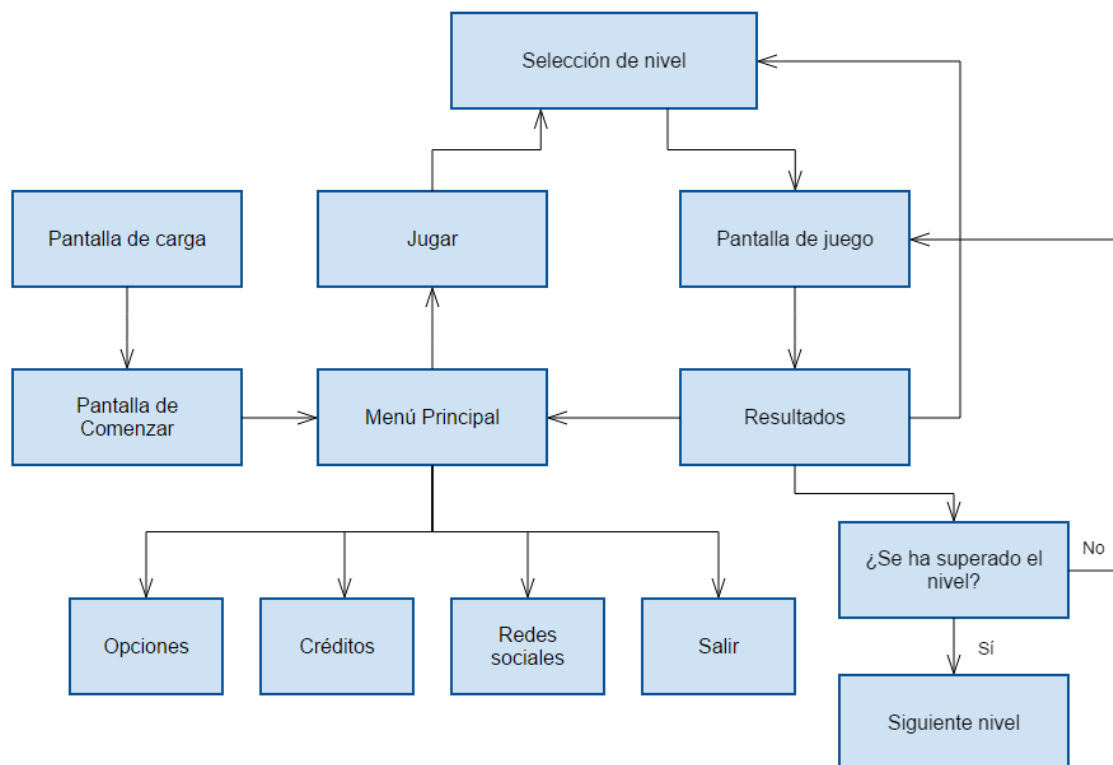


Figura 21: Diagrama de flujo general de pantallas

5.19.2 Pantallas del Juego

PANTALLA DE CARGA:

Esta será la primera pantalla que saldrá al iniciar la aplicación. Consistirá en una pantalla con el fondo en negro y el logo del autor en el centro. Justo abajo un pequeño texto con el contenido de "Cargando..."

PANTALLA DE COMENZAR:

Es la pantalla que saldrá justo después de la pantalla de carga. En esta se mostrará el logo del juego y el texto en la parte inferior, tal y como se muestra a continuación.



Figura 22: Concept pantalla de comenzar

MENÚ PRINCIPAL:

Al tocar la pantalla, accederemos al menú principal donde podremos ver el logo del juego en la parte izquierda de la pantalla y una serie de botones en la parte derecha, tal y como se muestra en la siguiente imagen.



Figura 23: Concept menú principal

Botones del menú principal:

- Jugar: al pulsar en el botón de Jugar aparecerá el menú de selección de zonas y niveles.
- Votar: al pulsar este botón la aplicación llevará al usuario al juego en Google Play para que pueda votarlo y comentarlo.
- Twitter: al pulsar este botón la aplicación abrirá el explorador de internet y llevará al usuario a la red social Twitter del juego o autor.
- Facebook: al pulsar este botón la aplicación abrirá el explorador de internet y llevará al usuario a la red social Facebook del juego o autor.
- Retos: al pulsar el botón de Retos aparecerá un panel con la información de los retos que el jugador ha desbloqueado y los que no.
- Opciones: al pulsar este botón aparecerá la pantalla de opciones que se describirá más adelante.
- Créditos: al pulsar el botón de créditos, aparecerá un panel donde se mostrarán los créditos del juego.

- Salir: al pulsar el botón salir aparecerá un mensaje emergente preguntando al usuario si realmente desea salir. Si se pulsa que sí la aplicación se cerrará, y si se pulsa que no el mensaje desaparecerá.

MENÚ DE SELECCIÓN DE ZONAS Y NIVELES

En esta pantalla aparecerán los distintos niveles bloqueados y disponibles de cada zona con la información de cada uno, es decir, el número de engranajes conseguidos y baterías en el caso de un nivel de recogida y si la fase se ha completado o no en el caso de un nivel de jefe.

Si el jugador pulsa en un nivel lo llevará a la pantalla de juego correspondiente.

MENÚ OPCIONES:

El menú de opciones será como se muestra en la siguiente imagen.

The image shows a conceptual layout for an 'OPCIONES' (Options) menu. At the top left, it says 'Juego TFG'. The title 'OPCIONES' is centered at the top. Below it, there are three settings: 'Música:' with checkboxes for 'Sí' (checked) and 'No'; 'Efectos de Sonido:' with checkboxes for 'Sí' (checked) and 'No'; and 'Idioma:' with checkboxes for 'Español' (checked) and 'English'. A 'Volver' button is located in the bottom right corner.

Figura 24: Concept menú de opciones

Es posible que se añadan nuevas opciones según se considere oportuno durante el desarrollo del juego.

PANTALLA DE RESULTADOS:

La pantalla de resultados aparecerá cada vez que se finalice un nivel. En esta pantalla se mostrará si se ha conseguido superar el nivel (si el jugador ha conseguido el mínimo requerido de engranajes, si no lo ha conseguido o si los ha conseguido todos) las baterías que ha conseguido y tres botones: uno para reintentar el nivel, otro para volver al mapa y uno para pasar al siguiente nivel de forma directa (que aparecerá bloqueado si no se ha superado el nivel).

6. Desarrollo e Implementación

En este apartado se explicará cómo se ha implementado el juego y los métodos y estructuras que se han usado en dicha implementación.

6.1 Estructura General

Lo primero a tener en cuenta a la hora de implementar un videojuego es cuáles son los tres bloques principales que lo componen y en qué consisten. Estos bloques son: *Input*, *Update* y *Render*.

- Input: Se trata de la entrada o lector de eventos. Es el encargado de recoger las acciones que ejecutará el jugador al pulsar una tecla o botón.
- Update: Es el bloque donde se realiza toda la lógica del juego, como por ejemplo el movimiento del personaje, la inteligencia artificial de los enemigos o los eventos que irán sucediendo a lo largo de la partida.
- Render: Se encarga de la visualización de todos los elementos del juego en la pantalla.

Además el orden en el que éstos se ejecutan es también importante para asegurar el buen funcionamiento del juego.

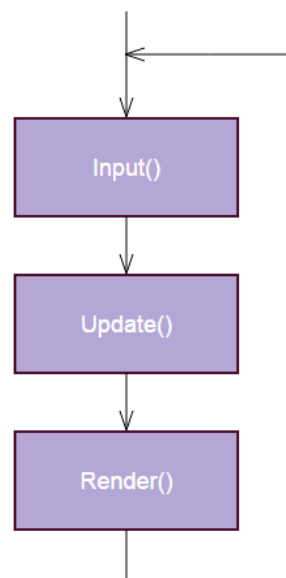


Figura 25: Bloques de un videojuego

Para este proyecto, se ha optado por crear una clase con cada bloque, es decir, una clase *Input* donde comprobaremos todos los eventos, en este caso, recibidos al tocar la pantalla del móvil en una zona determinada por el jugador, otra clase *Update* donde estará todo el código de la lógica, y una clase *Render*, donde se llamarán a los métodos de dibujo para mostrar los elementos del juego por pantalla. A parte de estas tres clases, tenemos la clase principal *GameScreen* que será la que se encargue de llamarlas de forma recursiva durante toda la ejecución del juego.

6.2 Patrones de Diseño

Los patrones de diseño es algo esencial para el desarrollo de un proyecto de grandes dimensiones como es el desarrollo de un videojuego. Los patrones de diseño tratan de dar soluciones simples a los problemas de diseño que surgen en la programación orientada a objetos o a lo largo del desarrollo del proyecto. Existen diversos patrones de diseño que se clasifican en:

- Estructurales
- Creacionales
- De comportamiento

Para este proyecto se ha utilizado el patrón Singleton, ya que es el más usado para el desarrollo de videojuegos y el que más se adaptaba a la estructura que se deseaba para el proyecto.

6.2.1 Patrón Singleton

El patrón Singleton consiste en restringir la creación de objetos pertenecientes a una clase, de modo que un objeto sea creado una sola vez y tenga una única instancia de la clase para toda la ejecución del proyecto. Con esto se garantiza que el objeto sea accesible desde todo el proyecto, ahorrando código y tiempo de ejecución.

6.3 Implementación de Efectos

Los efectos están presentes de una forma u otra en la mayoría de los videojuegos y es un elemento esencial ya que hace que el juego se vea mucho más vistoso y proporciona una considerable mejora visual. Los efectos pueden ser desde una transición de pantalla, como por ejemplo la navegación entre los menús, que no aparezcan de repente, si no

con un pequeño movimiento o un oscurecimiento de pantalla, hasta la vibración de la pantalla para indicar cuando un disparo te da o las partículas de chispas cuando disparas un arma. Estos efectos consiguen dar al jugador una mejor impresión del juego y hace que parezca más profesional, además de proporcionar un *feedback* importante al jugador.

A la hora de la planificación del proyecto se pensó en todos los efectos que se necesitarían. Para el cambio de pantalla

6.3.1 Fade In y Fade Out

El *Fade* consiste en un efecto de transición suave que se utiliza para la navegación entre pantallas y menús, muy común en los videojuegos. Para este proyecto, además del paso de pantallas, se ha implementado para otros momentos determinados del juego, como por ejemplo en los niveles de jefe justo antes de realizar un ataque especial, se oscurece la pantalla mediante el *fade* para dar paso a las combinaciones, o para dar ambientación a la partida mediante un pequeño *fade* de color rojo simulando una alarma.

Hay dos tipos de *fade*, el *fade In* y el *fade Out*. El primero es cuando comienza a producirse el efecto y el segundo cuando desaparece. A continuación se muestra un ejemplo de ambos *fades* en una transición de pantalla entre el menú de selección de niveles y el comienzo de la partida:



Figura 26: Fade In

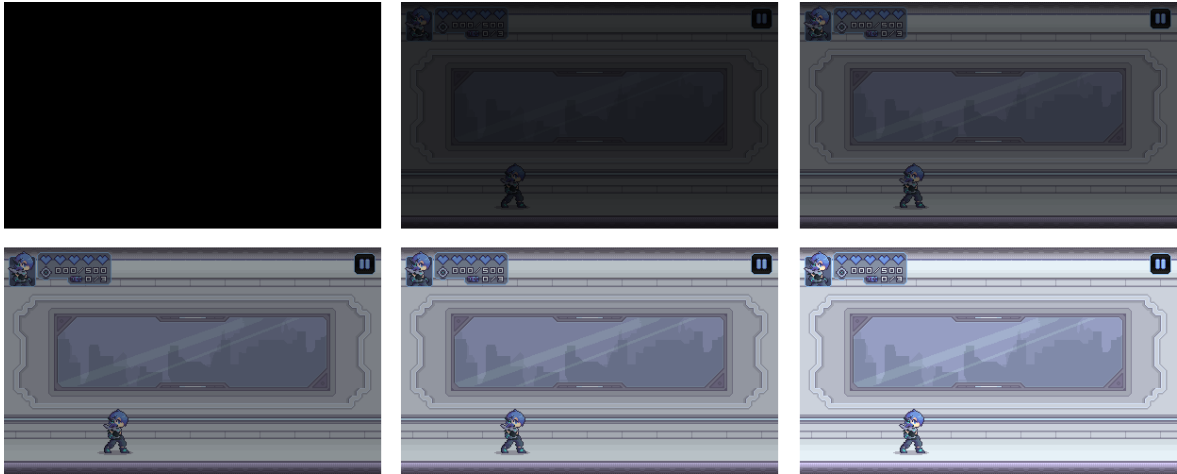


Figura 27: Fade Out

Para implementar este efecto lo primero que hemos hecho es crear un rectángulo con un color sólido que rellene toda la pantalla. El color se lo pasamos por parámetro, para así tener los diferentes *fades* que necesitamos, y también el tiempo que queremos que dure la transición.

Necesitamos dos métodos, uno para cada *fade*. Cuando activamos el método *Fade In*, el rectángulo creado comienza con opacidad 0 y en cada iteración se irá aumentando hasta llegar a la opacidad máxima, a una velocidad dependiendo del tiempo que le hemos pasado. Con el *Fade Out* será al contrario, empezará con opacidad máxima y disminuirá con cada iteración hasta quedar a 0.

Para los efectos de ambiente tenemos otro método distinto, ya que en estos no queremos que el *fade* llegue con la opacidad máxima, si no que llegue hasta una determinada opacidad que le pasaremos por parámetro y se esté ejecutando en bucle el tiempo determinado que le asignemos.



Figura 28: Fade de ambientación

6.3.2 Vibración de pantalla

La vibración de la pantalla es un movimiento de cámara que simula un temblor, es un efecto bastante utilizado en los videojuegos. Entre sus numerosos usos, los principales son para proporcionar un *feedback* al jugador, como por ejemplo cuando un proyectil impacta contra el personaje, una explosión, el movimiento de un objeto muy pesado, etc.

Para este proyecto se ha utilizado la vibración de pantalla en esos casos mencionados: cuando el jugador recibe daño, cuando se produce la explosión al detonar el obstáculo bomba, cuando se realizan los ataques especiales y cuando el enemigo del nivel del jefe aparece.

La implementación consiste en mover la cámara rápidamente en un eje, en nuestro caso el eje X (aunque puede implementarse para ambos), durante un determinado periodo de tiempo y de forma continua. Para eso se ha implementado el método *ShakeCam*, al que le pasaremos por parámetro el tiempo que queremos que dure la vibración y a la frecuencia con la que se ejecute.

6.3.3 Otros efectos

Se han implementado otros efectos fuera del gestor de la clase Efectos y mediante sprites, como son por ejemplos las partículas a la hora de disparar, al coger el personaje la esfera de salud, al hacer los ataques especiales y al recoger objetos.

Otro efecto es cuando el personaje sufre daño, que se producirá un pequeño *cooldown* mientras su sprite parpadea levemente.

6.4 Motor de Físicas

Como hemos mencionado, un motor de videojuegos está formado por distintos motores que se encargan de una función específica del juego. El motor de físicas es uno de ellos y es uno de los motores fundamentales, ya que hoy en día la mayoría de los juegos utiliza algún tipo de simulación física. Sin embargo, también es uno de los más difíciles de implementar. Debido a esta complejidad hay pocas compañías de videojuegos que implementan su propio motor de físicas y utilizan uno ya creado para incorporarlo a su motor.

El motor de físicas se encarga de realizar simulaciones de ciertos sistemas físicos, como puede ser la dinámica del cuerpo rígido, la colisión entre objetos o la propia gravedad. Pero en un videojuego no se busca la exactitud total con la física del mundo real, si no que parezca lo más realista posible con la menor complejidad posible.

Se pueden clasificar los motores de físicas en dos tipos por la capacidad de cálculo que requieren. Están los motores físicos de simulación en tiempo real y los de alta precisión. Estos últimos de alta precisión requieren demasiada capacidad de cómputo que hace que sea imposible hacer simulaciones en tiempo real.

A medida que van evolucionando tanto la tecnología como los videojuegos, la simulación física se hace más compleja, por lo que aumenta de forma considerable el coste de programar la física. Sin embargo, cada videojuego es muy distinto y necesita una física más compleja o no.

Para este proyecto se ha optado por crear el motor de físicas de forma propia usando solo las funciones de colisión entre objetos que nos proporciona LibGDX.

6.4.1 Colisiones

La detección de colisiones está presente de algún modo y otro en casi todos los videojuegos, consiste en detectar cuando dos cuerpos entran en contacto entre sí. Cuando esta colisión se produce, el código será el que controle qué va a pasar, por ejemplo si el personaje colisiona con un enemigo o un disparo o proyectil, lanzaremos una "señal" para que le baje vida al personaje por haber sufrido daño.

Las colisiones pueden ser de dos tipos, por área o por pixel a pixel. Las colisiones por área suceden cuando dos objetos que ocupan una determinada área, rectangular o circular, entran en contacto dichas áreas y se produce la colisión, mientras que las colisiones de pixel a pixel son más costosas que las de área y son cuando los objetos ocupan un área rectangular pero tienen una máscara que define qué píxeles son los que están visibles y cuáles no. Lo primero que se realiza es una detección de colisiones de área y cuando haya colisión se realiza otra comprobación pixel a pixel entre los píxeles superpuestos de ambos objetos. Si existen dos píxeles que se han superpuesto y ambos son visibles, se detecta la colisión.

Para este proyecto se han utilizado las colisiones por área, ya que al ajustar el área de colisión a los distintos elementos que colisionarán se ha obtenido un resultado

satisfactorio y con bajo coste de procesado. Para las áreas de colisión se han creado unos *bounds* para cada objeto, como se puede ver en la siguiente imagen:



Figura 29: Bounds

En este juego las colisiones que tendremos serán con los objetos que el personaje irá recogiendo, los obstáculos, los enemigos y los límites superiores e inferiores de la pantalla.

6.4.1.1 Colisiones con Objetos y Obstáculos

Para las colisiones con objetos y obstáculos, aunque son colisiones similares, se han implementado dos métodos diferentes para gestionar y tener organizados los eventos que ejecutarán:

- **ColisionarConItems:** en este método lo primero que comprobamos es si el vector de objetos se ha generado correctamente. A continuación recorremos todo este vector comprobando en cada posición si los *bounds* del personaje se solapan con los bounds de cada objeto mediante el método *OverlapRectangles*. Si existe un solapamiento entonces es que ambos objetos han colisionado y pasamos a comprobar de qué tipo es el objeto para lanzar un evento u otro. Si no hay colisión seguimos comprobando.

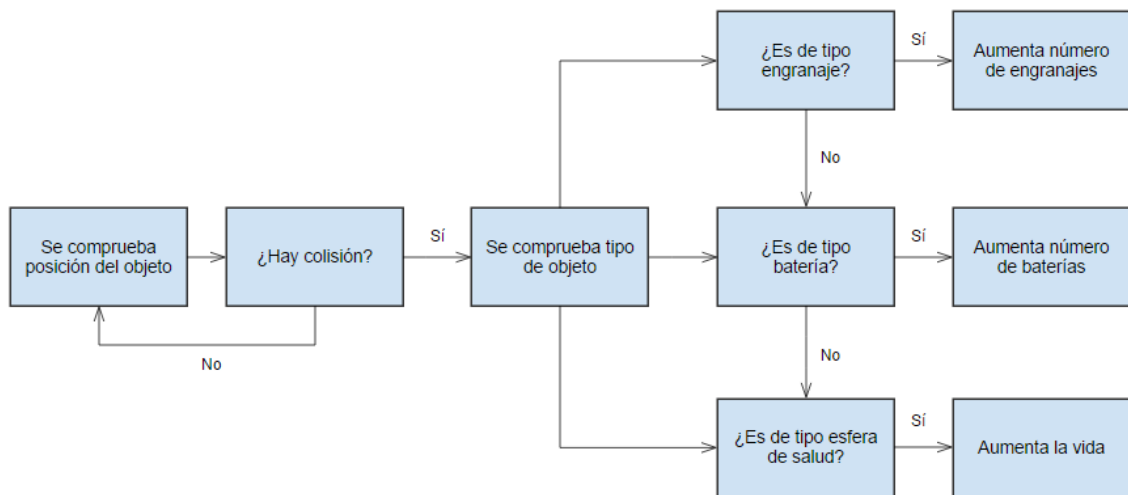


Figura 30: Diagrama funcional del método *ColisionarConItems*

- **ColisionarConObstaculos:** de igual forma que *ColisionarConItems*, recorreremos los obstáculos que hay en el vector de obstáculos y vamos comprobando si se produce colisión entre los *bounds* del personaje y del obstáculos. Si se produce colisión preguntamos qué tipo de obstáculo es y si no seguimos comprobando posiciones.

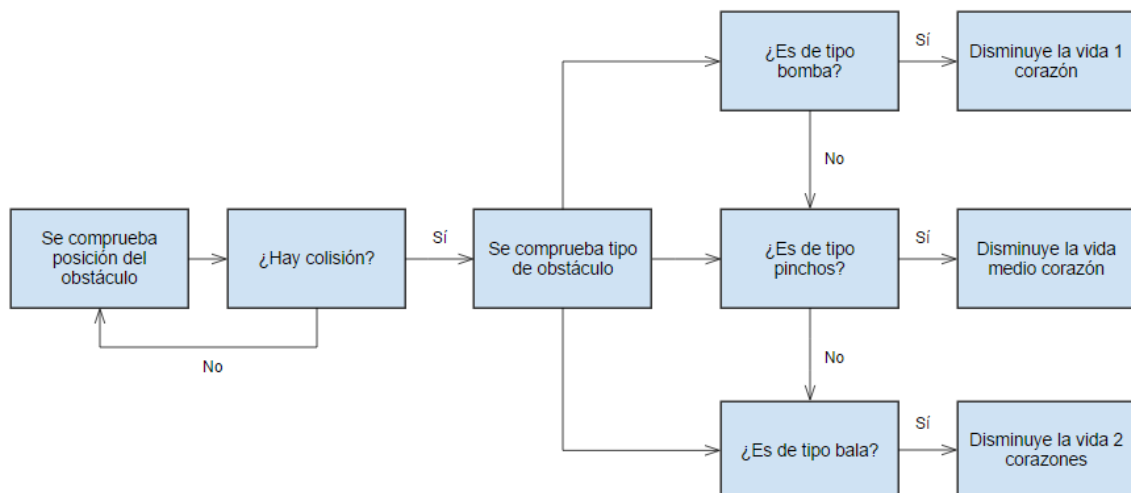


Figura 31: Diagrama funcional del método *ColisionarConObstaculos*

Una vez tanto los objetos como los obstáculos hayan colisionado con el jugador, se reproducirá su animación correspondiente y se eliminarán del vector, reordenándose de forma automática.

6.4.1.2 Colisiones con límites de pantalla

El personaje irá corriendo por el suelo siempre a la misma altura, y cuando el jugador salte o vuele llegará hasta el techo y no podrá seguir subiendo. Para comprobar estas colisiones y que el personaje no pueda salirse ni por arriba ni por abajo, se han establecido unos márgenes en la pantalla de una altura determinada. Se ha implementado un método *ComprobarPosición* que a lo largo de la partida va comprobando si la posición del personaje es mayor que el margen inferior y menor que el margen superior. Si el personaje al caer, su posición es menor que el margen inferior del suelo, se recolocará en la siguiente iteración, de igual forma que si se pasa del margen del techo.

6.5 Generadores de Objetos y Obstáculos

Para generar los obstáculos y los objetos se ha implementado una clase específica para ello. Todos los obstáculos se almacenarán en un vector de obstáculos y los objetos en un vector de objetos. En estos vectores se le pasará al elemento la posición en X e Y, y el tipo del obstáculo u objeto.

Un ejemplo de cómo se añadirían objetos u obstáculos a los vectores sería el que se muestra a continuación:

```
items.add(new Item(150, 20, 0); // engranaje
items.add(new Item(210, 60, 1); // batería
items.add(new Item(340, 100, 2); // esfera de salud
...
obstacles.add(new Obstacle(60, 40, 0); // bomba
obstacles.add(new Obstacle(60, 40, 0); // pinchos
...
```

6.6 Inteligencia Artificial

El enemigo que el jugador se encontrará a lo largo de la partida será en el nivel del jefe. Se trata de un enemigo que va desplazándose en una enorme nave acompañado de dos cañones flotantes y que actúan por separado. Estos cañones van disparando al personaje de forma activa, recargando su munición y cambiando sus posiciones.

Al aparecer el enemigo, los dos cañones ocuparán unos puestos iniciales determinados. Para el movimiento de los cañones se ha dividido la pantalla de juego en 8 partes, dejando como margen una arriba y otra abajo, para que los cañones no sobrepasen el techo ni se metan en el suelo. Los cañones decidirán de forma probabilística la posición a la que se moverán después de haber disparado, siendo más probable la que esté más cerca del jugador.

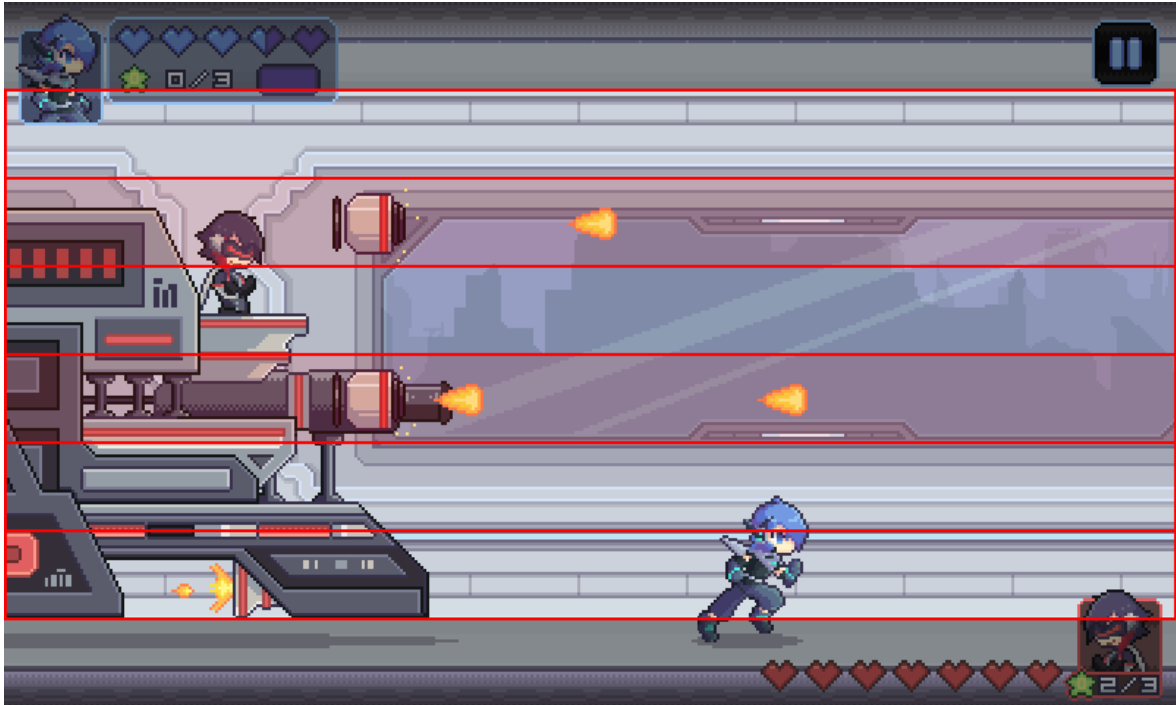


Figura 32: IA de los cañones - movimiento

Cada cañón tendrá unos estados determinados. Al comenzar la partida aparecerán quietos unos instantes y después comenzarán a atacar a la vez, lanzando un número variable de disparos cada uno, tras esto pasarán a un estado de descanso y a continuación elegirán nueva posición para moverse, comprobando si es válida o no. Si es válida efectuarán el movimiento, si no la volverán a calcular.

Dependiendo del número de munición que hayan gastado en el anterior ataque, necesitarán recargar o no. Cada cañón tiene como munición un número determinado de balas, que en cada partida variará. Si han gastado todas las balas necesitarán un tiempo para recargar, por lo que aparecerán parados con una pequeña animación para indicar la recarga. Si aún les quedan balas, atacarán al personaje hasta que se les gaste la munición.

Los tiempos de espera y de recarga variarán en cada ciclo, con esto conseguimos que cada cañón parezca que actúa distinto al otro y no efectúen movimientos a la vez.

A continuación se muestra un diagrama con el comportamiento de los cañones:

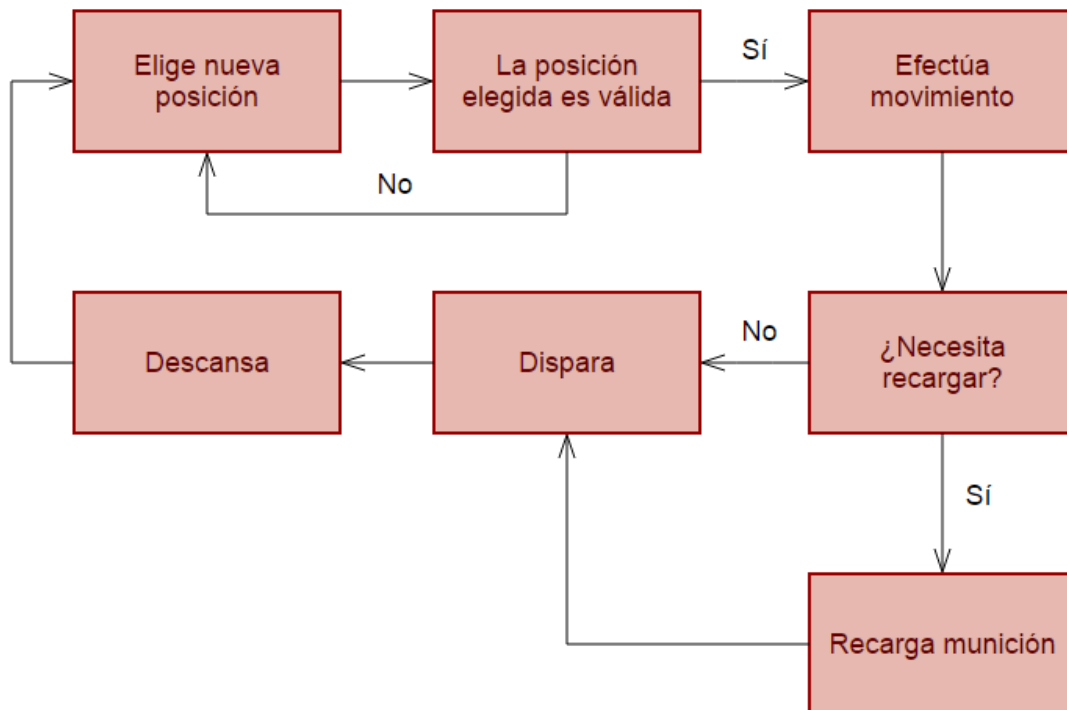


Figura 33: Comportamiento cañones del enemigo

Cuando el personaje muere, ambos cañones dejan de disparar y se mantienen quietos en su posición, sin cambiar de estado.

6.8 Gestión de Inputs

Al tratarse de un dispositivo móvil el medio por el que el usuario jugará a este proyecto, los *inputs* y botones los tiene que realizar todos a través de la pantalla táctil. Para ello debemos asignar un área para cada zona donde queremos que se pueda pulsar.

Para determinar un área tenemos que crear un rectángulo que nos sirva de *bounds*, de forma similar a las colisiones, pues queremos que se produzca una colisión entre el toque con el dedo del jugador y el área. En el caso de los botones, los *bounds* quedarán como se puede apreciar en la imagen a continuación:

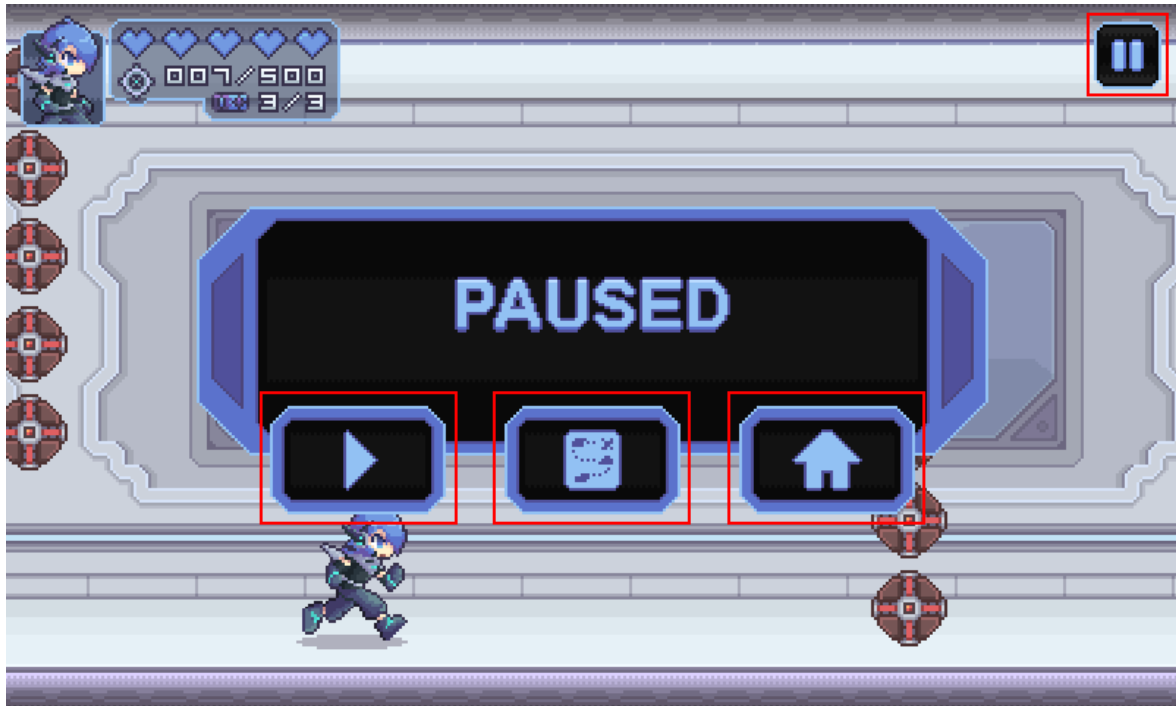


Figura 34: Área de inputs

Creamos los *bounds* algo más grandes que los botones para que se más sencillo pulsar, ya que los usuarios con dispositivos móviles más pequeños pueden tener dificultad si el botón es demasiado reducido. En nuestro caso, aunque se han pretendido que los botones sean lo suficientemente grandes para que no haya estos problemas, es preferirle aún así ampliar levemente los *bounds*. Por lo tanto, tenemos nuestra área para que cuando el jugador pulse se produzca un evento.

Lo que nos queda es determinar el punto donde se produce el toque del jugador sobre la pantalla. Para esto LibGDX nos proporciona un método *JustTouched*, que nos devolverá si se ha producido el toque, y si es así se guardarán las coordenadas X e Y de dicho toque en una variable llamada *touchPoint*, por lo que podremos comprobar si está dentro del rectángulo que hemos creado a través del método *pointInRectangle*. Este método nos devuelve verdadero o falso dependiendo si está dentro o no.

Con esto podemos crear todos los botones que queramos y asignarles un evento o acción cuando el toque coincida dentro de los *bounds*, como se ha realizado en el menú principal y en los paneles de Pausa, Game Over y Nivel Completo. Sin embargo, durante la partida no tenemos ningún botón a la hora de jugar, es decir, el jugador controla al personaje pulsando en cualquier área de la pantalla. Para esto hemos creado unos *bounds* que ocupen gran parte de la pantalla, dejando el espacio en la parte superior para el área del botón de pausa, tal y como se ve a continuación.

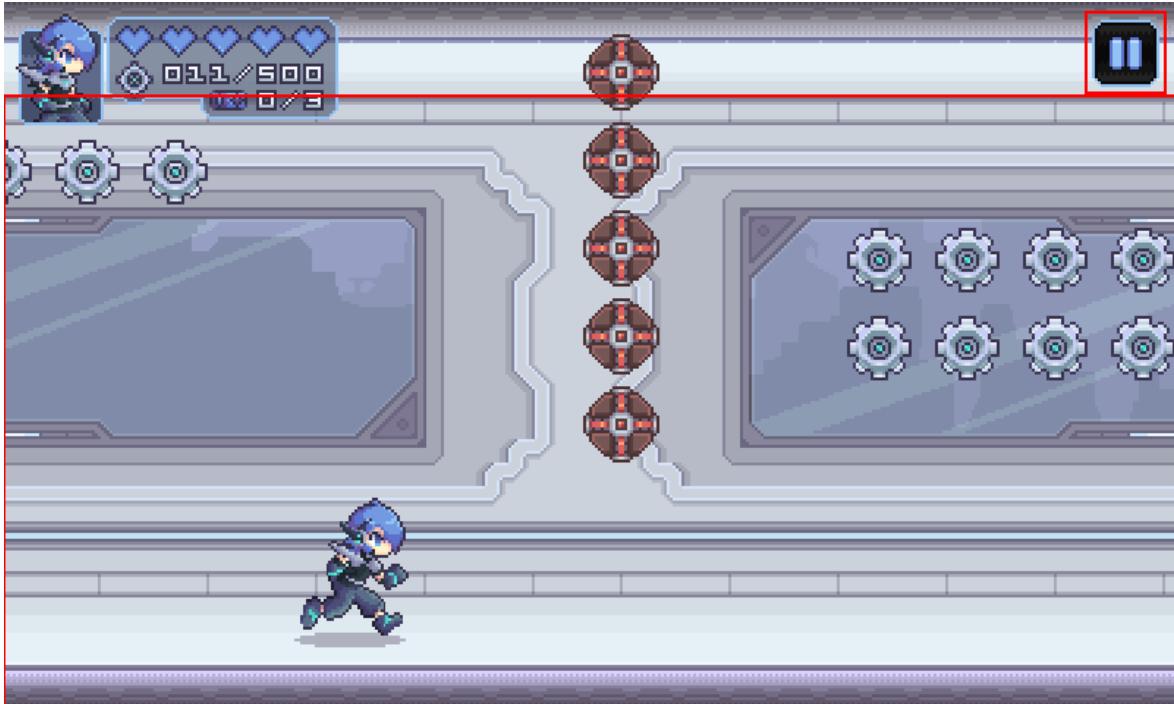


Figura 35: Inputs durante la partida

6.9 Gráficos

El aspecto gráfico es una de las partes más importantes a la hora de desarrollar un juego. Para que un juego atraiga a el mayor público posible, es importante que las todas las partes que lo componen tengan armonía entre sí y no falle ninguna de ellas. Por ejemplo, si un juego contiene unos gráficos increíbles y bonitos, pero luego la jugabilidad que ofrece al jugador da mucho que desear o es muy incómoda, el juego tiene más posibilidades de que sea rechazado por el jugador, al igual que pasaría del modo contrario. Para este proyecto se ha intentado que todas las partes estén igual de equilibradas y todos los gráficos han sido elaborados personalmente.

6.9.1 Assets

El término *assets* es muy utilizado en el mundo del desarrollo de videojuegos, hace referencia a los elementos que serán introducidos en el juego. Estos elementos incluyen, por ejemplo, modelos 3D, texturas, materiales, personajes y animaciones en la parte gráfica, como sonidos y melodías para la parte de audio.

Cada motor de juego trabaja de una manera distinta con los *assets*, de modo que hay *assets* que unos motores pueden utilizar y otros no. Pero los ejemplos mencionados antes son elementos que todos los motores de hoy en día usan.

Para nuestro proyecto, los *assets* que necesitaremos, ya que se trata de un juego en 2D, serán básicamente *sprites*, texturas, efectos de sonido y música para la parte de audio. En LibGDX, los *assets* se guardan en el proyecto de Android, del cual se accede desde el proyecto principal.

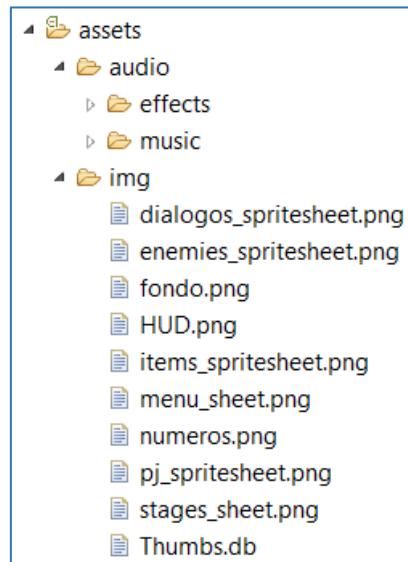


Figura 36: Assets del proyecto

6.9.2 Sprites

Los *sprites* son algo imprescindible en los videojuegos, sobre todo aquellos con gráficos en 2D. Un *sprite* es una imagen bidimensional que es integrada en una escena en el juego y mostrada por pantalla.

Para este proyecto, hemos organizado los *sprites* en distintos archivos, llamados *sprite sheet*, es decir, una hoja de *sprites*, de modo que cada elemento esté organizado en su respectivo archivo, por ejemplo, todos los *sprites* y animaciones del personaje en un mismo *sprite sheet*, etc.

Los archivos deben ser obligatoriamente potencia de 2 para que LibGDX pueda cargarlos, a continuación se mostrarán algunos de los *sprites* realizados para este proyecto:

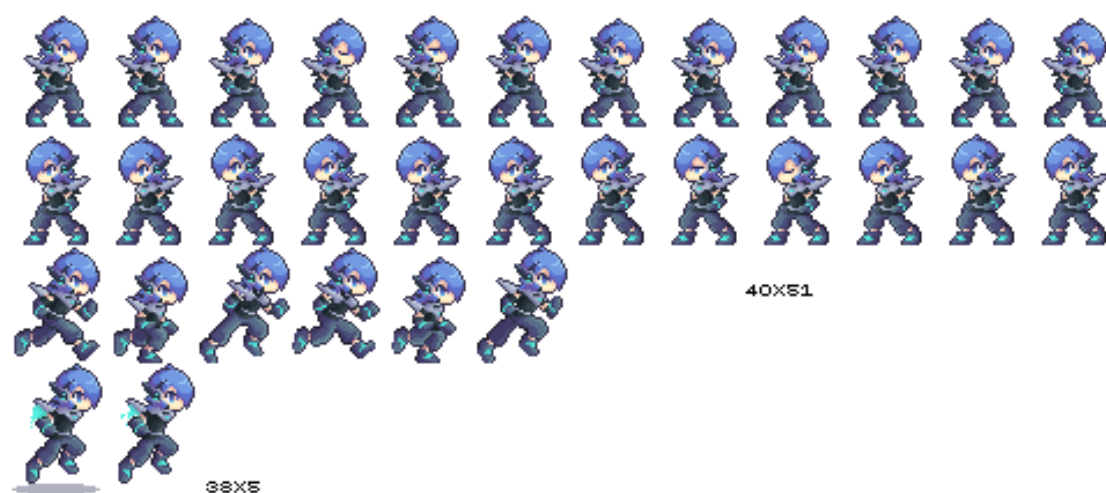


Figura 37: Spritesheet del Personaje Principal



Figura 38: Spritesheet de los obstáculos



Figura 39: Spritesheet de los objetos



Figura 40: Spritesheet del HUD

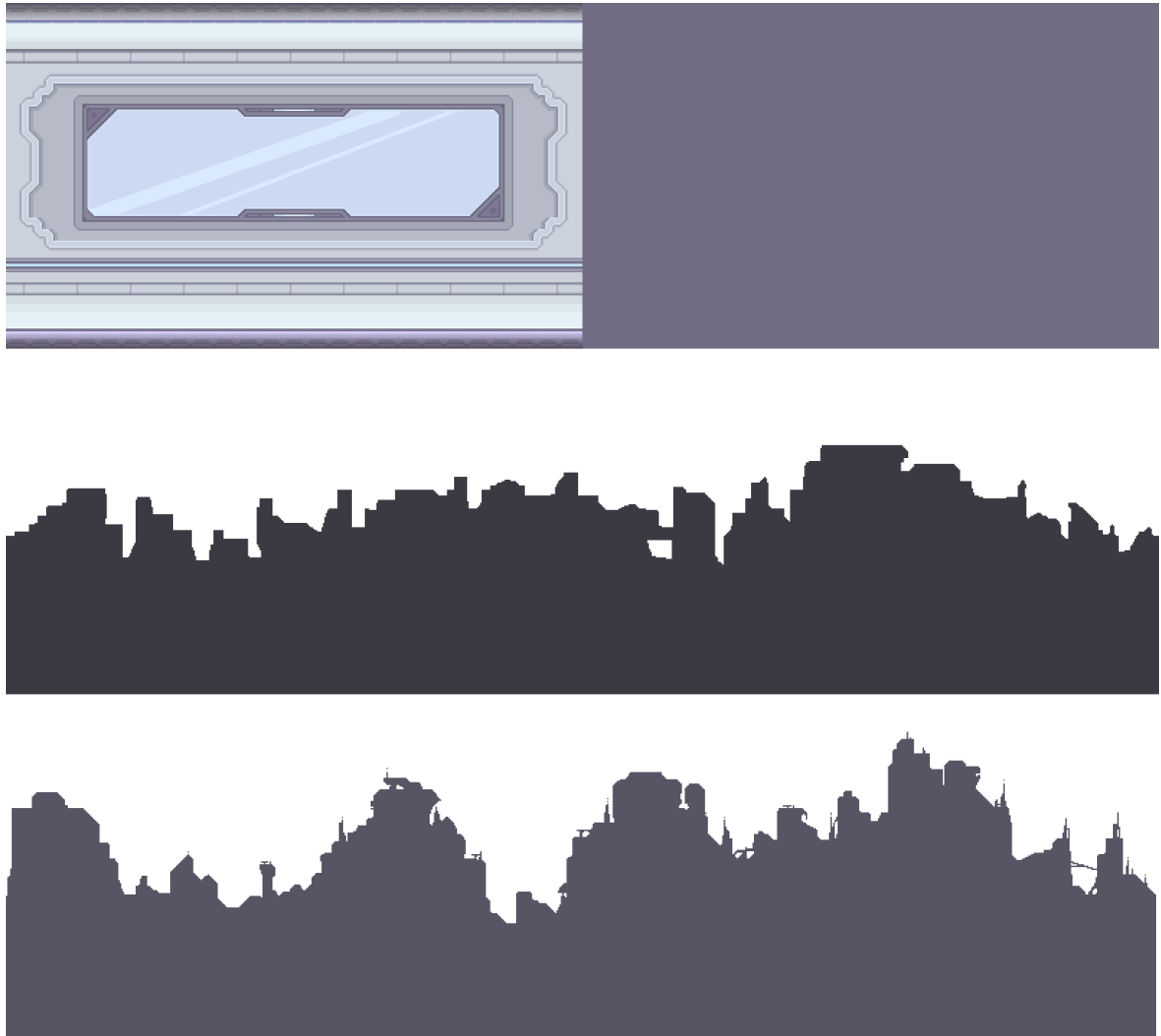


Figura 41: Spritesheet de los escenarios

6.9.3 Animaciones

La animación mediante *sprites* es una técnica usada en los videojuegos de dos dimensiones para dar la sensación de movimiento de un objeto o personaje. Está formada por una secuencia de *sprites*, también llamados *frames*, del objeto con diferente pose en cada uno, que son mostrados a una secuencia de intervalos establecidos. Por ejemplo, si queremos hacer la animación del personaje corriendo, tendríamos que tener en nuestro archivo un *sprite* con cada pose de correr por cada *frame* de la animación, como se ve en la imagen a continuación.

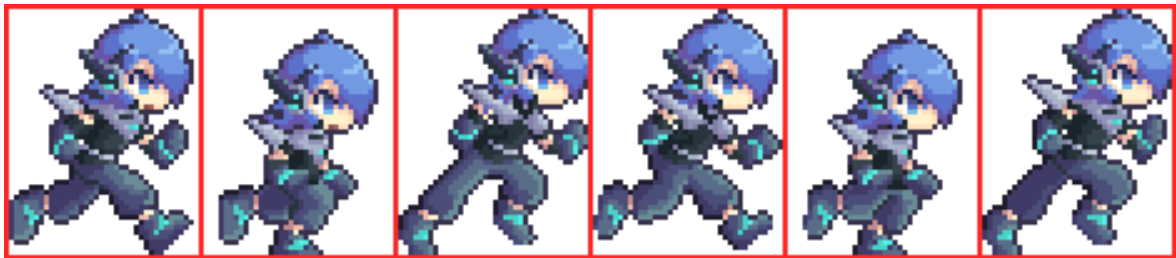


Figura 42: Frames de la animación del personaje corriendo

La animación de correr del personaje consta de 6 *frames* distintos que enlaza el último con el primero para poder ejecutar la animación en *loop*, es decir, en bucle. Para cada animación se tiene que determinar el tiempo de duración, es decir, el *frame rate*. El *frame rate* consiste en el número de veces que se cambiará entre *frames* por segundo, es decir, si nuestra animación tiene 6 *frames* en total y queremos que un ciclo de la animación dure 1 segundo, se tendrán que mostrar los 6 *frames* en un 1 segundo, con lo que el *frame rate* sería 6 FPS (en inglés *Frames Per Seconds*). Con esto se puede calcular el segundo valor que necesitamos, el *State Time*, o *Frame Time*, que calcula cuánto tiempo se mostrará un *frame* antes de sustituirlo por el siguiente. En nuestro caso, la fórmula del *State Time* sería la siguiente:

$$\text{State Time} = 1 \text{ segundo} / 6 \text{ frames} = 0.16$$

Esto quiere decir que con una animación de 6 *frames*, cada *frame* tiene que ser remplazado por el siguiente en un tiempo de 0.16 segundos.

Para crear nuestra animación en LibGDX le pasaremos a la clase *Animation* los siguientes parámetros: la duración de cada *frame*, es decir, el *frame rate*, y los distintos *frames* que tendrá la animación. Una vez cargada, a la hora de dibujarla tenemos que

llamar al método *getKeyFrame* y pasarle por parámetro el *State Time* y si queremos que se ejecute en *loop* o no.

6.9.4 Resoluciones

Uno de los mayores problemas del desarrollo de aplicaciones para dispositivos móviles es su amplia gama de modelos y su variedad de tamaños de pantalla. Cada compañía de telefonía tienen sus propios modelos y por tanto son los que deciden qué resolución ponerle a sus pantallas, esto nos dificulta la tarea a los desarrolladores que queramos que nuestras aplicaciones se vean bien en todos los dispositivos.

Una de las opciones para hacer esto es hacer nuestra aplicación o juego con diferentes resoluciones y que el usuario pueda elegirlos desde el menú por ejemplo, pero claro, aún así al existir un gran número de éstas, es difícil ajustar tu juego a todas, por lo que inevitablemente habrá en móviles en los que no se vea bien o se estire la imagen deformándose levemente.

En el caso de este proyecto se tiene aún más difícil, cuando se estira o reajusta una imagen puede pasar desapercibido, ya que al tener más resolución no se notaría demasiado la diferencia, pero al ser los gráficos de este proyecto en pixel art, si los estiramos aunque sea muy poco se notaría bastante, ya que el pixel art tiene muy poca resolución y se vería que los píxeles no son perfectos.

Existen diversos métodos y estrategias para evitar estos problemas, aunque no hay ninguno que sea el definitivo y que te solucione el problema por completo. LibGDX, en una de sus últimas actualizaciones, nos proporciona una nueva herramienta para lidiar con este problema de diferentes resoluciones, los *viewports*.

6.9.4.1 Viewports en LibGDX

Como se ha explicado, tratar con diferentes tipos de pantallas requiere la decisión de tomar una determinada estrategia a la hora de implementar nuestra aplicación o videojuego. Para ello necesitamos tratar con los diferentes tamaños de pantalla y la relación de aspecto, o *aspect ratio* en inglés.

La relación de aspecto de una imagen es la proporción entre su ancho y su alto. Este dato se calcula dividiendo el ancho de la imagen por su altura.

LibGDX nos proporciona una serie de *viewports* para poder tratar con este problema que explicaremos a continuación. Para ello tenemos que tener claro que en nuestro juego contamos con dos ventanas, una de ellas es la ventana del juego, o del mundo, que tiene la resolución verdadera del juego, en nuestro caso 240x400 píxeles, y luego tenemos la ventana del dispositivo, o pantalla virtual, que la cogerá del móvil en el que se juegue.

Por lo tanto, volviendo a los *viewports*, tenemos los siguientes:

StretchViewport:

El *StretchViewport* es el *viewport* que viene por defecto. Coge el tamaño que tiene siempre la pantalla virtual, por lo que siempre se ajusta a toda la pantalla y rellena todo el espacio, escalando la imagen. El problema de este *viewport* es que no mantiene la relación de aspecto, por lo que la imagen en algunos dispositivos móviles con resolución no proporcional a la resolución del juego se verá muy alterada, ya que los gráficos en pixel art son más propensos a que se noten estos desperfectos.

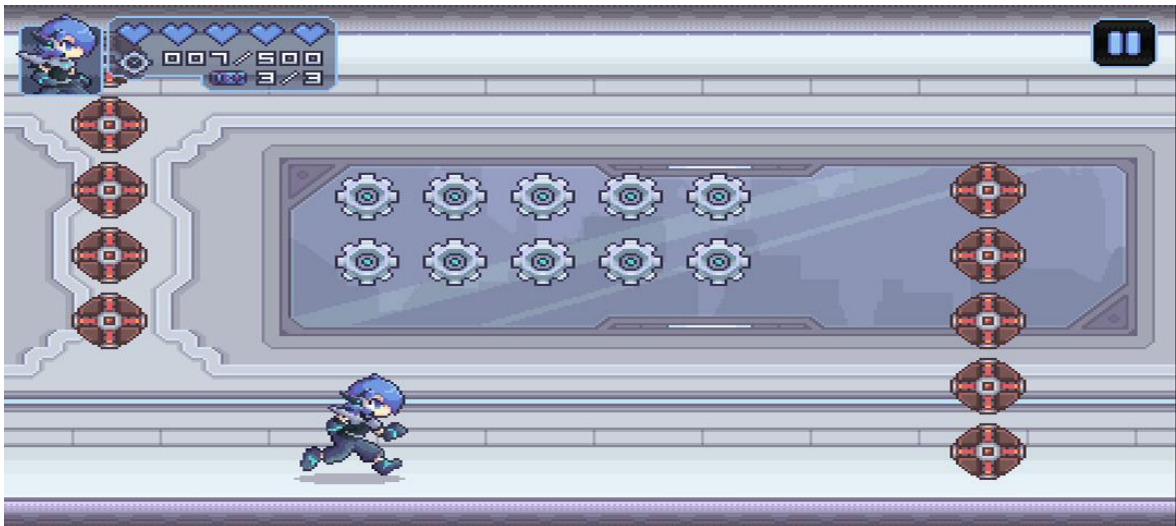


Figura 43: Visualización del *StretchViewport*

FitViewport:

Este *viewport* mantiene siempre la relación de aspecto del juego de la pantalla virtual, por lo que se puede escalar como cada uno desee sin que se pierda la proporción de la pantalla. La única desventaja de este *viewport* es que para lograr que se mantenga la relación de aspecto es que en algunos dispositivos móviles con resoluciones no proporcionales a la del juego aparecerán barras negras en los laterales de la pantalla.



Figura 44: Visualización del FitViewport

FillViewport:

El *FillViewport* es similar al *FitViewport*, pues también mantiene la relación de aspecto del juego, sin embargo tiene la diferencia de que este se ajusta rellenando por completo la pantalla, por lo que si el dispositivo móvil tiene una pantalla más ancha, con este *viewport* se ajustará la imagen al ancho y lo que sucederá es que las partes superiores de la pantalla se verán cortadas.



Figura 45: Visualización del FillViewport

ScreenViewport:

El *ScreenViewport* no tiene en cuenta el tamaño de la pantalla virtual, si no de la pantalla del dispositivo, esto quiere decir que no se escala y no aparece ningún tipo de barra negra en los laterales. La desventaja de este *viewport* es que en cada dispositivo móvil el juego cambiará, ya que el jugador con una pantalla más grande verá más zona de juego que un jugador con una pantalla más pequeña.



Figura 46: Visualización del ScreenViewport

ExtendViewport:

Este *viewport* consiste en escalar la imagen extendiéndose solo en una dirección. Mantiene la relación de aspecto, lo primero que realiza es el escalado hasta encajar dentro de la pantalla y a continuación la dimensión más corta se alarga para llenar la pantalla.

Para este proyecto se ha optado por utilizar el *FitViewport*. Al principio se pensó en usar el *ScreenViewport* porque se buscaba una solución que no se escalase la imagen en dispositivos con distinta relación de aspecto, pero el hecho de que hubiera jugadores que pudieran ver más parte de pantalla porque tuvieran móviles con pantallas más grandes se veía como desventaja para aquellos que tenían móviles más pequeños, debido al tipo de juego que se estaba desarrollando.

Por eso al final se optó por utilizar el *FitViewport*, que después del *ScreenViewport* era el que más se ajustaba a lo que se quería. Aunque en algunos dispositivos aparecieran las barras negras en los laterales, no alteraba el espacio que se veía en el juego y por tanto sería justo para todos los jugadores.

6.10. Audio

El audio en un videojuego es también un elemento clave para que el jugador consiga una mejor experiencia de juego. Para este proyecto, al ser un estilo futurista, se han buscado tanto efectos de sonido como melodías del mismo tipo para que concuerden con el entorno.

Para insertar los efectos y la música, LibGDX nos ofrece una interfaz de Audio muy sencilla de usar. Esta interfaz contiene la creación y la gestión de los recursos de audio y nos permite tener acceso directo al hardware de audio mediante las interfaces *AudioDevice* y *AudioRecorder*. Para los efectos de sonido tenemos la correspondiente interfaz *Sound* y para la música la interfaz *Music*.

Para cargar el audio tenemos que añadir los archivos correspondientes a nuestra carpeta de *Assets* del proyecto. En este caso, dentro de la carpeta de *Assets* se ha creado otra carpeta llamada *Audio*, y dentro de esta dos nuevas carpetas llamadas *SoundEffects* y *Musica*, para así tener el proyecto y los archivos más organizados y que sean más fáciles de encontrar.

Una vez añadidos los archivos, cargarlos en nuestro proyecto y hacer que se ejecuten en el juego se explicará en los siguientes apartados.

6.10.1 Efectos de Sonido

Para cargar un efecto de sonido en LibGDX se hace de esta sencilla manera. En nuestra clase de *Assets*, cargamos el sonido con esta línea de código:

```
Sound wavSound = Gdx.audio.newSound(Gdx.files.internal("data/wav.wav"));  
Sound oggSound = Gdx.audio.newSound(Gdx.files.internal("data/ogg.ogg"));  
Sound mp3Sound = Gdx.audio.newSound(Gdx.files.internal("data/mp3.mp3"));
```

Como podemos observar, LibGDX nos permite cargar archivos con 3 formatos diferentes: WAV, OGG y MP3. En este caso los efectos de sonido se han exportado como WAV ya que es por defecto cómo exporta los archivos el programa SFXR que explicaremos un poco más adelante.

Una vez cargados por código los archivos podremos hacer que suene durante el juego llamando al método *Play*. Además se le puede pasar por parámetro el volumen que queremos que se escuche ese sonido, por ejemplo, si le pasamos el valor de 0.5 el sonido se escuchará al 50%, o si queremos que se escuche con el volumen por defecto no se le pasará nada.

```
wavSound.play();  
oggSound.play(0.5f);
```

También tenemos los métodos de *Loop* si queremos que nuestro sonido se ejecute en bucle de forma infinita. Por defecto el sonido solo se ejecutaría una vez.

Para generar los efectos de sonido, se ha optado por utilizar la herramienta gratuita SFXR. Existe una versión superior a esta también gratuita y accesible desde la web de forma online, llamada BFXR, sin embargo, se ha considerado que para este proyecto la versión anterior ofrece todo lo necesario y además es una herramienta que ya se ha utilizado con anterioridad.

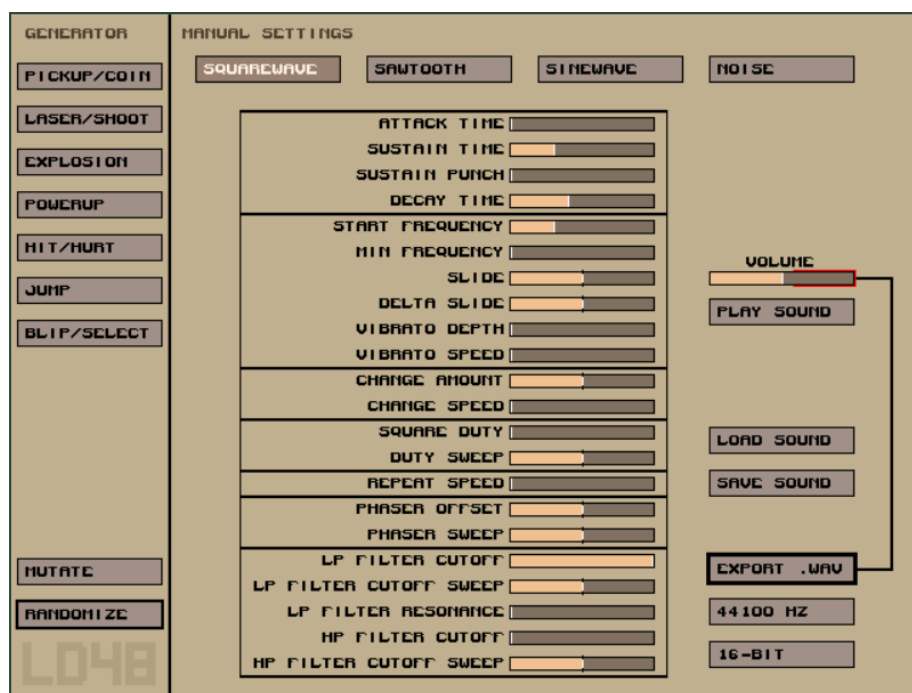


Figura 47: Panel de control del programa gratuito para efectos de sonido SFXR

Este programa es muy sencillo de usar y permite generar efectos de sonidos de 8 bits, usados en los videojuegos antiguos o retro. Ya que nuestro proyecto presenta un aspecto gráfico en pixel art y puede recordar a estos juegos retro, se ha visto acertado usar este tipo de sonidos.

Tal y como vemos en la imagen previa, SFXR tiene un amplio panel de control, podemos escoger generar el sonido de forma aleatoria, eligiendo un tipo de sonido del menú de la izquierda, así como modificar uno que se genere aleatoriamente hasta conseguir una variante cambiando los valores de los parámetros que vemos en la parte central. Gracias a esto podemos obtener un amplio abanico de sonidos muy diferentes entre sí.

Para nuestro juego se han generado un total de 23 efectos de sonido, en la siguiente tabla se muestran para qué se ha usado cada uno:

Nombre del archivo	Uso en el juego
menu_start.wav	Al pulsar en la primera pantalla
menu_option.wav	Al pulsar en una opción o botón
menu_back.wav	Al pulsar en volver en el menú
menu_play.wav	Al pulsar en un nivel y empezar el juego
start_level.wav	Cuando empieza el nivel
jump.wav	Al saltar el personaje
gear.wav	Al recoger un engranaje
fly.wav	Al volar el personaje
battery.wav	Al recoger una batería
heal.wav	Al recoger una esfera de salud
hurt.wav	Al recibir daño
dead.wav	Al morir el personaje
mode_item.wav	Al coger el objeto de modo
enemy_attack.wav	Ataque normal del enemigo
enemy_specialAttack.wav	Ataque especial del enemigo
enemy_hurt.wav	Al recibir daño el enemigo

enemy_dead.wav	Al morir el enemigo
explosion.wav	Al estallar la bomba
laser.wav	Sonido de láser
shield.wav	Sonido del escudo
gameover.wav	Sonido de finalizar partida
atage_clear.wav	Sonido de completar partida
dialog.wav	Sonido de los diálogos

Tabla 1: Efectos de sonido empleados

6.10.2 Música Ambiente

Para la música ambiente del juego, o banda sonora, se ha optado por usar música de páginas gratuitas como *Freesound* o *Jamendo* que tenga licencia libre para su uso comercial.

De forma similar a los efectos de sonido, para cargar los archivos de música con LibGDX

Se han utilizado para el proyecto un total de 4 pistas de música para los distintos niveles y menús.

Nombre del archivo	Uso en el juego
music_menu.mp3	Música para el menú
music_tutorial.mp3	Música para el tutorial
music_level.mp3	Música del nivel de recogida
music_boss.mp3	Música para el nivel del jefe

Tabla 2: Música ambiente empleadas

6.11 Requisitos funcionales

Los requisitos funcionales tratan de declaraciones sobre los servicios que ofrecerá el sistema, en este caso, el videojuego, la forma en la que éste debe reaccionar al usuario y de cómo se debe comportar en situaciones concretas. Muchas veces los requisitos funcionales también incluyen las acciones que no debe realizar la aplicación.

Como se ha dicho, los requisitos funcionales describen lo que el sistema debe hacer en casos particulares, estos requisitos dependen mucho del tipo del software. Para definir un requisito funcional lo primero que necesitamos es una necesidad o un problema.

A continuación se expondrán los requisitos funcionales de este proyecto, se enumerarán mediante la ID y se le asociará una prioridad entre 1 y 5, siendo mayor la prioridad cuánto más alta sea la cifra.

ID	Requisito	Descripción	Prioridad
RF01	Ejecutar el juego	El sistema debe ejecutar e iniciar el juego cargando todos los recursos necesarios.	5
RF02	Mostrar pantalla de carga	El sistema debe mostrar la pantalla de carga con el logo del autor.	1
RF03	Mostrar menú principal	El sistema debe mostrar el menú principal una vez cargados todos los recursos.	5
RF04	Mostrar menú de selección de niveles	Tras mostrar el menú principal y pulsar en la opción de comenzar partida, el sistema debe mostrar el menú de selección de niveles para que el jugador escoja el nivel que desea jugar.	5
RF05	Mostrar panel de opciones	En el menú principal, si el jugador pulsa en el botón de opciones, el sistema debe mostrar el panel con las opciones disponibles para configurar el juego.	4

RF06	Mostrar panel de créditos	En el menú principal, si se pulsa en el botón de créditos, el sistema debe mostrar el panel con los créditos del juego.	3
RF07	Mostrar panel de retos y logros	En el menú principal, si se pulsa en la opción de ver los retos y logros el sistema debe mostrar el panel con los retos y logros del juego.	4
RF08	Abrir votaciones	En el menú principal, si se pulsa en la opción de votar, el sistema debe abrir la aplicación de la tienda y mostrar el panel para efectuar la votación y valoración del juego.	3
RF09	Abrir redes sociales	En el menú principal, si se pulsa en los botones de las redes sociales, el sistema debe abrir el navegador de internet y abrir la respectiva red social del juego.	2
RF10	Iniciar partida	En el menú de selección de niveles, si se pulsa en un nivel el sistema debe de iniciar la partida mediante una transición de pantalla.	5
RF11	Mostrar pantalla de juego	Al iniciar la partida, el sistema debe de mostrar los recursos necesarios por pantalla.	5
RF12	Controlar el personaje principal	En la pantalla de juego, el sistema debe permitir que el jugador controle el personaje principal de la forma que desee y que esté establecida.	5
RF13	Interactuar con objetos	En la pantalla de juego, el sistema debe permitir interactuar al jugador con los objetos que se mostrarán por pantalla.	5

RF14	Interactuar con obstáculos	En la pantalla de juego, el sistema debe interactuar con los obstáculos que se mostrarán por pantalla cuando el jugador colisione con ellos y así perder vida.	5
RF15	Interactuar con enemigos	En la pantalla de juego, el sistema debe permitir interactuar al jugador con los enemigos que aparecerán por pantalla.	5
RF16	Mostrar y actualizar el HUD	En la pantalla de juego, el sistema debe mostrar y actualizar en todo momento el HUD para esté presente toda la información de la partida y el progreso del jugador en el juego.	5
RF17	Mostrar menú de pausa	En la pantalla de juego, si se pulsa en la opción de pausa, el sistema debe mostrar el menú de pausa con sus respectivas opciones.	4
RF18	Mostrar panel de Game Over	En la pantalla de juego, cuando el jugador pierda la partida, el sistema debe mostrar el panel de Game Over con la información del progreso que el jugador ha obtenido durante la partida.	5
RF19	Mostrar panel de Nivel Completado	En la pantalla de juego, cuando el jugador complete el nivel que ha elegido el sistema debe mostrar el panel de Nivel Completado.	5
RF20	Mostrar panel de Nivel Completado al 100%	En la pantalla de juego, cuando el jugador complete el nivel de forma 100%, el sistema debe mostrar el panel de Nivel Completado a 100%.	4
RF21	Mostrar transición entre pantallas	El sistema debe mostrar una transición entre las pantallas o menús.	4

RF22	Salir de la partida	En la pantalla del juego, el sistema debe permitir al jugador salir del juego si se pulsa la opción de pausa y se selecciona la opción de salir al menú principal o salir al menú de selección de niveles.	5
RF23	Salir al menú principal	En la pantalla de juego, el sistema debe permitir al jugador salir al menú principal si se pulsa en la opción de pausa y se selecciona la opción de salir al menú principal.	4
RF24	Salir al menú de selección de niveles	En la pantalla de juego, el sistema debe permitir al jugador salir al menú de selección de niveles si se pulsa en la opción de pausa y se selecciona la opción de salir al menú de selección de niveles.	3
RF25	Guardar partida	Al terminar un nivel, el sistema debe de guardar el progreso que ha obtenido el jugador para que al volver a iniciar el juego la próxima vez el jugador pueda continuar por donde lo había dejado.	5
RF26	Guardar configuración	En el menú de opciones, si el jugador configura cualquier opción el sistema debe guardar la configuración para que la próxima vez que se ejecute el juego las opciones estén como las dejó el jugador.	3
RF27	Cerrar aplicación	El sistema debe cerrar la aplicación cuando el jugador lo desee.	5

Tabla 3: Requisitos Funcionales

6.11.1 HUD

El HUD es un elemento muy importante y que está presente en todos los videojuegos de una forma u otra. Muestra en todo momento por pantalla al usuario la información durante la partida, generalmente esta información se suele mostrar mediante iconos y números. En el HUD es típico encontrar la vida del personaje, los puntos conseguidos en el nivel, un minimapa, etc, dependiendo del juego.

En nuestro proyecto se ha pretendido que el HUD sea lo más sencillo posible y muestre la información necesaria al jugador en todo momento durante la partida. Al ser un juego para móviles principalmente, se ha optado por un diseño claro e intuitivo, no muy sobrecargado, para que no entorpezca la visibilidad, ya que contamos con una pantalla de tamaño más reducido que por ejemplo la de un ordenador.

El HUD cambiará dependiendo del nivel que el jugador juegue. En un nivel de recogida, el HUD mostrará la siguiente información:



Figura 48: HUD del jugador - Nivel de recogida

- Retrato del personaje: muestra el *sprite* del personaje que irá cambiando si sufre daño o si muere, como se mostrará a continuación, además también estará animado cuando haya diálogos durante la partida y el personaje hable.



Figura 49: Retratos del jugador - Nivel de recogida

- Vida del personaje: es lo más importante del personaje, la vida estará representada por corazones. Cada corazón se divide en dos, por lo que si el personaje recibe tres toques de daño le quitarían un corazón y medio. Si los corazones acaban todos vacíos, el personaje morirá.



Figura 50: Corazones de vida del jugador

- Número de engranajes conseguidos: muestra la cantidad de engranajes que el jugador ha recolectado hasta el momento.
- Número de engranajes totales: muestra la cantidad exacta de engranajes que hay en ese nivel.
- Baterías conseguidas: muestra el número de baterías que el jugador ha recogido hasta el momento.
- Baterías totales: muestra el número de baterías que hay en el nivel, que siempre serán 3.

Sin embargo, si se juega a un nivel de jefe, el jugador se encontrará con otro tipo de HUD que variará levemente del de nivel de recogida:



Figura 51: HUD del jugador - Nivel jefe

En este HUD no aparecerán los engranajes ni las baterías conseguidas ni totales, ya que en este nivel no aparecerán estos elementos. En su lugar aparecerán las Estrellas de Energía recogidas, que se podrán llevar como máximo 3. Conservamos el retrato del personaje y los corazones de vida, además de añadir dos elementos nuevos:

- Bocadillo de diálogo: a diferencia del nivel de recogida, el bocadillo de diálogo aparecerá en el HUD, ya que en los niveles de jefe el personaje participará en conversaciones durante el desarrollo de la partida, por lo que en esta nueva posición no entorpecerá la visibilidad del jugador.
- Modo actual: en este recuadro se mostrará el modo del personaje en ese momento. Si está vacío el personaje está en estado normal, si no, aparecerán uno de los siguiente iconos de modo, de ataque o de defensa:



Figura 52: Modo ataque



Figura 53: Modo defensa

En este nivel, además de tener los retratos del personaje de herido, normal o muerto, se añaden las siguientes:



Figura 54: Retratos del jugador para diálogos

Por otra parte, en este nivel también se mostrará el HUD del enemigo, es decir, del jefe del nivel. Será similar al del personaje, se mostrará también el retrato del jefe, que irá cambiando si recibe daño o muere, la vida, que disminuirá de igual forma que la del personaje principal, el número de Estrellas de Energía que va consiguiendo y el bocadillo de diálogo.



Figura 55: HUD del jefe

6.11.2 Controles

Al ser el proyecto un videojuego para dispositivos móviles con pantalla táctil, el jugador podrá interactuar con el juego mediante toques por toda la pantalla. El sistema detectará la zona donde ha pulsado el jugador y lanzará los eventos correspondientes, tal y como se ha explicado con detalle en la sección [6.8 Gestión de Inputs](#).

Los controles básicos del juego pretenden ser lo más sencillos posibles, ya que se ha buscado que el juego sea lo más fácil de jugar y abarque un mayor número de público posible. El jugador controlará al personaje principal mediante toques en la pantalla que hará que el personaje salte o vuele en el caso de pulsar varias veces o mantener

pulsado, esto hará que sea posible esquivar los obstáculos y recoger los elementos que aparecerán por la pantalla y que serán necesarios para completar la partida.

6.12 Requisitos no funcionales

Además de los requisitos funcionales, existen también los requisitos no funcionales del sistema. Los requisitos no funcionales, o también llamados atributos de calidad, son los requisitos que especifican los criterios que pueden usarse para analizar la operación del sistema en vez de sus comportamientos determinados, ya que estos últimos son los que analizan los requisitos funcionales que hemos mencionado en el apartado [6.11 Requisitos funcionales](#).

Por lo tanto, los requisitos no funcionales definen las restricciones del sistema, como la el rendimiento del sistema, la disponibilidad, la capacidad de los dispositivos de entrada y salida, o la usabilidad y accesibilidad de la aplicación.

En la tabla que se muestra a continuación se definirán los requisitos no funcionales que están implicados en este proyecto. De igual forma que los requisitos funcionales, se irán enumerando mediante la ID y se le asignará la prioridad correspondiente que se ha considerado oportuna.

ID	Requisito	Descripción	Prioridad
RNF01	Usabilidad	El sistema debe ser fácil de usar y ofrecer toda la ayuda necesaria y de forma sencilla al jugador mediante el HUD y la interfaz.	5
RNF02	Accesibilidad	El sistema debe ser accesible para que cualquier persona pueda utilizarlo independientemente de sus capacidades técnicas, cognitivas o físicas.	2
RNF03	Multiplataforma	El sistema debe funcionar en distintos sistemas operativos y plataformas hardware.	5
RNF04	Rendimiento	El sistema debe soportar el manejo de la cantidad necesaria de datos y recursos durante su ejecución.	5

RNF05	Estabilidad	El sistema debe ser robusto y no dar ningún error crítico durante su ejecución que obligue a cerrar la aplicación.	4
RNF06	Seguridad	El sistema debe proporcionar seguridad con respecto a los datos que maneja. Los datos que este proyecto gestionará sobre los usuarios es su cuenta de correo electrónico para poder acceder a la API de retos de Google Play para dispositivos Android.	5
RNF07	Fiabilidad	El sistema debe proporcionar una fiabilidad lo más alta posible para que la experiencia del jugador no se vea entorpecida debido a fallos o errores del sistema.	5
RNF08	Desempeño	El sistema no debe presentar problemas para su manejo e implementación.	2

Tabla 4: Requisitos no Funcionales

7. Publicación

Una vez que tengamos terminada nuestra aplicación o juego es normal que queramos que la gente la pruebe y la descargue en sus dispositivos móviles. Para esto existen varias plataformas de distribución que nos ofrecen una forma sencilla de subir nuestras aplicaciones a la red para que sea accesible a cualquier usuario, eligiendo además la forma de monetizarlas.

Existe una plataforma de distribución, o también considerada tienda online, para cada sistema operativo de *smartphones*, desarrolladas por su respectiva compañía. En este proyecto analizaremos y explicaremos las principales y más usadas hoy en día: Google Play para dispositivos con sistema operativo Android y App Store para dispositivos con iOS.

7.1 Google Play

Google Play es el servicio que nos ofrece Google para disfrutar de cualquier tipo de contenido en nuestro dispositivo móvil Android de manera muy sencilla. Antes conocido como Android Market, Google Play se trata de una plataforma de distribución digital que contiene aplicaciones móviles desarrolladas con Android SDK, la herramienta de desarrollo para el sistema operativo Android. Es como una tienda online que permite a los usuarios navegar y descargar ya sean juegos, películas, música e incluso libros y revistas en su dispositivo móvil.

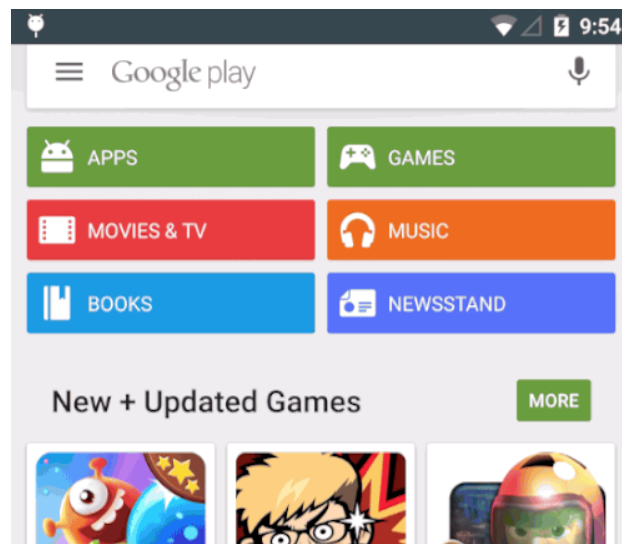


Figura 56: Interfaz de Google Play

Las aplicaciones que podemos encontrar en Google Play pueden ser gratuitas o de pago. Cada aplicación tiene su ficha correspondiente que nos proporciona información como la descripción de la aplicación, el tipo, la fecha de publicación, comentarios, valoraciones, si ha habido actualizaciones, imágenes previas, etc.

La cantidad de aplicaciones publicadas en esta plataforma sobrepasó el millón y medio y se registró más de mil millones de usuarios en el año 2014, según Carlos Álvarez, director de desarrollo de negocio de Google. Actualmente se considera la plataforma de distribución más utilizada por los usuarios debido a la cantidad de dispositivos móviles con Android disponibles que sigue creciendo día a día.

7.1.1 Publicación en Google Play

Para que nuestra aplicación esté disponible en Google Play lo primero que tenemos que hacer es, si disponemos de una cuenta de Google transformarla en cuenta de desarrollador a través de [Google Play Developer Console](#), en el caso de que no dispongamos de cuenta, tendremos que crearnos una previamente.

Una vez hayamos transformado nuestra cuenta nos pedirá el pago de la tasa que Google pone a todos los desarrolladores para que puedan incluir sus aplicaciones en la tienda. Esta tasa consiste en 25 dólares que solo es necesario pagar una sola vez. Nos pedirá también aceptar las condiciones de Google.

Una vez pagada la tasa y aceptadas las condiciones, nos aparecerá nuestro *Developer Console* con un menú de opciones a la izquierda y una lista con nuestras aplicaciones al lado, como podemos observar en la siguiente imagen:

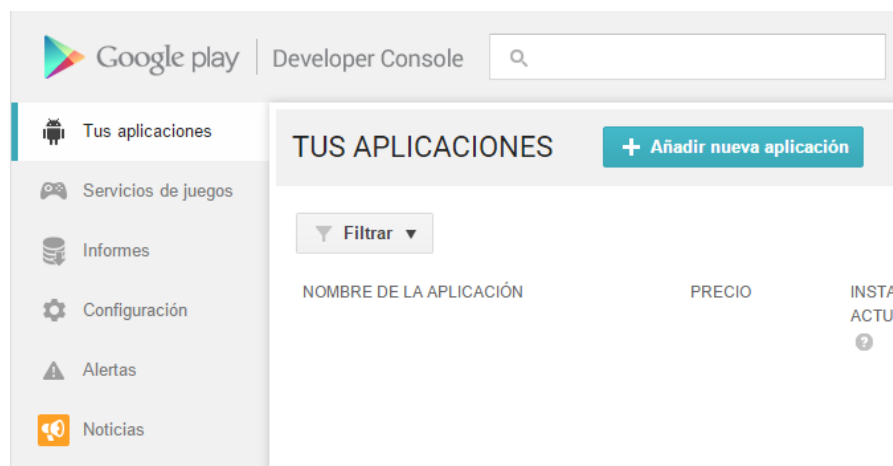
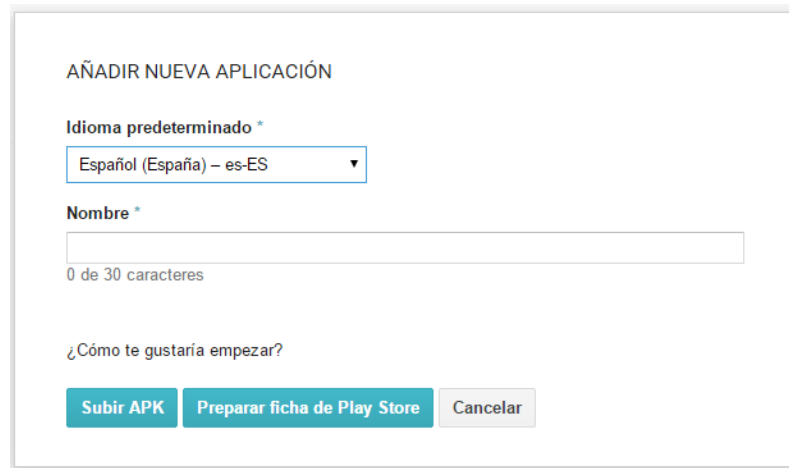


Figura 57: Google Play - Developer Console

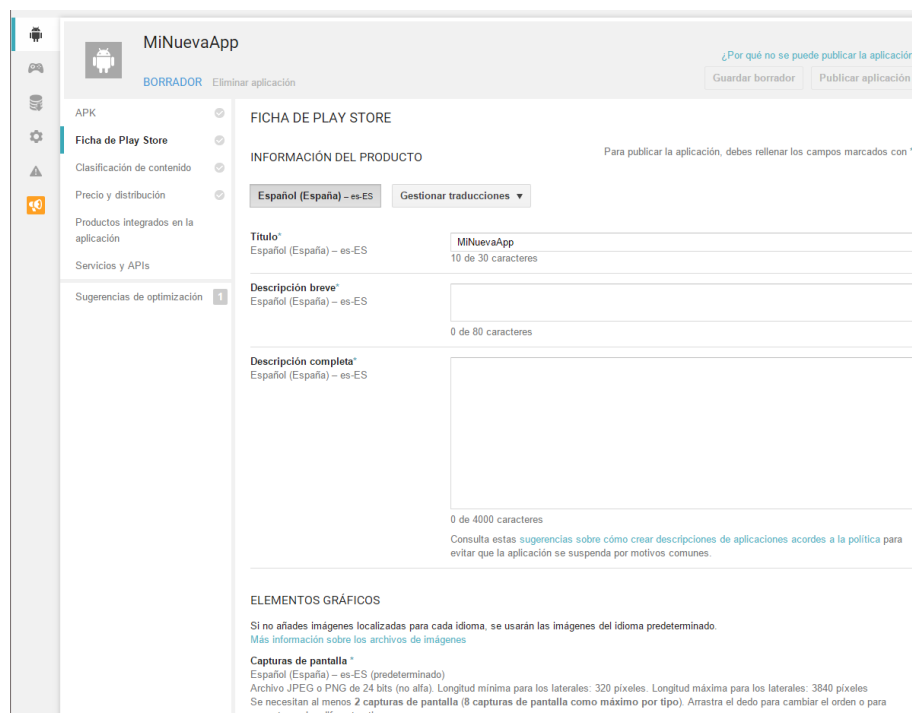
Para añadir una nueva aplicación tenemos que pulsar en el botón 'Añadir nueva aplicación' situado en la parte superior derecha. Esto nos abrirá una nueva ventana para seleccionar el idioma predeterminado y el título de la aplicación. Podremos elegir si subir el archivo APK directamente o preparar antes la ficha que aparecerá en Google Play. En nuestro caso prepararemos antes la ficha.



The screenshot shows a dialog box titled "AÑADIR NUEVA APLICACIÓN". It contains a dropdown menu for "Idioma predeterminado *" set to "Español (España) - es-ES". Below it is a text input field for "Nombre *" with a character count "0 de 30 caracteres". At the bottom, there is a question "¿Cómo te gustaría empezar?" and three buttons: "Subir APK" (blue), "Preparar ficha de Play Store" (blue), and "Cancelar" (grey).

Figura 58: Google Play - Añadir nueva aplicación

A continuación nos aparecerán los distintos campos de la ficha que tendremos que rellenar sobre nuestra aplicación, así como la descripción, capturas de pantalla, vídeo promocional, tipo de aplicación, público al que va dirigido, etc.



The screenshot shows the "Ficha de Play Store" page for an application named "MiNuevaApp". The left sidebar contains navigation options: "APK", "Ficha de Play Store" (selected), "Clasificación de contenido", "Precio y distribución", "Productos integrados en la aplicación", "Servicios y APIs", and "Sugerencias de optimización". The main content area is titled "FICHA DE PLAY STORE" and "INFORMACIÓN DEL PRODUCTO". It includes a language selector set to "Español (España) - es-ES" and a "Gestionar traducciones" button. The form fields include: "Título*" (filled with "MiNuevaApp", 10 de 30 caracteres), "Descripción breve*" (empty, 0 de 80 caracteres), and "Descripción completa*" (empty, 0 de 4000 caracteres). A note at the bottom states: "Si no añades imágenes localizadas para cada idioma, se usarán las imágenes del idioma predeterminado. Más información sobre los archivos de imágenes". Below this is the "ELEMENTOS GRÁFICOS" section, which includes instructions for "Capturas de pantalla *" and a note about the minimum number of screenshots required (8).

Figura 59: Google Play - Ficha para rellenar de la aplicación

En el menú desplegado que observamos a la izquierda tenemos más opciones para rellenar de nuestra aplicación, como la clasificación de contenido, si queremos asignarle un precio (en cuyo caso necesitaríamos una cuenta de pago válida) o que sea gratuita, etc. Es recomendable leerse las recomendaciones de optimización que Google te ofrece en la opción de 'Sugerencias de optimización'.

Una vez completada toda la ficha y subido el archivo APK de nuestra aplicación, podremos publicarla cambiando el estado de Borrador. Nuestra aplicación tardará al menos un día hasta que aparezca disponible en Google Play.

Para más información y más detallada Google nos proporciona a los desarrolladores una ayuda para la publicación, administración y políticas de las aplicaciones en la siguiente web oficial: <https://support.google.com/googleplay/android-developer#topic=3450769>

7.2 App Store

De forma similar a Google Play, Apple nos proporciona su propio servicio para iPhone, iPods Touch, iPad o MAC OS, la plataforma de distribución APP Store, que permite a los usuarios buscar y descargar aplicaciones desarrolladas con el iPhone SDK. Estas aplicaciones pueden ser tanto gratuitas como de pago.



Figura 60: Interfaz de App Store

7.2.1 Publicación en App Store

Para publicar nuestra aplicación en App Store debemos previamente registrarnos como [desarrollador o programador de aplicaciones](#) y darnos de alta en el programa *iOS Developer Program*. La cuota de registro es de 99 dólares por año.

Una vez creada nuestra cuenta y pagadas las tasas, a continuación crearemos los certificados necesarios mediante el *Member Center* para compilar nuestra aplicación.

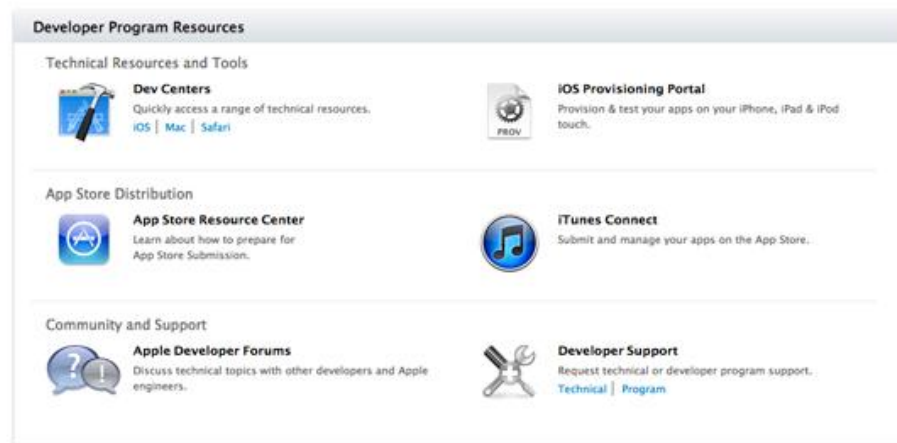


Figura 61: Developer Apple - Member Center

El paso que seguiremos a continuación es entrar en *iTunes Connect* identificándonos con nuestra Apple ID, la misma que hemos usado al registrarnos como desarrolladores. Una vez dentro, nos aparecerá la opción de '*Manage Your Applications*' o '*Gestionar tus Aplicaciones*' y procederemos a darle a '*Add new App*' o '*Añadir nueva App*'.

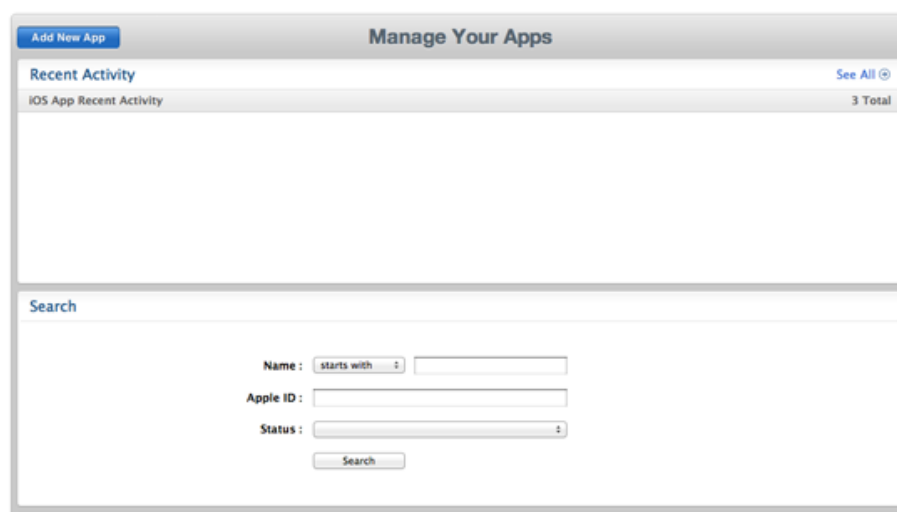


Figura 62: iTunes Connect - Gestionar Aplicaciones

Procederemos a rellenar los datos necesarios sobre nuestra aplicación, como puede ser el nombre, el idioma, una descripción, palabras clave, etc. También tendremos que definir a qué público está destinada nuestra aplicación, el número de versión y el precio que queremos asignarle.

Tendremos que ir rellenando formularios hasta que lleguemos a una pantalla para añadir el icono de la aplicación y dos imágenes que se mostrarán cuando se descargue. Estas imágenes tienen una medida específica que nos lo indica en la ayuda.

Una vez nos aparezca una pantalla con la información principal y el icono de nuestra aplicación, pulsaremos en la opción '*Ready to Upload Binary*' para indicar que hemos terminado. Solo nos quedará probar y subir nuestra aplicación usando el Xcode para que esté a la espera de confirmación y ser subida a App Store.

8. Publicidad

El tema de la monetización es uno de los dilemas más habituales para el desarrollador a la hora de lanzarse a crear y publicar una aplicación. Es normal que lo que se pretenda es recuperar la inversión empleada en el desarrollo de la aplicación.

Sin embargo, el mercado de las aplicaciones crece cada vez más y más y lo hace a una velocidad muy alta, por lo que cada vez resulta más competitivo y difícil conseguir dar visibilidad a nuestra aplicación para que llegue a los usuarios y la descarguen, y por tanto, llegar a conseguir algún beneficio.

Existen varias opciones para monetizar nuestra aplicación, una de las opciones es poner nuestra aplicación de pago, sin embargo, debido a la gran cantidad de aplicaciones gratuitas en este sector, el público cada vez se acostumbra más a descargar todo gratis y es reacio a pagar por una aplicación en el móvil, por lo que es difícil llegar a conseguir algo de ganancias con las primeras aplicaciones, a no ser que se tenga ya un buen público o mucha suerte. También se puede poner la aplicación gratuita y ofrecer compras integradas, una estrategia que se está utilizando cada vez con más frecuencia.

Otra solución es la publicidad dentro de nuestra aplicación, también conocida en inglés como *In-App Advertising*. Es una de las opciones más frecuentes y consiste en ofrecer la aplicación gratis y cobrar por la publicidad que aparezca en la misma. Lo más común es poner la publicidad en forma de banner en partes concretas de la pantalla, como por ejemplo, en un juego, entre el cambio de nivel o la pantalla de carga. En el apartado 8.1 Estrategias de publicidad se describen las estrategias más comunes y útiles para colocar de manera efectiva la publicidad en una aplicación.

Existen varias plataformas publicitarias para incluir publicidad en tu aplicación, pero para este proyecto se ha optado por incluir la publicidad mediante AdMob, que se explicará en el apartado 8.2 Estrategia de publicidad y negocio elegida.

8.1 Estrategias de publicidad y negocio existentes

Existen diversas estrategias para publicitar tu aplicación y que llegue al mayor número de personas posibles, además de conseguir algo de beneficio económico. Aunque algunas ya las hemos mencionado antes, las explicaremos aquí con más detalle.

1. Poner la aplicación de pago

Es la estrategia clásica y la más obvia, pero eso no quiere decir que sea la mejor. Al introducir una aplicación en una plataforma de distribución se nos da la posibilidad de indicar un precio, sin embargo, hay que tener en cuenta que la mayoría de las aplicaciones que se encuentran en las tiendas online del sector móvil, sobre todo en Google Play, son aplicaciones gratuitas, y la gente suele ser reacia a comprar aplicaciones si pueden obtener lo mismo o algo similar de forma gratis.

Por lo tanto, antes de decidir si poner nuestra aplicación de pago hay que pensar si realmente ofrece una función útil, si es adictiva, novedosa, etc. Si no, desgraciadamente es difícil que llegue a algún sitio.

2. Versión de prueba gratuita y versión de pago

Hoy en día es una estrategia bastante utilizada por las grandes empresas. Consiste en sacar tu aplicación o juego de pago, pero subiendo también una versión de prueba o demo con las funciones reducidas para que los usuarios la prueben y puedan hacerse una idea de cómo es la aplicación sin necesidad de realizar ningún pago inicial. Si al usuario le llega a gustar la aplicación o juego y le engancha lo suficiente tu juego, no dudarán en comprar la versión completa de pago.

3. Publicidad dentro de la aplicación

Es la estrategia más común utilizada por los desarrolladores. Consiste en introducir anuncios de publicidad dentro de tu aplicación mediante las plataformas publicitarias como AdMod, iAd o AdColony que irán incrementando tus ingresos a medida que los usuarios vayan pulsando en los anuncios. Cada vez que un usuario pulse en un anuncio, se ganará una cantidad determinada.

Esta publicidad puede estar presente de muchas formas, como por ejemplo, en determinadas pantallas de la aplicación o el juego, mediante banners pequeños, a pantalla completa, o de forma voluntaria para incitar a los usuarios a pulsar en la publicidad y conseguir algún objeto o característica en el juego, etc.

4. Compras integradas dentro de la aplicación

Conocidas también como micropagos *In-App*, son muy utilizadas por los desarrolladores de juegos, consisten en poner la aplicación gratis e incluir dentro compras de elementos del juego, como pueden ser armas, vidas, monedas, o incluso desbloquear o avanzar niveles de forma automática. Es una buena estrategia para ganar dinero extra.

5. Coste por instalación

Esta estrategia es similar a la de publicidad, pero en vez de anuncios de todo tipo, lo que se publicitan son otras aplicaciones con una temática similar a la nuestra. Cuando un usuario descarga la aplicación desde tu anuncio, ganarás una cantidad determinada, normalmente mayor a la de la publicidad normal.

8.2 Estrategia de publicidad y negocio elegida

AdMob es una plataforma publicitaria adquirida por Google en 2009. Se trata de uno de los gestores de publicidad más usados hoy en día por los desarrolladores de este sector que nos ayuda a dar visibilidad a nuestra aplicación y nos ofrece formatos de anuncios enriquecidos permitiéndonos orientarlos según los dispositivos.

Esta plataforma incluye el servicio de Google Analytics, que nos sirve para obtener información sobre los usuarios que prueben nuestra aplicación o juego, como por ejemplo, qué tipo de personas juegan a nuestro juego, cuál es la edad media, dónde es más popular, etc. Esto nos ayuda a preparar estrategias y descubrir nuevas oportunidades para mejorar la experiencia de juego y que llegue cada vez a más público .

Otra de las razones por la que se ha usado AdMob es la compatibilidad que tiene con las distintas plataformas como Android e iOS, y la fácil integración con LibGDX.

La publicidad en AdMod funciona mediante clics e impresiones. Una vez puesto nuestro anuncio mediante el código que nos proporciona la plataforma, cada vez un usuario pulse en esa publicidad ganaremos una cantidad determinada. Las impresiones nos indican las veces que el anuncio ha sido mostrado en la aplicación.

Para este proyecto, se ha considerado poner la publicidad a modo de banners en la parte superior de las pantallas del menú principal, game over, fase completada y pausa, de forma que no aparezca mientras el jugador está en plena partida y moleste, y llame la atención cuando esté descansando entre un nivel u otro, o navegando por el menú

principal. A continuación se muestra un ejemplo de un banner de prueba para mostrar cómo se quedaría en una de las pantallas.



Figura 63: Ejemplo de posición y tipo de banner publicitario

Algo importante que también tenemos que tener en cuenta es filtrar el tipo de publicidad que queremos que aparezca a los usuarios. Ya que nuestro juego es para todos los públicos, queremos decir que todo lo que aparece en pantalla es para todos los públicos, incluida la publicidad. Por lo que tendremos que entrar en las propiedades de los anuncios y configurar la audiencia.

9. Estudio de viabilidad

El estudio de viabilidad es un análisis que se realiza a un proyecto para identificar su éxito o fracaso a partir de una serie de datos basados en la rentabilidad del proyecto, legislación aplicable, necesidades del mercado, el medio físico, etc.

Este estudio de viabilidad sentará las bases para la toma de decisiones en la realización del proyecto. Aunque en un principio este estudio se realiza como un pronóstico, a lo largo de todo el desarrollo, a este estudio, se le debe realizar un seguimiento continuado, y poner en marcha todas las medidas paliativas y correctoras necesarias que vayan apareciendo.

El contenido del estudio de viabilidad varía según el proyecto y es muy aconsejable realizarlo por muy pequeña que sea la empresa o el proyecto, por lo que a continuación se explicarán en los siguientes apartados un análisis de la viabilidad económica-financiera del proyecto, es decir, se estimarán los costes que abarcará este proyecto. A continuación se realizará un análisis de la viabilidad conceptual mediante un DAFO y por último un análisis de los riesgos.

9.1 Viabilidad económica

Este proyecto se propone como una aplicación gratuita para incluir en Google Play, la plataforma de distribución de Google, por lo tanto, la justificación económica es la monetización a través de la publicidad que se incluye en el juego y no se puede concretar hasta que no se vea el nivel de aceptación por los usuarios. Aunque los resultados esperados en estos tipos de plataformas son alentadores por el poder de difusión y la gran cantidad de usuarios potenciales.

9.2 Viabilidad técnica

Técnicamente el proyecto es viable y como hemos visto anteriormente, existen las herramientas y tecnologías necesarias para su desarrollo.

9.3 Viabilidad legal

No se ha encontrado ningún problema legal con este proyecto, pues el material que se necesita para el desarrollo es de elaboración propia o está disponible de forma gratuita y libre para su distribución.

9.4 Estimación de costes

Una vez descompuesto los objetivos del proyecto en actividades, es posible comenzar a estimar el tiempo y realizar una planificación temporal que nos llevará a la realización del proyecto.

Crear un calendario para estas tareas es un punto importante a la hora de gestionar un proyecto y tiene como objetivo dejar claro lo que se espera y para cuándo, y si es posible realizarlo. El calendario además debe estar en constante revisión y ajustarlo siempre que sea necesario.

En cuanto a las tareas, existe varios métodos para representarlas de modo visual, entre ellos los siguientes:

- Diagrama de Gantt
- Diagrama de flechas
- Diagrama de red
- Diagrama de precedencias

A continuación se muestra una tabla con la planificación temporal que se estima para la realización de este proyecto. Las tareas expuestas son las que se han considerado más relevantes a la hora del desarrollo. A partir de esta tabla se pretende presentar un estimación del esfuerzo que lleva la realización de este proyecto.

Nombre de la tarea	Horas
Estructuración del proyecto	10
Programación	
Movimiento del personaje	15
Movimiento del escenario	5
Generador de los objetos	10

Generador de los obstáculos	10
Gestión del ciclo de la partida	40
HUD	30
<i>Viewports</i> para distintas resoluciones	35
Menú principal	30
Menú de selección de niveles	30
Menú de pausa	10
Pantalla de resultados	10
Efectos	15
Movimiento del enemigo jefe	5
Comportamiento de los cañones	45
Modos del personaje	15
Ataque especial del jefe	10
Introducción de los efectos de sonido	5
Introducción de la música ambiente	2
Escena del jefe	10
Diálogos	10
Gráficos	
Botones del menú principal	15
Botones y paneles de la selección de niveles	20
Panel de opciones	5
Panel de créditos	3
Escenario laboratorio	10
Personaje principal	
<i>Sprites</i> de la animación quieta	10
<i>Sprites</i> de la animación de correr	15
<i>Sprites</i> de la animación de morir	7

<i>Sprites</i> de la animación de atacar	10
<i>Sprites</i> de la animación de defenderse	10
<i>Sprites</i> de la animación de coger estrella de energía	3
<i>Sprites</i> de animaciones de obstáculos	5
<i>Sprites</i> de animaciones de objetos	5
HUD	8
Enemigo Jefe	
<i>Sprites</i> de la animación de moverse	10
<i>Sprites</i> de la animación de coger estrella de energía	3
<i>Sprites</i> de la animación de morir	5
<i>Sprites</i> de la animación de atacar	15
Panel de pausa	5
Panel de resultados	5
Documento	
Planificación del proyecto	7
Diseño del proyecto	10
Desarrollo de la implementación	20
Estudio de viabilidad	8
Estimación de costes	10
Redacción de la memoria	50
TOTAL	626

Tabla 5: Tareas y estimación de tiempos

Para obtener la estimación de costes total se tienen en cuenta los siguientes datos:

- La jornada de trabajo sería de 8 horas diarias
- Trabajo en los días de lunes a sábado (6 días), lo que conlleva a 48 horas por semana.
- Personal implicado, 1 programador con conocimientos de diseño, preferiblemente ingeniero multimedia.

- El coste por hora sería de 15 euros, el saldo medio de un desarrollador de juegos independiente.

Por lo tanto, obtenemos los siguientes resultados.

	Horas	Días	Coste (€)
Total	626	78,25	9,390

Tabla 6: Estimación de costes

El tiempo estimado de trabajo es de 78,25 días, lo que equivale a un desarrollo de aproximadamente 3 meses con un coste de 9,390 euros.

9.5 Análisis DAFO

El análisis DAFO consiste en valorar las debilidades, amenazas, fortalezas y oportunidades que abarca un proyecto o empresa en un momento determinado. Se trata de un método de análisis básico e imprescindible que se debe realizar al comienzo de cualquier proyecto y conviene elaborarlo cada cierto tiempo para conocer las situaciones que debemos evitar y cuáles son las que debemos aprovechar.

El uso de este análisis está muy extendido debido a que puede aplicarse a cualquier tipo de proyecto. Se encuentra dividido en dos partes principales, el análisis interno y el análisis externo:

- **Análisis interno:** se compone por las fortalezas y debilidades, en esta parte se debe analizar de forma meticulosa los puntos claves como la producción, el marketing, los recursos humanos o la situación financiera.
- **Análisis externo:** está compuesto de los dos puntos restantes, las amenazas y las oportunidades. Tenemos que tener presente la competencia en el sector de negocio, la situación del mercado o proveedores.

A continuación se mostrará el análisis DAFO de este proyecto en la siguiente tabla:

<p>Debilidades</p> <ul style="list-style-type: none"> • Falta de experiencia en el mercado • Falta de recursos financieros 	<p>Amenazas</p> <ul style="list-style-type: none"> • Fuerte competencia por empresas veteranas • Velocidad elevada en la evolución de las tecnologías, dificultad para adaptarse • Incertidumbre sobre la tendencia del mercado
<p>Fortalezas</p> <ul style="list-style-type: none"> • Expansión de la industria de los videojuegos móviles • Precios accesibles 	<p>Oportunidades</p> <ul style="list-style-type: none"> • Avance constante de la tecnología • Mercado de los dispositivos móviles en desarrollo • Plataformas de publicación accesibles y con costes asumibles

Tabla 7: Análisis DAFO

9.6 Análisis de riesgos

El análisis de riesgos es el que determina cuáles son los factores de riesgo y su impacto en un proyecto que potencialmente tendrían un mayor efecto sobre el desarrollo del mismo.

El número de riesgos que pueden aparecer puede ser incalculable, por eso a continuación se mostrarán algunos de los que se pueden presentar de forma representativa:

Riesgo	Avería en la estación de trabajo
Descripción	Imposibilidad temporal de poder continuar con el desarrollo del proyecto a causa de una avería en la maquinaria utilizada.
Tipo de riesgo	Hardware y Software

Impacto	Cumplimiento con los objetivos del proyecto
Probabilidad	Baja
Acción	Instalación de todo el software necesario en otro equipo disponible y continuar con el desarrollo intentando recuperar el tiempo perdido.

Tabla 8: Riesgo - Avería en la estación de trabajo

Riesgo	Error de planificación
Descripción	Encuentro con errores en la planificación del proyecto debido a un conocimiento insuficiente que impida cumplir con los objetivos a tiempo.
Tipo de riesgo	Organizacional
Impacto	Cumplimiento con los objetivos del proyecto
Probabilidad	Alta
Acción	Reajuste de la planificación y recuperación del tiempo atrasado de forma que la desviación de tiempo que se produzca sea la menor posible y cause un impacto mínimo.

Tabla 9: Riesgo - Error de planificación

Riesgo	Problemas de disponibilidad
Descripción	Encuentro con problemas de disponibilidad tales como una enfermedad o accidente del desarrollador que impida seguir con la planificación del proyecto.
Tipo de riesgo	Personal
Impacto	Cumplimiento con los objetivos del proyecto
Probabilidad	Media
Acción	Descanso del tiempo necesario para la recuperación y volver al trabajo lo antes posible para recuperar el tiempo perdido.

Tabla 10: Riesgo - Problemas de disponibilidad

Riesgo	Desmotivación
Descripción	Encuentro con problemas de desmotivación que causen una bajada en el rendimiento del desarrollo pudiendo llegar al abandono del mismo.
Tipo de riesgo	Personal
Impacto	Cumplimiento con los objetivos del proyecto
Probabilidad	Media
Acción	Desconexión el tiempo necesario realizando una tarea diferente del proyecto o ajena a este para recuperar la motivación e interés.

Tabla 11: Riesgo - Desmotivación

Riesgo	Errores de compatibilidad de herramientas
Descripción	Dificultad para continuar con el desarrollo del proyecto debido a problemas y errores encontrados a causa de incompatibilidad con las herramientas y <i>frameworks</i> utilizados.
Tipo de riesgo	Hardware y Software
Impacto	Cumplimiento con los objetivos del proyecto
Probabilidad	Alta
Acción	Dedicación del tiempo necesario y menor posible para la solución de la incompatibilidad para continuar con el trabajo y recuperar el tiempo perdido lo antes posible.

Tabla 12: Riesgo - Errores de compatibilidad de herramientas

Riesgo	Sobreestimación o infravaloración de las tareas
Descripción	Encuentro con errores en la estimación de tiempo o costes de las tareas, ya sean de sobreestimación o infravaloración debido a la falta de experiencia.
Tipo de riesgo	Estimación
Impacto	Cumplimiento con los objetivos del proyecto

Probabilidad	Alta
Acción	Dedicación de mayor o menor tiempo en la tarea y reajuste de la estimación y planificación.

Tabla 13: Riesgo - Sobreestimación o infravaloración de las tareas

9.7 Conclusión

Aunque no tengamos a priori una justificación económica clara de principio, por ser una aplicación que se difundirá por las plataformas de tiendas online, y en estas es impredecible el ingreso que nos pueden reportar y el tiempo. Podemos concluir con que el desarrollo de este proyecto es viable técnicamente, el riesgo principalmente es bajo, no refleja problemas legales, y no existe una app con idénticas características.

10. Hardware y Software

10.1 Herramientas hardware empleadas

Al ser nuestro proyecto un videojuego para dispositivos móviles, se ha utilizado como hardware un *smartphone* con sistema operativo Android y control táctil para comprobar su funcionamiento y hacer las pruebas necesarias para verificar su calidad. Mientras que para el desarrollo del mismo se ha utilizado un ordenador portátil.

10.2 Herramientas software empleadas

Se ha utilizado Windows 8.1 como sistema operativo a la hora del desarrollo del juego. Para portarlo a iOS, se ha hecho uso de una máquina virtual con el sistema operativo Mac OS X 10.9 Mavericks de 64 bits, junto al software Xcode 6.

El entorno de desarrollo empleado ha sido Eclipse con el SDK de Android e iOS. Para desarrollar el juego se ha utilizado el framework LibGDX.

A la hora del testeo del juego se ha empleado tanto el propio ordenador como emuladores de dispositivos móviles con los sistemas operativos Android e iOS.

Para la parte gráfica se ha utilizado Paint Tool SAI y Photoshop CS5 para los gráficos en dibujos 2D y Graphics Gale para los gráficos 2D en pixel art.

Para la parte sonora se ha empleado el software generador de sonidos SFXR para los efectos de sonido en estilo 8 bits.

10.3 Requisitos Hardware y Software

Los requisitos necesarios para ejecutar el juego de forma correcta son los siguientes:

- Dispositivo móvil con pantalla táctil
- Sistema operativo Android o iOS

11. Conclusiones

11.1 Objetivos alcanzados

Echando un vistazo atrás, recordamos que los objetivos de este proyecto eran el diseño y el desarrollo de un videojuego multiplataforma para diferentes dispositivos móviles.

Tras haber llegado al final, se puede decir que se han cumplido todos los objetivos y especificaciones que se habían planeado para este proyecto. El resultado que obtenemos es un videojuego funcional tanto en móviles Android como en iOS.

Este proyecto ha consistido el mayor reto de todo el trabajo realizado a lo largo del grado de Ingeniería Multimedia cursado y he de admitir que ha sido un esfuerzo superior al esperado, pero todo el esfuerzo empleado ha merecido la pena al ver los objetivos cumplidos y el videojuego funcionando en los dispositivos móviles.

A pesar de que se hayan producido retrasos según la planificación estimada, y muchas veces el trabajo se acumulara, todo se ha podido afrontar exitosamente sin que al final haya existido una desviación significativa sobre el tiempo proyectado.

Aunque en un principio la idea que se tenía era un poco más ambiciosa, ésta se ha tenido que ajustar al tiempo que se disponía para la entrega del proyecto. Quedando de todas formas un resultado muy satisfactorio, totalmente operativo, completo y dejando el proyecto abierto a nuevas funcionalidades y ampliaciones.

11.2 Líneas Futuras de trabajo

Después de cumplir con los objetivos planificados, como hemos mencionado antes, el videojuego nos abre las puertas a muchas funcionalidades y opciones nuevas, de hecho, es recomendable, pues un videojuego para dispositivos móviles si no está en constante cambio y ofrece cada vez más funciones nuevas el público llega a perder el interés en él.

Además, al ser publicado en las plataformas de distribución, mantenerlo actualizado y promocionarlo es la clave para que no se pierda entre la innumerable cantidad de aplicaciones que existen en estas tiendas online.

Por lo tanto, se le pueden añadir infinidad de cosas nuevas, algunas de ellas pueden ser las siguientes:

- Añadir más niveles y zonas para dar más variedad a la jugabilidad y que el jugador tenga una mejor experiencia en el juego.
- Añadir nuevos personajes.
- Añadir más obstáculos y nuevas metas.
- Más enemigos y jefes que supongan un verdadero reto al jugador.

12. Anexo: Manual de Instrucciones

Historia

LHED es un misterioso androide que despierta dentro de una máquina de pruebas en un extraño laboratorio. Incapaz de recordar qué le ha pasado y cómo ha llegado allí, encuentra un fragmento de una gema con vida propia que le ayuda a salir de allí dándole el poder suficiente. Su objetivo ahora es conseguir escapar del lugar, recuperar sus recuerdos y conocer el origen de esa extraña gema, pero no le será tan sencillo, ya que se encontrará con otros androides con fragmentos de gemas similares a la suya que tratarán de detenerle.

Personajes

LHED: Es el protagonista del juego, se trata de un joven androide que ha perdido sus recuerdos y se encuentra atrapado en un laboratorio desconocido. Muy obstinado y persistente, no dudará en enfrentarse a cualquiera que le impida salir de allí y averiguar lo que le ha pasado.



Figura 64: Manual de Instrucciones - LHED

Shiro: uno de los enemigos que impedirán que LHED escape del laboratorio. Shiro es también un androide poderoso que tiene a su control un enorme robot que le sirve como arma. Aunque se desconocen sus verdaderas intenciones, hará todo lo posible para acabar con LHED.



Figura 65: Manual de Instrucciones - Shiro

Iniciar la Partida

Menú de Inicio

Al ejecutar el juego, aparecerá una pantalla de carga que dará paso al menú principal. En este menú podrás observar las siguientes opciones:



- **Comenzar partida:** te permite empezar la partida. Cuando pulses este botón te aparecerá el siguiente menú para elegir el nivel.



- **Logros:** al pulsar este botón aparecerá un panel con los logros y la información de cada logro disponible en el juego.



- **Votar:** al pulsar en esta opción se abrirá la aplicación de Google Play o App Store para efectuar una votación al juego.



- **Créditos:** selecciona esta opción para ver un panel con los créditos del juego.



- **Opciones:** al pulsar en esta opción podrás ver un panel con las distintas opciones del juego.



- **Redes sociales:** al pulsar en estos botones se abrirá el navegador de tu dispositivo móvil y te redirigirá a la red social correspondiente: Facebook o Twitter.



Selección de nivel

Al comenzar partida podrás ver el menú de selección de nivel y de zonas. En este menú podrás elegir qué nivel de los disponibles jugar. Una vez completes un nivel, se desbloqueará el siguiente.



Figura 66: Manual de Instrucciones - Menú Selección de nivel

En este menú encontrarás los niveles de cada zona. Los niveles que estén de color son los que están disponibles, mientras que los que aparecen apagados o en color gris son los que aún están sin desbloquear.

En cada panel del nivel aparece la información del mismo, es decir, el número de engranajes y el de baterías que has conseguido en ese nivel. Puedes repetir niveles ya desbloqueados para conseguir el 100% del nivel.

En la primera zona hay un nivel especial que en las otras no se encuentra, el nivel Tutorial. Este nivel es un pequeño tutorial que te enseña los controles básicos y cómo progresar en el juego.

Niveles

En el juego hay dos tipos de niveles, el nivel de recogida y el nivel de jefe.

Nivel de Recogida

En el nivel de recogida el objetivo es recolectar un determinado número de objetos. Estos objetos son engranajes y baterías. En cada nivel de recogida hay un máximo de 3 baterías, mientras que de engranajes el número varía en cada nivel. Para superar el nivel tienes que tener al menos la mitad de la cantidad de engranajes totales que hay en el nivel.

En este tipo de nivel te puedes encontrar con los siguientes elementos:



Figura 67: Manual de Instrucciones - Nivel de recogida

1. LHED, el personaje principal que controlas.
2. Obstáculos, te harán daño si te chocas contra ellos.
3. Engranajes, son los objetos principales que tienes que recoger para pasarte el nivel.
4. Retrato de LHED, te indica cuando el personaje recibe daño o muere.

5. Vida de LHED, te indica la vida actual del personaje representada por corazones.
6. Indicador de engranajes conseguidos/totales.
7. Indicador de baterías conseguidas/totales.
8. Botón de pausa.

Nivel de Jefe

En el nivel de Jefe tendrás que enfrentarte a un poderoso enemigo y derrotarlo para completar el nivel. Al principio de este nivel podrás ver una pequeña escena entre el personaje principal y el enemigo.



Figura 68: Manual de Instrucciones - Nivel de Jefe

1. Enemigo, jefe actual del nivel.
2. Cañones del enemigo, irán cambiando de posición y disparándote sin cesar.
3. Estrella de energía, necesitarás recoger 3 de estas para poder realizar un ataque hacia el jefe. Si se te escapa alguna estrella, el enemigo las absorberá, y si consigue tres será el quién realice el ataque hacia ti.
4. Retrato de LHED, cambiará cuando aparezcan diálogos.
5. Indicador de Estrellas de Energía conseguidas/necesarias para realizar un ataque.
6. Casilla de modo, indica el modo en el que está el personaje.
7. Diálogos de LHED.
8. Retrato del enemigo.

9. Vida del enemigo.
10. Indicador de Estrellas de Energía recogidas/necesarias del enemigo para que realice un ataque contra ti.
11. Diálogos del enemigo.

Objetos

Durante los niveles podrás recoger ciertos objetos que te servirán para completar los niveles o te serán de ayuda. A continuación se muestran cuáles son estos objetos:



- **Engranaje:** los engranajes aparecerán solo en los niveles de recogida, son pequeñas piezas necesarias para completar un nivel. Para completar el nivel necesitarás al menos conseguir la mitad.



- **Batería:** las baterías solo aparecerán en los niveles de recogida, al igual que los engranajes, y siempre habrá 3 de ellas en cada nivel. Sirven para obtener el 100% del nivel.



- **Esfera de salud:** se trata de una esfera flotante con un corazón en su interior. Puede aparecer tanto en los niveles de recogida como en los niveles de jefe. Al cogerla recuperarás un corazón.



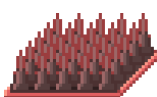
- **Estrella de Energía:** es una estrella que contiene una misteriosa energía. Solo aparecerá en los niveles de jefe y necesitarás recoger 3 de ellas para poder realizar un ataque.

Obstáculos

En cada nivel te topará con obstáculos que intentarán impedirte el paso y que si los tocas recibirás daño. Estos son los tipos de obstáculos que encontrarás:



- **Bomba:** esta peligrosa bomba explotará nada más rozarla. Si te da te quitará un corazón de vida. Suele aparecer muchas veces en hileras con otras bombas.



- **Pinchos:** estos afilados pinchos se encuentran situados tanto en el suelo como en el techo y si los rozas te dañarán quitándote medio corazón de vida.

Modos y Ataques

En los niveles de jefe LHED puede adoptar dos modos distintos, de ataque y de defensa. Este modo se indicará en la casilla de modo mediante un icono. Si está la casilla vacía significará que no ha adoptado de momento ningún modo. A continuación se muestran los iconos de cada modo:



- **Modo de ataque:** cuando LHED coge 3 Estrellas de Energía adoptará una posición ofensiva y se preparará para atacar al enemigo.

Cuando LHED adopte su posición de ataque, el juego se pausará por un momento, se oscurecerá la pantalla y aparecerán una serie de combinaciones que deberás acertar para realizar el ataque con éxito.



Figura 69: Manual de Instrucciones - Combinaciones de botones

- Si aciertas en todos los botones de la combinación LHED se situará a mitad de altura de la pantalla y realizará un poderoso ataque en forma de láser que dañará al enemigo quitándole un corazón y medio.
- Si aciertas la mitad de los botones de la combinación, LHED atacará de igual forma que la anterior, pero con un ataque más flojo que solo dañará al enemigo quitándole medio corazón.

- Si fallas todos los botones de la combinación, LHED no realizará ningún ataque y será el enemigo el que lo haga, quitándote un corazón de vida.



- **Modo de defensa:** cuando Shiro coge 3 Estrellas de Energía LHED adoptará una posición defensiva y se preparará para protegerse del ataque del enemigo.

De manera similar, si Shiro coge las 3 Estrellas de Energía será él el que realice el ataque lanzando una enorme bola de fuego y obligará a LHED a pasar a modo defensa. De esta forma aparecerán las combinaciones al igual que en el modo ataque:

- Si aciertas todos los botones de las combinaciones, LHED activará un burbuja a modo de escudo y se protegerá totalmente y con éxito, sin sufrir ningún daño.
- Si aciertas la mitad o más de los botones de las combinaciones, LHED activará igualmente la burbuja para protegerse, pero más débil, por lo que recibirá algo de daño que le quitará medio corazón de vida.
- Si fallas toda la combinación, LHED no podrá activar su escudo para defenderse y sufrirá todo el poder del ataque del enemigo, quitándole un corazón de vida.

Menú de Pausa

Al pulsar en el botón de pausa el juego se detendrá y aparecerá el siguiente panel con las siguientes opciones:



Figura 70: Manual de Instrucciones - Menú de pausa

1. Quitar la pausa y reanudar el juego.
2. Volver a la selección de zonas y niveles.
3. Volver al menú principal.
4. Quitar la pausa y reanudar el juego.

Game Over

Si la los corazones de vida de LHED llegan a estar todos vacíos la partida terminará y aparecerá el siguiente panel con las siguientes opciones:



Figura 71: Manual de Instrucciones - Game Over

1. Reintentar el nivel desde el principio.
2. Volver a la selección de zonas y niveles.
3. Volver al menú principal.

13. Bibliografía y referencias

Libros

- [1] Beginning Android Games (2011) Mario Zechner
- [2] Learning LibGDX Game Development (2013) Andreas Oehlke
- [3] Historia, teoría y práctica del diseño conceptual de videojuegos (2010) Borja López Barinaga [el libro es gratuito y se puede descargar de aquí:
<http://www.alesiagames.com/libros/Juego-Borja-Lopez-Barinaga.pdf>]

Páginas web

- [4] Página principal de LibGDX
<https://libgdx.badlogicgames.com/>
- [5] Documentación sobre LibGDX
<https://github.com/libgdx/libgdx/wiki>
<https://github.com/libgdx/libgdx/wiki/Deploying-your-application>
- [6] Fases del desarrollo de un videojuego
<http://www.cristalab.com/blog/las-fases-del-desarrollo-de-un-videojuego-guia-para-principiantes-c100401/>
<https://ecoperenlared.wordpress.com/2013/03/06/el-desarrollo-de-los-videojuegos/>
- [7] Videojuegos para móviles
<http://www.elandroidelibre.com/2015/03/videojuegos-para-moviles-la-gran-oportunidad-para-desarrolladores-con-pocos-recursos.html>
- [8] Google Play
<https://play.google.com/store>
<https://support.google.com/googleplay/android-developer/answer/113469?hl=es>
- [6] Plataforma publicitaria AdMob
<http://www.google.es/admob/>
- [7] Metodologías y metodología SUM
<http://definicion.de/metodologia/>
<http://www.gemserk.com/sum/>

http://www.fing.edu.uy/sites/default/files/biblio/22811/asse_2009_16.pdf

[8] Gliffy, herramienta online para crear diagramas de flujo

<https://www.gliffy.com/>

[9] StackOverflow, comunidad muy útil para cualquier tipo de duda con java

<http://stackoverflow.com/>

[10] Historia de los videojuegos

http://www.elotrolado.net/wiki/Historia_de_los_videojuegos

https://es.wikipedia.org/wiki/Historia_de_los_videojuegos